Book sharing agency

- Mahmoud Kamal Mahmoud: 56
- Mohammed Magdy Abd-elghany: 53
- Hamza Hassan Mohammed: 25
- Mahmoud Ebrahim Elsayed: 55
- Ahemd adel salama: 7
- Ahmed Ayman Mohamed : 4

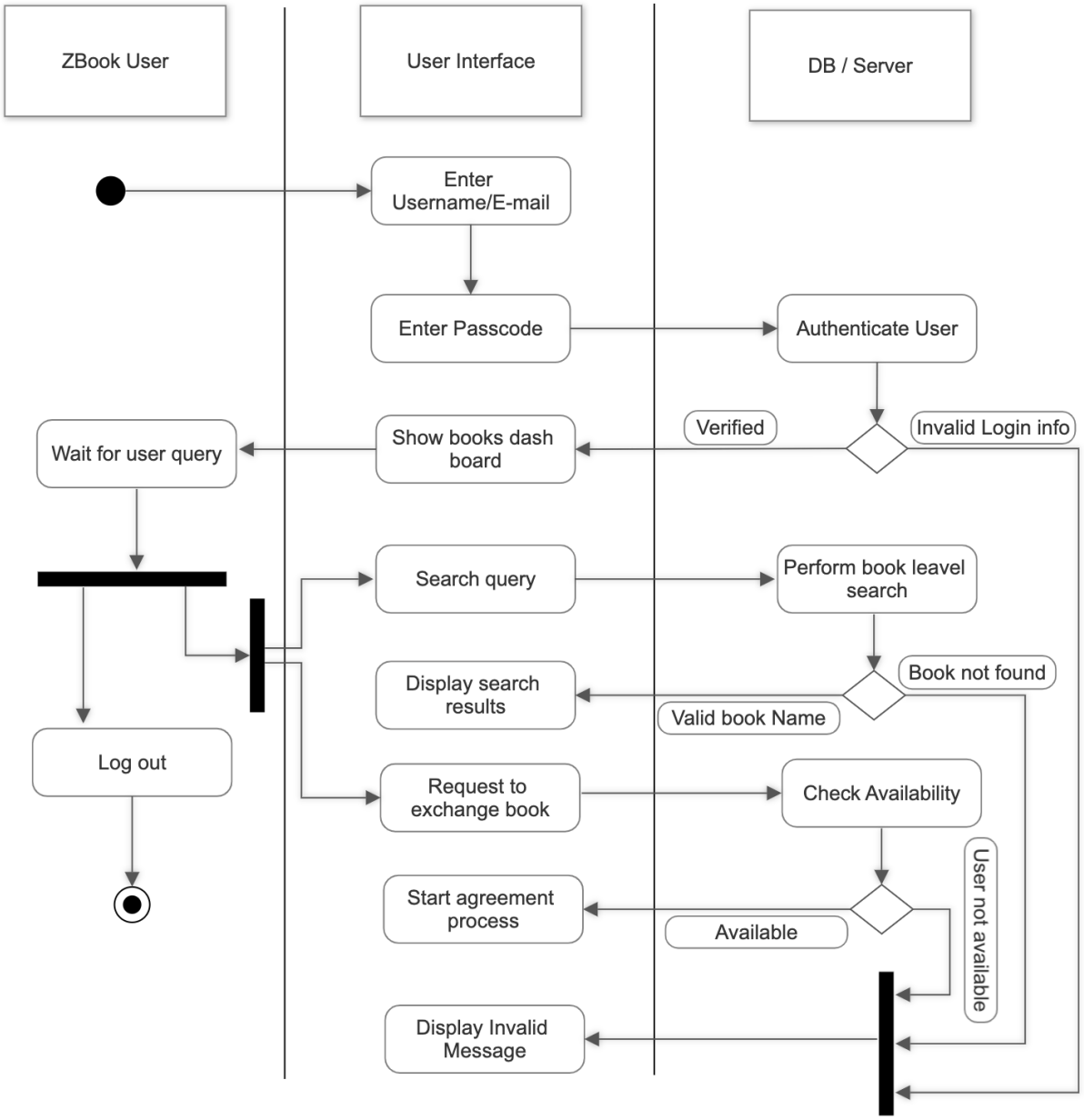
Requirements that are done:

1. **FrontEnd**
    - The user can log in
    - The user can add books to the system
    - List all books
    - The user can logout
2. **BackEnd**
    - Implementing the class model for the User(Reader) and the Book
    - Constructing the Schema of the database
    - Adding facilities to AddBook, AddReader, DeleteBooks, DeleteReaders, FileReader, FindBook
    - Adding the possibility for a User to acquire books
    - Creating endpoints to receive requests from the user.

Design

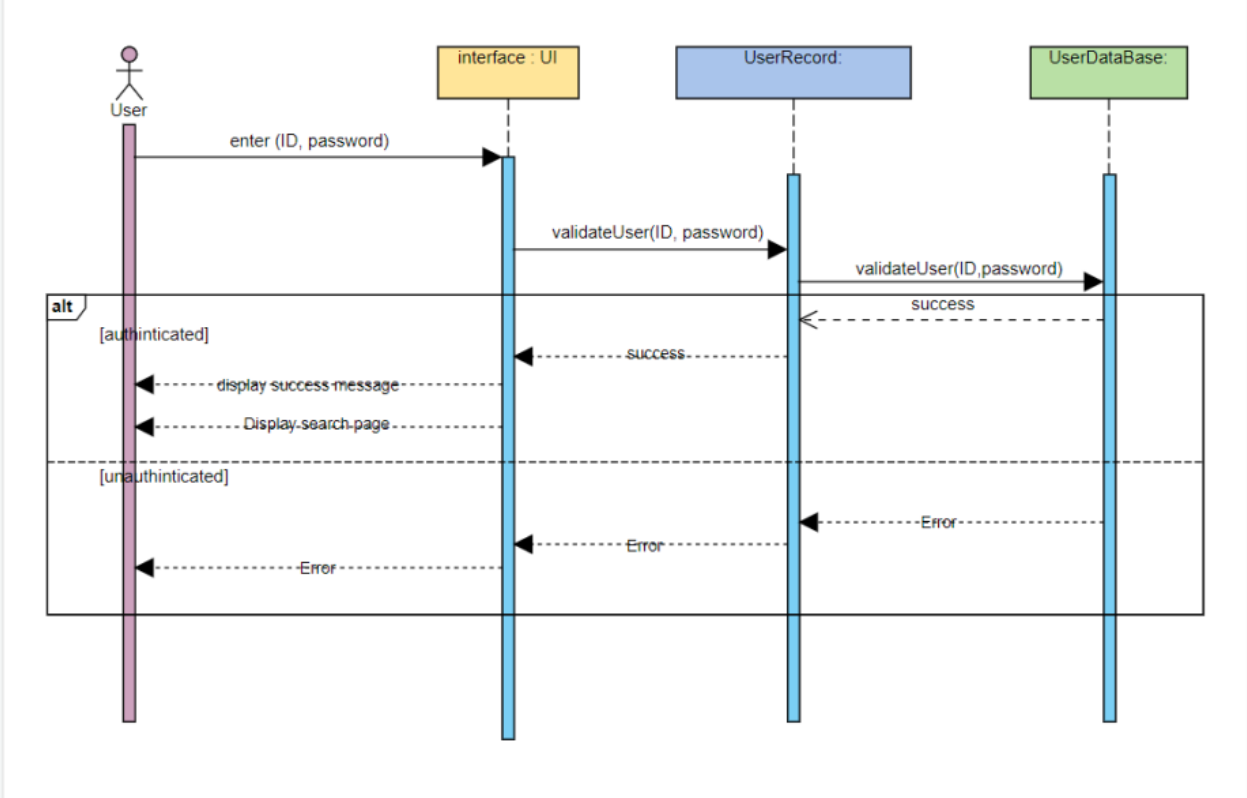
1. FrontEnd
    - React js framework
    - HTML
    - CSS
2. BackEnd
    - MangoDB
    - Nodejs

Activity Diagram

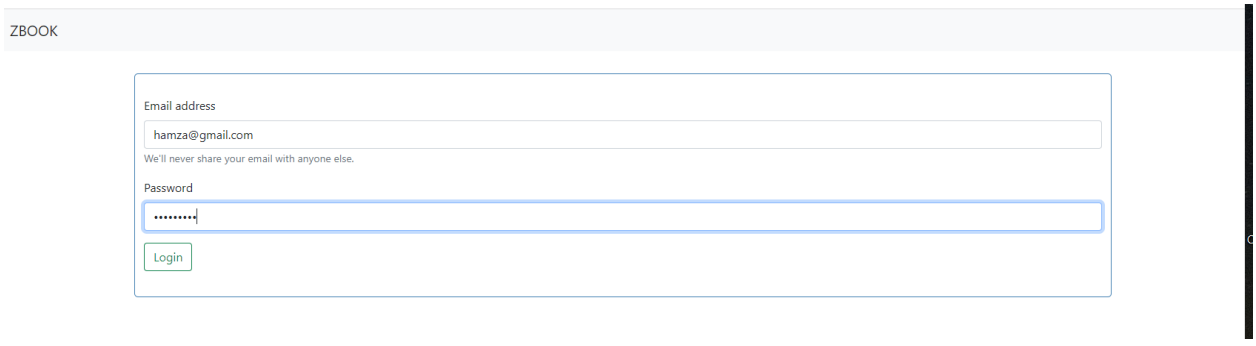


User Scenarios

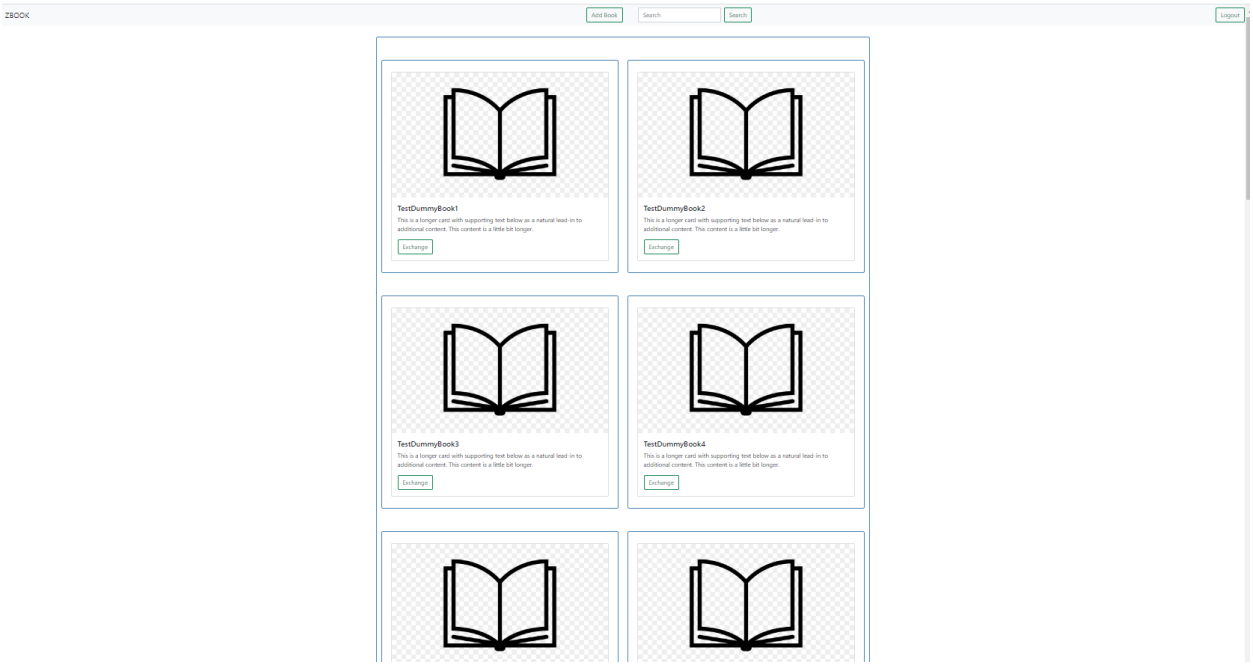
The user logs in and explores the books in the system



1. The user login to the server

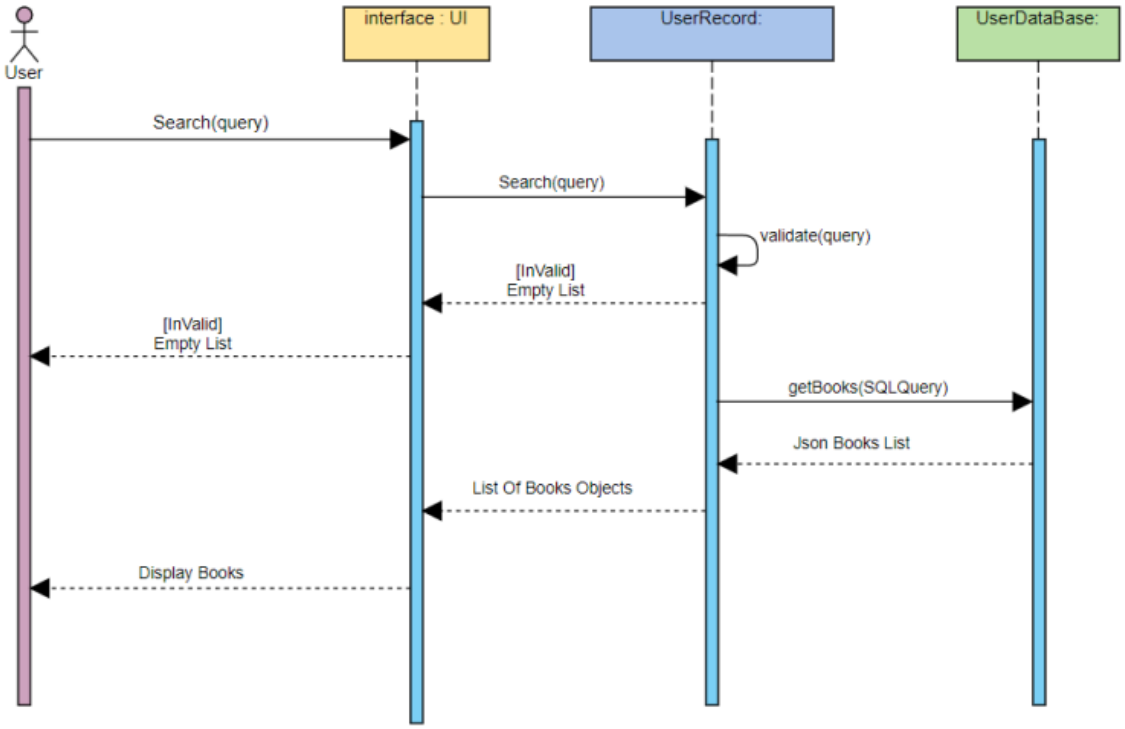


2. The user can explore the books existing in the system

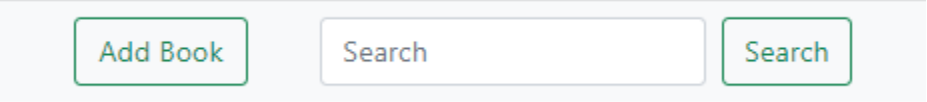


3. After exploring the books, they may log out of the system

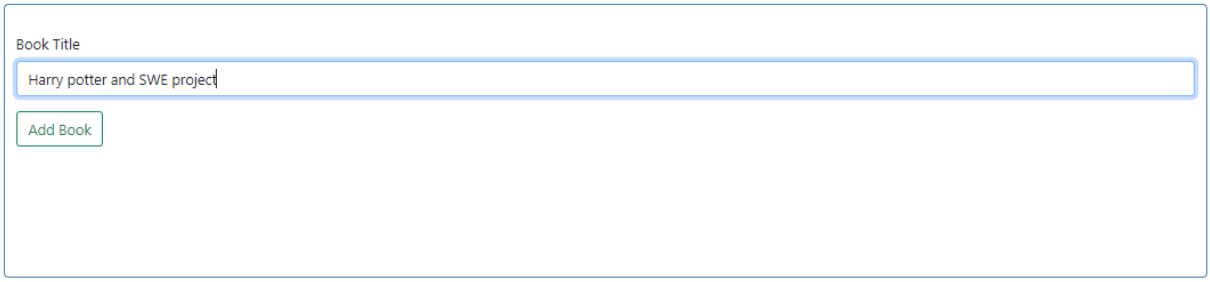
The user wants to add a book to the system



1. The user logs in to the system
2. The user presses the button “Add Book” at the top of the page



3. The user enters the book’s name they want to add into the system



4. Finally, the user logs in to the system and they will find the book in the books list. The book is also added to the database

Harry potter and SWE project

This is a longer card with supporting text below as a natural lead-in to additional content. This content is a little bit longer.

Exchange


```
_id: ObjectId("628e8573b7ed15808141b07b")
title: "Harry potter and SWE project"
userIds: Array
createdAt: 2022-05-25T19:37:23.780+00:00
updatedAt: 2022-05-25T19:37:23.780+00:00
__v: 0
```

Testing

We used the framework Jest with NodeJs for testing.

Our Testing strategy works as follows:

1. We constructed modules to enable the tester to write the tests before even starting implementation.
2. We used the modules' methods to create 26 tests that cover most of the functionalities we implemented in this milestone.

The tests don't depend on the existence of certain entries in the database's tables. They are fully automated, they add the needed entries in the database tables once they connect with it.

The Functionalities covered in the testing

1. Testing establishing a connection with the database
2. Testing adding books to the database
3. Testing removing books from the database
4. Testing adding users to the database
5. Testing removing users from the database
6. Testing the user when adding a book to the system
7. Testing users adding multiple books to their account
8. Testing the books get added by multiple users

```
 PASS  test/DB/userAndBook.test.ts (18.553 s)
  √ connectDatabase (2361 ms)
  √ Drop the entries from book table (1174 ms)
  √ Delete the users we are going to add don't exist (486 ms)
  √ Make sure that the users and books are deleted (1388 ms)
  √ add book 1  (169 ms)
  √ add book 2 (158 ms)
  √ add book 3  (160 ms)
  √ add book 4  (170 ms)
  √ add book 5 (159 ms)
  √ Count the number of book in the database (1 ms)
  √ create user 1 (166 ms)
  √ create user 2 (160 ms)
  √ create user 3 (168 ms)
  √ Get the Users objects and the books objects  (813 ms)
  √ Check that all users exist  (1 ms)
  √ Check that all books exist  (1 ms)
  √ Add book1 to User1 (662 ms)
  √ Add book2 to User1 (672 ms)
  √ Get all books that user 1 owns  (1 ms)
  √ Add book1 to User2 (703 ms)
  √ Add book1 to User3 (630 ms)
  √ Get all books that user 2 owns  (1 ms)
  √ Check that all users own book1  (1 ms)
  √ Check that only one user own book2
  √ check that user3 owns only one book
  √ Closing the database (5 ms)

Test Suites: 1 passed, 1 total
Tests:       26 passed, 26 total
Snapshots:   0 total
Time:        19.651 s
Ran all test suites.
PS D:\CSED\SWE\Project Milestone\final\Book-Sharing\backend>
```