



VAASAN AMMATTIKORKEAKOULU
UNIVERSITY OF APPLIED SCIENCES

Duong Minh Ha

METHOD TO MIGRATE A REACTJS
APPLICATION FROM BROWSER TO SMART
TV

Technology and Communication
2022

ACKNOWLEDGEMENT

Glory to you, O Lord, who guide and bless me in my journey. My profound gratitude is extended to my great family, Uyen Dinh, and the VOTVapps team, especially to my manager Kari Karkkainen, and my colleagues including Viljami Korhonen, and Matthew Everitt for the support, encouragement, and opportunity for me to join the project.

In addition, I would like to express my sincere gratitude to Dr. Chao Gao, my project supervisor, who was in charge of me this project and provided guidance throughout the thesis-writing process.

Finally, I am very grateful and incredibly thankful to the Finnish Government and all the teaching and non-teaching staff of Vaasa University of Applied Sciences. I would not be here today without their expertise and efforts to establish an excellent learning environment.

Duong Ha

ABSTRACT

Author	Duong Minh Ha
Title	Method to Migrate a ReactJS Application from Browser to Smart TV
Year	2023
Language	English
Pages	38
Name of Supervisor	Gao, Chao

The demand for smart television is expected to increase rapidly in the upcoming years. For integrated internet connection, the smart television offers a wide range of features and services for users. However, developing an application in order to run on the smart television is still very restricted due to the variety of operating systems in different smart television brands.

The primary objective of this thesis is to migrate a browser-based ReactJS application to webOS and Tizen OS smart TVs. The application provides streaming services such as movie video on demand, TV shows, and live television. The scope of this thesis principally focuses on front-end development.

As the results of this thesis, ReactJS applications can be applied according to the development of the project mentioned in this thesis. Therefore, the thesis is a useful guidance for the development of a single framework application for different smart television operating systems.

Keywords	Smart TV, television technology, Tizen OS, webOS, and ReactJS
----------	---

CONTENTS

ABSTRACT	2
1 INTRODUCTION	7
1.1 Background	7
1.2 Objectives	8
1.3 Overall Structure of Thesis	8
2 SMART TELEVISION OVERVIEW	9
2.1 What Is a Smart Television?	9
2.2 Smart Television Manufacturer	9
2.3 Smart Television Operating Systems	10
3 APPLICATION DESCRIPTION	11
3.1 Application Overview	11
3.2 Use Case Diagram and User Interface Sketches	12
4 RELEVANT TECHNOLOGIES	15
4.1 ReactJS Framework	15
4.2 Webpack 5	16
4.3 BabelJS	17
4.4 Norigin Spatial Navigation	17
5 Implementation	20
5.1 Setting Up the Development Tool and Environment for Tizen Project	20
5.2 Generating a Tizen project	23
5.3 Setting Up a WebOS Project	24
5.4 Setting Up a Webpack	25
5.5 Improvements to ReactJS Project	27
5.6 Applying Navigation Management for ReactJS Project	29
5.7 Installing the Application Into Devices	31
5.8 Performance Improvement	35
6 CONCLUSION	37
REFERENCES	38

LIST OF FIGURES AND TABLES

Figure 1. The user case of VOTVapps project	12
Figure 2. Whatsnew screen	13
Figure 3. AllLivePrograms screen.....	14
Figure 4. Package Manager in Tizen Studio	21
Figure 5. Creating an emulator on Emulator Manager.....	21
Figure 6. Tizen Certificate Manager.....	22
Figure 7. Creating a Tizen certificate	23
Figure 8. Certificate container folder.....	23
Figure 9. Tizen project and config.xml file.....	24
Figure 10. WebOS configuration file.....	25
Figure 11. Entry and output.....	25
Figure 12. Use HtmlWebpackPlugin	26
Figure 13. .babelrc file	26
Figure 14. Define rules in Webpack.....	27
Figure 15. Output folder after bundling	27
Figure 16. Required files for Tizen project.....	28
Figure 17. The project folder for Tizen	29
Figure 18. Declare init method	30
Figure 19. Make a focusable Button component	30
Figure 20. Wrap components into a Focusable Container	31
Figure 21. Transform Button component to be focusble and handle click and focus events	31
Figure 22. Enable Developer Mode on Samsung TV.....	32
Figure 23. Create the TV device on Device Manager and permit to install applications.....	33
Figure 24. enable Developer Mode for LG TV.....	34
Figure 25. Create the TV device and set up SSH Key on webOS TV Devices.....	34
Table 1. Platform supported by Norigin Spatial Navigation library	18

LIST OF ABBREVIATIONS

A/V	Audio/visual
CAGR	Compound annual growth rate
CLI	Command-line interface
CPU	Central processing unit
CRA	Create-react-app
CSS	Cascading Style Sheet
DOM	Document Object Model
EPG	Electronic program guide
ES5	European Computer Manufacturers Association Script 5
ES6	European Computer Manufacturers Association Script 6
HDTV	High-definition television
HTML	Hyper Text Markup Language
IDE	Integrated development environment
JS	JavaScript
JSON	JavaScript Object Notation
MVC	Model-view-controller
OS	Operating System
OTT	Over-the-top
PPV	Pay-per-view
SDK	Software development kit
SSH	Secure Shell
SVOD	Subscription video on demand
TV	Television
TVOD	Transactional video on demand
UI	User Interface
UX	User Experience
VOD	Video on demand

1 INTRODUCTION

During the COVID-19 epidemic, several regions of the world were placed on lockdown, requiring residents to remain in their residences. The majority of people's time was devoted to digital media at home and the number of users surfing websites, using applications, playing games, and accessing other media has increased. Digital video content on the internet gradually is used by a sizable portion of the population and it is an increasing trend. Also contributing to the growth of smartphone, tablet, and other portable device users, particularly for gaming and social media access, is the younger generation. Digital content is now easier to obtain in a number of formats because of the development of the Internet and technology. Therefore, choosing the optimal digital media for a given function is no longer a challenge. This section will provide a concise overview of the smart TV market, as well as a summary of the thesis's objective.

1.1 Background

The huge price decrease over the past few years has probably played the biggest role in the TV's rebirth. Today, shoppers may spend \$500 or less on a very decent 55-inch 4K TV. Due to the fact that laptops and cellphones may easily cost \$1,000 or more, the TV is now the least expensive screen in the home. /1/

The market for smart TVs is expanding, which creates a sizable possibility for creating applications for them. In the report published by Grand View Research (2023), the size of the global smart TV market was estimated at USD 197.82 billion in 2022, and it is anticipated to increase at a Compound Annual Growth Rate (CAGR) of 11.4% from 2023 to 2030. The market for smart TVs has benefited from the rising demand for content on Over-The-Top (OTT) services. To meet the needs of every sort of viewer, a number of TV content producers are constantly releasing new material in a variety of genres. Several content producers are working together to establish exclusive partnerships with various OTT platforms. Leading streaming services such as Amazon Prime, Netflix, Disney Plus, and others have created tailored applications for smart Televisions to offer their streaming

services. Furthermore, innovative capabilities on smart TVs such as voice control, screen mirroring and sharing, and video calling, among others, contribute to their increasing demand by making the user experience more engaging. /2/

Smart TV application providers can use new technologies to optimize their applications to capitalize on the trends, opportunities, and challenges of the smart TV industry. Therefore, the thesis aims to investigate a new direction for the development of the application by making use of the popular and high-performance framework ReactJS .

1.2 Objectives

Generally, different TV manufacturers are using different operating systems. For example, Samsung mainly uses Tizen OS which is a Linux-based mobile operating system for its smart television product lines. Meanwhile, webOS also known as LG webOS was made primarily a smart TV operating system for LG televisions. The primary aim of this thesis is to convert a web-based ReactJS application to webOS and Tizen OS smart TVs application. The topic of this thesis is inspired by the development of a smart TV application by Viaccess-Orca. The application offers streaming services for live television, catchup programs, TV shows, and on-demand movies. It also allows parents to manage their children via kid profile which will filter appropriate programs for children and anonymous mode for reviewing all programs without login. Front-end development is the main emphasis of the thesis.

1.3 Overall Structure of Thesis

The fundamental organization of the thesis is explained as follows:

The motivation and goals of the thesis are discussed in Chapter 1. Chapter 2 gives a general introduction to smart televisions. Chapter 3 will cover the overview of the application. The relevant technologies and project implementation are discussed in Chapter 4 while the conclusion is presented in Chapter 5.

2 SMART TELEVISION OVERVIEW

In today's high-tech world, devices that have integrated advanced technology and connectivity features to enhance their capabilities beyond traditional devices usually come with the "smart" prefix. This chapter aims to provide readers with a comprehensive understanding of smart TVs in today's digital landscape.

2.1 What Is a Smart Television?

The outdated "dumb Televisions" could only show content through a cable, HDTV antenna, or other A/V sources. The linked world today demands a little bit more sophistication. The current TV is more like a smartphone or tablet than the tube TV of the past thanks to strong CPUs, internet connectivity, and user-friendly software. /3/

Smart TVs provide internet connection and application compatibility, much like smartphones and other smart home devices. Besides, it allows the control of a house full of connected devices, including the best Alexa-compatible and Google Home-compatible gadgets, as well as the ability to play games, access social media, and watch Netflix and Hulu films. /3/

2.2 Smart Television Manufacturer

With the push toward making every set "smart," every major TV manufacturer has moved away from creating dumb TVs. The top manufacturers of smart TVs are Hisense, LG, Panasonic, Philips, Samsung, Sharp, Sony, TCL, Toshiba, and Vizio. However, consumers should be aware of low-cost televisions that merely hazily indicate having a smart TV capability, even though the majority of these companies claim that their smart platforms are the most cutting-edge. These off-brand smart TV platforms usually provide restricted application choices, poor performance, and concerning security flaws.

2.3 Smart Television Operating Systems

Similar to the world of desktop operating systems, the many operating systems for smart TVs each have their own benefits and drawbacks. Others offer a greater selection of applications, others are more compatible with third-party devices, some have better user interfaces, and so on. /4/

Currently available operating systems for smart TVs include Roku TV, Google TV, Fire TV, webOS, and Tizen OS. They have their own unique features and capabilities. However, based on the smart TV market share and customer demand from VOTVapps which is the basis to carry out this thesis, the scope of this thesis is focused on two operating systems TizenOS and webOS.

Samsung uses TizenOS, a Linux-based operating system that was created by The Linux Foundation. In 2022, Samsung began granting smaller third-party TV brands licenses to use TizenOS, hence TizenOS may also be found on non-Samsung Televisions.

An internet-focused smart TV platform called webOS TV has been powering LG Smart TVs for more than ten years. It not only confirmed its consistent and reliable performance around the world, but it also showed that it has the capability and qualifications to dominate the market for Linux-based smart display platforms.

3 APPLICATION DESCRIPTION

This section provides an overview of VOTVapps web application, beginning with an introduction to its most important features and functionalities. In addition, it contains a use case diagram depicting the numerous interactions between the application and its users. The section depicts the sketch of the user interface. This introduction sets the foundation for a more in-depth examination of how navigation management is applied to all screen components in subsequent sections.

3.1 Application Overview

VOTVapps is a customizable application for TV Everywhere, and Video multi-screen services by Viaccess-Orca.

- It offers powerful TV analytics, business management, content management, personalization and discovery of content, content security, a multi-platform media player, and an online messaging platform.
- It supports a wide range of sizes, shapes, types, and operating systems, as well as web, Android, and iOS platforms, and it regularly certifies new hardware and software.
- The app is successful in multi-screen service monetization techniques. The adaptable solution offers a number of content business offerings, such as VOD (TVOD, SVOD, and Movie Packages) and Live (Channel Packages, PPV, Start Over, and Catch Up), among others.
- The app can be completely customized with White-Label Approach by using brandable elements in your brandings, such as a logo, a name, a color scheme, labels, and more
- The modular services provided by the Apps, such as Live, VOD, Social TV, Purchase, Subscription, and other revenue models, can be turned on and off in accordance with the service provider's offerings.
- The display of several languages in captions, metadata, and messages is supported through language localization.

3.2 Use Case Diagram and User Interface Sketches

Figure 1 demonstrates an example of the fundamental features and options available to users. A user can often manage various profiles with a parental PIN level. They can choose a language from the list and watch live videos, movies, or TV shows. They can also add their favorite content to a wish list, purchase video content, or search for videos by title.

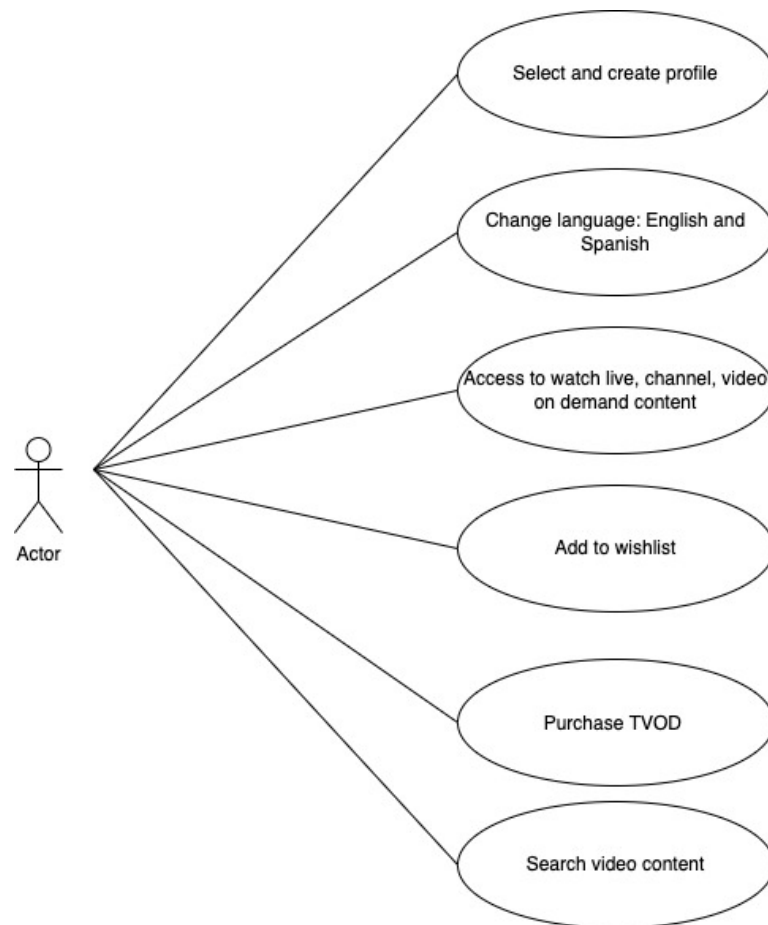


Figure 1. The user case of VOTVapps project

For each screen, the application has a different user interface. This section will only show the sketches of two screens, Whatsnew and AllLivePrograms, in order to clarify the operation of the navigation between components in the application.

After logging in and selecting a profile, the application will go to the Whatsnew screen. Figure 2 depicts all elements in the Whatsnew screen, one of the required

and primary screens in the application. The screen contains a banner, several program carousels, and a left menu.

The first icon of the left menu indicates the profile avatar. In vertical order, the following icons represent the Whatsnew screen, the AllLivePrograms screen, the VOD screen, the setting screen, the search screen, and the logout action.

To let users know where they are on the screen, an element being targeted will have a white outline and expand to 1.05 times its original size. Users can use the four-arrow keys on the TV remote to navigate the focus to other elements on the screen. For instance, if the current focus is the program card on a carousel and the user wants to move to the AllLiveProgram screen. They need to press the left arrow key of the TV remote. With every press, the focus will move left through each program card on the carousel, then after reaching the first program card on the row, it will change to the left menu and they need to press down to focus on the AllLivePrograms icon. Then, they need to press the enter key to trigger onClick event and the screen will change to AllLivePrograms screen.

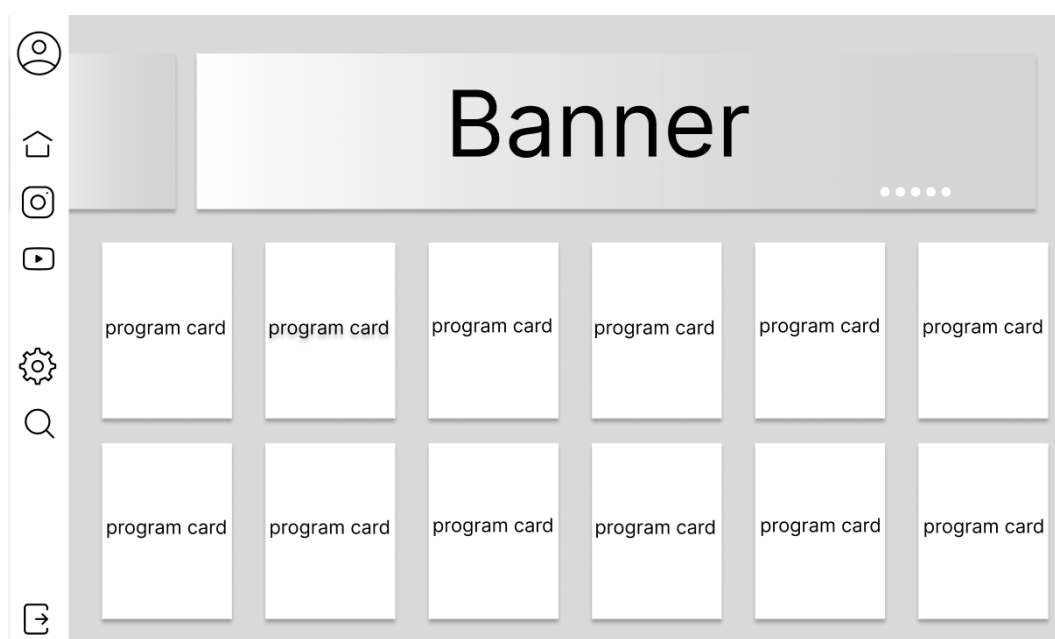


Figure 2. Whatsnew screen

AllLivePrograms screen includes a scrollable grid of all airing live program cards and a left menu as shown in Figure 3. To scroll down deeper cards, the user must

select the down arrow key on the TV remote, which will cause the focus to move through each card in a downward direction. After reaching the card at the screen edge, the gird will automatically scroll down to display more cards below.

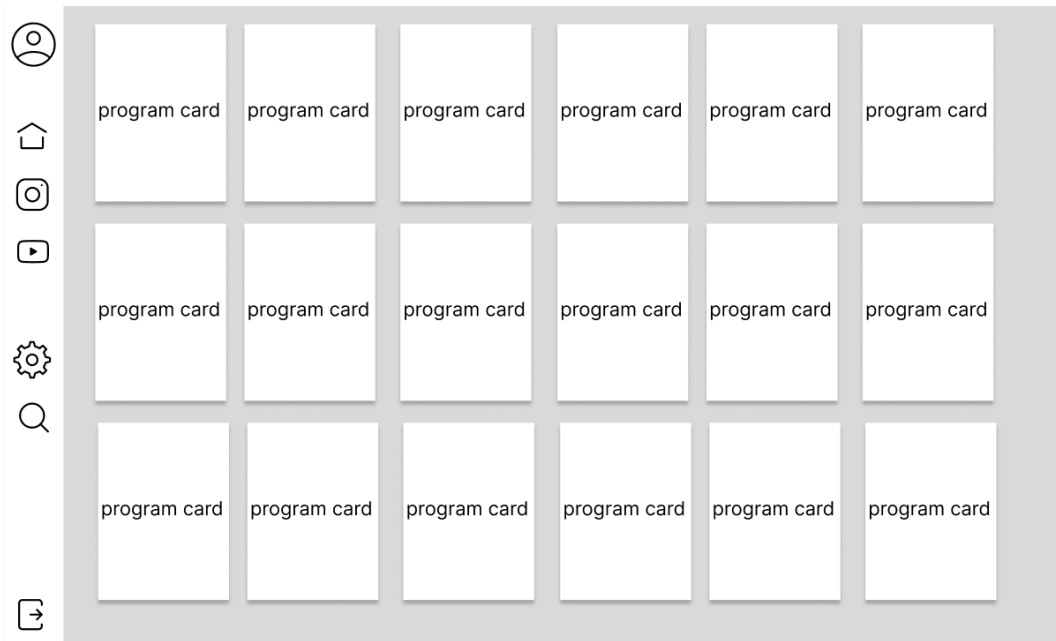


Figure 3. AllLivePrograms screen

4 RELEVANT TECHNOLOGIES

This chapter provides an overview of the relevant technologies used for the project. This project was developed using the ReactJS framework along with Webpack, BabelJS, and Norigin Spatial Navigation to convert from a web-based application to a smart TV application. ReactJS is the popular framework for developing web application. It is easy to add to current projects without having to start from scratch. Additionally, It also has a large and active developer community, which means there is extensive support, documentation, and resources available.

4.1 ReactJS Framework

ReactJS is the open-source framework and library for JavaScript by Facebook. It is used to quickly and efficiently construct interactive user interfaces and online Apps in comparison to utilizing pure JavaScript. /5/

Every component created by ReactJS can be reused as it is similar to individual Lego blocks. When combined, these pieces, which are individual components of a final interface, form the whole user interface of the application. /5/

In an application, the primary function of ReactJS is to manage the view layer of the application by providing the most efficient rendering execution, like the V in the Model-View-Controller (MVC) paradigm. Sophisticated user interfaces will be broken down into separated reusable components rather than considering the entire user interface as a single entity. The combination of JavaScript 's speed and efficiency with a more efficient method of manipulating the Document Object Model (DOM) will allow web pages to be rendered more quickly and highly dynamic. Furthermore, responsive online applications will be developed. /5/

ReactJS uses a component-based design that lets programmers make user interface (UI) parts that can be reused and do not depend on anything else. These components encapsulate their own logic and state, which makes the code modular and easy to update. ReactJS uses a Virtual DOM, which is a lightweight copy of the

real DOM, to update and display UI changes quickly. When the state or props of a component change, ReactJS figures out the difference between the old and new Virtual DOM and changes only the necessary parts. Therefore, this mechanism improves speed. This method encourages a declarative programming style, in which programmers describe how they want the UI to look, and ReactJS takes care of changing the UI quickly. ReactJS's architecture supports modularity, reusability, and fast rendering, which makes it a popular choice for building dynamic and responsive web applications. /5/

Before starting with ReactJS, NodeJS installation is necessary for local development. NodeJS can also be used in production, but it is not required. A static HTML/CSS/JS folder can be exported by many ReactJS frameworks. The documentation for setting up a ReactJS application can be found in their website <https://react.dev/>.

4.2 Webpack 5

A module bundler is called Webpack. Bundling can be handled by Webpack in addition to a different task runner. Nevertheless, due to community-developed Webpack plugins, the distinction between a bundler and a task runner has blurred. Although these tasks can be put off outside of Webpack, these tools are sometimes used to do things that are usually done outside of Webpack, like clean the build files or deploy the build to a server.

The majority of people are accustomed to building any ReactJS project with CRA (Create-React-App). It sets up every tool required to start a ReactJS project, including Webpack, by issuing a single command.

For ReactJS developers working on a modest ReactJS project, the way to automatically set up a ReactJS project along with Webpack by a single command is useful. Nevertheless, the project output after bundling contains a number of unadjustable defaults that prevent the application from functioning correctly on other platforms, as well as the inability to implement advanced Webpack features and plugins. For developing large-scale ReactJS projects, this technique might not

be the best option. It is best to choose a custom ReactJS Webpack configuration from start to obtain more control over the entire build process. We should comprehend Webpack ReactJS before delving further into using it with ReactJS.

Webpack is instructed on how to write the built files to disk by the output configuration parameters that are set. While there may be several entry points, there is only one output configuration listed. /6/

4.3 BabelJS

Edge JavaScript is converted by the JavaScript transpiler BabelJS into European Computer Manufacturers Association Script 5 standard for JavaScript (ECMAScript 5 or ES5) that may be used in any browser (even the old ones). It makes available all the classes, fat arrows, and multiline strings that the new ES6 specification brought to JavaScript as syntactical sugar. Using it to translate TypeScript into normal JavaScript that can execute in a browser is an optional additional usage for it.

In terms of this project, it is imperative to translate any new ES6 specification in ReactJS into standard ES5 JavaScript for some older models of smart TV. Otherwise, the software can crash or its user interface might malfunction. /7/

4.4 Norigin Spatial Navigation

Due to the growing number of Connected TVs and SmartTV devices on the market today, developing cross-platform web applications is increasingly necessary. The majority of them may be utilized to create applications for these devices because they are integrated with the interactive Web 2.0 and frontend JavaScript libraries such as ReactJS.

There are a couple of reasons that Norigin Media's spatial navigation library has been chosen to manage directional navigation for this project. Firstly, Norigin Spatial Navigation is designed to be compatible with many different web platforms and frameworks, including ReactJS. Secondly, it simplifies the development process by giving developers an API for handling spatial navigation that is easy to

use. Finally, it improves the accessibility of web applications by providing an alternative navigation method.

With the Norigin Spatial Navigation library, creating navigation logic for websites and applications that are accessed via the keyboard (browsers) or remote control (Smart TV or Connected TV) is made easier. All preparation consists of declaring the initial focus method in the root file and declaring hooks for focusable components. /8/

While using the directional keys to move, this spatial navigation library will automatically choose which elements to focus on next.

Theoretically, the Norigin Spatial Navigation library is designed to function on any web-based device, including browsers and smart TVs. The UI/UX is compatible with Samsung Tizen TVs, LG WebOS TVs, Hisense Vidaa TVs, and a variety of other Connected TVs as long as it was developed using the ReactJS Framework. On Apple TV and Android TV, it can also be used, but functionality will be constrained. This library is regularly updated in the table below and is actively used, and tested across a wide range of devices.

Table 1. Platform supported by Norigin Spatial Navigation library

Platform	Name
Web Browser	Chrome, Firefox
Smart TVs	Samsung Tizen, LG WebOS, Hisense
Other Connected TV devices	Browser Based settop boxes with Chromium, Ekioh or Webkit browsers
AndroidTV, AppleTV	Only React Native Apps, limited functionality

In the previous version, a component is converted into a navigable component by applying HOC (Higher-Order Component) to it. However, the HOCs approach and the most widely used library, recompose, which offered many practical HOCs, are

both regarded as deprecated. Additionally, it adds to "wrapper hell". The functionality has recently been migrated to ReactJS Hooks.

5 IMPLEMENTATION

This chapter focuses on the implementation of the project, covering various aspects of the development process. It starts with an introduction to Tizen Studio and how to build a Tizen project. The next subsection shows how to set up webOS and apply navigation management for the project. It also covers setting up Webpack so that modules can be bundled together and the application can be installed on devices quickly. Furthermore, the final section will collect the investigation into performance improvement. Collectively, these topics provide a comprehensive overview of the implementation process, ensuring a smooth and optimized development experience.

5.1 Setting Up the Development Tool and Environment for Tizen Project

Generally, to set up the development tool and environment, it is required to install Tizen Studio, Tizen Command-Line Interface (CLI), Tizen SDK, Tizen certificate, and Tizen emulator.

Tizen Studio is an integrated development environment (IDE) for the TizenOS. Tizen Studio supports many purposes of development, testing, and deployment of Tizen applications with a selection of features and tools. We must install Tizen Studio as our first task. The location to get it can be found at developer.tizen.org (the most recent version is advised). Additionally, we also need Tizen CLI in order to execute commands for packaging and generating a Tizen application or deploying the application to the emulator or the TV device. Installing SDK and extension SDK is required after installing Tizen Studio in order to develop TV platforms. To accomplish this, we must launch Tizen Studio's package management as shown in Figure 4 and install the following SDK:

- Tizen SDK tools

- TV Extensions-7.0 (depending on the latest version)

- Samsung Certificate Extension

- TV Extension Tools

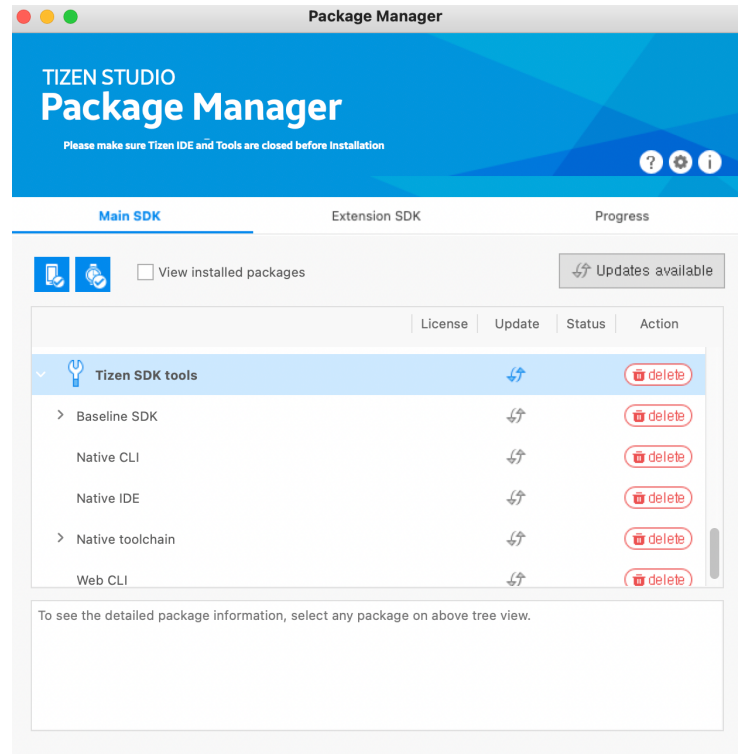


Figure 4. Package Manager in Tizen Studio

Figure 5 shows the Tizen Studio tool called "Emulator Manager". The Emulator Manager will help to make different versions of emulators so that the program can be tested in different environments. Developers can create, launch, edit, or delete by clicking on the buttons in the top menu. Creating an emulator that requires doing things like selecting the system picture, defining the device, finding the processor, and setting up the network.

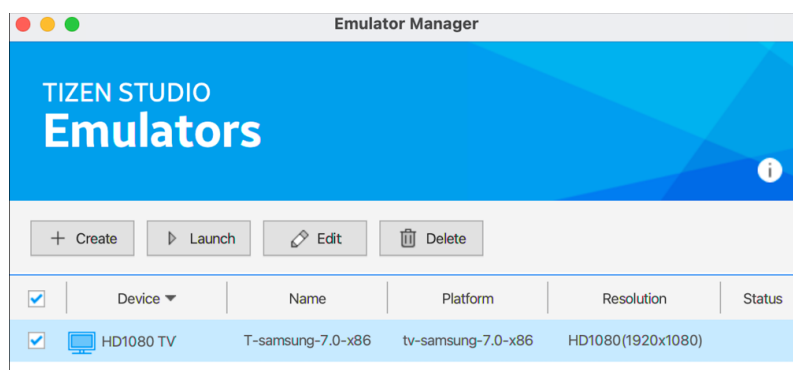


Figure 5. Creating an emulator on Emulator Manager

In terms of certification, in order to deploy the application to any emulator or TV device for both development and production purposes, we need to generate a

certificate for the application and grant permission to the emulator or TV device. The source of the application will be verified by the signature and ensure it has not been altered since it was published. A certificate profile is a combination of the signature certificates. To create a certificate, in Tizen Studio, we go to Tools and select Certificate Manager. The modal will open on the screen as shown in Figure 6.

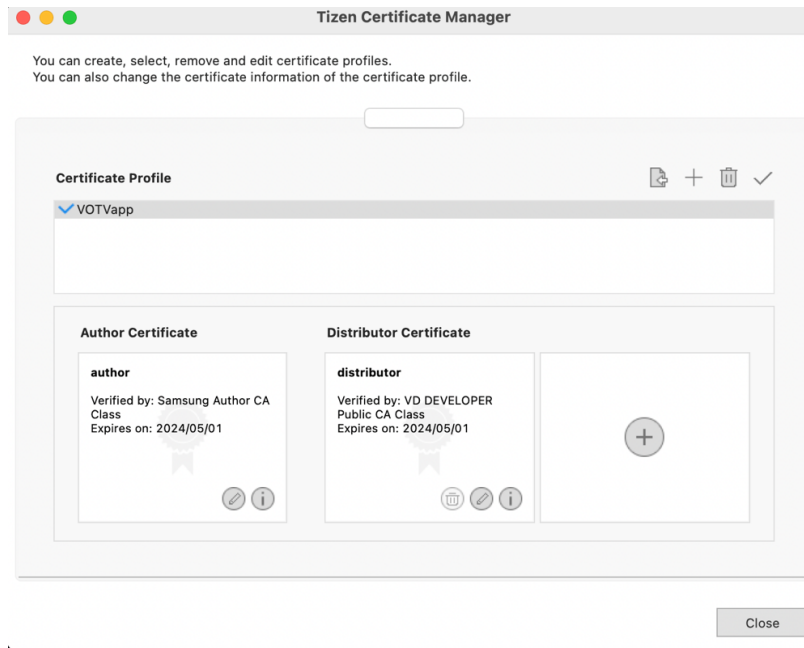


Figure 6. Tizen Certificate Manager

Clicking the plus icon in the menu's upper-right corner will reveal a second model. In Figure 7, if we wish to publish to the Galaxy store, we should select Samsung. Otherwise, Tizen certificate profiles only permit installation of the application on Tizen devices.

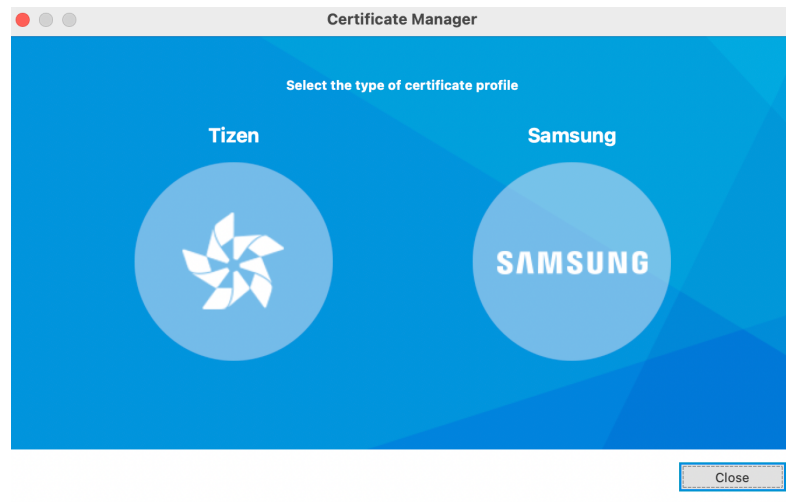


Figure 7. Creating a Tizen certificate

Then, we select the TV platform for TV application development and create the certificate profile and the author with name and password. We also need to create the distributor of the application. Finally, we can review all files created in SamsungCertificate folder as shown in Figure 8.



Figure 8. Certificate container folder

5.2 Generating a Tizen project

The structure of a ReactJS application contains numerous files and components, whereas the structure of the Tizen TV project is different. Therefore, in order for a ReactJS application to operate on a Tizen TV device, the structure of the ReactJS project must be modified, and the first step is to determine the structure of a Tizen project by generating a basic Tizen project on Tizen Studio.

In order to create a Tizen project, in Tizen Studio, we go to File, select New, and Tizen Project after that. The implementation process begins by selecting the type

of project, which sets the foundation for development. Following that, the application profile and version are carefully chosen, ensuring compatibility and access to desired features. The application type is then chosen based on the needs of the project, which decides the general structure and technologies. To expedite development, to speed up development, an application template is selected, providing a pre-configured starting point. Finally, project properties are defined, including the project name, package ID, and the location of the project. /9/

In Figure 9, Our project will then appear in the left Explorer. The overall structure contains an HTML file, a JavaScript file, a CSS file, an icon image, a configuration file, and some hidden setting files. We can declare privileges in the config.xml file so that the application running on the TV can be activated with more functions.

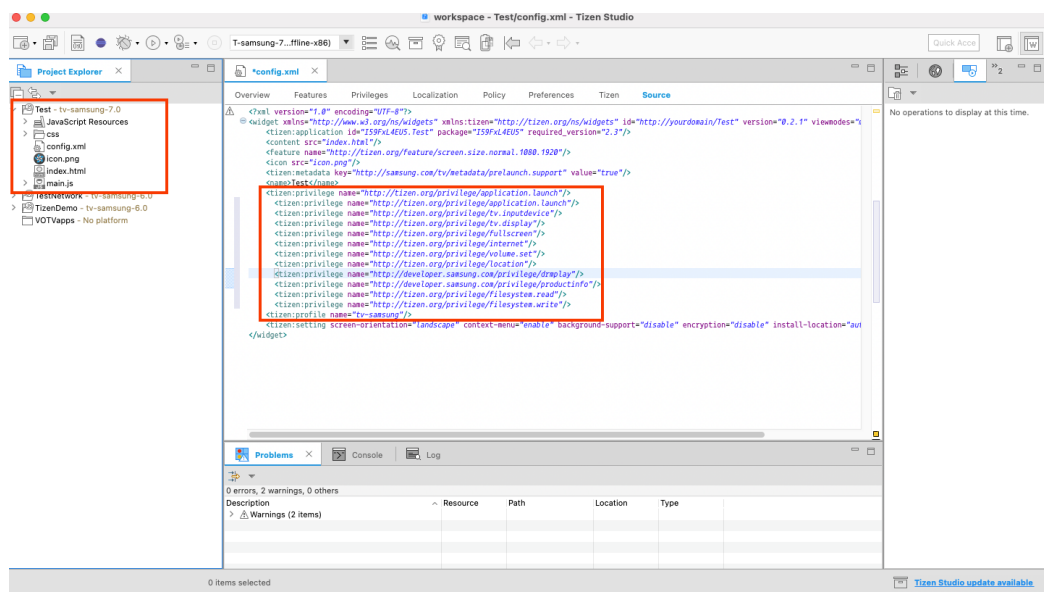
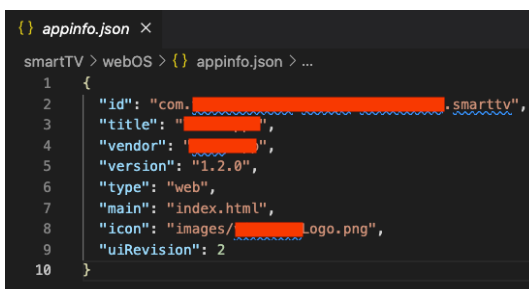


Figure 9. Tizen project and config.xml file

5.3 Setting Up a WebOS Project

Except for installing webOS CLI, we do not need to set up many environments for webOS applications. To port a ReactJS application operating in the browser to webOS TV devices, we must, similar to the Tizen project, modify the structure of the ReactJS project. To comprehend what to include in a ReactJS project, we must create a basic webOS and examine its components.

Essentially, webOS CLI is a Command-Line Interface utility used to develop, test, and deploy webOS applications. Then, we can create a new app by *ares-generate -t basic ./sampleApp* command (the option -t will specify the template, and the last part of the command is the location of the project). Eventually, it will generate a project with a similar structure to the Tizen project. The *appInfo.json* is the configuration file of the project. We need to define values for all properties in the file as shown in Figure 10.



```

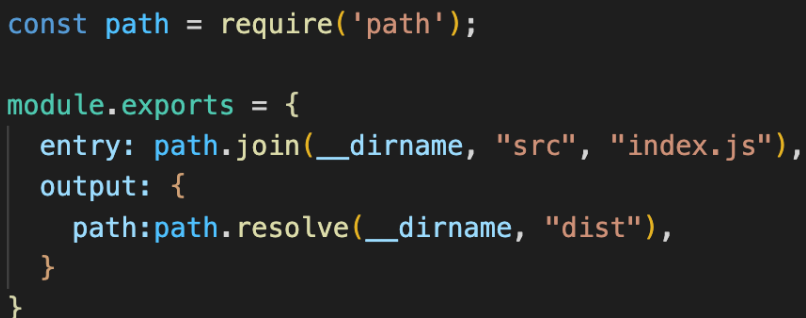
1  {
2    "id": "com. [REDACTED].smarttv",
3    "title": "[REDACTED]",
4    "vendor": "[REDACTED]",
5    "version": "1.2.0",
6    "type": "web",
7    "main": "index.html",
8    "icon": "images/[REDACTED]_Logo.png",
9    "uiRevision": 2
10 }

```

Figure 10. WebOS configuration file

5.4 Setting Up a Webpack

Firstly, we need to install Webpack into our project by `npm install Webpack Webpack-cli` and create *webpack.config.js* file. In Figure 11, we define where to start bundling the JavaScript file in *entry* property and to create the final bundled files.



```

const path = require('path');

module.exports = {
  entry: path.join(__dirname, "src", "index.js"),
  output: {
    path: path.resolve(__dirname, "dist"),
  }
}

```

Figure 11. Entry and output

In Figure 12, *HtmlWebpackPlugin* simplifies the process of generating HTML files that are automatically linked to the bundled JavaScript files produced by webpack.

```

plugins: [
  new HtmlWebpackPlugin({
    template: path.join(__dirname, "src", "index.html"),
  }),
],

```

Figure 12. Use HtmlWebpackPlugin

Then, we can setup babel by `npm install @babel/core babel-loader @babel/preset-env @babel/preset-react --save-dev` and create `.babelrc` file as shown in Figure 13.

- `@babel/core` is the core transpiler.
- Babel-loader is a Webpack loader for using babel transpiler.
- `@babel/preset-env` contains a collection of plugins that enable BabelJS to transpile modern JS code to operate in a variety of environments.
- `@babel/preset-react` contains a set of plugins to help the functionality and syntax of the React library.

```

{
  "presets": [
    ["@babel/preset-env", {
      "targets": {
        "uglify": true
      },
      "corejs": 3
    }],
    ["@babel/preset-react", {
      "runtime": "automatic"
    }]
  ]
}

```

Figure 13. `.babelrc` file

The rules in `webpack.config.js` tell webpack how to deal with certain types of files or modules when it bundles the application. They list the loaders or tools that transform or manipulate these files are used during the build process. Figure 14 shows an example of defining rules for JavaScript files, CSS files, and images.

```

module: {
  rules: [
    {
      test: /\.m?js$/,
      include: [sourcePath, path.resolve(__dirname, "node_modules")],
      use: {
        loader: "babel-loader",
        options: {
          presets: ['@babel/preset-env'],
        }
      }
    },
    {
      test: /\.css$/,
      use: [
        MiniCssExtractPlugin.loader,
        'css-loader',
      ],
    },
    {
      test: /\.(png|jpe?g|gif)$/i,
      include: [sourcePath, publicPath],
      type: 'asset/resource',
      generator: {
        filename: 'images/[name][ext]',
      }
    },
  ],
}

```

Figure 14. Define rules in Webpack

5.5 Improvements to ReactJS Project

After setting up Webpack and BabelJS, we can run `webpack --mode production` to start the bundling process of the application. It will generate a build folder as shown in Figure 15.

```

└─ build
  ├── assets
  ├── chunk-npm.
  ├── fonts
  ├── images
  ├── npm.
  ├── JS chunk-393.2c837734290cc1ccd574.js
  ├── JS chunk-738.5b6d5e5dc144ab752835.js
  ├── JS chunk-930.6f24f43aa7d1ef45ef57.js
  ├── JS chunk-6590.403202c9d4055d26e588.js
  ├── index.html
  ├── JS main.213fa06fba0de674611a.js
  ├── main.css
  ├── npm.null.css
  └── JS runtime.4443c79de9a5720c8e82.js

```

Figure 15. Output folder after bundling

When we compare the structure of this directory, we find similarities with the Tizen and webOS projects. However, we are still unable to generate a final package to install on TV based on this folder. To accomplish that, for Tizen, we must add

.settings, .project, .tproject, and config.xml as shown in Figure 16 (they are generated when creating a Tizen project in section 5.2) to this build folder, and for webOS, we must add appInfo.json file as mentioned in Figure 10. It is essential to remember that when generating the final webOS package for installation on webOS TV, all Tizen additional files must be removed, and vice versa for Tizen.

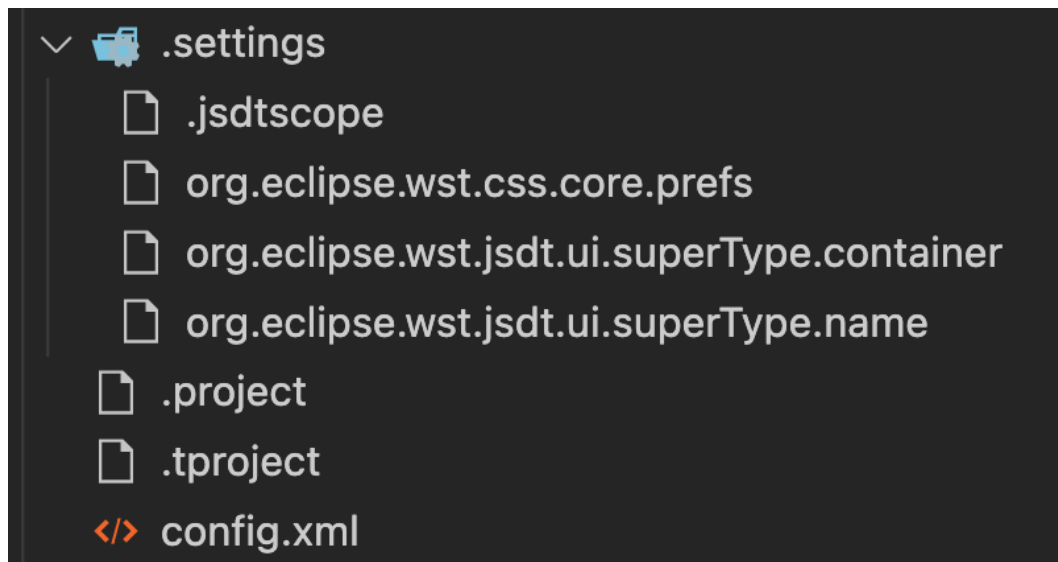


Figure 16. Required files for Tizen project

To add .settings, .tproject, .project, and config.xml files into the build folder, we use `cp -R .settings .tproject .project config.xml build`. When combining 2 commands for bundling and file copying, the project will generate a build folder containing all bundling files and Tizen configuration files as shown in Figure 17. We can carry out the same steps for webOS.

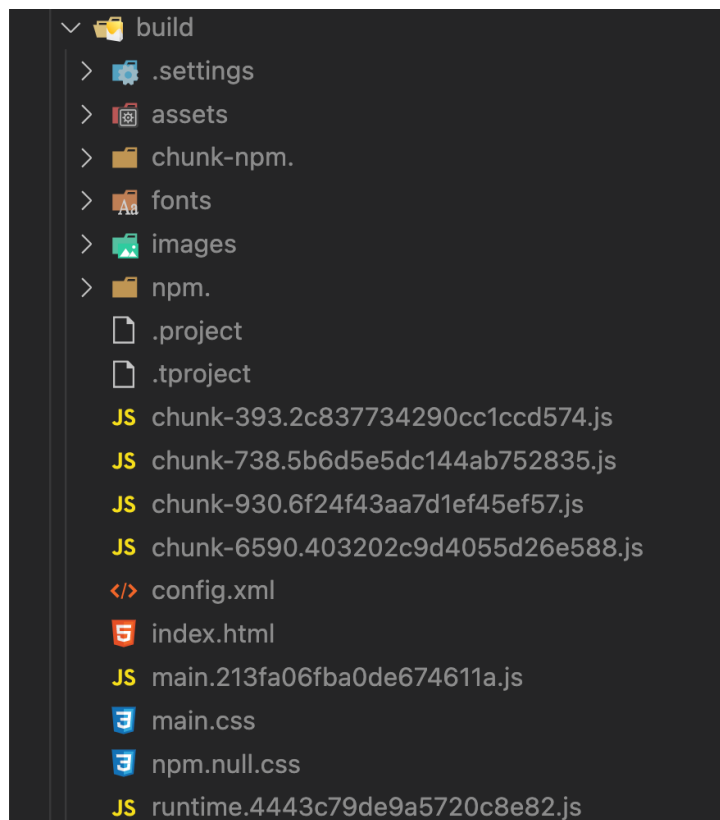


Figure 17. The project folder for Tizen

Basically, the application is now available to operate on Tizen and webOS TVs. However, the navigation manager is still required so that users can navigate the application and we will install it in the following section.

5.6 Applying Navigation Management for ReactJS Project

To install the library for the project, we can use the command `npm i @noriginmedia/norigin-spatial-navigation --save`. Figure 18 depicts the `init` method declaration that must be present in the root file before interacting with components in the project. We can add options such as `debug` (which enables console debugging) and `visualDebug` (which enables visual debugging) to the method.

```
// Called once somewhere in the root of the app

import { init } from '@noriginmedia/norigin-spatial-navigation';

init({
  // options
});
```

Figure 18. Declare init method

To convert a Leaf component (which does not have focusable children) into a focusable component, we can implement as Figure 19 with *ref* to bind the DOM element to the hook and *focused* to indicate the state of the component when it is focused.

```
import { useFocusable } from '@noriginmedia/norigin-spatial-navigation';

function Button() {
  const { ref, focused } = useFocusable();

  return (<div ref={ref} className={focused ? 'button-focused' : 'button'}>
    Press me
  </div>);
}
```

Figure 19. Make a focusable Button component

Components such as the Whatsnew and AllLivePrograms screens in Figures 3 – 4 have several child components. We will wrap all parent containers with `Focusable.Provider` to provide all children components with the container's `focusKey`. In Whatsnew case, `Focusable.Provider` will be declared and wrap the Whatsnew container, the side menu container, the banner container, and the program carousel container. For all children such as the program card, the side menu icon, and the banner block, those consider as final children so we can apply the spatial navigation as the instruction for Leaf components. By doing this way, focusable children components that are deep into the DOM tree still are able to identify their focusable parent. Instead of maintaining a focused state, `Focusable.Container` propagates focus to the relevant Child component.

```

import { useFocusable, FocusContext } from '@noriginmedia/norigin-spatial-navigation';
import ListItem from './ListItem';

function ContentList() {
  const { ref, focusKey } = useFocusable();

  return (<FocusContext.Provider value={focusKey}>
    <div ref={ref}>
      <ListItem />
      <ListItem />
      <ListItem />
    </div>
  </FocusContext.Provider>);
}

```

Figure 20. Wrap components into a Focusable Container

Every child component should have a unique focusKey in order to set the focus when the component renders or avoid confusion for the navigation manager when it handles navigation actions. We can use focusSelf to set the focus on the current component or setFocus to manually set the focus with an identified focusKey. Furthermore, we can manipulate click event or focus event by onEnterPress and onFocus respectively. To handle other events or actions, we can get more information at github.com/NoriginMedia/Norigin-Spatial-Navigation.

```

export const Button = (props) => {
  const {
    children, id, title, active, className, onClick, focusKeyParam, disabled, ...rest
  } = props;
  const [
    ref, focused, setFocus, getCurrentFocusKey,
  ] = useFocusable({
    focusKey: focusKeyParam,
    onEnterPress: () => onClick(),
    onFocus: () => ref.current?.scrollIntoView({ behavior: 'smooth', block: 'center' }),
  });
  useRestoreFocus(setFocus, getCurrentFocusKey);
  const classActive = active || styles.active;
  const style = className || styles.button;
  const buttonStyle = useMemo(() => focused ? `${style} ${classActive}` : style, [focused]);

  return (
    <button ref={ref} type="button" id={id} className={buttonStyle} disabled={disabled} {...rest} title={title || ''} onClick={() => onClick()}>
      {children}
      <div className={styles.hoverfix}>
        {children}
      </div>
    </button>
  );
};

```

Figure 21. Transform Button component to be focusble and handle click and focus events

5.7 Installing the Application Into Devices

Before the application can install in Tizen TV or WebOS TV devices, we must enable Developer Mode on the TV.

For Tizen TV, we need to do the following:

1. On the TV, open the “Smart Hub” and select the “Apps” panel
2. Use the remote control or the on-screen number keypad and enter “12345”
3. Turn “Developer mode” on and enter the IP address of the computer that you want to connect to the TV as shown in Figure 22

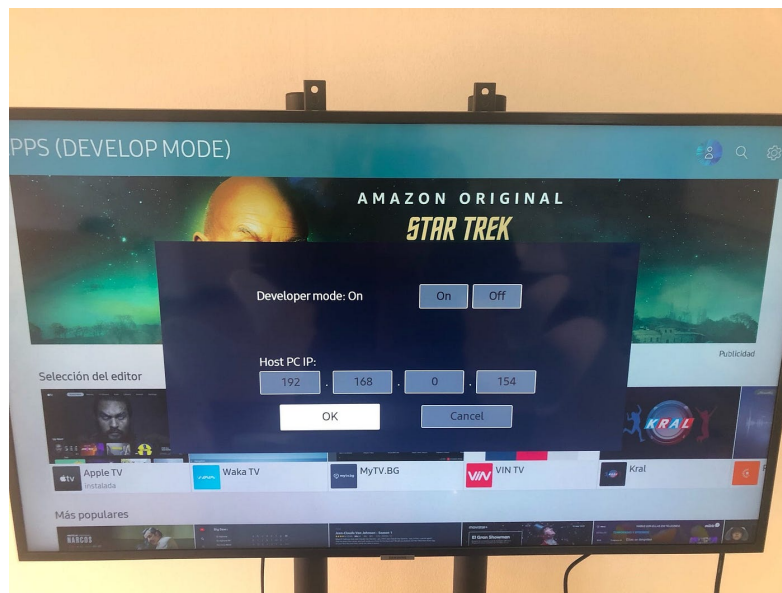


Figure 22. Enable Developer Mode on Samsung TV

4. Then, click “OK” and reboot the TV
5. Go to Device Manager in Tizen Studio and add a device
6. Enter the name, IP address, and Port of the TV
7. Check the connection of the TV on
8. Right-click the TV created and select “Permit to install applications” as shown in Figure 23

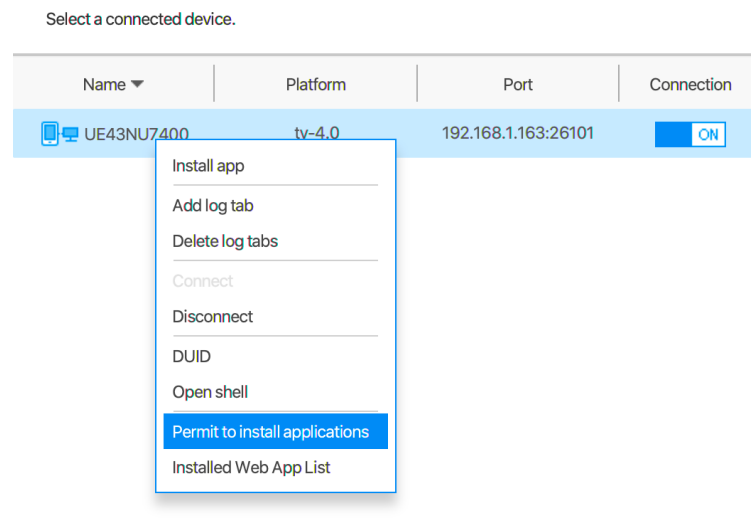


Figure 23. Create the TV device on Device Manager and permit to install applications

For webOS TV, we need to do the following:

1. Install the Developer Mode app
2. Login to the app by email and password of the LG Developer site
3. Switch the Dev Mode Status and Key Server to “On” as shown in Figure 24
4. Reboot the TV
5. Go to Visual Studio Code and download webOS TV extension
6. Select webOS TV Devices in the left menu
7. Click add device and enter device name, IP address, port, and username
8. Right-click on the device created and select “Set Up SSH Key” as shown in Figure 25
9. Enter Passphrase as displayed on the Developer Mode app

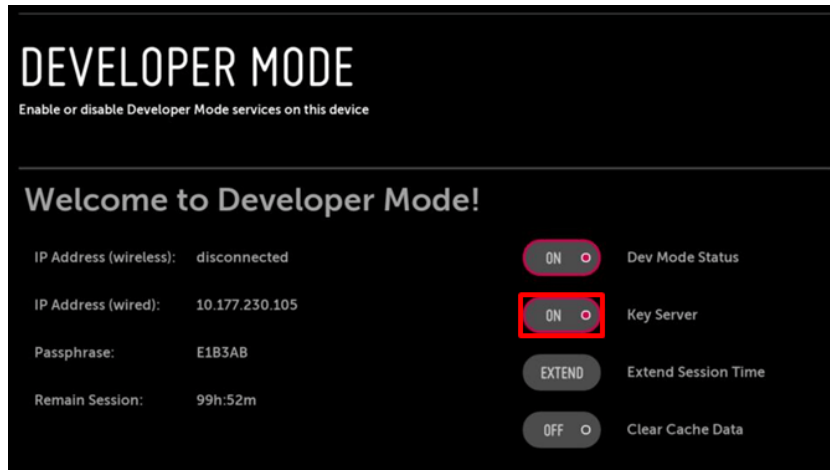


Figure 24. enable Developer Mode for LG TV

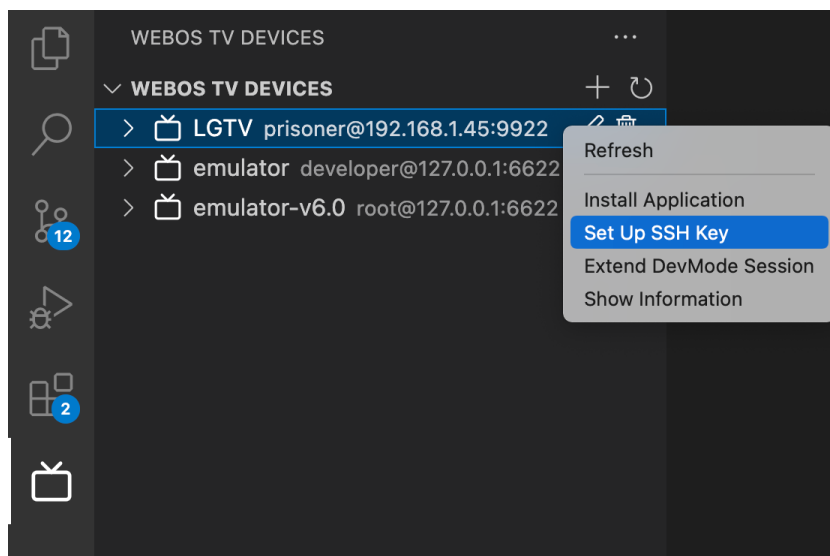


Figure 25. Create the TV device and set up SSH Key on webOS TV Devices

After completing these steps, we run `webpack --mode production` to start bundling the project.

To generate the package to install on the Tizen TV, we run the following commands:

1. `cp -R .settings .tproject .project config.xml build:` to copy `.settings`, `.tproject`, `.project`, and `config.xml` files into the build folder
2. `tizen build-web -- build:` To build the Tizen Web project with build folder as an input folder

3. `tizen package -t wgt -s package_file_path -- package_file_output:` To generate `.wgt` file in order to install on the TV
4. `tizen install -n package_file_name.wgt -- package_file_path -t target_name:`
To install the package on target

To generate the package to install on the webOS TV, we run the following commands:

1. `cp appinfo.json build:` to copy `appinfo.json` file into the build folder
2. `ares-package -o ./build ./build:` to generate `.ipk` file into the build folder in order to install on the TV

To install the application on the device for webOS, after generating the package file, we can right-click on the file on Visual Studio Code, select “webOS TV: Install Application” or “webOS TV: Run Application”, and choose the device that we created before.

5.8 Performance Improvement

In fact, in order to improve performance can include a variety of strategies can be included. After several investigations, here are a few implementations that are used for the application:

1. **Application Profiling:** In order to find areas for application performance optimization and improvement, a process of analysis and measurement is used. In order to increase overall performance, performance bottlenecks and other problems that may have a negative influence on the user experience are to be found via application profiling.
2. **welldone-software/why-did-you-render:** `wdyr` (why-did-you-render) is the library that helps to find reasons for component re-rendering. Based on the result, we can use `React.memo()`, `useMemo()`, or `useCallback()` to optimize application rendering by preventing unnecessary re-renders.
3. **Image Optimization:** For some huge images like 2300x2300px or more that are not displayed as full resolution on the screen, we can reduce

downloaded image sizes by using width? and height? parameters when fetching the image.

4. TerserWebpackPlugin: TerserWebpackPlugin is a Webpack plugin that is used to minify and compress JavaScript code. JavaScript files can be made smaller and downloaded more quickly thanks to this widely used web development tool. The plugin is built on the TerserJS library, a powerful code optimization tool.

6 CONCLUSION

The primary objective of this thesis was to convert a web-based ReactJS application to webOS and Tizen OS smart televisions. The application offers streaming services for live television, TV shows, and on-demand movies. The front-end development is the main emphasis of the scope.

This report primarily focuses on outlining the setup of environments for developing smart TV applications as well as the operation of relevant tools and libraries for a ReactJS project.

The VOTVapps is a large-scale project for multiple platforms that requires significant effort and time to develop. The report includes findings on new approaches to smart TV application development and performance improvement principles learned from project implementation. Making the application compatible with older TV Operating Systems, such as webOS 4.0, was the most difficult aspect. To enable new features, the project is still being developed and maintained by the author and VOTVapps team.

REFERENCES

/1/ Wolk, Alan. New Generation of Smart TVs Is Changing the Way We Watch TV. Accessed 10.04.2023. <https://www.forbes.com/sites/alanwolk/2021/09/15/the-tv-dongle-is-about-to-go-the-way-of-the-8-track/?sh=390a04a07c5a>

/2/ Grand View Research. TV Market Size, Share & Trends Analysis Report by Resolution, By Screen Size, By Screen Shape, By Operating System, By Distribution Channel, By Technology, By Region, And Segment Forecasts, 2023 – 2030. Accessed 10.04.2023. <https://www.grandviewresearch.com/industry-analysis/smart-tv-industry>

/3/ Brian Westover, John R. Quain. Smart TVs: Everything you need to know. Accessed 10.04.2023. <https://www.tomsguide.com/us/smart-tv-faq,review-2111.html>

/4/ Kathrine Teresa. What are the Best Smart TV Systems? Accessed 15.04.2023. <https://www.techowns.com/what-are-the-best-smart-tv-systems/>

/5/ David Herbert. What is React.js? (Uses, Examples, & More). Accessed 10.04.2023. <https://blog.hubspot.com/website/react-js>

/6/ Webpack. Accessed 10.04.2023. <https://Webpack.js.org/>

/7/BabelJS. Accessed 10.04.2023. <https://babeljs.io/>

/8/ Dmitro Brokhin. Spatial Navigation On Smart TV Apps. Accessed 10.04.2023. <https://noriginmedia.com/2022/03/29/react-based-spatial-navigation-on-smart-tv-app>

/9/ Tizen Docs. Create Your First Tizen TV Web Application. Accessed 10.04.2023. <https://docs.tizen.org/application/web/get-started/tv/first-app/>