

# **Technical side**

## **Bookish Corner Web Application**

**Yevheniia Monashko**

**Student number: 550003**

**DHI1V.Sr**

## Contents

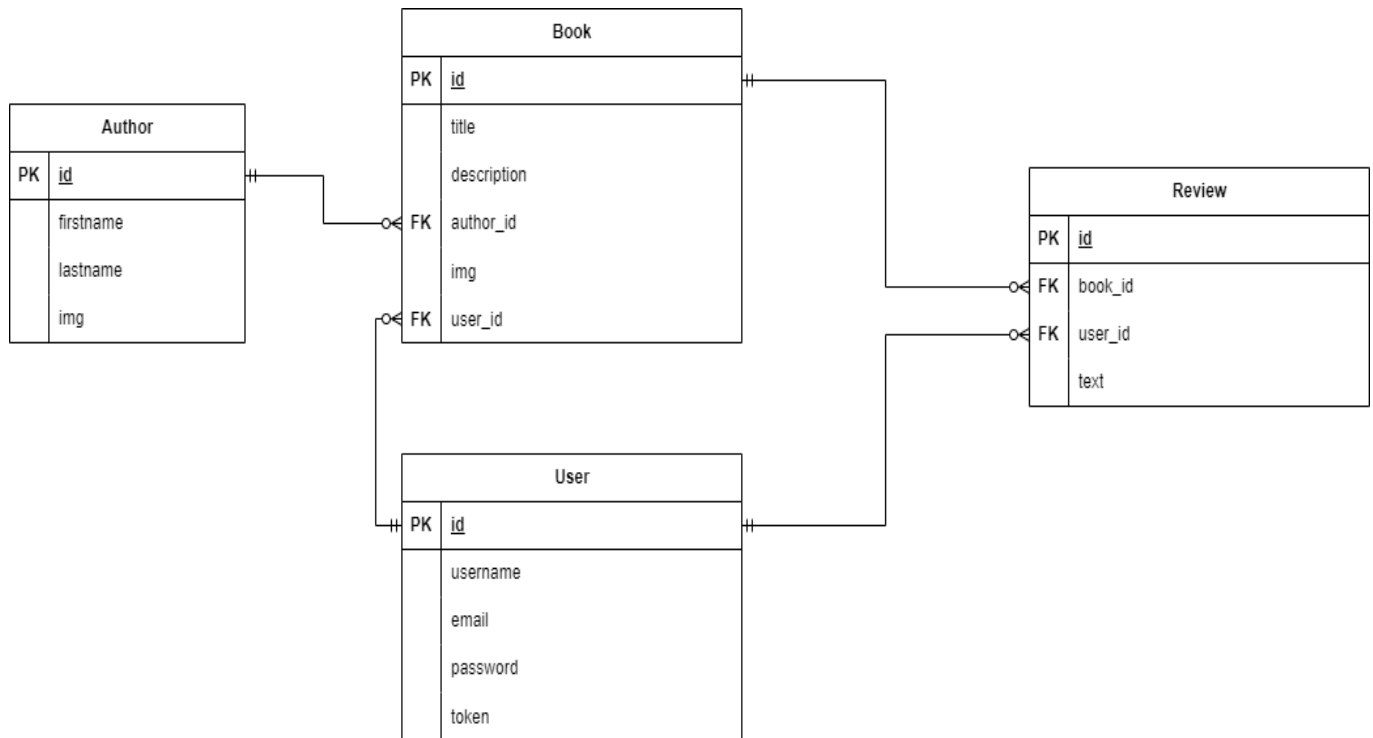
Context description .....	3
Entity-Relationship Diagram (ERD) .....	3
REST API Specification .....	5
<i>GET requests</i> .....	5
<i>POST requests</i> .....	12
<i>PUT requests</i> .....	14
<i>DELETE requests</i> .....	16
Sequence diagrams .....	17
Test cases .....	19
Test report .....	27
Justification for login/registration process .....	38

## Context description

The web application "Bookish Corner" serves as a platform connecting readers with a diverse list of authors and their literary works. The primary theme of this web application is reader engagement and literature exploration. Bookish Corner will be created to achieve the following key objectives:

- Provide a user-friendly platform that enables users to comfortably explore a diverse selection of authors and their books, promoting a culture of reading.
- Ensure detailed information for each book, including cover images, titles, and short descriptions to show the essence of the book, along with user reviews to make easier book discovery based on individual interests.
- Save readers significant time by allowing them to view reviews for a particular book before diving into it, ensuring a satisfying reading experience and minimizing the chances of choosing an unsuitable book due to information in the description, which cannot provide enough information to show pros and cons presented in a certain book.
- Allow users to create an account on our website or log into their existing accounts. Authenticated users can view and edit their user pages. One significant benefit for authenticated users is that they can add new books, as well as edit and delete existing books they have added. Furthermore, they can add authors to the website. This functionality helps users to expand the platform by adding books, thereby enlarging the collection of books for their favourite authors. Additionally, the ability to add authors allows users to widen the selection of authors and extend the community of the website by attracting more people who may be interested in the newly added authors.
- Engage authenticated users by allowing them to add reviews, and then edit and delete all reviews they have left. Each review contains a clickable title of the book it refers to, enabling easy navigation to the book under which the user's review was left. This fosters a sense of community among readers and assists others in book selection.

## Entity-Relationship Diagram (ERD)



**Note:** As specified by the teacher, if the project code doesn't contain classes, it should be an Entity-Relationship Diagram (ERD), not a class diagram. That's the justification for why this type of diagram is used.

## Diagram description

**Each user** has a username, email, and password, which are defined during account creation. Each user entity has a unique ID as the **primary key**. Additionally, **each user entity** contains a token that is generated automatically for each user and updated each time the user logs into their account. This attribute is used to track the authorized/unauthorized status of the user. **Each user** can add **many books** and write **many reviews**.

**Each book** contains a title, description, and image. **Each book** also has an **author**, with the data retrieved by the author\_id stored as a **foreign key** in each book entity. Additionally, **each book entity** has a user\_id as a **foreign key**, representing the **user who added the book**. Each book entity contains a unique ID as the **primary key**.

**Each review** contains a unique ID as the **primary key**. Additionally, **each review** contains a user\_id as a **foreign key**, corresponding to the **user who added the review**, and a book\_id as a **foreign key**, relating to the **specific book** for which the review was left.

**Each author** contains a unique ID as the **primary key**, along with a first name, last name, and image. **Each author entity** can be related to **many books**.

# REST API Specification

This chapter illustrates all GET, POST, PUT, and DELETE requests for the Bookish Corner web application. These requests fulfil the requirement that at least two resources should have full CRUD (Create, Read, Update, Delete) functionality. In the Bookish Corner web application, the book and review resources implement this full functionality. Additionally, at least three backend endpoints use query parameters, and at least three backend endpoints use path parameters, which also correspond to the requirements.

## GET requests

GET /books			
Retrieves all books along with their author details.			
Parameters:	Name	Type	Description
No parameters			
Responses:	Code	Description / example if successful	
	200 OK	List of all books with author information. [ { "id": 1, "title": "Nineteen Eighty-Four (1984)", "description": "George Orwell's novel \"1984\" is the most powerful dystopia of the 20th century...", "author_id": 1, "author_name": "George", "author_lastname": "Orwell", "img": "1717432622080.jpg", "user_id": null }, { "id": 2, "title": "Animal Farm", "description": "Animal Farm is a satirical allegorical novella, in the form of a beast fable...", "author_id": 1, "author_name": "George", "author_lastname": "Orwell", "img": "1717494954402.jpg", "user_id": null }, { "id": 3, "title": "Burmese days", "description": "\"Burmese Days\" by George Orwell is a gripping novel set in 1920s Burma...", } ]	

		<pre> "author_id": 1, "author_name": "George", "author_lastname": "Orwell", "img": "1717495108332.jpg", "user_id": null }, ... ] </pre>
	500 Internal Server Error	An error occurred on the server.

GET /book/{id}			
Retrieves details of a single book based on its ID and displays it on book page.			
Parameters:	Name	Type	Description
	id*	path	The ID of the book to retrieve.
Responses:	Code	Description / example if successful	
	202 OK	Book details with author information <pre> {   "id": 1,   "title": " Nineteen Eighty-Four (1984)",   "description": "George Orwell's novel \"1984\" is the most powerful dystopia of the 20th century...",   "author_id": 1,   "author_name": "George",   "author_lastname": "Orwell",   "img": "1717432622080.jpg.",   "user_id": 1 } </pre>	
	404 Not Found	Book not found.	

GET / author-books/{id}			
Retrieves all books written by a specific author based on the author's ID and displays them on the author books page.			
Parameters:	Name	Type	Description
	id*	path	The ID of the author.

Responses:	Code	Description / example if successful
	200 OK	<p>List of books written by the author.</p> <pre>[   {     "id": 1,     "title": "Nineteen Eighty-Four (1984)",     "description": "George Orwell's novel \"1984\" is the most powerful dystopia of the 20th century...",     "author_id": 1,     "img": "1717432622080.jpg",     "user_id": 1   },   {     "id": 2,     "title": "Animal Farm",     "description": "Animal Farm is a satirical allegorical novella, in the form of a beast fable...",     "author_id": 1,     "img": "1717494954402.jpg",     "user_id": 1   },   {     "id": 3,     "title": "Burmese days",     "description": "\"Burmese Days\" by George Orwell is a gripping novel set in 1920s Burma...",     "author_id": 1,     "img": "1717495108332.jpg",     "user_id": 1   },   ... ]</pre>
	404 Not Found	No books found for this author

GET /user-books/{id}			
Retrieves books added by a specific user based on user ID and displays books on user page.			
Parameters:	Name	Type	Description
	id*	path	The ID of the user.
Responses:	Code	Description / example if successful	
	200 OK	<p>List of books added by the specified user.</p> <pre>[</pre>	

		<pre>{   "id": 1,   "title": "Nineteen Eighty-Four (1984)",   "description": "George Orwell's novel \"1984\" is the most powerful dystopia of the 20th century...",   "author_id": 1,   "img": "1717432622080.jpg",   "user_id": }, {   "id": 5,   "title": "Harry Potter and the Philosopher's Stone",   "description": "\"Harry Potter and the Philosopher's Stone\" by J.K. Rowling is the enchanting beginning...",   "author_id": 2,   "img": "1717495916154.jpg",   "user_id": 1 }, ]</pre>
	500 Internal Server Error	An error occurred on the server.

GET /search-books? q={title}			
Searches for books by title and displays the book(s) based on the search request to the user.			
Parameters:	Name	Type	Description
	q*	query	The title or part of the title of the book to search for.
Responses:	Code	Description / example if successful	
	200 OK	Book or list of books matching the search query. <pre>{   "id": 1,   "title": "Nineteen Eighty-Four (1984)",   "description": "George Orwell's novel \"1984\" is the most powerful dystopia of the 20th century...",   "author_id": 1,   "img": "1717432622080.jpg",   "user_id": 1 }</pre>	
	400 Bad Request	Book title is required.	



GET /authors			
The request retrieves a list of authors available on the website and shows a page with authors to the user.			
Parameters:	Name	Type	Description
No parameters			
Responses:	Code	Description / example if successful	
	200 OK	The list of authors successfully retrieved. <pre>[   {     "id": 1,     "firstname": "George",     "lastname": "Orwell",     "img": "1717348083906.jpg"   },   {     "id": 2,     "firstname": "Joanne",     "lastname": "Rowling",     "img": "1717348175191.jpg"   },   ... ]</pre>	
	500 Internal Server Error	An error occurred on the server.	

GET /author/{id}			
Retrieves details of a single author based on their ID.			
Parameters:	Name	Type	Description
	id*	path	The ID of the author to retrieve.
Responses:	Code	Description / example if successful	
	200 OK	Author details successfully retrieved. <pre>{   "id": 1,   "firstname": "George",   "lastname": "Orwell",   "img": "1717348083906.jpg" }</pre>	
	404 Not Found	Author not found.	

GET	/search-authors?q={name}		
Searches for authors by either first name, last name, or full name and displays the author(s) based on search request to the user.			
Parameters:	Name	Type	Description
	q*	query	The name or part of the name of the author to search for. It can be the first name, last name, or full name.
Responses:	Code	Description / example if successful	
	200 OK	Author or authors retrieved successful. { "id": 1, "firstname": "George", "lastname": "Orwell", "img": "1717348083906.jpg" }	
	400 Bad Request	Author name is required.	

GET		/reviews/{bookId}	
Retrieves reviews for a specific book based on the book ID and displays it on book page.			
Parameters:	Name	Type	Description
	bookId*	path	The ID of the book to retrieve reviews for.
	page	query	The page number for pagination (default is 1).
	limit	query	The number of reviews per page (default is 10).
Responses:	Code	Description / example if successful	
	200 OK	List of reviews for the specified book, along with pagination information received successfully. { "reviews": [ {"text": "Great book!", "username": "user1"}, {"text": "Loved it!", "username": "user2"} ], "totalPages": 5, "currentPage": 1 }	
	400 Bad Request	Book ID is required.	

GET		/reviews/user/{id}	
Retrieves reviews written by a specific user based on user ID and displays on user page.			
Parameters:	Name	Type	Description
	id*	path	The ID of the user to retrieve reviews for.
Responses:	Code	Description / example if successful	
	200 OK	List of reviews written by the specified user retrieved successfully. [ {"id": 1, "text": "Great book!", "bookTitle": "Book Title 1", "bookId": 1 }, {"id": 2, "text": "Not bad", "bookTitle": "Book Title 2", "bookId": 2 } ]	
	400 Bad Request	User ID is required.	

GET	/user/{email}		
Retrieves user information based on email.			
Parameters:	Name	Type	Description
	email*	path	The email of the user to retrieve.
Responses:	Code	Description / example if successful	
	200 OK	User information retrieved successfully. { "id": 1, "username": "username", "email": "user@example.com", "password": "password", "token": "user_token" }	
	404 Not Found	User not found.	

GET		/user/id/{id}	
Retrieves user information based on user ID.			
Parameters:	Name	Type	Description
	id*	path	The ID of the user to retrieve.
Responses:	Code	Description / example if successful	
	200 OK	User information retrieved successfully. <pre>{   "id": 1,   "username": "username",   "email": "user@example.com",   "password": "password",   "token": "user_token" }</pre>	
	404 Not Found	User not found.	

GET		/login	
Logs in a user and updates the user token.			
Parameters:	Name	Type	Description
	email*	query	The email of the user.
	password*	query	The password of the user.
Responses:	Code	Description / example if successful	
	200 OK	Login successful. { "message": "Login successful", "token": "generated_token", "id": 1 }	
	401 Unauthorized	Invalid password.	
	404 Not Found	User not found.	

## POST requests

POST	/books		
Adds a new book to the database.			
Parameters:	Name	Type	Description

Add a * to the name of required parameters.	title*	body	The title of the book.
	description*	body	The description of the book.
	author_id *	body	The ID of the author of the book.
	user_id*	body	The ID of the user adding the book.
	img	body	The image of the book cover.
<b>Responses:</b>	<b>Code</b>	<b>Description / example if successful</b>	
	201 Created	Book added successfully with book details. <pre>{   "message": "Book added successfully!",   "book": {     "title": "The Great Gatsby",     "description": "A novel set in the Jazz Age...",     "author_id": 11,     "user_id": 1,     "img": "great_gatsby_cover.jpg"   } }</pre>	
	400 Bad Request	Missing required fields.	

POST		/authors	
Adds a new author to the system.			
Parameters:	Name	Type	Description
	firstname*	body	The first name of the author.
	lastname*	body	The last name of the author.
	img	body	The image of the author.
Responses:	Code	Description / example if successful	
	201 Created	Author added successfully. { "message": "Author added successfully!", "author": { "id": 11, "firstname": "Arthur", "lastname": "Conan Doyle", "img": "1717500000000.jpg" } }	
	400 Bad Request	Missing required fields.	

POST	/reviews		
Adds a new review to the system.			
Parameters:	Name	Type	Description
	text*	body	The text of the review.

	book_id*	body	The ID of the book to which the review will be added.
	user_id*	body	The ID of the user, who is writing the review.
<b>Responses:</b>	<b>Code</b>	<b>Description / example if successful</b>	
	201 Created	Review added successfully. { "message": "Review added successfully!", "review": { "text": "Great book!", "book_id": 1, "user_id": 1 } }	
	400 Bad Request	Missing required fields.	

POST		/register	
Creates a new user in system.			
Parameters:	Name	Type	Description
Add a * to the name of required parameters.	username*	body	The username of the user.
	email*	body	The email address of the user.
	password*	body	The password of the user.
Responses:	Code	Description / example if successful	
	201 Created	User created successfully. { "message": "User created successfully!", "token": "generated_token", "id": 1 }	
	400 Bad Request	User already exists. OR Missing required fields.	

### PUT requests

PUT		/book/{id}	
Updates details of an existing book.			
Parameters:	Name	Type	Description
Add a * to the name of required parameters.	Id*	path	The ID of the book to edit.
	title*	body	The new title of the book.
	description*	body	The new description of the book.
	author_id*	body	The new ID of the author of the book.
	img	body	The new image of the book cover (optional).
Responses:	Code	Description / example if successful	
	200 OK	Book updated successfully.	

		{ "message": "Book updated successfully!" }
	400 Bad Request	Missing required fields
	404 Not Found	Book not found.

PUT /reviews/{id}			
Edits an existing review based on review ID.			
Parameters:	Name	Type	Description
Add a * to the name of required parameters.	id*	path	The ID of the review to edit.
	text*	body	The new text of the review.
Responses:	Code	Description / example if successful	
	200 OK	Review updated successfully. { "message": "Review updated successfully!", "review": { "id": 1, "text": "Updated review text" } }	
	400 Bad Request	Review text is required.	

PUT /user/{userId}			
Updates the information of an existing user and reflects the changes on the user profile page.			
Parameters:	Name	Type	Description
	userId	path	The ID of the user to update.
	username*	body	The new username of the user or the old one( the field cannot be empty).
	email*	body	The new email address of the user or the old one( the field cannot be empty).
	password*	body	The new password of the user or the old one( the field cannot be empty).
Responses:	Code	Description / example if successful	
	200 OK	User updated successfully. { "message": "Profile updated successfully!"	

		}
	400 Bad Request	Missing required fields.

### DELETE requests

DELETE /book/{id}			
Remove a book from the system.			
Parameters:	Name	Type	Description
	id*	path	The ID of the book to delete.
Responses:	Code	Description / example if successful	
	200 OK	Book deleted successfully. { "message": "Book deleted successfully!" }	
	500 Internal Server Error	An error occurred on the server.	

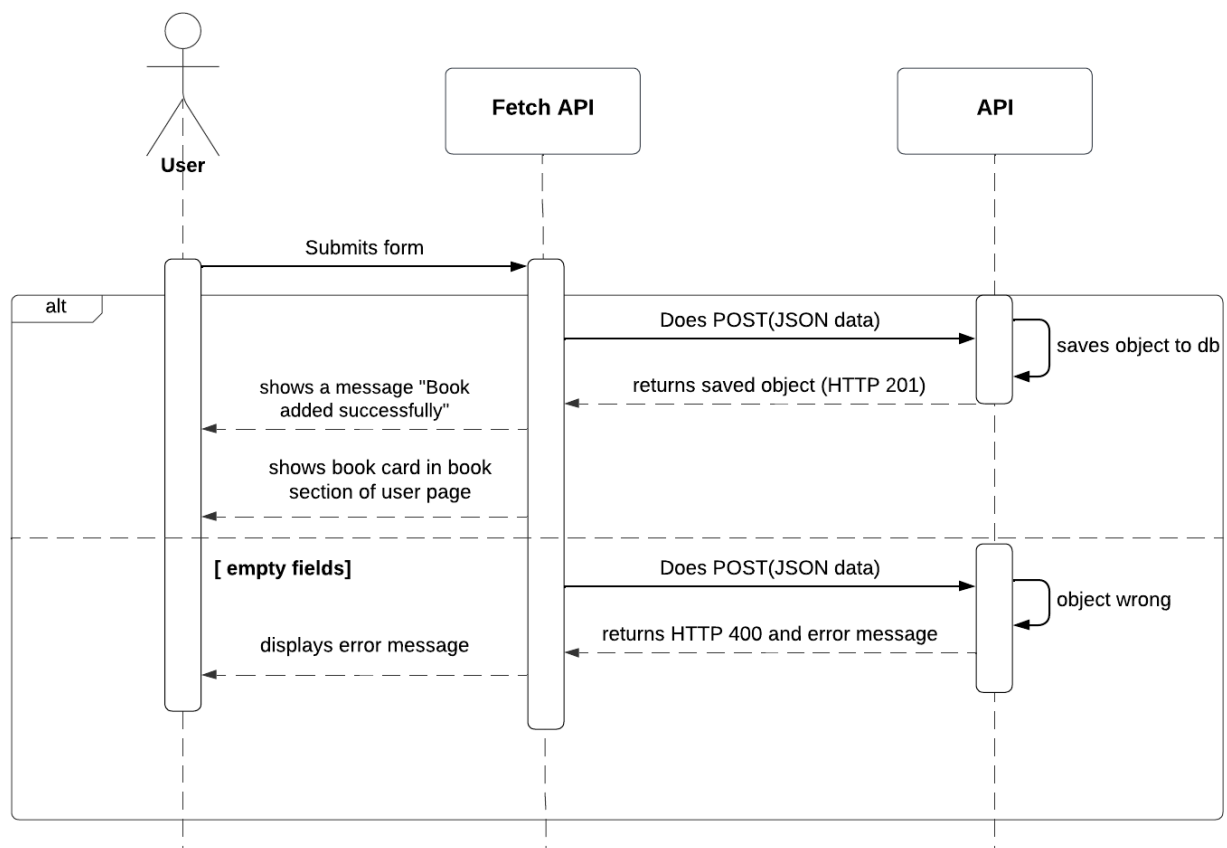
DELETE /reviews/{id}			
Deletes an existing review based on review ID from the system.			
Parameters:	Name	Type	Description
	id*	path	The ID of the review to delete.
Responses:	Code	Description / example if successful	
	200 OK	Review deleted successfully. { "message": "Review deleted successfully!" }	
	500 Internal Server Error	An error occurred on the server.	



# Sequence diagrams

## Diagram 1

This diagram represents the sequence of events when a user adds a book on the Bookish Corner web application.

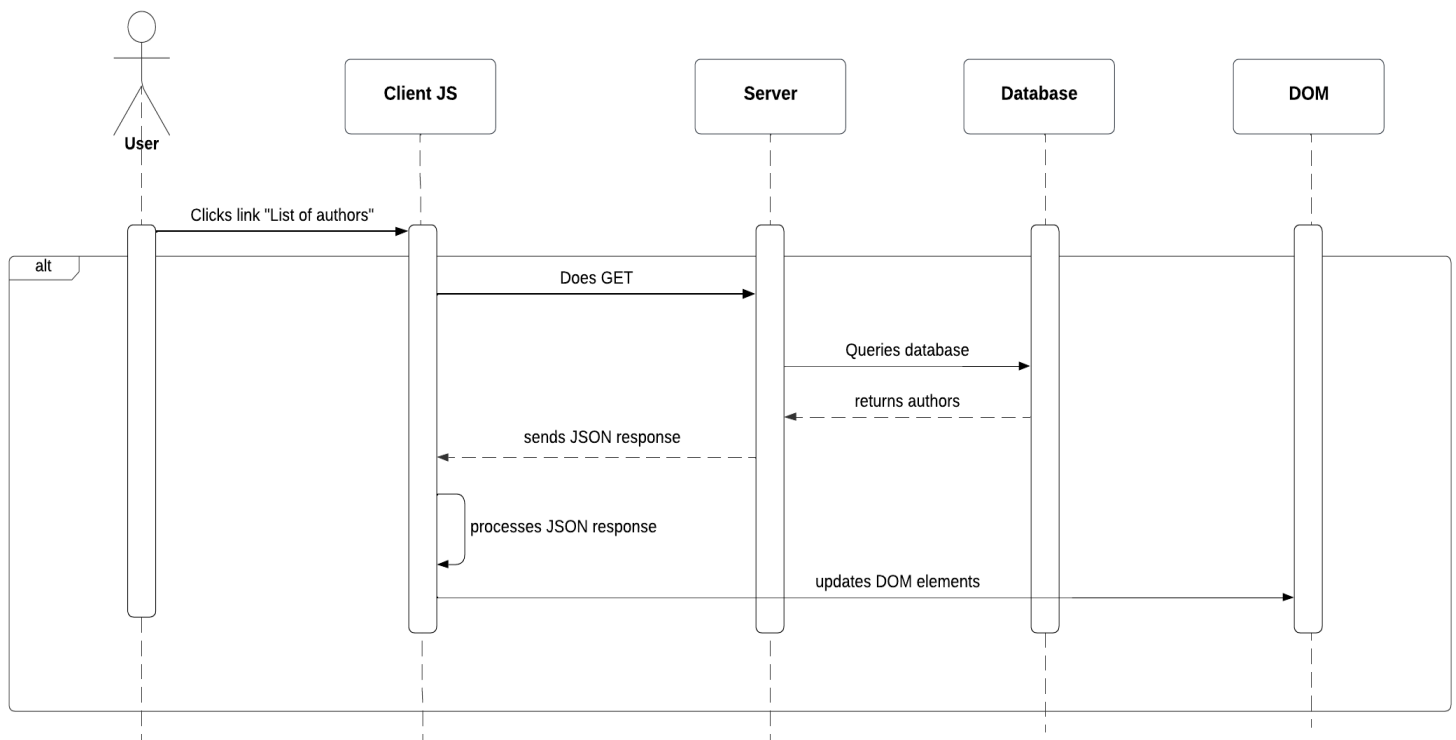


1. The sequence begins when the user submits a form to add a new book from the add book page.
2. The form data, including the title, description, author ID, and optionally the image of the book cover, as well as the data with the user ID of the user who submitted the form, is sent via a POST request to the Fetch API.
3. The Fetch API sends a POST request with the JSON data to the API endpoint **/books**.
4. The API processes the request and attempts to save the new book object to the database.
5. If the book object is saved successfully, the API returns an HTTP 201 status code along with the details of the saved book.
6. The Fetch API receives the response and shows a message indicating that the book was added successfully.
7. The Fetch API displays the new book card in the book section of the user's page, showing the book details.

8. If any required fields are empty when the form is submitted, the API returns an HTTP 400 status code along with an error message indicating that required fields are missing.
9. The Fetch API receives the error response and displays an error message to the user, indicating that the required fields are missing.

## Diagram 2

This diagram represents the sequence of events when a user views the list of authors on the Bookish Corner web application.



1. The sequence begins when the user clicks on the "List of Authors" link on the home page.
2. The client side sends a GET request to the Fetch API to retrieve the list of authors.
3. The Fetch API sends a GET request to the API endpoint **/authors**.
4. The API processes the request and queries the database for all authors.
5. The database retrieves the details of each author, including the ID, first name, last name, and image for each author.
6. The API returns a JSON response with the array of authors to the Fetch API.
7. The Fetch API receives the response and parses the JSON data.
8. The Fetch API updates the DOM to display the list of authors on the "List of Authors" page, showing cards with each author's details.

## Test cases

TC 1		
Name	User Registration	
Description	Test the functionality of the application responsible for registering a new user by using the provided username, email, and password.	
Preconditions	-	
Step	Action	Expected results
1	Navigate to the registration page in the footer of the home page or any other page.	The registration page is displayed.
2	Enter a username, email, and password in the corresponding fields.	Fields are filled with entered data.
3	Submit the form.	The user is registered successfully, receives a confirmation message, and is redirected to the user page.
Postconditions	The user is now authorized and has access to the user page and all features available for authorized users.	

TC 2		
Name	User Login	
Description	Test the functionality of the application responsible for logging in to an existing user by using the provided email and password.	
Preconditions	The user must have a created account.	
Step	Action	Expected results
1	Navigate to the login page in the footer of the home page or any other page.	The login page is displayed.
2	Enter an email and password used during the account creation.	Fields are filled with entered data.
3	Submit the form.	The user logs in successfully, receives a confirmation message and is redirected to the user page.
Postconditions	The user is now authorized and has access to his/her user page and all features available for authorized users.	

TC 3		
Name	Book Adding	
Description	Test the functionality of the application responsible for adding a new book.	
Preconditions	The user must be authorized.	
Step	Action	Expected results
1	Navigate to the "Add book" page through "Add book" link on the user page.	The "Add book" page is displayed.
2	Enter the title, select an author, provide a description, and upload an image.	Fields are filled with entered data and image is displayed after uploading.
3	Submit the form.	The book is added successfully, a confirmation message is displayed, and the

		book appears on the user page in the book section.
Postconditions	The book is created and stored in the database and is now accessible through search and displayed on the “Author books” page for the author selected during the book creation.	

TC 4		
Name	Book Editing	
Description	Test the functionality of the application responsible for editing books.	
Preconditions	The user must be authorized.	
Step	Action	Expected results
1	Click on the “Edit” link on the book card in the book section of the user page.	The “Edit book” page is displayed with prefilled information in all fields.
2	Enter the changes to the book properties, which the user wants to edit.	Fields are filled with the changed data, and if the user changes the image, the new image is displayed after uploading.
3	Submit the form.	The book is edited successfully, a confirmation message is displayed, and the book is now shown on the user's page with updated information.
Postconditions	The book is edited and updated in the database. It is now displayed through search with updated information and shown on the “Author books” page and the “Book page” with updated information.	

TC 5		
Name	Author Adding	
Description	Test the functionality of the application responsible for adding a new author.	
Preconditions	The user must be authorized.	
Step	Action	Expected results
1	Navigate to the “Add author” page through “Add author” link on the user page.	The “Add author” page is displayed.
2	Enter the first name and last name and upload an image of the author.	Fields are filled with entered data and the image is displayed after uploading.
3	Submit the form.	The author is added successfully, a confirmation message is displayed.
Postconditions	The author is created and stored in the database and is now accessible through search and displayed on the “List of authors” page. Additionally, the new author can be selected during the book creation in the dropdown field.	

TC 6	
Name	Profile Editing
Description	Test the functionality of the application responsible for updating user profile information.
Preconditions	The user must be authorized.

Step	Action	Expected results
1	Navigate to the "Edit profile" page through "Edit profile" link on the user page.	The "Edit profile" page is displayed with prefilled information in all fields.
2	Enter the changes to the user properties, which the user wants to edit.	Fields are filled with the changed data.
3	Submit the form.	The user is edited successfully, and a confirmation message is displayed. If the user changes the username, it is immediately reflected on the user page. Any other changed information is reflected on the edit profile page.
Postconditions	User data is updated, and changes are saved in the database.	

TC 7		
Name	Review Adding	
Description	Test the functionality of the application responsible for adding a review to a specific book.	
Preconditions	The user must be authorized.	
Step	Action	Expected results
1	Navigate to a book page.	The book page is displayed with the review field in the "Add Review" section.
2	Enter review text into the review field.	Review text is filled.
	Submit the review by clicking on the "Send" button on the review field.	Review is added successfully and displayed on the book page.
Postconditions	The review is created and stored in the database and displayed in the review section on the user's page.	

TC 8		
Name	Review Editing	
Description	Test the functionality of the application responsible for editing a review added by user.	
Preconditions	The user must be authorized.	
Step	Action	Expected results
1	Click on the "Edit" button on the review in review section of the user page.	The area around the review becomes grey, and the text of the review becomes editable. The "Edit" button changes to a "Save" button.
2	Enter changes to the review text.	The editable review text is displayed with the new content.
	Click on the "Save" button.	The changes are saved successfully, and the updated review text is displayed.
Postconditions	The review is updated and stored in the database with the new information. The updated review is displayed in the review section on the user's page and the corresponding book page.	

TC 9		
Name	Book deleting	
Description	Test the functionality of the application responsible for deleting a book.	
Preconditions	The user must be authorized.	
Step	Action	Expected results
1	Navigate to the book section on user page by clicking on the "Books" button and find the book for deletion.	The user's list of added books is displayed.
2	Click on the "Delete" link on the book card in the book section of the user page.	A confirmation prompt appears asking if the user is sure they want to delete the book.
3	Confirm the deletion.	The book is deleted successfully, and a confirmation message is displayed. The book is removed from the user's book list.
Postconditions	The book is removed from the database and no longer accessible through any part of the application.	

TC 10		
Name	Review deleting	
Description	Test the functionality of the application responsible for deleting a review.	
Preconditions	The user must be authorized.	
Step	Action	Expected results
1	Navigate to the review section on user page by clicking on the "Reviews" button and find the review for deletion.	The user's list of added reviews is displayed.
2	Click on the "Delete" button on the review in the review section of the user page.	A confirmation prompt appears asking if the user is sure they want to delete the review.
3	Confirm the deletion.	The review is deleted successfully, and a confirmation message is displayed. The review is removed from the user's review list.
Postconditions	The review is removed from the database and no longer accessible through any part of the application.	

TC 11		
Name	View List of Authors	
Description	Test the functionality of the application responsible for retrieving and displaying the list of authors.	
Preconditions	-	
Step	Action	Expected results
1	Click on the "List of Authors" link on the home page.	The user is navigated to the "List of Authors" page. The list of authors is displayed on the

		page, showing each author's card including full name, image and "View Books" link .
Postconditions	-	

TC 12		
Name	View Authors Books	
Description	Test the functionality of the application responsible for retrieving and displaying books written by a specific author.	
Preconditions	-	
Step	Action	Expected results
1	Click on an author card or "View Books" link from the list of authors.	The user is navigated to the author's books page. The list of books written by the author is displayed on the page, showing each book's card and displaying the name of the author above this list.
Postconditions	-	

TC 13		
Name	View Book Information	
Description	Test the functionality of the application responsible for retrieving and displaying detailed information about a book.	
Preconditions	-	
Step	Action	Expected results
1	Click on the "See more" button on a book card.	The user is navigated to the "Book" page. The book details are displayed on the page, showing title, description, author information, and cover image.
2	Scroll down to the "Reviews" section.	On the book page all reviews left for this book is displayed.
Postconditions	-	

TC 14		
Name	Search Books	
Description	Test the functionality of the application responsible for searching books by title.	
Preconditions	-	
Step	Action	Expected results
1	Click on the "Books" button above the search bar.	The button is now activated and displayed in black color with white text.
2	Enter a book title or part of the title in the search bar on the home page.	The search results are displayed, showing books cards that match the search request.
Postconditions	-	

TC 15	
Name	Search Authors
Description	Test the functionality of the application responsible for searching authors by name.

Preconditions	-	
Step	Action	Expected results
1	Click on the "Authors" button above the search bar.	The button is now activated and displayed in black color with white text.
2	Enter an author's name or part of the name in the search bar on the home page.	The search results are displayed, showing authors cards that match the search request.
Postconditions	-	

## Bad weather tests

TC 16		
Name	User registration with email of existing user.	
Description	Test the application's response when attempting to register with an email that is already in use.	
Preconditions	The email must already be registered in the system.	
Step	Action	Expected results
1	Navigate to the registration page in the footer of the home page or any other page.	The registration page is displayed.
2	Enter a username, email that was used before during the account creation, and password in the corresponding fields.	Fields are filled with entered data.
3	Submit the form.	The system displays an error message indicating that the user already exists.
Postconditions	The user is not authorized.	

TC 17		
Name	User login with email that was not used during registration of user.	
Description	Test the application's response when attempting to log in with an email that is not registered.	
Preconditions	The email must not be registered in the system.	
Step	Action	Expected results
1	Navigate to the login page in the footer of the home page or any other page.	The login page is displayed.
2	Enter an email that was not used during the registration and a password used during the account creation.	Fields are filled with entered data.
3	Submit the form.	The system displays an error message indicating that the user not found.
Postconditions	The user is not authorized.	

TC 18	
Name	User login with incorrect password.
Description	Test the application's response when attempting to log in with an incorrect password.
Preconditions	The user must have a created account.



Step	Action	Expected results
1	Navigate to the login page in the footer of the home page or any other page.	The login page is displayed.
2	Enter a registered email and an incorrect password.	Fields are filled with entered data.
3	Submit the form.	The system displays an error message indicating that the password is invalid
Postconditions	The user is not authorized.	

TC 19		
Name	Review editing with empty review text.	
Description	Test the application's response when attempting to edit a review with an empty review text field.	
Preconditions	The user must be authorized.	
Step	Action	Expected results
1	Click on the "Edit" button on the review in review section of the user page.	The area around the review becomes grey, and the text of the review becomes editable. The "Edit" button changes to a "Save" button.
2	Clear the review text	The editable review text field is empty.
	Click on the "Save" button.	The application should display an error message indicating that the review text cannot be empty.
Postconditions	The review is not updated, and the original review text remains unchanged in the database and on the user page.	

TC 20		
Name	Profile editing with empty fields.	
Description	Test the application's response when attempting to edit a profile with an empty fields.	
Preconditions	The user must be authorized.	
Step	Action	Expected results
1	Navigate to the "Edit profile" page through "Edit profile" link on the user page.	The "Edit profile" page is displayed with prefilled information in all fields.
2	Clear one or more fields (username, email, or password).	The cleared fields are empty.
3	Submit the form.	The application should display an error message indicating that all required fields must be filled out.
Postconditions	User data is not updated, and the original profile information remains unchanged in the database.	

TC 21	
Name	The author adding with empty first name and last name fields.


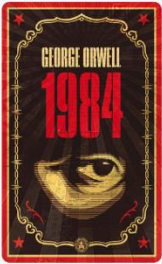

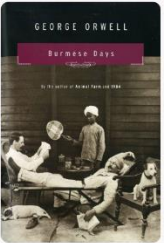
Description	Test the application's response when attempting to add an author with empty first name and last name fields.	
Preconditions	The user must be authorized.	
Step	Action	Expected results
1	Navigate to the "Add author" page through "Add author" link on the user page.	The "Add author" page is displayed.
2	Do not enter the first name and last name, upload an image of the author.	The image is displayed after uploading, but the first name and last name fields remain empty.
3	Submit the form.	The application should display an error message indicating that the first name and last name fields are required.
Postconditions	The author is not created, and no data is stored in the database.	

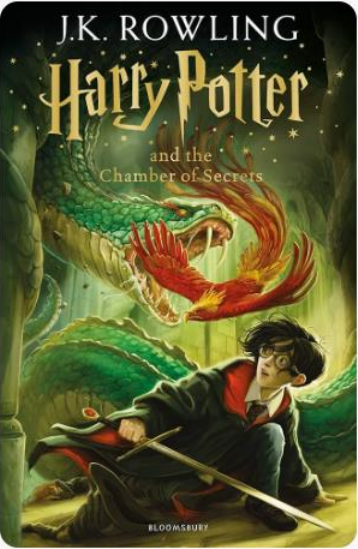
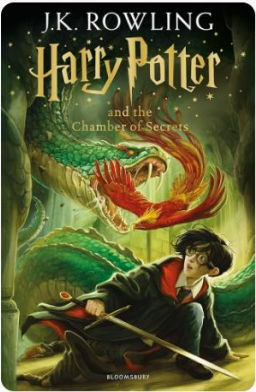
TC 22		
Name	Book editing with an empty title, description fields.	
Description	Test the application's response when attempting to edit a book with empty title and description fields.	
Preconditions	The user must be authorized.	
Step	Action	Expected results
1	Click on the "Edit" link on the book card in the book section of the user page.	The "Edit book" page is displayed with prefilled information in all fields.
2	Clear the title and description fields.	The cleared fields are empty, other information remains the same.
3	Submit the form.	The application should display an error message indicating that the title and description fields are required.
Postconditions	The book is not edited, and the original book information remains unchanged in the database.	


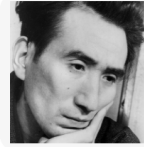


TC 23		
Name	Book adding with empty title, description fields.	
Description	Test the application's response when attempting to add a book with empty title and description fields.	
Preconditions	The user must be authorized.	
Step	Action	Expected results
1	Navigate to the "Add book" page through "Add book" link on the user page.	The "Add book" page is displayed.
2	Do not enter the title and description but select an author and upload an image.	The image is displayed after uploading and author selected, but the title and description fields remain empty.
3	Submit the form.	The application should display an error message indicating that the title and description fields are required.
Postconditions	The book is not created, and no data is stored in the database.	


# Test report

Test case	Result	Details
TC 1	OK	<div>The user is registered successfully.</div> <div><div>127.0.0.1:5500 says</div><div>User successfully created!</div><div>OK</div></div> <div><div>test user</div><div>test@gmail.com</div><div>12345</div><div>Create account</div></div> <div><div><div>Books</div><div>Reviews</div></div><div><div>test user</div><div>Add book</div><div>Add author</div><div>Edit profile</div><div>Logout</div></div><div>You didn't add any books yet!</div></div>
TC 2	OK	The user is logged in successfully.

		<div>127.0.0.1:5500 says</div> <div>Login successful</div> <div>OK</div> <div>admin@gmail.com</div> <div>.....</div> <div>Login</div> <div><div> Bookish Corner</div><div><div>Your added books</div><div><div>Books</div><div>Reviews</div></div><div><div><div>Nineteen Eighty-Four (1984)</div><div>Edit bookDelete book</div></div><div><div>Animal Farm</div><div>Edit bookDelete book</div></div><div><div>Burmese days</div><div>Edit bookDelete book</div></div></div><div><div>Super Admin</div><div><a href="#">Add book</a> <a href="#">Add author</a> <a href="#">Edit profile</a> <a href="#">Logout</a></div></div></div></div>
TC 3	OK	<div>The book is added successfully.</div> <div>&lt;-add.html</div> <div>127.0.0.1:5500 says</div> <div>Book added successfully!</div> <div>OK</div> <div>Sup</div> <div><a href="#">Go to</a> <a href="#">Logot</a></div>










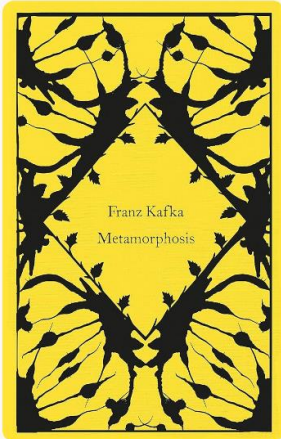
		<div><p><b>test book</b></p><p><a href="#">Edit book</a> <a href="#">Delete book</a></p></div>
TC 4	OK	<p>The book is edited successfully.</p> <div><p>127.0.0.1:5500 says Book updated successfully!</p><p>OK</p></div> <div><p><b>test book 2</b></p><p><a href="#">Edit book</a> <a href="#">Delete book</a></p></div>
TC 5	OK	<p>The author is added successfully.</p>

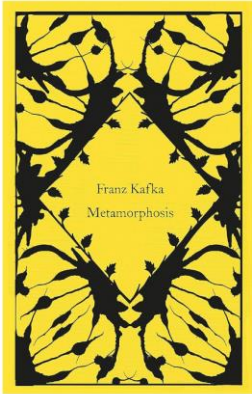


		<div> <div>  <b>Bookish Corner</b> </div> <div> 127.0.0.1:5500 says  Author added successfully! </div> </div> <div> <h2>Add author</h2> <div> <input type="text" value="Osamu"/> </div> <div> <input type="text" value="Dazai"/> </div> <div> <input type="text" value="Select image"/>  </div> <div> <input type="button" value="Add author"/> </div> <div> <b>Super Admin</b>  <a href="#">Go to profile</a>  <a href="#">Logout</a> </div> </div>
TC 6	OK	<div> <div>  <b>Bookish Corner</b> </div> <div> 127.0.0.1:5500 says  Profile updated successfully! </div> </div> <div> <h2>Edit your account</h2> <div> <input type="text" value="Super Admin 2"/> </div> <div> <input type="text" value="admin@gmail.com"/> </div> <div> <input type="text" value="admin"/> </div> <div> <input type="button" value="Save changes"/> </div> <div> <b>Super Admin 2</b>  <a href="#">Go to profile</a>  <a href="#">Logout</a> </div> </div> <div> <h2>Edit your account</h2> <div> <input type="text" value="Super Admin 2"/> </div> <div> <input type="text" value="admin@gmail.com"/> </div> <div> <input type="text" value="admin"/> </div> <div> <input type="button" value="Save changes"/> </div> <div> <b>Super Admin 2</b>  <a href="#">Go to profile</a>  <a href="#">Logout</a> </div> </div>
TC 7	OK	<div> <div>  <b>Bookish Corner</b> </div> <div> 127.0.0.1:5500 says  Review added successfully! </div> </div> <div> <h2>Add review</h2> <div> <input type="text" value="Book title"/> </div> <div> <input type="text" value="User name"/> </div> <div> <input type="text" value="Rating"/> </div> <div> <input type="text" value="Review text"/> </div> <div> <input type="button" value="Add review"/> </div> <div> <b>Super Admin</b>  <a href="#">Go to profile</a>  <a href="#">Logout</a> </div> </div>

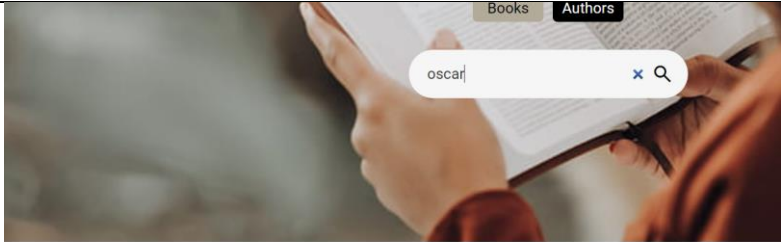

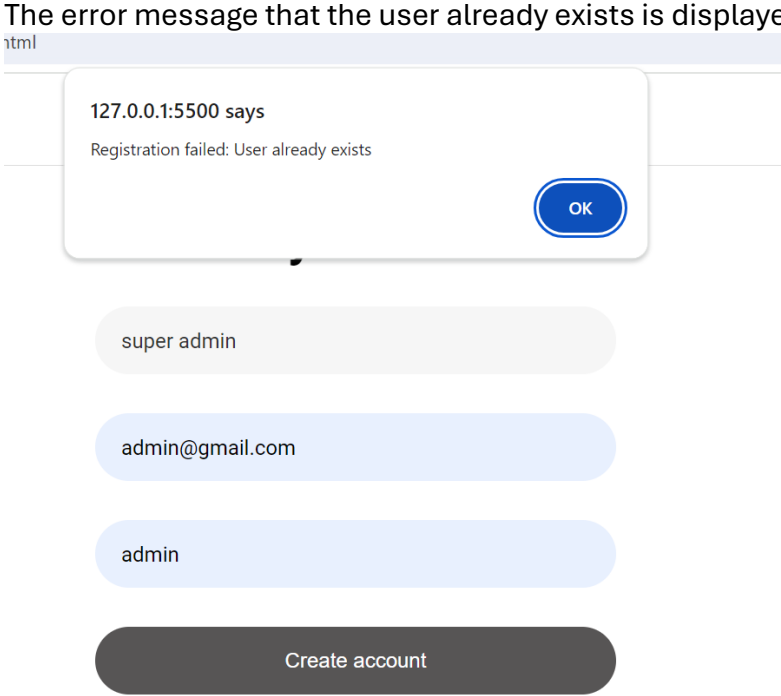
		<div><div>127.0.0.1:5500 says</div><div>Review added successfully!</div><div>OK</div></div> <div><h2>Reviews</h2><div>Write review... </div><div>Super Admin</div><div>test review</div></div>
TC 8	OK	<div>The review is edited successfully.</div> <div><div>127.0.0.1:5500 says</div><div>Review updated successfully!</div><div>OK</div></div> <div><h2>Metamorphosis</h2><div>test review 2</div><div><div>Save</div><div>Delete</div></div><h2>Your added reviews</h2><div><div>Books</div><div>Reviews</div></div><div><h3>Metamorphosis</h3><div>test review 2</div><div><div>Edit</div><div>Delete</div></div></div></div>
TC 9	OK	<div>The book is deleted successfully.</div>

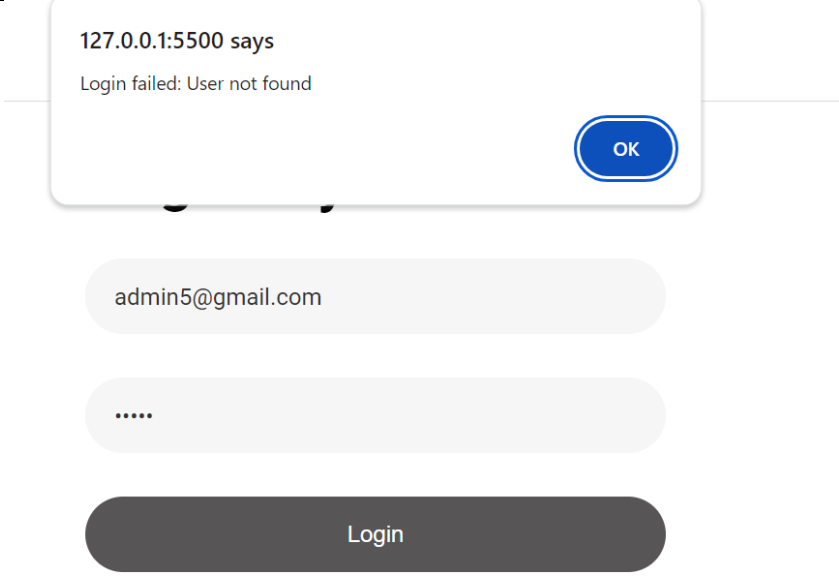
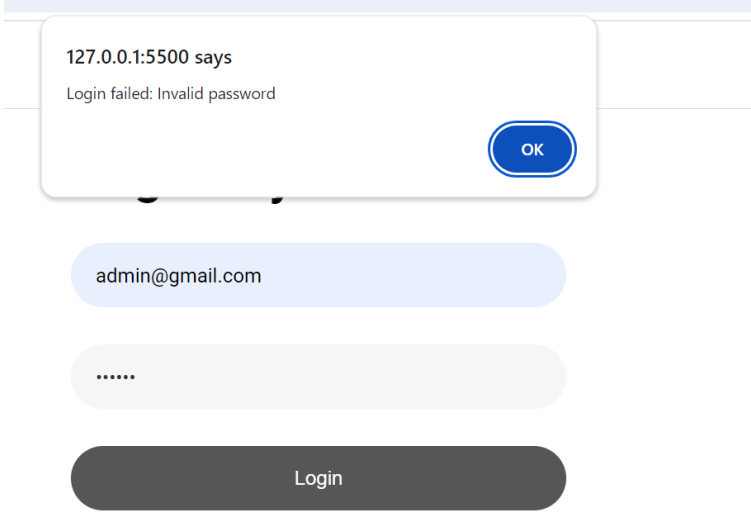
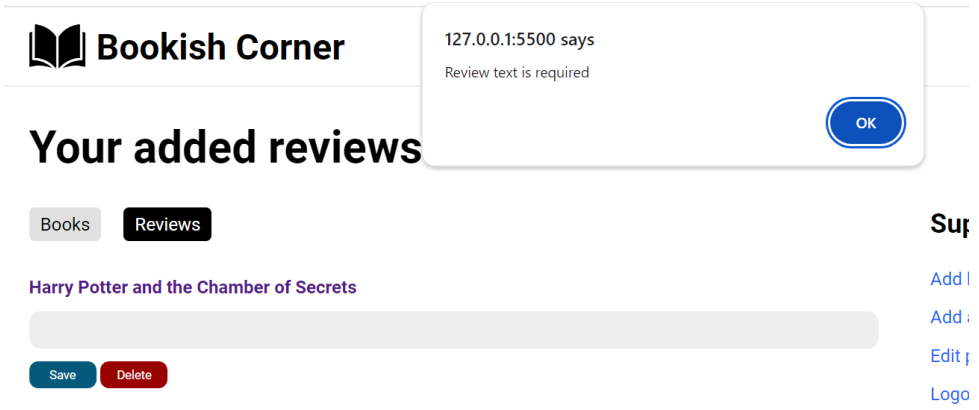
		<div>127.0.0.1:5500 says</div> <div>Book deleted successfully!</div> <div>OK</div> <div><p>test book 2</p><p><a>Edit book</a> <a>Delete book</a></p></div> <div><p>The Signature of All Things</p><p><a>Edit book</a> <a>Delete book</a></p></div>
TC 10	OK	<div>The review is deleted successfully.</div> <div>127.0.0.1:5500 says</div> <div>Review deleted successfully!</div> <div>OK</div>






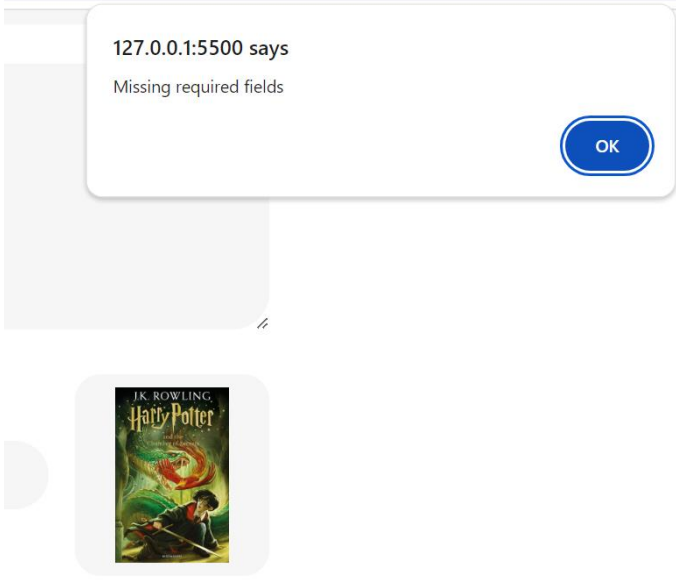
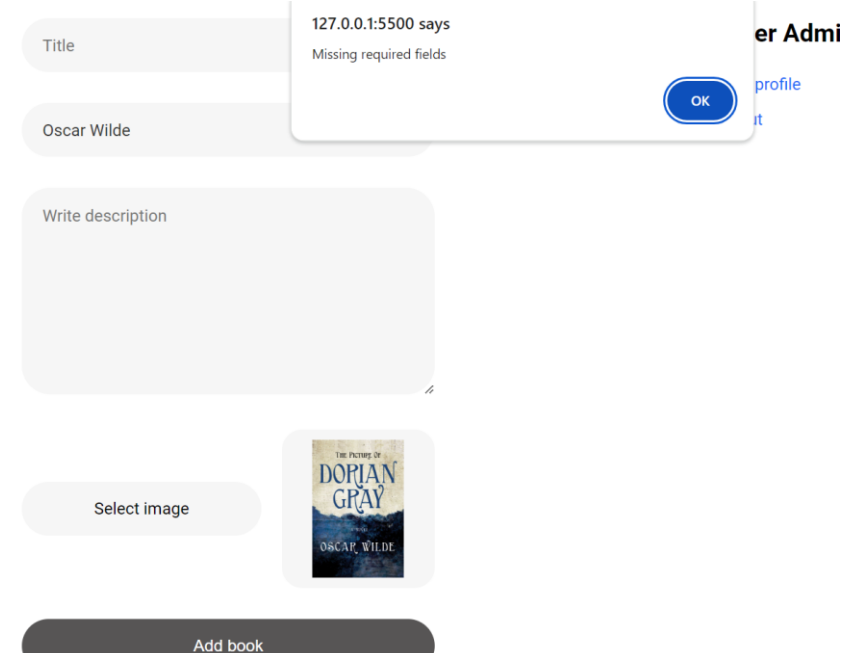
		<h1>Your added reviews</h1> <div> <a href="#">Books</a> <a href="#">Reviews</a> </div> <p>You didn't write any reviews yet!</p>
TC 11	OK	<p>The list of authors is displayed successfully.</p> <div>  Bookish Corner       <a href="#">Back</a> </div> <h2>List of authors</h2> <div> <div>  <p>George Orwell</p> <a href="#">View books</a> </div> <div>  <p>Joanne Rowling</p> <a href="#">View books</a> </div> <div>  <p>Franz Kafka</p> <a href="#">View books</a> </div> <div>  <p>Mikhail Bulgakov</p> <a href="#">View books</a> </div> </div> <div>     </div>
TC 12	OK	<p>A list of books for specific author is displayed successfully.</p> <div> <h2>Franz Kafka</h2> <div>  <p>Metamorphosis</p> <a href="#">See more</a> </div> </div>
TC 13	OK	<p>The information for a specific book is displayed successfully.</p>

		<div data-bbox="497 143 750 535">  </div> <div data-bbox="810 143 1051 181"> <h2>Metamorphosis</h2> </div> <div data-bbox="810 208 963 230"> <p>Author: Franz Kafka</p> </div> <div data-bbox="810 255 1219 624"> <p>Franz Kafka's "Metamorphosis" is a haunting and enigmatic tale that delves deep into the psyche of its protagonist, Gregor Samsa. One morning, Gregor awakens to find himself inexplicably transformed into a monstrous insect. This shocking metamorphosis serves as a catalyst for the unraveling of Gregor's life and the disintegration of his relationships. As Gregor grapples with the physical and emotional ramifications of his transformation, he is confronted with the cruel realities of human nature. His family's initial shock and horror soon give way to resentment and neglect, as they struggle to come to terms with his grotesque appearance and diminished capacity to contribute to the household. Trapped within the confines of his room, Gregor becomes increasingly isolated from the outside world, his existence reduced to a mere spectacle for his family's amusement and disdain. Through Gregor's surreal and tragic journey, Kafka masterfully explores themes of alienation, identity, and the absurdity of human existence.</p> </div> <div data-bbox="485 660 627 701"> <h3>Reviews</h3> </div> <div data-bbox="485 732 1286 788"> <div>Write review...</div> </div> <div data-bbox="485 817 826 842"> <p>There is no reviews for this book!</p> </div>
TC 14	OK	<div data-bbox="472 887 1391 916"> <p>Search displayed book cards corresponding to the search request.</p> </div> <div data-bbox="472 920 1323 1243">  </div> <div data-bbox="472 1274 679 1312"> <h3>Search result</h3> </div> <div data-bbox="472 1344 1323 1516">  </div>
TC 15	OK	<div data-bbox="472 1523 1396 1552"> <p>Search displayed author card corresponding to the search request.</p> </div>

		<div data-bbox="470 112 1254 353">  </div> <div data-bbox="470 380 793 425"> <p>Your search result</p> </div> <div data-bbox="885 452 1144 721">  <p>Oscar Wilde</p> </div>
TC 16	OK	<p>The error message that the user already exists is displayed.</p> <div data-bbox="470 728 1254 1422">  </div>
TC 17	OK	<p>The error message that the user is not found is displayed.</p>

		
TC 18	OK	<p>The error message that the user's password is invalid is displayed.</p> 
TC 19	OK	<p>The error message that the review text required is displayed, the review was not updated.</p> 
TC 20	OK	<p>The error message that the missing fields are required is displayed, the profile was not updated.</p>

		 <b>Bookish Corner</b> <div>127.0.0.1:5500 says Missing required fields</div> <div>OK</div> <h2>Edit your account</h2> <div>Username</div> <div>admin@gmail.com</div> <div>admin</div> <div>Save changes</div> <div>Super Admin</div> <div>Go to profile</div> <div>Logout</div>
TC 21	OK	<p>The error message that the missing fields are required is displayed, the author was not added.</p>  <b>Bookish Corner</b> <div>127.0.0.1:5500 says Missing required fields</div> <div>OK</div> <h2>Add author</h2> <div>First name</div> <div>Last name</div> <div>Select image</div>  <div>Add author</div> <div>Super Admin</div> <div>Go to profile</div> <div>Logout</div>
TC 22	OK	<p>The error message that the missing fields are required is displayed, the book was not updated.</p>

		
TC 23	OK	<p>The error message that the missing fields are required is displayed, the book was not added.</p> 

## Justification for login/registration process

The authentication for this project was not required because implementing robust authentication involves numerous aspects such as data hashing and password recovery, which require significant time. Due to the given reduced deadline for this module, it was not practical. However, while developing certain functionalities in my web application, I realized that a basic authentication process would be beneficial and help me achieve my planned functionality. During the registration process, the system simply saved the entered data from the registration form to the database without any hashing. During the login process, the system compares the entered data to the data stored in the database. If they match, the user is logged into the system.

I acknowledge that this is not a reliable or secure solution for a real system and understand the potential issues related to such an implementation of authentication. I will not repeat this approach and will enhance security for future web projects.