**Software Requirements and Design Document**

**For**

**Group 31**

Version 2.0

**Authors**:

Aiden Livingston

Dustin Haire

Maria Penalver

McKenna Warrick

William Gomer

# Overview (5 points)

FlashCardio is a web-based flashcard and study application designed to improve the way users organize, create, and learn with flashcards. Unlike other flashcard apps, FlashCardio allows users to store and organize flashcards and decks within a flexible hierarchical folder structure. It integrates a spaced-repetition algorithm that customizes study sessions, helping focus on less familiar cards to optimize learning and retention. The application aims to provide an intuitive and simple experience for managing complex knowledge.

The system will support user accounts to enable the personalized storage of flashcards and progress tracking. Users will be able to create flashcards, organize them into folders, and study them in an efficient/ customizable manner. As the application evolves, additional features such as AI-generated flashcards and a repository of shared notes will be considered to further enhance the learning experience.

# Functional Requirements (10 points)

**Allow users to create flashcards:** Flashcards are the core feature of the application, and users must be able to create them to begin using the app effectively. **Priority:** High

**Allow users to organize flashcards into folders and decks:** Organizing flashcards into folders is essential for the usability of the app, enabling users to manage large collections of cards in a structured manner. **Priority:** High

**Allow users to create and delete folders and decks:** Users need to manage their flashcards efficiently by creating and deleting folders and decks as necessary for their learning process. **Priority:** High

**Implement a spaced-repetition algorithm to optimize the study of flashcards:** The spaced-repetition algorithm will help users retain information over time by showing flashcards based on their familiarity with the content. **Priority:** High

**Allow users to track their progress on specific sets of flashcards:** Progress tracking is essential for users to understand which cards they need to focus on and to monitor their learning. **Priority:** High

**Allow users to create an account and associate their flashcards with their user profile:** Account creation enables personalized user experiences, including data storage and synchronization across multiple devices. **Priority:** Medium

**Allow users to delete individual flashcards:** Users must be able to remove cards that are no longer needed or relevant to their learning. **Priority:** Medium

**Provide a study viewer that presents flashcards for review in study sessions:** The study viewer is crucial to delivering the actual study experience, and it must be clear, easy to navigate, and effective in presenting flashcards. **Priority:** Medium

**Allow users to adjust the learning algorithm and select specific folders or decks for study sessions:** Providing flexibility to users by allowing them to customize study sessions enhances the UX, especially for those managing large flashcard collections. **Priority:** Medium

# Non-functional Requirements (10 points)

**Respond to user actions within 500ms:** Quick response times provide a smooth UX, particularly when managing large flashcard and folder collections. **Priority:** High

**Ensure data security for flashcards and personal information through encryption and authentication:** Security is critical for user privacy and trust in the application. Encryption and secure login methods will safeguard sensitive data. **Priority:** High

**Support scalability for many users and flashcards:** Scalability ensures that the app can grow with the user base without performance degradation, especially as users create and manage large numbers of flashcards. **Priority:** High

**Maintain 99.9% uptime, with minimal disruptions to user access:** High availability is essential for a cloud-based application, ensuring users can access their data and study without interruptions. **Priority:** High

**Compatible with major web browsers, including Chrome, Firefox, Edge, and Safari:** Ensuring compatibility across multiple browsers increases user base and accessibility. **Priority:** Medium

**Provide an intuitive and easy-to-navigate user interface with clear instructions:** A user-friendly interface is critical for engagement, especially for users who may not be tech savvy. **Priority:** Medium

# Use Case Diagram (10 points)

**Actors:**

1. **User** (Primary Actor)
   * A person who interacts with the system, creating and studying flashcards.
2. **Admin** (Secondary Actor)
   * An administrator who might manage users or oversee content

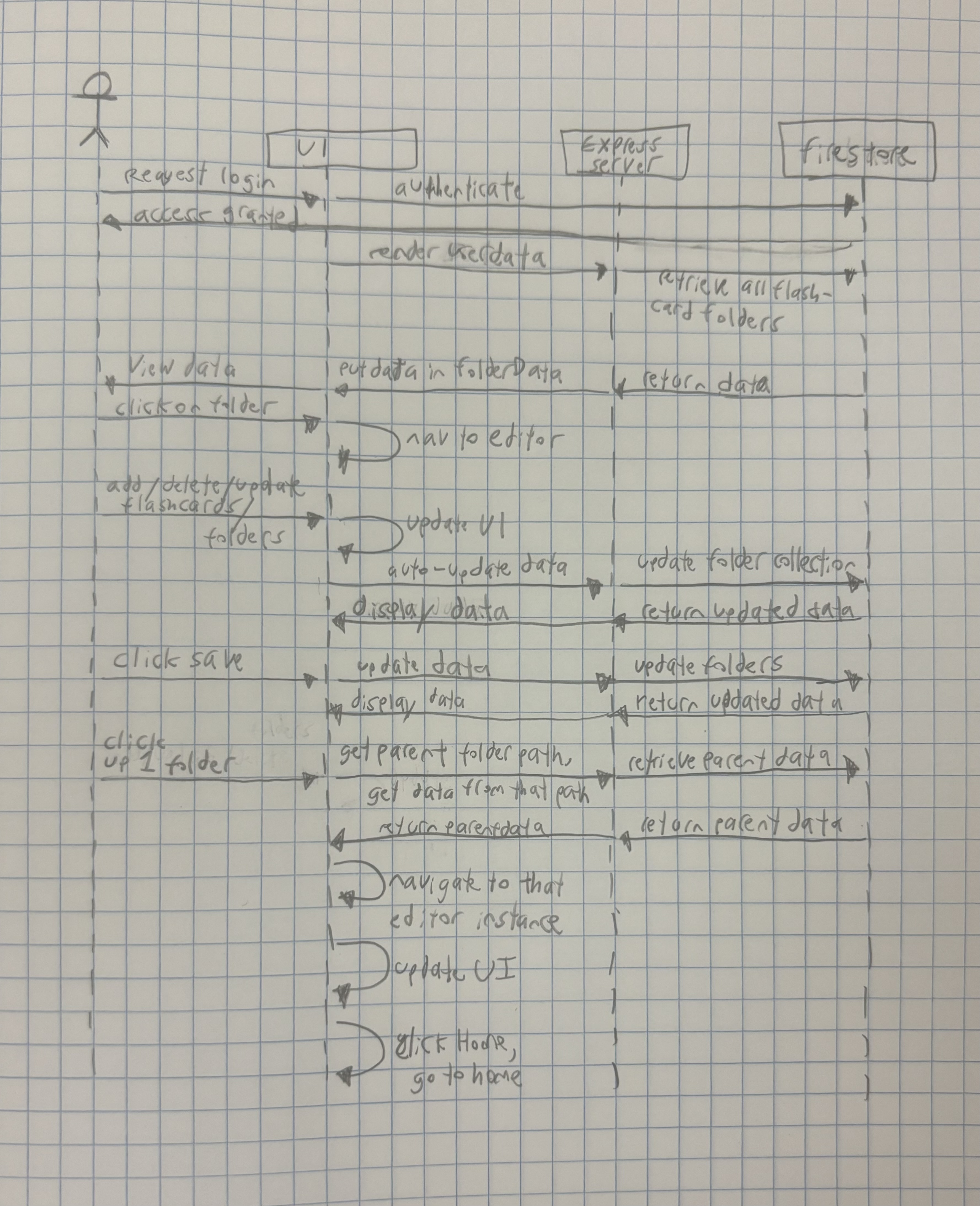
**Use Cases:**

1. **Sign In** 
   * **Description**: Users can sign in or sign up for an account to access and save their flashcard data.
   * **Actor**: User
   * **Precondition**: The user must have a valid account for sign-in or must create a new account to sign up.
2. **Create Flashcards**
   * **Description**: Users can create new flashcards, adding questions and answers.
   * **Actor**: User
   * **Precondition**: User is logged in.
3. **Edit Flashcards**
   * **Description**: Users can edit existing flashcards to modify the content.
   * **Actor**: User
   * **Precondition**: User has already created flashcards.
4. **Delete Flashcards**
   * **Description**: Users can delete flashcards they no longer need.
   * **Actor**: User
   * **Precondition**: User has created flashcards.
5. **Create Folders**
   * **Description**: Users can create folders to organize their flashcards into categories.
   * **Actor**: User
   * **Precondition**: User is logged in.
6. **View Folders and Flashcards**
   * **Description**: Users can view their folders and flashcards.
   * **Actor**: User
   * **Precondition**: User has created or added flashcards to folders.
7. **Study Flashcards**
   * **Description**: Users can study their flashcards by reviewing the questions and answers, possibly with a quiz mode or flashcard review system.
   * **Actor**: User
   * **Precondition**: User has flashcards created and organized.
8. **Delete Folders**
   * **Description**: Users can delete folders they no longer need.
   * **Actor**: User
   * **Precondition**: User has created folders.
9. **Admin: Manage Users**
   * **Description**: Admin can manage user accounts, such as adding, banning, or updating user information (optional).
   * **Actor**: Admin
   * **Precondition**: Admin has permission.
10. **Admin: Monitor Content**
    * **Description**: Admin can review user-generated content for appropriateness or to enforce terms of use (optional).
    * **Actor**: Admin
    * **Precondition**: Admin has permission.

A graph paper with a diagram

Description automatically generated

# Class Diagram and/or Sequence Diagrams (15 points)

 *A screenshot of a computer

Description automatically generated*

# Operating Environment (5 points)

FlashCardio will operate as a web application, accessible through modern browsers such as Google Chrome, Mozilla Firefox, Microsoft Edge, and Safari. The application will be platform-agnostic, meaning it can run on any device with a browser, including desktop computers, laptops, tablets, and smartphones.

The backend will run on a Node.js server hosted in a cloud environment, with Firebase being used for database management and authentication. The system will require no additional software installations beyond a browser and will integrate with other browser-based applications.

# Assumptions and Dependencies (5 points)

**Assumptions**:

* The target users will have access to reliable internet connections and modern web browsers.
* Firebase services will remain available and maintain compatibility with the technologies we are using.
* The chosen cloud hosting platform for the backend will support Node.js without disruptions or significant performance issues.
* Users will primarily interact with the app in environments where minimal processing and storage demands on their devices will suffice.

**Dependencies**:

* The project relies on libraries and frameworks including React.js, Express.js, Firebase, and Node.js. Any major updates or deprecations in these technologies could impact development or maintenance.
* The project depends on the continued availability and support of Firebase for real-time database functionality and user authentication.
* Deployment will depend on a compatible cloud hosting provider to ensure scalability and reliable performance.
* Performance optimizations will depend on the integration of caching strategies and query efficiencies to address current bottlenecks.