

Software Design Document

for

Open-Source Real-Time Video Player

Version 4 approved

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

AT&T

May 14, 2021

Table of Contents

Table of Contents.....	02
Revision History.....	03
1. Introduction.....	04
1.1. Purpose.....	04
1.2. Document Conventions.....	04
1.3. Intended Audience and Reading Suggestions.....	04
1.4. System Overview.....	04
2. Design Considerations.....	05
2.1. Assumptions and dependencies.....	05
2.2. General Constraints.....	05
2.3. Goals and Guidelines.....	06
2.4. Development Methods.....	06
3. Architectural Strategies.....	07
4. System Architecture.....	08
5. Policies and Tactics.....	09
5.1. Specific Products Used.....	09
5.2. Requirements traceability.....	09
5.3. Testing the software.....	09
5.4. Engineering trade-offs.....	09
5.5. Guidelines and conventions.....	09
5.6. Protocols.....	09
5.7. Maintaining the software.....	09
5.8. Interfaces.....	09
5.9. System's deliverables.....	09
5.10. Abstraction.....	09
6. Detailed System Design.....	10
6.x Name of Module.....	10
6.x.1 Responsibilities.....	10
6.x.2 Constraints.....	10
6.x.3 Composition.....	11
6.x.4 Uses/Interactions.....	11
6.x.5 Resources.....	11
6.x.6 Interface/Exports.....	11
7. Detailed Lower level Component Design.....	12
8. User Interface.....	12
9. Database Design.....	12
10. Requirements Validation and Verification.....	17
11. Glossary.....	18
12. References.....	19

Revision History

Name	Date	Reason For Changes	Version
First Draft	12-06-2020	Created	1
Second Draft v1	03-16-2021	Updating progress of the project	2
Second Draft v2	04-21-2021	Updating	2.1
Third Draft	05-01-2021	Updating for final approval	3
Final Draft	05-14-2021	Final revision	4

1. Introduction

1.1 Purpose

The purpose of the Software Design Documentation (SDD) is to outline the design of our Open-Source Real-Time Video Player. This document serves to expand the knowledge and understanding of our project's functionalities and purpose. We will illustrate the purpose, Level 0 DFD, Level 1 DFD, user interface, and the libraries used in our software.

1.2 Document Conventions

This document contains headings, subheadings and body text, all using the Calibri font. All headings are size 20pt and are bold, all subheadings are size 14 and are bold, and all body text is size 12pt and uses 1.15 spacing for the ease of readability. Bullet Points are used to list or outline specifications for ease of comprehension.

1.3 Intended Audience and Reading Suggestions

This documentation is intended for developers and users. It's broken down into documentational components in order to make it easier to understand our project in detail. To have the best understanding, it's best to have a general understanding of video streaming, live-streaming, and live-broadcasting. This document contains the overall idea of how our project works as a whole. A suggestion for users is to look into the sections of understanding the purpose, system overview, user interface, and acronyms, in order to better understand the project.

1.4 System Overview

Our software builds upon an open-source web-app available on GitHub. Our software provides information on the performance of the web-app by recording Key Performance Indicators(KPIs) such as Video Start Time, Rebuffering Ratio, Rebuffering Duration, Bit Rate, and various network data. Other important functionality includes automated testing, recording of data to an excel sheet, and throttling of the network to simulate various conditions in order to test the behavior of the web-app. All features are for the purpose of examining the behavior and capability of the web-app in order to make improvements to the client's video streaming services.

2. Design Considerations

AT&T initially proposed four different video web players for research and analysis. After preliminary testing and new information provided by AT&T, the number of web players were reduced to HLS.js, chosen by the team, and Shaka-player, chosen by AT&T due to the fact that we would only be working with HLS format videos and only these two players meet all requirements specified. Due to the difficulties involved with Shaka-player, the team opted to first focus on HLS.js. Due to the fact that both players are written in Javascript, the project is also in turn, written primarily in Javascript with uses of html and css for frontend development as necessary. Throttling of the network, a major component of the project, was found to be too difficult and thus the project instead simulates network conditions utilizing Selenium, a web browser automation tool also utilized to collect and record data by the project, to interact with the browser's development tools which include network simulation capabilities.

2.1 Assumptions and Dependencies

As this is a project involving building upon a web-app, a sufficient broadband connection is required for testing and development. Furthermore, the project utilizes the latest version of the Google Chrome web browser and the addition of a matching webdriver is required as a dependency.

2.2 General Constraints

General constraints are placed for the protection of the software and users. As this is a product intended for the use by AT&T, steps are taken such that the product's development adheres to their specifications.

- Some pre-loaded AT&T video urls require access from the AT&T team
- The Hls.js video player does not support DASH videos
- Security Considerations
 - We do not store any data that is collected by users, it is only downloaded on the user's local computer
- Web browser:
 - Google Chrome

2.3 Goals and Guidelines

Goals:

- Create a program which records accurate metric data
- Simulate specified changing network conditions in real-time
- Automate testing with specified conditions and parameters
- Store the data in a legible format for later access

Guidelines:

1. The software and accompanying documentation shall be completed and delivered by the end of Spring Semester 2021

2.4 Development Methods

- Task 1:
 - Create Clone Github repository of AT&T's private HLS.js repository
 - Create Branches for specified KPI metrics and functions
 - Video Start Time
 - Rebuffering
 - Adaptive Bitrate
 - Network Throttling
 - Implement KPI Metrics and features for testing
- Task 2:
 - Collect Test Data for Analysis
 - Utilization of Selenium for automated testing
 - Record all relevant data to excel file
 - Verify Accuracy of KPI metrics
 - Compare data produced by members for consistency
- Task 3:
 - Merge Branches to Master Branch
 - Utilization of Selenium for automated testing
 - Reverify all functionalities implemented
 - Prepare Deliverables to be presented to client

3. Architectural Strategies

For the web application we built on top of Netlify's open-source HLS.js demo and used Visual Studio's VSCode IDE to augment the application. The languages used include: JavaScript, HTML and CSS. HLS.js also uses typescript in the original source code, and the backend uses JSON to manage data. This data can be created, altered, and removed by the user and provides a platform for testing and evaluating results. Some of the libraries we use include:

HLS.js

Selenium

Mozilla MDN

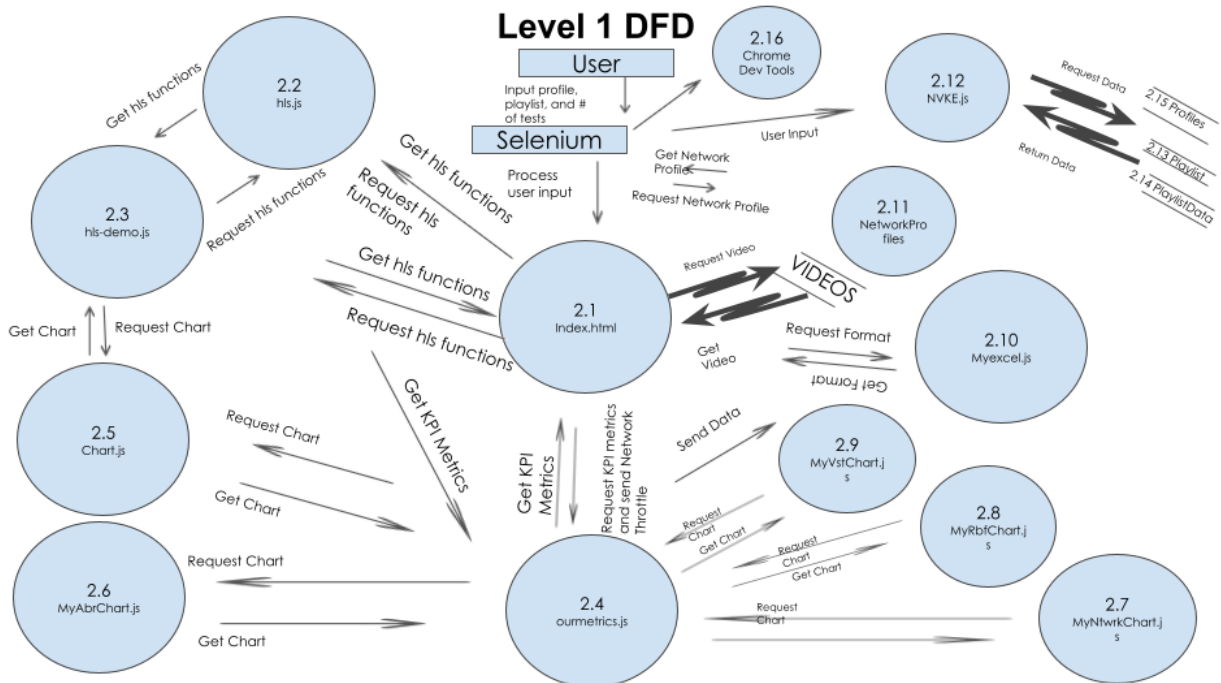
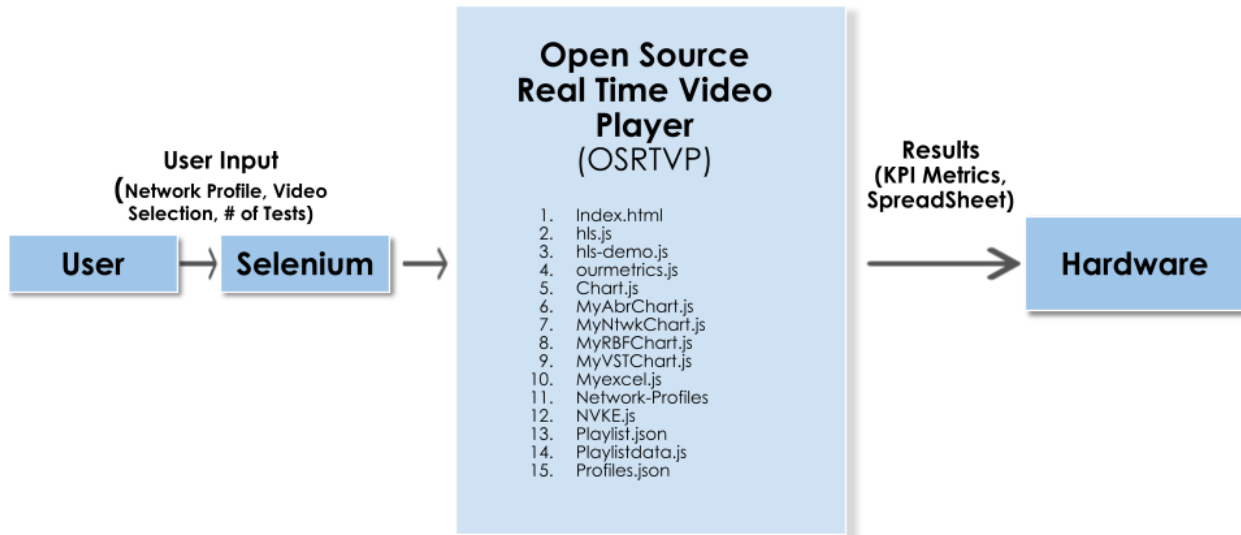
Node.js

MyExcel

Chart.js

4. System Architecture

Level 0 DFD



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

- HLS.js
- Selenium
- Mozilla MDN
- Node.js
- MyExcel
- Chart.js

5.2 Plans for ensuring requirements traceability

- By dividing the client's required KPI's into individual branches on Github, we ensured that each feature was implemented to our Client's satisfaction before merging the final product.
- Weekly meetings were hosted to keep track of latest code reviews on each branch, and every team member tested the application after every Github push.

5.3 Plans for testing the software

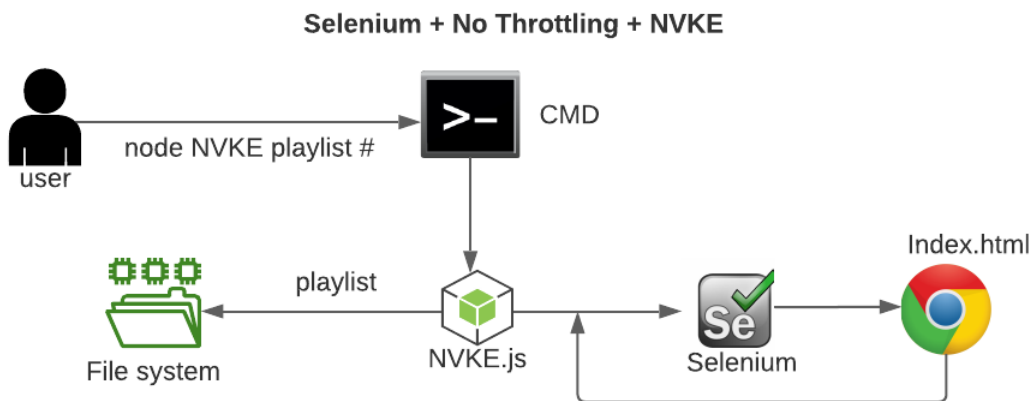
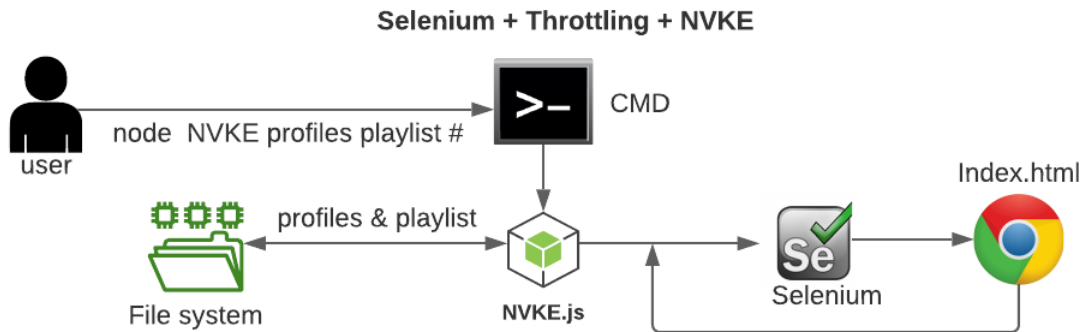
- Custom Business Practices

5.4 Plans for maintaining the software

- Once the project is completed on Spring 2021, 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.



The system requires several arguments to launch the Network Video KPI Emulator. Users can launch the Network Video KPI Emulator with throttling or no throttling.

The arguments are: node, NVKE, playlist, profiles, and number of tests desired.

node: is to run the application outside of a browser and instead start the application directly in the command line.

NVKE: A JavaScript file where all selenium code resides to run the automation.

profiles: A json file where users can append or remove network profiles they want to apply. If the profiles.json file is not passed in as an argument then that depicts no throttling.

```
{
  "NetworkProfiles": [
    { "name": "wifi" },
    { "name": "cellular" },
    { "name": "ss" }
  ]
}
```

playlistdata: Not a requirement to be passed in the command line. This Javascript file is the file where users append or remove video urls.

```
var att_playlist = {
  spr: {
    url: 'https://att-advavtech-output.s3.amazonaws.com/csula/HLS-TV1_At_S_SPR_1080p24.m3u8',
    description: 'SPR - Saving Private Ryan - adaptive qualities',
    abr: true
  },
  cnn: {
    url: 'https://att-advavtech-output.s3.amazonaws.com/csula/HLS-TV2_CNN-2597-03-131.m3u8',
    description: 'CNN - CNN News - adaptive qualities',
    abr: true
  },
  wbv: {
    url: 'https://att-advavtech-output.s3.amazonaws.com/csula/HLS-TV3_E1_S_WVB_1080p60.m3u8',
    description: 'WBV - Womans Beach Volleyball - adaptive qualities',
    abr: true
  },
  pjm: {
    url: 'https://att-advavtech-output.s3.amazonaws.com/csula/HLS-TV4_PJMasks_1080p24.m3u8',
    description: 'PJM - PJ Masks - adaptive qualities',
    abr: true
  }
};
```

playlist: A json file where the user determines what videos to play from the playlistdata JavaScript file.

```
{
  "Vectors": [
    { "name": "spr" },
    { "name": "cnn" },
    { "name": "wbv" },
    { "name": "pjm" }
  ]
}
```

7. Detailed Lower level Component Design

Does not apply.

8. Database Design

Does not apply.

9. User Interface

9.1 Chrome Overview of User Interface

When the user opens the application on a Chrome browser, they will see the entirety of the NVKE and its UI components. Visually going from top to bottom, the user will first see the Video Source Information which provides 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 are the KPI real time charts. These allow the user to view the metrics changing in real time. Lastly, the HLS-DEMO's real time metrics.

9.2 Screen Frameworks or Images

- I. General Overview
- II. Video Source Information
- III. Video Player Screen
- IV. CSULA implemented KPI Metric Stats
- V. Netlify's Hls-demo's real time metrics and features

I.

hls.js
Merged_Demo_v1

Enable streaming:
Auto-recover media-errors:
Dump transmuxed HMP4 data:
Metrics history (max limit, -1 is unlimited): -1
HTML video element width: 240px
Current player size: 825 x 464.1
Current video resolution: 1920 x 1080

file:///C:/Users/user/Documents/GitHub/CSULA/News20folder/ABR/demo/hls-Demo/index.html?src=https%3A%2F%2Fatt-advavtech-output.s3.amazonaws.com%2Fcsula%2FHLS-TV1_At_S_SPR_1080p24_e3u8d8demoConfig-ey71beF1b0V1dH01V61pbec10n9ydMuz1eF1d59S2hVvdeyR03yb3110nrydMz1eR1bxbnTVAR1jpeYkxzZSw1b0V2Zkx0YXhwaW5nIjotR5w1bG1tAXRNZ8yAMz1jotR5w-

Persist Apply

II.

SPR - Saving Private Ryan - adaptive qualities

https://att-advavtech-output.s3.amazonaws.com/csula/HLS-TV1_At_S_SPR_1080p24_e3u8d8demoConfig-ey71beF1b0V1dH01V61pbec10n9ydMuz1eF1d59S2hVvdeyR03yb3110nrydMz1eR1bxbnTVAR1jpeYkxzZSw1b0V2Zkx0YXhwaW5nIjotR5w1bG1tAXRNZ8yAMz1jotR5w-

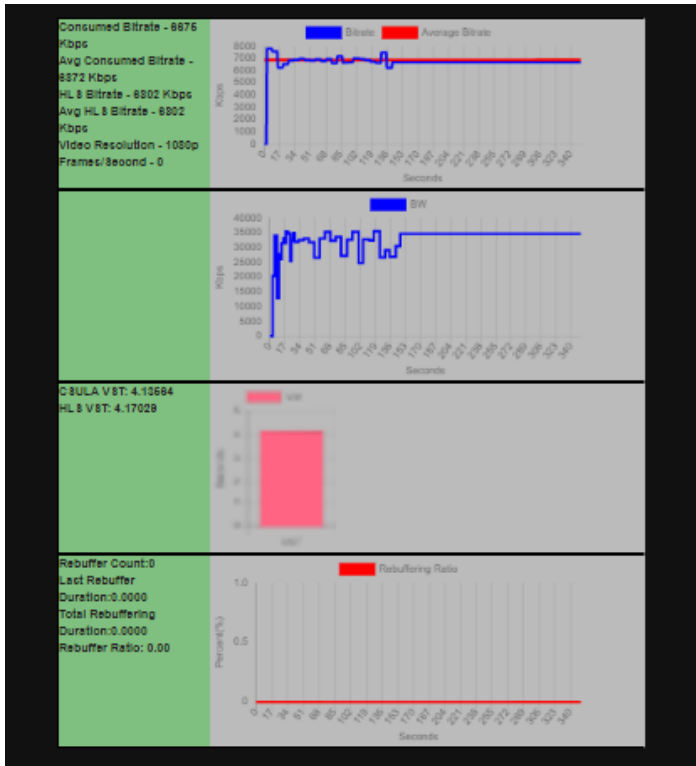
```
1 {  
2   "debug": true,  
3   "enableWorker": true,  
4   "liveBackBufferLength": 900,  
5   "abrMaxWithRealBitrate": true  
6 }
```

Persist Apply

III.



IV



V

Status:

```
0.616 | Loading https://att-advavtech-output.s3.amazonaws.com/csua/HLS-TV1_At_S_SPR_1080p24.m3u8
0.638 | Loading manifest and attaching video element...
```

Error:

Playback Timeline Quality-levels Audio-tracks **Real-time metrics** Buffer & Statist...

toggle sliding/fix window


window ALL 2s 5s 10s 20s 30s 60s 120s

Window Zoom In Window Zoom Out

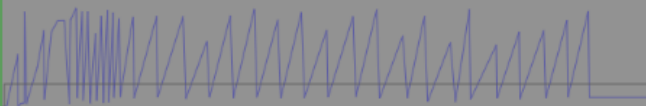
<<< Window Slide Window Slide >>>

metrics link metrics permalink copy metrics to clipboard


play pos/buffer
last pos: 142526 ms
last buffer: 73732 ms
max buffer: 216258 ms
nb samples: 112



last bitrate: 2.95Mb/s
min bitrate: 0.05Mb/s
max bitrate: 29.93Mb/s
min/last/max level: 7/7/7
nb level switch: 0
average level: NaN



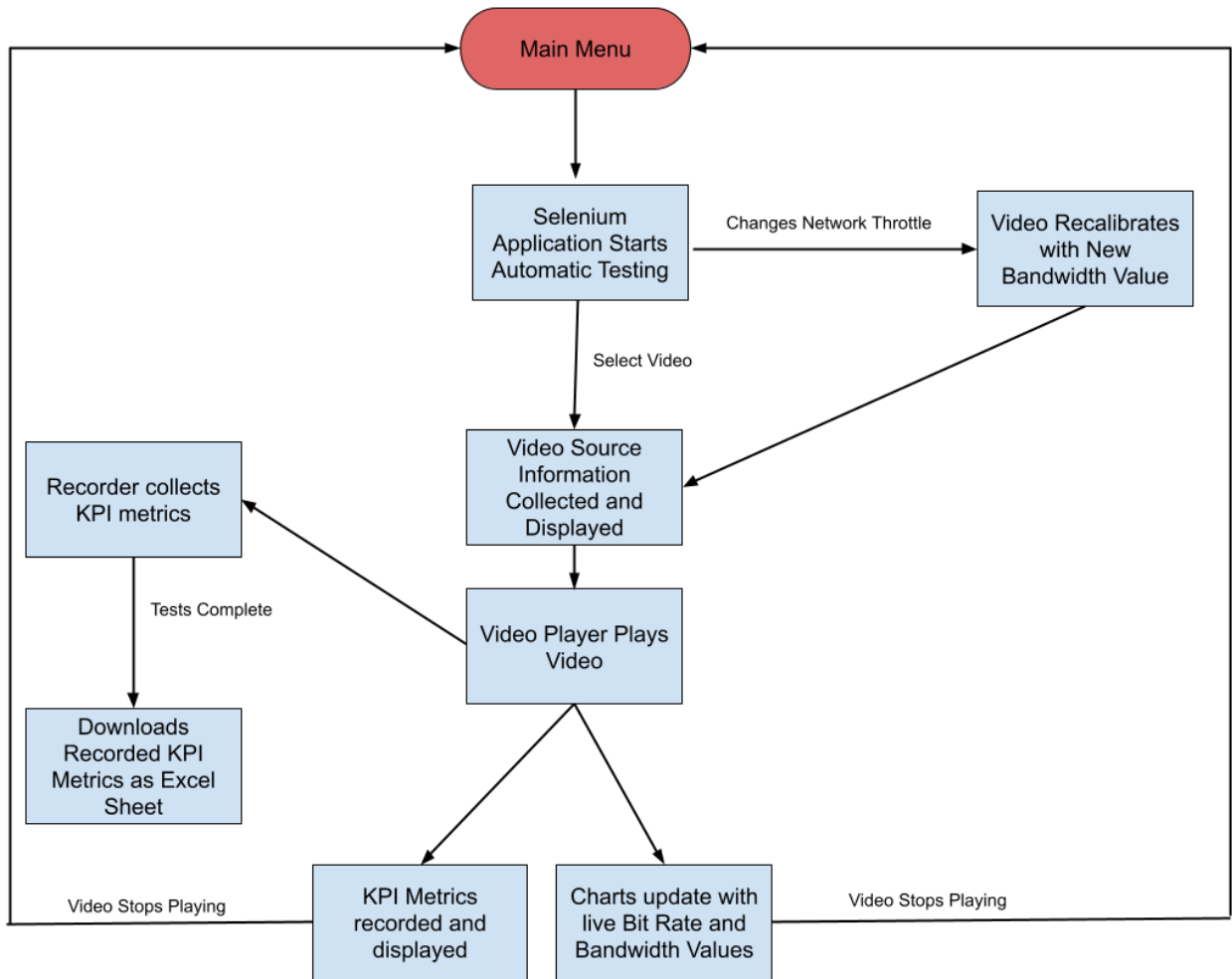
play pos/buffer zoomed
[422291.41499998514, 442291.41499998514]
focus time: 422291.41499998514
focus position: 142526 ms
focus buffer: 73732 ms



video event time [duration]

load event start - end [latency loading parsing appending] size bitrate

9.3 User Interface Flow Model



10. Requirements Validation and Verification

Guidelines for functionality that were stated on the SRS:

1. The software shall take a pre-set json file with a list of video urls by the user.
2. The software shall take a pre-set json file with a list of network profiles by the user.
3. The software shall perform analysis by recording data on the video playback.
4. The software shall locally download an xlsx file containing all the data recorded.
5. The software shall be able to play HLS videos.
6. The software shall display the different KPI metrics while running the web page.
7. The software shall plot the video start time.
8. The software shall display the buffering graphs.
9. The software shall display the average bitrate graphs.
10. The software shall display the bandwidth and network profile graphs.
11. The software shall handle errors by returning warnings with invalid data.
12. The software shall 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 over HTTP
HLS	HTTP Live Streaming
KPI	Key Performance Indicator
VST	Video Start Time
RBF	Rebuffering
ABR	Adaptive Bitrate
NTWK	Network
VMAF	Video Multimethod Assessment Fusionsa
Broadband	Telecommunication for Data transmission
REPOS	Repositories
JavaScript	An interpreting language not a compiler
SS	Stream Saver
Selenium	Automation for web applications
QA	Quality Assurance
Bandwidth	The maximum amount of data transmitted over an internet connection in a given amount of time in mbps
UI	User Interface

12. References

- HLS.js: <https://github.com/video-dev/hls.js/>
- HLS Demo: <https://github.com/video-dev/hls.js/tree/master/demo>
- Selenium: <https://www.selenium.dev>
- Chart.js: chartjs.org
- Mozilla MDN: <https://developer.mozilla.org/en-US/>
- MyExcel: <https://github.com/jsegarra1971/MyExcel>
- Node.js <https://nodejs.org/en/>
- Javascript: <https://www.javascript.com>
- Chrome Developer Tools: <https://developers.google.com/web>