Advanced Higher Computing Science Project

ESMS Book Bank

Matthew Zahra
101446964

Table of Contents

Analysis

Problem Description:

Outline of Problem:

My Project, ESMS Book Bank, will be a Web Design and Development project integrated with Database Design and Development.

My project will act as a website in which students who are looking to buy school textbooks can be connected with students who are looking to sell their textbooks. This will allow people to easily get the contact details of the seller/buyer, making communication easier, allowing both parties to sort out a transaction in their own time in a more manageable style.

Users will be able to register/login and then either browse books for sale or put their own books up for sale. If a potential buyer sees a book they wish to purchase, they will be able to select the book which will lead to an email being sent to the seller with the buyer's and book's details and vice versa. Users will also be able to delete any books that they have put up for sale, as well as being able to change their password.

The website will be managed by admin status users – they can view all normal users, delete normal users, delete the books that have been put up for sale by other users and can promote normal users to admin status.

The admin users will be managed by a super_admin – they are the only account(s) that can demote an admin user back to a normal user.

This system will be restricted to only be used by pupils of the school – this will be carried out by ensuring they have access to a '…@esms.org.uk' email address, which are issued by the school.

It will be suitable for an Advanced Higher project as a Web Design and Development Project as it will include:

- Form elements will be used many times – for example, to allow users to register, login, put a book up for sale etc…
- It will be styled by external CSS so that all the pages have a consistent look and feel.
- Media queries will be used so that the page can be resized when being viewed on a mobile device.
- PHP session variables will be used to:
    - Store a logged in user's details
    - Ensure only a logged in user can see pages beyond the login, register and forgot password pages
    - Greet each user individually as they login
    - Ensure only admin users can access the users.php page
    - Ensure only the super_admin can manage the other admin users

- PHP scripts will assign data to variables when working with the data submitted by the forms – the scripts will also be used to validate and handle the data on the server side.

It will also integrate with Database Design and Development as it will include:

- A database which:
    - Will have a table for the user details.
    - Will have a table for all the books for sale.
    - Will be accessed by the PHP scripts setting up a connection to the database – SQL SELECT statements will be executed to display all the books, INSERT statements will be used to add new books/users, DELETE statements will be used to delete books/users, UPDATE statements will be used to change a user's status/password etc…
    - The results of the queries will be formatted as appropriate – e.g. when selecting all the books available from the books table, they will be displayed in an html table.

Scope:

- Design of the whole project:
    - Pseudocode of all processes
    - Design for SQL queries used
    - Annotated wireframes indicating the page layout for viewing on both a desktop and a mobile device
    - Data dictionaries for all tables
    - Entity Relationship Diagram for all tables
    - Entity Occurrence Diagram for database system
    - A UML use case diagram of my solution
    - Printouts of all code
- Files (Implementation):
    - All the various PHP/HTML/CSS files as well as a completely set up database with all the necessary tables
- Testing:
    - A complete test plan with a description of test cases and personas
    - Screenshots of the tests displaying all the inputs and outputs as well as any errors that arise
- Evaluation:
    - An evaluation report as a result of the testing and comparing final project with requirements

Boundaries:

My Solution will contain/be able to do:

- Validate all user inputs:

    - This may take the form of presence checks (e.g. making sure a username and password have been entered, or that the necessary fields have been entered when putting a book up for sale), length checks (for text inputs) and ensuring special characters have been removed to avoid SQL injection, range checks (for ensuring the price is a positive number but not too large) – this validation will be applied to each form used
    - Length checks:
        - passwords must be > 5 characters and <= 256 characters
        - Emails must be <= 60 characters
        - Title/edition of book, author and condition must be <= 60 characters
        - ISBN Number must be 10 or 13 digits long (and contain only numeric characters)
    - Photo of book must be of suitable filetype (.jpg, .jpeg or .png) and size (<= 3Mb)
    - Emails must all contain "...@esms.org.uk" so that only pupils in the school can register, in order to maintain a safe environment – and also, all email addresses must be unique
    - Restricted choice:
        - For qualification and subject, users will only be able to choose from a pre-determined dropdown list:
- To be viewed on either a desktop or a mobile device – CSS will assume that users are using one of these two device types. Media queries will kick in when the screen size is <= 420px.
- Once a user logs in, their email, userID and user status will be saved to a session – this session will be checked at every page to ensure that only logged in users can progress past the login, registration and forgot password pages, and that only users of admin status can access the users.php page and can delete other users' books – and also that only the super_admin(s) can see and demote the other admin users.
- When a user registers, to ensure they are part of the school, their email (school email of the form '…@esms.org.uk') will get sent a confirmation email with a verification code – this needs to be entered the first time they log in and within 5 minutes of registering.
- When a user forgets their password, they will get emailed with a verification code to login with (they have 5 minutes to enter it) – they can then change their password once they are in
- Password gets hashed when stored in database
- **The only way to have a super_admin account is to have it directly put into the database. In the Implementation section, there is some SQL code which shows how to do this.**

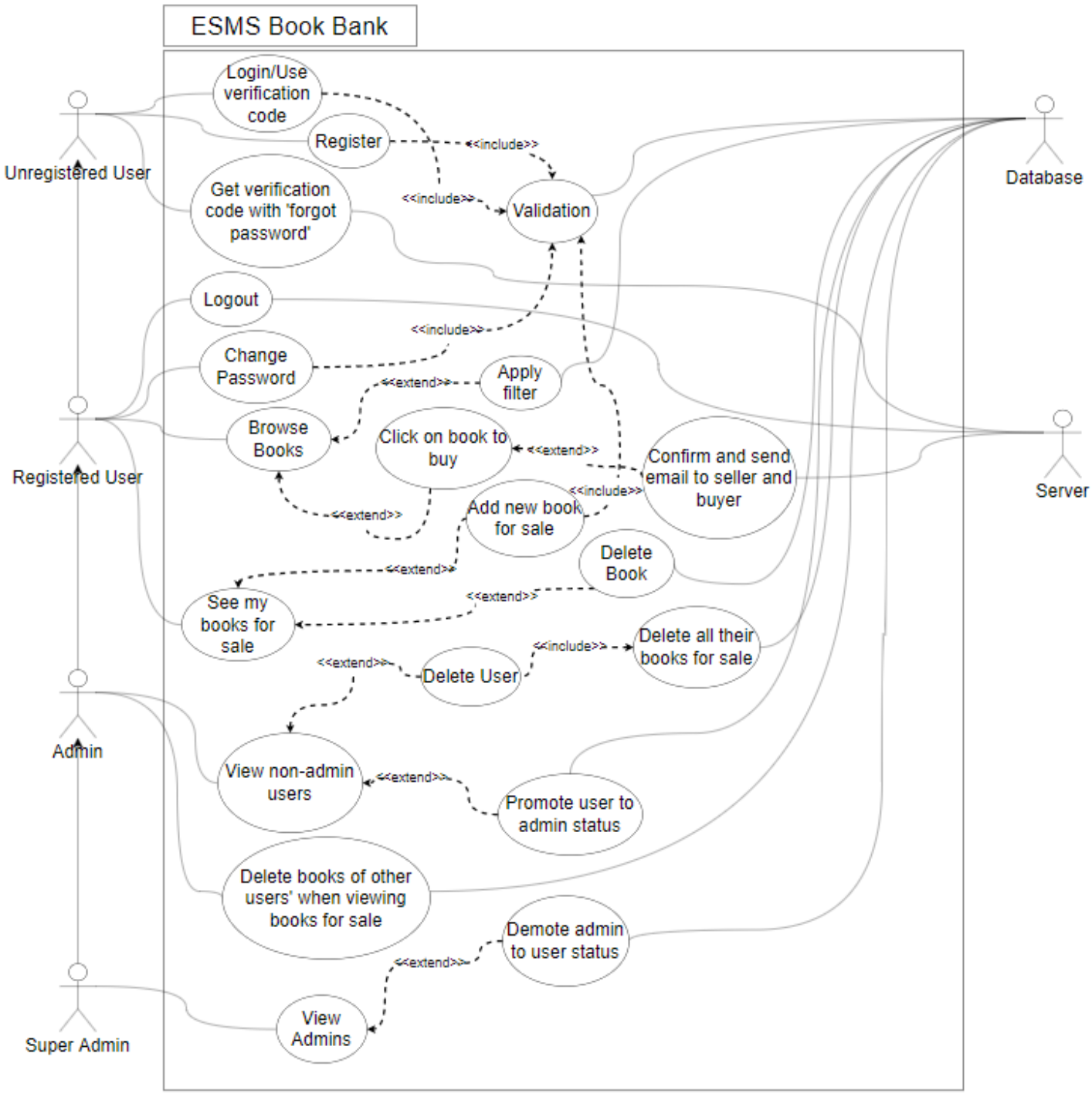
Constraints:

The project will be restricted by the following constraints:

- Time Constraints:
    - Project must be delivered by the 25th of March 2022

- Legal Constraints:
    - In order to satisfy the Copyright, Designs and Patents Act 1988, all images/icons used will be copyright-free or will be used with their owner's consent
    - GDPR will be taken into consideration as the school emails of users will need to be securely stored.
- Technical:
    - PHP, HTML, CSS, JavaScript and SQL will be used to develop this project
    - VSCode will be the main editor used
    - The finished project will be run on a Linux-based server
- Economic Constraints:
    - There are no issues here as all resources used are either free or already owned

UML Use Case Diagram:



Requirements Specification:

End-User Requirements:

- Before logging in:
  1. Users should be able to register an account with a unique, unverified '…@esms.org.uk' email address, and a password that is > 5 and <= 256 characters in length
  2. Users should be able to login with their email and password (and verification code if it is their first time logging in)
  3. Users should be able to select 'forgot password' and then use the verification code they have been emailed to login
- Once Logged in:
  4. A user will be able to navigate via the home page/nav bar of each other page to a different page
  5. A user will be able to change their password
  6. A user should be able to view all their books they have put up for sale, add new ones and delete any of their own books that are already up
  7. Users should be able to browse books for sale (other than their own) – applying a filter if they want to
  8. Users should be able to select a book to buy to be taken to a confirmation page
  9. Users should be able to confirm that they would like to buy a book – this should send an email to them with the details of the book and the seller
  10. Users should be able to logout
  11. Admin users should be able to delete other users' books from the browse table
  12. Admin users should be able to view, delete and promote standard users
  13. The super_admin should be able to view and demote all admin users

Functional Requirements:

14. The system should appropriately validate all inputs:
    1. Restricted choice for subjects/qualifications for books (only client side)
    2. Length checks for emails, book titles/editions/condition – all <= 60 characters; passwords which are > 5 characters and <= 256 characters; ISBN numbers are 10 or 13 digits long
    3. Ensure that the price is a positive number, less than 99999999.99 and is no more accurate than 2 decimal places
    4. Ensure all emails (at registration) are unique and of the correct format – '…@esms.org.uk'
    5. Presence checks for emails and passwords and necessary fields (such as the subject) when putting a book up for sale
    6. Escape all form inputs to avoid SQL injection

7. Photo of book is of correct filetype and size (.jpg, .jpeg or .png and <= 3Mb)
8. Validate any verification codes by checking with the database (ensure they are correct and have been entered within 5 minutes of being issued)
9. Ensuring that any login/registration details are valid etc…
10. Validate changing of passwords by ensuring that they match

15. When a user registers, the system should put details into database (if valid)
16. If a user registers with an email address that's already been registered, but the verification code was never entered (so no one ever logged in), they should be allowed to register again, and their new details will be saved by the system, after validation, to the users table, and a new verification code should be sent to their email
17. Only logged in users should be able to progress past login/registration/forgot password pages as the system should store the user's details in a session to enforce this
18. If a user selects 'forgot password', the system should email them a verification code to login with and update the code in the database
19. The system should allow a user to put a new book up for sale – adding it to the database and any photos uploaded to the files/ directory
20. The system should allow a user to remove a book that they have put up for sale – removing it from the database and removing any photos from the files/ directory
21. The system should display all the books that are for sale to each user (before a filter is applied)
22. The system should display only the books that fall under the applied filter (once the user selects a filter)
23. The system should allow users to select a book to buy and then take them to a confirmation page with the data of the corresponding book on it and in the URL
24. Once a user confirms they would like to buy a book, the system should send an email to the seller with the buyer's details and a confirmation email to the buyer
25. The system should allow admin users to delete other users' books from the site/database, also deleting the images of the books from the files/ directory
26. The system should let users logout – once this happens, all session should terminate in order to maintain security
27. The system should allow admin users to see/delete/promote standard users
28. The system should allow the super_admin to see/demote admin users

Project Plan:

Task:	Sub Task:	Estimated Time:	Resources to be Used:	Target Date of Completion:
Design	Wireframes for all pages	6 hours	Draw.io	December 1st 2022
	Low-fidelity prototypes of pages	3 hours	Draw.io	
	Design of PHP code – handling form data, email system etc. – use pseudocode where appropriate	4 hours	Draw.io/Microsoft Word	
	Data dictionary – all datatypes/constraints on tables	30 mins	Microsoft Word	
	Wireframes for media queries	2 hours	Draw.io	
	Entity relationship diagram for all tables	10 mins	Draw.io	
	Entity occurrence Diagram	10 mins	Draw.io	
	Query design for all queries that will be executed	3 hours	Microsoft Word	
Implementation	Create login.php page	1 hour	VsCode and webserver	February 1st 2022
	Create register.php page	1 hour	VsCode and webserver	
	Create home.php page	1 hour	VsCode and webserver	
	Create accountDetails.php page	1 hour	VsCode and webserver	
	Create myBooks.php page	3 hours	VsCode and webserver	
	Create browse.php page	4 hours	VsCode and webserver	
	Create pageHeader.php page	2 hours	VsCode and webserver	
	Create entrancePageHeader.php	30 mins	VsCode and webserver	
	Create users.php page	2 hours	VsCode and webserver	
	Create adminsHandler.php	30 mins	VsCode and webserver	

Implementation	Create buyBook.php	1 hour	VsCode and webserver	February 1st 2022
	Create codeHandler.php	2 hours	VsCode and webserver	
	Create connectDB.php	10 mins	VsCode and webserver	
	Create deleteBooks.php	30 mins	VsCode and webserver	
	Create email.php	4 hours	VsCode and webserver	
	Create forgotPass.php	1 hour	VsCode and webserver	
	Create forgotPassHandler.php	1 hour	VsCode and webserver	
	Create loadPage.php	30 mins	VsCode and webserver	
	Create loginHandler.php	1 hour	VsCode and webserver	
	Create logout.php	10 mins	VsCode and webserver	
	Create passwordChange.php	30 mins	VsCode and webserver	
	Create registerHandler.php	3 hours	VsCode and webserver	
	Create uploadBook.php	3 hours	VsCode and webserver	
	Create usersHandler.php	1 hour	VsCode and webserver	
	Create files directory	5 mins	-	
	Do CSS for all standard pages	12 hours	VsCode and webserver	
	Write media queries for formatting each page	6 hours	VsCode and webserver	
	Set up database and tables (and all their constraints)	1 hour	VsCode and SQL	
	Document research and development of new skills	3 hours	Microsoft Word	
	Gather all ongoing testing information and format it	3 hours	Microsoft Word	
Testing	Construct final test plan. Plan: • Requirements • Persona and test cases	2 hours	Microsoft Word	March 20th 2022

Testing	Requirements testing – screenshots of program's inputs, outputs and all error messages to demonstrate validation	8 hours	Microsoft Word	March 20th 2022
	Persona testing and use of test cases	2 hours	Microsoft Word and person	
Evaluation	Evaluation report: • Fitness for purpose • Maintainability • Robustness	3 hours	Microsoft word	March 23rd 2022

Design

Web Design:

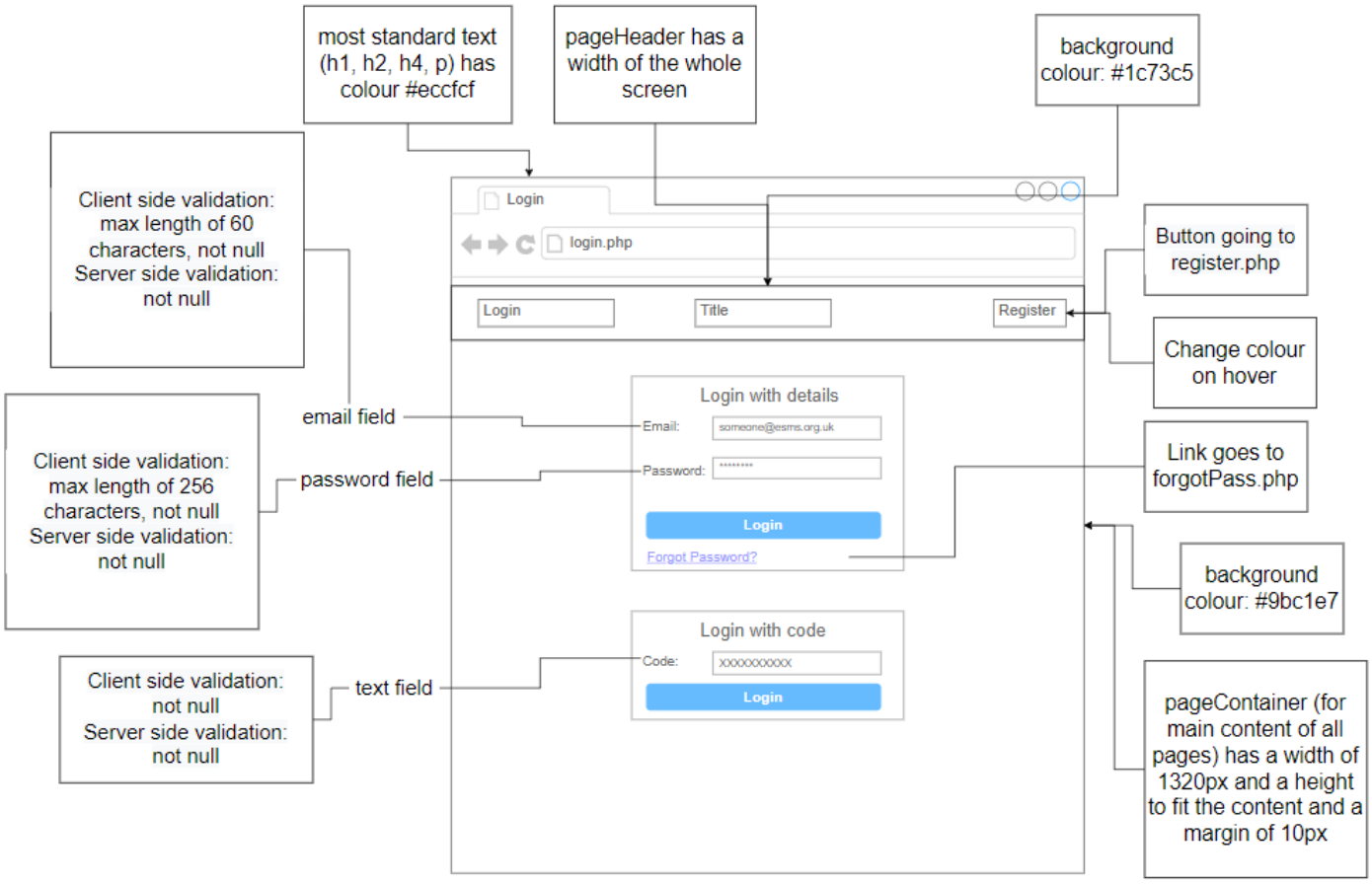
User-Interface Design – Wireframes:

All client-side validation creates error messages at the fields of input on the forms. All server-side validation will print the errors in a div on the page – this will be indicated by a '*Errors Box*'.

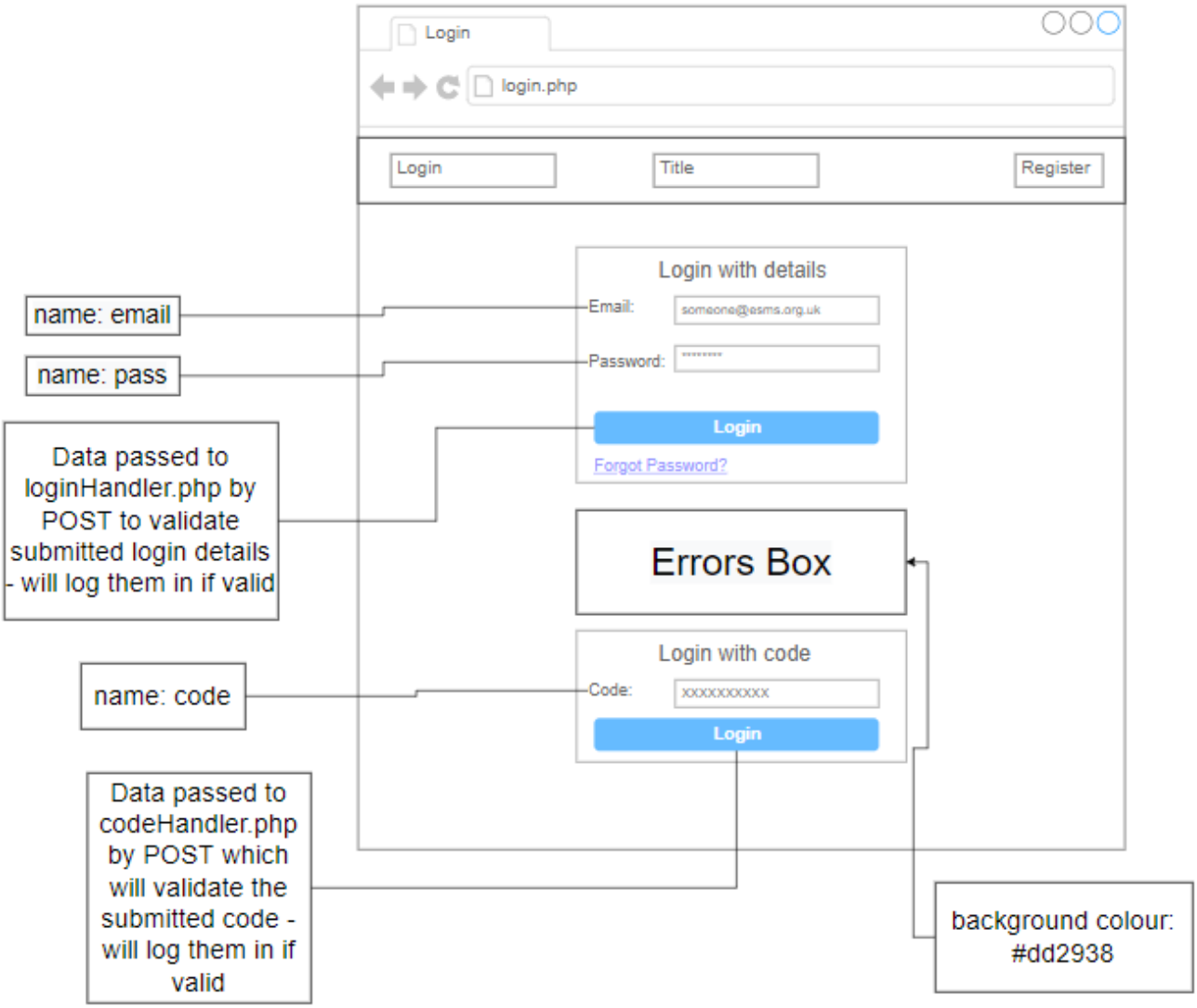
Media queries will be activated when the device screen size is 420px or less.

Underlying processes are indicated on the 'errors and dataflow' wireframe of each page, as they indicated what happens when each form gets submitted.

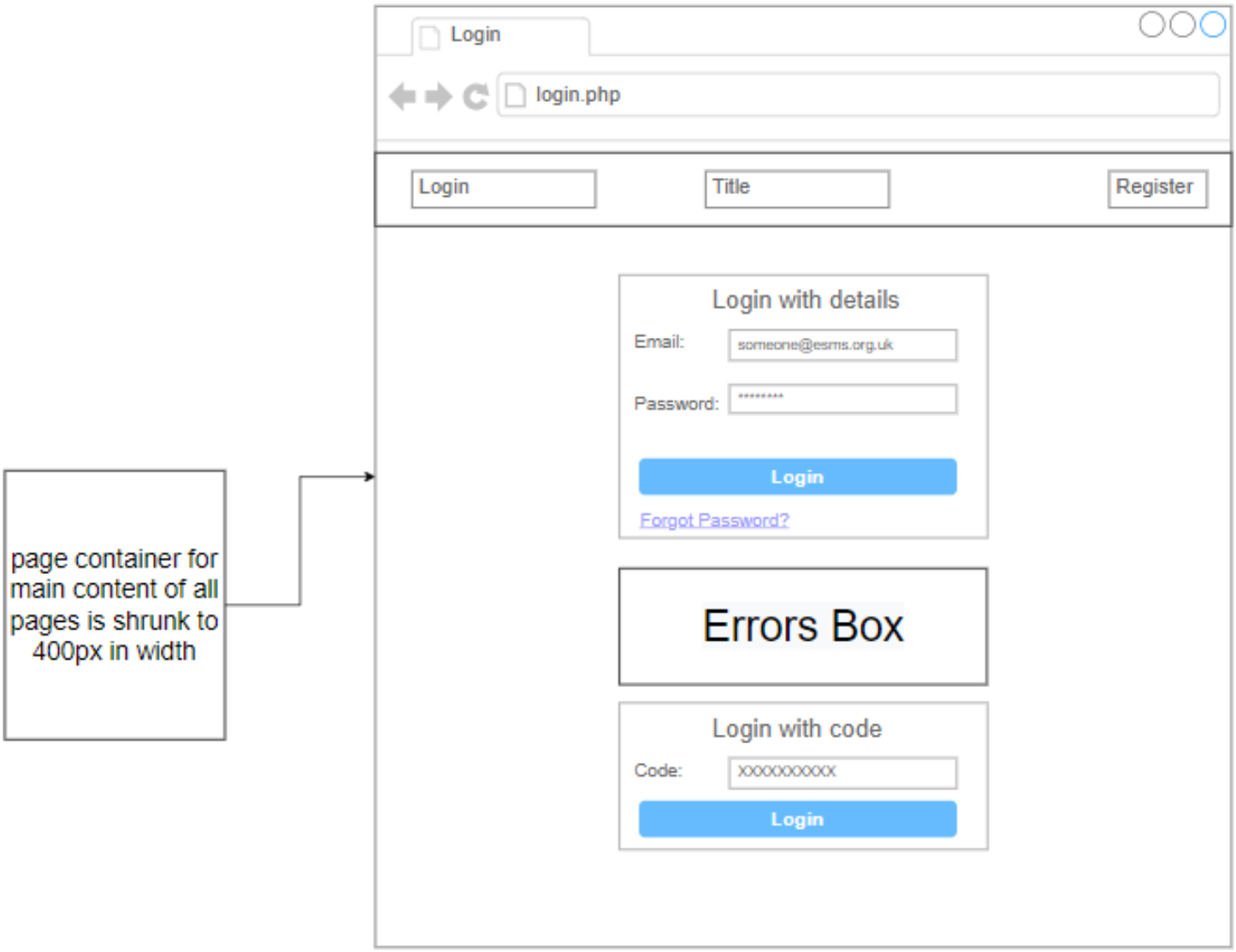
login.php



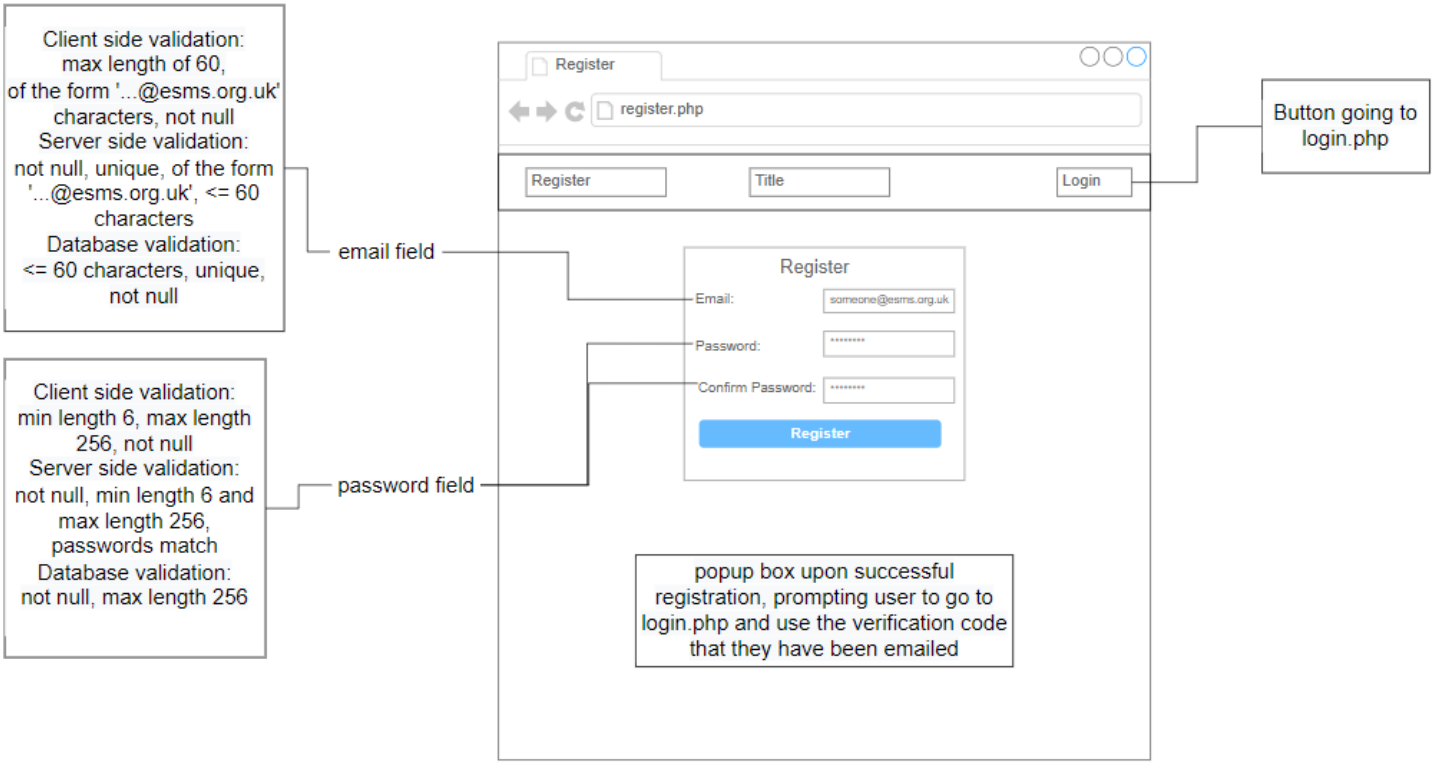
login.php – errors and dataflow



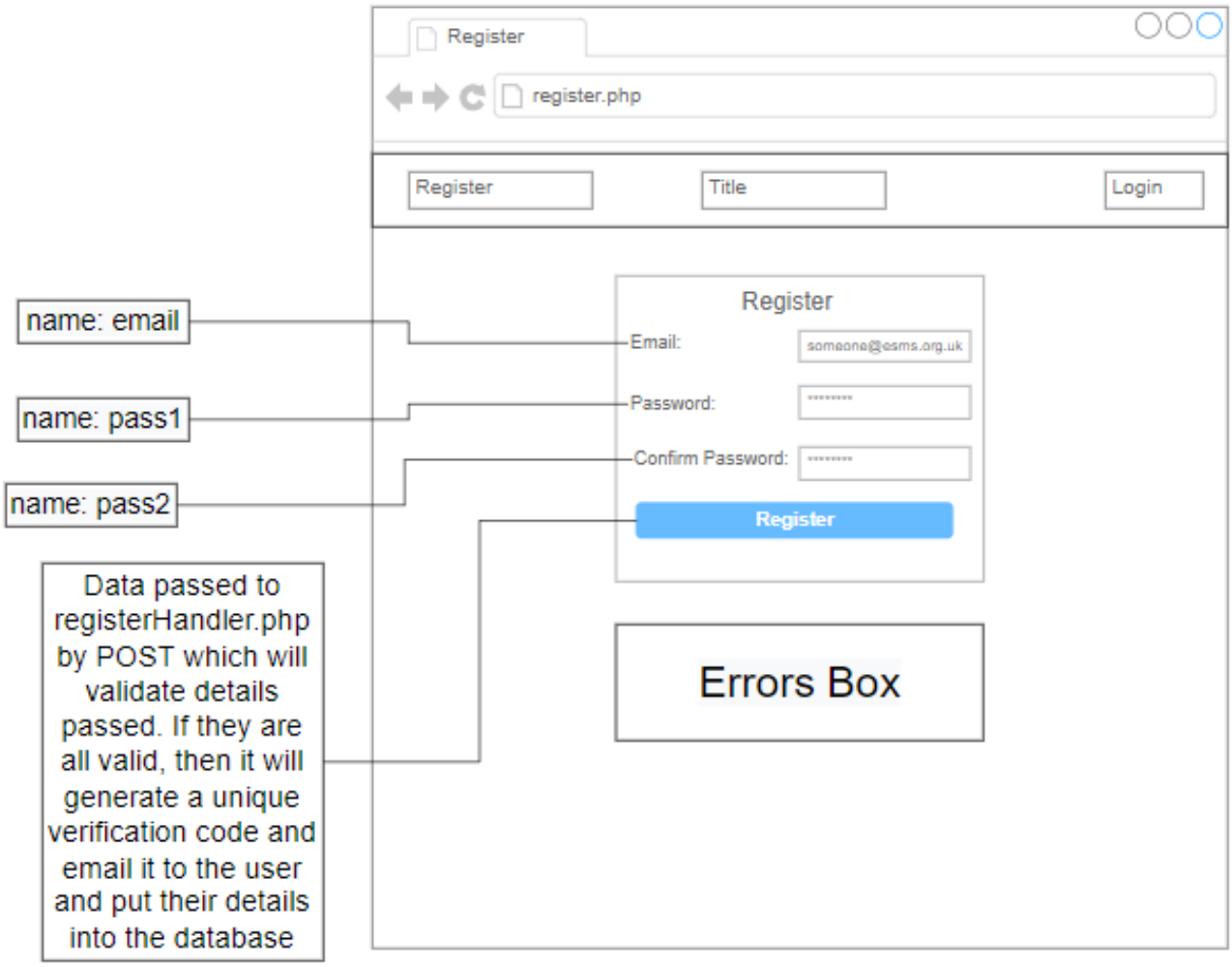
login.php – media query



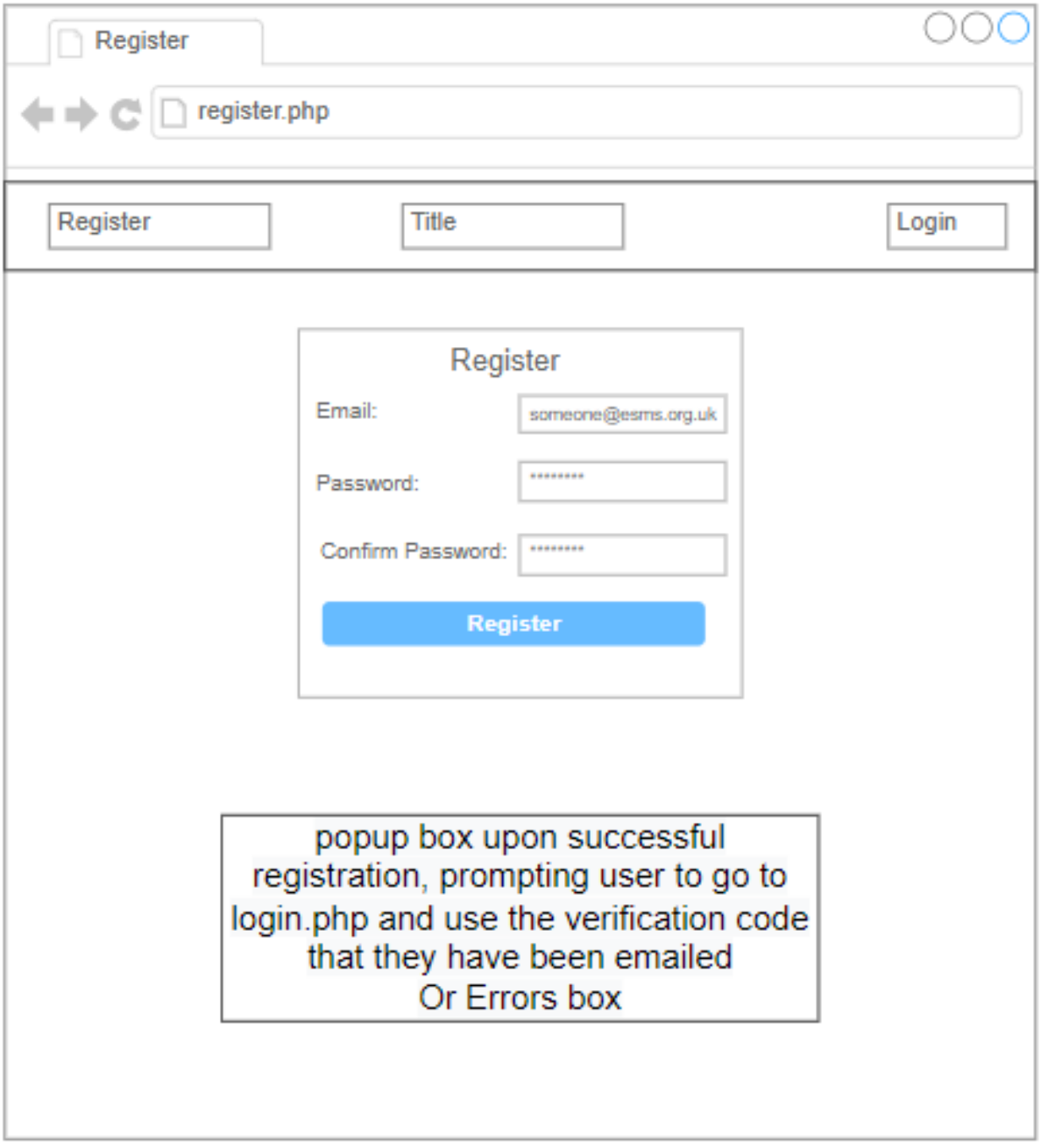
register.php



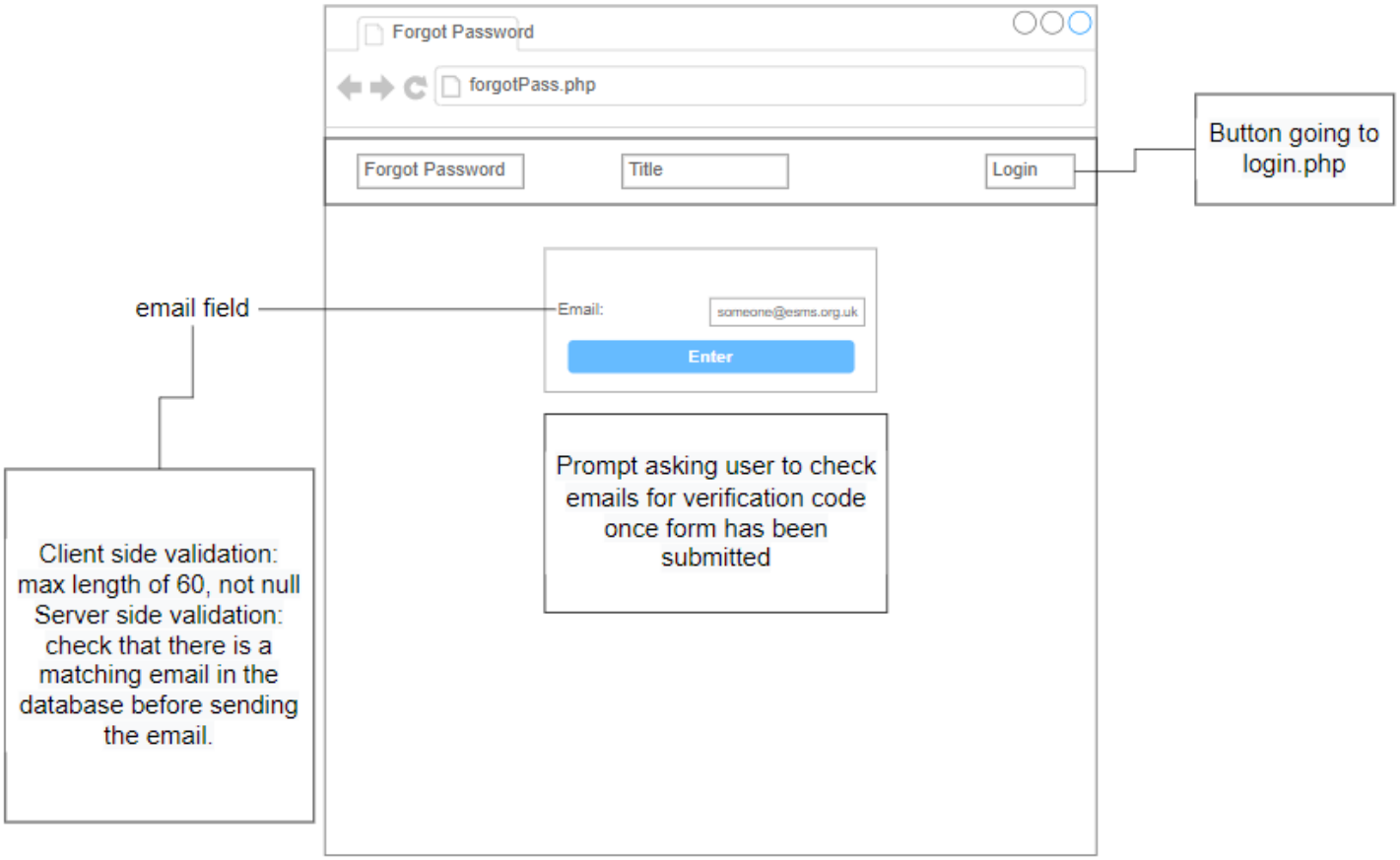
register.php – errors and dataflow



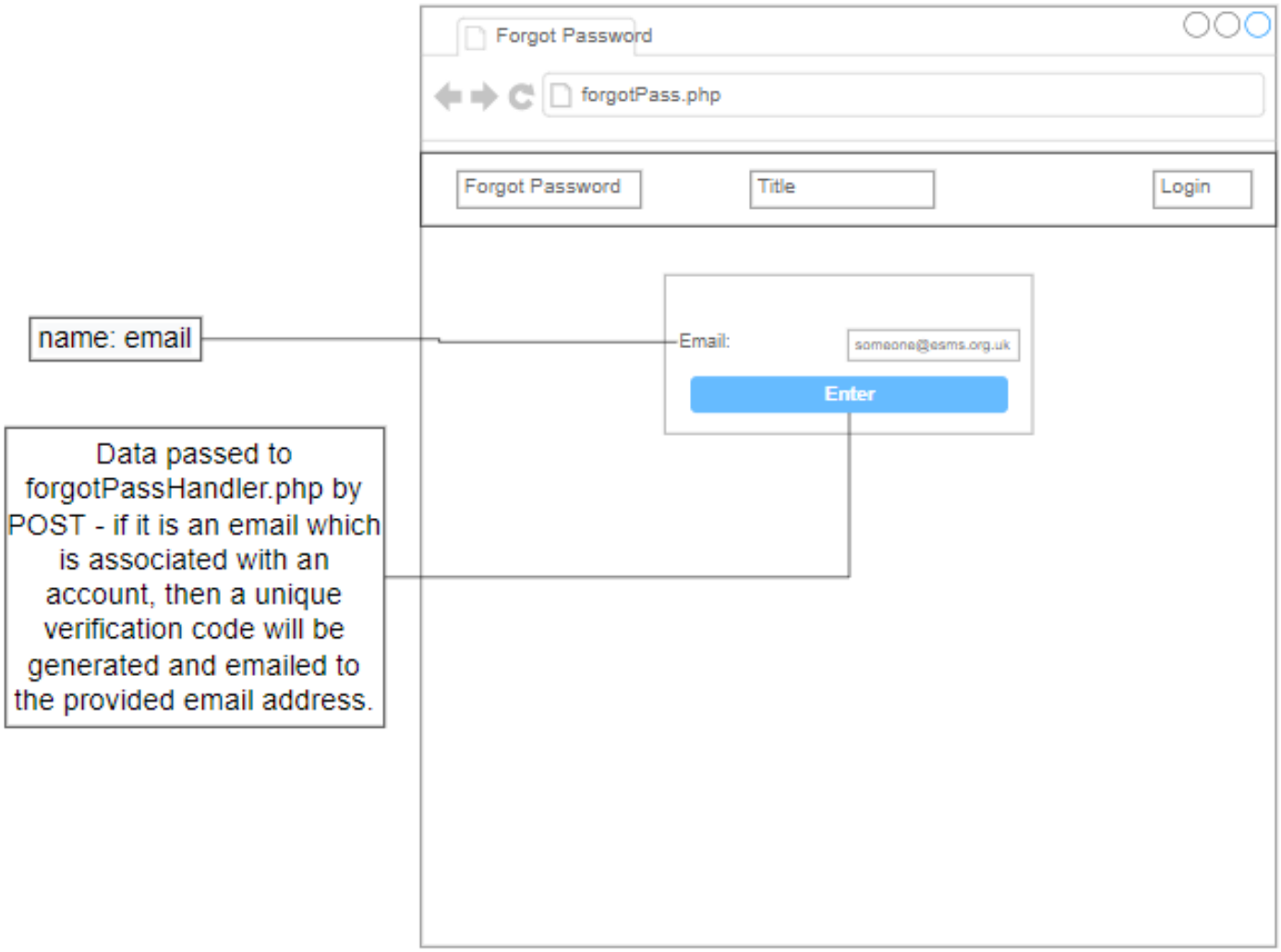
register.php – media query



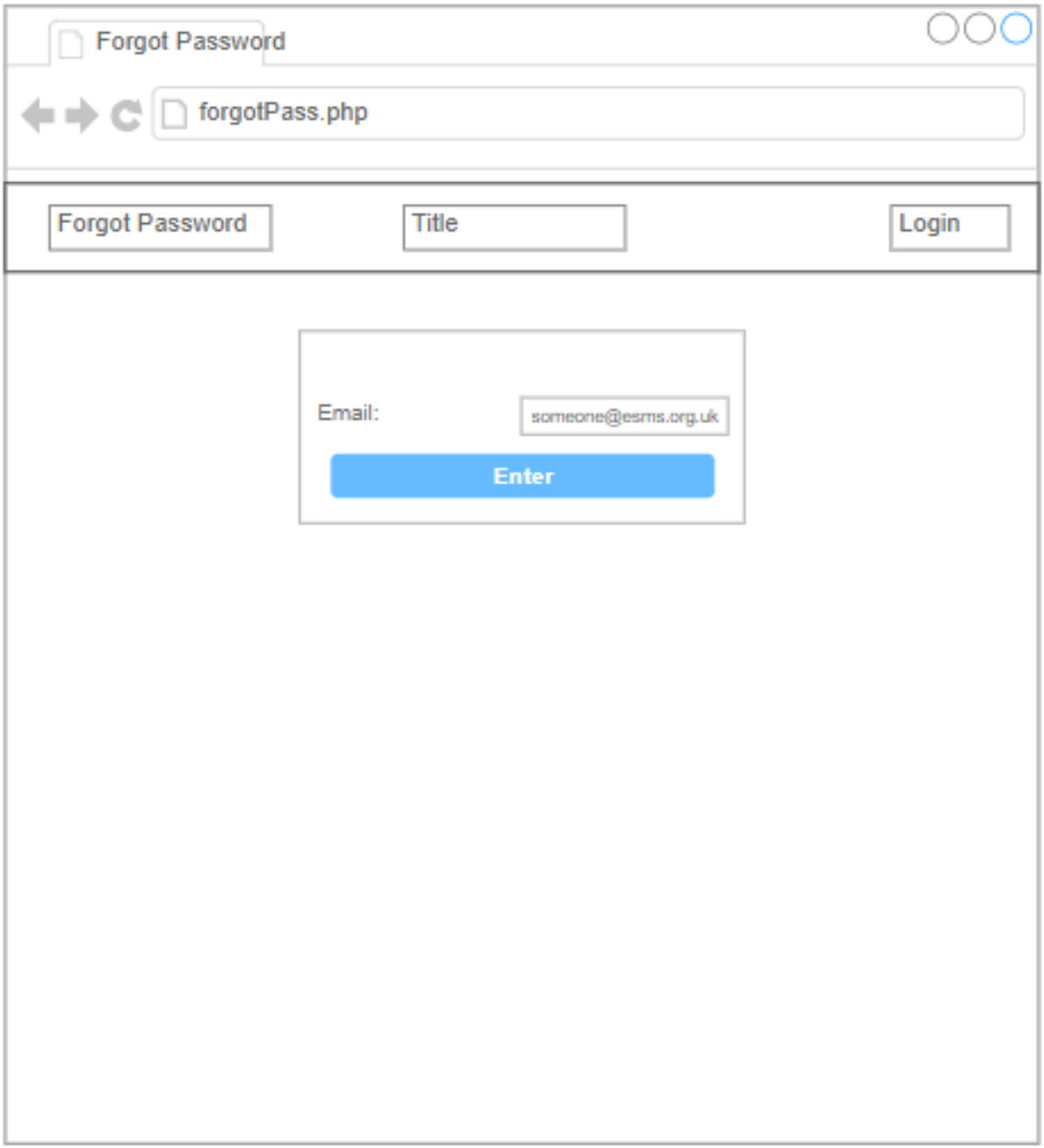
forgotPass.php



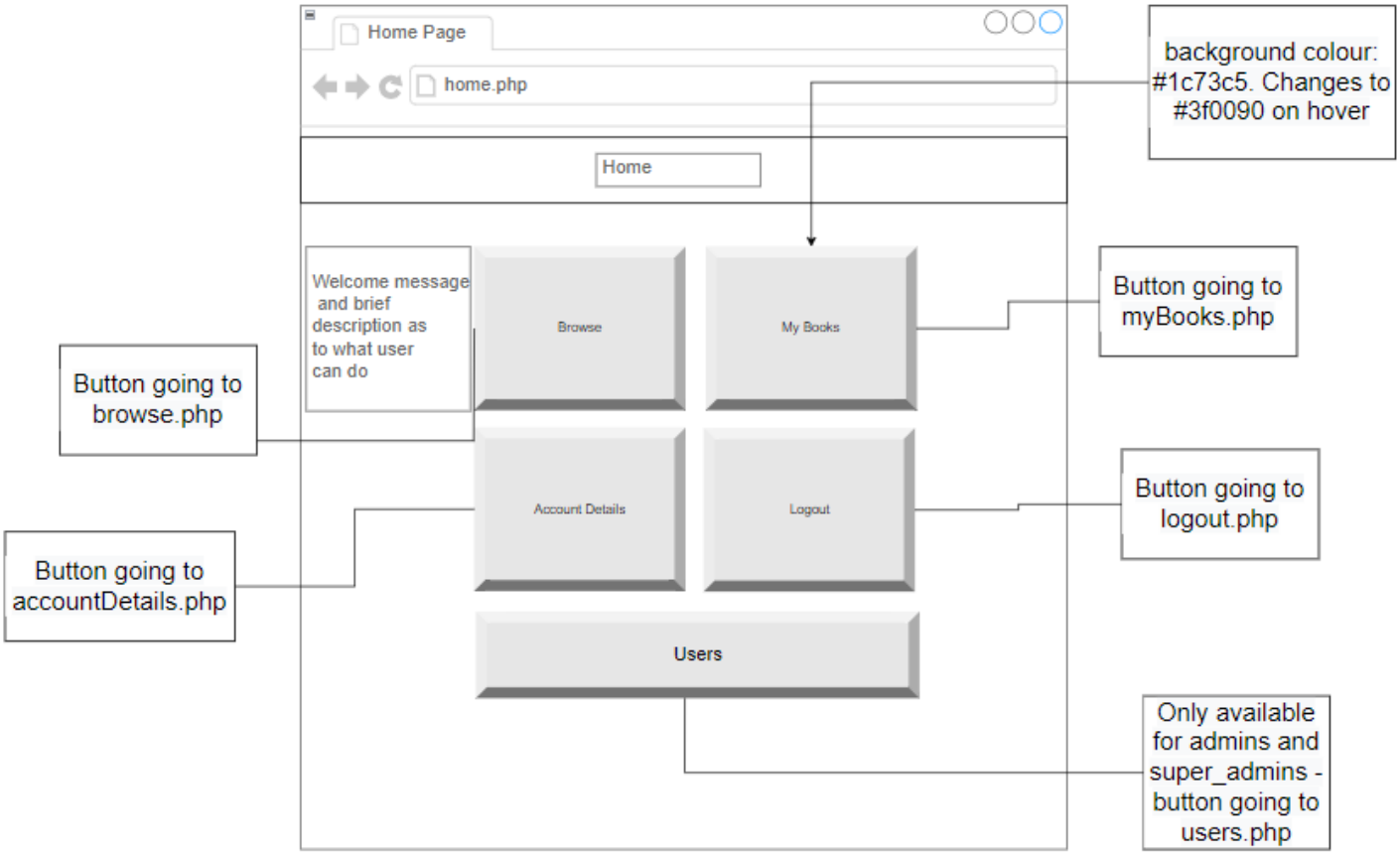
forgotPass.php – errors and dataflow



forgotPass.php – media query



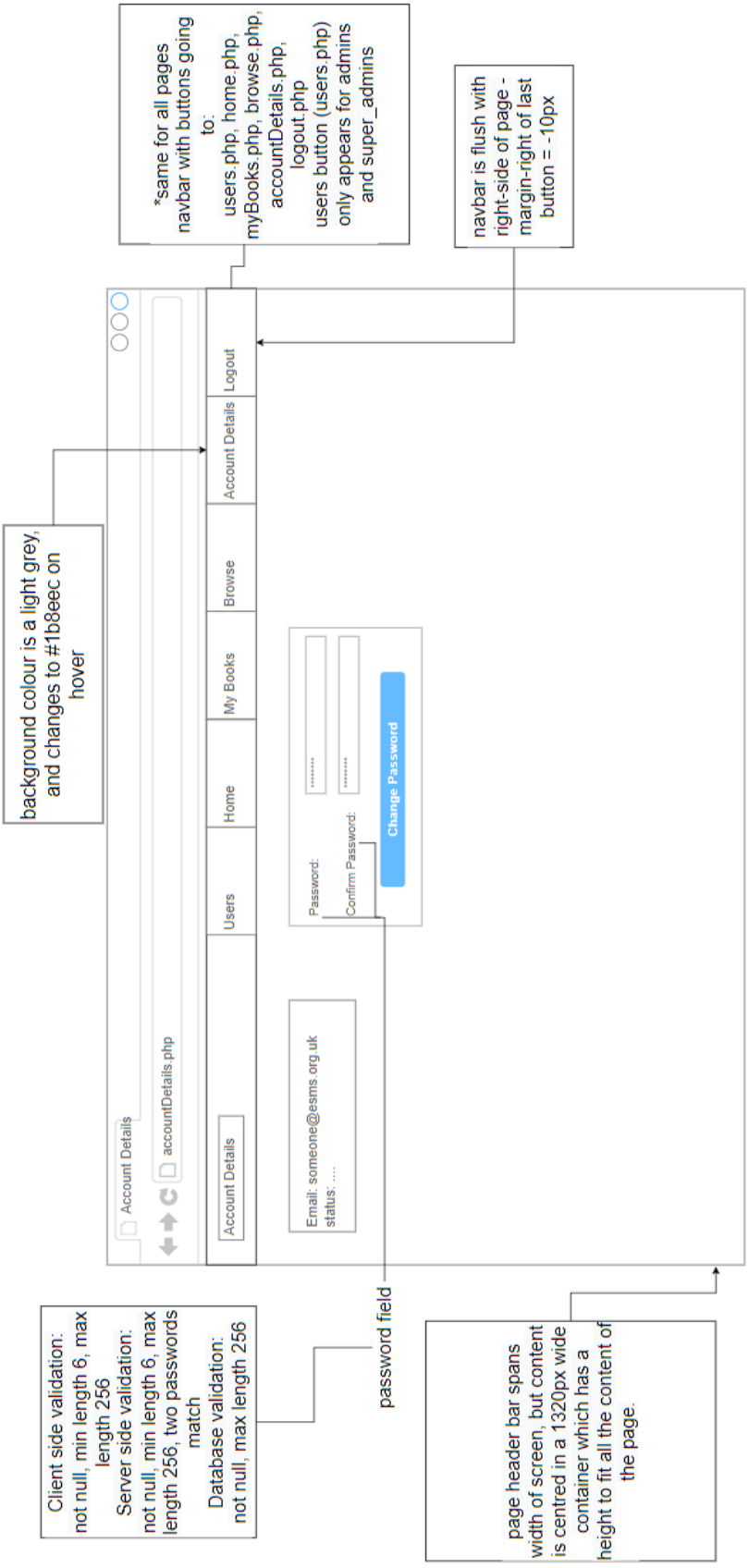
home.php


Home Page
home.php
Home
Welcome message and brief description as to what user can do
Browse
My Books
Account Details
Logout
Users
background colour: #1c73c5. Changes to #3f0090 on hover
Button going to myBooks.php
Button going to logout.php
Only available for admins and super_admins - button going to users.php
Button going to browse.php
Button going to accountDetails.php

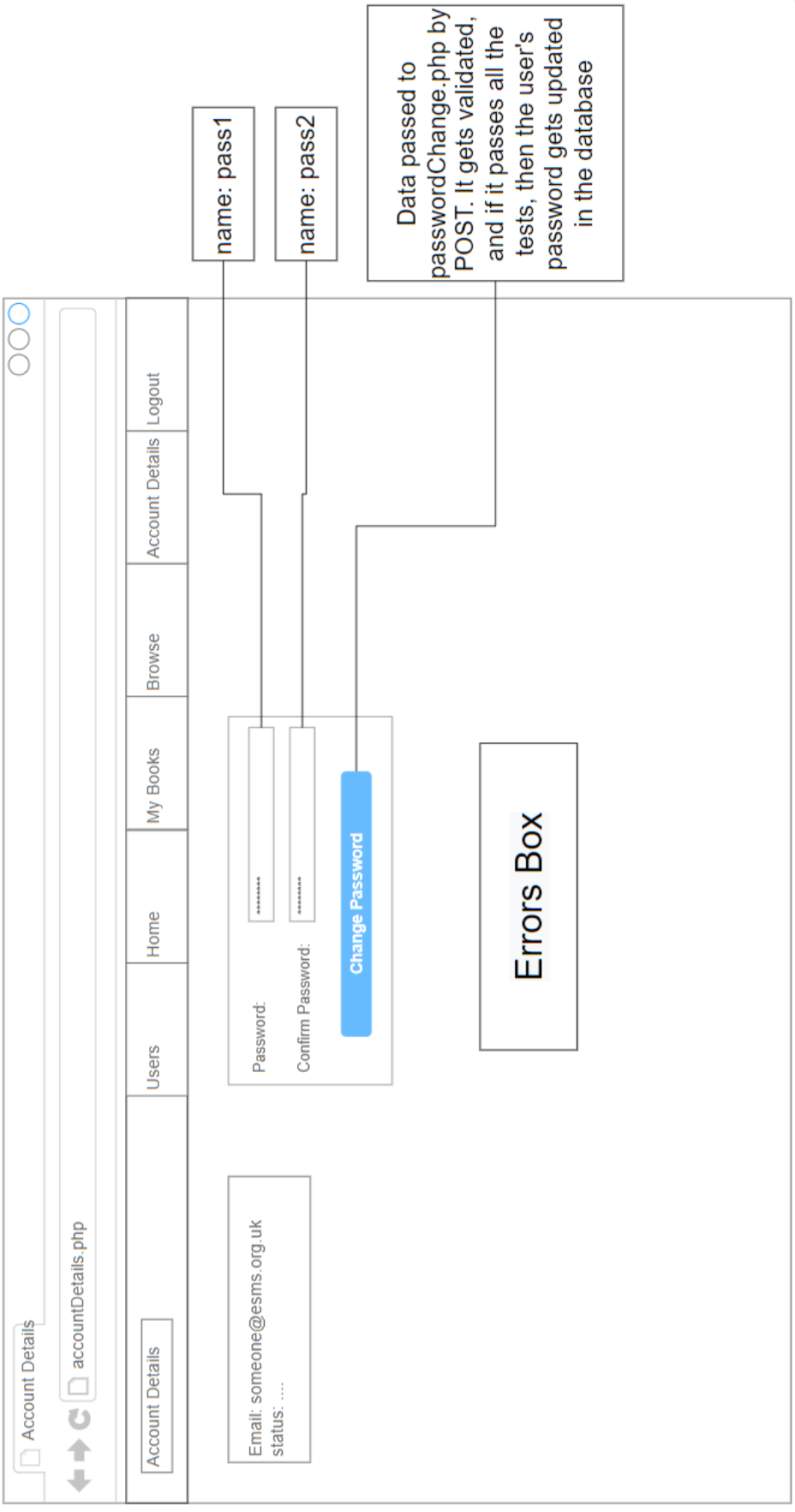
home.php – errors and dataflow

home.php – media query

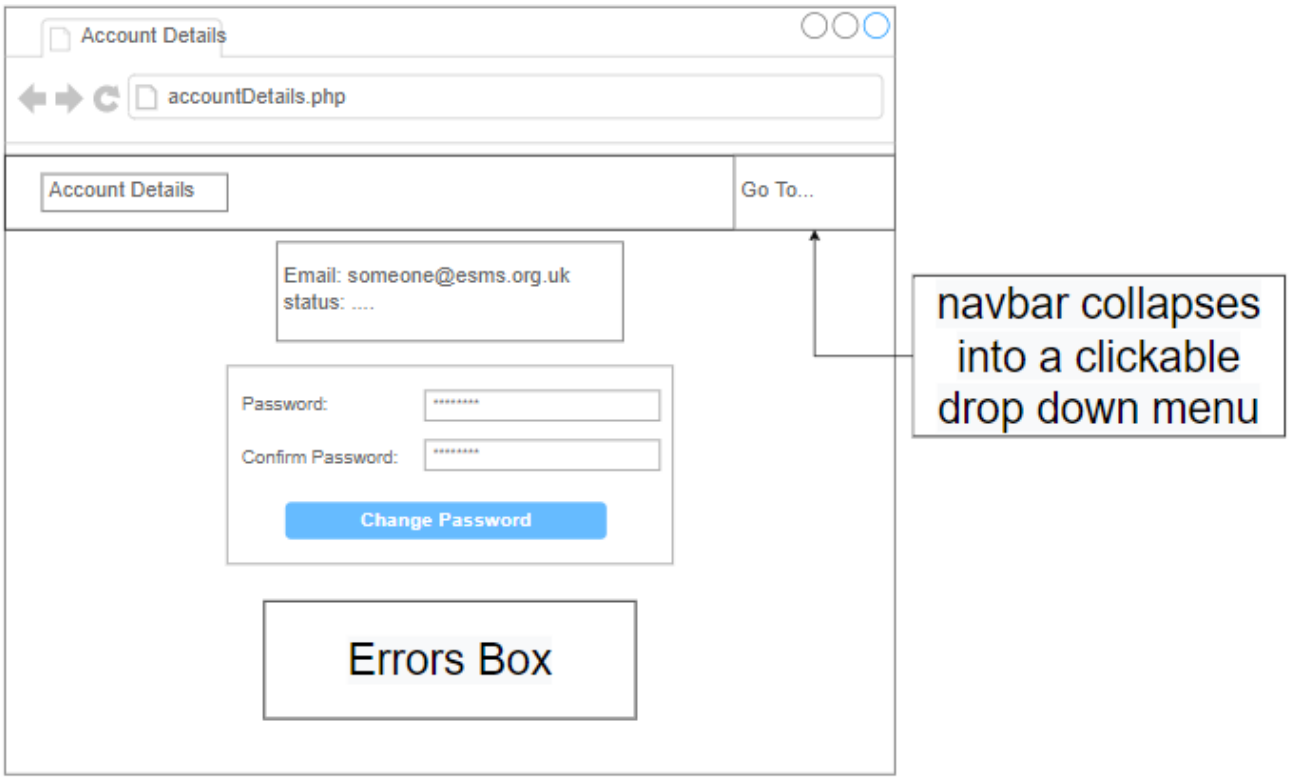
accountDetails.php



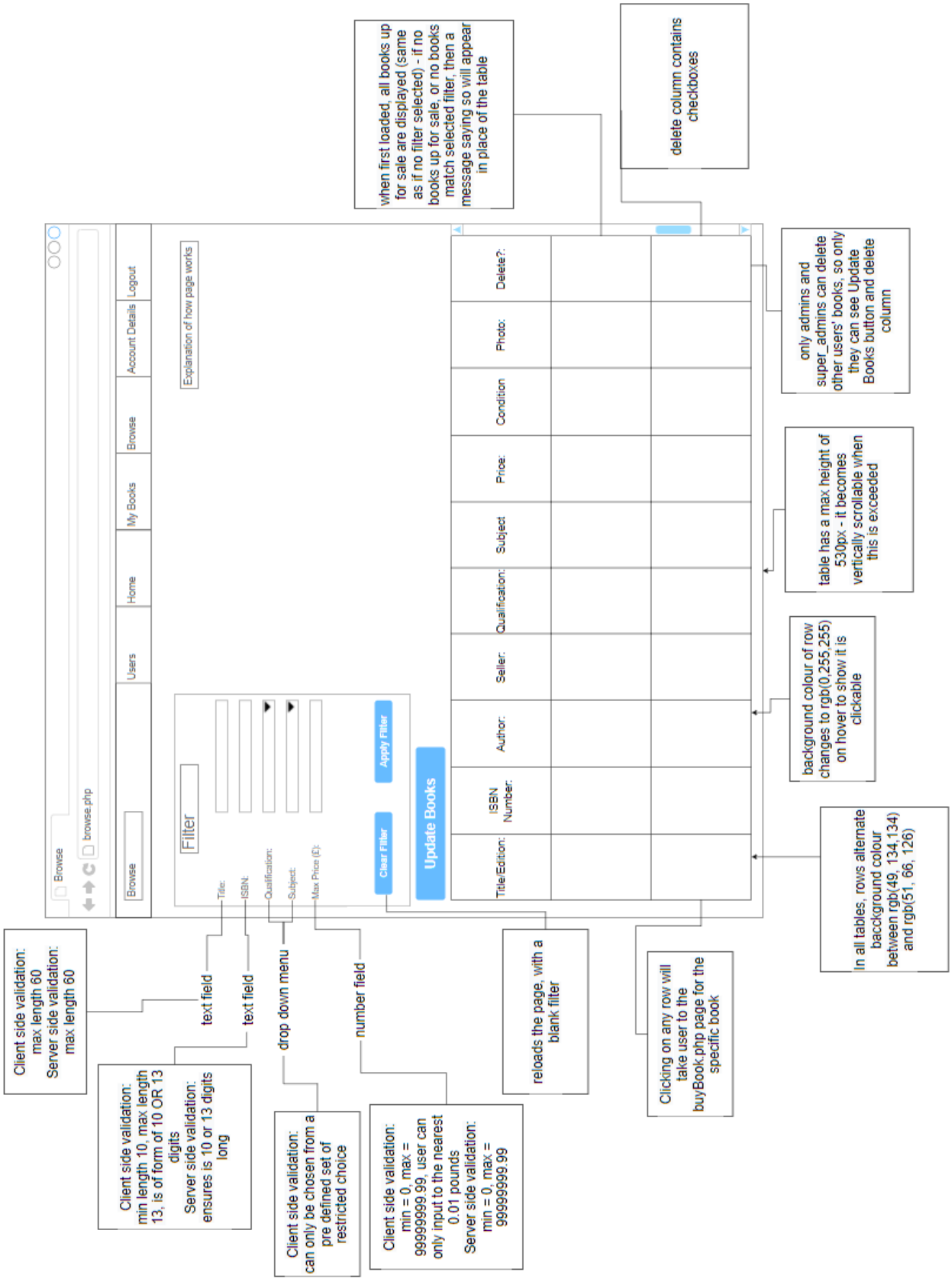
accountDetails.php – errors and data flow



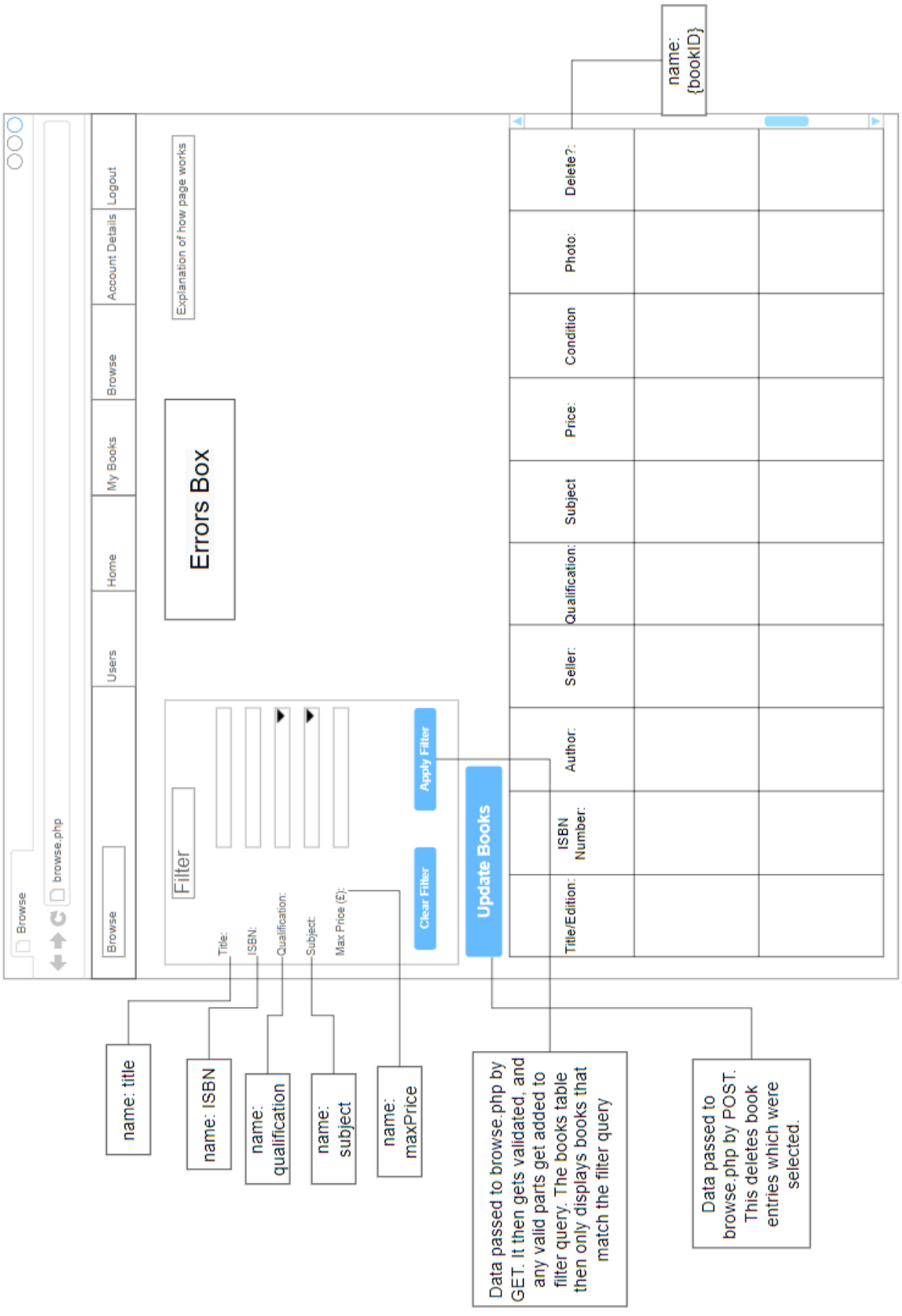
accountDetails – media query



browse.php

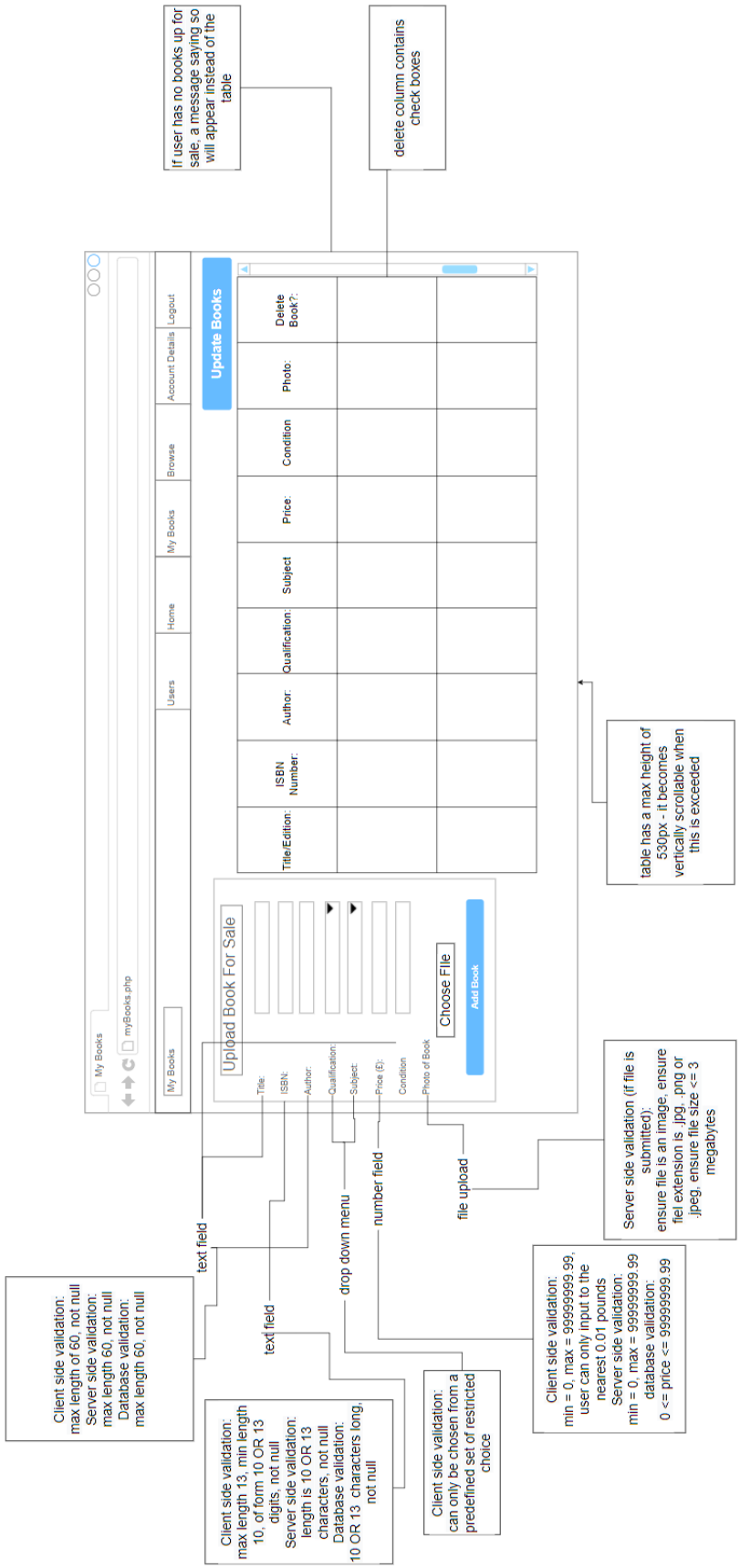

when first loaded, all books up for sale are displayed (same as if no filter selected) - if no books up for sale, or no books match selected filter, then a message saying so will appear in place of the table
delete column contains checkboxes
only admins and super_admins can delete other users' books, so only they can see Update Books button and delete column
table has a max height of 530px - it becomes vertically scrollable when this is exceeded
background colour of row changes to rgb(0,255,255) on hover to show it is clickable
In all tables, rows alternate backcground colour between rgb(49, 134,134) and rgb(51, 66, 126)
Clicking on any row will take user to the buyBook.php page for the specific book
reloads the page, with a blank filter
Client side validation: max length 60 Server side validation: max length 60
text field
text field
drop down menu
number field
Client side validation: min length 10, max length 13, is of form of 10 OR 13 digits Server side validation: ensures is 10 or 13 digits long
Client side validation: can only be chosen from a pre defined set of restricted choice
Client side validation: min = 0, max = 99999999.99, user can only input to the nearest 0.01 pounds Server side validation: min = 0, max = 99999999.99
Browse
Users
Home
My Books
Browse
Account Details
Logout
Explanation of how page works
Filter
Apply Filter
Clear Filter
Update Books
Delete?:
Photo:
Condition
Price:
Subject
Qualification:
Seller:
Author:
ISBN Number:
Title/Edition:

browse.php – errors and dataflow

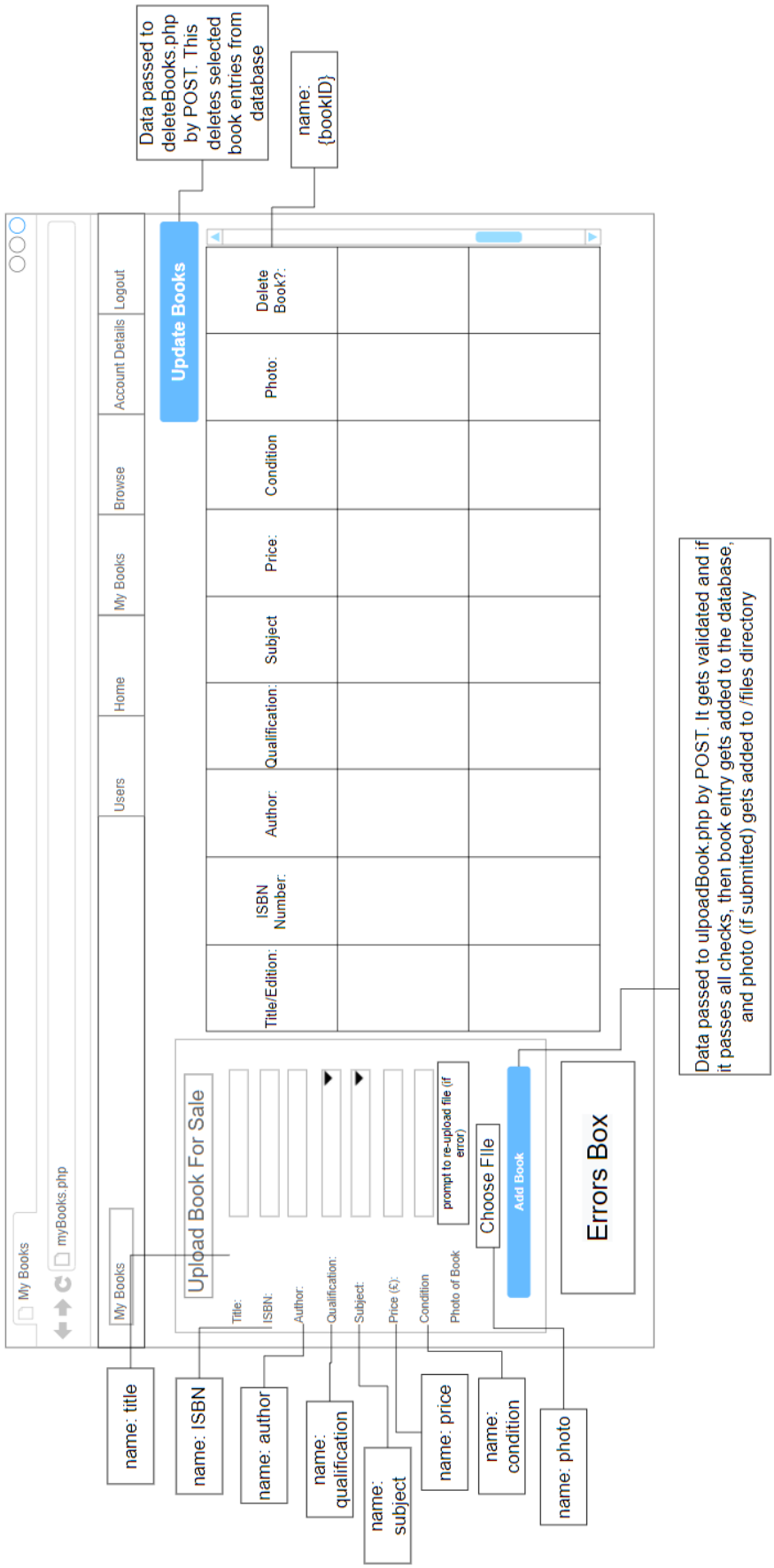


browse.php – media query

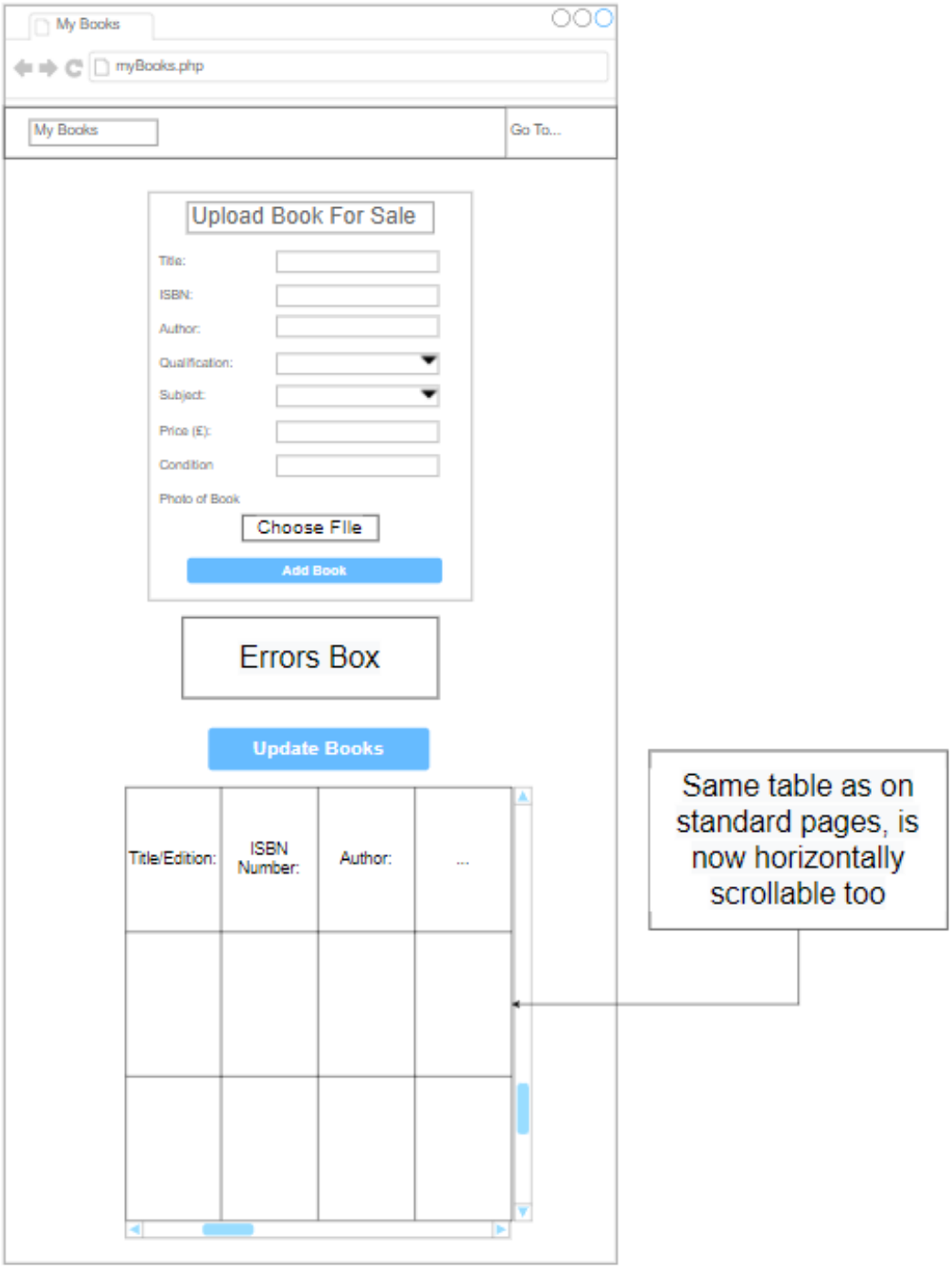
myBooks.php


If user has no books up for sale, a message saying so will appear instead of the table
delete column contains check boxes
My Books
myBooks.php
My Books
Users
Home
My Books
Browse
Account Details
Logout
Update Books
Title/Edition:
ISBN Number:
Author:
Qualification:
Subject
Price:
Condition
Photo:
Delete Book?:
Upload Book For Sale
Title:
ISBN:
Author:
Qualification:
Subject:
Price (£):
Condition
Photo of Book
Choose File
Add Book
table has a max height of 530px - it becomes vertically scrollable when this is exceeded
text field
text field
drop down menu
number field
file upload
Client side validation: max length of 60, not null
Server side validation: max length 60, not null
Database validation: max length 60, not null
Client side validation: max length 13, min length 10, of form 10 OR 13 digits, not null
Server side validation: length is 10 OR 13 characters, not null
Database validation: 10 OR 13 characters long, not null
Client side validation: can only be chosen from a predefined set of restricted choice
Client side validation: min = 0, max = 99999999.99, user can only input to the nearest 0.01 pounds
Server side validation: min = 0, max = 99999999.99
database validation: 0 <= price <= 99999999.99
Server side validation (if file is submitted): ensure file is an image, ensure file extension is .jpg, .png or .jpeg, ensure file size <= 3 megabytes

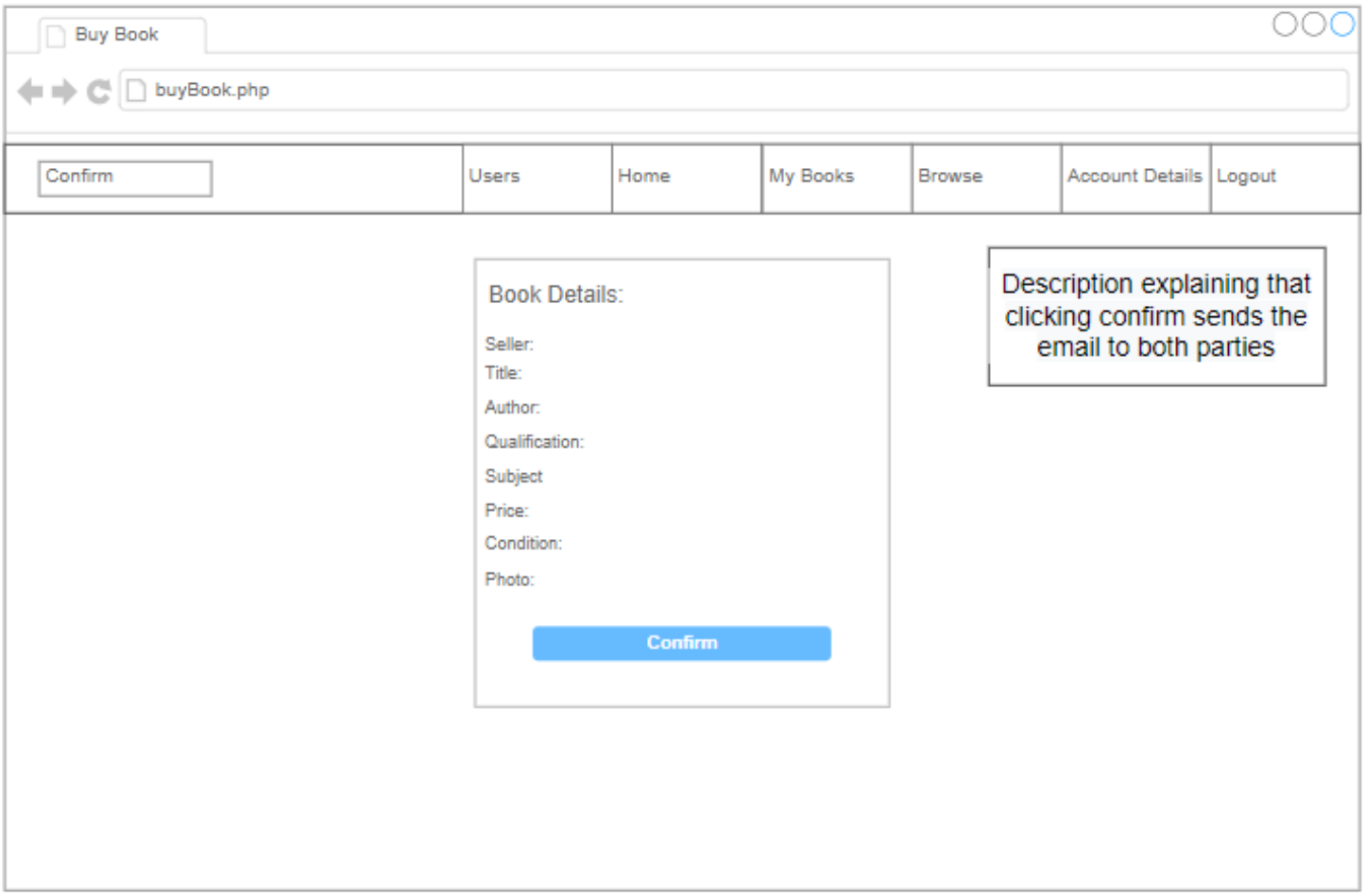
myBooks.php – errors and dataflow



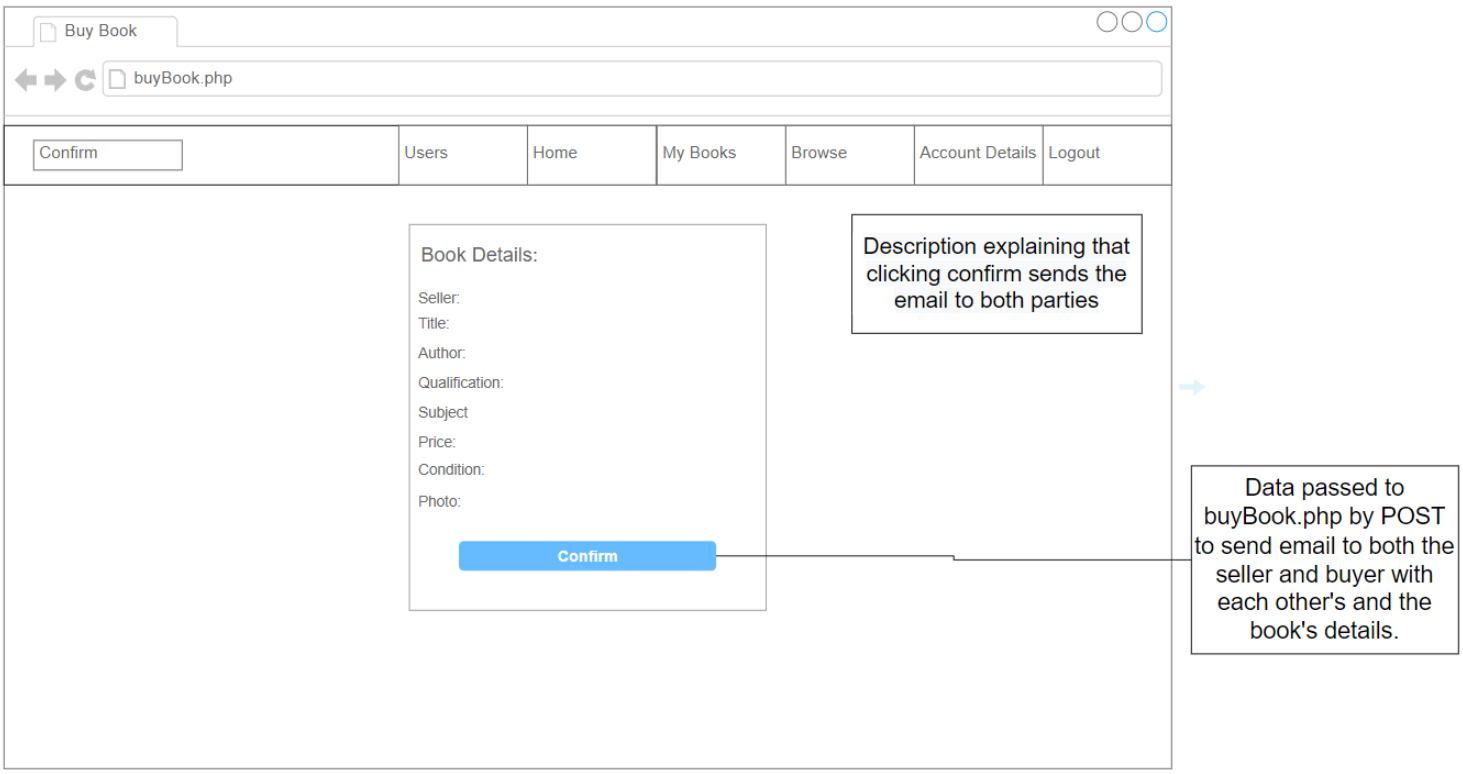
myBooks.php – media query



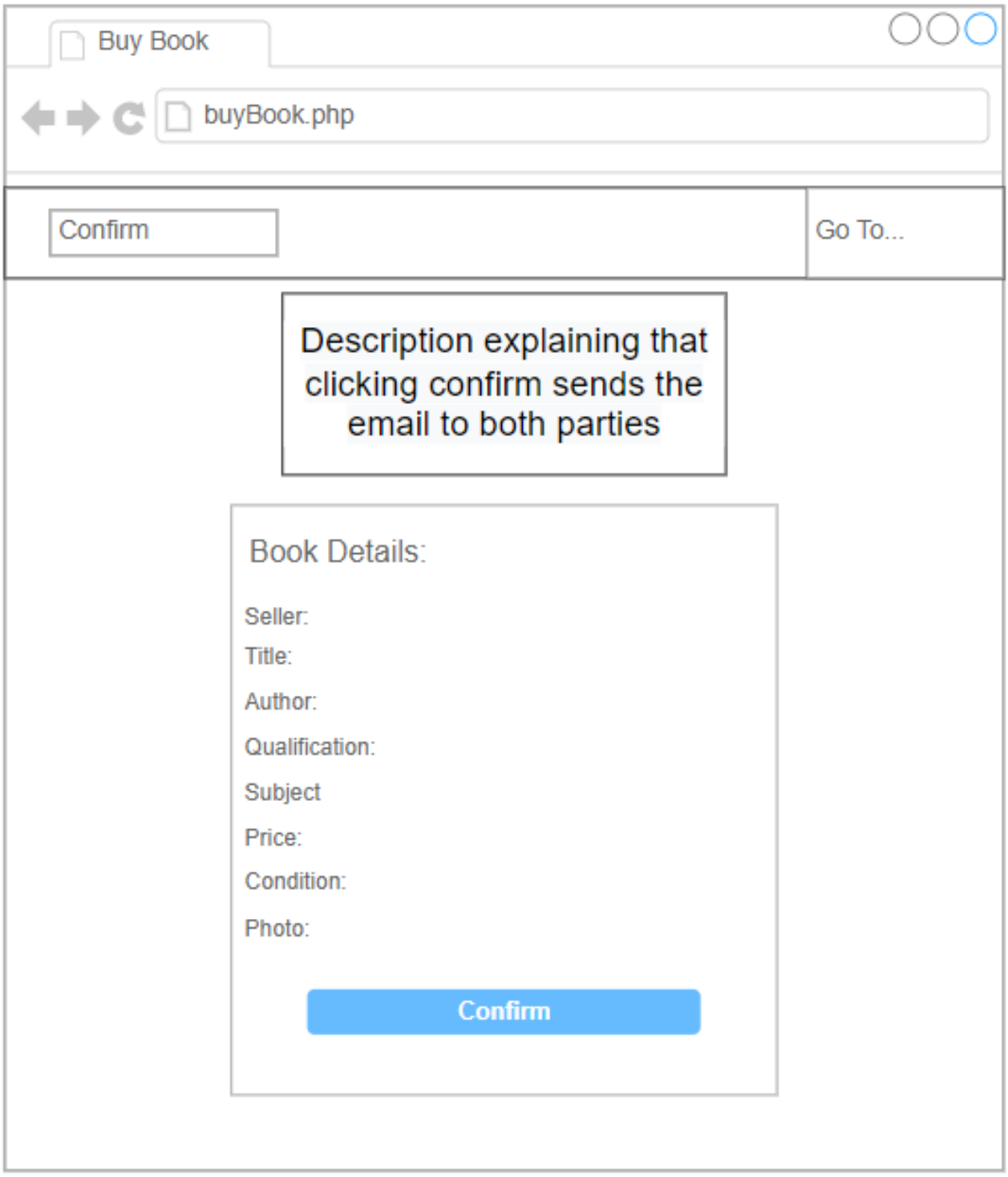
buyBook.php


Buy Book
buyBook.php
Confirm
Users
Home
My Books
Browse
Account Details
Logout
Book Details:
Seller:
Title:
Author:
Qualification:
Subject
Price:
Condition:
Photo:
Confirm
Description explaining that clicking confirm sends the email to both parties

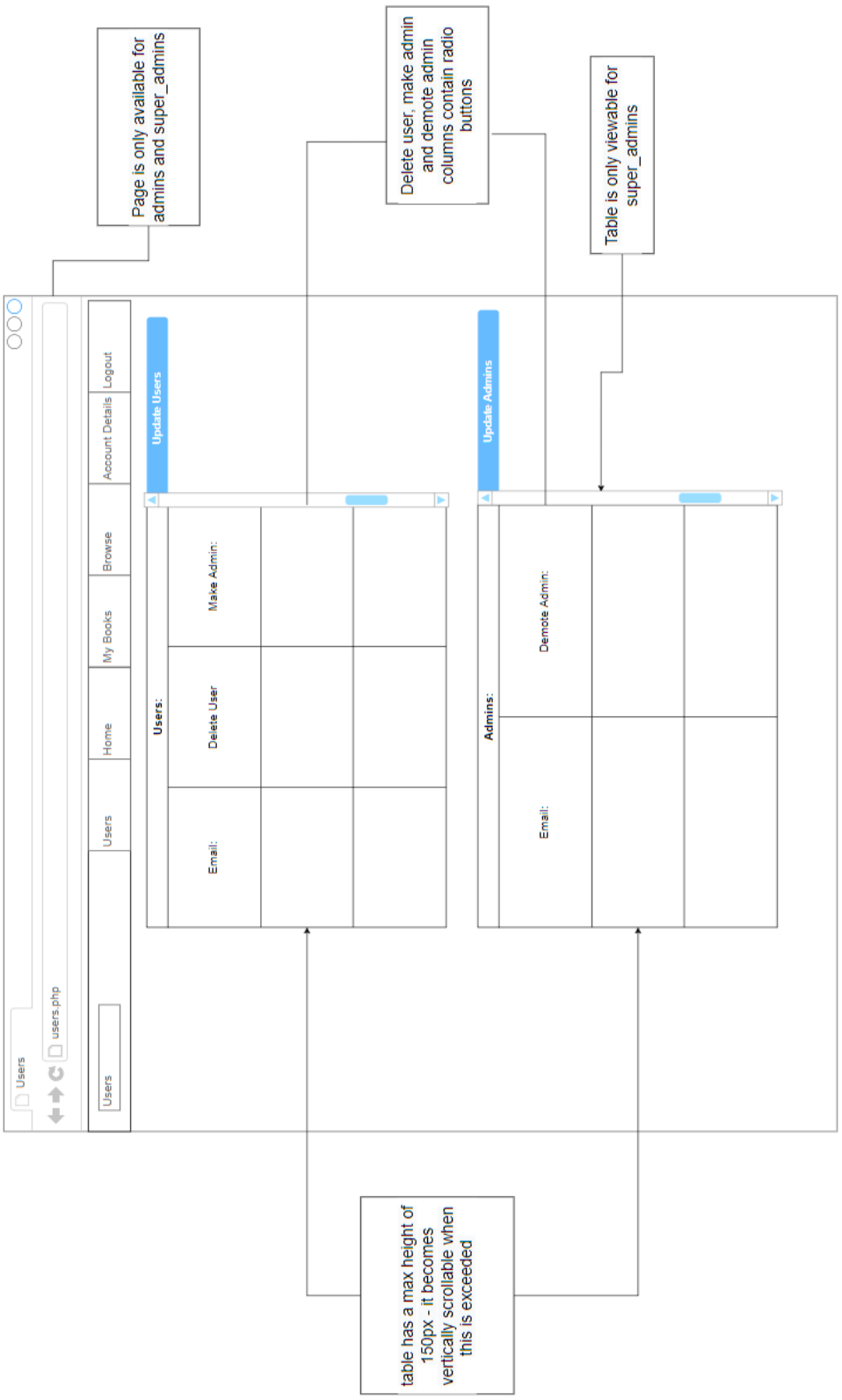
buyBook.php – errors and dataflow



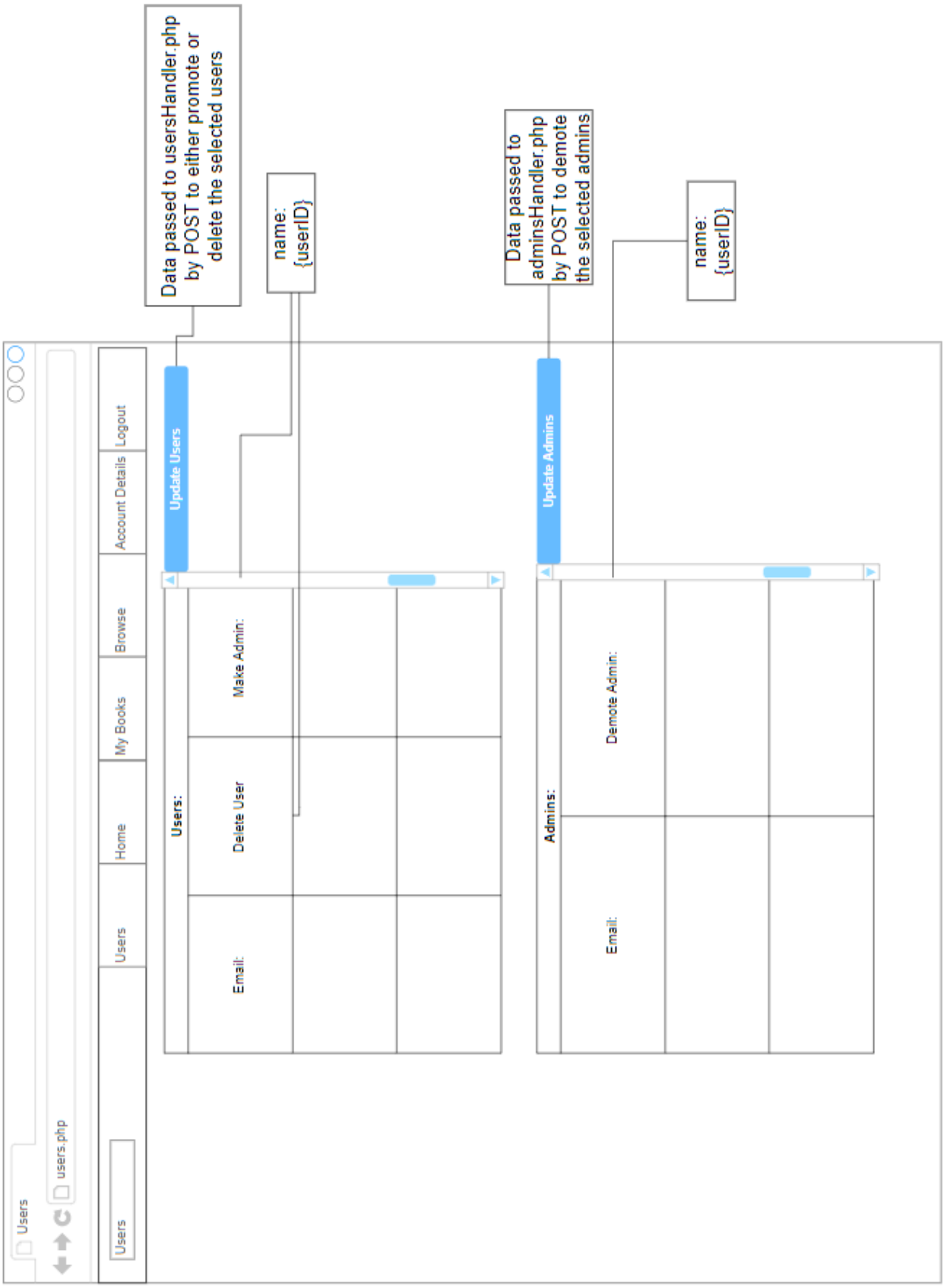
buyBook.php – media query



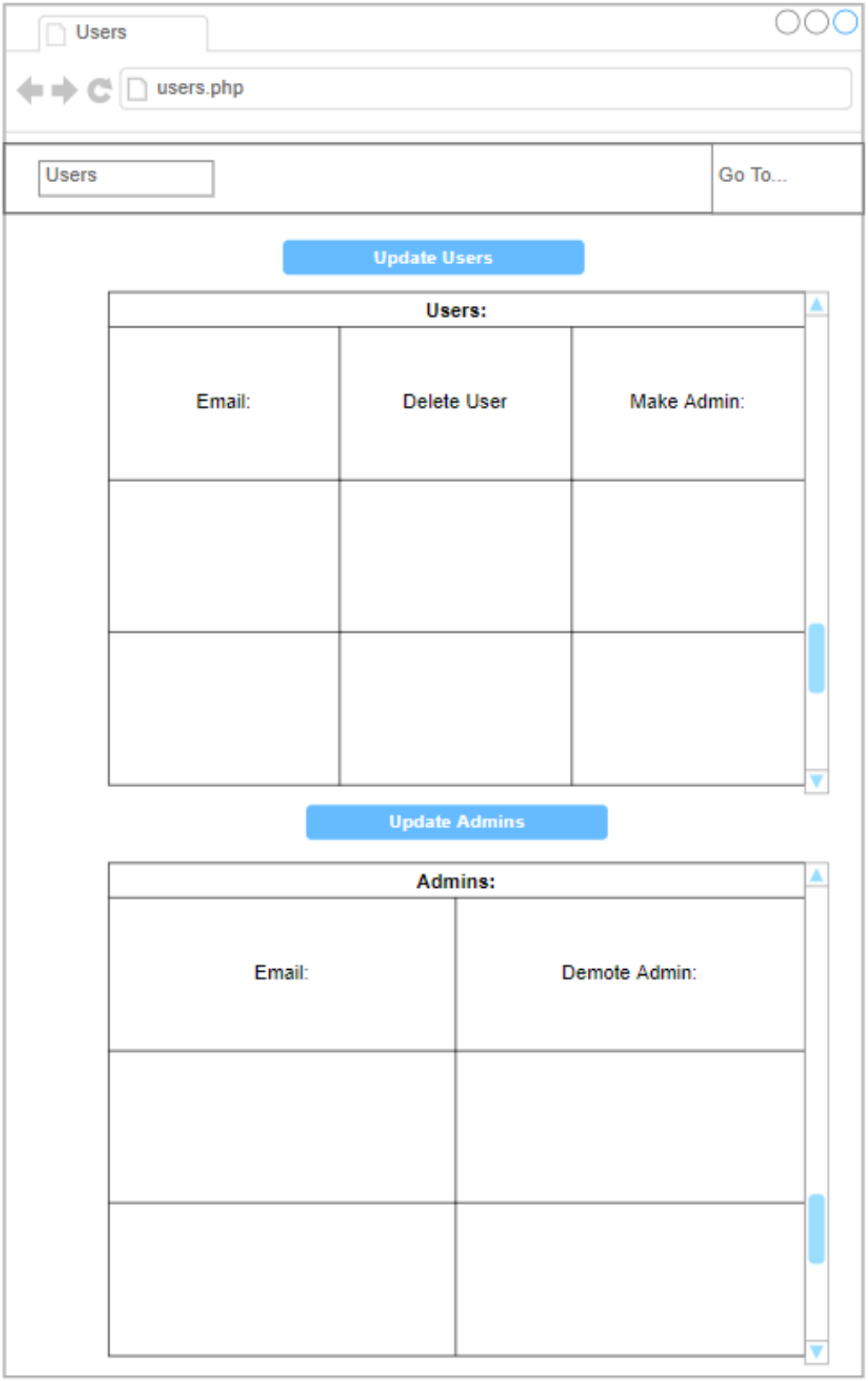
users.php


Page is only available for admins and super_admins
Delete user, make admin and demote admin columns contain radio buttons
Table is only viewable for super_admins
Users
users.php
Users
Users
Home
My Books
Browse
Account Details
Logout
Update Users
Users:
Make Admin:
Delete User
Email:
Update Admins
Admins:
Demote Admin:
Email:
table has a max height of 150px - it becomes vertically scrollable when this is exceeded

users.php – errors and dataflow



users.php – media query



Low-Fidelity Prototypes:

login.php

Login

ESMS Book Bank

Register

Login with Details:

Email Address:

someone@esms.org.uk

Password:

Login

Forgot Password

Login with Code:

Code:

Login

register.php

Register

ESMS Book Bank

Login

Register

Email Address:

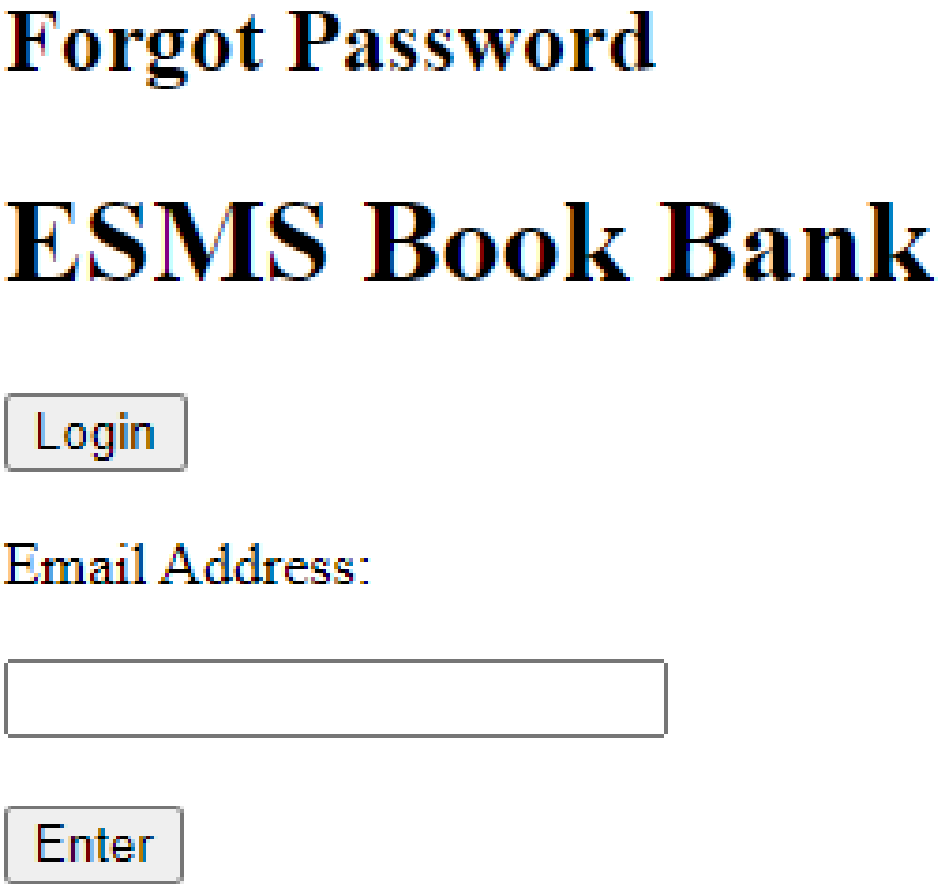
someone@esms.org.uk

Password:

Confirm Password:

Register

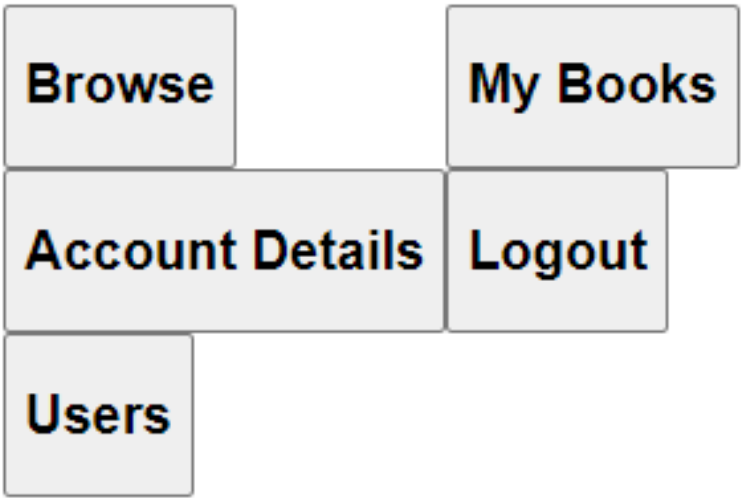
forgotPass.php



home.php

Home Page

Welcome noreplybooksale. Here you can change your details, browse books that have been put up for sale or manage your own books for sale.



accountDetails.php

Account Details

- Users
- Home
- My Books
- Browse
- Account Details
- Logout

Email: noreplybooksale@gmail.com

Status: super_admin

New Password:

Confirm Password:

Change Password

myBooks.php (navbar has been left out so it fits but it is the same as other pages)

Upload Book For Sale

Title/Edition:

ISBN Number:

1111111111111

Author:

Qualification Type:

Select a qualification

Subject:

Select a subject

Price (£):

Condition:

Photo of Book:

Choose File No file chosen

Add Book

Update Books

Title/Edition:	ISBN Number:	Author	Qualification	Subject	Price	Condition	Photo	Delete Book?
Learning PHP	1111111111111	me	higher	biology	£2.30	good	n/a	☐

browse.php

Browse

- Users
- Home
- My Books
- Browse
- Account Details
- Logout

Welcome to the browse page, feel free to use the filter to narrow down the selection, and then **click on a book** to be taken to the confirmation page.

Filter

Title:

ISBN:

1111111111111

Qualification:

Any

Subject:

Any

Max Price (£):

Apply Filter Clear Filter

Update Books

Title/Edition:	ISBN Number	Author:	Seller Email:	Qualification:	Subject:	Price:	Condition:	Photo:	Delete?
freedom of speech	1111111111111	you	zahramm@esms.org.uk	national5	chemistry	£2.30	Brand New	n/a	☐
Learning PHP	3333333333333	Bob Jones	zahramm@esms.org.uk	advancedHigher	economics	£10.00	good	n/a	☐

buyBook.php

Confirm

- Home
- My Books
- Browse
- Account Details
- Logout

Clicking **confirm** will send both yourself and the seller of this book an email with the necessary details to communicate together to complete the transaction of the book.

Book Details:

Seller: noreplybooksale@gmail.com

Title: French Dictionary

ISBN: 3333333333333

Author: Jeff White

Qualification: higher

Subject: french

Price: £8.99

Condition: Slightly Used

Photo : n/a

Confirm

users.php

Users

- Users
- Home
- My Books
- Browse
- Account Details
- Logout

Users

Update Users

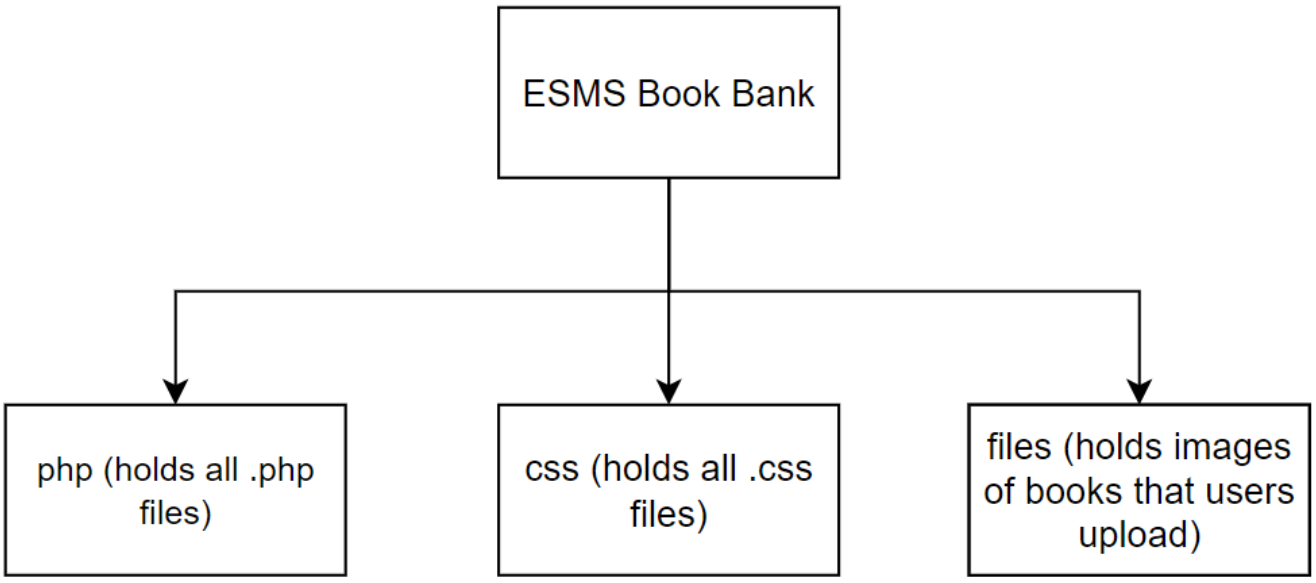
Email:	Delete:	Make Admin:
zahramm@esms.org.uk	○	○

Admins

Update Admins

Email:	Demote Admin:
bobjones@esms.org.uk	○

Folder Structure for Files:



Code Design for PHP processes – Pseudocode

Register a new user – registerHandler.php:

1. Connect to database
2. Initialize errors array
3. Get email and both passwords from form and set to variables
4. Validate email (add to errors array as needed):
    a. Check email isn't empty
    b. Check that email is of the form '…@esms.org.uk'
    c. Check email isn't > 60 characters
5. Validate passwords (add to errors array as needed):
    a. Check that both passwords aren't empty
    b. Check that both passwords match
    c. Check that password length is > 5 and <= 256
6. If errors array is empty:
    a. Query to see if email address is already in use
    b. If it is, query to check if user_type = 'pending'
    c. If it is pending, delete it from the database so it can be overwritten, if not, then email is already in use and has been verified by use of issued verification code so this isn't a valid email to register with – add to errors array
7. If errors array is empty:
    a. Generate random unique 10 digit code
    b. Insert user details with code into database
    c. Email user with registration code
8. Else:
    a. Display errors

Log in with email and password (loginHandler.php):

1. Connect to databse
2. Initialize errors array
3. Get email and password from form and assign to variables
4. Validate details (add to errors array as necessary):
    a. Check to make sure email isn't empty
    b. Check to make sure password isn't empty
5. If errors array is empty:
    a. SELECT all entries from database that match with entered email and password
    b. If the result returned is not 1 single entry, then the combination is invalid, add this to the errors array
    c. Else:
        i. From the returned results, if the user_type = 'pending', then add to the errors array that the account hasn't been verified with a registration code yet

6. If the errors array is empty:
    a. Start a session
    b. Add the userID to the session
    c. Add the email to the session
    d. Add the user_type to the session
    e. Load home.php
7. Else:
    a. Display errors

Login with verification code (codeHandler.php):

1. Connect to database
2. Initialize errors array
3. Get code from form and assign it to a variable
4. Check if code is empty, if so, add to errors array
5. Else:
    a. SELECT timeOfCode FROM users table WHERE code = '<entered code>'
    b. If there are no results, then code is invalid, so add to errors array
    c. Else:
        i. Set timeOfCode to a DateTime object by passing in the timeOfCode string from the query result
        ii. Set currentTime to a DateTime object with the current time stamp
        iii. Set diff to a DateInterval object by passing in timeOfCode and currentTime to date_diff()
        iv. Set timeElapsed to 0
        v. Add number of seconds from diff to timeElapsed by accessing each time property (seconds, minutes, hours, days, months, years) of the diff object and converting each to seconds
        vi. If timeElapsed > 5 minutes (5 * 60 seconds):
            1. Code has expired, add to errors array and suggest redirection to register page so a new code can be issued
6. If errors array is empty:
    a. If user who entered the code has user_type = 'pending', update it so that their user_type = 'user' (verifying email upon logging in for the first time)
    b. SELECT all details FROM users table of user with corresponding code
    c. Start session
    d. Add the userID to the session
    e. Add the email to the session
    f. Add the user_type to the session
    g. Load home.php
7. Else:
    a. Display errors

Issue a new verification code for forgot password feature (forgotPassHandler):

1. Connect to database
2. Get user's email from form and assign it to a variable
3. Query database to select details of users with an email that matches the entered one
4. If the query returns a result:
    a. Generate a random unique 10 digit code
    b. Update users table for record of user which matches email entered – set code to new code and timeOfCode to current time stamp
    c. Email user new code

Display username (home.php):

1. Start a session
2. Get user's email from session variable
3. Split user's email on '@' symbol and take first element as username
4. Display username

Display a user's books up for sale (myBooks.php):

1. Connect to database
2. Start a session
3. Get userID from session variable
4. Query database, selecting all details from books table where userID = '<userID from session variable>'
5. If there are no entries, display 'No books up for sale yet'
6. Else:
    a. Display all the details of each book in a table with the option (provided by a form) to delete each book

Allow a user to delete books they have put up for sale (deleteBooks.php):

1. Connect to database
2. Get bookIDs for deleting from POST variable
3. Loop for each bookID in POST variable:
    a. Query books table, selecting photo where bookID = '<current bookID>'
    b. If query returns a result:
        i. Delete from books table where bookID = '<current bookID>'
        ii. If current photo is not = 'n/a':
            1. Set fname to 'files/' + photo
            2. If exists(fname):
                a. Delete fname (photo file in files/ directory)

Allow a user to upload a new book for sale (uploadBook.php):

1. Connect to database
2. Initialize errors array
3. Get book details from form and assign them to variables
4. Validate form data (add to errors array as necessary):
    a. Check that title, ISBN number, author, qualification, subject, price and condition aren't empty
    b. Check that title <= 60 characters
    c. Check that ISBN number is 10 or 13 digits
    d. Check that author is <= 60 characters
    e. Check that 0 <= price <= 99999999.99
    f. Check that condition <= 60 characters
    g. If filename is not empty (add to errors array as necessary):
        i. Ensure file is actually an image by trying to get the image size
        ii. Ensure extension of file is '.jpg', '.png', or '.jpeg'
        iii. Ensure filesize <= 3Mb
5. If errors array is empty:
    a. Start a session
    b. Get userID from session variable
    c. If a file was submitted:
        i. Set newFname to '<current userID>' + current time stamp (format yymmddhhmmssuuuuuu) + file extension
        ii. Upload file to 'files/' + newFname
        iii. If there was an error uploading the file:
            1. Add 'Error when uploading photo' to errors array
    d. Else:
        i. Set newFname to 'n/a'
    e. Insert into books table all book details from form and newFname
6. Else:
    a. Display errors

Allow a user to change their password (passwordChange.php):

1. Connect to database
2. Initialize errors array
3. Get both passwords from form and assign to variables
4. Validate passwords (add to errors array as necessary):
    a. Check passwords aren't empty
    b. Check passwords match
    c. Check that passwords are > 5 characters and <= 256 characters
5. If errors array is empty:
    a. Start session
    b. Get userID from session variable

c. Update users table, set password to new password where userID = '<current userID>'

6. Else:
    a. Display errors

Display books to browse that are included in the filter (browse.php):

1. Connect to database
2. Initialize errors array
3. Start session
4. Get userID from session variable
5. Set query to "SELECT * FROM books, users WHERE users.userID = books.userID AND NOT users.userID = '<current userID>'" (don't display user's own books for sale)
6. Get and validate data from form (add to errors array whenever a constraint is broken – a field being empty is not a breach here as then it just won't be included in the filter):
    a. If title not empty:
        i. If title > 60 characters add to errors array
        ii. Else add "AND title LIKE '%<given title>%'" to query
    b. If ISBN number not empty:
        i. If ISBN number is not 10 or 13 characters, add to errors array
        ii. Else if ISBN contains a character which is not numeric, add to errors array
        iii. Else add "AND ISBN = '<given ISBN number>'" to query
    c. If qualification is not empty:
        i. Add "AND qualification = '<given qualification>'" to query
    d. If subject is not empty:
        i. Add "AND subject = '<given subject>'" to query
    e. If max price is not empty:
        i. If max price < 0 or max price > 99999999.99 add to errors array
        ii. Else add "AND price <= '<given max price>'" to query
    f. Add "ORDER BY title ASC" to query
    g. Execute query
    h. If query returns no results:
        i. Display "No books matched this search"
    i. Else:
        i. Display book details in a table
        ii. Get user_type from session
        iii. If user_type = admin or super_admin:
            1. Add option to delete each book
7. If errors array is not empty:
    a. Display each error

Allow an admin or super_admin to delete a book another user has put up for sale (browse.php):

1. Connect to database
2. If POST variable is not empty:
    a. Loop for each bookID in POST:
        i. SELECT photo from books table where bookID = '<current bookID>'
        ii. Delete from books table where bookID = '<current bookID>'
        iii. If photo is not = 'n/a':
            1. Set fname to 'files/' + photo:
            2. If fname exists:
                a. Delete fname

Allow a user to send an email to a seller with their details (buyBook.php):

1. Start session
2. Get email from session
3. Get email from POST (email of seller)
4. Send email to seller with details of interested buyer and book
5. Send email to interested buyer (current user) with details of seller and book

Display all users of user_type = 'user' to admins and super_admins with option to delete/promote (users.php):

1. Connect to database
2. SELECT userID, email from users table where user_type = 'user' ORDER BY email ASC
3. Display details in a table in a form with the option to delete or promote each user

Display all users of user_type = 'admin' to super_admins with option to demote (users.php):

1. Connect to database
2. SELECT userID, email from users table where user_type = 'admin' ORDER BY email ASC
3. Display details in a table in a form with the option to demote each admin

Allow an admin or super_admin to promote a user of user_type = 'user' to admin status or delete the user and all their books (usersHandler.php):

1. Connect to database
2. Get POST variable (associate array where keys are userIDs and values are either 'admin' or 'delete')
3. Loop for each userID in POST:
    a. If value = 'admin':
        i. Update users table, set user_type = 'admin' where userID = '<current userID>'
    b. If value = 'delete':

i. SELECT photo FROM books table WHERE userID = '<current userID>'
ii. Loop for each photo:
    1. If photo is not = 'n/a':
        a. Set fname to 'files/' + photo
        b. If fname exists:
            i. Delete fname
iii. DELETE from books table where userID = '<current userID>'
iv. DELETE FROM users table where userID = '<current userID>' AND user_type = 'user'

Allow super_admin to demote admin users to user_type = 'user' (adminsHandler.php):

1. Connect to database
2. Get POST variable (associate array where keys are userIDs and values are 'demote')
3. Loop for each userID in POST variable:
    a. UPDATE users table, set user_type = 'user' WHERE userID = '<current userID>'

Load a given page (loadPage.php):

1. Function load(page):
    a. Set URL to 'http://' + current domain + current directory
    b. Remove trailing slashes from URL
    c. Add '/' + page to end of URL
    d. Then load URL

Refinement of generating a unique 10 digit code:

1. Function genCode():
    a. Set character set to [a-z, A-Z, 0-9, @, _, -, .]
    b. Set code to empty string
    c. Loop ten times:
        i. Generate random number from 0 to length of character set – 1
        ii. Append character in character set at index of random number to code
    d. Query from users table to see if code appears in it
    e. If it appears in it (so query returns a result), repeat from beginning until code is unique
    f. If it is unique, then return the code

Refinement of sending an email:

1. Function email(recipientEmail, subject, body, altBody):
    a. Create a new PHPMailer object
    b. Set the sender email address and name to 'noreplybooksale@gmail.com' and 'Admin'
    c. Set the receiving email address to recipientEmail passed in

d. Set body content type to HTML (for formatting purposes)
e. Set the email subject to subject passed in
f. Set the body to the body passed in
g. Set the alternative body to altBody passed in
h. Tell the system to use SMTP to send the email
i. Set the host to ‘smtp.gmail.com’
j. Ask to use SMTP authentication
k. Set encryption system to ‘tls’ (not ssl)
l. Set username to ‘noreplybooksale@gmail.com’
m. Set password to SMTP password
n. Set port to 25
o. Send the email

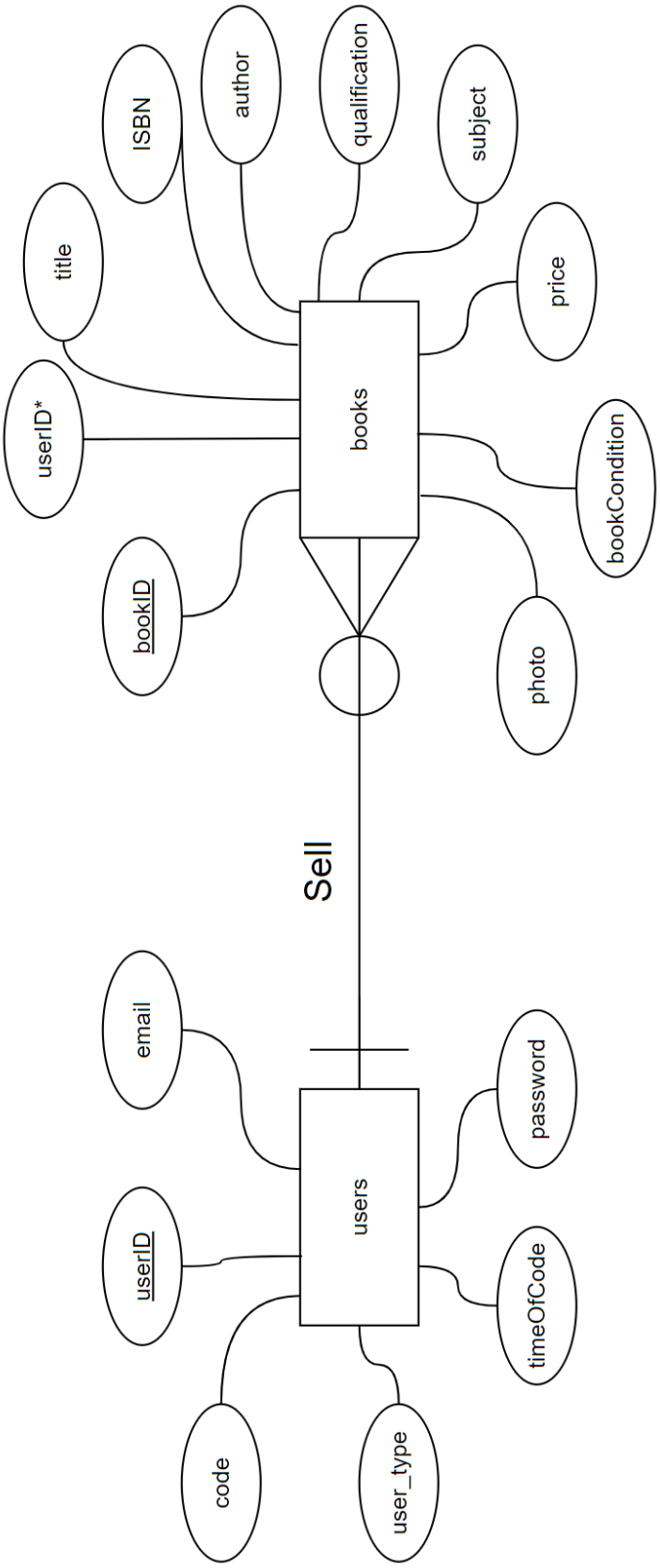
Database Design:

Data Dictionaries:

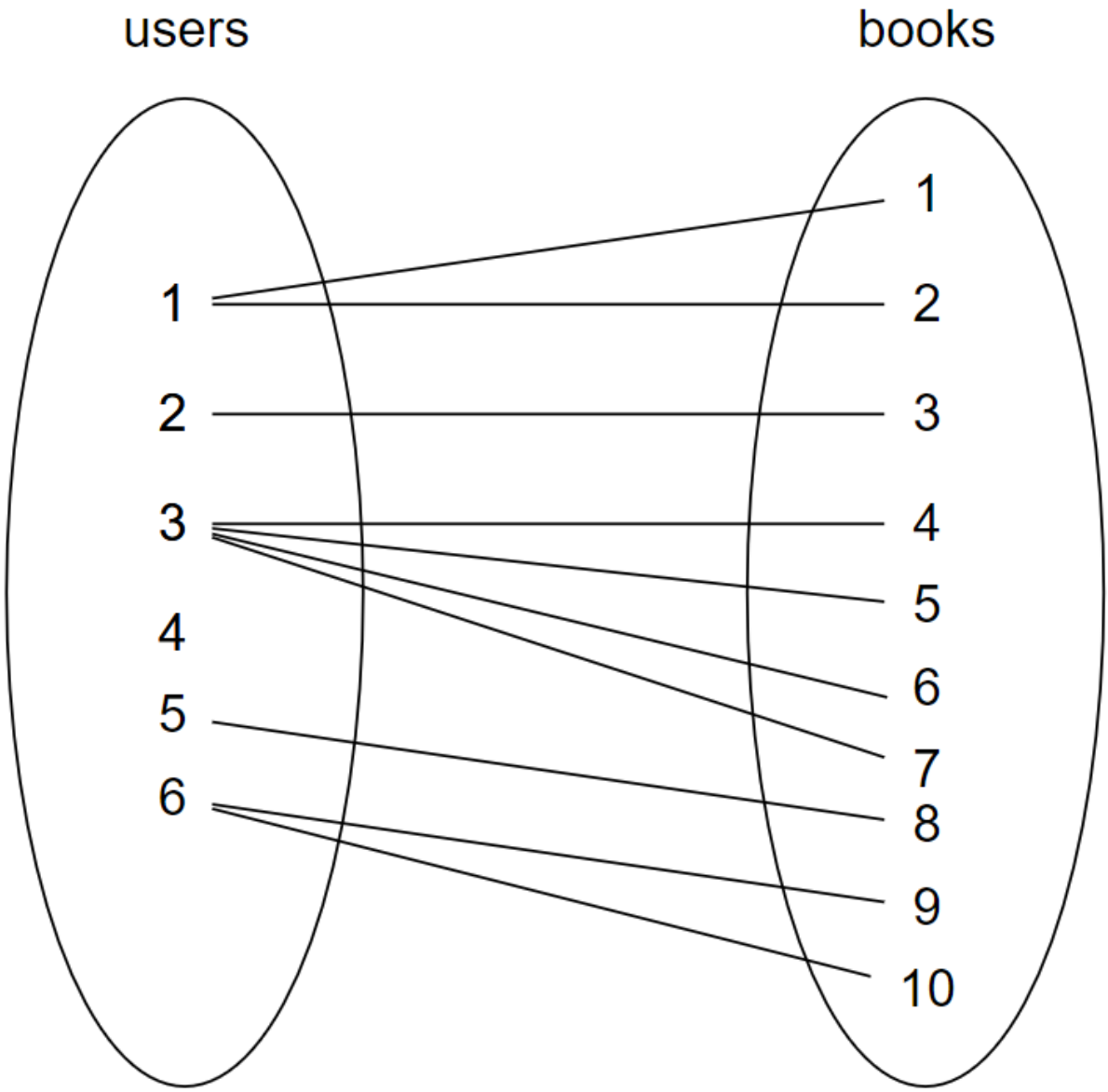
Entity: users					
Attribute Name	**Key**	**Type**	**Size**	**Required**	**Validation**
userID	PK	Integer		Yes	Auto increment
email		varchar	60	Yes	Unique
password		varchar	256	Yes	
user_type		varchar	15	Yes	Restricted Choice: super_admin, admin, user, pending
code		varchar	256	No	Unique
timeOfCode		timestamp		No	

Entity: books					
Attribute Name	**Key**	**Type**	**Size**	**Required**	**Validation**
bookID	PK	Integer		Yes	Auto increment
userID	FK	Integer		Yes	Must exist in users table
title		varchar	60	Yes	
ISBN		varchar	13	Yes	Must be of length 10 or 13
author		varchar	60	Yes	
qualification		varchar	60	Yes	
subject		varchar	60	Yes	
price		decimal(10,2)		Yes	Price >= 0 and <= 99999999.99
bookCondition		varchar	60	Yes	
photo		varchar	255	Yes	

Entity Relationship Diagram:



Entity Occurrence Diagram



Query Designs:

adminsHandler.php

Change user_type from 'user' to 'admin' (demote an admin)	
UPDATE TABLE	users
SET	user_type = 'user'
CONDITION(S)	userID = '<given userID>'

browse.php

Display books to meet filter (presence of conditions in curly braces depends on the filter the user applies)	
SELECT FIELDS	*
TABLE(S)	books, users
CONDITION(S)	users.userID = books.userID NOT users.userID = '<userID of current user>' {title LIKE '%<given title>%'} {ISBN = '<given ISBN number>'} {qualification = '<given qualification>'} {subject = '<given subject>'} {price <= '<given max price>'}
ORDER BY	Title ASC

Select file names of book photos for deleting	
SELECT FIELDS	photo
TABLE(S)	books
CONDITION(S)	bookID = '<given bookID>'

Delete books admins select	
DELETE FROM TABLE	books
CONDITION(S)	bookID = '<given bookID>'

codeHandler.php

Find when a certain code was issued	
SELECT FIELDS	timeOfCode
TABLE(S)	users
CONDITION(S)	code = SHA2('<given code>', 256)

Change user_type from 'pending' to 'user' when a valid code is entered and a user first logs in	
UPDATE TABLE	users
SET	user_type = 'user'
CONDITION(S)	code = SHA2('<given code>', 256) user_type = 'pending'

Get details of user corresponding to entered code	
SELECT FIELDS	*
TABLE(S)	users
CONDITION(S)	code = SHA2('<given code>', 256)

deleteBooks.php

Get image name to delete	
SELECT FIELDS	photo
TABLE(S)	books
CONDITION(S)	bookID = '<given bookID>'

Delete book entries from database	
DELETE FROM TABLE	books
CONDITION(S)	bookID = '<given bookID>'

email.php

See if there's a match for a code to test for uniqueness	
SELECT FIELDS	*
TABLE(S)	users
CONDITION(S)	code = SHA2('<given code>', 256)

forgotPassHandler.php

Get user details of a corresponding email	
SELECT FIELDS	*
TABLE(S)	users
CONDITION(S)	email = '<given email>'

Update code and timeOfCode when a new code is requested with 'forgot password' feature	
UPDATE TABLE	users
SET	code = SHA2('<generated code>', 256), timeOfCode = NOW()
CONDITION(S)	email = '<given email>'

loginHandler.php

SELECT user details to see if there is a username and password match	
SELECT FIELDS	*
TABLE(S)	users
CONDITION(S)	email = '<given email>' code = SHA2('<given code>', 256)

myBooks.php

SELECT all of the details of all the books a user has for sale for display	
SELECT FIELDS	*
TABLE(S)	books
CONDITION(S)	userID = '<current user's userID>'
ORDER BY	title ASC

passwordChange.php

Update a user's details with a new password	
UPDATE TABLE	users
SET	password = SHA2('<given new password>', 256)
CONDITION(S)	userID = '<current user's userID>'

registerHandler.php

SELECT details to see if email entered is already in use	
SELECT FIELDS	*
TABLE(S)	users
CONDITION(S)	email = '<given email>'

See if an email in use is 'pending' (so code can be overwritten) by seeing if this query returns anything	
SELECT FIELDS	*
TABLE(S)	users
CONDITION(S)	email = '<given email>' user_type = 'pending'

Delete registered account which was never activated with code (so code has been overwritten)	
DELETE FROM TABLE	users
CONDITION(S)	email = '<given email>'

Insert new registered user details into database	
INSERT	email password user_type code timeOfCode
TABLE(S)	Users
VALUES	'<given email>' SHA2('<given password>', 256) 'pending' SHA2('<generated code>', 256) NOW()

uploadBook.php

Insert details of book put up for sale	
INSERT	userID title ISBN author qualification subject price bookCondition photo
TABLE(S)	Books
VALUES	'<current userID>' '<given title>' '<given ISBN number>' '<given author>' '<given qualification>' '<given subject>' '<given price>' '<given condition>' '<file name (if a file is provided, it will be the files/userID concatenated with the current timestamp of the format yymmddhhmmssuuuuuu, otherwise will be 'n/a')>'

users.php

SELECT all user emails and IDs of user_type = 'user'	
SELECT FIELDS	userID, email
TABLE(S)	Users
CONDITION(S)	user_type = 'user'
ORDER BY	email ASC

SELECT all user emails and IDs of user_type = 'admin'	
SELECT FIELDS	userID, email
TABLE(S)	users
CONDITION(S)	user_type = 'admin'
ORDER BY	email ASC

usersHandler.php

Promote a user to an admis status	
UPDATE TABLE	users
SET	user_type = ‘admin’
CONDITION(S)	userID = ‘<given userID>’

Select file names of book photos for a user for deleting when deleting their account	
SELECT FIELDS	photo
TABLE(S)	books
CONDITION(S)	userID = ‘<given userID>’

Delete details about a user’s books when they get deleted by an admin	
DELETE FROM TABLE	books
CONDITION(S)	userID = ‘<given userID>’

Delete user details of a user that is getting deleted by an admin	
DELETE FROM TABLE	users
CONDITION(S)	userID = ‘<given userID>’ user_type = ‘user’

Implementation

Code: (screenshots of pages as viewed in a browser are shown in testing)

connectDB.php

```
1. <?php
2.
3. $dbc =
   mysqli_connect('localhost','book_admin','C7B?mJrv$P=q?x6n','bookstore_db')
   OR die(mysqli_connect_error());
```

login.php

```
1. <!DOCTYPE html>
2. <html>
3.     <head>
4.         <meta name="viewport" content="width=device-width" />
5.         <title>Login</title>
6.
7.         <link rel='stylesheet' type='text/css'
   href='../css/pageStyles.css'>
8.         <link rel='stylesheet' type='text/css'
   href='../css/mediaQueries.css'>
9.
10.        <?php #terminate session
11.        session_start();
12.        $_SESSION = array(); #clear $_SESSION superglobal
13.        if (isset($_SESSION['userID']))
14.        {
15.            session_destroy(); #end session
16.        }
17.        ?>
18.    </head>
19.    <body>
20.
21.    <?php
22.        $pageName = 'Login';
23.        $buttonTitle = 'Register';
24.        $buttonDestination = 'register.php';
25.        include('entrancePageHeader.php');
26.    ?>
27.
28.    <div id='outerPageContainer'>
29.    <div id='pageContainer'>
30.    <!-- form to login with email and password -->
31.    <div class='loginForm'>
32.        <div class='formContainer'>
33.        <h2>Login with Details:</h2>
34.
35.        <form action='loginHandler.php' method='POST'>
36.            <p>Email Address:</p>
37.            <input type="email" name="email" value='<?php if
   (isset($email)){echo $email;}?>' required maxlength='60'
   placeholder='someone@esms.org.uk'>
38.
39.            <br>
40.            <br>
41.
42.            <p>Password:</p>
43.            <input type="password" name="pass" value='<?php if
   (isset($pass)){echo $pass;}?>' required maxlength='256'>
44.            <br>
45.            <br>
46.
47.            <input class='submitButton' type="submit" value="Login">
48.        </form>
49.        <div class='formLink'>
50.        <p><a href='forgotPass.php'>Forgot Password</a></p>
51.        </div>
```

```
52.         </div>
53.     </div>
54.
55.     <!--Only display div if input errors exist-->
56.     <div class='errorsBox' <?php if (!isset($errors) or
    empty($errors)){echo "style='display: none';";}?>>
57.     <div class='errorsContainer' <?php if (!isset($errors) or
    empty($errors)){echo "style='display: none';";}?>>
58.     <?php
59.             if (isset($errors) and !(empty($errors)))
60.             {
61.                 foreach ($errors as $error)
62.                 {
63.                     echo "<p>$error</p>";
64.                     echo '<br>';
65.                 }
66.             }
67.         ?>
68.     </div>
69.     </div>
70.
71.     <!-- form to login with verification code -->
72.     <div class='loginForm' id='codeForm'>
73.         <div class='formContainer'>
74.
75.         <h2>Login with Code:</h2>
76.         <form action="codeHandler.php" method='POST'>
77.             <p>Code:</p>
78.             <input type="text" name="code" value='<?php if
    (isset($code)){echo $code;}?>' required>
79.             <br>
80.             <br>
81.
82.             <input type="submit" class='submitButton' value="Login">
83.         </form>
84.         </div>
85.     </div>
86.     </div>
87.     </div>
88.     </body>
89. </html>
```

loginHandler.php

```
1. <?php
2.
3. if ($_SERVER['REQUEST_METHOD'] != 'POST') #ensure data has been posted
4. {
5.     require('loadPage.php');
6.     load('login.php');
7. }
8.
9. require('connectDB.php');
10.
11.#protect against SQL injection/special characters
12.
13.$email = mysqli_real_escape_string($dbc,trim($_POST['email']));
14.$pass = mysqli_real_escape_string($dbc,$_POST['pass']);
15.
16.$errors = array(); #initialise errors array
17.
18.if (empty($email)) #ensure that they entered an email and password
19.{
20.    $errors[] = 'Please enter an email address';
21.}
22.
23.if (empty($pass))
24.{
25.    $errors[] = 'Please enter a password';
26.}
27.
28.if (empty($errors))
29.{
30.    #see if any users match submtted email and password combination
31.    $sql = "SELECT * FROM users WHERE email = '$email' AND password =
   SHA2('$pass',256)";
32.    $r = mysqli_query($dbc,$sql);
33.    if (mysqli_num_rows($r) != 1)#make sure the combination is valid
34.    {
35.        $errors[] = 'Email and password combination is invalid';
36.    }
37.    else
38.    {
39.        #make sure user is not of 'pending' user_type (they have confirmed
   they have access to an email of form '...@esms.org.uk')
40.        $row = mysqli_fetch_array($r, MYSQLI_ASSOC);
41.
42.        if ($row['user_type'] == 'pending')
43.        {
44.            $errors[] = 'Account has not been verified yet. Please login
   with the code emailed to you for the first time, or re-register if the code
   has expired/been lost.';
45.        }
46.    }
47.}
48.
49.if (empty($errors))
50.{
51.    #initialise session variables
52.    session_start();
```

```
53.     $_SESSION['userID'] = $row['userID'];
54.     $_SESSION['email'] = $email;
55.     $_SESSION['user_type'] = $row['user_type'];
56.
57.     require('loadPage.php');
58.     load('home.php');
59. }
60. else
61. {
62.     require('login.php');
63. }
```

codeHandler.php

```
1. <?php
2. 
3. #ensure user has reached here appropriately
4. if ($_SERVER['REQUEST_METHOD'] != 'POST')
5. {
6.     require('loadPage.php');
7.     load('login.php');
8. }
9. 
10. require('connectDB.php');
11. 
12. $errors = array();
13. 
14. $code = mysqli_real_escape_string($dbc,trim($_POST['code']));
15. if (empty($code))
16. {
17.     $errors[] = 'Please Enter a Code';
18. }
19. 
20. else
21. {
22.     #find when a certain code was issued
23.     $sql = "SELECT timeOfCode FROM users WHERE code = SHA2('$code',256)";
24.     $r = mysqli_query($dbc,$sql);
25. 
26.     if (mysqli_num_rows($r) == 0)#if code is invalid
27.     {
28.         $errors[] = 'Code is invalid';
29.     }
30.     else
31.     {
32.         #need to check if the the code has been entered within 5 minutes of
    being issued
33.         #unpack query result - it will be an array of length 1 as each code
    is unique
34.         $row = mysqli_fetch_array($r,MYSQLI_NUM);
35.         #this is the DateTime when the code was issued, stored as a string
36.         $timeOfCode = $row[0];
37. 
38.         #turns string into a DateTime object
39.         $timeOfCode = date_create($timeOfCode);
40. 
41.         #defaults to a DateTime object of the current time
42.         $currentTime = date_create();
43. 
44.         #creates a DateInterval object whose properties make up the numeric
    time interval
45.         $diff = date_diff($timeOfCode, $currentTime);
46. 
47.         $timeElapsed = 0;
48. 
49.         #each unit of time will be converted into seconds for comparison
50.         #seconds
51.         $timeElapsed += $diff->s;
52.         #minutes
53.         $timeElapsed += ($diff->i) * 60;
```

```
54.         #hours
55.         $timeElapsed += ($diff->h) * 60 * 60;
56.         #days
57.         $timeElapsed += ($diff->d) * 24 * 60 * 60;
58.         #months
59.         $timeElapsed += ($diff->m) * 30 * 24 * 60 * 60;
60.         #years
61.         $timeElapsed += ($diff->y) * 365 * 24 * 60 * 60;
62.
63.         #to stop an automated script brute forcing the code, only 5 minutes
    (5 * 60 seconds) is allowed to enter the code
64.         if ($timeElapsed > (5 * 60))
65.         {
66.             $errors[] = "Code expired. Click <a
    href='register.php'>here</a> to re-register or if you have forgotten your
    password, click where appropriate to get issued with a new code.";
67.         }
68.     }
69. }
70.
71. if (empty($errors))
72. {
73.     #change user_type from 'pending' to 'user' as code is valid if
    necessary (only needed when code is used for first login, not for use of
    forgotten password)
74.     $sql = "UPDATE users SET user_type = 'user' WHERE code =
    SHA2('$code',256) AND user_type = 'pending'";
75.     mysqli_query($dbc,$sql);
76.
77.     #initialise the session variable
78.     $sql = "SELECT * FROM users WHERE code = SHA2('$code',256)";
79.     $r = mysqli_query($dbc,$sql);
80.     $row = mysqli_fetch_array($r,MYSQLI_ASSOC);
81.
82.     session_start();
83.     $_SESSION['userID'] = $row['userID'];
84.     $_SESSION['email'] = $row['email'];
85.     $_SESSION['user_type'] = $row['user_type'];
86.
87.     require('loadPage.php');
88.     load('home.php');
89. }
90. else
91. {
92.     require('login.php');
93. }
94.
95.
```

register.php

```
1.  <!DOCTYPE html>
2. <html>
3.     <head>
4.     <meta name="viewport" content="width=device-width" />
5.         <title>Register</title>
6.         <link rel='stylesheet' type='text/css'
   href='../css/pageStyles.css'>
7.         <link rel='stylesheet' type='text/css'
   href='../css/mediaQueries.css'>
8.
9.         <?php if (isset($_SESSION['email'])){session_destroy();}?><!--
   Terminate session -->
10.    </head>
11.    <body>
12.
13.    <?php
14.        $pageName = 'Register';
15.        $buttonTitle = 'Login';
16.        $buttonDestination = 'login.php';
17.        include('entrancePageHeader.php');
18.    ?>
19.
20.        <div id='outerPageContainer'>
21.        <div id='pageContainer'>
22.
23.        <!-- registration form -->
24.        <div class='loginForm'>
25.        <div class='formContainer'>
26.        <h2>Register</h2>
27.
28.        <form action='registerHandler.php' method='POST'>
29.            <p>Email Address:</p>
30.            <input type="email" name="email" value='<?php
   if(isset($email)){echo $email;}?>' required pattern='^[a-zA-Z0-9.-
   _]+@esms.org.uk' maxlength='60' title='someone@esms.org.uk'
   placeholder='someone@esms.org.uk'>
31.            <br>
32.            <br>
33.
34.            <p>Password:</p>
35.            <input type="password" name="pass1" value='<?php
   if(isset($pass1)){echo $pass1;}?>' required maxlength='256' minlength='6'>
36.            <br>
37.            <br>
38.
39.            <p>Confirm Password:</p>
40.            <input type="password" name="pass2" value='<?php
   if(isset($pass2)){echo $pass2;}?>' required maxlength='256' minlength='6'>
41.            <br>
42.            <br>
43.
44.            <input class='submitButton' type="submit" value="Register">
45.        </form>
46.        </div>
47.        </div>
48.
```

```
49.             <!--Only display div if input errors exist-->
50.         <div class='errorsBox' <?php if (!isset($errors) or
    empty($errors)){echo "style='display: none';";}?>>
51.         <div class='errorsContainer' <?php if (!isset($errors) or
    empty($errors)){echo "style='display: none';";}?>>
52.         <?php
53.                 if (isset($errors) and !(empty($errors)))
54.                 {
55.                     foreach ($errors as $error)
56.                     {
57.                         echo "<p>$error</p>";
58.                         echo '<br>';
59.                     }
60.                 }
61.             ?>
62.         </div>
63.         </div>
64.         </div>
65.         </div>
66.     </body>
67. </html>
```

registerHandler.php

```
1. <?php
2.
3. if (!($_SERVER['REQUEST_METHOD'] == 'POST'))
4. {
5.     require('loadPage.php');
6.     load('register.php');
7. }
8. require('connectDB.php');
9.
10.$errors  = array(); #initialise errors array
11.
12.#retrieve data from POST superglobal
13.#remove extra spaces in email address
14.$email = mysqli_real_escape_string($dbc,trim($_POST['email']));
15.$pass1 = mysqli_real_escape_string($dbc,$_POST['pass1']);
16.$pass2 = mysqli_real_escape_string($dbc,$_POST['pass2']);
17.
18.#validate email
19.if (empty($email))
20.{
21.    $errors[] = "Please enter an email address";
22.}
23.else
24.{
25.    #regex to ensure that email address starts with some character(s) and
   is then followed by '@esms.org.uk'
26.    if (!(preg_match("/^[a-zA-Z0-9.-_]+@esms.org.uk/",$email)))
27.    {
28.        $errors[] = "Email is not valid - must be of form
   '...@esms.org.uk'";
29.    }
30.    else
31.    {
32.        if (strlen($email) > 60)
33.        {
34.            $errors[] = 'Email address is too long';
35.        }
36.    }
37.}
38.
39.#validate password
40.if (empty($pass1) and empty($pass2))
41.{
42.    $errors[] = "Please enter a password";
43.}
44.else
45.{
46.    #ensure two passwords match
47.    if ($pass1 != $pass2)
48.    {
49.        $errors[] = "Passwords don't match";
50.    }
51.    else
52.    {
53.        if (strlen($pass1) <= 5)
54.        {
```

```
55.             $errors[] = "Password is too short";
56.         }
57.         else
58.         {
59.             if (strlen($pass1) > 256)
60.             {
61.                 $errors[] = 'Password is too long';
62.             }
63.         }
64.     }
65. }
66. 
67. #check that email isn't already in use
68. if (empty($errors))
69. {
70.     $sql = "SELECT * FROM users WHERE email = '$email'";
71.     $r = mysqli_query($dbc,$sql);
72.     if (mysqli_num_rows($r) != 0)
73.     {
74.         #if an email address is already in use, but it is 'pending' (no one
    has logged into it with a code for the first time), then the old entry
    should be deleted so that it can be re-registered - this is to provide a
    new verification code
75.         $sql = "SELECT * FROM users WHERE email = '$email' AND user_type =
    'pending'";
76.         $r = mysqli_query($dbc,$sql);
77.         if (mysqli_num_rows($r) != 0)
78.         {
79.             $sql = "DELETE FROM users WHERE email = '$email'";
80.             mysqli_query($dbc,$sql);
81.         }
82. 
83.         else
84.         {
85.             $errors[] = 'Email already in use';
86.         }
87.     }
88. }
89. 
90. #now need to put user data into database and email them a code
91. if (empty($errors))
92. {
93.     require('email.php');
94. 
95.     #generate unique 10 digit code
96.     $code = genCode();
97. 
98.     #to ensure users have access to an email of the form '...@esms.org.uk'
    their status will be set to 'pending' until they log in with the code for
    the first time
99.     #password and code are both hashed with inbuilt function
100.      $sql = "INSERT INTO users (email,password,user_type,code,timeOfCode)
    VALUES ('$email', SHA2('$pass1',256),'pending', SHA2('$code',256), NOW())";
101.      mysqli_query($dbc,$sql);
102. 
103.      #send the email with the code to the user
104.      email($email,'Registration',"<h2>Thank you for registering an account
    with <b>ESMS Book Bank</b>.</h2><p>If you don't remember registering with
    us, please ignore this email.</p><p>Otherwise, please use the code:
```

```
    <b>$code</b> to login for the first time.</p><p><i>ESMS Book
    Bank</i></p>","Thank you for registering an account with ESMS Book Bank.If
    you don't remember registering with us, please ignore this email.Otherwise,
    please use the code: $code to login for the first time.ESMS Book Bank");
105.      $errors[] = "Thank you for registering. Please use the code which has
    been emailed to you to register <a href='login.php'>here</a> for the first
    time.";
106.  }
107.
108.  #load register page to display error/success message
109.  require('register.php');
```

forgotPass.php

```
1. <!DOCTYPE html>
2. <html>
3.     <head>
4.     <meta name="viewport" content="width=device-width" />
5.         <title>Forgot Password</title>
6.         <link rel='stylesheet' type='text/css'
   href='../css/pageStyles.css'>
7.         <link rel='stylesheet' type='text/css'
   href='../css/mediaQueries.css'>
8.
9.     </head>
10.    <body>
11.
12.    <?php
13.        $pageName = 'Forgot Password';
14.        $buttonTitle = 'Login';
15.        $buttonDestination = 'login.php';
16.        include('entrancePageHeader.php');
17.    ?>
18.
19.    <div id='outerPageContainer'>
20.    <div id='pageContainer'>
21.    <!-- Email address is then sent an email with a new code to input -->
22.    <!-- email address form -->
23.    <div class='loginForm'>
24.    <div class='formContainer'>
25.    <form action='forgotPassHandler.php' method='POST'>
26.        <p>Email Address:</p>
27.        <input type="email" name="email" value='<?php
   if(isset($email)){echo $email;}?>' required maxlength='60'>
28.        <br>
29.        <br>
30.
31.        <input type="submit" class='submitButton' value="Enter">
32.    </form>
33.    </div>
34.    </div>
35.
36.    <?php
37.        #info bubble confirming sending of email
38.        if ($_SERVER['REQUEST_METHOD'] == 'POST')
39.        {
40.            echo "
41.                <div class='infoBubbleContainer'
   id='forgotPassBubbleContainer'>
42.                <div class='infoBubbleContainerDetails'
   id='forgotPassBubbleContainerDetails'>
43.                <div class='infoBubble'>
44.                <p>Email has been sent, make sure to check your spam folder
   if nothing arrives shortly. Click <a href='login.php'>here</a> to
   login.</p>
45.                </div>
46.                </div>
47.                </div>";
48.        }
49.    ?>
```

```
50.
51.     </div>
52.     </div>
53. </body>
54. </html>
```

forgotPassHandler.php

```
1. <?php
2.
3. if ($_SERVER['REQUEST_METHOD'] != 'POST') #ensure form has been submitted
4. {
5.     require('loadPage.php');
6.     load('login.php');
7. }
8.
9. require('connectDB.php');
10. require('email.php');
11.
12. $email = mysqli_real_escape_string($dbc,trim($_POST['email']));
13.
14. $sql = "SELECT * FROM users WHERE email = '$email'";
15. $r = mysqli_query($dbc,$sql);
16.
17. #if email address is not valid, no error message will appear in order to
    protect the email addresses of the users - so others cannot spam emails
    until they find a valid one
18.
19. if (mysqli_num_rows($r) == 1)
20. {
21.     $code = genCode();
22.
23.     #update database with new hashed code and new time
24.     $sql = "UPDATE users SET code = SHA2('$code',256), timeOfCode = NOW()
    WHERE email = '$email'";
25.     mysqli_query($dbc,$sql);
26.
27.     email($email,'Code for Access',"<h2>Dear User,</h2><p>We have been
    alerted that you have forgotten your password.</p><p>If this is true, use
    the following code to login to then be able to change your password:
    <b>$code</b></p><p><b>If you did not request this code then please ignore
    this email!</b></p><p><i>ESMS Book Bank</i></p>","Dear User, We have been
    alerted that you have forgotten your password. If this is true, use the
    following code to login to then be able to change your password: $code If
    you did not request this code then please ignore this email! ESMS Book
    Bank");
28. }
29.
30. require('forgotPass.php');
```

entrancePageHeader.php

```
1. <?php
2. echo "
3. <div id='entrancePageHeader'>
4.         <div>
5.             <h2>$pageName</h2>
6.         </div>
7.         <div id='titleDiv'>
8.             <h1>ESMS Book Bank</h1>
9.         </div>
10.        <div>
11.            <a href='$buttonDestination'><button id='movePageButton'
   type='button'>$buttonTitle</button></a>
12.        </div>
13.    </div>
14.    ";
15.
```

home.php

```
1. <!DOCTYPE html>
2. <html>
3.     <head>
4.     <meta name="viewport" content="width=device-width" />
5.         <title>Home Page</title>
6.
7.         <link rel='stylesheet' type='text/css'
   href='../css/pageStyles.css'>
8.         <link rel='stylesheet' type='text/css'
   href='../css/mediaQueries.css'>
9.
10.        <?php
11.        #ensure users login to reach here
12.        session_start();
13.
14.        if (!(isset($_SESSION['userID'])))
15.        {
16.            require('loadPage.php');
17.            load('login.php');
18.        }
19.        ?>
20.    </head>
21.    <body>
22.
23.        <div id='pageHeader'>
24.            <div id='homeTitle'>
25.                <h2>Home Page</h2>
26.            </div>
27.        </div>
28.
29.        <div id='outerPageContainer'>
30.        <div id='pageContainer'>
31.
32.        <?php
33.            #generate a message personal to the user. Take their username
   to be the bit of their email that precedes the '@'
34.            $email = $_SESSION['email'];
35.
36.            #split on '@' symbol and then take first item in array (based
   on assumption that all emails are of the form '...@esms.org.uk' so only
   have one '@' symbol)
37.            $username = preg_split("/@/",$email)[0];
38.            echo "<div class='infoBubbleContainer'
   id='homeInfoBubbleContainer'><div class='infoBubble'><p>Welcome $username.
   Here you can change your details, browse books that have been put up for
   sale or manage your own books for sale.</p></div></div>";
39.        ?>
40.        <table id='buttonTable' cellspacing=0 cellpadding=0>
41.            <tr>
42.            <td>
43.                <div>
44.                <a href='browse.php'><button
   type='button'><h2>Browse</h2></button></a>
45.                </div>
46.            </td>
47.            <td>
```

```
48.                 <div>
49.                 <a href='myBooks.php'><button type='button'><h2>My
   Books</h2></button></a>
50.                 </div>
51.             </td>
52.             </tr>
53.             <tr>
54.             <td>
55.                 <div>
56.                 <a href='accountDetails.php'><button
   type='button'><h2>Account Details</h2></button></a>
57.                 </div>
58.             </td>
59.             <td>
60.                 <div>
61.                 <a href='logout.php'><button
   type='button'><h2>Logout</h2></button></a>
62.                 </div>
63.             </td>
64.
65.         <?php
66.             #ensure only admins and the super_admin can access the 'users'
   page
67.             if ($_SESSION['user_type'] == 'admin' OR $_SESSION['user_type']
   == 'super_admin')
68.             {
69.                 echo "<tr><td colspan='2'><div id='usersButtonContainer'><a
   href='users.php'><button type='button'
   id='usersButton'><h2>Users</h2></button></a></div></td></tr>";
70.             }
71.         ?>
72.         </table>
73.     </div>
74.     </div>
75.     </body>
76. </html>
```

pageHeader.php

```
1. <!DOCTYPE html>
2. <html>
3.     <head>
4.         <meta name="viewport" content="width=device-width" />
5.     </head>
6.
7.     <body>
8.         <div id='pageHeader'>
9.             <div id='pageTitle'>
10.                <h2><?php echo $title;?></h2>
11.            </div>
12.
13.            <!-- standard navbar -->
14.            <div id='navContainer'>
15.            <nav id='mainNavBar'>
16.            <ul>
17.                <?php
18.                    if ($_SESSION['user_type'] == 'admin' or
   $_SESSION['user_type'] == 'super_admin')
19.                    {
20.                        echo "<li><a
   href='users.php'><button>Users</button></a></li>";
21.                    }
22.                ?>
23.                <li><a href='home.php'><button>Home</button></a></li>
24.                <li><a href='myBooks.php'><button>My
   Books</button></a></li>
25.                <li><a href='browse.php'><button>Browse</button></a></li>
26.                <li><a href='accountDetails.php'><button>Account
   Details</button></a></li>
27.                <li id='logoutButton'><a
   href='logout.php'><button>Logout</button></a></li>
28.            </ul>
29.            </nav>
30.            </div>
31.
32.            <!-- dropdown navbar - for mobile devices -->
33.            <div id="dropDownNav">
34.                    <ul>
35.                        <li><button class='navButton'
   onclick='toggleNavBar()'>Go To...</button></li>
36.                        <ul id="navContent" style='display:none'>
37.                            <li><a
   href='home.php'><button>Home</button></a></li>
38.                            <li><a href='myBooks.php'><button>My
   Books</button></a></li>
39.                            <?php
40.                                if ($_SESSION['user_type'] == 'admin' or
   $_SESSION['user_type'] == 'super_admin')
41.                                {
42.                                    echo "<li><a
   href='users.php'><button>Users</button></a></li>";
43.                                }
44.                            ?>
45.                            <li><a
   href='browse.php'><button>Browse</button></a></li>
```

```
                            <li><a
    href='accountDetails.php'><button>Account Details</button></a></li>
                            <li><a
    href='logout.php'><button>Logout</button></a></li>
                        </ul>
                    </ul>
                </div>
            </div>

            <!-- JS for toggling dropdown navbar -->
            <script>
                function toggleNavBar() {
                    var navBar = document.getElementById('navContent');
                    if (navBar.style.display === 'none') {
                        navBar.style.display = 'block';
                    }
                    else {
                        navBar.style.display = 'none';
                    }
                }
            </script>
        </body>
</html>
```

accountDetails.php

```
1. <!DOCTYPE html>
2. <html>
3.     <head>
4.     <meta name="viewport" content="width=device-width" />
5.         <title>Account Details</title>
6.
7.         <link rel='stylesheet' type='text/css'
   href='../css/pageStyles.css'>
8.         <link rel='stylesheet' type='text/css'
   href='../css/mediaQueries.css'>
9.         <?php #ensure users login to reach here
10.        session_start();
11.
12.        if (!(isset($_SESSION['userID'])))
13.        {
14.            require('loadPage.php');
15.            load('login.php');
16.        }
17.        ?>
18.    </head>
19.
20.    <body>
21.        <?php
22.            $title = 'Account Details';
23.            include('pageHeader.php');
24.        ?>
25.    <div id='outerPageContainer'>
26.    <div id='pageContainer'>
27.
28.    <!-- info bubble with email and user_type -->
29.    <div class='infoBubbleContainerDetails'
   id='accountDetailsInfoBubbleContainer'>
30.    <div class='infoBubble' id='accountDetailsInfoBubble'>
31.    <p>Email: <?php echo $_SESSION['email']?></p>
32.    <p>Status: <?php echo $_SESSION['user_type']?></p>
33.    </div>
34.    </div>
35.
36.    <!-- password change form -->
37.    <div class='loginForm'>
38.    <div class = 'formContainer'>
39.    <form action='passwordChange.php' method='POST'>
40.    <p>New Password:</p>
41.    <input type='password' name='pass1' value='<?php if(isset($pass1)){echo
   $pass1;}?>' required maxlength='256' minlength='6'>
42.    <br>
43.    <br>
44.
45.    <p>Confirm Password:</p>
46.    <input type='password' name='pass2' value='<?php if(isset($pass2)){echo
   $pass2;}?>' required maxlength='256' minlength='6'>
47.    <br>
48.    <br>
49.
50.    <input class='submitButton' type='submit' value='Change Password'>
51.    </form>
```

```
52.     </div>
53.     </div>
54.
55.     <!--Only display div if input errors exist-->
56.     <div class='errorsBox' <?php if (!isset($errors) or
    empty($errors)){echo "style='display: none';";}?>>
57.     <div class='errorsContainer' <?php if (!isset($errors) or
    empty($errors)){echo "style='display: none';";}?>>
58.     <?php
59.             if (isset($errors) and !(empty($errors)))
60.             {
61.                 foreach ($errors as $error)
62.                 {
63.                     echo "<p>$error</p>";
64.                     echo '<br>';
65.                 }
66.             }
67.         ?>
68.     </div>
69.     </div>
70.
71.     </div>
72.     </div>
73.     </body>
74.
75. </html>
```

passwordChange.php

```
1. <?php
2.
3. #make sure data is posted
4. if ($_SERVER['REQUEST_METHOD'] != 'POST')
5. {
6.     require('loadPage.php');
7.     load('accountDetails.php');
8. }
9.
10. require('connectDB.php');
11.
12. $errors = array(); #instantiate errors array
13.
14. $pass1 = mysqli_real_escape_string($dbc,trim($_POST['pass1']));
15. $pass2 = mysqli_real_escape_string($dbc,trim($_POST['pass2']));
16.
17. #validate password
18. if (empty($pass1))
19. {
20.     $errors[] = "Please enter a password";
21. }
22. elseif ($pass1 != $pass2)
23. {
24.     $errors[] = "Passwords don't match";
25. }
26. elseif (strlen($pass1) <= 5)
27. {
28.     $errors[] = "Password is too short";
29. }
30. elseif (strlen($pass1) > 256)
31. {
32.     $errors[] = "Password is too long";
33. }
34.
35. if (empty($errors))
36. {
37.     #update database with new password
38.     session_start();
39.     $userID = $_SESSION['userID'];
40.     $sql = "UPDATE users SET password = SHA2('$pass1',256) WHERE userID =
    '$userID'";
41.     mysqli_query($dbc,$sql);
42.     $errors[] = "Password Successfully Changed";
43. }
44.
45. #since accountDetails.php calls session_start() to ensure a user is logged
    in, must close a session (while not destroying its contents) to avoid
    trying to start a session when one is already open (generates an
    error/warning)
46. session_write_close();
47. require('accountDetails.php');
```

myBooks.php

```
1. <!DOCTYPE html>
2. <html>
3.     <head>
4.     <meta name="viewport" content="width=device-width" />
5.         <title>My Books</title>
6.
7.         <link rel='stylesheet' type='text/css'
   href='../css/pageStyles.css'>
8.         <link rel='stylesheet' type='text/css'
   href='../css/mediaQueries.css'>
9.
10.        <?php #ensure user is logged in
11.
12.        if (!(isset($_SESSION['userID'])))
13.        {
14.            session_start();
15.        }
16.
17.        if (!(isset($_SESSION['userID'])))
18.        {
19.            require('loadPage.php');
20.            load('login.php');
21.        }
22.
23.        ?>
24.    </head>
25.    <body>
26.
27.        <?php
28.            $title = 'My Books';
29.            include('pageHeader.php');
30.        ?>
31.
32.        <div id='outerPageContainer'>
33.
34.        <div id='pageContainer'>
35.        <div id='filterAndDisplayBooks'>
36.
37.        <!-- Upload Book Form -->
38.        <div id='uploadBookContainer'>
39.        <div id='uploadBookForm'>
40.        <form action='uploadBook.php' method='POST'
   enctype="multipart/form-data">
41.            <!--Only make form sticky if there is a mistake when a user
   tries to uplaod a book-->
42.            <h2>Upload Book For Sale</h2>
43.
44.            <p>Title/Edition:</p>
45.            <input type='text' name='title' value='<?php if
   (isset($bookTitle)){if (isset($errors)) {if (!(empty($errors))){echo
   $bookTitle;}}}?>' required maxlength='60'>
46.            <br><br>
47.
48.            <p>ISBN Number:</p>
49.            <input type='text' name='ISBN' value='<?php if
   (isset($ISBN)){if (isset($errors)) {if (!(empty($errors))){echo
```

```
   $ISBN;}}}?>' required maxlength='13' minlength='10' pattern='\d{13}|\d{10}'
   title='ISBN 13 or ISBN 10' placeholder='1111111111111'>
50.             <br><br>
51.
52.             <p>Author:</p>
53.             <input type='text' name='author' value='<?php if
   (isset($author)){if (isset($errors)) {if (!(empty($errors))){echo
   $author;}}}?>' required maxlength='60'>
54.             <br><br>
55.
56.             <p>Qualification Type:</p>
57.             <!--Make dropdown menu sticky if there are errors and an input
   was given-->
58.             <select name='qualification' class='dropDownMenu' required>
59.                 <option value='<?php if (!(isset($qualification)) or
   !(isset($errors)) or empty($errors)){echo'';} else {echo
   $qualification;}?>'><?php if (!(isset($qualification)) or
   empty($qualification) or !(isset($errors)) or empty($errors)){echo 'Select
   a qualification';} else {echo $qualification;}?></option>
60.                 <option value='advancedHigher'>Advanced Higher</option>
61.                 <option value='higher'>Higher</option>
62.                 <option value='national5'>National 5</option>
63.                 <option value='national4'>National 4</option>
64.                 <option value='GCSE'>GCSE</option>
65.                 <option value='alevel'>A Level</option>
66.                 <option value='other'>Other</option>
67.             </select>
68.             <br><br>
69.
70.             <p>Subject:</p>
71.             <select name='subject' class='dropDownMenu' required>
72.                 <option value='<?php if (!(isset($subject)) or
   !(isset($errors)) or empty($errors)){echo'';} else {echo
   $subject;}?>'><?php if (!(isset($subject)) or empty($subject) or
   !(isset($errors)) or empty($errors)){echo 'Select a subject';} else {echo
   $subject;}?></option>
73.                 <option value='art'>Art</option>
74.                 <option value='biology'>Biology</option>
75.                 <option value='chemistry'>Chemistry</option>
76.                 <option value='computerScience'>Computer Science</option>
77.                 <option value='economics'>Economics</option>
78.                 <option value='English'>English</option>
79.                 <option value='environmentalScience'>Environmental
   Science</option>
80.                 <option value='french'>French</option>
81.                 <option value='geography'>Geography</option>
82.                 <option value='german'>German</option>
83.                 <option value='history'>History</option>
84.                 <option value='latin'>Latin</option>
85.                 <option value='mathematics'>Mathematics</option>
86.                 <option value='mechanics'>Mechanics</option>
87.                 <option value='media'>Media</option>
88.                 <option value='modernStudies'>Modern Studies</option>
89.                 <option value='music'>Music</option>
90.                 <option value='physicalEducation'>Physical
   Education</option>
91.                 <option value='physics'>Physics</option>
92.                 <option value='productDesign'>Product Design</option>
93.                 <option value='spanish'>Spanish</option>
```

```
94.                 <option value='statistics'>Statistics</option>
95.                 <option value='other'>Other</option>
96.             </select>
97.             <br><br>
98.
99.             <p>Price (£):</p>
100.              <input type='number' step='0.01' name='price' value='<?php if
    (isset($price)){if (isset($errors)) {if (!(empty($errors))){echo
    $price;}}}?>' required min='0' max='99999999.99'>
101.              <br><br>
102.
103.              <p>Condition:</p>
104.              <input type='text' name='condition' value='<?php if
    (isset($condition)){if (isset($errors)) {if (!(empty($errors))){echo
    $condition;}}}?>' required maxlength='60'>
105.              <br><br>
106.
107.              <p>Photo of Book:</p>
108.              <!--$fname is defined in uploadBook.php as
    basename($_FILES['photo']['name']) which essentailly extracts the filename
    of the inputed file-->
109.              <?php if (isset($fname) and !(empty($fname)) and
    isset($errors) and !(empty($errors))){echo "<br><h4 id='reUploadFile'>Image
    of book will need to be re-uploaded as an error occured</h4>";}?>
110.              <input id='chooseFile' type='file' name='photo'>
111.              <br><br>
112.
113.              <input class='submitButton' type='submit' value='Add Book'>
114.          </form>
115.          </div>
116.          </div>
117.
118.          <!--Only display if on a mobile (screen width <= 420px)-->
119.          <div class='errorsBox' id='myBooksErrorsBoxMobile' <?php if
    (!isset($errors) or empty($errors)){echo "style='display: none';";}?>>
120.          <div class='errorsContainer' <?php if (!isset($errors) or
    empty($errors)){echo "style='display: none';";}?>>
121.              <?php
122.                  if (isset($errors) and !(empty($errors)))
123.                  {
124.                      foreach ($errors as $error)
125.                      {
126.                          echo "<p>$error</p>";
127.                          echo '<br>';
128.                      }
129.                  }
130.              ?>
131.          </div <?php if (!isset($errors) or empty($errors)){echo
    "style='display: none';";}?>>
132.          </div <?php if (!isset($errors) or empty($errors)){echo
    "style='display: none';";}?>>
133.
134.          <?php
135.          require('connectDB.php');
136.
137.          #check to see if logged in user has any books up for sale
138.          $userID = $_SESSION['userID'];
139.          $sql = "SELECT * FROM books WHERE userID = '$userID' ORDER BY
    title ASC";
```

```
140.         $r = mysqli_query($dbc,$sql);
141.
142.         #books table
143.         echo "<div class='booksTableContainer'
   id='myBooksTableContainer'>";
144.         echo "<div class='booksTable'>";
145.         if (mysqli_num_rows($r) == 0)
146.         {
147.             echo "<div class='tableHeader'>";
148.             echo '<h2>No books up for sale yet</h2>';
149.             echo "</div>";
150.         }
151.         else
152.         {
153.
154.             echo "<form action='deleteBooks.php' method='POST'>";
155.             echo "<div class='tableHeader'>";
156.             echo "<input type='submit' id='updateBooksButton'
   value='Update Books'>";
157.             echo "</div>";
158.
159.             echo "<div id='bookTableScroll'>";
160.             echo "<table border=1>";
161.
162.             echo "<thead>";
163.             echo "<tr><th><h3>Title/Edition:</h3></th><th><h3>ISBN
   Number:</h3></th><th><h3>Author</h3></th><th><h3>Qualification</h3></th><th
   ><h3>Subject</h3></th><th><h3>Price</h3></th><th><h3>Condition</h3></th><th
   id='photoCell'><h3>Photo</h3></th><th><h3>Delete Book?</h3></th>";
164.             echo "</thead>";
165.
166.             echo "<tbody>";
167.             while ($row = mysqli_fetch_array($r,MYSQLI_ASSOC))
168.             {
169.                 echo
   "<tr><td><p>{$row['title']}</p></td><td><p>{$row['ISBN']}</p></td><td><p>{$
   row['author']}</p></td><td><p>{$row['qualification']}</p></td><td><p>{$row[
   'subject']}</p></td><td><p>£{$row['price']}</p></td><td><p>{$row['bookCondi
   tion']}</p></td>";
170.                 if ($row['photo'] == 'n/a')
171.                 {
172.                     echo "<td
   id='photoCell'><p>{$row['photo']}</p></td>";
173.                 }
174.                 else
175.                 {
176.                     echo "<td><img src=files/{$row['photo']}
   width=130px></img></td>";
177.                 }
178.                 echo "<td><input type='checkbox' name={$row['bookID']}
   value='delete'></td>";
179.             }
180.             echo "</tbody>";
181.             echo "</table>";
182.             echo "</form>";
183.             echo "</div>";
184.         }
185.
186.         echo "</div></div>";
```

```
187.         ?>
188.
189.     </div>
190.     <!--Only display div if input errors exist-->
191.     <div class='errorsBox' id='myBooksErrorsBox' <?php if
   (!isset($errors) or empty($errors)){echo "style='display: none';";}?>>
192.     <div class='errorsContainer' <?php if (!isset($errors) or
   empty($errors)){echo "style='display: none';";}?>>
193.         <?php
194.             if (isset($errors) and !(empty($errors)))
195.             {
196.                 foreach ($errors as $error)
197.                 {
198.                     echo "<p>$error</p>";
199.                     echo '<br>';
200.                 }
201.             }
202.         ?>
203.     </div <?php if (!isset($errors) or empty($errors)){echo
   "style='display: none';";}?>>
204.     </div <?php if (!isset($errors) or empty($errors)){echo
   "style='display: none';";}?>>
205.     </div>
206.     </div>
207.     </body>
208. </html>
209.
```

uploadBook.php

```php
<?php
if ($_SERVER['REQUEST_METHOD'] != 'POST') #ensure data has been posted
{
    require('loadPage.php');
    load('myBooks.php');
}

require('connectDB.php');

$errors = array(); #initialise errors array

$bookTitle = mysqli_real_escape_string($dbc,trim($_POST['title']));
$ISBN = mysqli_real_escape_string($dbc,trim($_POST['ISBN']));
$author = mysqli_real_escape_string($dbc,trim($_POST['author']));
$qualification = mysqli_real_escape_string($dbc,$_POST['qualification']);
$subject = mysqli_real_escape_string($dbc,$_POST['subject']);
$price = mysqli_real_escape_string($dbc,$_POST['price']);
$condition = mysqli_real_escape_string($dbc,trim($_POST['condition']));

#validate inputs
if (empty($bookTitle))
{
    $errors[] = "Please provide a title";
}
elseif (strlen($bookTitle) > 60)
{
    $errors[] = "Title is too long";
}

if (empty($ISBN))
{
    $errors[] = "Please provide an ISBN number";
}
elseif (strlen($ISBN) != 13 and strlen($ISBN) != 10)
{
    $errors[] = "All ISBN numbers must be 13 or 10 digits long";
}
#ensure that ISBN number is entirely numeric
elseif (!ctype_digit($ISBN))
{
    $errors[] = "Invalid ISBN Number";
}

if (empty($author))
{
    $errors[] = "Please provide an author";
}
elseif (strlen($author) > 60)
{
    $errors[] = "Author name is too long";
}

if (empty($qualification))
{
    $errors[] = "Please provide a qualification level";
}
```

```
57.
58. if (empty($subject))
59. {
60.     $errors[] = "Please provide a subject";
61. }
62.
63. if (empty($price))
64. {
65.     $errors[] = "Please provide a price";
66. }
67. elseif ($price < 0 or $price > 99999999.99)
68. {
69.     $errors[] = "Please enter a valid price";
70. }
71.
72. if (empty($condition))
73. {
74.     $errors[] = "Please provide a description of the book's condition";
75. }
76. elseif (strlen($condition) > 60)
77. {
78.     $errors[] = "Description of book's condition is too long";
79. }
80.
81. #get file name
82. $fname = basename($_FILES['photo']['name']);
83.
84. #check if a file has been uploaded (this is not necessary for a book to be
    uploaded)
85. if ($fname != "")
86. {
87.     $upload = TRUE;
88.     $extension = strtolower(pathinfo($fname,PATHINFO_EXTENSION));
89.
90.     #check to see if file is actually an image
91.     if (getimagesize($_FILES['photo']['tmp_name']) == False)
92.     {
93.         $errors[] = "File is not an image";
94.     }
95.
96.     #check extension is an accepted one
97.     elseif ($extension != 'jpg' && $extension != 'png' && $extension !=
    'jpeg')
98.     {
99.         $errors[] = "File is not an accpted format (must be a .jpg,.png or
    .jpeg)";
100.     }
101.
102.     #ensures image size is <= 3 megabytes
103.     elseif ($_FILES['photo']['size'] > 3145728)
104.     {
105.         $errors[] = "Image is too large";
106.     }
107. }
108. else
109. {
110.     $upload = FALSE;
111. }
112.
```

```
113. if (empty($errors))
114. {
115.     session_start();
116.     $userID = $_SESSION['userID'];
117.     #a file was submitted
118.     if ($upload)
119.     {
120.         #each photo file needs a unique name - they will be the the
    userID of the uploader concatenated with the current time stamp of the time
    uploaded of the form (yymmddhhmmssuuuuuu) where u = microseconds
121.         #this degree of accuracy makes it impossible for one user to
    upload 2 books at the same time stamp, so that they all have a unique name
122.
123.         $newFname =
    $userID.date_format(date_create(),'ymdHisu').'.'.$extension; #date_create
    creates an object of the current time stamp
124.
125.         if(!(move_uploaded_file($_FILES['photo']['tmp_name'],
    'files/'.$newFname)))#returns False if it is unsuccessful
126.         {
127.             $errors[] = "Error when uploading photo";
128.         }
129.     }
130.     else
131.     {
132.         $newFname = 'n/a';
133.     }
134.
135.     $sql = "INSERT INTO books
    (userID,title,ISBN,author,qualification,subject,price,bookCondition,photo)
    VALUES
    ('$userID','$bookTitle','$ISBN','$author','$qualification','$subject','$pri
    ce','$condition','$newFname')";
136.     mysqli_query($dbc,$sql);
137. }
138.
139. require('myBooks.php');
140.
```

deleteBooks.php

```
1. <?php
2. 
3. if ($_SERVER['REQUEST_METHOD'] != 'POST') #ensure data has been posted
4. {
5.     require('loadPage.php');
6.     load('myBooks.php');
7. }
8. 
9. require('connectDB.php');
10. 
11. foreach($_POST as $id => $action)
12. {
13.     #get image name to delete
14.     $sql = "SELECT photo FROM books WHERE bookID = '$id'";
15.     $r = mysqli_query($dbc,$sql);
16.     #condition below so if page is re-loaded it doesn't try and delete a
    book that doesn't exist
17.     if (mysqli_num_rows($r) > 0)
18.     {
19.         $photo = (mysqli_fetch_array($r, MYSQLI_ASSOC))['photo'];
20. 
21.         #delete book entries from database
22.         $sql = "DELETE FROM books WHERE bookID = '$id'";
23.         mysqli_query($dbc,$sql);
24. 
25.         #if exists, delete image from directory
26.         if ($photo != 'n/a')
27.         {
28.             $fname = 'files/'.$photo;
29.             #check file exists (so ignores all n/a photos)
30.             if (is_writeable($fname))
31.             {
32.                 #delete file
33.                 unlink($fname);
34.             }
35.         }
36.     }
37. }
38. require('myBooks.php');
39. 
```

browse.php

```
1. <!DOCTYPE html>
2. <html>
3.     <head>
4.     <meta name="viewport" content="width=device-width" />
5.         <title>Browse</title>
6.
7.         <link rel='stylesheet' type='text/css'
   href='../css/pageStyles.css'>
8.         <link rel='stylesheet' type='text/css'
   href='../css/mediaQueries.css'>
9.
10.        <?php
11.        require('connectDB.php');
12.
13.        session_start();
14.        if (!(isset($_SESSION['userID'])))
15.        {
16.            require('loadPage.php');
17.            load('login.php');
18.        }
19.
20.        $errors = array(); #instantiate errors array
21.
22.        #filter to select books (don't display their own books for sale)
23.        $userID = $_SESSION['userID'];
24.        $sql = "SELECT * FROM books, users WHERE users.userID =
   books.userID AND NOT users.userID = '$userID'";
25.
26.        #validate filter inputs and append onto sql to select books to see
27.        if (isset($_GET['title']) and !(empty($_GET['title'])))
28.        {
29.            $_GET['title'] =
   mysqli_real_escape_string($dbc,trim($_GET['title']));
30.            if (strlen($_GET['title']) > 60)
31.            {
32.                $errors[] = 'Title is too long';
33.            }
34.            else
35.            {
36.                $bookTitle = trim($_GET['title']);
37.                $sql .= " AND title LIKE '%$bookTitle%'";
38.            }
39.        }
40.
41.        if (isset($_GET['ISBN']) and !(empty($_GET['ISBN'])))
42.        {
43.            $_GET['ISBN'] =
   mysqli_real_escape_string($dbc,trim($_GET['ISBN']));
44.            if (strlen($_GET['ISBN']) != 13 and strlen($_GET['ISBN']) !=
   10)
45.            {
46.                $errors[] = 'All ISBN numbers must be 13 or 10 digits
   long';
47.            }
48.            elseif (!ctype_digit($_GET['ISBN']))
49.            {
```

```
50.                 $errors[] = 'Invalid ISBN Number';
51.             }
52.             else
53.             {
54.                 $ISBN = $_GET['ISBN'];
55.                 $sql .= " AND ISBN = '$ISBN'";
56.             }
57.         }
58.
59.         if (isset($_GET['qualification']) and
    !(empty($_GET['qualification'])))
60.         {
61.             $_GET['qualification'] =
    mysqli_real_escape_string($dbc,trim($_GET['qualification']));
62.             $qualification = $_GET['qualification'];
63.             $sql .= " AND qualification = '$qualification'";
64.         }
65.
66.         if (isset($_GET['subject']) and !(empty($_GET['subject'])))
67.         {
68.             $_GET['subject'] =
    mysqli_real_escape_string($dbc,trim($_GET['subject']));
69.             $subject = $_GET['subject'];
70.             $sql .= " AND subject = '$subject'";
71.         }
72.
73.         if (isset($_GET['maxPrice']) and !(empty($_GET['maxPrice'])))
74.         {
75.             $_GET['maxPrice'] =
    mysqli_real_escape_string($dbc,trim($_GET['maxPrice']));
76.             if ($_GET['maxPrice'] < 0 or $_GET['maxPrice'] > 99999999.99)
77.             {
78.                 $errors[] = 'Please enter a valid max price';
79.             }
80.             else
81.             {
82.                 $maxPrice = $_GET['maxPrice'];
83.                 $sql .= " AND price <= '$maxPrice'";
84.             }
85.         }
86.
87.         #delete books that admins select
88.         if ($_SERVER['REQUEST_METHOD'] == 'POST')
89.         {
90.             if (!(empty($_POST)))
91.             {
92.                 foreach ($_POST as $id => $action)
93.                 {
94.                     $id = mysqli_real_escape_string($dbc, $id);
95.                     $Deletesql = "SELECT photo FROM books WHERE bookID =
    '$id'";
96.                     $r = mysqli_query($dbc,$Deletesql);
97.                     $photo =
    (mysqli_fetch_array($r,MYSQLI_ASSOC))['photo'];
98.
99.                     #delete book entries from database
100.                      $Deletesql = "DELETE FROM books WHERE bookID =
    '$id'";
101.                     mysqli_query($dbc,$Deletesql);
```

```
102.
103.                     #if exists, delete image from directory
104.                     if ($photo != 'n/a')
105.                     {
106.                         $fname = 'files/'.$photo;
107.                         #check file exists (so ignores all n/a photos)
108.                         if (is_writeable($fname))
109.                         {
110.                             #delete file
111.                             unlink($fname);
112.                         }
113.                     }
114.                 }
115.             }
116.         }
117.         ?>
118.     </head>
119.     <body>
120.         <?php
121.             $title = 'Browse';
122.             include('pageHeader.php');
123.         ?>
124.
125.         <div id='outerPageContainer'>
126.         <div id='pageContainer'>
127.         <div id='filterContainer'>
128.
129.         <div class='infoBubbleContainer' id='browseInfoBubbleContainer'>
130.         <div class='infoBubble' id='browseInfoBubble'>
131.             <p>Welcome to the browse page, feel free to use the filter to
    narrow down the selection, and then <b>click on a book</b> to be taken to
    the confirmation page.</p>
132.         </div>
133.         </div>
134.
135.         <!-- filter form -->
136.         <div class='loginForm' id='filterFormContainer'>
137.         <div class='formContainer'>
138.         <h2>Filter</h2>
139.         <form action='browse.php' method='GET'>
140.             <p>Title:</p>
141.             <input type='text' name='title' value='<?php
    if(isset($bookTitle)){echo $bookTitle;}?>' maxlength='60'>
142.             <br><br>
143.
144.             <p>ISBN:</p>
145.             <input type='text' name='ISBN' value='<?php
    if(isset($ISBN)){echo $ISBN;}?>' maxlength='13' minlength='10'
    pattern='\d{13}|\d{10}' title='ISBN 13 or ISBN 10'
    placeholder='1111111111111'>
146.             <br><br>
147.
148.             <p>Qualification:</p>
149.             <select name='qualification' class='dropDownMenu'>
150.                 <option value='<?php if(isset($qualification)){echo
    $qualification;}?>'><?php if(isset($qualification)){echo
    $qualification;}else{echo 'Any';}?></option>
```

```
151.                 <?php if(isset($qualification)){echo "<option
    value=''>Any</option>";}?> <!--allow for 'Any' option once sticky form has
    been activated-->
152.                 <option value='advancedHigher'>Advanced Higher</option>
153.                 <option value='higher'>Higher</option>
154.                 <option value='national5'>National 5</option>
155.                 <option value='national4'>National 4</option>
156.                 <option value='GCSE'>GCSE</option>
157.                 <option value='alevel'>A Level</option>
158.                 <option value='other'>Other</option>
159.             </select>
160.             <br><br>
161.
162.             <p>Subject:</p>
163.             <select name='subject' class='dropDownMenu'>
164.                 <option value='<?php if(isset($subject)){echo
    $subject;}?>'><?php if(isset($subject)){echo $subject;}else{echo
    'Any';}?></option>
165.                 <?php if(isset($qualification)){echo "<option
    value=''>Any</option>";}?> <!--allow for 'Any' option once sticky form has
    been activated-->
166.                 <option value='art'>Art</option>
167.                 <option value='biology'>Biology</option>
168.                 <option value='chemistry'>Chemistry</option>
169.                 <option value='computerScience'>Computer Science</option>
170.                 <option value='economics'>Economics</option>
171.                 <option value='english'>English</option>
172.                 <option value='environmentalScience'>Environmental
    Science</option>
173.                 <option value='french'>French</option>
174.                 <option value='geography'>Geography</option>
175.                 <option value='german'>German</option>
176.                 <option value='history'>History</option>
177.                 <option value='latin'>Latin</option>
178.                 <option value='mathematics'>Mathematics</option>
179.                 <option value='mechanics'>Mechanics</option>
180.                 <option value='media'>Media</option>
181.                 <option value='modernStudies'>Modern Studies</option>
182.                 <option value='music'>Music</option>
183.                 <option value='physicalEducation'>Physical
    Education</option>
184.                 <option value='physics'>Physics</option>
185.                 <option value='productDesign'>Product Design</option>
186.                 <option value='spanish'>Spanish</option>
187.                 <option value='statistics'>Statistics</option>
188.                 <option value='other'>Other</option>
189.             </select>
190.             <br><br>
191.
192.             <p>Max Price (£):</p>
193.             <input type='number' step='0.01' name='maxPrice' value='<?php
    if(isset($maxPrice)){echo $maxPrice;}?>' min='0' max='99999999.99'>
194.             <br><br>
195.
196.             <input id='applyFilterButton' type='submit' value='Apply
    Filter'>
197.             <a
    href='browse.php?title=&ISBN=&qualification=&subject=&maxPrice='><button
    type='button' id='clearFilterButton'>Clear Filter</button></a>
```

```
198.          </form>
199.          </div>
200.          </div>
201.
202.          <!--Only display div if input errors exist-->
203.          <div class='errorsBox' id='filterErrorsBox' <?php if
   (!isset($errors) or empty($errors)){echo "style='display: none';";}?>>
204.          <div class='errorsContainer' <?php if (!isset($errors) or
   empty($errors)){echo "style='display: none';";}?>>
205.          <?php
206.              if (isset($errors) and !(empty($errors)))
207.              {
208.                  foreach ($errors as $error)
209.                  {
210.                      echo "<p>$error</p>";
211.                      echo '<br>';
212.                  }
213.              }
214.          ?>
215.          </div>
216.          </div>
217.
218.          </div>
219.
220.
221.          <?php #display books in books table
222.          $sql .= ' ORDER BY title ASC';
223.
224.          $r = mysqli_query($dbc,$sql);
225.
226.          echo "<div id='browseBooksContainer'>";
227.          echo "<div class='booksTable' id='browseBooksTable'>";
228.          if (mysqli_num_rows($r) == 0)
229.          {
230.              echo "<div class='tableHeader'>";
231.              echo "<h2>Sorry, we couldn't find any books for sale with
   this filter...</h2>";
232.              echo "</div>";
233.          }
234.          else
235.          {
236.
237.              echo "<form action='browse.php' method='POST'>";
238.
239.              if ($_SESSION['user_type'] == 'admin' or
   $_SESSION['user_type'] == 'super_admin')
240.              {
241.                 echo "<div class='tableHeader'><input type='submit'
   value='Update Books' id='updateBrowseBooksButton'></div>";
242.              }
243.
244.              echo "<div id='bookTableScroll'>";
245.              echo '<table border=1>';
246.              echo '<thead>';
247.              echo "<tr><th><h3>Title/Edition:</h3></th><th><h3>ISBN
   Number</h3></th><th><h3>Author:</h3></th><th><h3>Seller
   Email:</h3></th><th><h3>Qualification:</h3></th><th><h3>Subject:</h3></th><
   th><h3>Price:</h3></th><th><h3>Condition:</h3></th><th
   id='photoCell'><h3>Photo:</h3></th>";
```

```
248.
249.             if ($_SESSION['user_type'] == 'admin' or
    $_SESSION['user_type'] == 'super_admin')
250.             {
251.                 echo '<th><h3>Delete?</h3></td>';
252.             }
253.
254.             echo '</tr>';
255.             echo '</thead>';
256.             echo "<tbody>";
257.             while ($row = mysqli_fetch_array($r,MYSQLI_ASSOC))
258.             {
259.                 #need to replace all spaces with '%20' so that they can
    be used in URL - onclick=window.location='...' can't deal with spaces like
    a GET can - this is to make each cell (except delete cell) clickable
260.                 $location = "buyBook.php?seller=".str_replace('
    ','%20',$row['email'])."&title=".str_replace('
    ','%20',$row['title'])."&ISBN={$row['ISBN']}&author=".str_replace('
    ','%20',$row['author'])."&qualification=".str_replace('
    ','%20',$row['qualification'])."&subject=".str_replace('
    ','%20',$row['subject'])."&price={$row['price']}&condition=".str_replace('
    ','%20',$row['bookCondition'])."&photo={$row['photo']}";
261.                 #each cell is clickable except cell with checkboxes so
    that they can be used without a new page appearing
262.                 echo "<tr id='bookRow'><td
    onclick=window.location='$location'><p>{$row['title']}</p></td><td
    onclick=window.location='$location'><p>{$row['ISBN']}</p></td><td
    onclick=window.location='$location'><p>{$row['author']}</p></td><td
    onclick=window.location='$location'><p>{$row['email']}</p></td><td
    onclick=window.location='$location'><p>{$row['qualification']}</p></td><td
    onclick=window.location='$location'><p>{$row['subject']}</p></td><td
    onclick=window.location='$location'><p>£{$row['price']}</p></td><td
    onclick=window.location='$location'><p>{$row['bookCondition']}</p></td>";
263.                 if ($row['photo'] == 'n/a')
264.                 {
265.                     echo "<td onclick=window.location='$location'
    id='photoCell'><p>{$row['photo']}</p></td>";
266.                 }
267.                 else
268.                 {
269.                     echo "<td onclick=window.location='$location'
    id='photoCell'><img src=files/{$row['photo']} width=130px></img></td>";
270.                 }
271.
272.                 #admins get the option delete another person's books
273.                 if ($_SESSION['user_type'] == 'admin' or
    $_SESSION['user_type'] == 'super_admin')
274.                 {
275.                     echo "<td><input type='checkbox'
    name={$row['bookID']} value='delete'></td>";
276.                 }
277.                 echo '</tr>';
278.             }
279.             echo "</tbody>";
280.             echo "</div>";
281.             echo '</form></table>';
282.         }
283.
284.         echo "</div>";
```

```
285.          echo "</div>";
286.          ?>
287.      </div>
288.      </div>
289.      </body>
290. </html>
```

buyBook.php

```
1. <!DOCTYPE html>
2. <html>
3. <head>
4. <meta name="viewport" content="width=device-width" />
5.     <title>Buy Book</title>
6.
7.     <link rel='stylesheet' type='text/css' href='../css/pageStyles.css'>
8.     <link rel='stylesheet' type='text/css' href='../css/mediaQueries.css'>
9. <?php
10.
11. session_start();
12. if (!(isset($_SESSION['userID'])))
13. {
14.     require('loadPage.php');
15.     load('login.php');
16. }
17. ?>
18. </head>
19. <body>
20. <?php
21. #send email when confirmation button is pressed
22. if ($_SERVER['REQUEST_METHOD'] == 'POST')
23. {
24.     require('email.php');
25.
26.     # &#163; is the code to display the '£' in HTML
27.     #email buyer
28.     email($_SESSION['email'],'Seller Contact Details',"<h2>Dear
    {$_SESSION['email']},</h2><p>Here are the details about the book you showed
    interest in. The seller has received a similar emails with your details -
    <b>it is up to you two to now contact each other</b> and sort out any
    transaction/handover of the book.</p><p><i>Seller:
    {$_POST['seller']}<br>Title: {$_POST['title']}<br>ISBN:
    {$_POST['ISBN']}<br>Author: {$_POST['author']}<br>Qualification:
    {$_POST['qualification']}<br>Price: &#163;{$_POST['price']}<br>Condition:
    {$_POST['condition']}</i></p><p><i>ESMS Book Bank</i></p>","Dear
    {$_SESSION['email']}, Here are the details about the book you showed
    interest in. The seller has received a similar emails with your details -
    it is up to you two to now contact each other and sort out any
    transaction/handover of the book. Seller: {$_POST['seller']} Title:
    {$_POST['title']} ISBN: {$_POST['ISBN']} Author: {$_POST['author']}
    Qualification: {$_POST['qualification']} Price: £{$_POST['price']}
    Condition: {$_POST['condition']} ESMS Book Bank");
29.
30.     #email seller
31.     email($_POST['seller'],"Potential Buyer", "<h2>Dear
    {$_POST['seller']},</h2><p>Here are the details about a potential buyer and
    the book they wish to buy from you - <b>it is up to you two to now contact
    each other</b> and sort out any transaction/handover of the
    book:</p><p><i>Buyer: {$_SESSION['email']}<br>Title:
    {$_POST['title']}<br>ISBN: {$_POST['ISBN']}<br>Author:
    {$_POST['author']}<br>Qualification: {$_POST['qualification']}<br>Price:
    $#163;{$_POST['price']}<br>Condition:
    {$_POST['condition']}</i></p><p><i>ESMS Book Bank</i></p>","Dear
    {$_POST['seller']}, Here are the details about a potential buyer and the
    book they wish to buy from you - it is up to you two to now contact each
```

```
    other and sort out any transaction/handover of the book: Buyer:
    {$_SESSION['email']} Title: {$_POST['title']} ISBN: {$_POST['ISBN']}
    Author: {$_POST['author']} Qualification: {$_POST['qualification']} Price:
    £{$_POST['price']} Condition: {$_POST['condition']} ESMS Book Bank");
32.
33.     #return to browse.php page
34.     require('loadPage.php');
35.     load('browse.php');
36. }
37. ?>
38.
39. <?php
40.     $title = 'Confirm';
41.     include('pageHeader.php');
42. ?>
43. <div id='outerPageContainer'>
44. <div id='pageContainer'>
45.
46. <div class='infoBubbleContainer' id='browseInfoBubbleContainer'>
47. <div class='infoBubble' id='browseInfoBubble'>
48.     <p>Clicking <b>confirm</b> will send both yourself and the seller of
    this book an email with the necessary details to communicate together to
    complete the transaction of the book.</p>
49. </div>
50. </div>
51.
52. <!-- book details container -->
53. <div id='bookDetailsContainer'>
54. <div id='bookDetails'>
55. <h2>Book Details:</h2>
56. <?php
57. echo "<p>Seller: {$_GET['seller']}</p><br>";
58. echo "<p>Title: {$_GET['title']}</p><br>";
59. echo "<p>ISBN: {$_GET['ISBN']}</p><br>";
60. echo "<p>Author: {$_GET['author']}</p><br>";
61. echo "<p>Qualification: {$_GET['qualification']}</p><br>";
62. echo "<p>Subject: {$_GET['subject']}</p><br>";
63. echo "<p>Price: £{$_GET['price']}</p><br>";
64. echo "<p>Condition: {$_GET['condition']}</p><br>";
65. if ($_GET['photo'] == 'n/a')
66. {
67.     echo "<p>Photo : n/a</p>";
68. }
69. else
70. {
71.     echo "<p>Photo: </p><div id='bookPhoto'><img src=files/{$_GET['photo']}
    width=200px></img></div>";
72. }
73.
74. echo "<form action=buyBook.php method='POST'>";
75. echo "<input type='hidden' name='seller' value='{$_GET['seller']}'>";
76. echo "<input type='hidden' name='title' value='{$_GET['title']}'>";
77. echo "<input type='hidden' name='ISBN' value='{$_GET['ISBN']}'>";
78. echo "<input type='hidden' name='author' value='{$_GET['author']}'>";
79. echo "<input type='hidden' name='qualification'
    value='{$_GET['qualification']}'>";
80. echo "<input type='hidden' name='subject' value='{$_GET['subject']}'>";
81. echo "<input type='hidden' name='price' value='{$_GET['price']}'>";
82. echo "<input type='hidden' name='condition' value='{$_GET['condition']}'>";
```

```
83. echo "<input type='hidden' name='photo' value='{$_GET['photo']}'>";
84. echo "<input type='submit' class='submitButton' id='confirmBook'
    value='Confirm'>";
85. echo "</form>";
86.
87. ?>
88.
89. </div>
90. </div>
91. </div>
92. </div>
93. </body>
94. </html>
```

users.php

```
1. <!DOCTYPE html>
2. <html>
3.     <head>
4.     <meta name="viewport" content="width=device-width" />
5.         <title>Users</title>
6.
7.         <link rel='stylesheet' type='text/css'
   href='../css/pageStyles.css'>
8.         <link rel='stylesheet' type='text/css'
   href='../css/mediaQueries.css'>
9.
10.        <?php
11.        session_start();
12.
13.        #make sure they login
14.        if (!(isset($_SESSION['userID'])))
15.        {
16.            require('loadPage.php');
17.            load('login.php');
18.        }
19.
20.        #ensure only admins can access this page
21.        elseif ($_SESSION['user_type'] != 'admin' and
   $_SESSION['user_type'] != 'super_admin')
22.        {
23.            require('loadPage.php');
24.            load('home.php');
25.        }
26.        ?>
27.    </head>
28.    <body>
29.    <?php
30.        $title = 'Users';
31.        include('pageHeader.php');
32.    ?>
33.        <div id='outerPageContainer'>
34.        <div id='pageContainer'>
35.        <?php
36.        require("connectDB.php");
37.
38.        #users table
39.        $sql = "SELECT userID, email FROM users WHERE user_type = 'user'
   ORDER BY email ASC";
40.        $r = mysqli_query($dbc,$sql);
41.
42.        echo "<div class='usersFormContainer'><div class='usersForm'>";
43.        if (mysqli_num_rows($r) == 0)
44.        {
45.            echo "<div class='tableHeader'>";
46.            echo '<h2>No users</h2>';
47.            echo "</div>";
48.        }
49.        else
50.        {
51.            echo "<div class='tableHeader'>";
52.            echo "<h2>Users</h2>";
```

```
53.             echo "<form action='usersHandler.php' method='POST'>";
54.             echo "<input class='updateTable' type='submit' value='Update
    Users'>";
55.             echo "</div>";
56.
57.             echo "<div class='scrollTable'>";
58.             echo "<table border=1>";
59.             echo "<thead>";
60.             echo
    "<tr><th><h3>Email:</h3></th><th><h3>Delete:</h3></th><th><h3>Make
    Admin:</h3></th></tr>";
61.             echo "</thead>";
62.
63.             echo "<tbody>";
64.
65.             while ($row = mysqli_fetch_array($r,MYSQLI_ASSOC))
66.             {
67.                 #each row is an array of length 2 that contains each userID
    and email
68.                 #using 'userID' as name instead of 'email' as POST turns
    all dots into underscores, destroying emails
69.                 echo "<tr><td><p>{$row['email']}</p></td><td><p><input
    type='radio' name={$row['userID']}
    value='delete'></input></p></td><td><p><input type='radio'
    name={$row['userID']} value='admin'></input></p></td></tr>";
70.             }
71.
72.             echo "</tbody>";
73.
74.             echo "</table>";
75.             echo"</div>";
76.
77.             echo "</form>";
78.         }
79.         echo "</div></div>";
80.
81.         #admins table - only to be displayed to the super_admin
82.         if ($_SESSION['user_type'] == 'super_admin')
83.         {
84.             $sql = "SELECT userID,email FROM users WHERE user_type =
    'admin' ORDER BY email ASC";
85.             $r = mysqli_query($dbc,$sql);
86.
87.             echo "<div class='usersFormContainer'><div class='usersForm'>";
88.             if (mysqli_num_rows($r) == 0)
89.             {
90.                 echo "<div class='tableHeader'>";
91.                 echo '<h2>No Admins</h2>';
92.                 echo "</div>";
93.             }
94.             else
95.             {
96.                 echo "<div class='tableHeader'>";
97.                 echo "<h2>Admins</h2>";
98.                 echo "<form action='adminsHandler.php' method='POST'>";
99.                 echo "<input class='updateTable' type='submit'
    value='Update Admins'>";
100.                  echo "</div>";
101.
```

```
102.                 echo "<div class='scrollTable'>";
103.                 echo "<table border=1>";
104.                 echo "<thead>";
105.                 echo "<tr><th><h3>Email:</h3></th><th><h3>Demote
    Admin:</h3></th>";
106.                 echo "</thead>";
107.
108.                 echo "<tbody>";
109.
110.                 while ($row = mysqli_fetch_array($r,MYSQLI_ASSOC))
111.                 {
112.                     #each row is an array of length 2 that contains each
    userID and email
113.                     echo
    "<tr><td><p>{$row['email']}</p></td><td><p><input type='radio'
    name={$row['userID']} value='demote'></input></p></td></tr>";
114.                 }
115.
116.                 echo "</tbody>";
117.
118.                 echo "</table>";
119.                 echo "</div>";
120.
121.                 echo "</form>";
122.                 echo "</div></div";
123.             }
124.         }
125.         ?>
126.         </div>
127.         </div>
128.     </body>
129. </html>
130.
```

usersHandler.php

```php
<?php

if (!($_SERVER['REQUEST_METHOD'] == 'POST'))#ensure form has been submitted
{
    require('loadPage.php');
    load('users.php');
}

require('connectDB.php');

foreach($_POST as $userID => $action)#key value pairs
{
    #POST turns '.' characters into '_' characters, which means all the '@.org.uk' parts of the emails get turned to '@_org_uk'
    #to avoid this, I've passed the userIDs instead

    #change user_type from 'user' to 'admin'
    if ($action == 'admin')
    {
        $sql = "UPDATE users SET user_type = 'admin' WHERE userID = '$userID'";
        mysqli_query($dbc,$sql);
    }

    #delete user from table
    elseif ($action == 'delete')
    {
        #delete any image files from directory that are part of this user's books that are up for sale
        $sql = "SELECT photo FROM books WHERE userID = '$userID'";
        $r = mysqli_query($dbc,$sql);
        while ($row = mysqli_fetch_array($r, MYSQLI_ASSOC))
        {
            #an image needs to be deleted from the directory
            if ($row['photo'] != 'n/a')
            {
                $fname = 'files/' . $row['photo'];
                if (is_writeable($fname))
                {
                    #delete image
                    unlink($fname);
                }
            }
        }

        #delete user books for sale from books table
        $sql = "DELETE FROM books WHERE userID = '$userID'";
        mysqli_query($dbc,$sql);

        #delete user from users table
        #user_type check is so that an admin can't inspect html code and delete other admins by putting in their userID as the value
        $sql = "DELETE FROM users WHERE userID = '$userID' AND user_type = 'user'";
        mysqli_query($dbc,$sql);
    }
```

```
52. }
53.
54. #reload users page
55. require('loadPage.php');
56. load('users.php');
```

adminsHandler.php

```
1. <?php
2. 
3. #ensure form has been submitted
4. if (!($_SERVER['REQUEST_METHOD'] == 'POST'))
5. {
6.     require('loadPage.php');
7.     load('users.php');
8. }
9. 
10. require('connectDB.php');
11. 
12. foreach($_POST as $userID => $action)
13. {
14.     #change user_type from 'admin' to 'user'
15.     if ($action == 'demote')
16.     {
17.         $sql = "UPDATE users SET user_type = 'user' WHERE userID =
    '$userID'";
18.         mysqli_query($dbc,$sql);
19.     }
20. }
21. 
22. require('loadPage.php');
23. load('users.php');
24. 
```

logout.php

```
1. <?php
2. 
3. #log out user by loading the login page - this will destroy the session as
   the login page gets loaded
4. 
5. require('loadPage.php');
6. load('login.php');
7. 
```

loadPage.php

```
1. <?php
2. 
3. #function to load a specified page
4. function load($page)
5. {
6.     $url = 'http://'.$_SERVER['HTTP_HOST'].dirname($_SERVER['PHP_SELF']);
7.     $url = rtrim($url,'/\\');
8.     $url .= '/'.$page;
9.     header("Location: $url");
10.    exit();
11. }
12. 
```

email.php

```php
<?php

#function to send an email
use PHPMailer\PHPMailer\PHPMailer;
use PHPMailer\PHPMailer\Exception;
require 'C:\Abyss Web Server\composer\vendor\autoload.php';
function email($recipientEmail,$subject,$body,$altBody)
{
    #creates a PHPMailer object
    $mail = new PHPMailer(TRUE);
    try {
       #$mail->SMTPDebug = 3; #used for testing connection
       $mail->setFrom('noreplybooksale@gmail.com', 'Admin');
       $mail->addAddress("$recipientEmail");

       #display body of email with html tags being considered
       $mail->isHTML(true);

       $mail->Subject = "$subject";
       $mail->Body = "$body";
       $mail->AltBody = "$altBody";


       #use SMTP
       $mail->isSMTP();

       #server address
       $mail->Host = 'smtp.gmail.com';

       #Use SMTP authentication.
       $mail->SMTPAuth = TRUE;

       #set encryption system to tls
       $mail->SMTPSecure = 'tls';

       #SMTP authentication username
       $mail->Username = 'noreplybooksale@gmail.com';

       #SMTP authentication password
       $mail->Password = [redacted];

       #Set the SMTP port (25 rather than 587)
       $mail->Port = 25;

       #Send the mail
       $mail->send();
    }
    catch (Exception $e)
    {
       echo $e->errorMessage();
    }
    catch (\Exception $e)
    {
       echo $e->getMessage();
    }
}
```

Email password has been redacted

```
57.
58. #function to generate a 10 random code
59. function getCode()
60. {
61.     #possible characters in the randomly generated string
62.     $chars =
    'abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ1234567890@_-.';
63.
64.     $code = '';
65.
66.     #concatenates 10 random characters from $chars variable onto $code
67.     for ($i=0; $i<10; $i++)
68.     {
69.         $code .= $chars[rand(0, strlen($chars) - 1)];
70.     }
71.     return $code;
72. }
73.
74. function genCode()
75. {
76.     require('connectDB.php');
77.     $code = getCode();
78.
79.     #check that code is unique
80.     $unique = FALSE;
81.     while ($unique == FALSE)
82.     {
83.         $sql = "SELECT * FROM users WHERE code = SHA2('$code',256)";
84.         $r = mysqli_query($dbc,$sql);
85.
86.         #code is unique
87.         if (mysqli_num_rows($r) == 0)
88.         {
89.             $unique = TRUE;
90.         }
91.         else
92.         {
93.             $code = getCode();
94.         }
95.     }
96.     return $code;
97. }
98.
```

pageStyles.css

```
* {
    margin:auto;
    font-family:Arial, sans-serif;
    scroll-behavior: smooth;
}

body {
    height:100%;
    width: fit-content;
    min-width: 100%;
    background-color: #9bc1e7;
}

html {
    height:100%;
    min-width:100%;
}

h1,h2,h4,p {
    color: #eccfcf;
}

/*ensures that pageContainer div is centered on each page, no matter page width*/
#outerPageContainer{
    display: flex;
    justify-content: center;
}

/*Login/registration*/
#entrancePageHeader {
    border-bottom: #002a44 solid 3px;
    margin: 0px;
    margin-bottom: 10px;
    background-color: #1c73c5;
    padding: 10px;
    padding-right: 0px;
    height:50px;
    display: flex;
    justify-content: center;
    align-items: center;
    overflow: hidden;
}

#entrancePageHeader > div {
    width: 33%;
}

#entrancePageHeader > #titleDiv {
    text-align: center;
}

#pageHeader {
    border-bottom: #002a44 solid 3px;
    margin: 0px;
    margin-bottom: 10px;
```

```
    background-color: #1c73c5;
    padding: 10px;
    padding-right: 0px;
    height:50px;
    display: flex;
    justify-content: center;
    align-items: center;
}

@media screen and (min-width:421px){
    #pageHeader > div {
        margin:10px;
    }
}

#pageHeader > div > h1 {
    text-align: center;
}

#pageContainer {
    margin:10px;
    width: 1320px; /*standardised width of each page*/
    height: fit-content;
}

#movePageButton{
    border-style: solid;
    border-width: 3px;

    float:right;
    font-size: 20px;
    padding:10px;
    border-radius: 10px;

    background-color: #002a44;
    border-color: #09d6cc;
    color: white;
}

#movePageButton:hover {
    background-color:#09d6cc;
    color: rgb(0, 0, 0);
    border-color: #002a44;
}

 .submitButton {
     font-size: 15px;
     width:290px;
     display: block;
     border-radius: 5px;
 }

 .loginForm {
     background-color: #002a44;
     width: 350px;
     padding:10px;
     border-radius: 15px;
     text-align: center;
 }
```

```
115.
116. #codeForm {
117.     margin-top: 15px;
118. }
119.
120. .loginForm p {
121.     width: 150px;
122. }
123.
124. .formContainer {
125.     background-color: #1c73c5;
126.     border-radius: 15px;
127.     padding:5px;
128.     overflow: hidden;
129. }
130.
131. .formContainer p
132. {
133.     float:left;
134.     text-align: left;
135. }
136.
137. .errorsBox {
138.     background-color: #dd2938;
139.     width: 350px;
140.     padding:10px;
141.     margin-top: 25px;
142.     border-radius: 15px;
143.     text-align: center;
144.     color: white;
145. }
146.
147. .errorsContainer {
148.     background-color: #cc0707;
149.     border-radius: 15px;
150.     padding:5px;
151.     overflow: hidden;
152. }
153.
154. #errorsContainer p {
155.     color: white;
156. }
157.
158. .formLink {
159.     background-color: #ffffff;
160.     border-radius: 10px;
161.     margin:5px;
162.     overflow: hidden;
163.     padding: 3px;
164.     display: flex;
165.     width: 130px;
166. }
167. .formLink p, .formLink p a:visited{
168.     text-align: center;
169.     color: #1b8eec;
170. }
171.
172. input {
173.     border-width: 0px;
```

```
174.     border-radius: 8px;
175.     background-color: #edf5e1;
176.     width:150px
177. }
178.
179. /*Forgot Password*/
180. #forgotPassBubbleContainer {
181.     float: none;
182.     width:370px;
183.     margin: 0 auto;
184.     margin-top: 15px;
185.     position: static;
186.     padding: 1px;
187. }
188.
189. #forgotPassBubbleContainerDetails{
190.     position: static;
191.     float: none;
192. }
193.
194. /*Home Page*/
195. #homeInfoBubbleContainer {
196.     width:300px
197. }
198.
199. #buttonTable tr{
200.     line-height: 0px;
201. }
202.
203. #buttonTable td > div{
204.     background-color: #002a44;
205.     padding: 5px;
206.     width:300px;
207.     height: 150px;
208.     border-radius: 15px;
209.     display: flex;
210.     justify-content: center;
211.     margin: 5px;
212.     margin-top: 0px;
213. }
214.
215. #buttonTable button {
216.     border-radius: 10px;
217.     background-color: #1c73c5;
218.     height:140px;
219.     border-radius: 15px;
220.     width: 290px;
221.     font-size: 40px;
222.     border-width: 0px;
223. }
224.
225. #buttonTable #usersButton {
226.     width:610px;
227. }
228.
229. #buttonTable #usersButtonContainer {
230.     width:620px;
231. }
232.
```

```
#buttonTable button:hover {
    background-color: #3f0090;
}

button:hover, input:hover{
    cursor:pointer;
}

#buttonTable h2 {
    color: rgb(255, 255, 255);
    font-size: 50px;
}

/*Navbar*/
#dropDownNav {
    display: none;
}

#navContainer {
    width:100%;
    margin:0px;
}

#mainNavBar {
    width:100%;
    min-width: 580px;
}

#mainNavBar h2 {
    font-size: 20px;
}

#mainNavBar ul {
    list-style-type: none;
    float: right;
    height:70px;
}

#mainNavBar ul li {
    float:left;
    text-align: center;
    display: flex;
    height:100%;
}

#logoutButton {
    margin-right: -10px;
}

#mainNavBar ul li button {
    width:100px;
    height:70px;
    border-width: 0px;
    margin:0px;
    /*keep text in middle of each button (counteract the margin-right: -10px; of the li)*/
    padding-right: 10px;

}
```

```
291.
292. #mainNavBar ul li button:hover {
293.     background-color: #1b8eec;
294. }
295.
296. /*Account details*/
297. .infoBubbleContainer {
298.     float:left;
299.     background-color: rgb(93, 92, 92);
300.     border-radius: 10px;
301.     position: absolute;
302. }
303.
304. .infoBubbleContainerDetails {
305.     float:left;
306.     background-color: rgb(93, 92, 92);
307.     /*on accountDetails.php - allows long email addresses to fit inside
    box*/
308.     width: fit-content;
309.     border-radius: 10px;
310.     position: absolute;
311. }
312.
313. .infoBubble {
314.     margin: 5px;
315.     padding: 5px;
316.     background-color: antiquewhite;
317.     border-radius: 10px;
318. }
319.
320. .infoBubble p {
321.     color: #2f2e2d;
322. }
323.
324. /*Users*/
325. .usersFormContainer {
326.     width:700px;
327.     padding:10px;
328.     border-radius: 15px;
329.     background-color: #002a44;
330.     margin-bottom: 15px;
331. }
332.
333. .usersForm {
334.     border-radius: 15px;
335.     padding:10px;
336.     background-color: #1c73c5;
337. }
338.
339. .usersForm input[type = 'submit'] {
340.     display: flex;
341.     justify-content: center;
342.     width: 190px;
343. }
344.
345. .usersForm .scrollTable thead th {
346.     position: sticky;
347.     top:0
348. }
```

```
349.
350. .tableHeader {
351.     font-size: 40px;
352.     /*allows title to be floated left, which allows submit button to be
    in header - without overflow:auto, div collapses*/
353.     overflow: auto;
354. }
355.
356. .tableHeader h2 {
357.     float:left;
358.     color: #edf5e1;
359.     margin:15px;
360. }
361.
362. .tableHeader input{
363.     font-size: 25px;
364.     margin: 5px;
365. }
366.
367. .scrollTable{
368.     max-height: 150px;
369.     overflow-y: scroll;
370. }
371.
372. .updateTable {
373.     float:right;
374. }
375.
376. .usersForm table {
377.     width:100%;
378.     border-width: 0px;
379.     height:10px;
380.     border-color: black;
381.     border-collapse: collapse;
382. }
383.
384. .usersForm table input[type='radio'] {
385.     display: block;
386. }
387.
388. .usersForm table thead th{
389.     background-color: #210153;
390.     border-color: #210153;
391.     color: #eccfcf;
392. }
393.
394. .usersForm table tbody {
395.     text-align: left;
396. }
397.
398. .usersForm table td {
399.     max-width: 130px;
400.     word-wrap: break-word;
401. }
402.
403. .usersForm table tr:nth-child(even) {
404.     background-color: rgb(49, 134, 134);
405. }
406.
```

```
.usersForm table tr:nth-child(odd) {
    background-color: rgb(51, 66, 126);
}

/*My Books*/
#pageContainer #uploadBookContainer {
    float:left
}

#pageContainer .booksTableContainer {
    float: right;
}

#uploadBookContainer {
    background-color: #002a44;
    width: 350px;
    padding:10px;
    border-radius: 15px;
    text-align: center;
}

#uploadBookForm {
    background-color: #1c73c5;
    border-radius: 15px;
    padding:5px;
}

#uploadBookForm p
{
    float:left;
    text-align: left;
    width: 150px;
}

#chooseFile {
    width:300px;
    color: black;
}

#reUploadFile {
    background-color: #cc0707;
    border: solid 3px #dd2938;
    border-radius: 8px;
    margin:5px;
}

input[type="file"] {
    display: flex;
    font-size: 17px;
    color: #b8b8b8;
    margin-left: 20px;
  }

input[type = 'checkbox'] {
    width:100%
}

.dropDownMenu {
    width:154px;
```

```
466.     border-radius: 8px;
467. }
468.
469. #updateBooksButton {
470.     float:right;
471.     margin:15px;
472. }
473.
474. .booksTableContainer {
475.     background-color: #002a44;
476.     width:900px;
477.     padding:10px;
478.     border-radius: 15px;
479. }
480.
481. .booksTable {
482.     background-color: #1c73c5;
483.     border-radius: 15px;
484. }
485.
486. .booksTable input[type = 'submit'] {
487.     display: flex;
488.     justify-content: center;
489.     width: 190px;
490. }
491.
492. .booksTable table {
493.     margin-top: 10px;
494.     margin-bottom: 10px;
495.     text-align: center;
496.     border-width: 0px;
497.     border-color: black;
498.     border-collapse: collapse;
499. }
500.
501. .booksTable table td {
502.     height:160px;
503.     max-width:85px;
504.     min-width: 85px;
505.     word-wrap:break-word;
506. }
507.
508. #photoCell {
509.     width:130px
510. }
511.
512. .booksTable table tr:nth-child(even) {
513.     background-color: rgb(49, 134, 134);
514. }
515.
516. .booksTable table tr:nth-child(odd) {
517.     background-color: rgb(51, 66, 126);
518. }
519.
520. .booksTable table thead th {
521.     background-color: #210153;
522.     border-color: #210153;
523.     color: #eccfcf;
524. }
```

```
525.
526. #bookTableScroll {
527.     max-height: 530px;
528.     overflow-y: scroll;
529. }
530.
531. #bookTableScroll thead th {
532.     position: sticky;
533.     top:0
534. }
535.
536. table h3 {
537.     font-size: 15px;
538. }
539.
540. #filterAndDisplayBooks {
541.     overflow:visible;
542.     height: 530px;
543. }
544.
545. #myBooksErrorsBox {
546.     float: left;
547.     margin-bottom: 15px;
548. }
549.
550. @media screen and (min-width:421px)
551. {
552.     #myBooksErrorsBoxMobile {
553.         display: none;
554.     }
555. }
556.
557. /*Browse books*/
558. #filterContainer {
559.     width: 1320px;
560.     /*stop parent div from collapsing when child div is floated*/
561.     overflow:auto;
562. }
563.
564. #filterFormContainer {
565.     float:left;
566.     margin-top: 0px;;
567. }
568.
569. #applyFilterButton {
570.     float:right;
571.     height: 30px;
572.     width:150px;
573.     font-weight: bold;
574.     font-size: 15px;
575.     margin:5px;
576. }
577.
578. #clearFilterButton {
579.     float:left;
580.     font-size: 15px;
581.     border-radius: 5px;
582.     border-width: 0px;
583.     height:30px;
```

```
584.     width:150px;
585.     margin:5px;
586.     font-weight: bold;
587. }
588.
589. #browseInfoBubbleContainer {
590.     position:static;
591.     float:right;
592.     margin:0px;
593. }
594.
595. #browseInfoBubble {
596.     position:static;
597.     width:370px;
598. }
599.
600. #filterErrorsBox {
601.     margin-top:0px;
602. }
603.
604. #updateBrowseBooksButton {
605.     margin: 10px 0px 0px 25px
606. }
607.
608. #browseBooksContainer {
609.     background-color: #002a44;
610.     padding:10px;
611.     margin-top: 25px;
612.     border-radius: 15px;
613. }
614.
615. #bookRow:hover {
616.     background-color: rgb(0, 255, 255);
617.     cursor: pointer;
618. }
619.
620. #bookRow:hover p {
621.     color: black;
622. }
623.
624. .buyBookButton:hover p {
625.     color: black;
626. }
627.
628. /*Buy book*/
629. #bookDetailsContainer {
630.     padding:10px;
631.     background-color: #002a44;
632.     width: 400px;
633.     border-radius: 15px;
634. }
635.
636. #bookDetails {
637.     background-color: #1c73c5;
638.     padding:10px;
639.     border-radius: 15px;
640. }
641.
642. #bookDetails p {
```

```
643.     word-wrap: break-word;
644. }
645.
646. #bookPhoto {
647.     display: flex;
648.     justify-content: center;
649. }
650.
651. #confirmBook {
652.     margin-top: 10px;
653. }
```

mediaQueries.css

```
@media screen and (max-width:420px) {
    #pageContainer {
        /*has margin of 10px on all sides, so this makes total page have a width of 420px*/
        width:400px;
    }

    /*Login Page*/
    #titleDiv {
         width: 100%;
    }

    #titleDiv h1 {
        font-size: 25px;
    }

    /*Home Page*/
    #homeTitle {
        text-align: center;
    }

    #homeInfoBubbleContainer {
        position:static;
        float:none;
        margin-top: 0px;
        margin-bottom: 10px;
        padding:1px
    }

    #buttonTable td > div {
        width:180px;
    }

    #buttonTable button {
        width: 170px
    }

    #buttonTable #usersButtonContainer {
        width:380px;
    }

    #buttonTable #usersButton {
        width: 370px;
    }

    #buttonTable h2 {
        font-size: 40px;
    }

    #usersButtonContainer h2{
        font-size: 60px;
    }

    /*NavBar*/
    #pageHeader div {
        width:50%;
```

```
56.         overflow: auto;
57.     }
58.
59.     #navContainer {
60.         display: none;
61.     }
62.
63.     #dropDownNav {
64.         display: flex;
65.         height:fit-content;
66.         overflow: auto;
67.     }
68.
69.     #navContent {
70.         height:fit-content;
71.         padding: 0;
72.     }
73.
74.     #dropDownNav ul {
75.         list-style-type: none;
76.         width: fit-content;
77.         height: fit-content;
78.         padding:0;
79.         z-index: 1;
80.     }
81.
82.     #dropDownNav button {
83.         width:100px;
84.         height:50px
85.     }
86.
87.     #dropDownNav li {
88.         text-align: center;
89.     }
90.
91.     #navContent ul li a {
92.         display: block;
93.     }
94.
95.     /*Browse*/
96.     #filterContainer {
97.         width: 400px;
98.         padding: 0px;
99.     }
100.
101.      #browseInfoBubbleContainer {
102.          float:none;
103.          width:370px;
104.          padding:1px;
105.          position: static;
106.          margin:0 auto;
107.      }
108.
109.      #browseInfoBubble {
110.          width:350px
111.      }
112.
113.      #filterFormContainer {
114.          float:none;
```

```
115.         margin-top: 15px;
116.     }
117.
118.     #browseBooksContainer {
119.         width: 350px;
120.         margin-top: 15px;
121.     }
122.
123.     #browseBooksTable {
124.         width:100%;
125.         margin:0 auto;
126.     }
127.
128.     #browseBooksTable form {
129.         margin: 0 5px 0 5px;
130.     }
131.
132.     #filterErrorsBox {
133.         margin-top: 15px;
134.     }
135.
136.     #updateBrowseBooksButton {
137.         margin:0 auto;
138.         margin-top:10px
139.     }
140.
141.     /*My Books*/
142.     #pageContainer #uploadBookContainer {
143.         float:none
144.     }
145.
146.     #pageContainer .booksTableContainer {
147.         float: none;
148.     }
149.
150.     #filterAndDisplayBooks {
151.         height:fit-content
152.     }
153.
154.     #myBooksTableContainer {
155.         width:350px;
156.         margin-top: 15px;
157.     }
158.
159.     #updateBooksButton {
160.         margin:0 auto;
161.         margin-top:10px;
162.         float: none;
163.     }
164.
165.     #myBooksTableContainer form {
166.         margin: 0 5px 0 5px;
167.     }
168.
169.     #myBooksErrorsBox {
170.         display: none;
171.     }
172.
173.     #myBooksErrorsBoxMobile {
```

```
174.         margin-top: 15px;
175.     }
176.
177.     /*Users*/
178.     .usersFormContainer {
179.         width:350px
180.     }
181.     .usersForm .tableHeader {
182.         text-align: center;
183.
184.     }
185.     .usersForm .tableHeader h2{
186.         float: none;
187.     }
188.
189.     .usersForm input[type='submit'] {
190.         float:none;
191.         margin:0 auto;
192.         margin-bottom: 10px;
193.     }
194.
195.     .usersForm input[type='radio'] {
196.         margin: 0 auto;
197.         width:100%
198.     }
199.     /*Account Details*/
200.     #accountDetailsInfoBubbleContainer {
201.         float:none;
202.         width:370px;
203.         padding:1px;
204.         position: static;
205.         margin:0 auto;
206.         margin-bottom: 15px;
207.     }
208.
209.     #accountDetailsInfoBubble {
210.         text-align: center;
211.     }
212.
213.     /*Buy Book*/
214.     #bookDetailsContainer {
215.         margin-top: 15px;
216.         width:350px
217.     }
218. }
```

Database code:

SQL to create database:

```
1. CREATE DATABASE IF NOT EXISTS bookstore_db;
```

SQL to create tables:

users:

```
1. CREATE TABLE IF NOT EXISTS users(
2. userID INT PRIMARY KEY AUTO_INCREMENT,
3. email VARCHAR(60) NOT NULL UNIQUE,
4. password VARCHAR(256) NOT NULL,
5. user_type VARCHAR(15) NOT NULL CHECK(user_type
   IN('super_admin','admin','user','pending')),
6. code VARCHAR(256) UNIQUE,
7. timeOfCode TIMESTAMP);
```

books:

```
1. CREATE TABLE IF NOT EXISTS books(
2. bookID INT PRIMARY KEY AUTO_INCREMENT,
3. userID INT NOT NULL,
4. title VARCHAR(60) NOT NULL,
5. ISBN VARCHAR(13) NOT NULL CHECK(CHAR_LENGTH(ISBN) IN (10,13)),
6. author VARCHAR(60) NOT NULL,
7. qualification VARCHAR(60) NOT NULL,
8. subject VARCHAR(60) NOT NULL,
9. price DECIMAL(10,2) NOT NULL CHECK(price >= 0 and price <= 99999999.99),
10. bookCondition VARCHAR(60) NOT NULL,
11. photo VARCHAR(255) NOT NULL,
12. FOREIGN KEY (userID) REFERENCES users (userID));
```

Implemented tables:

```
mysql> explain users;
+------------+--------------+------+-----+---------+----------------+
| Field      | Type         | Null | Key | Default | Extra          |
+------------+--------------+------+-----+---------+----------------+
| userID     | int          | NO   | PRI | NULL    | auto_increment |
| email      | varchar(60)  | NO   | UNI | NULL    |                |
| password   | varchar(256) | NO   |     | NULL    |                |
| user_type  | varchar(15)  | NO   |     | NULL    |                |
| code       | varchar(256) | YES  | UNI | NULL    |                |
| timeOfCode | timestamp    | YES  |     | NULL    |                |
+------------+--------------+------+-----+---------+----------------+
```

```
mysql> explain books;
+---------------+---------------+------+-----+---------+----------------+
| Field         | Type          | Null | Key | Default | Extra          |
+---------------+---------------+------+-----+---------+----------------+
| bookID        | int           | NO   | PRI | NULL    | auto_increment |
| userID        | int           | NO   | MUL | NULL    |                |
| title         | varchar(60)   | NO   |     | NULL    |                |
| ISBN          | varchar(13)   | NO   |     | NULL    |                |
| author        | varchar(60)   | NO   |     | NULL    |                |
| qualification | varchar(60)   | NO   |     | NULL    |                |
| subject       | varchar(60)   | NO   |     | NULL    |                |
| price         | decimal(10,2) | NO   |     | NULL    |                |
| bookCondition | varchar(60)   | NO   |     | NULL    |                |
| photo         | varchar(255)  | NO   |     | NULL    |                |
+---------------+---------------+------+-----+---------+----------------+
```

Create database user – for database connection:

```
1. CREATE USER IF NOT EXISTS 'book_admin'@'localhost' IDENTIFIED BY
   'C7B?mJrv$P=q?x6n';
2.
3. GRANT SELECT, UPDATE, DELETE, INSERT ON bookstore_db.* TO
   'book_admin'@'localhost';
```

SQL for inserting in super_admin user:

```
1. INSERT INTO users (email, password, user_type) VALUES
   ('noreplybooksale@gmail.com', SHA2('x@a;2qE+ASd:_4BL',256),
   'super_admin');
```

Log of Ongoing Testing:

What is being tested:	Issues encountered:	Solution:	References:	Date:
Email function	Failed connection – connected party didn't respond after a period of time	I was using port 587 (standard SMTP port) – but this was blocked on the school internet system (where the program is to be used). The solution was to switch over to port 25 which wasn't blocked.	https://kinsta.com/blog/smtp-port/	December 3rd
Email function	Emails were getting picked up as potential spam, and being held in quarantine for several hours	I just had to pick the subject of my emails carefully – certain keywords in the subject seemed to be triggering this, so I avoided subjects like 'New Password Code' as they led to the emails being placed in quarantine.		December 5th
Using verification code to login	All codes seemed to have expired (exceeded the 5 minutes) when being entered, even if they were entered immediately after their issue. The database wasn't updating the users table with the new timeOfCode when a new code was issued when forgotPassHandler.php was called, so each code was recorded as have being issued at the time the user first registered, so any attempts to login with this code were invalid if they tried to use the Forgot Password feature 5 minutes or more after they had registered.	Needed to set timeOfCode to NOW() in the update statement, alongside updating the code.		December 8th

Changing a user's password at accountDetails.php	Since passwordChange.php used session_start(); to access user details and then required accountDetails.php, which used session_start(); as well (to ensure the user is logged in) this flagged up a warning as it was trying to start a session when one was already active.	To fix this, I closed the session in passwordChange.php without deleting its contents – I used the function session_write_close(); to do this.	https://www.php.net/manual/en/function.session-write-close.php	December 11th
Trying to delete a user from the users.php	Initially, the name of each row in the users table on users.php was each user's email address (as these were all unique) – however, there would be no matches with the rows in the database, as the POST variable was converting all the dots in the email addresses to underscores	Initially, I was going to use some regex to change all the underscores back to dots, but this wouldn't work as if the original email address contained an underscore, it would get changed to a dot (and I couldn't simply change all the dots after the @ as there could've been dots before it) – so instead, I changed the name of each row to each user's userID – this was an integer, so didn't cause any problems.	https://www.php.net/manual/en/language.variables.external.php	December 19th
Putting a book entry into the books table	Wasn't working at all, and was throwing up errors	I had called one of the columns 'condition' in the books table in the database – and after research, it turns out this is a reserved keyword. So I changed it to bookCondition	https://dev.mysql.com/doc/refman/8.0/en/keywords.html	December 20th
Trying to save photo files to a directory	I didn't have a good way of naming the files (didn't use original names as there could've been two photos named the same	I named each photo the current dateTime, concatenated with the userID of the person uploading it. I went to a	https://www.tutorialspoint.co	December 22nd

	thing, and auto increment would make it tricky to retrieve the correct photo)	very high degree of accuracy (up to microseconds) so that it would be practically impossible for one user to upload two photos at the exact same time.	m/php/php_function_date_create.htm	
Creation of navbar	Navbar wasn't flush with edge of page on right	This was caused by the pageHeader having some padding – solved by giving the rightmost button negative margin on the right.	https://www.educba.com/negative-margin-css/	December 28th
Creation of scrollable tables (e.g. browse.php)	Tried to put a div inside of a table, and make its contents scrollable, but the div was appearing in another place when inspected	I found out that you can't just put a div in a table like that so that it wraps around the tbody – the fix was to put the whole table in a div and then make that scrollable	https://www.sitepoint.com/community/t/how-to-put-div-between-table-rows/3611/2	January 2nd
Ensure that headers in scrollable tables stay fixed	After doing the above fix, the headers would also scroll (as whole table scrolled)	To fix this, gave the <th> tags in the thead the properties: position: sticky; top: 0;	https://www.w3docs.com/snippets/html/how-to-create-a-table-with-a-fixed-	January 2nd

			header-and-scrollable-body.html	
Testing placement of errors box on myBooks.php	It was appearing inside the wrong div	Tags were not matched up properly – originally was <table> </table> which needed to be changed to <table></table>		January 5th
Making upload book form sticky	Was unable to set a default value for the upload file input (as in, if there's an error and the user needs to adjust their inputs, it keeps their original inputs after the form submission)	Turns out this is not possible for security reasons – as could allow files from user's machine to be pulled off by form and without consent – instead, put a prompt asking them to re-upload the file	https://stackoverflow.com/questions/2665058/set-default-value-for-a-input-file-form	January 7th
Making default values for sticky forms	Was only displaying first word of each value, and stopping at each space	Needed to put the php in quotes e.g. value='<?php … ?>' so that the whole string was taken as the value, not just the first word	https://stackoverflow.com/questions/3078192/how-to-set-html-value-attribute-with-spaces	January 10th

Making clickable rows in browse.php book table	Tried to wrap the <tr> tags in <a> tags, but this didn't work as the row didn't become clickable	This is invalid html, so I used the onclick method to change the location by applying it to the <tr> tag. This method didn't deal with spaces well, and broke when some of the key value pairs in the URL had spaces – so used str_replace() to change each space to '%20' – as they appear in the URL – so that they can be accessed by a GET	https://www.geeksforgeeks.org/how-to-make-whole-row-in-a-table-clickable-as-link-in-bootstrap/	January 16th
Allowing checkboxes to be used in clickable rows	After implementing the above, it became impossible to click the checkboxes as an admin user, in the browse.php books table, without being taken to the buyBook.php page (as the whole row is clickable)	I applied the onclick method to each <td>, except the one with the checkbox (it was no longer clickable), instead of applying the method to the entire row		January 17th
Trying to centre a div in a div	It would sometimes gravitate towards the top left of the parent div	I needed to apply margin: 0 auto; to the inner div	https://www.freecodecamp.org/news/how-to-center-anything-with-css-align-a-div-text-and-more/	January 21st

Testing media queries on an android device	The media queries weren't being applied on an android device	In order to make android devices give their real size, apply: <meta name="viewport" content="width=device-width" />	https://www.freecodecamp.org/news/how-to-center-anything-with-css-align-a-div-text-and-more/	January 25th
Testing drop down navbar for mobile devices	Navbar was activated by hover, so on a mobile device, needed to be held down, which made it awkward to then click one of the buttons that appeared	Made the drop down navbar clickable, by getting some JS to toggle the rest of the navbar's display between 'none' and 'block'	https://stackoverflow.com/questions/55442477/make-text-appear-disappear-on-button-click	January 29th

Research and development of new skills and/or knowledge

SMTP Email system:

In order to ensure that users were part of the school system, I wanted to make sure that they had access to an email address of the form '…@esms.org.uk', as such emails are only issued to members of the school. To ensure this, I emailed them a verification code to login for the first time, to prove they have access to a valid email address. I also use the email system to send verification codes if they forgot their password, and to send the details for interactions between sellers and potential buyers.

To achieve this, I used PHPMailer to send the emails, following this: https://alexwebdevelop.com/phpmailer-tutorial/ tutorial to install (using Composer), set up and use PHPMailer.

I then created a function to send the email, which took the recipient email, subject, email body and email body without html as parameters. I used Gmail's SMTP servers, so created a Gmail account (noreplybooksale@gmail.com) from which all the emails would be sent from. The function creates a PHPMailer object, and then sets up all the parameters of the email, including picking the port (25), picking the encryption system (tls) and several other things. Gmail's SMTP servers allow me 500 free emails per day, which is more than enough for the scale of this project.

Input type='file':

To allow users to upload a file of the book they were uploading, I used input type='file' on the myBooks.php page. The submitted file had its details stored in the superglobal $_FILES. I used the function getimagesize on the submitted file to see if it was an image (it returned false if it wasn't) – the actual file was stored in $_FILES['photo']['tmp_name'] as this is the temporary filename of the file once it has been uploaded.

I then extracted the extension from the file and checked to see if it was valid (I only accepted .jpg, .jpeg and .png). I then checked the actual file size and ensured it didn't exceed the limit of 3Mb.

If no errors were flagged up, I called the file the user's userID concatenated with the current time stamp (up to microseconds in accuracy) so that the filename was unique (as practically impossible for a user to upload two files at the exact same time with this degree of accuracy). I then saved the file into a directory called files/ with the function move_uploaded_file().

I used the following for my research:

https://www.php.net/manual/en/features.file-upload.post-method.php

https://www.w3schools.com/php/php_file_upload.asp

Regex:

In order to ensure that people registered with email addresses of the form '…@esms.org.uk', I used regex to pattern match any inputs. I used it in two places, as client-side and as server-side validation. On the client-side, I put pattern='^[a-zA-Z0-9.-_]+@esms.org.uk' in the form and on the server-side preg_match("/^[a-zA-Z0-9.-_]+@esms.org.uk/"). The '^' looks for a match at the start of a string and [a-zA-Z0-9.-_] signifies any combination of lowercase, uppercase, numbers, dots, dashes and underscores. Altogether, with the '+', it looks for a string that starts with any number (at least one) of the characters previously listed, followed by '@esms.org.uk'. Most of my research was found here: https://www.w3schools.com/php/php_regex.asp https://www.sitepoint.com/client-side-form-validation-html5/

Getting current Date and Time:

I needed to be able to get the current date and time so that when a verification code is issued, the time it is issued can be recorded, and when one is entered, it is possible to tell if it was entered within 5 minutes of it being issued. I was able to fetch the timeOfCode for each code from the database, and pass the string into date_create() to create a DateTime object. I could then use date_create() again without giving it any parameters to get a DateTime object for the current time. I could then pass both objects into the date_diff() function, which would create a DateInterval object – I was able to see the total time elapsed by accessing each of this object's properties in turn (e.g. number of minutes = dateIntervalObject->i).

I also used the date_create() function to get the current timestamp for the naming of the photo files. I then passed the DateTime object into date_format() along with 'ymdHisu' to get the dateTime as a string, of the form yymmddhhmmssuuuuuu.

Most of my research came from here:

https://www.geeksforgeeks.org/php-date_diff-function/

https://www.php.net/manual/en/class.dateinterval.php

https://www.w3schools.com/Php/func_date_date_create.asp

Testing

Final Test Plan:

What is to be tested:	Reason for test:	How it will be tested:	Expected output:
Page layout (CSS) for desktop viewing of each page	Ensure each page displays its content correctly in an easy to read way	Screen shots of each page showing all the content	All content displays correctly
Page layout (CSS media queries) for mobile devices (screen-width <= 420px) of each page	Ensure each page is mobile friendly, and displays its content well on a smaller screen	Screen shots of each page as viewed from a device of screen-width <= 420px	CSS media queries adjusts placement/sizing of content to appropriately fit the new screen size
Requirement 1	Ensure it is possible for school students to register an account	Attempt to register an account with an '…@esms.org.uk' email address and a valid password (>5 and <= 256 characters)	Get verification code sent to email address and confirmation message appearing saying they have registered
Requirement 2	Ensure that a registered user can login	Attempt to login with the verification code and then with the email and password	Both methods should work and allow the user to progress past the login page
Requirement 3	Ensure that users can still login if they have forgotten their password	Request a verification code from the Forgot Password page and try and login with it	The code should be emailed and then the user should be able to login with it
Requirement 4	Ensure that users are able to navigate between the different pages	Clicking the buttons on the home page and navbar, and ensuring that they all go to the desired pages	Each button takes the user to the indicated page
Requirement 5	Ensure that users can change their password	First login with old password, then attempt to change a user's password on the account details screen, and then login with the new password	User should be able to login with new password – and the database will update the password in the user's record

Requirement 6	Ensure that a user can manage their own books, by seeing the ones up for sale, and adding or removing entries	Try to add a book on the my books page for sale as a user. View the new book is there and then delete it by ticking the checkbox and then submitting the form	Books added for sale should appear in table and deleting a book should remove it from the table
Requirement 7	Ensure that users can browse the books that are up for sale (other than their own) with optional use for a filter	Check to see if a user can see all the available books for sale – by checking with the database by using a select statement. Then try applying several filters by using the form (again, checking by running a select statement on the database to ensure that they are seeing the correct entries)	That without a filter, a user can see each book up for sale (except their own) and then can see only books that meet the filter once it is applied
Requirement 8	Make sure that the book rows in the table are clickable, and take the user to the correct confirmation page	Try clicking on one of the rows in the browse books table, and ensuring that the page the user is then taken to contains the information of the correct book	The user is taken to a confirmation page that contains the details of the book they clicked on
Requirement 9	Ensure that users can confirm they would like to buy a book, sending them an email	Get a user to click confirm on the buy book page and then check if they received an email with the correct information (with the seller's details and the information about the book being looked at)	That the buyer receives an email with the details of the seller and the book

Requirement 10	Ensure that a user is able to logout	Click the logout button	It should take the user back to the login page
Requirement 11	Ensure that an admin user is able to delete another user's book entry – so they can control what gets posted	By going onto the browse page and filling in the checkbox by a book entry, and then submitting the form	The entry should be removed from the browse table
Requirement 12	Ensure that admin users can view, promote and delete standard users	Use select statement to see all users and confirm that admin users can view them all on users page, then promote a user – check it changes their user_type to 'admin'. Then delete a user – check it deletes that user from the users table, as well as any books they had up for sale	That the admins can see all standard users, and can delete/promote any of them
Requirement 13	Ensure that the super_admin can view/demote admin users – so they remain in ultimate control of the website	Select all admins from the users table and see if this matches the admins that the super_admin can see in the admins table on the users page. Then demote one of them, and see if they have been moved to the users table and check the database to support this claim by checking their user_type	The super_admin is able to view all admins, and demote them to 'user' status
Requirement 14.1 – 14.10	Ensure that forms are capable of validating inputs	Show restricted choice on form, enter invalid inputs – emails, book titles, editions,	All forms validate user inputs, providing error messages where appropriate

		conditions that are > 60 characters, passwords that are < 5 or > 256 characters, ISBN numbers that aren't 10 or 13 digits, a negative price, an email not of the form '…@esms.org.uk', or one that has already been verified, not entering required fields (emails, passwords, titles etc…), SQL injection, enter a book photo of wrong file type (not .jpg, .jpeg or .png), enter a photo that's > 3Mb, incorrect verification code, verification code that has expired (entered more than 5 minutes after issue), invalid login details/registration details and invalid (non-matching) passwords when trying to change a password	
Requirement 15	Ensure that a user's details get saved to the users table in the database	Get a user to register and then check the users table before and after with select statements to see if their details appear	That the registered user's details appear in the users table with user_type = 'pending'
Requirement 16	Ensure that a user can re-request a verification code in case they lose it or it expires	Register an account but don't use verification code within 5 minutes (so	That the user can re-register since their code expired, and get a new verification

		user_type remains 'pending'), then re-register and see if a new verification code gets sent by email (and check it was updated in the users table)	code emailed to them – the code will also be updated in the users table
Requirement 17	Ensure that only logged in users can progress past the login/registration/forgot password pages	By going to the login page, and then entering the URL of pages that exist behind it – e.g. home.php	It should not let them access it and just direct them to the login page
Requirement 18	Ensure that users can request a verification code if they forget their password	Get a registered user to go to the forgotten password page and request a verification code – see if it gets sent by email and then try and login with it – check database before and after to see if it has been updated	They should receive the verification code by email (see it gets updated in the database) and then be able to login with it
Requirement 19	Ensure that users can upload a book – it gets added to database and any images get added to the files/ directory	Get a user to upload a book, then use select statement before and after upload to see if book details have been added to the books table – then also check if any photos uploaded have been added to the files/ directory	That the uploaded book details get added to the books table and any uploaded files get saved in the files/ directory
Requirement 20	Ensure a user can delete a book from the my books page	Get a user to select the delete checkbox next to a book entry and then submit the form. Then check to see if the book details have been removed from the database and check	When the user submits the form, the book details get removed from the database (and consequently the table on the my books page) and also any photos in the

		to see if any photos have been removed from the files/ directory	files/ directory have been removed
Requirement 21	Ensure that the books that are up for sale are displayed	Login and view the books in the browse section – check these match with those in the database by running a select statement	It displays all the books (except those of the current user) in the table on the browse page
Requirement 22	Ensure that a user can only see books that fit the current filter	Login and apply a filter – ensure those are the correct books on display by running a query that only selects book entries that match the filter	It only displays book entries that match the filter in the browse table
Requirement 23	Ensure that users can select a book to buy and be taken to confirmation page	Get a user to click on a book entry in the browse table – check that the data displayed is of the correct book, and check this data also appears correctly in the URL	Clicking on the entry takes them to the buy book page with the corresponding details of the book that they selected and the correct key value pairs in the URL
Requirement 24	Ensure that an email with the right information gets sent to the buyer and seller when a user confirms they would like to buy a book	Once a user is on the buy book page of a book, they click confirm – the two recipients then need to check their emails to ensure that the correct information has been sent – this can be verified by comparing it with the book's details and ensuring both parties receive the other's correct email address	An email gets sent to both parties with the correct book's information and each other's details

Requirement 25	Ensure that admins are able to control what is posted and delete other users' book entries from the site/database	Get an admin to check the delete checkbox by a book entry on the browse page, and then submit the form – check if the book entry is removed from the database (by comparing with select statements before and after submitting the form), and that if the book had a corresponding image, it gets deleted from the files/ directory	Book details/image get deleted from the system
Requirement 26	Ensure that once a user logs out, the session gets terminated, for security reasons	Once a user clicks logout, they should try and access a page behind the login page by typing it in the URL – e.g. home.php	It should just redirect them to login.php as the session should have been destroyed when they logged out and landed on the login.php page
Requirement 27	Ensure that admins can view/delete/promote users	SELECT all users from users table and see if this corresponds to the users that an admin can see in the users table on the users.php page. Then, promote one of them, and SELECT their data from the database and see if their user_type field has changed from 'user' to 'admin'. Then try deleting another user, and check the database to see if their account, their book entries, and any	Admins are able to view/delete/promote all standard users

		images they put up have all been removed	
Requirement 28	Ensure that the super_admin can see/demote admins	Ensure that the admins that the super_admin sees in the admins table on the users.php page correspond to the admins shown when the admin type users are selected from the users table in the database. Then, try and demote one of them – they should appear in the users table now, and their user_type in the users table should have changed from 'admin' to 'user' when checked in the database	The super_admin is able to view/demote all admin type users

Persona Testing Plan:

The following test cases will be completed by another Advanced Higher Computer Science student. They will first act as a standard 'user' using the site to register, login, request a new verification code, change their password, put a book up for sale, delete one of their books and place an order for a book. They will then be promoted to an 'admin' status and asked to delete another user's book, promote and delete another user. Finally, they will be given access to an account with 'super_admin' status and asked to demote a specific user.

Test:	**Task to be completed:**
Test case 1	A user will be asked to register and login for the first time.
Test case 2	A user will be asked to login without the use of their password (to test the forgot password feature) and then go and change their password (mimicking someone forgetting their password).
Test case 3	A user will be asked to put a book up for sale.
Test case 4	A user will be asked to delete a book they have put up for sale.
Test case 5	A user will be asked to place an order for a specific book – e.g. find a French dictionary in the system (this is to test the filter and email systems).
Test case 6	A user with an admin account will be asked to delete a specific book from the database.
Test case 7	A user with an admin account will be asked to promote a specific user and delete a specific user.
Test case 8	A user with a super_admin account will be asked to demote a specific admin.

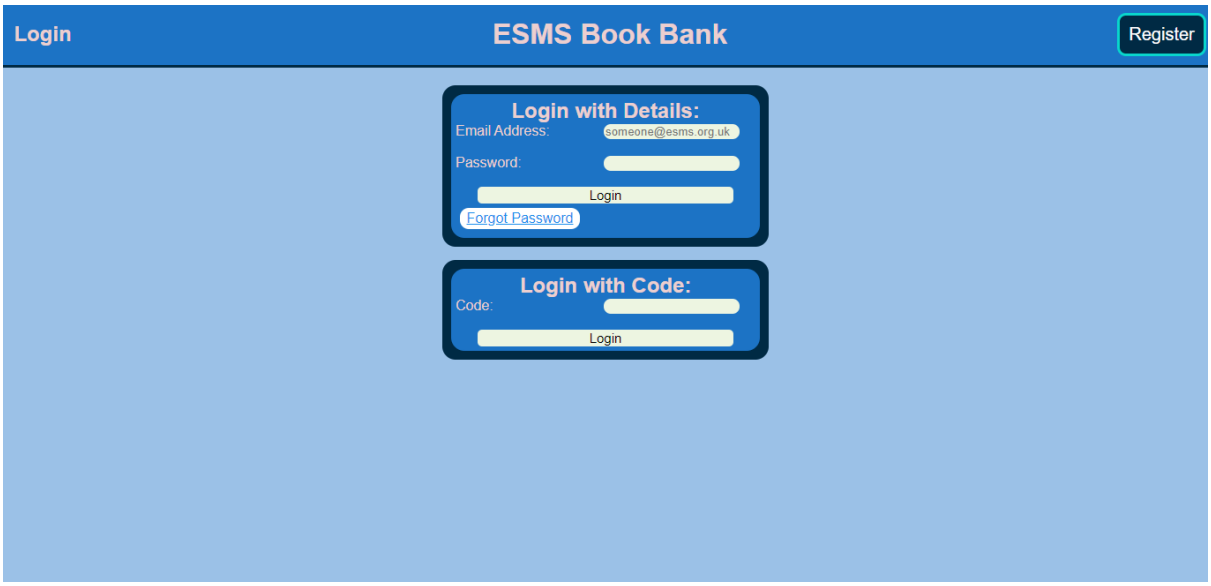
Requirements Testing:

Main CSS

Test: Page layout (CSS) for desktop viewing of each page (screenshots of pages as viewed in browser)

login.php

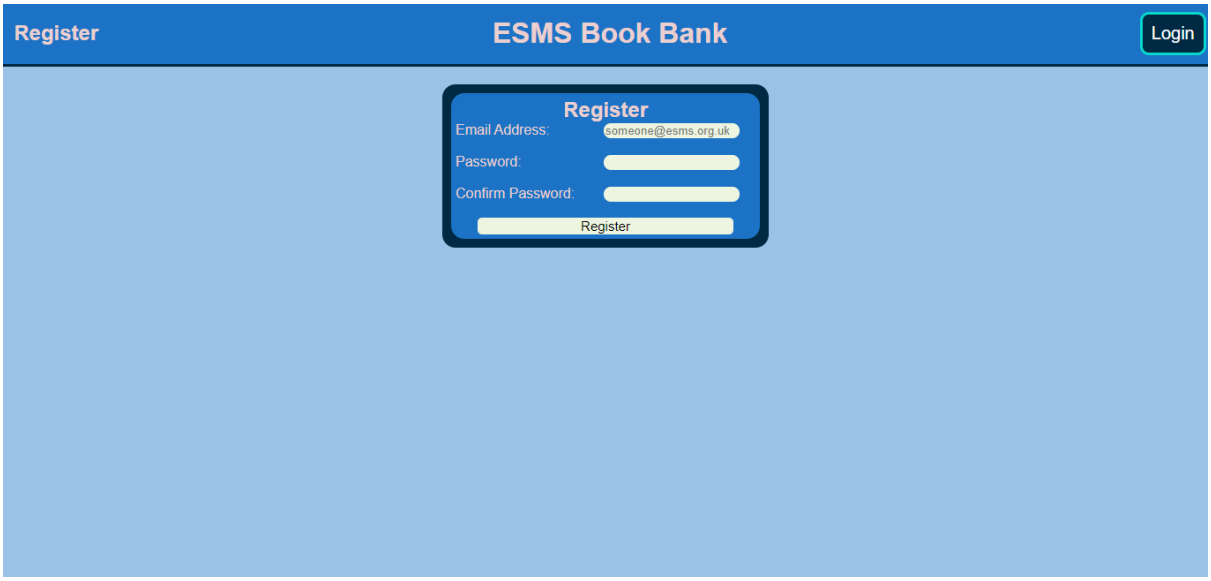
Expected output: both 'Login with Details' and 'Login with Code' forms are correctly displayed in middle of screen and there's a button to register.php and to forgotPass.php



Actual output: all elements displayed correctly

register.php

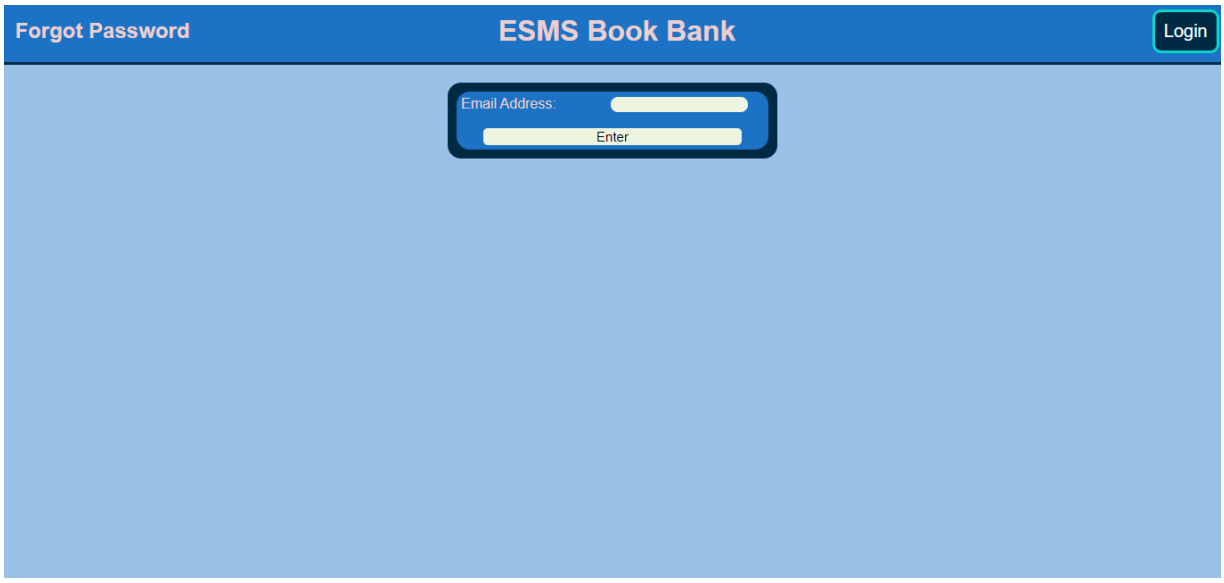
Expected output: 'Register' form is displayed in middle of screen and there's a button to login.php



Actual output: all elements display correctly

forgotPass.php

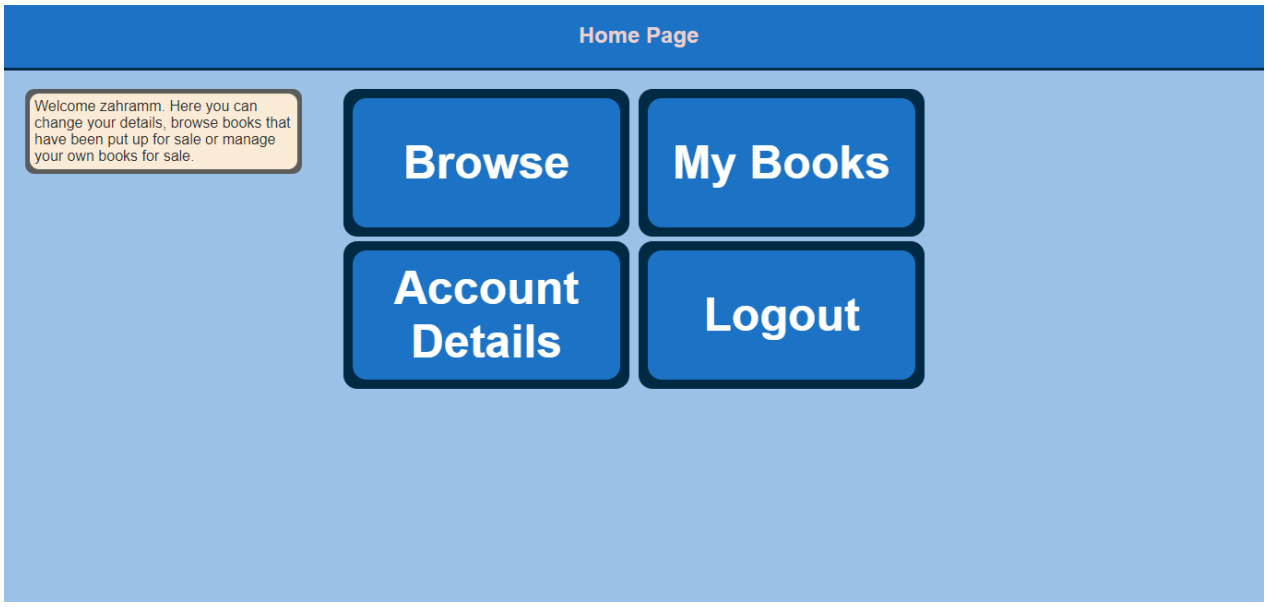
Expected output: forgot password form displays in middle of the page and there's a button to login.php



Actual output: all elements display correctly

home.php (for a standard user)

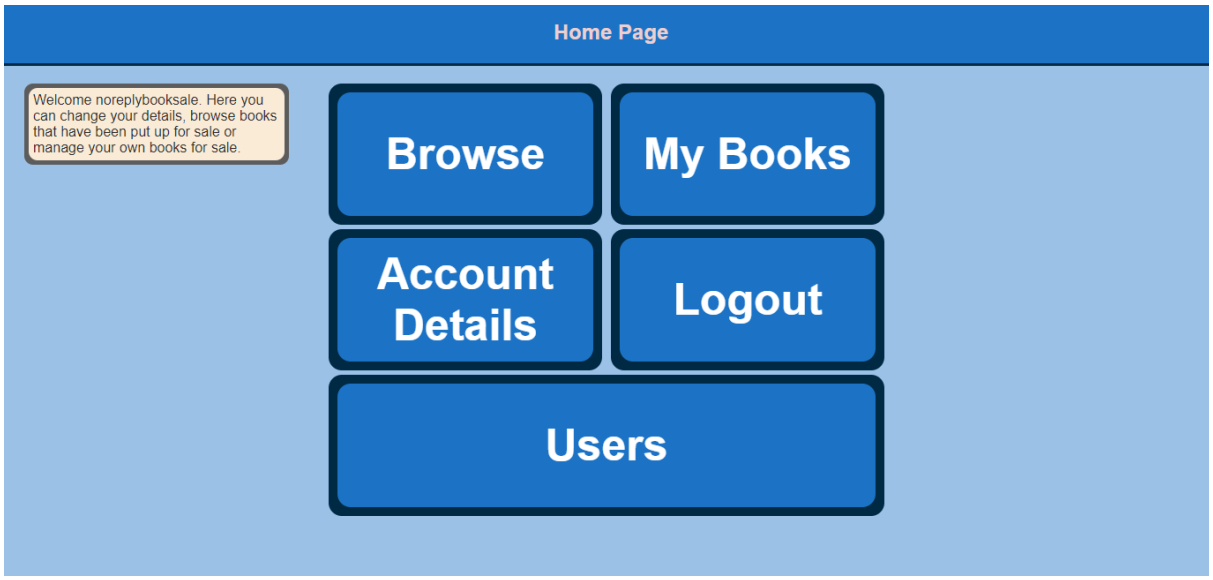
Expected output: buttons to browse.php, myBooks.php, accountDetails.php, logout.php and a greeting message on the left-hand side



Actual output: all elements display correctly

home.php for an admin/super_admin user:

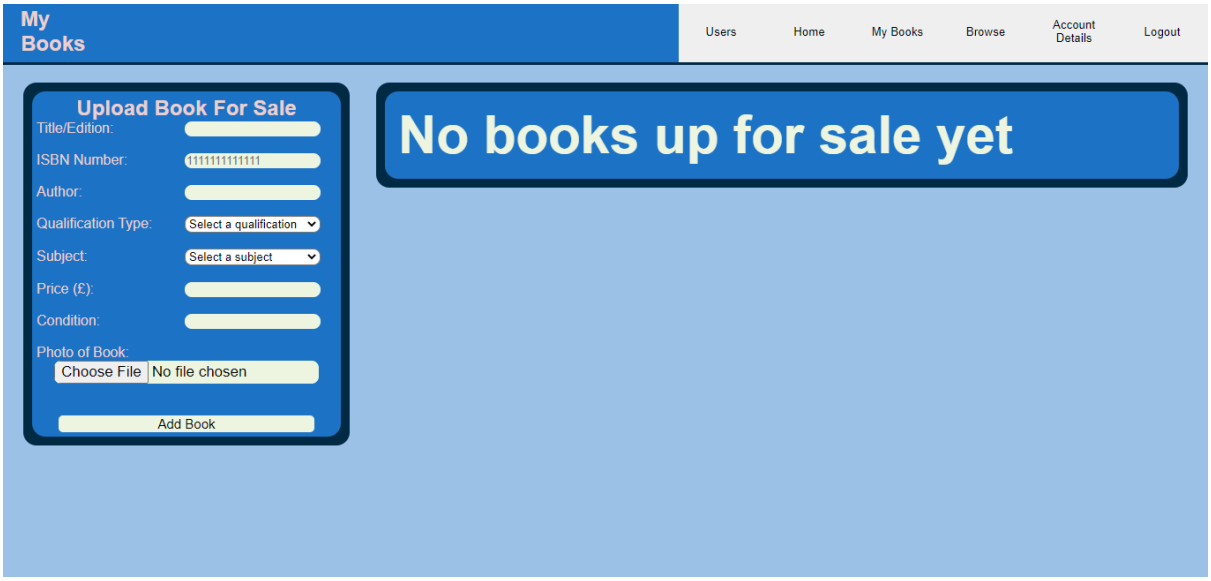
Expected output: same as above but there's also a button to users.php



Actual output: all elements display correctly

myBooks.php for someone without any books for sale

Expected output: upload books form on left and empty books table on right with message saying there are no books up for sale yet **(with navbar displaying at top right – same for all pages)**



Actual output: all elements display correctly

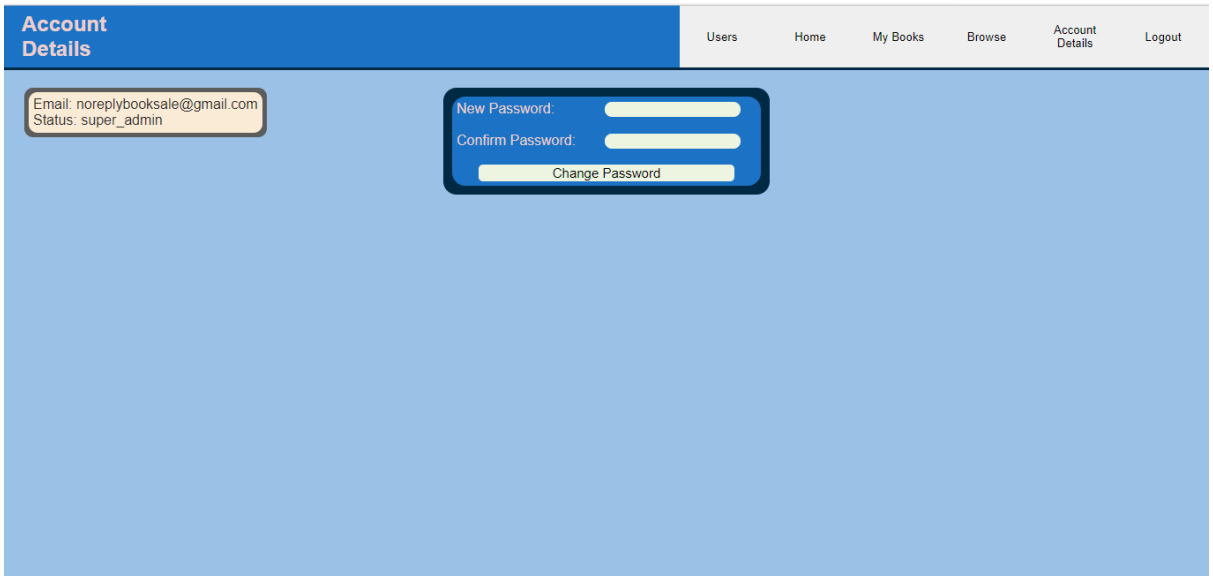
myBooks.php for someone with a few books for sale

Expected output: same as above, but book table contains rows with book details and is vertically scrollable

Actual output: all elements display correctly

accountDetails.php

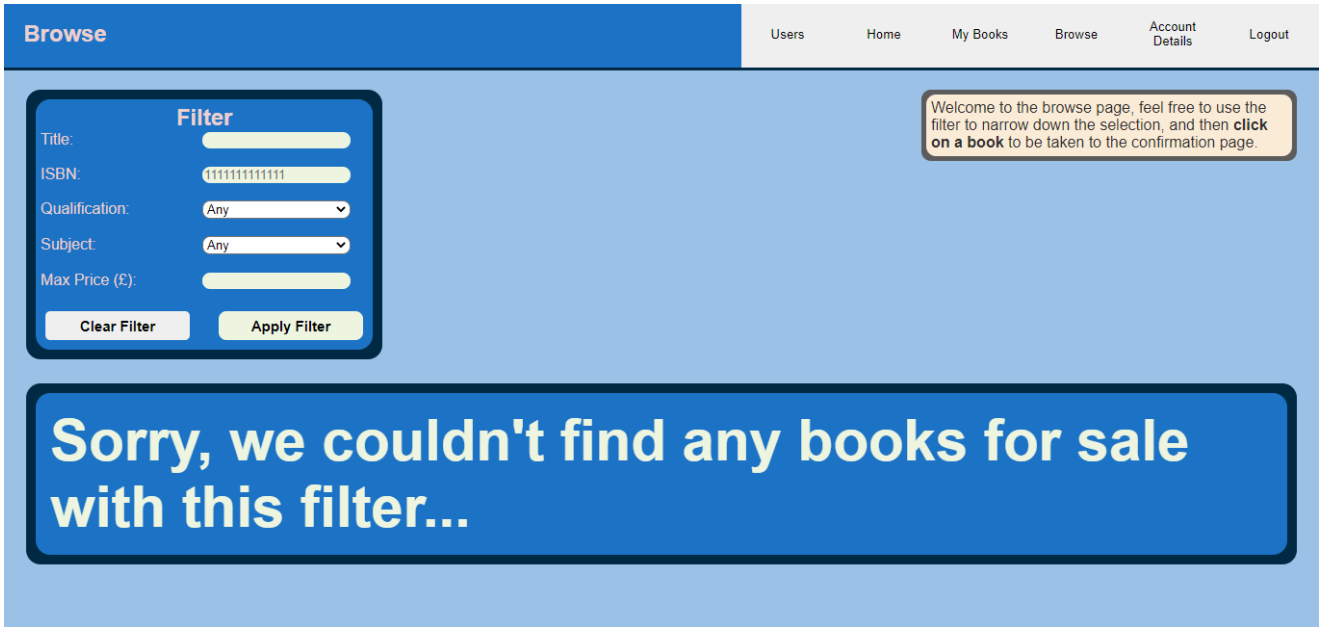
Expected output: info bubble on left side of page and form in middle of page



Actual output: all elements are displayed correctly

browse.php for when there are no books for sale (or none that match the current filter)

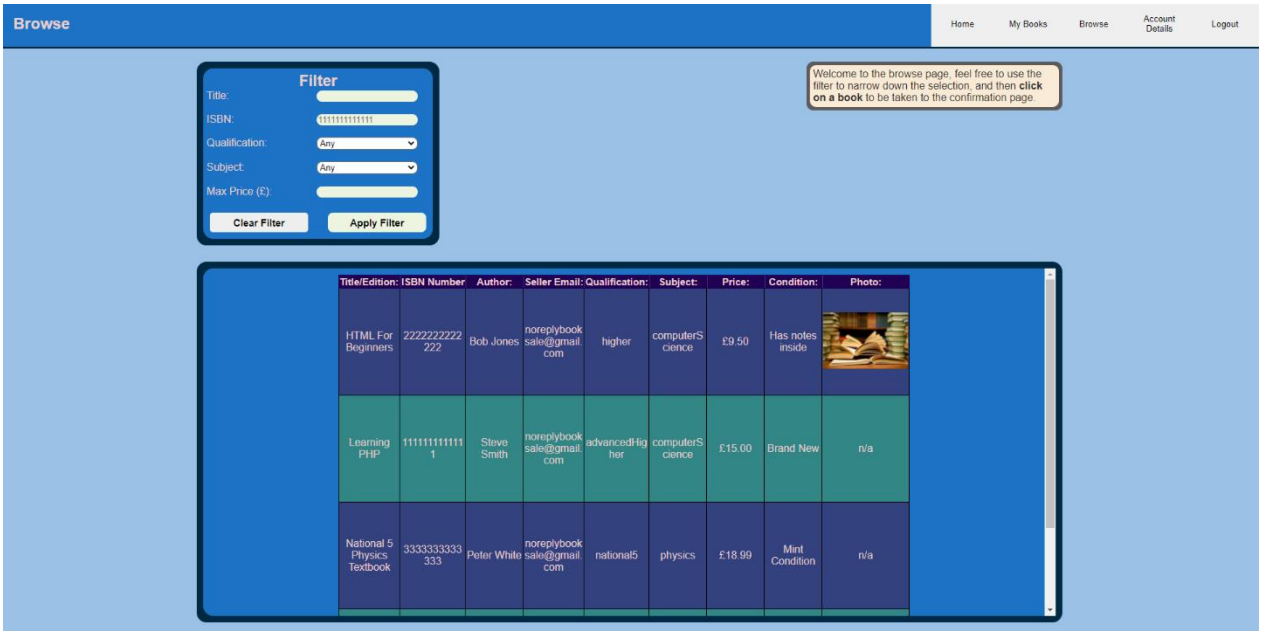
Expected output: filter form on left, info bubble on right, and then empty book filter below saying that there are no books



Actual output: all elements displayed correctly

books.php for standard user where there are books for sale that match current filter

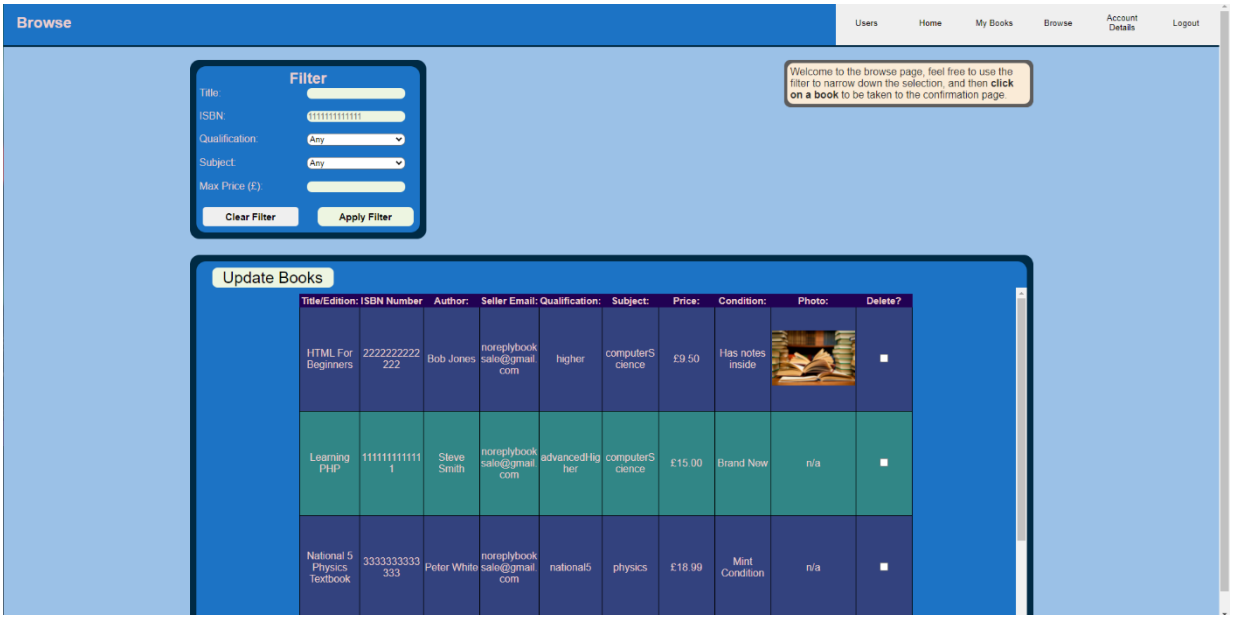
Expected output: same as above except books table has book entries



Actual output: all elements display correctly

books.php for an admin or super_admin where there are books for sale that match current filter

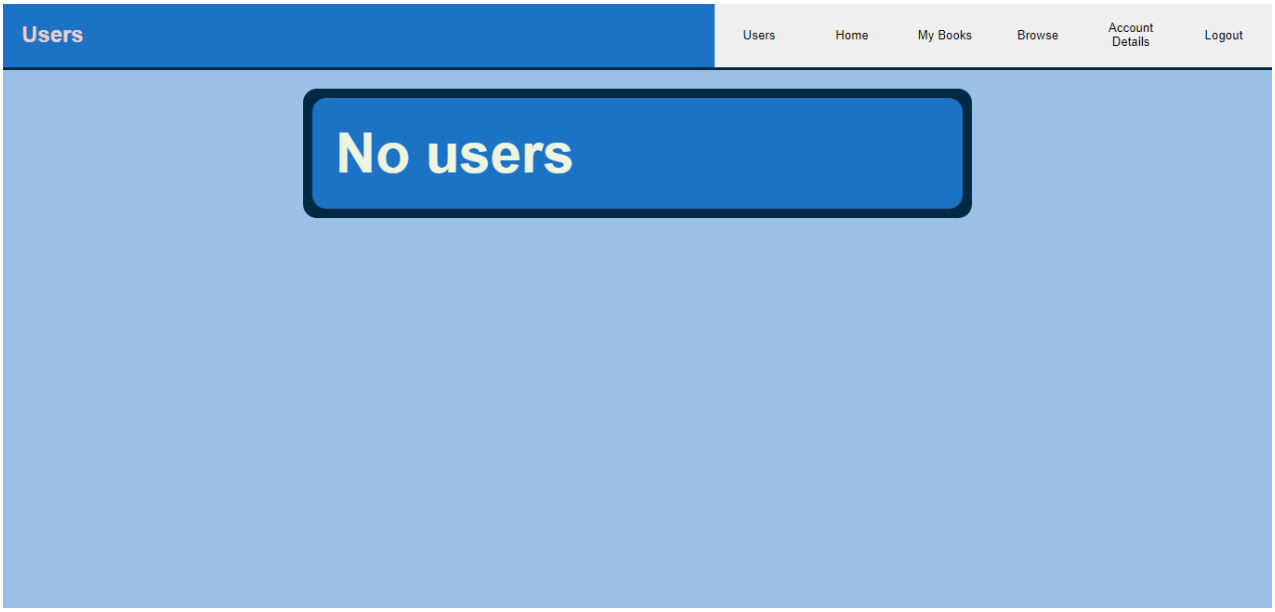
Expected output: same as above except there's also the update books button and the delete column



Actual output: all elements display correctly

users.php for an admin where there are no users

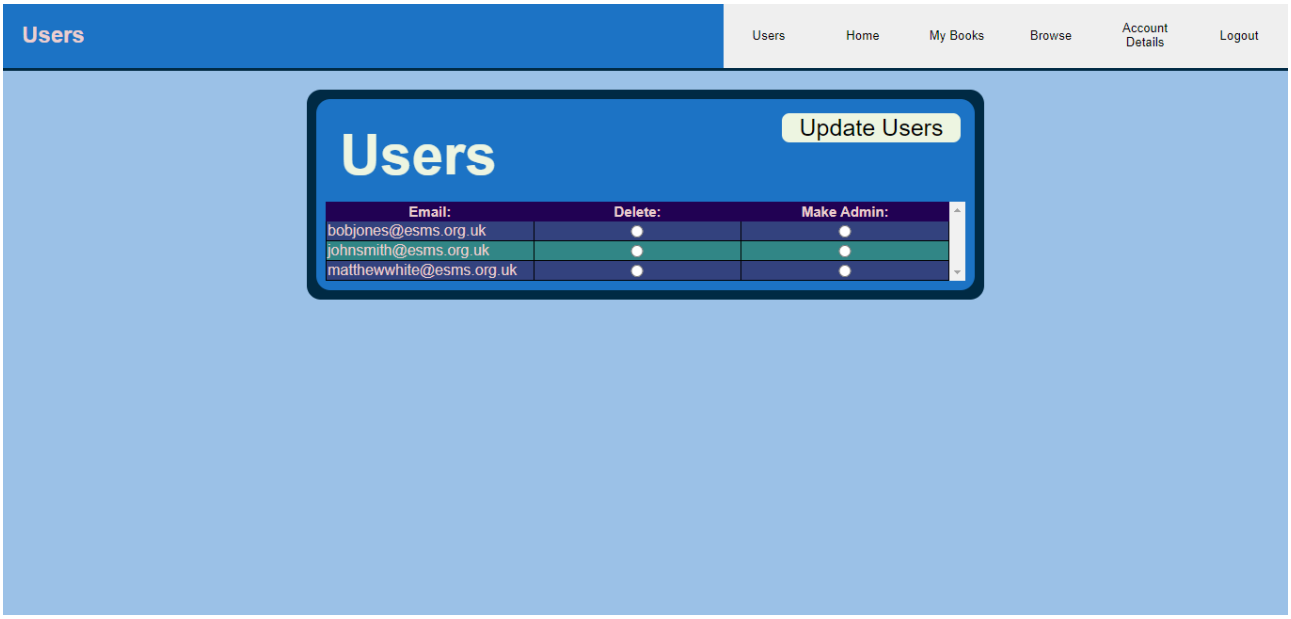
Expected output: users table is empty with a message saying so



Actual output: all elements display correctly

users.php for an admin when there are users

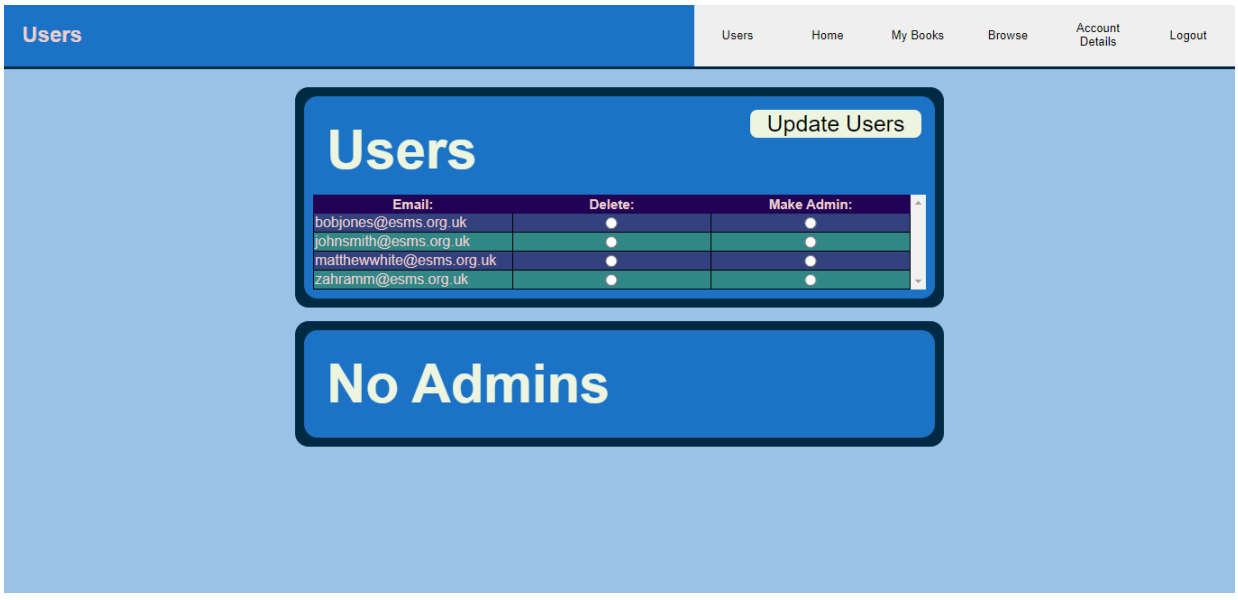
Expected output: same as above except table is populated with users, with delete and promote columns and an update users button



Actual output: all elements display correctly

users.php for a super_admin when there are no admins

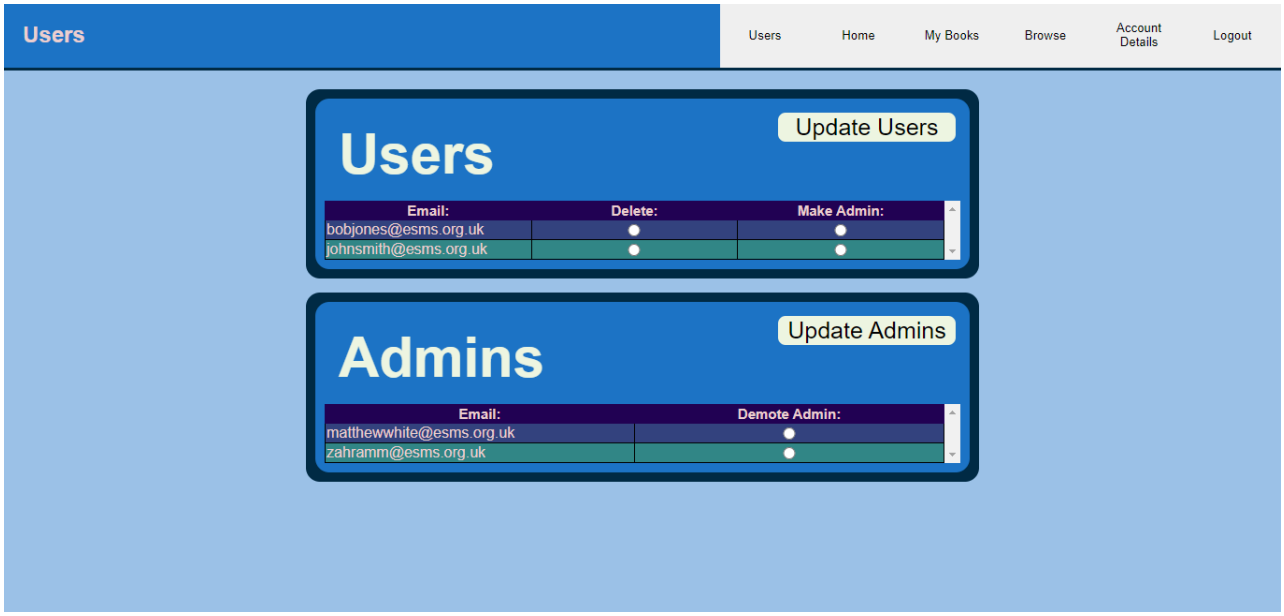
Expected output: same as above but now there's also an empty admins table with a message saying this



Actual output: all elements display correctly

users.php for a super_admin when there are admins

Expected output: same as above except now there are entries in the admins table, with the demote column and the update admins button



Actual output: all elements display correctly

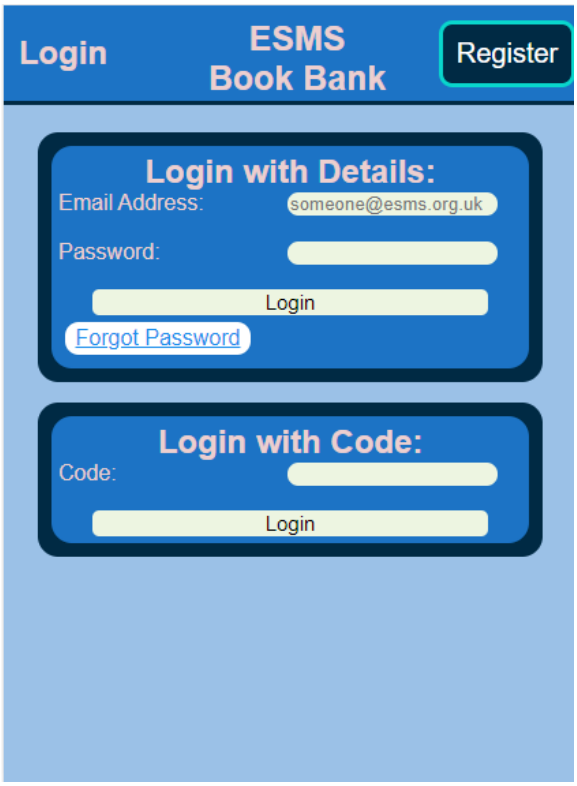
Conclusion: all content for all pages displays correctly.

Media Queries

Test: Page layout (CSS media queries) for mobile devices (screen-width <= 420px) for each page

login.php

Expected output: Title text shrinks to fit (rest remains same)

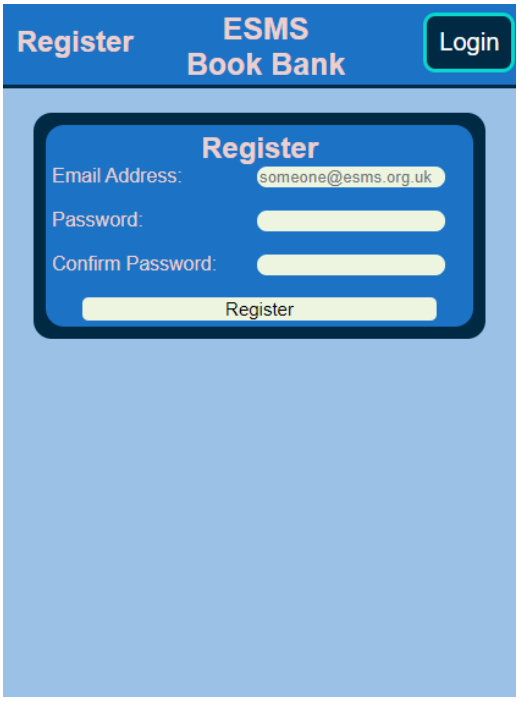


Actual output: all elements display correctly

register.php

Expected output: same as above

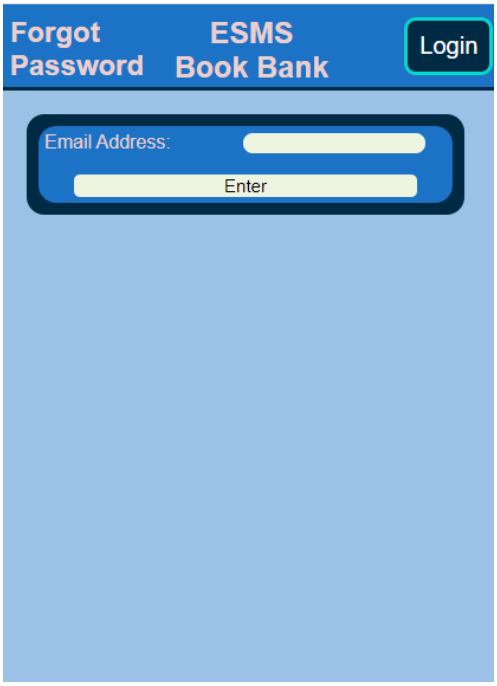
Actual output: all elements display correctly



forgotPass.php

Expected output: same as above

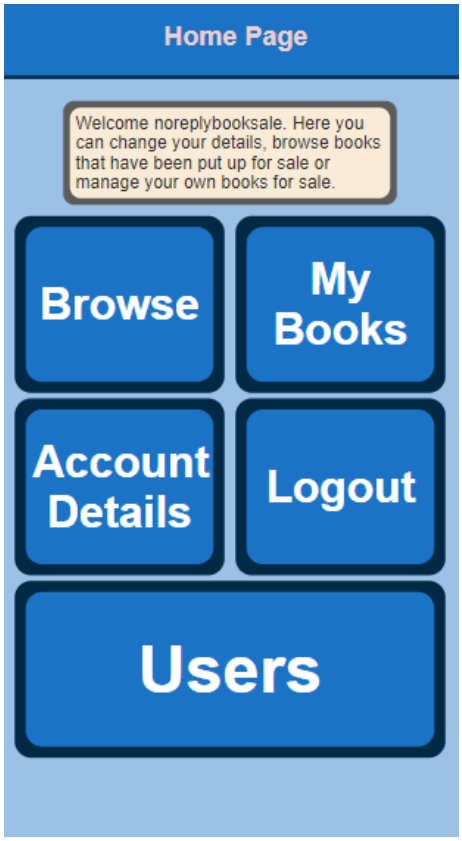
Actual output: all elements display correctly



home.php (for admin/super_admin – standard user is the same except users button is omitted)

Expected output: info bubble/greeting message gets centred, and buttons get shrunk to fit

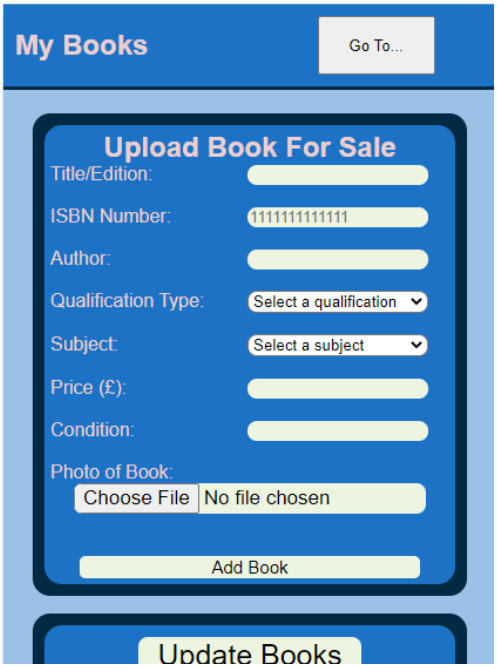
Actual output: all elements displayed correctly

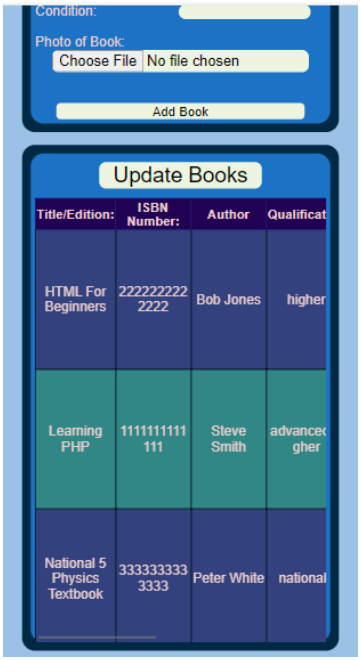


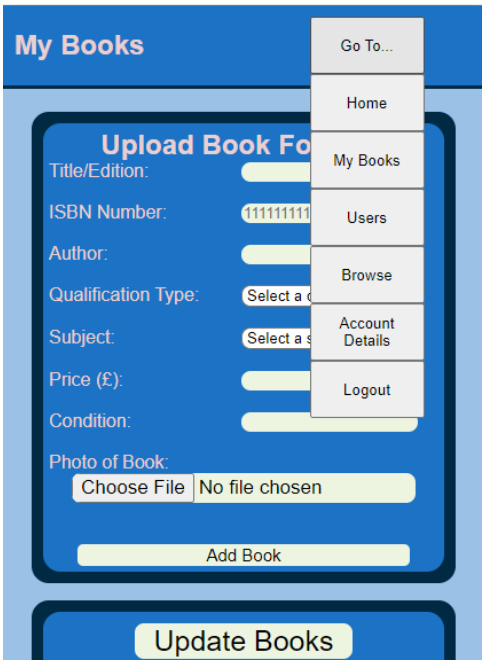
myBooks.php

Expected output: Upload form and table are both centred and displayed on top of each other – books table is now also horizontally scrollable (also – for all pages from now on, navbar becomes a clickable drop-down menu – shown in 3rd image)

Actual output: all elements display correctly



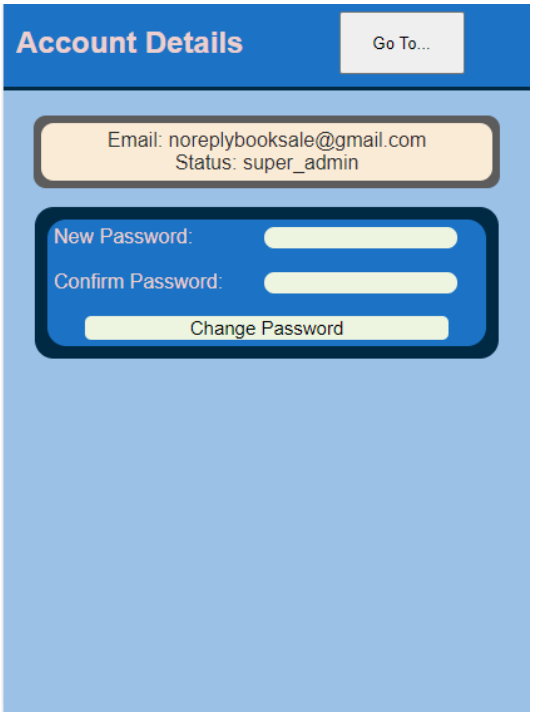




accountDetails.php

Expected output: info bubble is centred in middle of page and form appears just below it (also centred)

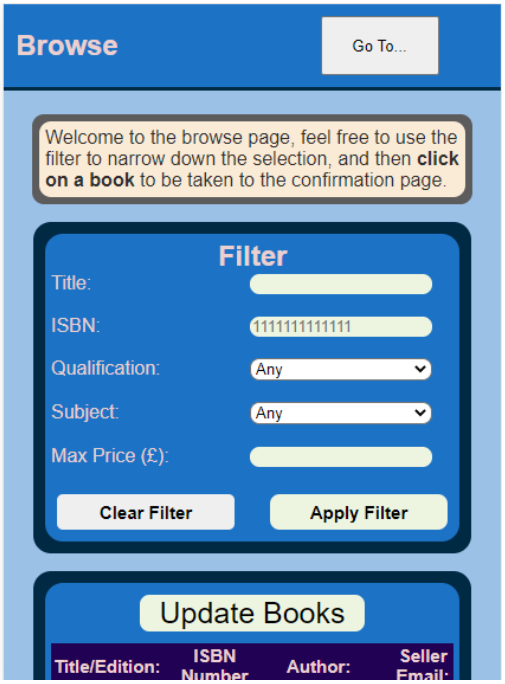
Actual output: all elements displayed correctly

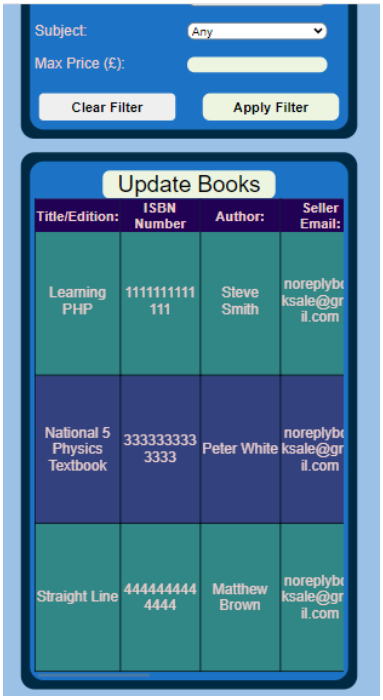


browse.php

Expected output: info bubble, filter form and books table are all centred and displayed on top of one another. Books table is now also horizontally scrollable (update books button appears because it is viewed from an admin account).

Actual output: all elements display correctly

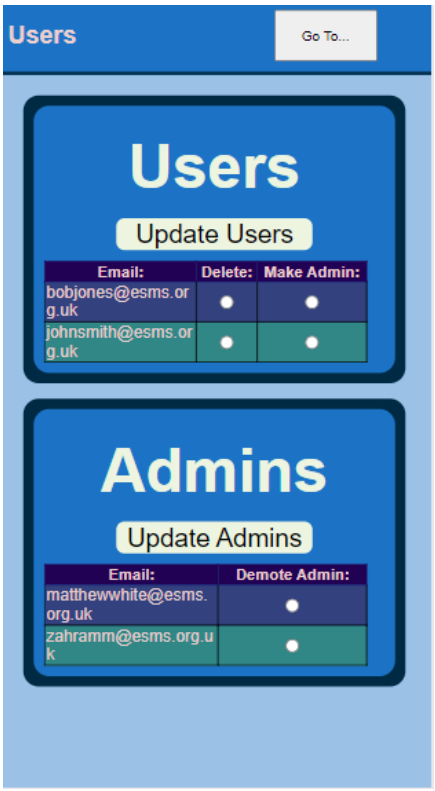




users.php (viewed from a super_admin account)

Expected output: users and admins table remain centred and on top of one another, but are slightly shrunk to fit

Actual output: all elements display correctly

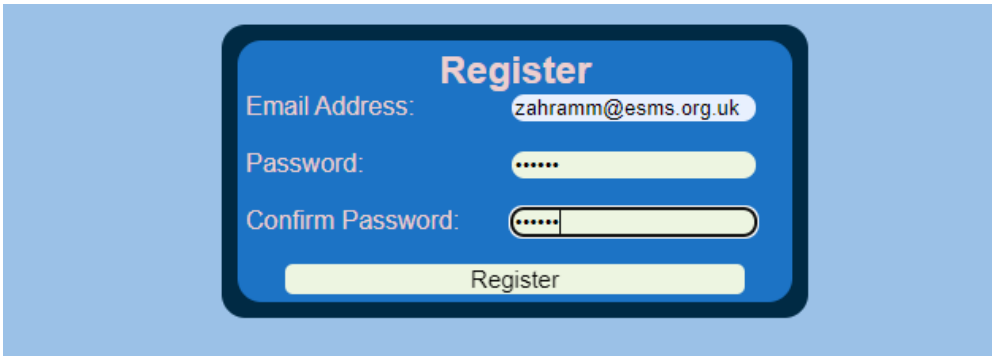


Conclusion: CSS media queries successfully adjust placement/sizing of page content so that it is easily viewable from a mobile device.

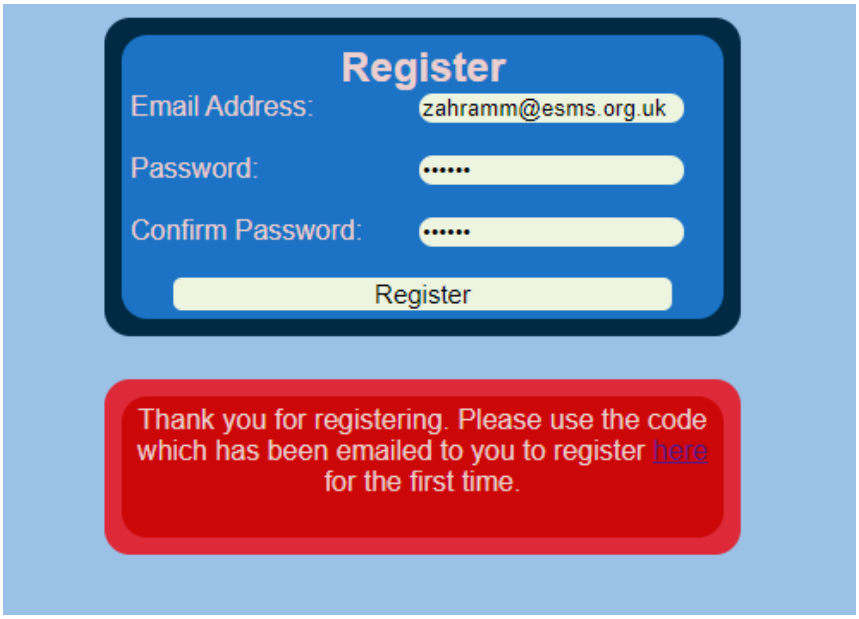
Test: Requirement 1

Actual output:

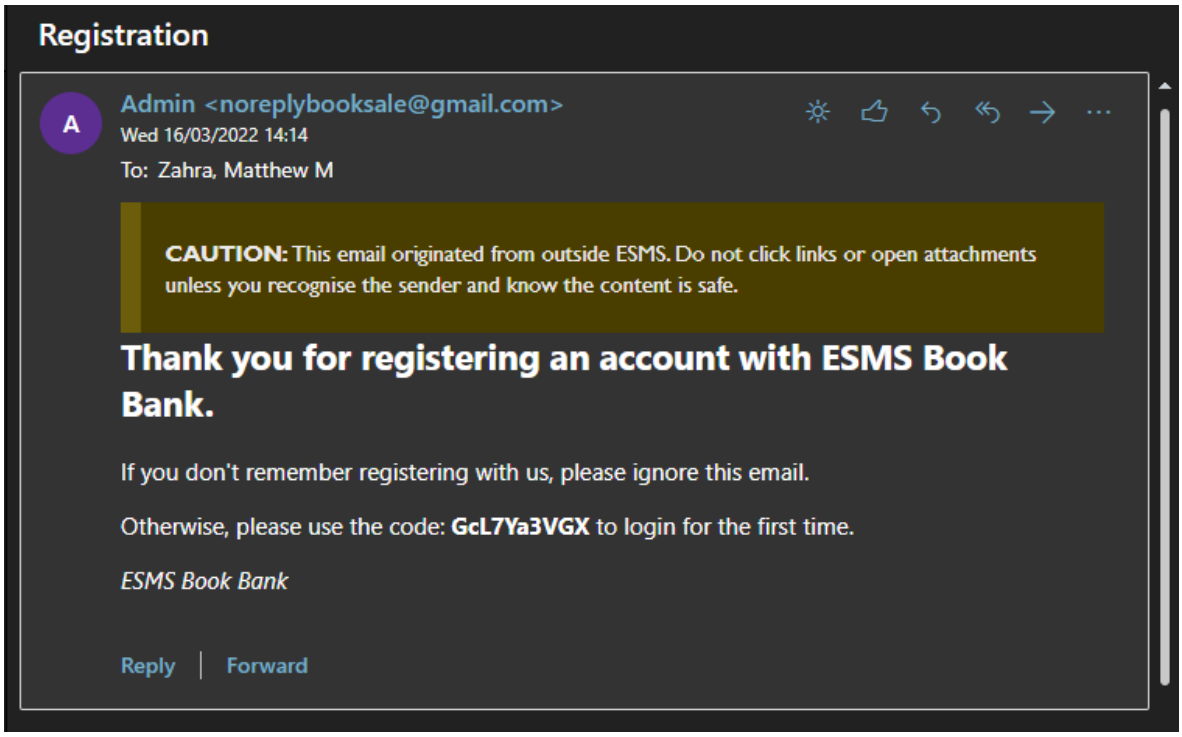
User registering with a valid email:



Confirmation message:



Email with confirmation code in inbox of registering user:

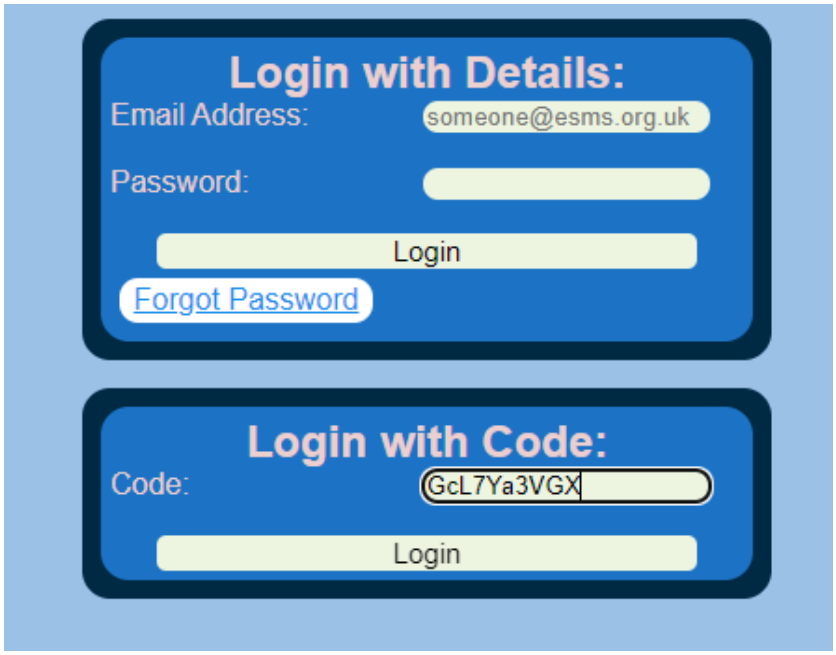


Conclusion: requirement 1 is satisifed as a user can register with a valid password and a valid email '…@esms.org.uk'

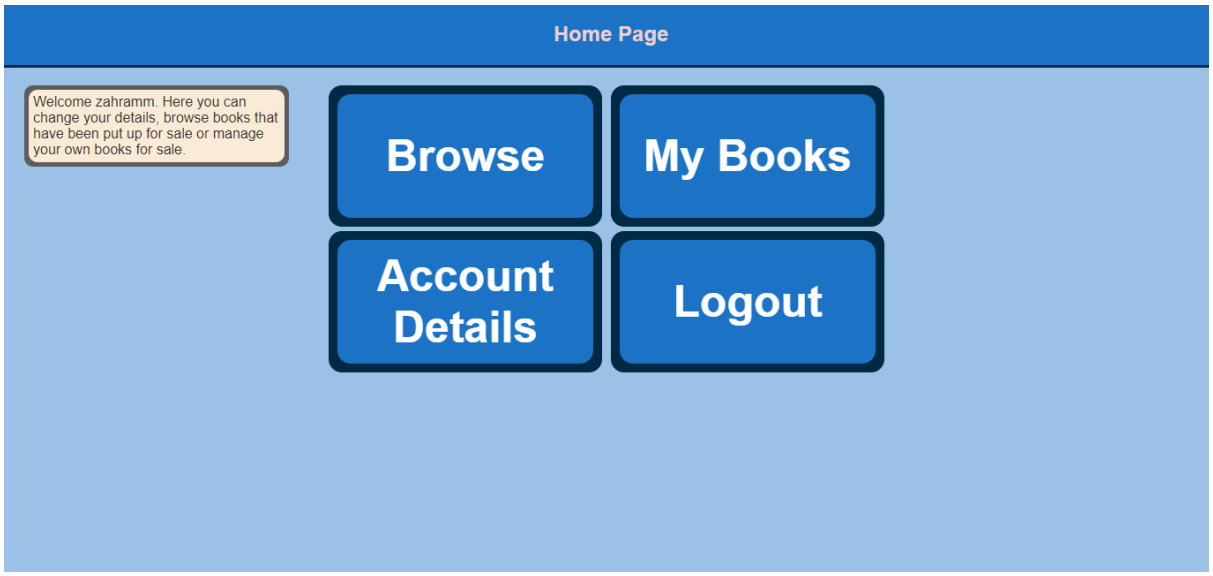
Test: requirement 2

Actual output:

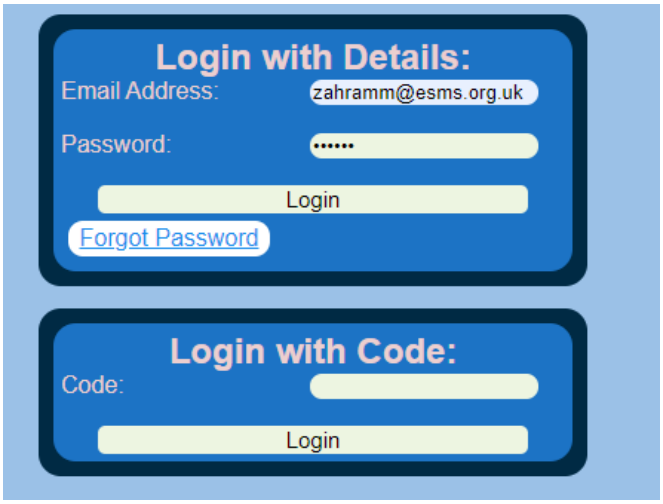
User logging in with verification code that was sent to email address:



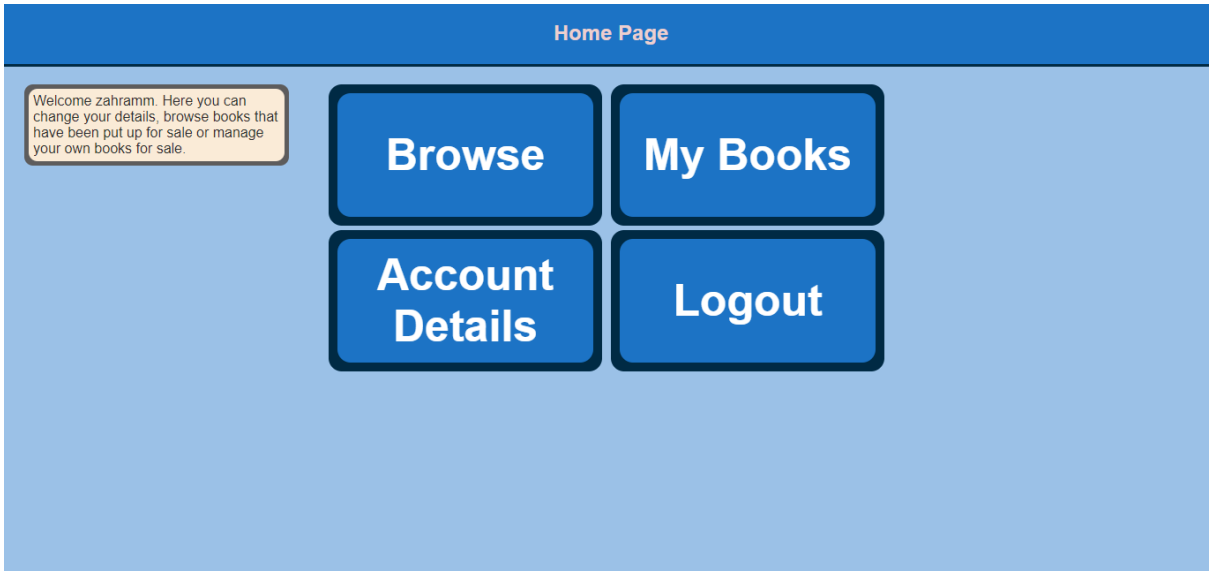
Redirects to home.php:



User logging in with email and password:



Redirects to home.php:

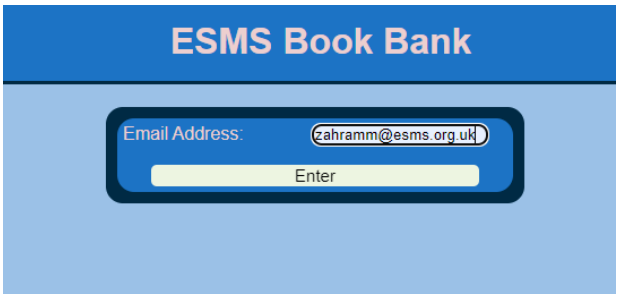


Conclusion: since the user is able to login with both the verification code they got emailed and their email and password, therefore requirement has been satisfied.

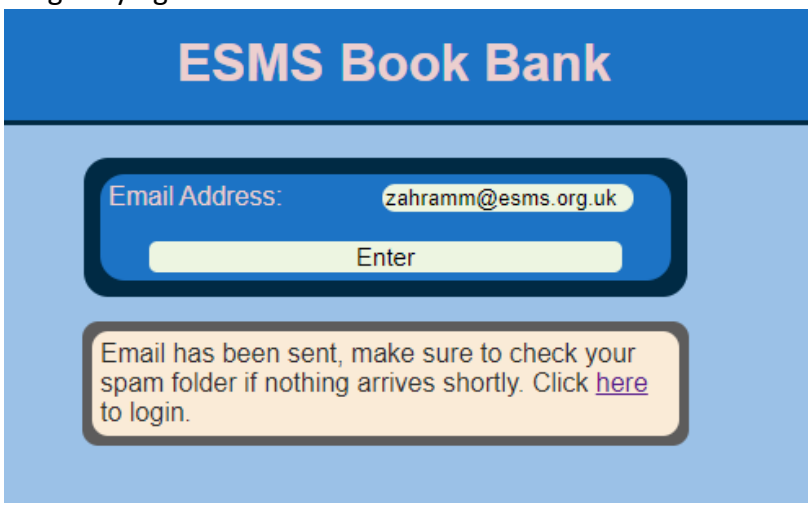
Test: requirement 3

Actual output:

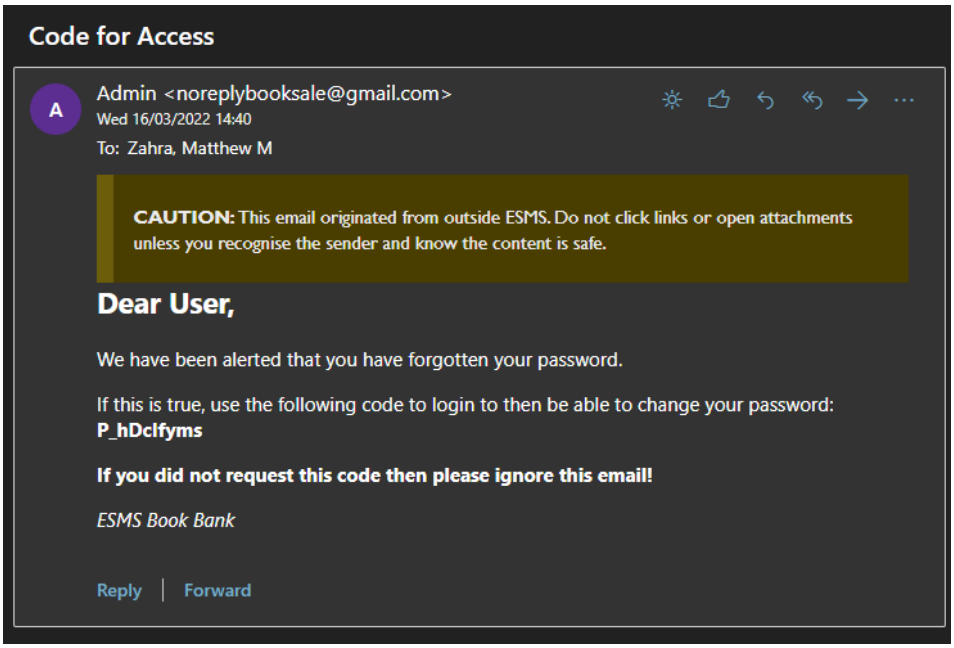
User requesting a new code by entering their email on forgotPass.php:



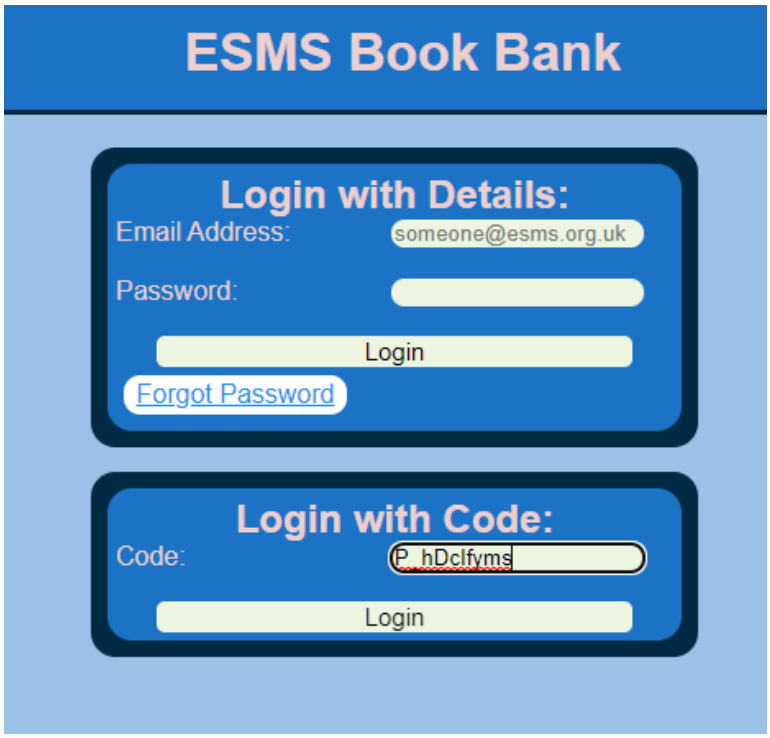
Prompt message saying code has been sent:



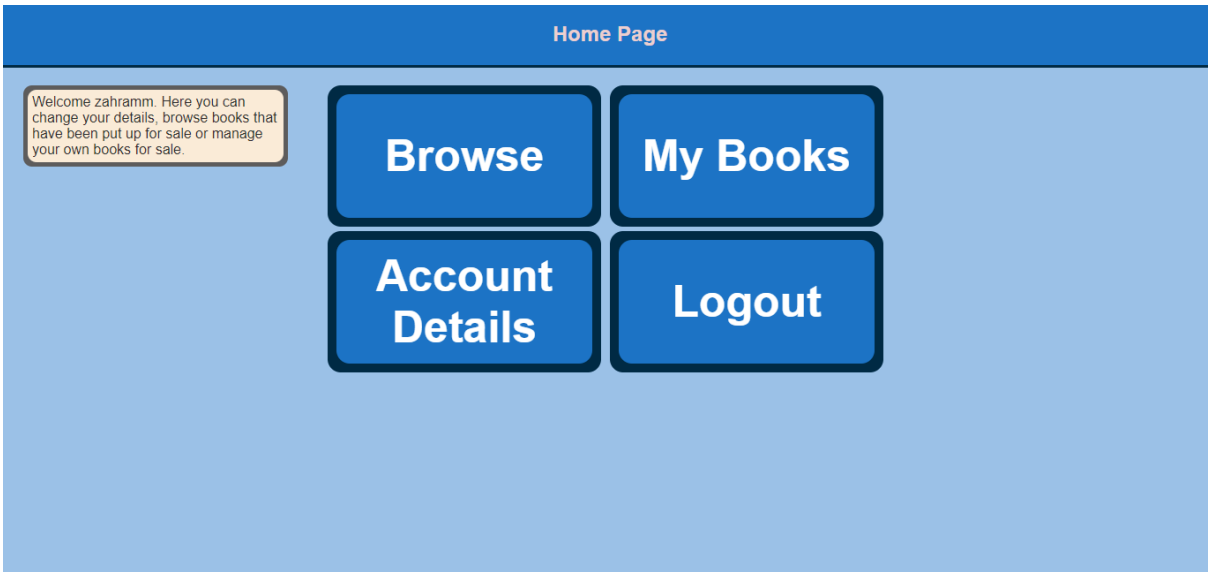
Code sent to user's email upon request:



User logging in with code:



This redirects them to home.php:

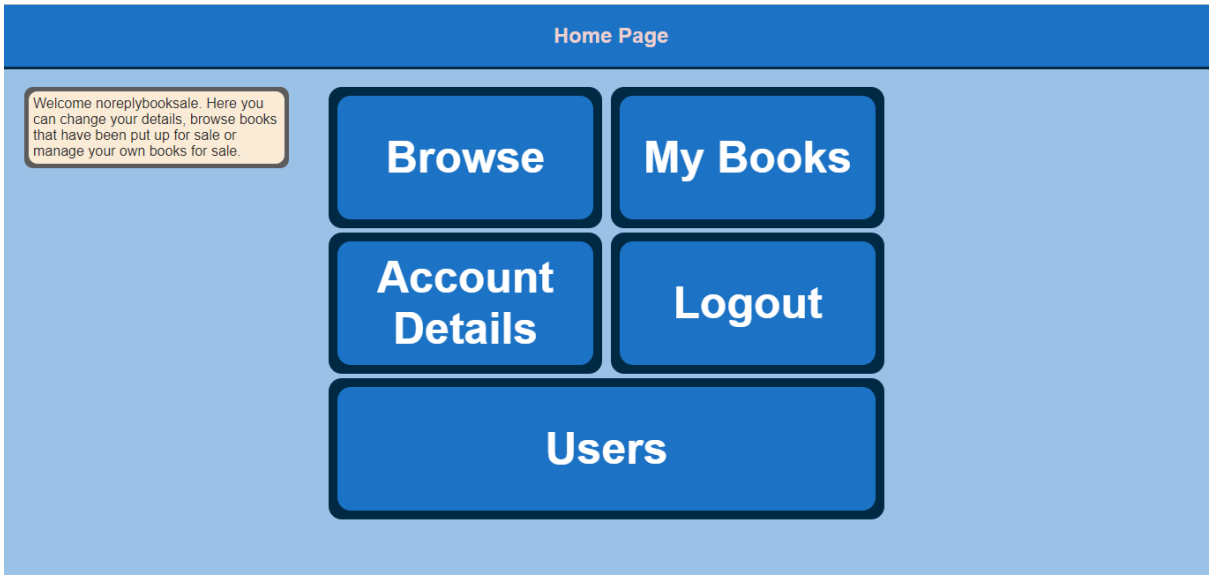


Conclusion: Requirement 3 is satisfied as a user can request a verification code from forgotPass.php and login with it

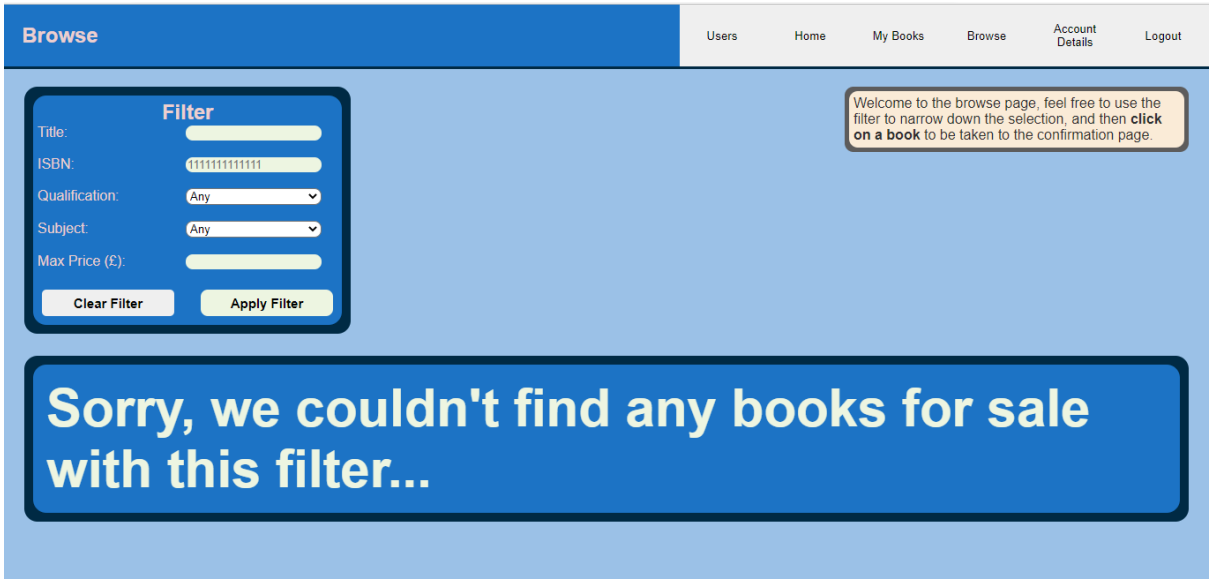
Test: Requirement 4

Actual output:

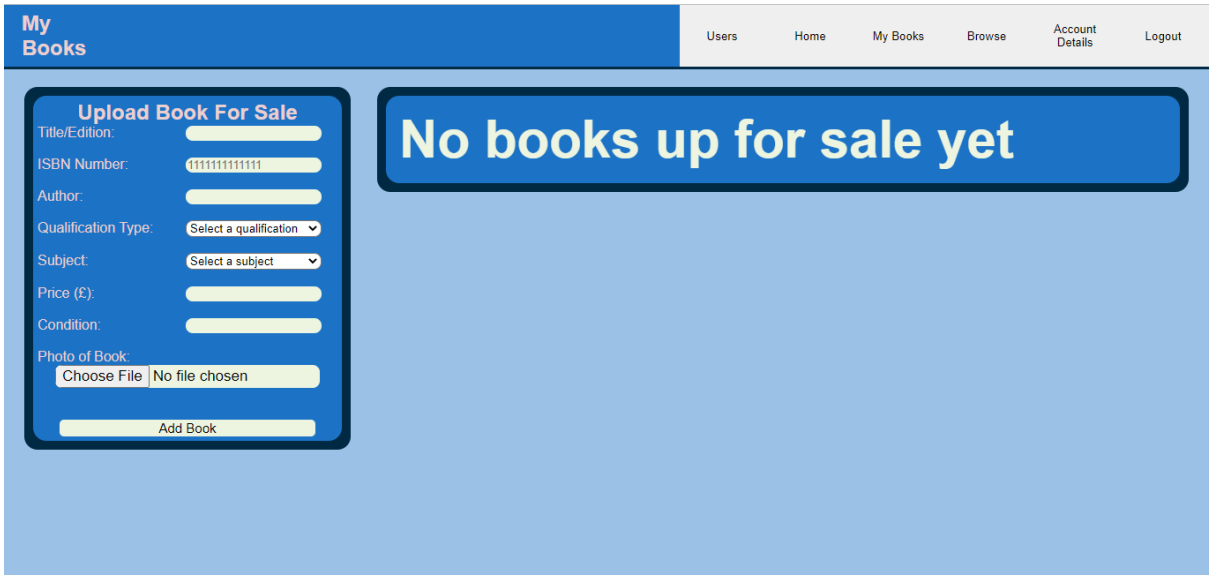
From home.php (from an admin account)



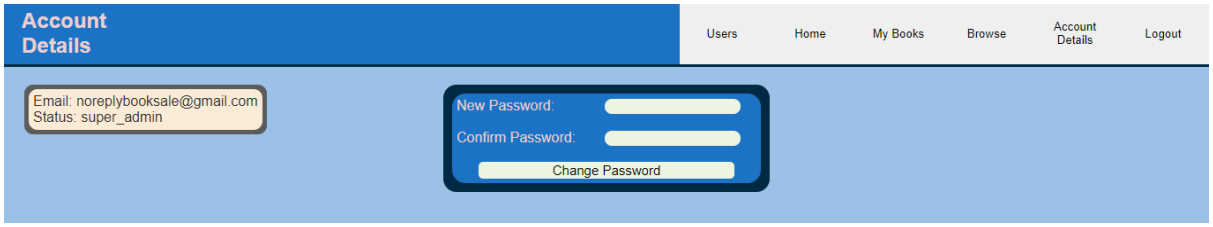
Clicking browse button:



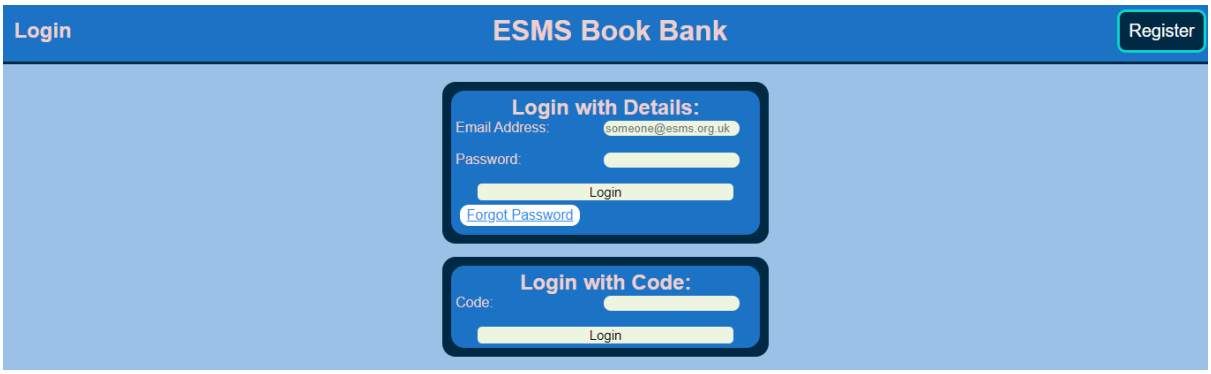
Clicking My Books button:



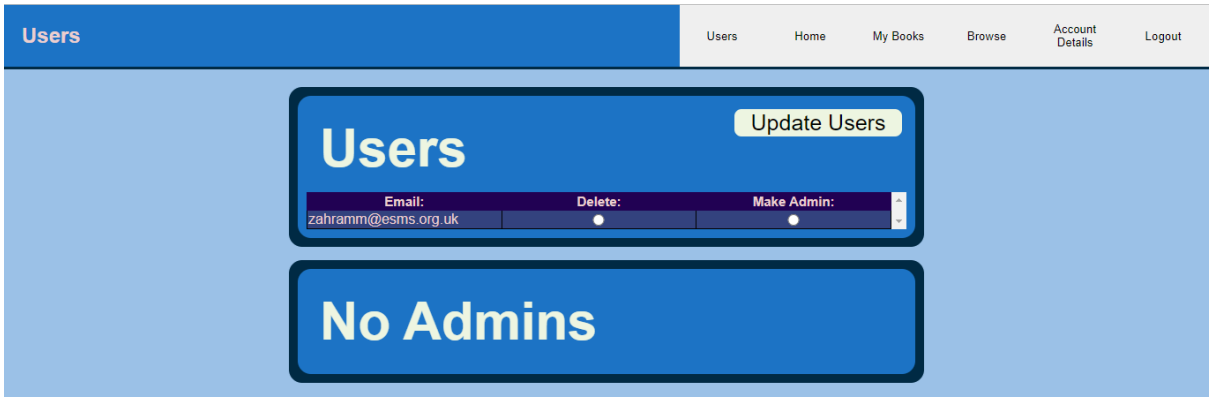
Clicking Account Details Button:



Clicking logout button:



Clicking users button:



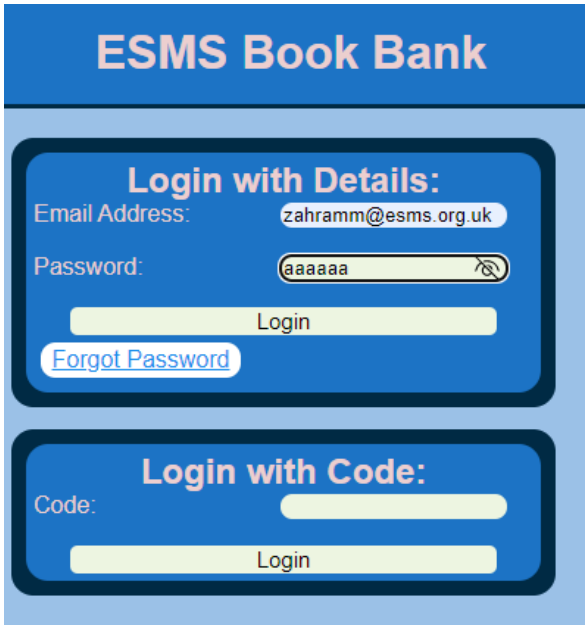
From navbar, clicking the home button takes us back to home.php, and clicking the other buttons on each navbar yield the exact same result as clicking the buttons on home.php (so they haven't been shown here).

Conclusion: Requirement 4 is met as a user can navigate to any page using the buttons on home.php or any of the navbars.

Test: requirement 5

Actual output:

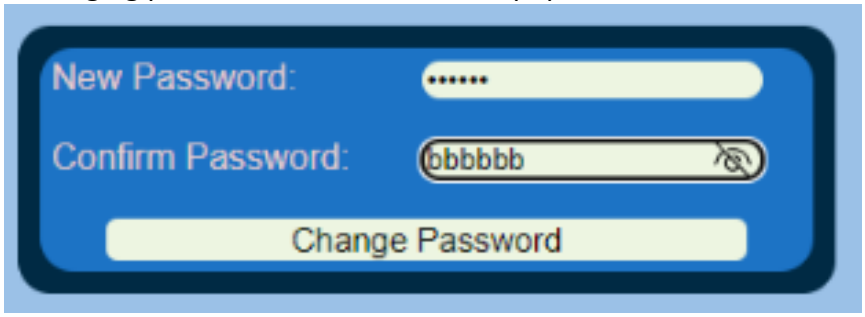
User logging in with original password:

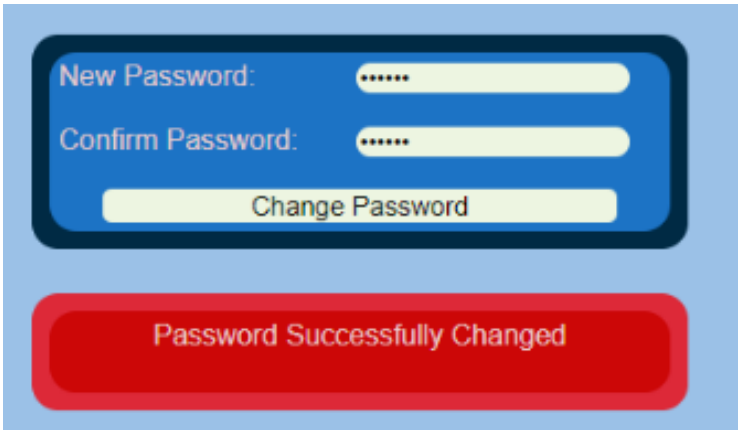


Database showing original (hashed) password:

```
mysql> SELECT email, password FROM users WHERE email = 'zahramm@esms.org.uk';
+----------------------+------------------------------------------------------------------+
| email                | password                                                         |
+----------------------+------------------------------------------------------------------+
| zahramm@esms.org.uk | ed02457b5c41d964dbd2f2a609d63fe1bb7528dbe55e1abf5b52c249cd735797 |
+----------------------+------------------------------------------------------------------+
```

User then changing password at accountDetails.php:

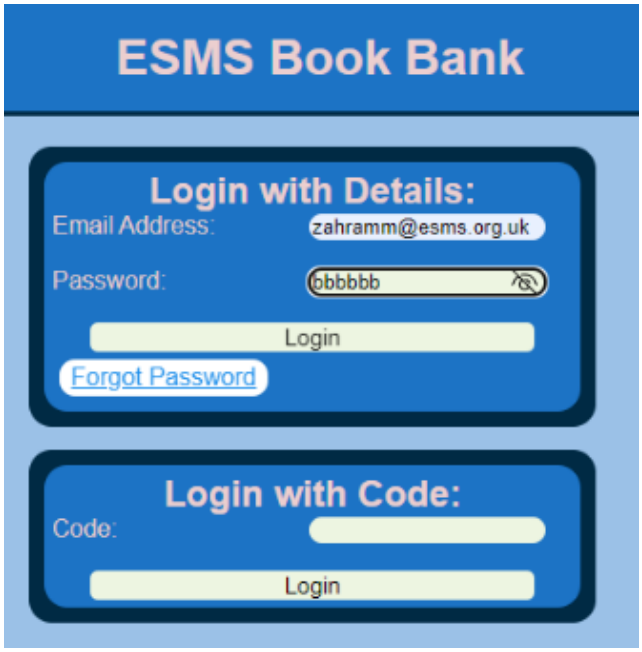




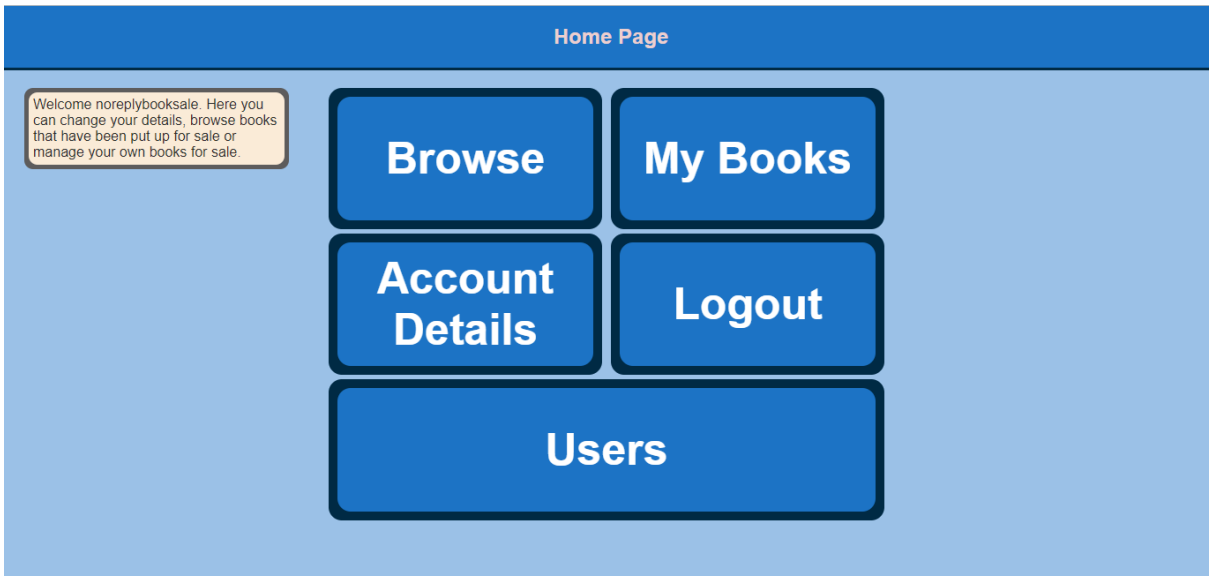
Updated password (hashed) in database:

```
mysql> SELECT email, password FROM users WHERE email = 'zahramm@esms.org.uk';
+---------------------+------------------------------------------------------------------+
| email               | password                                                         |
+---------------------+------------------------------------------------------------------+
| zahramm@esms.org.uk | 4625fd63b0e96fc0d656ae7381605e48d4a0f63a319fc743adf22688613883c7 |
+---------------------+------------------------------------------------------------------+
```

User logging in with new password:



User is then redirected to home.php:

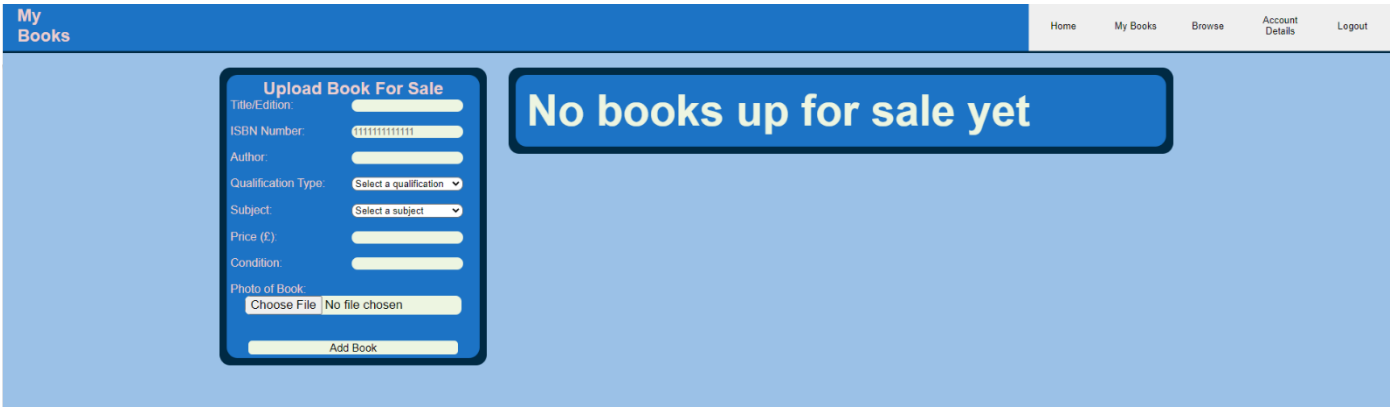


Conclusion: Requirement 5 is satsified as a user can change their password, and then login with new password as database gets updated with new (hashed) password

Test: requirement 6

Actual output:

myBooks.php before adding any books:



Form with book details in it:

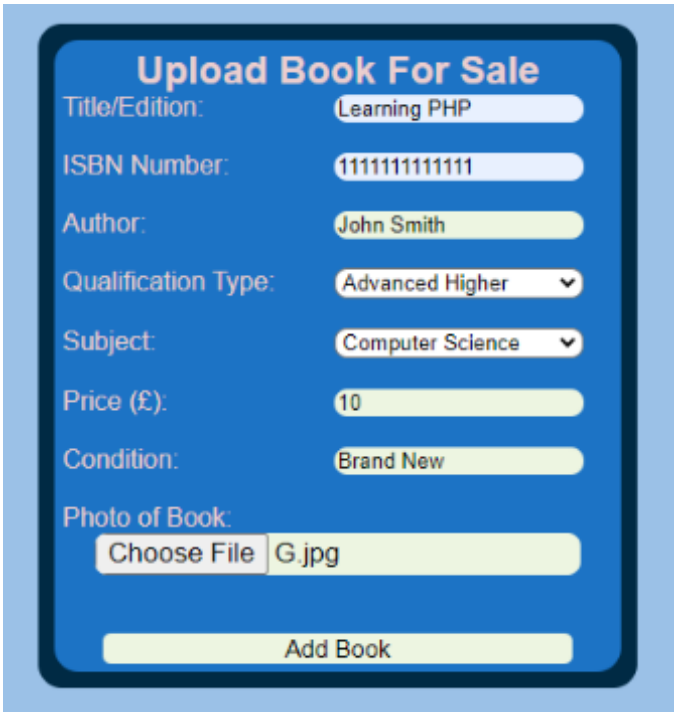
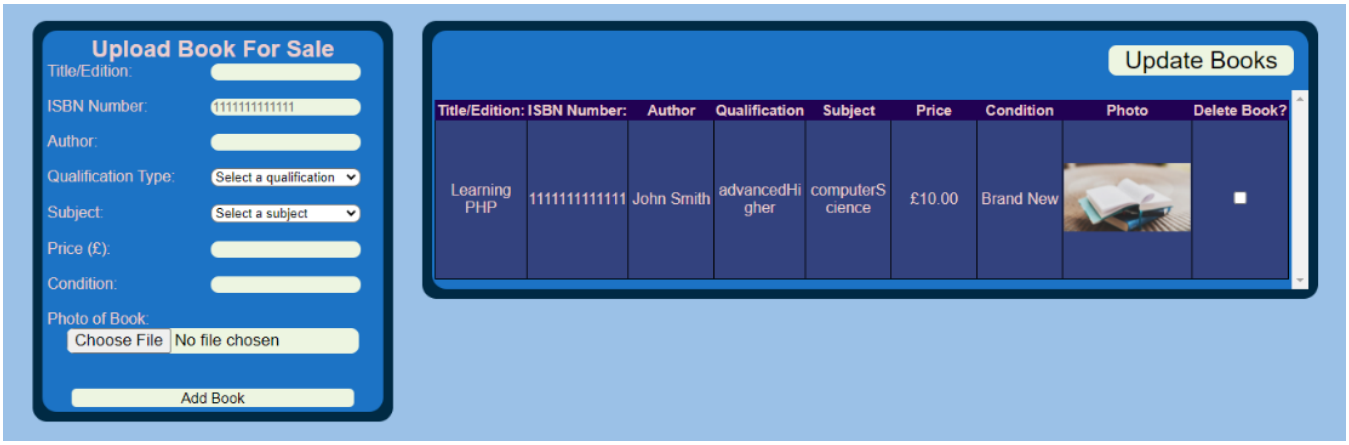
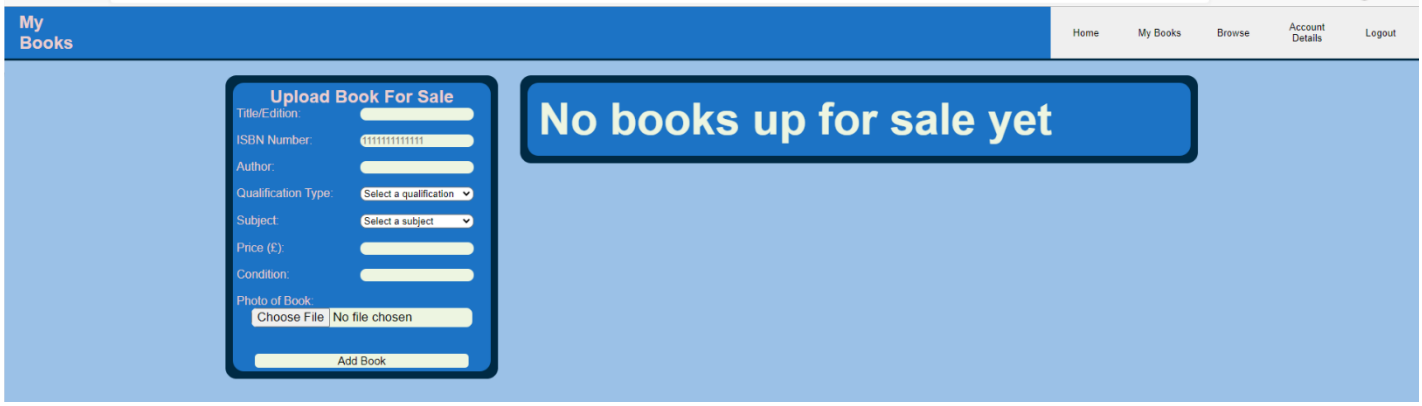


Table once form has been submitted (with book entry in it):



Delete Book checkbox has been ticked:

Table once form has been submitted (book gets deleted):



Conclusion: requirement is satisified as a user can upload a book for sale, view their books that they have put up for sale, and then delete a book they have put up for sale

Test: requirement 7

Actual output:

User (zahramm@esms.org.uk) viewing books table browse.php page without a filter being

Title/Edition:	ISBN Number	Author:	Seller Email:	Qualification:	Subject:	Price:	Condition:	Photo:
French Dictionary	3333333333333	Jeff White	noreplybooksale@gmail.com	higher	french	£8.99	Slightly Used	n/a
Learning PHP	1111111111111	John Smith	noreplybooksale@gmail.com	advancedHigher	computerScience	£10.00	Brand New	
Physics Study Book	2222222222222	Bob Jones	noreplybooksale@gmail.com	higher	physics	£7.50	Used	

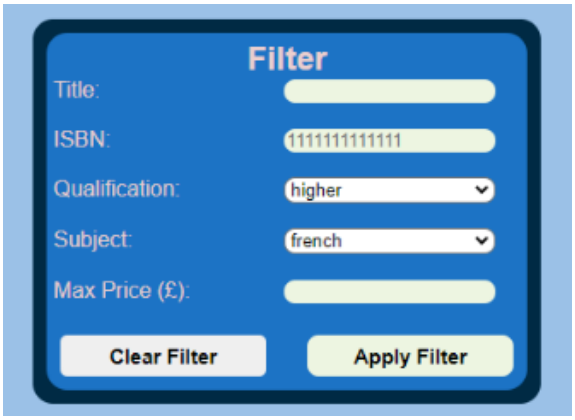
applied:

All books available for sale in database that are up for sale for users other than logged in user (zahramm@esms.org.uk):

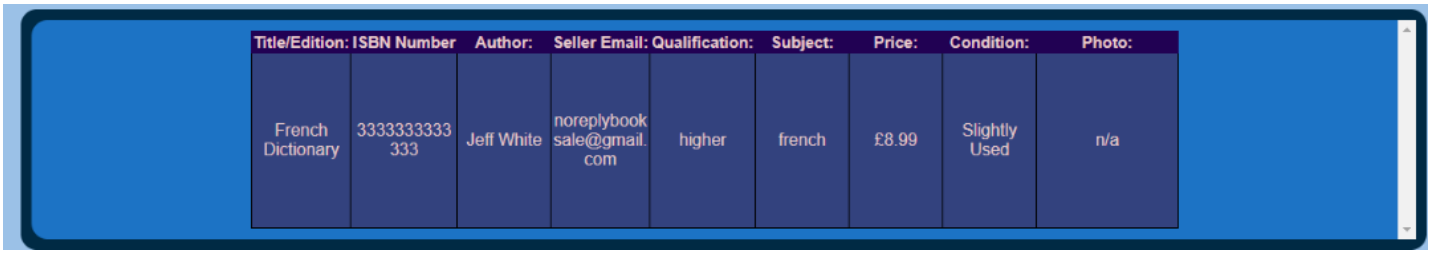
```
mysql> SELECT title, ISBN, author, email, qualification, subject, price, bookCondition, photo FROM books, users WHERE users.userID = books.userID and NOT email = 'zahramm@esms.org.uk';
+--------------------+---------------+-----------+--------------------------+---------------+----------------+-------+---------------+--------------------------+
| title              | ISBN          | author    | email                    | qualification | subject        | price | bookCondition | photo                    |
+--------------------+---------------+-----------+--------------------------+---------------+----------------+-------+---------------+--------------------------+
| Learning PHP       | 1111111111111 | John Smith | noreplybooksale@gmail.com | advancedHigher | computerScience | 10.00 | Brand New     | 1220322213842784378.jpg |
| Physics Study Book | 2222222222222 | Bob Jones  | noreplybooksale@gmail.com | higher         | physics        |  7.50 | Used          | 1220322214010696877.jpg |
| French Dictionary  | 3333333333333 | Jeff White | noreplybooksale@gmail.com | higher         | french         |  8.99 | Slightly Used | n/a                      |
+--------------------+---------------+-----------+--------------------------+---------------+----------------+-------+---------------+--------------------------+
```

These match the ones that zahramm@esms.org.uk can see in the books table

Apply a filter:



See books avaialble:



Title/Edition:	ISBN Number	Author:	Seller Email:	Qualification:	Subject:	Price:	Condition:	Photo:
French Dictionary	3333333333333	Jeff White	noreplybooksale@gmail.com	higher	french	£8.99	Slightly Used	n/a

See books in database that match the filter:

```
mysql> SELECT title, ISBN, author, email, qualification, subject, price, bookCondition, photo
    -> FROM books, users WHERE users.userID = books.userID and NOT email = 'zahramm@esms.org.uk'
    -> AND qualification = 'higher' AND subject = 'french';
+--------------------+---------------+------------+--------------------------+---------------+---------+-------+---------------+-------+
| title              | ISBN          | author     | email                    | qualification | subject | price | bookCondition | photo |
+--------------------+---------------+------------+--------------------------+---------------+---------+-------+---------------+-------+
| French Dictionary | 3333333333333 | Jeff White | noreplybooksale@gmail.com | higher        | french  |  8.99 | Slightly Used | n/a   |
+--------------------+---------------+------------+--------------------------+---------------+---------+-------+---------------+-------+
```

This entry matches the one seen in the table

Apply another filter:

See books in table:

See books in database that match filter:

```
mysql> SELECT title, ISBN, author, email, qualification, subject, price, bookCondition, photo
    -> FROM books, users WHERE users.userID = books.userID and NOT email = 'zahramm@esms.org.uk'
    -> AND price <= 9;
+--------------------+---------------+------------+--------------------------+---------------+---------+-------+---------------+-------------------------+
| title              | ISBN          | author     | email                    | qualification | subject | price | bookCondition | photo                   |
+--------------------+---------------+------------+--------------------------+---------------+---------+-------+---------------+-------------------------+
| Physics Study Book | 2222222222222 | Bob Jones  | noreplybooksale@gmail.com | higher        | physics |  7.50 | Used          | 1220322214010696877.jpg |
| French Dictionary  | 3333333333333 | Jeff White | noreplybooksale@gmail.com | higher        | french  |  8.99 | Slightly Used | n/a                     |
+--------------------+---------------+------------+--------------------------+---------------+---------+-------+---------------+-------------------------+
```

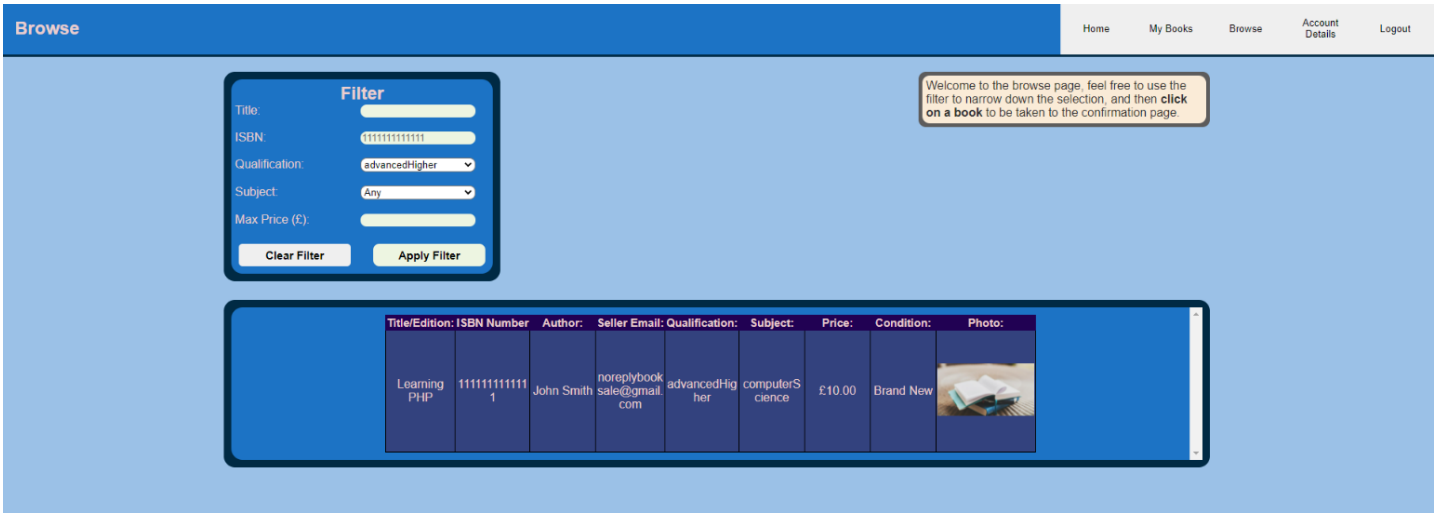
Books in database match those in table

Conclusion: requirement is satisified as a user can see each book up for sale (other than their own) and when they apply a filter, only books that match the filter get displayed in the books table.

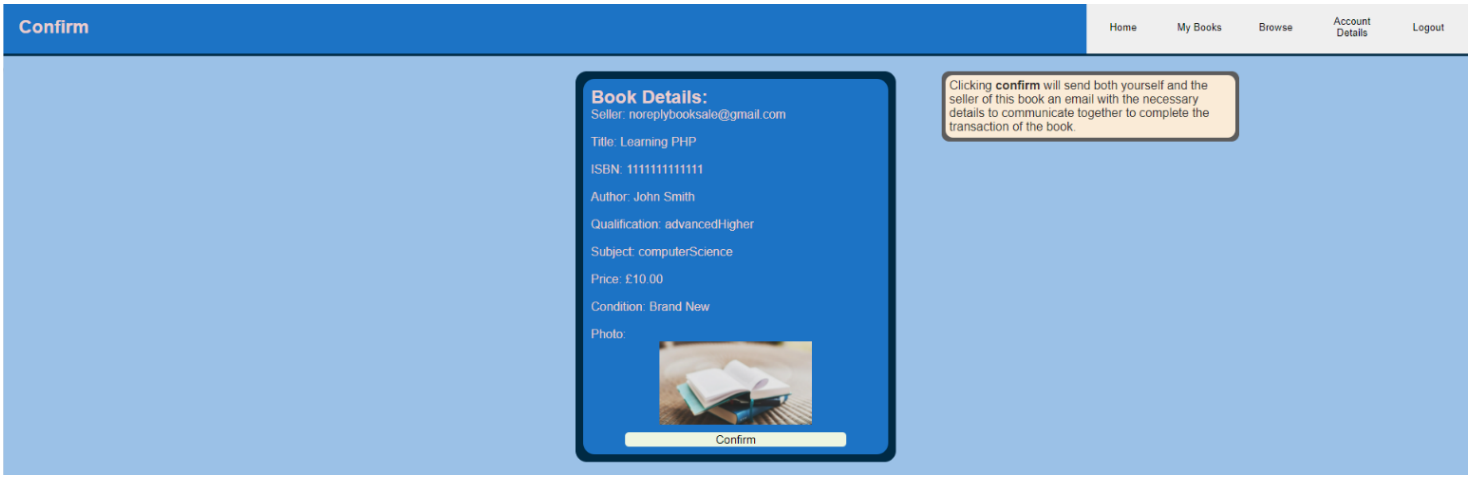
Test: requirement 8

Actual output:

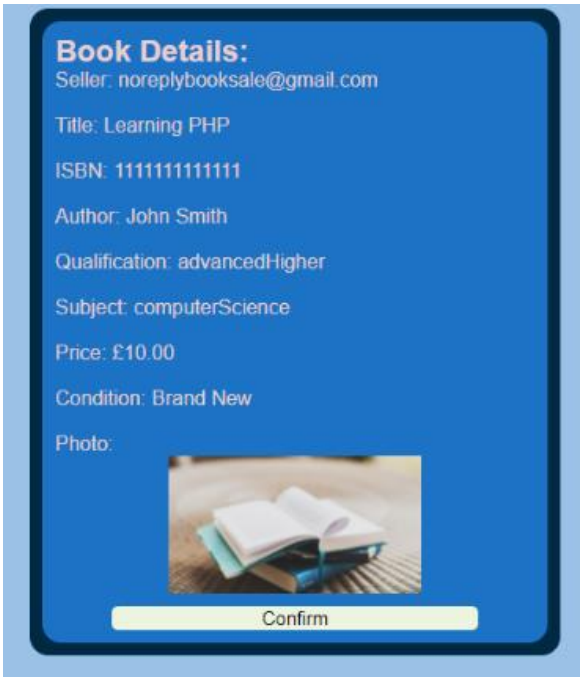
User viewing books in table:



They click on book entry and get taken to buyBook.php:



Book details:

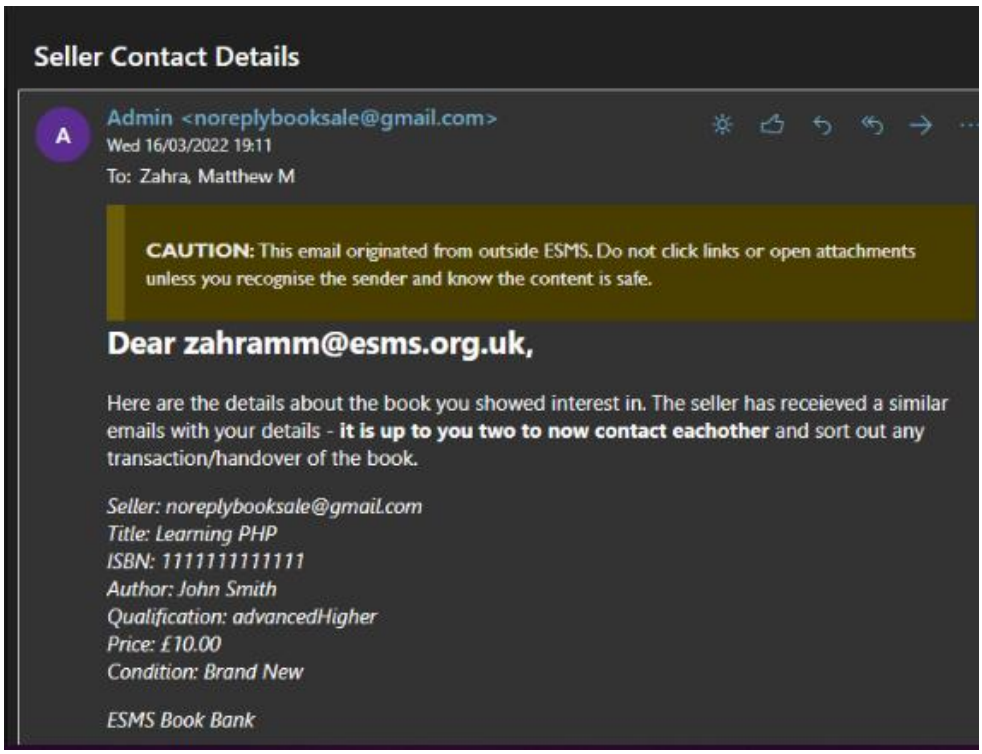


Conclusion: since book details on buyBook.php page match the details of the book that the uer clicked on, the requirement is satisified

Test: requirement 9

Actual output:

With the above buyBook.php page, the user clicks confirm to trigger the email process. They then receive this email:

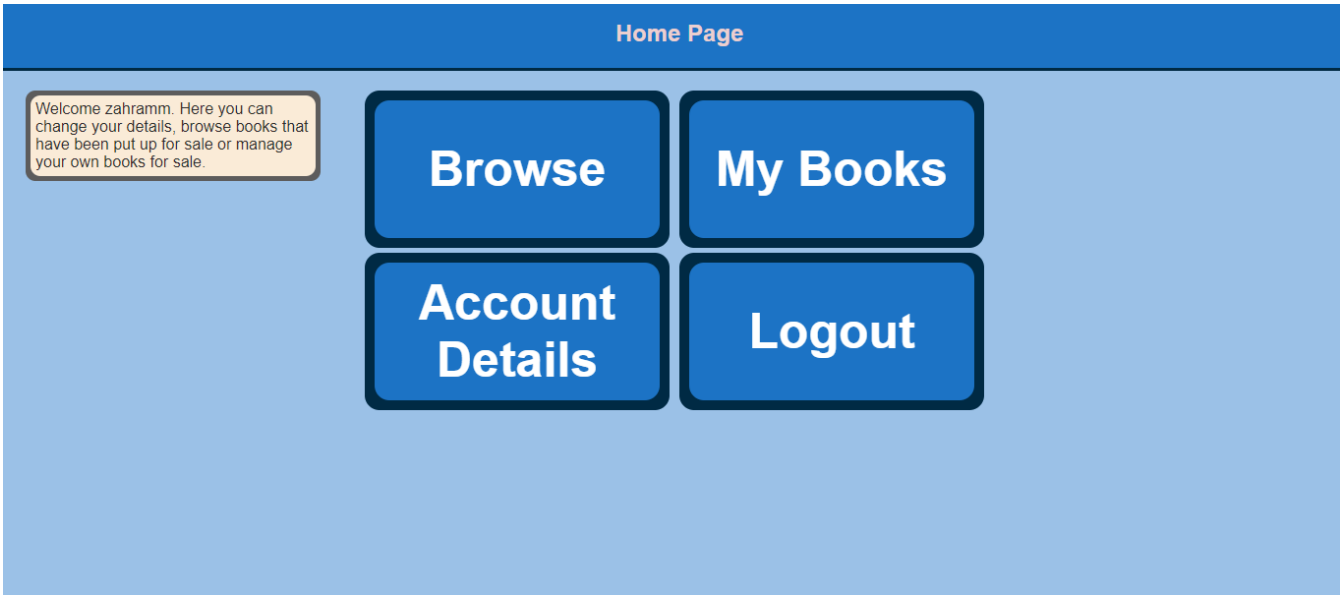


Conclusion: requirement is satisfied as the buyer receives an email with the correct book's information, including the seller's email

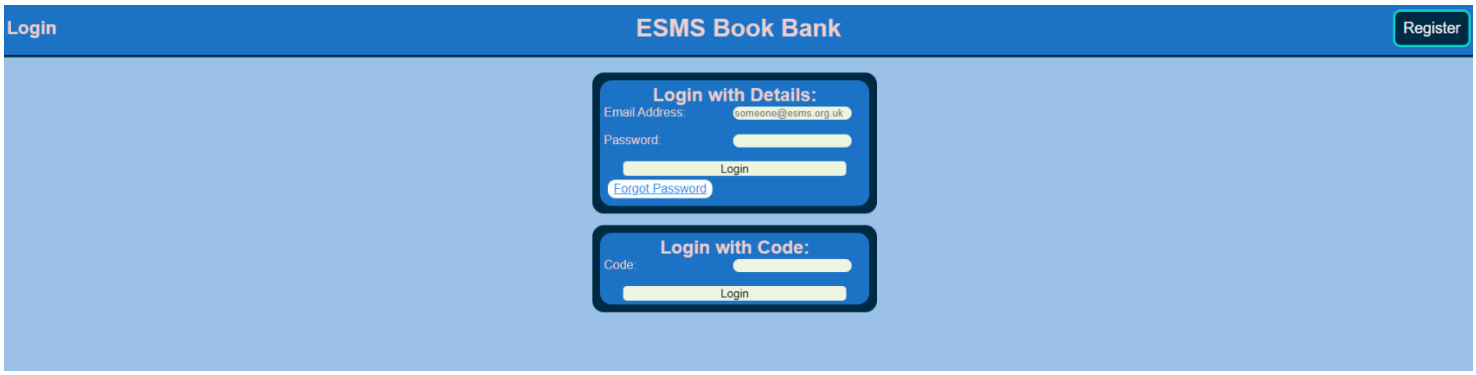
Test: requirement 10

Actual output:

User clicks logout button from home.php or the navbar:



This loads login.php:

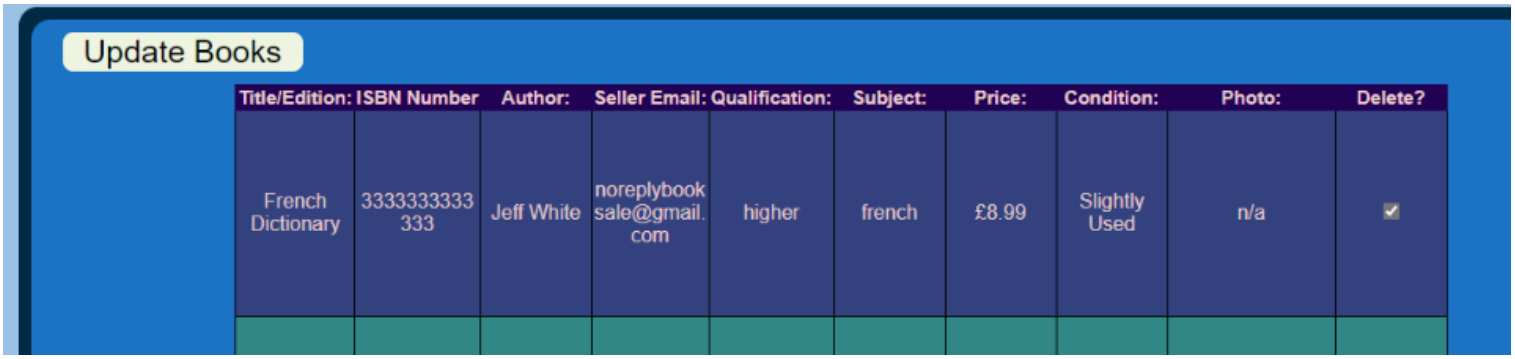


Conclusion: requirement is satisfied as user can logout

Test: requirement 11

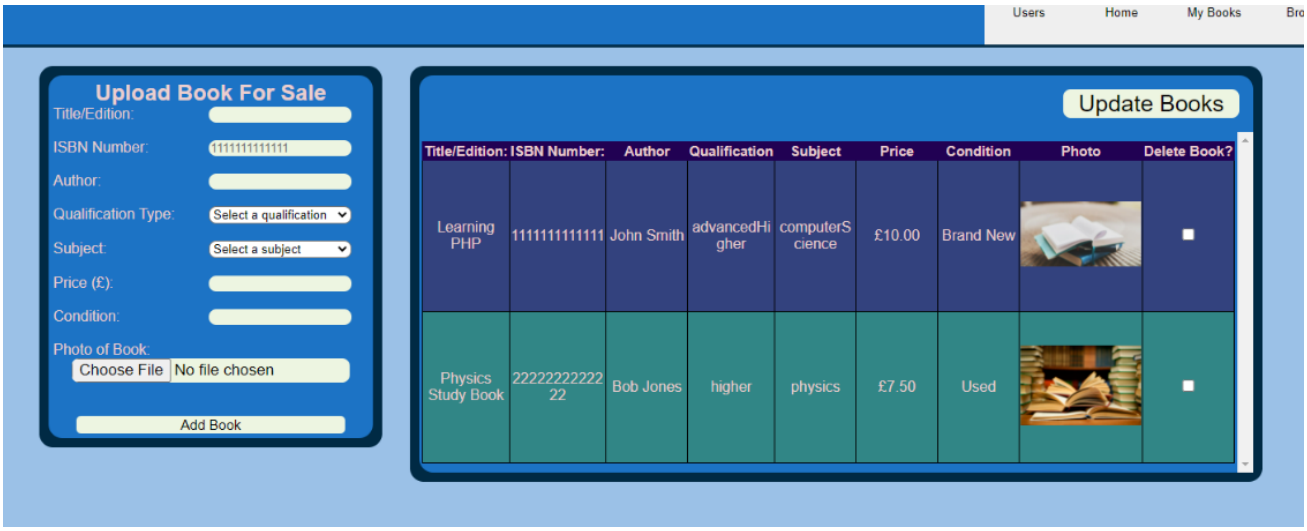
Actual output:

Admin looking at books table on browse.php page, and has ticked checkbox for delete column for first book entry:



They then click the Update Books button, submitting the form. This removes the book 'French Dictionary' from the table:

If we login as that user, we see that that book no longer appears in their myBooks.php page



Conclusion: requirement is satisfied as an admin can successfully delete another user's books that are up for sale, which removes them from the browse table and that user's own books on their myBooks.php page

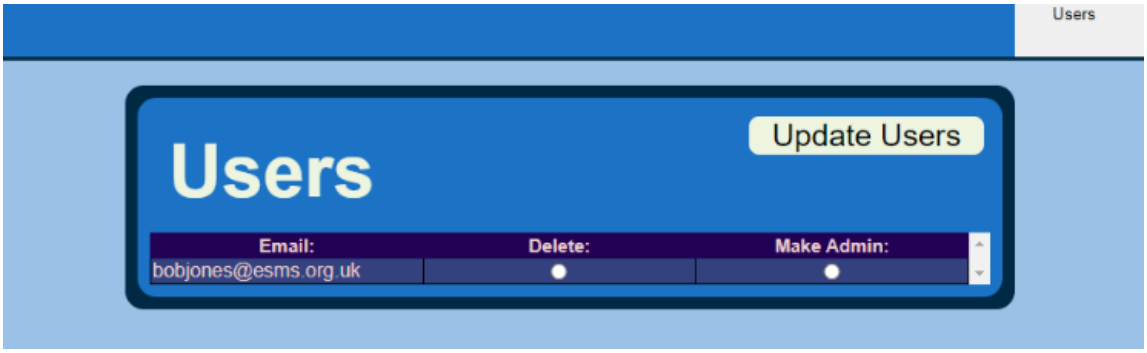
Test: requirement 12

Actual output:

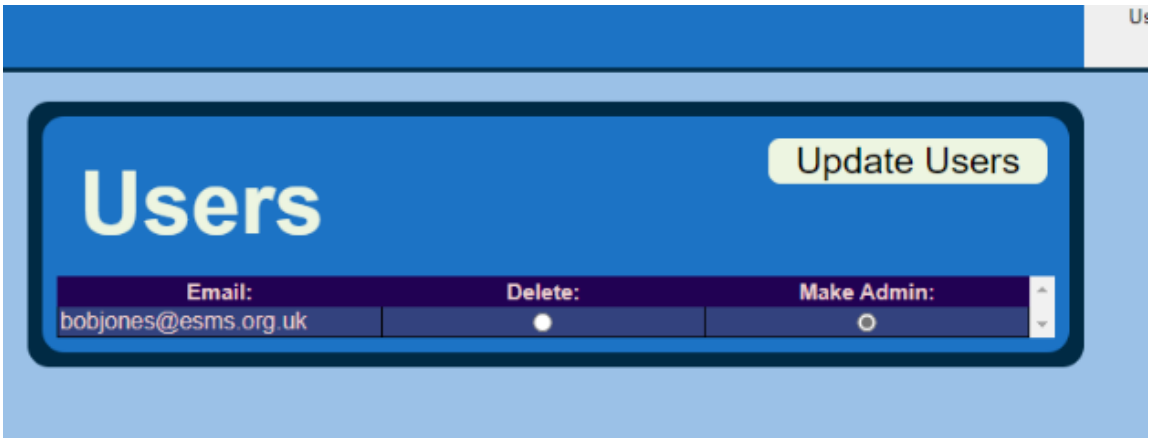
Check users in database:

```
mysql> SELECT * FROM users WHERE user_type = 'user';
+--------+---------------------+------------------------------------------------------------------+-----------+------+------------+
| userID | email               | password                                                         | user_type | code | timeOfCode |
+--------+---------------------+------------------------------------------------------------------+-----------+------+------------+
|      9 | bobjones@esms.org.uk | ed02457b5c41d964dbd2f2a609d63fe1bb7528dbe55e1abf5b52c249cd735797 | user      | NULL | NULL       |
+--------+---------------------+------------------------------------------------------------------+-----------+------+------------+
```

See that this matches the users that an admin can see:



Now check the make admin radio button:



And submit the form with the Update Users button:

And check database for status of bobjones@esms.org.uk:

```
mysql> SELECT * FROM users WHERE email = 'bobjones@esms.org.uk';
+--------+---------------------+------------------------------------------------------------------+-----------+------+------------+
| userID | email               | password                                                         | user_type | code | timeOfCode |
+--------+---------------------+------------------------------------------------------------------+-----------+------+------------+
|      9 | bobjones@esms.org.uk | ed02457b5c41d964dbd2f2a609d63fe1bb7528dbe55e1abf5b52c249cd735797 | admin     | NULL | NULL       |
+--------+---------------------+------------------------------------------------------------------+-----------+------+------------+
```

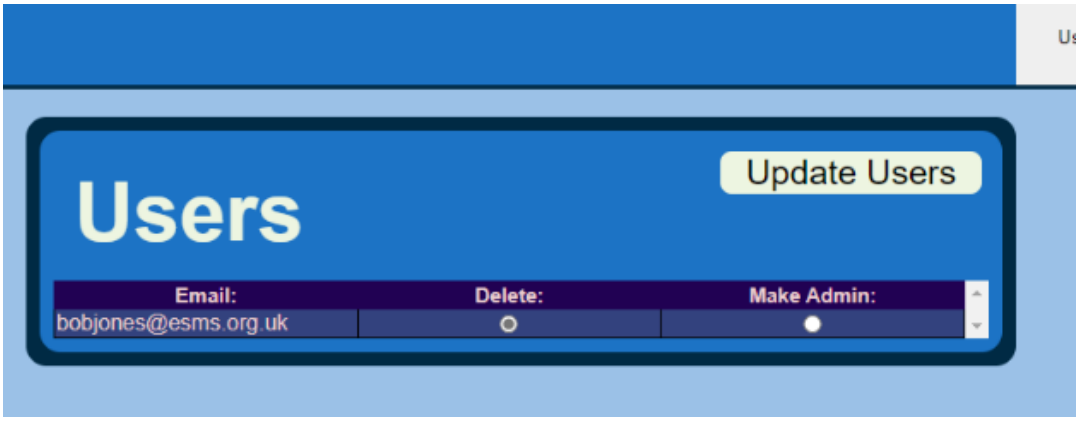
Here, we can see their status has been updated to ‘admin’

bobjones@esms.org.uk has had their user_type changed back to ‘user’

check books that bobjones@esms.org.uk has for sale in database:

```
mysql> SELECT title FROM books, users WHERE books.userID = users.userID AND email = 'bobjones@esms.org.uk';
+--------------+
| title        |
+--------------+
| Learning PHP |
+--------------+
```

Now check delete button for bobjones@esms.org.uk in users table:



Submit form:

Now check books for bobjones@esms.org.uk:

```
mysql> SELECT title FROM books, users WHERE books.userID = users.userID AND email = 'bobjones@esms.org.uk';
Empty set (0.00 sec)
```

Can see that their books have been deleted from the database

Check users table:

```
mysql> SELECT * FROM users WHERE user_type = 'user';
Empty set (0.00 sec)
```

Can see that their account has been deleted

Conclusion: requirement is satisfied as admins can view standard users, promote them to admin status, and can then delete them, which also deletes any books that they had up for sale

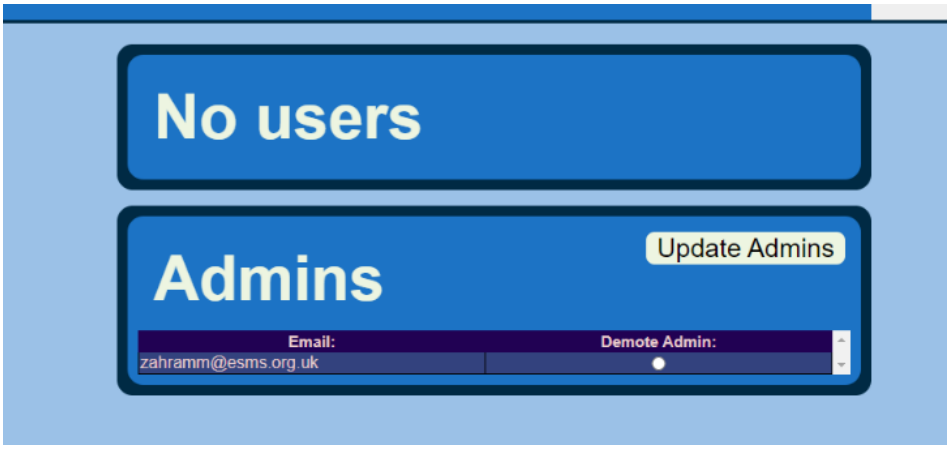
Test: requirement 13

Actual output:

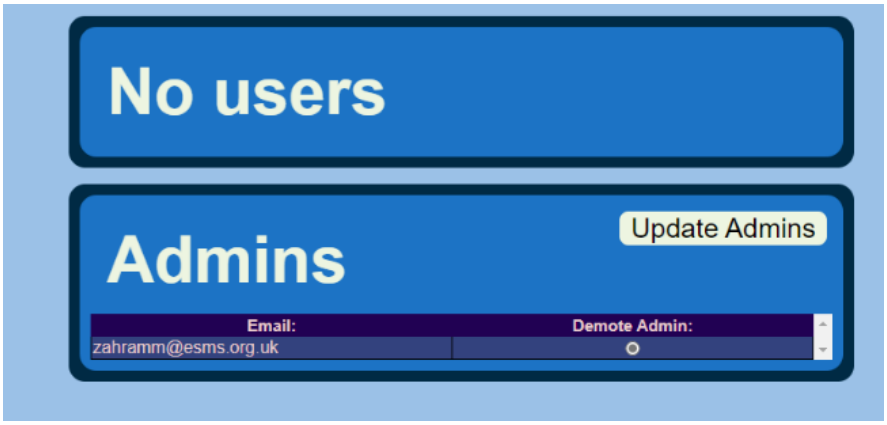
SELECT admins from database:

```
mysql> SELECT email FROM users WHERE user_type = 'admin';
+---------------------+
| email               |
+---------------------+
| zahramm@esms.org.uk |
+---------------------+
```

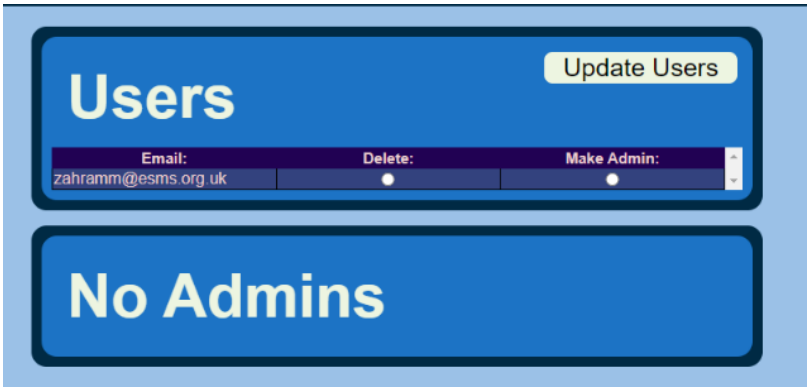
Then view admins in admins table on users.php (from a super_admin account) – can see this matches database:



Check demote button next to zahramm@esms.org.uk:



Submit form and then view tables:



Can see that admin zahramm@esms.org.uk has been been demoted to user_type = 'user'– we can prove this in the database:

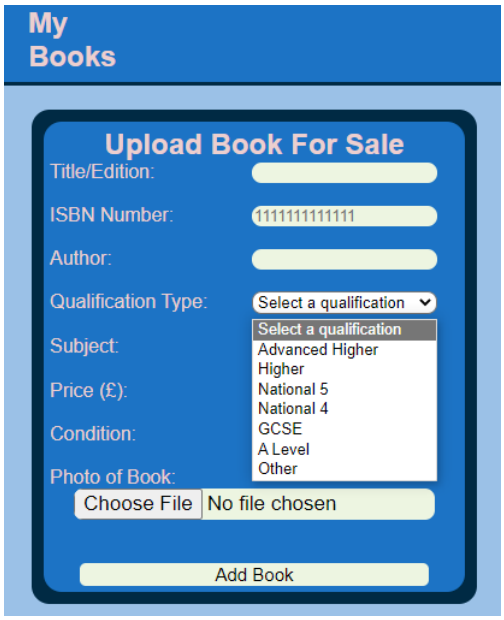
```
mysql> SELECT user_type FROM users WHERE email = 'zahramm@esms.org.uk';
+-----------+
| user_type |
+-----------+
| user      |
+-----------+
```

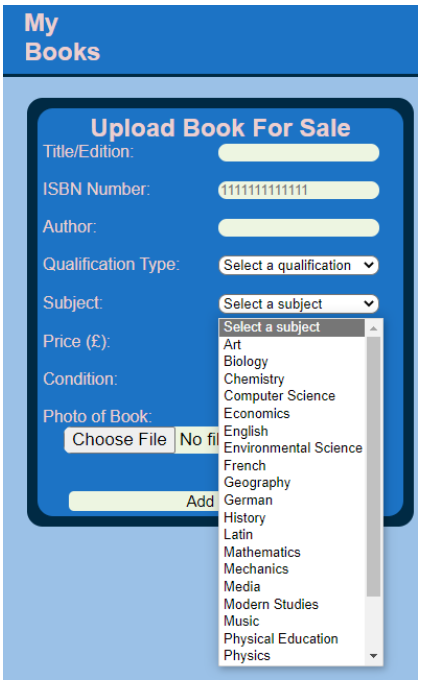
Can see here that zahramm@esms.org.uk has user_type of 'user', proving that they got demoted from 'admin'.

Conclusion: requirement has been satisfied as super_admin can view all admins and can demote any of them to 'user' status.

Test: requirement 14

14.1: restricted choice for subject/qualification for books – can only pick from a pre-defined list of options – same for filter form on browse.php

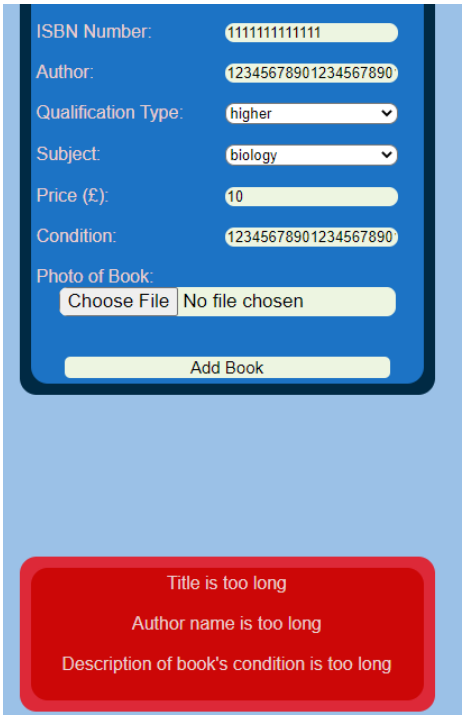




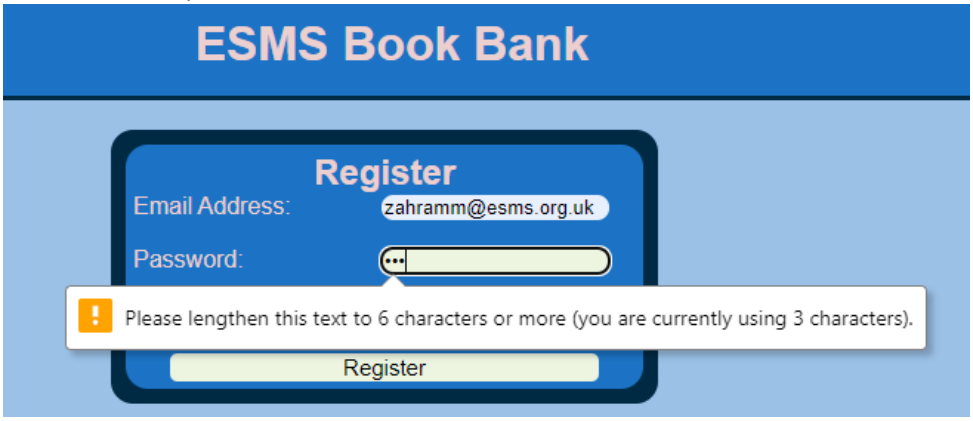
Conclusion: requirement satisfied as subject and qualification are limited by a restricted choice, provided by a drop down menu

14.2:

Client-side validation ensures that titles, emails, conditions of books and author names are <= 60 characters, so form doesn't let you type in anything that's longer than 60 characters for these fields. However, if the form is inspected and the "maxlength='60'" gets removed from the input field, then the server-side validation kicks in if the entered data is > 60 characters long:

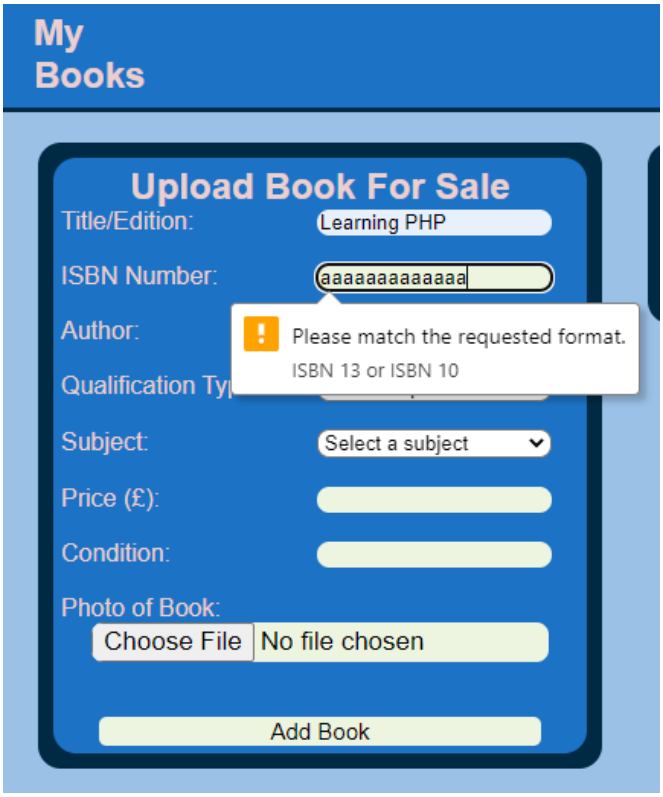


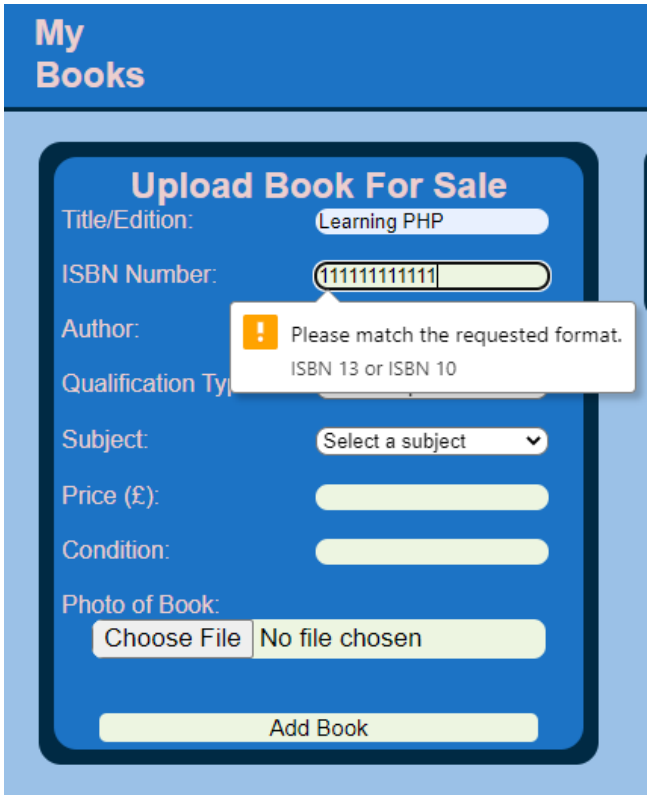
Client-side validation also ensures that passwords are > 5 characters (only when registering) and <= 256 characters (it doesn't allow you to enter a password that's more than 256 characters in length when registering and logging in – not by error message, but as demonstrated above, by stopping the user from being able to type any more characters once the limit is reached):



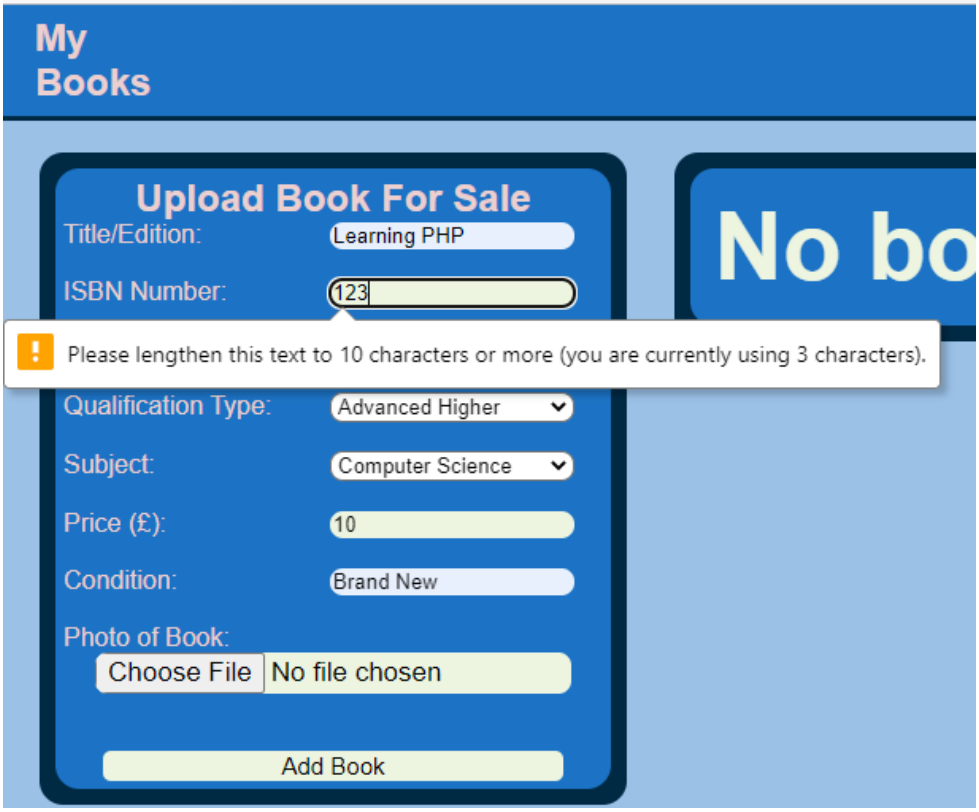
However, if client-side validation is disabled, then the server-side validation kicks in and provides error messages (same validation is in place on the accountDetails.php page when a user is trying to change their password so screenshots have been left out):

Client-side validation also ensures that ISBN numbers are completely numeric, and are either 10 or 13 digits long by prompting the user to use ISBN 10 or ISBN 13 format (exact same for filter on browse.php page so screenshots have been left out):

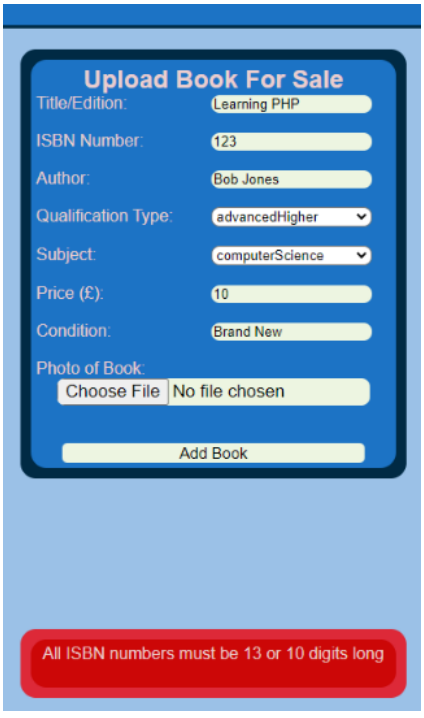


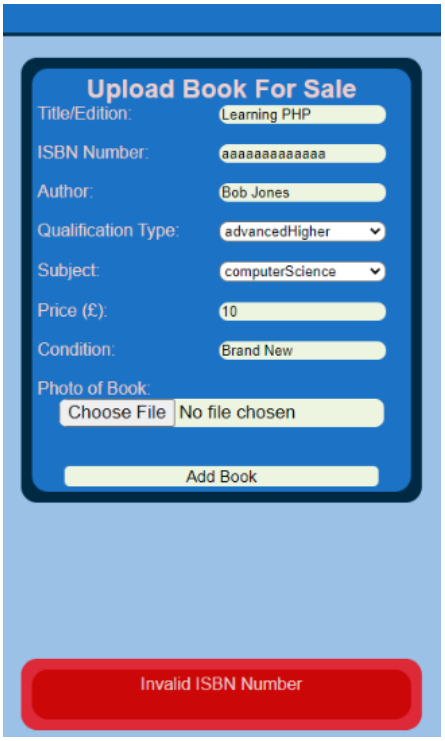


If client-side validation for the pattern (10 or 13 digits) is disabled then there's still more client-side validation to ensure that it is between 10 and 13 characters long (error message if less than 10 digits and doesn't let user type more than 13):



If all client-side validation gets disabled, then server-side validation kicks in:

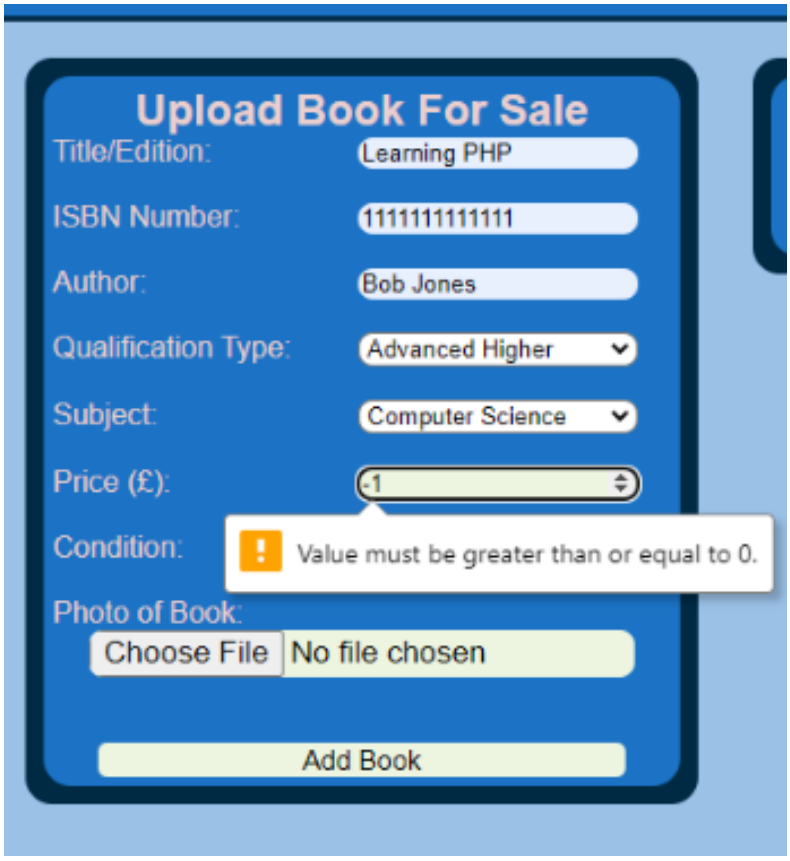




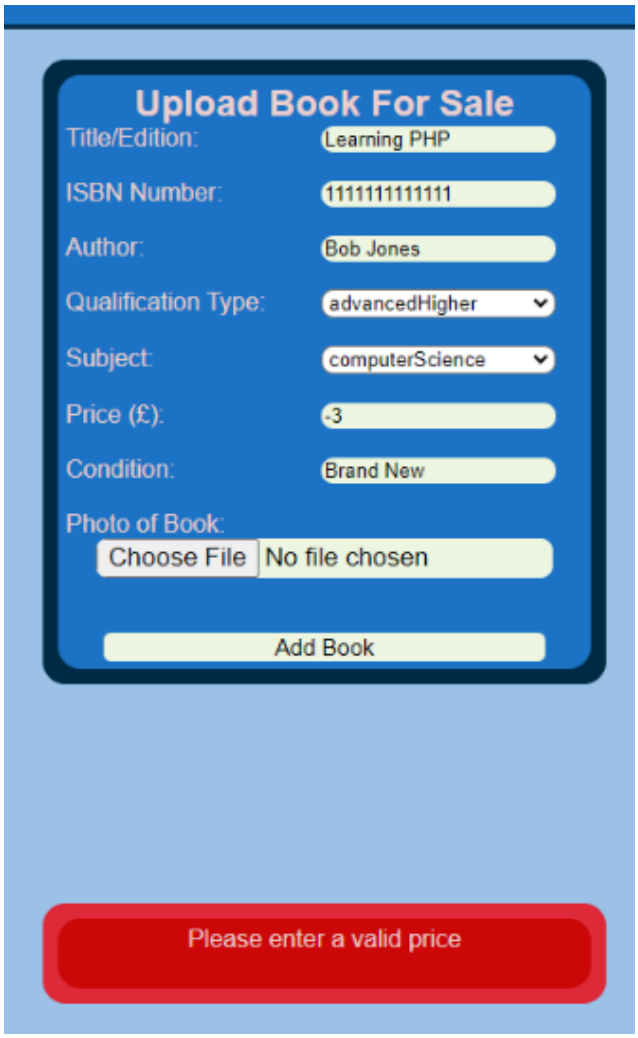
Conclusion: requirement satisfied as all emails, titles, conditions, passwords and ISBN numbers are validated to be of the correct length and format

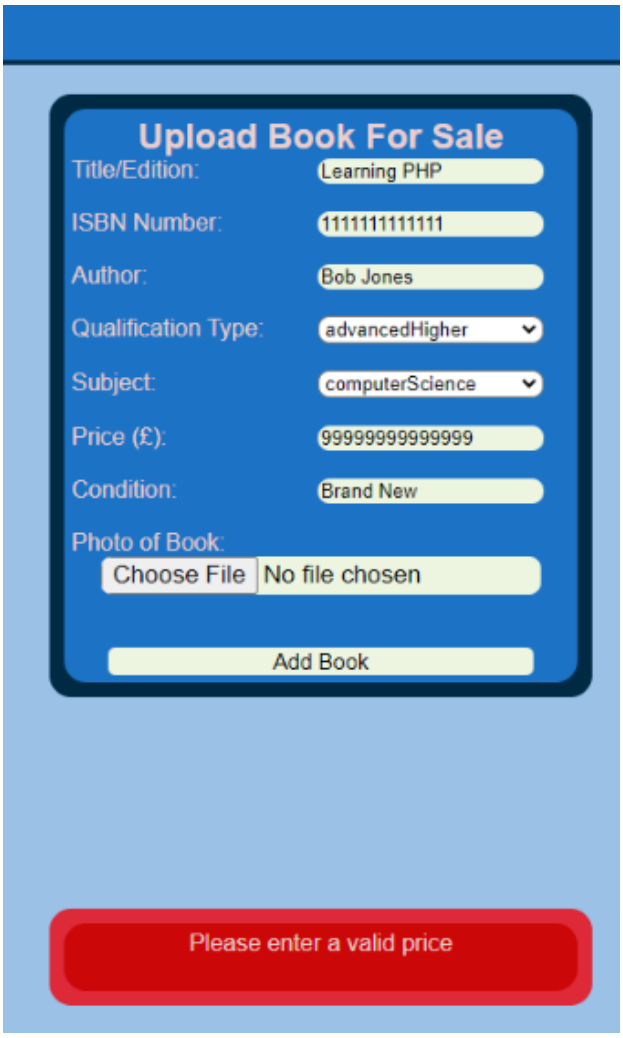
14.3:

Client-side validation to ensure price is >= 0 and <= 99999999.99:



Server-side validation in case client-side validation is disabled to ensure price is in correct





range:

Client-side validation to ensure that price is no more accurate than 2 decimal places:

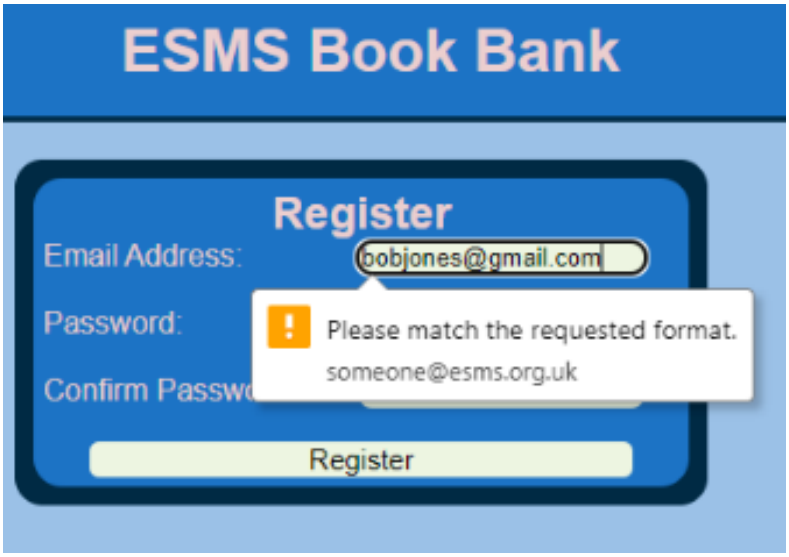
14.4:

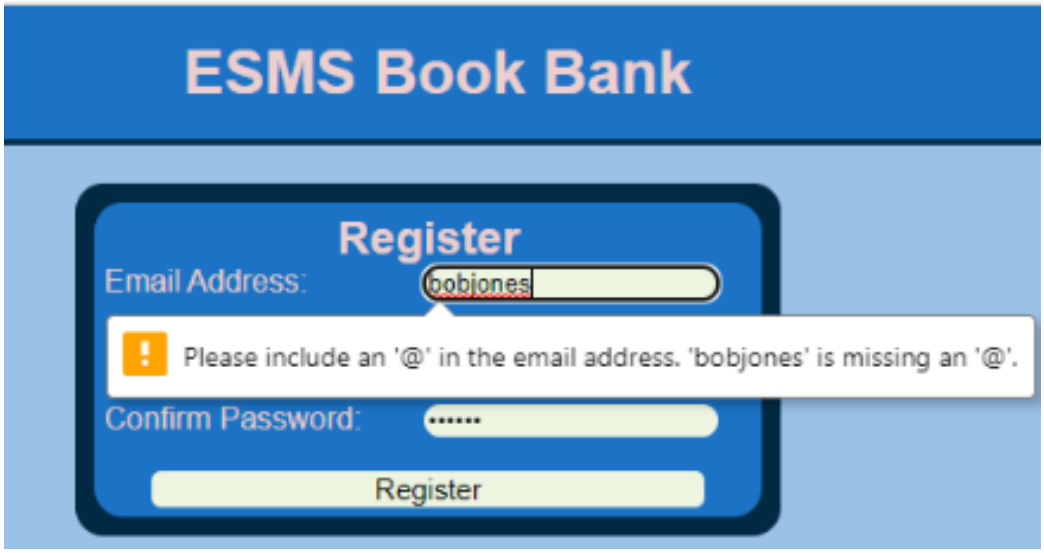
Select current users from database:

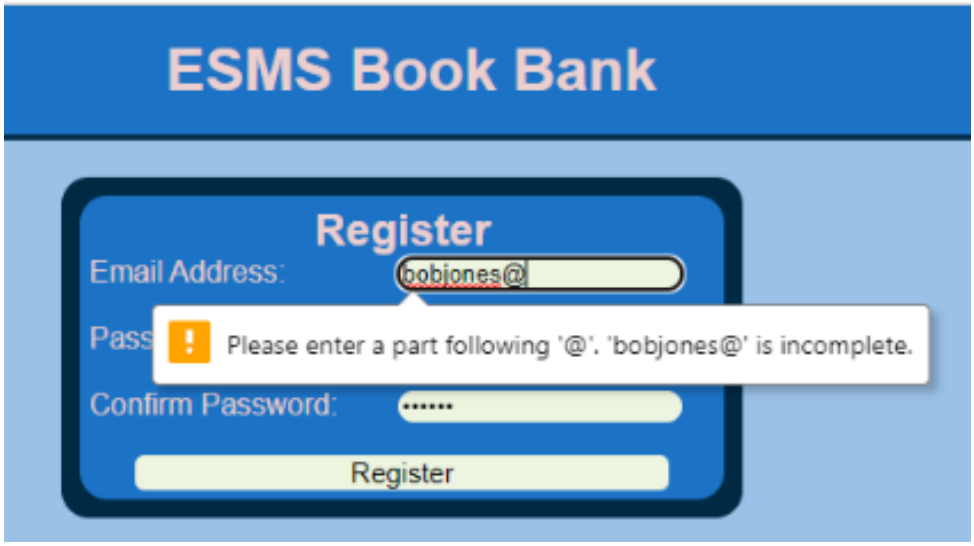
```
mysql> SELECT email, user_type FROM users;
+-----------------------------+-------------+
| email                       | user_type   |
+-----------------------------+-------------+
| noreplybooksale@gmail.com | super_admin |
| zahramm@esms.org.uk         | user        |
+-----------------------------+-------------+
```

Try and register with an email that's already in use (server-side validation kicks in):

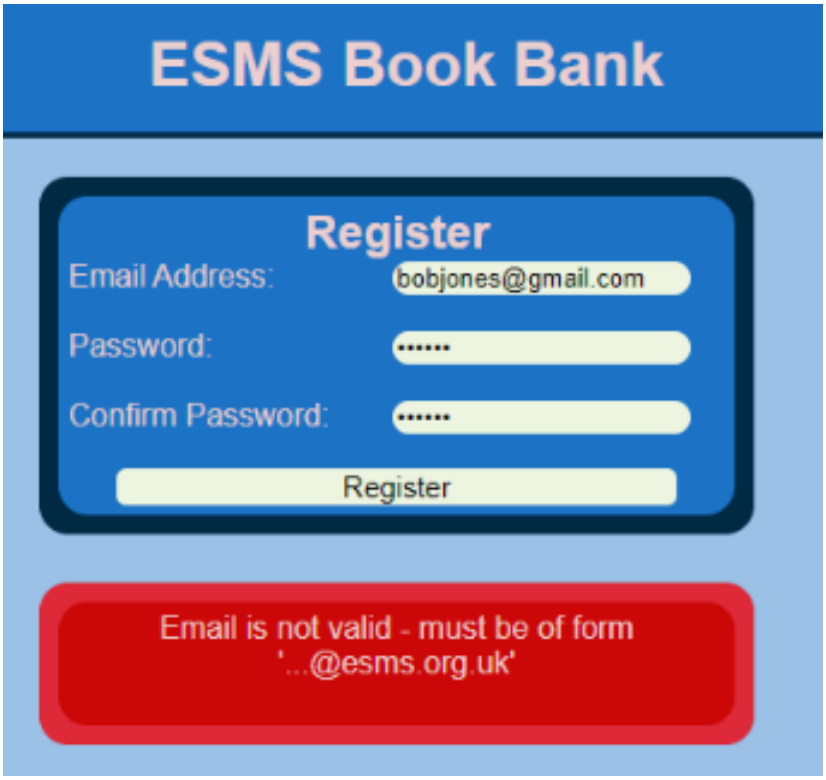
Try and register with an email not of the form '…@esms.org.uk':







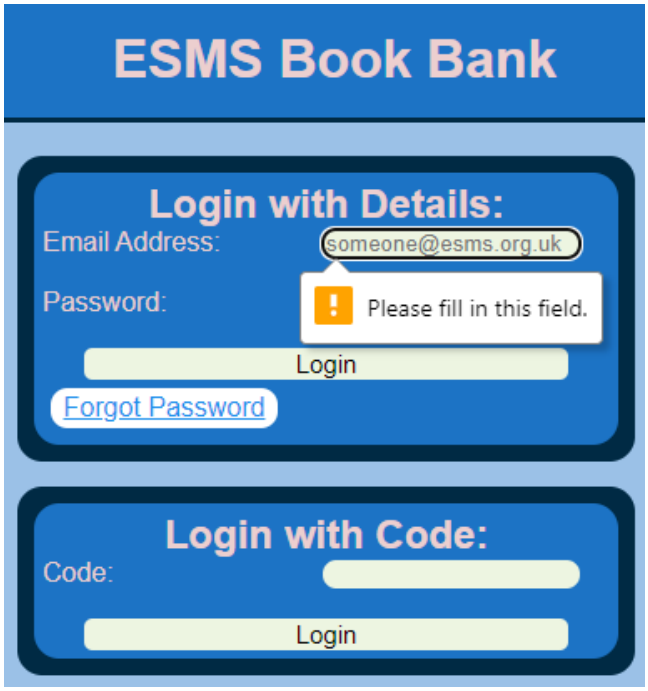
If client-side validation has been disabled, then server-side kicks in:

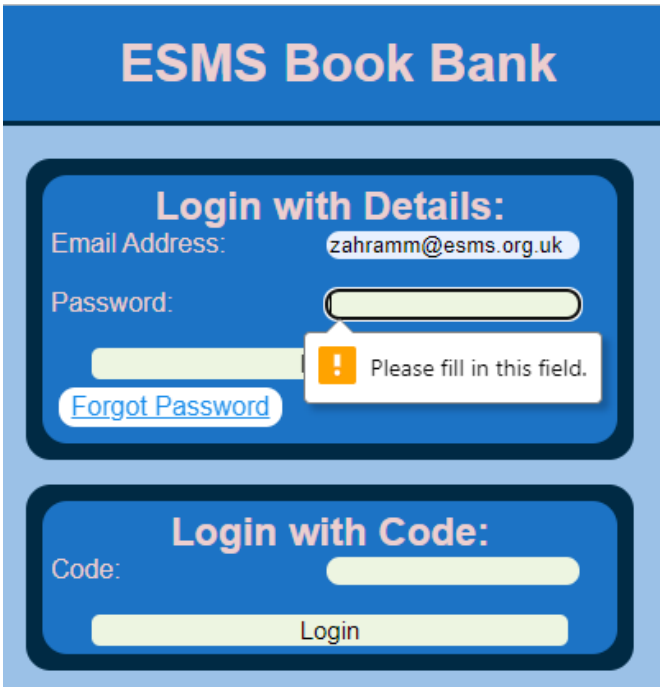


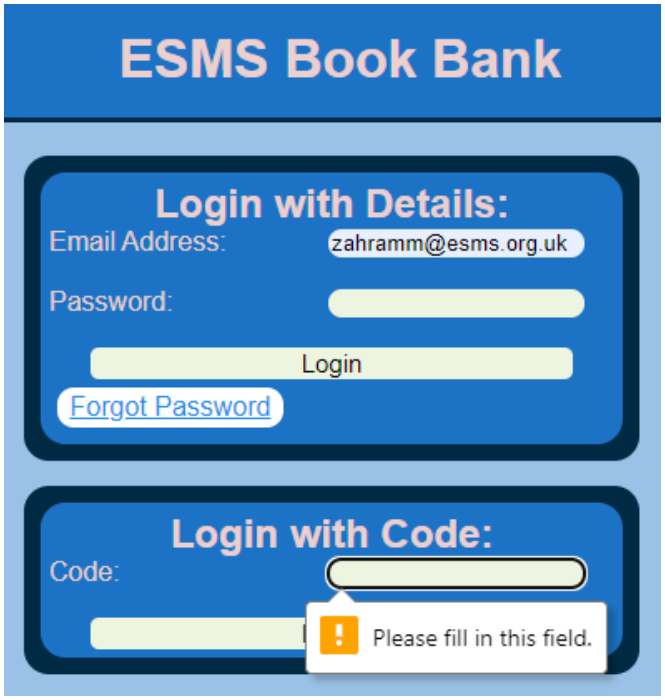
Conclusion: requirement satisfied as vaidation ensures that a user can' t register with an email already in use, and must use an email of the form '…@esms.org.uk'

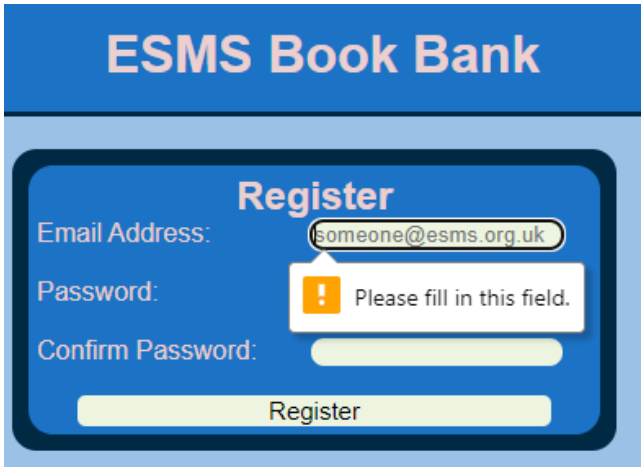
14.5:

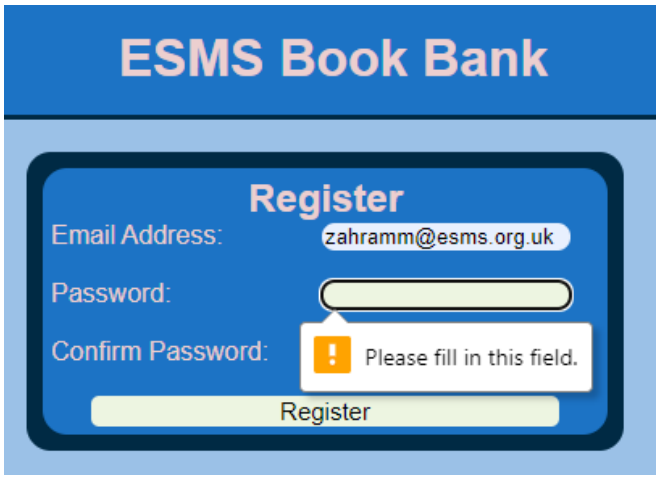
Validation for presence checks, to ensure necessary fields are entered:

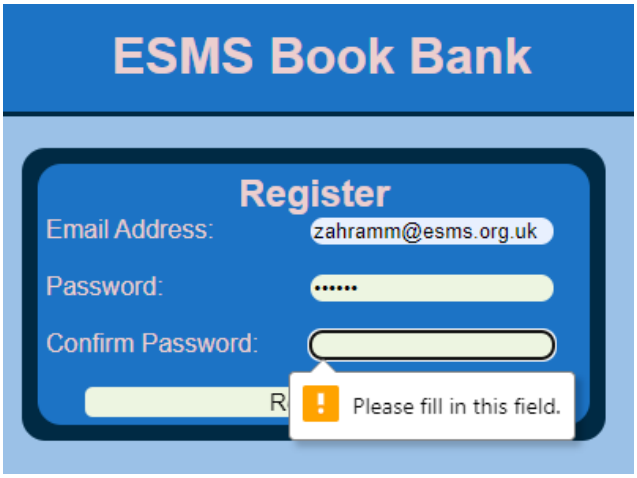


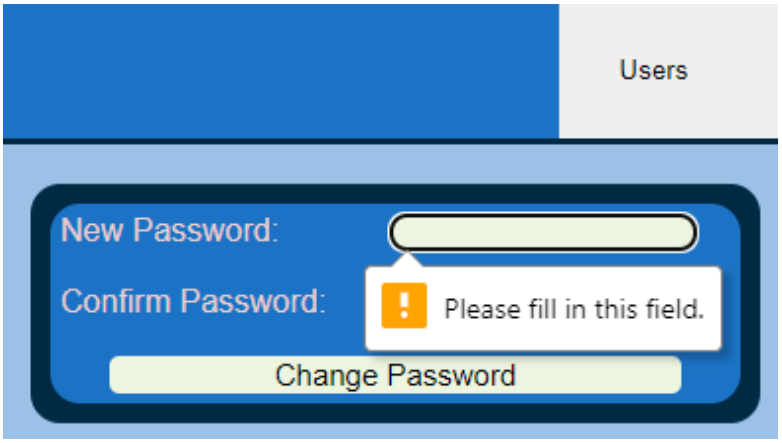


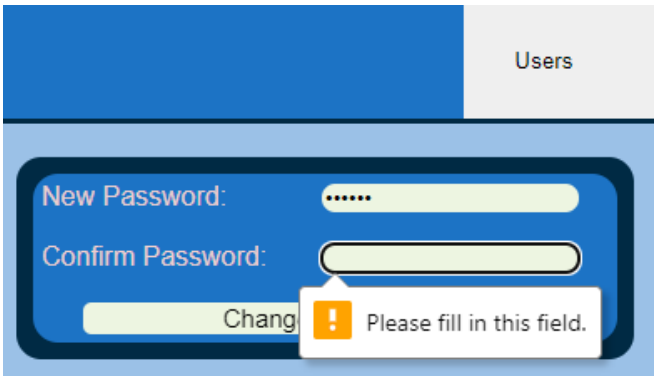


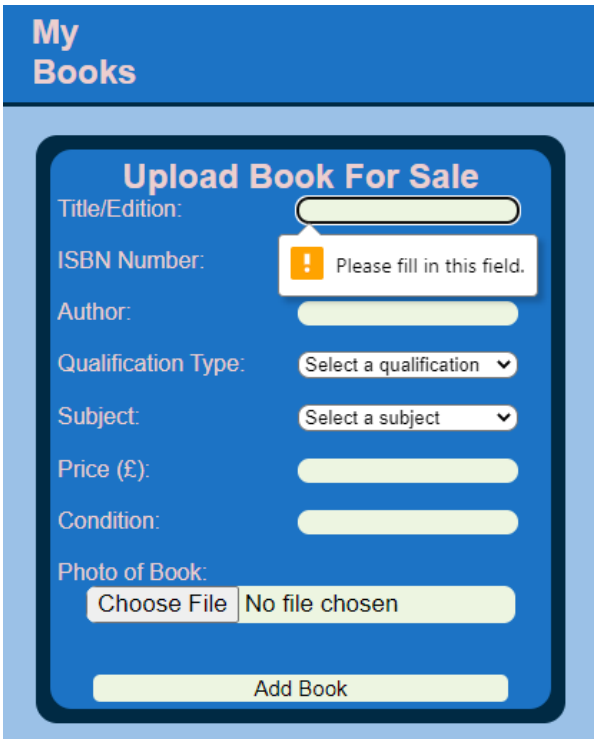


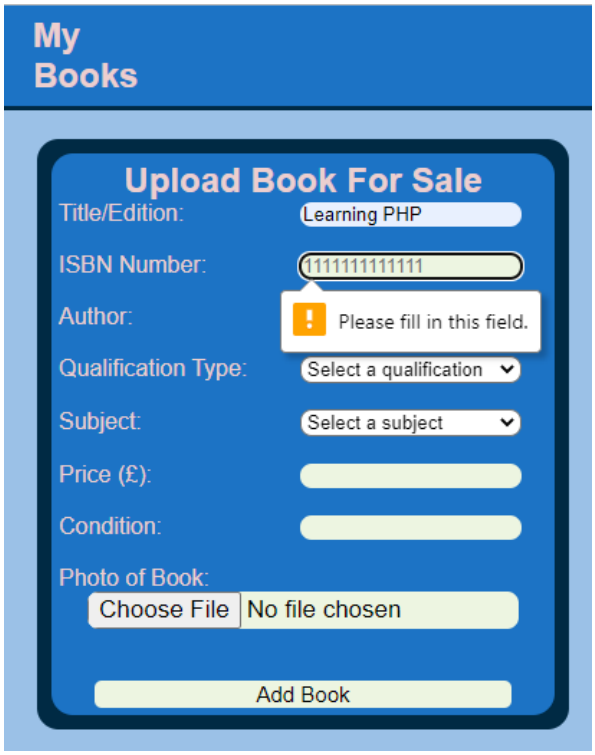


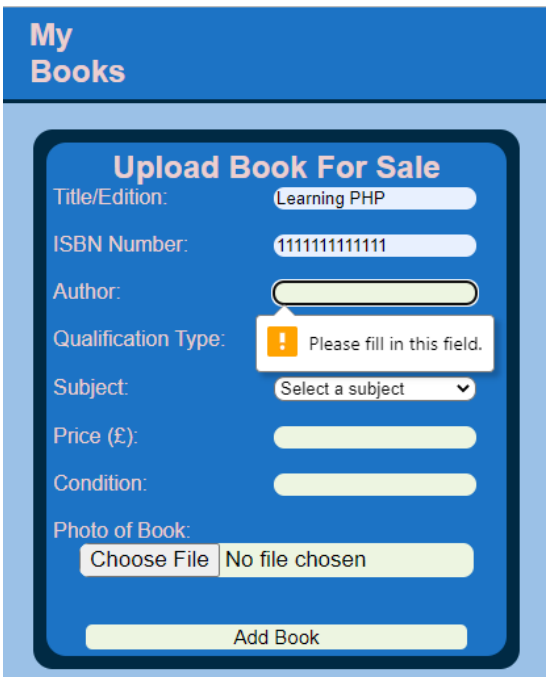

ESMS Book Bank
Register
Email Address:
zahramm@esms.org.uk
Password:
Confirm Password:
R
Please fill in this field.

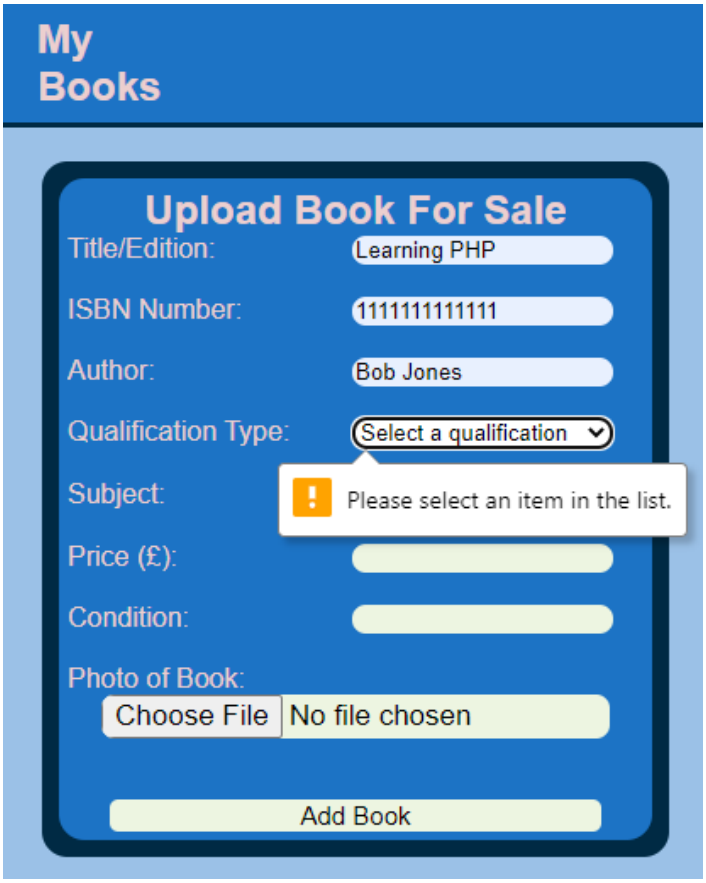

Users
New Password:
Confirm Password:
Please fill in this field.
Change Password

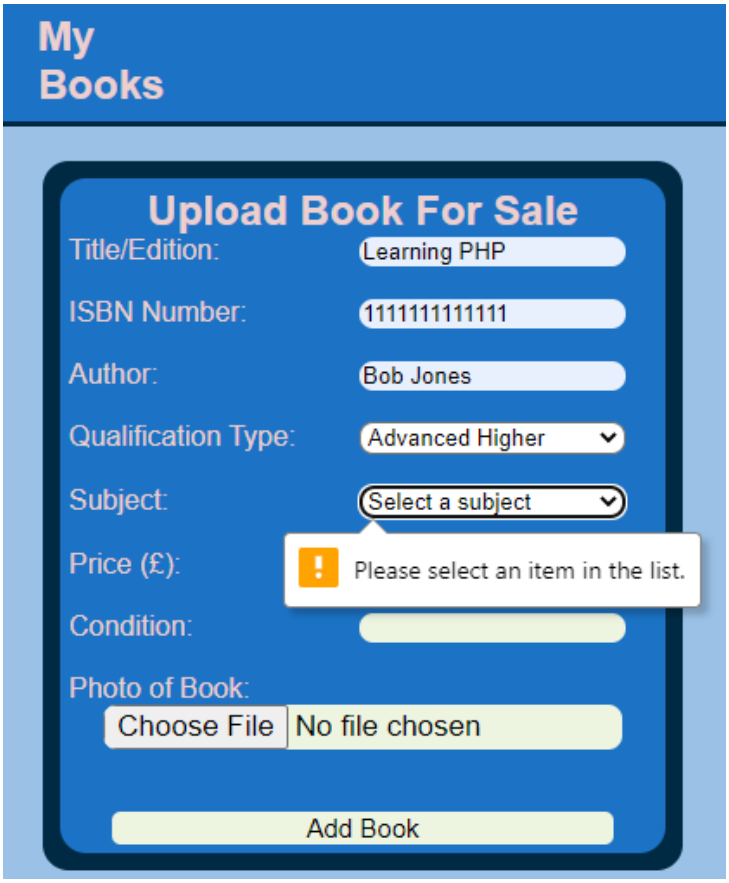

Users
New Password:
Confirm Password:
Chang
Please fill in this field.

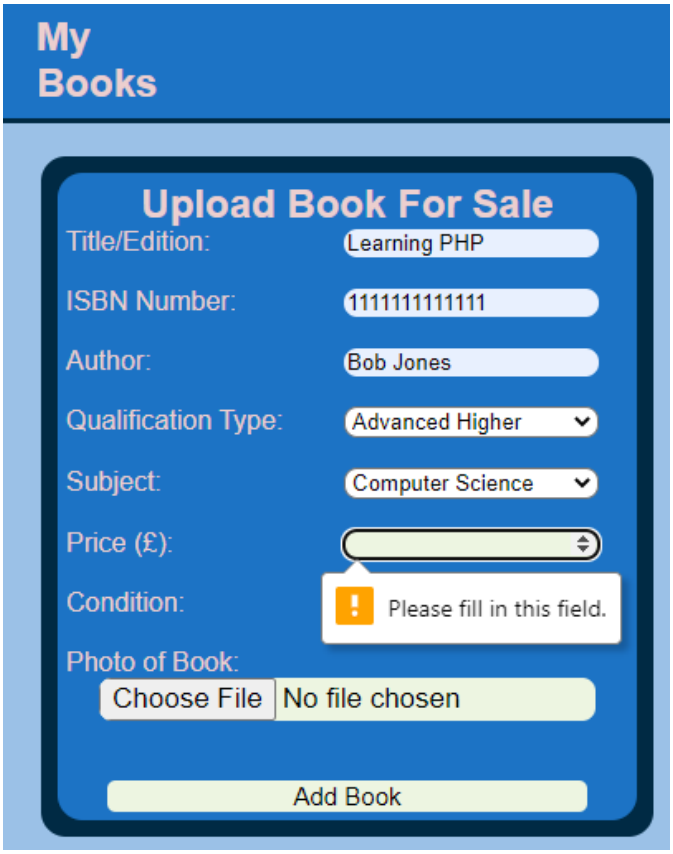

My Books
Upload Book For Sale
Title/Edition:
ISBN Number:
Please fill in this field.
Author:
Qualification Type:
Select a qualification
Subject:
Select a subject
Price (£):
Condition:
Photo of Book:
Choose File
No file chosen
Add Book

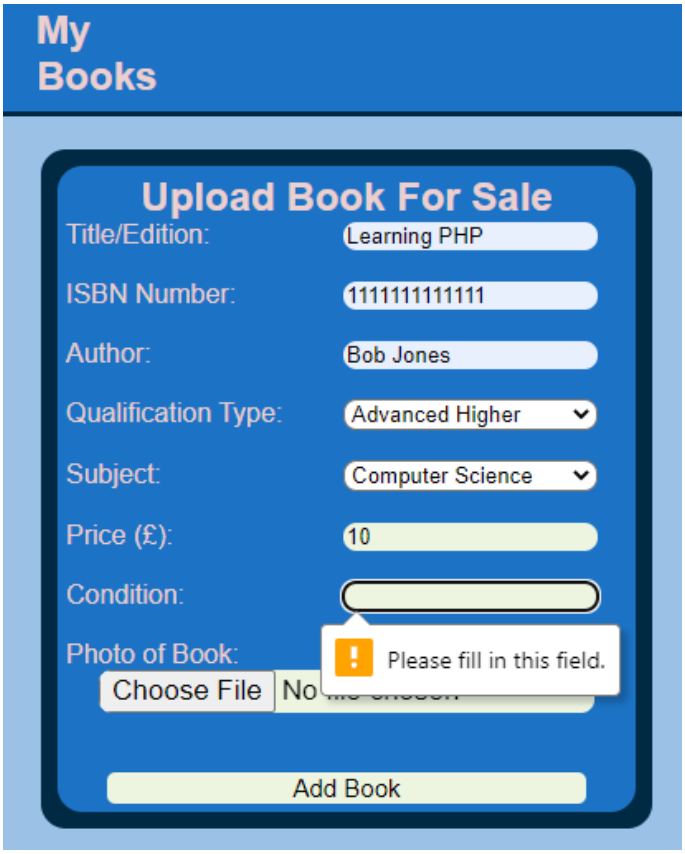

My Books
Upload Book For Sale
Title/Edition:
Learning PHP
ISBN Number:
1111111111111
Author:
Please fill in this field.
Qualification Type:
Select a qualification
Subject:
Select a subject
Price (£):
Condition:
Photo of Book:
Choose File
No file chosen
Add Book

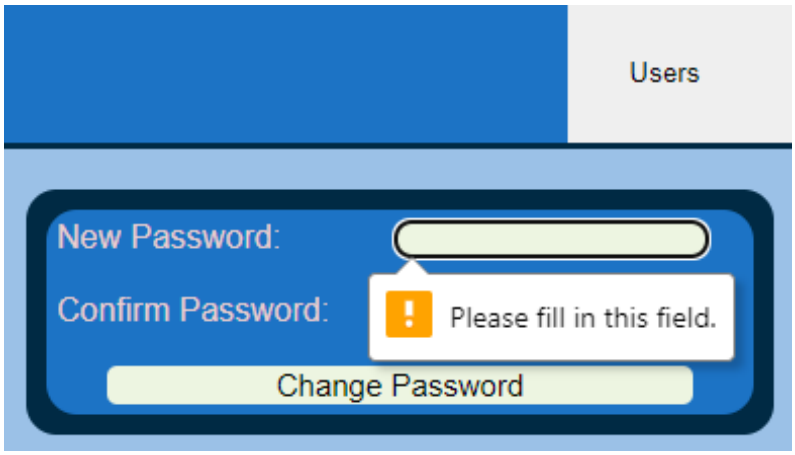

My Books
Upload Book For Sale
Title/Edition:
Learning PHP
ISBN Number:
1111111111111
Author:
Qualification Type:
Please fill in this field.
Subject:
Select a subject
Price (£):
Condition:
Photo of Book:
Choose File
No file chosen
Add Book

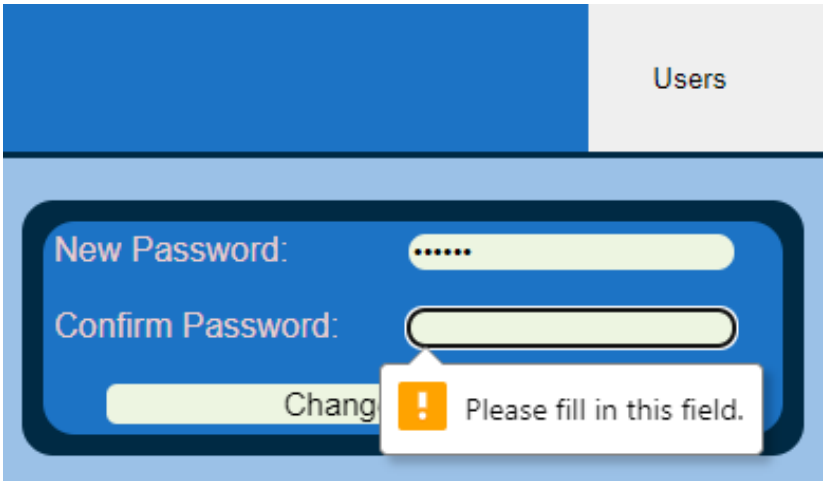

My Books
Upload Book For Sale
Title/Edition:
Learning PHP
ISBN Number:
1111111111111
Author:
Bob Jones
Qualification Type:
Select a qualification
Subject:
Please select an item in the list.
Price (£):
Condition:
Photo of Book:
Choose File
No file chosen
Add Book


My Books
Upload Book For Sale
Title/Edition:
Learning PHP
ISBN Number:
1111111111111
Author:
Bob Jones
Qualification Type:
Advanced Higher
Subject:
Select a subject
Price (£):
Please select an item in the list.
Condition:
Photo of Book:
Choose File
No file chosen
Add Book


My Books
Upload Book For Sale
Title/Edition:
Learning PHP
ISBN Number:
1111111111111
Author:
Bob Jones
Qualification Type:
Advanced Higher
Subject:
Computer Science
Price (£):
Condition:
Please fill in this field.
Photo of Book:
Choose File
No file chosen
Add Book


My Books
Upload Book For Sale
Title/Edition:
Learning PHP
ISBN Number:
1111111111111
Author:
Bob Jones
Qualification Type:
Advanced Higher
Subject:
Computer Science
Price (£):
10
Condition:
Please fill in this field.
Photo of Book:
Choose File
Add Book



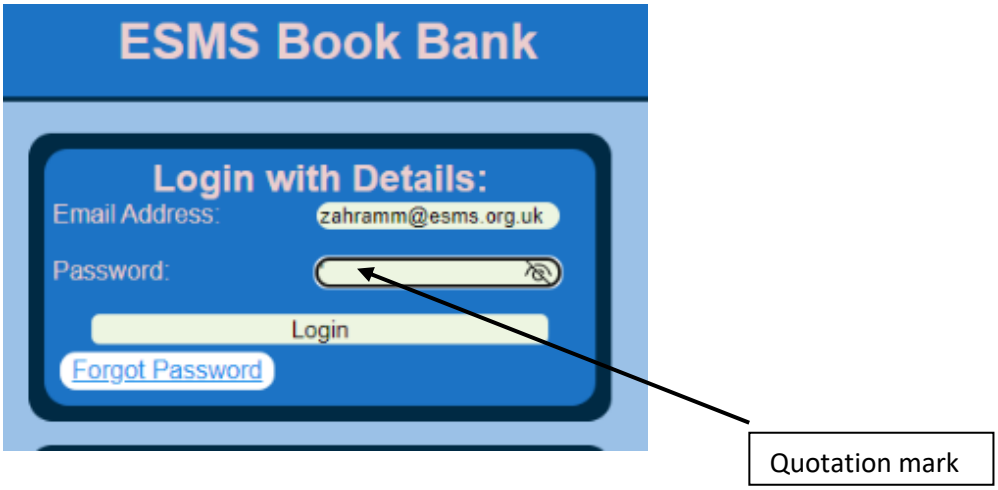


As with before, it the client-side validation gets disabled, then the server-side validation will kick in and will display error messages if any necessary fields are missing.

Conclusion: requirement satsified as validation ensures that all required fields are entered into forms – if disabled, then server-side validation does the same job, displaying error messages where appropriate.

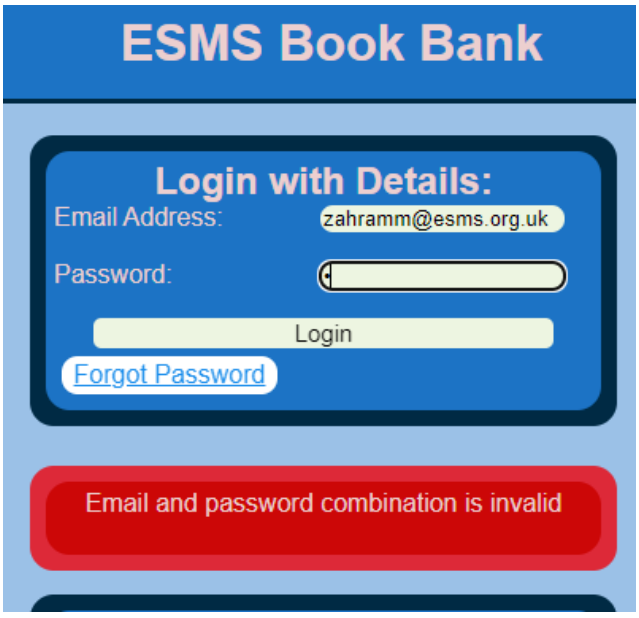
14.6:

All forms use the function mysqli_real_escape_string() on the server-side in order to ensure that SQL injection is impossible. When entering a password, if the input was a quotation mark, then without the function, this would throw up an error:



Fatal error: Uncaught TypeError: mysqli_num_rows(): Argument #1 ($result) must be of type mysqli_result, bool given in C:\Abyss Web Server\htdocs\ahProject\php\loginHandler.php:36 Stack trace: #0 C:\Abyss Web Server\htdocs\ahProject\php\loginHandler.php(36): mysqli_num_rows() #1 {main} thrown in **C:\Abyss Web Server\htdocs\ahProject\php\loginHandler.php** on line **36**

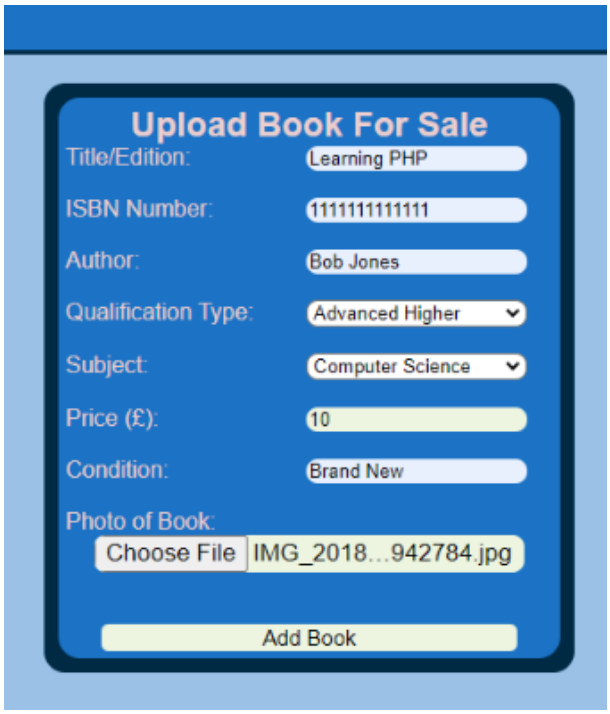
More clever SQL injection could lead to the database contents being stolen, manipualted or deleted. Since each form uses the function, SQL injection is not possible at any point, and the forms deal with the inputs:

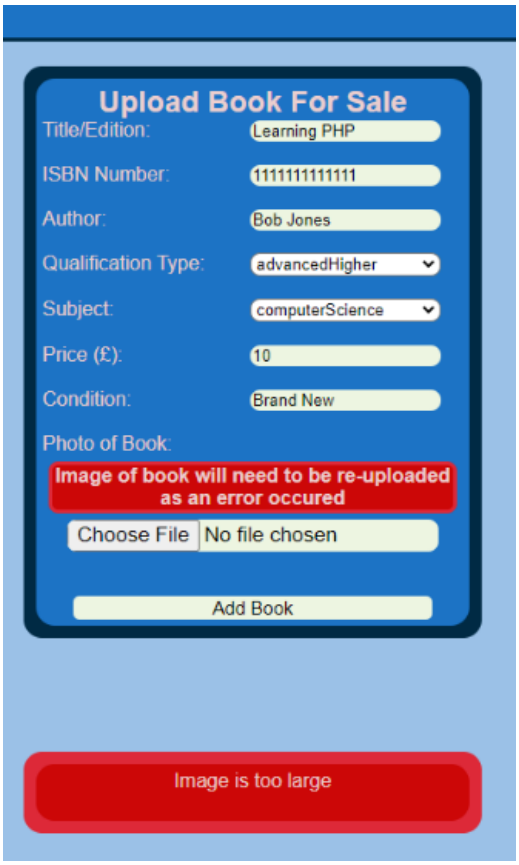


Conclusion: requirement satisfied as each form uses function to escape SQL injection

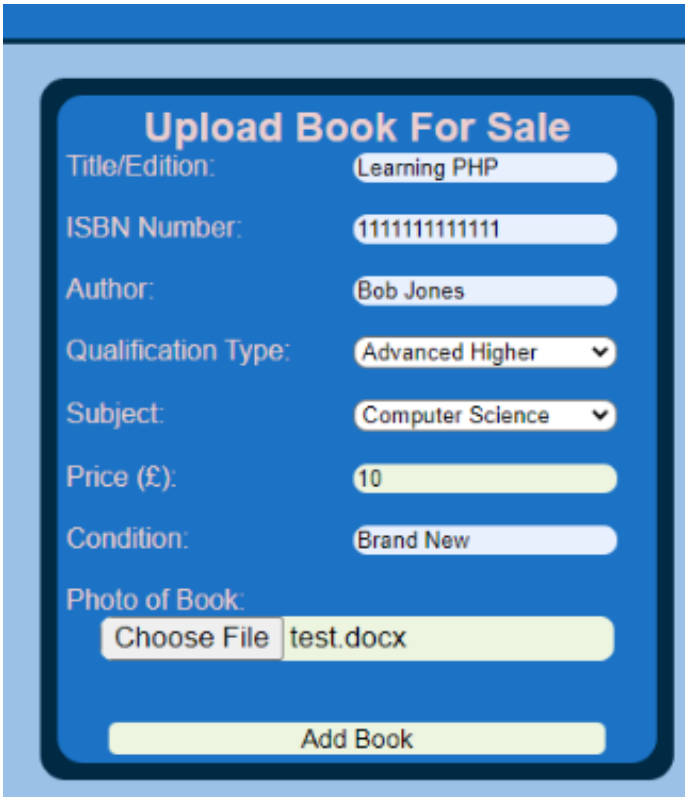
14.7:

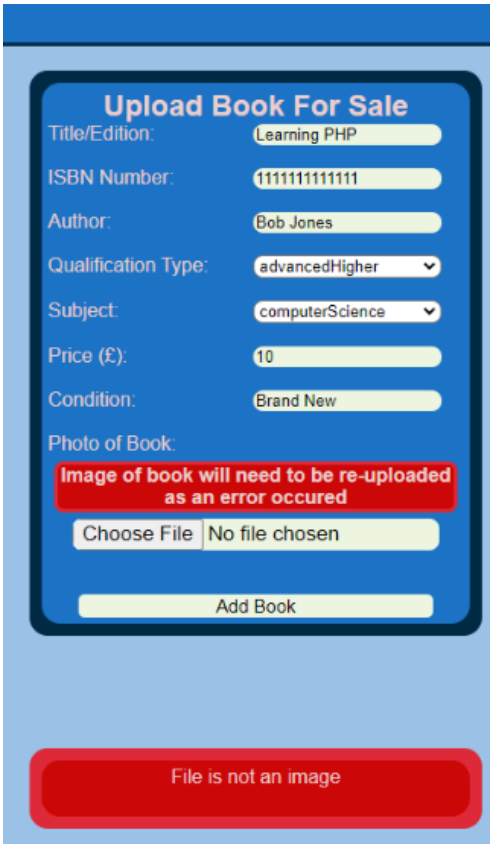
Trying to upload a file that's > 3Mb in size (server-side validation catches this):





Trying to upload a file of the wrong type (not .jpg, .jpeg or .png):

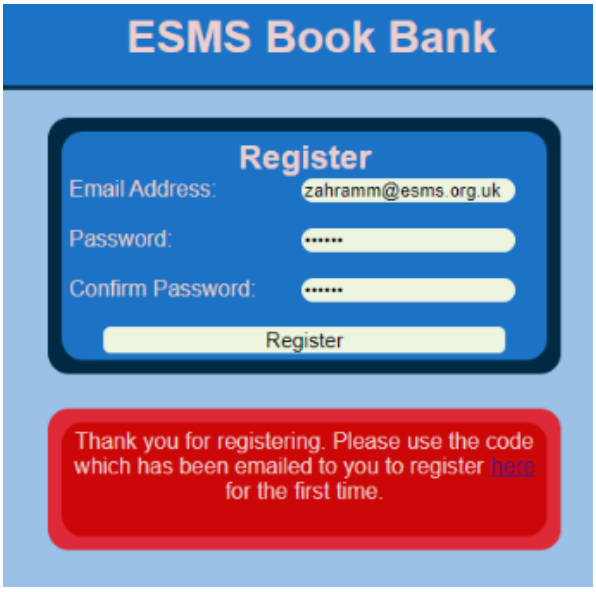


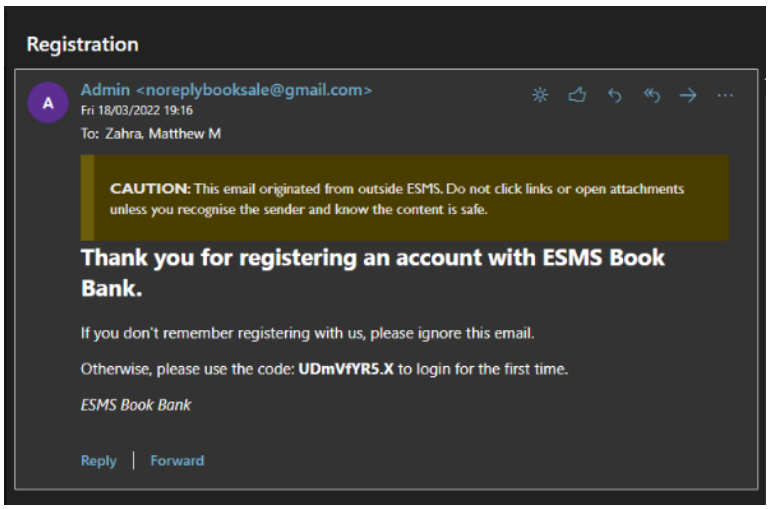


Conclusion: requirement is satisfied as server-side validation ensures that file has to be of the correct type and <= 3Mb in size.

14.8:

Register and get a verification code by going to register.php and typing in a valid email and password and see code sent in email:

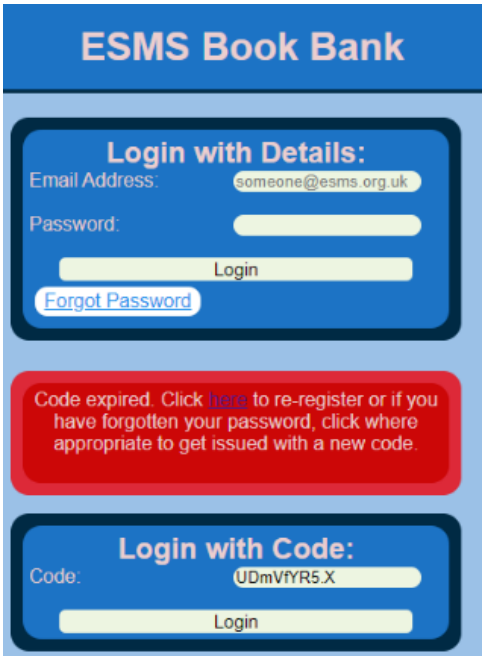




Extract timeOfCode for this code from database:

```
mysql> SELECT email, timeOfCOde FROM users WHERE email = 'zahramm@esms.org.uk';
+---------------------+---------------------+
| email               | timeOfCOde          |
+---------------------+---------------------+
| zahramm@esms.org.uk | 2022-03-18 19:16:14 |
+---------------------+---------------------+
```

I will now wait for more than 5 minutes after the above timestamp, so that the code has expired:



If I now enter an invalid code:

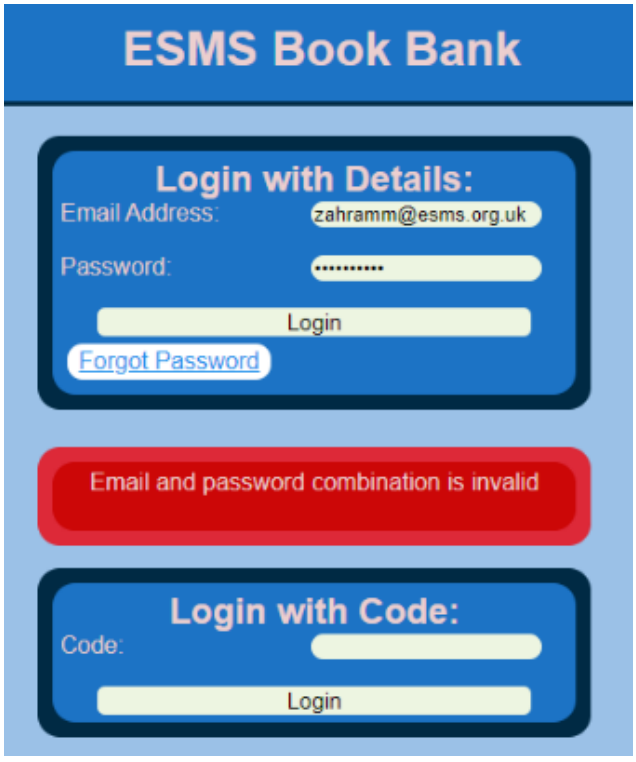
Conclusion: requirement met as verification code is valdiated – checked it is correct and checked it has been entered within 5 mintues of its issue

14.9:

Registering with an email already in use has already been shown to be validated above.

If the user tries to register without providing matching passwords:

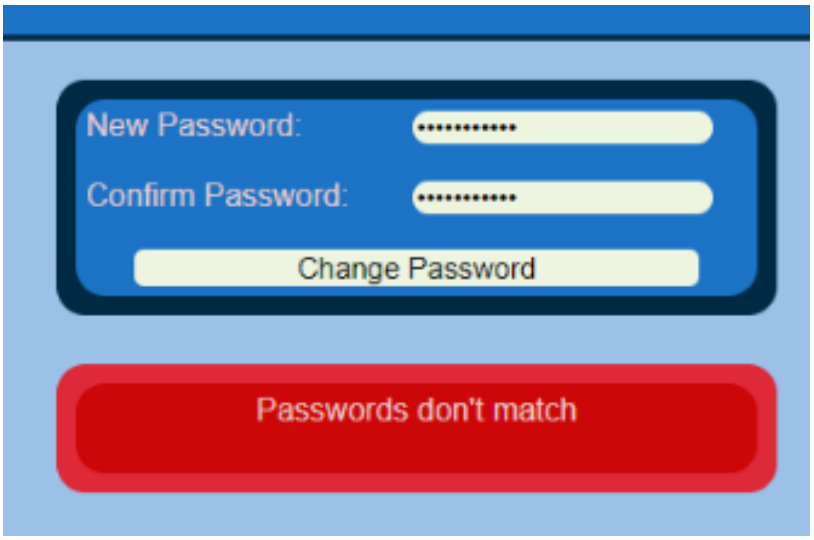
If user tries to login with invalid details:



Conclusion: requirement satisfied as login and registration details are all validated (length and presence checks have been shown in previous tests above)

14.10:

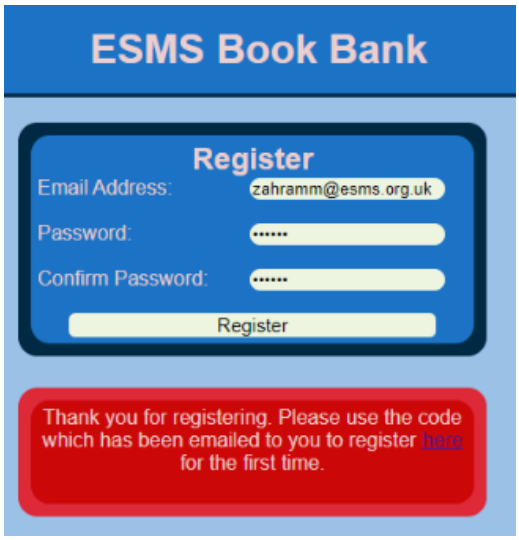
If a user tries to change their password by providing 2 non-matching passwords, then validation kicks in (presence and length checks shown above in previous tests):

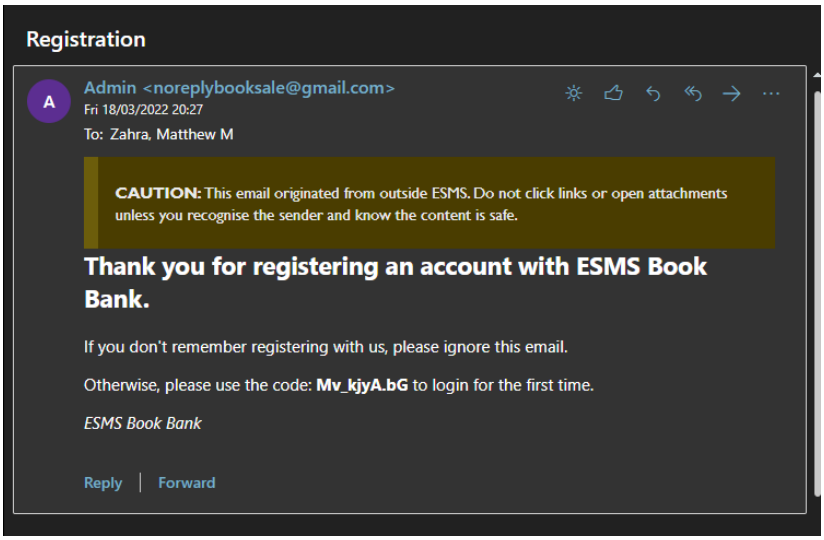


Conclusion: requirement satisfied as validation ensures that passwords match when a user attempts to change their password

Test: requirement 15

User registers and recieves code in email :





Check to see if their details have been added to the database:

```
mysql> SELECT * FROM users WHERE email = 'zahramm@esms.org.uk';
+--------+-----------------------+------------------------------------------------------------------+-----------+------------------------------------------------------------------+---------------------+
| userID | email                 | password                                                         | user_type | code                                                             | timeOfCode          |
+--------+-----------------------+------------------------------------------------------------------+-----------+------------------------------------------------------------------+---------------------+
|     11 | zahramm@esms.org.uk | ed02457b5c41d964dbd2f2a609d63fe1bb7528dbe55e1abf5b52c249cd735797 | pending   | 063bc00012371f6ead4bc2bbdb564116e7025c86b0e202e9080e00d467fbece8 | 2022-03-18 20:27:38 |
+--------+-----------------------+------------------------------------------------------------------+-----------+------------------------------------------------------------------+---------------------+
```

User email and (hashed) password have been successfully added, user_type has been set to ‘pending’ (waiting for verification code) and timeOfCode has correctly been set to the time that the user registered.

Conclusion: requirement is satisfied as user details are correctly added to the database when the user registers

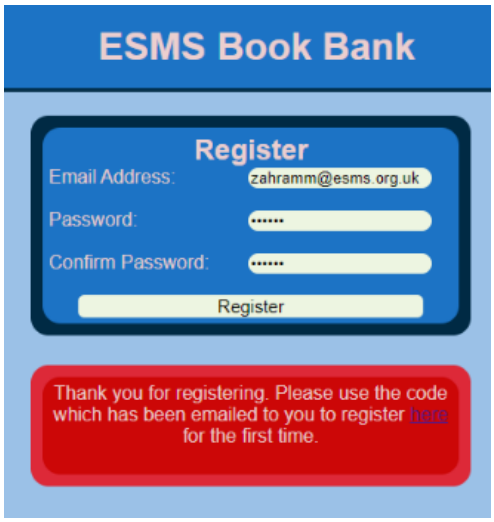
Test: requirement 16

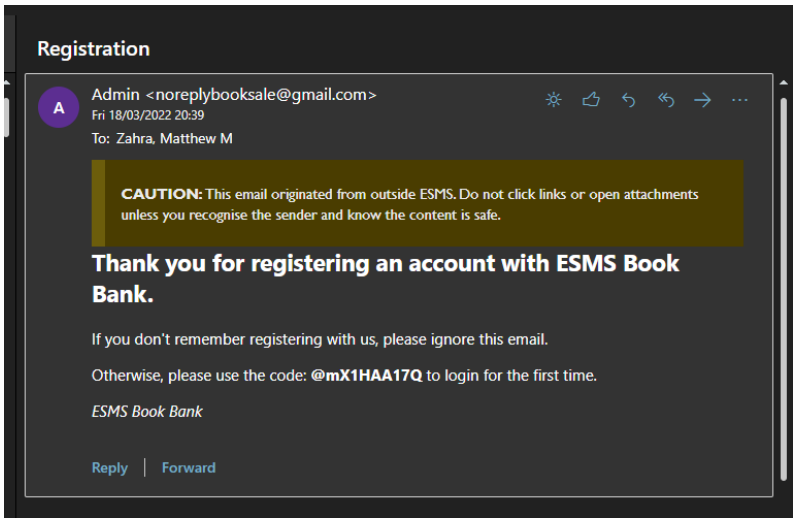
```
mysql> SELECT * FROM users WHERE email = 'zahramm@esms.org.uk';
+--------+-----------------------+------------------------------------------------------------------+-----------+------------------------------------------------------------------+---------------------+
| userID | email                 | password                                                         | user_type | code                                                             | timeOfCode          |
+--------+-----------------------+------------------------------------------------------------------+-----------+------------------------------------------------------------------+---------------------+
|     11 | zahramm@esms.org.uk | ed02457b5c41d964dbd2f2a609d63fe1bb7528dbe55e1abf5b52c249cd735707 | pending   | 063bc00012371f6ead4bc2bbdb564116e7025c86b0e202e9080e00d467fbece8 | 2022-03-18 20:27:38 |
+--------+-----------------------+------------------------------------------------------------------+-----------+------------------------------------------------------------------+---------------------+
```

Here, we can see that the account 'zahramm@esms.org.uk' has been registered, but its user_type is still set to 'pending', meaning that it has never been verified by its owner logging in with the verification code that they were emailed.

In case they lose the code, or it expires, they should be able to re-register (only when user_type is pending, as this is when the account hasn't been verified)

Try and re-register:





User is able to re-register and receives a new code

Check database:

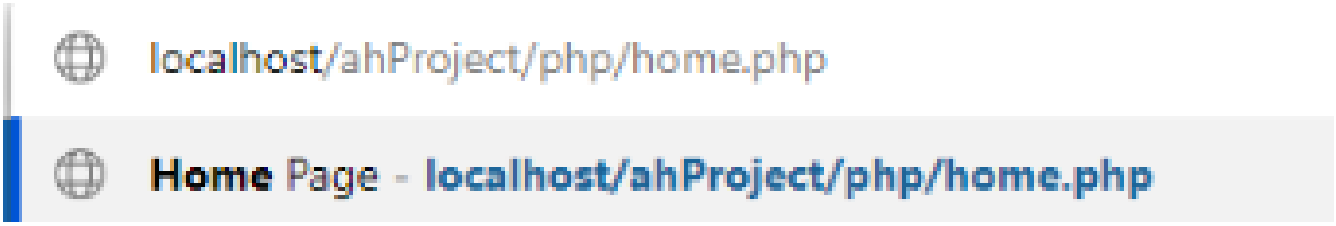
```
mysql> SELECT * FROM users WHERE email = 'zahramm@esms.org.uk';
+--------+-----------------------+------------------------------------------------------------------+-----------+------------------------------------------------------------------+---------------------+
| userID | email                 | password                                                         | user_type | code                                                             | timeOfCode          |
+--------+-----------------------+------------------------------------------------------------------+-----------+------------------------------------------------------------------+---------------------+
|     11 | zahramm@esms.org.uk | 8c0e24f73bf5fe793dc45e5f90b5a9682c0cb345f524ddef77ab2859b9352d52 | pending   | 27ff6a382592f7defd3cf2cc203d66c2d253667537184a580e36790518b0c4c6 | 2022-03-18 20:30:51 |
+--------+-----------------------+------------------------------------------------------------------+-----------+------------------------------------------------------------------+---------------------+
```

Can see that code and timeOfCode has been updated.

Conclusion: requirement is satisifed as if an account is never verified, a user can re-register it, in order to get sent a new verification code and reset the 5 minutes to enter it (in previous tests, it was shown that if an account has been verified (user_type NOT = 'pending') then it doesn't let the user register, but tells them that the email is already in use).

Test: requirement 17

Try to access home page from login.php by typing in its location into the URL



Since the session hasn't been set or was destroyed upon logging out, the user doesn't gain access to home.php, but is redirected back to login.php

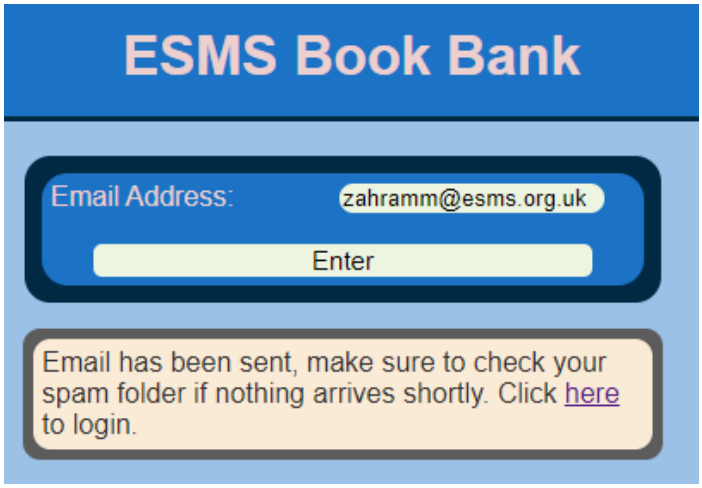
Conclusion: requirement is satisfied as a user has to be logged in to access any pages past the login, registration or forgot password pages, and can't skip them by typing in the correct URL of the next pages.

Test: requirement 18

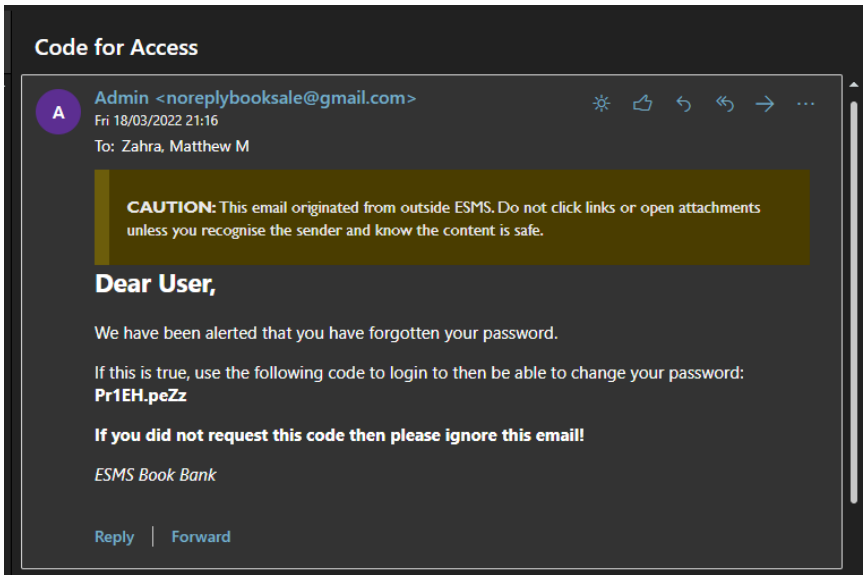
Check database for code, and timeOfCode of a user:

```
mysql> SELECT email, code, timeOfCode FROM users WHERE email = 'zahramm@esms.org.uk';
+---------------------+------------------------------------------------------------------+---------------------+
| email               | code                                                             | timeOfCode          |
+---------------------+------------------------------------------------------------------+---------------------+
| zahramm@esms.org.uk | 5271cd887e47d3e9a15d27fec0d5487ac43aad0ae88a0b4bb94526680aedb218 | 2022-03-18 21:13:28 |
+---------------------+------------------------------------------------------------------+---------------------+
```

Then, fill out forgot password form:



New code gets sent:



Check database:

```
mysql> SELECT email, code, timeOfCode FROM users WHERE email = 'zahramm@esms.org.uk';
+---------------------+------------------------------------------------------------------+---------------------+
| email               | code                                                             | timeOfCode          |
+---------------------+------------------------------------------------------------------+---------------------+
| zahramm@esms.org.uk | 3529ca3299954c4010a01a831440175171ae91413f242c5b4012f48e17ca12a7 | 2022-03-18 21:16:30 |
+---------------------+------------------------------------------------------------------+---------------------+
```

Can see that code and timeOfCode have been updated – the user is able to successfully login with the code that they have been emailed.

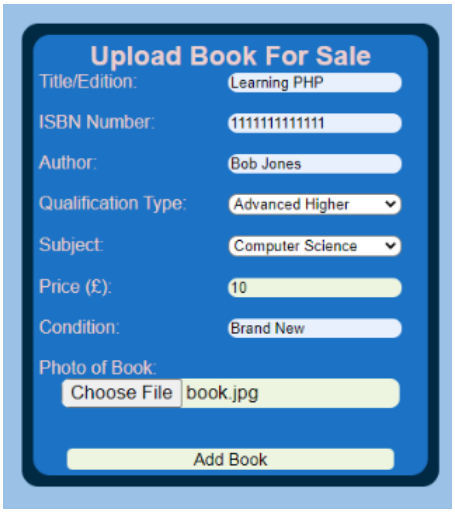
Conclusion: requirement satisfied as when a user uses the forgot password form, a new code is sent to them, and the new code and its time of issue are updated in their record in the database

Test: requirement 19

Check books table in database:

```
mysql> SELECT title FROM users, books WHERE users.userID = books.userID AND email='zahramm@esms.org.uk';
Empty set (0.00 sec)
```

User fills in valid details for a book:

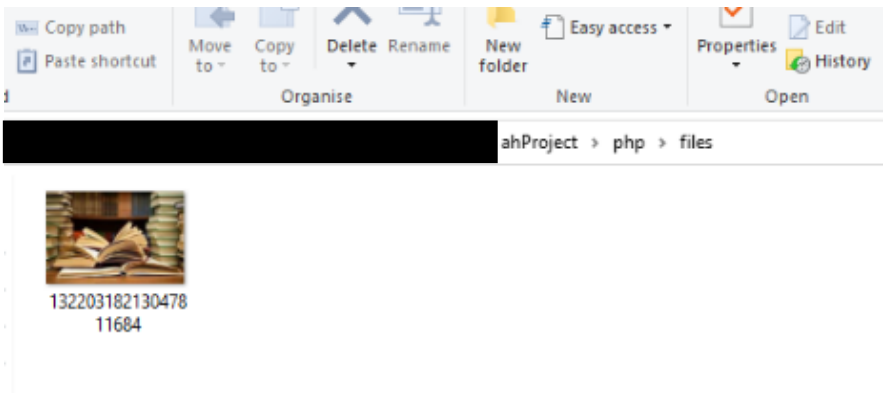


Submits the form, and as seen in previous tests, book appears in table:

Check database to see if entry has been uploaded (only title and photo selected for ease):

```
mysql> SELECT title,photo FROM users, books WHERE users.userID = books.userID AND email='zahramm@esms.org.uk';
+--------------+-------------------------+
| title        | photo                   |
+--------------+-------------------------+
| Learning PHP | 13220318213047811684.jpg |
```

Can see that it has been successfully uploaded. Now check files/ directory to see if image has been uploaded:



Can see that the image has been uploaded (has been given a unique name).

Conclusion: requirement satisfied as when a user uploads a book, its details get added to the database and its photo gets added to the files/ directory.

Test: Requirement 20

I will now attempt to delete the book uploaded in the previous test.

Check the delete checkbox in the table:

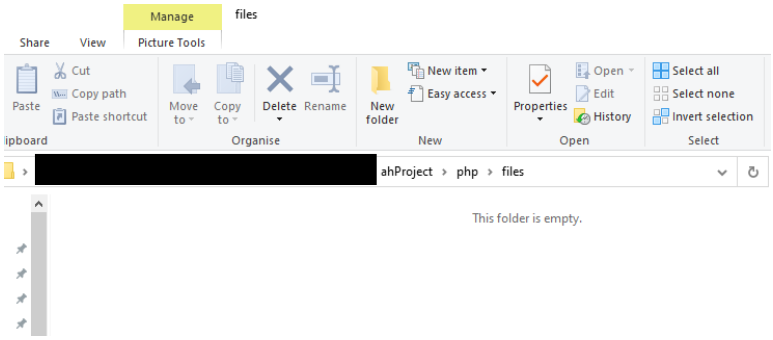
This removes this entry from the table when the form gets submitted:

If we now check the database:

```
mysql> SELECT title,photo FROM users, books WHERE users.userID = books.userID AND email='zahramm@esms.org.uk';
Empty set (0.01 sec)
```

We saw that there was a book entry here in the previous test, and it has now been removed from the database.

Checking the files/ directory:



Can see that photo has also been deleted from the files/ directory.

Conclusion: requirement satisfied as the system allows a user to remove a book, deleting its details from the database, and removing its image file (if it exists) from the files/ directory

Test: requirement 21 and requirement 22

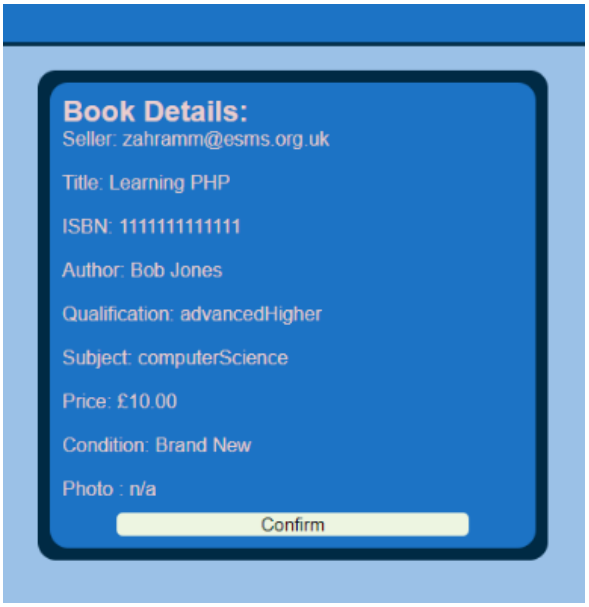
The test for these two is the exact same as when testing requirement 7, and therefore yielded the same results. To avoid repetition, the screenshots for these tests have not been included.

Conclusion: both requirements are satisifed as the system correctly displays all the right books to a user, both with and without the use of a filter.

Test: requirement 23

User selects book from table by clicking on the row:

This takes them to a buyBook.php page:



Here we can see that the information displayed is correct as it matches the information of the book that the user selected.

The URL of the page is:

localhost/ahProject/php/buyBook.php?seller=zahramm@esms.org.uk&title=Learning%20PHP&ISBN=1111111111111&author=Bob%20Jones&qualification=advancedHigher&subject=computerScience&price=10.00&condition=Brand%20New&photo=n/a

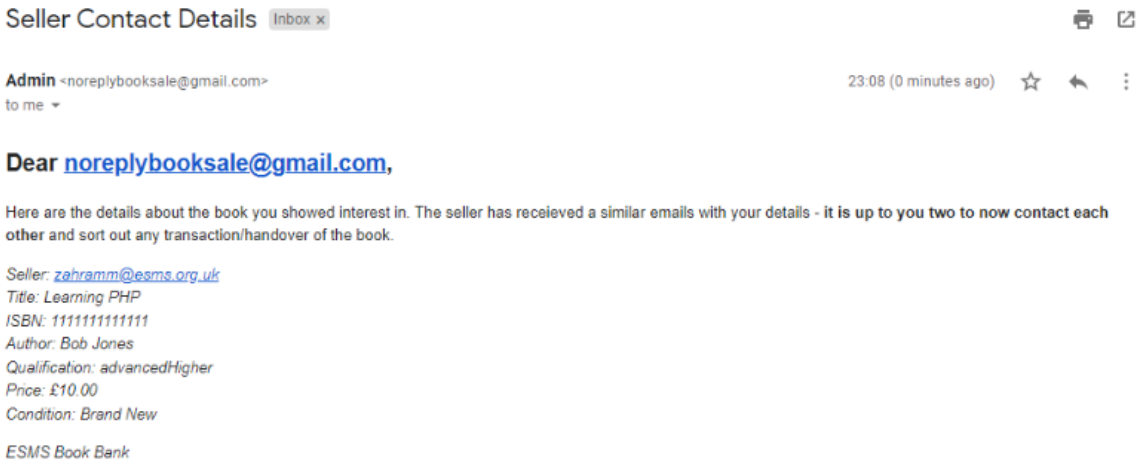
Here, we can see that this is correct, as all the key value pairs in the URL match the details of the book that the user selected.

Conclusion: requirement satisfied as when a user selects a book from the browse table, they get taken to a buyBook.php page which displays the correct information and contains the correct key value pairs in the URL.

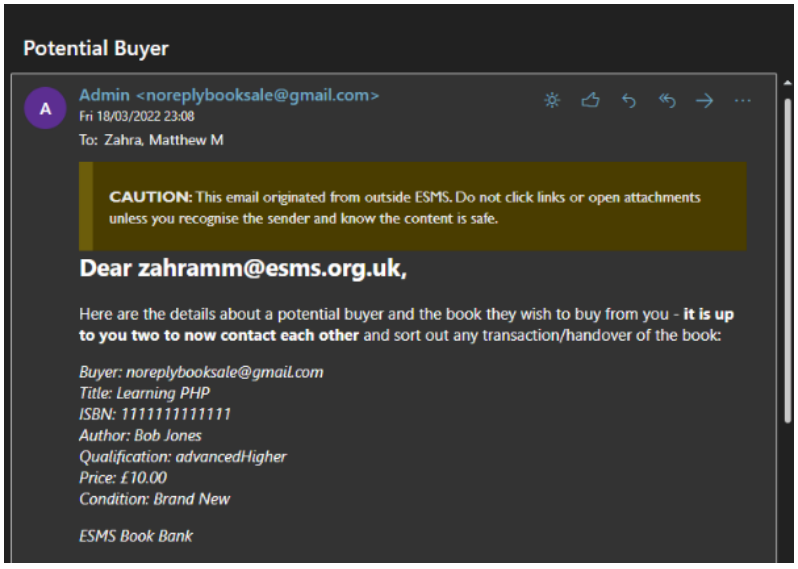
Test: requirement 24

Clicking confirm on the buyBook.php page from the previous test sends the two following emails:

To the buyer:



To the seller:



We can see that both parties received an email with the correct information.

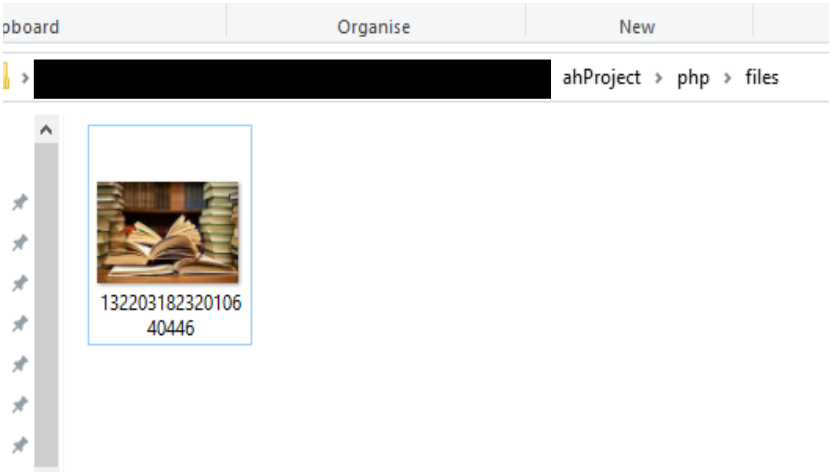
Conclusion: requirement satisfied as when a buyer clicks 'Confirm' on a buyBook.php page, both the buyer and the seller receive an email with the correct details of the other party and the book that the buyer selected.

Test: requirement 25

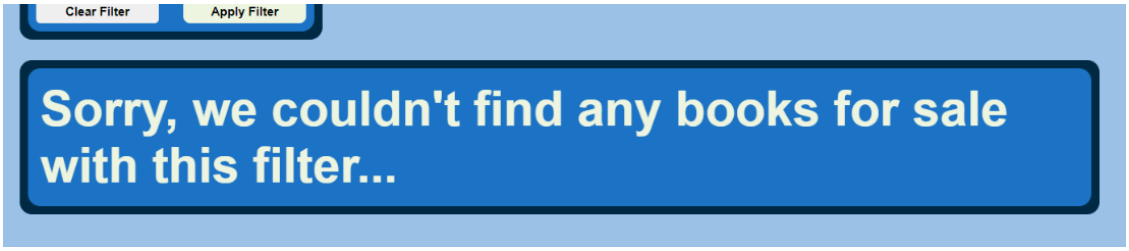
View browse books as an admin:

Show that this book exists in database, and that photo exists in files/ directory:

```
mysql> SELECT title, photo FROM books, users WHERE users.userID = books.userID AND email = 'zahramm@esms.org.uk';
+--------------+--------------------------+
| title        | photo                    |
+--------------+--------------------------+
| Learning PHP | 13220318232010640446.jpg |
+--------------+--------------------------+
```

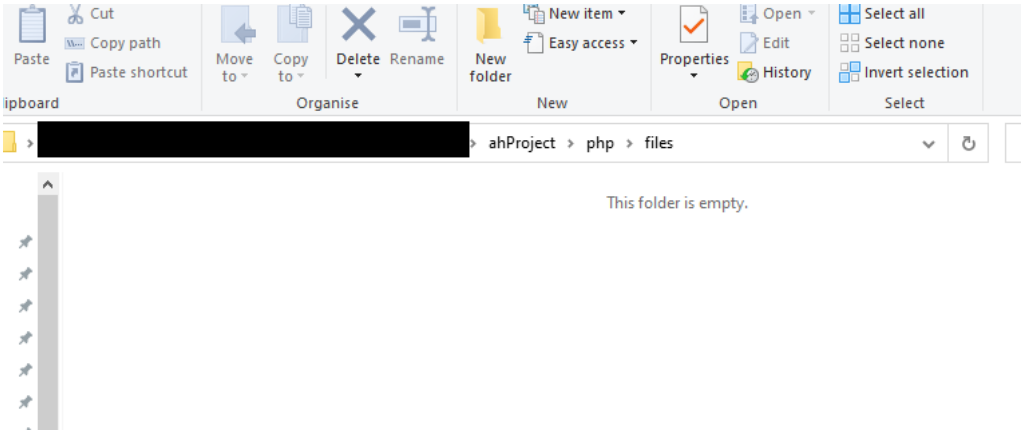


Admin then checks checkbox in delete column in the table, and submits the form. Checking the table:



Checking database and files/ directory:

```
mysql> SELECT title, photo FROM books, users WHERE users.userID = books.userID AND email = 'zahramm@esms.org.uk';
Empty set (0.00 sec)
```

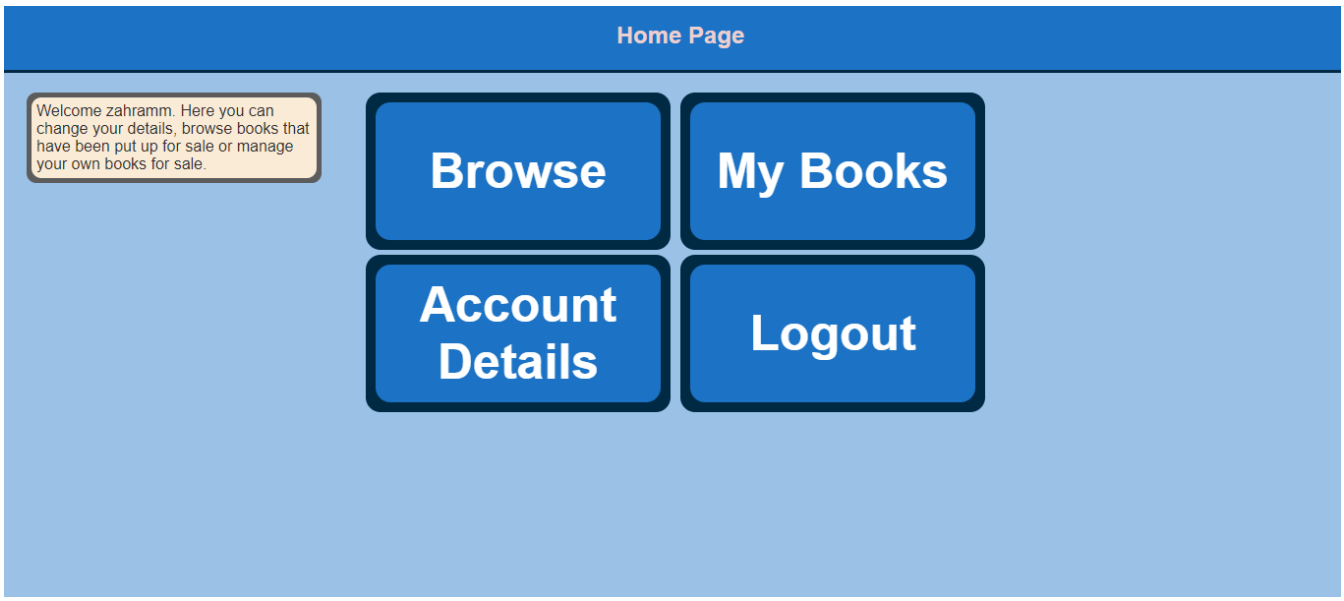


Can see that book details and photo have been removed from database and files/ directory respectively.

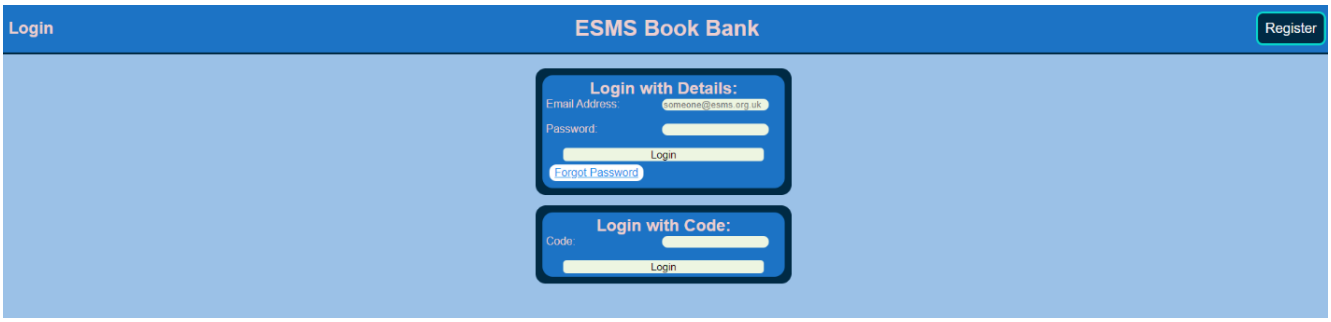
Conclusion: requirement satisfied as admins are able to delete the books of other users, which removes them from the browse table, the database, and also removes their photo from the files/ directory.

Test: requirement 26

User can be on any page:



Once they click the logout button (on home.php or on any of the navbars), it redirects them to login.php:



As shown when testing requirement 17, since the user is on login.php, all sessions are destroyed, meaning that if they type in the URL of a page other than the login, register or forgot password page, they will get redirected back to the login page – this is because, for security, it is ensured that only registered accounts can gain access to the site, as each page checks for the user's details inside the session superglobal – since login.php destroys the session variable, each page will redirect them back to login.php. The only way to restore the session variable is to login with valid details or a valid verification code.

To prove that the session gets destroyed, if I try and echo the userID which is stored in the session:

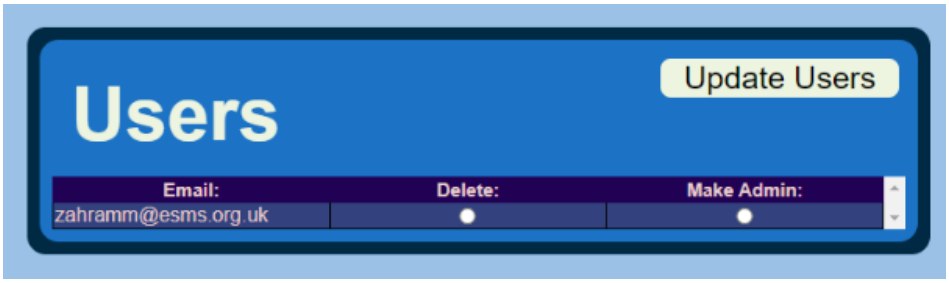
We can see that the session doesn't have a userID, as it has been destroyed.

Conclusion: requirement satisifed as when a user logs out, they are taken to login.php, which destroys the session, meaning that users can't gain access to the main part of the site without

logging in properly, as skipping the login process by typing in the URL of a page behind the preliminary pages will redirect them back to login.php as their details aren't then stored in a session variable.

Test: requirement 27

View users as an admin:

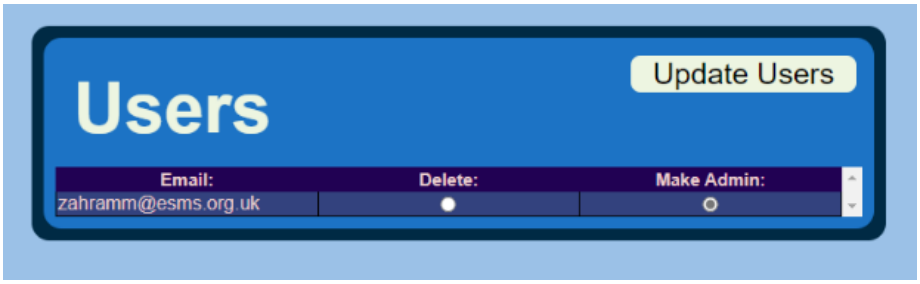


Ensure this corresponds with the database:

```
mysql> SELECT email FROM users WHERE user_type = 'user';
+---------------------+
| email               |
+---------------------+
| zahramm@esms.org.uk |
+---------------------+
1 row in set (0.00 sec)
```

This is correct.

Now, check ‘make admin’ radio button:



Submit form and then check users table and database:

We can see that zahramm@esms.org.uk has been promoted to an admin by checking the database:

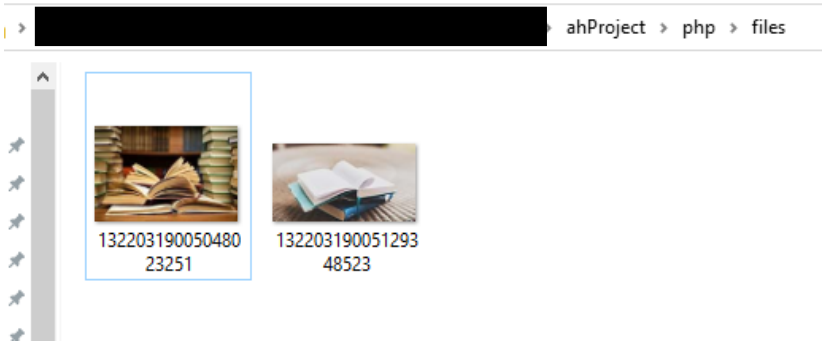
```
mysql> SELECT email FROM users WHERE user_type = 'user';
Empty set (0.00 sec)

mysql> SELECT user_type FROM users WHERE email = 'zahramm@esms.org.uk';
+-----------+
| user_type |
+-----------+
| admin     |
+-----------+
1 row in set (0.00 sec)
```

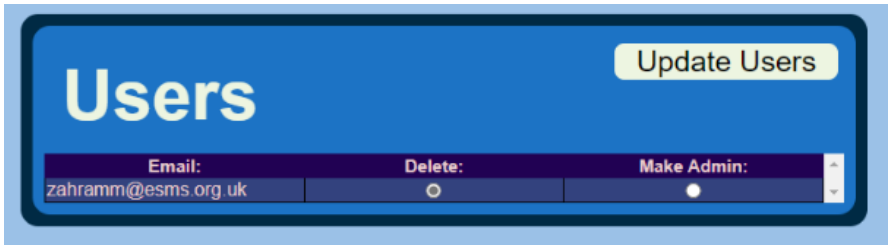
For this next part, they have been demoted back to user status.

Let's check the books that zahramm@esms.org.uk has up for sale in the database, and check the photos in the files/ directory:

```
mysql> SELECT title, photo FROM books, users WHERE users.userID = books.userID AND email = 'zahramm@esms.org.uk';
+------------------+--------------------------+
| title            | photo                    |
+------------------+--------------------------+
| Learning PHP     | 13220319005048023251.jpg |
| French Dictionary | 13220319005129348523.jpg |
+------------------+--------------------------+
2 rows in set (0.00 sec)
```



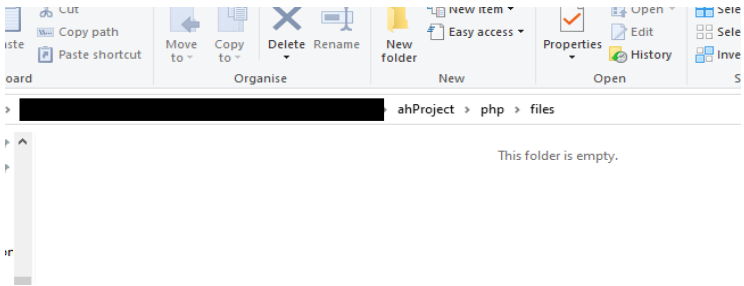
Now, select the delete radio button next to this user:



Submit form, and then check table, database and /file directory:


```
mysql> SELECT * FROM users WHERE email = 'zahramm@esms.org.uk';
Empty set (0.00 sec)
```

```
mysql> SELECT title, photo FROM books, users WHERE users.userID = books.userID AND email = 'zahramm@esms.org.uk';
Empty set (0.00 sec)
```

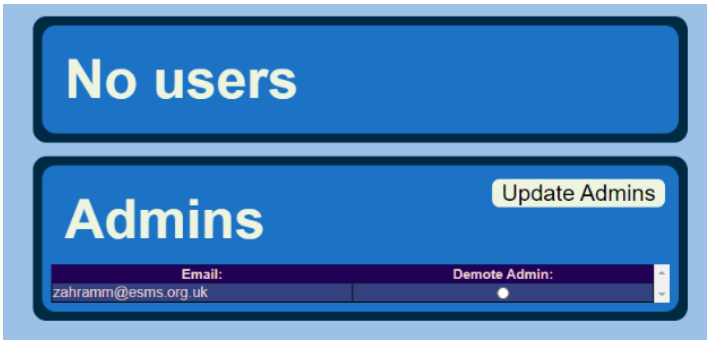


We can see that the user has been deleted, and so have all the books they had up for sale, and each image for these books has also been removed from the files/ directory.

Conclusion: requirement satisfied as admins can view all standard users, admins can promote standard users to ‘admin’ status and admins can delete standard users – this not only deletes the user, but all the books that they had up for sale, and any images the books had in the files/ directory are also removed.

Test: requirement 28

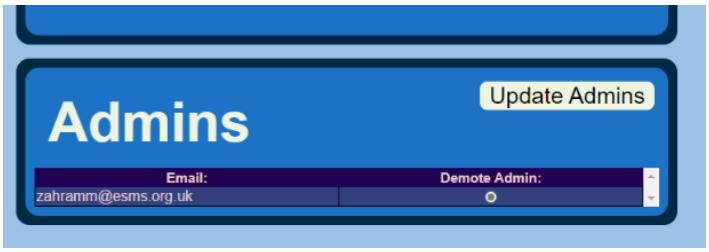
Check that admins the super_admin sees correspond to admin users in database:



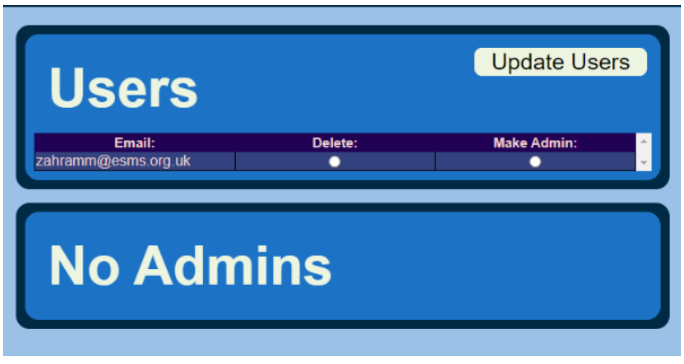

```
mysql> SELECT email FROM users WHERE user_type = 'admin';
+---------------------+
| email               |
+---------------------+
| zahramm@esms.org.uk |
+---------------------+
1 row in set (0.00 sec)
```

This is correct.

Now, check demote radio button in form:



Check table and database upon submitting the form:




```
mysql> SELECT user_type FROM users WHERE email = 'zahramm@esms.org.uk';
+-----------+
| user_type |
+-----------+
| user      |
+-----------+
1 row in set (0.00 sec)
```

We can clearly see that the super_admin is able to view all admins and demote them to ‘user’ status.

Conclusion: requirement satisfied as the super_admin can view all the current ‘admin’ users and can successfully demote any of them to ‘user’ status.

Results of persona test cases:

Test case 1:

The user started on login.php and quickly found the 'Register' button in the top right, taking them to register.php. Here they successfully filled in their details and had the email with the verification code sent to them. Following the prompted link, they were then taken back to the login page. Despite the instructions in the email and at registration, the user still tried to login with their email and password for the first time, not with the verification code. For the next version, the use of the verification code could potentially be made more intuitive by putting the 'Login with Code' and 'Login with Details' forms on different pages and once they register, they get taken to a page with only the option to use the verification code. However, after re-reading the email, they soon realised their error and were able to login. So, overall, this test was successful.

Test case 2:

The user immediately went to the 'Forgot Password' link and then correctly filled in their email. They were then able to quickly extract the code from the email, and use it to login. Once they had logged in, they immediately clicked the 'Account Details' button, and were able to successfully change their password. Overall, this test was a success as the speed that the user completed the task demonstrated the intuitiveness of this function.

Test case 3:

The user was able to quickly navigate to the 'My Books' page by use of the 'My Books' button on the home page. Here, they began filling out the form to upload a book. They commented that they thought it was strange that the 'condition' input for the book was a text field, and not a drop down menu. Perhaps, for the next version, this could be turned into a drop down menu in order to standardise the ways in which people describe the conditions of the books they put up for sale (e.g. brand new, slightly used etc…) rather than allowing ambiguous descriptions. While filling in the form, the user tried to use a photo for the book that was not of the correct type (not .png, .jpg or .jpeg) and they commented that the error message 'File is not an image' wasn't helpful enough, as it didn't specify the acceptable file types – this could be more specific and indicate the valid file types in the next version. After completing the form, they commented that they thought it was strange to insist on an image for each book – even though this is not the case, as a book can be uploaded without an image, it is not indicated anywhere which fields are necessary. Perhaps, for the next version, all the necessary fields could be indciated with an asterisk. Despite some ambiguity, the test was a success as the user was able to successfully upload a book for sale.

Test case 4:

The user was able to quickly click the radio button in the delete column next to the book entry that they had just uploaded on the 'My Books' page. They then clicked the 'Update Books' button, deleting the book entry. This test was a success as it showed that deleting a book was intuitive and easy to do.

Test case 5:

The user was able to successfully use the navbar to navigate to the browse page, and when asked to find a French Dictionary for Higher French, they were able to successfully apply the filter to find the desired book. They however were unsure how to procede from here, as even though there were instructions in the info bubble in the top right, telling them to click on a book entry, these were not very eye catching – these instructions could be made more obvious in a future version of the system. Once they spotted this, they clicked on the correct entry to be taken to the 'Buy Book' page for the book. They eventually clicked 'Confirm', once they had read the contents of the info bubble in the top right, explaining what this did – they did comment that 'Confirm' was rather misleading, as it suggested they would be making some sort of payment – this could perhaps be re-labled to something like 'Send Details' in the next version of the system. Once the email came through, the user was able to successfully extract the details of the seller so that they could contact them. Overall this test was a success as the user was able to use the filter to find a book, click on it, and then confirm that they would like to send the email with the details to both them and the seller.

Test case 6:

The user, now with an admin account, navigated to the browse page, and quickly saw the new 'Delete' column for each book entry, and the new 'Update Books' button at the top of the books table. They used the filter to find the specific book and then clicked the 'Delete' radio button and submitted the form with the 'Update Books' button, deleting the book entry. This test was a success as it was very intuitive for the admin user to find and then delete another user's book entry.

Test case 7:

The user immediately spotted the new 'Users' button and used this to navigate to the correct page. Since the list of users is sorted alphabetically, the user was able to find the two specific users to delete and promote. They clicked on the respective radio buttons for both of these users and then submitted the form. This test was a success as the user was able to easily complete the task, finding the new page quickly and then promoting and deleting the correct users.

Test case 8:

The user, now with a super_admin account, again navigated to the 'Users' page, and saw the new 'admins' table. Again, due to the admins being sorted alphabetically, they were able to quickly locate the correct admin user, click the 'demote' radio button and then submit the form with the 'Update Admins' button. This test was a success as the user was able to quickly complete the task, showing that this bit of the system is very usable.

Evaluation

Fitness for Purpose:

In order to be fit for purpose, each of the requirements identified in the analysis stage would need to have been met.

Requirements:

- End user requirements:
    - Before logging in:
        29. Users should be able to register an account with a unique, unverified '…@esms.org.uk' email address, and a password that is > 5 and <= 256 characters in length
        30. Users should be able to login with their email and password (and verification code if it is their first time logging in)
        31. Users should be able to select 'forgot password' and then use the verification code they have been emailed to login
    - Once Logged in:
        32. A user will be able to navigate via the home page/nav bar of each other page to a different page
        33. A user will be able to change their password
        34. A user should be able to view all their books they have put up for sale, add new ones and delete any of their own books that are already up
        35. Users should be able to browse books for sale (other than their own) – applying a filter if they want to
        36. Users should be able to select a book to buy to be taken to a confirmation page
        37. Users should be able to confirm that they would like to buy a book – this should send an email to them with the details of the book and the seller
        38. Users should be able to logout
        39. Admin users should be able to delete other users' books from the browse table
        40. Admin users should be able to view, delete and promote standard users
        41. The super_admin should be able to view and demote all admin users

- Functional requirements:
        42. The system should appropriately validate all inputs:
            1. Restricted choice for subjects/qualifications for books (only client side)
            2. Length checks for emails, book titles/editions/condition – all <= 60 characters; passwords which are > 5 characters and <= 256 characters; ISBN numbers are 10 or 13 digits long

3. Ensure that the price is a positive number, less than 99999999.99 and is no more accurate than 2 decimal places
4. Ensure all emails (at registration) are unique and of the correct format – '…@esms.org.uk'
5. Presence checks for emails and passwords and necessary fields (such as the subject) when putting a book up for sale
6. Escape all form inputs to avoid SQL injection
7. Photo of book is of correct filetype and size (.jpg, .jpeg or .png and <= 3Mb)
8. Validate any verification codes by checking with the database (ensure they are correct and have been entered within 5 minutes of being issued)
9. Ensuring that any login/registration details are valid etc…
10. Validate changing of passwords by ensuring that they match

43. When a user registers, the system should put details into database (if valid)
44. If a user registers with an email address that's already been registered, but the verification code was never entered (so no one ever logged in), they should be allowed to register again, and their new details will be saved by the system, after validation, to the users table, and a new verification code should be sent to their email
45. Only logged in users should be able to progress past login/registration/forgot password pages as the system should store the user's details in a session to enforce this
46. If a user selects 'forgot password', the system should email them a verification code to login with and update the code in the database
47. The system should allow a user to put a new book up for sale – adding it to the database and any photos uploaded to the files/ directory
48. The system should allow a user to remove a book that they have put up for sale – removing it from the database and removing any photos from the files/ directory
49. The system should display all the books that are for sale to each user (before a filter is applied)
50. The system should display only the books that fall under the applied filter (once the user selects a filter)
51. The system should allow users to select a book to buy and then take them to a confirmation page with the data of the corresponding book on it and in the URL
52. Once a user confirms they would like to buy a book, the system should send an email to the seller with the buyer's details and a confirmation email to the buyer
53. The system should allow admin users to delete other users' books from the site/database, also deleting the images of the books from the files/ directory

54. The system should let users logout – once this happens, all session should terminate in order to maintain security
55. The system should allow admin users to see/delete/promote standard users
56. The system should allow the super_admin to see/demote admin users

I created a test for each requirement in my test plan, and carried each of these out, providing evidence in my testing section. Each test was successful, showing that the project satisfied each end-user and functional requirement. This means that users are able to successfully register, ask for a new verification code with 'Forgot Password', login, navigate to each page, view and manage their own books up for sale, view other books for sale with the option of using a filter, click on a book and then confirm they wish to buy the book to send the email to them and the seller with each other's and the book's details, change their password and logout. Admin users can additionally manage users, being able to view, promote and delete users and also delete the individual book entries of other users. And the super_admin can also view and demote admin users.

Following the results of the persona and test cases, it was clear that the site was usable, but a few minor changes to the next version could better improve the user experience: put the 'Login with Code' on a different page to 'Login with Details' to make it more intuitive for first time users, so they don't get confused between which to use for the first time; make the 'condition' input for uploading a book a drop down menu instead of a text input to better standardise the way people describe the conditions of their books; make it clear that a photo is not necessary when uploading a book for sale, but all the other fields are; specify the valid file types when a user is uploading a book; make the info bubble indicating to click on a book entry on the browse page easier to spot; change the 'Confirm' button to something more intuitive, such as 'Send Details', so it is clear that clicking this button doesn't activate a transaction. These suggestions were minor changes and the person attempting the test cases was able to complete each of them successfully, meaning that the system is useable, user-friendly, overall mostly intuitive and fit for purpose.

Since all the requirements tests were successful, my project fulfils all the end-user and functional-requirements, so therefore is fit for purpose.

Maintainability:

In order to make my project as maintainable as possible, I tried to modularise my code where possible, so that if changes needed to be made, they would only need to be made in one place. I used a single external CSS file for the main CSS, and a second external CSS file for the media queries, meaning that any design changes would be easier to carry out as each page referenced both these two files. I also made a few of my PHP files modular and imported them in several places. For example, I made load.php a general function to load any page I passed into it. As I often moved the user from page to page, it made sense to do this and just import the file when it is needed, rather than write the same code in many different places. I also made email.php a file with several modular functions inside. The email function was here as all my emails were sent using the same code, and if I ever needed to make changes to it (such as changing the sender details, SMTP server or the port) then I could just make changes to the one file. This file also contained a function to generate the verification code, which was used in a few different places – this was so that if I ever decided to change how the verification codes were generated, I only had to edit here. I also made connectDB.php modular, so that it could be imported many times, as each time I connected to the database, I used the same details. And finally, I made the pageHeader.php and entrancePageHeader.php files to be used across many pages (I just had to declare what the title of each page was), so that I just import these when needed – this is again so that if I needed to make any changes to these aspects, I would only need to change the one file as they were consistent over many pages.

I however, used the same code for the red error boxes (triggered when server-side validation found an error) on many different pages and I could have made my code more maintainable by putting this code into a modular file and then importing it when needed, instead of copy and pasting the code each time it was used, as if I need to change this aspect of the system currently I would need to manually change it in every single file that it gets used.

The code below is the block that has been repeated across many pages, and could have been put into a file to import when needed:

```
1. <!--Only display div if input errors exist-->
2.          <div class='errorsBox' id='filterErrorsBox' <?php if
   (!isset($errors) or empty($errors)){echo "style='display: none';";}?>>
3.          <div class='errorsContainer' <?php if (!isset($errors) or
   empty($errors)){echo "style='display: none';";}?>>
4.          <?php
5.              if (isset($errors) and !(empty($errors)))
6.              {
7.                  foreach ($errors as $error)
8.                  {
9.                      echo "<p>$error</p>";
10.                     echo '<br>';
11.                 }
12.             }
13.         ?>
14.         </div>
15.         </div>
16.
```

I could have also made some of my validation more modular. A lot of the server-side validation is very similar e.g. ensuring things are not NULL, that strings are <= 60 characters etc… I could have made a PHP file with a validate function that could have been imported several times instead of writing out very similar code in multiple places.

To also make my project more maintainable, I made sure to use sensible variable names, white space between different sections of code, indentation to preserve the structure of loops and if statements (where appropriate) and lots of internal commentary to explain the purpose of each bit of code or any tricky bits of logic.

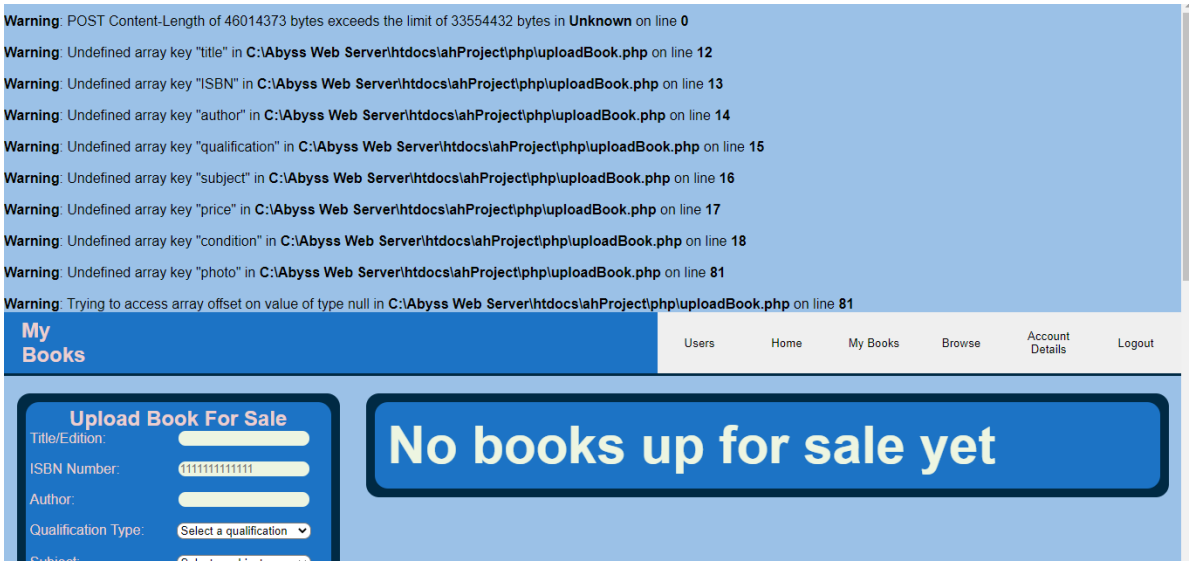
The extract below is from uploadBook.php and clearly demonstrates good use of whitespace, indentation and sensible variable names:

```
1. #check if a file has been uploaded (this is not necessary for a book to be
   uploaded)
2. if ($fname != "")
3. {
4.     $upload = TRUE;
5.     $extension = strtolower(pathinfo($fname,PATHINFO_EXTENSION));
6.
7.     #check to see if file is actually an image
8.     if (getimagesize($_FILES['photo']['tmp_name']) == False)
9.     {
10.        $errors[] = "File is not an image";
11.    }
12.
13.    #check extension is an accepted one
14.    elseif ($extension != 'jpg' && $extension != 'png' && $extension !=
   'jpeg')
15.    {
16.        $errors[] = "File is not an accpted format (must be a .jpg,.png or
   .jpeg)";
17.    }
18.
19.    #ensures image size is <= 3 megabytes
20.    elseif ($_FILES['photo']['size'] > 3145728)
21.    {
22.        $errors[] = "Image is too large";
23.    }
24.}
25.else
26.{
27.    $upload = FALSE;
28.}
29.
```

Robustness:

My project is robust enough to be fully usable, and as seen in my implementation, I ensured that every input was validated on server-side and where possible, on the client-side and at the database end too. This ensured that my system could handle exceptional/incorrect inputs. I used the function mysqli_real_escape_string() in order to protect the system from SQL

injection. However, the system could have been made even more robust. In the current version, there's nothing to really stop other types of injection, for example, JavaScript injection. In the next version, I could use functions like htmlspecialchars() when displaying user inputs (e.g. all the book details) as it encodes special characters so that they get displayed but not interpreted, and could also use strip_tags() to remove all HTML and PHP tags from user inputs. These additional functions would make the system much more secure and robust, as it would be much less vulnerable to injection attacks. The other area I have identified as not being as robust as possible is when a user uploads a file – the default maximum file size that PHP will POST is 32Mb. If a user tries to upload an image that is > 32Mb, then the system encounters some issues. It still sets the server request method to POST and goes to the action set by the form (the page handler) but it doesn't end up posting any of the data in the form, meaning the form handler tries to access a lot of undefined variables in the POST superglobal, and throws up a lot of errors:



This however is fixed by refreshing the page, and although isn't ideal, no data or functionality is compromised. Overall, the system is more than robust enough as the server-side validation ensures that all inputs are validated appropriately and provides useful error messages where appropriate to help guide the user. In the next version, the robustness could be further improved by further protecting against injection attacks, with the addition of the functions above, and also by removing the PHP errors that appear when a file is > 32Mb and replacing them with some more user-friendly warnings.