Joseph Bec
CS-314

Final Project Report

Code Structure and Tests Report

Project Overview

This project is a full-stack web application with a React frontend and Node.js/Express backend. The application appears to be a social messaging platform with features for user authentication, friend management, direct messaging, and group chats.

Frontend Architecture

- **React Components**: The main component is **Dashboard.js** which handles most of the application's functionality
- **Context Providers**:
    - **AuthContext** for user authentication state
    - **SocketContext** for real-time communication
- **Directory Structure**:
    - **/src/components**: UI components including Dashboard, Settings, Message
    - **/src/context**: Context providers for state management
    - **/src/utils**: Utility functions and formatters

Backend Architecture

- **Express Routes**:
    - **/routes/auth.js**: Authentication endpoints
    - **/routes/users.js**: User management
    - **/routes/friends.js**: Friend request handling
    - **/routes/messages.js**: Message management
    - **/routes/groupChats.js**: Group chat functionality
- **Database**: MongoDB
- **Socket.IO**: Implemented for real-time features

Testing Strategy

Frontend Tests

1. **Component Tests**:
    - **Dashboard.test.js**: Main test file covering basic component functionality
    - **Settings.test.js**: Tests for the settings component
2. **Extended Dashboard Tests**:
    - **Dashboard.additional.test.js**: Basic rendering tests
    - **Dashboard.functions.test.js**: Tests for key functions (fetchFriends, fetchMessages, sendMessage)
    - **Dashboard.modals.test.js**: Tests for modals (friend requests, image upload)
    - **Dashboard.events.test.js**: Tests for socket events

Backend Tests

1. **Unit Tests**:
    - **api.test.js**: Tests API endpoints
    - **auth.test.js**: Tests authentication functionality
    - **database.test.js**: Tests database operations
    - **messages.test.js**: Tests message handling
    - **group.test.js**: Tests group chat functionality
2. **Integration Tests**:
    - API integration tests using SuperTest
3. **Mock Server**:
    - **mockServer.js** for testing API endpoints without a real database

Additional Features

- **I did the whole thing full stack**
- **sending images**
- **live typing indicators (work for both DMs and group chats)**
- **read receipts**
- **notification sounds**
- **browser tab unread message count**
- **in application unread message counter**

Testing Approaches

1. **Jest & React Testing Library**: For frontend component testing
2. **Mock Functions**: Extensive use of Jest mocks for context providers, axios, sockets
3. **Function Testing**: Direct tests of individual functions to isolate functionality
4. **Event Testing**: Simulating socket events and user interactions
5. **Cypress**: End-to-end tests appear to be implemented in the Cypress directory

Challenges Encountered

All the features presented their own unique challenges, but I specifically had great trouble trying to implement the notifications. Getting tab notification to function properly was challenging, what added to my struggle was also getting the notification to clear properly when a message was viewed. Luckly this tied in well with the read receipts feature, which I had already implemented by this point.

I also had trouble getting the group chats to function, specifically with the live indicators. As they had to show up for everyone currently viewing the group chat which added much more complexity than their implementation in DMs. In addition to this, getting group chats to function at all was also troubling and required lots of troubleshooting.

And lastly, testing in general was very challenging simply due to the sheer volume of things to test. It always felt like I could just keep writing more, and more tests. Also, when I changed something, numerous tests would fail causing me to enter into an infinite loop of debugging. In addition, I also set out to implement feature testing on top of unit testing. I chose to do this by using Cypress, this was very, very hard. I got it to work thus ensuring that the basic, core features operate properly. But cypress and Jest together easily took up most of my development time invested in this project.

Conclusion

The difficulty of this project was greatly increased because of my decision to work on it as a full stack application and my other poor decisions of adding way too many features that I thought sounded cool. Overall, I am happy with the final product as it has many utilities and works as expected.