CMPT 275

Assignment 2B – Design Document

By Team X

Project Group #10

10/20/2017

Project Website
https://sites.google.com/view/blackbox21o

Ivana Jovasevic | 301247058

Shayne Kelly II | 301193702

Aaron Nguyen | 301254559

Peter Saffold | 301228703

Ya Qi (Jerry) Liu | 301255583

Table of Contents

0. Revision History

Revision	Status	Date	By
1.0	Template of Design Document	September 26th, 2017	Jerry
2.0	Added introduction, guidelines and feature priority	October 1st, 2017	Jerry, Peter, Ivana
3.0	Added state diagrams, and class diagrams, data requirements	October 7th, 2017	Ivana, Aaron, Shayne
4.0	Detailing and formatting	October 15th, 2017	All
5.0	Finalize document	October 17th, 2017	All

Table 1: *Revision History*

1. Introduction

This design document will outline the high-level system organization for the iOS application Personal Emotion Trainer (P.E.T.). The document will address technical guidelines, relevant ethical and legal issues, and showcase system diagrams to outline interactions between major modules, classes, and functions of the system. A summary of all input and output of the system will be provided to showcase data communication for the application. Finally, features priority are listed to display the main module of focus for each version during the development of the application.

2. Guidelines

Throughout the development of this application there are technical, ethical, and legal guidelines to follow. Programmers will refer to the guidelines below when creating the application.

2.1 Technical Guidelines

- This application is designed for iPad pro (Apple tablet), a working camera is recommended.
- This application is designed for a resolution of 1366x1024.
- This application will require iOS version 9.X or later to operate.
- This application will be designed and implemented using the IDE Xcode 8 or 9.
- The source code will be written in Swift 3 or 4.
- Unit and integrated testing will be done through the Xcode IDE and manually on an iPad.
- Internet connection is needed for complete functionality of the application.
- Firebase will be the free real-time database and backend service provider.
- SQLite XCode Database will be the local database option on the iPad.
- Stock photos, videos, animation and SFX will be used to produce assets.
- Adobe Photoshop and Premiere Pro will be used to create additional assets.
- Facial Recognition technology provided by Azure or Affectiva (Choice of API will be finalized soon) will be used for detecting facial expression made made by the user under the emotion recognition feature.
- Github will be used as the version control repository.
- Instrument, CLOC, and Tailor will be used for project complexity measurement.
- Documentation of the application will be done on MS Word and Google docs.

2.2 Ethical Guidelines

- Personal information will remain private and only accessible by and developers.
- All users will be treated fairly, equally and without discrimination by the application.
- The application will not charge its users for using the app.
- The application will not contain any adult content or offensive material.
- The application will follow all Apple development guidelines [1].
- No responsibility will be taken by development team for material uploaded to database by user.

2.3 Legal Guidelines

- The development team takes no responsibility for injury or illnesses resulting from use of the application.
- Copyrighted materials will not be used as assets unless a copyright license is obtained or written permission granted by the owner.
- The application will not be monetized through advertising or any other affiliation.
- The application will not feature any micro-transactions.
- The application will not ask user to perform illegal or unethical tasks.
- The user may not use this application for any illegal or unethical tasks.
- The application will not share user personal information or transmit user data without user consent as defined in the App Store Review Guidelines described by Apple [1].

3. System Diagrams

3.1 High Level System Class Diagram

The overall system architecture will consist of three different components. The first component is the part of the system contained on the iOS device. This is shown in blue below, consisting application that will run on the device and its local SQLite database (for storing facial image data, questions and other related information). The second component (shown below in purple) of the system is our internal server which will be used to store account information and game statistics for a user account. This will be implemented using a REST API and a NoSQL database on a server running in the cloud (Firebase Services). The third component (shown in red) consists of the external APIs that will be used. An emotion recognition API by Affectiva will be used to implement the facial recognition portion of the game.

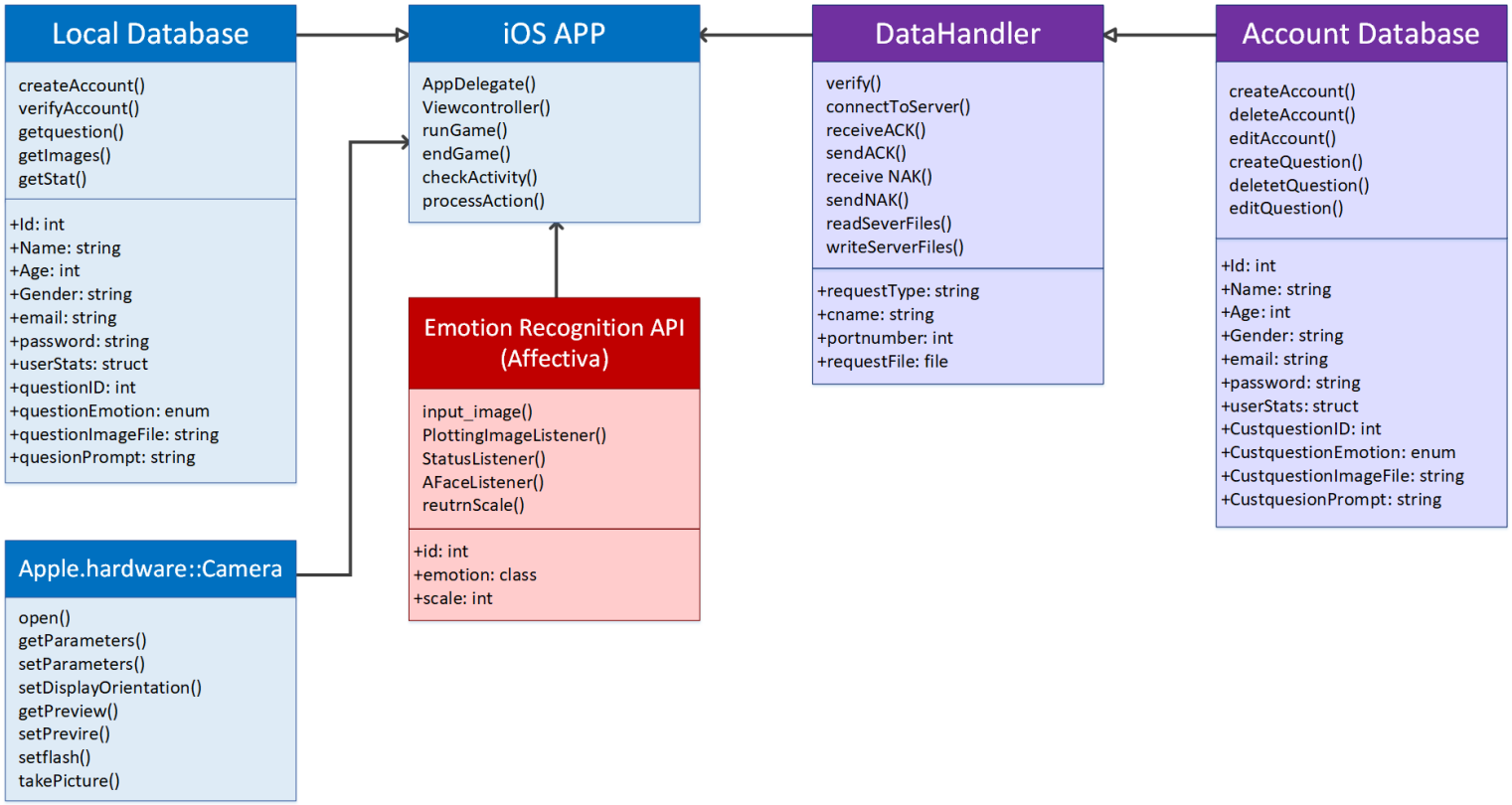

Figure 1: *High Level Class Diagram*

3.2 UI State Diagram

The UI State Diagram models the high-level interactions between user interface elements. The use cases (Refer to Requirements Document) reference major modules and provide insight to how the application is used. Each box represent a major user interface element, and the arrows connecting the boxes represent the possible transactions between them. The blue dot on the left indication entrance of the application.

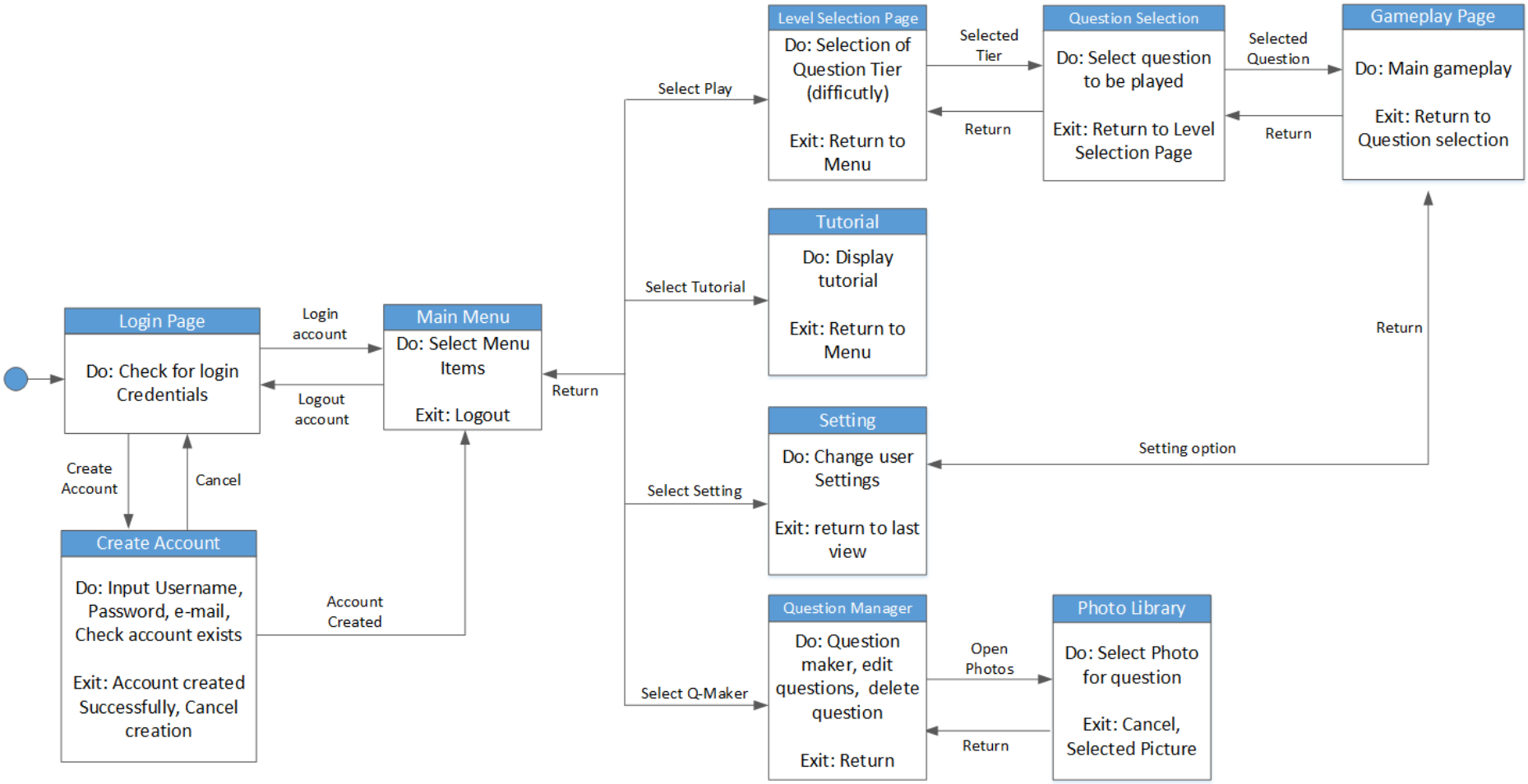

Figure 2: *UI State Diagram*

3.3 Application MVC UML Class Diagram

The Class Diagram shows the components that the application will be written in, and where classes are derived from. The application will be designed using Model View Controller architectural pattern in order to separate the functionality of the application from the UI itself.

The first component is the Models of the program, shown in red, which will implement all the main functionality of the program, including fetching and sending data to and from the remote database.

The second component are the Views of the program, shown in purple. The view is the UI display which the user interacts with, and is completely separate from the Model. Each page in the program has its own view, which is designed using XCode's interface builder, which allows the developer to drag and drop buttons, text boxes and displays onto the screen to create the UI. All Views are derived from UIView, which is the base class provided by Swift.

The third component is the Controllers of the program, shown in blue. The controllers act as a bridge between the View and the Model, and provide a layer of abstraction. The Controller has direct access to both its View and the Model. When a user action occurs in the view, the controller is notified and accesses the Model to gather the corresponding data necessary, and then returns to the View and tells it what to do. All Controllers are derived from ViewController provided by Swift, and ViewController is derived from iOS base class AppDelegate.

Below is a legend which outlines different components in the Class Diagram (Figure 3):

- **Red Arrows** → Connection from Controllers to Model: The controllers use the functionality of the Local Data Model to get the data they need for each individual Controller and View.
- **Purple Arrows** → User-action initiated connection from View to controller. Notifies the controller when a button has been pressed or input has been made, so that the controller can tell the view what to do next.
- **Blue Arrows** → Connection from Controllers to View: The controller controls its view and uses the user input that is fed to it to perform the necessary action.
- **Green Notification** 🟢 Signal sent from Model to all Controllers listening to indicate that the Model has been updated. When a controller gets this notification, it is able to connect to the model to get the new data. This maintains the separation of the model from each individual Controller.

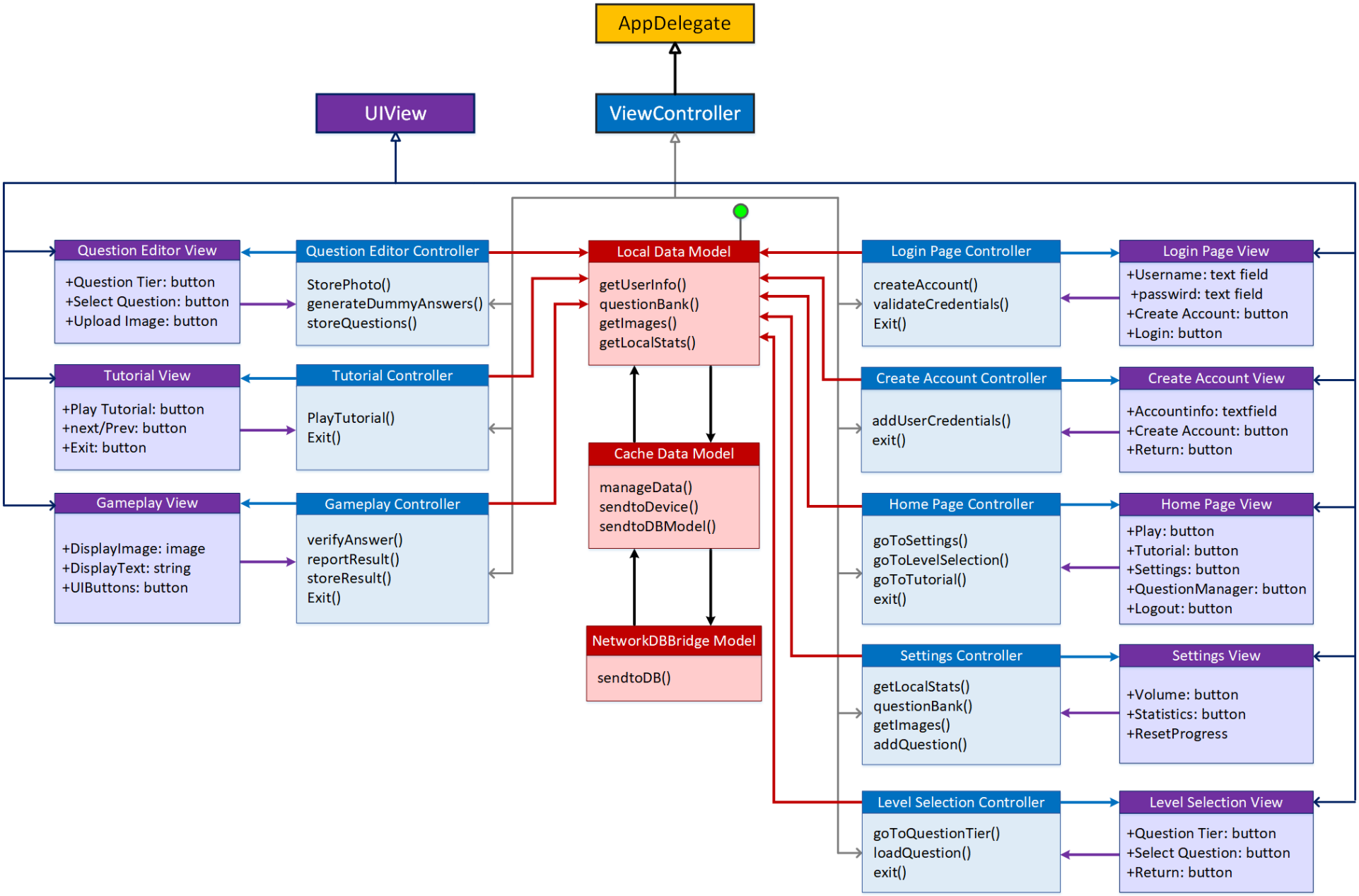

Figure 3: *Application UML Class Diagram*

4. Data Requirements

There are several data input/output requirements for the various functionality of this application. Local data will be store in SQLite XCode Database, this includes account information, questionnaire, images and game progression. Server-side and backend management will be done by web-based real time database provider Firebase. User interaction input with the application will mostly be done by the iPad touch screen. Additionally, images from either the photo gallery or direct from the camera is also required as inputs by some features of the application.

4.1 User Input:

- Registration & Authentication
    - User is required to register account in order to access the application. User will input account information such as username, email address, and password; as well as general user information such as name, age, and gender. User data will be entered manually via the on-screen keyboard for input.
    - User will enter account information when opening app. User is required to input username or email, and the password. Account information will be sent to server for authentication.
- Question Maker
    - User can import images using the camera or photo gallery features of the iPad. These images will be used for custom question maker. User is required to provide question prompt to give context to question, input will be provided by keyboard.

4.2 Database Input:

- Account information
    - General account information from registration under user input will be send to the web-based server (Firebase). This include username, email address, password, and general user information such as name, age, gender.
- Gameplay Statistics
    - Statistics such as id of levels completed, question response time, number of correct and incorrect selections, and number of skips will be sent to server.
    - Statistics base on which emotion or facial express class the user is most familiar with or had the most difficulty recognizing will be send to server.
    - Statistics of time used for answering each question will be send to the server.
- Custom Question Maker Data
    - Images, question prompt, and answer will be bundled into data packet and uploaded to server, and will be retrieved when requested.

4.3 Database Output:

- Account Authentication
    - User input username, email, and password will be authenticated with account information retrieved from the servers.
- User Statistics (Global)
    - Combined global statistics of all player will be outputted onto web portal. Here the statistics is viewable by anyone. User's information will remain anonymous.

5. Feature Priority

There will be three implementation iterations during the course of this project - version 1, version 2, and version 3. Each version will have its own major features/functions. The list below will indicate the priority of each major features and functions. For detailed feature and functional requirements for each function please refer to Requirements Document - User Manual.

Feature Priority Definitions

P1 - Priority 1 (Highest: This is a required feature that can't be left out)
P2 - Priority 2 (Medium: The app can function without this feature)
P3 - Priority 3 (Low: Optional extra feature that can be added if there is time)

5.1 Version 1 - Prototype

Version 1 will be focusing on creating the GUI and linking each view together. The contents will mostly be placeholders until features and function have passed unit testing. The core navigation will exist for testing of general flow of application. Then the GUI will be integrated with other major features and functions. Account authentication system will be implemented in this version. The server-side database will be setup, since it will be tested heavily by the account authentication system and question storage system. Other key features such as web portal, tutorial, etc. will be in development as well at lower priority, and will be integrated after unit testing.

User Manual Section	Feature/Function	Priority
3.1	Account Authentication System	**P1**
3.1	Database Implementation	**P1**
3.2	Game Configuration Settings	**P3**
3.5	Tutorial	**P2**
3.6	Gameplay Development	**P1**
N/A	Graphical User Interface	**P1**
N/A	Asset Development	**P2**
N/A	Web portal	**P2**

5.2 Version 2

Version 2 will focus on the major features and functionality of the application which include custom question maker. Additionally, the camera and photo gallery integration will begin development as well. Early asset development will start during this phase, GUI elements will be refined with better graphical assets. Voice and audio elements will be added. Additional questions will be added to the core game. Database will need improvement with increase in data communication.

User Manual Section	**Feature/Function**	**Priority**
3.1	Database Improvement I	**P3**
3.4	Custom Question Maker	**P1**
3.6	Additional Questions	**P2**
3.7	Gameplay - Custom Questions	**P1**
3.7	Camera, Photo Gallery Integration	**P1**

5.3 Version 3

Version 3 will be dedicated to the development and integration of the emotion recognition API, and polishing and improvement of the overall application. In this phase, additional assets will be incorporated as well as addition of more questions to the gameplay. A reward system is considering to be implemented at this stage. Terms and Agreement will be implemented.

User Manual Section	**Feature/Function**	**Priority**
3.1	Database Improvement II	**P1**
3.6	Additional Questions	**P1**
3.8	Emotion Recognition API Integration	**P1**
N/A	Asset Development	**P2**
N/A	Unplanned features arise during development	**P3**
N/A	Refine Reward System	**P3**

6. Reference

[1] A. Inc., “App Store Review Guidelines,” *App Store Review Guidelines - Apple Developer*. [Online]. Available: https://developer.apple.com/app-store/review/guidelines/. [Accessed: 09-Oct-2017].