Software Requirements and Design Document

For

Group <6>

Version 3.0

Authors:
Frankie Messina
Zach Porcoro
Raul Rodriguez
Andrew Stade
Peter Vasiljev

1. Overview (5 points)

The application is intended to give the user the ability to quickly view their progress towards completing their degree. Any outstanding degree requirements will be able to be viewed at a quick glance. With inspiration from FSU's flowchart system, this application will provide a checkpoint for those remaining classes that are required by the computer science department. The project will incorporate a user-friendly UI, with easy to navigate buttons and a design that focuses on accessibility.

2. Functional Requirements (10 points)

1. Login (High priority): The functionality of the application requires the user to have an existing account. Authentication will be handled in a secure fashion by utilizing Firebase's API calls. The user will also be able to reset their password if needed through the use of a password reset link that is sent to a valid email address. Login functionality can also be performed with the use of various login providers (Google, GitHub, Yahoo).
2. Registration (High priority): User's who do not have an existing account or do not use a login provider are able to register a new account. Firebase is used to perform account creation within the database.
3. Store class data for users (High priority): Using Firebase, the application will store the user's class data. The user will select which classes they have already completed after creating their account for the first time. The user will be able to update their class data as often as they would like through the UI.
4. Compare class data to CS flowchart (High priority): There will be a view, similar to the CS flowchart, that will allow the user to quickly see what classes they have and have not taken.
5. Display degree progress (High priority): The system will provide a progress bar/percentage that indicates at a glance how close the user is to meeting graduation requirements.
6. Course information page (Medium priority): This page contains information about the courses that are offered by the Computer Science department and is based on the requirements for a Bachelor of Science degree.
7. Recommend a potential schedule for the next semester (Medium priority): The user will receive a suggestion of which classes they should take for their next semester.
8. Professor ratings (Medium priority): Shows the user what professors usually teach a certain class and shows their rating.
9. Additional CS Resources Page (Low priority): Has links to useful FSU Computer Science webpages. It also contains the contact information for the Computer Science advisors.
10. Polished User Interface (Low priority): Focused on user accessibility with sleek design.

3. Non-functional Requirements (10 points)

<u>Security:</u>
Authentication is performed whenever the user navigates to any page. This means that the user cannot access certain pages (e.g. home, account settings) without being authenticated. If a user attempts to access a page they are not authorized to view, they are redirected to the login screen.

<u>Reliability:</u>
We use Firebase for many different things in our project. Not only do we use it to login and create new users, but we also use it to store and retrieve data about the user's. So, if Firebase goes down, then the functionality of our application will be interrupted.

<u>Performance:</u>
The application must feel responsive to the user and there should not be any issues with website performance. Usage of proper Web/React development practices will help to ensure that the user has a smooth experience. Some dynamic information is stored into the browser's local storage after retrieving

the information from the database, so that loading times can be shortened after loading the app for the first time.

4. Use Case Diagram (10 points)

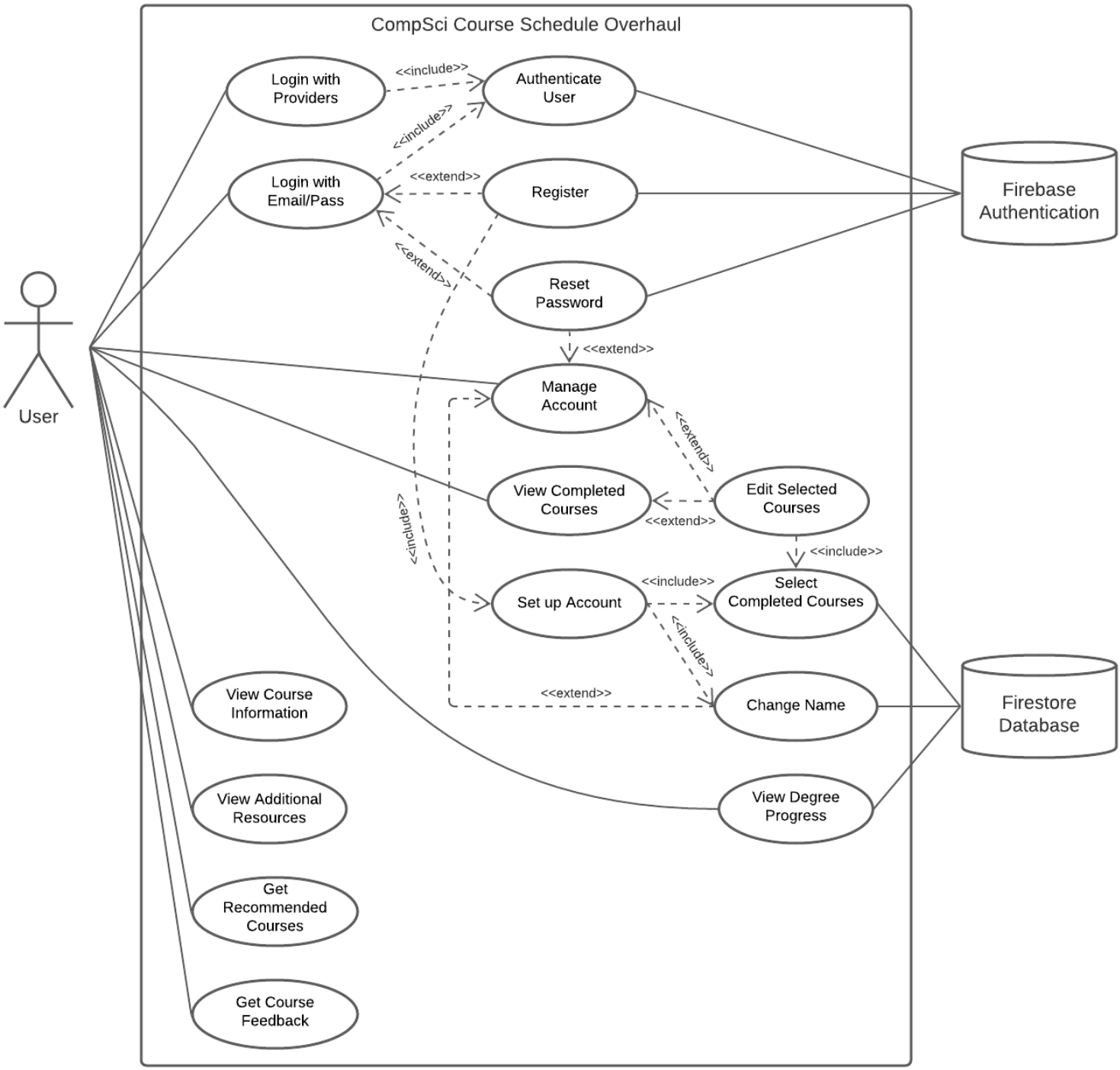

Figure 1. CompSci course schedule overhaul use case diagram

4.1. Use Case Textual Descriptions

Disclaimer: We provided extended textual descriptions for the most important use cases and only provided some brief descriptions for the other use cases.

1. Name: Login with Email/Password
2. Description: Allows the user to log into their account using Firebase Authentication.
3. Participating Actors: User, Firebase Authentication
4. Preconditions:
    - User has a valid account
    - App has connection to Firebase
    - Email must be in valid email form
5. Postconditions:
    - User must be authenticated with Firebase
    - User is redirected to the homepage
6. Basic Flow of Events:
    1. User enters an email and password
    2. User clicks on the login button
    3. The app checks that the email/passwords are valid
    4. User's credentials are authenticated with Firebase
    5. The user is redirected to the homepage
7. Alternative Flows:
    - Invalid email or password entered in step 1
        - Displays error messages
    - Invalid credentials in step 3
        - User remains on the login screen and may retry
    - User does not have an account
        - User clicks the register button
    - User forgot account password
        - User clicks on the reset password button
    - App cannot connect to Firebase
8. Special Requirements:
    - N/A

---

1. Name: Register
2. Description: Allows a new user to create an account with Firebase Authentication.
3. Participating Actors: User, Firebase Authentication
4. Preconditions:
    - User must have a valid email address
    - App has connection to Firebase
    - Email address must not already be registered in the database
    - Password must be at least 6 characters
    - Both password and password confirmation must match
5. Postconditions:
    - Account gets created in Firebase
    - User is redirected to account setup
6. Basic Flow of Events:
    1. User enters email and password
    2. User confirms the password they entered again
    3. User clicks on the register button
    4. The app checks that the email/passwords are valid
    5. The app sends the credentials to Firebase for account creation
    6. The user is redirected to account setup

7. Alternative Flows:
    - Email already exists in step 1
        - Displays error message
    - Password is too short in step 1
        - Displays error message
    - Passwords don't match in step 2
        - Displays error message
    - App cannot connect to Firebase
    - User clicks on the return to login button
        - Redirects to login page
8. Special Requirements:
    - N/A

---

1. Name: Set up Account
2. Description: Allows the user to set their name and completed classes and have them stored in Firestore
3. Participating Actors: User, Firestore Database
4. Preconditions:
    - User must not have any data in Firestore yet
    - The user is logged in
    - The user is a BS in Computer Science student
5. Postconditions:
    - The name and selected classes are saved in Firestore
    - User is redirected to the home page
6. Basic Flow of Events:
    1. User enters their first and last name
    2. User selects the classes they have and are currently taking
    3. User clicks the Save button
    4. The app checks that the user entered something for their name
    5. The app sends the data to Firestore
    6. The user is redirected to the home page
7. Alternative Flows:
    - No first or last name was entered in step 1
        - Displays error message
    - App cannot connect to Firebase
8. Special Requirement:
    - N/A

---

1. Name: Login with Providers
2. Description: Allows the user to login using a Google, GitHub, or Yahoo account, using Firebase additional login provider APIs.

---

1. Name: Authenticate User
2. Description: Authenticates the user's credentials using Firebase Authentication.

---

1. Name: Reset Password
2. Description: Allows a user to reset their password. This uses Firebase's password reset feature. This will send an email to the user which will allow them to set their new password.

1. Name: Manage Account
2. Description: Let the user change their name, reset their password, or change their completed courses. When they change their name, it updates their new name in the Firestore database. When they reset their password, it uses the same feature as described in the description above. Finally, when changing their completed courses, they can modify their selected courses and have the new ones saved in Firestore.

---

1. Name: View Completed Courses
2. Description: Shows the user what courses they have completed, one ones they can take next, and what ones they still need to take the prerequisite for.

---

1. Name: Edit Selected Courses
2. Description: Redirects to the page that allows the user to select their completed courses.

---

1. Name: Select Completed Courses
2. Description: Allows the user to change what courses they have completed and saves the new information in Firestore.

---

1. Name: Change Name
2. Description: Allows the user to change their name and save the new name in Firestore.

---

1. Name: View Degree Progress
2. Description: Gets what percent of the way done the user is from Firestore and shows that percentage in a progress bar.

---

1. Name: View Course Information
2. Description: Displays all the required BS Computer Science classes and all their information.

---

1. Name: View Additional Resources
2. Description: Provides links to FSU CS resources and also provides contact information for advisors.

---

1. Name: Get Recommended Courses
2. Description: Provides the user with recommended courses to take for the next semester based on their current degree progress and how many courses they feel comfortable taking.

---

1. Name: Course Feedback Data Page
2. Description: Allows you to easily search FSU's SPCI website by course code, instructor name, or both.

5. Class Diagram and/or Sequence Diagrams (15 points)

Disclaimer: It was a little tricky for us to map the structure of our React app to the format of a traditional class diagram. We did our best to represent the components used in our app and how they relate to each other.

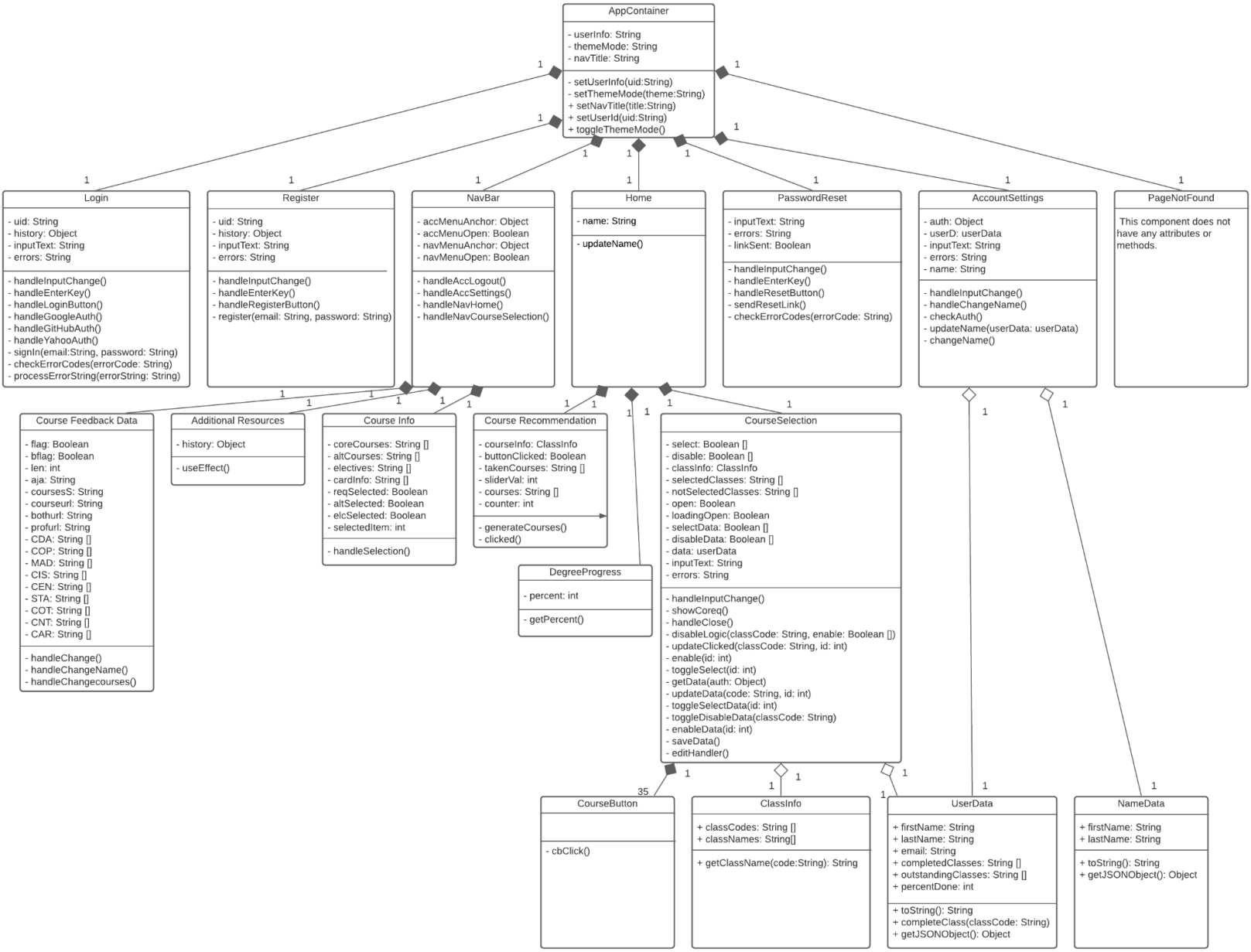

Figure 2. Class diagram

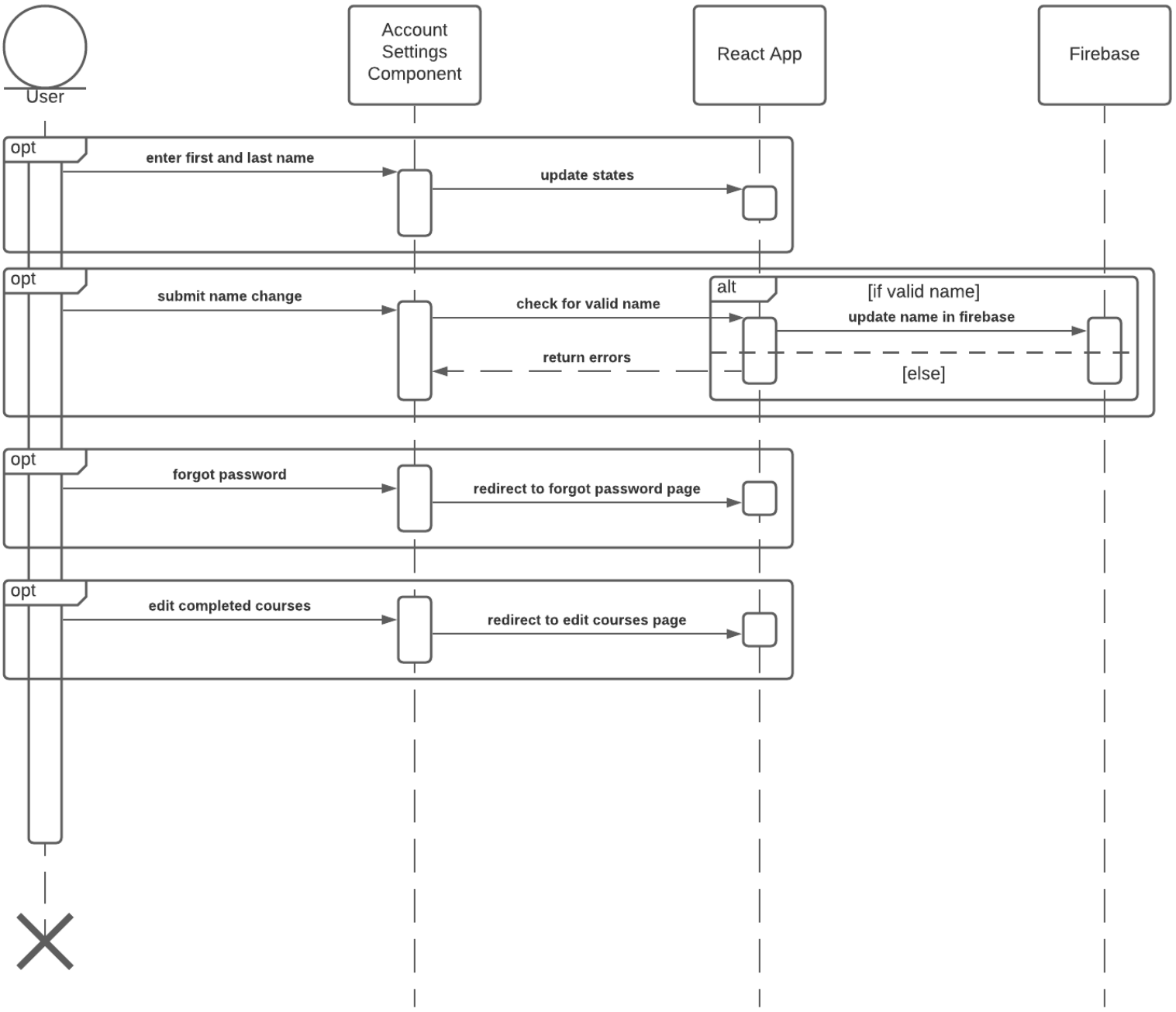

Figure 3. Account settings component sequence diagram

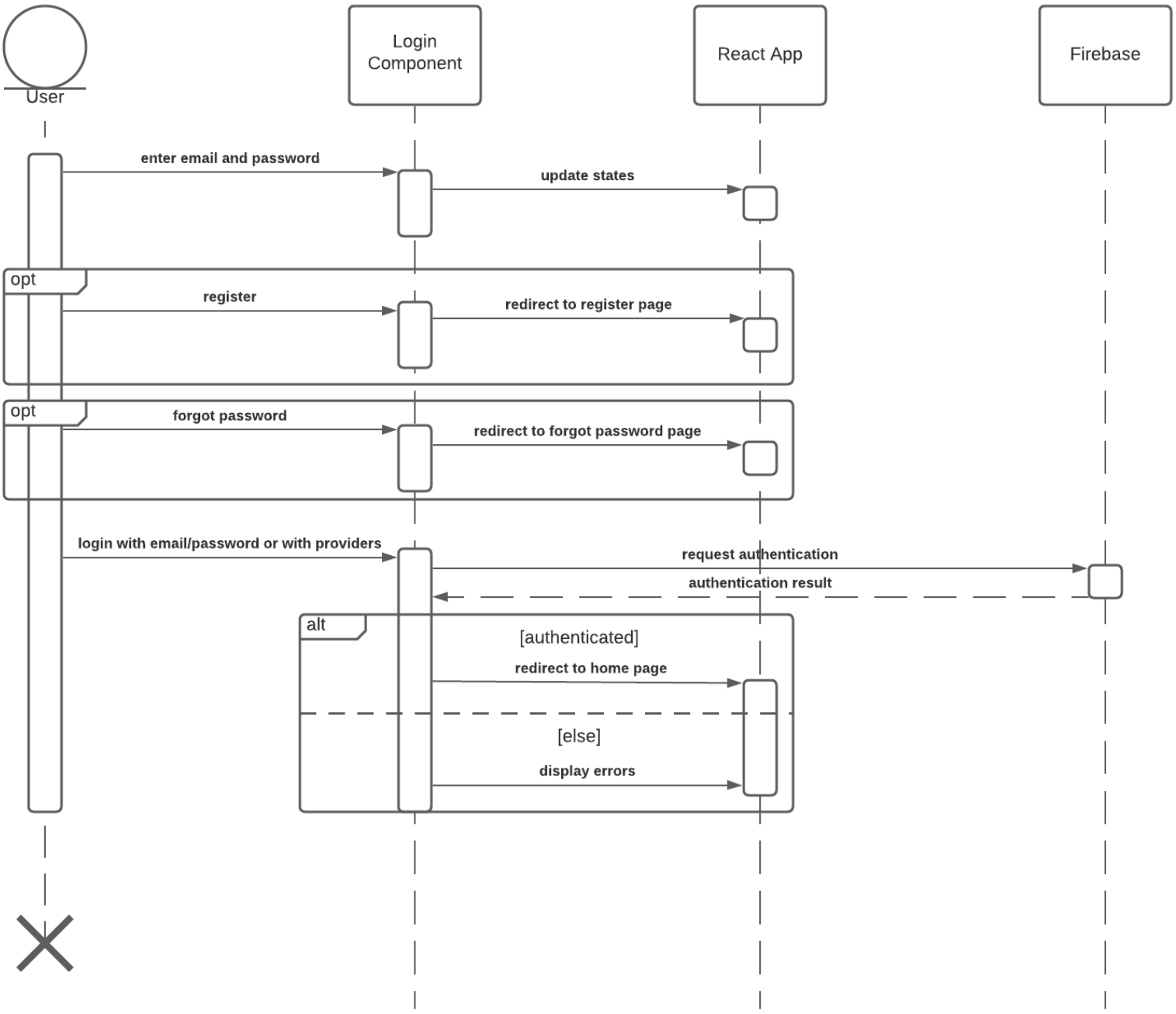

Figure 4. Login component sequence diagram

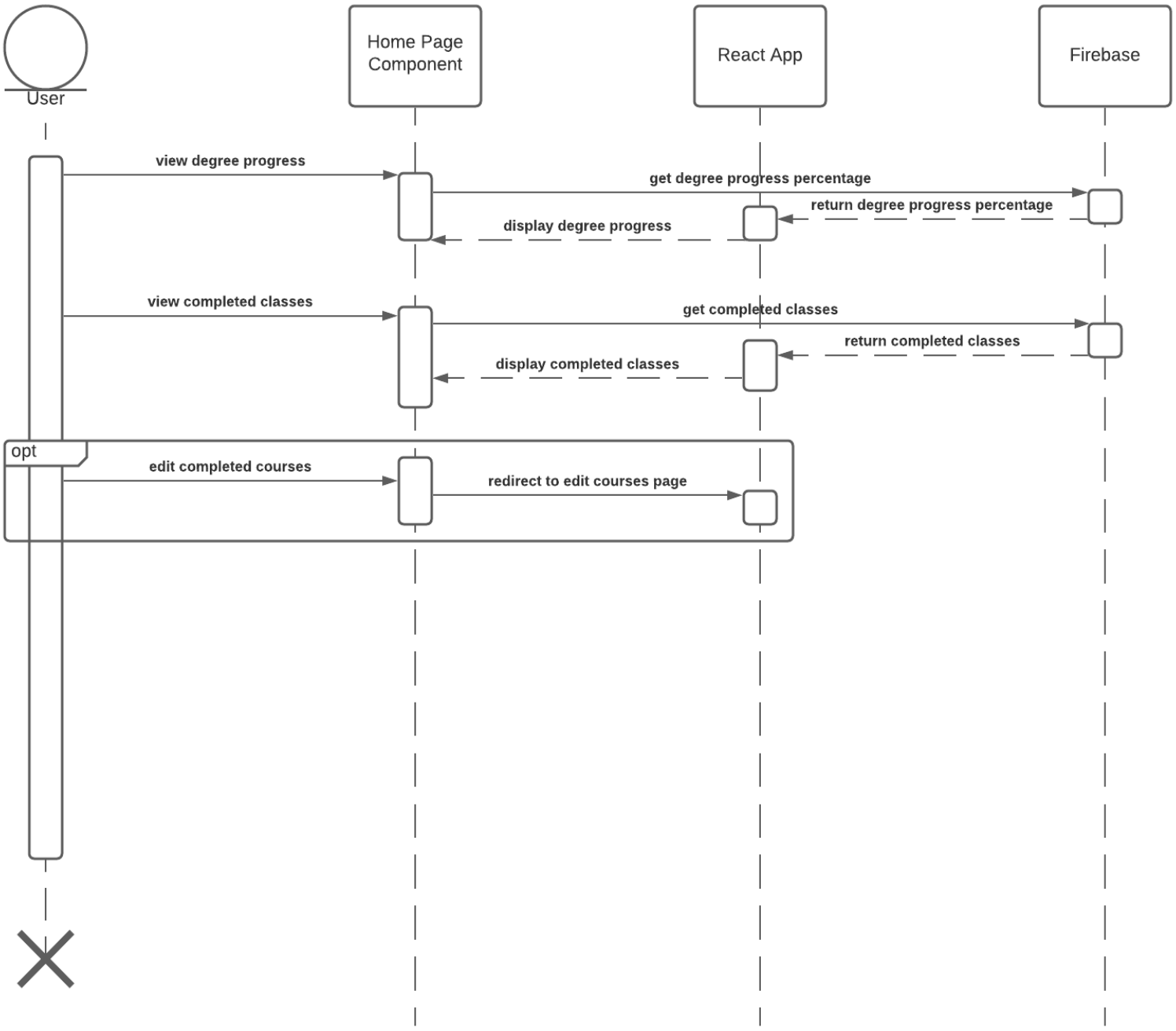

Figure 5. Home page component sequence diagram

6. Operating Environment (5 points)

The application will operate through most web browsers, with a main focus on desktop or laptop functionality (Windows, Linux, and MacOS). Given that this application is web-based, there are no other software components to consider for peaceful coexistence.

7. Assumptions and Dependencies (5 points)

- The application must be able to connect to Firebase in order to perform many of the features outlined in the functional requirements
- The application is designed for students that are pursuing a Bachelor of Science in Computer Science degree
- The application is designed to work with modern browsers and may not be compatible with older browsers such as internet explorer.
- We use MaterialUI for most of the user interface for the project. So, if there is a change/update to it, it could affect our project.