Software Requirements Specification

for

Perceptual Video Quality Data Acquisition and Analysis Tool

Version 1.0

Prepared by

California State Los Angeles Senior Student Team

Deanna Thomas (Lead)

Ponaroth Eab

Daniel Ramirez

Nelson Huynh

George Beltran

Project Advisor

Mark Sargent

Liaison

Harrison L. Hays V

Peshala V. Pahalawatta

Ross Castillo

March 22, 2019

Table of Contents

Revision History

Name	Date	Reason For Changes	Version
Initial Draft	2019-11-14		1.0
Second Semester Updates	2020-03-22	Updates as project progressed	2.0

1. Introduction

1.1 Purpose

The purpose of this document is to specify the requirements of the AT&T Video Quality Analysis system. We will specify requirements for all pieces of the system, including an iOS and Android application, a web admin tool, an API, and a database. We will also specify any external dependencies that the system relies on so that it can be managed by another team of engineers.

1.2 Intended Audience and Reading Suggestions

This document is intended for engineers to understand the project deeply. In addition, they should be able to continue development on the project should additional features need to be added in the future.

1.3 Product Scope

The product is a Video Quality Rating and Analysis Tool. It is a mobile application that can be downloaded onto your smartphone or tablet device. The user of this application will be shown two videos that capture the same scene but have different video properties such as resolution, bit rate, and frame rate. The user will decide which video they prefer after they have finished watching them and the application will display a graph showing the user how other users rated and a button that will display a detailed list of properties for each video when pressed. The Video Quality Rating and Analysis Tool will be used internally at AT&T (Direct TV) to generate data and to analyze whether a change in a certain video property can be noticeable or not to the general public. If a certain change isn't noticeable and would cost less than what is currently being streamed to the public through Direct TV, then it would be more beneficial to stream videos using the new method. Currently it takes AT&T hundreds of thousands of dollars to buy the equipment needed to rate video quality and maintain it. This application will save money by allowing more people to rate without the need to buy costly equipment since all that is needed is a mobile device. The Video Quality Rating and Analysis Tool will end up creating a better Direct TV streaming service while saving AT&T a huge amount of money in the long run.

1.4 Definitions, Acronyms, and Abbreviations

The acronyms and abbreviation list has been moved to Appendix A: Glossary in section 6 due to it being fairly long.

1.5 References

Swift.org - Swift programming language documentation (https://swift.org/documentation/)
Kotlin - Kotlin programming language documentation (https://kotlinlang.org/)
Javascript programming language documentation (https://devdocs.io/javascript/)
React.org - React JS documentation (https://reactjs.org/docs/getting-started.html)
Cocoapods for library manager - (https://cocoapods.org/)
Gradle - Build Tool documentation (https://gradle.org/)

2. Overall Description

2.1 Product Perspective

This software system runs independently of any other software system. To collect data using this system, the mobile app, web admin app, database, and API are all that is needed.

2.2 Product Functions

Mobile Applications:

- Side-by-Side video quality rating application allows the user to rate which video was better out of two. One application for iOS and one for Android.
- Absolute rating application allows the user to rate a single video on a 1-5 scale. One application for iOS and one for Android

Web Admin Application:

- Allows the admin to manage videos, collections
- Allows the admin to view and analyze result data

API:

- Provides access to the database with different endpoints for the mobile applications and web admin application

Database:

- Stores videos, collections, users, and score data

2.3 User Classes and Characteristics

End users of the Mobile Applications will be random people, likely AT&T employees, who use the application in a lab setting. They will not be installing it on their local devices.

Users of the web admin tool will be the AT&T video team. Therefore, it has administrative access to viewing and managing data.

The API will only be accessed by the mobile and web apps themselves. It has no direct with the end user. The database will only be accessed by the API and will also not have direct interaction with the end user.

2.4 Operating Environment

The mobile applications will be available for iOS and Android. The iOS app will be developed specifically for iOS 12, the previous iOS version, but is also optimized for iOS 13. The Android

app will be developed for Android 10, but should be compatible with devices running Android 4.4 or greater.

The web app will only be run on the admin's local machine to ensure security. The user must install npm on their machine, add the project files to their computer, then run `npm start` in the command line.

The API is running on a Node.js server which is hosted on Heroku. Lastly, the database is MySQL and will require a web server which is able to host a MySQL database.

2.5 Design and Implementation Constraints

Due to the fact that this application will not be distributed in an app store, and that the admin tool can simply be run locally, we did not have any significant design and implementation constraints.

2.6 User Documentation

The mobile applications have been designed to be very easy to use, so that anyone can use them and provide useful data. Therefore, user documentation does not need to be provided. However, we will provide a basic user guide for the web admin tool. This will be provided in an "Info" page within the application itself.

2.7 Assumptions and Dependencies

The applications are designed for current versions of iOS and Android. Should any major changes take place, the software requirements may need to be revisited or updated.

We will be using Heroku to host the API. Heroku is a cloud platform that allows us to deploy the API. It is free to use for systems with minimal requests. Should the system scale up and become a public application, this free hosting plan will need to be upgraded to ensure accessibility.

Lastly, some sort of MySQL database server will be required. Currently, the system is being hosted on a team member's personal web server. When it comes time to hand off the application, we will export the database for the customer to import and use on their own server. No specific version is required.

2.8 Apportioning of Requirements

At this time, there are no requirements planned to be implemented in a future version of the system.

3. External Interface Requirements

3.1 User Interfaces

For the mobile applications, our requirement was to ensure that there was the least amount of friction as possible. The easier to use, the more results will be able to be collected. We used a similar theme throughout the applications, using the AT&T blue color, AT&T logo, as well as built in system design practices (i.e. Google's Material Design for Android). In terms of visual design, we were not given any additional requirements. While details may be different between iOS and Android, the user flow remains the same.

The web application will only be used by the project leaders/engineers at AT&T. Therefore, we did not have many restrictions on design. However, it was a priority for us that the UI is intuitive and easy to use. Like many apps, we use a sidebar which displays on all pages with the navigation options, and each link in the sidebar displays the respective page.

3.2 Hardware Interfaces

Our system does not have hardware interface requirements.

3.3 Software Interfaces

The iOS application requires a minimum version of 12.
The Android application requires a minimum version of 4.4.
The web app requires npm version 6 or above to run locally.
The API is currently run on Heroku, but could be moved to any cloud location in which the server can be accessed securely via a URL.
Lastly, MySQL version should be 5.6.41-84.1 or above.

3.4 Communications Interfaces

The mobile and web applications will communicate directly with the API. Each request will be sent with a BEARER TOKEN which will authenticate the request. This token will be kept securely in the code for the mobile applications and web app so that it cannot be compromised. The applications send GET, POST, PATCH, and DELETE HTTP requests to our API.

The mobile and web applications will require internet access (WiFi, 4G, etc) at all times in order to use the system.

4. Requirements Specification

4.1 Functional Requirements

1. Mobile Application
    a. Login
        i. The app shall accept a username for login
        ii. The login shall not be secured and shall not require a password
        iii. The app shall allow the user to continue as a guest
        iv. The app shall call the API to get the user's information upon login
    b. Rating Choice
        i. The app shall allow the user to choose between watching a random video and browsing a list of available videos
        ii. The app shall allow the user to logout
    c. Random Video Flow
        i. The app shall query the API to get the random collection and the two random videos from that collection
        ii. The app shall show the videos in the following order:
            1. The app shall show the user instructions for rating the videos
            2. The app shall show the first randomly selected video
            3. The app shall prompt the user to watch the second video
            4. The app shall show the second randomly selected video
    d. Browse Video Flow
        i. The app shall query the API to get all collections and display them for the user to select
        ii. After selecting a collection, the app shall query the API to get two random videos from that collection
        iii. The app shall show the videos in the following order:
            1. The app shall show the user instructions for rating the videos
            2. The app shall show the first randomly selected video
            3. The app shall prompt the user to watch the second video
            4. The app shall show the second randomly selected video
    e. Video Rating
        i. The app shall show a rating screen which allows the user to select which video was better with the options Definitely Video A, Maybe Video A, About the Same, Maybe Video B, and Definitely Video B. The user must select a rating to proceed

ii. The app shall allow the user to rewatch the videos
iii. The app shall send the user's chosen rating to the API. It must include the username, videos rated, device type, if mirrored, screen brightness, and rewatch count

f. Video Results
i. The app must query the API to check the show_results setting.
ii. If the show_results setting is ON, the app shall show the current rating data for that video in a chart
iii. if the show_results setting is OFF, the app shall only display a "Thank you for rating" message
iv. The user shall be able to select "Next Video" to watch another random video
v. The user shall be able to logout from the application
vi. The user shall be able to view the video's technical details by clicking an "info" button

2. Absolute Rating Mobile Application
a. The app shall have all the same login requirements as #1.a above
b. The app shall have all the same rating choice requirements as #1.b above
c. The app shall show a tutorial before rating videos with the following user flow:
i. The user will be shown a first video and instructed that this should be considered "Excellent"
ii. The user will be shown a second video and instructed that this should be considered "Good"
iii. The user will be shown a final video and instructed that this should be considered "Poor"
d. Random Video Flow
i. The app shall query the API to get the random collection a single random video from that collection
ii. The app shall show the videos in the following order:
1. The app shall show the user instructions for rating the video
2. The app shall show the randomly selected video
e. Browse Video Flow
i. The app shall query the API to get all collections and display them for the user to select
ii. After selecting a collection, the app shall query the API to get a single random video from that collection
iii. The app shall show the videos in the following order:
1. The app shall show the user instructions for rating the videos

2. The app shall show the randomly selected video

f. Video Rating
   i. The app shall show a rating screen which allows the user to select the video rating with the options Excellent, Very Good, Good, Fair, and Poor. The user must select a rating to proceed
   ii. The app shall allow the user to rewatch the video
   iii. The app shall send the user's chosen rating to the API. It must include the username, video rated, device type, if mirrored, screen brightness, and rewatch count

g. The app shall have all the same video rating requirements as #1.f above

3. Web Application
   a. Video Management
      i. The user shall be able to view all collections
      ii. The user shall be able to edit all collections
         1. The user shall be able to hide a specific collection
      iii. The user shall be able to delete a collection
      iv. The user shall be able to delete a collection along with all associated videos and scores
      v. The user shall be prompted to confirm before making any deletions
      vi. The user shall be able to view all videos for each collection
      vii. The user shall be able to edit all videos
      viii. The user shall be able to delete a video
      ix. The user shall be able to delete a video along with all associated scores
   b. Settings
      i. The user shall be able to view the system settings
      ii. The user shall be able to toggle the system settings
   c. Data Analysis
      i. The user shall be able to run a custom query on the database and return the result
      ii. The user shall be able to view rankings of videos for each question
      iii. The user shall be able to view graphs which display data to prove/disprove the following hypotheses:
         1. Higher framerate is preferable to higher resolution if bitrate is constant.
         2. Bits per pixel (bitrate / height * width * framerate) is a good way to predict user preference.
         3. Users can't tell the difference between low bitrate (< 1Mbps AVC) variants.

4. Users can't tell the different between high bitrate (> 6Mbps AVC) variants.
5. HEVC doesn't provide noticeably better quality than AVC at low bitrates (< 1Mbps)
6. Frame rate is still preferable to resolution on HEVC encoded content.
7. The HLS recommended 1.5x bitrate ABR ladder always provides noticeable VQ improvement between variants.

4. API
    a. The user shall be able to send a query to get data from the database:
        i. Send a score
        ii. Get all scores
        iii. Get a list of all collections
        iv. Get a single random collection
        v. Get two random videos from the collection
        vi. Get collection details from collection ID
        vii. Get video details from video ID
        viii. Login a user and get user data
        ix. Add points to a user
        x. Add a collection
        xi. Edit a collection
        xii. Delete a collection
        xiii. Add a video
        xiv. Edit a video
        xv. Delete a video
        xvi. Get config/settings
        xvii. Update config/settings
        xviii. Run a custom query which has been sanitized only to allow SELECT*
        xix. Get video rankings for each collection
        xx. Get average score per AVC video
        xxi. Compare ratings for HEVC and AVC codecs
        xxii. Average scores per bitrate
        xxiii. Average scores per bit-per-pixel ratio
        xxiv. Average scores per Framerate/Resolution pair
        xxv. Average scores per Framerate/Resolution pair for HEVC videos only
5. Database
    a. The database shall be relational and use MySQL
    b. The database shall store the following information:

i. Users
    1. Table for Rating App
    2. Table for Absolute Rating App
ii. Video ratings
    1. Table for Rating App
    2. Table for Absolute Rating App
iii. Videos
iv. Collections
v. Absolute Rating tutorial videos

4.2 External Interface Requirements

1. Mobile Apps:
    a. Login screen:
        - Username: String
        - Login: Button
        - Gust Login: Button
    b. Welcome screen:
        - AT&T Logo: Image
        - Start Watching: Button
        - Browse Video: Button
        - Log out: Button
    c. Start watching screen:
        - Instructions: String
        - Begin video sequence: Button
        - Back: Button
    d. Browse video screen:
        - Instruction: String
        - Search bar: String input
        - Collection: Image. On-click leads to Start watching screen.
        - Back: Button
    e. Video A and Video B screens:
        - Video: Video player streams the video from api.
        - Video player has play, pause, fast-forward and go-backward for 15 sec buttons.
        - Video player has an X button for exiting the video
        - Video player has a sound button to adjust the audio volume.
    f. Continue screen:

- Instruction: String
- Continue: Button

g. Rating screen:
   - Instruction: String
   - Definitely A: Button
   - Maybe A: Button
   - About the same: Button
   - Maybe B: Button
   - Definitely B: Button
   - Watch again: Button

h. Thank you screen:
   - Thank you for rating!: String
   - Next video sequence: Button
   - Done: Button
   - More information about the videos: Button

2. Web Application:
   a. Side bar: contains links to all pages
   b. Home page:
      - Instruction: String
   c. Setting page:
      - Show graph on mobile app: button
      - Choose low bitrate value: number input
      - Choose high bitrate value: number input
   d. Add/Edit Videopage:
      - Video name: Text input
      - Video url: Text input
      - Video type: Dropdown menu
      - Collection name: Text input
      - Video resolution details: Text input
      - Video setting details: Text input
      - Video tag: Text input
      - Send: Button
   e. Add/Edit Collection page:
      - Test collection name: Text input
      - Thumbnail: URL link to an image
      - VideoID: Text input
      - Remove video field: Button

- Add video field: Button
- Send change to database: Button
- Hide or show collection: Switch

f. Search Tags page:
- Tag name: Text input
- Search tagged videos: Button

g. Delete Tool page:
- Test collection: Dropdown menu
- Delete collection: Button
- Delete collection with all associations: Button
- Video ID: Text input
- Delete video and associated scores: Button

h. Custom Query page:
- Custom query: Text input
- Run: Button
- Data table: Table showing queried data

i. Ranking Page:
- Ranking algorithm tables
- Download data: Button

j. Analysis Page:
- Refresh data: Button
- Data Graphs

4.3 Logical Database Requirements

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)
    - 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

4.4 Design Constraints

Because of time and labor constraints, the applications are still in development phase at the moment. Each application is yet to be fully tested. Each application still need improvement in user interface so that they can be pleasantly and intuitively used.

5. Other Nonfunctional Requirements

5.1 Performance Requirements

- Each mobile application requires at least 50 MB for storage.
- The mobile phone should have at least 2 GB of RAM.
- Web Application requires a web browser that can handle HTML5 and is preferably running on a desktop or a laptop.

5.2 Safety Requirements

The application shall not cause damage to the device that it is running on. Prolonged usage of the applications might cause eye fatigue.

5.3 Security Requirements

The user will have an option to not enter a username and instead log into the application as a guest to preserve their privacy while rating videos. If the user does decide to use a username login, data on the number of videos the user has watched will be stored. There should be no sensitive data stored from mobile apps users since the apps require neither password nor sensitive information. Since identity authorization is not implemented, Web application and the database should only be accessible by authorized developers and AT&T employees.

5.4 Software Quality Attributes

For all the applications mentioned, the user must have an internet connection speed of at least 3 Megabits per second to let the applications run smoothly. The application shall have the following characteristics: ease of use, usability, efficiency, and simplicity.

5.5 Business Rules

All applications mentioned are reserved under the rights of AT&T and DirectTV corporations.

6. Other Requirements

All requirements are subject to change according to the liaison.

Appendix A: Glossary

SRS	Software Requirement Specification

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

Appendix B: Analysis Models

Not applicable.

Appendix C: To Be Determined List

Not applicable.