
Bilkent University - Computer Science

CS492 - Senior Design Project II

Final Report

T2401

Ömer Fırat Bekiroğlu 22002239
Faruk Uçgun 22003016
Cenk Merih Olcay 22002408
Enes Bektaş 22002401
Akif Erdem Tanyeri 22003537

2025 Spring

# 1. Introduction

Medicine is one of the study fields that requires a vast amount of memorization. Medical school students face an excessive amount of information during their studies. Research conducted by Dattathreya and Shillingford found that medical school students have trouble remembering all of this information [1]. Although the causes of this problem differ, research shows that most students avoid textbooks and rely mostly on lecture slides [2]. Consequently, medical students learn their lecture content passively, which leads to ineffectiveness in their studies and performance loss on their exams [3]. ReMediCard.io is a mobile application seeking to address these issues and facilitate medical school students' learning.

One of the solutions for memorization problems is flashcard, which is a study tool that typically consists of a small card with a question, term, or prompt on one side and the answer or explanation on the other. Flashcards are used to promote active recall, where learners test themselves repeatedly to reinforce memory. Although there are a number of applications available for medical students currently, none of them truly tackle the difficulties associated with creating flashcards. Widely used programs like Anki and Quizlet offer limited customization options and tools for creating flashcards by hand, but they demand a large time commitment from students, which can be difficult for medical students with heavy workloads. Anki in particular is unable to generate flashcards automatically from handwritten notes and other legacy teaching materials. Quizlet still relies on the user to manually generate flashcards, even though it features a document scanning tool that allows users to choose particular words from the provided document.

ReMediCard.io is an AI-powered flashcard application intended to help medical students study more efficiently, in recognition of the difficulties that come with medical school. Our product reduces the manual work needed by automatically creating flashcards from lecture notes, videos, and images by utilizing state-of-the-art computer vision, machine learning, and natural language processing techniques. With this, students can concentrate more on studying and less on preparation.

Furthermore, ReMediCard.io uses adaptive learning algorithms to customize study sessions based on each user's performance and progress. This personalized approach enhances learning efficiency, and also aligns with the diverse needs of medical students. The integration of these cutting-edge technologies is the innovative part of our project, differentiating us from similar products in the market.

The remainder of this report outlines the ReMediCard.io in detail. Section 2 presents the requirement details of the system. Section 3 includes a detailed description of the proposed software design and architecture of our system. Section 4 explains the development and implementation details. Section 5 represents the test cases that the final product is assessed against. Section 6 provides our maintenance plan in detail. Section 7 discusses the various aspects of the project, including constraints of the project, the standards we followed, our ethical

considerations and teamwork details. Finally, Section 8 presents the conclusion and future plans.

## 1.1 Purpose of the System

The main purpose of the system is to enhance the academic success and study efficiency of medical students by modernizing how they interact with their learning materials. Medical education is content-heavy, and requires students to master and retain vast amounts of information across multiple disciplines. Traditional study methods, such as reading textbooks, reviewing lecture notes, and manually creating flashcards, are inefficient and time-consuming. These challenges frequently result in passive learning, and suboptimal exam performance.
The purpose of ReMediCard.io is to provide a solution to these issues by introducing an intelligent, AI-powered flashcard and quiz generation platform. The system makes it possible for medical students to convert traditional lecture materials — including handwritten notes, PDFs, slides, videos, and audio recordings — into structured, and interactive study content as flashcards and quizzes. By using the technologies in natural language processing, speech recognition, computer vision, and large language models (LLMs), the application automates the content transformation process that students would otherwise do manually.

Furthermore, ReMediCard.io offers a personalized learning experience by incorporating adaptive algorithms, particularly spaced repetition techniques, to ensure that study sessions are optimized for memory retention. These algorithms track user performance over time and adjust review intervals based on individual learning curves, helping users learn information more effectively and remember for longer periods.

The system's primary objective is to diminish the difference between studies with traditional lecture materials and using technological capabilities, providing an intelligent and effective method of studying for medical students.

## 1.2 Definitions, acronyms, and abbreviations

**AI (Artificial Intelligence):** AI is the field of computer science that investigates how machines provide intelligent responses to encountered situations.

**API (Application Programming Interface):** Application Programming Interface, is the set of rules that allows a software's functionality or data to be used by third-party software or clients.

**Computer Vision:** Computer vision is a field of AI that uses machine learning and neural networks to teach computers and systems to derive meaningful information out of visual inputs.

**CPU (Central Processing Unit):** The CPU is the main processing unit that is responsible for computations and management of the infrastructure of a computer.

**Flashcard:** Flashcards are cards with both sides containing information related to each other. Usually, one side is for questions, and the other side is for answers. These cards are used for studying.

**GDPR (General Data Protection Regulation):** An EU regulation that sets guidelines for collection and processing and ensuring personal data privacy and data security.

**GPU (Graphic Processing Unit):** GPUs are specialized strong processors that are primarily used for rendering graphics. GPUs are also widely used for machine learning practices and computations.

**IEEE (Institute of Electrical and Electronics Engineers):** An organization that sets standards and promotes studies in engineering fields, especially electrical, electronics, and computer science.

**IEC (International Electrotechnical Commission):** An organization that publishes safety and compatibility standards for engineering fields.

**ISO (International Organization for Standardization):** An international organization that publishes global quality and safety standards.

**LLM (Large Language Model):** Large Language Models are advanced AI systems trained on huge datasets to understand and generate human-like text.

**Machine Learning:** Machine learning is a branch of AI that focuses on using data and algorithms to enable AI to imitate the way humans learn.

**Natural Language Processing (NLP):** NLP is the field of computer science that focuses on making machines understand and use the languages used by humans.

**NoSQL:** NoSQL is a scalable and flexible data storage solution that is not fixed with a schema.

**OCR (Optical Character Recognition):** OCR is a technology for extracting text data from images or scanned documents.

**Spaced Repetition Algorithm:** Evidence-based learning technique where newly introduced or harder problems are shown more frequently. On the other hand, older and well-understood problems are shown to the user less frequently.

**Speech-to-Text:** A technology that converts spoken language into written text using automatic speech recognition systems.

**SQL (Structured Query Language):** SQL is a programming language used to interact with relational databases.

**UML (Unified Modeling Language):** A standardized modeling language to help software engineers visualize and document the artifacts of software systems.

**User Interface (UI):** UI is the design of the application that is offered directly to the user to interact with.

### 1.3 Overview

For medical school students who have a hard time remembering vast amounts of information in their studies, ReMediCard.io is a mobile application which facilitates the high memorization based learning process for medical students. For this purpose, the system utilizes AI-based technologies to transform traditional lecture materials, such as lecture notes, handwritten documents, and audio or video recordings, into interactive and engaging study tools.

The application's core feature is the intelligent flashcard and quiz auto-generator, which takes lecture materials as inputs, captures the key terms and concepts, and creates easy-to-use flashcard decks or quizzes. This system employs Large Language Model(LLM) based Machine Learning(ML) services to extract compact but significant information from the traditional lecture materials.

To further enhance learning efficiency, ReMediCard.io integrates a spaced repetition algorithm based on the Beta distribution. This algorithm dynamically adjusts review intervals by estimating the probability that a user will recall a given flashcard. The algorithm updates itself with the historical information of the users' past results. This adaptive scheduling ensures that study sessions are optimized for long-term memorization.

In addition to these core features, the application supports social and motivational components. Users can share their decks and quizzes with their friends or make them publicly accessible for all of the users. The platform also allows students to set personalized study goals by defining deadlines and repetition intervals. Users receive timely notifications to help them stay on track and complete their study goals before the set deadline.

Overall, ReMediCard.io aims to provide a structured, intelligent, and collaborative environment that empowers medical students to study more efficiently.

# 2. Requirements Details

## 2.1. Functional Requirements

- The user can register/login by using their email.
- The user can register/login by using their Google accounts.
- The user can reset their password using their email.
- The user can delete their account.
- The user can create custom flashcard decks.
- The user can create/edit custom flashcards and add them to the decks.
- The user can attach an image to each side of a flashcard.
- The user can upload a video lecture record and generate a deck of cards automatically

- The user can upload a voice record and generate a deck of cards automatically
- The user can upload their lecture notes and generate a deck of cards automatically.
- The user can upload a video lecture record and generate a quiz of multiple choice questions automatically
- The user can upload a voice record and generate a quiz of multiple choice questions automatically
- The user can upload their lecture notes and generate a quiz of multiple choice questions automatically.
- The user can be shown flashcards based on the spaced repetition algorithm. For example, if they struggled last time to remember the backside of the flashcard, they can expect to encounter it more often when reviewing the corresponding deck or vice versa.
- The user can get feedback for the solutions of the practice test questions in the application.
- The user can get indirect hints while solving test questions
- The user can search flashcards, questions or media(if available) based on keywords.
- The user can monitor their statistics for a certain deck or a quiz after completing it.
- The user can share their flashcard decks with other users
- The user can share their multiple choice quizzes with other users
- The user can see and use other user's flashcard decks if the publishing user consents to share.
- The user can see and use other user's multiple choice quizzes if the publishing user consents to share.
- The user can set study goals and receive notifications about these goals.
- Users can see and save other users' decks or quizzes from the discover page if shared publicly
- Users can rate the decks or quizzes shared publicly on discover page by liking or disliking them
- Users can receive notifications for certain actions, for example reminder of study goals or for indicating that auto generation is completed
- Users can change the system language between Turkish and English when they want

## 2.2. Nonfunctional Requirements

### 2.2.1. Usability

To make our project easy to use and accessible for everyone we focused on building user friendly UI, ensuring easy adaptability and responsive design. For this we completed the following improvements:

- A clean and simple UI is applied to make the application easy to understand

- All main features—flashcards, quizzes, discover, and study goals—can be accessed directly from the homepage with one step
- Each page includes a navbar on the bottom of the screen to easily navigate to certain pages
- Generation requests are async and does not interrupt user's experience
- The instructions provided in the application are easily understandable
- Consistent design elements are used (colors, buttons, fonts) so users could get used to the app more easily
- Google authentication service is provided
- UI design is responsive for different mobile screen sizes and devices
- Users get simple and understandable feedback about their actions
- Explanatory warnings are included for incorrect and incomplete actions

### 2.2.2. Reliability & Availability

To make sure the system works reliably, we focused on data consistency, smooth updates, and preventing interruptions. Our aim was to provide users with a stable and safe experience at all times. The following measures were taken:

- The application runs with high availability and stays accessible under normal conditions.
- In case of any crash or issue, the system can be restarted quickly without major downtime.
- User data is stored accurately and kept up to date at all times.
- Backups are taken regularly to prevent any data loss.
- If data loss happens, the latest backup can be used to recover the system fast.
- We used version control throughout the development process to track and manage all updates.
- Updates are tested before release and are applied without affecting existing user data.
- Before deployment, the system is tested in different situations to make sure everything works as expected.
- After deployment, the system is monitored to catch performance or reliability problems early.

### 2.2.3. Performance

ReMediCard.io features frequent interaction with users and gives instant responses to users. For smooth and uninterrupted experience in this setting, high performance is essential. To achieve this high performance we focused on optimizing user interactions, choosing the right hosting environment, and improving system efficiency and following steps are taken

- The system is designed to respond quickly to user interactions, such as navigating between screens or sending requests, ensuring a seamless user experience.

- Requests to AI services (like LMM or speech-to-text) are designed asynchronous, so users can keep using the app while requests are handled in the background.
- AI and speech-to-text services are hosted on spot group EC2 instances, which provide cost-effective compute resources while maintaining good performance.
- For storage, we use Amazon Aurora for structured data and S3 for larger assets such as audio and image files. These services help in fast data access and scalability.
- The rest of the system infrastructure is built on AWS EC2 which allows us to utilize the high speed connections inside the AWS infrastructure when transferring data from S3 storage or databases.
- To manage background tasks and communication between services, we use AWS SQS and SNS, which help maintain responsiveness and system stability under varying loads.
- We also followed performance best practices in the frontend, like minimizing unnecessary renders and using efficient data-fetching techniques.

### 2.2.4. Supportability & Localization

We want ReMediCard.io to be accessible to use for every individual interested in the project. To achieve this, we focused both on support tools like documentation and compatibility and localization efforts like language options. The following measures were taken:

- We prepared detailed, well structured and easy to read documentations that explained how the system works, how it is designed, and how different parts interact.
- Technical reports and update notes are written to make future maintenance easier.
- Users can get help or report problems easily via support channels such as contact us pages on our website and project itself
- The application is compatible with major mobile operating systems, and it is tested to work properly on recent OS versions.
- We added multi-language support, starting with English and Turkish, so users can use the system in their preferred language.
- Our infrastructure allows new languages to be added easily when needed.

### 2.2.5. Extensibility & Scalability

We designed our system to be able to grow and improve according to user needs and demands. New features, tools, or services may be added in the future, that is why creating an extensible and scalable design was important for us. This way, we can improve the platform continuously while maintaining stable and efficient performance. For this aim the following measures were taken:

- The system architecture is modular, allowing new features to be added without affecting existing components.
- Core components have been developed independently of each other to reduce the risk of functionality degradation during updates

- The database is scalable,  new structures or content can be added without a major redesign.
- As the number of users grows, the system can handle increased load using scalable AWS services like EC2, Aurora, and S3.
- Spot group EC2 instances are used for AI and Speech-to-Text services to allow flexible compute resource management.
- Queueing and notification services are used to manage different parts of the system in a scalable and decoupled way.
- Computational resources are dynamically allocated based on user activity, which helps reduce latency and avoid service interruptions.

### 2.2.6. Security & Privacy

System handles user data and personalized study content, thus ensuring privacy and protecting the system against unauthorized access were among our top priorities. We aimed to build a secure and trustworthy platform by applying secure authentication, encryption, and safe data handling. The following measures were taken:

- Google OAuth 2.0 is used for secure user authentication, which also allows integration with third-party services in a safe way.
- User data is encrypted both in transit and at rest, ensuring confidentiality during storage and communication.
- All database operations are designed to be atomic, meaning that each operation is either fully completed or fully rolled back to prevent data inconsistency or corruption.
- Sensitive actions (such as login or password reset) are handled through secure and validated endpoints and require email verification.
- AWS infrastructure services like Aurora and S3 provide encryption and secure access control by default, further enhancing system-level security

### 2.2.7. Maintainability

In order to keep the system alive and open to improvement in the long term, maintainable system design is aimed. This allowed future improvements, fixes, additions and deletions can be made without breaking or interrupting the working system. Thus our team took the following measures to support maintainability:

- The system follows a modular and service-oriented architecture, so each component can be developed, tested, and updated independently.
- When a new feature is added or a bug is fixed, only the related module needs to be modified, reducing the risk of affecting unrelated parts.
- Version control is handled with Git, which allows us to track all code changes, collaborate efficiently, and revert to previous versions when necessary.

### 2.2.8. Marketability

ReMediCard.io focused on the needs and demands of our target audience from the early stages of the development. Our aim was to build a tool not only functional but useful for medical students. We also aimed to stand out from the existing tools by integrating necessary and relevant technologies for our audience. Here are measures for this purpose:

- The application was specially designed according to the demands sent to us by medical students.
- We conducted user surveys and feedback sessions to understand real user needs and made improvements accordingly.
- Constant communication with people who would represent the target audience was maintained during the development phase.
- Project design was kept simple for the sake of adaptability of new users
- We conducted several research sessions at different stages of the development to understand similar projects in the market
- We integrated relevant technologies (like LLMs, speech-to-text, spaced repetition algorithms) according to our audience's needs

### 2.2.9. Flexibility

In order to support a wide variety of users and to provide a more personalized experience for each user, our system was designed with flexibility measures. This ensures that users can study in ways that best suit their individual needs and styles.

- Users can create their own flashcard decks and multiple choice question quizzes based on their own sources like lecture notes, which allows them to focus on their own relevant topics
- The spaced repetition algorithm tracks the user's performance on each flashcard for every deck, optimizing learning efficiency based on that performance by ordering the decks and flashcards.
- ReMediCard.io supports various lecture materials as visuals, audio, handwritten notes and digital lecture notes to generate new content on the application ensuring accessibility for different learning preferences and generating
- User's can set personalized study goals and the application will inform the user accordingly via notifications and visual responses

# 3. Final Architecture and Design Details

## 3.1. Overview

ReMediCard.io's architecture consists of multiple subsystems that work together to provide an efficient learning experience for medical students. The system prioritizes modularity so that each subsystem is responsible for a specific function. The client subsystem provides the user interface for students, interacting with the gateway subsystem in the backend, which acts as the entry point for handling requests securely. The system also includes critical components such as user management for authentication and profile tracking, flashcard management for content organization, AI-powered automation for generating flashcards and quizzes, and quiz management for evaluating knowledge retention. Additional features like performance tracking with statistics for study goals, reminder notifications, and a database management system ensure an optimized learning process.

ReMediCard.io consists of a combination of mobile and cloud-based infrastructure to support its operations. The mobile application is compatible with both iOS and Android, and it utilizes local device resources for rendering and tracking study progress, while the backend infrastructure, hosted in the cloud, processes tasks such as speech-to-text conversion, natural language processing, and AI-powered automation. A hybrid data storage strategy is employed, using PostgreSQL for structured data, ElasticSearch for text indexed content for searching, and Amazon S3 for storing large multimedia files. These choices ensure scalable and efficient data management, allowing faultless access to user materials and AI-generated content.

Security and access control are fundamental aspects of the system. Therefore, we need to ensure user data privacy and secure interactions. For this reason, ReMediCard.io employs a robust authentication and authorization mechanism based on JWT (JSON Web Tokens), which allows users to securely access their accounts and interact with the system. We enforce the principle of least privilege through role-based access control, ensuring that only authorized users can perform specific actions. Additionally, authentication tokens are periodically refreshed or invalidated based on user activity to maintain session security. Through these measures, ReMediCard.io ensures a reliable and secure learning environment for medical students while optimizing the study process through automation and data-driven insights.

## 3.2. Subsystem decomposition

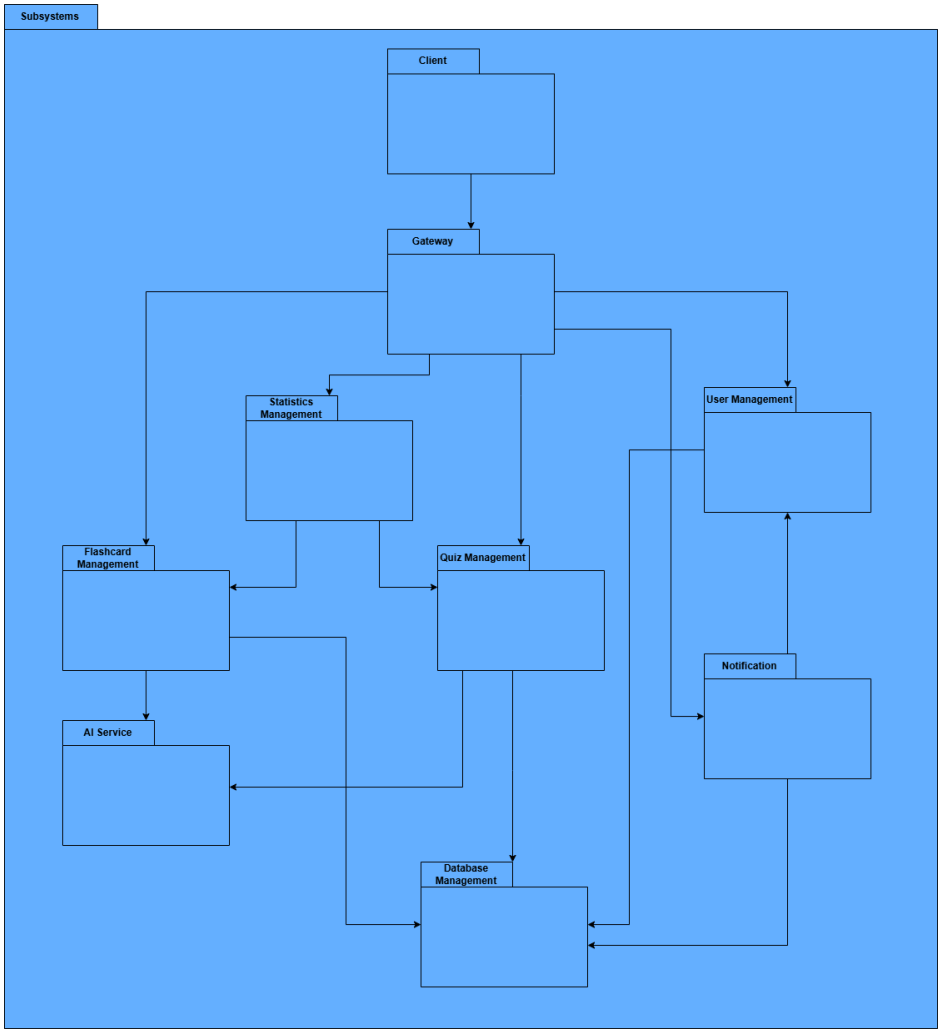


**Fig. 1: Subsystem Decomposition.**

The ReMediCard.io system is structured into several interconnected subsystems, each responsible for specific functionalities.

**Client Subsystem**

- Represents the user interface where students interact with the system.
- Sends user requests to the Gateway Subsystem for processing.

**Gateway Subsystem**

- Acts as the central entry point for all client requests.
- Routes requests to the appropriate subsystems.
- Provides security.

**User Management Subsystem**

- Handles user authentication, authorization, and account management.
- Manages user profiles, study preferences, and progress data.

**Flashcard Management Subsystem**

- Allows users to create, edit, and organize flashcards and decks.
- Supports multimedia attachments (images, audio, and video).
- Integrates with the AI Service Subsystem for automatic flashcard generation.

**AI Service Subsystem**

- Processes the input lecture material internally to use transcripts for generation
- Uses LLMs to generate flashcards from text, images, and speech.
- Enhances learning by providing automatic question generation and summarization.

**Quiz Management Subsystem**

- Allows users to create, edit, and organize quizzes.
- Evaluates quiz responses and provides feedback.
- Interacts with the AI Service Subsystems for automatic quiz generation

**Search Management Subsystem**

- Tracks and indexes decks and quizzes based on content
- Provides results for the search functionalities in various pages such as for discover or users' deck list

**Notification Subsystem**

- Sends reminders to users about scheduled study sessions.
- Sends notifications for upcoming quizzes, progress milestones, and system updates.

**Database Management Subsystem**

- Centralized storage for all kinds of data in the system.
- Divided into relational (PostgreSQL) and document (Elasticsearch) databases. Also, uses Amazon S3 for storing large media files.
- Ensures data integrity, security, and retrieval.

## 3.3. Hardware/software mapping

ReMediCard.io has two main components that involve software/hardware mapping: the mobile application used by medical students and the backend infrastructure on the cloud that powers the automated flashcard generation process.

The mobile application runs on both iOS and Android devices, making use of the device's CPU and GPU for rendering the user interface.

The backend system that is hosted on the cloud, leverages powerful GPUs and CPUs to handle computationally intensive tasks like image processing, speech-to-text conversion, and natural language processing for flashcard automation. The application relies on machine learning models to analyze the material uploaded by the user. These models need to be optimized for performance to allow for a seamless user experience.

## 3.4. Persistent data management

ReMediCard.io requires efficient and reliable persistent data storage to manage user account information, flashcards, study progress, multimedia content, and AI-generated materials. To achieve this, we use a hybrid database architecture combining PostgreSQL for structured data, and Elasticsearch for unstructured text-based content. Additionally, Amazon S3 is used for large media file storage.

PostgreSQL is used to store structured and transactional data such as user profiles, flashcard decks and metadata, user study statistics and progress tracking. Elasticsearch is used to store text-heavy content such as lecture note transcriptions, AI-generated summaries, and user-uploaded text data. Amazon S3 is used for storing large media files such as lecture recordings, and user-uploaded figures and diagrams.

As data persistence and access mechanisms, we use three different frameworks. Spring Data JPA & Hibernate is used for PostgreSQL to facilitate ORM (Object-Relational Mapping) and efficient database transactions. Spring Data Elasticsearch is used for direct interaction with Elasticsearch documents for text indexed data storage. AWS SDK for S3 is used to manage the storage and retrieval of multimedia content securely.

## 3.5. Access control and security

In our system, we have implemented the authorization and authentication ourselves. The user submits their credentials when registering, then we encode this data and store it in the database. When the user wants to access the system, we check the credentials with the ones from the database and we return a JWT (JSON Web Token) if they are matching. This token is used to transmit secure information to the backend. We validate the request based on the token.

After a predetermined time period, the token is refreshed if the user is still active, invalidated otherwise. The user will be logged out once the token is invalidated.
There are also roles in the system. Only the users with predetermined roles can send valid requests to the specified controllers. This “principle of least privilege” idea allows us to implement a more robust application, where only the necessary resources are accessible.

## 3.6. Client Subsystem

The client side of the application is React-Native based mobile application. In the figure below, is the hierarchy of pages showing the navigation paths from the pages starting from the Login and Register.
Users can access only to the Forgot Password, About, Login and Register Pages without authentication. To access the other pages users should either login or register from the corresponding pages.

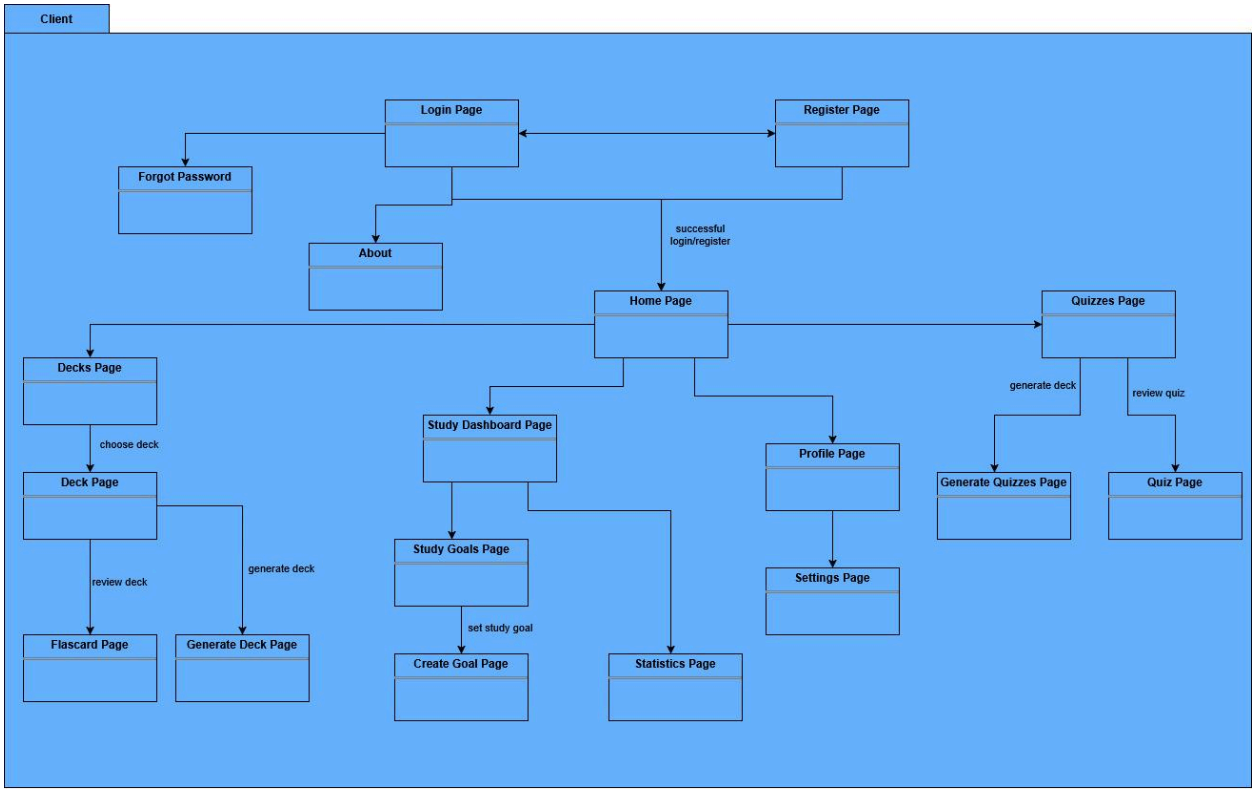


**Figure 2: Client Subsystem Architecture**

## 3.7. Backend Subsystem

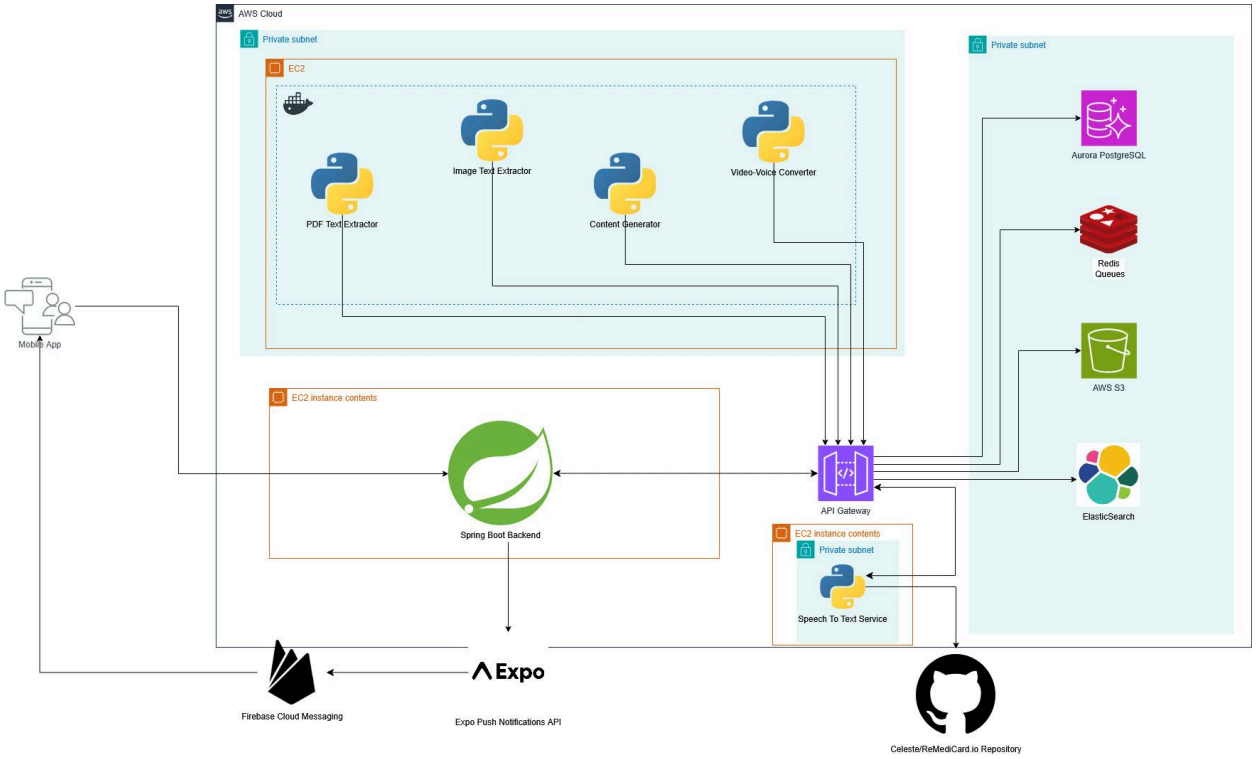


**Figure 3: Backend Architecture**

The detailed backend architecture is given above. Every request first arrives at the gateway and then is navigated to the responsible microservice which interacts with the respective database to persistently store the application state and returns to corresponding response to the client app. Workers in Python containers are responsible to fetch and process the user input data for generative features.

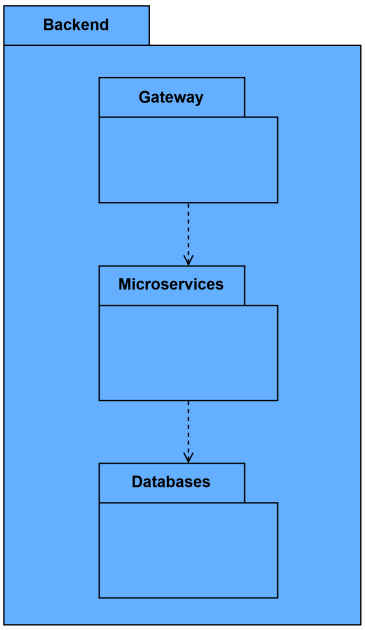


**Figure 4: Backend Subsystem Architecture**

In the backend gateway, all the incoming requests except the ones for forgotten passwords, login and register are first authenticated by the authenticator in the gateway. Then the requests are navigated by the router to the related microservice. Finally, a unified response from the executed microservices are created and returned to the client application by the API Composition component. This scheme is shown below.

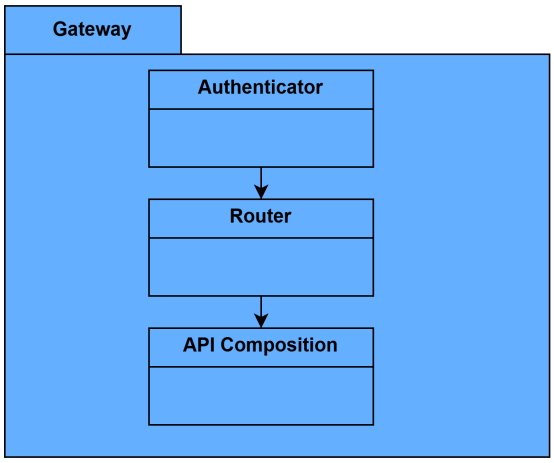


**Figure 5: Backend Gateway Architecture**

The diagram below shows the composition of the User Microservice which is responsible for the user functionalities. Handlers inside the microservice represent a group of functional components(controllers, services and repository interfaces) for the relevant user actions.

Interactions between the handlers are represented with dashed lines implying that the pointing handler is the client for the functionalities provided by the pointed component.

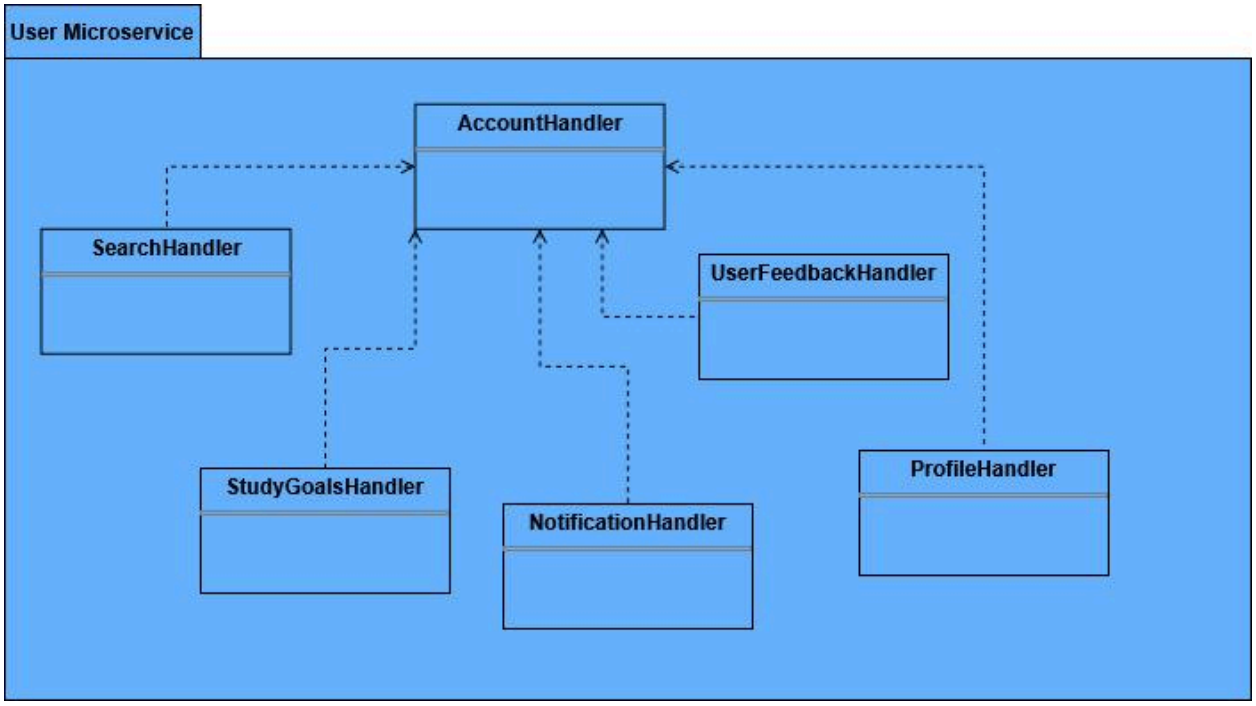


**Figure 6: Backend User Microservice Architecture**

Machine Learning(ML) Microservice is mainly responsible for the preprocessing of the various types of media inputs such as video, voice record, image, etc. and generating the flashcard decks or quizzes from the given traditional lecture content by using queues to asynchronously communicate. For example, OCRHandler manages the transcript creation task for image files while SpeechToTextHandler does the same thing for video or voice records and both enqueue a message to notify the generator services that a transcript to generate a quiz or deck out of it is ready.  On the other hand, QuizGenerationHandler and FlashcardGenerationHandler generate quizzes and flashcard decks respectively from the uploaded lecture content.

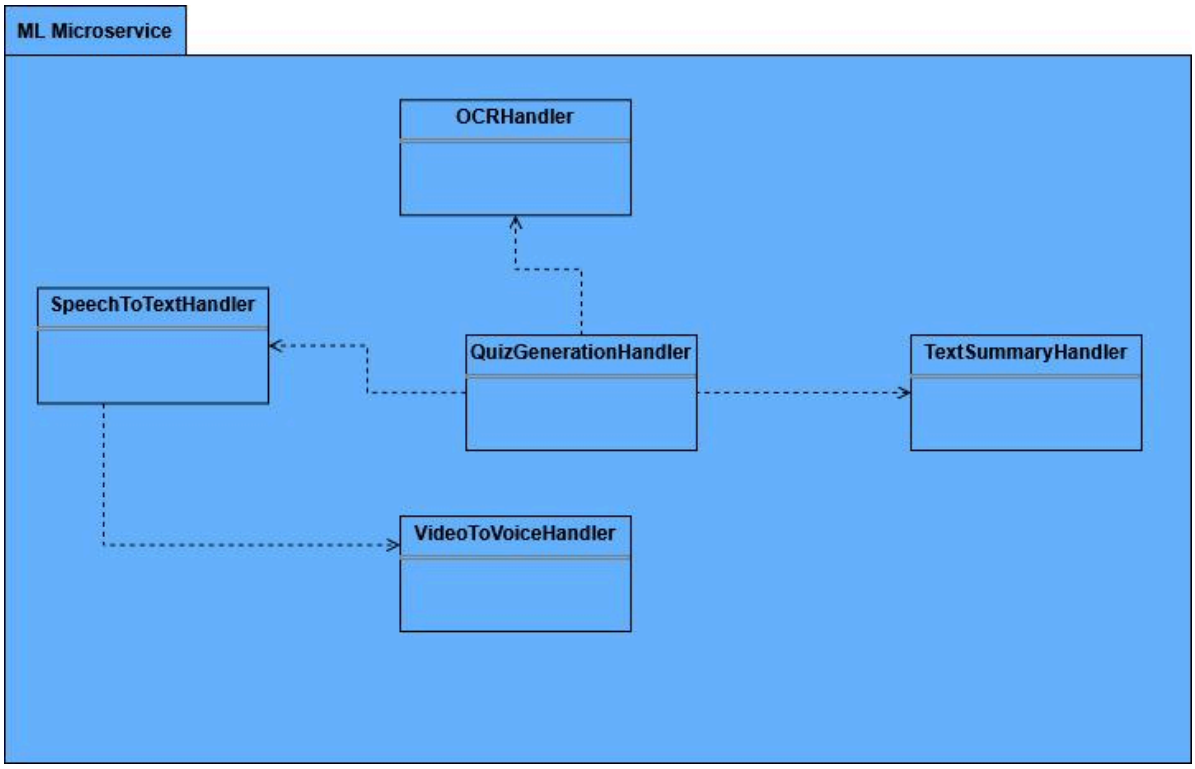


**Figure 7: Backend ML Microservice Architecture**

Flashcard Microservice provides general create, read, update and delete functionalities by making use of handlers such as FlashcardHandler, DeckHandler and QuizHandler along with business logic for the spaced repetition studies with flashcards and quizzes. SpaceRepetitionAlgorithmHandler manages the operations for the spaced repetition algorithm adjusting the displaying schedule/frequencies and scores for the flashcards in decks and questions in the quizzes.

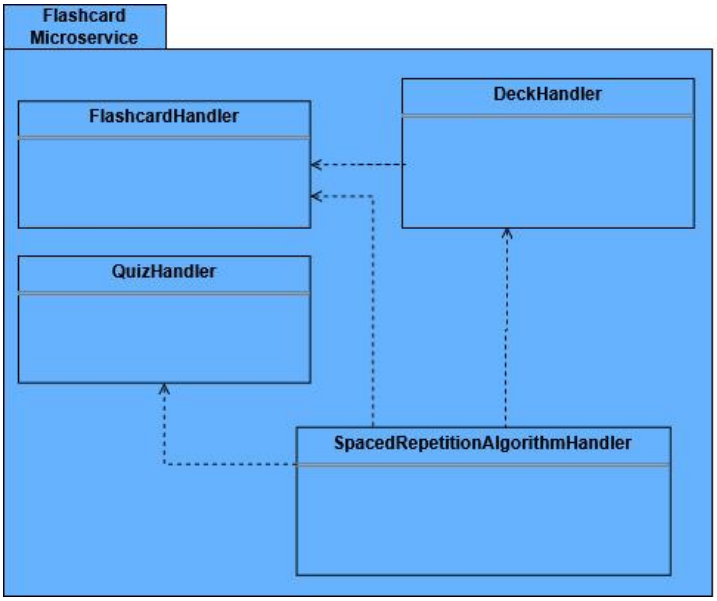


**Figure 8: Backend Flashcard Microservice Architecture**

# 4. Development/Implementation Details

## 4.1. Development as a Team

During the development process, we have adapted sprints ranging from 2 to 4 weeks, flexible for the semester plans. We planned our sprint objectives, assigned relevant tasks to each group member, and then tried to complete as many tasks as possible on our backlogs during the sprints. Although we couldn't fit daily scrum meetings and tried to be as flexible as possible about everyone's completely distinct weekly schedule, we tried our best to keep communicating with each other using our WhatsApp group, Jira Board, and GitHub issues for our repository.

## 4.2. System Design Process

Before planning which technologies to use and ultimately deciding the architecture scheme, our team first examined the underlying architectures in similar flashcard products in the market, such as Anki and Medical FlashNotes, to get a good knowledge of what kind of tools and systems are out there and optimal to create a mobile flashcard application.

For instance, at first, we planned to design and use our spaced repetition algorithm for flashcard deck reviews. Yet, after considering our lack of knowledge and the substantial amount of time and effort a new design would require, we decided to use a third-party library to integrate. Our app has an innovative spaced repetition system for flashcard reviews.

We utilized a fine-tuned Llama 2 model from Hugging Face named MedAlpaca for our machine learning subsystem in our backend. However, because of the lack of computing power, long waiting times, and lack of realistic methods to reduce the compute requirements of the LLM, we opted to use Gemini, which can be used with a free API key, as an alternative to generate quiz or deck content from the transcripts of the traditional lecture materials. Also, in the first development phase, our generator service was part of our backend. Later, we moved the functionalities of our generator service to a completely independent worker running on a Python container in Docker, which made it easier for us to integrate it into our pipeline along with other media processor workers.

## 4.3. Implementing the Architecture and Features

In the backend, we used a microservice architecture to handle various functions: media processing operations, content generation operations, search operations, and user data management operations. Our Spring Boot application acted as a gateway for the user actions, ultimately leading to the operations mentioned earlier. We integrated our system with AWS, utilizing services like AWS API Gateway as an entry point, Aurora PostgreSQL for storing relational user data, EC2 for hosting our worker containers, and S3 to store user media. We also used services like Expo Push Notifications and Google Firebase Cloud Messaging to handle push notifications.

To facilitate local development, we utilized Docker with Docker Compose on all of our microservices. We used Github issues to address our applications' missing elements or bugs.

## 4.4. Design Decision-Making and Iterative Feedbacks

For the features where we had to use our creativity to develop a system, we exchanged our ideas and plans in face-to-face and online meetings in an iterative fashion to produce the most suitable and feasible decisions on our design plans. Multiple developers have examined every implemented feature in various iterations to ensure each one is robust and user-friendly. Regular demos with medical students ensured that every aspect and feature of our application suited their daily study routine. This regular feedback helped us to eliminate the spots where our application lacked user-friendliness. For example, we have changed how our generator service creates the flashcards or questions a few times to ensure they contain compact and relevant content from the input lecture materials and avoid the references for the sources, such as mentioning the texts or videos.

# 5. Test Cases and Results

| Test ID | 1 | Type | Non Functional | Priority | Major |
|---|---|---|---|---|---|
| Objective | This test case is to verify that system response times should not exceed 5 seconds per action. | | | | |
| Steps | Step 1. Simulate multiple concurrent users performing typical actions (login, deck creation, flashcard review).<br>Step 2. Monitor response times and system throughput. | | | | |
| Expected | System response times remain within < 5 seconds per action under expected load. | | | | |
| Date-Result | 30 April 2025 - Passed | | | | |

| Test ID | 2 | Type | Non Functional | Priority | Critical |
|---|---|---|---|---|---|
| Objective | This test case is to verify that Docker containers auto-restart upon failure. | | | | |
| Steps | Step 1. Force a failure in each service container separately.<br>Step 2. Observe if each container restarts automatically. | | | | |
| Expected | The container should restart without manual intervention | | | | |

| | |
|---|---|
| Date-Result | 30 April 2025 - Passed |

| | | | | | |
|---|---|---|---|---|---|
| Test ID | 3 | Type | Non Functional | Priority | Major |
| Objective | This test case is to verify that auto flashcard deck generation after document upload should not exceed 15 minutes. | | | | |
| Steps | Step 1. Login to the application<br>Step 2. Go to the Edit/Create page.<br>Step 3. Click Generate Cards with AI button.<br>Step 4. Upload a source document.<br>Step 5. Measure the time taken after uploading the documents for the generation. | | | | |
| Expected | The time to respond should be less than 15 minutes<br>The system should not freeze during process | | | | |
| Date-Result | 30 April 2025 - Failed: The processing time heavily depends on document size, for smaller files the time constraint is achieved but it is not certain to work on larger files (such as 1-hour long video data) | | | | |

| | | | | | |
|---|---|---|---|---|---|
| Test ID | 4 | Type | Non Functional | Priority | Minor |
| Objective | This test case is to verify that any functionality should be 5 steps away from where the user is at. | | | | |
| Steps | Step 1. Go to any page<br>Step 2. Go depth-first to any linked page from your current page | | | | |
| Expected | Navigation from a page to a functionality should take at most 5 steps | | | | |
| Date-Result | 30 April 2025 - Passed | | | | |

| | | | | | |
|---|---|---|---|---|---|
| Test ID | 5 | Type | Non Functional | Priority | Major |
| Objective | This test case is to verify that the system should be compliant with Data Privacy Regulations (GDPR). | | | | |
| Steps | Step 1. Manually review the user registration, profile update, and account deletion processes to ensure clear consent is obtained and can be withdrawn. | | | | |

| | |
|---|---|
| | Step 2. Verify that data encryption and anonymization measures are in place by checking system notifications and documentation provided.<br>Step 3. Simulate a data deletion request and verify that the system fully purges user data as per GDPR guidelines<br>Step 4. Check that any user consent dialogs are clear and that the data deletion process is fully functional. |
| Expected | The system meets GDPR requirements by obtaining explicit user consent, encrypting data, and providing a complete data deletion mechanism. |
| Date-Result | 30 April 2025 - Passed |

| Test ID | 6 | Type | Non Functional | Priority | Major |
|---|---|---|---|---|---|
| Objective | This test case is to verify that the application data is persistently backed-up should periodically, so that in the event of a failure, the system can recover with minimal downtime and no significant data loss. | | | | |
| Steps | Step 1. In the test environment, simulate a data loss scenario by deleting entries from a test database or corrupting a backup file.<br>Step 2. Follow the documented backup restoration procedure step-by-step.<br>Step 3. Keep track of the restoration process by noting the time it takes for the system to begin and complete recovery. | | | | |
| Expected | Step 1. The backup restoration process should be completed within.<br>Step 2. All critical data is fully restored without loss or corruption.<br>Step 3. The system returns to normal operational status, and all functionalities work as expected. | | | | |
| Date-Result | 30 April 2025 - Passed | | | | |

| Test ID | 7 | Type | Non Functional | Priority | Critical |
|---|---|---|---|---|---|
| Objective | This test case is to verify that the system should handle multiple concurrent user sessions without significant performance degradation. | | | | |
| Steps | Step 1. Simulate or have multiple users log in simultaneously.<br>Step 2. Have each user perform typical actions (create decks, upload files, review flashcards) within a short time span.<br>Step 3. Monitor response times and resource usage. | | | | |

| | |
|---|---|
| Expected | Step 1. Response times remain within acceptable limits (e.g., under 2 seconds).<br>Step 2. No server crashes or significant slowdowns occur.<br>Step 3. System usage metrics (CPU, RAM) stay within defined thresholds. |
| Date-Result | 30 April 2025 - Passed |

| | | | | | |
|---|---|---|---|---|---|
| Test ID | 8 | Type | Non Functional | Priority | Critical |
| Objective | This test case is to verify that the system scales effectively when user demand increases | | | | |
| Steps | Step 1. Simulate a gradual increase in user load (e.g., from 10 to 500 users).<br>Step 2. Measure response times and resource usage.<br>Step 3. Identify any performance bottlenecks. | | | | |
| Expected | The system should maintain stable performance under high loads | | | | |
| Date-Result | 30 April 2025 - Passed | | | | |

| | | | | | |
|---|---|---|---|---|---|
| Test ID | 9 | Type | Non Functional | Priority | Critical |
| Objective | This test case is to verify that the user can’t access some other user’s resources. | | | | |
| Steps | Step 1. The user logs in to the system.<br>Step 2. The user tries to access another user's deck or quiz via its URL. | | | | |
| Expected | The user won’t be permitted to access the deck or quiz and will be redirected to the unauthorized page. | | | | |
| Date-Result | 30 April 2025 - Passed | | | | |

| | | | | | |
|---|---|---|---|---|---|
| Test ID | 10 | Type | Non Functional | Priority | Critical |
| Objective | This test case is to verify that the system should automatically log out inactive users. | | | | |
| Steps | Step 1. Log in to the application with valid credentials. | | | | |

| | |
|---|---|
| | Step 2. Remain idle for a preset session timeout period (e.g., 15 minutes).<br>Step 3. Attempt to perform an action (e.g., open a deck) after the idle period. |
| Expected | Step 1. The user session is terminated automatically.<br>Step 2. The user is redirected to the login screen upon the next action.<br>Step 3. No unauthorized actions are possible after the timeout. |
| Date-Result | 30 April 2025 - Passed |

| Test ID | 11 | Type | Functional | Priority | Critical |
|---|---|---|---|---|---|
| Objective | This test case is to verify that users can register upon entering their username, email and password. | | | | |
| Steps | Step 1. The user enters the registration information.<br>Step 2. The user clicks on the sign up button. | | | | |
| Expected | The user will be redirected to the main page. | | | | |
| Date-Result | 30 April 2025 - Passed | | | | |

| Test ID | 12 | Type | Functional | Priority | Critical |
|---|---|---|---|---|---|
| Objective | This test case is to verify that users can log in upon entering their username and password. | | | | |
| Steps | Step 1. The user enters the login information.<br>Step 2. The user clicks on the login button. | | | | |
| Expected | The user will be redirected to the main page. | | | | |
| Date-Result | 30 April 2025 - Passed | | | | |

| Test ID | 13 | Type | Functional | Priority | Critical |
|---|---|---|---|---|---|
| Objective | This test case is to verify that users can log in via Google. | | | | |
| Steps | Step 1. The user clicks on the Continue with Google button.<br>Step 2. The user fills in their information and logs into their Google account. | | | | |
| Expected | The user will be redirected to the main page. | | | | |

| | |
|---|---|
| Date-Result | 30 April 2025 - Passed |

| | | | | | |
|---|---|---|---|---|---|
| Test ID | 14 | Type | Functional | Priority | Critical |
| Objective | This test case is to verify that users can edit their credentials. | | | | |
| Steps | Step 1. The user clicks on the profile button.<br>Step 2. The user clicks on the edit credentials button.<br>Step 3. The user fills in the information and submits. | | | | |
| Expected | User credentials must have changed. | | | | |
| Date-Result | 30 April 2025 - Passed | | | | |

| | | | | | |
|---|---|---|---|---|---|
| Test ID | 15 | Type | Functional | Priority | Critical |
| Objective | This test case is to verify that users can create decks. | | | | |
| Steps | Step 1. The user navigates to the Decks page from the main menu.<br>Step 2. The user clicks on the Create New Deck button.<br>Step 3. The user fills in the information and submits. | | | | |
| Expected | The created deck must be available among the others. | | | | |
| Date-Result | 30 April 2025 - Passed | | | | |

| | | | | | |
|---|---|---|---|---|---|
| Test ID | 16 | Type | Functional | Priority | Critical |
| Objective | This test case is to verify that users can edit deck information | | | | |
| Steps | Step 1. The user navigates to the Decks page from the main menu.<br>Step 2. The user clicks on the edit button of a deck.<br>Step 3. The user fills in the information and submits. | | | | |
| Expected | 30 April 2025 - Passed | | | | |
| Date-Result | This part will be completed in the final report. | | | | |

| | | | | | |
|---|---|---|---|---|---|
| Test ID | 17 | Type | Functional | Priority | Minor |

| | |
|---|---|
| Objective | This test case is to verify that users can sort the decks. |
| Steps | Step 1. The user navigates to the Decks page from the main menu.<br>Step 2. The user chooses the sorting option from the panel. |
| Expected | The decks must be sorted in the order chosen. |
| Date-Result | 30 April 2025 - Passed |

| | | | | | |
|---|---|---|---|---|---|
| Test ID | 18 | Type | Functional | Priority | Minor |
| Objective | This test case is to verify that users can search for decks. | | | | |
| Steps | Step 1. The user navigates to the Decks page from the main menu.<br>Step 2. The user enters the search word in the search bar and presses enter. | | | | |
| Expected | The decks related to the search word must appear on the screen. | | | | |
| Date-Result | 30 April 2025 - Passed | | | | |

| | | | | | |
|---|---|---|---|---|---|
| Test ID | 19 | Type | Functional | Priority | Minor |
| Objective | This test case is to verify that users can sort the quizzes. | | | | |
| Steps | Step 1. The user navigates to the Quizzes page from the main menu.<br>Step 2. The user chooses the sorting option from the panel. | | | | |
| Expected | The quizzes must be sorted in the order chosen. | | | | |
| Date-Result | 30 April 2025 - Passed | | | | |

| | | | | | |
|---|---|---|---|---|---|
| Test ID | 20 | Type | Functional | Priority | Critical |
| Objective | This test case is to verify that users can search for quizzes. | | | | |
| Steps | Step 1. The user navigates to the Quizzes page from the main menu<br>Step 2. The user enters the search word in the search bar and presses enter | | | | |
| Expected | The quizzes related to the search word must appear on the screen. | | | | |
| Date-Result | 30 April 2025 - Passed | | | | |

| Test ID | 21 | Type | Functional | Priority | Critical |
|---|---|---|---|---|---|
| Objective | This test case is to verify that a user's best score on a quiz is calculated correctly. | | | | |
| Steps | Step 1. The user navigates to the Quizzes page from the main menu<br>Step 2. The user selects a quiz that doesn't have a best score<br>Step 3. The user solves a number of questions and ends the quiz | | | | |
| Expected | The best score statistic must be calculated correctly | | | | |
| Date-Result | 30 April 2025 - Passed | | | | |

| Test ID | 22 | Type | Functional | Priority | Critical |
|---|---|---|---|---|---|
| Objective | This test case is to verify that a user's progress while solving the quiz is accurate. | | | | |
| Steps | Step 1. The user navigates to the Quizzes page from the main menu<br>Step 2. The user selects a quiz<br>Step 3. The user solves a number of questions | | | | |
| Expected | The question progress indicator must show the correct information | | | | |
| Date-Result | 30 April 2025 - Passed | | | | |

| Test ID | 23 | Type | Functional | Priority | Critical |
|---|---|---|---|---|---|
| Objective | This test case is to verify that the users can finish a quiz successfully. | | | | |
| Steps | Step 1. The user navigates to the Quizzes page from the main menu<br>Step 2. The user selects a quiz<br>Step 3. The user solves the questions and ends the quiz | | | | |
| Expected | User should be navigated to the quizzes page | | | | |
| Date-Result | 30 April 2025 - Passed | | | | |

| Test ID | 24 | Type | Functional | Priority | Critical |
|---|---|---|---|---|---|
| Objective | This test case is to verify that a user's last score on a quiz is calculated correctly. | | | | |

| | |
|---|---|
| Steps | Step 1. The user navigates to the Quizzes page from the main menu<br>Step 2. The user selects a quiz<br>Step 3. The user solves the quiz and ends the quiz |
| Expected | The last score on the quiz must represent the latest score calculated |
| Date-Result | 30 April 2025 - Passed |

| Test ID | 25 | Type | Functional | Priority | Critical |
|---|---|---|---|---|---|
| Objective | This test case is to verify that a new flashcard can be created and stored in the database. | | | | |
| Steps | Step 1. The user creates a new flashcard with front and back text.<br>Step 2. The system processes the request and saves it to the database.<br>Step 3. Fetch the flashcard by ID from the database. | | | | |
| Expected | The flashcard is stored with the correct front and back text. | | | | |
| Date-Result | 30 April 2025 - Passed | | | | |

| Test ID | 26 | Type | Functional | Priority | Critical |
|---|---|---|---|---|---|
| Objective | This test case is to verify that users can update flashcard content. | | | | |
| Steps | Step 1. The user updates the front or back of an existing flashcard.<br>Step 2. The system processes and updates the flashcard in the database.<br>Step 3. Fetch the flashcard and check if updates are applied. | | | | |
| Expected | The flashcard contains the updated content. | | | | |
| Date-Result | 30 April 2025 - Passed | | | | |

| Test ID | 27 | Type | Functional | Priority | Critical |
|---|---|---|---|---|---|
| Objective | This test case is to verify that users can delete a flashcard. | | | | |
| Steps | Step 1. The user selects a flashcard and deletes it.<br>Step 2. The system processes the request and removes the flashcard from the database.<br>Step 3. Attempt to fetch the deleted flashcard. | | | | |
| Expected | The flashcard no longer exists in the database. | | | | |

| | |
|---|---|
| Date-Result | 30 April 2025 - Passed |

| | | | | | |
|---|---|---|---|---|---|
| Test ID | 28 | Type | Functional | Priority | Critical |
| Objective | This test case is to verify that users can access public flashcard decks. | | | | |
| Steps | Step 1. The user selects a deck they own.<br>Step 2. The user changes the visibility of the deck to public.<br>Step 3. Other users access the public decks on the discover page. | | | | |
| Expected | The public deck is accessible to other users on the discover page. | | | | |
| Date-Result | 30 April 2025 - Passed | | | | |

| | | | | | |
|---|---|---|---|---|---|
| Test ID | 29 | Type | Functional | Priority | Major |
| Objective | This test case is to verify that users can share quizzes via URL. | | | | |
| Steps | Step 1. The user selects a quiz they own.<br>Step 2. The user clicks on the share quiz option and can share the quiz via URL generated on a platform of their choosing. | | | | |
| Expected | The view-only quiz is accessible to other users once they click on the link. | | | | |
| Date-Result | 30 April 2025 - Passed | | | | |

| | | | | | |
|---|---|---|---|---|---|
| Test ID | 30 | Type | Functional | Priority | Major |
| Objective | This test case is to verify that users can share flashcard decks via URL. | | | | |
| Steps | Step 1. The user selects a deck they own.<br>Step 2. The user clicks on the share deck option and can share the deck via URL generated on a platform of their choosing. | | | | |
| Expected | The view-only deck is accessible to other users once they click on the link. | | | | |
| Date-Result | 30 April 2025 - Passed | | | | |

| | | | | | |
|---|---|---|---|---|---|
| Test ID | 31 | Type | Functional | Priority | Critical |

| | |
|---|---|
| Objective | This test case is to verify that users can add a shared deck among their own decks |
| Steps | Step 1. The user clicks on the URL of a shared deck.<br>Step 2. On the page where they can see the deck in view-only mode, they can opt for adding the deck among their own. |
| Expected | Users should be able to see the shared deck on their decks page. |
| Date-Result | 30 April 2025 - Passed |

| | | | | | |
|---|---|---|---|---|---|
| Test ID | 32 | Type | Functional | Priority | Critical |
| Objective | This test case is to verify that users can add a shared quiz among their own quizzes | | | | |
| Steps | Step 1. The user clicks on the URL of a shared quiz.<br>Step 2. On the page where they can see the quiz in view-only mode, they can opt for adding the quiz among their own. | | | | |
| Expected | Users should be able to see the shared quiz on their quizzes page. | | | | |
| Date-Result | 30 April 2025 - Passed | | | | |

| | | | | | |
|---|---|---|---|---|---|
| Test ID | 33 | Type | Functional | Priority | Major |
| Objective | This test case is to verify that flashcards reappear at the correct intervals based on user recall strength. | | | | |
| Steps | Step 1. The user marks a flashcard as "Incorrect" during review.<br>Step 2. The system updates the flashcard's next review date to an earlier time.<br>Step 3. The user marks another flashcard as "Correct."<br>Step 4. The system delays the next review date for that flashcard. | | | | |
| Expected | The system schedules flashcards appropriately based on difficulty ratings. | | | | |
| Date-Result | 30 April 2025 - Passed | | | | |

| | | | | | |
|---|---|---|---|---|---|
| Test ID | 34 | Type | Functional | Priority | Minor |

| | |
|---|---|
| Objective | This test case is to verify that the algorithm updates review intervals dynamically over time. |
| Steps | Step 1. The user studies a deck with 10 flashcards.<br>Step 2. The user marks three cards as "Correct," four cards as "Uncertain," and three cards as "Incorrect."<br>Step 3. The system updates intervals accordingly.<br>Step 4. The user returns after 24 hours.<br>Step 5. The system presents flashcards based on the updated schedule. |
| Expected | Flashcards appear according to their assigned difficulty levels and time intervals. |
| Date-Result | 30 April 2025 - Passed: Flashcards marked as "Incorrect" appear before the other flashcards. |

| | | | | | |
|---|---|---|---|---|---|
| Test ID | 35 | Type | Functional | Priority | Major |
| Objective | This test case is to verify that newly added flashcards are prioritized in the review cycle. | | | | |
| Steps | Step 1. The user creates a new deck and adds five flashcards.<br>Step 2. The user starts a review session.<br>Step 3. The system presents new flashcards before older ones. | | | | |
| Expected | New flashcards appear more frequently in initial sessions compared to previously learned ones. | | | | |
| Date-Result | 30 April 2025 - Failed (Design Change): By the nature of the spaced repetition algorithm deployed in ReMediCard.io, new objects are not prioritized by default without going through a review cycle | | | | |

| | | | | | |
|---|---|---|---|---|---|
| Test ID | 36 | Type | Functional | Priority | Critical |
| Objective | This test case is to verify that notifications are stored in the database and can be retrieved. | | | | |
| Steps | Step 1. The system triggers a notification event.<br>Step 2. Fetch notifications from the database.<br>Step 3. Verify the notification's content, timestamp, and user association. | | | | |
| Expected | The notification is correctly stored in the database and linked to the correct user. | | | | |
| Date-Result | 30 April 2025 - Passed | | | | |

| Test ID | 37 | Type | Functional | Priority | Major |
|---|---|---|---|---|---|
| Objective | This test case is to verify that email notifications are sent for account-related actions (e.g., password reset). | | | | |
| Steps | Step 1. The user requests a password reset.<br>Step 2. The system triggers an email notification.<br>Step 3. Check the email inbox or email logs. | | | | |
| Expected | The user receives a properly formatted email with a reset link. | | | | |
| Date-Result | 30 April 2025 - Passed | | | | |

| Test ID | 38 | Type | Functional | Priority | Major |
|---|---|---|---|---|---|
| Objective | This test case is to verify that a notification is created when a study goal deadline is approaching. | | | | |
| Steps | Step 1. The user sets a study goal with a deadline.<br>Step 2. Time progresses to the predefined reminder threshold.<br>Step 3. The system triggers a notification event.<br>Step 4. Fetch user notifications from the database. | | | | |
| Expected | A notification showing the correct goal name and deadline is stored and retrieved correctly. | | | | |
| Date-Result | 30 April 2025 - Passed (Design Change): We decided to notify the users on a regular basis according to their preferences instead of notifying specifically before the deadline | | | | |

| Test ID | 39 | Type | Functional | Priority | Critical |
|---|---|---|---|---|---|
| Objective | This test case is to verify the creation of relevant and engaging flashcards by the generative LLM service. | | | | |
| Steps | Step 1. The user uploads the file corresponding to the lecture content.<br>Step 2. LLM generates a deck of flashcards based on the content of the lecture material. | | | | |
| Expected | The generated flashcards are relevant and expressed information is definitely inferable from the given lecture material. | | | | |
| Date-Result | 30 April 2025 - Passed | | | | |

| Test ID | 40 | Type | Functional | Priority | Critical |
|---|---|---|---|---|---|
| Objective | This test case is to verify that LLM generates the flash card decks in Turkish when the given lecture material is in Turkish. | | | | |
| Steps | Step 1. The user opens the auto-generation section and uploads the lecture content.<br>Step 2. The given lecture material is successfully processed by the pipeline and a deck is created. | | | | |
| Expected | All the flashcards in the generated deck are in Turkish. | | | | |
| Date-Result | 30 April 2025 - Passed | | | | |

| Test ID | 41 | Type | Functional | Priority | Critical |
|---|---|---|---|---|---|
| Objective | This test case is to verify that LLM generates flashcard decks in English when the given lecture material is in English. | | | | |
| Steps | Step 1. The user opens the auto-generation section and uploads the lecture content.<br>Step 2. The given lecture material is successfully processed by the pipeline and a deck is created. | | | | |
| Expected | All the flashcards in the generated deck are in English. | | | | |
| Date-Result | 30 April 2025 - Passed | | | | |

| Test ID | 42 | Type | Functional | Priority | Critical |
|---|---|---|---|---|---|
| Objective | This test case is to verify that the flashcard generation section accepts various forms of lecture materials as input. | | | | |
| Steps | Step 1. The user opens the auto-generation section.<br>Step 2. The user chooses a lecture video as the input format.<br>Step 3. The user uploads the corresponding file to the system.<br>Step 4. Application created the generated deck and returns.<br>Step 5. Repeat Step 2 for different content formats: voice record, images of lecture notes and text/pdf files. | | | | |
| Expected | Relevant flashcard decks are generated by the LLM for the given input type. | | | | |
| Date-Result | 30 April 2025 - Passed | | | | |

| Test ID | 43 | Type | Functional | Priority | Critical |
|---|---|---|---|---|---|
| Objective | This test case is to verify that the lecture materials uploaded by a user that include hate speech, sexual content, or discriminatory content are prevented from further processing for deck auto-generation. | | | | |
| Steps | Step 1. The user opens the auto-generation section.<br>Step 2. Users intentionally upload hate speech, sexual content, or discriminatory content in any form.<br>Step 3. The application receives the input and creates the input transcript. | | | | |
| Expected | The generator LLM recognizes the content corresponding to hate speech, sexual content or discriminatory content and avoids generating a flashcard deck. | | | | |
| Date-Result | 30 April 2025 - Passed | | | | |

| Test ID | 44 | Type | Functional | Priority | Critical |
|---|---|---|---|---|---|
| Objective | This test case is to verify that the application ensures the input file given for the deck auto-generation is in an acceptable format and size and then is processed asynchronously. | | | | |
| Steps | Step 1. The user opens the auto-generation section.<br>Step 2. The user chooses the lecture content format and tries to upload the corresponding file. | | | | |
| Expected | The application returns a success message indicating the deck generation is queued or returns an error message expressing the reason for the failure. | | | | |
| Date-Result | 30 April 2025 - Passed | | | | |

| Test ID | 45 | Type | Functional | Priority | Critical |
|---|---|---|---|---|---|
| Objective | This test case is to verify that the users are notified once the auto-generated flashcard decks are created and they can view the decks. | | | | |
| Steps | Step 1. The user opens the auto-generation section.<br>Step 2. The user chooses the lecture content format and uploads the content file.<br>Step 3. The user sees the queuing successful message after the application receives it.<br>Step 4. LLM service creates the corresponding flashcard deck. | | | | |
| Expected | The application notifies the user immediately and the user can see the flashcards and the deck. | | | | |

| | |
|---|---|
| Date-Result | 30 April 2025 - Passed |

| | | | | | |
|---|---|---|---|---|---|
| Test ID | 46 | Type | Functional | Priority | Critical |
| Objective | This test case is to verify that the user document or the file for auto-generation is stored and indexed so that when the user searches for relevant keywords the document is visible. | | | | |
| Steps | Step 1. The user opens the auto-generation section<br>Step 2. The user chooses the lecture content format and uploads the content file.<br>Step 3. The user sees the queuing successful message after the file is received.<br>Step 4. LLM service creates the corresponding flashcard deck.<br>Step 5. The user enters keywords related to the input file content to the search bar on the main page. | | | | |
| Expected | The application returns the input file as the relevant search result. | | | | |
| Date-Result | 30 April 2025 - Failed: We decided not to keep the user documents on the system for the sake of storage space(which can be also abused as a cloud storage otherwise) and make them accessible for the users as it will severely increase the resource consumption. This can be further changed if a change in our infrastructure lead to storage redundancy that can be handy for such purposes | | | | |

| | | | | | |
|---|---|---|---|---|---|
| Test ID | 47 | Type | Functional | Priority | Critical |
| Objective | This test case is to verify that the number of auto-generated flashcards satisfies the lower limit corresponding to the summarized transcript of the uploaded file. | | | | |
| Steps | Step 1. The user opens the auto-generation section<br>Step 2. The user chooses the lecture content format and uploads the content file.<br>Step 3. The user sees the queuing successful message after the file is received.<br>Step 4. LLM service creates the corresponding flashcard deck. | | | | |
| Expected | The application creates at least as many flashcards as the lower limit corresponding to the transcript summary of the input document. | | | | |
| Date-Result | 30 April 2025 - Passed | | | | |

| Test ID | 48 | Type | Functional | Priority | Critical |
|---|---|---|---|---|---|
| Objective | This test case is to verify the creation of relevant and engaging quiz questions by the generative LLM service. | | | | |
| Steps | Step 1. The user uploads the file corresponding to the lecture content.<br>Step 2. LLM generates a quiz based on the content of the lecture material. | | | | |
| Expected | The generated questions are relevant and inexplicitly expressed information in the answers is definitely inferable from the given lecture material. | | | | |
| Date-Result | 30 April 2025 - Passed | | | | |

| Test ID | 49 | Type | Functional | Priority | Critical |
|---|---|---|---|---|---|
| Objective | This test case is to verify the creation of text indexed documents for the input files of the quiz auto-generation feature for obtaining relevant hints while solving the quizzes. | | | | |
| Steps | Step 1. The user uploads the file corresponding to the lecture content.<br>Step 2. LLM generates a quiz based on the content of the lecture material.<br>Step 3. Generated quiz questions are created, the user is notified and the quiz visible to the user.<br>Step 4. The user clicks to the generated quiz in the quizzes section and clicks get a hint button for one of the questions. | | | | |
| Expected | LLM Service returns a relevant hint to the user using the indexed transcript of the corresponding lecture content. | | | | |
| Date-Result | 30 April 2025 - Passed (Design change): Hints are given for the quizzes but documents aren’t currently accessible after the time of generation | | | | |

| Test ID | 50 | Type | Functional | Priority | Critical |
|---|---|---|---|---|---|
| Objective | This test case is to verify the creation of detailed explanations for the generated questions in quiz auto-generation feature by the LLM Service. | | | | |
| Steps | Step 1. The user views a generated quiz by choosing it in the quizzes section.<br>Step 2. The user chooses an option for one of the test questions.<br>Step 3. The user clicks to see the answer and the explanation button. | | | | |
| Expected | The application displays a neat explanation for the solution of the test question. | | | | |
| Date-Result | 30 April 2025 - Passed | | | | |

| Test ID | 51 | Type | Functional | Priority | Critical |
|---|---|---|---|---|---|
| Objective | This test case is to verify that the users are notified once the auto-generated quizzes are created and they can view the quizzes. | | | | |
| Steps | Step 1. The user opens the quiz auto-generation section.<br>Step 2. The user chooses the lecture content format and uploads the content file.<br>Step 3. The user sees the queuing successful message after the file is received.<br>Step 4. LLM service creates the corresponding quiz. | | | | |
| Expected | The application notifies the user immediately and the user can see the questions and the quiz. | | | | |
| Date-Result | 30 April 2025 - Passed | | | | |

| Test ID | 52 | Type | Functional | Priority | Critical |
|---|---|---|---|---|---|
| Objective | This test case is to verify that the quiz generation section accepts various forms of lecture materials as input. | | | | |
| Steps | Step 1. The user opens the quiz auto-generation section.<br>Step 2. The user chooses lecture video as the input format.<br>Step 3. The user uploads the corresponding file to the system.<br>Step 4. The application created the generated quiz and returns.<br>Step 5. Repeat Step 2 for different content formats: voice record, images of lecture notes and text/pdf files. | | | | |
| Expected | Relevant questions are generated by the LLM for the given input type. | | | | |
| Date-Result | 30 April 2025 - Passed | | | | |

| Test ID | 53 | Type | Functional | Priority | Minor |
|---|---|---|---|---|---|
| Objective | This test case is to verify that users can contact the developers within the app for suggestions and problems | | | | |
| Steps | Step 1. The user navigates to the profile page<br>Step 2. The user clicks on contact us option<br>Step 3. The user fills in the subject and message fields | | | | |
| Expected | Mail generated with the contents filled by the user is sent to the developers | | | | |
| Date-Result | 30 April 2025 - Passed | | | | |

| Test ID | 54 | Type | Functional | Priority | Major |
|---|---|---|---|---|---|
| Objective | This test case is to verify that users can access public quizzes. | | | | |
| Steps | Step 1. The user selects a quiz they own.<br>Step 2. The user changes the visibility of the quiz to public.<br>Step 3. Other users access the public quizzes on the discover page. | | | | |
| Expected | The public quiz is accessible to other users on the discover page. | | | | |
| Date-Result | 30 April 2025 - Passed | | | | |

# 6. Maintenance Plan and Details

So far, the project has reached the majority of the objectives in the project plan. We will pursue a new plan that will prioritize the refinements and bug fixes on our current system. Later on refactoring and stabilization on our code repository will be our primary focus. Then, our new objectives will cover the features we couldn't completely integrate along with the new ones and changes in our infrastructure that will improve the study experience within ReMediCard.io.

In our system, the generation pipeline supports only PDF, JPEG/PNG, MP3 and MP4 files; it can be further extended to accept other file formats for the supported media types. Also for converting the speech data into text and for using the medically fine-tuned generational LLM model, our pipeline is currently restricted by the hardware resources. If we can move our pipeline to better compute alternatives through cloud, we can use more robust models that can improve the authenticity of the content generator along with the quality or uniqueness of the questions in quizzes or flashcards in decks.

So far we designed and implemented a simple and user friendly UX but we are looking forward to gathering user feedback for possible improvements and creating even smoother UX. Possible improvement areas are enhancing the visual design according to user habits and demands, maintaining and improving smooth user interactions, workflow simplification by reducing unnecessary or unused steps and adding tooltip information. Also according to the demand new system languages can be added to the application. These refinements are expected to increase user satisfaction.

In time, as the application gains more users and its use expands, we will consider adding more features that will assist our target audience. We plan to make changes that will broaden the applications educational impact. Even though these new features will be formed according to the user feedback and their usage data, here are some ideas that we could implement in the future:

- Case study feature to simulate doctor-patient interaction
- Leaderboards for the flashcard decks and quizzes, gamifying the learning process.
- Flashcard creation from figures

These features and more are discussed in section 8.2 in more detail.

Currently, our infrastructure doesn't have a vast storage space and GPU powered compute instances that can speed up the scale of computations especially for the speech to text and generation services in our pipeline. Also the system can be scaled up to multiple regions to provide services in distinct locations more effectively. Caching and asynchronous data prefetching strategies can also be implemented within services like Aurora or API Gateway in AWS to improve the performance further, providing a smoother user experience.

# 7. Other Project Elements

## 7.1. Consideration of Various Factors in Engineering Design

### 7.1.1. Constraints

#### 7.1.1.1. Implementation Constraints

Technological Requirements:

- The integration of machine learning models and computer vision tools requires robust frameworks such as TensorFlow, PyTorch, and OpenCV, along with substantial backend development for data processing.
- The AI model must handle various data types, including image, audio, and text, and provide similar output for each.

Computational Resources:

- High computational power is necessary for processing images, running ML algorithms, and transcribing audio to text. This may require the use of adequate cloud-based services.

Data Privacy:

- To protect user data and securely manage sensitive information, such as lecture videos and personal notes, it is essential to implement robust encryption and comply with established data protection protocols.

Platform Compatibility:

- The application needs to work across multiple mobile platforms (e.g., iOS, Android), leading to a need for cross-platform development tools and technologies.

- GPU spot-instance volatility is crucial because our inference jobs run on AWS g4dn instances obtained through EC2 Spot, sudden terminations can occur.

#### 7.1.1.2. Economic Constraints

Development Costs:

- Leveraging libraries or APIs for advanced functionality, such as speech-to-text conversion or specialized ML tools, could lead to licensing fees.
- There could be lease fees involved for the use of medical books, presentations, sample questions, and any other resources.

Server and Infrastructure Costs:

- Running AI models and storing multimedia data (images, audio, and video) may involve significant server costs, especially for real-time processing and storage.

Maintenance and Updates:

- Continuous updates, bug fixes, and model retraining to keep up with the latest advances in AI and maintain accuracy will require further funding.

Subscription Models:

- The base app will be free to use, limiting the auto-generation features. The premium version of the application will offer affordable pricing to obtain unlimited use of the features offered.

#### 7.1.1.3. Ethical Constraints

Bias and Fairness:

- Machine learning models might introduce biases if not properly trained on a diverse dataset.
- Generated flashcards and questions should be checked to ensure fairness and inclusivity.

User Consent:

- Users should be fully informed about data collection and use. User data can only be collected after the user consents to it.
- Consent for processing audio and image data is crucial, and transparency regarding data storage and privacy should be maintained.

Intellectual Property:

- Care must be taken to avoid infringing on proprietary medical content and copyrighted materials when generating content.
- Data uploaded by the users should be checked by administrative personnel to prevent any abuse.

Accuracy and Reliability:

- Providing students with accurate information is crucial; incorrect or misleading content could have serious implications for their education and future medical practice.
- Professional help from the medical field is required to check the content used in the application.

Mental Health Considerations:

- A balanced approach should be maintained to prevent stress and cognitive overload in students by not overwhelming them with difficult questions or excessive repetitions.

#### 7.1.1.4. Global Factors

The application should be fit for a global audience with varying medical curricula and study practices used. Multi-language support is essential for students across the globe. This can be done with the help of contributors all around the world or with the use of natural language processing tools.

#### 7.1.1.5. Cultural Factors

Cultural differences in medical practices and terminologies should be respected. Generated content could be reviewed by an Admin before it is deemed ready for use.

#### 7.1.1.6. Social Factors

Features like sharing decks and quizzes could help build a collaborative community, enabling peer-to-peer learning that contributes to each and every student. The application should be accessible by students from varying socioeconomic backgrounds.

#### 7.1.1.7. Environmental Factors

Cloud-based infrastructure with on-demand resource usage should reduce energy consumption compared to traditional methods. By digitizing study materials and studying sessions, we could contribute to the reduction of paper use.

**Table 1: Factors that can affect analysis and design.**

| Factors | Effect Level | Effect |
|---|---|---|
| Public health | 7/10 | Enhances medical education by improving study efficiency, indirectly contributing to better-trained healthcare professionals. |

| | | |
|---|---|---|
| Public safety | 2/10 | Ensures accurate content delivery, reducing risks associated with misinformation in medical training, which could impact patient safety in later stages. |
| Public welfare | 7/10 | Offers equal access to educational tools, empowering students from diverse backgrounds and improving the quality of future medical practitioners globally. |
| Global factors | 4/10 | Adapts to diverse curricula and languages, promoting inclusivity and incentivizing global collaboration among medical students. |
| Social factors | 8/10 | Builds a collaborative community through sharing decks, quizzes and peer learning. |
| Environmental factors | 5/10 | Encourages sustainability by reducing paper usage and leveraging energy-efficient on-demand cloud resources for AI processing and storage. |
| Economic factors | 8/10 | Balances cost-efficiency with affordability, providing a free base app and a premium version for advanced features. |

### 7.1.2. Standards

ReMediCard.io's standards cover user interaction, data handling, and software development:

#### 7.1.2.1. Software Engineering Standards

- **IEEE 830-1998**: Used for documenting requirements specifications [7].
- **IEEE 1016-2009**: Used for system design description [8].
- **UML 2.5.1**: Used for system modeling, including use-case diagrams, class diagrams, activity diagrams, and state diagrams [9].

#### 7.1.2.2. Data Security and Privacy Standards

- **ISO/IEC 27001**: Will be used for information security management to protect sensitive data and prevent unauthorized access [10].
- **GDPR Compliance**: Will be used to ensure the system is designed to meet European data protection standards for handling user data [11].

#### 7.1.2.3. AI/ML Development Standards

- **IEEE 7001-2021**: Will be used for ethical considerations in AI design and development [12].

#### 7.1.2.4. Coding Standards

- **Google Style Guide (Java)**: Will be used for Java development [13].
- **RESTful API Design Standards**: Will be used for integration and communication between system components.

#### 7.1.2.5. Testing Standards

- **ISO/IEC/IEEE 29119**: Will be used for software testing to ensure reliability for testing of application features [14].

## 7.2. Ethics and Professional Responsibilities

As ReMediCard.io is planned to be an everyday application for medical students, it contains professional and ethical issues. Such issues can be grouped in the below subheadings:

### 7.2.1. Data Privacy and Security

The application will utilize private data such as lecture notes, voice recordings, and video recordings. Ensuring the safety and confidentiality of the user data is a must. In our project, we will encrypt all kinds of data during storage and transmission.
We will use data protection regulations such as the General Data Protection Regulation (GDPR) [4] to ensure user data privacy.

### 7.2.2. Potential Problems of AI Usage

The use of large language models (LLMs) such as ChatGPT or T5 may introduce bias in generated content. We have taken steps to minimize the harmful biases in flashcards, questions, and feedback.
The system includes disclaimers about the limitations of generated content to prevent reliance on potentially incomplete or incorrect information.

### 7.2.3. Accessibility

The application aims to reach a large number of users. Hence, the system supports both Turkish and English to ensure people from different backgrounds can utilize the application.

### 7.2.4. Accuracy and Trustworthiness

As the accuracy of information in flashcards and quizzes is crucial, the application incorporates quality assurance measures, such as validating content against trusted medical sources.

### 7.2.5. Copyright

We will ensure that copyright laws are compiled by reviewing any third-party content, such as medical images, external datasets or lecture materials carefully.

## 7.3. Teamwork Details

### 7.3.1. Contributing and functioning effectively on the team to establish goals, plan tasks, and meet objectives

To ensure effective collaboration within Team Celeste, we employ a combination of industry-standard tools, project management techniques, and formal meetings. With these tools and techniques, an efficient workflow is achieved so that the tasks are appropriately distributed, progress is tracked, and a high degree of communication is ensured.

We use Jira for task and issue tracking, Confluence for team documentation, such as project configuration and information on various backend services, and Google Drive for storing and sharing important files, such as course report drafts or our internal project plans. These tools ensure that all project information is easily accessible and well-documented so that any member who wants to learn more about any part of the project encounters no problem finding or receiving relevant material.

Git and GitHub drive our development workflow, hosting our repositories and website, following changes, and working on code together. GitHub Issues helps us handle bug reports and feature requests efficiently. We can report encountered problems or expected work to the related members and quickly solve these issues.

Constant communication is needed for our success as a team. We have a WhatsApp group for quick discussion, and Zoom is our default platform for scheduled virtual meetings for planning, reviewing, and even handling urgent issues. In this way, anyone who wants to consult other members regarding any problem can reach out to the others as quickly as possible.

As a group of five Computer Science students, we do not have the exact schedules; we have developed a customized Scrum framework that suitably works well around our available time without compromising self-organization and efficiency. This ensures individual plans are not inadvertently disrupted by the tasks involved in the project.  We have daily work in 2-week sprint iterations so that we can plan, do, and inspect work sequentially.

We define our objectives at the start of each sprint and break them down into tasks. We also conduct weekly planning and review meetings to evaluate progress, identify obstacles, and

make necessary changes. While doing these meetings, each member's participation is essential; thus, we arrange the meetings with care for everyone's suitability.

We hold retrospective and planning meetings after each sprint. During these meetings, we discuss what went right, what went wrong, and how we can improve in the next iteration. In this way, every team member is made aware of the updates in different aspects of the project, keeping everyone on the same page regardless of their ongoing responsibilities. Additionally, we have progress meetings with course instructors where we briefly present our progress and plan and discuss with them what can be or should be improved, leveraging their experience and knowledge.

With a nice combination of well-organized and customized meetings, effective channels and tools of communication, and a neatly and consistently documented development process, our team facilitates easy collaboration despite our completely distinct schedules. Applying project management software, Agile principles, and sufficient documentation allows us to work efficiently and flexibly in the implementation phases.

### 7.3.2. Helping creating a collaborative and inclusive environment

Providing a collaborative and healthy working environment is very important for the smooth progress of this project. To create this environment, we respect the contribution of each member. In addition, it is very useful for us that everyone can give their own ideas and feedback and we can discuss them. For these discussions we use our communication channels which are reserved for the project and this keeps the conversations clean.

One of the biggest factors that ensures the progress of the project is that we arrange the work distribution appropriately and help each other in case of a problem. We all are mindful of our schedule and work pace, we arrange the deadlines and meetings in a way that is acceptable for everyone. If a team member encounters difficulties, others step in to provide support. This cooperative approach not only helps us stay on track but also enhances our overall teamwork

To facilitate teamwork, we arrange ourselves to work together in some subtasks. Working in this way ensures that any possible problems are resolved as quickly as possible. Additionally, collaborating on certain tasks allows us to learn from each other's approaches and share different problem-solving techniques.

### 7.3.3. Taking lead role and sharing leadership on the team

For a successful team, successful leadership is mandatory since this sets the work pace and manages the collaborations among peers. In this project, we have a shared leadership model. This model makes each team member responsible for specific areas. Distributing the leadership role to each member of the team, we make sure that every aspect of our solution is gaining notice by at least one individual and considered carefully. This cooperative approach entails

more shared encouragement and communication-based collaborations within the team so that every peer can improve as a professional and team member.

When making important decisions about the project, we meet as a group over zoom. Members share their ideas and how they want to approach the problem or a topic at hand. Then we do research about it and the final decision is made following a consensus-based approach. If we want to be responsible for a part in the project and take the lead role, we share it with others and focus more on it then others if it is agreed upon by the group members.

The list below specifies the regions or aspects where each team member acted as a solo or shared leader. Our commitment to the project as a group and each individual's contribution is highlighted in this way:

- **Ömer Fırat Bekiroğlu:** Frontend Design and Development, ML Model/Speech-to-text Service, Input Media Handling/Processing, User Experience Design.
- **Cenk Merih Olcay:** Backend Design and Development, ML Model/LLM based Summarizer and Generator Services, Software Architecture Design.
- **Faruk Uçgun:** Backend Design and Development, Frontend development, Frontend - Backend connection, Software Architecture Design.
- **Enes Bektaş:** Backend Design and Development, Frontend - Backend connection, Software Architecture Design. Survey analysis.
- **Akif Erdem Tanyeri:** Frontend Design and Development.

### 7.3.4. Meeting objectives

In this section, it is shown how much we have met our project plan that we set in our Analysis phase with the details of the completeness level.

**Table 2: Work Packages**

| Work Package | Work Package Title | Leader | Members Involved |
|---|---|---|---|
| WP#1 | Jira and GitHub Project Management Infrastructure | Faruk Uçgun | Akif Erdem Tanyeri, Ömer Fırat Bekiroğlu |
| WP#2 | Full Stack Project Setup | Cenk Merih Olcay | Enes Bektaş, Faruk Uçgun |
| WP#3 | User Authentication, Profiles and Account Development | Cenk Merih Olcay | Faruk Uçgun |
| WP#4 | General UI Design | Ömer Fırat Bekiroğlu | Akif Erdem Tanyeri |

| | | | |
|---|---|---|---|
| WP#5 | Platform Main Page, Account and Profile UI Development | Ömer Fırat Bekiroğlu | Akif Erdem Tanyeri |
| WP#6 | Card Deck Creation and Editing Development | Enes Bektaş | Faruk Uçgun |
| WP#7 | Deck and Flashcard Creation and Management Pages UI Development | Ömer Fırat Bekiroğlu | Akif Erdem Tanyeri |
| WP#8 | Deck Sharing and Flashcard Media Attachment Feature Development | Cenk Merih Olcay | Enes Bektaş |
| WP#9 | Summarizer and Flashcard Generator LLM Models Research | Enes Bektaş | Cenk Merih Olcay, Ömer Fırat Bekiroğlu |
| WP#10 | Spaced Repetition and Study Tracking Development | Faruk Uçgun | Enes Bektaş, Cenk Merih Olcay |
| WP#11 | AI-Generated Text Summaries and Flashcards Model Development | Enes Bektaş | Ömer Fırat Bekiroğlu |
| WP#12 | Flashcard Generation from Lecture Notes Feature Development | Cenk Merih Olcay | Faruk Uçgun |
| WP#13 | Study Dashboard, Goal Notifications, Statistics UI Development | Ömer Fırat Bekiroğlu | Akif Erdem Tanyeri, Cenk Merih Olcay |
| WP#14 | Video/Voice Record Processing Development | Enes Bektaş | Cenk Merih Olcay |
| WP#15 | AI Model Integration with Backend Deck/Flashcard Services | Faruk Uçgun | Enes Bektaş |
| WP#16 | Quizzes and Questions Features UI Development | Akif Erdem Tanyeri | Ömer Fırat Bekiroğlu |
| WP#17 | Quizzes and Questions Features Development | Faruk Uçgun | Enes Bektaş |

| | | | |
|---|---|---|---|
| WP#18 | Dockerization and Microservice Architecture | Faruk Uçgun | Cenk Merih Olcay |
| WP#19 | Case Study AI Model Development | Enes Bektaş | Akif Erdem Tanyeri |
| WP#20 | Case Study Feature Development | Enes Bektaş | Faruk Uçgun |
| WP#21 | Case Study UI Development | Akif Erdem Tanyeri | Ömer Fırat Bekiroğlu |
| WP#22 | Testing and Refinement Phase | Faruk Uçgun | Enes Bektaş, Ömer Fırat Bekiroğlu, Cenk Merih Olcay |

| **WP #1:** Jira and GitHub Project Management Infrastructure | |
|---|---|
| **Start Date:** 10/10/2024 **End Date:** 20/10/2024 | |
| **Leader:** Faruk Uçgun | **Members Involved:** Akif Erdem Tanyeri, Ömer Fırat Bekiroğlu |
| **Objectives:**<br>To initialize project management infrastructure using Jira and Github to ensure project tracking and collaboration | |
| **Tasks:**<br>**T1.1:** Setup Jira environment<br>**T1.2:** Setup Github organization and repository infrastructure<br>**T1.3:** Setup project website hosting with GitHub Pages | |
| **Deliverables:**<br>**D1.1:** Jira Board<br>**D1.2:** Github repository<br>**D1.3:** Project website | |
| **100% COMPLETED** | |

| **WP #2:** Full Stack Project Setup | |
|---|---|
| **Start Date:** 18/10/2024 **End Date:** 30/10/2024 | |
| **Leader:** Cenk Merih Olcay | **Members Involved:** Enes Bektaş, Faruk Uçgun |
| **Objectives:**<br>Creation of the default Front-end & Back-end projects and Database setups for development | |
| **Tasks:**<br>**T2.1:** Create a React Native project with Expo and Android Studio<br>**T2.2:** Create a Spring Boot project<br>**T2.3:** Create PostgreSQL and Elasticsearch containers using Docker | |
| **Deliverables:**<br>**D2.1:** React Native Project Source Code<br>**D2.2:** Spring-Boot Project Source Code<br>**D2.3:** Dockerfile for database containers | |
| **100% COMPLETED** | |

| **WP #3:** User Authentication, Profiles and Account Development | |
|---|---|
| **Start Date:** 18/10/2024 **End Date:** 30/10/2024 | |
| **Leader:** Cenk Merih Olcay | **Members Involved:** Faruk Uçgun |
| **Objectives:**<br>Implement the features for user accounts, authentication/authorization, and account | |

| |
|---|
| settings |
| **Tasks:**<br>**T3.1:** Define the classes and their relations to the accounts in the system<br>**T3.2:** Implement a secure user registration and login functionality<br>**T3.3:** Implement functionality for users to view, edit, and update their profiles<br>**T3.4:** Design and implement role-based permissions for different user roles |
| **Deliverables:**<br>**D3.1:** Class Diagram<br>**D3.2:** Backend Authentication System<br>**D3.3:** User profile management functionalities<br>**D3.4:** Role-Based Authorization |
| **100% COMPLETED** |

| **WP #4:** General UI Design | |
|---|---|
| **Start Date:** 18/11/2024 **End Date:** 12/12/2024 | |
| **Leader:** Ömer Fırat Bekiroğlu | **Members Involved:** Akif Erdem Tanyeri |
| **Objectives:**<br>Design a visually appealing, simple, and useful user interface that enhances user experience | |
| **Tasks:**<br>**T4.1:** Draw simple design to decide on UI design approaches<br>**T4.2:** Create mockups for core features | |
| **Deliverables:**<br>**D4.1:** UI Mockups | |
| **100% COMPLETED** | |

| **WP #5:** Platform Main Page, Account, and Profile UI Development | |
|---|---|
| **Start Date:** 18/10/2024 **End Date:** 30/10/2024 | |
| **Leader:** Ömer Fırat Bekiroğlu | **Members Involved:** Akif Erdem Tanyeri |
| **Objectives:**<br>Implement the design of the home page, account, and profile-related pages following designed mockups | |
| **Tasks:**<br>**T5.1:** Implement the home page for the application<br>**T5.2:** Implement account-related UI elements for login, registration, and account settings<br>**T5.3:** Implement profile-related UI elements for editing profile and user preferences<br>**T5.4:** Integrate UI with Back-end Endpoints | |

| |
|---|
| **Deliverables:**<br>**D5.1:** Home Page UI<br>**D5.2:** Login Page UI<br>**D5.3:** Registration Page UI<br>**D5.4:** Profile Page UI<br>**D5.5:** Profile Settings UI |
| **100% COMPLETED** |

| | |
|---|---|
| **WP #6:** Card Deck Creation and Editing Development | |
| **Start Date:** 05/12/2024 **End Date:** 16/12/2024 | |
| **Leader:** Enes Bektaş | **Members Involved:** Faruk Uçgun |
| **Objectives:**<br>Design and development of the deck and flashcard management features in the backend | |
| **Tasks:**<br>**T6.1:** Update the Class Diagram to involve the relevant Classes and Relations<br>**T6.2:** Implement the features in the backend using Spring Data Library to create Entities, Repositories, and relevant service methods | |
| **Deliverables:**<br>**D6.1:** Updated Class Diagram<br>**D6.2:** Back-end Source Code for New Services, Entities, and Repositories | |
| **100% COMPLETED** | |

| | |
|---|---|
| **WP #7:** Deck and Flashcard Creation and Management Pages UI Development | |
| **Start Date:** 12/12/2024 **End Date:** 12/01/2025 | |
| **Leader:** Ömer Fırat Bekiroğlu | **Members Involved:** Akif Erdem Tanyeri |
| **Objectives:**<br>Implement the design of flashcards and flashcard decks with their interactions and management pages | |
| **Tasks:**<br>**T7.1:** Implement the design of a single flashcard and its edit and create pages<br>**T7.2:** Implement the design of a flashcard deck and its edit and create pages<br>**T7.3:** Integrate UI with Back-end Endpoints | |
| **Deliverables:**<br>**D7.1:** Flashcard UI Source Code<br>**D7.2:** Edit Flashcard UI Source Code<br>**D7.3:** Create Flashcard UI Source Code<br>**D7.4:** Flashcard Deck UI Source Code<br>**D7.5:** Edit Flashcard Deck UI Source Code<br>**D7.6:** Create Flashcard Deck UI Source Code | |
| **100% COMPLETED** | |

| **WP #8:** Deck Sharing and Flashcard Media Attachment Feature Development | |
|---|---|
| **Start Date:** 26/12/2024 **End Date:** 16/01/2025 | |
| **Leader:** Cenk Merih Olcay | **Members Involved:** Enes Bektaş |
| **Objectives:**<br>Development of the shared-deck features between the users in the backend | |
| **Tasks:**<br>**T8.1:** Update class diagram to cover shared decks and new related classes and relations<br>**T8.2:** Implement the features by updating the related backend services | |
| **Deliverables:**<br>**D8.1:** Updated Class Diagram<br>**D8.2:** Updated Source Code for the Back-end Services Responsible for Decks and Flashcards | |
| **100% COMPLETED** | |

| **WP #9:** Summarizer and Flashcard Generator LLM Models Research | |
|---|---|
| **Start Date:** 26/12/2024 **End Date:** 15/01/2025 | |
| **Leader:** Enes Bektaş | **Members Involved:** Cenk Merih Olcay, Ömer Fırat Bekiroğlu |
| **Objectives:**<br>Finding suitable methods and data sets to fine-tune fundamental models for the summarization and text generation capabilities in medicine | |
| **Tasks:**<br>**T9.1:** Research on the performance of fundamental models in the medical text<br>**T9.2:** Finding relevant datasets to optimize the fundamental models<br>**T9.3:** Searching for ways to enhance the optimization with different means of hyperparameter tuning | |
| **Deliverables:**<br>**D9.1:** Medical Text Datasets<br>**D9.2:** Report on the performance of state of art fundamental models<br>**D9.3:** Report on the candidate methods to optimize the fine-tuning | |
| **100% COMPLETED** | |

| **WP #10:** Spaced Repetition and Study Tracking Development | |
|---|---|
| **Start Date:** 26/12/2024 **End Date:** 15/01/2025 | |
| **Leader:** Faruk Uçgun | **Members Involved:** Enes Bektaş, Cenk Merih Olcay |

| |
|---|
| **Objectives:**<br>Find a suitable spaced repetition algorithm and implement the study tracking by developing an existing spaced repetition algorithm |
| **Tasks:**<br>**T10.1:** Conduct research on spaced repetition algorithms<br>**T10.2:** Implement spaced repetition algorithm<br>**T10.3:** Develop study tracking functionality using a spaced repetition algorithm |
| **Deliverables:**<br>**D10.1:** Spaced repetition algorithm<br>**D10.2:** Study tracking functionality |
| **100% COMPLETED** |

| **WP #11:** AI-Generated Text Summaries and Flashcards Model Development | |
|---|---|
| **Start Date:** 20/01/2025 **End Date:** 30/01/2025 | |
| **Leader:** Enes Bektaş | **Members Involved:** Ömer Fırat Bekiroğlu |
| **Objectives:**<br>Fine-tune the fundamental models using the obtained reports, dataset, and methods from WP#9 | |
| **Tasks:**<br>**T11.1:** Choosing the target fundamental models from the reports<br>**T11.2:** Preparing the datasets for the training<br>**T11.3:** Training the target multiple models in multiple iterations with different methods<br>**T11.4:** Creating the services for inference | |
| **Deliverables:**<br>**D11.1:** Fine-tuned AI Models<br>**D11.2:** Hosted AI Model Services for Inference | |
| **100% COMPLETED** | |

| **WP #12:** Flashcard Generation from Lecture Notes Feature Development | |
|---|---|
| **Start Date:** 30/01/2025 **End Date:** 09/02/2025 | |
| **Leader:** Cenk Merih Olcay | **Members Involved:** Faruk Uçgun |
| **Objectives:**<br>Implement the backend services and image processing service to extract the labels out of figure images and create the flashcards/puzzles through them | |
| **Tasks:**<br>**T12.1:** Creating an OCR service for text extraction<br>**T12.2:** Creating backend services for figure flashcards | |

| |
|---|
| **Deliverables:**<br>**D12.1:** OCR Service Source Code<br>**D12.2:** Source Code of Back-end Services for figure-based flashcard management |
| **50% COMPLETED** |
| **A model is developed to scan and modify the images but due to high cost of acceleration and maintenance for such a service, we deferred this feature for later versions.** |

| | |
|---|---|
| **WP #13:** Study Dashboard, Goal Notifications, Statistics UI Development | |
| **Start Date:** 30/01/2025 **End Date:** 09/02/2025 | |
| **Leader:** Ömer Fırat Bekiroğlu | **Members Involved:** Akif Erdem Tanyeri, Cenk Merih Olcay |
| **Objectives:**<br>Implement the design of a study tracking system, including study goals and statistics as well as their interactions with users, such as notifications | |
| **Tasks:**<br>**T13.1:** Implement study dashboard UI for users to set study goals<br>**T13.2:** Implement goal notifications for interactions of study goals and users<br>**T13.3:** Implement a statistics page for users to show what they achieve<br>**T13.4:** Integrate UI with Back-end Endpoints | |
| **Deliverables:**<br>**D13.1:** Study Dashboard UI<br>**D13.2:** Goal Notifications<br>**D13.3:** Statistics Page UI | |
| **100% COMPLETED** | |

| | |
|---|---|
| **WP #14:** Video/Voice Record Processing Development | |
| **Start Date:** 09/02/2025 **End Date:** 19/02/2025 | |
| **Leader:** Enes Bektaş | **Members Involved:** Cenk Merih Olcay |
| **Objectives:**<br>Implement the features for extracting the transcripts from videos and voice records and generating flashcards | |
| **Tasks:**<br>**T14.1:** Create a Speech to Text<br>**T14.2:** Create a Back-end service to manage voice records and text-indexed transcripts | |
| **Deliverables:**<br>**D14.1:** Speech-to-Text Service Source Code<br>**D14.2:** Source Codes for the Back-end Services | |
| **100% COMPLETED** | |

| **WP #15:** AI Model Integration with Backend Deck/Flashcard Services | |
|---|---|
| **Start Date:** 19/02/2025 **End Date:** 05/03/2025 | |
| **Leader:** Faruk Uçgun | **Members Involved:** Enes Bektaş |
| **Objectives:**<br>Integrating the implemented Back-end services for flashcard generation features with AI Models | |
| **Tasks:**<br>**T15.1:** Integrating the Lecture Notes Services with AI Models<br>**T15.2:** Integrating the Figures Services with AI Models<br>**T15.3:** Integrating the Video/Voice Recorded Services with AI Models | |
| **Deliverables:**<br>**D15.1:** Integrated Back-end Services Updated Source Codes | |
| **100% COMPLETED** | |

| **WP #16:** Quizzes and Questions Features UI Development | |
|---|---|
| **Start Date:** 19/02/2025 **End Date:** 20/03/2025 | |
| **Leader:** Akif Erdem Tanyeri | **Members Involved:** Ömer Fırat Bekiroğlu |
| **Objectives:**<br>Implement quiz questions and quizzes as well as quiz related UI pages such as creating and editing quiz questions and quizzes. Additionally interactions with quiz questions and quizzes going to be implemented by following the mockups | |
| **Tasks:**<br>**T16.1:** Prepare visual appearance of quiz questions and quizzes<br>**T16.2:** Implement editing and creating quiz questions and quizzes<br>**T16.3:** Implement user interactions with questions and quizzes<br>**T16.4:** Integrate UI with Back-end Endpoints | |
| **Deliverables:**<br>**D16.1:** Quiz Question UI<br>**D16.2:** Quiz UI<br>**D16.3:** Generating, Creating, and Editing Quiz Question Pages UI<br>**D16.4:** Generating, Creating and Editing Quiz Pages UI<br>**D16.5:** Question Interactions<br>**D16.6:** Quiz Interactions | |
| **100% COMPLETED** | |

| **WP #17:** Quizzes and Questions Features Development | |
|---|---|
| **Start Date:** 05/03/2025 **End Date:** 15/03/2025 | |
| **Leader:** Faruk Uçgun | **Members Involved:** Enes Bektaş |

| |
|---|
| **Objectives:**<br>Implement the Features for saving Test questions and Question Generation |
| **Tasks:**<br>**T17.1:** Implement Back-end Services for Questions and Quizzes<br>**T17.2:** Integrate the Back-end Service with generational AI Models |
| **Deliverables:**<br>**D17.1:** Source Codes for the created Back-end Services |
| **100% COMPLETED** |

| **WP #18:** Dockerization and Microservice Architecture Implementation | |
|---|---|
| **Start Date:** 30/11/2024 **End Date:** 20/01/2025 | |
| **Leader:** Faruk Uçgun | **Members Involved:** Cenk Merih Olcay |
| **Objectives:**<br>Configure the execution environments of the back-end, database services, and input processing services(image, videos, voice records) for deployment with Microservice Architecture | |
| **Tasks:**<br>**T18.1:** Dockerize Back-end, Database, and Input Processing Services<br>**T18.2:** Integrate All the Services with each other | |
| **Deliverables:**<br>**D18.1:** Dockerfiles and Docker Compose File | |
| **100% COMPLETED** | |

| **WP #19:** Case Study AI Model Development | |
|---|---|
| **Start Date:** 09/02/2025 **End Date:** 01/03/2025 | |
| **Leader:** Enes Bektaş | **Members Involved:** Akif Erdem Tanyeri |
| **Objectives:**<br>Finding suitable machine learning approaches and develop an AI model capable of generating case studies using LLMs | |
| **Tasks:**<br>**T19.1:** Conduct research on machine learning techniques that are suitable<br>**T19.2:** Gather a dataset that is sufficient to generate new case studies<br>**T19.3:** Try LLMs for case study generation<br>**T19.4:** Develop a sufficient AI model using LLMs | |
| **Deliverables:**<br>**D19.1:** Case Study Generating AI Model | |
| **75% COMPLETED** | |
| **Tasks T19.1, T19.2 and T19.3 are completed, yet, due to lack of compute power and** | |

**priorities of other tasks, we moved forward without this feature.**

| **WP #20:** Case Study Feature Development | |
|---|---|
| **Start Date:** 05/03/2025 **End Date:** 15/03/2025 | |
| **Leader:** Enes Bektaş | **Members Involved:** Faruk Uçgun |
| **Objectives:**<br>Develop case study features using the model that will be built at WP#19 | |
| **Tasks:**<br>**T20.1:** Case study creation functionalities development<br>**T20.2:** Case study editing functionalities development<br>**T20.3:** Development of user actions on case studies<br>**T20.4:** Progress and performance tracking development | |
| **Deliverables:**<br>**D20.1:** Fully Functional Back-end Case Study Service Source Code | |
| **0% COMPLETED** | |
| **Due to the work package having low priority within our time constrainted schedule and the prerequisite work package WP#19 being incomplete, this work package is also deferred.** | |

| **WP #21:** Case Study UI Development | |
|---|---|
| **Start Date:** 05/03/2024 **End Date:** 20/03/2025 | |
| **Leader:** Akif Erdem Tanyeri | **Members Involved:** Ömer Fırat Bekiroğlu |
| **Objectives:**<br>Implement a case study design that enables users to interact with those scenarios | |
| **Tasks:**<br>**T21.1:** Prepare case study main page<br>**T21.2:** Prepare interactive scenario UI<br>**T21.3:** Integrate UI with Back-end Endpoints | |
| **Deliverables:**<br>**D21.1:** Case Study Main Page UI<br>**D21.2:** Interactive Scenario UI<br>**D21.3:** Dynamic Response and Statistics Components for Case Study | |
| **0% COMPLETED** | |
| **Due to the work package having low priority within our time constrainted schedule and the prerequisite work package WP#20 being incomplete, this work package is also deferred.** | |

| **WP #22:** Testing and Refinement Phase | |
|---|---|
| **Start Date:** 01/04/2024 **End Date:** 30/04/2025 | |
| **Leader:** Faruk Uçgun | **Members Involved:** Enes Bektaş, Ömer Fırat Bekiroğlu, Cenk Merih Olcay |
| **Objectives:**<br>Test the system considering all the cases in the design report and refine the features. | |
| **Tasks:**<br>**T22.1:** Test the system on the integration tests declared on the reports.<br>**T22.2:** Finish the final refinement of the source code. | |
| **Deliverables:**<br>**D22.1:** Final Report Test Results<br>**D22.2:** Project Final Source Code | |
| **100% COMPLETED** | |

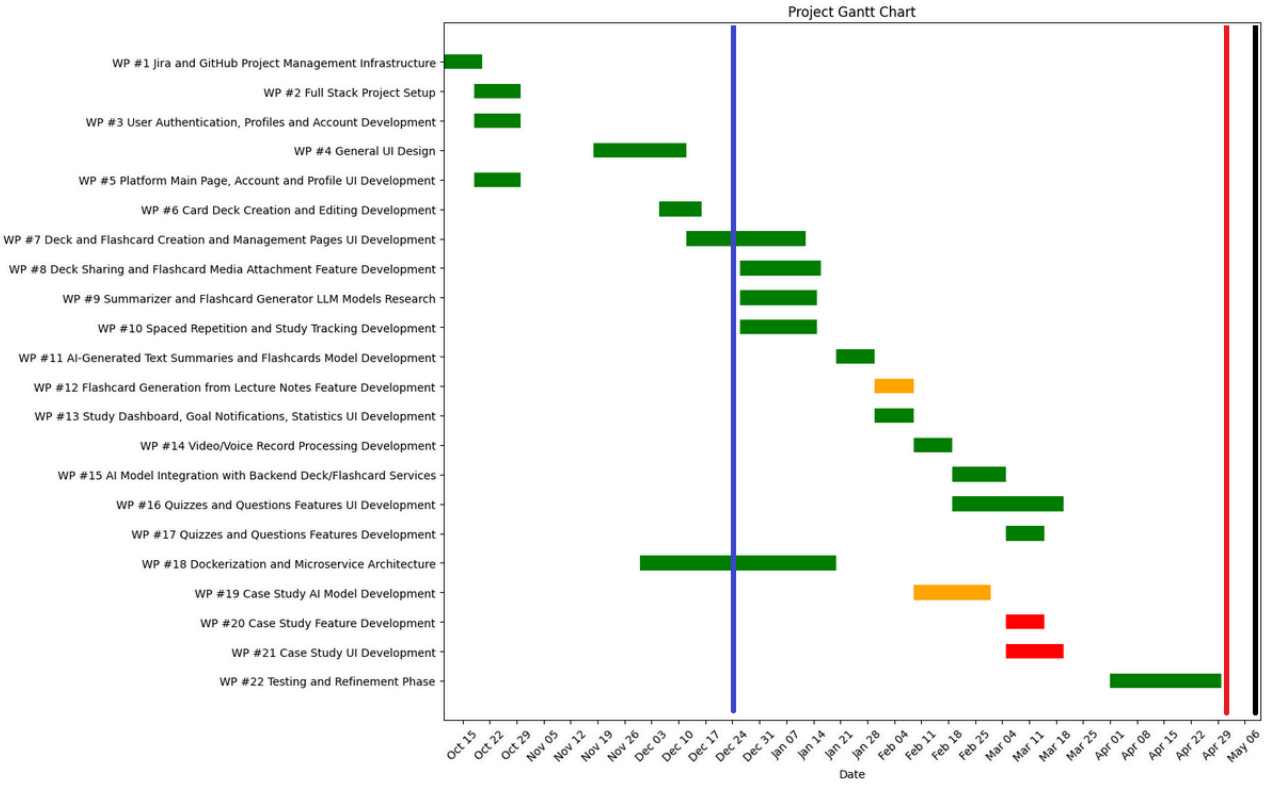


**Figure 9: Project Plan**

**Vertical - Blue Line:** CS491 Final Version
**Vertical - Red Line:** Current Progress
**Vertical - Black Line:** CS492 Final Version
**Green Blocks:** Completed Work Packages
**Orange Blocks:** Partially Completed Work Packages
**Red Blocks:** Incomplete/Aborted Work Packages

## 7.4. New Knowledge Acquired and Applied

Throughout the development of ReMediCard.io, our team gained significant new knowledge across multiple domains—ranging from cloud computing and machine learning to project management and team collaboration.

**1. Cloud Infrastructure & DevOps**
We gained hands-on experience with AWS services, including API Gateway, EC2, S3, and Aurora PostgreSQL. These tools taught us how to build scalable and cost-effective systems. Moreover, using Docker for local development and containerized microservices deepened our understanding of DevOps principles.

**2. Microservice Architecture**
Designing a backend using a microservice architecture allowed us to understand how to decompose a large system into independent services that are loosely connected to each other. This approach required us to learn inter-service communication, API composition, asynchronous communication and service orchestration.

**3. Machine Learning Integration**
We learned how to incorporate pre-trained LLMs and models like Llama 2 and Gemini into real-world applications. This involved understanding their limitations, API usage, and performance tuning. Handling OCR and speech-to-text pipelines also taught us how to preprocess various forms of input data for ML needed for the auto-generation feature.

**4. Full-Stack Development with Spring Boot & React Native**
On the backend, we became proficient in Spring Boot, especially with Spring Data JPA, Spring Security, and RESTful API design. On the frontend, working with React Native helped us understand mobile UI/UX concerns like making sure the app looks the same on all kinds of phones. Implementing secure JWT-based authentication and role-based access control was a particularly valuable experience.

**5. Agile Project Management & Scrum**
Customizing an Agile Scrum framework around our academic schedules helped us appreciate the importance of adaptable planning. We improved in sprint planning, writing clear Jira tasks, and delivering results on time.

**6. Effective Collaboration & Leadership**
Using a shared leadership model encouraged us to take initiative and ownership in different areas of the project. This experience helped us develop soft skills like conflict resolution, taking-giving constructive feedback, and taking responsibility for our parts of the project.

**7. User-Centered Design**
Engaging with medical students for feedback taught us how to design with users in mind. Iterating based on real-world feedback helped us refine the UI as we moved along and improve

the relevance of generated content, making the application genuinely useful to the target audience.

# 8. Conclusion and Future Work

## 8.1. Conclusion

In summary, ReMediCard.io provides medical students with an alternative way to accelerate and enhance their studies. Using modern machine learning technologies and intelligent algorithms in the background facilitates a personalized and efficient learning experience for its users. The innovative spaced repetition algorithm and the automated deck and quiz generation features help students easily review the lecture content.

In addition to personalization features, it includes discovering and content-sharing features to enforce students' efforts to create cumulative study materials or archives. Along with study tracking and study goal features, the application encourages students to make collective efforts for the exams by including competition elements and information exchange.

Last but not least, the application engages with its users through periodic and smartly timed notifications, motivating them to consistently work on their personalized studies with friendly reminders and insightful study stats.

## 8.2. Future Work

As the usage of ReMediCard.io expands to many more medical faculties, several features can be added to our system or several aspects that can be improved to handle higher workloads from new users and assist their studies:

- Automatically creating flashcard for figures using the label
- Creating a chat based case study feature to simulate patient interaction
- Specialized automatic custom classification for public decks and quizzes so that a pile of decks or quizzes are archived together based on lecture domain
- Acceleration and scaling on the generation pipeline for faster responses
- Transforming the mobile app to have compatibility with more devices
- Potential partnership with cloud providers for more robust system infrastructure
- Potential partnership with medical study brands for branding
- More detailed profile features for users to introduce a competitive dimension through leaderboards and duels with friends for deck and quiz reviews
- Enhanced statistics and visualization for students to provide insightful feedback about their studies
- To further enhance the user experience and personalization in our application achievements and badges can be added for deck and quiz reviews.

## 8.3. Potential Business Strategies

We are currently looking at two primary strategies for monetizing the usage in our app. The app will have a free tier; users can make quizzes and decks for several rounds. This will enable users to experience the platform's basic functionality without an initial expense so they can get started with the features.

For the ability to access additional generation rounds beyond this free limit, we are looking at the following two monetization strategies:

In-App Purchase Model: Consumers in this model can purchase packages of limited numbers of additional auto-generating credits. They can then utilize these to create either decks or quizzes, with consumers having some choice in how they would like to utilize them. This approach provides a pay-as-you-go model for users who prefer one-off buy or infrequent use needs.

Subscription Model: Our subscription model offers users unlimited generation capabilities for decks and quizzes within a certain subscription period. This model suits well for users who always require the regular utilization of the app's generation feature. Subscription levels may be based on duration (monthly, quarterly, annually) or bundled with other premium features in the future.

We plan to evaluate these options based on user interest, usage patterns, and long-term revenue sustainability. The goal is to balance accessibility for casual users with value and flexibility for sophisticated users.

# 9. References


[1] P. Dattathreya and S. Shillingford, “Identifying the ineffective study strategies of first year medical school students,” *Medical Science Educator*, vol. 27, no. 2, pp. 295–307, Mar. 2017, doi: 10.1007/s40670-017-0396-2.

[2] M. A. Aljaffer *et al.*, “The impact of study habits and personal factors on the academic achievement performances of medical students,” *BMC Medical Education*, vol. 24, no. 1, Aug. 2024, doi: 10.1186/s12909-024-05889-y.

[3] K. Baba, N. Cheimanoff, and N.-E. E. Faddouli, “A comparative study of active and passive learning approaches in hybrid learning, undergraduate, educational programs,” in *Advances in intelligent systems and computing*, 2020, pp. 715–725. doi: 10.1007/978-3-030-52249-0_48.

[4] “General Data Protection Regulation”. https://gdpr-info.eu. [Accessed: Nov 18, 2024].

[5] “IEEE Recommended Practice for Software Requirements Specifications”. https://standards.ieee.org/ieee/830/1222/. [Accessed: Nov 18, 2024].

[6] “IEEE Standard for Information Technology--Systems Design--Software Design Descriptions”. https://standards.ieee.org/ieee/1016/4502/. [Accessed: Nov 18, 2024].

[7] “About the Unified Modeling Language Specifications Version 2.5.1”. https://www.omg.org/spec/UML/2.5.1/About-UML. [Accessed: Nov 18, 2024].

[8] “ISO/IEC 27001:2022 – Information Security Management”. https://www.itgovernance.co.uk/iso27001. [Accessed: Nov 18, 2024].

[9] “IEEE Standard for Transparency of Autonomous Systems”. https://standards.ieee.org/ieee/7001/6929/. [Accessed: Nov 18, 2024].

[10] “Google Java Style Guide”. https://google.github.io/styleguide/javaguide.html. [Accessed: Nov 18, 2024].

[11] “ISO/IEC/IEEE International Standard - Software and systems engineering --Software testing”. https://standards.ieee.org/ieee/29119-1/10779/. [Accessed: Nov 18, 2024].