Team Number: 011 - 6

Team Name: JAAAC

We will be testing three features: our login get API (which takes the given username and password and retrieves the matching user profile from the users database), the registration post API (which adds the user into the users database as long as the given username does not already exist), and the home get API (which auto-populates cards with book information by pulling rows from the books database).

Unit Testing

To do our unit testing, we will be using Mocha and Chai. Using this framework, we will be testing for the following:

- Login Get
    - Must have property username
    - Must have property password
    - Username must be a string
    - Password must be a string
- Registration Post
    - Must have property username
    - Must have property password
    - Username must be a string
    - Password must be a string
    - Password must have a length of at least 8
    - Password must contain a number
- Home Get
    - Must have property title
    - Must have property category

Integration Testing

We wrote all of our front-end pages before we started working on the back-end. So to test that they are integrated correctly, we will look through our list of features and APIs and test each of them (to include drop-downs, buttons, etc.)

Features:

- Login Submission
- Registration Submission
- Clicking on NextPage logo (redirects to the home page)
- Clicking on user profile (which produces a dropdown)
    - Clicking on Home (which renders the home page)

    - Clicking on User Profile (which renders the user profile page)
    - Clicking on Recommendations (which renders the recommendations page)
    - Clicking on Logout (which renders the login page)

APIs:

- Get: /
    - Returns a welcome message used when testing our Mocha/Chai setup
- Get: /login
    - Renders the login page
- Post: /login
    - Retrieves the user information from the users database
- Get: /register
    - Renders the registration page
- Post: /register
    - Adds the user information to the users database
- Get: /home
    - Renders the home page and autopopulates a set number of book cards
- Get: /recommendations
    - Renders the recommendations page
- Get: /recommendations/genre
    - Takes the information from the user's genre selection, then autopopulates book cards where the category matches the user's choice

Our list of features and APIs is relatively short, so by dividing up

User Acceptance Testing

We would like to test the following cases:

- Users should be able to log in with correct credentials
- User authentication fails when the user provides incorrect credentials
- The login page provides the user with specific feedback about the error
- A user cannot submit the registration form without completing all the mandatory fields (username, password, and confirm password)
- Once the user has selected a category on the recommendations page, they should only see books under this category
- If no category is selected, the user should see books from an assortment of categories

Risks:

Organizational Risks:

- Though we would have liked to have additional features such as saving books and authors for future access, we may have to cut them depending on how much time is left after implementing key features. There were several extenuating circumstances that left

us with too much work for too few team members, which combined with our overall lack of experience in website development, means we will likely cut these features before the presentation.
- We may not have enough features to satisfy a 15 minute presentation window, so a lot of thought will need to be put in regarding how we present.

Technical Risks:

- We completed the Testing lab two weeks before our presentation, so we have limited time to ensure that our website can handle invalid inputs, such as putting unrecognized characters into text fields.
- We are unable to produce much data in regards to books, as we are struggling to find a way to import a CSV file into our PostgreSQL database. This impacts our ability to test the recommendation and search pages.

Business Risks:

- We do not have a rubric for this project, so may not have completed it to the grader's specifications.

Individual Contributions:

Cody Aker: I wrote the recommendations API and updated recommendations.ejs to better help with another feature.

Al Haddad: For this project milestone I wrote the "Risk" section. For the project I tinkered with the docker-compose to get it working, wrote the login get and post requests, registration get and post requests, set up the home get request, .env file, and wrote the scripts.js file to handle onload messages, and make sure passwords match upon registration.

Anna Rahn: I wrote the home API, added the auto-populating cards to our home page, moved some code to partials, updated the links in our header, set up the Mocha and Chai test cases, started commenting our individual contributions in the code, and wrote the unit testing/integration testing/user acceptance testing parts of this milestone.

Abigail Sullivan: I wrote the recommendations API to get book recommendations based on specified genres and updated the recommendations ejs to display those genre options accordingly.