LetuStudy

System Design Document

Version 1.0

03/10/2023

Table of Contents

List of Figures

List of Tables

1. Introduction

This System Design Document (SDD) is intended to describe the proposed system design for LetuStudy app, a comprehensive learning management app that allows educators and students to interact and manage study materials in a collaborative and efficient manner. This document outlines the system requirements, architecture, design, and implementation details necessary to develop and deploy the LetuStudy App.

1.1 Purpose of the SDD

The purpose of this SDD is to provide a comprehensive description of the proposed LetuStudy app design, including the system architecture, components, data flow, and other critical details necessary for developers to implement the system. This document also serves as a reference for a variety of users, including the development team, project managers, and customers, to understand the proposed system's design and functionality.

2. General Overview and Design Guidelines/Approach

The general overview of the Study App design is to provide a user-friendly and efficient platform for students to organize their study materials, collaborate with peers and instructors, and track their academic progress.

2.1 General Overview

The LetuStudy app is a mobile application designed to provide users with a comprehensive study tool. The app includes various features like flashcards, statistics, and calendars to help users study for any subject or class. In addition to these features, the app will also provide customization options to allow users to enhance their study experience by their needs. The app will be available on iOS and planned to adopt on Android platform in the future. The LetuStudy app will be designed with a user-friendly interface to enhance usability. This document outlines the software design of the LetuStudy app, including the architecture, design patterns, and development methodologies.

2.2 Assumptions/Constraints/Risks

2.2.1 Assumptions

- The user has access to a stable and reliable internet connection to access the app's services. App is usable without internet, but some functions (Login, Register, Sync etc.) might not work.
- Users have compatible devices running on iOS 16.0 or later with at least 2GB of RAM.
- The user has basic knowledge of using mobile applications and will be able to navigate the application without assistance.
- The application will be subject to regular maintenance and updates to ensure compatibility with the latest version of iOS.
- The user will provide accurate information during registration and usage of the application.

2.2.2 Constraints

- The app needs to be developed using Swift programming language and the Xcode IDE.
- The app is majorly designed to run on iOS devices.
- The app must not collect any personal user data without their explicit consent.
- The app needs to comply with all relevant security standards and regulations.
- The app must be designed to be easily navigable and user-friendly.
- The app must be compatible with iOS/iPadOS version 16 and higher.
- The app is designed to work either offline or online.
- The app must be able to synchronize user data with the cloud.

2.2.3 Risks

1. **Data Security Risks**: As the app stores personal information and sensitive data, there is a risk of data breaches or cyber-attacks, which could result in loss of user data, legal issues and loss of reputation.
2. **Technical Risks**: Technical risks such as system crashes, errors, and bugs could lead to data loss, system downtime, and damage to user experience.
3. **Integration Risks**: As the app integrates with third-party tools and software, there is a risk of incompatibility and integration issues, which could impact app performance and functionality.
4. **Performance Risks**: The app's performance could be impacted by factors such as server downtime, slow network speed, and poor device performance, leading to poor user experience and negative feedback.
5. **Legal and Compliance Risks**: The app must comply with legal regulations and data protection laws, failure to do so could lead to legal issues, fines, and loss of reputation.

3. Design Considerations

3.1 Goals and Guidelines

Goals:

One of the main objectives is to give users the ability to utilize the study features in offline mode, while ensuring that study sets are uploaded to the cloud as quickly as possible when the system is online. The implementation of the codebase will adhere strictly to the Swift guidelines and Xcode standards. Additionally, enabling the sharing and publication of study sets to other users is a significant goal. The design must emphasize user-friendliness and ensure that the overall user experience is both simplistic and efficient.

3.2 Development Methods

Development Methods:

Agile development will be used to allow for flexibility and adaptability in the development process, with regular iterations and user feedback during the app development process. The prototype is already made for UI (User Interface). Also, to illustrate the workflow and interactions between various elements of the LetuStudy application, an event-driven process diagram will be developed. This will make it easier for team members to communicate and will assist detect any process problems or bottlenecks. Events, activities, and gateways are all included in the diagram, which will be changed as needed throughout the development process.

Contingencies:

In case of compatibility issues or unexpected changes in the iOS platform or Xcode, contingency plans will be in place to adapt the development process accordingly. This may involve updating dependencies, finding workarounds, or implementing alternative solutions.

3.3 Architectural Strategies

The LetuStudy application shall be developed by utilizing industry-standard programming languages, Swift and Object-C, and shall be built using Xcode development environment. Furthermore, the application shall leverage the use of a reliable and efficient SQLite Database Management System (DBMS) for securely storing and managing the user's sensitive information, credentials, and other study materials.

4. System Architecture and Architecture Design

This section describes the overall structure of the application, including the hardware and software components, and their interactions and relationships. It also defines the architectural styles, patterns, and principles that are being used to guide the design and implementation of the system. This section aims to provide a clear and concise representation of the system's architecture, allowing stakeholders to understand the design choices and make informed decisions about the system's development and evolution.

4.1 Logical View

The logical view of the LetuStudy app is divided into several views to manage different aspects of the app's functionality. The main view, ContentView, serves as the entry point to the app and includes a tab bar to switch between different views. The following are the logical views in the LetuStudy app:

HomeView: This view displays a welcome screen with user’s information including avatar and name. It also shows recently used study sets and last time using the app.

CardsView: This view allows users to create, edit or delete a set of flashcards for a specific subject. It includes options to add, edit or delete flashcards, a clickable flashcard that can be flipped, and the previous/next button to switch between flashcards.

QuizView: This view allows users to create a quiz from a study set or take a quiz from an existing set. It includes options to customize the quiz, such as the number of questions and the time limit.

ResultView: This view displays the results of a quiz, including the number of correct and incorrect answers, and the user's score.

StatsView: This view displays the statistics of user, including total time spent in each study function of the app.

SettingsView: This view provides options for users to customize the app's settings, such as notification preferences, sound settings, and disability mode.

4.2 Hardware Architecture

The LetuStudy app is developed and designed to run on iOS / iPadOS devices with the following minimum requirements:

- iPhone 8 or later
- iPad (5th generation) or later
- iOS / iPadOS 16.0 or later

4.2.1 Security Hardware Architecture

- The user's login credentials will be stored on the device's Keychain securely. No external hardware component will be able to access sensitive data.

- The app will use Touch ID or Face ID to authenticate the user and grant access to the app and their data. When a user creates an account, they will be asked to choose if to use the Touch ID or Face ID for login.
- The app will use HTTPS to encrypt all data transmissions between the app and any online servers.
- To guarantee the safety and availability of sensitive user data, it follows a standard procedure for business. Since the LetuStudy app will be storing and transmitting sensitive user data as inputs, such as login credentials, study progress, and personal data, it is an essential component of the overall design.
- The servers hosting user data will use AWS, Google Cloud, Microsoft Azure, or other similar level cloud services providers, which provide firewalls, intrusion detection systems, and other security measures to prevent unauthorized access.
- The servers will be in secure data centers with restricted physical access and backup power supplies to ensure uptime and prevent data loss.
- The app will have a timeout feature that automatically logs the user out after a period of inactivity and requires re-authentication to regain access. It utilizes the iOS internal timer feature to log user's inactive time. Once the inactive time exceeded a threshold, the app will log user out. Each user's operation in app will reset the inactive timer.
- The app will use the latest version of iOS and Xcode and will regularly receive security updates to address any new vulnerabilities or threats.

4.2.2 Performance Hardware Architecture

To ensure optimal performance, the app will utilize several hardware components including:

- **CPU**: The app requires a minimum Apple A11 Bionic Processor to ensure that it can handle multiple tasks simultaneously without lag. The app is optimized to utilize the multicore CPUs that are available on most modern devices after 2017.
- **GPU**: The app is optimized to make use of the latest GPU technologies to provide a smooth and responsive user experience. With Apple's Metal API, GPU on user's devices will be used to accelerate the generation of charts, forms and animations.
- **Storage**: The app will make use of the device's internal storage to store user data, study materials, and other relevant information. The app is optimized to minimize the amount of storage space required, while still providing users with a rich and engaging experience.
- **Networking**: The app will utilize the device's networking capabilities to access online resources and provide account and syncing functions. An internet connection with 5 Mbps or above is recommended.

4.3 Software Architecture

The software architecture is modeled after an event driven design. The software will communicate with a database (SQLite) and the devices' local filesystem. Currently, the only API being referenced is OAuth2.0. This API allows the user to login to the application with their google accounts. The software is currently coded in Swift. As the application is expanded to Android users, there will need to be a re-write as Android cannot be supported purely with Swift. All the

data shown in the figure below will be stored locally on the device and in cloud storage. The relationship between the specific actions and related events is the reason that an event driven design pattern was chosen. There are specific events that then update the user, system, or both. Observing the software as event driven allows the observation of the specific relationships between events and their actions.

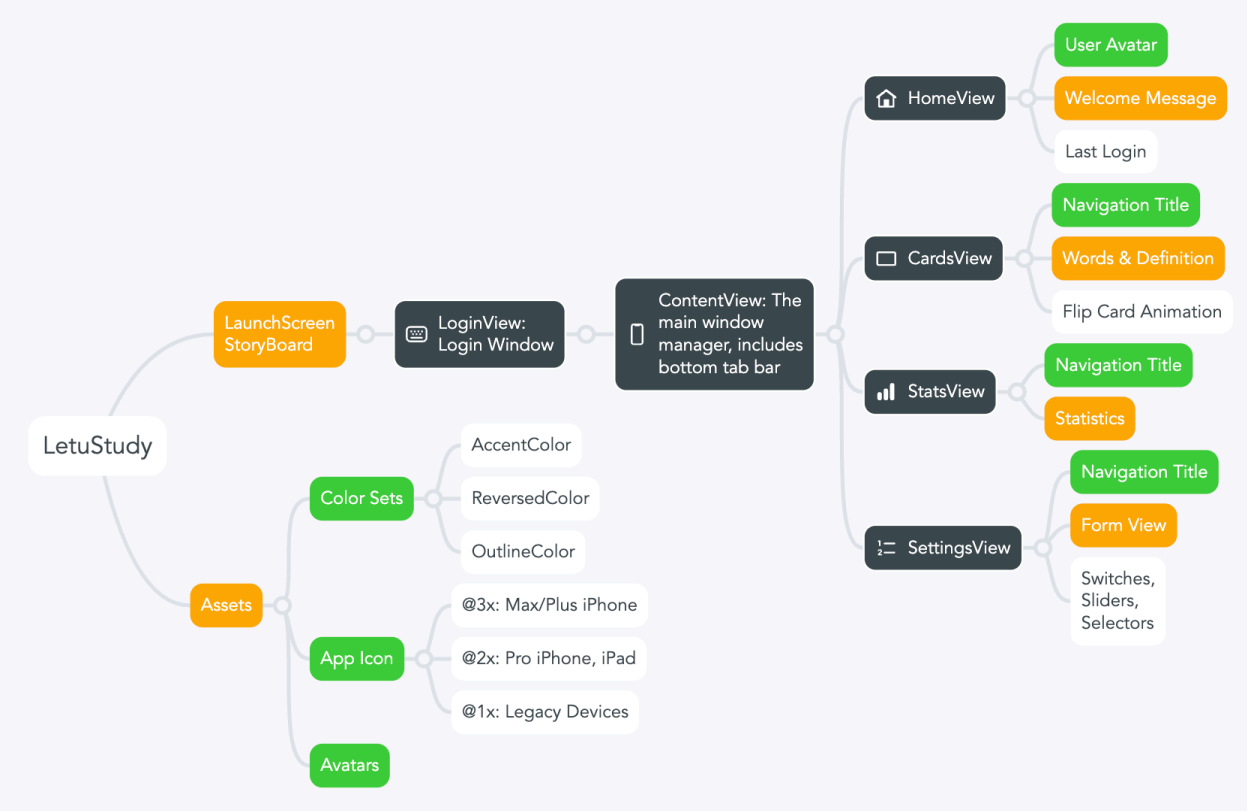

Figure 1 App Architecture

4.3.1 Performance Software Architecture

A cloud database stores and manages the study app's data will be held on Amazon Web Services or similar platform, which provides high availability through a database cluster and load-balanced configuration to provide redundancy and fault tolerance. The LetuStudy app itself can track system performance, detect errors, and events for troubleshooting and analysis through the internal logs.

Therefore, the performance tracking will be supported by the implementors as they will have access to the internal logs of the app. They will be able to analyze the logs and troubleshoot any errors or issues that comes up, or AWS may also provide built-in tols for monitoring and tracking system performance if that is being used.

4.4 Information Architecture

All flashcard data will be stored securely on the device and the cloud. There will be no critical personal or health information associated with this system. All information will be passed between the user, the device, and the cloud. A secure and encrypted storage option for sensitive data, including user's credentials, is provided via Keychain, a built-in iOS feature. The threat of unauthorized access or data breaches can be decreased by using the Keychain to ensure that user credentials are maintained in a secure and protected manner. Other relative files and database will be stored in isolated directories secured by iOS/iPadOS file system.

The user's credentials will undergo a SHA512 hashing algorithm, with the addition of salt for password hashing to enhance security. The hashed credentials will then be stored to the database under iOS Security Framework.

4.4.1.1 Data

LoginView

- Input:
  - Username – Strings
  - Password – Strings
  - Both will be unique to each user and will be stored in a user database for future logins.
- Output:
  - None

ContentView

- Input:
  - tabViewSelection – Integer 0 -3, represents a @State
- Output:
  - The state that you selected, which will be the different view described below.

HomeView

- Input:
  - None
- Output:
  - Users first and last name – Strings
  - Last study set opened – Strings
  - Last time opened – Date and Time

CardsView

- Input: (When creating a study set)
  - Study Set
    - Title – Strings
    - Flash Card
      - Question – Strings

- Answer – Strings
- Study Set description – Strings

- Each study set will be stored on the device and the cloud for use on multiple devices.

- Output:
  - Also, the study sets for the user to view and use.

StatsView

- Input:
  - totalMinutes – float, difference of the endMinutes and startMinutes. Calculations will be done from ContentView page.
  - datesList – object list of date and times.
- Output:
  - Time statistics using built in statistics functions in Xcode.
  - Calendar showing a streak of dates List.

SettingsView

- Input:
  - Notifications – Boolean, saves user's preference.
  - Dark Mode – Boolean, saves user's preference.
  - Sound – Boolean, saves user's preference.
  - Accessibility Settings – Boolean, saves user's preference.
    - Reverse Color – Boolean, saves user's preference.
    - Voiceover – Boolean, saves user's preference.
  - Brightness Adjustment – float between 0 – 100
  - Select your role – selector with 2 parts. The selector list, and then the integer associated to a specific role.
- Output:
  - The users will see the "output" corresponding to the settings that they chose.

4.4.1.2 Manual/Electronic Inputs

The app will support manual input through a user-friendly interface with fields for entering questions and answers, and an option to add multimedia content such as images, study cards or other files.

Electronic inputs will be supported by a file picker interface that will allow users to select files from their local device, iCloud, or other cloud-based storage services. The app will parse and extract the text from the selected files and store them in the SQLite database as separate question and answer pairs. To ensure proper processing, the input files must be in TXT format. The study cards' questions should occupy every odd-numbered line, while the study card answers should occupy every even-numbered line.

4.4.1.3 Master Files

The master files may contain the source code for each view in SwiftUI, user's study set and other settings or analytics data.

5. System Design

5.1 Business Requirements

This is a school project, so potentially, in the future, our project can be qualified for a business and its requirements, as we develop. Design aspects for Business Requirements would include user interface design, accessibility design, data scalability and management design, and design for premium services. To make sure that the LetuStudy app satisfies the commercial goals specified in the Business Requirements, it is crucial for the project team to take certain design considerations into account. To deliver a valuable and competitive solution in the education technology market, the LetuStudy app attempts to satisfy several important commercial needs. These prerequisites are:

- Providing a user-friendly and intuitive interface for users to create/update/delete flashcards and quizzes, promoting ease of use and learning engagement.
- Adding accessibility features, such as voice commands or switching the dark mode, helps ensure the software is accessible and reaches a wider audience.
- Allowing app settings to be altered to accommodate certain users, encouraging individualized learning experiences.
- As the program develops and gains more users, scalability, and the capacity to handle large numbers of users and data types must be guaranteed. The data can be used later for improvement and business.
- Offering premium services or subscription plans to attract customers and fund operations, sometimes through collaborations with organizations or educational institutions other than Letourneau.

5.2 Database Design

SQLite DBMS will be integrated in the LetuStudy app, which contains user’s credentials, study data and other essential data.

5.2.1 Data Objects and Resultant Data Structures

- **Flashcard**: The question and answer on each flashcard will be represented by strings referred to as the “front” string and the “back” string, respectively. These strings could be kept in a data structure like a list or an array. The two string parameters, "front" and "back", representing the question and answer, respectively.
- **Quiz**: As a quiz is made up of a series of flashcards, the data structure for the quiz is either an array or a list of flashcards. Each flashcard will have a "correct" Boolean flag to show whether the user successfully responded to the question. An array or list of flashcards, and the "correct" Boolean value will be considered as the parameters.
- **User**: Data, including the user's name, avatar, and login credentials will be kept in the user object. This information can be kept as files in an object or on a database table. The parameters includes user's name, avatar, login credentials and other information.

- **Quiz Result**: The user's score, the number of questions they properly and incorrectly answered, and the amount of time it took them to finish the quiz will all be included in the quiz result. This information can be kept in a data structure like an object or dictionary. The parameters include an integer for the number of correctly answered questions, an integer for the number of incorrectly answered questions, and a time for the total time for finishing the quiz.

5.2.2 File and Database Structures

To store and manage user data, LetuStudy requires a database. A cloud-based database or data storage solution is appropriate because the app must be able to synchronize user data with the cloud. Therefore, if the database solution is used, data distribution and network needs must be considered. The app should also handle network interruptions, allowing users to utilize it even if it was offline. There are plans to expand support to Android devices, meaning SQLite can still be used. However, Swift is not an android supported language, so there will need to be changes made to the LDM.

The file directories of LetuStudy App should look like the structure below:

StudyApp/

- Classes/
    - Flashcard.swift
    - Quiz.swift
    - User.swift
    - QuizResult.swift
- Resources/
    - StudyCards.txt
    - AvatarImages/
        - user1.png
        - user2.png
        - ...
    - ColorSets/
        - AccentColor
        - BackgroundColor
        - ...
- Views/
    - QuizView.swift
    - StudyCardView.swift

    - UserProfileView.swift
- Supporting Files/
    - Info.plist
    - AppDelegate.swift
- Tests/
    - FlashcardTests.swift
    - QuizTests.swift
    - UserTests.swift
    - QuizResultTests.swift

5.2.2.1 Database Management System Files

This section outlines the database management system (DBMS) files required for the LetuStudy app. The DBMS will be used to store user information, credentials, and study materials.

User's information will be stored in the format below with an offline/local account (All credentials except ID will be hashed with SHA512 algorithm, Password will add salt before hashing):

id	firstName	lastName	DoB	username	PW_SHA512
00000001	41fcb5b770	0e98e341b8	86df65cf9b	3eb167d5a7	84ad37613c624dg8052

Table 1 Local/Offline User Table

User's information will be stored in the format below for an OAuth account (All credentials except ID will be hashed with SHA512 algorithm, Password will not be required):

OAuth_id	firstName	lastName	DoB	username
ya29.Glins-oLtuljNVfthQU2bpJVJPTu	0d960dbdaa	71d365430e	1b4f1714f6	8f7fd15c7b7

Table 2 OAuth User Table

5.2.2.2 Non-Database Management System Files

- **Configuration files:** This may include files that store configuration parameters for the application, such as the database, app initialization, and other settings required for the application to run.

- **Log files** - This may include files that store logging information about the application such as error messages, warnings, and other diagnostic information that is helpful for troubleshooting and debugging.
- **Data files -** This may include files that store application-specific data not stored in the database**.** For example, this may include files that store user preferences, session information, or other data that needs to persist across different sessions.
- **Image files** - This may include files that store image files used in the application such as icons, logos, and other graphical elements.
- **Script files** - This may include files that store scripts used in the application such as JavaScript files used in the front-end, or Python scripts used in the back end.
- **Documentation files** - This may include files that store documentation related to the application such as user manuals, technical specifications, and other related documents.

5.3 Data Conversion

The study app will need to import and export data on a cloud database to populate it with relevant information. The cloud database stores data in a compact format that cannot be easily read by the study app to minimize the database size. Therefore, a data conversion process will be necessary to transform the data into a format that can be used by the app locally.

The data structure for cloud database includes the following fields:

- User ID
- Name
- Age
- Gender
- User Group

The data structure for the study app includes the following fields:

- User ID
- First name
- Last name
- Date of birth
- Gender
- User Group

To map the existing data to the new structure, the following transformations will need to be made:

- **User ID**: This field can be directly mapped between the local app and the cloud server.
- **Name**: The first name and last name fields in the local app would need to be combined into a single name field on the cloud server.
- **Age**: The date of birth field in the local app can be used to calculate the age and then store it as a separate field on the cloud server.
- **Gender**: This field can be directly mapped between the local app and the cloud server.
- **User Group**: This field can be directly mapped between the local app and the cloud server.

Once the fields have been mapped, the actual data conversions converting the data into a format that is suitable for transmission over the network, such as JSON or XML. Any sensitive data is encrypted before being transmitted to the cloud server to protect it from unauthorized access.

5.4 User Machine-Readable Interface

- **Students**: They are the primary users of the Study App. They use the app to access study materials, take quizzes and tests, calendar, and reminders, and communicate with instructors and other students. It is estimated that there will be thousands of students using the app, with a maximum of several hundred concurrent users at any given time.
- **Instructors**: They use the Study App to create and manage courses, upload study materials, set assignments and tests, facilitate discussions, and provide feedback to students. The number of instructors using the app is expected to be much smaller than the number of students, with a maximum of a few dozen concurrent users at any given time.
- **Administrators**: They use the Study App to manage user accounts, monitor system performance, troubleshoot issues, and perform other administrative tasks. The number of administrators is expected to be very small, with a maximum of a few concurrent users at any given time.
- **Technical Support Personnel**: They use the Study App to provide technical support to users who experience issues with the app. The number of technical support personnel is also expected to be very small, with a maximum of a few concurrent users at any given time.

5.4.1 Inputs

The LetuStudy app uses touchscreen as the main input method from the users. The AR function will utilize the camera and ToF sensors from the devices as input.

See Section 4.4.1.1.

5.4.2 Outputs

See Section 4.4.1.1.

5.5 User Interface Design

The user interface (UI) is a crucial component of the LetuStudy app, as it directly affects the user experience. The UI must be intuitive, user-friendly, and aesthetically pleasing to ensure maximum usability and user engagement.

- The LetuStudy app will feature a home screen, where users can view their login information and progress, as well as access additional features such as study cards, statistics and settings.
- The UI will also include sharing features, allowing users to share their study sets with others via cloud, files or to social media.

- The design will be optimized for both mobile and tablet devices, with responsive layouts and appropriate use of screen real estate.

To ensure that the UI meets the needs of users and provides an optimal experience, usability testing will be conducted throughout the development process. This will involve user testing sessions with individuals of various skill levels and demographics to evaluate the effectiveness of the design and identify areas for improvement.

6. Operational Scenarios

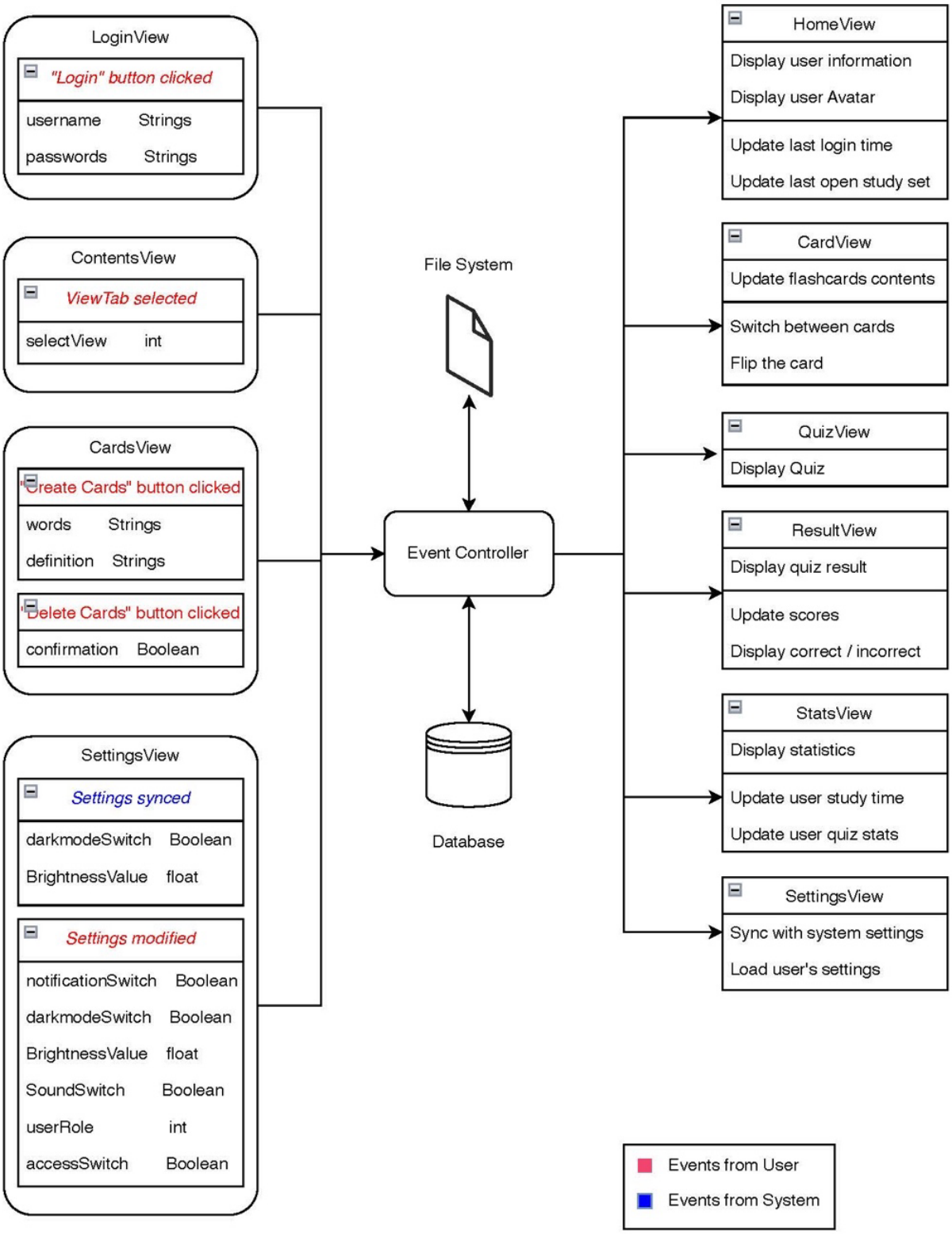

Figure 2 Event Driven Diagram

The figure above shows how the user's actions will trigger different events. The system will provide immediate feedback, whether the event succeeded or failed. As the user interacts with each button or toggle switch, the controller will consume the event and update the user or system based on the event. Should the system crash for any reason, the user should be notified with a message and not just an app closure. The app will collect the last running process and dump, log into crash report, and utilize the Apple Push Notification Service (APNS) to send notification to user's devices even the app is closed.

7. Detailed Design

7.1 Hardware Detailed Design

- The study app will be designed for use on the Apple iPhone 8, iPad 5th Gen and later models running iOS/iPadOS 16 and later versions.
- The app will require at least 2GB of RAM and 64GB of storage space on the user's device.
- The app will utilize the device's hardware components such as the camera and microphone for certain features such as AR scan and Voiceover.
- The app will be optimized for battery usage and will not drain the user's battery excessively during use. The LetuStudy app should minimize their background activity, and providing Low-Power Modes in case the device power if below a certain threshold that limits the active components and Extends widgets refresh interval.

7.2 Software Detailed Design

- **Service Identifier** – com.LetuSoftEng.LetuStudy
- **Classification** – Study Service Application
- **Definition** – The specific purpose of this service is to provide students, professors, and guest users with the ability to create and study flash cards. Some key features are statistics on time studied, scanning data into study sets, and sharing / distributing study sets.
- **Internal Data Structures** – The LetuStudy app's internal data structures include user profiles, study sets, flashcards, and progress tracking records, all of which are stored in the SQLite database.
- **Constraints** – The LetuStudy app has a minimum system version requirement for iOS/iPadOS. As a result, only iPhones and iPads that meet the specified criteria can access the app.
- **Composition** – As of now, there are no subservices in the software design. As the system grows, there may be a change in the design that merits the addition of a separate service. Currently, the application is one whole service, providing the study function to the users.
- **Users/Interactions** – The study app's system service will communicate with the cloud-based database, and due to the absence of subsystems, there will be no communication between the systems.
- **Processing** – To optimize performance and responsiveness, the app will make use of background processing and multithreading, allowing users to continue studying while the app performs background tasks such as syncing study data with the cloud. The app will also make use of caching to reduce network traffic and improve performance.

7.3 Security Detailed Design

- Authentication and Authorization: The app will allow users to use it as a guest, and an account is not necessary. However, some functions such as cloud syncing require users to create a unique login ID and password during the registration process. Passwords will be hashed and stored securely in the local database and online server. Access to certain parts of the app, such as user profile information and settings, will be restricted to authorized users only.
- Data Encryption: All sensitive user data, including login credentials and study progress, will be encrypted using industry-standard encryption algorithms such as AES or RSA. Encrypted data will be stored securely on the device and server and transmitted between the server and the app.
- Input Validation: All user input, including login credentials, study data, and user-generated content, will be validated on both the client-side and server-side to prevent SQL injection attacks and other security vulnerabilities.

7.4 Performance Detailed Design

- **Capacity and Volume Requirements**:
    - The study app is expected to have at least 100 users locally, and 10,000 users access the server simultaneously at any given time.
    - The average user session time is estimated to be around 30 minutes.
- **Performance Expectations**:
    - The app and server should be able to handle all the requests without significant delay.
    - The average response time for each request from the server should be less than 500ms, from the app should be less than 200ms.
    - The app should be able to handle peak loads such as bulk scan and upload without crashing.
- **Availability Requirements**:
    - The app should be available 24/7, with and without internet connection.
    - The server should be available for at least 99.99999% of the time.

7.5 Internal Communications Detailed Design

Communication Protocols:

The LetuStudy app uses several communication protocols to facilitate communication between its various components. These protocols include HTTP, HTTPS, and TCP/IP. The use of these protocols ensures that data is transmitted securely and reliably between components.

Interfaces:

The LetuStudy app provides several interfaces to facilitate communication between its components. These interfaces include APIs for accessing data stored in the database management system and for communicating with external services such as cloud storage

providers. The interfaces are designed to be simple and easy to use, allowing for efficient communication between components.

Security:

The LetuStudy app cares about user's security, and the internal communication architecture is designed to ensure that data is transmitted securely between components. All communication is encrypted using industry-standard encryption algorithms, and access to sensitive data is restricted to authorized users only.

8. System Integrity Controls

There are multiple security considerations that necessitate the implementation of system integrity controls for the LetuStudy app:

- Internal security should be developed to limit unauthorized users' access to sensitive data items.
- The appropriate reporting, control, and retention of operational and management reports should all be ensured via audit procedures.
- To monitor and trace retrieval access to critical data, application audit trails must be made.
- To verify data fields, standard tables should be used.
- When adding, removing, or updating vital data, verification processes should be created.

Audit data collected could include user activities (using time and functions), errors and crashes (logs), security events (succeeded and failed login attempt), system-level events (Battery level, network, etc.). The data will be collected in ways of iOS built-in frameworks (UIKit, Core Location Framework), logging, third-party analytics tools (OAuth, Google Analytics, etc.).

All audit data should be identifiable by user identification, network terminal identification, date, time, and data viewed or altered, according to the app. It is collected through an automated mechanism that tracks user activity and stores it for later analysis and monitoring, depending on the choice of how the tracking implementation is done. Authorized personnel can access this data for inspection and analysis in a centralized area, where they can utilize it to detect security concerns, examine incidents, and keep track of system performance.

For cloud database, the system integrity controls are implemented through:

- Authentication and authorization protocols to ensure only authorized users can access the database.
- Encryption of sensitive data to protect against unauthorized access or interception during transmission.
- Regular backups and disaster recovery plans to prevent data loss and ensure business continuity.
- Data validation and error checking mechanisms to ensure data accuracy and completeness.
- Access controls limit the privileges of users and prevent unauthorized modification or deletion of data.
- Monitoring and logging of all database activity to detect and respond to security threats in real-time.

Appendix A: Acronyms

Table 3 - Acronyms

Acronym	Literal Translation
AES	Advanced Encryption Standard
AR	Augmented Reality
API	Application Programming Interface
APNS	Apple Push Notification Service
AWS	Amazon Web Services
DBMS	Database Management System
HTTPS	Hypertext Transfer Protocol
LDM	Logical Data Model
RSA	Rivest-Shamir-Adleman (Encryption Algorithm)
SDD	System Design Document
TCP	Transmission Control Protocol
ToF	Time of Flight (sensor)

Appendix B: Glossary

Table 4 - Glossary

Term	Acronym	Definition
Application Programming Interfaces	API	An API is a set of functions and corresponding instructions that allows two or more applications to communicate with each other.
Database Management System	DBMS	A database management system is a computerized/electronic way of maintaining data.
Graphical User Interfaces	GUI	A GUI is a graphical way for a user to interact or interface with an application. Usually showing the users things they need and prioritizing flow of design for the user.
System Design Document	SDD	This is a system design document, and it is for the in-depth detailed design of a software system.

Appendix C: Referenced Documents

Table 5 - Referenced Documents

Document Name	Document Location and/or URL	Issuance Date
Software Requirements Specification	https://github.com/dlsxhm1/LetuStudy/blob/main/Documents/LetuStudy_SRS.pdf	02/06/2023

Appendix D: Approvals

The undersigned acknowledge that they have reviewed the SDD and agree with the information presented within this document. Changes to this SDD will be coordinated with, and approved by, the undersigned, or their designated representatives.

Table 6 - Approvals

Document Approved By	Date Approved
Name: <Name>, <Job Title> - <Company>	Date
Name: <Name>, <Job Title> - <Company>	Date
Name: <Name>, <Job Title> - <Company>	Date
Name: <Name>, <Job Title> - <Company>	Date