**Progress Report**

**- Increment 2 -**

**Group #31**

# Team Members

|  |  |  |
| --- | --- | --- |
| Aiden Livingston | atl21g | lvngstn |
| Dustin Haire | Dmh23a | DHaire92 |
| McKenna Warrick | Mlw21c | MLW21C |
| Maria Penalver | Mpb22b | marypenalver |
| William Gomer | wjg21a |  |

1. **Project Title and Description**

FlashCardio is a flashcard application designed to provide an efficient, and customizable learning experience. Unlike other flashcard apps, FlashCardio emphasizes advanced organizational tools and intuitive study modes to help users grasp complex concepts and manage extensive knowledge bases with ease.

Key features include a hierarchical file structure allowing users to create and organize flashcards, decks, and folders, and a study mode powered by a spaced-repetition algorithm to optimize retention. Additional functionalities like user progress tracking, personalized databases, and (in the future) AI-generated flashcards aim to simplify the learning process and cater to diverse user needs.

1. **Accomplishments and overall project status during this increment**

During this increment, we successfully implemented the hierarchical file structure, allowing users to organize flashcards, folders, and decks intuitively. The UI components for flashcards, folders, and pages have also been fully designed and developed. Additionally, we finalized the theme, fonts, and color scheme, providing the application with a cohesive and professional appearance.

On the backend, we developed basic API functionalities to perform CRUD operations for both flashcards and folders. These APIs enable the interactions necessary for managing content within the app. Progress indicated the project remains on track, with the core infrastructure in place to support the next stages of development.

1. **Challenges, changes in the plan and scope of the project and things that went wrong during this increment**

One of the primary challenges during this increment was refactoring/ optimizing the foundational source code from the previous increment. During the initial creation of the logic and components, there were various unintended inefficiencies in the code that were ignore in favor of creating the code foundation. Examples include:

* In the nested folders field for each folder, they were stored as the actual folder data instead of links to the folder in the database. This made folder data rendering very slow and inefficient.
* The folder and flashcard CRUD operations were implemented in the react frontend, which is generally not preferred by basic code organization principles.
* Folder state was inconsistent throughout each UI component. In the Editor component, there was just one ‘Folder’ variable, but when passed to other components, they fields in the composite ‘Folder’ variable was split in inconsistent ways. This made the state confusing to change/ work with

However, after a secondary look, all of the major points of unhealthy code were refactored. After abstracting the CRUD login into the APIs, another challenge that emerged was the recursive folder rendering feature returning slower-than-desired API response times, which may impact the user experience when managing large or deeply nested folder structures. This bottleneck reflects a common issue in software development, where performance tuning becomes necessary after building out complex data hierarchies. To address this, we are exploring caching mechanisms and query optimizations for the next increment.

1. **Team Member Contribution for this increment**

*Please list each individual member and their contributions to* ***each of the deliverables in this increment*** *(be as detailed as possible). In other words, describe the contribution of each team member to:*

* 1. The **progress report*:***
     1. Aiden: Sections 1, 2, 3, 4, 5, 6, 7
  2. the **requirements and design document**:
     1. Aiden:
  3. the **implementation and testing document**:
     1. Aiden: Sections 1, 2, 3, 4, 5
  4. the **source code**:
     1. Aiden
        1. Implemented recursive folder structure
        2. Created Folder – Edit Mode UI components
        3. Abstracted folder and flashcard CRUD operations into Express.js REST APIs
        4. Implemented dynamic URL paths for each folder/ deck
     2. Dustin
        1. Brainstormed application deployment methods
        2. Strategized on how to implement multiple users
     3. Maria
        1. Finalized fonts and themes for the application design
  5. *the* ***video or presentation***

1. **Plans for the next increment**

**Functional Goals:**

Study Mode:

* Implement a new page for users to select desired folders and flashcards to study them within the study mode.

Spaced Repetition:

* The system shall implement a spaced repetition algorithm to influence study sessions based on user familiarity with each flashcard.
* The system shall allow users to study specific folders or a mix of flashcards across multiple folders.

User Progress Tracking:

* The system shall display the user's progress on selected flashcards or folders, including familiarity ratings and mastery percentages.
* The system shall store data on each flashcard, such as familiarity levels, priority, and (wishlist) prerequisites.

Account Management:

* The system shall allow users to create and manage accounts tied to their flashcards and progress data.

Optional Features (Wishlist):

* The system may include functionality to auto-generate flashcards using AI from typed notes or spreadsheets.
* The system may feature a university-wide repository where students can share and unlock lecture notes.

**Non-Functional/ Architectural Goals:**

Performance:

* The system should load user flashcard data within 500 milliseconds on average. (Retrieval currently around 750ms)
* The Flashcard study mode should handle up to 500 flashcards without noticeable performance degradation.

Usability:

* The system interface should be intuitive and require no more than 5 minutes for a new user to understand basic functionality.
* The app should emphasize simplicity and clarity, minimizing configuration complexity for study sessions.

Scalability:

* The system should support up to 1,000 users without significant performance issues.
* The hierarchical file structure should efficiently manage flashcards and folders at various levels of nesting.

Security:

* User accounts and associated data should be securely stored, with passwords hashed and encrypted.
* The system should prevent unauthorized access to user data.

Maintainability:

* The codebase should be modular, enabling the addition of new features like AI-generated flashcards or shared repositories with minimal changes to existing functionality.

Data Integrity:

* All user-created flashcards, folders, and progress data should be reliably stored and retrievable, even in the event of system crashes.

1. **Stakeholder Communication**

**Subject:** Project Update: Progress and Next Steps

Dear Stakeholders,

We are happy to share an update on the progress of *FlashCardio – gains for your brain!* and outline our next steps as we move toward delivering a functional, efficient, and user-friendly flashcard application.

**Progress Highlights:**

The project has made significant strides. Our team has successfully implemented the hierarchical file structure that will allow users to organize flashcards and folders intuitively. Additionally, we have finalized the app's theme, fonts, and color scheme, giving it a polished, professional look.

On the development side, all core UI components for flashcards, folders, and pages have been completed. Basic API functionalities have been implemented to enable users to create, read, update, and delete (CRUD) both flashcards and folders. These features provide the foundation for a seamless user experience.

**Current Focus Areas:**

As we continue development, our primary objectives for the next phase include:

Study Mode: We are building a study viewer where users can select flashcards and study them using a spaced-repetition algorithm. This feature will optimize learning and retention and is central to the app's value proposition.

Deployment: We aim to deploy the application in its current form on the web, allowing users to explore and interact with the product. This milestone will also facilitate valuable feedback to refine the app further.

Personalized Data Storage: Each user will have a dedicated database instance to ensure that their flashcard data remains private and unique.

Performance Optimization: We are addressing API response times, particularly in recursive folder rendering. Our goal is to reduce response times to under 500ms, ensuring a fast and responsive experience.

**Challenges and Resolution Plans:**

One of the key challenges we've encountered is optimizing the recursive folder rendering, which is currently slower than desired. This type of performance bottleneck is common in hierarchical data systems and reflects the complexity of balancing functionality and efficiency in real-world software development. We are leveraging caching strategies and database query optimizations to address this issue.

**Next Steps:**

As we integrate these enhancements, we remain committed to delivering a reliable, high-quality product that meets the needs of our users. We are confident these efforts will strengthen FlashCardio’s functionality and user experience, moving us closer to our vision of a streamlined and effective learning tool.

Thank you for your continued support and interest in our project. Please don’t hesitate to reach out with any questions or feedback.

Best regards,

The FlashCardio Development Team

1. **Link to video**

*Paste here the link to your video.*