Bilkent University - Computer Science

CS492 - Senior Design Project II

Detailed Design Report

T2401

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

2025 Spring

1. Introduction

With the vast amount of lecture content to be covered regularly, it is challenging to be a medical student. Research by Dattatreya and Shillingford suggests that medical studies are not adjusted to allow medical students to process all this information easily [1]. Despite the various factors contributing to this mismatch, a recent study shows that students prefer lecture slides over textbooks [2]. As a result, medical students learn their lecture content passively, which leads to ineffectiveness in their studies and performance loss on their exams [3]. ReMediCard.io's primary goal is to handle these problems and enforce and encourage medical students' studies.

There are several flashcard applications. However, using these applications and generating flashcards still takes time and effort, which are the main difficulties in studying medical topics. Current applications like Quizlet and Anki offer customization options and allow for the creation of flashcards, but the time consumption regarding these interactions can be frustrating for medical students. Additionally, both Quizlet and Anki rely on users for this flashcard generation, and they cannot generate flashcards automatically from students' study resources. Which is a significant setback for these applications.

ReMediCard.io is an AI-powered flashcard application intended to help medical students study more efficiently, considering 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. This way, students can concentrate more on studying and less on preparation.

Moreover, ReMediCard.io utilizes adaptive learning algorithms to personalize study sessions based on users' statistics. Along with enhancing learning efficiency, this personalized approach also meets the diverse needs of medical students. These cutting-edge technologies are the innovative part of ReMediCard.io, which differentiates our application from similar products.

The remainder of this report outlines the design of ReMediCard.io in detail. Section 2 presents the market's current systems and their architectures to give technical insights about similar software solutions. Section 3 includes a detailed description of the proposed software design of our system. Section 4 explains the different subsystems involved in our solutions and their insights. Section 5 represents the test cases that the final product will be assessed against. Section 6 discusses the various aspects of our solution, ranging from technical to non-technical dimensions. Finally, Section 7 presents the work allocation for the project and the organization of our team.

1.1 Purpose of the system

This project aims to improve the learning experience of medical students. This improvement will be achieved by integrating computer vision and machine learning practices with medical students' study resources. Traditional study methods mainly rely on textbooks and lecture notes. However, studying and using these resources may be time-consuming. Our project aims to offer an interactive platform that generates flashcards and quiz questions created from summarized information from study resources such as study notes, textbooks, and audio and video records.

Our application ensures that users focus on the concepts they find more challenging by using spaced repetition algorithms. Additionally, the system can extract key information and generate flashcards or quizzes from lecture videos or notes. This automation allows students to concentrate on studying rather than using a manual note-taking approach.

Moreover, the application intelligently considers the user's preferences, study schedule, and past performances while solving decks, providing a more personalized learning experience. The system recognizes the spots in students' studies to provide intelligent feedback and recommendations for the following sessions. This flexible setting aims to improve their performance on school exams significantly.

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 Design goals

1.2.1 Usability

User Friendly UI:

- A clean, simple, and visually appealing design will be applied
- The application will have easily understandable content where users can manage learning materials and create flashcards.
- The application will be designed in a way that users can easily learn and use different features of the application.

Easy Registration and Login:

- Users will be able to log in to the system more practically by registering, logging in via email, and integrating with their Google accounts

Responsive Design:

- The system should be designed to adapt to different screen sizes and devices.

Feedback:

- Users will be able to learn whether their actions were successful or not with visual or text-based feedback when using the application.
- Explanatory warnings will be presented to the user when incorrect or incomplete actions are taken.

1.2.2 Reliability

Uninterrupted Service:

- The system will be accessible to users with high levels of availability.
- In case of possible interruptions, the application will be put back into service as soon as possible.

Data Consistency and Management:

- User data and system logs should always be stored accurately and up to date.
- Systematic backups will be made to prevent data loss.
- In case of data loss, the system will be able to be quickly restored from the latest backup.

Version Control and Updates:

- Updates will be performed without harming existing user data and the ongoing system.
- A rollback procedure will be prepared, and in case of possible version update errors, this procedure can be followed to revert to the last working version.

Monitoring and Testing:

- The system will undergo extensive testing before deployment.
- New updates will undergo necessary tests before being released to the user.
- Once the system is deployed, it will be continuously monitored to identify performance and reliability issues.

1.2.3 Performance

User Interaction:

- The system should respond instantly to user actions, providing smooth interaction.
- The system should handle simultaneous active users without a reduction in performance.

Hosting:

- The project will be hosted in an environment strong enough to meet the required user demands.
- The hosting environment will be scalable to adapt to different levels of traffic and provide uninterrupted service during peak usage times.

Optimization:

- The application will be optimized to utilize system resources (CPU, memory, network) efficiently, ensuring no waste of resources.
- The system will be designed to meet user demands without compromising performance.

1.2.4 Supportability

Documentation:

- The reports and documentation we prepare will contain the necessary information about how the system is built and works.
- The structure and working procedure of new updates will be added to the documentation.

User Support:

- Detailed user manuals and FAQs will be provided for ease of use.
- Users will be able to report problems they encounter, and they can seek assistance through support channels such as email.

Compatibility:

- The mobile application will be compatible with commonly used mobile operating systems.
- The system will be tested regularly to ensure compatibility with newer versions of browsers and operating systems.

1.2.5 Extensibility

Architecture:

- The system will be built in a way that allows for the easy addition of new features or components without breaking the existing ones.
- New functions can be integrated into the core system with minimal changes, providing flexibility for future updates.
- The data will be scalable as the user needs and traffic increases.
- Additional data can be added without requiring significant changes to the database.

Third-party Integrations:

- The system will support easy integration with third-party tools and services, such as cloud storage platforms and Google accounts

1.2.6 Scalability

Dynamic Resource Allocation:

- The system will allocate computational resources dynamically based on user activity.
- Resources will be adjusted elastically to prevent latency and downtime.

Cost Efficiency:

- The system will optimize computational resource usage to ensure cost-effectiveness.
- Unused resources will be deallocated to reduce overhead costs.

1.2.7 Security & Privacy

Authentication & Authorization:

- The system will implement role-based access control (RBAC) to ensure users only access relevant features.
- OAuth2 and similar modern authentication mechanisms will be supported for secure login and third-party integrations.
- Multi-factor authentication (MFA) will be considered for additional security.

Data Protection & Encryption:

- User data will be encrypted in transit and at rest to ensure confidentiality.

Secure Transactions & Atomicity:

- All database operations will be performed atomically to prevent partial updates and inconsistencies.

1.2.8 Maintainability

Modular & Service-Oriented Design:

- The system will follow a modular architecture where each feature or service can be updated independently.
- New features or fixes can be introduced independently without affecting the rest of the system.

Version Control:

- Git will be used for version control to track source code changes.

1.2.9 Availability

High Uptime & Fault Tolerance:

- The system will be built with a fault-tolerant infrastructure to minimize downtime.
- Load balancing mechanisms will distribute traffic efficiently across multiple servers.

Dynamic Scalability:

- Cloud-based solutions will be utilized to scale resources on demand

1.2.10 Localization

Multi-Language Support:

- The system will support Turkish and English as primary languages.
- Additional languages can be added easily with minimal effort.

Regional Adaptations:

- Different date formats units will be supported to accommodate international users.

1.2.11 Marketability

Target Audience

- Designed for medical students to enhance and ease the learning process.
- RemediCard.io proceeds in line with what is obtained from the target audience through surveys and discussions.
- Easy-to-use interface encourages users to adapt quickly.

Competitive Advantage:

- RemediCard.io combines AI-powered learning tools, flashcard automation, and spaced repetition.

1.2.12 Flexibility

Personalization:

- Users can obtain their own decks on the topics they choose and from the sources they prefer
- Spaced repetition algorithm provides repetition intervals that are individually optimal for each user

Support For Various Data Format:

- Supports visual, auditory, and text-based learning methods.
- Allows integration of videos, images, and notes into flashcards.

1.2.13 Modularity

Component-Based System:

- Each feature is developed as an independent module.
- Modules can be updated or replaced without affecting the core system.

Easy to Develop:

- New features can be easily added and run compatible with other modules

1.3 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.

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.

1.4 Overview

The primary objective of ReMediCard.io is to facilitate the high memorization-based learning process for medical students by utilizing AI-based technologies to transform traditional lecture materials, such as lecture notes and recordings, into interactive and engaging study tools. The application's core feature is the intelligent flashcard auto-generator, which takes lecture materials as inputs, captures the key terms and concepts, and creates easy-to-use flashcard decks. This system employs Large Language Model(LLM) based Machine Learning(ML) services to extract compact but significant information from the traditional lecture materials.

ReMediCard.io offers a clean and user-friendly UI. Users can adapt quickly to UI because of its simplicity, which enhances the overall experience. Its scalable architecture allows the system to respond to high user load and provide services without problems. Also, having a robust design ensures that possible struggles are overcome with minimal downtime. While delivering this robust and helpful structure, security measures have been taken into account to secure user data and ensure compliance with the relevant regulations.

Converting the pure lecture materials into compact forms of information in flashcards, the application encourages medical students to focus more on the critical aspects of their study, optimizing their efficiency. An innovative spaced repetition algorithm that adapts to the student's learning rates adjusts the schedules for flashcard review intervals through reminders to optimize the information retention in students' memories. This algorithm matches the brain's information absorption schedule with medical students' studies, allowing them to remember crucial lecture content more quickly and efficiently.

Additionally, we have an extensive pipeline for media processing, such as a speech-to-text feature for audio data and an Optical Character Recognition(OCR) feature for handwritten lecture notes and figures. Utilizing these technologies, ReMediCard.io offers a practical study experience.

2. Current software architecture

Remedicard.io is designed to overcome the limitations of traditional flashcard applications such as Anki and Quizlet by AI-generated content specifically for medical education. A good understanding of the current architectures is a sounder basis for Remedicard.io to build upon. In particular, the open source Anki GitHub repository [4] reveals a layered design that separates the user interface, core logic, and data management. In this section, we will examine these architectural components to provide context for Remedicard.io's design improvements.

2.1 Layered Architecture

2.1.1 Presentation Layer

Anki uses the PyQt framework to create its cross-platform graphical user interface. This layer handles all user interactions such as displaying decks, cards, and scheduling information while providing a responsive experience. The UI of Anki includes various lists, buttons, forms, and dialogs. These elements are arranged to display decks, flashcards and study schedules. Anki uses Qt's layout managers (e.g., QHBoxLayout, QVBoxLayout, QGridLayout) to ensure arranging the UI components dynamically based on the screen size which guarantees a consistent user experience across different devices and platforms [5]. Furthermore, PyQt employs a robust signal and slot mechanism to manage events [6]. When a user interacts with a UI component such as clicking a button to create a new flashcard deck, signals are emitted and corresponding slots in the business logic layer are triggered to handle the event. This decoupling allows the UI to remain responsive and maintain a clear separation from application logic.

2.1.2 Business Logic Layer

The core application logic is written in Python. This layer implements key functionalities such as deck and card management, the spaced repetition scheduling algorithm, and data processing. It's separated into modules for handling note creation, card generation, and review scheduling etc. This structured approach not only makes data processing and updating across different modules easier but also it provides easier maintenance and flexibility through defined interfaces and modules. By spaced repetition abstraction and integrating dynamic data processing, it is ensured that user actions like adding or updating a card trigger consistently, and it gets more predictable to make changes in the system's state, therefore it provides a robust and adaptive learning experience.

2.1.3 Data Storage Layer

Anki stores its data which includes decks, notes, cards, and scheduling information in a local SQLite database. This database is the single source for the application. It delivers fast data retrieval and updates in offline mode while ensuring that changes can be seamlessly synchronized with cloud services such as AnkiWeb. So, it maintains consistency across multiple devices.

3. Proposed software architecture

3.1 Overview

ReMediCard.io's structured system contains multiple subsystems that work together to provide an efficient learning experience for medical students. The system is designed with a modular approach, where each subsystem is responsible for a specific function. The client subsystem provides the user interface for students, interacting with the gateway subsystem, 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, timely 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, MongoDB for unstructured content, 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

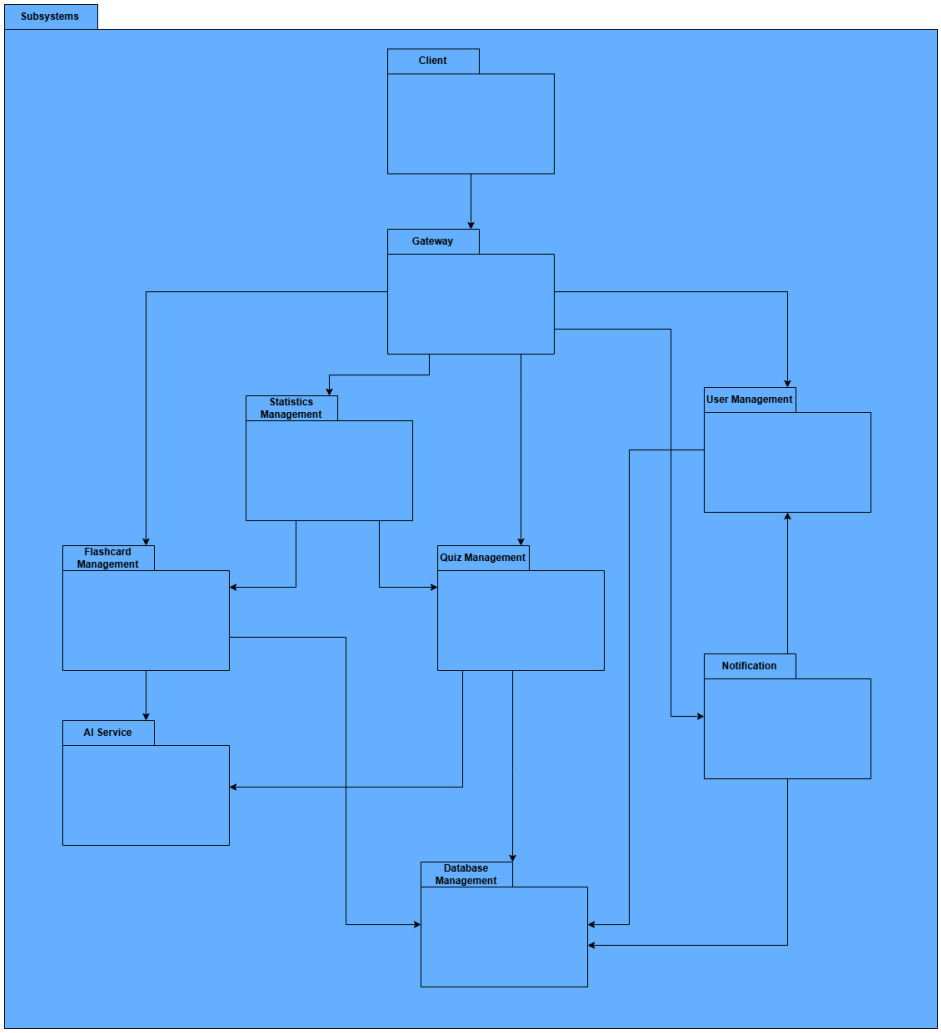

Fig. 1: Subsystem Decomposition.

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

- 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

Statistics Management Subsystem

- Tracks and analyzes user performance and study habits.
- Provides insights such as accuracy rates, progress trends, and study recommendations.
- Helps optimize spaced repetition algorithms for effective learning.

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 non-relational (MongoDB) 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 and executing simple computations such as keeping track of user statistics.

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 MongoDB for unstructured text-based content. Additionally, Amazon S3 is used for large media file storage.

PostgreSQL (relational database) is used to store structured and transactional data such as user profiles, flashcard decks and metadata, user study statistics and progress tracking. MongoDB (document-based database) 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 (audio/video), 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 MongoDB is used for direct interaction with MongoDB collections for unstructured 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.

4. Subsystem services

4.1. 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.

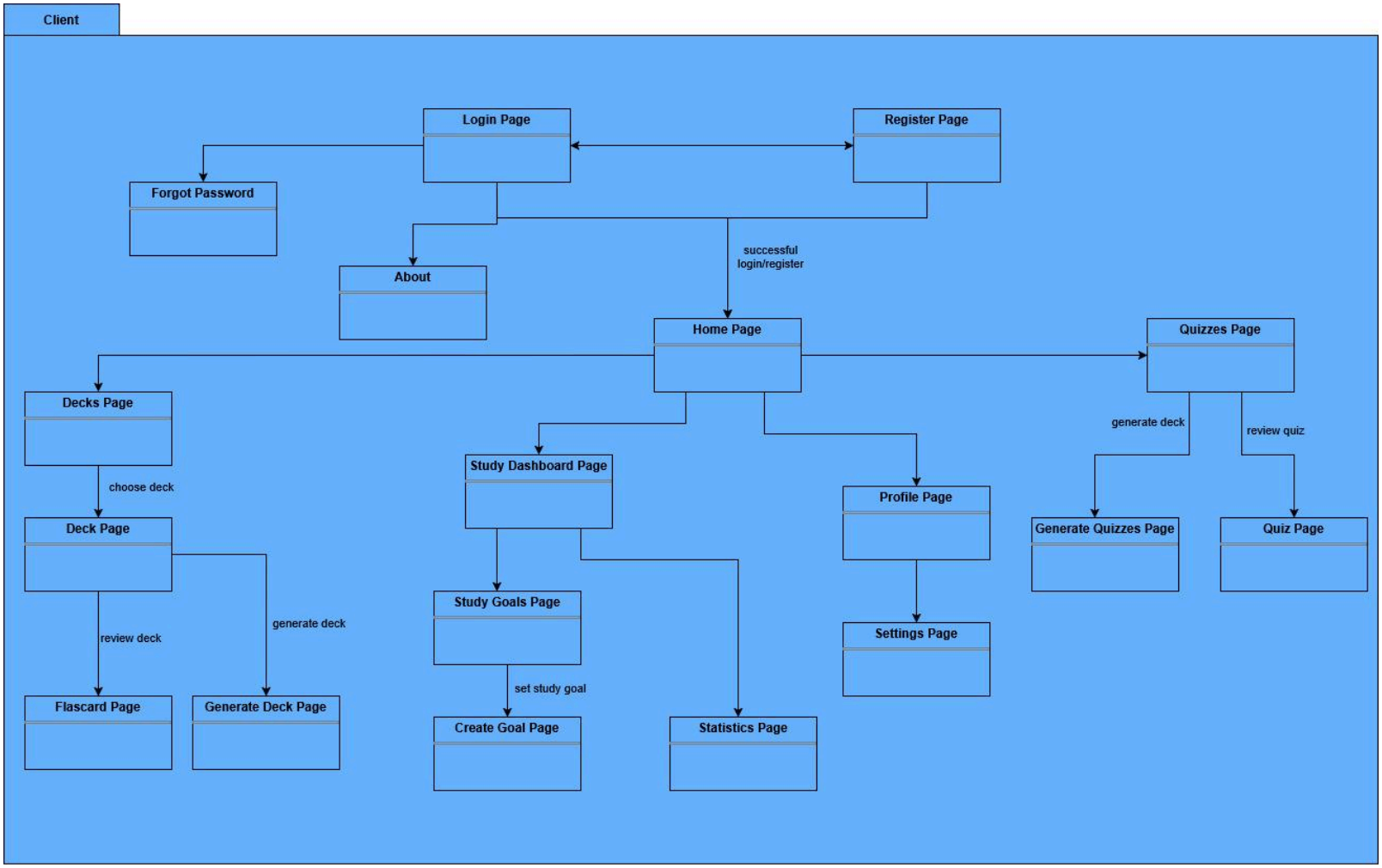

Figure 2: Client Subsystem Architecture

4.2. Backend Subsystem

A simple overview of the backend architecture is given below. 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 .

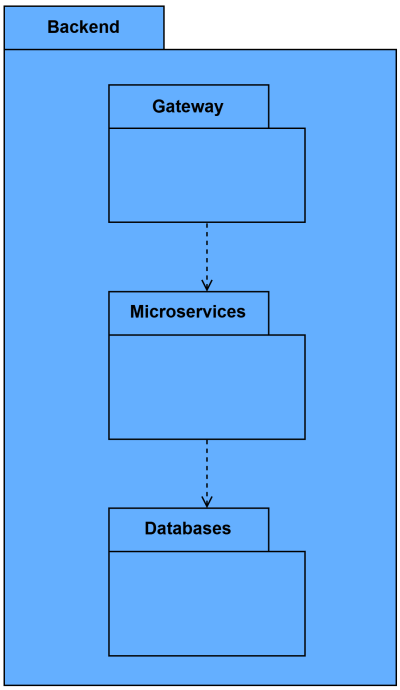

Figure 3: Backend Subsystem Architecture

In the 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.

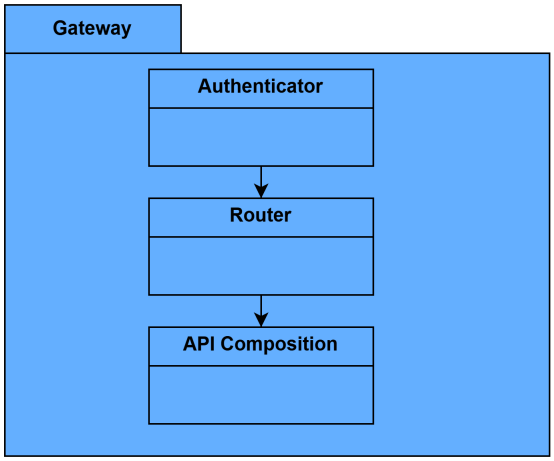

Figure 4: Backend Gateway Architecture

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.

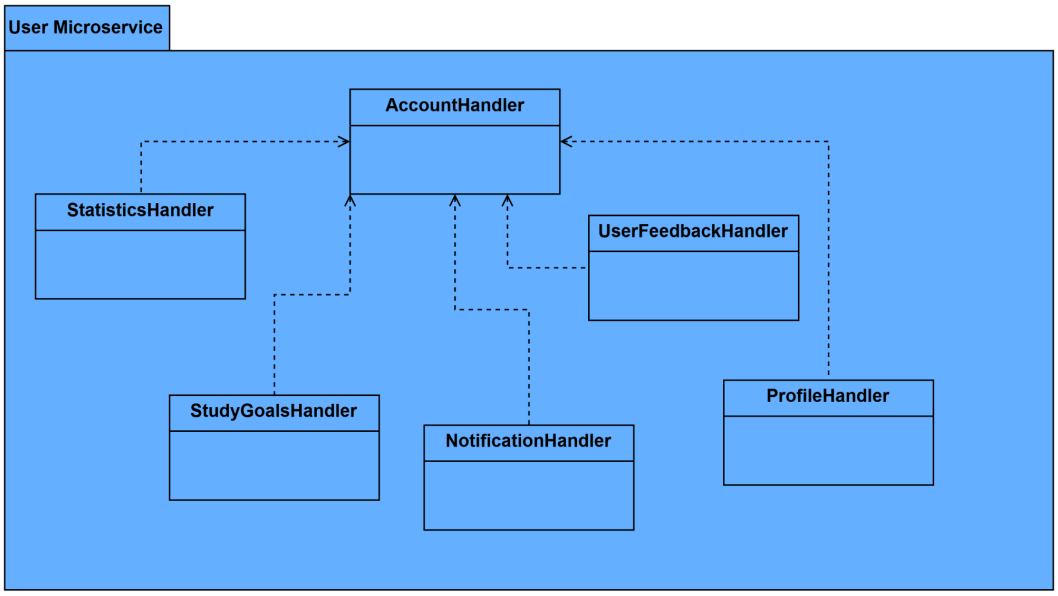

Figure 5: Backend User Microservice Architecture

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. For example, OCRHandler manages the transcript creation task for image files while SpeechToTextHandler does the same thing for video or voice records. On the other hand, QuizGenerationHandler and FlashcardGenerationHandler generate quizzes and flashcard decks respectively from the uploaded lecture content.

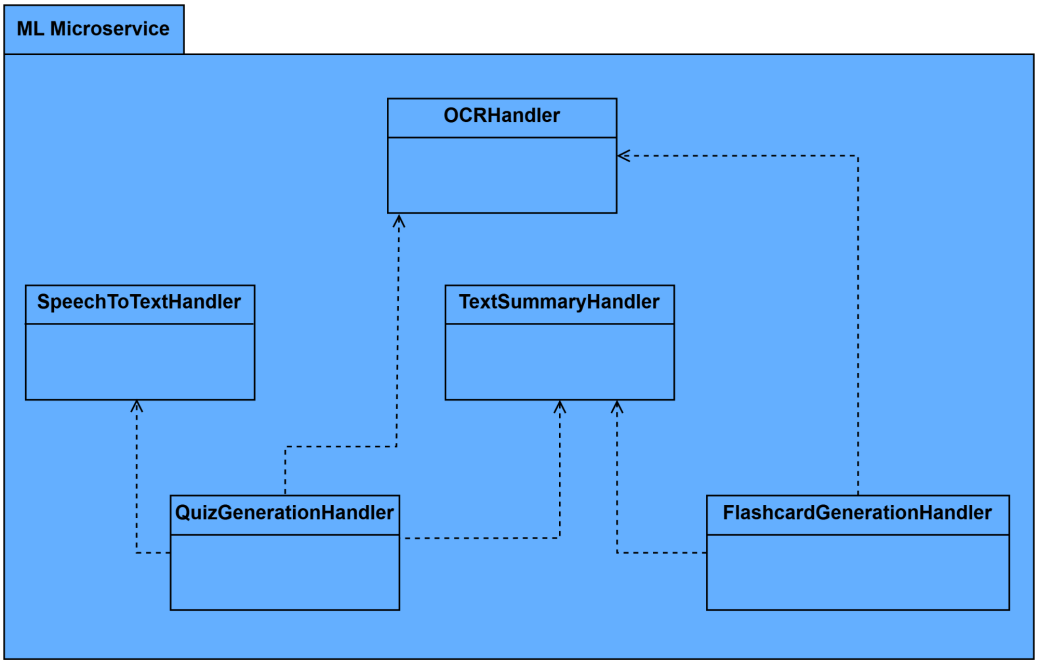

Figure 6: Backend ML Microservice Architecture

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. RepetitionAlgorithmHandler 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.

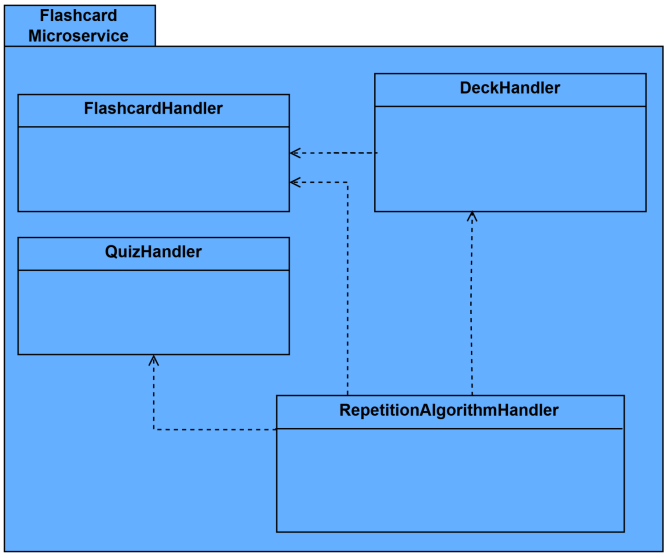

Figure 7: Backend Flashcard Microservice Architecture

5. Test Cases

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). Step 2. Monitor response times and system throughput.				
Expected	System response times remain within < 5 seconds per action under expected load.				
Date-Result	This part will be completed in the final report.				

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. Step 2. Observe if each container restarts automatically.				
Expected	The container should restart without manual intervention				
Date-Result	This part will be completed in the final report.				

Test ID	3	Type	Non Functional	Priority	Major
Objective	This test case is to verify that auto flashcard deck generation after the document should not exceed 15 minutes.				
Steps	Step 1. Login to the application Step 2. Go to the Edit/Create page. Step 3. Click Generate Cards with AI button. Step 4. Upload a source document. Step 5. Measure the time taken after uploading the documents for the generation.				
Expected	The time to respond should be less than 15 minutes				

	The system should not freeze during process
Date-Result	This part will be completed in the final report.

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 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	This part will be completed in the final report.				

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. Step 3. Simulate a data deletion request and verify that the system fully purges user data as per GDPR guidelines 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	This part will be completed in the final report.				

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. Step 2. Follow the documented backup restoration procedure step-by-step. 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. Step 2. All critical data is fully restored without loss or corruption. Step 3. The system returns to normal operational status, and all functionalities work as expected.
Date-Result	This part will be completed in the final report.

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. Step 2. Have each user perform typical actions (create decks, upload files, review flashcards) within a short time span. Step 3. Monitor response times and resource usage.				
Expected	Step 1. Response times remain within acceptable limits (e.g., under 2 seconds). Step 2. No server crashes or significant slowdowns occur. Step 3. System usage metrics (CPU, RAM) stay within defined thresholds.				
Date-Result	This part will be completed in the final report.				

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). Step 2. Measure response times and resource usage. Step 3. Identify any performance bottlenecks.				

Expected	The system should maintain stable performance under high loads
Date-Result	This part will be completed in the final report.

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. 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	This part will be completed in the final report.				

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). 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. Step 2. The user is redirected to the login screen upon the next action. Step 3. No unauthorized actions are possible after the timeout.				
Date-Result	This part will be completed in the final report.				

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. Step 2. The user clicks on the sign up button.				

Expected	The user will be redirected to the main page.
Date-Result	This part will be completed in the final report.

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. Step 2. The user clicks on the login button.				
Expected	The user will be redirected to the main page.				
Date-Result	This part will be completed in the final report.				

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. 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	This part will be completed in the final report.				

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. Step 2. The user clicks on the edit credentials button. Step 3. The user fills in the information and submits.				
Expected	User credentials must have changed.				
Date-Result	This part will be completed in the final report.				

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. Step 2. The user clicks on the Create New Deck button. Step 3. The user fills in the information and submits.
Expected	The created deck must be available among the others.
Date-Result	This part will be completed in the final report.

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. Step 2. The user clicks on the edit button of a deck. Step 3. The user fills in the information and submits.				
Expected	The deck information must be updated.				
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. Step 2. The user chooses the sorting option from the panel.				
Expected	The decks must be sorted in the order chosen.				
Date-Result	This part will be completed in the final report.				

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. 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	This part will be completed in the final report.

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. Step 2. The user chooses the sorting option from the panel.				
Expected	The quizzes must be sorted in the order chosen.				
Date-Result	This part will be completed in the final report.				

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 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	This part will be completed in the final report.				

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 Step 2. The user selects a quiz that doesn't have a best score Step 3. The user solves a number of questions and ends the quiz				
Expected	The best score statistic must be calculated correctly				
Date-Result	This part will be completed in the final report.				

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 Step 2. The user selects a quiz Step 3. The user solves a number of questions
Expected	The question progress indicator must show the correct information
Date-Result	This part will be completed in the final report.

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 Step 2. The user selects a quiz Step 3. The user solves the questions and ends the quiz				
Expected	User should be navigated to the quizzes page				
Date-Result	This part will be completed in the final report.				

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 Step 2. The user selects a quiz 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	This part will be completed in the final report.				

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. Step 2. The system processes the request and saves it to the database.				

	Step 3. Fetch the flashcard by ID from the database.
Expected	The flashcard is stored with the correct front and back text.
Date-Result	This part will be completed in the final report.

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. Step 2. The system processes and updates the flashcard in the database. Step 3. Fetch the flashcard and check if updates are applied.				
Expected	The flashcard contains the updated content.				
Date-Result	This part will be completed in the final report.				

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. Step 2. The system processes the request and removes the flashcard from the database. Step 3. Attempt to fetch the deleted flashcard.				
Expected	The flashcard no longer exists in the database.				
Date-Result	This part will be completed in the final report.				

Test ID	28	Type	Functional	Priority	Critical
Objective	This test case is to verify that users can share flashcard decks with others.				
Steps	Step 1. The user selects a deck and enables sharing. Step 2. The system makes the deck visible to other users. Step 3. Other users access the shared deck.				
Expected	The shared deck is accessible to other users.				
Date-Result	This part will be completed in the final report.				

Test ID	29	Type	Functional	Priority	Critical
Objective	This test case is to verify that user feedback is stored in the database.				
Steps	Step 1. The user submits feedback through the feedback form. Step 2. The system processes the request and stores it in the database. Step 3. Retrieve the feedback entry from the database.				
Expected	Feedback is correctly stored with the user ID, timestamp, and content.				
Date-Result	This part will be completed in the final report.				

Test ID	30	Type	Functional	Priority	Critical
Objective	This test case is to verify that submitted feedback can be retrieved by the admin.				
Steps	Step 1. The feedback is submitted by users. Step 2. Admin logs in and accesses the feedback management page. Step 3. The system fetches stored feedback entries from the database.				
Expected	Feedback entries appear in the admin panel with correct details (user, date, message).				
Date-Result	This part will be completed in the final report.				

Test ID	31	Type	Functional	Priority	Minor
Objective	This test case is to verify that users can see their previous feedback submissions.				
Steps	Step 1. The user navigates to the "My Feedback" page. Step 2. The system retrieves and displays past feedback entries of the user.				
Expected	The user sees all their past feedback messages with timestamps.				
Date-Result	This part will be completed in the final report.				

Test ID	32	Type	Functional	Priority	Minor
Objective	This test case is to verify that users cannot submit empty feedback.				

Steps	Step 1. The user attempts to submit feedback without typing any content. Step 2. The system processes the request.
Expected	The system returns an error message and prevents submission.
Date-Result	This part will be completed in the final report.

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 "Hard" during review. Step 2. The system updates the flashcard's next review date to an earlier time. Step 3. The user marks another flashcard as "Easy." Step 4. The system delays the next review date for that flashcard.				
Expected	The system schedules flashcards appropriately based on difficulty ratings.				
Date-Result	This part will be completed in the final report.				

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. Step 2. The user marks three as "Easy," four as "Medium," and three as "Hard." Step 3. The system updates intervals accordingly. Step 4. The user returns after 24 hours. 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	This part will be completed in the final report.				

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. Step 2. The user starts a review session. 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	This part will be completed in the final report.

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. Step 2. Fetch notifications from the database. 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	This part will be completed in the final report.				

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. Step 2. The system triggers an email notification. Step 3. Check the email inbox or email logs.				
Expected	The user receives a properly formatted email with a reset link.				
Date-Result	This part will be completed in the final report.				

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. Step 2. Time progresses to the predefined reminder threshold.				

	Step 3. The system triggers a notification event. 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	This part will be completed in the final report.

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. 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	This part will be completed in the final report.				

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. 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	This part will be completed in the final report.				

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.				

	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	This part will be completed in the final report.

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. Step 2. The user chooses a lecture video as the input format. Step 3. The user uploads the corresponding file to the system. Step 4. Application created the generated deck and returns. 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	This part will be completed in the final report.				

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. Step 2. Users intentionally upload hate speech, sexual content, or discriminatory content in any form. 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	This part will be completed in the final report				

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. 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	This part will be completed in the final report

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. Step 2. The user chooses the lecture content format and uploads the content file. Step 3. The user sees the queuing successful message after tThe application received. 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	This part will be completed in the final report.				

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 Step 2. The user chooses the lecture content format and uploads the content file. Step 3. The user sees the queuing successful message after the file is received. Step 4. LLM service creates the corresponding flashcard deck. 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	This part will be completed in the final report

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 Step 2. The user chooses the lecture content format and uploads the content file. Step 3. The user sees the queuing successful message after the file is received. 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	This part will be completed in the final report.				

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. 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	This part will be completed in the final report				

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. Step 2. LLM generates a quiz based on the content of the lecture material.				

	Step 3. Generated quiz questions are created, the user is notified and the quiz visible to the user. 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	This part will be completed in the final report.

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. Step 2. The user chooses an option for one of the test questions. 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	This part will be completed in the final report.				

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. Step 2. The user chooses the lecture content format and uploads the content file. Step 3. The user sees the queuing successful message after the file is received. 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	This part will be completed in the final report.				

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. Step 2. The user chooses lecture video as the input format. Step 3. The user uploads the corresponding file to the system. Step 4. The application created the generated quiz and returns. 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	This part will be completed in the final report.

6. Consideration of Various Factors in Engineering Design

6.1 Constraints

6.1.1. 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.

6.1.2. 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.

6.1.3. 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.

6.1.4. 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.

6.1.5. 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.

6.1.6. 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.

6.1.7. 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.

6.1.8. Economic Factors

The base app will be free to use. The premium version of the application will offer affordable pricing to obtain unlimited use of the features offered. Leveraging advanced AI models requires significant use of resources but promises high returns in terms of educational outcomes.

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.

6.2 Standards

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

6.2.1. 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].

6.2.2. 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].

6.2.3. AI/ML Development Standards

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

6.2.4. 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.

6.2.5. Testing Standards

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

7. Teamwork Details

7.1 Contributing and functioning effectively on the team

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.

7.2 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 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.

8. 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] Ankitects, “GitHub - ankitects/anki: Anki’s shared backend and web components, and the Qt frontend,” *GitHub*. https://github.com/ankitects/anki [Accessed: Mar 6, 2025]

[5] “Layout Management | QT Widgets 6.8.2.” https://doc.qt.io/qt-6/layout.html

[6] “Old-style signal and slot support — PYQT 4.12.3 Reference guide.” https://www.riverbankcomputing.com/static/Docs/PyQt4/old_style_signals_slots.html

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

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

[9] “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].

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

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

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

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

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