Developing Fresh Fitz - Group 015-6

Fresh Fitz Developers:

Name	Github Username
Ryan Kerwick	RyanKerwick
Kaythy Liao	Arayla
Elizabeth Long	frizzyliz
Gavin Schoew	gavinschoew

Description:

Fresh Fitz is an interactive outfit builder website that allows users to browse, wish list, and create outfits. The website uses an external API call to populate its database with various different pieces of clothing such as shoes, bottoms, tops, and hats. Upon logging in, the user is directed to the home page that displays all the clothing items for users to browse and discover their aesthetic. When the user finds a piece of clothing that they like, they can wish list the item. The wish listed item will show up in their profile and outfits creator page in their own section. When the user is ready to create an outfit, they can navigate to the outfits or profile page. Users have the option to choose up to four items to create their outfit. Wishlisted items will show up first, followed by the rest of the items in the database. After they have submitted their items, their outfit will be generated and saved in the outfits and profile page. If the user no longer likes an article of clothing or outfit, they have the option to delete the item or outfit. When the user is done, they can log out.

VCS Project Repository:

https://github.com/RyanKerwick/CSCI-3308-Final-Project

Project Board:

https://github.com/users/RyanKerwick/projects/1/views/1?filterQuery=epic%3A%22Search+for+clothing+items%22

Demo Video:

https://youtu.be/PkFsFg5N6Sk

Contributions:

Elizabeth - I was responsible for the UI development of the Profile and Home pages. I also fixed some backend API bugs, created the header and title partials, contributed to the Navbar, and added stylistic page elements to the website overall. I primarily used Handlebars and Bootstrap to dynamically display and align the Home page and Wishlist cards. These tools along with HTML/CSS allowed me to customize the Navbar, buttons (ie. search bar, wishlist, and delete buttons), page background images/colors, text fonts/colors, and image displays.

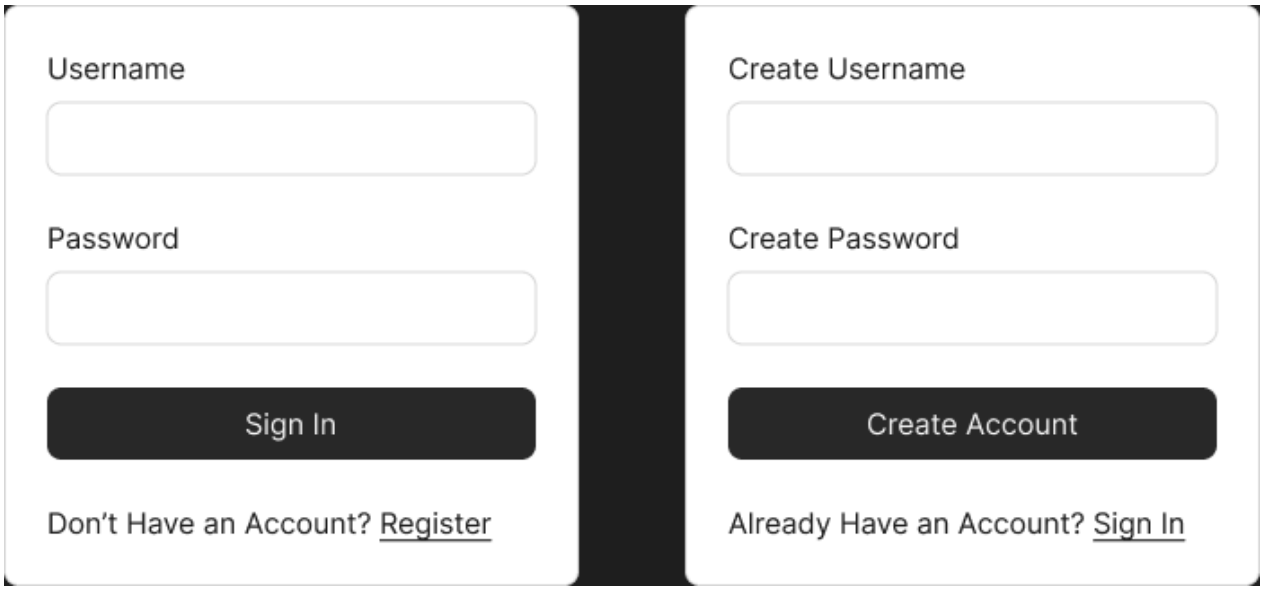
Ryan - Contributed Primarily to the backend, with some front-end contributions. I made major contributions to nearly every API route in our application, including profile, wishlist, deleteWishlist, outfit, and deleteOutfit. I created useful helper functions within our index.js file that were used in the final application, including populate_items, which populated the items table of the database with the external API, and several print functions to debug the database. I wrote functional test cases for login, register, and wishlist API routes with mocha and chai. I designed most of the outfit page, including some client-side javascript to help with functionality. Additionally, I wrote the table create queries for outfits and wishlist.

Kaythy - My main contributions were in the front end. I was in charge of creating the login, register, and logout page. I helped make the layout of the items in the profile and outfits page look good. I also deployed the website using render, recorded the demo, and added additional links to the navigation bar and the footer.

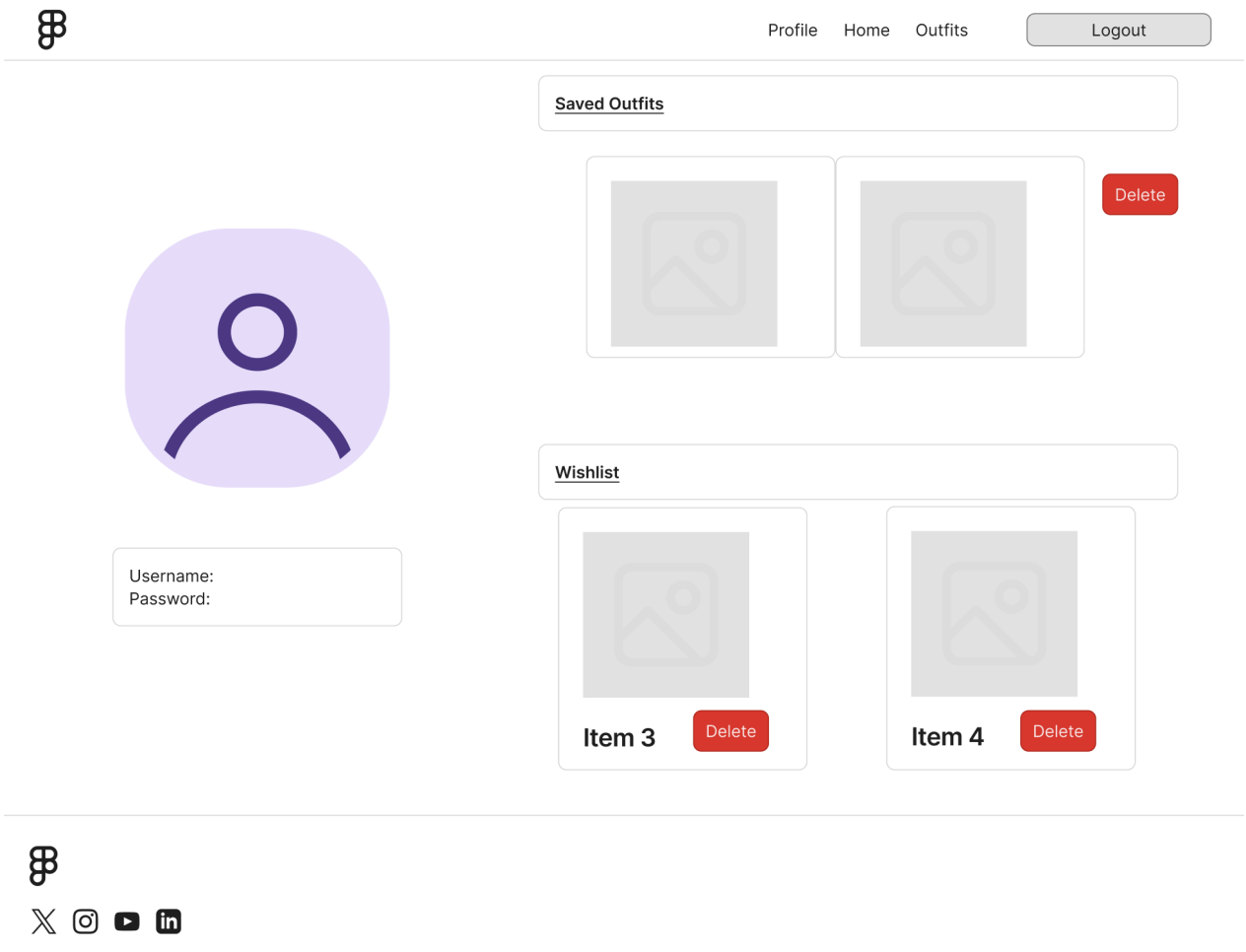
Gavin - I mainly contributed to the backend of our website. I designed and implemented the original database functionality, including the users and items tables, and wrote/modified many PostgreSQL queries in our API routes. I tested several different external APIs to get clothing items to populate our database and implemented the calls to the selected external API in the populate_items function. I also created the search API route and refactored the profile route. I made small modifications to the Home page, ensuring that Handlebars displayed our data correctly.

Wireframes

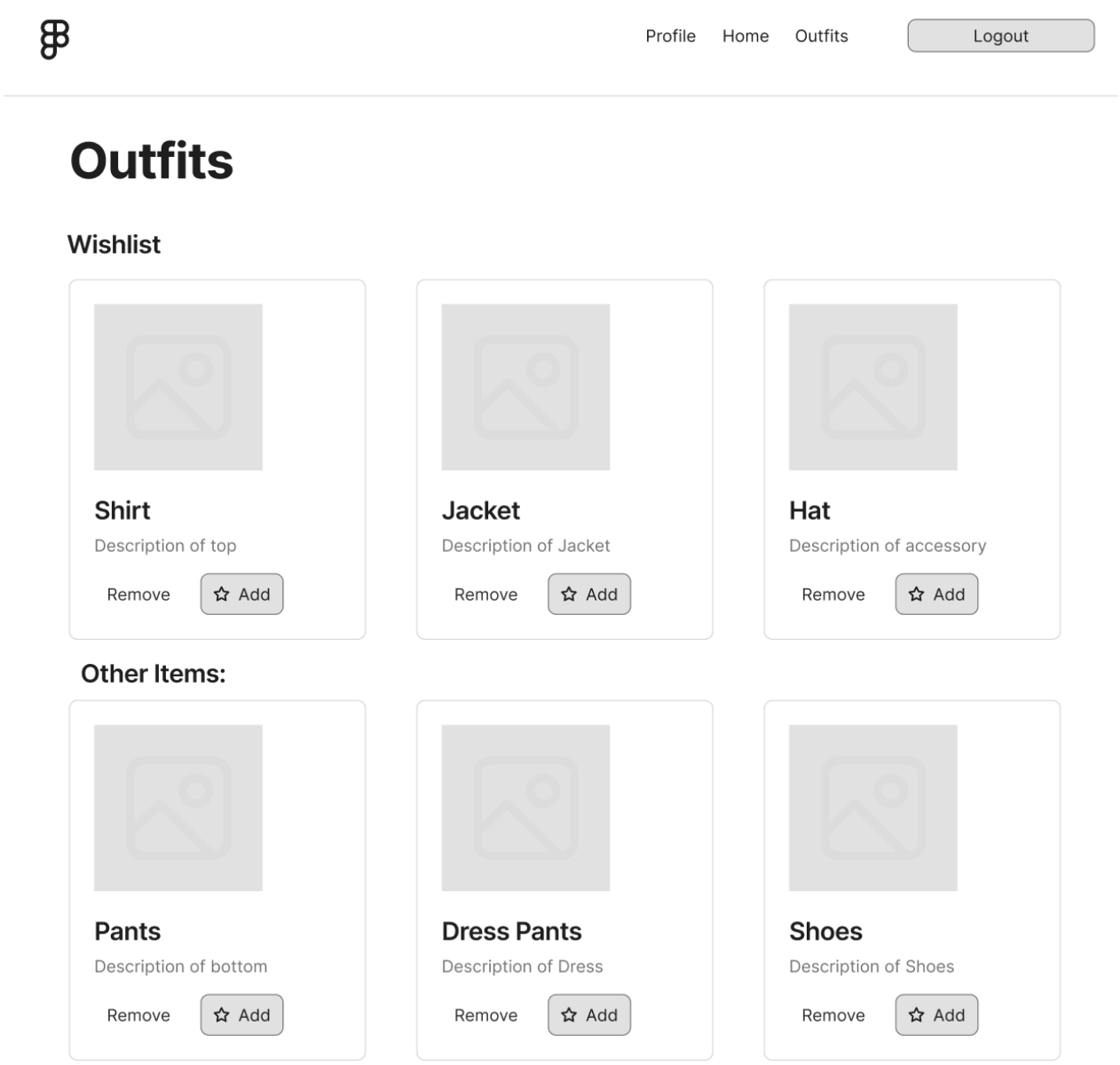
Login/Register:



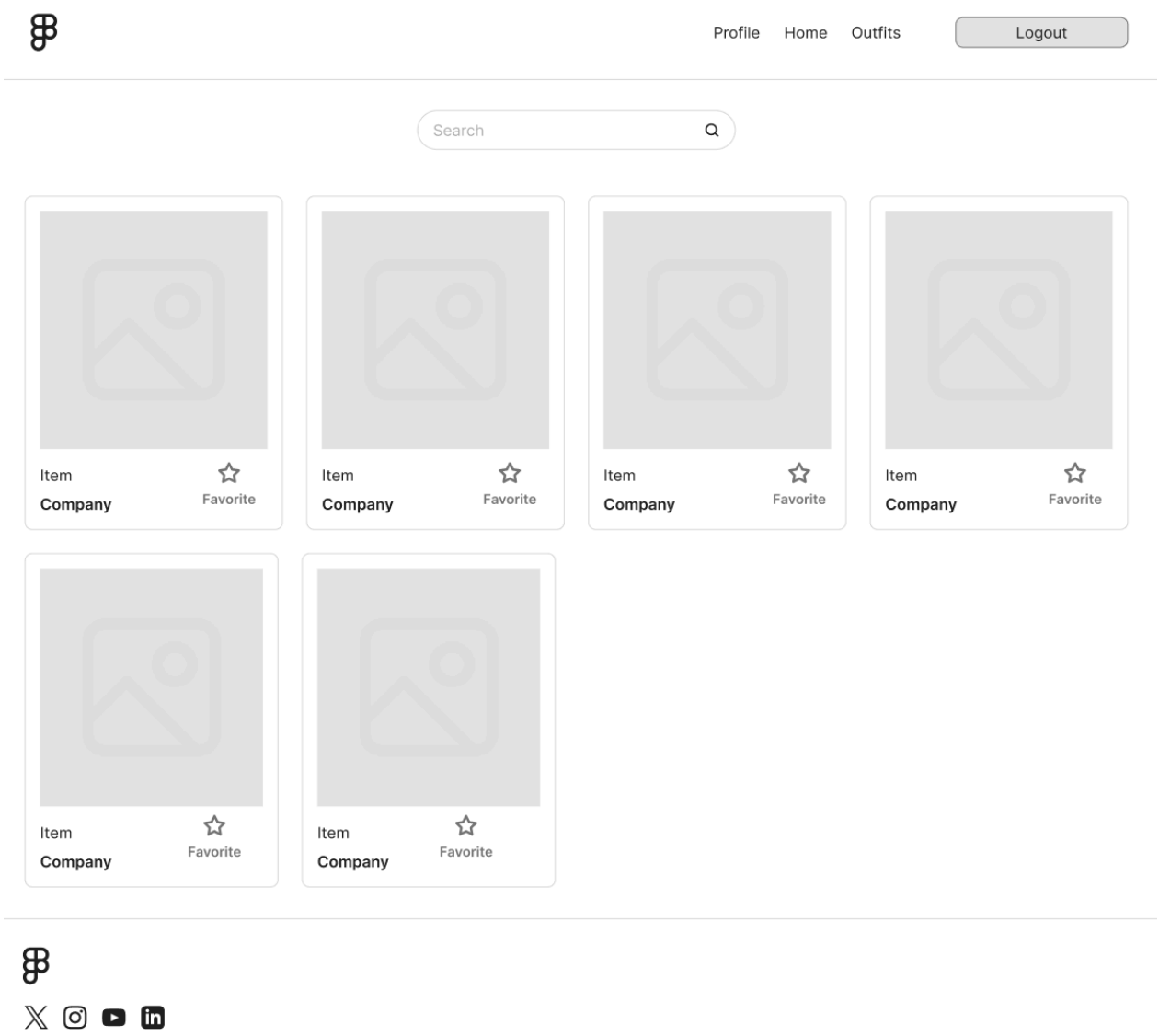
Profile:


Profile
Home
Outfits
Logout
Saved Outfits
Delete
Wishlist
Username:
Password:
Item 3
Delete
Item 4
Delete

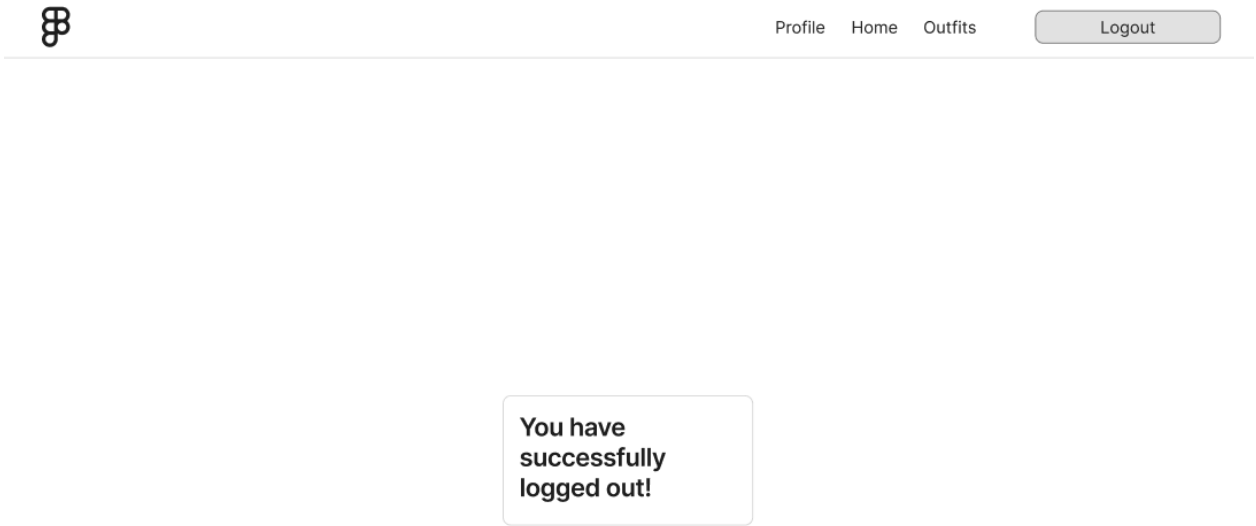
Outfit Creator:



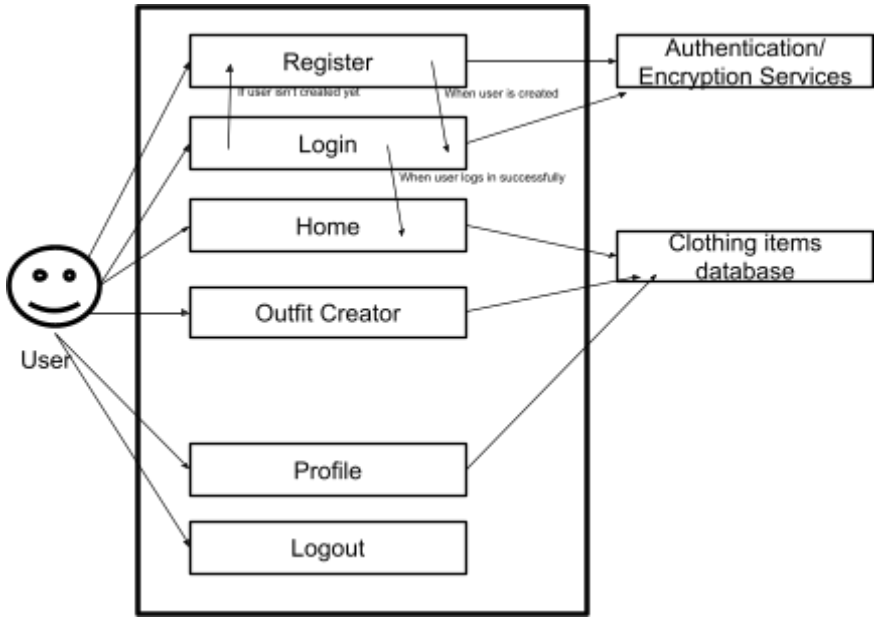
Home:


Profile
Home
Outfits
Logout
Search
Item
Company
Favorite
Item
Company
Favorite
Item
Company
Favorite
Item
Company
Favorite
Item
Company
Favorite
Item
Company
Favorite

Logout:


Profile
Home
Outfits
Logout
You have successfully logged out!

Use Case Diagram:



Testing:

Test cases were written for login and register. The use cases consist of a positive and negative case for each of the APIs. For register, we had a test case creating a unique account, then another case creating an account with the same username, which would fail due to username being a primary key. For Login, we had 3 use cases, one where the username and password were correct, one where the username was incorrect, and one where the password was incorrect. Below is a screenshot of all the tests passing.

```
web-1  | > test
web-1  | > npx mocha
web-1  |
web-1  | Server is listening on port 3000
web-1  |
web-1  |
web-1  |   Testing Register API
web-1  | Database connection successful
web-1  |     ✓ positive : /register (256ms)
db-1   | 2024-12-09 06:36:31.611 UTC [75] ERROR:  duplicate key value violates unique constraint "use
rs_pkey"
db-1   | 2024-12-09 06:36:31.611 UTC [75] DETAIL:  Key (username)=(John Doe) already exists.
db-1   | 2024-12-09 06:36:31.611 UTC [75] STATEMENT:  INSERT INTO users (username, password) values (
'John Doe', '$2a$10$00rr6QKU87bRnp9OhEdDdeRgPOcF5X.U6ZS07IYWsN16TcZfJMqCC') returning *;
web-1  |     ✓ Negative : /register. Checking invalid name (93ms)
web-1  |
web-1  |   Testing Login API
web-1  |     ✓ positive : /login. Checking User Acceptance (89ms)
web-1  |     ✓ Negative : /login. Checking invalid password (101ms)
web-1  | QueryResultError {
web-1  |     code: queryResultErrorCode.noData
web-1  |     message: "No data returned from the query."
web-1  |     received: 0
web-1  |     query: "SELECT * FROM users where username = 'Jane Woe'"
web-1  | }
web-1  | Error during login: TypeError: Cannot read properties of null (reading 'password')
web-1  |     at /ProjectSourceCode/index.js:129:64
web-1  |     at process.processTicksAndRejections (node:internal/process/task_queues:95:5)
web-1  |     ✓ Negative : /login. Checking invalid username
web-1  |
web-1  |
web-1  |   5 passing (663ms)
web-1  |
```

When testing the application with real users, the following observations were made:

- Some users immediately tried to login without registering for an account first. They received a message about an incorrect username or password, and then clicked the link to register. This is consistent with the use case where the username and password were incorrect because the profile didn't exist.
- When registering, some users tried using usernames taken by previous users. They would see a message saying username already taken, please use another. This prompted the user to choose a different unique username. This is consistent with the use case where different accounts were trying to be made with the same username.
- All users had no problems with registering with a unique username and password. This is consistent with the use case that created a unique account.
- After registering, users were directed to the login page and logged in successfully. This is consistent with the use case where the password and username were correct.
- Sometimes users accidentally had a typo in their username when logging in. This prompted the incorrect username or password to pop up. They would notice their mistake

and then login successfully. This is consistent with the use case where the username was incorrect.

- Users would sometimes type in their password incorrectly. This prompted a message that said incorrect username or password. Upon retyping their password, they were able to successfully login. This is consistent with the use case where the password was incorrect when logging in.

Deployment:

https://csci-3308-final-project.onrender.com