VAASAN AMMATTIKORKEAKOULU
UNIVERSITY OF APPLIED SCIENCES

Anh Minh Hoang

SMART TELEVISION APPLICATION DEVELOPMENT

Technology and Communication
2022

ACKNOWLEDGEMENTS

First of all, I would like to express my gratitude to my supervisor Anna-Kaisa Saari for her patience, motivation, enthusiasm and immense knowledge. Her guidance and advice were invaluable during the research and writing of this thesis.

Besides my supervisor, I would like to thank all my teachers at VAMK, especially Dr. Ghodrat Moghadampour and Mr. Timo J Kankaanpää for helping, supporting and teaching me a lot during my university.

Mr. Ari Pöyhtäri, Mr. Erkki Salminen and Mr. Mika Kanerva deserve special appreciation for providing me with summer trainee chances at Sofia Digital Oy and allowing me to work on a variety of intriguing projects.

Last but not the least, I am extremely grateful to my parents for their love, care and sacrifices in educating and preparing me for the future.

VAASAN AMMATTIKORKEAKOULU
UNIVERSITY OF APPLIED SCIENCES
Information Technology

ABSTRACT

Author	Hoang Anh Minh
Title	Smart television Application Development
Year	2022
Language	English
Pages	46
Name of Supervisor	Anna-Kaisa Saari

The topic of this thesis is the development of the smart television application providing streaming services such as live television, movies, tv shows and radios that run on HbbTV (Hybrid Broadcast Broadband TV) and Smart TV.

The professional background needed for the application contains an understanding of HbbTV and Smart TV development, programming language skills in JavaScript, familiarity with the Vue framework and third-party players that support adaptive media formats.

The main objective of the thesis was to develop a television application that allows the user to experience and watch the VOD (Video on Demand), live streaming and radio with different media formats (MPEG-DASH, HLS and Smooth Streaming) and search the program and add to their favorite list. In addition, the player interface is very important for the user experience so the application contains the thumbnail images on the progress bar, the subtitles and the full control buttons for the player.

Keywords Smart TV, HbbTV, VOD, VUE.js video media formats

CONTENTS

LIST OF FIGURES AND TABLES

LIST OF SNIPPETS

LIST OF ABBREVIATIONS

OS	Operating System
GUI	Graphical User Interface
DRM	Digital Right Management
SDK	Software Development Kit
HTTP	HyperText Transfer Protocol
NPM	Node Package Manager
HBBTV	Hybrid Broadcast Broadband TV
VOD	Video On Demand
HLS	HTTP Live Streaming
DASH	Dynamic Adaptive Streaming over HTTP
HTML	HyperText Markup Language
CSS	Cascading Style Sheets
API	Application Programming Interface
DOM	Document Object Model
MP4	MPEG-4 Part 14
CDN	Content Delivery Network

1 INTRODUCTION

Especially during the COVID-2 pandemic certain parts of the world were on lockdown, which caused people to get locked in their homes. At home, people spend the majority of their time on digital media. Entertainment, news, games, applications and websites have all seen an increase in visitors. It is no longer difficult to determine which digital media is the best for each function. The rise of the Internet and technology has made digital material more accessible in a variety of formats, making people's lives easier. Even though the younger generation and the wealthy are increasingly using smartphones, tablets and other forms of portable devices for gaming and social media access, a significant section of the public continues to expand their internet usage for watching digital video content.

Television and broadcasting were at the top of the entertainment rankings for several decades. The popularity of television waned as the Internet became more prevalent, although it was not completely replaced. The developers grew into the smart TV application development generation, which entices clients with a diverse set of features and a simple user interface. /22/

According to a survey commissioned by Traficom*, as seen in figure 1, internet TV streaming services surpassed traditional linear TV viewing among Finns under 45 in 2021. The total viewing time was still evenly balanced as late as spring last year, but by October, the percentage of linear TV had declined to 46%, while the share of streaming services, such as Yle Areena and Netflix, had climbed to 54%. Linear TV viewership has declined dramatically in this age group since 2014 when streaming services accounted for 22% of total TV viewing time. /20/

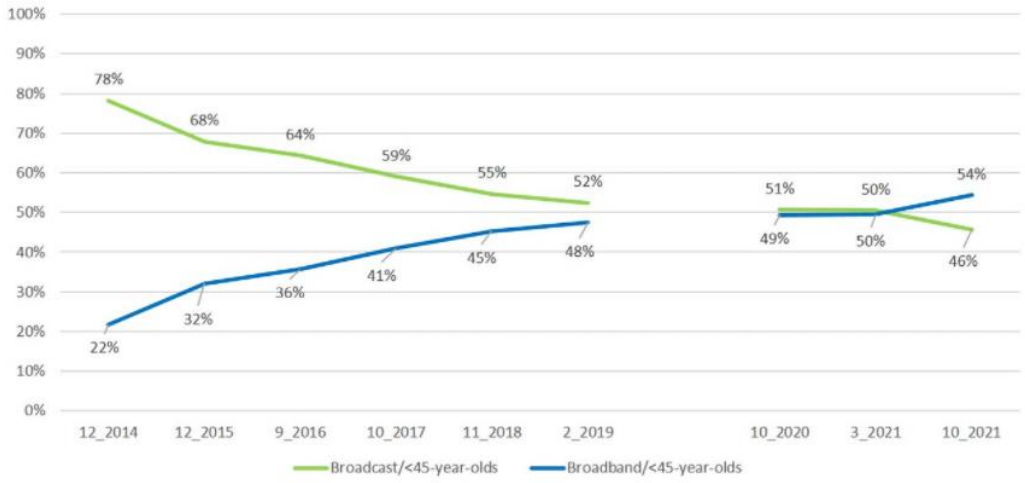

Figure 1. Traficom's Internet TV surpasses survey

The goal of this thesis is to create a television application that allows users to view VOD (Video on Demand), live streaming, and radio in various media types (MPEG-DASH, HLS and Smooth Streaming). There are numerous screens in the application (menu, program, player and screen screens). Furthermore, because the player interface is so vital to the user experience, the program includes thumbnail images on the progress bar, subtitles, and full player control buttons.

The thesis consists of four sections. The first section is an overview of the use of television and broadcasting for entertainment and the introduction of the thesis and the application. The second section reviews the television application development in multiple parts (operating system, HbbTV, resolutions, technologies, digital rights management, media formats and the navigation management). The third part shows how to set up build the smart tv application. The last part is the conclusion that presents the purpose of the thesis and the most challenge when developing the application.

2 TELEVISION APPLICATION DEVELOPMENT

A smart TV is an internet-connected television that includes on-demand programming via different applications such as BBC iPlayer, ITV Hub and All 4. It also includes access to streaming services such as Netflix and the ability to link television to other wireless devices such as smartphones. /16/

This chapter introduces key concepts related to smart tv application development, such as technologies, media formats, platforms and DRM content support.

2.1 Smart TV operating system

The television operating system is bootable software that allows users to access and operate advanced features and linked devices on smart TVs and set-top boxes. TA television OS has a graphical user interface (GUI) for interaction, just like the personal computer has. In essence, smart TVs can be called Internet-connected entertainment-specialized computers that can wirelessly link to a variety of devices. /1/

Users of TV operating systems can access not only satellite or cable TV channels, but also on-demand video services. Pictures, music and video content stored on connected storage devices or streamed can also be accessed by the systems. /1/

2.1.1 Most popular operating system on a smart TV

The most popular operating systems on smart televisions are WebOS, Android TV and Tizen OS.

1. **WebOS**: Despite popular assumptions, the original Linux kernel-based system was created and developed by a Palm company, which was later sold to Hewlett-Packard and then to LG Electronics. In 2009, WebOS was initially used on a PDA and although LG's operating system, it was first deployed on television in 2014. WebOS has proven to be an absolute protagonist since its debut and it has a simple and unobtrusive user interface./2/

The evolution of WebOS has been ongoing over time. For example, support for Alexa, Google Assistant, Apple HomeKit 2 and a variety of home automation tasks have been developed. New applications include Apple TV, Disney+ and a slew of others./2/

2. **Android TV**: The Android operating system is one of the most widely used mobile operating systems. The Android TV operating system, created by Google, offers everything from the Android phone to the large screen./2/

   Sony, Philips and Sharp are among the companies that employ the Android TV operating system. One of the most appealing features of an Android TV is the ability to cast entertainment from users' smartphones to their TV. The home screen offers a clean, easy-to-navigate design and you can connect to the Google Play store can be connected to download all favorite applications. /2/

3. **Tizen OS** is a Samsung-developed system. Tizen OS is also based on the Linux operating system. The original project began in 2011 with the implementation of a platform for HTML5 apps for mobile devices. Tizen OS has made its way into smartphones and smartwatches as the market has evolved. It has been substantially modified with heavy inspiration from WebOS and landed on televisions since 2015, where it has also become the most popular smart-TV system in the world thanks to Samsung TV sales. /2/

   The strength of Tizen OS has been the ease of use, functionality and App store offered over the years. Aside from the WebOS pointing system, there is not a single feature that is not n available on this system, except Tizen's multi-view feature, which allows users to view two sources at once. /2/

2.1.2 Software Development Kit for the Operating System

The software development kit (SDK) is a term used to describe a collection of tools that can be used. The SDK, sometimes known as a devkit, is a collection of software development tools for a certain platform, which often includes building blocks,

debuggers and a framework or group of code libraries, such as a set of operating-system-specific procedures (OS). /3/

For developing the Smart TV operating system webOS SDK, Android Studio and Tizen Studio are needed. The webOS SDK is a set of tools for developing web applocations for the webOS TV platform as well as an emulator for testing.

2.2 Hybrid Broadcast Broadband TV

The hybrid broadcast broadband TV, HbbTV, is a global movement aiming at harmonizing the distribution of entertainment services to customers via linked TVs, set-top boxes and multiscreen devices. Industry leaders created the HbbTV specification to improve the video user experience for customers by enabling novel, interactive services over broadcast and broadband networks. The protocol incorporates aspects from several specifications such as OIPF, CEA, DVB, MPEG-DASH and W3C. /18/

HbbTV applications are launched like DVB linear TV channels overlays, whereas Smart TV applications must be downloaded from a TV app store such as Google Play for Android or the App Store for iOS. /18/

2.3 Smart TV Resolutions

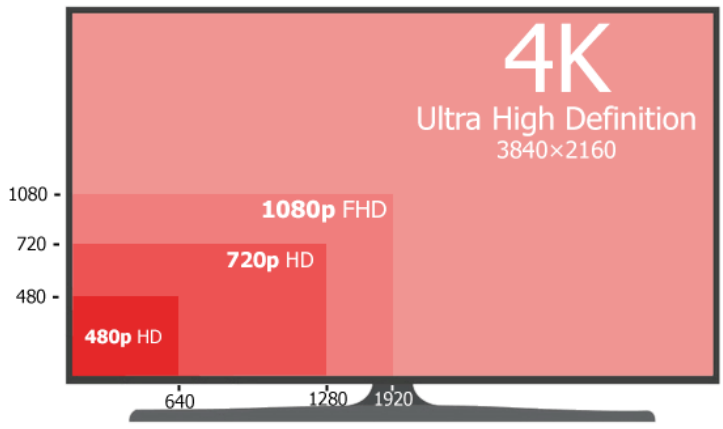

Figure 2. Smart TV resolutions

Smart TV resolutions are usually 16x9, as seen in figure 2 and the most common are 1280x720 (HD) and 1920x1080 (Full HD). The TV and monitor industry is growing day by day, so TV resolutions are moving to 4k, 8k and even 10k. Because the developers know the exact size of the screen, they do not need to implement resizable and responsive application screen by screen, instead they just need to create a variable for each screen into this variable will change according to the screen resolution. For example, if the variable for 720x1280 is 1 unit, then 1080x1920 is one 1*1080/720 equal to 1.5 units and the same for other resolutions. Common examples of television resolutions are shown in table 1. /4/

Table 1. Television resolutions

Resolution	Description	Aspect Ratio	Width in pixels	Height in pixels
720p	High Definition (HD)	16:9	1280	720
1080p	High Definition (HD)	16:9	1920	1080
2160p	Ultra-High Definition (UHD) or 4K	16:9	3840	2160

2.4 Technologies for smart tv application

This chapter briefly introduces some of the technologies that can be used in television applications. Building and developing the smart tv application involves technologies from both the server-side and client-side. On the client-side, besides the technologies and framework used, the application needs the third-party players that be adaptive media formats like HLS or Dash.

2.4.1 Frontend Technologies

The application running on television mainly uses HTML, CSS, JavaScript, TypeScript and DOM. The exceptions are Native Android TV Apps which are built using Java and Apple TV Apps, which are built using Swift. Besides using Vanilla JavaScript (using plain JavaScript without any additional libraries and frameworks), the

applications nowadays are more and more developed using frameworks due to the following reasons:

+ The framework has built-in features that are common to the application/software. For example, almost all e-commerce websites need registration, login, user data management, etc. The framework has built-in these features and programmers only need to put them to use when building a website.

+ Frameworks help programmers to save time and effort when developing software/applications.

+ Frameworks follow application products to inherit standardized features and structures. It makes operating and maintaining or /troubleshooting the application easier.

+ Frameworks allow users to expand arbitrarily based on what the Framework has to offer. Developers can extend features by selectively overriding existing classes or writing new functionality on top of the Framework, as long as certain standards are followed.

JavaScript is aprogramming language that is one of the core technologies of the World Wide Web, alongside HTML and CSS. JavaScript is a high-level, often just-in-time compiled language that conforms to the ECMAScript standard. It has dynamic typing, prototype-based object orientation and first-class functions. JavaScript is multi-paradigm, supporting event-driven, functional and imperative programming styles. It has application programming interfaces (APIs) for working with text, dates, regular expressions, standard data structures and the Document Object Model (DOM). /5/

React (also known as React.js or ReactJS) is an open-source front-end JavaScript library for building user interfaces specifically for single-page applications. React allows developers to create reusable UI components and makes the website to be fast, simple and scalable. /6/

Vue is a framework for creating progressive user interfaces and built from the ground up to be incrementally adoptable, unlike other monolithic frameworks. The core library is focused on the view layer only and is easy to pick up and integrate with other libraries or existing projects. /7/

Svelte is a free and open-source front-end compiler created by Rich Harris and maintained by the Svelte core team members. Not similar to React or Vue, the Svelte application generates the necessary code to manipulate the DOM within each component, which may reduce the size of transferred files and give better client startup and run-time performance. /8/

2.4.2 Backend Technologies

The backend is all the parts that support the operation of a website or application that cannot be seen by the user. Arguably the backend is like the human brain. It processes requests, and commands and selects the correct information to display on the screen.

The backend is made up of 3 components: server, application and database. Similar to frontend development, the backend comes with multiple features that are helpful to the developers. These are for example database, architecture, server, API, router and security.

Express.js: is a free and open-source framework for Node.js. Express.js is used to design and build web applications and APIs simply and quickly. /9/

2.4.3 Third-party Player Technologies

HTML has three supported video formats, WebM, OGG and MP4. To run these on Smart TV, third-party player technologies, such as Shaka Player and Dash.js are needed. Some televisions do not support video tag on HbbTV, so the Open IPTV Forum (OIPF) has developed a solution by using object tag to play the video.

Shaka Player is an open-source JavaScript library for adaptive media. It plays adaptive media formats such as DASH and HLS in a browser, without using plugins or

Flash. Instead, Shaka Player uses the open web standards Media Source Extensions and Encrypted Media Extensions. /10/

Dash.js is a reference client implementation for the playback of MPEG DASH via JavaScript and compliant browsers. Dash.js supports the in-band events, multiple periods, and cross-browser DRM /11/

2.5 Digital Rights Management (DRM)

Digital rights management (DRM) is a way to protect copyrights for digital media. DRM is essentially a way of controlling the copyright of digital data content based on encryption. By using DRM, the copyright holder can control how users (buyers of digital products) use their products. For example, control can be done by limiting the number of times to install the file, limiting the time of use or limiting the object of using the file./12/

DRM works is based on encrypting file content with a secret key. When there is a need to use the file, a separate application to read the file will proceed to decrypt the file. After the file has been decrypted, it can be used in the application. The functionality of DMR is shown in Figure 3. /13/

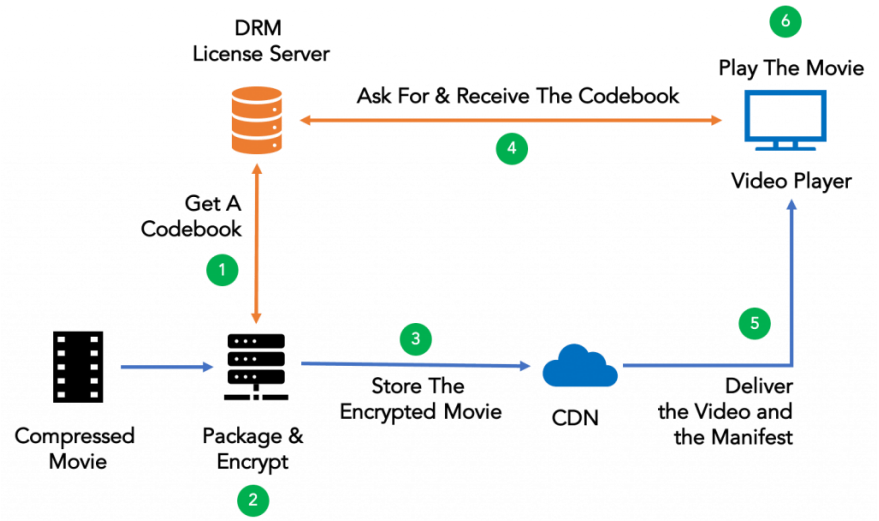

Figure 3. Digital Rights Management functionality

Encryption

First, the package file will send a request to the DRM System to receive the encryption key (Figure 3 [1][2][4]). After receiving the encryption key, it is used to encrypt the file, Sometimes the encryption key is generated by the person who packaged the file. This key is then stored on the DRM System (Figure 3 [3]). /13/

Decryption

The application will download the encrypted content. After having the Encrypted file, the application will ask to receive the Decryption key from the DRM System (Figure 3 [4]). If the credentials are accepted, the DRM System will resend the decryption key. The application will decrypt the DRM file with this decryption key for users to use (Figure 3 [5][6]). /13/

There most popular types of DRM are Google's Widevine, Apple's FairPlay and Microsoft's PlayReady.

2.6 Media Formats

Any video file has two components: a container and a codec. A video format is a file that contains audio, video, subtitles, and other metadata. HTTP Live Streaming and Dynamic Adaptive Streaming over HTTP are the main formats used for delivering adaptive bitrate video. /23/

2.6.1 HTTP Live Streaming

HTTP Live Streaming, which is also known as HLS, is an HTTP-based adaptive bitrate streaming communications protocol developed by Apple Inc. HLS was released in 2009. First, HLS breaks down the video files into smaller HTTP-based file downloads, then delivers them using the HTTP protocol. Initially, HLS was supported only on iPhone but now it has become the most popular protocol and HLS is widely supported by most of the available devices and browsers. /15/

HLS protocol includes various features like subtitles, fast forward and rewind, alternative audio and video, contingency alternatives, ads insertion, and content protector. /15/

First an HLS stream comes from a server where the media file is kept (in on-demand streaming) or where the stream is created (in live streaming). Because HLS is based on HTTP, it may be started by any regular web server. When client devices request the stream, the encoded video chunks are sent over the Internet to them. Typically, a content delivery network (CDN) will help disseminate the stream to geographically different places. A CDN will also cache the stream so that it may be served to clients even faster. The client device — for example, a user's smartphone or laptop – receives the stream and plays the video. The client device utilizes the index file as a guide for putting the video together in the right order, switching from higher to lower quality (and vice versa) when appropriate./21/

2.6.2 Dynamic Adaptive Streaming over HTTP

Dynamic Adaptive Streaming over HTTP, which is also known as DASH or MPEG-DASH, is an adaptive bitrate streaming technique that enables high-quality streaming of media content over the Internet delivered from conventional HTTP web servers. Similar to Apple's HTTP Live Streaming (HLS) solution, MPEG-DASH works by breaking the content into a sequence of small segments, which are served over HTTP./17/

In MPEG-DASH the video file is divided into smaller segments by the origin server, each of which is a few seconds long. The server also creates an index file for the video parts, similar to a table of contents. The segments are then encoded, which means they are presented in a way that multiple devices can understand. Any encoding standard can be used with MPEG-DASH. The encoded video chunks are pushed out to client devices via the Internet once consumers begin watching the broadcast. A content delivery network (CDN) almost always aids in the more efficient distribution of the stream. When a user's device receives streamed data, it

decodes it and plays the video back. To respond to network conditions, the video player automatically converts to a lower or better-quality picture. For example, if the user currently has very limited bandwidth, the movie will play at a lower quality level that requires fewer data to be downloaded over HTTP./17/

2.7 Navigation Management

On a computer, a mouse can be used to navigate through web pages, but televisions do not have a mouse available. Instead, navigation needs to be done with the help of a television remote. The navigation on TV remotes is referred to as the "D-Pad" (Directional Pad). The most popular means for a user to navigate around a TV interface are the up, down, left and right arrow buttons. A pointer input (magic remote) may be provided in some circumstances, such as with LG TVs and Apple TV features a directional touch pad./4/

3 BUILDING SMALL SMART TV APPLICATION

3.1 Setup and Development

In order to develop and implement the smart tv application, the development machine (computer or laptop) has to have NodeJS pre-installed. Yarn and Vue CLI can be installed globally by running **npm install -g yarn** and **npm install -g @vue/cli**.

3.2 Size of the Application

The Smart TV resolutions are usually 16x9 so the default application size should be 1280px for width and 720px for height. The ratio variable needs to be defined for the bigger screen. For example, the ratio for the screen 1920x1080 is:

$$ratio = 1 \times \frac{1920}{1280} = 1.5$$

For example, if the position of the navbar is 100px from the left, it can be styled to a 100px x ratio to make the navbar responsive on the different television.

3.3 Navigation and d-pad Focusable

There are two ways to install and use the JavaScript Spatial Navigation. The first way is to install the Spatial Navigation from npm by running the command**: yarn add spatial-navigation-js**. Another way is copying **spatial_navigation.js** from URL https://github.com/luke-chang/js-spatial-navigation/blob/master/spatial_navigation.js to the project folder **/src/assets/js/spatial_navigation.js**. The application will follow the second way because the developer can customize, change, or add any properties or any behaviors of spatial navigation to be suitable for the project.

The spatial navigation with the section id **default** needs to be initialized and the navigation configuration needs to be done. There are three common and important properties needed on the configuration object: selector, straightOnly and rememberSource. Selector id is needed for navigable elements, whereas **straightOnly** (Boolean) defines the direction for navigation (the vertical or horizontal elements will be navigated when it is true and ignores the diagonal or

oblique elements) and **rememberSource** (Boolean) defines if the previous element focused will be chosen as the next candidate with a greater priority. Spatial navigation configuration is presented in Snippet 1.

```
window.SpatialNavigation.init()
window.SpatialNavigation.add('default', {
  selector: '.dpad-focusable',
  straightOnly: true,
  rememberSource: true,
})
```

Snippet 1. Initialization and adding the spatial navigation

After initializing and defining spatial navigation, the element will be the focusable element when having the **dpad-focusable** class name. Snippet 2 shows how the navbar items were made to be focusable and navigable elements.

```
<router-link
  v-for="item in navbarItem"
  :key="item.title"
  class="navbar-item dpad-focusable"
  :to="item.routerLink"
>
  {{item.title}}
</router-link>
```

Snippet 2. Making navbar items focusable

Making the currently existing navigable elements focusable and focusing on the first navigable element by using two prototypes of SpatialNavigation are **makeFocusable** and **focus**. Both prototypes receive the section id as an argument, as shown in Snippet 3.

```
window.SpatialNavigation.makeFocusable('default')
window.SpatialNavigation.focus('default')
```

Snippet 3. Focus on the section with the specified section id (default)

In some special ways, it is easy to enable and disable the section by id temporarily by using two prototypes of the library **enable** and **disable**. As for previous prototypes, these receive also the section id as an argument, as shown in Snippet 4.

```
window.SpatialNavigation.enable('default')
window.SpatialNavigation.disable('default')
```

Snippet 4. Enabling and disabling the section

Before destroying the application, the spatial navigation should be destroyed, uninitialized and cleared. For this, the SpatialNavigation provides **uninit** and **clear** prototypes, as shown in Snippet 5. Uninitialization resets the variable state with/without unbinding the event listeners.

```
window.SpatialNavigation.uninit()
window.SpatialNavigation.clear()
```

Snippet 5. Uninitialization and clearing SpatialNavigation

3.4 Banner and Program Carousel

To implement the banner and program carousel, the Vue awesome swiper package for the project is needed. It is installed with the following command: **yarn add swiper vue-awesome-swiper.**

```
swiperOptions: {
  slidesPerView: 'auto',
  centeredSlides: true,
  spaceBetween: 1,
  initialSlide: 2,
  lazy: {
    loadPrevNext: true,
    loadOnTransitionStart: true,
  },
  navigation: {
    nextEl: '.swiper-button-next',
    prevEl: '.swiper-button-prev',
  },
  autoplay: {
    delay: 5000,
  },
  loop: true,
}
```

Snippet 6. Setting up banner carousel using vue-awesome-swiper

When the Vue awesome swiper package has been taken into use, **s**etting up the banner carousel can be done. The carousel includes auto play in five seconds, loop, the navigation and the initialization slide. The initialization slide is the second slide, normally the banner carousel has one or three slides or images per view. Settings needed for the banner carousel are shown in Snippet 6.

Snippet 7 below shows how to set up the basics program carousel. The carousel includes five slides per view and a lazy loader for the banner images.

```
swiperOptions: {
  slidesPerView: 5,
  lazy: {
    loadPrevNext: true,
    loadOnTransitionStart: true,
  },
}
```

Snippet 7. Setting up program carousel using vue-awesome-swiper

3.5 Keyboard for Searching Screen

One of the key functionalities of the Smart TV application is the keyboard, which is needed on the search screen. The simple keyboard package can be installed for the project using the command: **yarn add simple-keyboard**.

Once the keyboard package has been taken into use, import simple keyboard can be imported to the file using keyboard component as shown in Snippet 8:

```
import Keyboard from 'simple-keyboard'
import 'simple-keyboard/build/css/index.css'
```

Snippet 8. Import simple-keyboard to the project

Snippet 9 is split into two sections. The first section (from the beginning to "...") shows how to set up the keyboard with four different layouts: lowercase text, uppercase text, number, and char. The second section (codes after "...") also demonstrates how to construct a div element with the class name simple-keyboard and incorporate the simple keyboard user interface into the project.

```
this.keyboard = new Keyboard({
  layout: {
    shift: [
      'q w e r t y u i o p',
      'a s d f g h j k l z',
      'x c v b n m',
      '{shift} {numbers} {chars} {space} {enter} {backspace}',
    ],
    default: [
      'Q W E R T Y U I O P',
      'A S D F G H J K L Z',
      'X C V B N M',
      '{numbers} {chars} {space} {enter} {backspace}',
    ],
    numbers: [
      '1 2 3 4 5 6 7 8 9 0',
      '{abc} {chars} {space} {enter} {backspace}',
    ],
    chars: [
      '_ # + - ! ? @ $ % ( ) . , :',
      '£ & / = * € [ ]',
      '{abc} {numbers} {space} {enter} {backspace}',
    ],
  },
}
...
<template>
  <div class="simple-keyboard"></div>
</template>
```

Snippet 9. Set up keyboard with the layout

3.6 Player UI

To create a user interface for the player, several elements were needed, which are:

+ Player position element, which is needed for showing the current position of the video.

+ Player time element, which is needed for showing the total duration of the video.

+ Progress bar element, which is needed for displaying the status of the video timestamp.

+ Pause/play, rewind and forward button elements, which are needed for controlling the video player (play/pause the video, moving/skipping to a new position).

+ Subtitle text element, which is needed for displaying subtitle options for example depending on the subtitle language or showing/hidden subtitle text.

+ Audio text element, which is needed for displaying audio selection if the audio has more than one audio source.

+ Loading icon, which is needed for displaying when the video is loading.

The implementation for player user interface is shown in Snippet 10.

```
<div id="player">
  <div id="playposition"></div>
  <div id="playtime"></div>
  <div id="progress_currentTime" style="left: 130px"></div>
  <div id="progressbarbg"></div>
  <div id="progressSeekable" style="transition03all"> </div
  ><div id="progressbar" style="transition03all"></div>
  <div id="prew"></div>
  <div id="ppauseplay" class="pause">
    <div class="vcrbtn"></div>
    <span id="pauseplay"></span>
  </div>
  <div id="pff"></div>
  <div id="subtitleButton"><div
id="subtitleButtonText">Subtitles</div></div>
  <div id="audioButton"><div id="audioButtonText">Audio</div></div>
</div>
```

Snippet 10. Creating the player UI

3.7 Key Events for the Application

The key names and key codes had to be defined before adding to the application. For example, VK_RED is the name and 82 is the code of the red button on the remote control, as shown in Snippet 11.

```
const defaults = {
  VK_RED: 82,
  VK_GREEN: 71,
  VK_YELLOW: 89,
  VK_BLUE: 66,
  VK_LEFT: 37,
  VK_UP: 38,
  VK_RIGHT: 39,
  VK_DOWN: 40,
  VK_ENTER: 13,
  VK_0: 48,
  VK_1: 49,
  VK_2: 50,
  VK_3: 51,
  VK_4: 52,
  VK_5: 53,
  VK_6: 54,
  VK_7: 55,
  VK_8: 56,
  VK_9: 57,
  VK_PLAY: 415,
  VK_PAUSE: 19,
  VK_PLAY_PAUSE: 463,
  VK_STOP: 413,
  VK_FASTFWD: 417,
  VK_REWIND: 412,
  VK_HOME: 771,
  VK_END: 35,
  VK_BACK: 220,
  VK_BACK_SPACE: 8,
  VK_TELETEXT: 459,
}
```

Snippet 11. Determine the keycode for the application

After defining the keycodes for the application, they needed to be exported to the code. Exporting all of the key names defined on the object is shown in Snippet 12.

```
export const VK_LEFT = 'VK_LEFT'
export const VK_RIGHT = 'VK_RIGHT'
export const VK_UP = 'VK_UP'
export const VK_DOWN = 'VK_DOWN'
export const VK_ENTER = 'VK_ENTER'
export const VK_RED = 'VK_RED'
export const VK_BLUE = 'VK_BLUE'
export const VK_YELLOW = 'VK_YELLOW'
export const VK_GREEN = 'VK_GREEN'
export const VK_0 = 'VK_0'
export const VK_1 = 'VK_1'
export const VK_2 = 'VK_2'
export const VK_3 = 'VK_3'
export const VK_4 = 'VK_4'
export const VK_5 = 'VK_5'
export const VK_6 = 'VK_6'
export const VK_7 = 'VK_7'
export const VK_8 = 'VK_8'
export const VK_9 = 'VK_9'
export const VK_PLAY = 'VK_PLAY'
export const VK_PAUSE = 'VK_PAUSE'
export const VK_PLAY_PAUSE = 'VK_PLAY_PAUSE'
export const VK_STOP = 'VK_STOP'
export const VK_FAST_FWD = 'VK_FAST_FWD'
export const VK_REWIND = 'VK_REWIND'
export const VK_HOME = 'VK_HOME'
export const VK_END = 'VK_END'
export const VK_BACK = 'VK_BACK'
export const VK_BACK_SPACE = 'VK_BACK_SPACE'
export const VK_TELETEXT = 'VK_TELETEXT'
export const VK_GUIDE = 'VK_GUIDE'
export const VK_INFO = 'VK_INFO'
```

Snippet 12. Export all of the key names

To be able to react to user events, a function that receives the keycode and key name as parameters needed to be implemented. The function returns true if the keycode belongs to the key name (463 is a keycode of VK_PLAY_PAUSE) and vice versa. The implemented function is shown in Snippet 13.

```
export function isKey(code, key) {
  if (
    typeof window.KeyEvent !== 'undefined' &&
    typeof window.KeyEvent.VK_LEFT !== 'undefined'
  ) {
    if (key === VK_PLAY_PAUSE) {
      const playPause =
        typeof window.KeyEvent.VK_PLAY_PAUSE !== 'undefined'
          ? window.KeyEvent.VK_PLAY_PAUSE
          : 463
      return code === playPause
    } else if (window.KeyEvent[key]) {
      return code === window.KeyEvent[key]
    } else if (key === VK_TELETEXT || key === VK_BACK_SPACE) {
      return (
        (key === VK_TELETEXT && code === 459) ||
        (key === VK_BACK_SPACE && code === 8)
      )
    }
  } else {
    return defaults[key] && defaults[key] === code
  }

  return false
}
```

Snippet 13. Function for checking the keycode and key name

In order to improve the user experience, back navigation (return to the previous screen) and exit behavior (exit from the application) had to be be added. Snippet 14 shows how to add the key down event listener for the back and exit navigation.

```
document.addEventListener('keydown', this.onKeyDown, false)

onKeyDown(event) {
  event.preventDefault()
  if (!this.loading) {
    if (
      isKey(event.keyCode, VK_BACK) ||
      isKey(event.keyCode, VK_BACK_SPACE)
    ) {
      if (this.$route.name === 'home') {
        this.showExitDialog()
      } else {
        // Return to previous view
        this.$router.go(-1)
      }
      return true
    } else if (isKey(event.keyCode, VK_RED)) {
      if (
        this.$route.name === 'home' ||
        this.$route.name === 'menu' ||
        this.$route.name === 'search'
      ) {
        this.showExitDialog()
        return true
      }
    }
  } else {
    if (
      isKey(event.keyCode, VK_BACK) ||
      isKey(event.keyCode, VK_BACK_SPACE)
    ) {
      this.$router.go(-1)
    }
  }
}
```

Snippet 14. Key down event for the application

3.8 Event Listeners for the Player

In order to interact with the player, the event listeners have to be added. Common events for the player are error event, ended the event, playing event, pause event, time update event and seeking event.

- Error event occurs when the source could not be loaded.
- Ended event occurs when the audio/video was reached to the end.
- Playing event occurs when the audio/video is playing.
- Pause event occurs when the audio/video is pause by user or programmatically
- Time update event occurs when the audio/video position has changed.
- Seeking event occurs when the audio/video position moving, skipping, or seeking to the new position.

Adding event listeners is shown in Snippet 15.

```
this.video.addEventListener('error', this.onErrorHander, false)
this.video.addEventListener('ended', this.onEndedHandler, false)
this.video.addEventListener('abort', this.onAbortHandler, false)
this.video.addEventListener('loadstart', this.onLoadStartHandler,
false)
this.video.addEventListener('waiting', this.onWaitingHandler, false)
this.video.addEventListener('playing', this.onPlayingHandler, false)
this.video.addEventListener('pause', this.onPauseHandler, false)
this.video.addEventListener('emptied', this.onEmptiedHandler, false)
this.video.addEventListener('timeupdate', this.onTimeUpdateHandler,
false)
```

Snippet 15. Adding event listeners for the player

Before destroying or disposing of the video player component, the event listeners of the video player need to be removed to avoid the memory leaks. Removing event listeners is shown in Snippet 16.

```
this.video.removeEventListener('error', this.onErrorHander, false)
this.video.removeEventListener('ended', this.onEndedHandler, false)
this.video.removeEventListener('abort', this.onAbortHandler, false)
this.video.removeEventListener('loadstart', this.onLoadStartHandler,
false)
this.video.removeEventListener('waiting', this.onWaitingHandler,
false)
this.video.removeEventListener('playing', this.onPlayingHandler,
false)
this.video.removeEventListener('pause', this.onPauseHandler, false)
this.video.removeEventListener('emptied', this.onEmptiedHandler,
false)
this.video.removeEventListener('timeupdate',
this.onTimeUpdateHandler ,false)
```

Snippet 16. Removing player's event listeners

3.9 Creating a Player

3.9.1 OIPF Object Tag

In some HbbTV environments, the video does not work with the video tag and has to use an object tag instead. An object tag is created with the mime type and media URL. Table 2 shows video extensions and their mime types. The code to create an object tag is shown in Snippet 17.

Table 2. Video extension and its mime type

Video extension	Mime type
mpd	application/dash+xml
m3u8	application/vnd.apple.mpegurl
mp3	audio/mpeg
mp4	video/mp4

```
<object id="video" type="mime type" data="URL"></object>
```

Snippet 17. Object tag for the HbbTV player

3.9.2 Shaka

In order to install Shaka Player for the project, running this command **yarn add shaka-player.** Shaka Player required to be imported to the top of the file after the library was already in the project.

Snippet 18 shows how to create a player using Shaka player based on the video element tag. Shaka Player has a load method that loads the media source **this.player.load(url)** after the player has been initialized.

```
this.video = document.getElementById("video")
this.player = new shaka.Player(this.video)
```

Snippet 18. Creating a player using Shaka player

3.9.3 Dash.js

In order to use Dash.js for the project, Dash.js needed to be installed by using the command **yarn add dashjs**

First, as shown in Snippet 19, dashjs was imported into the file, then the play-er was created with dashjs and the player initialized with three arguments: video element, media source, and auto play that are shown in Snippet 20.

```
const dashjs = require("dashjs")
```

Snippet 19. Creating a player using Shaka player

```
this.player = dashjs.MediaPlayer().create()
this.player.initialize(this.video, url, true)
```

Snippet 20. Integrating dashjs to player

3.10 DRM Configuration

If the video has license management to access, the application needs to provide the player with two things: the URL(s) and its license server(s).

3.10.1 OIPF object Tag

In order to send DRM to the DRM system, DRM OIPF object tag with the type is application/oipfDrmAgent needs to be created, as shown in Snippet 21.

```
<object id="oipfDrm" type="application/oipfDrmAgent" width="0"
height="0"> </object>
```

Snippet 21. Creating an object tag

When sending the DRM message with the function **sendDRMMessage**, it takes three arguments: msgType msg and DRMSystemID.

- **msgType**: The DRM system defines a globally unique message type (for example **application/vnd.ms-playready.initiator+xml** is the MIME type of PlayReady Action Token).
- **msg**: it is to be sent to the underlying DRM agent, formatted according to the message type specified by the msgType property.
- **DRMSystemID**: is defined by DRMSystemID. For example, for PlayReady, the DRMSystemID value is urn:dvb:casystemid:19219.

There are two DRM systems available, PlayReady DRM system and Marlin DRM system. XML License acquisition for PlayReady system is shown in Snippet 22 and for marlin system is shown in Snippet 23.

```
var msgType = "application/vnd.ms-playready.initiator+xml";
var xmlLicenceAcquisition =
'<?xml version="1.0" encoding="utf-8"?>' +
'<PlayReadyInitiator
xmlns="http://schemas.microsoft.com/DRM/2007/03/protocols/">' +
  '<LicenseServerUriOverride>' +
  '<LA_URL>' +
    this.drm.la_url +
  '</LA_URL>' +
  '</LicenseServerUriOverride>' +
'</PlayReadyInitiator>';
var DRMSysID = "urn:dvb:casystemid:19219";
```

Snippet 22. XML License Acquisition for PlayReady system

```
var msgType = "application/vnd.marlin.drm.actiontoken+xml";
var xmlLicenceAcquisition =
'<?xml version="1.0" encoding="utf-8"?>' +
'<Marlin xmlns="http://marlin-
drm.com/epub"><Version>1.1</Version><RightsURL><RightsIssuer><URL>'+
this.drm.la_url +'</URL></RightsIssuer></RightsURL></Marlin>';
var DRMSysID = "urn:dvb:casystemid:19188";
```

Snippet 23. XML License Acquisition for marlin system

After defining the message, message type and the DRM system ID, the sendDRMMessage has to be called to send DRM message to DRM service, as shown in Snippet 24. The result message will be returned to the client after it has been sent, and the result message as well as the description may be found in Table 3.

```
this.oipfDrm = document.getElementById('oipfDrm')
this.oipfDrm.sendDRMMessage(msgType, xmlLicenceAcquisition, DRMSysID);
```

Snippet 24. Running sendDRMMessage function with arguments

Table 3. Result message and its description after sending the DRM message defined by DRM system

Result message	Description
0	Successful
1	Unknown error
2	Cannot process request
3	Unknown MIME type
4	User consent needed
5	Unknown DRM system
6	Wrong format

3.10.2 Shaka

Snippet 25 shows how to configure the DRM for Shaka. Shaka provides the configure method to set up the multiple DRM playback. It requires two important things, the first thing is the DRM server (for example for PlayReady is com.microsoft.playready) and the second one is the license server.

```
this.player.configure({
  drm: {
    servers: {
      'com.widevine.alpha': 'the license server url for widevine',
      'com.microsoft.playready': 'the license server url for playready'
    }
  }
})
```

Snippet 25. DRM Configuration for Shaka player

3.10.3 Dashjs

Snippet 26 demonstrates how to set up the DRM for Dashjs. The setProtectionData method in Dashjs is used to set up multiple DRM playback. It receives the DRM

server and the server url, and the licensing token is optional, much like when setting up DRM for Shaka player.

```
this.player.setProtectionData({
  'com.widevine.alpha': {
    serverURL: 'the server url for widevine',
    httpRequestHeaders: {
      'X-AxDRM-Message': 'license token',
    },
  },
  'com.microsoft.playready': {
    serverURL: 'the server url for playready',
    httpRequestHeaders: {
      'X-AxDRM-Message': 'license token',
    },
  },
})
```

Snippet 26. DRM Configuration for Dashjs player

3.11 Creating and Adding Subtitles to the Video

The format of HTML5 video is WebVTT with .vtt extension. Snippet 27 shows how to create a subtitle file named example.vtt. For example “This is an example subtitle from 01:27 to 01:30” will be the subtitle text of the video from 1 minute 27 seconds and 600 milliseconds to 1 minute 30 seconds and 160 milliseconds.

```
WEBVTT

1
00:01:27.600 --> 00:01:30.160
This is an example subtitle from
01:27 to 01:30

2
00:01:31.000 --> 00:01:36.000
This is an example subtitle from
01:31 to 01:36

3
00:01:37.600 --> 00:01:42.000
This is an example subtitle from
01:37 to 01:42

4
00:01:42.400 --> 00:01:47.480
This is an example subtitle from
01:42 to 01:47
```

Snippet 27. Subtitle file with the WebVTT format

In order to create and add the track to the video element tag, the subtitle source and source language are required as well as the kind of the track has to be “subtitles”. The track below has a subtitle source as an example.vtt and subtitle language with English. One video can have multiple subtitle tracks.

```
<track src="example.vtt" label="English" kind="subtitles" srclang="en" >
```

If the player running on the object text does not support track tag, the developer needs to create a subtitle element and the function that reads the subtitle file and passes the text on the created element.

```
<div id="subcontainer"> </div>
```

```
convertSubtitleFileToArray(url, index) {
  const self = this
  axios.get(url).then((response) => {
    self.arraySubtitles[index] = response.data
      .split('\n\r\n')
      .map(function(item) {
        var parts = item.split('\r\n')
        if (item && parts && parts[1]) {
          const time = parts[1].split(' --> ')
          return {
            number: parts[0],
            starttime: self.convertFormatTimeToSeconds(
              time[0].split('.')[0],
              time[0].split('.')[1]
            ),
            endtime: self.convertFormatTimeToSeconds(
              time[1].split('.')[0],
              time[1].split('.')[1]
            ),
            text: parts.slice(2).join(' '),
          }
        }
      })
      .splice(1, response.data.length - 1)
  })
}

convertFormatTimeToSeconds(time, ms) {
  const [hh = '0', mm = '0', ss = '0'] = (time || '0:0:0').split(':')
  const hour = parseInt(hh, 10) || 0
  const minute = parseInt(mm, 10) || 0
  const second = parseInt(ss, 10) || 0
  return hour * 3600 + minute * 60 + second + ms / 1000
}
```

Snippet 28. Reading and converting the data from subtitle file to array

Reading and converting the subtitle file to the array function is done as shown in Snippet 28. There are two functions. The first function is converting the data from the subtitle file to the object with four keys (the number is subtitle index, starttime is the start time of subtitle showing in seconds, endtime is the end time of subtitle showing in seconds and text is the subtitle text). The second function is converting the format time on the subtitle file to seconds. For example: 00:01:27.600 to 87.6 seconds.

Snippet 29 describes the function that generates and transmits the subtitle text to the subtitle element. The generateSubText function loops through the subtitle array, looking for a subtitle object with a start time less than the current video time and an end time greater.

```
generateSubText() {
  const self = this
  clearInterval(this.showSubIntervalTime)
  if (this.subtitlesSelected > -1) {
    this.showSubIntervalTime = setInterval(() => {
      const subItem = self.arraySubtitles[self.subtitlesSelected].filter(
        (item) =>
          item &&
          item.starttime &&
          item.endtime &&
          item.starttime - 1 <= self.getPlayPosition() &&
          item.endtime >= self.getPlayPosition()
      )
      if (
        !self.arraySubtitles ||
        self.arraySubtitles.length === 0 ||
        self.arraySubtitles[self.subtitlesSelected].length === 0
      ) {
        clearInterval(self.showSubIntervalTime)
      }
      let currentSub = ''
      if (subItem && subItem.length > 0) {
        currentSub = subItem[0].text
      } else {
        currentSub = ''
      }
      document.getElementById(
        'subcontainer'
      ).innerHTML = `<span>${currentSub}</span>`
    }, 100)
  } else {
    document.getElementById('subcontainer').innerHTML = ''
    clearInterval(this.showSubIntervalTime)
  }
}
```

Snippet 29. Generating subtitle text and showing it on the screen

3.12 Implementing Thumbnail Images on the Player

Snippet 30 shows the function that displays the thumbnail image on the progress bar of the player. The function receives two parameters, the first parameter is the video duration and the second one is the current position of the player. The thumbnail image height and width have to be defined to ensure the calculation of the position of the thumbnail image is in the right place on the progress bar. The example of the original thumbnail image is shown in Figure 4. The original thumbnail includes multiple images and it should be cut to the single one based on the duration and position of the video like the image shows in Figure 5.

```
displayThumbnailImage(duration, position) {
  const imageHeight = 68;
  const imageWidth = 120;
  if (this.player && duration && this.thumbnailImageExist) {
    const progressBar = document.getElementById('progressbarbg')
    const thumbnail = document.getElementById('thumbnail')
    thumbnail.style.display = 'block'

    // Calculate 1 thumbnail takes how many pixel
    const thumbnailPixel =
      progressBar.getBoundingClientRect().width / (duration / 30)

    const progressBarLeft = progressBar.getBoundingClientRect().left
    const currentProgressBarWidth =
      (progressBar.getBoundingClientRect().width / duration) *
position

    const row = Math.floor((currentProgressBarWidth / thumbnailPixel)
% 8)
    const column = Math.floor(currentProgressBarWidth / (8 *
thumbnailPixel))

    thumbnail.style.left =
      Math.floor(currentProgressBarWidth + progressBarLeft -
imageHeight - imageWidth * row) +
      'px'
    thumbnail.style.top = -60 - imageHeight * column + 'px'
    thumbnail.style.clip = `rect(${imageHeight * column}px,
${imageWidth * (row + 1)}px, ${imageHeight *
      (column + 1)}px, ${imageWidth * row}px)`
  }
}
```

Snippet 30. Displaying thumbnail image on the progress bar of the player

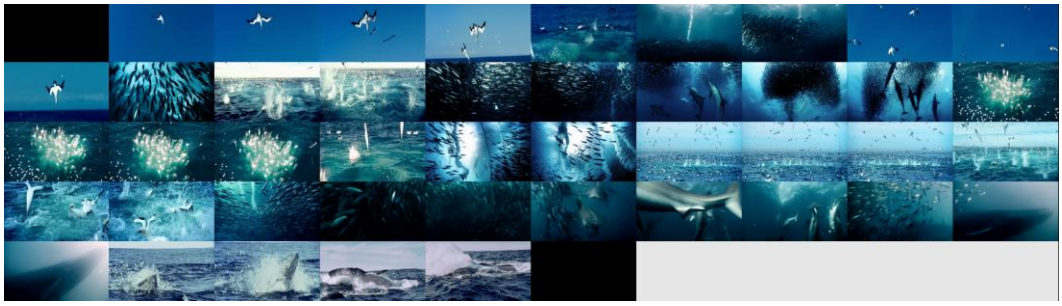

Figure 4. Original thumbnail image

Figure 5. Thumbnail image after getting cut off

4 CONCLUSIONS

The purpose of the thesis was to introduce television development in general and to design a smart television application for the streaming service industry in particular. The application should have a user-friendly user interface that includes a banner and program carousel, searching, a player screen and a focus on the user experience with the application's full button bind (back, exit and enter buttons) as well as navigation.

The most challenging part of the application was doing the player elements, components and functionalities as well as making sure the player worked correctly, smoothly without crashing and handling the errors.

Throughout the course of designing the application, I have learned a lot about developing TV apps for HbbTV and SmartTv. In addition, I gained more expertise with problem-solving and testing skills to ensure that the application works well on a variety of televisions such as LG, Samsung and others.

The work on this application continues to be developed and maintained by me at Sofia Digital Oy after graduation.

REFERENCES

/1/ What is the Best Smart TV Operating System? WhatIs. Accessed 18/02/2022 https://whatis.techtarget.com/definition/TV-operating-system#:~:text=A%20TV%20operating%20system%20(TV,interaction%20in%20a%20TV%20OS.

/2/ What is the Best Smart TV Operating System?. Source. Accessed 10/02/2022 https://www.sourceht.com/what-is-the-best-smart-tv-operating-system/

/3/ SDK vs. API: What's the Difference?. Ibm. Accessed 13/02/2022 https://www.ibm.com/cloud/blog/sdk-vs-api

/4/Smart-TV Development: The Basics. Medium. Accessed 04/02/2022 https://mlangendijk.medium.com/smart-tv-development-the-basics-5ec22a9ea2ad

/5/ JavaScript. Wikipedia. Accessed 04/02/2022 https://en.wikipedia.org/wiki/JavaScript

/6/ React. Wikipedia. Accessed 04/02/2022 https://en.wikipedia.org/wiki/React_(JavaScript_library)

/7/ What is Vue.js. Vue.js. Accessed 04/02/2022 https://v2.vuejs.org/v2/guide/

/8/ Svelte. Wikipedia. Accessed 04/02/2022 https://en.wikipedia.org/wiki/Svelte

/9/ Express.js. Wikipedia. Accessed 04/02/2022 https://en.wikipedia.org/wiki/Express.js

/10/ shaka-player. Github. Accessed 04/02/2022 https://github.com/google/shaka-player

/11/ dash.js Github. Accessed 04/02/2022 https://github.com/Dash-Industry-Forum/dash.js?

/12/ Digital rights management Wikipedia. Accessed 04/02/2022 https://en.wikipedia.org/wiki/Digital_rights_management

/13/ EME, CDM, AES, CENC and Keys – The Essential Building Blocks of DRM. Ottverse. Accessed 05/02/2022 https://ottverse.com/eme-cenc-cdm-aes-keys-drm-digital-rights-management/

/14/ HOW TO PRODUCE PROTECTED CONTENT: UNDERSTANDING DIGITAL RIGHTS MANAGEMENT. Brightcove. Accessed 05/02/2022

https://www.brightcove.com/en/resources/blog/dealing-drm-understanding-drm-and-how-produce-protected-content/

/15/ HTTP Live Streaming. Accessed 05/02/2022 https://en.wikipedia.org/wiki/HTTP_Live_Streaming

/16/ What is a smart TV? Best smart TVs for 2022. Accessed 16/03/2022 https://www.which.co.uk/reviews/televisions/article/what-is-smart-tv-aNeqk6F0RAAn

/17/ Dynamic Adaptive Streaming over HTTP. Accessed 05/02/2022 https://en.wikipedia.org/wiki/Dynamic_Adaptive_Streaming_over_HTTP

/18/ HbbTV Overview. Accessed 16/03/2022 https://www.hbbtv.org/overview/

/19/ HbbTV Overview. Accessed 16/03/2022 https://medium.com/@tvgames/hbbtv-vs-smart-tv-70baaca4448f

/20/ Online streaming overtakes traditional TV viewing among Finns under the age of 45. Accessed 16/03/2022 https://www.traficom.fi/en/news/online-streaming-overtakes-traditional-tv-viewing-among-finns-under-age-45

/21/ What is HTTP Live Streaming? | HLS streaming Accessed 16/03/2022 https://www.cloudflare.com/learning/video/what-is-http-live-streaming/

/22/ How To Develop A Smart Tv Mobile App: Features, Cost & Tech Stack. Accessed 16/03/2022 https://www.emizentech.com/blog/smart-tv-app-development.html

/23/ Media type Accessed 16/03/2022 https://en.wikipedia.org/wiki/Media_type