

Software Design

Specification

( High-Level )

**26 February 2015, V1**

**Team3:**

Derek Grove

Douglas Harshberger

Dylan Small

Evan Connors

Michael Nazzario

Table of Contents

**I. Introduction 2**

*I.I → Purpose*

*I.II → Scope*

*I.III → Intended Audience*

**II. System Overview 3**

*II.I → System Context Diagram*

*II.II → System Context Explanation*

**III. Design Considerations 4**

*III.I → Assumptions and Dependencies*

*III.II → General Constraints*

*III.III → Goals and Guidelines*

*III.IV → Development Methods*

**IV. Architectural Strategies 5**

*IV.I → Engineering Paradigm*

*IV.II → System Hierarchy (Packaging)*

*IV.III → System IO*

**V. System & Architectural Design 6**

*V.I → High-Level Class Diagram*

*V.II → Detailed Class Diagram*

*V.III → Class Diagram Explanation*

*V.IV → High-Level Sequence Diagram*

*V.V → Sequence Diagram Explanation*

*V.VI → DB Relational Schema*

*V.VII → DB E-R Diagram*

*V.VIII → DB Glass-Box Explanation*

*V.IX → GUI Cases Diagram*

*V.X → GUI Explanation*

**VI. Glossary 15**

*→ Terms and Definitions*

I. Introduction

***.I Purpose:***

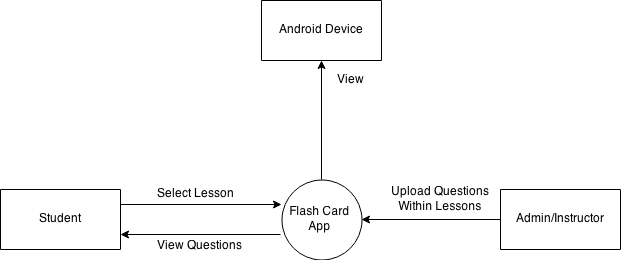
The purpose of this document is to be a reference to any person who wants to implement or any person interested in the architecture of the Flash Card application and its respective database. This document describes the application’s architecture and each sub-architecture along with their associated interfaces, 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. Kim (current client), and future developers of the application.

II. System Overview  
***.I System Context Diagram:***  


***.II System Context Explanation:***

This application will be run on the Android operating system. The database will be online, and will be accessible via the application. The online database questions and answers will have the ability to be edited via mobile device in future implementation. Admin has the ability to edit the data in the database as well as all the students’ capabilities. The student has the ability to view the data through the application.

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 access the database. The admin/teacher shall also have SQLiteBrowser properly installed on the system in which they plan to edit the data.

***.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, and 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 is Object Oriented Programming (OOP). We will be using java as our language of choice. Using an Object Oriented language like Java allows us to easily design and implement modules. We will also be using SQLite for the database and database management system (DBMS). The database can currently be edited/updated using the SQLiteBrowser tool, which provides a graphical and tabulated user interface for the database.

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

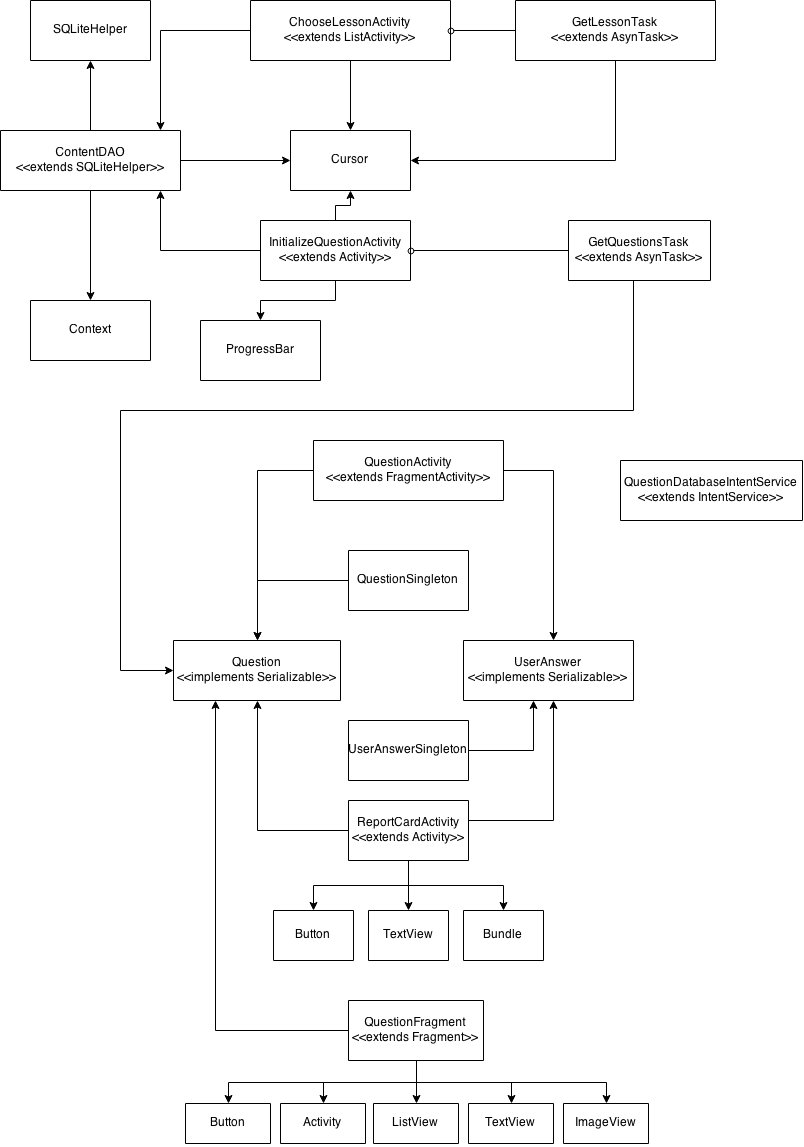
The Java user-generated source-code is organized into three different packages. The JapaneseFlashCard package contains java classes pertaining to interactions with GUIs. The JapaneseFlashCard.dao package contains classes that provide an interface between the SQLite database and the android application. The JapaneseFlashCard.Models package holds information pertaining to lesson questions and answers.

***.III System IO:***

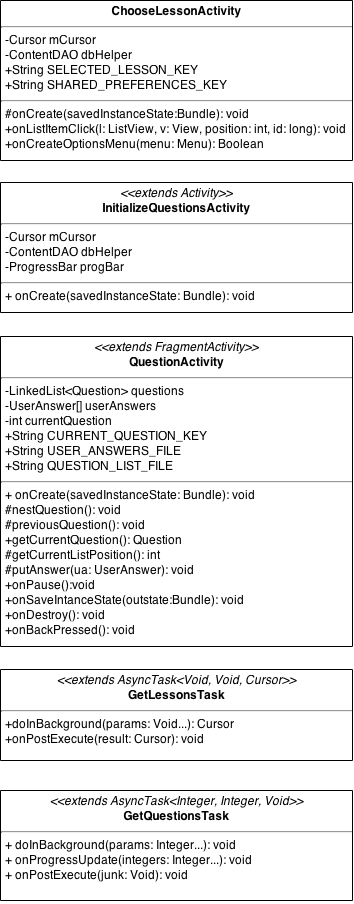
Currently, all application data will reside on the android device that it is installed on. The application requires no internet access to function. The database will lie locally within the application, so access to a remote database is not required.

V. System & Architectural Design

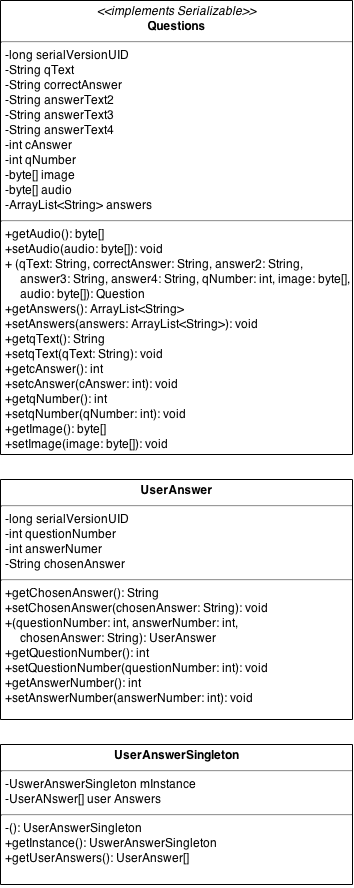
***.I High-Level Class Diagram:***

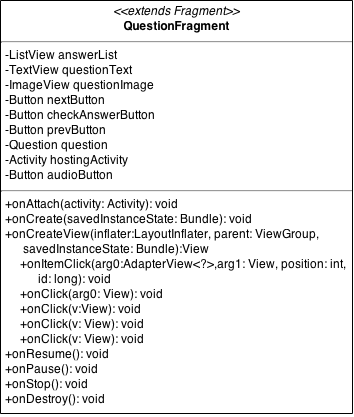


***.II Detailed Class Diagram:***





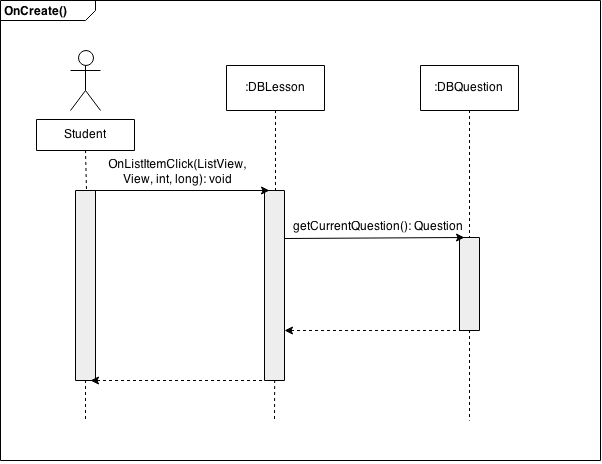




***.III Class Diagram Explanation:***

* ChooseLessonActivity is the first thing to appear when the app is launched. 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.

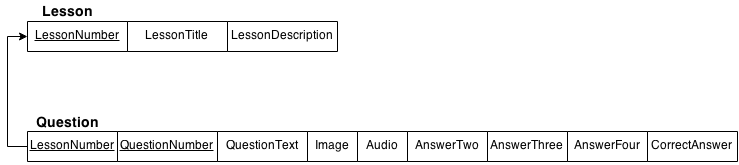
***.IV High-Level Sequence Diagram:***



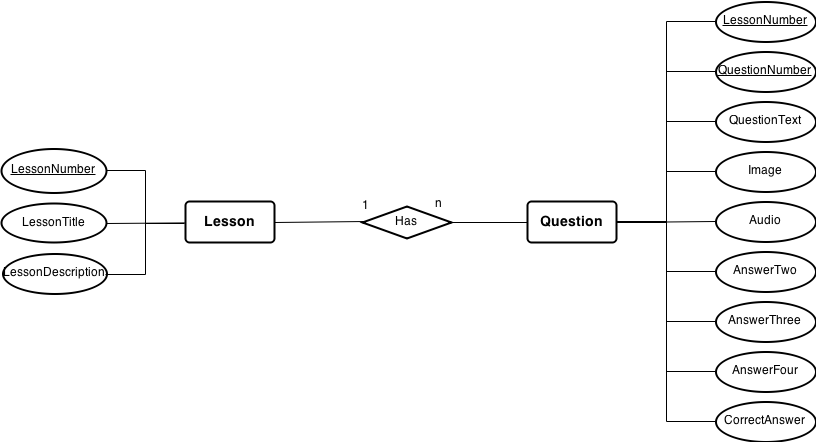
***.V Sequence Diagram Explanation:***

Once the application is started, it looks for a saved location in which the user has cached on their device. If there is no starting point (i.e. question they were on) the system loads up a list of lessons. A user selects a lesson from the list which is populated by the Lesson table of the database, in which then loads the frame with the first question which is populated by the Question table of the database. The questions are in a linked list in which the previous or next question is shown based on the event instantiated.

***.VI DB Relational Schema:***



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

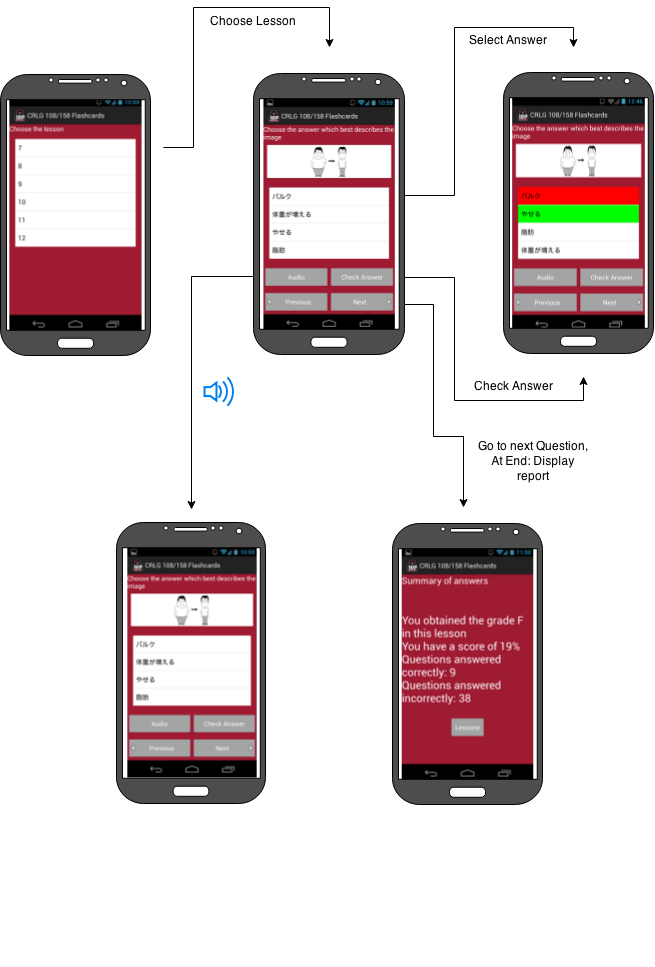


***.VIII DB Glass-Box Explanation:***

The current database consists of two different tables. The first, titled Lesson, has three attributes. The first attribute is LessonNumber, which is type NUMERIC. LessonNumber is the primary key of Lesson, which uniquely identifies each lesson. The second attribute, LessonTitle, is type TEXT, and the third, Lesson Description, is type TEXT. Each individual Lesson row has many Questions associated with it.

The second 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.

***.IX GUI Cases Diagram:***



***.X GUI Explanation:***

The application opens up with the last question you were on if one previously used the app, otherwise it opens up with a list of lessons, displayed in rows. The user picks a lesson which loads the screen with the first question of the lesson. For a question fragment, there are Button, ListView, TextView, and ImageView objects in which each have an Activity event handler assigned respectively. A user is able to select an answer and click the “Check Answer” button in which case if wrong, will highlight the selected answer in red, and highlight the correct answer in green. If “Audio” button is pressed, the application will play the mp3 file associated with that question. “Previous Question” button is in placed in case the user should choose to skip a question and come back to it later. After the user has clicked “Next Question” and exhausted the list, a Report will pop-up allowing the user to see the logistics in how they performed. Once the user is done viewing the report, they can click the “Lessons” button to go back to the opening screen.

VI. 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.

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