CIS472 Software Engineering

Project: Flashcards

The Senioritis Solution: Technical Documentation

Senioritis x Spelly: Zuri Levene-Harvell, Brittanie Rice, Jordan Still

Table of Contents

User Stories

USER STORY ID	As a *<type of user>*	I want to *<perform some task>*	so that I can *<achieve some goal>*
1	Teacher	Create a way for my students to be able to review their vocabulary words when outside of the classroom	Ensure that my students are able to complete my course successfully
2	High School Student	Study concepts that I learned in class in a more kinetic way than reading over my notes	Perform well in my class since exams are coming up
3	Parent of Student	Help my child study by quizzing them on topics, but I do not have the time to go and learn them myself	Help my child do better in the subjects that they are struggling with
4	College Student	Order all of the vocabulary words in my notes to make them easier to study	Have a clear and concise plan of what to study so that I do not become overwhelmed
5	Tutor	Come up with a fun way to test my student's knowledge on previously learned topics	Ensure that my students are retaining the information that I teach them
6	ESOL	Become better at english by studying certain words and parts of speech	Be more comfortable in any and all settings
7	Parent of Child with Learning Disability	Aid in the process of teaching my child to speak/read	Make things easier for them in any way possible

Above is the table of user stories of characters that would want/need to interact with the application. These users range in age and desired use of the application. The main character that would be interacting with this platform would be the teachers and students. The students would want to make sure they are able to successfully study the category/topic they need to study. The teacher would want to ensure that the flashcards have enough information.

Functional Requirements

Enumerated Functional Requirements

NOTE: As a group we decided to change the functionality of the application, instead of the user being able to make their own flashcards, they now are able to choose from a database of questions with a large number of subjects. We also added the feature of controlling how many cards appear on screen. The shuffle aspect is still present.

Functional Requirement	Priority Weight	User Story
Selecting the subject for the flashcards	5	As a student I should have the ability to choose which subject I want to study, and have only cards containing questions in said subject appear on the screen.
Changing the number of flashcards visible	5	As any user I would like to have the option to increase or decrease the number of flashcards shown at any given time.
Generating a new set of flashcards	5	As a student in an effort to prevent me from getting questions that I have already answered, I should be able to generate a new group of flashcards.
Reveal the answer to the question presented on the flashcard	5	As a parent or teacher I should be able to check and see the answer to the question printed on the flashcard.
Redisplay the question	4	As any user, if I get the question incorrect I want the

		ability to flip the flashcard back over to try again later

Enumerated Non-Functional Requirements

Non- Functional Requirement	Priority Weight	User Story
Website	5	As a user I should be able to visit this application via website.
Trustworth Data	5	As a user I be sure that the questions and answers on the flash cards are reliable and will return relevant and accurate answers
Updated Data	5	As a user I can be confident that all of the flashcards contain up to date questions and answers due to the backend api updating as new data comes in.

User Interface Requirements

Priority	Requirement
5	The interface shall have a large section displaying all of the flashcards in an easy to read setup.
5	The interface shall have a dropdown menu containing all of the category options
5	The interface shall have a box to change the number of flashcards displayed on the screen
5	The interface shall have a generate button
3	The interface shall have the flashcards light up when the user hovers their mouse over it

2	The interface shall be Spelman blue to match the theme of our webpage

Figure B, is a diagram pictured below gives a broad representation of the on-screen requirements for our application. Please note that this is just a general design and the final patterns/CSS elements will be decided at a later date. This diagram does however demonstrate what the user can expect to see when the application is released.

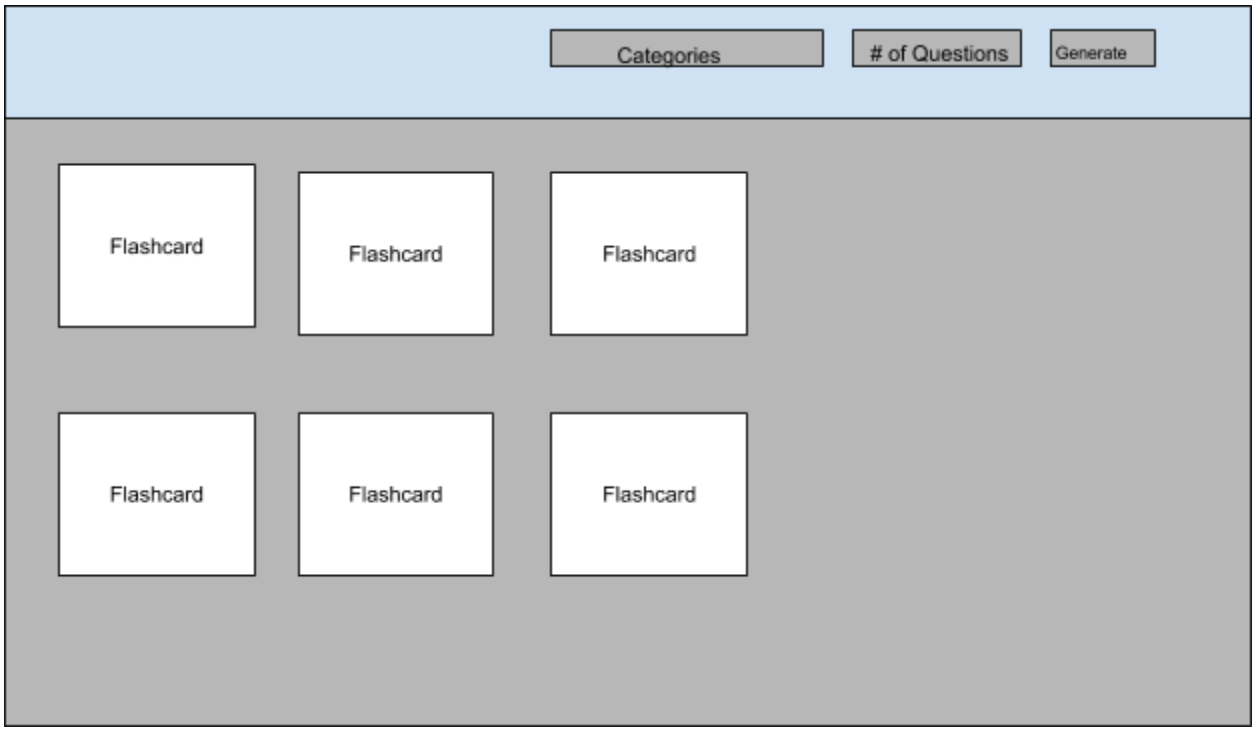


Figure B: a broad representation of the on-screen requirements

System Architecture and System Design

Architectural Styles

For this project, our group is using a layered architectural design. These layers consist of the following aspects:

UI Layer:
The layer where the user is able to interact with the application and make any changes they need without affecting any of the technical aspects of the application.

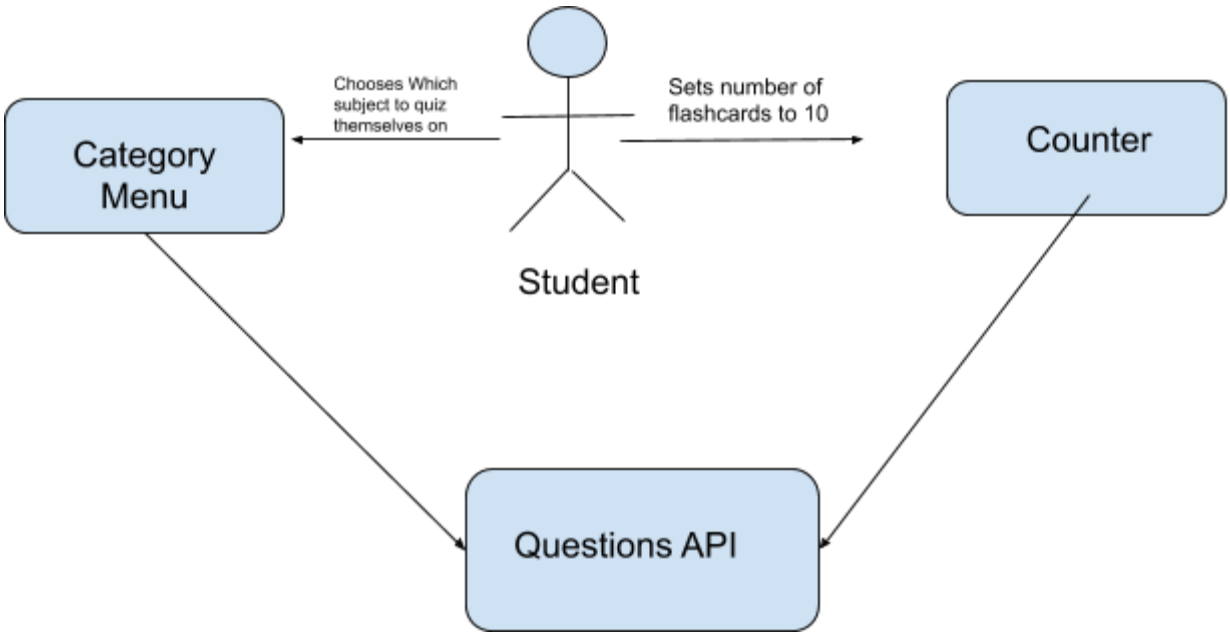
Application Layer:
All of the services of our application will exist on this layer, this includes the dropdown menu and the number of flashcards selection features of the application.

Domain Layer:
This layer will contain the logic of our application that explains how all of the aspects of our application connect together to produce a final product

Data Access Layer:
Our database that contains all of the questions and answers will be housed in this layer.

We chose to use the layered style due to the numerous benefits present. Lack of dependency on other layers allowed for easy reuse of code, which made collaboration between each team members located in different states especially easy.
Layers would allow more reuse of code, which is important in a team of this size and on multiple apps. Each layer's encapsulation and abstractions made testing very simple and straight to the point. Additionally, a layered approach made the application much easier to manage as more functions were added. It will be easier to manage as the software becomes more and more available. The only main drawback of the layered approach is performance, and since our application is fairly simple response time is never an issue, making performance a fairly small requirement.

Identifying Subsystems



This subsystem displays a rough picture of the user choosing which and how many flashcards will be pulled from the applications online API. The diagram shows that the application is dependent upon the student's choices. Once the student makes their selections, the result of their choices will be displayed and they may begin quizzing themselves on their chosen subject.

Global Control Flow

Execution Order:
This entire application is event-driven and said events are dependent on what the user chooses to do. There are however some procedurally driven functionalities present in the application. For example, if the user wants to shuffle the cards and display 20 new questions, they cannot just press the “Generate” button, the user must select the number of questions they wish to appear on the screen first. The same goes for if the user wishes to change subjects, they must first trigger the dropdown menu and select a new subject before generating a new set.

Time Dependency:
This application runs in real-time and there are no time constraints in place.

Hardware Requirements

The system resources that we depend on are the internet and laptop/desktop computers. The communication between the user’s device and the application is dependent upon a stable internet connection. Without a stable connection not only will the website that houses the application be unable to load, but the online API that the questions are being pulled from will be rendered useless.