Software Design Document

for

Open-Source Real-Time Video Player

Version 1 approved

Prepared by Brian Hernandez, Mirasol Davila, Wendy Joya, Ashley Jetty, Israel Lopez-Diaz, Tim Ellis, Rafael Mendoza, Jeffrey Luu

AT&T

December 6, 2020

Table of Contents

Table of Contents..... 02

Revision History..... 05

1. Introduction..... 06

    1.1. Purpose..... 06

    1.2. Document Conventions..... 06

    1.3. Intended Audience and Reading Suggestions..... 06

    1.4. System Overview..... 06

2. Design Considerations..... 07

    2.1. Assumptions and dependencies..... 07

    2.2. General Constraints..... 07

    2.3. Goals and Guidelines..... 08

    2.4. Development Methods..... 08

3. Architectural Strategies..... 09
4. System Architecture..... 10

    4.1. 10

    4.2. 10

5. Policies and Tactics..... 11

    5.1. Specific Products Used..... 11

    5.2. Requirements traceability..... 11

    5.3. Testing the software..... 11

    5.4. Engineering trade-offs..... 11

    5.5. Guidelines and conventions..... 11

5.6. Protocols..... 11

5.7. Maintaining the software..... 11

5.8. Interfaces..... 11

5.9. System's deliverables..... 11

5.10. Abstraction..... 11

6. Detailed System Design..... 12

6.x Name of Module..... 12

6.x.1 Responsibilities..... 12

6.x.2 Constraints..... 12

6.x.3 Composition..... 12

6.x.4 Uses/Interactions..... 12

6.x.5 Resources..... 12

6.x.6 Interface/Exports..... 12

7. Detailed Lower level Component Design..... 14

7.x Name of Class or File..... 14

7.x.1 Classification..... 14

7.x.2 Processing Narrative(PSPEC)..... 14

7.x.3 Interface Description..... 14

7.x.4 Processing Detail..... 14

7.x.4.1 Design Class Hierarchy..... 14

7.x.4.2 Restrictions/Limitations..... 14

7.x.4.3 Performance Issues..... 14

7.x.4.4 Design Constraints..... 14

7.x.4.5 Processing Detail For Each Operation..... 14

8. User Interface

8.1. Overview of User Interface..... 14

8.2. Screen Frameworks or Images..... 14

8.3. User Interface Flow Model..... 14

9. Database Design..... 14
10. Requirements Validation and Verification..... 19
11. Glossary..... 20
12. References..... 21

Revision History

Name	Date	Reason For Changes	Version
First Draft	12-06-2020		1

1. Introduction

1.1 Purpose

The Software Design Documentation (SDD) dives into our projects' software documentation. Where it will expand the knowledge of whomever reads this document to serve its purpose to help understand the descriptions in our documentation. We will illustrate the purpose, Level 0 DFD, Level 1 DFD and our user interface.

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 and comprehension.

1.3 Intended Audience and Reading Suggestions

This documentation is intended for developers and users. It's broken down into its documentational components where it will make it easier to understand step-by-step our project. This document contains the overall idea of how our project works as a whole. A suggestion for users will be to look into the sections of understanding the purpose, system overview, user interface and acronyms to better understand the project.

1.4 System Overview

Our software system is an open-source web-app available on GitHub. Our software allows users to stream videos and they provide you with real-time video playback for the purpose of research and analysis. Our software provides useful information through many metrics, such as Video Start Time, Rebuffering Ratio, Rebuffering Duration, video resolution, and many more. Other important functionality includes the ability to upload Network Profiles to throttle the network speeds and simulate other network conditions for the purpose of testing. On our software, we also have the functionality to store all the metric information in an excel file, which downloads after you are done playing your video. All these features are for the purpose of research for companies that want to test video playback and make improvements to their video streaming services.

2. Design Considerations

In our proposal, AT&T illustrated four different video players where we analyzed and researched. After the extensive research we were narrowed down to two video players. Before starting any design of the project we began by creating a simple web page and did tests considering the different KPI's. This is where we resolved that the two video players selected were the right choice because they performed better than the rest. This is because sample videos used in the video player were originally HLS supported making HLS.js and Shaka player the highest from all four video players. In Network throttling we had to look into different libraries because Clapper-Stats was its own player. We finally ended up finding a library that helped us ease the process to analyze our throttling data.

2.1 Assumptions and Dependencies

Requirements:

- Broadband
    - WIFI Speed
    - IP Address

The main concern would be broadband because without internet connection the webpage wouldn't be able to display the html file. It's speed would affect the buffering ratio and video quality as well.

2.2 General Constraints

General constraints are placed for the protection of the software and users.

- Repos were provided by AT&T
- IP Addresses were granted access with the help of the AT&T team
- Data Constraints
- HLS Videos
    - Video players are HLS and DASH supported
        - HLS.js player is only HLS video supported
- Security Considerations
    - Data collected should only be used for testing purposes

2.3 Goals and Guidelines

Goals:

- Create a program where there's accuracy in our data.
- Video player with high quality resolution
- Real-Time network throttling

Guidelines:

1. The software is needing to be completed by the end of Spring 2021
2. Full functionalities of the KPI metrics and network throttling
3. Hosting of both web pages

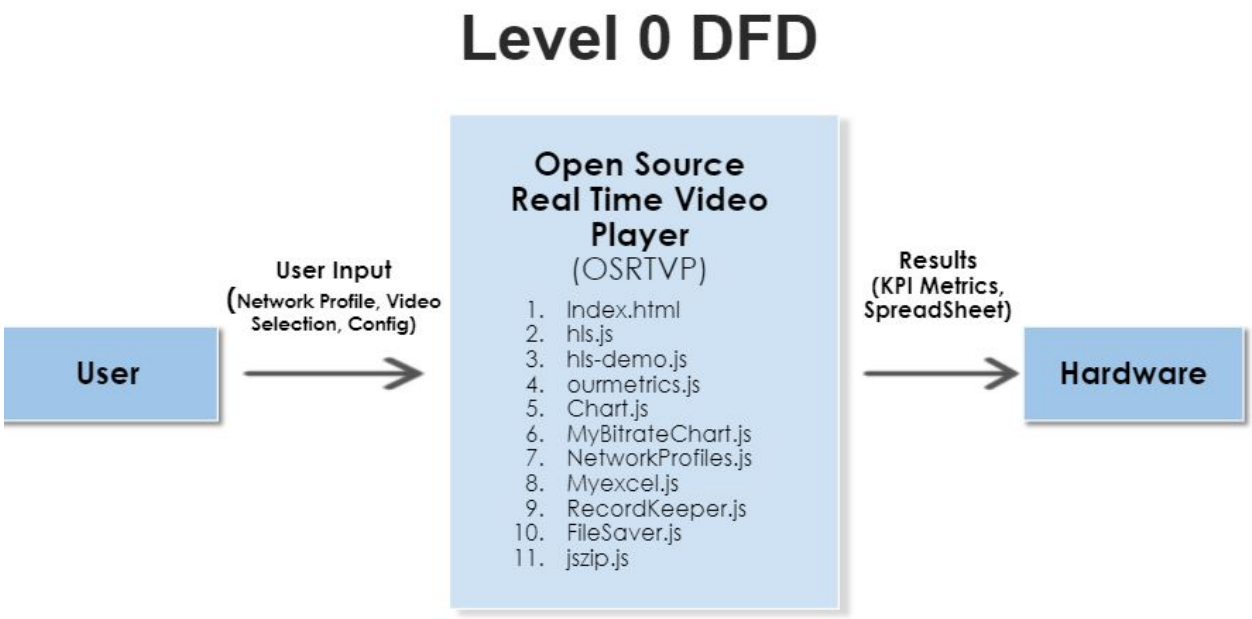
2.4 Development Methods

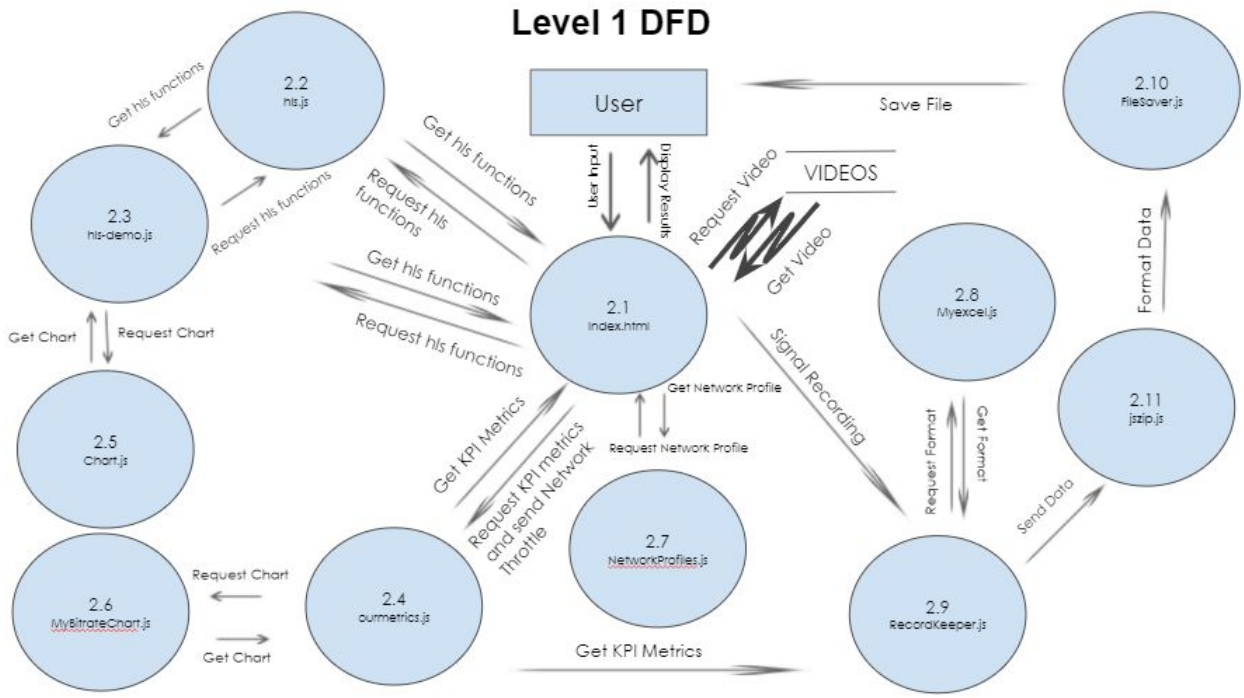
Our approach for this software is to look into the repos given by the AT&T team and start outsourcing in finding different libraries applicable to this project. After, start improving in the libraries found by integrating them into the web page. Weekly testing of the KPI metrics, if one of the metrics does not apply to the proposal then it will be scratched off our list of KPI metrics. Also, looking into clapper-stats for network throttling but it was its own player so looked into different libraries where eventually we were able to throttle the network.

3. Architectural Strategies

The web application was done by using the IDE of visual studio. The language in which it was coded was in JavaScript, html and CSS. We haven't used a database because it was all placed as part of the program. Some libraries used areClappr-stats, HLS.js, Mux.js, Chart.js and MyExcel.

4. System Architecture





5. Policies and Tactics

5.1 Choice of which specific products used

IDE

- Visual Studio Code

Programming Language

- JavaScript (An interpreted language, not a compiled language)
- HTML
- CSS

Database

- TBD

Libraries

- Clappr-stats
- HLS.js
- Mux.js
- Chart.js
- MyExcel

5.2 Plans for ensuring requirements traceability

TBD

5.3 Plans for testing the software

- Custom Business Practices

5.4 Plans for maintaining the software

- Once the project is completed on Spring 2020, this software will be available for the use of the AT&T team
- Continue being a software to test their programs and videos

6. Detailed System Design

The following are the components to the System Architecture.

hls.js

- Hls.js is responsible for HTTP Live Streaming Client. When the browser has built-in HLS support an HLS manifest is provided directly to the video element through the source property.
- Relies on HTML5 video and MediaSource Extensions for playback.
- Only compatible with browsers supporting MediaSource (MSE) API with ‘video/mp4’ mime types inputs.
- hls.js does not need any player, it works directly on top of a standard HTML <video> element
- Instantiate a hls object to load the video src link to the hls objects load source method. Then attach the video element to the hls objects media method.

hls-demo.ls

- Where all hls functions reside.
- This file is called by hls.js to get the functions needed for playback.

Index.html

- The main file that runs the web application for hls.js.
- Added features are implemented in this html file.
- Integrates all JavaScript files that are used for the System Architecture.

ourmetrics.js

- Responsible for measuring all kpi metrics.
- Count - How many times rebuffering occurs.
- Frequency - How often does rebuffering occur.
- Duration - How long does a rebuffer take.
- Ratio - A percentage of how long the player rebuffered versus the overall playtime.

Chart.js

- Displays the bitrate value per second based on the video src.

MyBitrateChart.js

- Our bitrate chart keeps track of how the video is performing when it's loaded.

NetworkProfiles.js

- Set up properties for the line chart to display bits/sec to the user.
- Properties such as time in seconds, bit/sec, and different network categories such as: WiFi Cellular, and Stream Saver.
- Executes regardless if a user selects a network preset.
- Dynamically displays the changes of bit/sec when the network is changed.

MyExcel.js

- A required JavaScript library to export from a web application data into Excel Format.

RecordKeeper.js

- Writes the recorded data to an excel sheet so the client can view themselves for analyzing.
- Writing four dynamic values that consist of Playtime, Rebuffering Ratio, Estimated Bandwidth, and Bitrate Mean.

FileSaver.js

- FileSaver.js is the solution to saving files on the client-side.
- Perfect to generate files and for saving sensitive information.

Jszip.js

- A JavaScript class for generating and reading zip files.

7. Detailed Lower level Component Design

TBD

8. Database Design

TBD

9. User Interface

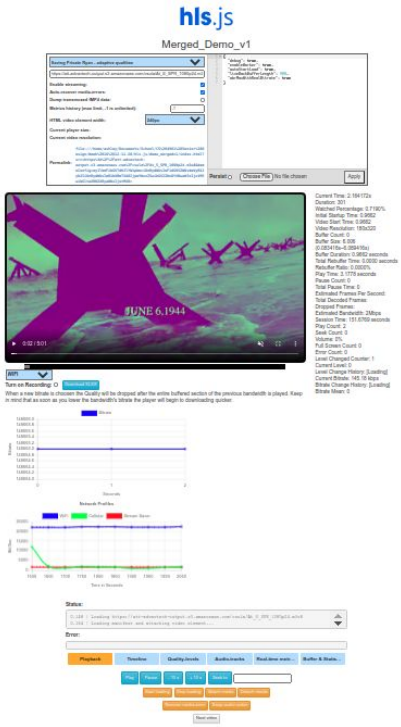
9.1 Chrome/Safari Overview of User Interface

When the user opens the application on a Chrome or Safari browser, they will see the entirety of the OSRVP and its UI components. Visually going from top to bottom and left to right, the user will first see the Video Source Information which will provide title and source information of the video selected. Beneath the Video Source Information is the Video Player Screen which is the viewer that plays the video and the KPI Metric Stats that are values of the real time metrics of the video being played. Underneath the Video Player Screen is the Bandwidth and Record Metric Menu. These allow the user to select a Bandwidth Network Throttle Value and gives the user the option of recording KPI Metric Stats to an excel sheet based on playtime. Next comes the Bit Ratio and Network Profile Charts which gives the user real time charted value changes of both the video's bit rate and the bandwidth value it's using. Lastly our HLS Playback Stats and HLS Quality Level menu allows for the user to view the libraries playback values of the video and also change the level of bit rate quality during playtime.

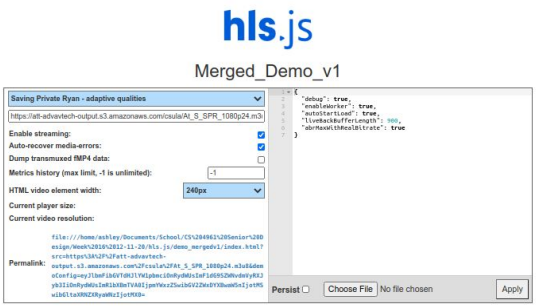
9.2 Screen Frameworks or Images

I. General Overview
II. Video Source Information
III. Video Player Screen
IV. KPI Metric Stats
V. Bandwidth and Record Metric Menu
VI. Bit Ratio and Network Profile Charts
VII. HLS Playback Stats
VIII. HLS Quality Level Menu

I.


hls.js
Merged_Demo_v1
JUNE 6,1944

II.


hls.js
Merged_Demo_v1
Saving Private Ryan - adaptive qualities
Enable streaming:
Auto-recover media-errors:
Dump transmuxed fMP4 data:
Metrics history (max limit, -1 is unlimited):
HTML video element width:
240px
Current player size:
Current video resolution:
Permalink:
Persist
Choose File
No file chosen
Apply

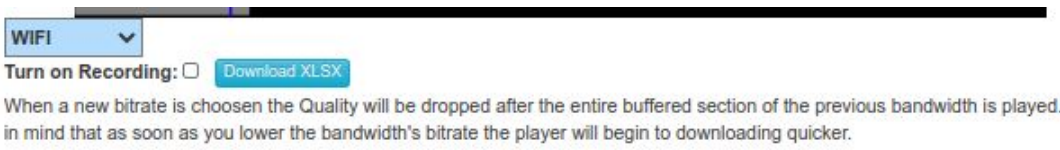
III.


JUNE 6,1944
0:02 / 5:01

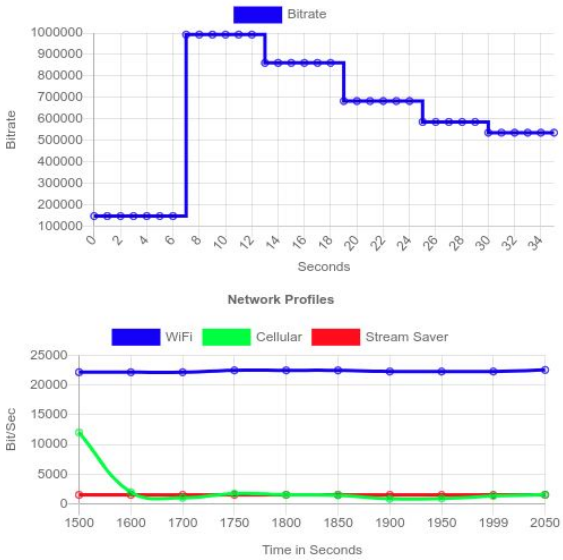
IV.

Current Time: 47.953038s
Duration: 301
Watched Percentage: 15.9312%
Initial Startup Time: 0.9662
Video Start Time: 340.0842
Video Resolution: 360x640
Buffer Count: 0
Buffer Size: 54.012292
(0.083416s~54.095708s)
Buffer Duration: 0.9662 seconds
Total Rebuffer Time: 0.0000 seconds
Rebuffer Ratio: 0.0000%
Play Time: 49.0140 seconds
Pause Count: 1
Total Pause Time: 340.0842 seconds
Estimated Frames Per Second: 24
Total Decoded Frames: 980
Dropped Frames: 29
Estimated Bandwidth: 7Mbps
Session Time: 407.6770 seconds
Play Count: 3
Seek Count: 0
Volume: 0%
Full Screen Count: 0
Error Count: 0
Level Changed Counter: 4
Current Level: 3
Level Change History: [Loading, 0, 1, 2]
Current Bitrate: 326.82 kbps
Bitrate Change History: [Loading, 145.18 kbps, 254.62 kbps, 438.91 kbps, 321.14 kbps, 304.39 kbps, 312.55 kbps, 306.04 kbps]
Bitrate Mean: 294.35 kbps

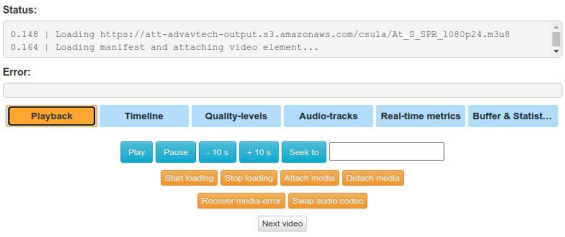
V.



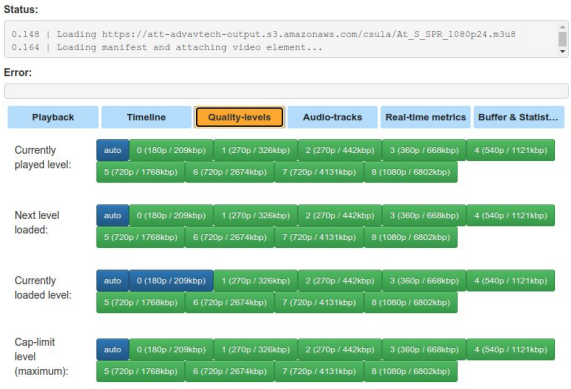
VI.



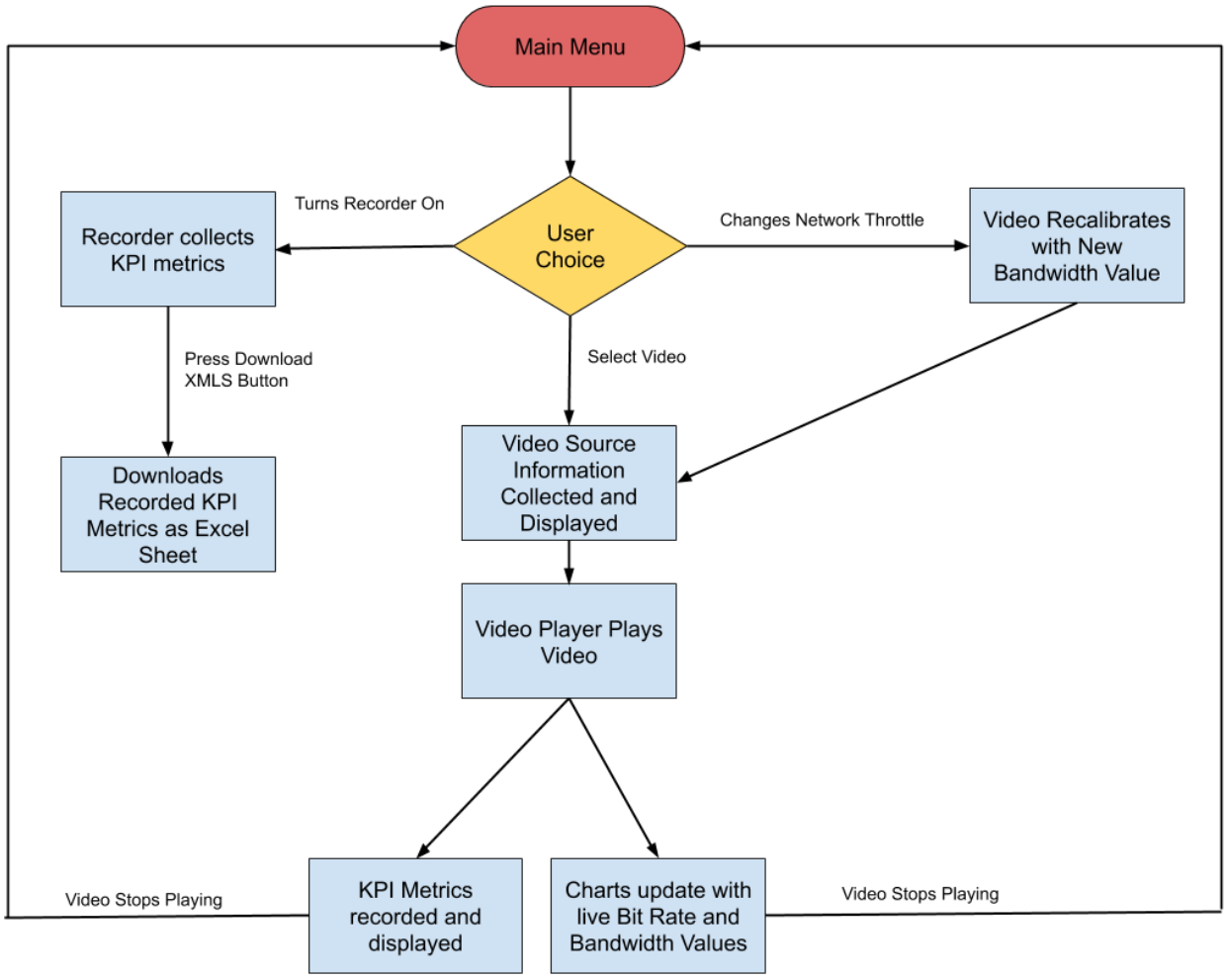
VII.



VIII.



9.3 User Interface Flow Model


Main Menu
User Choice
Turns Recorder On
Recorder collects KPI metrics
Changes Network Throttle
Video Recalibrates with New Bandwidth Value
Press Download XMLS Button
Downloads Recorded KPI Metrics as Excel Sheet
Select Video
Video Source Information Collected and Displayed
Video Player Plays Video
KPI Metrics recorded and displayed
Charts update with live Bit Rate and Bandwidth Values
Video Stops Playing
Video Stops Playing

10. Requirements Validation and Verification

Guidelines for functionality that were stated on the SRS:

1. The software shall take a pre-set amount of videos to be chosen by the user.
2. The software shall perform analysis by recording its data.
3. The software may have an alternate button for alternating between HLS.js or Shaka video player.
4. The software may be able to play both HLS and DASH videos even if they aren't supported by the player.
5. The software shall display the different KPI metrics while running the web page.
6. The software shall display the buffering graphs when video is playing.
7. The software shall handle errors by returning invalid data.
8. The software should be able to maintain the video that was previously loaded even if the web browser was reloaded.

The components are satisfied by the methods described above. There are some requirements that we would like to be implemented in a future version of the software.

11. Glossary

ORVP	Open-Source Real-Time Video Player
SRS	Software Requirement Specification
DASH	Dynamic Adaptive Streaming
HLS	HTTP Live Streaming
KPI	Key Performance Indicator
VST	Video Start Time
VMAF	Video Multimethod Assessment Fusion
Broadband	Telecommunication for Data transmission
REPOS	Repositories
JavaScript	An interpreting language not a compiler

12. References

- ExoPlayer: https://github.com/rc728m/ExoPlayer.git
- Exoplayer Demo: https://exoplayer.dev/demo-application.html
- Shaka: https://github.com/rc728m/shaka-player.git
- HLS.js: https://github.com/rc728m/hls.js
- HLS Demo: https://hls-js-dev.netlify.app/demo
- Video.js: https://github.com/rc728m/video.js.git
- Clappr Stats: https://github.com/clappr/clappr-stats
- Mux.js: https://github.com/videojs/mux.js#readme
- Chart.js: chartjs.org
- MyExcel: https://github.com/jsegarra1971/MyExcel