Alabass Alkhrsany, Blendi Grajqevci, Teemu Laasio, Aleksi Putkonen

Project Planning Group 3

1. Project Overview

Project Title
Online Flashcard Learning Application

Problem Summary
Many students struggle with remembering study materials and organizing their learning in an effective way. Traditional studying methods are not always flexible enough, and teachers may not have enough time or tools to provide additional study materials that support self-paced learning. The goal of this project is to address these issues by providing a simple digital tool that supports studying through flashcards.

Intended Audience / Users
The main users of the application are students, including university, vocational students, and high school students, who want to improve their learning experience and memory retention. The system is also intended for teachers who want to support students by creating and sharing flashcard sets as additional resources.

Main Features / Components
The core functionality of the project is based on flashcard learning. Users can study using flashcards, create their own flashcard sets, or use pree-created flashcard sets for different subjects. The application focuses on being easy to use, accessible, and suitable for self-paced learning.

2. Project Objectives

The core objectives of the project include creating an effective and all-round accessible learning tool for students and all other learners of numerous needs. The project aims to deliver a free, reliable and intuitive tool that students and teachers can effectively utilize along with the traditional learning methods. The use case objectives of the project include:

- **Flashcard usage**
  The most important and relevant goal for the project is to deliver an application that enables its users to seamlessly use any flashcards created on the platform, whether that is by an approved teacher, or a pre-created set of cards.

- **Flashcard creation**

Alongside flashcard usage, it is as important of a feature to be able to create new sets of flashcards for use. The project aims to develop the capability of a separate teacher role, that allows users with the aforementioned role to create their own sets of flashcards. These sets of flashcards can then be shared for use via a link or be found in the browsing section of the application if set to public visibility.

- **Administration & moderation**
  When allowing users to create their own flashcards, it is also important to have a dedicated role for moderation and administration, as user provided content can never be guaranteed to be appropriate and rule following. The moderator/administrator role will be implemented simply as a parallel role to the teacher and student roles.

There are also a number of technical objectives the project aims to achieve. The biggest ones consist of scalability, maintainability and the possibility for easily implementing new features in the future.

- **Scalability**
  As for this project, we aim to deliver a product that can be used by multiple people simultaneously without breaking apart or losing performance. However in this phase, we are not expecting to build an application that would support upwards of tens or hundreds of thousands concurrent users.

- **Maintainability**
  The goal is to make the codebase altogether well structured, modular, as well as readable and well documented. The application logic will be built so that it follows good OOP principles, and is easily maintainable for the future being.

3. Scope And Deliverables

The project focuses on implementing a web-based flashcard learning application. The primary objective of the application is to maximise ease-of-use while minimizing unnecessary functionality and feature-bloat. The scope is restricted exclusively to flashcard learning. The following table defines the project deliverables.

Category	**Included**	**Excluded**
User Management	Account creation, login / logout and role-based access for Teachers and Students.	External third-party authentication like **OAuth**.
Learning Features	Flashcard creation, quiz (flashcard sets) taking and basic quiz performance analysis.	Game features such as: leaderboards, streaks, badges.

Classrooms	Student and class management. Class creation and uploading of materials.	Real-time communication between students and teachers. (Websocket)
Uploadable content (Classrooms)	Uploadable PDF-material	Videos, custom learning environments.
Administration	Dashboard for content moderation, deleting of quizzes, teacher status validation, support ticket system and user management.	AI-driven content moderation.
Platforms	The product will be exclusively on the web.	Mobile-app, downloadable application.

4. Project Timeline

The goal of the project is to launch a Minimum Viable Product (MVP) within 8 weeks, divided into four two-week sprints. For organizing tasks, we will be using Trello.

4.1 Sprint 1: Planning (15.01 - 29.01)

The initial phase of the project focuses on planning and initial setup.

- **Product vision and planning.**
  Implementation of the product vision and project plan documents.

- **Technology stack.**
  Choosing the appropriate technologies that will be used in the project, this includes the backend, frontend and necessary dependencies.

- **Project management setup.**
  Create the initial project backlog and fill it with user stories, tasks and key features

- **GitHub repository**
  Create a GitHub repository for the project.

- **UI/UX design**.
  Designing the application UI with Figma and other tools.

4.2 Sprint 2: Core Infrastructure and Testing (29.01 - 12.02)

The second phase of the project focuses on the establishment of the technical foundation by developing the database, starting the UI and integrating unit tests.

- **Database implementation.**
  Design and create the relational database (MariaDB) and connect the Jakarta Object-Relational Mapping (ORM) to the database.

- **Initiate UI development.**
  Start the development of the frontend based on the Figma design and implement basic interactivity like authentication and routing.

- **Testing and Quality Assurance.**
  Integrate JUnit for backend testing and JaCoCo for code coverage. Integrate Playwright for End-to-End (E2E) testing and Vitest for unit testing in the frontend.

4.3 Sprint 3: CI/CD, Feature Extension and Docker (12.02 - 05.03)

The third phase of the project focuses on integrating Jenkins for CI/CD, improving automated testing, and developing functionality.

- **Integrate Jenkins.**
  Implement a CI/CD pipeline using Jenkins to automate the build, test and deployment process.

- **Feature development and authentication.**
  Implement the core product features listed in the project backlog and plan. This involves authentication, the backend, and the frontend. Ensure the feature is well tested for quality assurance.

- **Advanced testing**
  Ensure all new features are thoroughly tested. Test with Playwright for E2E user journeys.

- **Functional review**.
  Ensure the newly developed features are working as expected and can be demonstrated. Integrate the backend with the frontend and document the code functionality thoroughly.

- **Containerization**
  Create and test local Docker images for both backend and frontend services.

4.4 Sprint 4: Finalizing (05.03 - 12.03)

The focus of the final phase of the project is to finalize the product and push it to Docker Hub.

- **Finalize the product**
  Ensure all features are fully implemented and tested. Fix any remaining issues and ensure all data is validated and secure.

- **Docker**
  Containerize the entire application, create the Dockerfile, build the image and test it locally. After testing and ensuring the product works, proceed to push it to Docker Hub.

- **Sharing**
  Share the product and receive feedback.

5. Resource Allocation

As a four-person team our goal is to assign our working resources in a way that makes the project run smoothly. We will split our team into front-end and back-end developers or alternatively we will have part of the team doing full-stack development to even out the workload. Specific tasks will be assigned once the full feature list is finalized, ensuring that responsibilities match each members' strengths.

- **Team members:**

  Alabass Alkhrsany — programmer
  Blendi Grajqevci — programmer
  Teemu Laasio — programmer
  Aleksi Putkonen — programmer

Each team member will also participate in writing the necessary reports and documents for the project and will take part in creating and presenting the final presentation. One member will be assigned as the Scrum master for each two-week sprint of the project. The order is determined randomly and each member will be Scrum master for only one sprint.

Programming will be done on each team members' personal laptop on an Integrated Development Environment, or IDE. The team has a shared GitHub repository that will be used for version control and sharing code. The software tools used for the project will be Docker, Kubernetes Minicube, and Jenkins.

Possible external resources the team would need for the project are programming help and information from the internet and global coding community or help from the software providers for the software tools used in the project.

6. Risk Management

For every project it is important to think about potential risks beforehand so you can plan ahead and minimize the likelihood of said risks occurring. Here is a list of potential risks with their descriptions, likelihood of happening, impact on the project, and strategies to mitigate them.

1. A team member falls sick

- **Description:** A team member falls sick and is unable to participate in the project for a short or a long period of time
- **Likelihood:** High
- **Impact:** Low/Medium (depending on the length of absence)
- **Mitigation strategies:** This is a difficult risk to mitigate among students who study in crowded environments where flu and viruses may spread, but one thing that can be done is to manage the project and workload well to minimize stress and keep team members happy and healthy.

2. A team member calls it quits

- **Description:** For one reason or another a team member decides to leave the team in the middle of the project.
- **Likelihood:** Low
- **Impact:** High
- **Mitigation strategies:** Create a good working environment, keep the team morale high, listen to each other so everyone can be heard and make their mark in the project, and manage the workload to minimize stress.

3. Product not delivered in time

- **Description:** The project schedule fails in a way that the team cannot provide a functioning product before the final deadline.
- **Likelihood:** Medium
- **Impact:** High
- **Mitigation strategies:** Thorough planning, active participation, regular progress reviews, good time management and sticking to the schedule.

4. Product does not function as intended

- **Description:** The project does not work as it should or fails to perform its core function.
- **Likelihood:** Low
- **Impact:** High
- **Mitigation strategies:** Thorough planning and sticking to the plan.

5. Missing features

- **Description:** The final product is missing some of the features that were planned to be implemented.
- **Likelihood:** Medium
- **Impact:** Medium
- **Mitigation strategies:** Thorough planning, regular progress reviews, avoiding feature bloat, and good time management.

These are some of the more common risks on a software development project but the team should be mindful of the possibility of other risks too and be prepared to adjust and react accordingly.

7. Testing and Quality Assurance

To achieve the highest possible quality, the business logic of the application will be implemented using test-driven development (TDD). Unit tests shall be written before logic parts, ensuring that modules and methods work as expected from the get-go. The target is to achieve code coverage of over 70%. Tools like JaCoCo and JUnit will be used to technically achieve these goals.

We also want to gather user input and feedback on the application during development, so that it can be truly tailored for the target audience.

8. Documentation and Reporting

Documentation plays an important role throughout the project to ensure clarity, transparency, and good collaboration within the team. All project related documents, such as the product division, sprint planning reports, and other planning materials, are stored in the project´s github repository.

During each sprint, progress and decisions are documented to keep track of the development process. Sprint planning and sprint review documents are used to describe what was planned, what was completed, and what can be improved in the next sprint.

In addition to written documentation, the project results will be presented at the end of the course. The documentation supports both the development process and the final evaluation by providing clear feedback of the project and its outcomes.