Team B
CSCI 441

Flashcard Application

Prepared for:
Dr. Mireku Kwakye

Project URL: https://sites.google.com/view/teambflashcard/home
GitHub URL: https://github.com/ahmedjaff90/CSCI441_flash_card_app

Table of Contents

Contents

Work Assignment

a. Contributions and Breakdown of Work in Report - RACI Chart

	Megan Hall	Ahmed Alzehhawi	Joseph Tracy	Steven Smith
Problem Statement	C	A/R	C	C
Sub-problems	C	A/R	C	C
Glossary of Terms	A/R	C	C	C
Business Goals	A/R	C	C	C
Functional Requirements	C	C	C	A/R
Nonfunctional Requirements	C	C	C	A/R
UI Requirements	C	C	A/R	C
Use Cases	A/R	A/R	C	A/R
User Interface Specification	C	C	A/R	A/R
System Architecture	A/R	A/R	C	A/R
Project Size Estimation	C	C	A/R	C
Conceptual Model	A/R	C	C	A/R
System Operation Contracts	C	A/R	C	C
Data Model and Persistent Storage	C	C	A/R	C
Interactions Diagram	C	A/R	C	C
Class Diagram	C	C	C	A/R
Data Types and Operation Signatures	C	C	A/R	C
Class Traceability Matrix	A/R	C	C	C
Data Structures	C	C	A/R	C
Concurrency	A/R	C	C	C
Design of Tests	C	A/R	A/R	C
Project Management	A/R	C	C	C
References	C	A/R	C	C

Key	R: Responsible	A: Accountable	C: Consulted	I: Informed

Summary of Changes

Modifications have been made to this report based on implementation:

- Removed decomposition into sub-problems items: Enable Virtual Study Group and Helpful Study Tips.
- We noted that Functional Requirements 6 – 8 are outlined, but not implemented. Instead, these requirements are casually described upon disclosing goals for future features.
- Photos of the UI design mock-ups have been replaced with captures of the functioning Web Application.
- Section 4, System Sequence Diagram for Use Case "Sign-up Email" has been modified to reflect current implementation. At the current time, we are unable to implement a verification link send through email.
- Revised Section 5 from original reports to reflect Effort Estimation using Use Case points assuming the productivity factor is 28 hours per use case.
- Section 6: Traceability Matrix, and diagrams have been revised to reflect implementation including the Domain model, concept definitions, association definitions, attribute definitions, and system operation contracts.
- Section 7: Interaction diagrams updated to reflect implemented use case and show design patterns.
- Added Class OCL Contracts to section 8.
- Added changes to our System Architecture design to show changes to Connectors and Network protocol. This change reflects that the successful database connection was achieved through a MongoDB Atlas NoSQL database, which has been altered since our first demonstration to allow scalability and success.
- Section 10 has updated captures of User Interface specifications and Design implementation that is currently functioning, along with descriptions of changes made from the preliminary UI design to the current UI design and how these designs satisfy Use Cases.
- There have been changes to our project management section. Plan of work has become "History of Work".

1. Customer Problem Statement

a. Problem Statement:

Computer Science stands as a discipline that demands a considerable amount study and effort for students to grasp and apply its intricate concepts effectively. Students in this realm of study often struggle to discover engaging and entertaining methods to meet these rigorous academic goals. This is especially true when trying to seek methods to make learning a more portable and accessible endeavor. Traditionally, studying has long been perceived as an activity that demands a stationary approach for success; but as technology advances, students are progressively seeking out alternative approaches that help them achieve their educational targets.

History of Flashcards

Historically, flashcards have played a pivotal role in improving student success rates. Research indicates that students who incorporate flashcards into their study routine receive higher exam scores compared to their peers who do not employ this method (Golding, 2012). Although researchers have proven the value of flashcards, the current physical application of this tool leaves much to be desired and warrants improvement. The same could also be said about current studying websites. In theory, developments in technology should allow students more freedom and modernized tools. Flashcard applications could be updated to reflect a more contemporary style of learning simply adding innovative features such as quizzes in real time. Current applications can tend to forego or exclude such innovative features. One can only assume that in updating the outdated idea of physical flashcard by building a digital application, a student's learning experience can improve drastically.

Insight to Imperfections

Upon scrutiny, conventional studying websites reveal a lack enjoyment and proper engagement for students. Interaction with these interfaces exposes common design flaws and issue, leading to user frustration. Issues such as: JavaScript errors, limited collaboration, lack of interactivity, and scarcity of well-timed application updates plague these platforms, obstructing the learning experience. Additionally, the information provided to students for studying purposes can often lack quality and introduce more confusion in the process. Incorporating solutions to these current issues can improve satisfaction while studying and increase chances of student success.

Current Issues:

- **JavaScript Errors:** Existing flashcard applications frequently encounter JavaScript errors, disrupting study sessions and causing frustration.

- **Limited Collaboration:** Many flashcard applications, through lack of features, are incapable of facilitating group study, which limits collaboration between friends and classmates.
- **Lack of interactivity:** Current flashcard applications lack engaging features, which makes diminishes the overall excitement in the learning process.
- **Application Updates:** Irregular and infrequent updates can compromise the learning experience and lessen the quality of a student's studying experience.
- **Quality of Information:** Information that is not reflecting the current standard and train of thought can cause confusion, decrease in student confidence, and the stagnation of skill mastery.
- **"Clicks" Issue:** Many current flashcard applications have issues specifically with button clicks. This issue can cause confusions for users and may at times, not even function.
- **Inactive search:** The search functionality in most flash card applications is available, and that makes it challenging for some users to find the content that they are looking for.
- **User Feedback:** Some popular flashcard applications exclude insights that let the users know how well they are progressing through a subject or where they need to improve.
- **Poor Design Application:** If an application is poorly designed, discrepancies in the sizes and styles font and colors used within the application can cause issues with accessibility and make the application ineffectual to students with accessibility needs.

Improvement Suggestions:

The previously stated issues suggest there is a need for educational tools that change the monotonous, traditional ways in which students are accustomed to studying. A flashcard application tailored to Computer Science concepts and terminology could address these challenges and improve a student's overall proficiency. The application should be well-designed and modern. This would give students the opportunity for virtual group study and access across numerous devices. Creating a simple user-friendly application can allow students of all ages the opportunity to utilize the application. Frequent and timely updates allow new ideas and exciting improvements to keep students engaged. Providing collaborative features to increase participation could allow students to study not only with their friends, but other students who are interested in the same areas of Computer Science study. Quality assurance of materials allows students to feel confident in the information they are studying. Allowing users to create free accounts permits students to access knowledge without paying a large sum of money in the process.

b. Decomposition into Sub-problem:

- **Design and Modern Application:** Encourages that users have a good experience using the application and motivates users' engagement.

- **Accessible From Numerous Devices:** The application should be accessible from Numerous Devices such as Laptops, Tablets, Phones, and iPads.
- **Simple User-Friendly Application:** The application should be easy to use and accessible to students of all ages.
- **Frequent Updates:** The application should be updated regularly to avoid any issues, bugs, and security threats.
- **Quality Assurance of Information:** Databases housing information for the application are checked and updated regularly to reflect correctness and adhere to current education standards.
- **Free Sign-up:** The application should offer an excellent learning experience that free of charge to all general users.
- **Helpful Study Tips:** For concepts that metrics and feedback show the student is struggling, the application could provide "tips" that offer alternative explanations or hints to help better understanding.
- **Accessibility Accountability:** The application should meet the standards of Web Accessibility for students with disabilities and needs for accessible technology.

By directing these Sub-problems, the suggested flashcard application aims to transform the studying experience for Computer Science students, providing a modern, secure, and collaborative learning environment.

2. Glossary of Terms

a. Glossary of Terms

Flashcard: A digital card used for memorization of concepts. Information is stored on both sides.

Deck: A collection of flashcards. Decks will be grouped depending on the subject and category.

Revision: The practice of reviewing previously learned information.

Shuffle: A randomization feature that cycles through flashcard sets. Allows for the prevention of rote memorization based on sequence.

Animation: The effect that simulates flipping a flashcard to reveal the answer initially hidden from the user.

Progress: Advancement and achievement through studying subject material

Tracking: Quantification of progress through metrics

Metrics: standard of measurements (number of active users, percentage of materials viewed, etc.).

Spaced Repetition: A flashcard learning technique. A student reviews flashcards at systemic intervals.

Active Recall: A learning strategy. Students actively retrieve information from memory as opposed to passive learning methods.

Sync: Synchronization of data across multiple devices.

Offline Mode: The ability to access features and flashcard sets without an internet connection.

Student: The user.

Rating: Assigning a score to a set of flashcards (e.g. 1 to 5 stars).

Quiz: A feature that presents information from the flashcards in a quiz format, testing the user's knowledge.

Study Group: A group of students that collaborate and share study resources.

Feedback: Allows users to provide feedback about the application for future improvements.

User Profile: a personal account that the student can use to keep track of progress, stores user settings, and preferences.

Computer Science: The study of the principles and use of computers.

Sign-In: The process of signing into an individualized user account.

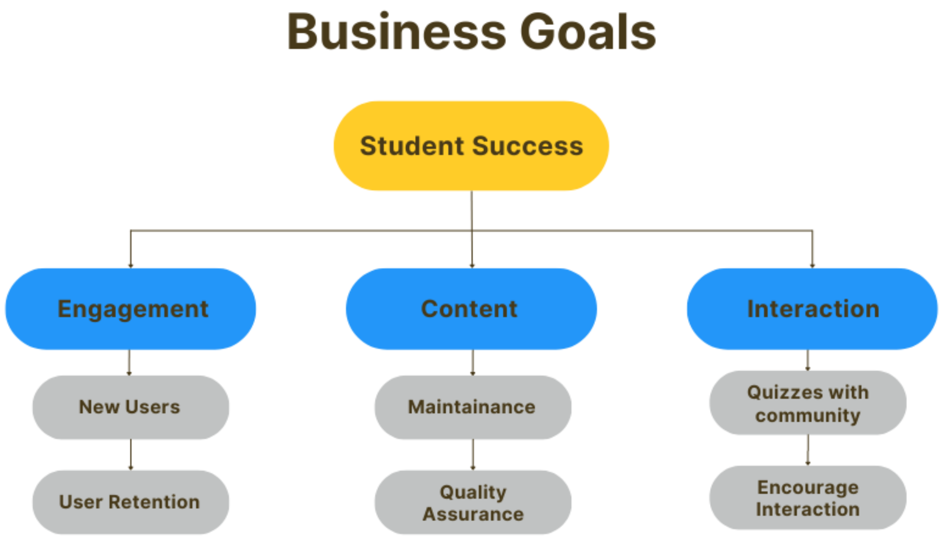
Sign-Up: The process of creating an individual user account.

3. System Requirements

a. Business Goals

The overall Business goal that we're trying to reach is ultimately achieving success not only for the application, but for the students themselves. Reaching these goals could later lead to monetization and scalable improvements within the application. To achieve this success, three subsequent goals will need to initially be met: Engagement, Content, Interaction.

By focusing on engagement, we can aim to quickly form and grow our user base through acquiring new users and retaining active members with established accounts. Assuring that content is upheld to a quality standard with regular maintenance, helps to ensure students are maximizing subject mastery. Likewise, interaction between users bolsters community and allows users to challenge each other to achieve their personal learning goals.



b. Functional Requirements:

Identifier	Priority	Requirement
REQ-1	5	The system will provide links to view flashcards for each provided section of Computer Science.
REQ-2	5	The system will provide a search bar for the user to query a specific section they are wanting to study.
REQ-3	5	The system sends a request to the database via API and retrieves the data from the corresponding query.
REQ-4	4	The system will provide a sent email to the user, when they sign up for a free account.
REQ-5	2	The system will provide an email address for users to contact with questions or concerns.
REQ-6 *	3	The system will alert the user once they reached a certain number of flashcards studied or time spent studying on the application.
REQ-7 *	3	The system will suggest other flashcards or activities to participate in, depending on previous searches utilized by a user.
REQ-8 *	2	The system will provide a section for users to save flashcards for future use in a catalogued group.

Requirements denoted with an * are future features that are not yet implemented, but that we do find value and worth noting.

c. Nonfunctional Requirements:

Identifier	Priority	Requirement
NFR REQ-1	5	The system will provide an interactive screen that will show flashcards being used for studying.
NFR REQ-2	5	The system must keep user-provided information private
NFR REQ-3	4	The system should be able to handle multiple levels of user traffic.
NFR REQ-4	3	The system should be able to handle any changes in OS, devices, or any other change on the users end. Also, for any change in system specs.
NFR REQ-5	3	The system should be able to run efficiently on any of the acceptable devices.

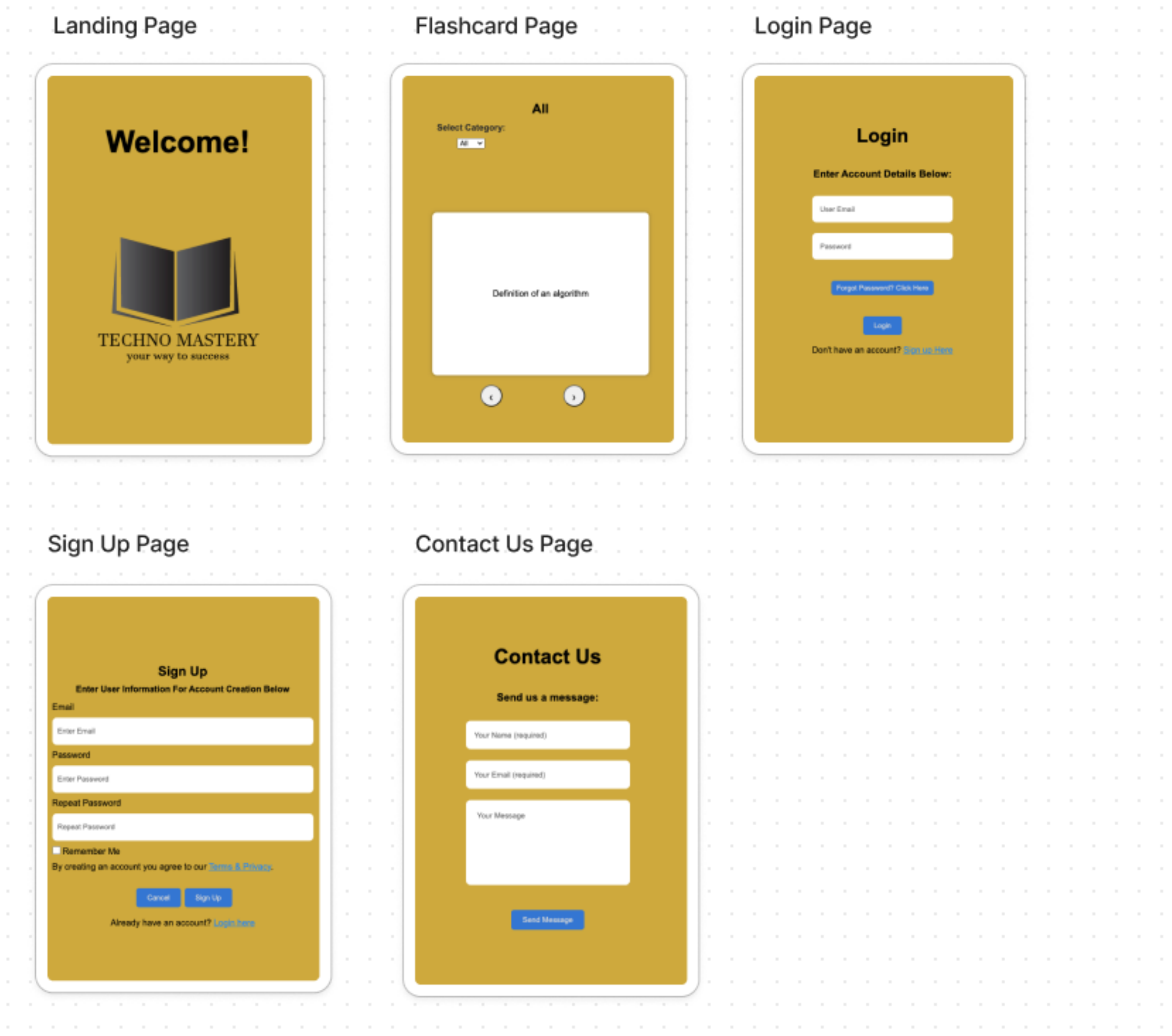
d. User Interface Requirements:

Identifier	Priority	Requirement
US REQ-1	5	UI will have the title head on the homepage of the application as well as on top of each set of flashcards.
US REQ-2	5	The UI will include a “flashcards” button that will incorporate a dropdown menu, leading to the various study categories provided.
US REQ-3	5	UI will contain a login and sign-up window for users.
US REQ-4	3	UI will contain home button, flashcards, login, and sign-up buttons to navigate.
US REQ-5	3	UI will contain a search bar.
US REQ-6	3	Flashcards will have effects such as appearing and disappearing.

Product Logo:

TECHNO MASTERY
your way to success

On-Screen Appearance:



4. Functional Requirement Specifications

a. Stakeholders

Many different groups of people will be stakeholders for our Computer Science Flashcard application. Stakeholders will range from everyday users seeking to learn new skills to technology businesses desiring to provide tools necessary for their employees to review concepts of Computer Science that they might need refreshed. What follows here is a detailed list of some potential stakeholders for our application:

- Users (Educational Purpose or General Everyday Use): These users are everyday people who are seeking to learn a new subject or require an educational tool to aid them through their current educational journey. These users will be affected by this application through direct access to specific sections of Computer Science that they may be interested in.
- Businesses that relate to Computer Science or other technology: This group, which could be business organizations, managers, or business owners, would have a stake in this application for a couple of different reasons. Utilizing this flash card application can help their employees refresh their knowledge of certain concepts or recollect specified vocabulary. The flashcard application could also be a part of a sponsorship with certain businesses, which would create a relationship between the application and the business, thus showing that our application is relevant.
- Schools: Our Computer Science flashcard application would be very beneficial to students, teachers, and other educational institutions, globally. Our sole purpose for this application is to provide the tools for learning topics of Computer Science to anyone who wishes to wield them, thus making schools a major stakeholder for our application. Many schools of varying grade levels have computer science programs. A flashcard application dedicated to the subject of Computer Science would be very useful for students to use throughout their educational journey.

b. Actors and Goals

Actors:

Primary Initiating Actors:

- User: An individual who uses the application and interacts with the system.
- Business Organizations
- Schools

These actors will interact with different devices such as desktops, mobile phones, or tablets, to initiate and utilize the Computer Science Flash Card Application. The system developers will ensure the system to be functions and does what it needs to when users interact with it.

Secondary Participating Actors:

- Devices used to access the application
- Application System
- API
- Database

Goals:

Users/Business Organizations/Schools:

- Uses devices to enter application via the internet.
- Accesses sign-up information to create new account or log into existing account.
- Clicks on different navigational links/buttons to search or access specific flashcards.
- Accesses email link provided to ask questions or leave feedback.
- Uses the system to share flash cards with other users.
- Uses the system to initiate a study session or start up a quiz for a certain section.

Devices:

- Devices, such as desktops, mobile phones, and tablets, work in direct relation with the user to allow access to the system to be for general use.
- Devices allow flash cards from the application to be saved for future use.
- Devices utilize their screens to enable users to see the application and navigate where they need to go.

System to Be:

- The source of the application being utilized by the user.
- Houses all flashcards needed for the system to be function and for users to be able to study.
- Stores and saves user information needed to access the site such as login information and information needed to set up a user profile.
- Provides navigational links for users to interact with the study whatever section of Computer Science they are needing.

c. Use Cases

I. Implemented Features Description

Following is the casual text description of the identified used cases against the requirements that each use case respond of our Flashcard Application:

→ View Flashcards

Description: The system will provide links to view flashcards for each provided section of Computer Science. These sections are connected via our database and API connections using Mongo DB Atlas and display both the definition and answer on the respective flashcard face.
Requirements: REQ-1, REQ-5, NFREQ-1, NFREQ-2, NFREQ-4, NFREQ-5 US REQ-1, US REQ -2, US REQ -4, US REQ -6

→ Sign-up Email

Description: The system will provide a unique account to the user when they sign up for a free with an email address.
Requirements: REQ-4, NFREQ-3, NFREQ-4, NFREQ-5, US REQ -3, US REQ -4

→ Retrieve Flashcard Data

Description: The system will retrieve flashcard data stored in the application's database to study a Computer Science concept or language.
Requirements: REQ-3, USREQ-6

→ Contact Support

Description: The system will provide an email address for users to contact with questions or concerns.
Requirements: REQ-5

II. Future Features Use Case Casual Description

The following Use Cases casually describe features that we were unable to implement by the time of our final demonstration. Although, they are not currently functioning, our team believes that these Use Cases and features are not only valuable to our web application but would positively help in the products growth.

→ Search Flashcards

Description: The system will provide a search bar for the user to query a specific section they are wanting to study.
Requirements: REQ-2, NFREQ-3, US REQ -5

→ Track Progress Alert

Description: The system will alert the user once they reached a certain number of flashcards studied or time spent studying on the application

→ Personalized Suggestions

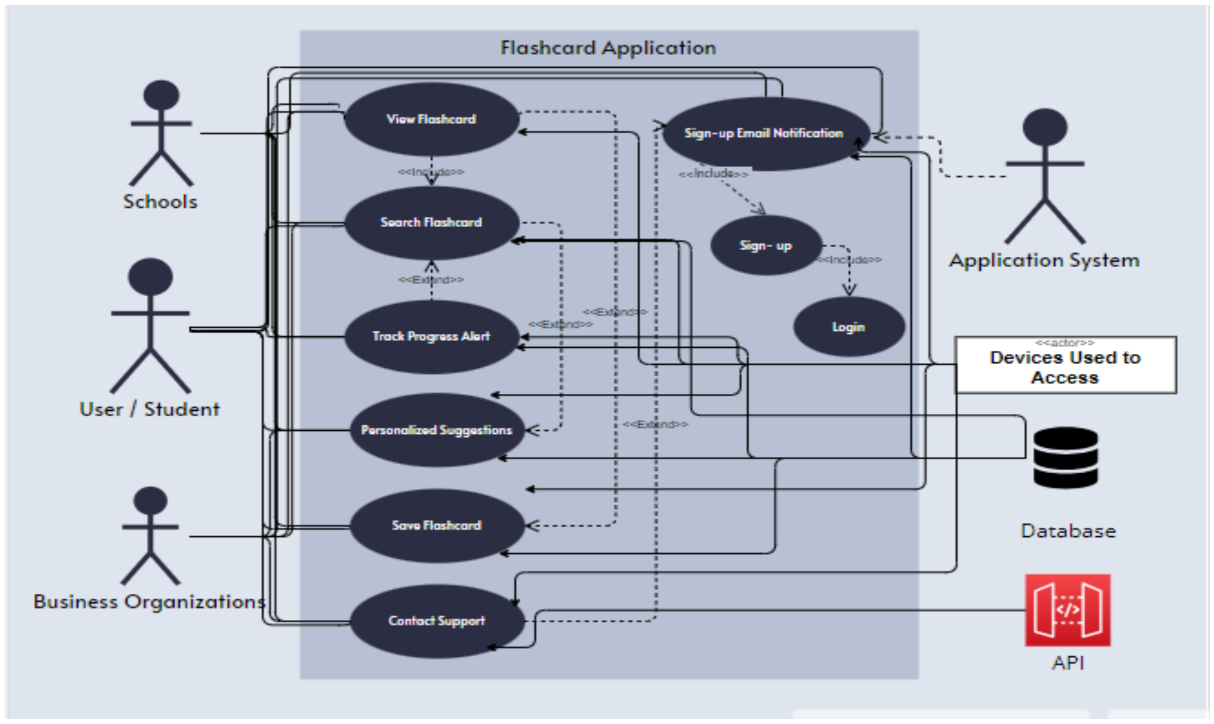
Description: The system will suggest other flashcards or activities to participate in, depending on previous searches utilized by a user.

→ Save Flashcards

Description: The system will provide a section for users to save flashcards for future use in a cataloged group.

III. Use Case Diagram

The following is the use case diagram of our Flashcard Application project:



Use case diagram displays current and future feature design.

IV. Traceability Matrix – Implemented Features

Use Case	View Flashcards	Sign-up Email	Retrieve Flashcard Data	Contact Support	Search Flashcards
REQ - 1	✓				
REQ - 2					✓
REQ - 3			✓		
REQ - 4		✓			
REQ - 5	✓			✓	
NFREQ - 1	✓				
NFREQ - 2	✓				
NFREQ - 3		✓			✓
NFREQ - 4	✓	✓			
NFREQ - 5	✓	✓			
US REQ - 1	✓				
US REQ - 2	✓				
US REQ - 3		✓			
US REQ - 4	✓	✓			
US REQ - 5					✓
US REQ - 6	✓		✓		
Total Weight	7 + 16 + 16 = **39**	4 + 10 + 8 = **22**	5 + 0 + 3 = **8**	2 + 0 + 0 = **2**	5 + 4 + 3 = **12**

We can summarize the table as following:

Use Case	Requirements	Priority Weight
View Flashcards	REQ-1, REQ-5, NFREQ-1, NFREQ-2, NFREQ-4, NFREQ-5 US REQ-1, US REQ -2, US REQ -4, US REQ -6	5+2+5+5+4+3+5+5+3+3 = 39
Search Flashcards	REQ-2, NFREQ-3, US REQ -5	5+4+3 = 12
Sign-up Email Notifications	REQ-4, NFREQ-3, NFREQ-4, NFREQ-5, US REQ -3, US REQ -4	4+4+3+3+5+3 = 22
Retrieve Flashcard Data	REQ-3, USREQ-6	5 + 3 = 8
Contact Support	REQ-5	2

The weights calculated above are including the weights of Nonfunctional and User Interface Requirements as well. If we exclude them, the use case of "View Flashcards" has the highest priority followed by the "Sign-up Email Notifications" use case.

Elaboration:

1. **View Flashcards:** With a total highest priority weight, this use case contains the top priority. It addresses several significant requirements such as allowing users to view flashcards for each provided section of Computer Science, while providing an interactive screen to show flashcards and ensuring the system handles multiple levels of user traffic.

2. **Sign-up Email:** With the second highest total weight of, this use case involves sending an email to the user upon signing up for a free account, meeting the requirement for email notification and keeping user-provided information private. Also, the window should contain navigation and particular buttons to process with sign-in, sign-up, and logout followed by this feature.

The above use cases should be given priority for elaboration and planning of our first demonstration. The use cases address the most critical requirements needed in our flashcard application.

V. Fully Dressed Description of Use Cases

The most important use cases as specified by traceability matrix include “View Flashcards” and “Sign-up Email Notification”. Both fully dressed descriptions are as follows.

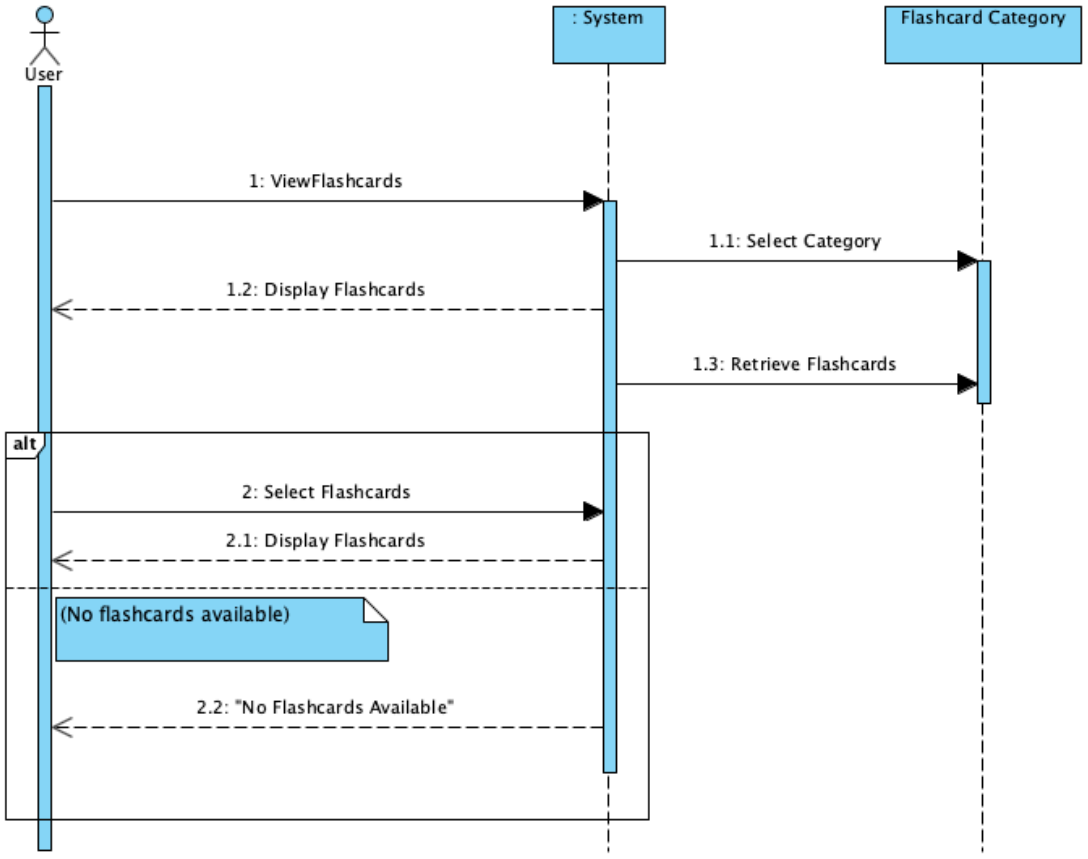
Use Case ID	1
Use Case Name	View Flashcards
Description	The system will provide links to view flashcards for each provided section of Computer Science.
Preconditions	• The user is successfully logged into the system or visiting the application as a guest. • Flashcards must be available for the selected section.
Postcondition	• The user can study flashcards for the selected section.
Main Flow Steps	• User navigates to the “View Flashcards” section. • User selects a specific section of Computer Science • System displays all the available flashcards for the selected section on both front and back sections of the flashcard.
Alternative Steps	If no flashcards are available for the selected section, the category will not be populated into the dropdown menu. Likewise, if flashcard data is not available due to an unforeseen error, the flashcard will remain blank.

Use Case ID	3
Use Case Name	Sign-up Email
Description	The system will provide a unique account to the user, when they sign up for a free account using a valid email address.
Preconditions	User has initiated the sign-up process by clicking on the “sign-up” button.
Postcondition	User data is successfully received into the database creating a unique account.

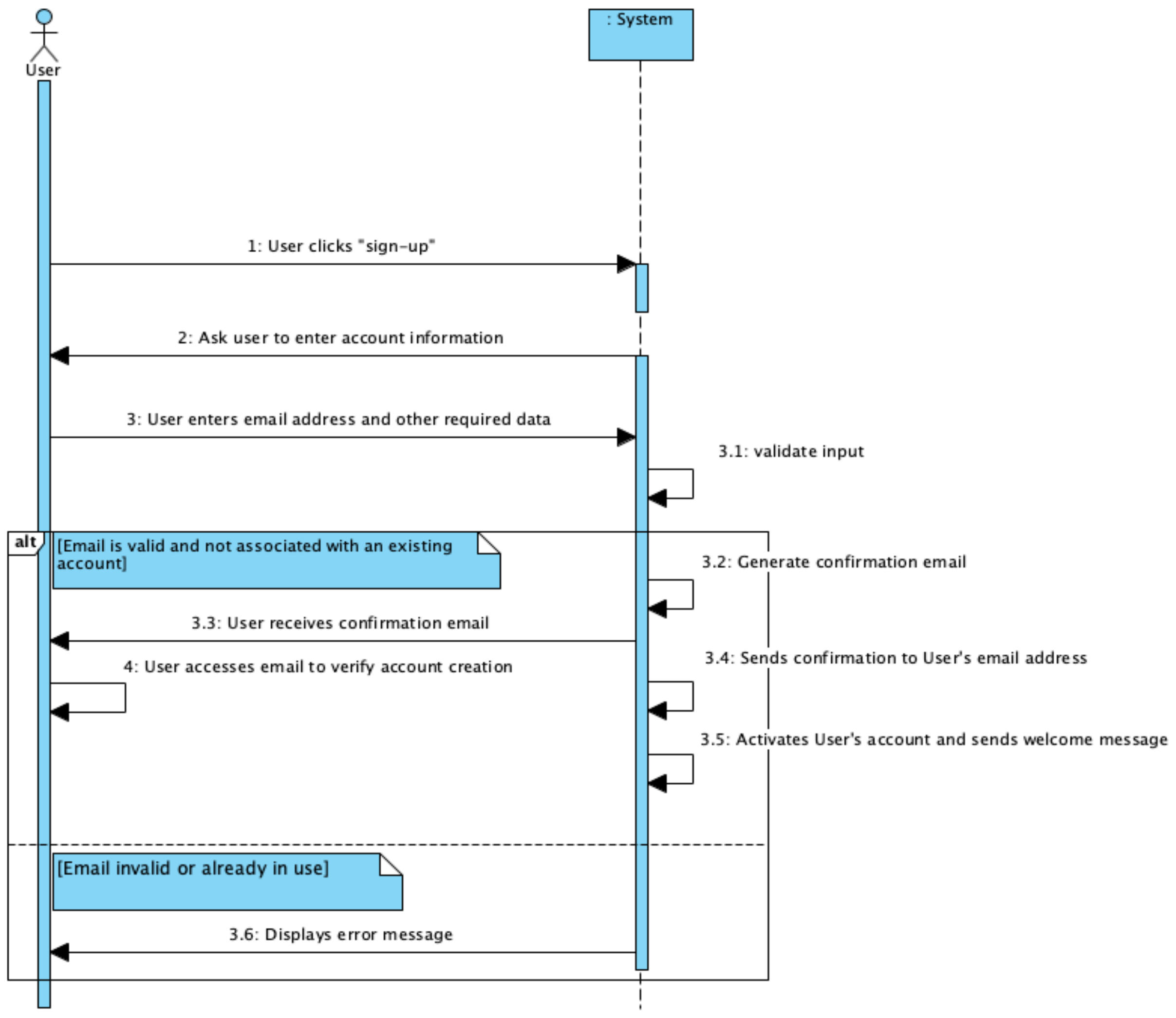
Main Flow Steps	• User clicks on the “sign-up” button. • The system initiates the sign-up process and asks user to enter necessary information including email address. • User enters the email address and other required details. • System validates the input provided by the user and checks for authenticity or existing accounts associated with the provided email address. • If the email address is valid and not already associated with an existing account, the system generates a confirmation email. • System sends the confirmation email to the user’s provided email. • Users receive the confirmation to their inbox.
Alternative Steps	If the email address is invalid or already associated with an existing account, the system shows the error message to the user to provide a valid email address or else log in if the email address is already registered.

d. System Sequence Diagrams

UC-1 View Flashcards:



UC-3 Sign-up Email:


User
: System
1: User clicks "sign-up"
2: Ask user to enter account information
3: User enters email address and other required data
3.1: validate input
alt
[Email is valid and not associated with an existing account]
3.2: Generate confirmation email
3.3: User receives confirmation email
4: User accesses email to verify account creation
3.4: Sends confirmation to User's email address
3.5: Activates User's account and sends welcome message
[Email invalid or already in use]
3.6: Displays error message

5. Effort Estimations

a. User Effort Estimation

- User clicks on website link for the Computer Science Flash Card Application through the internet, to gain access to the system itself: Potential clicks or keystrokes could be 1 click to start up the internet, 1 click to click the search bar, upwards of 20 keystrokes to type in the application name, and 1 click to access the system website. **Total= 3 clicks and approx. 20 keystrokes.**
- User locating and accessing specific section of the application: 1 click to access search bar, upwards of 20 keystrokes to enter information needed to search, 1 click to access navigational links if needed, 1 click to access specific flash card set, 1 click to access quizzes, 1 click to access link provided for sharing flash cards. **Total = Approx. 5 clicks depending on path taken through application and approx. 20 keystrokes if necessary**.
- User sets up account or logs into existing account: After arrival to system website, 1 click to access log in text bar, 1 click to access new user sign up, upwards of 30 keystrokes to enter in log in credentials, upwards of 50 keystrokes to enter new sign up information, 1 click to hit the log in button, and 1 click to hit finish/sign up after new user information inserted. Total = **Approx. 4 clicks depending on path taken and approx. 50 keystrokes depending on just logging in or setting up new account**

Throughout all the potential scenarios that the user can take to access and utilize the application, 70% of the time the user is clicking to navigate, which correlates to the user interface navigation. The other 30% relates to the information that the user types in to provide their log in or sign-up credentials as well as typing in specific search criteria to access.

b. Project Duration Effort Estimation Using Use Case Points

Using the following use case points for each use case:

- View Flashcards: 39
- Sign-up Email Notifications: 2
- Retrieve Flashcard Data: 8
- Contact Support: 2
- Search Flashcards: 12

We can then calculate the overall use case points for the application:

Total = 39 + 22 + 8 + 2 + 12 + 3 + 3 + 2 = **83 UCP**

When the Productivity Factor (PF = 28 hours) is applied and inserted into the effort estimation formula we can conclude:

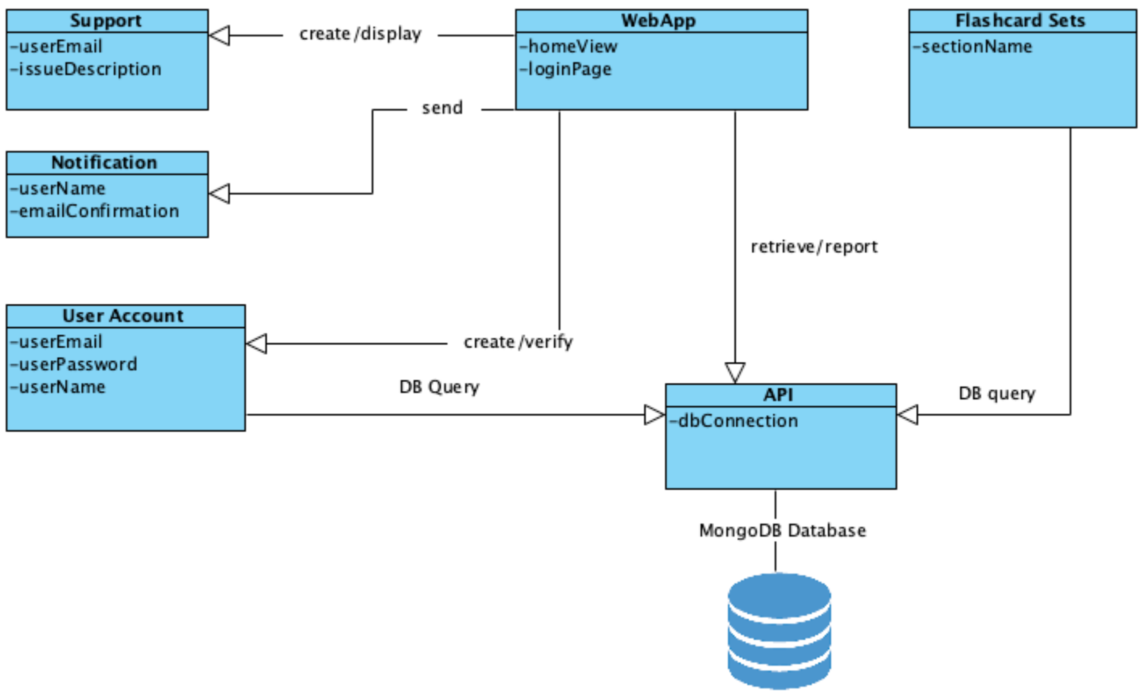
Duration = UCP x PF = 83 x 28 = **2324 hours.**

6. Domain Analysis

a. Conceptual Model

In this section, we have created a conceptual model for our Flashcard Application. This model showcases the structure and the foundation on which our application will operate in relation to its everyday use. This conceptual model will reveal the key concepts, concept associations, attributes, and matrix traceability of our application, which will compare the relationships between its development, user requirements, and system functionality.

Domain Model



I. Concept Definitions for Implemented Design

Responsibility Description	*Type (D=Doing, K=Knowing)*	*Concept Name*
Contains all the information regarding User Accounts: Login, Username, Registration Email	K	User Account
Contains the section title of subject being studied	K	Flashcard Sets
Contains the User Email for the support ticket	K	Support
Enables the issue described on the support ticket to be sent to the Support Team	D	Support
Securely connects with databases to fetch or store information related to users and flash cards	D	API
Provides User Interface for User Interaction with System	D	Web App
Collects and Utilizes User Information related to Account Creation	D	Web App
Provides notification confirmation of Account Creation	D	Web App
Enables use of flash cards for studying	D	web app
Provides interface for System admins to observe information related to user accounts and System reports	D	Web app
Contains Information related to Account and Confirmation Email	K	Notification

II. Association Definitions for Implemented Design

Concept Pair	*Association Definition*	*Association Name*
API ↔ User Account	Creates User Account Information and stores in system database	DB query
API ↔ Flashcard Sets	Flash card information is fetched from database and saved and stored on separate database for use on the application	DB query
WebApp ↔ Support	creates and displays support ticket information	create/display
WebApp ↔ User Accounts	Creates User Accounts and verifies user emails to allow access to system	create/verify
WebApp ↔ Notification	Sends notifications to users related to account and system activities	Send
WebApp ↔ API	Retrieves information related to usage and system reports	retrieve/report

III. Attribute Definitions for Implemented Design

Concept	Attribute	Description
WebApp	homeView	A user interface element responsible for displaying flashcards
	loginPage	User interface element that allows the user to securely log into their account
User Account	userEmail	The email associated with the email address and password within the webapp.
	userPassword	The password associated with the email address and password of the user.

	userName	An identifier chosen by the user to access their account
Flashcard Sets	sectionName	The title of the section associated with a set of flashcards.
Support	userEmail	The email associated with the user submitting the support request.
	issueDescription	A description of the issue reported by the user.
API	dbConnection	Connection to the information to be displayed on the flashcards and configuration details required for the API to connect to the database.
Notifications	userName	The name of the account associated with a particular notification.
	emailConfirm	An email sent to the email address associated with the user account notifying account creation has been successful.

IV. Traceability Matrix for Implemented Domain Concepts

Use Case	PW	Domain Concepts					
		WebApp	User Account	Flashcard Sets	Support	API	Notifications
View Flashcards	5	✓		✓			
Search Flashcards	5	✓		✓			
Sign-up Email Notifications	5	✓					✓
Retrieve Flashcard Data	3	✓				✓	
Contact Support	4	✓	✓		✓		

b. System Operation Contracts

Based on the provided descriptions and associated requirements for each fully dressed use case, here are the corresponding system operation contracts:

The System Operation Contracts for the use cases elaborated in 3c for their system operations identified in 3d particularly are as follows:

For View Flashcards:

Contract	When a user clicks on the "View Flashcard" option, the system will provide links to view flashcards for each provided section of Computer Science.
Input	Particular Sections
Output	Link to "View Flashcard" / user will be navigated to the flashcards to view information.
Exceptions	If the section or flashcards are not available (i.e. No flashcard exists or uploaded), then the system will provide a proper exception or error prompt.

For Sign-up Email Notifications:

Contract	When the user clicks on the "Sign-up" option, the system will provide a sent email to the user, when they sign up for a free account.
Input	User has initiated the sign-up process by clicking on the "Sign-up" button.
Output	User successfully receives an email notification and confirms their email address.
Exceptions	If the email address is invalid or already associated with an existing account, the system displays an error message to the user asking to provide a valid email address or else log in if the email address is already registered.

Other Use Cases System Operation Contracts are as follows:

For Search Flashcards:

Contract	When a user clicks on the search bar, and tries to search any particular flashcard, the system must retrieve and display all relevant flashcards that satisfies the search query.

Input	Click on search bar and write query to search
Output	Particular flashcards retrieved and displayed.
Exceptions	If the flashcard is not available or there are no relevant flashcards available (i.e. no flashcard exists or uploaded as indicated by search), then the system will provide a proper exception or error prompt.

Retrieve Flashcard Data:

Contract	The system will retrieve flashcard data stored in the application's database to study a Computer Science concept or language.
Input	Click on search bar and write any flashcard to search for
Output	Particular Flashcards retrieved and displayed.
Exceptions	If the flashcard is not available or there are no relevant flashcards available (like no flashcard exists or uploaded as per searching) then the system will provide a proper exception or error prompt.

Description: The system will retrieve flashcard data stored in the application's database to study a Computer Science concept or language.
Requirements: REQ-3, USREQ-6

→ Contact Support

Description: The system will provide an email address for users to contact with questions or concerns.
Requirements: REQ-5

→ Contact Support

Contract	The system will retrieve flashcard data stored in the application's database to study a Computer Science concept or language.
Input	Click on Search bar and write any flashcard to search for
Output	Particular Flashcards retrieved and displayed.
Exceptions	If the flashcard is not available or there are no relevant flashcards available (like no flashcard exists or uploaded as per searching) then the system will provide a proper exception or error prompt.

Retrieve Flashcard Data
- Contract: When a user requests to study a Computer Science concept or language, the system shall retrieve relevant flashcard data from the application’s database.
- Inputs: Requested concept or language
- Outputs: Flashcard data
- Exceptions: If the requested data is not available, the system shall provide an appropriate error message.

5. **Contact Support**
- Contract: The system shall provide users with an email address to contact for questions or concerns.
- Inputs: User's request to contact support
- Outputs: Support email address
- Exceptions: If the support email address is unavailable, the system shall provide an alternative method for contacting support.

The following Use Case Operation Contracts we not able to be implemented by the time of the demonstration. Adding these use cases will help users to interact with the web application more frequently and with other users.

6. **Track Progress Alert**
- Contract: The system shall alert the user when they reach a certain number of flashcards studied or time spent studying on the application.
- Inputs: User's study progress
- Outputs: Progress alert
- Exceptions: If the progress alert feature is disabled or unavailable, the system shall not provide alerts.

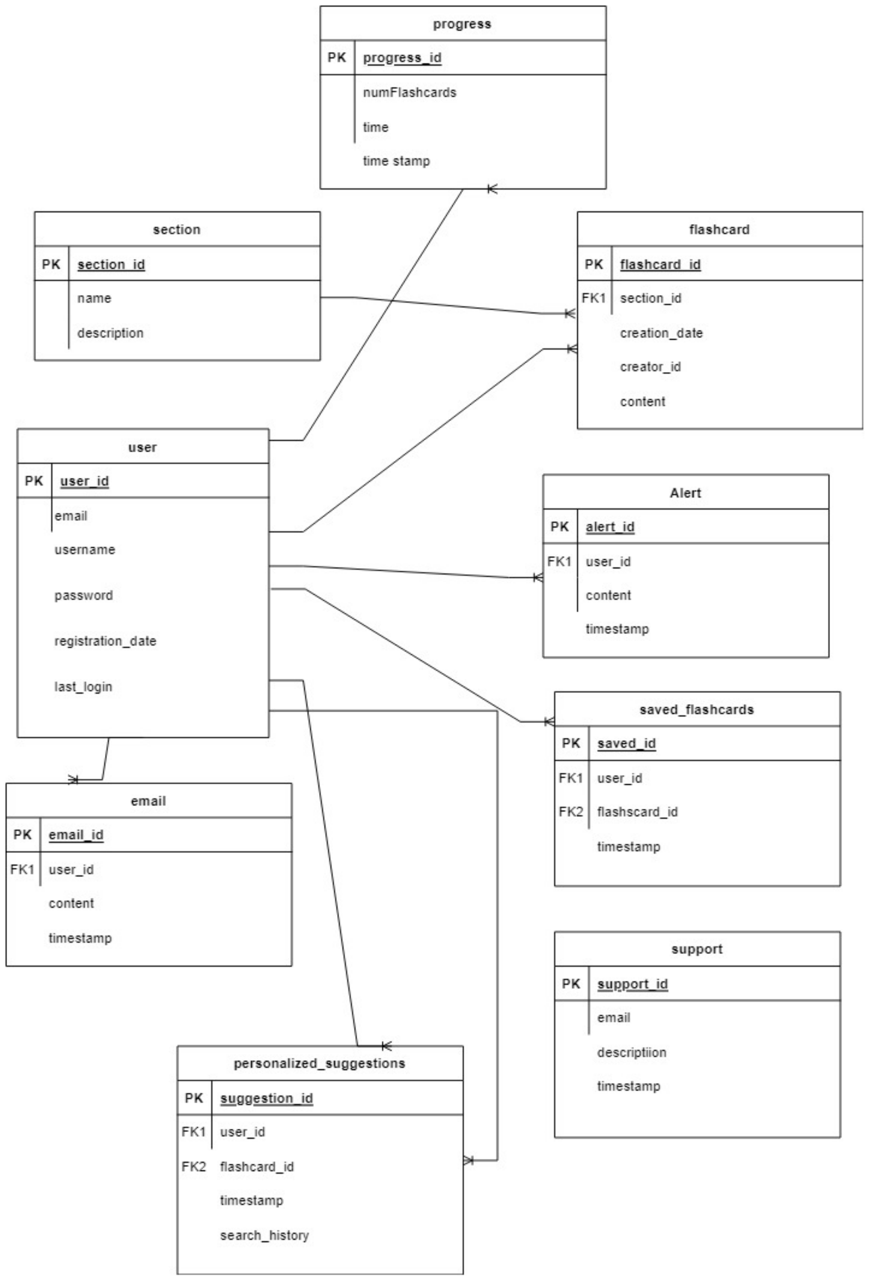
7. **Personalized Suggestions**
- Contract: Based on a user's previous searches, the system should suggest relevant flashcards or activities.
- Inputs: User's previous searches
- Outputs: Personalized suggestions
- Exceptions: If the user's search history is insufficient to provide suggestions, the system shall not provide any suggestions.

8. **Save Flashcards**
- Contract: The system shall provide users with a section to save flashcards for future use in a cataloged group.
- Inputs: Flashcards to be saved

- Outputs: Confirmation of successful save
- Exceptions: If the save operation fails, the system shall provide an appropriate error message.

These contracts outline the expected behavior of the system for each use case, including inputs, outputs, and exceptions. They serve as guidelines for the development and operation of the flashcard application.

c. Data Model and Persistent Data Storage



Our flashcard application needs to save data that outlives single execution of the system. The application program will rely on a relational database management system. The database will correctly store user's login information, flashcard sets, alerts, email information, and data based on how long and how much they have been studying. The data will be stored in a NoSQL database. It is imperative that user data is stored and tracked because in future features, the user's progress will be updated in real time. The relationships between entities are established through foreign key constraints.

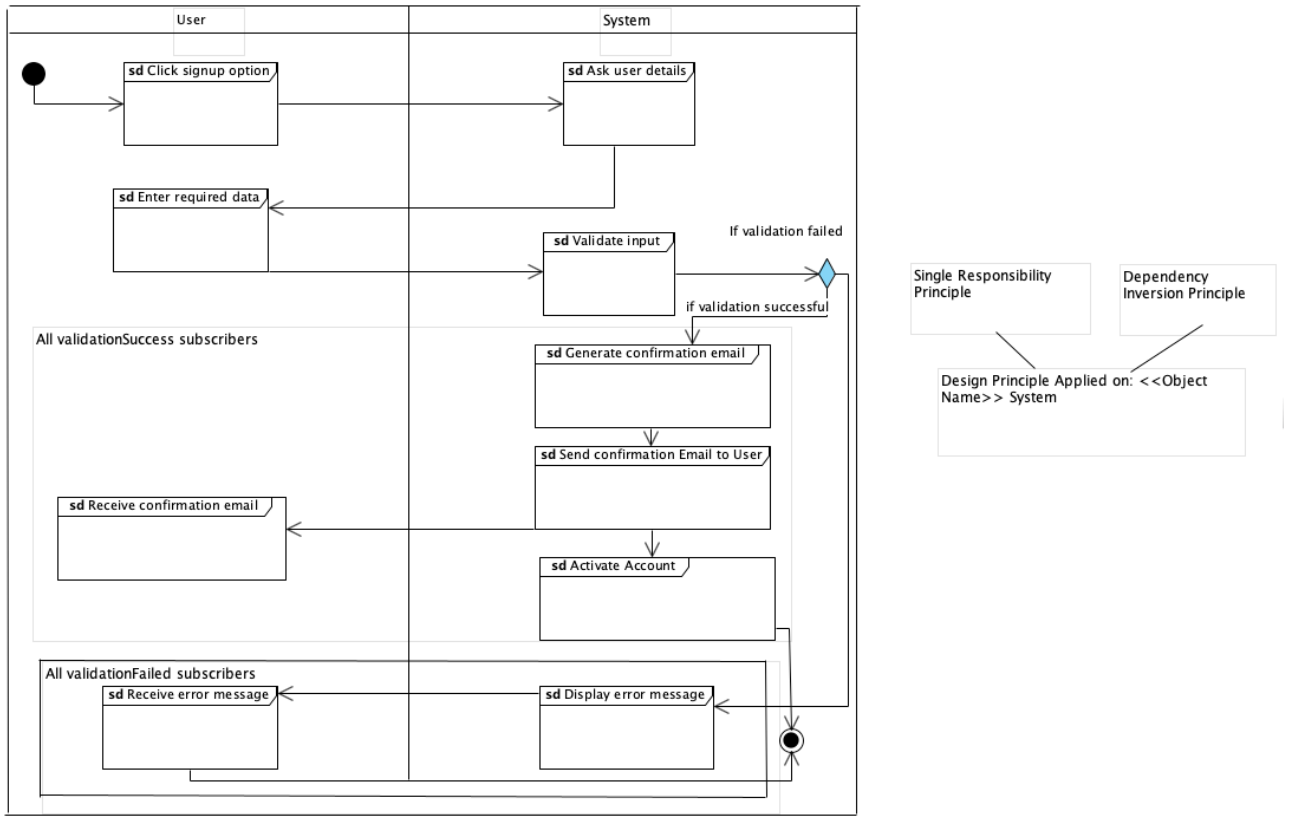
Our data model shows current implementation as well as plans for future implementation.

7. Interaction Diagrams

a. Interaction Diagram

The interaction diagram particularly activities communication diagram of fully-dressed use cases are as follows:

For Sign-up



User
System
sd Click signup option
sd Ask user details
sd Enter required data
sd Validate input
If validation failed
if validation successful
All validationSuccess subscribers
sd Generate confirmation email
sd Send confirmation Email to User
sd Receive confirmation email
sd Activate Account
All validationFailed subscribers
sd Receive error message
sd Display error message
Single Responsibility Principle
Dependency Inversion Principle
Design Principle Applied on: <<Object Name>> System

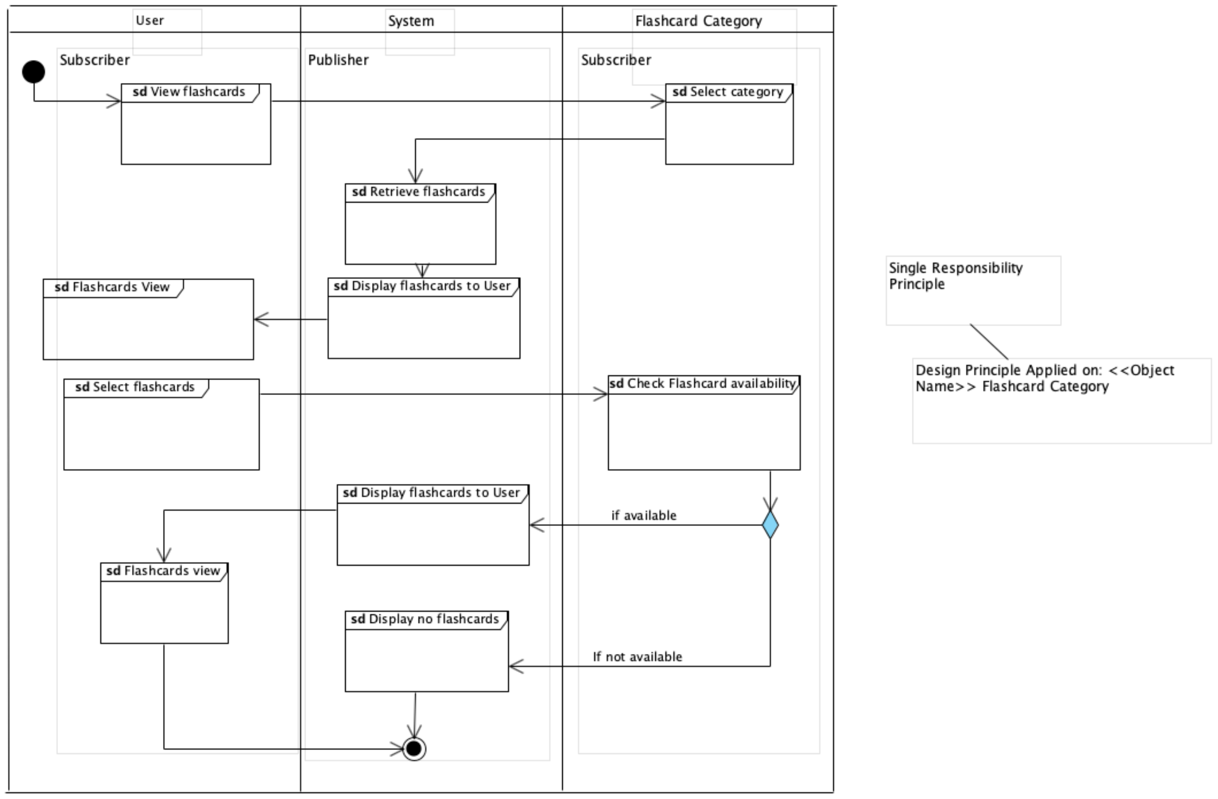
Objects and Design Principles

The objects in the above diagram includes User and System.
The design principles that can be applied are described as following:

- **Single responsibility Principle:** This principle is applicable here for the separation of concerns. In System object, particularly the Email module, handles email-related tasks, while the Account module manages user accounts, adhering to the SRP.

- **Dependency Inversion Principle:** This principle is applicable here because System object depends on abstractions (Email module, Account module) rather than concrete implementations. This shows flexibility and modifiability.

For View Flashcards



Objects and Design Principles

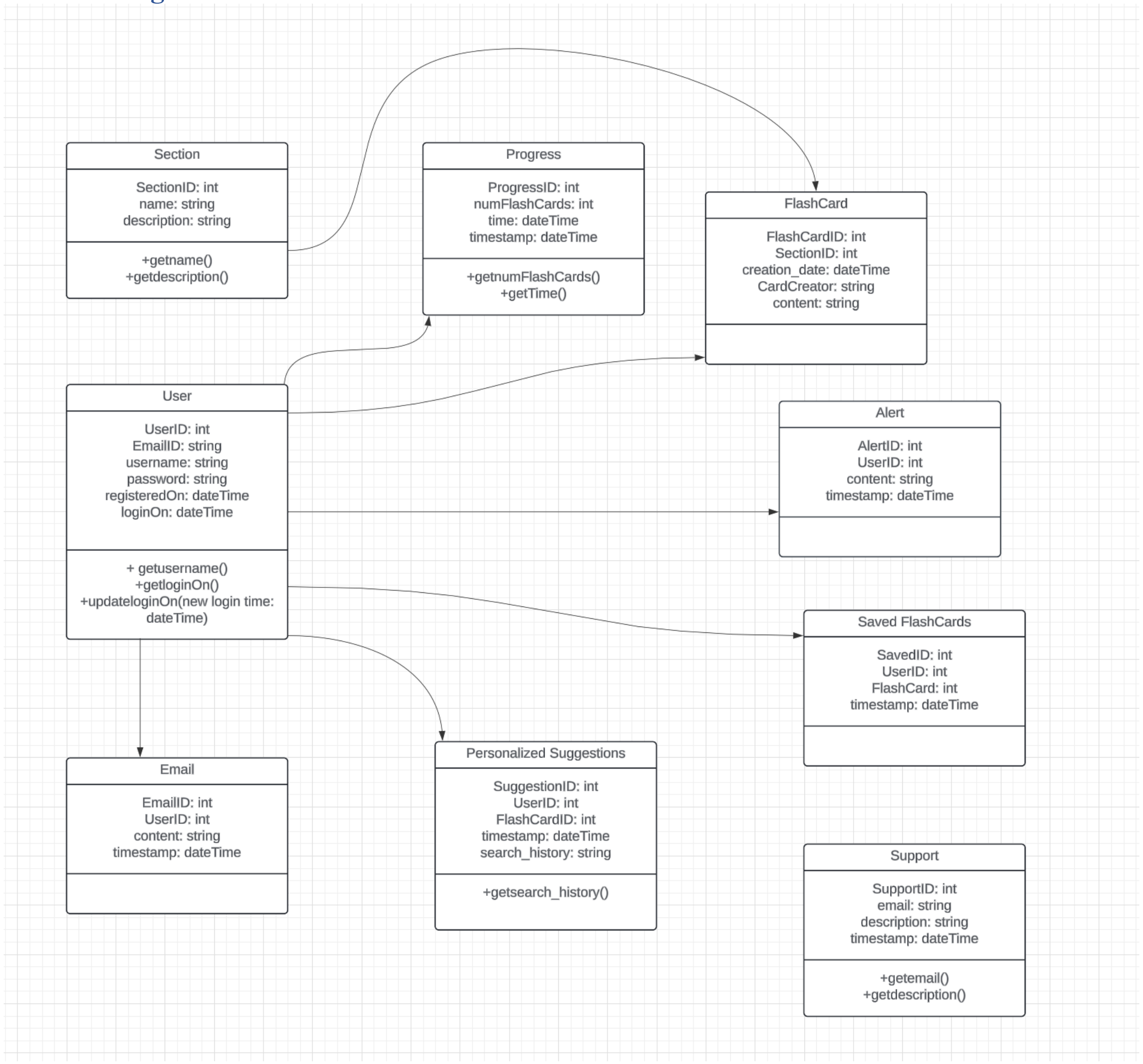
The objects in the above diagram includes User, System and Flashcard Category.

The design principle that can be applied are described as following:

- **Single responsibility Principle:** This principle is applicable here because the Flashcard category object here is responsible solely for managing flashcards which supports separation of concerns.

8. Class Diagram and Interface Specification

a. Class Diagram



b. Data Types and Operation Signatures

Class:Section - represents section or category within the study material

Attributes:
SectonID:int -unique identifier for section
name:string - name of the section (i.e. Data Structures)
Description:string - description of the section being studied

Operations:
+getname():string - retrieve name of section
+getdescription():string - retrieves description of the section

Class:User - represents user of the system

Attributes:
UserID:int - unique identifier for the user
EmailID:string - email address of the user
Username:string - username chosen by the user
Password:string - password chosen by user
registeredOn:dateTime - date and time when user registered account
loginOn:dateTime - date and time of the user's last login

Operations:
+getLogin():string - get valid username and password for login
+getLoginOn():dateTime - retrieves the date and time of user's last login
+updateloginOn(new login time) datetime:void - update date and time of user's last login in realtime

Description:

Class:Email - represents email message

Attributes:
EmailID:int - unique identifier for the email
UserID:int - identifier of the user who sent or received the email
Content:string - text content of the email
timestamp:dateTime - date and time when email was sent or received

Class:Progress - represents the progress made by a user studying

Attributes:
ProgressID:int - unique identifier for the progress entry
numFlashcards:int - number of flashcards reviewed by user
totalStudytime:dateTime - total time spent studying
timestamp:dateTime - date and time of the progress entry

Operations:
+getnumFlashcards():int - retrieve the number of flashcards reviewed
+getTimeStudyTime():dateTime - retrieve total time spent studying

Class:Flashcard - represents flashcard that contains study material

Attributes:
FlashcardID:int - unique identifier for the flashcard
SectionID:int - identifier of the section to which the flashcard belongs
creation_date:dateTime - date and time when flashcard was created by user
CardCreator:string - creator of the flashcard
Content:string - text content of flashcard(front and backside of flashcard)

Class:Alert - represents alert notification for a user

Attributes:
AlertID:int - unique identifier for the alert
UserID:int - identifier of the user who received the alert
Content:string - text content of the alert
timestamp:dateTime - date and time when alert was generated

Class:Saved Flashcards - represents flashcard saved by users

Attributes:
SavedID:int - unique identifier for the saved flashcard entry
UserID:int - identifier of the user who saved the flashcard
Flashcard:int - the flashcard itself
timestamp:dateTime - date and time when the flashcard was saved

Class:Support - Represents a support request from a user

Attributes:
SupportID:int - unique identifier for the support request
Email:string - email address of the user requesting support
Description:string - description of the support request
timestamp:dateTime - date and time when the support request was made

Operations:
+getemail():string - retrieves email address of the user
+getdescription():string - retrieves description written from user for support request

Class:Personalized Suggestions - Represents personalized study suggestions for users based on their search history

Attributes:
SuggestionsID:int - unique identifier for the suggestion
UserID:int - identifier of the user receiving the suggestion
FlashcardID:int - identifier of the suggested flashcard
timestamp:dateTime - date and time when the suggestion was made
Search_history:string - search history of the user

Operations:
+getsearch_history():string - retrieves search history of user and stores in memory

I. Traceability Matrix

Classes	Domain Concepts					
	WebApp	User Account	Flashcard Sets	Support	API	Notifications
Section	✓		✓		✓	
User	✓	✓				✓
Email						
Progress	✓					
Flashcard	✓		✓		✓	
Support	✓			✓		

User Account:
- User: Permits the user to create account and add personal details.

Flashcard Sets:
- Section: Allows the user to select a section of study
- Flashcard: Allows the user to view study material

Support:
- Support: Creates a request description for support and links user email to the request.

API:
- Section: Sends request for sections of study.

- Flashcard: Requests desired flashcards.

Notifications:
- User: Incorporates user info into alerts (i.e. name).
- Alert: Manages user alerts and notifications.

c. Design Patterns

Publisher-Subscriber Pattern: This design pattern is applicable for use in our web application. The System object would act as the Publisher and the User acts as the subscriber (*See section 7 Interaction Diagrams*). This can be implemented for scalability in the future to notify users when new improvements or notifications are released via the Email module.

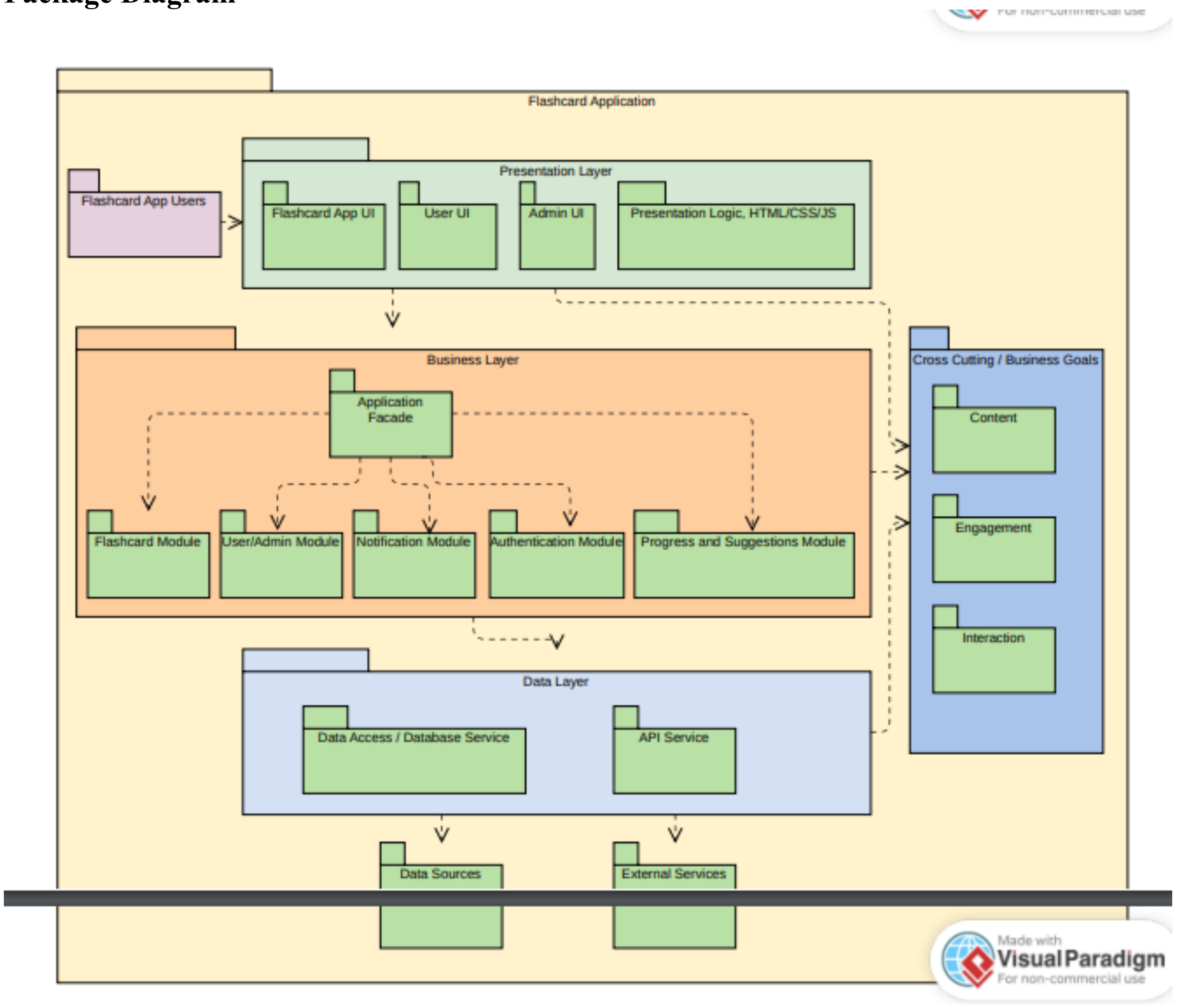
d. Object Constraint Language (OCL)

Section:
Context Section inv:
self.Section->get(name)->get(description->database()
User:
Context User inv:
self.User->get(Username)->get(Password)->database()
Email:
Context Email inv:
self.Email->get(Content)->get(timestamp)->get(User)->database()
Flashcard:
Context Flashcard inv:
self.Flashcard->get(Content)->database->post(Flashcard)
Support:
Context Support inv:
self.Support->get(Email)->get(Description)->get(timestamp)->database()

9. System Architecture and System Design

a. Identifying Subsystems

Package Diagram



Package Diagram Description:

The above presented package diagram illustrates the architecture of our system. This system is organized into three layers: Presentation Layer, Application Layer, and Data Access Layer. The purpose of portraying subsystems in the form of a package diagram is to display that we have selected a layered architecture style for our project.

All subsystems within each layer are inclined to the core business values of our project. Overall, this package diagram provides a comprehensive overview of the flashcard system covering its individual subsystems.

b. Architecture Styles

Our flashcard applications will utilize and follow a Layer Architecture style. This architecture style allots benefits via its modularity and flexibility. By using a Layer Architecture, we can easily distribute our flashcard application into various layers and manage them seamlessly. This also allows us to make modifications to one layer of the system without impacting the others. A Layer Architecture structure is also advantageous when using HTML, CSS and JS: the three main markup-up and scripting languages used in building the applications and its features. The description of each layer is as follows:

- In the first layer, Presentation Layer, the various user interface components will be presented, including Flashcard UI for display or manage flashcard module, User UI for managing user particularly here student profiles and settings, and Administrative UI for managing administrator profiles and settings and administrative tasks. These components involve presentation logic and HTML, CSS as well as JS has been used to define presentation logic. The main task of this layer is to handle user interactions that come from our flashcard application users and provide a seamless user experience.
- Then the middle layer, the Application Layer, encapsulates the core functionalities or our flashcard application logic. It is divided into divided into several modules. The Flashcard Module handle the creation, fetching, manipulation and deletion of flashcards, while the User Module manages user-related functionalities such as registration, login, and profile management. The Authentication Module ensures secure access to the system, Notification Module manages notification services such as email alerts and user notifications, and the progress and suggestion module deals with the feature of progress tracking, sending progress alerts and showing personalized suggestions based upon interest.
- The Data Access Layer interacts with external data sources or services to retrieve or store data. It includes the API Service the Database Service which basically interacts with the database to perform CRUD operations on our flashcard app persistent data.

c. Mapping Subsystems to Hardware

The server for the flash card application will run on a single server computer that houses its own database for user info, the database API which fetches and stores information needed for the user and the flash cards, and handles all requests made from the users' browsers while utilizing the client-side application to the API. The API will communicate with two different servers in relation to the application. The server will be utilized using Apache. Users will be able to access the application from many different devices, such as desktops, mobile phones, or tablets.

d. Connectors and Network Protocols

These will be the forms of communication that will be used with the application:

HTTP/HTTPS: This form of communication will be used between the user via client-side application and the database API. This allows information requested or sent to the API, to reach the server in a quick and protected manner.

MongoDB Atlas:
Our application will utilize the NoSQL database known as MongoDB Atlas. All information created on the application such as user information (login, password, username) and flash card data will be implemented into this database. The database will then be accessed using Fetch Requests which will retrieve and post the flashcard information on the HTML page.

Peer to Peer: This form of communication will be used between users who wish to share flash cards with each other, study together, or initiate a quiz to collaborate on. This connection is protected and allows the users to keep their interactions private.

e. Global Control Flow

Execution orderliness: The system should follow an event-driven architecture rather than linear. User interaction is essential in this application. Different user actions such as clicking to reveal an answer, navigating between multiple sets of flashcards, or even interacting with the system to contact support will trigger corresponding actions within the application that are dependent on individual user behaviors. Using an event-driven approach allows for greater responsiveness and flexibility to provide users with a better experience.

Time Dependency: Since the flashcard application tracks user progress, there would be a need to rely on timestamps to display metrics and show visual representation of a user's achievements. Other time dependencies can also extend to updates regarding flashcard content and timestamps users save a card or set.

f. Hardware Requirements

Optimal performance and functionality of the flashcard application are reliant on the following hardware:

- **Screen Display:** A screen display with resolution capable of displaying text and images. Examples: computer monitor, mobile device screen, etc. Specific display requirements may vary depending on device type and size.
- **Disk Storage:** The database subsystem and server will require disk storage to store user data, application files, backups, and logs to ensure performance.
- **Communication Network:** A stable internet connection for establishing and accessing online features. A minimum bandwidth of 56 Kbps will be needed for proficient data exchange.
- **Database Server:** A dedicated database server for managing user accounts, flashcard data, and application information efficiently to ensure scalability upon growth.
- **Security Measures:** Security measures such as HTTPS encryption, data encryption, and user authentication to keep all user information private and safe. This should prevent unauthorized users from accessing sensitive ser information.

10. Algorithms and Data Structures

a. Data Structures

We will not be using any complex algorithms. The first data structure to be implemented is an array. The array will store the flashcards that the user populates. Each element in the array will represent a collection of flashcards/study sets. Most user data and flashcard definitions will be accessed and stored inside our relational database. Using an array data structure provides the application with simplicity and efficiency when accessing and organizing the flashcards.

Another data structure implemented will be a hash table. Hash tables will be used to store flashcard data in the application database. This will be efficient for the user's retrieval of specific flashcard sections/categories. The hash table data structure was chosen because of its efficiency for retrieval. Using hashing will provide faster lookup times for cards, especially when there are many categories and sections.

When the user wants to load up a study session, a queue data structure will be used. The queue will be populated with IDs or indexes of all the flashcards the user will want to study. During the session, it will dequeue from the queue one by one. This will present a flashcard list in sequential order. In addition to the queue data structure, the user may want to review their flashcards in reverse order, that is where a stack will be implemented. The queue and stack data structures are implemented because of their simplicity and efficiency for the user.

b. Concurrency

Concurrency is often a factor when designing software. The front end of the application will utilize Asynchronous JavaScript to handle multiple tasks at once such as fetching data, performing animations, or executing other tasks without the risk of blocking the main thread. The back end of the application will utilize concurrency control features within Node.js to manage requests and connections or preventing resource exhaustion during high loads of traffic or requests.

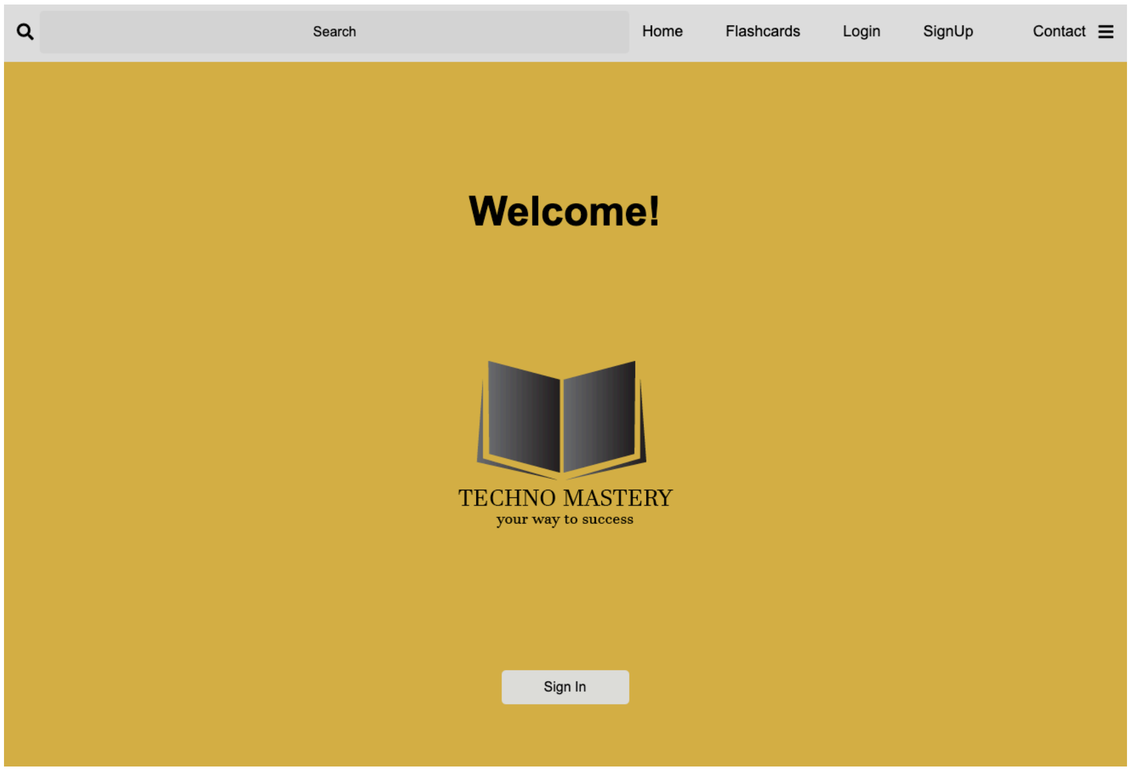
11. User Interface Design and Implementation

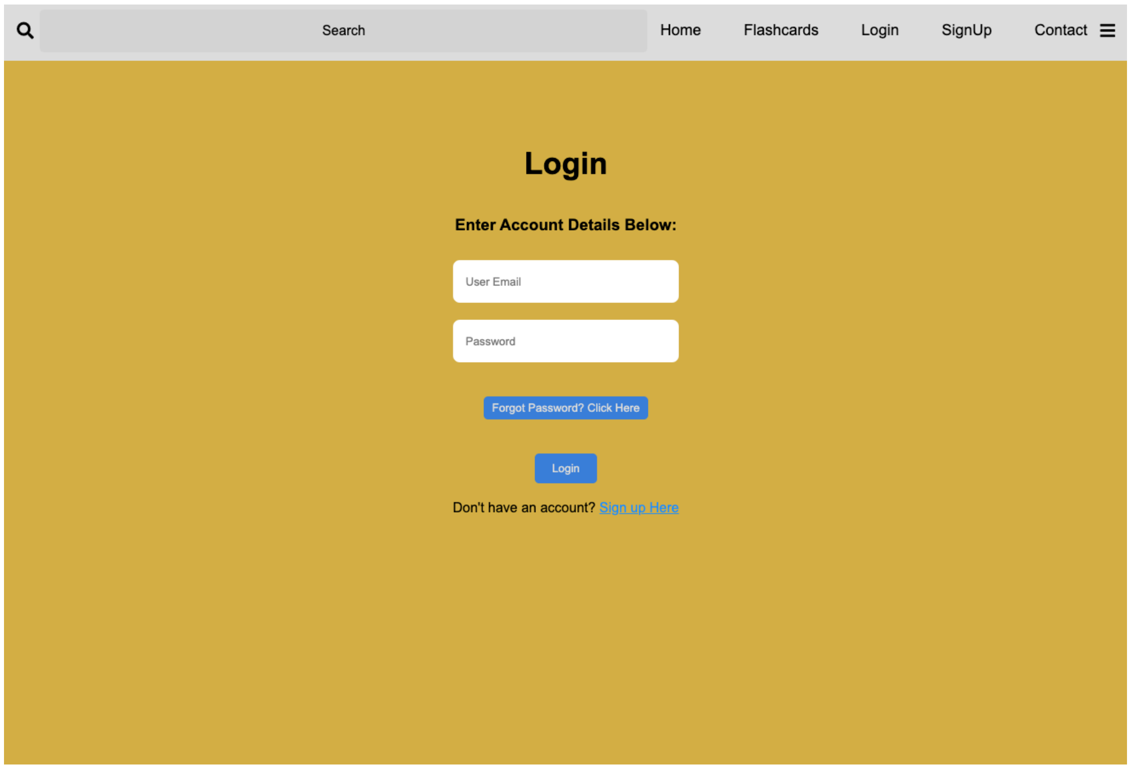
Design: A few changes were made to the initial application design throughout the implementation process. A few key changes were:

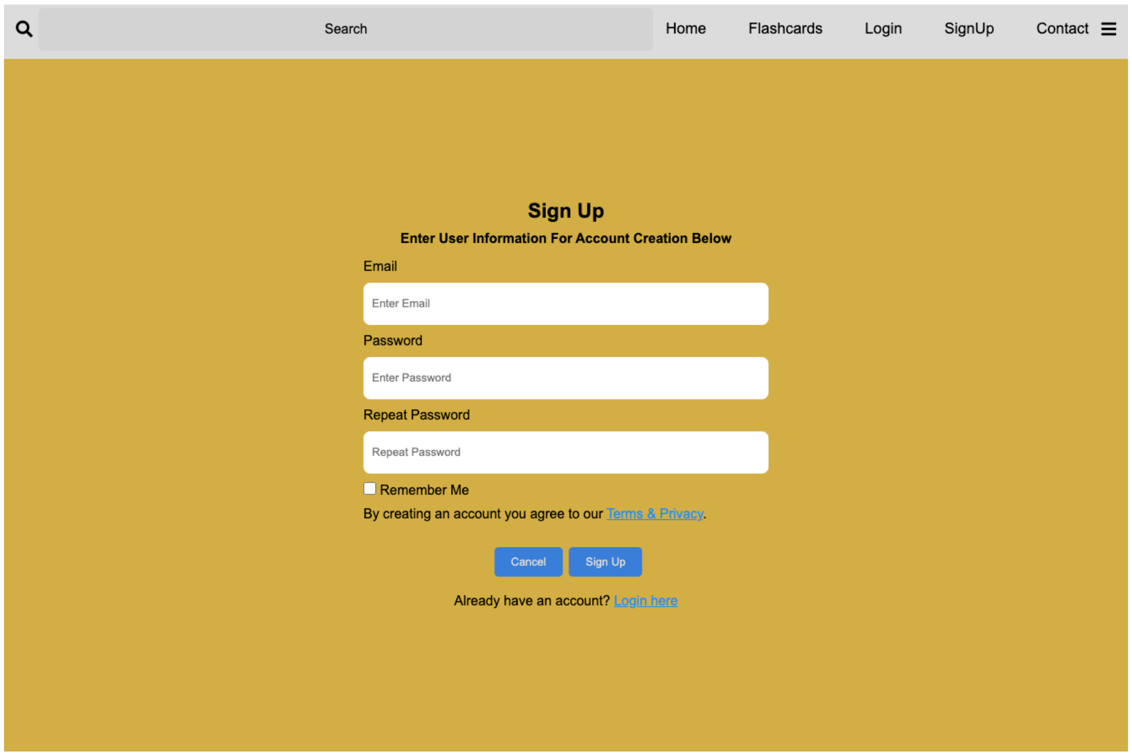
- The Home Page was changed to now show a logo and a title for our application
- The Flash Card Page was altered to show more categories, as well as more definitions and answers pertaining to those categories.
- A functional sign up and login page have been implemented to allow users to sign up for an account and login to their accounts if they have one created already
- The Contact Page allows users to send messages to the Support team, which prompts a notification that the message was sent

The use case for our application has been altered to better showcase a more functional application. The application has been around 90% realized, with the most important functions, such as the flash cards, login, signup, and contact pages, being fully implemented and functional.
Use Case #1:

- User accesses application and clicks on sign in button through the home page to enter login information. If the user does not have a login, they can either click on the Sign Up navigation tab to create an account or they can click on Sign Up Here link on the bottom of the login page.
- Once the user enters login information, they are now logged in and can proceed to access the different pages within the application. This can be done by clicking on the different navigational buttons on the top of the page.


Search
Home
Flashcards
Login
SignUp
Contact
Welcome!
TECHNO MASTERY
your way to success
Sign In


Search
Home
Flashcards
Login
SignUp
Contact
Login
Enter Account Details Below:
User Email
Password
Forgot Password? Click Here
Login
Don't have an account? Sign up Here


Search
Home
Flashcards
Login
SignUp
Contact

Sign Up

Enter User Information For Account Creation Below

Email

Enter Email

Password

Enter Password

Repeat Password

Repeat Password

Remember Me

By creating an account you agree to our Terms & Privacy.

Cancel Sign Up

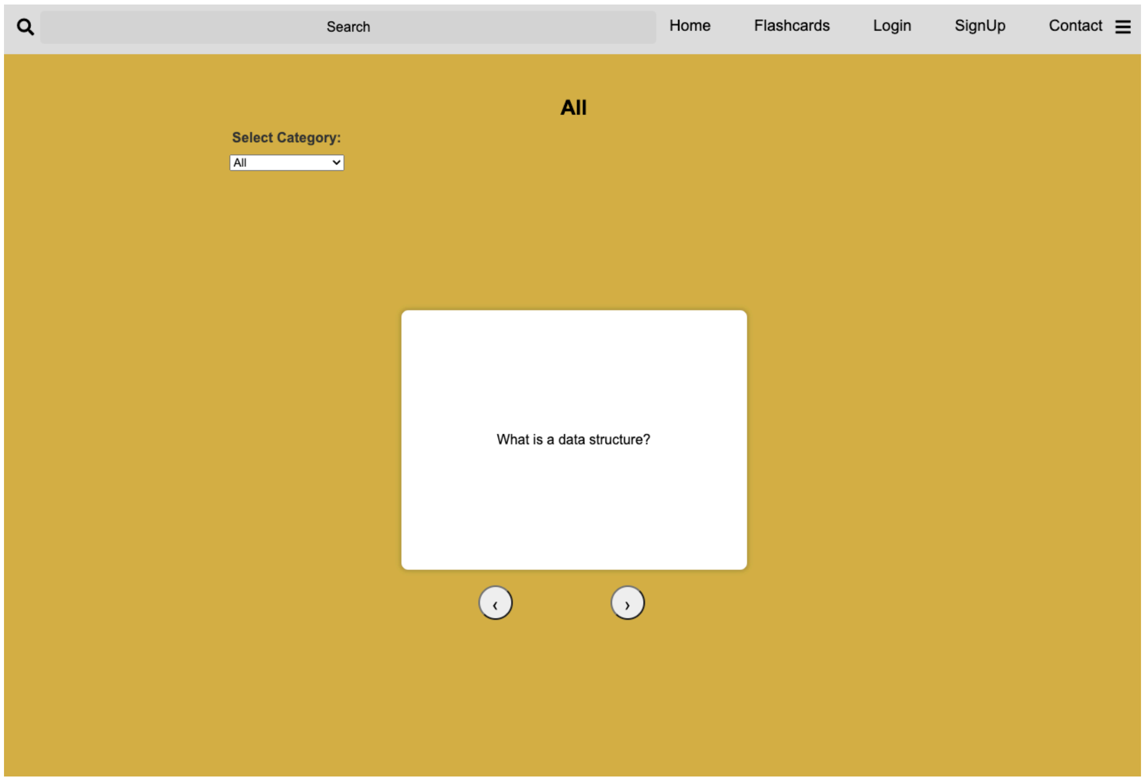
Already have an account? Login here

Use Case # 2:

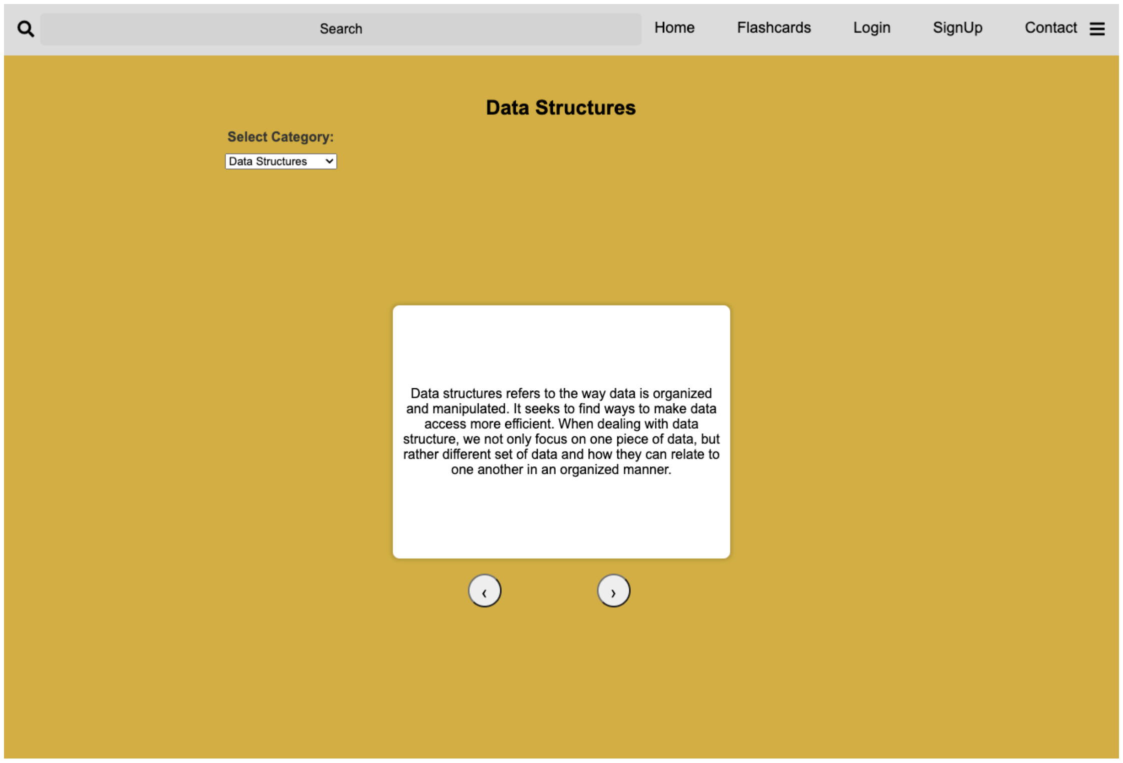
- The user can click on the different navigational buttons, and it will take them to their respective pages

Use Case #3

- After clicking on the flash card navigation tab, the user can access different categories of flash cards through a drop-down box.
- They can then cycle through each flash card and see its definition and answer by clicking on them.



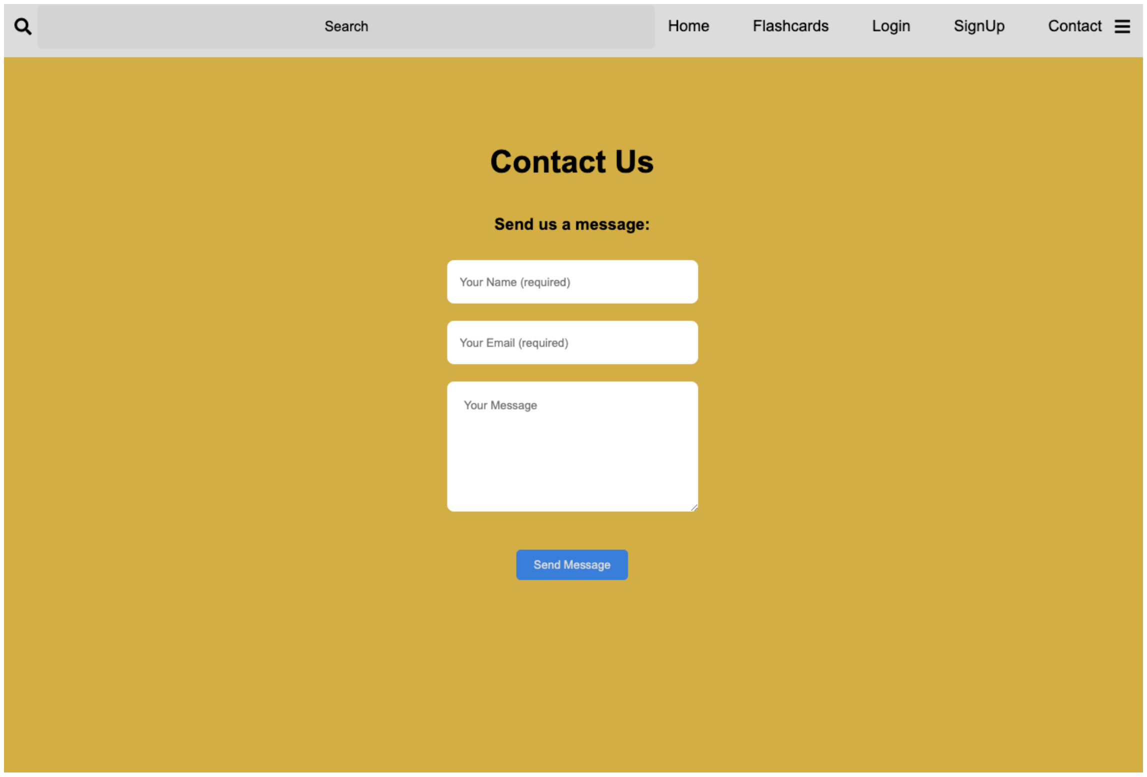
What is a data structure?

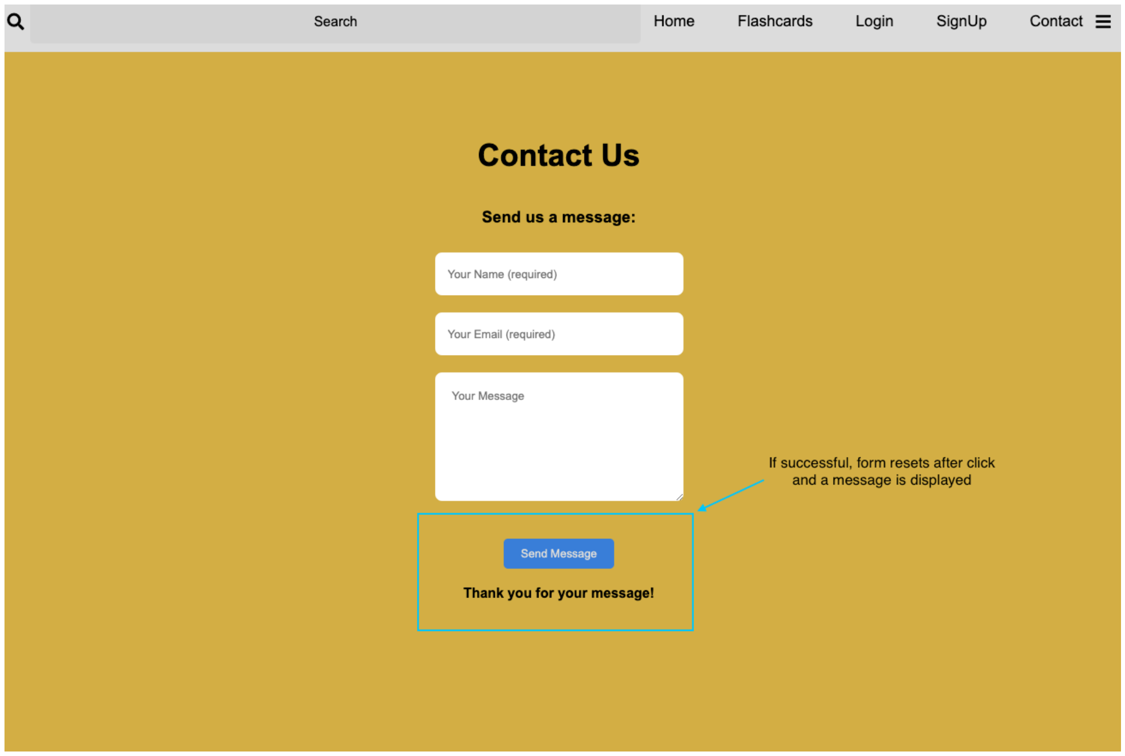


Data structures refers to the way data is organized and manipulated. It seeks to find ways to make data access more efficient. When dealing with data structure, we not only focus on one piece of data, but rather different set of data and how they can relate to one another in an organized manner.

Use Case #4:

- User can send an Email to the Support Team by clicking on the Contact navigation tab on the top of the screen.
- The user will then enter their information, along with the message they are trying to send. Afterwards the user will click the Send Message button on the bottom of the page.
- A notification will appear stating that the email was sent.





Contact Us

Send us a message:

Thank you for your message!

If successful, form resets after click and a message is displayed

12. Design of Tests

a. Test Cases:

To ensure that our application is running as expected, the following test cases will be designed and implemented for testing:

- Initial Application Home Page Functionality: Ensure that once the application is searched for and engaged with, the application is functioning and that interactive UI components are present.
- Compatibility Testing/Functionality: Test that compatibility settings are initialized to allow application usage on multiple devices, which testing will be done by use of desktop and mobile devices.
- Login/Account Creation Functionality: Interact with the login/account creation button and verify that the dialogue box to login or create an account appears and works.
- Navigational Link Functionality: Interact with navigational links and verify that they travel to respective linked pages.
- Flash Card Functionality: Interact with a set of flash cards and verify that flash cards are functioning and show respective data.

b. Test Coverage:

The test cases provided will utilize roughly 60 to 70% of the applications programmed code. Certain linked pages, UI components, and data sources may not be present during specific testing cycles, thus the percentage provided being used. The test cases will have enough finished code to ensure that the test cases can be utilized and show that the application is in a functional state. The percentage of code used will also ensure that a quality portion of the application can be tested without many errors occurring.

c. Solution:

To perform integration testing of our Flashcard application, the following strategy will be implemented:

- We will first ensure that unit testing has been performed completely.
- Then, we will identify the points where integration takes place between modules (i.e. interaction between the User module with flashcard module and modification module).

- We will follow a proper test plan document in which for each use case all possible test cases will be defined that will basically verify the working of integration points. For instance, here we may test the feature of user registration and test if it will trigger the feature of appropriate email generation that is handled by the notification module.
- We will determine a test date to enter against each test case to be used.
- Finally, we will execute our integration tests to assure that our functional requirements, non-functional requirements, and user interface requirements are properly working together.
- After making changes or updates to modules, wherever required, we will perform regression testing to ensure that all existing integration points remain unaffected.
- We will document test results including all the risks identified and issues encountered and then track them for future testing and monitoring. This would help us out in identifying and addressing integration issues efficiently.

Additionally, the testing plans for non-functional requirements, and user interface requirements are highlighted as follows:

➔ **For Non-functional Requirements Testing:** In this we will perform different types of testing which satisfies the non-functional requirements stated implementation. Following testing here we will perform following steps:

  - We will assess our Flashcard application ability to handle multiple levels of user traffic (NFREQ-3) by   - performance testing. In this we will give varying loads and measure response times. Such as we will give a load of more than 100 users using our application view flashcard or sign-up notification and check in how much time the system is taking to respond.
  - We will assess whether our Flashcard application runs efficiently on different devices (NFREQ-5) by performing compatibility testing and is compatible with different operating systems and browsers.
  - We will verify that user-provided information remains private (NFREQ-2) by conducting security testing in which we perform privacy assessments and penetration testing.

➔ **For User Interface Testing:** In this we will conduct usability testing to ensure that the user interface (UI) basically meets requirements or not such as having a search bar (US REQ-5) and providing intuitive navigation (US REQ-4). Apart from that, we will verify that UI elements like flashcards (US REQ-6) fetch out and display correctly and interact as expected, helping in making sure a seamless user experience.

d. Conclusion:

By following these outlined testing strategies and plans, we will try our level best to ensure the correct working of functionalities upon integration, reliability, security, compatibility, performance, and usability of our flashcard application while meeting the stated requirements and objectives in our report documentation.

13. Project Management

a. Merging the Contributions from Individual Team Members

The team leader, Megan Hall, formatted the draft of the finished final report with the assistance of Steven Smith. Tasks and deliverables were divided equally amongst the team at weekly group meetings. After everyone completed their sections, Megan would review and edit materials for cohesion.

b. Project Coordination and Progress Report

Our current project functions with the top priority use cases as outlined in previous sections. The working web application is capable successfully fetching data from our database to view desired flashcard categories, terms, definitions, and answers. Likewise, Users are able to create a unique account, login with an existing account, and contact support from the current user interface.

c. History of Work

History of Work and Current Work

January 21 – January 28
After forming our team, we created a discord channel to communicate project ideas and begin our project proposal. After our first voice channel meeting, we all agreed to produce an education flashcard web application for the subject of computer science. Through Discord and the file exchange located on our university's Blackboard application, we were able to draft our project proposal.

January 28 – February 4
Utilizing the feedback presented from our proposal plan, we began to find solutions and further Our application designs. During this time, we focused on the reasons why a customer would use our application and the problems that our application would remedy. We also took this time to identify business goals, functional requirements, nonfunctional requirements, and user interface requirements. During this time Joe made the mockup of our basic user interface design.

February 4 -February 11
During this time, the team took the opportunity to compile the use cases and begin looking at the functional specification of the application in more detail. Ahmed created the use case diagram to help us visualize these requirements in depth.

February 11 – February 18
The team plotted the system architecture and design that will be required for our application. We identified which architecture style our application would use, which subsystems would be required, which connectors and network protocols would be utilized, etc. Once we identified these concepts, we were able to estimate the project size.

February 18 – February 25
The team used this time to create our conceptual models, identify the system operation contracts, and create a data model. Steven and Megan worked together to create the concept definitions, association definitions, and attribute definitions. They then visually rendered these concepts in a domain model diagram.

February 25 – March 3
During this week, the classes that would need to be utilized within the application were identified and described. In doing so, the team was able to render class diagram, identify data types and operation signatures, and compile an interaction diagram.

March 3 – March 10
The team identified any complex algorithms or data structures that would be utilized, any notable user interface design changes, designed the test cases that will be utilized and compiled our plan of work for implementation.

March 10-March 17
Team began implementing the different components of the application. Weekly meeting conducted to determine what approach to take and list of responsibilities for each team member.

March 17- March 24
Team continued implementation process to ensure that application was functional enough in preparation for first live demonstration. Weekly meeting conducted to tackle remaining issues that may have presented themselves.

March 24-March 31
Final preparations for the first demonstration, which entailed tidying up any code and ensuring that the main components were functional. Weekly meeting conducted to discuss remaining work needing done and deciding who was presenting which parts of the application as well as which parts of the PowerPoint presentation.
March 31-April 7: Last team meeting before Demo # 1. Team decided who was presenting which parts and the PowerPoint presentation was finalized. April 3rd was our first demonstration.

April 7-April 14
Weekly meeting conducted to go over how to make our application more functional, after feedback from first demonstration. Further implementation was done on the application. Database was created and set up to allow flash card data to be entered in correctly onto the application.

April 14-April 21
Weekly meeting conducted to go over final preparations for our final demonstration in a couple of weeks. Login and Sign-up Pages of our application were fixed and made functional, in correlation with the database.

April 21-April 28
Weekly meeting conducted to go over finalizing our revisions for final project report. Finishing touches on the backend side of our application were made. Full functionality of all major components of our application is present.

April 28-May 5
Weekly meeting conducted to go over preparations for final demonstration and deciding who will be presenting which sections.

May 5-Future
Weekly meeting conducted, and final projects reports and self-reflection essays created for our project.

Key Accomplishments:
d. Project Team created
e. Excellent collaboration among team members in project creation
f. Weekly progress that showed our teams improvement
g. Functional application created
h. Application created that can be considered beneficial in today's world
i. Database created, which showcases a more modern application
j. All deadlines reached with no issues
k. Key communication among team members, which makes for better progress
l. An application that our entire team can be proud of and did not seem possible at the start of the project, but we accomplished it

Future Work
Regarding the future work of our project, there are still a few minor components that have not been fully realized yet. This pertains to our personalized suggestion function, alert functions that

let you know how many flash cards you have studied, the saving of flash cards into a personalized storage area, and peer to peer sharing of saved flash cards. These components can be implemented later if decided, but as of now the application is mostly functional and can be used by anyone.

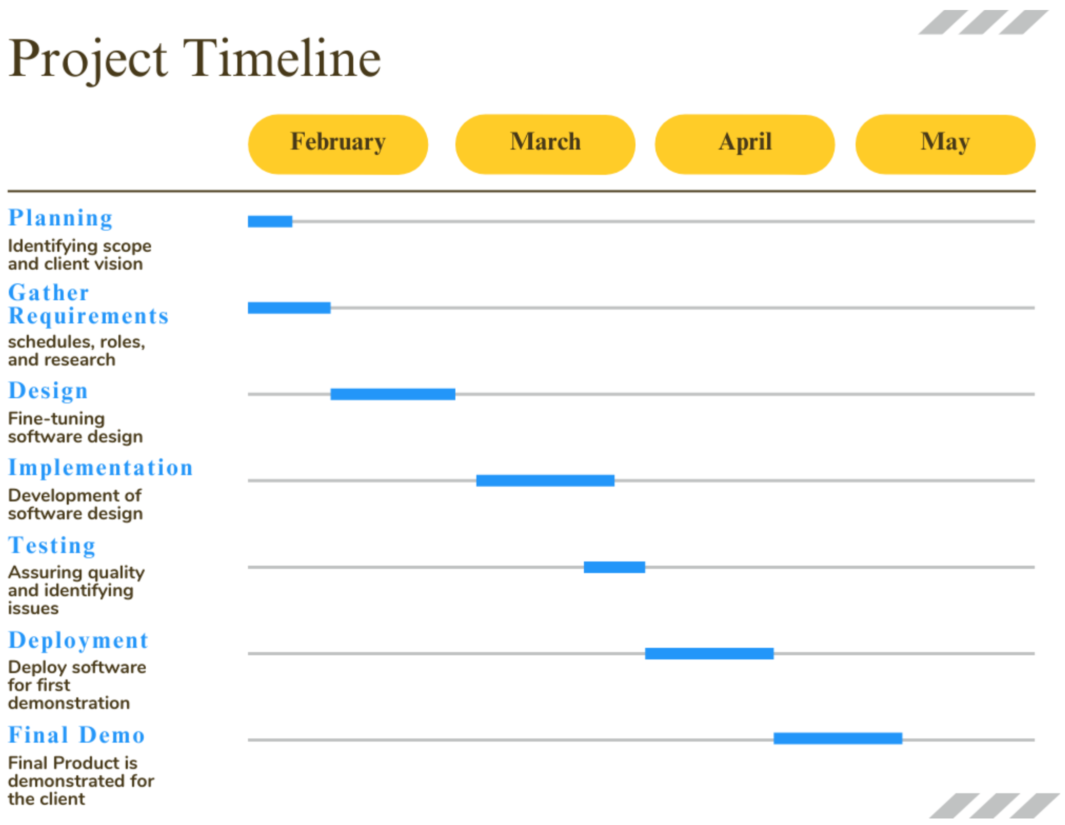
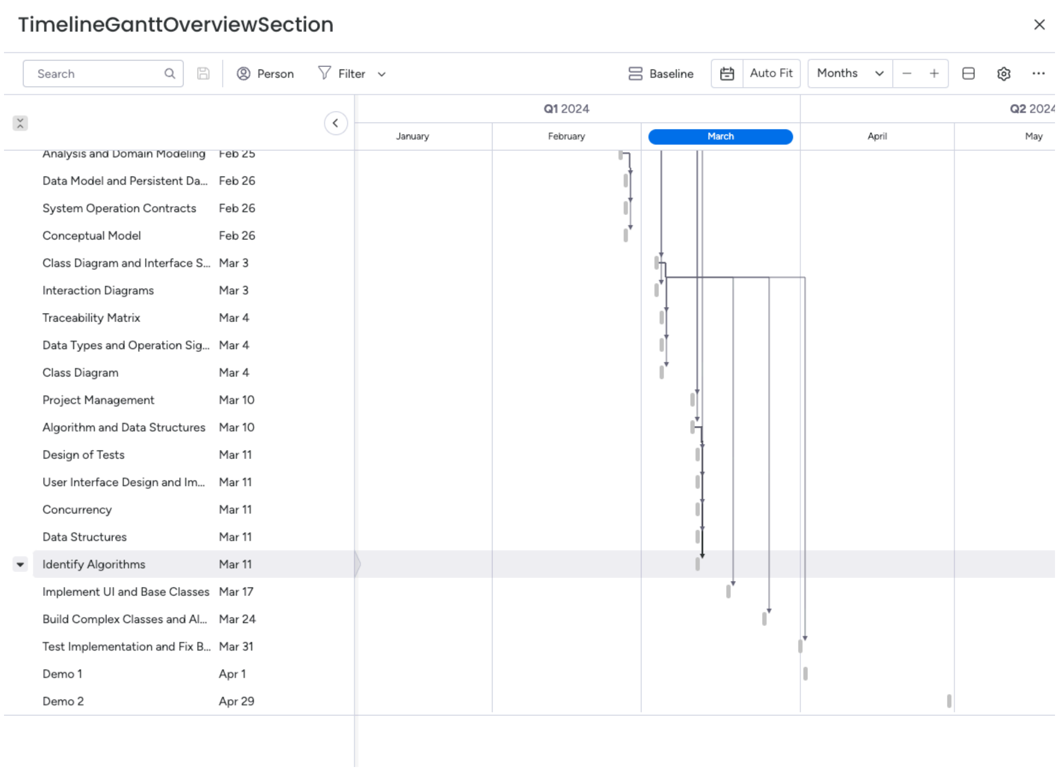


Figure 1.1 – *A timeline from February 2024 through May 2024 of deliverables and general tasks*



Analysis and Domain Modeling	Feb 25
Data Model and Persistent Da...	Feb 26
System Operation Contracts	Feb 26
Conceptual Model	Feb 26
Class Diagram and Interface S...	Mar 3
Interaction Diagrams	Mar 3
Traceability Matrix	Mar 4
Data Types and Operation Sig...	Mar 4
Class Diagram	Mar 4
Project Management	Mar 10
Algorithm and Data Structures	Mar 10
Design of Tests	Mar 11
User Interface Design and Im...	Mar 11
Concurrency	Mar 11
Data Structures	Mar 11
Identify Algorithms	Mar 11
Implement UI and Base Classes	Mar 17
Build Complex Classes and Al...	Mar 24
Test Implementation and Fix B...	Mar 31
Demo 1	Apr 1
Demo 2	Apr 29

Figure 1.3 – *A more detailed timeline from February 2024 through May 2024. You can access the timeline here on Monday.*

d. Breakdown of Responsibilities - Implementation

Class	Planned Team Member	Actual Team Member
Section	Joe	Megan, Ahmed, Joe, Steven
User	Ahmed	Ahmed, Joe
Email	Steven, Megan	Joe, Ahmed, Megan
Flashcard	Steven	Joe, Ahmed, Steven
Support	Joe	Megan, Joe, Ahmed, Steven

14. References

Dixon S. What Is a RACI Chart? Definition, Template, and Examples. Jul 5, 2023. https://www.wrike.com/blog/what-is-a-raci-chart/

Golding, J. M., Wasarhaley, N. E., & Fletcher, B. (2012). "The Use of Flashcards in an Introduction to Psychology Class." Teaching of Psychology, 39(3), 199-202, https://doi.org/10.1177/0098628312450436

Ideas For Software Engineering Team Project. Accessed 1Feb, 2024. https://www.fhsu.edu/cs/csci441/se/sampleprojects

Learning Tools, Flashcards, and Textbook Solutions | Quizlet, quizlet.com/. Accessed 1 Feb, 2024.

Miles M. Effective Problem Statements Have These 5 Components. Better Up. May 26, 2023. https://www.betterup.com/blog/problem-statement

Schneiders I. Why is a Real Customer Problem Statement important? Mar 4, 2017. https://productcoalition.com/how-to-write-a-good-customer-problem-statement-a815f80189ba