Senior Design Document
for
Perceptual Video Quality Data Acquisition and Analysis Tool
Version 1.0

Prepared by

California State Los Angeles Senior Student Team
Deanna Streffer (Lead)
Ponaroth Eab
Daniel
Nelson
George Beltran

Project Advisor
Mark Sargent

Liaison
Harrison L. Hays V
Peshala V. Pahalawatta
Ross Castillo

November 14, 2019
## Revision History

<table>
<thead>
<tr>
<th>Name</th>
<th>Date</th>
<th>Reason For Changes</th>
<th>Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initial Draft</td>
<td>2019-11-29</td>
<td></td>
<td>1.0</td>
</tr>
<tr>
<td>Revision</td>
<td>2019-12-8</td>
<td>Additional information</td>
<td>2.0</td>
</tr>
</tbody>
</table>
Revision History.................................................................................................................2
Table of Content................................................................................................................3
1. Introduction...........................................................................................................4
   1.1. Purpose......................................................................................................4
   1.2. Document Conventions..............................................................................4
   1.3. Intended Audience and Reading Suggestions............................................4
   1.4. System Overview......................................................................................4
2. Design Considerations............................................................................................5
   2.1. Assumptions and dependencies.................................................................5
   2.2. General Constraints....................................................................................5
   2.3. Goals and Guidelines..................................................................................5
   2.4. Development Methods..............................................................................6
3. Architectural Strategies..........................................................................................7
4. System Architecture..............................................................................................8
   4.1. Level 0 .......................................................................................................8
5. Policies and Tactics................................................................................................8
   5.1. Specific Products Used..............................................................................8
   5.2. Requirements Traceability........................................................................8
   5.3. Testing the software..................................................................................8
   5.4. Guidelines and conventions......................................................................8
   5.5. Building the Code......................................................................................8
6. Database Design .....................................................................................................9
7. User Interface.......................................................................................................11
8. Glossary................................................................................................................12
1. Introduction

1.1 Purpose
The purpose of this document is to identify the software requirements for the iOS app, database and administration tool web application. The iOS app collects data from users and sends it to the database. Then, the administration tool can communicate with the database to view the data, alter, or export it.

1.2 Document Conventions
All text for the document will be single-spaced, size 12pt, and Calibri font; however, the headers will be size 24pt, Calibri font, and bolded. Bullet points are used to list or outline specifications for ease of readability.

1.3 Intended Audience and Reading Suggestions
This document is intended for the AT&T engineers who will be utilizing the system, as well as for any other engineer who may expand or otherwise work on it.

1.4 System Overview
The system contains a MySQL database which stores video rating data. A Node.js server provides API endpoints to send, receive, and modify data. A web application is used as an admin tool in which the data can be analyzed and visualized, and, in addition, video data can be added and modified. Lastly, Android and iOS apps are created to collect the data by allowing users to rate the quality of the videos.
2. Design Considerations

- Compatibility with multiple mobile devices
- Compatibility with multiple web browsers
- Communication between mobile app, web application and database
- Display will vary and change slightly depending on the platform and screen sizes being used.
- Swift programming language and libraries.
- React JS and React libraries.

2.1 Assumptions and Dependencies

This software was intended for internal AT&T use. Thus, we had a lot of flexibility in terms of design limitations and constraints. For example, in order to avoid adding advanced encryption techniques to log-in to the admin portal, the liaison was satisfied with simply running the React application tool on his local machine. In addition, we did not need to worry about preparing the application for widespread distribution, such as via the Apple Store or Google Play store. The engineers frequently gave feedback regarding new feature ideas for the system, but we have not gotten and do not anticipate to receive major modification requests to the system.

2.2 General Constraints

Because the software was intended for internal use, we did not have significant constraints. However, in terms of collecting data, AT&T only was licenced to use the video clips in a laboratory setting. Therefore, we would not be able to distribute the application to get more data.

2.3 Goals and Guidelines

The system had already been worked on for one year by seniors at Harvey Mudd college. Thus, a lot of the basic parts were completed. However, we needed to make several modifications.

- The liaisons were not convinced that the NoSQL database was the best way to store and represent their data. Therefore, we decided to migrate to a MySQL database.
- The liaisons described a lot of “friction” in the existing iOS application which made it difficult to use and discouraged people from participating in rating videos.
- The liaisons were interested in adding an Android app, which would run on Android TV devices as well be easily distributable to internal AT&T employees.
- The liaisons wanted a way to easily view and analyze their data, as well as be able to prove or disprove key hypotheses that they had about video quality.
- We also wanted it to be easier for the liaisons to manage their videos and video collections without having to directly access the database.
• We wanted better code quality by implementing design patterns as well as implementing unit testing and continuous integration.

2.4 Development Methods
We used Agile development methods. Each two-week sprint, we met on Monday, met with the liaisons to discuss progress and updates, then tasked out our work on our Trello board, which our faculty advisor and liaisons also had access to. We set goals for the sprint in our “To-Do” column with tasks from our backlog, and closed them out by the end of the sprint, when we worked on the Bi-Weekly reports. We allowed our project to be flexible and change as we worked by integrating our liaison’s ideas into our design.
3. Architectural Strategies

The iOS application was designed with the programming language “Swift”. Different libraries like “Alamo fire”, “Reachability Swift”, “Charts”, and “SD Web Image” were all used to design features and pass data into the application.

The Android application was designed using Kotlin. Different libraries such as “RXJava”, “OkHTTP”, “Retrofit”, “Android Architecture Components”, “MPCharts” were all used to design features and pass data into the application. A form of MVVM architecture pattern was used.

The Web application was designed using React.js. The API was designed with Node.js.
4. System Architecture

Modules:
1. iOS Application
2. Android Application
3. Web Application
4. Cloud API
5. MySQL Database
5. Policies and Tactics

5.1 Choice of which specific products used

The iOS application was developed on Xcode.
Different libraries like “Alamo fire”, “Reachability Swift”, “Charts”, and “SD Web Image” were all used to design features and pass data into the application.
The Android application was developed on IntelliJ using Kotlin. Kotlin has been declared Android’s official development language as opposed to Java as of 2019 and was therefore chosen for this project.
The web application was developed using React.JS.
The cloud API was developed with Node.js and hosted using Heroku.
The database was migrated from Firebase (Non-relational) to MySQL.

5.2 Plans for ensuring requirements traceability

We have used an Agile development system and have been organizing tasks as items on our Trello Board. All tasks are assigned and can be viewed per sprint on Trello after they have been tested and merged.

5.3 Plans for testing the software

To generate data, we will be using our mobile applications as well as having other people use the application. By using the product, we can thoroughly test it to ensure we are getting the correct data before handing off to the liaisons.

The cloud API will use unit testing to ensure that the correct data is being sent to and retrieved from the database.

5.4 Coding guidelines and conventions

Code is managed via private repositories on GitHub. Everytime a task is completed, a pull request is made for the subsequent branch. After a brief code review, the code is merged.

5.5 How to build and/or generate the system’s deliverables (how to compile, link, load, etc.)

An apple device is needed to build and run the iOS application. Xcode must have the most recent update if tested on iPhone or iPad, if not the build will fail.
To run the Android app, Android Studio must be installed.
To run the Web App, npm must be installed. It can be run by using the command “npm start”.
To run the API, npm must be installed. It can be run by using the command “npm start”.
6. Database Design
The database for this system is a relational database with MySQL.

Tables:
- **collections**
  - id - int(32) - auto increment - primary key
  - name - varchar(255)
  - thumbnail - text
  - hidden - tinyint(1)
- **config**
  - id - int(32) - auto increment - primary key
  - setting - varchar(255)
  - value - varchar(255)
- **scores**
  - id - int(32) - auto increment - primary key
  - user_name - varchar(255)
  - first_video - varchar(255)
  - second_video - varchar(255)
  - score - int(2)
  - device - varchar(255)
  - mirroring - tinyint(1)
  - brightness - double
  - rewatch - tinyint(32)
  - notes - text
- **scores_absolute**
  - id - int(32) - auto increment - primary key
  - user_name - varchar(255)
  - video - varchar(255)
  - score - int(2)
  - device - varchar(255)
  - mirroring - tinyint(1)
  - brightness - double
  - rewatch - tinyint(32)
  - notes - text
- **tutorial**
  - id - int(32) - auto increment - primary key
  - url - text
  - expected_rating - tinyint(2)
- **users**
- id - int(32) - auto increment - primary key
  - name - varchar(255)
  - score - int(32)
- users_absolute
  - id - int(32) - auto increment - primary key
  - name - varchar(255)
  - score - int(32)
- videos
  - id - int(32) - auto increment - primary key
  - bitrate - int(32)
  - codec - varchar(32)
  - color_space - varchar(32)
  - content - varchar(255)
  - eotf - varchar(32)
  - frame_rate - int(32)
  - original_resolution_height - int(32)
  - original_resolution_width - int(32)
  - output_resolution_height - int(32)
  - output_resolution_width - int(32)
  - url - text
  - name - varchar(255)
  - tags - text
7. User Interface

7.1 Overview of User Interface

When the user opens the mobile application, they will see a screen that contains a textbox and two buttons for logging into the application with a username and logging into the application as a guest user. The next screen will give the user a choice to press one of two buttons, one to view random videos and another to view a searched video. If the random video button is pressed, the user will watch two randomly selected videos from a randomly selected collection. If the searched video button is pressed, the user will be displayed a list of collections to choose from. After the user chooses a collection, the user will view two randomly selected videos from that collection. When the user watches two videos from both options, the user will be shown a screen to rate each video and compare which one is better than the other video according to what they have seen by pressing one of five buttons. After a button is pressed, the user will be shown a pie graph that displays the percentages of how everyone else rated the two videos and where the user falls in that graph. There will be a button for the user to press so they can view the video properties for both videos. Two more buttons will also be on the graph screen to allow the user to either watch videos again or exit the application.

The user will be informed of what everything on the user interface of the web application by the initial page containing a list explaining how each component functions. An application bar will be displayed at the top of the window with the name of the application. There will be a sidebar on the screen that will direct users to pages that are listed on it when the user click on the name of the page.

7.2 User Interface Flow Model
8. Glossary

<table>
<thead>
<tr>
<th>SRS</th>
<th>Software Requirement Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td>SDS</td>
<td>Software Design Specification</td>
</tr>
<tr>
<td>js</td>
<td>Java Script</td>
</tr>
<tr>
<td>Android OS</td>
<td>A mobile operating system developed by Google</td>
</tr>
<tr>
<td>iOS</td>
<td>A mobile operating system developed by Apple</td>
</tr>
<tr>
<td>MacOS</td>
<td>Macintosh Operating System</td>
</tr>
<tr>
<td>JDK</td>
<td>Java Development Kit</td>
</tr>
<tr>
<td>App</td>
<td>Application</td>
</tr>
<tr>
<td>OS</td>
<td>Operating System</td>
</tr>
<tr>
<td>GUI</td>
<td>Graphic User Interface</td>
</tr>
<tr>
<td>MMI</td>
<td>Main Menu Interface</td>
</tr>
<tr>
<td>API</td>
<td>Application Program Interface</td>
</tr>
<tr>
<td>HEVC</td>
<td>High Efficiency Video Coding</td>
</tr>
<tr>
<td>SQL</td>
<td>Structured Query Language</td>
</tr>
<tr>
<td>HTTP</td>
<td>Hyper-Text Transfer Protocol</td>
</tr>
<tr>
<td>AVC</td>
<td>Advanced Video Coding</td>
</tr>
<tr>
<td>MB</td>
<td>Megabyte</td>
</tr>
<tr>
<td>GB</td>
<td>Gigabyte</td>
</tr>
<tr>
<td>RAM</td>
<td>Random Access Memory</td>
</tr>
<tr>
<td>NPM</td>
<td>Node Package Manager</td>
</tr>
<tr>
<td>URL</td>
<td>Uniform Resource Locator</td>
</tr>
</tbody>
</table>