

DEVELOPING AN INFORMATION SYSTEM SOLUTION WITH REACT

Case: FPT Information System (FIS)

Abstract

Author VU, VIET LINH	Type of publication Bachelor's Thesis, UAS	Published Autumn 2020
	Number of pages 39	Supervisor Aki Vainio
Title of publication Developing an Information System solution with React Case: FPT Information System (FIS)		
Name of Degree Bachelor of Business Administration, Business Information Technology		
Abstract <p>The subject of front-end JavaScript frameworks has been a popular topic for discussion within the IT community recently. Over the recent years there has been a numerous number of frameworks released on the market. These frameworks varied in terms of features and performances, and they all offer various options and benefits for front-end development, which can be challenging for new developers to choose the right framework for their project.</p> <p>The research presented in this thesis focused on the React JavaScript library, developed by Facebook. Multiple articles and studies regarding React were used to demonstrate the unique features and the benefits as well as drawbacks of the library. Furthermore, the thesis also covered some additional theory on two other equally popular and frequently used frameworks, Angular and Vue, to provide a comparison of these frameworks.</p> <p>A case study was also used for this research to provide a practical example of how React was implemented for the development of an IT solution. The thesis went through a detailed process of describing the usage of React in this case study to provide an evaluation of how React affected the project development. From here the benefits of using React were analyzed and demonstrated to help differentiate React from other</p>		
Keywords React, JavaScript frameworks, front-end development		

CONTENTS

1	INTRODUCTION	1
1.1	Research Background	2
1.2	Research Question and Objective	3
1.3	Research Motivation	3
2	THESIS DESIGN AND STRUCTURE	4
2.1	Thesis Structure	4
2.2	Research Method	5
2.3	Research Approach	6
2.4	Data Collection and Analysis	6
3	LITERATURE REVIEW	8
3.1	JavaScript Frameworks	8
3.2	React	9
3.2.1	Overview	9
3.2.2	Features	9
3.2.3	Pros and Cons of React.....	14
3.2.4	React Redux.....	16
3.3	Other Similar Frameworks	16
3.3.1	Angular	16
3.3.2	Vue.....	17
3.3.3	Comparison of These Frameworks	18
4	CASE STUDY	21
4.1	Project Description and Objectives	21
4.2	Structure and Design	21
4.2.1	Structure.....	21
4.2.2	Create Stock Export Request	24
4.2.3	View Stock Export Request	26
5	DATA ANALYSIS.....	29
5.1	Implementation of React.....	29
5.2	Effects of React	32
5.3	Other Choices of Frameworks	33
6	CONCLUSION.....	35
6.1	Results	35
6.2	Limitations	35

6.3	Validity.....	36
6.4	Suggestions for Future Research	36
7	REFERENCES	37

LIST OF ABBREVIATIONS

CSS	Cascading Style Sheets
DOM	Document Object Model
ES6	ECMAScript 6
HLR	Home Location Register
HTML	HyperText Markup Language
IT	Information Technology
MVC	Model-View-Controller
UI	User Interface

1 INTRODUCTION

As time progresses, web applications using JavaScript have gone a long way in terms of design. Nowadays with the help of modern JavaScript frameworks we can create a much more fluid and interactive design that allows users to navigate through with ease. Making use of JavaScript frameworks can drastically cut down the development time, while still maintain a functioning software with minimal errors and hassle.

With the popularity of JavaScript frameworks quickly rising comes the creation of many JavaScript web frameworks that allows different options for developers to choose. A quick search on Google will show millions of results about the latest JavaScript frameworks. One that gathers many attentions within the community is ReactJS (or React for short) developed by Facebook. React is a JavaScript library to help creating interactive user interfaces (UI) that can update and render as the user changes the data in real time (React 2020a).

Among the community, React is considered one of the top choices. According to a survey conducted by Stack Overflow in 2019, 74.5% of the developers who have had experiences with React were interested in continue working with it again, and 21.5% of the developers who have not yet tried it were willing to try it out (Stack Overflow 2019). It is noticeable that React is one of the most favorable choices, with a thriving community of up to 1500 contributors and growing on GitHub as of 2020 (Facebook 2020).

Most Loved, Dreaded, and Wanted Web Frameworks

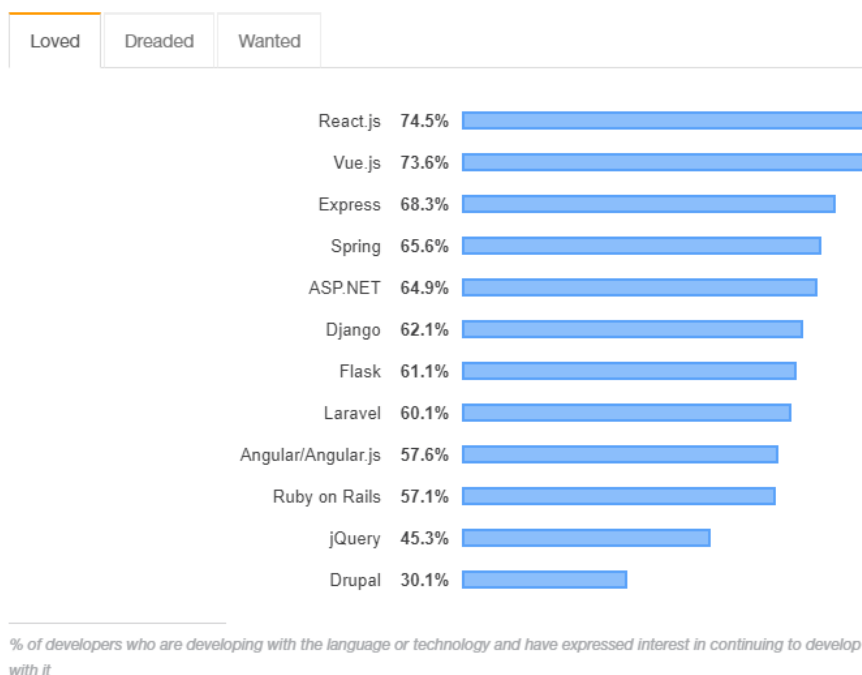


Figure 1: The most favorable frameworks in 2019 (Stack Overflow 2019)

1.1 Research Background

As there are many different choices for JavaScript frameworks it can be overwhelming for new web developers to choose the right one for their project. It is crucial to understand that these frameworks are not implemented in the same way and the benefits they provide depend on the purpose of the project. This thesis will focus on the React specifically by analyzing how it is implemented in a full-scale project to demonstrate the advantages and disadvantages it brings.

The thesis project is carried out at FPT Information System (FIS) – located in Hanoi, Vietnam – as an internship program. The company specializes in system integration, software development and IT services in many sectors. The project is to develop a web-based centralized sales management solution for a major Vietnamese mobile network operator using React. The project was initiated in early summer of 2019, and development and testing process was completed in June 2020.

1.2 Research Question and Objective

As a demonstration of how React JavaScript library can be utilized, the primary purpose of this thesis is to answer the following question:

- **In what ways can React benefit the development of a web application?**

To answer the question above, this thesis will go through the literature review process of describing the fundamentals of the React JavaScript library along with other equally well-known and frequently used frameworks, Angular and Vue, to help visualize the possibilities of said frameworks and how they can be implemented. The thesis will also investigate a case study where React was used during the front-end development process in order to provide necessary data and information regarding the research question.

1.3 Research Motivation

This thesis will be used mainly as an example to demonstrate the capabilities of React with the intention to differentiate React among other commonly used web frameworks. Another reason worth mentioning is that the majority of documents and articles regarding React focuses mainly on development on mobile platform rather than web-based platform. It should be noted that the case study used in this thesis is only one example of how React is implemented and preferably be used as a reference for those who want to learn more about React.

2 THESIS DESIGN AND STRUCTURE

This chapter will go through the methods used to form the structure of the thesis and demonstrate how the data is collected. By going through this process, it will illustrate the reasoning and effectiveness of the methodology used for this research.

2.1 Thesis Structure

The thesis will be divided into four main sections with a total of six chapters and an additional chapter for references and sources. These sections are: Introduction, Literature review, Case study and finally Conclusion.

Introduction: Provides an insight on the topic of the thesis and describes the process to collect the data	Chapter 1: Introduction
	Chapter 2: Thesis Design and Structure
Literature review: Theoretical research on JavaScript frameworks and examples of them.	Chapter 3: Literature Review
Case study: Practical research on a real-life case to understand how the web framework was implemented and the benefits it brings.	Chapter 4: Case Study
	Chapter 5: Data Analysis
Conclusion: Assesses the data collected and compare to the original findings.	Chapter 6: Conclusion

Table 1: Structure of the thesis

The first section includes the introduction of the thesis topic as well as the methods and approach used for the research. The second section will cover the theory of JavaScript frameworks to help familiarize with terminologies of JavaScript frameworks, followed by documentation on React library which includes its features and functionalities. This is then followed by the review of other popular JavaScript frameworks (Vue and Angular) to compare the similarities and differences between them. The third section will introduce the case study and describes how React was implemented into the project. This is followed by the collection and analysis of data from the project. The final section will summarize the findings by assessing the data collected from the case study and compare

them to the data from theoretical study. In addition to that there are also some insights from the author on the topic as well.

2.2 Research Method

Since the nature of this research is based on theory and case study, the type of data that will be used for this research will be qualitative. Information for literature review are collected from online journals and articles and data for the case study are collected throughout the internship. No survey was required as the research focus was on a single project.

There are three approaches to research design methods: inductive, deductive, and abductive approach. The differences between these approaches are shown in the table below (Table 2). Since the purpose of this thesis is to answer the research question of how React can benefit a web application development using a case study as a primary example, an inductive approach is used to achieve this objective. By choosing an inductive approach, the thesis will create a data-driven approach and form a connection between the data collected from the objectives of the research and the data collected from both theoretical and practical research (Chetty 2016).

	Deduction	Induction	Abduction
Logic	In a deductive inference, when the premises are true, the conclusion must also be true	In an inductive inference, known premises are used to generate untested conclusions	In an abductive inference, known premises are used to generate testable conclusions
Generalizability	Generalising from the general to the specific	Generalising from the specific to the general	Generalising from the interactions between the specific and the general
Use of data	Data collection is used to evaluate propositions or hypotheses related to an existing theory	Data collection is used to explore a phenomenon, identify themes and patterns and create a conceptual framework	Data collection is used to explore a phenomenon, identify themes and patterns, locate these in a conceptual framework and test this through subsequent data collection and so forth
Theory	Theory falsification or verification	Theory generation and building	Theory generation or modification; incorporating existing theory where appropriate, to build new theory or modify existing theory

Table 2: Differences between deductive, inductive, and abductive approaches (Business Research Methodology 2020)

2.3 Research Approach

The primary focus of this thesis is on the effects of React during the development of an Information System solution, specifically on performance and workload, and finally decide whether if React is suitable for developing other applications. As such it should be treated as a reference for those who are interested in or planning on using React for their next web application project.

2.4 Data Collection and Analysis

The thesis is divided into two main parts: Theoretical and Practical research. Theoretical research is done through collection of data from reliable online articles and journals. Practical research was conducted during and after the development stage of the project is completed.

For the theoretical research, the thesis includes various materials from different sources. Articles, studies along with books about React and JavaScript frameworks as well are collected from LAB's and Google Scholar online libraries, while documentation from official websites and developer's blogs are also included to help demonstrate the capabilities of these frameworks. Data collected from the case study consists of screen captures during the development and testing processes as well as extracts of code from the solution, both are used to demonstrate how React was implemented and its effects on the project.

The research will be based on one of Alan Hevner's Design-Science Research Guidelines, which are shown in the table below (Table 3). According to Hevner, a research must include at least one of these contributions:

- Design Artifact: The artifact used in the research must enable the solution to unsolved problems by extending the knowledge base or present existing knowledge in a new way.
- Foundations: The development of the research must extend and improve the existing design-science knowledge base.
- Methodologies: The use of evaluation methods and measures are crucial and must contribute to the design-science research. (Hevner et al. 2004, 87.)

Guideline	Description
Guideline 1: Design as an Artifact	Design-science research must produce a viable artifact in the form of a construct, a model, a method, or an instantiation.
Guideline 2: Problem Relevance	The objective of design-science research is to develop technology-based solutions to important and relevant business problems.
Guideline 3: Design Evaluation	The utility, quality, and efficacy of a design artifact must be rigorously demonstrated via well-executed evaluation methods.
Guideline 4: Research Contributions	Effective design-science research must provide clear and verifiable contributions in the areas of the design artifact, design foundations, and/or design methodologies.
Guideline 5: Research Rigor	Design-science research relies upon the application of rigorous methods in both the construction and evaluation of the design artifact.
Guideline 6: Design as a Search Process	The search for an effective artifact requires utilizing available means to reach desired ends while satisfying laws in the problem environment.
Guideline 7: Communication of Research	Design-science research must be presented effectively both to technology-oriented as well as management-oriented audiences.

Table 3: Guidelines to Design-Science research (Hevner et al. 2004)

Following Hevner's guidelines to Research Contributions, this thesis will include the contributions of the design artifact and methodologies for the research. The design artifact in this case is the web application developed with React which will be used as an demonstration of how React is utilized for a specific purpose, while the methodologies consist of the observations and analysis made during and after the development of the project.

Analysis was done by comparing the data collected from the theoretical study and case study in order to evaluate the advantages and disadvantages React brings to the project. In addition to that hypotheses are also made to determine whether the use of other frameworks would affect the development differently. The analysis also provides some insights from the perspective of the author as well.

3 LITERATURE REVIEW

The following chapter will cover the theoretical background of the research. It will explain the fundamentals of JavaScript web frameworks, follow by the introduction to React and examination of the features that React brings to JavaScript development. It will also cover some additional information on two other popular and similar JavaScript frameworks (Vue and Angular) to compare the similarities as well as differences between these frameworks.

3.1 JavaScript Frameworks

JavaScript is the most popular language used for designing and building web pages. It is a cross-platform, object-oriented language that is mainly used in a browser environment and sometimes can also be used in a non-browser environment as well. JavaScript runs on the client side of the web and is used alongside with HTML and CSS to create web pages and control their behavior on screen. (Mozilla 2020a.) With the increasing popularity of JavaScript over recent years, many developers created customized tools that can help with problems and issues in JavaScript. These tools are packaged into what is called a library. A JavaScript framework is a library that offers more choices for development in terms of how the application is built. These choices allow for predictability and maintainability of the application's lifespan. JavaScript frameworks are created with the intention to build code that can update the UI with every changes made in the application as well as a way to help visualize what the UI should look like during development. (Mozilla 2020b.)

A feature that is prominent in JavaScript frameworks is the usage of the Document Object Model (DOM). DOM is a programming interface that offers dynamic access to the content and structure of a HTML web page. It allows the web page to be loaded into the browser as a document object and its HTML elements are then manipulated by JavaScript. Additionally, DOM does not require to be installed individually and most web browsers implement DOM according to the W3DOM standard. (Levlin 2020, 16.)

Most JavaScript frameworks follow the principal of the Model-View-Controller (MVC) design pattern. The pattern involves three elements: the Model which

handles the storage and process of data; the View displays the data collected from the Model to the user; the Controller manages all the data from user inputs and updates them to the Model (Mariano 2017, 32-33). MVC pattern is primarily used for desktop application development but it can be utilized for web applications as well (Levlin 2020, 26-27).

3.2 React

3.2.1 Overview

An early prototype of React, developed by a software engineer at Facebook named Jordan Walke, was first introduced in 2011 under the name of FaxJS. Walke finalized the prototype and created React in 2012, which was soon integrated into Facebook and Instagram in the same year. In 2013 React became open-source and later became available in Ruby on Rails and Python Applications. This is followed by the release of React Native, an extension of React for mobile development on Android and iOS, in 2015. Ever since then React consistently pushes out many releases throughout the years, improving and introducing new features for users. (Hámori 2020.)

React is known to be a versatile tool that can be utilized on both desktop and mobile platforms with many distinct features. One of them is the ability to create interactive rather than simple static pages that can update and render data after each input from the user, thus allows for a seamless UI without the need to refresh the entire page every time a change is made. The reason for this is due to React's structure is based on components that allows for the design of complex UIs due to the component logic is written in JavaScript (React 2020a).

3.2.2 Features

As a JavaScript library, React can be easily integrated into a website or can be used to build an application from scratch. React only requires a basic understanding of HTML and JavaScript and are suitable for both beginners and experienced developers. Many features of React are listed below:

- **JSX:** JSX is a syntax extension to JavaScript that can be used alongside with the JavaScript code to help visualize what the UI would look like. Each

JSX attributes that are enclosed in quotes will become a string (Singh & Tanna 2018, 48). Using JSX can help reduce the amount of code needed to be written, thus allow for a cleaner and more comprehensible code structure (Mariano 2017, 54).

```
const element = <h1>Hello, world!</h1>;
```

Figure 2: Example of a JSX element

- React DOM: An alternate version of the traditional DOM, React DOM is used to handle the rendering of elements in React. These elements are different from traditional DOM elements, in which React elements are simple objects used by React DOM for translation purposes. Rendering in React is done using the *ReactDOM.render()* command. (Levlin 2020, 45.) When updating, React compares the elements together in order to ensure the right element whose content have changed is updated by the DOM, thus reducing the probability of bugs in the code (React 2020b).

```
function tick() {  
  const element = (  
    <div>  
      <h1>Hello, world!</h1>  
      <h2>It is {new Date().toLocaleTimeString()}.</h2>  
    </div>  
  );  
  ReactDOM.render(element, document.getElementById('root'));  
}  
  
setInterval(tick, 1000);
```

Figure 3: Example of the *ReactDOM.render()* command

Rendering with React DOM is done by using a form of virtual DOM, which allows the React elements to be used and updated as DOM nodes every time changes occur. React DOM can render components as HTML strings that can be used to generate content on the server side as well as on the browser side as an HTML file. (Wilson 2018, 219.) Another characteristic of React's virtual DOM usage is the minimal DOM manipulation handled in the

application, therefore reducing computing resources required and making React updates faster (Levlin 2020, 44).

- Components: One of the core elements of a React application is components which are what the application is based around on. Components are treated as JavaScript functions and they accept inputs and render them as React elements on screen (React 2020c). The type of inputs that components receive are called props which will be discussed later on. There are two types of components: Function components and Class components, which are shown in the figures below (Figure 4 and 5). Function components are plain JavaScript functions that receive data from other components called props, while Class components are defined by ES6 classes to create Stateful components and render React elements using the *render()* method. Components provide a way to divide web applications into reusable parts of code for easier management and development. (Wilson 2018, 217.)

```
function Welcome(props) {  
  return <h1>Hello, {props.name}</h1>;  
}
```

Figure 4: Example of a Function Component

```
class Welcome extends React.Component {  
  render() {  
    return <h1>Hello, {this.props.name}</h1>;  
  }  
}
```

Figure 5: Example of a Class Component

React components are consisted of elements which describe the information on the screen and are rendered by React DOM. Each time an element is rendered it is passed to *ReactDOM.render()*. Once rendered its attributes cannot be changed so to update the rendered element a new

element is created and also passed to `ReactDOM.render()`. This explains how components rendering work: calling the component using `ReactDOM.render()` with the user input as the props, which the component returns a new element as the result. (React 2020c.)

```
function Welcome(props) {  
  return <h1>Hello, {props.name}</h1>;  
}  
  
const element = <Welcome name="Sara" />;  
ReactDOM.render(  
  element,  
  document.getElementById('root')  
);
```

Figure 6: Example of a Component rendered by the React DOM

- Props: Properties, or props for short, are plain JavaScript objects that contain raw data from components and help keeping the consistency of the UI. Props is used along with State to build interactive application and generally do not change over time. (Singh & Tanna 2018, 59.) A general rule of React is that props cannot be modified within a component since function components must always return the same value for the same inputs they have and as such they are prohibited from changing their inputs (React 2020c).

```
function sum(a, b) {  
  return a + b;  
}
```

Figure 7: Example of Props *a* and *b* inside *sum* function

- State: State is similar to props in terms of functionality – they are both objects used to store raw data within the component. However, unlike props, data can be merged into State and re-rendered in the component

every time an update is made to the UI, meaning the data in the State can be changed. (Singh & Tanna 2018, 59.) A State can be modified and updated by using the `setState()` method which allows for React to detect changes in the State and update the component. Components that manage their own State are called Stateful components and are written using ES6 classes. (Wilson 2018, 228.)

```
constructor(props) {  
  super(props);  
  this.state = {  
    posts: [],  
    comments: []  
  };  
}
```

Figure 8: Example of declaring the 2 State posts and comments

When utilizing multiple components in a React application, it is important that every class components have a life cycle method to prevent resources being wasted when components are destroyed (React 2020d). Adding a life cycle method provide control over the creation and termination of the components as well as their properties, such as when they receive new properties or when should they be updated. Some commonly used life cycle methods include:

- *constructor(props)*: This method is used when initializing the first instance of the component as well as the initial State of the component.
- *componentDidMount()*: This method is used after the first render call and is can be useful for accessing the DOM or making HTTP requests.
- *componentWillUnmount()*: This method is used before a component is destroyed and is commonly used to clear timers or cancel network requests.

- *componentDidUpdate()*: This method is used after the render method or when an update occurs, with the exception of the first render method. (Wilson 2018, 230.)

When modifying State there are three things to know in order to ensure the correct usage of State. The first is that State can only be modified using *setState()* and also cannot be modified directly. The next thing to know is that props and State can be updated asynchronously and does not have to depend on the same update. Multiple *setState()* calls can be grouped into a single update by React to preserve performance. Lastly, when calling *setState()*, React may merge the component's object into the current State. This means that a State with multiple variables can be updated independently using multiple separate *setState()* calls. (React 2020d.)

- Hooks: The latest addition in React version 16.8, Hooks allow for the use of State without the need for a class. In theory, Hooks are functions that can be used to call for State and life cycle features from function components. There are two types of Hooks: State Hook and Effect Hook. State Hooks are used to add State to function components. Effect Hooks are used to perform side effects in function components, for example data fetching or manually changing the DOM. Hooks are backwards compatible and can be integrated into existing code alongside classes without the need to re-write the components. (React 2020e.)

3.2.3 Pros and Cons of React

The most notable benefit of using React is the ability to re-use components throughout the application. Applications with React can be split into many individual pieces of code, which they can be worked with asynchronously (Wilson 2018, 217). By isolating these components React lets developers work on each part of the application individually, making the process much more efficient and less opportunities for bugs. In addition to that the usage of a virtual DOM also helps React with proficient rendering capabilities, allowing it to perform consistently faster compare to other popular frameworks (Levlin 2020, 65-66). This enables React to be a powerful tool for developers to create fast and interactive complex UIs efficiently with less time required. Another benefit of using React is

the option for 3rd party extensions thanks to React being open-source, which allows for even more customizable options for UI design. A few examples include Ant Design, PrimeReact, react-i18next, and reactstrap. Lastly, React has one of the largest community on GitHub with over 1500 contributors, including more than 158 thousand Git repository stars and over 4 million active users (Facebook 2020). This provides React with a strong supporter base that can offer many improvements to its functionality with new updates as well as providing guidance to others who are willing to learn React.

As one of the most well-known choice for building web pages and applications, React still has its own drawbacks, however. The biggest of which is the use of JSX in React. JSX is a relatively new feature for JavaScript that is still not fully optimized for many browsers, which causes application that utilizes it to perform much slower compare to using React without it (Mariano 2017, 76-77). In addition to that, the lack of clear documentation is another disadvantage of React due to it still being a fairly new technology, having only been on the market for less than 10 years. Another disadvantage of React is the fact that it is only a JavaScript library rather than a complete framework. This causes React to lack some features that other frameworks have to make it a fully equipped tool for front-end development.

Regardless, React is still a powerful tool to consider among other choices in the JavaScript frameworks landscape that will be discussed next. The table below summarizes the pros and cons of React that were previously mentioned.

Pros	Cons
Re-usable components	JSX needs more improvements
Virtual DOM allows for fast rendering	Poor documentation
Many options for 3 rd party extensions	Not a complete framework
Large community	

Table 4: Pros and Cons of React

3.2.4 React Redux

As UIs become more and more complex, State management can become increasingly more difficult. As a solution to this issue, React introduces a separate library called Redux that works as a State manager for React applications. Redux is designed with the intention to make State mutations become more predictable by dictating how and when updates should happen. Redux consists of three core concepts:

- **Store:** an object that is used as a State container that manages the State within the application and cannot be manipulated or mutated directly.
- **Action:** another object that displays the changes within the application and is used to direct changes to the store.
- **Reducer:** a function used to combine State and Action together, it takes State and Action as arguments and returns new State to the application. (Singh & Tanna 2018, 119-120.)

There are three fundamental principles to remember when using Redux. The first is each application can only contain a single store which acts as a single source of truth. The second is that State is read only. This means that State can only be altered by Actions instead of other functions. Finally, any changes made to the State must be described using Pure Functions, which are functions that always return the same value with every set of arguments passed to them. (Singh & Tanna 2018, 120.)

3.3 Other Similar Frameworks

3.3.1 Angular

Originally created as AngularJS by a Google employee, Miško Hevery, Angular became open-source in 2010 and was re-written as Angular2+ (Gavigan 2018). Since support for the original AngularJS has stopped, this thesis will focus on the newer version which is Angular2+, or Angular. It is a TypeScript-based framework rather than a pure JavaScript framework like AngularJS, the reason for this is because of additional technical and performance improvements (Manjunath 2018).

Angular utilizes a component-based architecture with a root component in every application. Each component consists of a class for handling the logic and a template for the view layer. Angular also includes the template structure which was brought from AngularJS and reworked with new features. One of which is that each component has its own template attached to it. Another feature of Angular is dependencies injection, a design pattern that handles dependencies in the application and insert them into components. (Manjunath 2018.)

When first released, AngularJS was the top choice for quick prototype building despite having many issues relating to performance. These issues were soon resolved in Angular with improvements and changes to the core features of AngularJS which leads to many switching to Angular for their projects. (Manjunath 2018.) Angular is currently being improved by Google and the community to help build feature-rich UI components and provide tools to develop custom components as well. (Angular 2020a)

3.3.2 Vue

A popular JavaScript framework that shares many similarities with React, Vue is developed by Evan You after working with Google. According to You, Vue was created as a lightweight alternative to Angular that has many similar features but with less extra concepts. (Cromwell 2016.) An early version was released around 2013 and followed by an official release in 2015. Vue is a progressive open-source framework for building UIs and designed to be incrementally adoptable. The core library of Vue focuses exclusively on the view layer and can be integrated with other libraries and projects. It also supports development of complex single-page applications. (Vue 2020a.)

Vue includes an HTML-based template syntax. A signature characteristic of Vue is the declaration of syntaxes using the “v-” prefix which is its way of identifying Vue-specific attributes. Examples of this includes *v-bind* and *v-on*, which are two of the most commonly used commands in Vue. (Vue 2020b.)

```

1 <!-- full syntax -->
2 <a v-bind:href="url"> ... </a>
3
4 <!-- shorthand -->
5 <a :href="url"> ... </a>
6
7 <!-- shorthand with dynamic argument -->
8 <a :[key]="url"> ... </a>

```

Figure 9: Example of the *v-bind* command

```

1 <!-- full syntax -->
2 <a v-on:click="doSomething"> ... </a>
3
4 <!-- shorthand -->
5 <a @click="doSomething"> ... </a>
6
7 <!-- shorthand with dynamic argument -->
8 <a @[event]="doSomething"> ... </a>

```

Figure 10: Example of the *v-on* command

Being one of the most recent framework on the market, Vue may seem to have a moderate community with less than 400 contributors and 100 thousand users, but it also has over 174 thousand repository stars, showing a significant amount of attention for Vue (Vue 2020a). With a small but steadily rising community, it is easy to notice that popularity is increasing fast and this shows that Vue can be an effective technology for JavaScript development that can rival with older frameworks on the market.

3.3.3 Comparison of These Frameworks

The most noticeable common feature between these frameworks is that they all share a components-based structure. By separating the application into various independent parts, it allows developers to work on each individual functions of the application asynchronously, thus leading to a more flexible development process. Each of these components-based frameworks has different ways of utilizing

components templates: React uses .jsx files; Angular with a combination of .html, .css, .ts, and .spec files; Vue uses proprietary .vue files (Levlin 2020, 48,51).

A similar feature that is seen in React and Vue is the use of a virtual DOM. React utilizes the virtual DOM to handle many of the rendering tasks using as little DOM manipulation as possible, thus reducing the time and resources needed for updates. Similarly, Vue handles the management of the real DOM using a separate virtual DOM. The virtual DOM in Vue consists of multiple virtual nodes that manage what information should be displayed and interacts with real DOM to update the data on the HTML page. In contrast, Angular does not make use of virtual DOM, but instead manages all DOM manipulations directly by converting components into classes that can be displayed onto the DOM. (Levlin 2020, 44, 47, 50.) Another similarity between React and Vue is that their core libraries both focus exclusively on the View layer of the MVC model (Mariano 2017, 25), whereas Angular and Angular JS are based around the MVC architecture, with Angular's newer architecture leans towards a component-based architecture more (Manjunath 2018). This means that React and Vue are best optimized for UI development while Angular is suited for general web application development.

As for popularity, React is considered to be a well-known library with a large following. Compared to Angular, both of them share a relatively large community with thousands of contributors on GitHub (Facebook 2020, Angular 2020b). Additionally, they are also developed and supported by major leading industries, with React by Facebook and Angular by Google, respectively. In contrast, Vue is developed and managed by a community of developer and receives financial support from donations.

Perhaps the biggest difference that separates React from the other two frameworks is the fact that React is only a UI library rather than a full-fledged JavaScript framework. In terms of features, React falls behind Angular and Vue due to the fact that the other two are complete and packed frameworks. Despite this React is still a popular choice for front-end development. This is because unlike large and feature-packed JavaScript frameworks, React only focuses on building the UI, and as a result makes it an optimal tool for those who wants to put an emphasis on interfaces (JSComplete 2020).

The diagram below (Figure 11) summarizes all the similarities and differences that were described earlier.

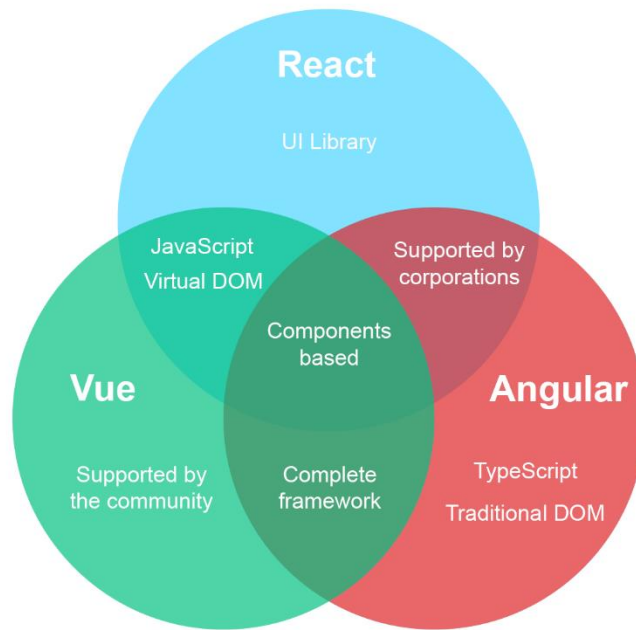


Figure 11: Comparison of React, Vue, and Angular

4 CASE STUDY

The following chapter will introduce the client company and the project developed for the client company using React. It will also demonstrate the structure as well as functionalities of the solution during development.

4.1 Project Description and Objectives

The project was carried out by FIS, a subsidiary of FPT Group. FPT Group is the largest IT service company in Vietnam that specializes in ICT-related services. FIS is a branch of FPT Group that is dedicated in providing IT solutions for many services and public sectors. The client company that the project was developed for is MobiFone, a major mobile network operator and telecommunications provider in Vietnam.

The goal of the project was to create an IT solution that works as a centralized sales management system. The system is designed for the purpose of managing the sales along with inventory of products and services related to telecommunications. The solution is created specifically for the client company as a new upgrade to their existing system and can be accessed through web browsers on desktop platform.

4.2 Structure and Design

4.2.1 Structure

When accessing the system, the user will be taken to the login page. The design of the login page is minimalistic, with the inclusion of 2 input fields for the username and password, and a sign in button below. In order to access the system, the user must login by entering their username and password. Due to the nature of the system designed exclusively for MobiFone, only employees of MobiFone can use their credentials to login. This means that user information is entered into the database separately by the administrator rather than directly on the browser. Users are required to enter their credentials every time they access the system as well as after a long period of inactivity when using the system, for the purpose of security.

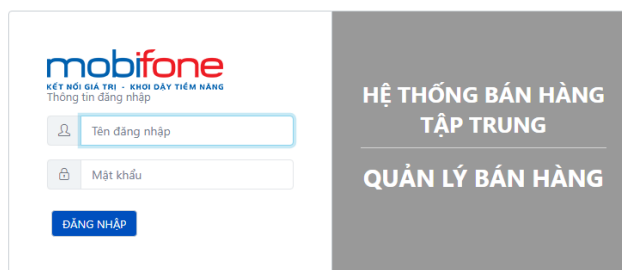


Figure 12: The login page

As a centralized sales management system, it features roughly a hundred different functions within the system. Each of these functions are contained within several categories that manage various sectors of the system. These sectors include inventory, invoice, e-invoice, sales, revenues, distribution channels, and many more. The categories are shown on a side menu that can be accessed at any time in the system after logging in by clicking on the list icon located on the top left corner of the screen.

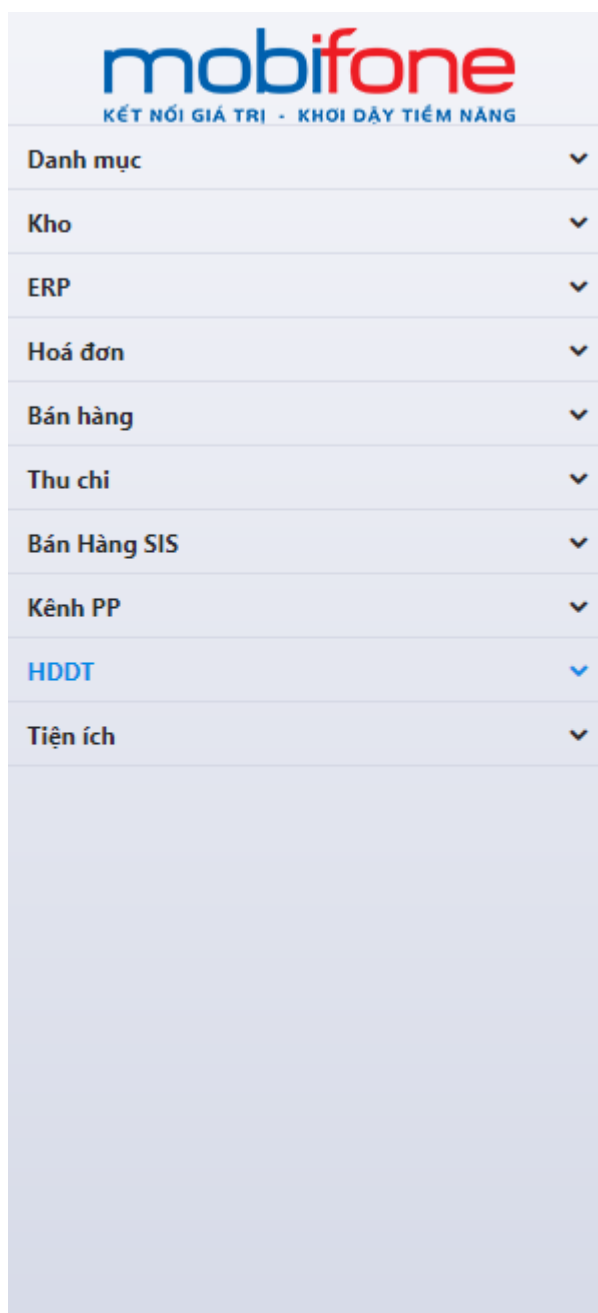


Figure 13: The side menu

The entire system consists of multiple different categories, each category consists of a number of functions that serve a certain objective that belongs in that category. These functions act as multiple smaller application that compose the entire solution. Users can access them by clicking on a category, which will extend into a drop-down menu that lists the functions. Some categories can contain a sub-category or more if they include functions for a specific objective. When the user

accesses a function, the browser will load a new page and direct the user to the said function page. Only one function can be worked with at a time.

4.2.2 Create Stock Export Request

An example of a function from the solution is OrderEstablish. This function was designed to build the Create Stock Export Request page, where users can create new request for stock exports, or to modify or delete existing requests. The page can be accessed by the side menu, through the Inventory category and followed by the Import and Export with Superiors sub-category.

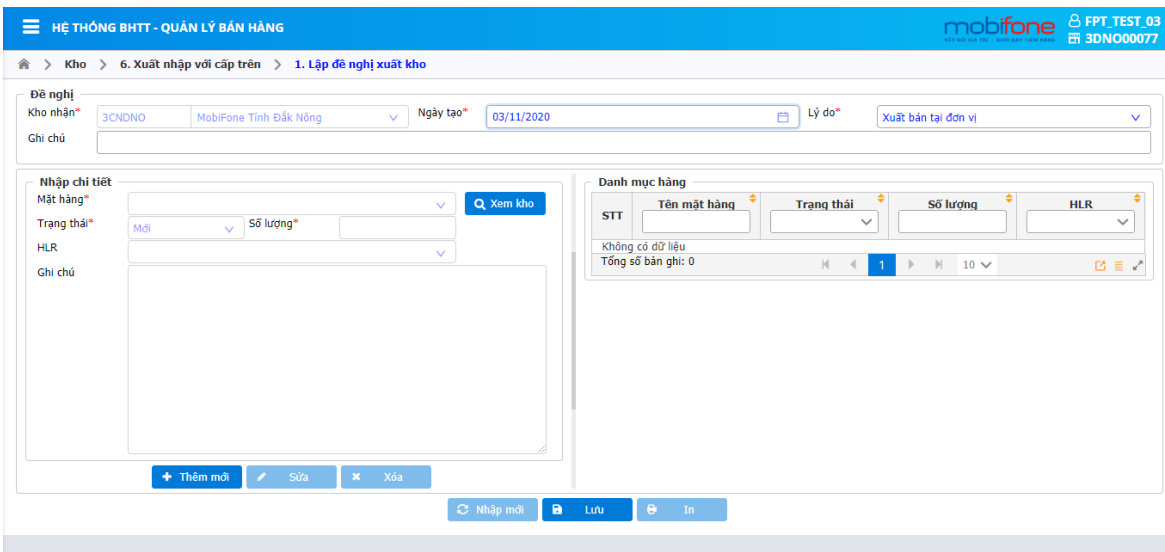


Figure 14: Create Stock Export Request page

The page consists of three main parts: the Request tab, Details tab, and Products tab. The Request tab is located on top of the page while the other two are grouped together and placed below, with the Details tab on the left and the Products tab on the right. At the bottom of the page are 3 buttons. From left to right are the Insert New button for submitting a new request; the Save button to save the changes made on the page; and the Print button, which will let the user print out the document from the page.

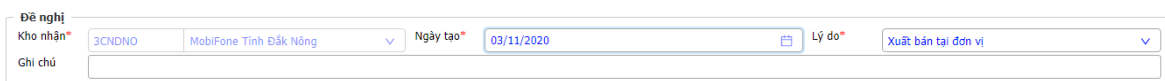
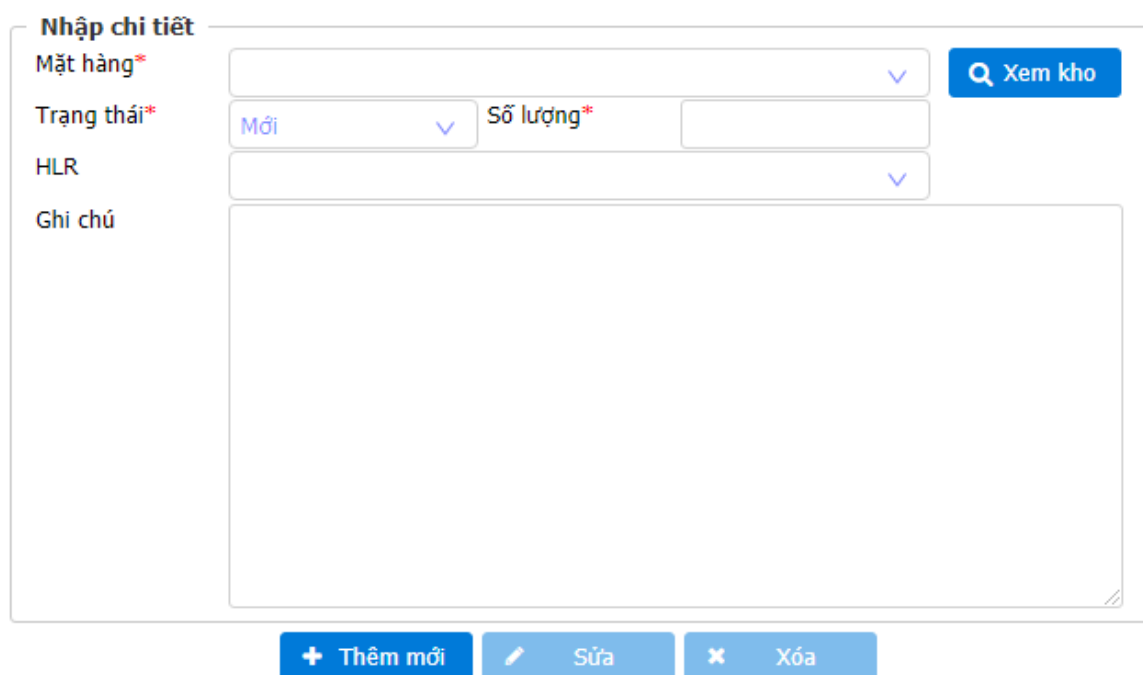


Figure 15: The Request tab of Create Stock Export Request page

On the request tab are 4 input boxes that receive data from the user. The 3 boxes on top let the user input the required data for the name and serial code of the receiving warehouses, the date of request creation, and the reason for the request. All of these boxes only allow the user to input a fixed type of data using drop down menus and calendar to prevent the user from inputting invalid data. The bottom box is a text field which allows the user to insert any additional note to their request if needed.



Nhập chi tiết

Mặt hàng*

Trạng thái* Số lượng*

HLR

Ghi chú

Figure 16: The Details tab of Create Stock Export Request page

The Details tab consists of 5 input boxes that let the user to put in the data for name of product, product's status, as well as quantity, HLR (Home Location Register), and additional notes. Each input boxes that contain the data required for the operation are marked with a red asterisk on the right side of their titles. On the top right of the tab is the View Inventory button that will display a list of all the products in the inventory on the Products tab. At the bottom of the tab are 3 buttons, which let the user to add a new product, modify an existing product, and delete a product from the list.

Danh mục hàng				
STT	Tên mặt hàng	Trạng thái	Số lượng	HLR
Không có dữ liệu				
Tổng số bản ghi: 0				

Page navigation: 1 / 10

Figure 17: The Products tab of Create Stock Export Request page

All the items that will be used for export will be displayed on the Products tab. This part consists of a table that shows the order, name, status, quantity, and HLR of the products. The top row of the table where the headings of each columns are displayed contains small input boxes for each of the product's classification, with the exception of the order column. These boxes receive data from the user in forms of a string, an integer, or a fixed data from a drop-down menu to help with searching for a specific product. At the bottom of the table are the total number of products, page navigation buttons, a drop-down menu to select the number of items to display on the table and options to expand or extend the table.

4.2.3 View Stock Export Request

Another example of a function is the OrderView. This function was designed to work alongside with the previously mentioned OrderEstablish function and is used to build the View Stock Export Request page. Unlike the former, however, this page is dedicated to viewing and sorting the requests and does not make any changes to the database of the system. The page shares the same category as the Create Stock Export Request page and can be accessed by the same category path.

Figure 18: View Stock Export Request page

The page includes 3 main parts: the Search tab, located at the upper part of the page, and 2 tabs below, both consist of tables that display information regarding stock export requests which are the Requests List and Products List. There are 2 buttons positioned at the bottom of the page that corresponds to printing the request or to cancel the request and reset the data on the page.

Figure 19: The Search tab of View Stock Export Request page

The Search tab is used by the user as a tool to fetch the data regarding stock export requests from the system's database. It consists of 3 input boxes, 2 of which are for inputting the start and end date of the desired request, and the other one includes a drop-down menu for user to select the status of the request. To search for a request, the user clicks on the Search button located at the bottom right corner of the tab which will read the input data and filter the results into the tables below.

Danh mục đề nghị							
STT	Ngày đề nghị	Người lập	Trạng thái	Người duyệt/hủy	Ngày duyệt/hủy	Lý do hủy	Ghi chú
Không có dữ liệu							
Tổng số bản ghi: 0							
<< 1 >> 5							

Danh mục hàng						
STT	Tên MH	Trạng thái	Số lượng	SL duyệt	HLR	Ghi chú
Không có dữ liệu						
Tổng số bản ghi: 0						
<< 1 >> 5						

Figure 20: The Requests List and Products List of View Stock Export Request page

The other 2 parts of the page are the Requests List and the Products List tabs. Both are displayed using tables that are formatted to display all necessary information regarding the inventory of the company. Any changes made to the database using the OrderEstablish function will be displayed on the Requests List tab as detailed lists of requests with information regarding names of individuals who made and accepted or rejected, the dates of creation and/or cancelation, and current status of the requests. This tab also includes built-in search boxes to look for specific requests. The Products List tab includes information of products involved in the requests, such as product's name, status, quantity, and so on. Both tables also have a sorting feature that lets users reorganize the lists by the table's categories in ascending or descending order that can be access by clicking on the arrow icon on the top right corner of each category.

5 DATA ANALYSIS

The following chapter will examine how React was implemented in the case study to analyze the importance of React in the project. This chapter also demonstrates a few hypotheses made by the author to show the differences in the development process if other frameworks like Vue or Angular were chosen instead of React.

5.1 Implementation of React

The project makes use of React's re-usable components feature by implementing various components into the structure of the system. These components comprise of many React functions and classes that are used for various tasks. These include formatting input from the user, rendering UI elements such as dialog boxes and error notifications, adding extra features, for example a date and time picker, to the UI. These components provide developers with more options to add customizable features to the UI and they are an integral part to the structure of the system.

An example of a React component that is prominently used is FTUComponent, which is a collection of functions and classes that is used frequently during development. This component was designed by the developers of FIS with the intention to perform several tasks in terms of functionalities that caters to various operations of the solution. One of which is formatting user input from Vietnamese to English text to ensure the data provided by the user can be processed by the browser. This makes FTUComponent a crucial factor to the operation of the system and thus it is used in virtually all of functions in the solution.

```

formatInput = (str, filter) => {
  if (filter === "number") {
    const _ = require('lodash');
    str = str.toString().trim();
    str = str.replace(/[^0-9]/g, ''); //Loại bỏ ký tự không phải số.
    return str ? _.toInteger(str) : null;
  }
  if (filter.includes("alpha")) {
    // Xóa space
    str = str.replace(/\s+/g, '');
    //Bỏ dấu tiếng Việt.
    str = str.replace(/à|á|ạ|â|ã|â|ã|â|ã|â|ã|â|ã|â|ã|ã|ã/g, "a");
    str = str.replace(/è|é|ê|ë|è|é|ê|ë|è|é|ê|ë/g, "e");
    str = str.replace(/ì|í|î|ï|ì|ï/g, "i");
    str = str.replace(/ò|ó|ọ|õ|ô|õ|ô|õ|ô|õ|ô|õ|ô|õ|ô|õ/g, "o");
    str = str.replace(/ù|ú|ụ|ủ|ừ|ứ|ử|ữ|ừ|ử/g, "u");
    str = str.replace(/ỳ|ý|ỷ|ỹ|ỳ|ỹ/g, "y");
    str = str.replace(/đ/g, "d");
    str = str.replace(/À|Á|À|Á|À|Á|À|Á|À|Á|À|Á|À|Á|À|Á/g, "A");
    str = str.replace(/È|É|È|É|È|É|È|É|È|É|È|É|È|É|È|É/g, "E");
    str = str.replace(/Ì|Í|Ì|Í|Ì|Í/g, "I");
    str = str.replace(/Ò|Ó|Ò|Ó|Ò|Ó|Ò|Ó|Ò|Ó|Ò|Ó|Ò|Ó|Ò|Ó/g, "O");
    str = str.replace(/Ù|Ú|Ù|Ú|Ù|Ú|Ù|Ú|Ù|Ú|Ù|Ú|Ù|Ú|Ù|Ú/g, "U");
    str = str.replace(/Ỡ|Ỡ|Ỡ|Ỡ|Ỡ|Ỡ/g, "Y");
    str = str.replace(/Đ/g, "D");
    // Combining Diacritical Marks
    str = str.replace(/\u0300|\u0301|\u0303|\u0309|\u0323/g, ""); // huyền, sắc, hỏi, ngã, nặng
    str = str.replace(/\u0306|\u030c|\u031b/g, ""); // mũ â (ê), mũ ă, mũ ơ (ư)
    if (filter === "alpha_number")
      //Bỏ ký tự không phải chữ cái và số.
      str = str.replace(/[^0-9a-z_]/gi, '');
    else if (filter === "alpha")
      //Bỏ ký tự không phải chữ cái.
      str = str.replace(/[^a-z]/gi, '');
  }
  return str;
}

```

Figure 21: A snippet of the translation function from FTUComponent

The project also implemented several 3rd party libraries that React supports into the design and development of the UI. One of which is the use of the DatePicker template by Ant Design, which allows for the input of data as date and time (Ant Design 2020). With DatePicker, instead of manually input the date into text fields, users can choose the desired day, month, and years from a mini calendar that shows up when they click on the input box. Examples of this template can be seen in the input boxes of the OrderEstablish and OrderView functions in the previous chapter.

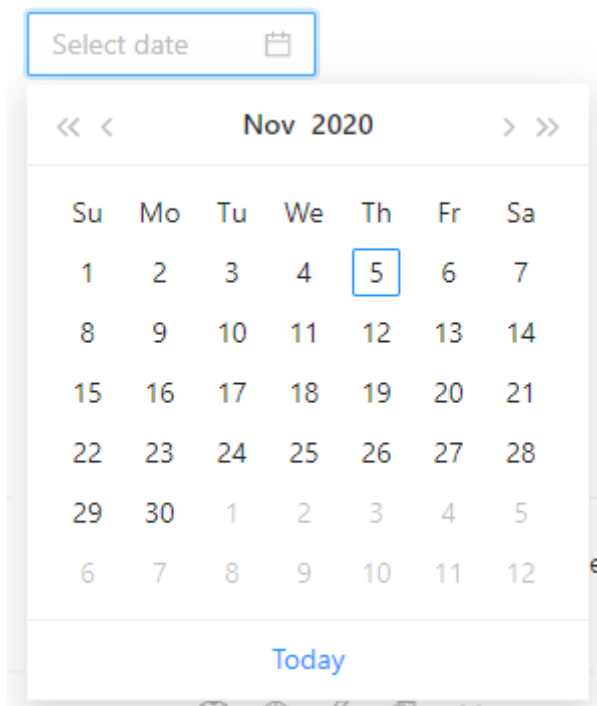


Figure 22: Example of DatePicker (Ant Design 2020)

Another example of usage of 3rd party extensions is the use of PrimeReact's template Fieldset. With this template, each section of the page can be separate into individual pieces or group together into one general section (PrimeFaces 2020). Using Fieldset helps developers to divide the UI into multiple sections that can work side by side while still maintaining a clean and functional interface. As shown in the functions mentioned in the prior chapter, Fieldset was used to separate the sections of the page by using a thin border surrounding each parts, with their title shown on the top right corner.

```

<div>
  <DialogDetailStock/>
  <Fieldset legend={this.trans("suggestTitle")} style={{paddingBottom: "10px"}}>
    <div className="row">
      <div className="col-md-1">
        <Label value={this.trans("importWarehouse")} required={true}/>
      </div>
    </div>
  </Fieldset>
</div>

```

Figure 23: Example of Fieldset used in OrderEstablish function

Due to the high complexity of the solution, React Redux was utilized to help with State management for the project. An example of React Redux usage can be shown below (Figure 24). In this example, the *mapStateToProps* function is called

every time an update is made to the Store. The `connect()` command is called to connect a React component to the Redux store, in this case is the `OrderView` function. This is done so that the State inside the `OrderView` function can pass on data to other functions such as `OrderEstablish` when needed as well.

```
const mapStateToProps = state => ({
  user: state.user.info
});

export default withTranslation("partner")(connect(mapStateToProps)(OrderView));
```

Figure 24: Example of React Redux used in `OrderView` function

5.2 Effects of React

As a front-end JavaScript library, applications using React can benefit from its client-side rendering capabilities. It allows the JavaScript elements to be rendered almost instantaneously in the browser, which lets data to be changed without the need to reload the entire page. This also puts less pressure on server rendering as well, especially in the case of multiple users access the system. React's rendering capabilities can be seen in the `OrderEstablish` function where changes made on the Details tab are updated and displayed immediately on the Products tab, allowing for faster data processing speed.

Another evident advantage of React is that it is an open-source library. This means that it can be utilized freely for the project without the need for licensing, and that it can be freely modified depending on the development status. Being open-source also means that there are various supported extensions made by 3rd parties that can help the design of the UI without having to modify the HTML and CSS of the pages. As demonstrated before, the usage of UI templates provides developers a short-cut when building pages, therefore requires less time needed for front-end development and more for debugging and back-end aspects as well.

Furthermore, the use of React components also helps with the development of the project. It allows developers to work with each pieces of the solution individually rather than having to worry about the entire solution as a whole, thus leads to a more efficient working progress and less chance of making errors. Additionally, these individual pieces can be called into other components as well when needed

without having to reprogram them again, which helps with saving time and effort for development.

As previously described in the case study, the project contains a high number of functions for multiple different operations of the sales system, which means the amount of State to manage between these components can be overwhelming. This is why the utilization of React Redux also holds a crucial role in the development of the project. By using Redux, developers can always have control over the State manipulations within the functions and avoid monitoring changes made in State manually, which can be particularly time consuming and leads to mistakes.

The usage of React in the project also has a few shortcomings. One of which is the heavy UI-oriented nature of React. Due to being only a JavaScript library for front-end development, React does not support additional features in terms of functionality of the application that can aid the development. This means that while the design of the UI can be done with ease, the application can still be prone to errors and bugs. Another drawback is the high usage of extra React extensions when designing the UI. While rich in features, the aesthetics of the UI lack any originality due to most of the designs are done by 3rd party libraries. Fortunately, these downsides of React are not impactful and does not severely affect the development of the project.

5.3 Other Choices of Frameworks

When looking at other alternative options for JavaScript frameworks that has been mentioned in this thesis, Vue would be the most suitable choice. Both of them share many similar characteristics such as components-based architecture, focus on View, virtual DOM usage and more (Vue 2020c). Vue can theoretically be implemented the same way as React since they are both heavily UI-oriented frameworks. However, in terms of popularity, Vue still falls behind React and therefore it may not have as many extra supported extension as React has. This means that the selection for UI customizability is more limited and not as diverse when compare to React.

The other option to consider is Angular. Compare to React, Angular is more feature-rich due to being a complete framework. In terms of functionality, Angular is suited for large scale application development due to its programming style, features, and documentation (Levlin 2020, 47). In spite of this, Angular is developed to be worked better with TypeScript, which requires developers to know extra knowledge on the TypeScript language to fully utilize it. Furthermore, due to having more features, Angular has a larger total file size when compare to React, which could cause the project to be heavier and takes up more resources.

In conclusion, all of the frameworks mentioned earlier are suitable for development of the centralized sales system and each of them have their own benefits as well as drawbacks. Regardless, React is still a fitting choice for this project with its diverse range of choices for front-end development.

6 CONCLUSION

This chapter will summarize the findings from the previous chapters. From this, the research's validity and limitations are reviewed in order to propose suggestions for future research.

6.1 Results

As mentioned in the beginning of this thesis, the reason for conducting this research was to answer the following question:

- **In what ways can React benefit the development of a web application?**

The thesis was able to answer this question through both a literature review of documentation and articles regarding the React JavaScript library as well as a detailed case study of a large-scale project developed with React. From the results collected, the thesis presented a variety of advantages React brought to the case study project as a demonstration of the library's capabilities. In terms of application quality, React can help design and build interactive UIs that allows for seamless interactions between the user and the application. React also offers developers many options to cut down development time with various official and 3rd party extensions. In addition to this, the thesis also managed to differentiate React with 2 other popular choices for JavaScript front-end development, which are Vue and Angular, through reviewing additional documents about these 2 frameworks.

6.2 Limitations

A few limitations were present during the research of this thesis. The first is that the thesis manages to cover only some notable aspects of React that was implemented for this project. Due to the high complexity of the artifact used for the research, many other functionalities of the solution could not be demonstrated, hence the thesis could not display more advanced usages of React. The second is that the examples used in this research were taken during the early development process of the project, which made them outdated when compare to the final completed product.

6.3 Validity

Qualitative research was conducted through a combination of scholarly articles, books and journals, and online documentation and blog posts to achieve the purpose of this research, which provided a diverse source of information and detailed analysis on the research topic. The thesis also presented a research artifact through comprehensive examination of the case study that can be used as a reference of React's usages and functionalities. Furthermore, the use of JavaScript frameworks is also a commonly discussed subject among the community and the topic of this research will be relevant for many years to come.

6.4 Suggestions for Future Research

The main focus of this research was on the effects of React on the development of a large-scale IT web solution. A few other topics that can broaden the scope of this research includes:

- The effects of React on smaller scale web applications development.
- The effects of React on mobile application development.
- Comparison of React with other front-end JavaScript frameworks besides Vue and Angular.

7 REFERENCES

Angular. 2020a. GitHub. Retrieved on 22 October 2020. Available at

<https://github.com/angular/components>

Angular. 2020b. GitHub. Retrieved on 30 October 2020. Available at

<https://github.com/angular/angular>

Ant Design. 2020. Retrieved on 3 November 2020. Available at <https://ant.design>

Business Research Methodology. 2020. Research Approach. Retrieved on 6

October 2020. Available at [https://research-methodology.net/research-](https://research-methodology.net/research-methodology/research-approach)

[methodology/research-approach](https://research-methodology.net/research-methodology/research-approach)

Chetty, P. 2016. Importance of Research Approach in a Research. Project Guru.

Retrieved on 9 Nov 2020. Available at [https://www.projectguru.in/selecting-](https://www.projectguru.in/selecting-research-approach-business-studies/)

[research-approach-business-studies/](https://www.projectguru.in/selecting-research-approach-business-studies/)

Cromwell, V. 2016. Evan You. Between the Wires. Retrieved on 21 October 2020.

Available at

<https://web.archive.org/web/20170603052649/https://betweenthewires.org/2016/11>

[/03/evan-you/](https://web.archive.org/web/20170603052649/https://betweenthewires.org/2016/11/03/evan-you/)

Facebook. 2020. GitHub. Retrieved on 8 September 2020. Available at

<https://github.com/facebook/react>

Gavigan, D. 2018. The History of Angular. The Startup Lab. Retrieved on 21

October 2020. Available at [https://medium.com/the-startup-lab-blog/the-history-of-](https://medium.com/the-startup-lab-blog/the-history-of-angular-3e36f7e828c7)

[angular-3e36f7e828c7](https://medium.com/the-startup-lab-blog/the-history-of-angular-3e36f7e828c7)

Hámori, F. 2020. The History of React.js on a Timeline. RisingStack. Retrieved on

18 October 2020. Available at [https://blog.risingstack.com/the-history-of-react-js-](https://blog.risingstack.com/the-history-of-react-js-on-a-timeline)

[on-a-timeline](https://blog.risingstack.com/the-history-of-react-js-on-a-timeline)

Hevner, A.R. & March S.T. & Park J. & Ram S. March 2004. Design Science in

Information Systems Research. MIS Quarterly. Volume 28 Issue 1. Pages 75 –

106. Retrieved on 24 September 2020. Available at

https://www.researchgate.net/publication/201168946_Design_Science_in_Informat

[ion_Systems_Research](https://www.researchgate.net/publication/201168946_Design_Science_in_Informat)

jsComplete. 2020. React.js Beyond the Basics. Retrieved on 4 November 2020. Available at <https://jscomplete.com/learn/react-beyond-basics/introduction>

Levlin, M. 2020. DOM benchmark comparison of the front-end JavaScript frameworks React, Angular, Vue, and Svelte. Åbo Akademi. Master's thesis. Retrieved on 20 October 2020. Available at <http://urn.fi/URN:NBN:fi-fe2020051838212>

Manjunath, M. 2018. AngularJS and Angular 2+: a Detailed Comparison. Sitepoint. Retrieved on 21 October 2020. Available at <https://www.sitepoint.com/angularjs-vs-angular>

Mariano, C.L. 2017. Benchmarking JavaScript Frameworks. Technological University Dublin. Masters dissertation. Retrieved on 20 October 2020. Available at <https://doi.org/10.21427/D72890>

Mozilla. 2020. Introduction. Retrieved on 20 October 2020. Available at <https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Introduction>

Mozilla. 2020. Introduction to Client-Side Frameworks. Retrieved on 20 October 2020. Available at https://developer.mozilla.org/en-US/docs/Learn/Tools_and_testing/Client-side_JavaScript_frameworks/Introduction

Primefaces. 2020. Retrieved on 3 November 2020. Available at <https://www.primefaces.org/primereact>

React. 2020a. Retrieved on 17 October 2020. Available at <https://reactjs.org>

React. 2020b. Rendering Elements. Retrieved on 18 October 2020. Available at <https://reactjs.org/docs/rendering-elements.html>

React. 2020c. Components and Props. Retrieved on 18 October 2020. Available at <https://reactjs.org/docs/components-and-props.html>

React. 2020d. State and Lifecycle. Retrieved on 18 October 2020. Available at <https://reactjs.org/docs/state-and-lifecycle.html>

React. 2020e. Hooks at a Glance. Retrieved on 18 October 2020. Available at <https://reactjs.org/docs/hooks-overview.html>

Singh, H & Tanna, M. 2018. Serverless Web Applications with React and Firebase. Packt Publishing. Retrieved on 29 October 2020. Available at <https://ebookcentral-proquest-com.ezproxy.saimia.fi/lib/lab-ebooks/detail.action?pq-origsite=primo&docID=5345875>

Stack Overflow. 2019. Developer Survey Results 2019. Retrieved on 6 September 2020. Available at <https://insights.stackoverflow.com/survey/2019#most-loved-dreaded-and-wanted>

Vue. 2020a. GitHub. Retrieved on 21 October 2020. Available at <https://github.com/vuejs/vue>

Vue. 2020b. Template Syntax. Retrieved on 29 October 2020. Available at <https://v3.vuejs.org/guide/template-syntax.html>

Vue. 2020c. Introduction. Retrieved on 21 October 2020. Available at <https://v3.vuejs.org/guide/introduction.html>

Wilson, E. 2018. MERN Quick Start Guide: Build Web Applications with MongoDB, Express.js, React, and Node. Packt Publishing. Retrieved on 20 October 2020. Available at <https://ebookcentral-proquest-com.ezproxy.saimia.fi/lib/lab-ebooks/detail.action?pq-origsite=primo&docID=5405683>