

Software Design

Specification

( Detailed )

**26 March 2015, V2**

**Team3:**

Derek Grove

Douglas Harshberger

Dylan Small

Evan Connors

Michael Nazzario

Table of Contents

**I. Introduction 3**

*I.I       → Purpose*

*I.II      → Scope*

*I.III     → Intended Audience*

**II. System Overview 4**

*II.I → System Architecture*

*II.II → System Explanation*

**III. Design Considerations 5**

*III.I → Assumptions and Dependencies*

*III.II → General Constraints*

*III.III → Goals and Guidelines*

*III.IV → Development Methods*

**IV. Architectural Strategies 6**

*IV.I → Engineering Paradigm*

*IV.II → System Hierarchy (Packaging)*

*IV.III → System IO*

**V. Class Design 7**

*V.I → High-Level Class Diagram*

*V.II → Detailed Class Diagram*

*V.III → Class Explanation*

**VI. Processes      16**

*VI.I → High-Level Sequence Diagram*

*VI.II → High-Level Sequence Diagram Explanation*

*VI.III → Manage Sequence Diagram*

*VI.IV → Manage Sequence Diagram Explanation*

*VI.V → Practice Sequence Diagram*

*VI.VI → Practice Sequence Diagram Explanation*

*VI.VII → Quiz Sequence Diagram*

*VI.VIII → Quiz Sequence Diagram Explanation*

**VII. Database Design     20**

*VII.I → DB Relational Schema (High-Level)*

*VII.II → DB Schema Definitions*

*VII.III → DB E-R Diagram*

*VII.IV → DB Glass-Box Explanation*

**VIII. Data Flows      24**

*VIII.I → Context Data Flow Diagram*

*VIII.II → Context Data Flow Explanation*

*VIII.III → Level 1 Data Flow Diagram*

*VIII.IV → Level 1 Data Flow Explanation*

*VIII.V → Level 2 Data Flow Diagram*

*VIII.VI → Level 2 Data Flow Explanation*

**IX. GUI      28**

*IX.I → Login Case Diagram*

*IX.II → Login Explanation*

*IX.III → Manage Case Diagram*

*IX.IV → Manage Explanation*

*IX.V → Practice Case Diagram*

*IX.VI → Practice Explanation*

*IX.VII → Quiz Case Diagram*

*IX.VIII → Quiz Explanation*

**X. Glossary      36**

*→ Terms and Definitions*

I. Introduction

***.I Purpose:***

The purpose of this document is to be a reference to any person who wants to implement or any person interested in the architecture of the Flash Card application. This document describes the application’s architecture and each sub-architecture along with their associated interfaces, database schemas, and the motivations behind the chosen design.  Both high-level and detailed-level designs are included in this document.

In order to properly understand this document one should have a technical background and be able to interpret Data Flow Diagrams (DFD) as well as Unified Modeling Language (UML) diagrams.

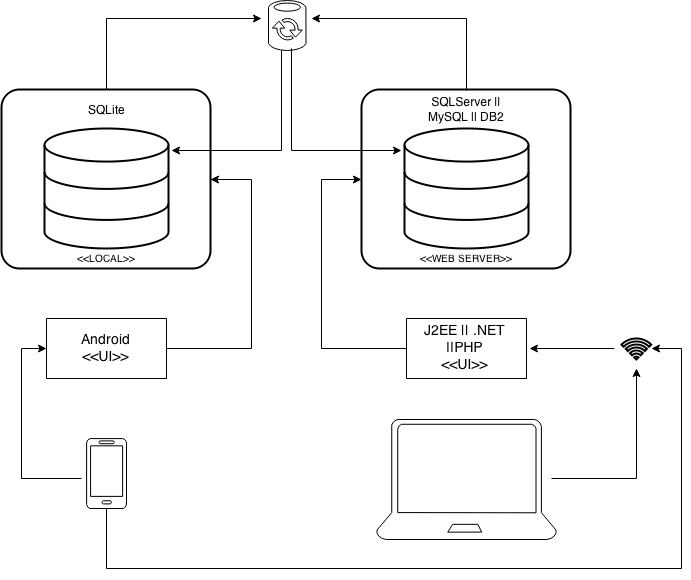
***.II Scope:***

The scope of this document will include the architecture and design of the Flash Card application. This application is to provide a student with a flash card based quiz and assist with the memorization of terms that an instructor or an administrator inputs.

***.III Intended Audience:***

The intended audience of this document includes the technical team currently developing the application, managers, Dr. Yongtaek (current client), and future developers of the application.

II. System Overview  
***.I System Architecture:***

***[](https://drive.draw.io/#G0B3Y_Ai05HOYwQ3pJbGZuLWxfWVk)***

***.II System Architecture Explanation:***

There will be two UIs. One interface which will be the client-side viewing UI will be that with an Android device and the app installed. This system will use a local DB, SQLite, to view and manipulate data. The other UI will be a server-side language UI to a server DB. This will be the primary way for admins to go in and manipulate their databases in which the clients can sync up with the server. The server-side UI and DB are undecided currently, however it is probable that the UI will be a PHP web-page to a MySQL DB.

III. Design Considerations

***.I Assumptions and Dependencies:***

Assuming that we will have the DBMS on the internet, we will need the user to have an internet connection in order to synchronize with the server DB. The admin/teacher should have a desktop or laptop in which they can ping the admin web portal to manipulate the server DB in a efficient manner. However data is able to be manipulated from the application, in which can be synced from local to server.

***.II General Constraints:***

Our biggest constraint will be that the application is only for the Android operating system. Time will also be a constraint, we only have until the end of the semester to complete the application for the client so that he may present the application at a conference.

***.III Goals and Guidelines:***

Our main goal is to generalize this application and implement more features so that the student and admin will have a better experience overall with the application. We want to make the Flash Card application into a tool for future professors to use in their classes, in an arbitrary fashion.

***.IV Development Methods:***

This project will utilize a paradigm with 3 builds: Alpha, Beta, Gamma. Each time becoming more stable and robust with more features implemented. We will make the final product very user intuitive. Also allowing the Admin to easily access data and options to edit the data.

IV. Architectural Strategies

***.I Engineering Paradigm:***The architectural design of this application uses the Object Oriented Paradigm (OOP). Android application are coded in Java along with XML referencing. Using an Object Oriented language like Java allows us to easily design and implement modules. We will also be using SQLite for the local DB. The database can currently be edited/updated using the SQLiteBrowser tool, which provides a graphical and tabulated user interface for the database.

The web management portal will most likely be coded in the PHP language which is also an O-O language. The server-side DB that the PHP web-page will provide a UI for will be comprised with MySQL.

***.II System Hierarchy (Packaging):***

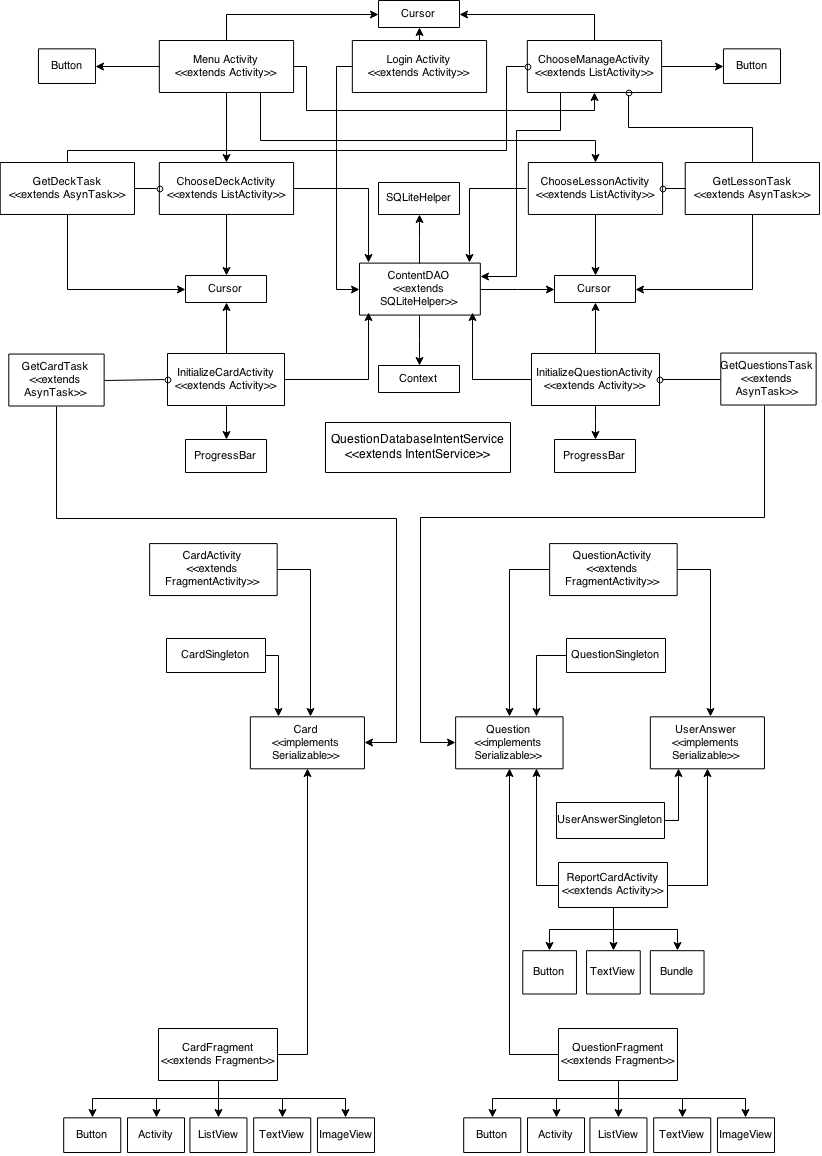
The Java developer-generated source-code is organized into three different packages. The JapaneseFlashCard package contains java classes pertaining to interactions with GUIs. The JapaneseFlashCard.dao package contains classes that provide an interface between the SQLite database and the android application. The JapaneseFlashCard.Models package provides views of the data.

***.III System IO:***

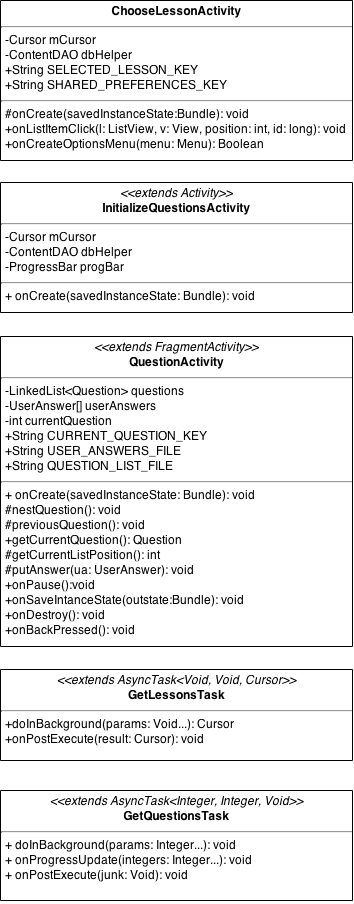
Currently, all application data will reside on the android device that it is installed on. The application requires no internet access to function. The SQLite database will lie locally within the application, so access to a remote database is only required if content has been manipulated since the application was installed.

V. Class Design

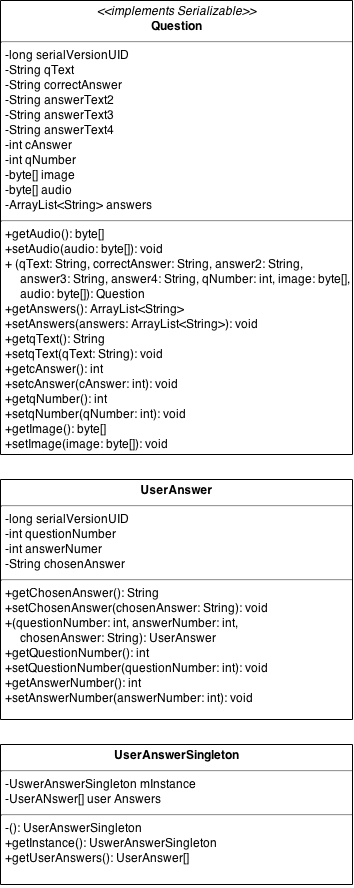
***.I High-Level Class Diagram:***

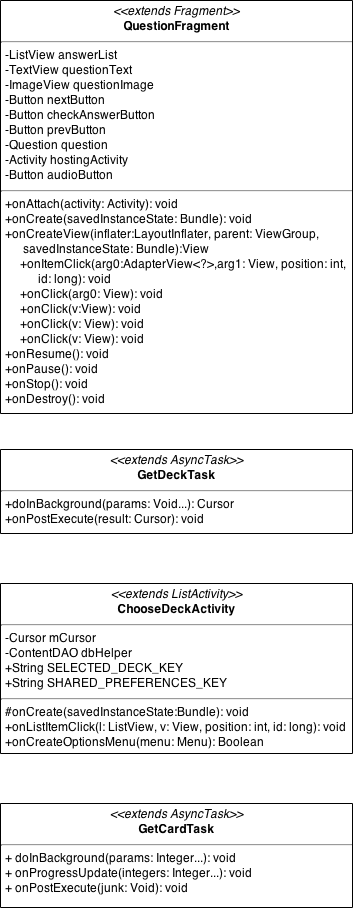
***[](https://drive.draw.io/#G0B3Y_Ai05HOYwRHF0TGotMXE4Qms)***

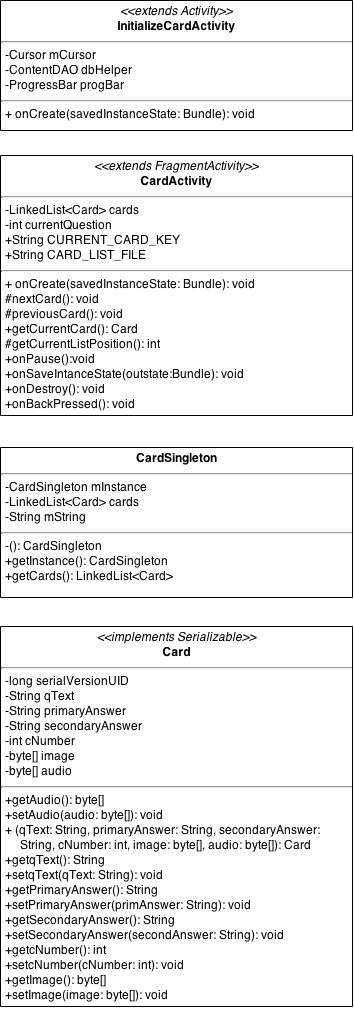
***.II Detailed Class Diagram:***

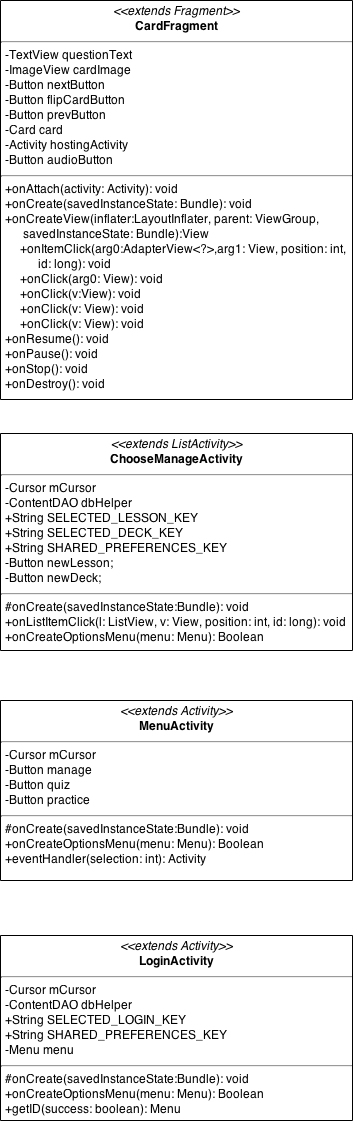
***[](https://drive.draw.io/#G0B3Y_Ai05HOYwWktISDZlVy1ubWs)***

***[](https://drive.draw.io/#G0B3Y_Ai05HOYwWktISDZlVy1ubWs)***

***[](https://drive.draw.io/#G0B3Y_Ai05HOYwWktISDZlVy1ubWs)***

***[](https://drive.draw.io/#G0B3Y_Ai05HOYwWktISDZlVy1ubWs)***

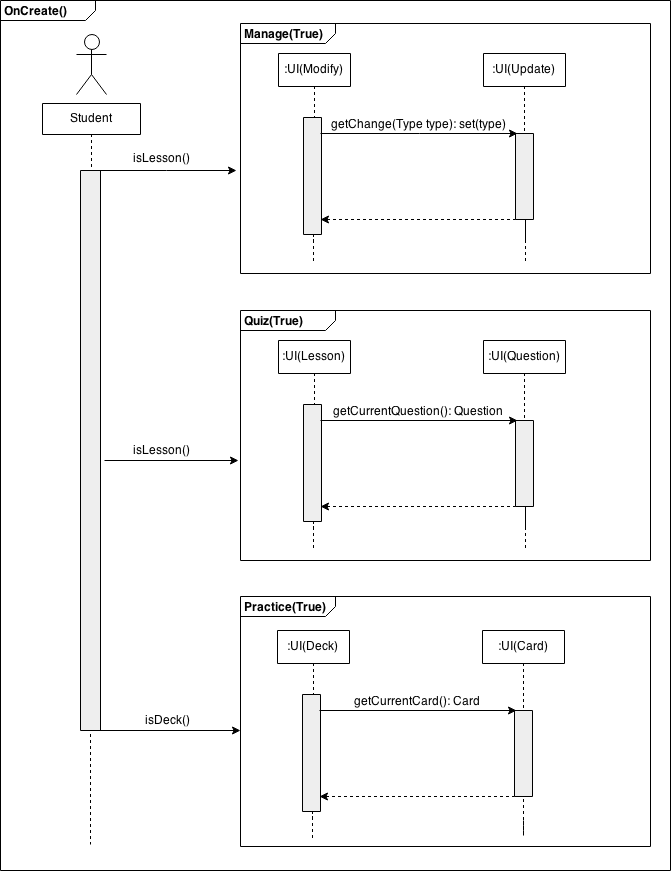
***[](https://drive.draw.io/#G0B3Y_Ai05HOYwWktISDZlVy1ubWs)***

***[](https://drive.draw.io/#G0B3Y_Ai05HOYwWktISDZlVy1ubWs)***

***.III Class Explanation:***

* LoginActivity is the screen a user will be brought to when initially launching the application if away for a long period of time. User logs in which will determine the credentials as a student or admin/teacher.
* MenuActivity is the home screen that is initially launched when the user logs in. Also serves as the home page that is accessible by an action bar shortcut.
* ChooseManageActivity is the first thing to appear when the manage avenue is selected. It displays the list of lessons and decks.
* ChooseLessonActivity is the first thing to appear when the quiz avenue is selected.  It displays a list of lessons which can be chosen by the user.
* InitializeQuestionsActivity gathers all of the questions for the selected lesson, displaying a spinning progress indicator in the meantime.
* QuestionActivity keeps track of the current question in the lesson set, the student’s recorded answers, and other information relevant to the session.
* GetLessonsTask returns the lesson chosen by the user so that the program knows which set of questions to retrieve.
* GetQuestionsTask is the specific entity tasked with gathering the questions for a given lesson.
* ReportCardActivity calculates and displays the student’s performance on the lesson’s question set.
* ContentDAO deals with all direct interactions between the database and the rest of the program.
* QuestionDatabaseIntentService handles internal database updates by connecting to a remote server.
* QuestionSingleton creates a container for a set of questions.
* Question defines the format of all the information involved in a single question.
* UserAnswer defines the format for a record of a student’s answer to a given question.
* UserAnswerSingleton creates a container for all of the student’s answers to questions.
* QuestionFragment sets up and organizes the components on the screen when a question is being displayed.
* ChooseDeckActivity is the first thing to appear when the practice avenue is selected.  It displays a list of decks which can be chosen by the user.
* InitializeCardsActivity gathers all of the cards for the selected deck, displaying a spinning progress indicator in the meantime.
* CardActivity keeps track of the current card in the deck set, and other information relevant to the session.
* GetDeckTask returns the deck chosen by the user so that the program knows which set of cards to retrieve.
* GetCardsTask is the specific entity tasked with gathering the cards for a given deck.
* CardSingleton creates a container for a set of cards.
* Card defines the format of all the information involved in a single card.
* CardFragment sets up and organizes the components on the screen when a card is being displayed.

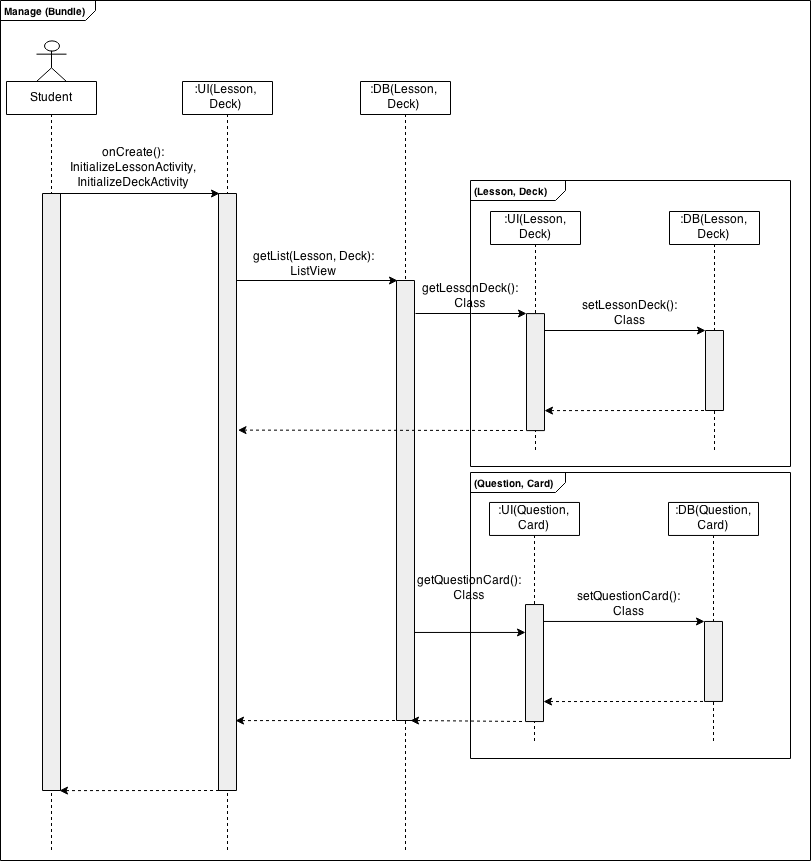
VI. Processes

***.I High-Level Sequence Diagram:******[](https://drive.draw.io/#G0B3Y_Ai05HOYwU0NuN0xyempiQlk)***

***.II High-Level Sequence Diagram Explanation:***

The Flash-Card application is broken down into three main processes. They are, manage, practice and quiz. Once logged in, a user may enter any of the portals. Manage serves as a way for the user to manipulate the data stored on the database. Practice serves as a way for users to view the content in a non-weighted system. Quiz serves as a way for users to view the content in a weighted system that prints out a report at the end.

***.III Manage Sequence Diagram:***

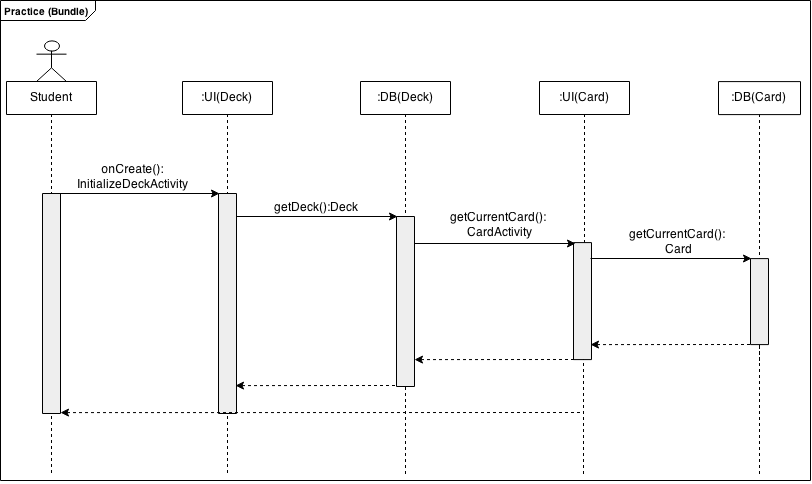
***[](https://drive.draw.io/#G0B3Y_Ai05HOYwSTE0YlEyRktDMTQ)***

***.IV Manage Sequence Diagram Explanation:***

There are two management portals, one for an admin and one for a general user. The general user may manipulate/delete/add any data pertaining to the practice avenue. The user is able to upload pictures and audio from their phone as well as put a primary and secondary answer. Those changes if saved will be written to the local DB which can then later be written to the server DB with an internet connection

The admin portal has all of the same capabilities as well as the ability to manage the quiz track. The lessons and questions on the quiz track are the same for everyone unlike the practice track being that of a client-side UI. The admin can manipulate/delete/add any data pertaining to both avenues. The local DB will not automatically synchronize with the server DB until the admin commits those changes. This is in case of a malfunction or human-error, the admin can sync with the server DB if changes to the local were unintended.

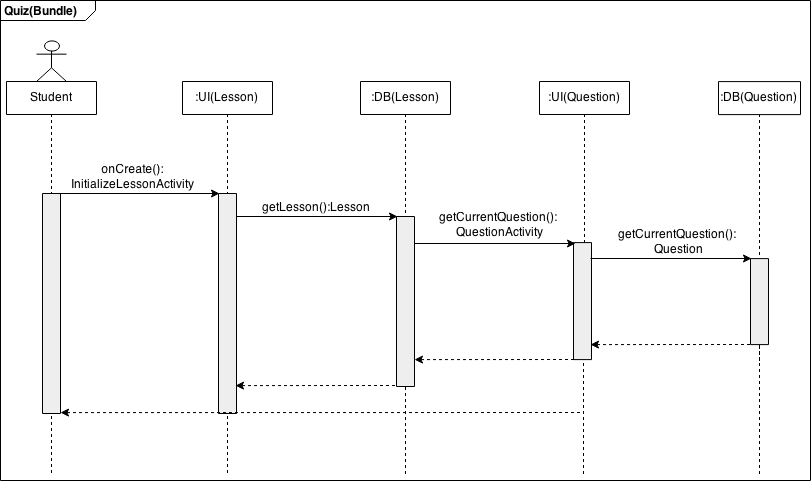
***.V Practice Sequence Diagram:***

***[](https://drive.draw.io/#G0B3Y_Ai05HOYwUDg5bmtGU0hoa1k)***

***.VI Practice Sequence Diagram Explanation:***

If the practice avenue is selected in the home screen, a user will be prompted with a list of custom and admin made decks from the local DB. A deck contains cards, which will be displayed if their parent (deck) was chosen by the user. The content will be filled programmatically from selecting fields from the Card table in the local DB.

***.VII Quiz Sequence Diagram:***

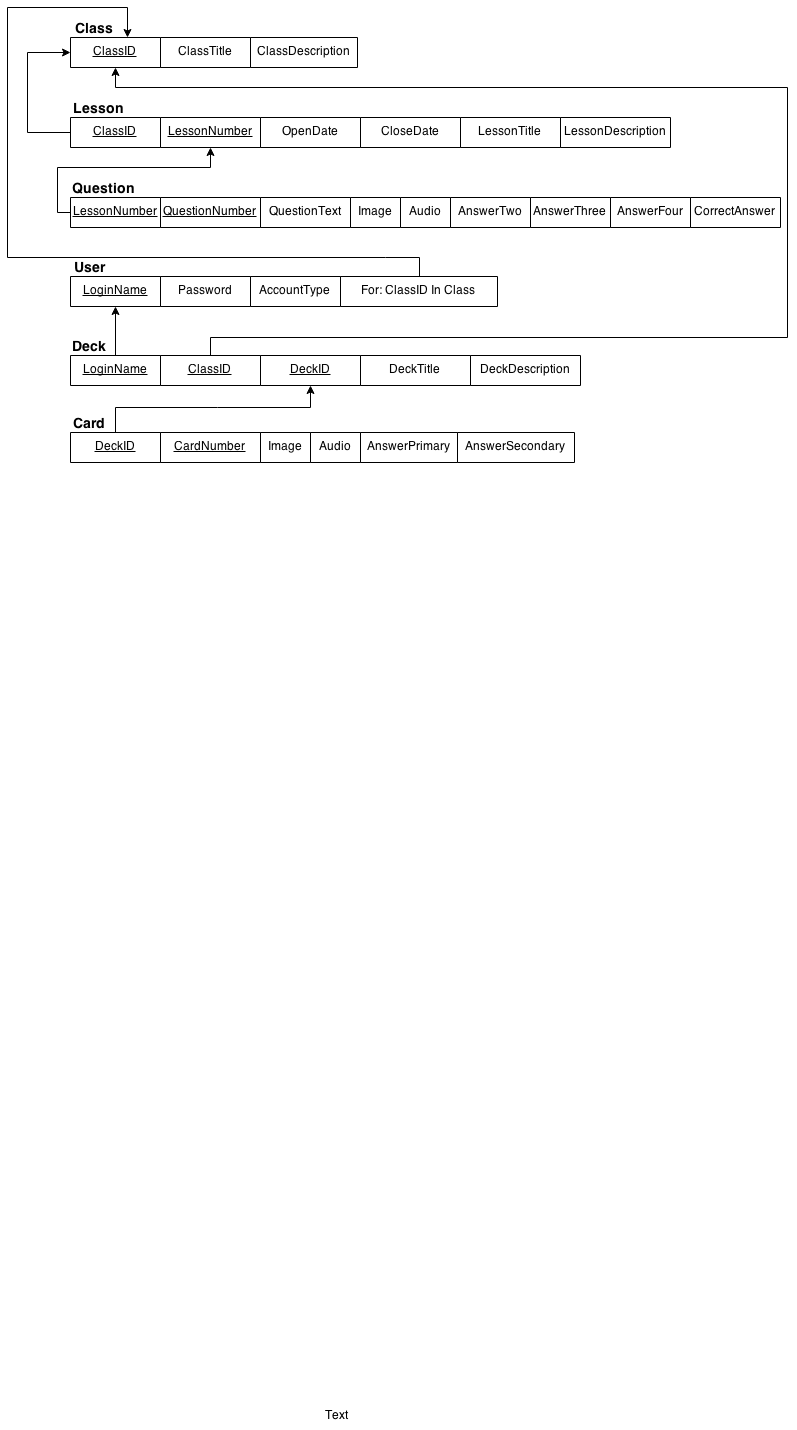
***[](https://drive.draw.io/#G0B3Y_Ai05HOYwUTFvUmVobl9WYzg)***

***.VIII Quiz Sequence Diagram Explanation:***

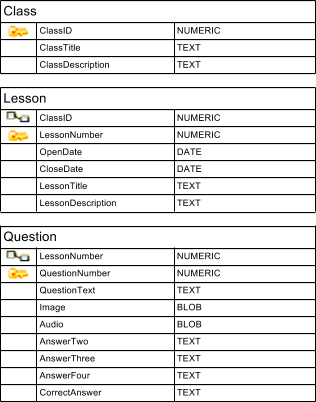
If the quiz avenue is selected in the home screen, a user will be prompted with a list of admin made lessons from the local DB. A lesson contains questions, which will be displayed if their parent (lesson) was chosen by the user. The content will be filled programmatically from selecting fields from the Question table in the local DB.

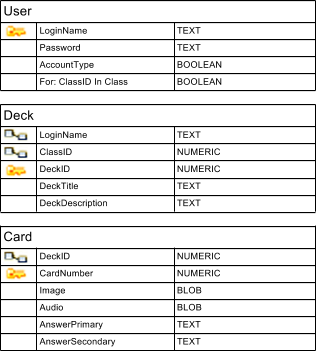
VII. Database Design

***.I DB Relational Schema:***

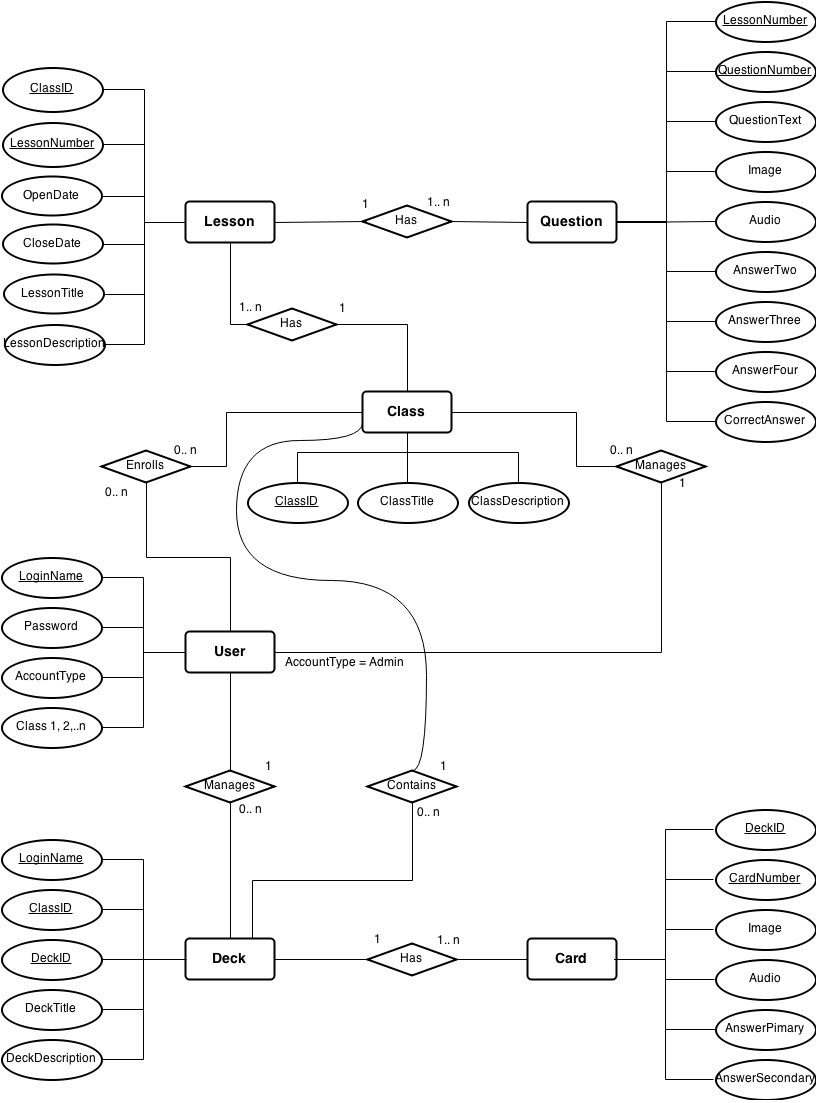
***[](https://drive.draw.io/#G0B3Y_Ai05HOYwV3BXLVNhY2dWTHc)***

***.II DB Schema Definitions***

******

******

***.III DB E-R Diagram:***

***[](https://drive.draw.io/#G0B3Y_Ai05HOYwak5KLUhiTk5NSjQ)***

***.IV DB Glass-Box Explanation:***

The current database consists of six different tables. The first table being a User consists of four attributes. LoginName which is type TEXT is the primary key which shall uniquely identify a user. To confirm a User, a Password attribute of type TEXT is checked. There is an AccountType, type BOOLEAN, which identifies if the User is an admin type or not. The last is the flagged Classes that the User is enrolled in.

The second table, titled Class, has three attributes. The first attribute is ClassID, which is NUMERIC and will consist of three numbers to uniquely identify a Class. The second attribute is ClassTitle which is type TEXT. ClassDescription describes the class in a TEXT format.

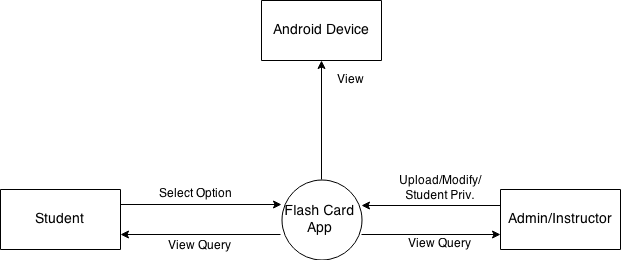
The third, titled Lesson, has six attributes. The first attribute is LessonNumber, which is type NUMERIC. The second attribute is ClassID which is also NUMERIC. LessonNumber and ClassID serve as the primary keys for Lesson, which uniquely identifies each lesson. The third attribute, LessonTitle, is type TEXT, and the fourth, LessonDescription, is type TEXT. There is also a OpenDate and CloseDate attributes which are both defined as a DATE. Each individual Lesson row has many Questions associated with it. Also, ClassID is a foreign key that is synced with ClassID in the Class table to ensure referential integrity.

The  fourth table in the database, titled Question, which has nine different attributes. The primary key of Question is a combination of QuestionNumber and LessonNumber, which are both type NUMERIC. QuestionText is type TEXT, Image is type BLOB, and Audio is type BLOB. AnswerTwo, AnswerThree, Answer Four, and Correct Answer are type TEXT. Every row in the Question table is associated with only one Lesson. Also, LessonNumber is a foreign key that is synced with LessonNumber in the Lesson table to ensure referential integrity.

The fifth, titled Deck, has three attributes. The first attribute is DeckNumber, which is type NUMERIC. DeckNumber is the primary key of Deck, which uniquely identifies each deck. The second attribute, DeckTitle, is type TEXT, and the third, DeckDescription, is type TEXT. Each individual Deck row has many Cards associated with it.

The sixth table in the database, titled Card, which has six different attributes. The primary key of Card is a combination of CardNumber and DeckNumber, which are both type NUMERIC. QuestionText is type TEXT, Image is type BLOB, and Audio is type BLOB. AnswerPrimary, and AnswerSecondary are type TEXT. Every row in the Card table is associated with only one Deck. Also, DeckNumber is a foreign key that is synced with DeckNumber in the Deck table to ensure referential integrity.

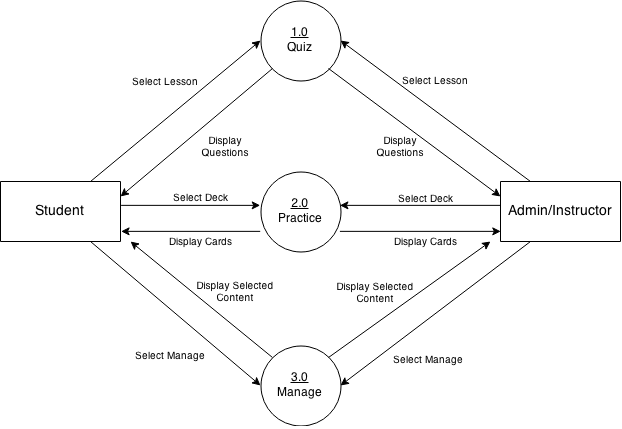
VIII. Data Flows

***.I Context Data Flow Diagram:***  
[](https://drive.draw.io/#G0B3Y_Ai05HOYwb1Utc2pmdW40UXc)

***.II Context Data Flow Explanation:***

The main application will be run on an Android device. There will be two types of users (Student and Admin). A student is able to manipulate local DB data which can be synced with the server. The main function of a student however is to view the data from the local DB. It only needs synced with the server DB if an admin changes the DB. The main function of the admin is to manipulate and manage the data, and manage who can enter their class.

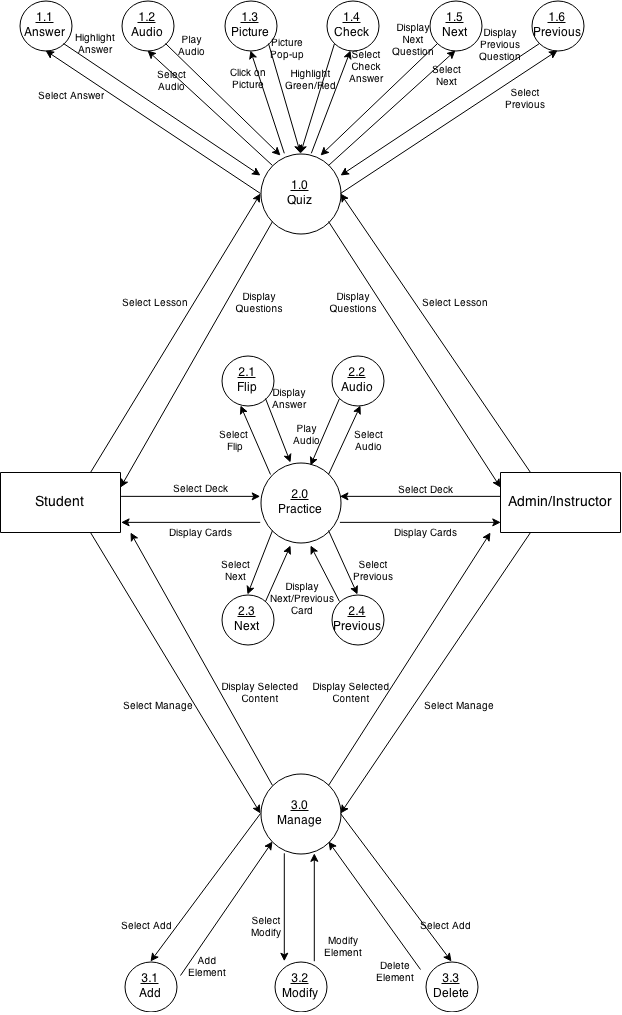
***.III Level 1 Data Flow Diagram:***

***[](https://drive.draw.io/#G0B8wVQUrDBEuibEM4aXJEcXlUQVU)***

***.IV Level 1 Data Flow Explanation:***

There are two main streams of data that is to manipulate or view for this application. The views will differ depending on if the user chooses option 1, 2, or 3. Option 1 presents the user with admin-made content. Option 2 presents the user with admin-made and self-made data if the user made a custom deck. Option 3 presents the user with both lists of decks and lessons (lessons are able to be manipulated if and only if the user is of type admin).

***.V Level 2 Data Flow Diagram:***

***[](https://drive.draw.io/#G0B3Y_Ai05HOYwellST3UwUm50alk)***

***.VI Level 2 Data Flow Explanation:***

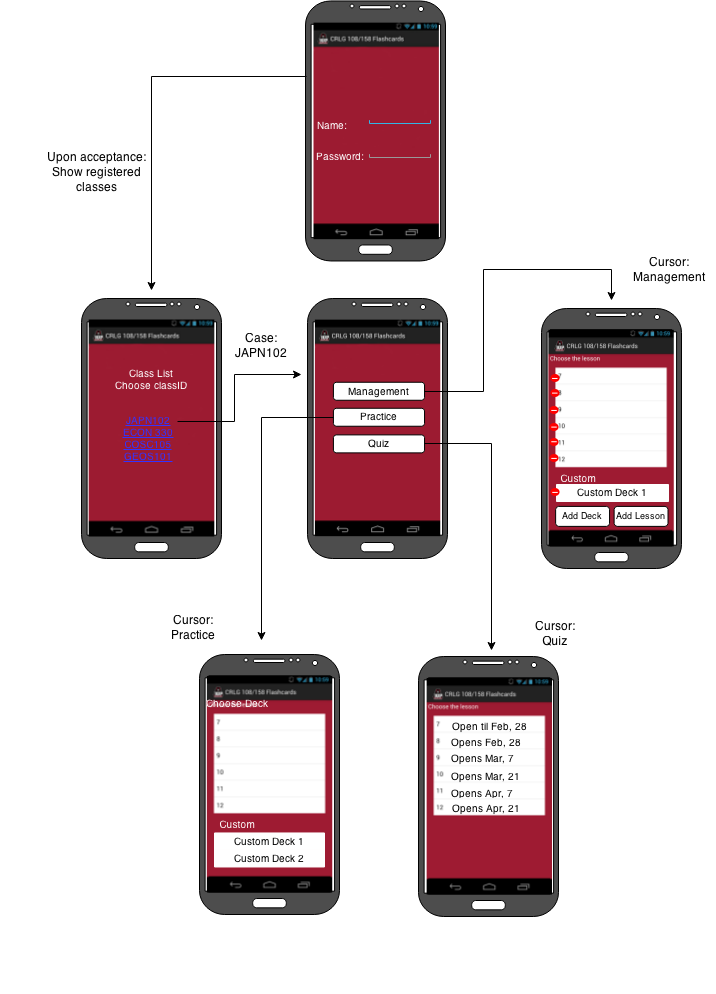
When the Quiz portal is selected, the user will be presented with a list of lessons for that class. Based upon the selection of the lesson, questions will be queued in for it. The first question will populate the screen unless there was a saved instance. Based upon the operation the user selects, will determine the interaction of the data.

When the Practice portal is selected, the user will be presented with a list of decks for that class. Based upon the selection of the deck, cards will be queued in for it. The first card will populate the screen unless there was a saved instance. Based upon the operation the user selects, will determine the interaction of the data.

When the Manage portal is selected, the user will be presented with a list of decks and lessons for that class. From here a user may edit the lesson or deck properties or even add or delete a lesson or deck. Based upon the selection of the deck or lesson, cards or questions [respectively] will be queued in for it. The list of cards or questions may be individually modified or deleted. User may also add to the list. Admin of the class may edit quiz portal data as well as practice portal data, which a student only has modifying privileges to.

IX. GUI

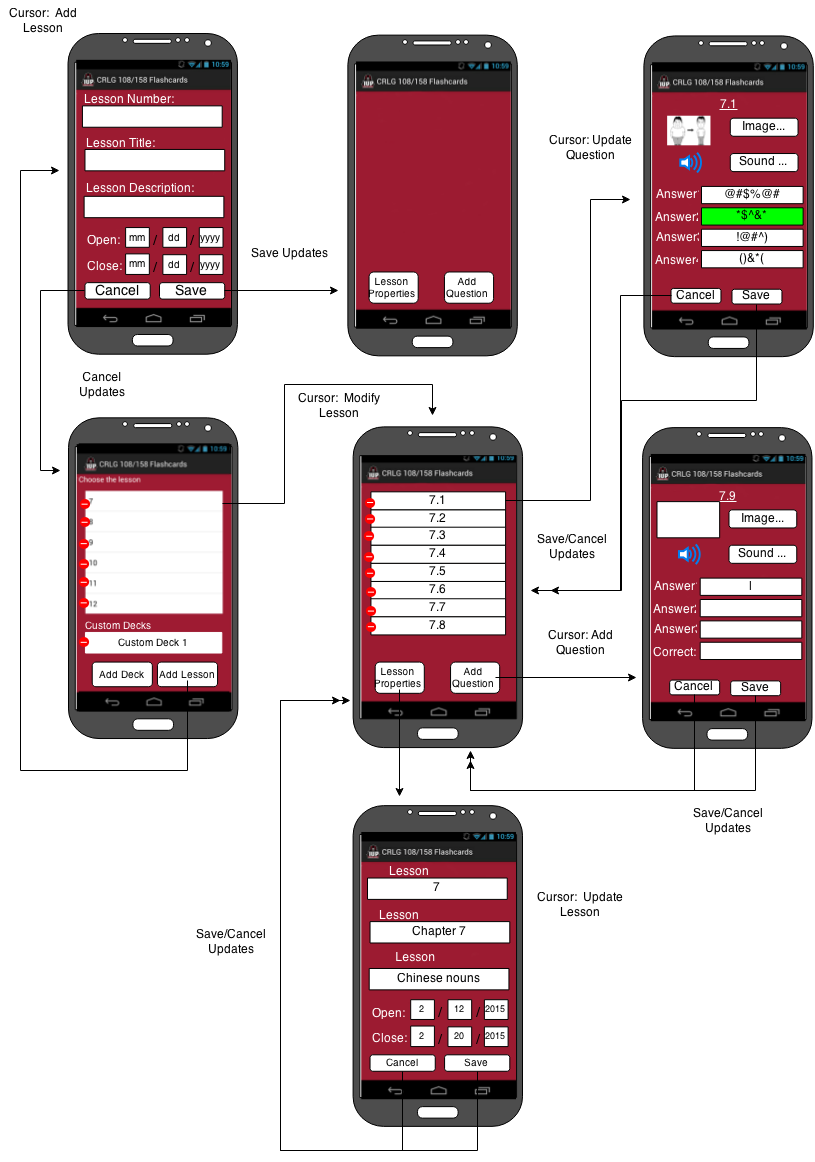
***.I Login Case:***

***[](https://drive.draw.io/#G0B3Y_Ai05HOYwMlM0UVdSWlRsUWc)***

***.II Login Explanation:***

Once opening the app you will be prompted with a login screen where the user will input his/her credentials to access the app. Once logged in the user may choose which class they want to access (only classes currently enrolled in will display). The user will then be prompted the home screen which will have a shortcut actionbar item referencer, and provide the user with the three main functions. Login will only be necessary if the application has been idle for some time, or is the first time opening up the app.

***.III Manage Case:***

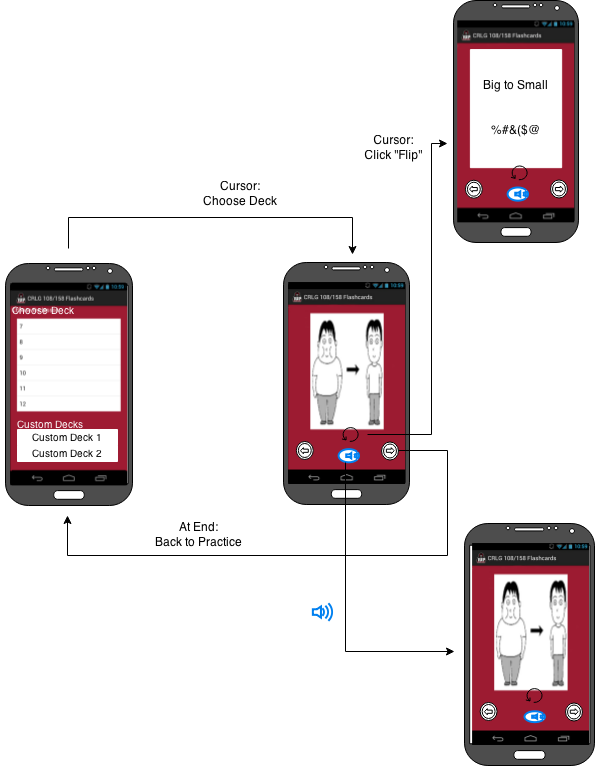
***[](https://drive.draw.io/#G0B3Y_Ai05HOYwUmNfdFFEMkw5SzQ)***

***.IV Manage Explanation:***

When in the management section of the app the user, if admin, has the ability to modify/delete/insert lessons and decks (students have these abilities for decks only). Since there is a general hierarchy of lesson to question and deck to card, all modifications to the parent (lesson or deck) will be done in the initial screen. Each attribute will be filled to ensure the DB will have valid content which will be required in order for the save functionality to work. If saved then the changes will be written to the DB, otherwise cancel the modification.

If a specific deck or lesson is selected, the connection to the DB will provide a view of the cards or questions [respectively], for it. A user may delete from the list from the list view, or add an item to the list from the list view. An item may be modified if chosen, in which the user will be presented all fields that are present in the DB. When the changes are done, if saved and completed, the changes will be written to the DB, otherwise the modifications will be canceled.

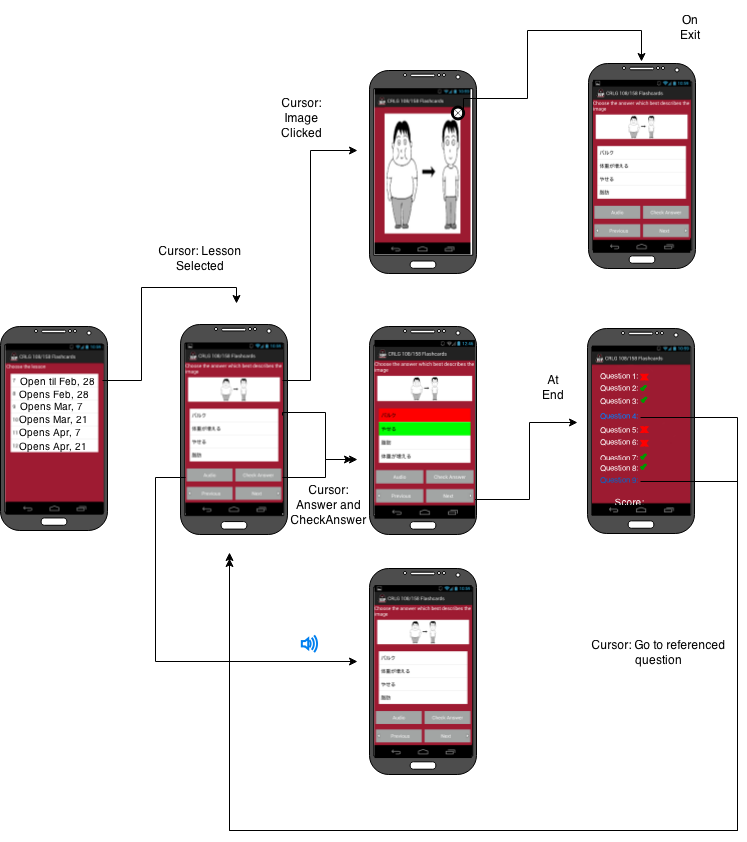
***.V Practice Case:***

***[](https://drive.draw.io/#G0B3Y_Ai05HOYwa0NSdE1NTWxXOXc)***

***.VI Practice Explanation:***

When in the practice portal, a user will be presented with a list of decks available to view. Admin-made decks will be available as well as client-made decks that were instantiated in the management portal. The first card will be instantiated given the deck, or the last saved card. Once that view is provided, a user will be shown the image, in which they may flip the card which will show them the primary language and the secondary language interpreted translations. A user has the ability to listen to the audio associated with the image to help assist the learning process. A user will traverse through the list of cards by using the next and previous buttons.

***.VII Quiz Case:***

***[](https://drive.draw.io/#G0B3Y_Ai05HOYwWTl5U1pZLTRmNTQ)***

***.VIII Quiz Explanation:***

When in the quiz portal, a user will be presented with a list of lessons available to view. Admin-made lessons will be available. The first question will be instantiated given the lesson chosen, or the last saved question. Once that view is provided, a user will be shown the image as well as the possible answers for the question. A user has the ability to listen to the audio associated with the image to help assist the learning process. Once a user is ready to answer the question, they may pick from the array of answers in which the app will highlight the answer chosen. A user will then be able to check the answer in which the app will highlight their answer red if wrong and highlight the correct answer in green. If the selection was correct then it will be highlighted green. A user will traverse through the list of questions by using the next and previous buttons.

X. Glossary

**Class Diagram** – An illustration of the relationships and source code dependencies among classes drawn/written in the UML.

**DB (Database)** – A storing of data for a system, accessible in various ways.

**DBMS (Database Management System)** – Controls the security and integrity of a database, as well as the controlling of organization, storage, and retrieval of data in the database.

**DFD (Data Flow Diagram)** – Often used as a means of overviewing the system graphically and preliminary. Provides a graphical representation of the flow of data in a system, modeling its process and aspects.

**E-R (Entity-Relationship) Diagram** – A graphical representation of objects or concepts within a system or organization, and their relationship to one another.

**GUI (Graphical User Interface)** – An interface that allows the user a graphical representation of the system with an electronic device.

**IO (Input/Output)** – Declares any operation, program, or device that transfers data to or from a system.

**Relational Schema** – A logical definition of a table; the table name, and the names of each column.

**Sequence Diagram** – A representation of object interactions within a given time, showing how processes operate with one another and what is in their order.

**UI (User Interface)** –A view in which the end-user sees.

**UML (Unified Modeling Language**) – A general purpose modeling language in the field of software engineering, providing a standardized way of visualizing a system.