DECEMBER 10, 2024

INQUISITIVE STUDY SITE (ISS)

Design Documentation -- SENIOR PROJECT I

Caleb Massey
Caleb Ruby
Caleb Rachocki
Ibrahim Alani

Instructor Comments / Evaluation:

Table of Contents:

Abstract:

This document is designated to provide a final overview of the Inquisitive Study Site (ISS) project. It is meant to be read by anyone ready and willing to develop the ISS project. This project will be developed over the course of 3 to 4 months by our team with an original codebase and minimal experience in many areas. This project is to serve as a means to expand our abilities and provide a service to our fellow students in an appropriate academic setting. The ISS will serve as a webpage, with both frontend and backend capabilities and processing to determine what the user would like to study as a subject, as well as how they would like to study it. This design document is meant to further explain the details of the inner workings of the ISS. In this document, we will discuss the modules and their architecture, the flow of our website's data, and various other key factors of the ISS.

Document Description:

Purpose and Use:

The purpose of the Inquisitive Study Site (ISS) design document is a proper and in-depth exploration of the project. This document will provide anybody willing and able to develop this project the capability and understanding to do so. It will also serve as a means for the professor to grade our final project, to determine if everything we wanted to include was, in fact, included. The design document will also serve as a guide for those who develop the ISS project, to ensure that developers do not take too many liberties in creating the project.

Ties to Specification Document:

The design document will expand upon the original explanation of the project in the specification document. This document explains which languages we need for the ISS to be successful. The design document will also explain various structures, libraries, architectures, and planning required to implement the ISS properly. This document serves as the guideline of the ISS, indicating how pieces function, and how they work together to produce the final result.

The intended audience of this document will be the developers of the ISS project and the professor(s) grading the project. We will be using terminology that is based on and revolves around the ISS project. If the reader does not know what a term specific to the ISS means, the reader should refer to the appendix, where we have provided the definitions of our vocabulary. The design document is not limited to only those working on the project and may be used as examples of what to do when creating a design document. The audience should possess a better understanding of the ISS as a whole when finished with this document.

Project Block Diagram:

The complexity of the project calls for simplifying certain parts of this document for conciseness and clarity. This diagram abstractly represented the general flow of the project and details on how each component will function. The following figure serves as a generalized explanation of what each webpage would do.

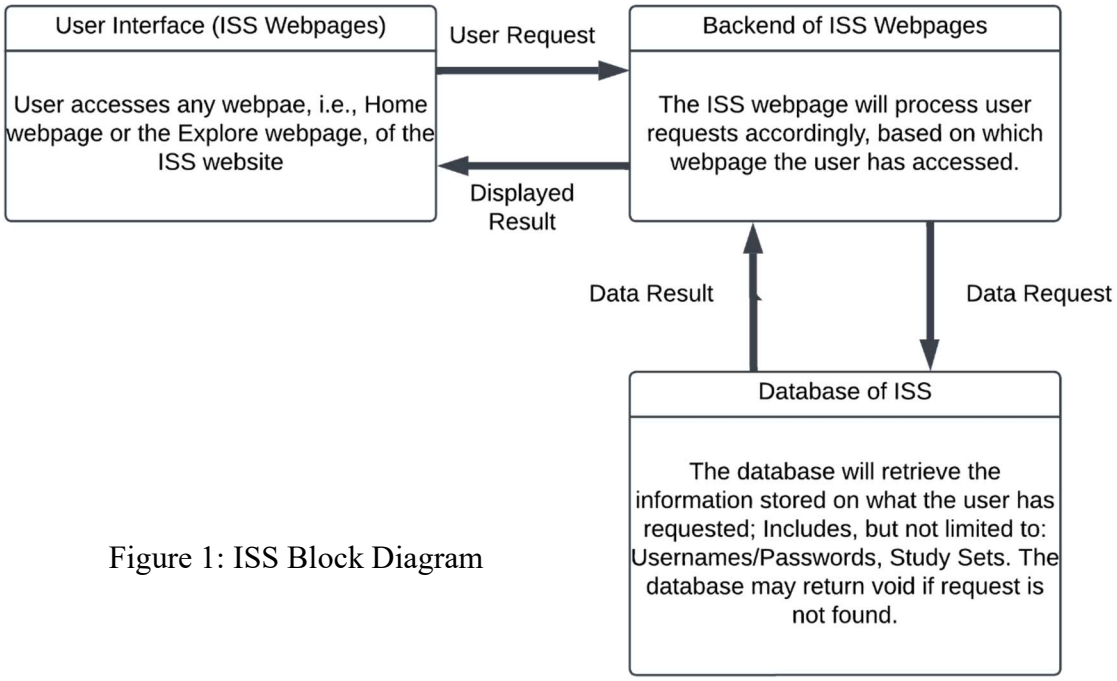

Figure 1: ISS Block Diagram

Design Details:

System Modules and Responsibilities:

Architecture Diagram:

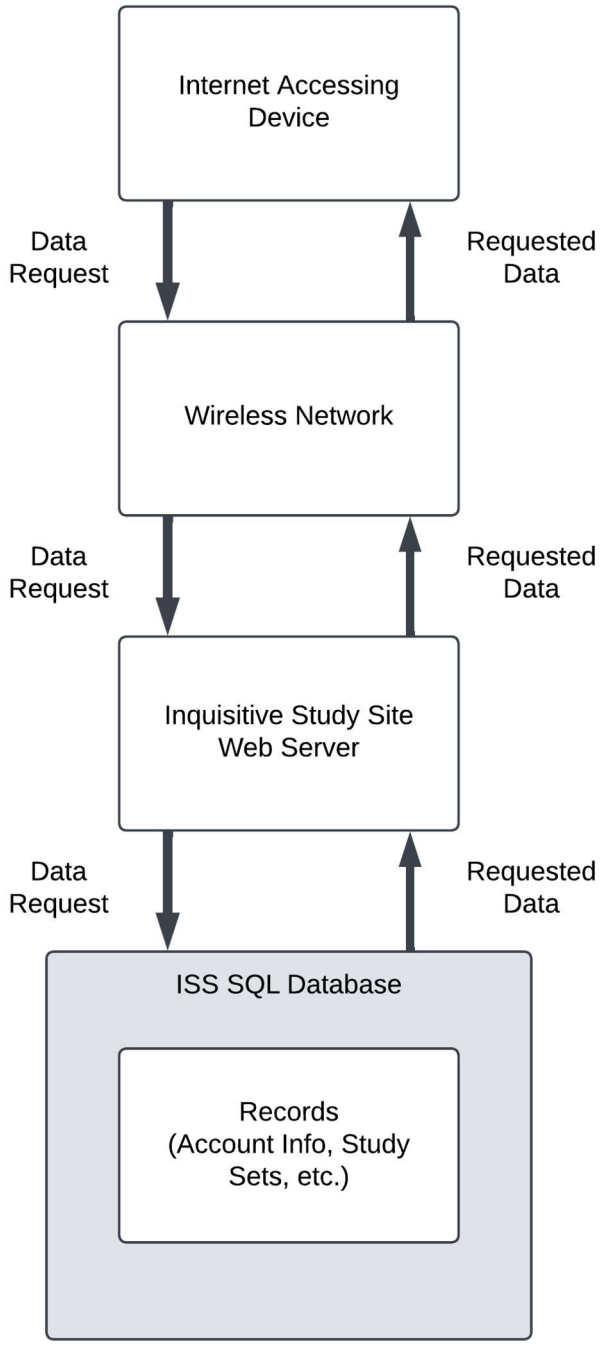

Figure 2: Architectural Structure of the ISS

Description of Figure 2: The user will have a device that is able to access the internet; this includes, but is not limited to, personal computer and laptops. It's in this layer that the user will access the internet with their device over a wireless network to reach the Inquisitive Study Site's domain. Once on the website, the user will be able to access any requested data they want (i.e., account login, account information, study sets, etc.). After the user has made a request for a particular set of data, the ISS will make a request to the backend SQL server to retrieve it; this data will then be sent back to the user for their viewing pleasure on their own device. To clarify, the data will not be stored on their device in any way, but will simply be displayed on it.

Module Cohesion:

The ISS project module cohesion is expressed in its functionality. The ISS will consist of various classes, including "user," "userDA," "studySet," "StudySet&TermDA," and "Term." Users are not required to have an account to access the site, and therefore may not utilize all of the functions available to them; however, others will. The naming convention of the classes in the project is simple; we have the existence of the "user" class, and now have a data access class, "userDA" to access the data being inputted into the SQL database. To simply state this, one function adds and removes, and the other will simply view. All of the classes will work in unison to provide users with the full experience of the ISS's capabilities.

For all of the "non-DA" classes (data, studySet, and Term), the user will be able to freely add and remove data within reason. This is to say that they will be able to alter some details of their account, but not all; this also applies to study sets (time of creation, time of update to data, etc.). The "DA" and "Term" classes will be dependent on the "studySet" and "user" classes. The "studySet" and "user" will be independent of the other classes.

Module Coupling:

Regarding the user's device, so long as it has an internet connection to any of the ISS webpages, the user will be able to use the ISS and all of the functionality that comes with it. With that said, their devices will be able to indirectly utilize all of the classes and functions of the ISS; this is to say that they will need to perform some series of actions to access the functionality of the classes, as well as the classes themselves. For example, if a user wants to use the "userDA" class, they need to navigate to the account login webpage, then access their account; this directs the ISS to retrieve their data and allow them access to their account.

Design Analysis:

Data Flow:

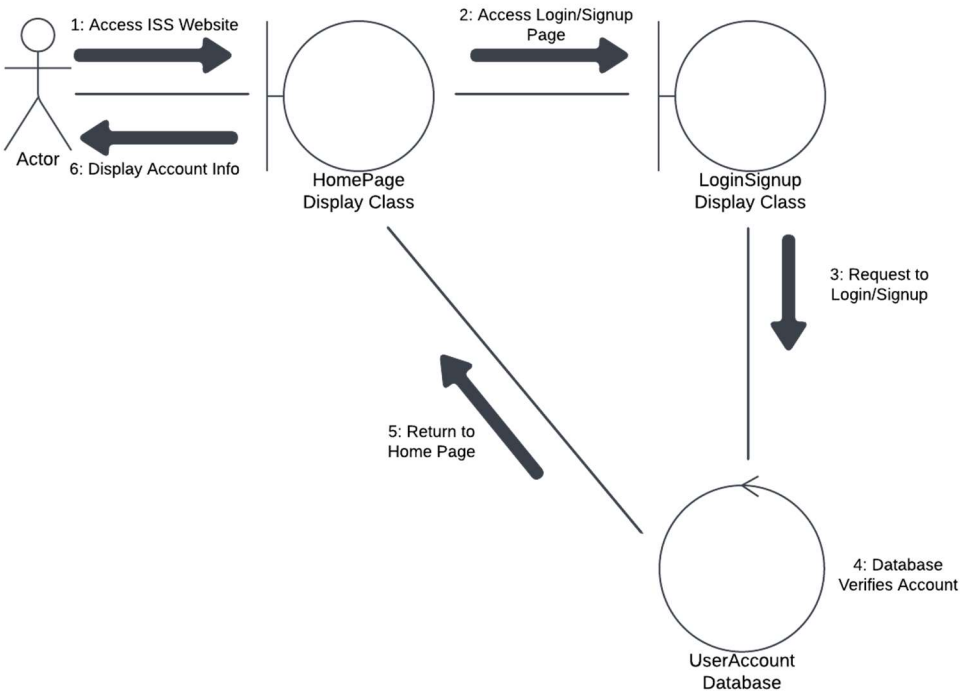

Figure 3: Dataflow Diagram of Account Login Process

Description of Figure 3: A simplified dataflow diagram of the Account Login Process

Design organization (Object-Oriented Design):

Detailed Tabular Description of Classes/Objects:

Class Name: user

Class Description: This class defines what role a visitor to the website has, and thereby granting or restricting their actions on the ISS.

Class Data Members: userName, password, isAdmin, ownedStudySets, creationDate, isLoggedIn, LoginManager()

Class Member Functions: setUserName(), setPassword(), setIsAdmin(), setAccountCreationTime(), createStudySet()

Class Name: userDA

Class Description: This class acts as the "Data Accessor" for the user class; dependent on the user class.

Class Data Members: userName, password, isAdmin, ownedStudySets

Class Member Functions: getUserName(), getPassword(), getIsAdmin(), getAccountCreationTime(), getCreatedStudySets()

Class Name: studySet

Class Description: This class provides the main structure of our study sets.

Class Data Members: studySetName, creatorName, terms[], tags[], creationTime, updateTime, studySetID

Class Member Functions: setName(), setCreatorName(), addTerm(), addTags(), setCreationTime(), setUpdateTime(), setStudySetID

Class Name: studySet&TermDA

Class Description: Connection to the SQL database

Class Data Members: studySetName, creatorName, terms[], tags[], creationTime, updateTime, studySetID

Class Member Functions: getName(), getCreatorName(), getTerm(), getTags(), getCreationTime(), getUpdateTime() getStudySetID()

Class Name: Term

Class Description: This class provides the actual structure of a term provided by the user, which will then be stored in a study set.

Class Data Members: termID, term, definition

Class Member Functions: setID(), getID, setTerm(), getTerm(), setDefinition, getDefinition()

Class Name: studySetSearch

Class Description: This class provides searching capabilities for study

Class Data Members: searchInput, tags[], searchType

Class Member Functions: setSearchInput(), setSearchTags(), setSearchType(), search(), searchDisplay()

Class Name: LoginManager

Class Description: This class handles the login and authentication of users

Class Data Members: userName, password, isLoggedIn

Class Member Functions: login(), logout(), createUser(), validateCreation(), validateLogin(), setLoginStatus()

Class Name: tagManager

Class Description: This class manages tags related to study sets, including adding, removing, and retrieving tags.

Class Data Members: tagName. studySetID

Class Member Functions: addTag(), removeTag(), getTags()

Class Name: StudyEnvironment

Class Description: This class manages flashcards, practice quizzes, and Q/A study formats

Class Data Members: studyType, studyResult[], isCorrect, questionOutput, answerInput

Class Member Functions: setStudyType(), displayFlashcards(), displayPracticeQuiz(), displayQ&A(), displayResults(), setCorrect(), endSession()

Functional Description:

user Class

setUserName()

Input:

The user will input a string of their choosing that will be tied directly to their account as their username at the time of account creation or account updates.

Output:

The output of the function will be displayed as either a successful sign up, or as a failure to sign up. Dependent on the user's input. Should a user be successful, their username will be displayed on banners across the website, and an account will be created with this username and corresponding password. May also be used to update a username of preexisting account.

Return Parameters:

The values returned by the function will either be the user's successful account generation, or it will return an error message.

Types:

Strings

setPassword()

Input:

The user will input a string of their choosing that will be tied directly to their account as their password at time of account creation.

Output:

The output of the function will be displayed as either a successful sign up, or as a failure to sign up. Dependent on the user's input. Should a user be successful, their password will be accepted, and an account will be created with this password and the corresponding username. May also be used to update a password of preexisting accounts.

Return Parameters:

The values returned by the function will either be the user's successful account generation, or it will return an error message.

Types:

Strings

setIsAdmin()

Input:

setIsAdmin() has no input; this will act as a flag for ISS accounts.

Output:

setIsAdmin() has no output; this will act as a flag for ISS accounts.

Return Parameters:

setIsAdmin() has no return parameters aside from the possibility of an error.

Types:

Boolean

setAccountCreationTime()

Input:

Retrieves current date and time for account upon creation; account will have a timestamp of the current date and time upon creation. Stored as a string for mutability.

Output:

Time of account creation will be linked to and displayed with the account's information.

Return Parameters:

The value returned is that of the timestamp, or a possible error exception.

Types:

String

createStudySet()

Input:

Users will be redirected to the study set creation webpage. Users will then be prompted to create a name for their study set, and add a minimum of 1 term and its corresponding definition.

Output:

Outputs a user generated study set (i.e., their own terms and definitions). Otherwise, the output may be an error if the user is for some reason unable to properly create the study set.

Return Parameters:

Possible errors thrown during study set creation.

Types:

Strings

End of user class functions.

---

userDA Class

getUserName()

Input:

The getUserName() function has no input.

Output:

This function should display the requested username of a given account.

Return Parameters:

This function will return a requested username, or it will return an error message; this is dependent on whether the username exists within the database.

Types:

Strings

getPassword()

Input:

The getPassword() function has no input.

Output:

This function should display the requested password of a given account.

Return Parameters:

This function will return a requested password, or it will return an error message; this is dependent on whether the password exists within the database.

Types:

Strings

getIsAdmin()

Input:

getIsAdmin() has no input; this will act as a flag for ISS accounts.

Output:

getIsAdmin() has no output; this will act as a flag for ISS accounts.

Return Parameters:

This function will return a boolean to determine whether an account has administrator privileges.

Types:

Boolean

getAccountCreationTime()

Input:

getAccountCreationTime() has no input.

Output:

This function will display the time a given account was created on thew account's respective webpage.

Return Parameters:

Returns the account creation timestamp.

Types:

String

getCreatedStudySets()

Input:

getCreatedStudySets() has no input.

Output:

getCreatedStudySets() will display all of the study sets a user has created up to that point in time.

Return Parameters:

getCreatedStudySets() will return objects of the class “studySet” that are tied to a specific account.

Types:

Objects of the class “studySet”

studySet Class

setName()

Input:

A string is required for the input of the study set's name.

Output:

setName() only sets the name of a given study set, and therefore has no output aside from a successful name input.

Return Parameters:

Possible error thrown during execution.

Types:

String

setCreatorName()

Input:

The name of the user who created the study set, pulled directly from the user's account information.

Output:

setCreatorName() has no output other than successfully creating a study set.

Return Parameters:

Possible error thrown during execution.

Types:

String

addTerm()

Input:

A term to be added to the study set at the user's request.

Output:

No specific output aside from successfully creating a study set.

Return Parameters:

Possible error thrown during execution.

Types:

String

addTags[]

Input:

A tag to be added to a given study set at the user's request.

Output:

No specific output aside from successfully creating a study set.

Return Parameters:

Possible error thrown during execution.

Types:

String

setCreationTime()

Input:

The date and time of when the study set was created stored as either a string, or a language's specific DateTime data type.

Output:

Acts as a setter for the study set; has no specific output.

Return Parameters:

Possible error thrown during execution.

Types:

String and/or DateTime type.

setUpdateTime()

Input:

The current date and time of when a study set was updated stored as either a string or a language's specific DateTime data type.

Output:

Acts as a setter for the study set; has no specific output.

Return Parameter:

Possible error thrown during execution.

Types:

String and/or DateTime Type

setStudySetID()

Input:

A given identifier for a study set, unique to that specific study set.

Output:

setStudySetID() does not have any kind of output.

Return Parameters:

Possible error thrown during execution.

Types:

Integer

studySet&TermDA Class

getName()

***Input**:*

This function does not have any input.

***Output**:*

Prints the name of the given study set to the area the webpage is requesting it.

Return Parameters:

Retrieves the name of a given study set to be used in the requested area of the webpage.

***Types**:*

String

getCreatorName()

***Input**:*

This function does not have any formal input.

***Output**:*

Prints the name of the creator of a given study set.

***Return Parameters**:*

Retrieves the creator's name of a given study set to be used in the requested area of the webpage.

***Types**:*

String

getTerm()

***Input**:*

This function does not have any formal input.

***Output**:*

Retrieves the terms in a given study set to be used in the requested area of the webpage.

Return Parameters:

Returns an array or linked list of terms that correspond to the given study set.

Types:

Array / Linked List of Strings

getTags()

***Input**:*

This function does not have any formal input.

***Output**:*

Retrieves the tags of a given study set to be used in the requested area of a webpage.

***Return Parameters**:*

Returns an array or linked list of terms that correspond to the given study set.

***Types**:*

Array / Linked Lists of Strings

getCreationTime()

***Input**:*

This function does not have any formal input.

***Output**:*

Retrieves the creation time of a given study set for use in the requested area of a webpage.

***Return Parameters**:*

Returns the creation time of the study set.

***Types**:*

String (or DateTime variable if available).

getUpdateTime()

***Input**:*

This function does not have any formal input.

***Output**:*

Retrieves the last time a given study set was updated for use in the requested area of a webpage.

***Return Parameters**:*

Returns the last update time of the study set.

***Types**:*

String (or DateTime variable if available).

getStudySetID()

***Input**:*

This function does not have any formal input.

***Output**:*

Retrieves the study set's ID for use in the backend of the ISS.

***Return Parameters**:*

Returns a study set's unique ID.

***Types**:*

Integer

Term Class

setID()

***Input**:*

This function requires a unique integer input that no other term already has. Will be generated by the ISS.

***Output**:*

This function does not have any formal output.

***Return Parameters**:*

This function does have any formal return parameters.

***Types**:*

Integer

getID()

***Input**:*

This function does have any formal input.

***Output**:*

Retrieves the unique ID of a given term for use in a requested area of a webpage.

***Return Parameters**:*

Returns the associated termID of a study set.

***Types**:*

Integer

setTerm()

***Input**:*

The actual termValue provided by the user.

***Output**:*

This function does not have any formal output.

***Return Parameters**:*

This function does not have any formal return parameters.

***Types**:*

String

getTerm()

***Input**:*

This function does not have any formal input.

***Output**:*

Retrieves the associated term for use in a requested area on a webpage.

***Return Parameters**:*

Returns the term requested based on the termID.

***Types**:*

String

setDefinition()

***Input**:*

The actual definitionValue provided by the user for its corresponding term.

***Output**:*

This function does not have any formal output.

***Return Parameters**:*

This function does not have any formal return parameters.

***Types**:*

String

getDefinition()

***Input**:*

This function does not have any formal input.

***Output**:*

Retrieves the definition of a given term based on the termID. Will be used in the requested area of a webpage.

***Return Parameters**:*

Returns the associated definition of a term.

***Types**:*

String

studySetSearch Class

setSearchInput()

Input*:*

The search query from the user.

Output*:*

This function does not have any formal output.

Return Parameters*:*

This function does not have any formal return parameters.

Types*:*

String

setSearchTags()

***Input**:*

The tag query from the user.

***Output**:*

This function does not have any formal output.

***Return Parameters**:*

This function does have any formal return parameters.

***Types**:*

String

setSearchType()

***Input**:*

Input selection based on checkbox from user.

***Output**:*

This function does not have any formal output.

***Return Parameters**:*

This function does not have any formal return parameter.

***Types**:*

searchType data member.

search()

***Input**:*

Works reliantly on the setSearchInput() function to search the database.

***Output**:*

This function performs the search operation based on the searchInput, tags[], and searchType.

***Return Parameters**:*

Returns various objects of studySet class.

***Types**:*

Objects of the studySet class.

searchDisplay()

***Input**:*

This function does not have any formal input.

***Output**:*

This function displays the results of the search() function in a readable way.

***Return Parameters**:*

This function does not have any forma return parameters.

***Types**:*

Objects of “studySet” class.

LoginManager Class

login()

Input:

This function allows inputs for username and password

Output:

This function will check for authenticates for the user if valid info then logs user in.

Return Parameters:

This function will return a true or a successful message if logged in with correct info.

Types:

Strings

logout()

Input:

This function have no inputs

Output:

This function will log the user out

Return Parameters:

This function will return a success message if logged out correctly.

Types:

Boolean flag

createUser()

Input:

This function will input username and password.

Output:

This function will output and create a new user account.

Return Parameters:

This function will return a true or false, if is successful.

Types:

Strings

validateCreation()

Input:

This function will have inputs of username and password.

Output:

This function output will check if username and passwords requirements are all met.

Return Parameters:

This function doesn't have a return parameter.

Types:

Strngs

validateLogin()

Input:

This function will have inputs of username and password.

Output:

This function output will check user identity.

Return Parameters:

This function have no return parameters

Types:

String

setLoginStatus()

Input:

This function will set a boolean flag for user status to be login status.

Output:

This function will update user status.

Return Parameters:

This function have no return parameters.

Types:

Boolean

tagManager Class

addTag()

Input*:*

Requires an input string based on the creator's desire.

Output*:*

This function does not have any formal output.

Return Parameters*:*

This function will return a boolean based on whether the tag is successfully added.

Types*:*

String, Boolean

removeTag()

***Input**:*

Requires an input based on the desire of the creator's account.

***Output**:*

This function outputs the removed tag should the tag be successfully removed.

***Return Parameters**:*

This function will return a boolean based on whether the tag was successfully removed.

***Types**:*

String, Boolean

getTags()

***Input**:*

This function does not have any formal input.

***Output**:*

Retrieves the tags applied to a given study set.

***Return Parameters**:*

Returns the tags associated with a given study set.

***Types**:*

Array or List of Strings

StudyEnvironment Class

setStudyType()

Input:

Takes in integer value representing which study method the user requests

Output:

This function sets the study type for a session, calling one of the display functions

Return Parameters:

N/A

Types:

String

displayFlashcards()

Input:

ID number for study set

Output:

Signals website to display flashcard study style, fetching data relevant to currenty study set

Return Parameters: N/A

Types: strings

displayPracticeQuiz()

Input:

ID number of study set

Output:

Signals website to display practice quiz study style, fetching data relevant to currenty study set

Return Parameters:

N/A

Types:

strings

displayQ&A()

Input:

ID number for study set

Output:

signal for website to display question and answer format page for the given study set

Return Parameters:

N/A

Types:

strings

displayResults()

Input:

Results array

Output:

List of terms with a grade next to each one (correct/incorrect)

Return Parameters:

N/A

Types:

Function deals with boolean array and outputs strings

setCorrect()

Input:

The ID for the current term being displayed to the user as well as the users response

Output:

Message to the user telling them if they were correct or not, followed by the correct answer if they were not

Return Parameters:

Returns a boolean value to the corresponding spot in the results array

Types:

The function will mostly process and output strings

endSession()

Input:

Study-session type (integer)

Output:

Flag signaling website to display a results page and a call to the displayResults() function

Return Parameters:

There will not be return parameters for this function

Types:

This function will handle a Boolean flag and a Boolean array corresponding to the terms of the study set

Files Accessed:

Most data will be stored server-side, such as user's study sets, usernames, and passwords. The only data that will be stored locally is the user's login status, so their browser will remember if they were logged in or not, eliminating the need to re-enter their username and password each time they wish to use our service.

Real-Time Requirements:

To ensure a positive user experience, our website must respond quickly and efficiently, minimizing any potential frustration for our users. However, it's important to note that the speed at which elements load on the site is largely influenced by the user's internet connection. While we strive to optimize our site's performance, factors such as bandwidth and connection quality play a significant role in the overall loading times. Therefore, users with slower internet speeds may experience longer load times, despite our best efforts to maintain a fast and responsive website.

Messages:

Code	***Description***
001	Failure to find user account
002	Failure to find study set
003	Failure to load study set
004	Failure to load term / definition
005	Invalid User Input

Narrative/PDL:

Home Page:

1. Sign In
2. Search
3. Study Set
4. Explore
5. About

Sign-In Webpage

1. “Enter Username” Text Field
2. “Enter Password” Text Field
3. “Login / Signup” Button
4. Account Not Found?
    a. Error Code – 001
5. Create Study Set
    a. Invalid Input?
        i. Error Code – 005

Search Webpage

1. “Enter Study Set Name” Text Field
2. Study Set Not Found?
    a. Error Code – 002

Study Set Webpage

1. Flash Card
2. Practice Exam / Quiz
3. Term Not Found?
    a. Error Code – 004
4. Study Set Not Found?
    a. Error Code – 003

Explore Webpage

1. “Enter Study Set Name” Text Field
2. Study Set Not Found?
    a. Error Code – 002

About Webpage

1. View Webpage

Decision: Programming Language/Reuse/Portability:

The ISS is designed to work on any given platform that has internet access. This includes personal computers, laptops, and cellphones. This is to ensure that any student may be able to access the resources we wish to provide them with. Most of the ISS will be written in HTML, CSS, JavaScript, and SQL. This ensures that the ISS is compatible with most, if not all devices that can properly utilize the internet and a given web browser at the time of writing this.

At least two of our team members have experience in SQL, HTML, CSS, and JavaScript, which does provide an advantage in creating the project. This also creates an opportunity for the other team members to further expand their knowledge by learning new concepts and languages for use in the field.

Implementation Timeline:

JANUARY	FEBRUARY	MARCH	APRIL	MAY
Developing	Developing	Developing	Developing	Website Launch
Testing	Testing	Testing	Testing	Bug Fixing/ Polish
Alterations*			Peer Reviews	Presentation

* May have small changes to the specification of the ISS in early stages.

Design Testing:

Testing will be performed throughout the entirety of the development of the ISS. We will be reviewing the code we all write as a team. This is to say that 3 members will review the work of 1 team member. This applies to all team members and their work. Group members will also work with people outside of the project, i.e. family and friends, to determine if the project is user-friendly to the extent team members desire it to be.

References:

Raspberry Pi 4 Model B; Broadcom BCM2711 Quad Core Cortex-A72 (ARM V8) 64-bit SOC; 8GB DDR4 RAM. Micro Center. (n.d.). https://www.microcenter.com/product/622539/pi4modelB8gb?src=raspberrypi

Appendix A: Team Details:

Each team member added content to the design document and helped in creating visual aids displayed in aforementioned document. There was a large amount of overlap in our work, and nothing was completed by one single team member. Listed below are the specific contributions that each team member spearheaded.

- Caleb Massey
    - Initial draft of the design document.
    - Creation of most visual aids and appearance of design document.
- Caleb Ruby
    - Proof and rewrite of portions of the document; majority of the editing of the document.
    - Reviewed classes and their functions.
- Caleb Rachocki
    - Reviewed functions of the classes
    - Continued work on the mock-up website for more visualization of our functions and classes required by the ISS.
- Ibrahim Alani
    - Reviewed the beginning portions of the document
    - Led the description of each diagram and thoroughly explained them.

Appendix B: Writing Center Report:

Grammar:
- Commas and Conjunctions: I noticed a few times that there is a comma where it is not needed or there is a conjunction missing where it needs to be. Remember that the only time a sentence with "and" needs to be separated by a comma is when it's connecting two independent clauses. I suggest reading both portions of the sentence without the “and” to determine if you need the comma.

Hyphens:
- There were only a couple of words that either needed a hyphen added or needed the hyphen removed.

Wording:
- At the end of the document, there were a few places where word choice didn't flow.

I added comments where all of these errors occurred. Overall, the document had very little and minute errors. It was clear and easy to understand, even as someone who is not in that class or specific major.

Appendix C: Workflow Authentication:

I, Caleb Massey, testify that any and all work on this document, as well as future work on the project, was done so by adhering to the requirements description of the document.

SIGNATURE: 

Date: 12/9/24

I, Caleb Rachocki, testify that any and all work on this document, as well as future work on the project, was done so by adhering to the requirements description of the document.

Caleb Rachocki

SIGNATURE: 

Date: 12/9/24

I, Caleb Ruby, testify that any and all work on this document, as well as future work on the project, was done so by adhering to the requirements description of the document.

SIGNATURE:

Date: 12/9/24 

I, Ibrahim K. Alani, testify that any and all work on this document, as well as future work on the project, was done so by adhering to the requirements description of the document.

SIGNATURE:

Date: 12/9/24