San Francisco State University

SW Engineering CSC 648/848 Section 1

Student Connect

Team 5

Team Lead: Kao Saephan, Scrum Master: Sean Ibarra Diaz, Back End Lead: Luke Thilgen, Front End Lead: Collins Gichohi, Git-Master: Zacharia Angha, Front Assist: Salvador Avila.

Milestone 4

April 15th, 2024

History Table:

Milestone 1	3/4/2024
Milestone 2	3/18/2024
Milestone 3	4/15/2024
Milestone 4	5/6/2024

1) QA testing

I. Unit Test

Select 5 P1 features to be tested.

1. **User Registration Functionality:** Testing the registerUser function to ensure it properly handles new user registration. The test verifies that the function returns a success message when a new user is successfully registered and handles duplicate registrations by returning an error message.
2. **Password Update Functionality**: Validating the changePassword function to ensure that it updates a user's password correctly when given a valid username and new password. The test checks that the function returns a success message on successful password update and fails when the new password does not meet the system's security requirements.
3. **User Login Functionality**: Testing the authenticateUser function to verify that it correctly authenticates a user with valid credentials and rejects invalid login attempts. This includes checking that the function returns the correct user profile on successful login and an error message on failure.
4. **Forum Post Creation and Deletion:** Validating that the system correctly handles the creation and subsequent deletion of forum posts. This involves tests to ensure that a post can be added, is retrievable after creation, and is no longer accessible after being deleted.
5. **Class Enrollment and Unenrollment Functionality**: Testing class enrollment and unenrollment features using the enroll and unenroll functions. These tests check that a student can be successfully added to or removed from a class, verifying that the system

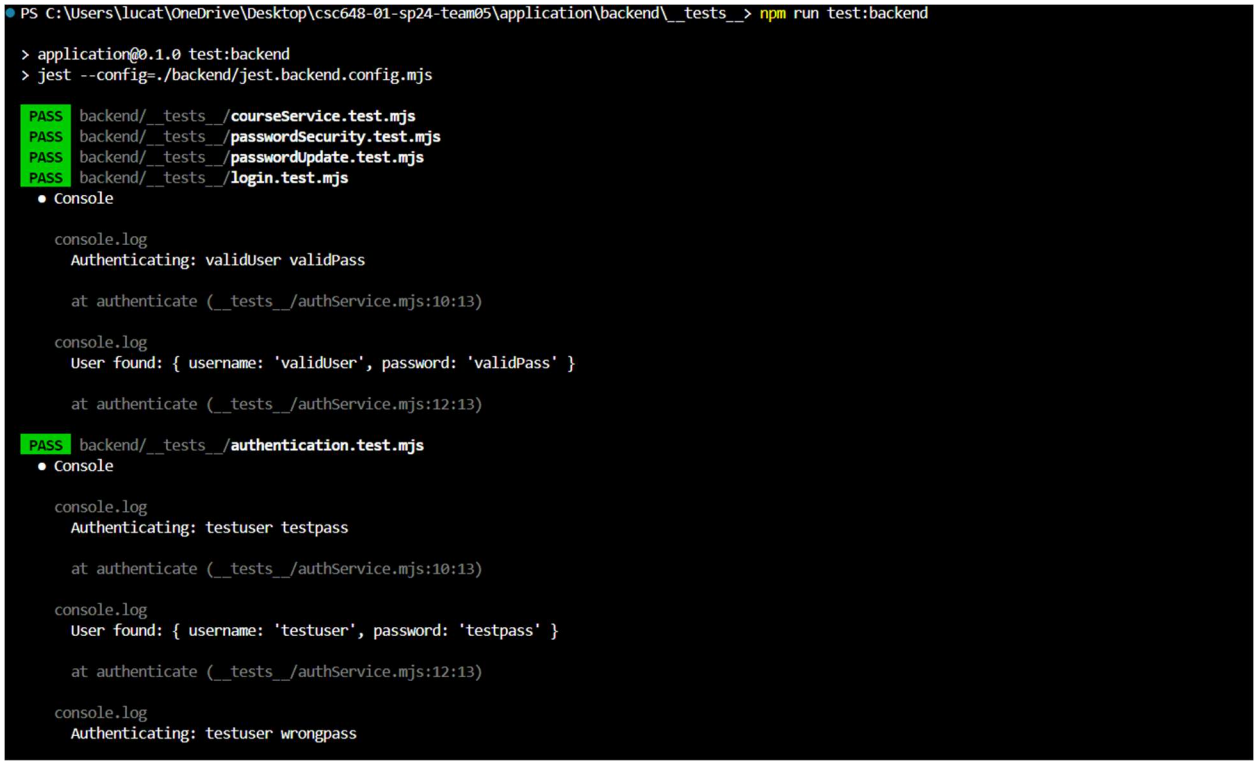
updates the class's student list appropriately and handles errors like attempting to unenroll a student not currently enrolled.

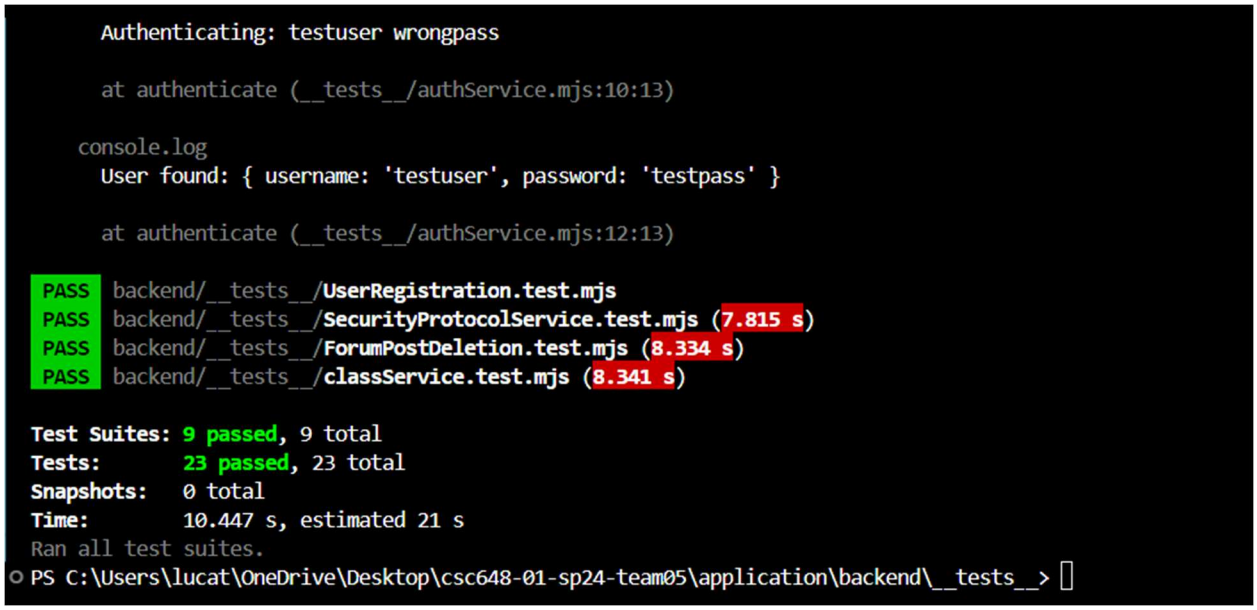
II. Integration Test

1. **Authentication Validation:** Verifying that the authenticate function successfully validates a user's login credentials (validUser and validPass).
2. **Session Creation:** Checking that a session is successfully created and retrieved via getSession after authentication.
3. **User Profile Access:** Ensuring that the user profile information can be accessed via getUserProfile after successful authentication and session creation.
4. **Password Update and Login**: Testing that after updating a password, a user can log in with the new password and not with the old one.
5. **User Registration and Login Flow**: Ensuring a new user can register and subsequently log in with the newly created credentials.
6. **Course Creation and Student Enrollment**: Testing that a course can be created and students can be successfully enrolled in the course.
7. **Course Update and Information Retrieval:** Validating that updates to course information are correctly stored and retrieved.
8. **Forum Post Deletion and Non-retrievability:** Confirming that after deleting a forum post, it is no longer retrievable.
9. **Class Deletion and Error Handling**: Testing the system's response to attempting to delete non-existent classes and unauthorized deletion attempts.

10. **Lockout Mechanism on Failed Logins:** Verifying that the system correctly locks out a user after multiple consecutive failed login attempts.

Test Results: There are nine test files (the ones with .test) but combined they have the 5 unit tests and 10 integration tests, all passed.





Github test folder: https://github.com/CSC-648-SFSU/csc648-01-sp24-team05/tree/master/application/backend/__tests__

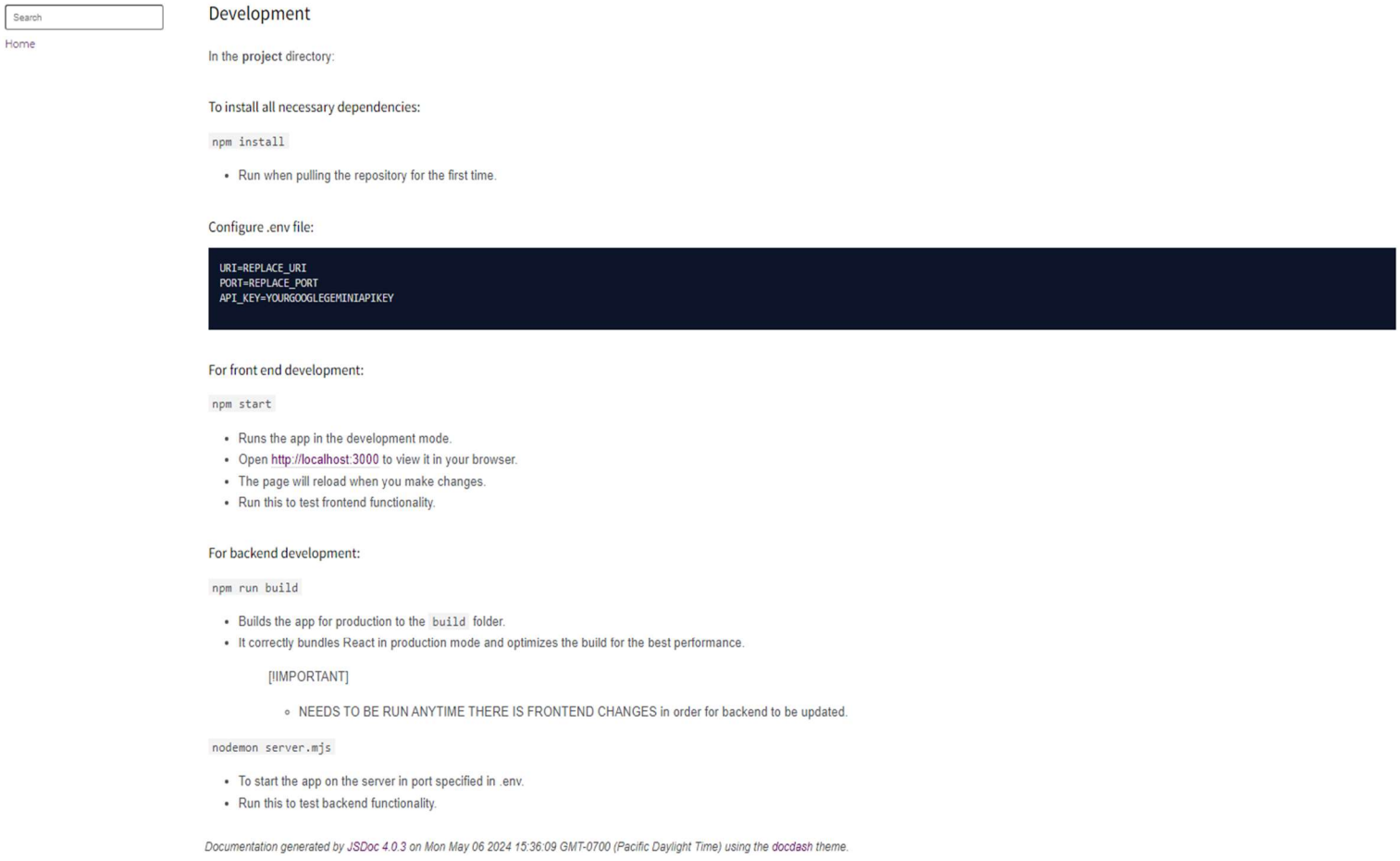
2) Coding practices:

I. A coding style.

- We used the Prettier extension to enforce our coding style. Prettier will get rid of unnecessary lines, add lines to separate code blocks, remove unnecessary spaces and indents. In addition, Prettier will wrap code blocks if needed and in a consistent manner.
- Please list source files related to 5 P1 features which demonstrate your coding style.
    1) aiTutor.js
    2) classes.js
    3) addClasses.js
    4) profile.js
    5) announcementview.js

II. [Extra Credit of 10 Pts] Documentation Generation

- For each public method/classes in source files for 5 P1 features, generate HTML documentation from code comments in the source files (refer to slides).



Search

Home

Development

In the **project** directory:

To install all necessary dependencies:

```
npm install
```

- Run when pulling the repository for the first time.

Configure .env file:

```
URI=REPLACE_URI
PORT=REPLACE_PORT
API_KEY=YOURGOOGLEGEMINIAPIKEY
```

For front end development:

```
npm start
```

- Runs the app in the development mode.
- Open http://localhost:3000 to view it in your browser.
- The page will reload when you make changes.
- Run this to test frontend functionality.

For backend development:

```
npm run build
```

- Builds the app for production to the `build` folder.
- It correctly bundles React in production mode and optimizes the build for the best performance.

  [!IMPORTANT]

  - NEEDS TO BE RUN ANYTIME THERE IS FRONTEND CHANGES in order for backend to be updated.

```
nodemon server.mjs
```

- To start the app on the server in port specified in .env.
- Run this to test backend functionality.

Documentation generated by JSDoc 4.0.3 on Mon May 06 2024 15:36:09 GMT-0700 (Pacific Daylight Time) using the docdash theme.