

# DEVELOPING MOBILE APPLICATION WITH VUE.JS FRAMEWORK

Case: Osmi application and SuperApp Oy

**LAHTI UNIVERSITY OF APPLIED  
SCIENCES LTD**  
**Bachelor of Business Administration**  
**Degree Programme in Business**  
**Information Technology**  
**Autumn 2019**  
**Hoang An Pham**

## Abstract

Author Pham, Hoang An	Type of publication Bachelor's thesis	Published Autumn 2019
	Number of pages 54	
Title of publication Developing mobile application with Vue.js framework Case: Osmi application and SuperApp Oy		
Name of Degree Bachelor's Thesis in Business Information Technology		
Abstract <p>Over the past few years there has been a rising number of mobile development frameworks and Vue.js is one of the most popular choices among those.</p> <p>This thesis discusses and examine how Vue.js framework could benefit tech companies in developing mobile application and how choosing Vue.js can enhance the performance of different development roles in the same project. The principal purpose of this study is to provide knowledge about Vue.js' advantages especially in emphasizing individual's strength while working on a project simultaneously.</p> <p>The theoretical sections of this thesis present background theory of different tasks commonly needed in mobile development process as well as knowledge about Vue.js along with some other frameworks such as React and Angular. The thesis also demonstrates a case study of the same application created by two developer teams. The goal of this case study was to illustrate the benefits of Vue.js in separating workload. For this, the author adopted qualitative research method and used inductive approach to answer research questions.</p> <p>By applying Vue.js framework, the author was able to identify the benefits gained in time productivity and efficiency in development collaboration. The thesis also suggests that Vue.js framework can optimize the benefits of website programming languages and make the coding process more flexible.</p>		
Keywords: JavaScript frameworks, Vue.js, front-end development, back-end development, Osmi		

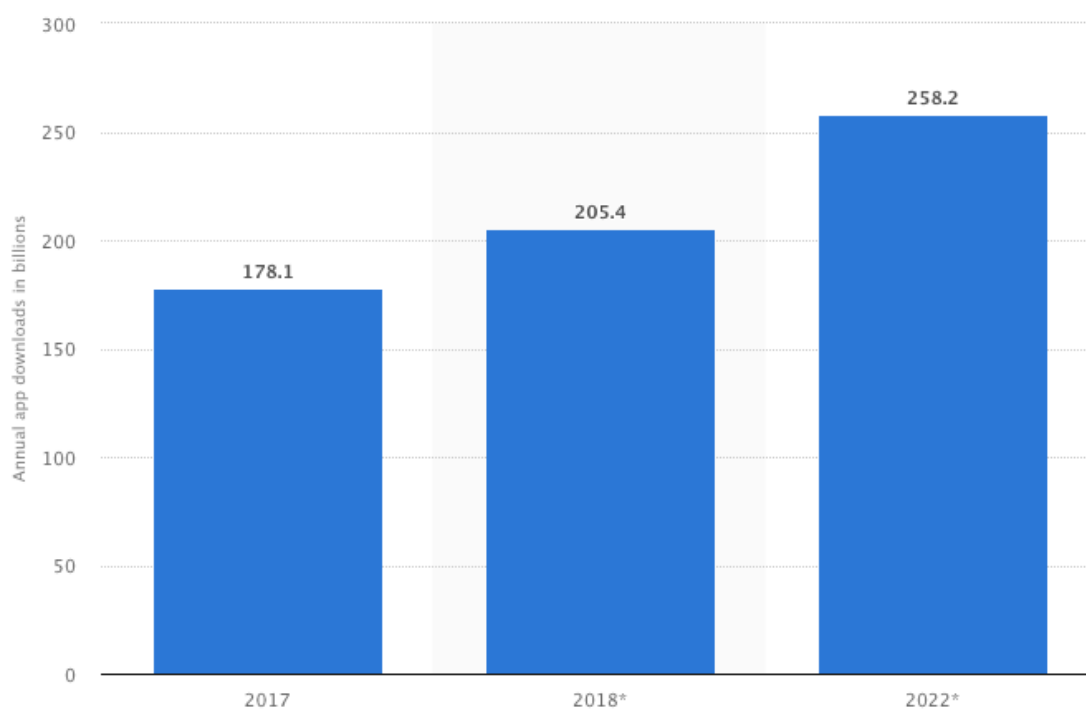
## CONTENTS

1	INTRODUCTION .....	1
1.1	Background .....	2
1.2	Thesis motivation.....	3
1.3	Thesis structure .....	3
2	RESEARCH DESIGN .....	6
2.1	Thesis objectives and research questions .....	6
2.2	Research approach .....	6
2.3	Research methods.....	7
2.4	Data collection and data analysis.....	8
3	THEORETICAL FRAMEWORK .....	10
3.1	Different roles in mobile application development .....	10
3.1.1	User interface designer.....	10
3.1.2	Front-end developer .....	11
3.1.3	Back-end developer.....	12
3.2	Mobile application frameworks.....	14
3.2.1	Vue.js .....	14
3.2.2	Other frameworks .....	17
3.2.3	Pros and cons of Vue.js compare to other frameworks .....	18
3.2.4	How Vue.js enhance collaboration between different roles .....	22
4	CASE DESCRIPTION.....	25
4.1	Introduction of Osmi application.....	25
4.2	Project goal.....	25
4.3	Two developer teams .....	28
4.4	Case study plan.....	29
5	COLLECTION OF DATA .....	32
5.1	Data collecting process.....	32
5.1.1	Collecting data time consumption .....	32
5.1.2	Collecting data on finalizing phases, difficulties and other supporting data .....	35
5.2	Collected data results .....	36
5.2.1	Data on development time .....	36
5.2.2	Data on finalizing phases and production quality .....	37
5.2.3	Data on difficulty during development process.....	38
5.2.4	Data on difficult levels of cooperation using Vue.js .....	39
6	STUDY AND ANALYSIS.....	41

6.1	Comparing development time of two developer team .....	41
6.2	Comparing quality of production .....	43
6.3	Comparing the difficulty levels of tasks .....	43
6.4	Analysis on convenience level of separating workload with Vue.js.....	45
7	CONCLUSION.....	46
7.1	Answering research questions.....	46
7.2	Limitations .....	47
7.3	Reliability and Validation.....	47
7.4	Suggestions for further study .....	47
8	SUMMARY .....	48
9	LIST OF REFERENCES.....	49
	APPENDICES.....	53

## 1 INTRODUCTION

The past decade has witnessed an inexorable growth of mobile applications with various categories focusing on all aspects of our life: business, entertainment, shopping social media, education, health & fitness...The figure below indicates the annual app downloads in billions in 2017 and 2018 as well as the procrastination data in 2022.



Data visualized by  + a b | e a u

© Statista 2019

Figure 1 Number of annual app downloads from 2017 to 2022 (Blair 2019)

Total app downloads in 2018 is more than 205 billion and expected to surge to 258.2 billion in 2022. These significant numbers along with 15% annual increase illustrate how rapidly the demand for mobile application expands.

## 1.1 Background

Since the number of mobile application users is growing day by day, a company would prefer to take advantage of this by having their mobile app alongside a website. Nowadays there are a lot of enterprises and businesses focusing on launching mobile apps for their customers. Figure 2 shows the worldwide mobile app revenues in billion US dollars.

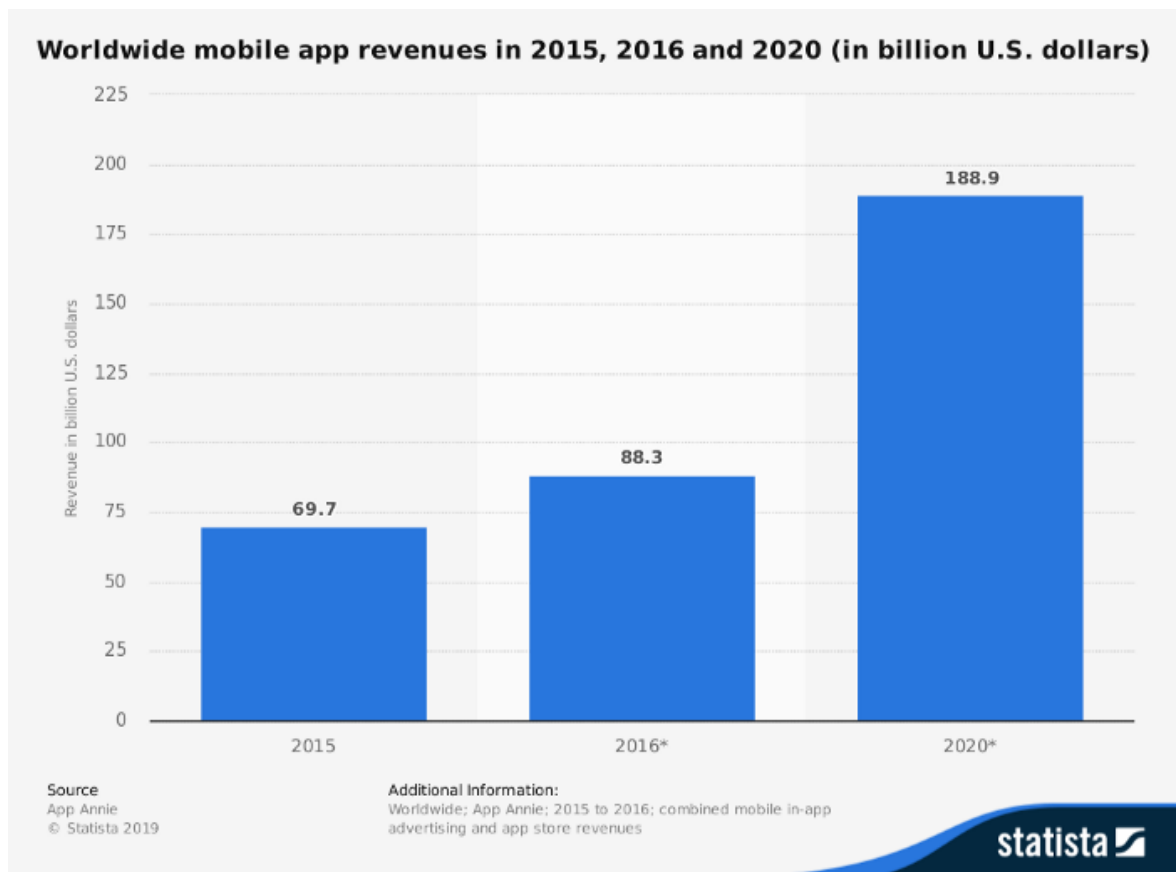


Figure 2 Global mobile app revenues in 2015, 2016 and 2020 (Martin 2019)

As displayed in the figure above, worldwide mobile application revenue are estimated to produce an income of \$188.9 billion via app stores and in-app advertising in 2020. This trend open up a huge opportunity for IT companies to yield profits in a promising industry: mobile application development.

According to Margaret Rouse (2019), a manager of TechTarget's IT encyclopedia and learning center, mobile application development is the set of procedures involved in creating software that run on small, wireless computing devices. One fundamental distinguishing factor of mobile apps when compared to other software development, for example web

development, is that it is often written precisely to optimize of the unique features available on particular mobile device. For instance, a business app can be written to take advantage of mobile calendar's adding event feature to remind important dates. Moreover, a network and Internet connection is usually utilized in an application to communicate with remote computing resources. Hence, the mobile development process includes generating installable software bundles (code, binaries, assets, etc.), employing backend services such as data handling with APIs, and testing the application on several target devices (AWS 2019)

## 1.2 Thesis motivation

Nowadays Vue.js, React and Angular are three most popular mobile framework based on GitHub frontend-end frameworks usage statistics which replicates developers' tendency on each frameworks and libraries (Xing, Huang, Lai 2019, 3). When creating apps, all tech companies must start with choosing a base framework and library. Because there are usually different developers working on the same project at the same time, reducing difficulty in coding, maximizing flexibility in programming and enhancing closer collaboration between programmers are the most fundamental priorities (Nikula 2019). Among those framework mention above, Vue.js is not only one of the fastest growing Javascript framework (Kaluža, Troškot, Vukelić 2018) but also has the capability of fulfilling that requirement. Developing mobile apps with Vue.js can produce time efficiency, diminish training time and resources as well as promote productivity by teamwork.

In the book "State of Vue.js 2019" (Monterail 2019, 9-10), a survey conducted over a five-week period in November and December of 2018 with 1,553 responses indicated that "92% respondents would use Vue.js again" and "more than a half of the respondents describes Vue.js as easy to start with". In this thesis, the author will analyze and examine how Vue.js framework can bring advantages to IT company.

## 1.3 Thesis structure

The author draws the diagram below to describe and outline main sections of this thesis.

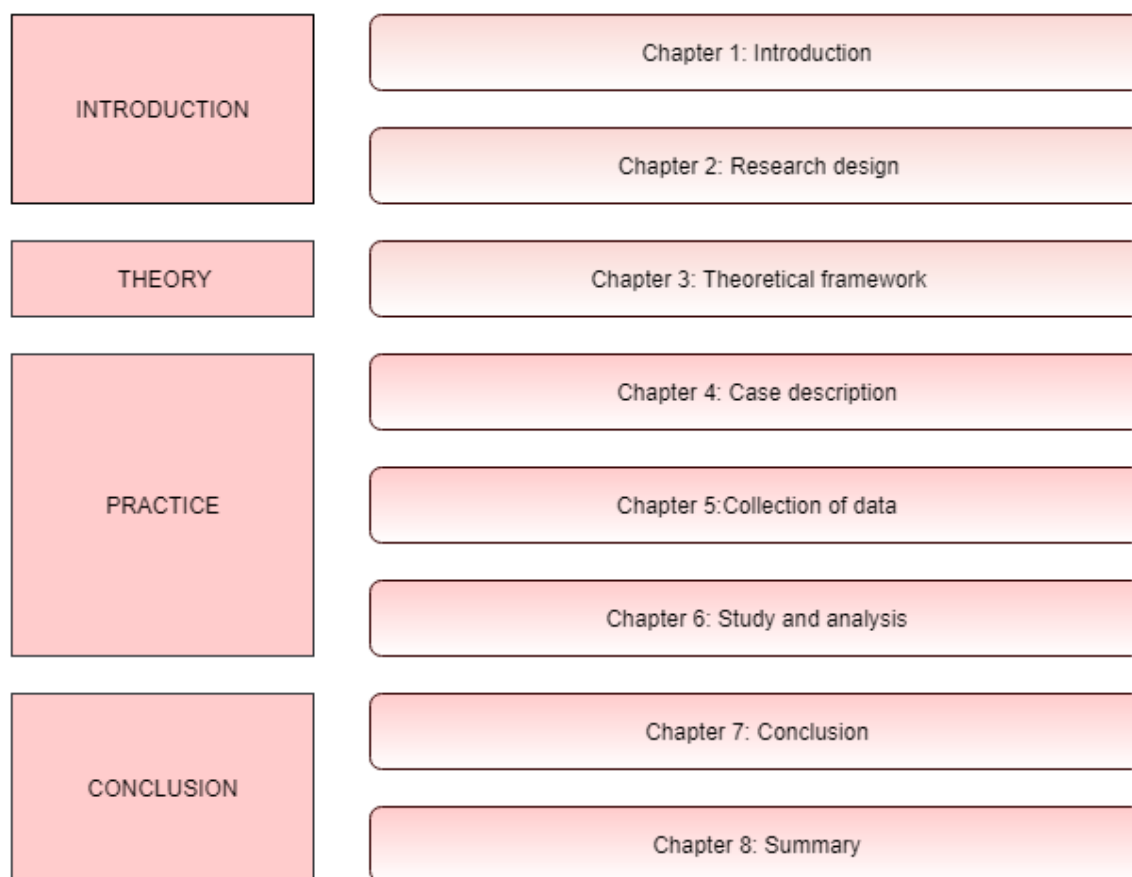


Figure 3 Thesis structure

As displayed in the above diagram, the thesis consists of 4 main parts divided into eight chapters. The first section is “Introduction”, in which Chapter 1 introduces about the background situation and motivation why this study is composed. In chapter 2, the author demonstrates how it is experimented through research questions, research methods and research approach. Next, “Theory” section provides reader with a theoretical background of the study. Particularly, technical knowledge about related mobile development frameworks (Vue.js, React and Angular) as well as different roles in mobile application development (UI designer, Front-end developer, Back-end developer) is discussed in terms of definition and concept. These information is eventually form the theory of how choosing Vue.js is a deliberate decision outstanding the others when it comes to enforcing collaboration between different roles. The third part, “Practice”, presents 3 chapter: “Case description”, “Collection of data” and “Study and analysis”. To begin with, Chapter 4 introduces the application used as case study, two developer teams, development phases and schedule along with study plan of how it is implemented. After that, Chapter 5 describes



the collected data results regarding development time, app performance, number of obstacles in tasks and difficulty level of team 1 which have two developer collaborating by Vue.js. These information is analyzed in Chapter 6 to prove that team 1 work smoothly with Vue.js and has better productivity in development time and production quality. Finally, “Conclusion” section consisted of Chapter 7 and 8 states the conclusion and summary respectively. This final section answers research questions as well as presenting limitations, reliability and validation of this research and suggestions for further study.

## 2 RESEARCH DESIGN

### 2.1 Thesis objectives and research questions

The thesis aims to determine and testify the advantages of utilizing Vue.js framework in mobile application development. Coupled with the theoretical background of Vue.js's different characteristics compared to other JavaScript frameworks, a case study of two developer teams is also presented to illustrate the benefits of it in dividing workload and encouraging individual's productivity. In other words, the author will research and examine the capability of Vue.js in allowing developers with different strength and preferences to cooperate in a same project simultaneously thanks to its framework architecture.

Consequently, this thesis is targeted to answer two main research questions:

- How the advantages of Vue.js can be optimized in developing mobile application in general?
- How choosing Vue.js can benefit IT company by enhancing the efficiency of different development roles in the same project?

### 2.2 Research approach

There are three most common research approaches: abductive, deductive and inductive. With deductive reasoning, researchers apply a theory into a set of data and progress to the application of the theory. This way they can examine that theory or make predictions. With inductive approach, in contrast, a theory is generalized by presenting specific observations, researches or case study. (Prince, Felder, 2006, 123.) Creswell and Plano Clark (2007, 23) defined the elementary difference between these two approaches is that deductive researcher "works from the 'top down', from a theory to hypotheses to data to add to or contradict the theory" while inductive researcher goes from the "bottom-up, using the participants' views to build broader themes and generate a theory interconnecting the themes". Figure 4 describes the distinction of these two research approaches.

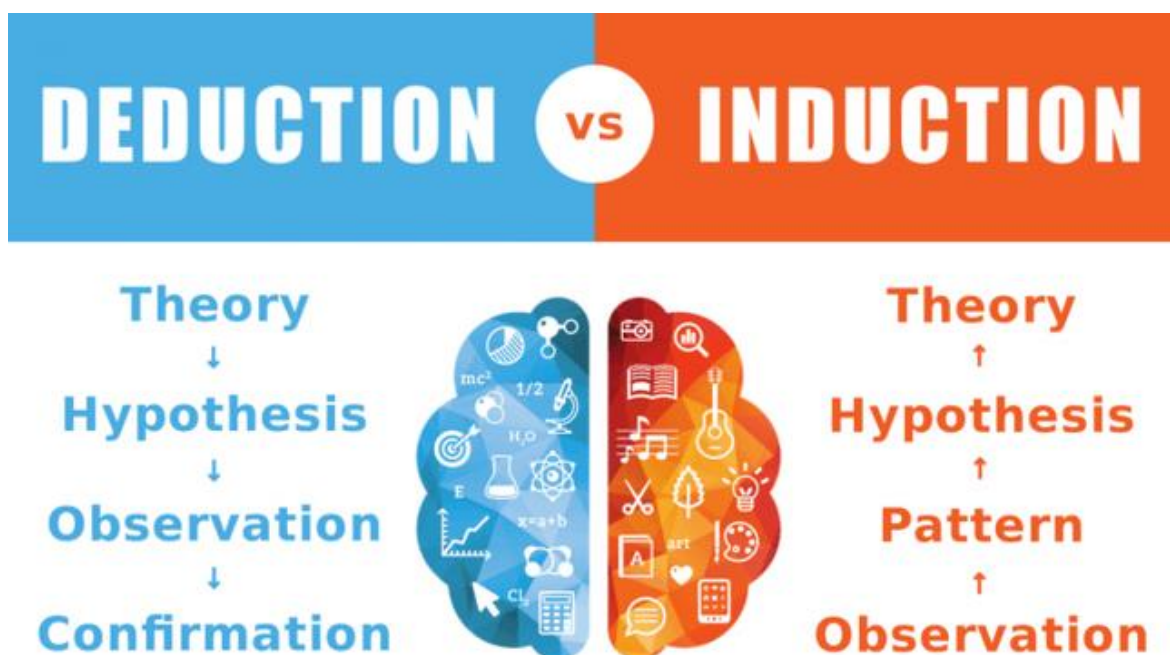


Figure 4 The difference between deduction and induction research approach

This paper studies an already existing solution for the demands and requirements stated in “Thesis motivation” section then applies it to the case study to compare two development processes. In addition, the data and observation collected from the case study is analyzed to testify and guarantee the outstanding advantages of Vue.js regarding cooperate programming. For those reasons, deductive approach will be employed in this research.

### 2.3 Research methods

Design science research is usually applied in technical and procedural aspects such as engineering, architect or information technology. This method focuses on the creation serving one or several particular purposes or in other words: an artifact. (Simon 1996). Hevner, March, Park and Ram (2004) state that design science research should investigate an unsolved problems in an innovative way or suggest a new solution to raise effectiveness.

Quantitative and qualitative are two types of design science research method. According to Creswell (2003, 18), a quantitative approach proves a theory by the numeric information, which is easy to handle in large quantities such as “experiments and surveys, and collects data on predetermined instruments that yield statistical data”. Alternatively, he believed qualitative method as an approach in which the researcher use “the multiple meanings of individuals experiences” such as “grounded theory studies, or case studies” which

allow readers to understand better where the numbers come from. To put it another way, qualitative research method implicates one-to-one interview, research on focus groups, ethnographic research, case study research, record keeping and process of observation. In some cases, a mixed methodology can be utilized which gathers both numeric information and specific findings. The differences between Qualitative Research and Quantitative Research are summarized in the table below.

Table 1 Comparison between Qualitative Research and Quantitative Research (Fernandez, 2019)

## QUALITATIVE RESEARCH VS QUANTITATIVE RESEARCH

	Qualitative Research	Quantitative Research
Objective / Purpose	To gain an understanding of underlying reasons and motivations  To uncover prevalent trends in thought and opinion	To quantify data and generalize results from a sample to the population of interest  Sometimes followed by qualitative research which is used to explore some findings further
Sample	Usually a small number of non-representative cases	Usually a large number of cases representing the population of interest
Data analysis	Non - statistical	Statistical data is usually in the form of tabulations (tabs). Findings are conclusive and usually descriptive in nature
Example	Focus Groups, individual depth interviews , group discussions	Survey, Simulations,

Applying case study and personal interview, the author adopts design science method with qualitative approach.

### 2.4 Data collection and data analysis

Hevner (2004, 17-18) summarized 5 evaluation methodologies in design sciences:

- Observational: Using case study (to research about artifact in specific case) or field study (to perceive use of artifact in manifold projects)
- Analytical: Applying static analysis, architecture analysis, optimization or dynamic analysis.
- Experimental: Utilizing controlled experiment or simulator.
- Testing: Executing functional (black box) testing or structural (white box) testing.

- Descriptive: Implementing informed argument or scenarios.

Because this thesis presents a case study of two developer team to examine the performance of Vue.js and the author herself was one of the developers, participant observational evaluation method will be applied. This method involves “active looking, improving memory, informal interviewing, writing detailed field notes” (DeWALT & DeWALT 2002, p.vii).

The case study collected data from 2 developer teams utilized Vue.js with a same project in order to emphasize the effectiveness of Vue.js in maximizing different role’s performance while working on a project simultaneously. Team 1 had one senior developer who was good at back-end and one junior developer who was good at front-end (the author) while Team 2 was one senior developer who could do both front-end and back-end. The statistics were gathered for the comparative purpose, concerning development time and number of obstacles in each phases, number of bugs after the apps have finished, difficult point of tasks and app quality in general. The author recorded data by an organization management tool called Trello, time recording systems and developer’s personal notes and reports through the whole project. All data and analysis will be presented and illustrated in Chapter 5 and 6 of this thesis.

### 3 THEORETICAL FRAMEWORK

#### 3.1 Different roles in mobile application development

A core development team for building a fully implemented mobile application typically includes the following roles: UI Designer, Front-end Developer and Back-end developer. In many cases, one developer can be in charge of both front-end and back-end which brands him a Full-stack Developer. The author performed a Front-end Developer role in the case study of this thesis.

##### 3.1.1 User interface designer

“A mobile user interface (mobile UI) is the graphical and usually touch-sensitive display on a mobile device, such as a smartphone or tablet, that allows the user to interact with the device’s apps, features, content and functions” (Rouse 2015). Figure 5 introduces 4 main stages of designing an application.

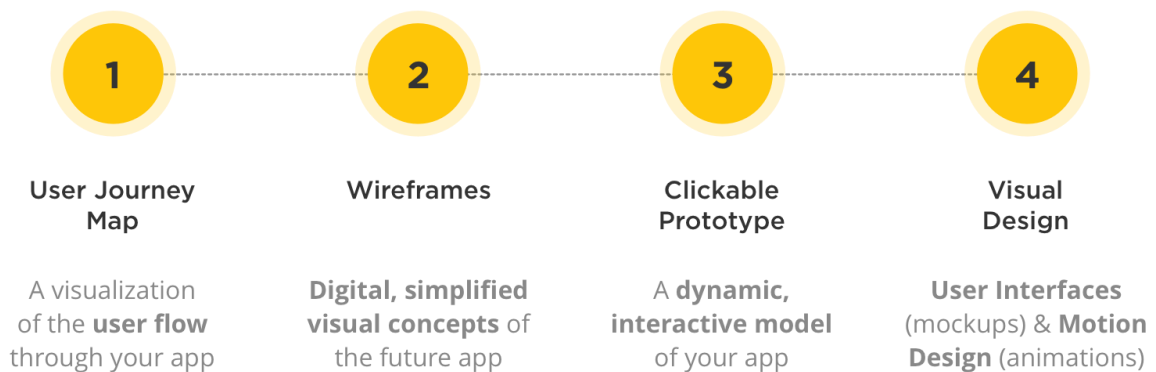


Figure 5 App design process (Mroczkowska 2018)

User interface (UI) designer is the one tailoring user experience by the understanding of user’s goals, desires, common preferences and behaviors pattern. Designer’s responsibility is to compose graphical user interfaces that is logical in function, visually beautiful and convenient for user. Usually the working instrument of UI designer is Adobe Illustrator which assists them in drawing sketches as well as creating all view’s layouts and app prototype. (Rybachuk 2016.) This role is required to have expertise in graphic editor tools such as Adobe Photoshop, Adobe Illustrator, MockFlow, Elementor... In some cases, UI designer also has some basic skills in HTML and CSS.

### 3.1.2 Front-end developer

The front-end of an application is everything that user sees and interact with (Pluralsight 2015). Also referred to as the client-side of the application, it is in essence all the attributes related to visual aspect: text, images, sliders, buttons, selectors, navigation menus, pages, input fields of all types... (Ferguson 2018). To this point, a front-end developer is sound very similar to UI designer. Nevertheless, he is the person who actually bringing a designer's concept to life through programming. The figure below demonstrates distinguishes between UI designer and front-end developer.



Figure 6 The characteristics of UI designer and front-end developer (Rybachuk 2016)

Front-end developer is the person that responsible for constructing the front-end base of an application or in other words, forming the basis of what users can touch and experience on their devices. The main purposes of this role is to guarantee the application is accessible to all type of users and deliver a responsive app in all views and devices. In other words, front-end developer generates dynamic conversions to the appearance and layout of the app depending on various screen sizes of the devices (Schade 2014).

In his book “Front-end Developer Handbook 2019”, Cody Lindley (2019, 16-17) summarized that the most common front-end job titles can be mentioned are: Front-End Developer, Front-End Engineer (aka JavaScript Developer or Full-stack JavaScript Developer), CSS/HTML Developer, UI (User Interface) Developer/Engineer and Mobile/Tablet Front-End Developer. There are several bare-bones and fundamental technologies employed by front-end developers:

- Hyper Text Markup Language (aka HTML)
- Cascading Style Sheets (aka CSS)
- Uniform Resource Locators (aka URLs)
- Hypertext Transfer Protocol (aka HTTP)
- JavaScript Programming Language (aka ECMAScript 262)
- JavaScript Object Notation (aka JSON)
- Document Object Model (aka DOM)
- Web APIs (aka HTML5 and friends or Browser APIs)
- Web Content Accessibility Guidelines (aka WCAG) & Accessible Rich Internet Applications (aka ARIA)

(Lindley 2019, 19.)

### 3.1.3 Back-end developer

Pluralsight, an online programming education company, referred back-end as the “server-side” which includes everything not directly displayed in app for user to see, for examples: databases and servers. The figure below indicates how the front-end part of a site (or a mobile application) connects and retrieves data from the back-end.



## FRONT-END DEVELOPMENT

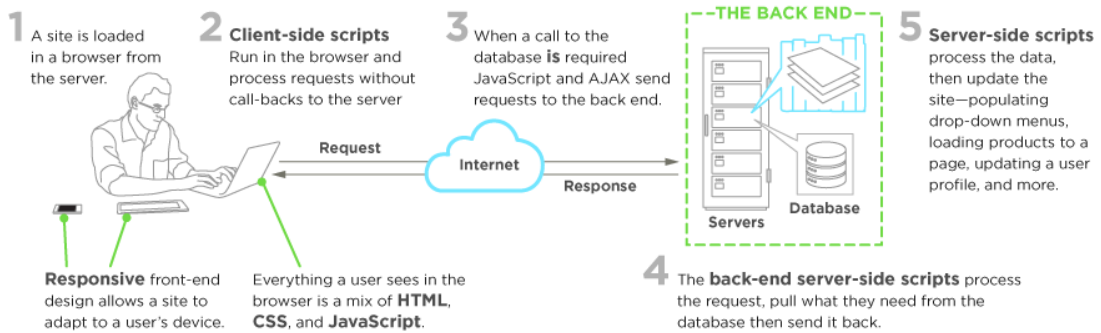
Upwork<sup>®</sup>

Figure 7 The connection between Front-end and Back-end development (Wodehouse 2019)

As can be observed from Figure 7, the client-side scripts receive user's interaction and send request to server to obtain and gather information from database which will be handled to response the action. These actions is delivered and transferred through Internet connection.

Back-end developer is responsible for various fully implemented functionalities as well as all the application's logic which runs across devices. To be more specific, the focused tasks of this role are managing APIs resources, storing data to displayed on the application, constructing and establishing algorithms in order to solve system related requirements as well as implementing additional services such as payment and purchase, in-app advertisement, chat function, taking pictures, adding events to device's calendar, QR code scanning... To accomplish these duties, essential skill set of a back-end developer contains server-side programming languages (PHP, Java, Python, Ruby, .Net...), database design and implementation, knowledge about server and API. (Guru99 2019)

From the characteristics and workload of the two roles declared above, the author interprets that front-end developer and back-end developer have a closely association yet still are two different roles in nature. The table below summarizes their discrepancies.

Table 2 Differences between Front-end and Back-end development

Front-End	Back-End
Refers to client-side of the application, user can observe	Refers to server-side of the application, user cannot observe
Determines how the elements look	Determines how the elements works
Receives user's interaction and actions	Handles user's interaction and actions
Displays data on screen	Loads data from app database
Usually uses HTML, CSS and JavaScript...	Usually uses PHP, Ruby, Python, Java, .Net...

### 3.2 Mobile application frameworks

This section introduces three most popular JavaScript frameworks for mobile development: Vue.js, React and Angular. The analysis on Vue.js's pros and cons compared to the others is also illustrated to explain why Vue.js is an adequate choice for this specific research objective.

#### 3.2.1 Vue.js

##### Background information

Vue.js is introduced by Evan You, who had worked in various AngularJS projects while working at Google. Evan You shared his story with Vue on GitHub: "I started Vue as a personal project when I was working at Google Creative Labs in 2013". After the experience of building UI prototypes with Vanilla JavaScript and Angular 1, he flourished an idea of capturing the nature of Angular in something simpler, more lightweight and approachable (GitHub 2019). Therefore, the first version of Vue.js 1.0.0 was released in October 2015 (Wohlgethan 2018, 41).

On Vue.js's official website, the organization define their own product as "a progressive framework for building user interfaces". As a consequence, its utility is diverse and manifold. For instance, if there is already an existing server-side application, the developer can plug Vue into just one part of the application that needs a richer and more interactive experience. Likewise, if a company wants to build more business logic from the baseline,

Vue.js has the core libraries and the ecosystems which includes Vuex and Vue-Router for developer to scale and utilize. (Vue.js 2019a.)

### Framework structure

Vue.js is a component based framework. Reusable components can be compiled as a small full functional piece of code and embed into different pages of the application. The basic structure of a Vue component is presented below.

```
1 <template>
2 |   <h1 class="heading">Hello World!</h1>
3 </template>
4
5 <script>
6 | // Import libraries here
7
8   export default {
9 |     computed: {
10
11 |     },
12     props: {
13
14 |     },
15     data() {
16 |       return {}
17 |     },
18     mounted()
19     {
20 |       // Runs whenever the component is loaded
21 |     },
22     methods: {
23 |       // Write app's functionalities here
24 |     }
25 |   }
26 </script>
27
28 <style lang="scss">
29 | .heading {
30 |   color: brown;
31 | }
32 </style>
```

Figure 8 Structure of Vue component

As can be seen in Figure 8, each Vue.js component contains 3 separate segment: <template> (HTML), <script> (JavaScript) and <style> (CSS). Vue uses an HTML-based template syntax that enables developer to bind the rendered DOM (Document Object Model) to the declared Vue instance's data. These templates are plain HTML that can be parsed and understood by all browsers, operation systems and HTML parsers. (Template Syntax 2019.) All the back-end functions including API calls, event handling, database communications, server-side methods... are handled in the <script> section. Last but not least, in <style> segment, the styling of component is generated with CSS or any preprocessor which allows developer to use features that aren't a part of the wider CSS standard.

There are several core features of Vue.js framework to support the connection between `<template>` and `<script>` while still performing as separate sections:

- **Declarative Rendering:** Mustache syntax empowers us to declaratively render data to the DOM. The figure below illustrates how the data “product” is interpreted in current component’s template.

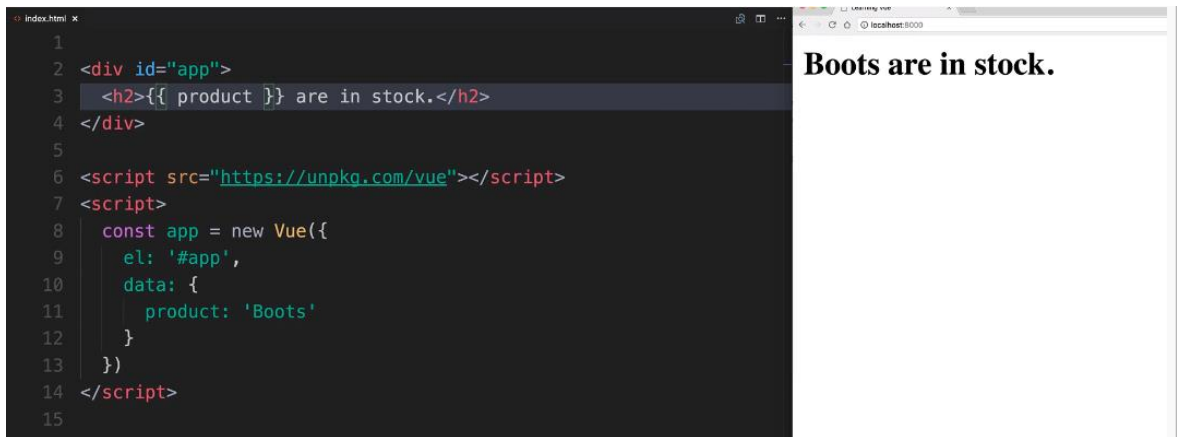


Figure 9 The use of declarative rendering in template syntax of Vue.js (Template Syntax 2019)

- **Directives:** A directive is “some special token in the markup that tells the library to do something to a DOM element”, starting with the prefix “v-” (Directives 2019). Most popular and powerful Vue directives can be mentioned are “v-model” for binding data which will be explained more precisely in next feature, “v-on” to handle event, “v-for” to run through a loop, “v-if” and “v-else” to render conditions...
- **Data binding:** Vue enables us to bind any data received in `<script>` through backend functions to the data displayed in `<template>`. Figure 10 shows how the value of user’s input field can be connected to the “message” data.

```

1  <template>
2  <div class="single-input">
3    <label>Message</label>
4    <input
5      type="text"
6      v-model="message"
7    >
8  </div>
9  </template>
10
11 <script>
12
13 export default {
14   data() {
15     return {
16       message: ""
17     };
18   },
19 }
20 </script>

```

Figure 10 Code example of Vue.js

### 3.2.2 Other frameworks

#### Angular

Wohlgethan (2018, 14) stated that “Angular was originally created by Google employees Misko Hevery and Adam Abrons in 2008”. However, it was not until 2012 that Google officially announced the first version of Angular which back then is called AngularJS (Aldwin 2019).

In summer 2014, Angular 2 was released, introducing a brand-new transformation in framework’s core concepts. Angular 2 is in essence formulated and constructed completely on a hierarchy of components as many of other JavaScript frameworks. This version is associated with another innovative novelty: TypeScript. (Wohlgethan 2018, 15.) This is the ES6 version of JavaScript added several customized features for Angular to work (Asim 2019).

As presented in the last section, Evan You – the creator of Vue.js is inspired by Angular. Therefore, Angular shares a lots of similarities to Vue.js when it comes to framework structure and features including routing, component interaction and aforementioned capabilities of declarative rendering and two-way data binding. (Wohlgethan 2018, 17-21.)

## React

Jordan Walke, a software engineer of Facebook, created React in 2011 and utilized this library internally for Facebook's newsfeed as the first use case. Two year later, having realized about React's potentials, Facebook decided to make it open source in May 2013 at JSConf US. (React.js history 2019.)

The core of React is also constructed by components. However, unlike Vue.js and Angular, React's component is not written in template principle but with two approaches instead: "Components as Functions" which return one `ReactDOMElement` or "Components as ES6 Classes" returning one root element in a "`render()`" method (Wohlgethan 2018, 30-31). Furthermore, React does not apply a specific template language to build component's layout but an extension called JSX is used instead. This is a syntax extension which enriched HTML with the full power of JavaScript so that the logical handling can be operated inside UI tags. (React 2019.)The figure below demonstrate how JSX is used to build an `<h1>` element.

```
function formatName(user) {
  return user.firstName + ' ' + user.lastName;
}

const user = {
  firstName: 'Harper',
  lastName: 'Perez'
};

const element = (
  <h1>
    Hello, {formatName(user)}!
  </h1>
);

ReactDOM.render(
  element,
  document.getElementById('root')
);
```

Figure 11 Code example of JSX (Introducing JSX 2019)

### 3.2.3 Pros and cons of Vue.js compare to other frameworks

#### Advantages

Firstly, easy learning curve is an outstanding advantage of Vue.js. According to The State of JavaScript Survey (2018) participated by 20,268 developers in 153 different countries,

this privilege was the most valued aspect among 13 categories of Vue’s strength. This was rooted in the fact that Vue.js empowers HTML which make it much more favorable for designers, beginners and experienced developers who are familiar with basic web technologies. On the other hand, Angular has TypeScript-based structure so the HTML template is full of additional syntax and the larger component is, the higher complexity the code gets. The figure below shows an example of input and submit button elements in Angular.

```

1  <input type="text"
2      class="input-search"
3      placeholder="Last Name"
4      [(ngModel)]="name">
5  [...]
6  <button type="submit"
7      class="clear-input style-button-shadow"
8      (click)="clearForm()"
9      *ngIf="isFormValid">
10 </button>

```

Figure 12 Simple form elements (Wohlgethan 2018)

In the example above there are several special syntaxes from Angular such as “*[(ngModel)]*” for two-way data-binding, “*\*ngIf*” for controlling conditional value, “*(click)*” for event handler. Consequently, learning all these exclusive syntaxes without any formula (Vue.js has the prefix “v-” for all syntaxes) is compulsory if you want to use Angular. (Wohlgethan 2018, 24-25.) The learning curve is even steeper when it comes to React as JSX is a bone of contention. JSX intermixes the JavaScript logic and UI styling related part of the application making it complicated and difficult for designer or front-end developer to modify app layout. Figure 13 illustrates how React binds the “message” data which undoubtedly is more obscure than the code example in Figure 10 showing Vue doing the same function.

```

// React.js (jsx)

var Message = React.createClass({
  getInitialState() {
    return {
      message: ''
    }
  },

  handleMessageChange(e) {
    this.setState({message: e.target.value});
  },

  render() {
    return (
      <div>
        <input type="text" onChange={this.handleMessageChange} />
        <span>{this.state.message}</span>
      </div>
    )
  }
});

```

Figure 13 Code example of React (Malhotra 2019)

Another benefit of Vue.js compared to the other frameworks is high flexibility. Vue.js is in essence an approachable and versatile framework that helps developer to create maintainable and testable code base (Vue.js 2019a). With the principal target to emphasize progressive factor, Vue provides developer with plenty of internal features: Vuex for data management (state management) which will be described more precisely in the next section of this thesis, Vue Router for in-app link (URL) management, Vue Server-Side Renderer for server-side rendering. All these core modules are built-in and developed by Vue.js team. On the contrary, React utilizes Redux for state management, but more official advanced features for organizing data managing router is not offered. React uses a lot of third party package which makes the official framework itself moderately limited in implementation. (Ahuvia 2018.) Angular shares the same concern of restraint.

*As Angular is a complete framework it provides a full set of homogeneous APIs. It takes time to gain an overview of all possibilities that are offered. The framework predetermines many decisions for the developer on how to handle certain situations.*

(Wohlgethan 2018, 26)

Last but not least, adaptability is also a considerable advantage of Vue.js. In term of programming skill, the switching period from other JavaScript framework to Vue is fairly short. This is because of the resemblance and similarity with React and Angular regarding structural design and architecture. (TechMagic 2018.) Moreover, Vue also has high adaptability in integration as developer is able to integrate Vue into existing project. In other words,



Vue proposes the capability of customizing the application according to developer's preferences and requirements. The book "State of Vue.js 2019" (2019, 12) reported that "Ease of integration" is the biggest advantage of Vue with 76% votes.

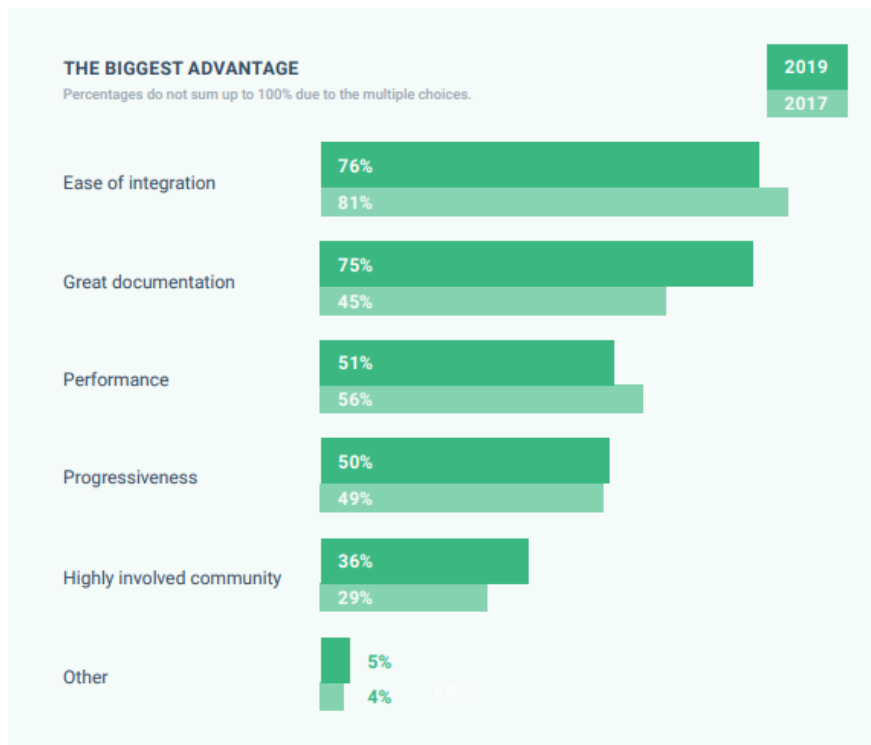


Figure 14 Statistic of responses for the biggest advantage of Vue.js (Monterail 2018, 12)

## Disadvantages

Despite of all the benefits that Vue.js may bring, it also has several downsides compared to other frameworks. A relatively small developer community is one aspect to consider when IT companies deliberate whether to utilize Vue or not. Speaking about popularity, at the time this paper is written, React is the leading competitor with a significant number of 69,831 public repositories on GitHub. Vue is far behind with only 20,662 projects related to this framework, followed by Angular which has 19,314 repositories. (GitHub 2019.) The small size of Vue community narrows the available resources (plugins, add-on, ready-made scripts...) and possibility of supports when needed. This means that if a developer experiences an obstacle during his project, it may take time to seek an adequate solution. Furthermore, in some cases over-flexibility can be one of Vue.js's disadvantage. In large-scale projects with many developers involved, the flexibility of Vue can be exploited in an

excessive and inappropriate way since there are various ways of programming. This generates inconsistency in the code base as well as increasing complexity and raising project's budget. To avoid this, a mutual guidance with comprehensible instructions about coding style should be composed and shared among developers in the same organization.

In summary, all the basic characteristics and aspects of comparison of Vue.js, Angular and React are outlined in the table below.

Table 3 Summary of comparisons between Vue.js, React and Angular

	<b>Vue.js</b>	<b>React</b>	<b>Angular</b>
<b>Architecture</b>	Component based	Component based	Component based
<b>Template</b>	HTML + JavaScript	JSX + JavaScript	HTML + TypeScript
<b>Learning curve</b>	Easy	Moderate	Steep
<b>Flexibility</b>	High (Can be over-flexible)	Low	Low
<b>Adaptability</b>	High	Moderate	Moderate
<b>Community</b>	Small but growing fast	Large	Medium

### 3.2.4 How Vue.js enhance collaboration between different roles

As explained in the section 3.1.1, Vue has the component-based architecture which allows developer to partition the application into reusable and self-contained components. This can transform a complex project into small and reasonable pieces with a shared workload for several people in the development team. Moreover, each Vue template and Vue component has its own HTML, JavaScript and style sections needed to render that subordinate segment of the app so these parts can be easily developed independently. By this mean, Vue allows not only two but three roles of developer to work on the same project simultaneously:

- An UI designer with a basic coding knowledge of HTML, CSS can make the layouts and styles of different app views.

- A junior developer can also do the front-end of application, add some complex handler for template (for example conditional handling), write some simple functions in <script> section as well as connect front-end to back-end end points.
- A senior developer can do the back-end, for example, PHP and Vuex functions to manage data and handle user requests.

In this way, different roles of developer are able to concentrate on what they are confident with and perform in the aspect of their expertise. (Nikula 2019.)

In order to support the connection between these roles, Vue.js offers Vuex, a state management system and library for Vue.js applications. Vuex assists as a centralized and general store for sharing information and data between components. (What is Vuex? 2019.) The figure below presents the basic structure of Vuex.

```
const store = new Vuex.Store({
  state: {
    ...
  },
  mutations: {
    ...
  },
  actions: {
    ...
  }
})
```

Figure 15 Basic structure of Vuex

“State” represents the actual data stored and can be accessed in all components at any points. The state tree is modified by mutations. However, mutations are synchronous and directly change the store so “actions” are needed to employ more sophisticated demands such as API calls and user interaction. Actions are asynchronous and modify the state object by committing mutations. (Ball 2018). With the phenomenal support of Vuex, all the back-end handlers and API can be implemented and called in “actions” through “axios” requests, while “state” provides information for front-end developer to use in component’s

template and data. An action can be dispatched in `<script>` of a component on event handlers (for example when clicking on a button). Figure 15 illustrates the connection between Vue's components and Vuex.

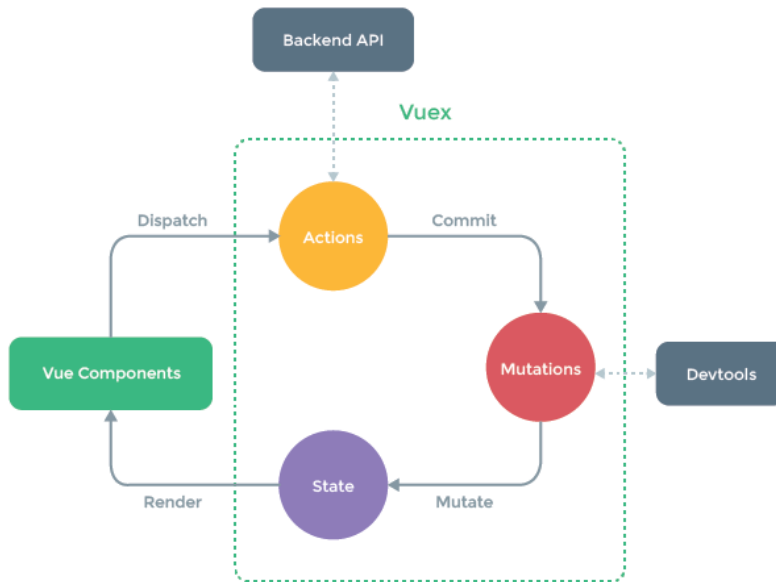


Figure 16 Concept of Vuex with one-way data flow (What is Vuex? 2019)

## 4 CASE DESCRIPTION

### 4.1 Introduction of Osmi application

SuperApp Oy is a Finnish company based in Lahti which focuses on selling web and mobile applications. The author has been working here since April 2019. Osmi is one of SuperApp's clients whose case is assigned for the author and one other developer. The development process is in the final check stage and about to be delivered to customer by the time this thesis is written. The author analyzed Osmi application as a case study of her research.

Osmi application is an idea of a newly startup founded by two Finnish man. They have an ambition to transform the traditional way of renting apartment by creating a platform for landlord and tenant. All the information and interaction related to renting process can be accessed through the app.

### 4.2 Project goal

Osmi was a cross-platform application that could run on both Android and iOS with two target user roles: landlord and tenant. The project goal was to build a platform for these two user roles to communicate and manage housing rental information. In other words, the whole application was a digital progress of renting an apartment in which all data was stored and could easily be observed and modified. The figure below demonstrates how Osmi app worked and the logical flow.

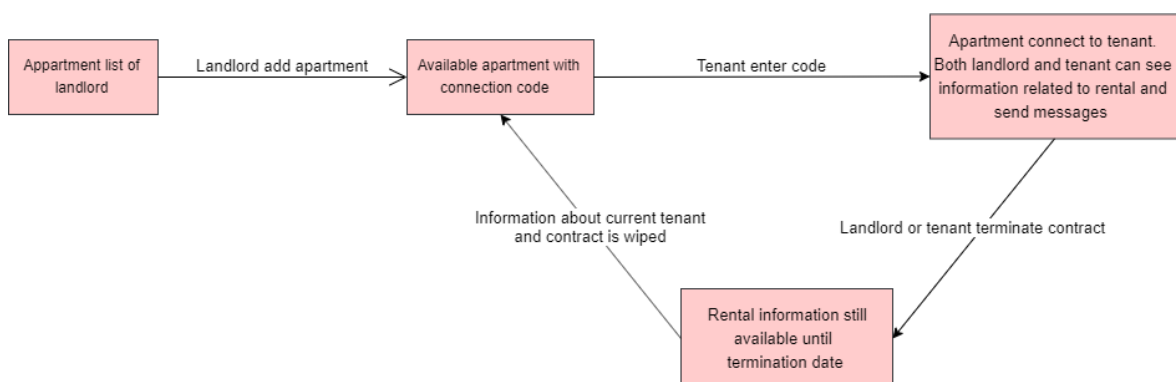


Figure 17 Logical flow of Osmi application.

After registration, a property owner could add multiple apartments as his possessions. Each available house could be linked to one tenant user account with a unique auto-generated connection code. Figure 18 and 19 respectively presents the landlord's view which listed all apartments of that specific user and single apartment view appeared when clicking on one of those cards. Once connected to a house, a renter could view all data as well as interacting with the landlord of that particular apartment.

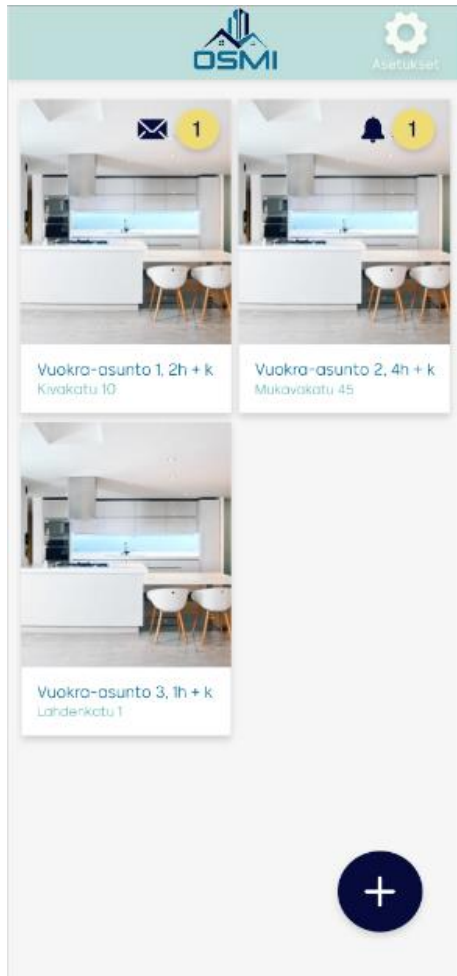


Figure 18 Landlord's view: All apartments listing page

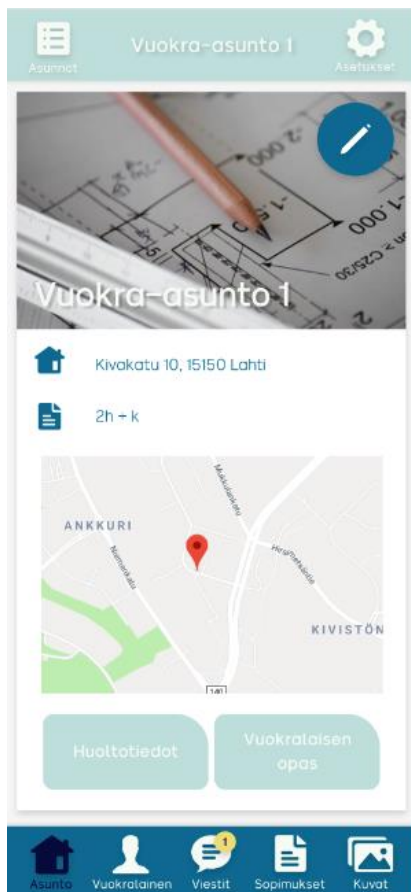


Figure 19 Both tenant and landlord's view: Single apartment

As can be observed from Figure 19, there were five tabs on the bottom navigation menu: "Asunto" (Apartment), "Vuokralainen" (Tenant), "Viestit" (Messages), "Sopimukset" (Contract), "Kuvat" (Images). The structure of a single apartment is summarized in Figure 20.

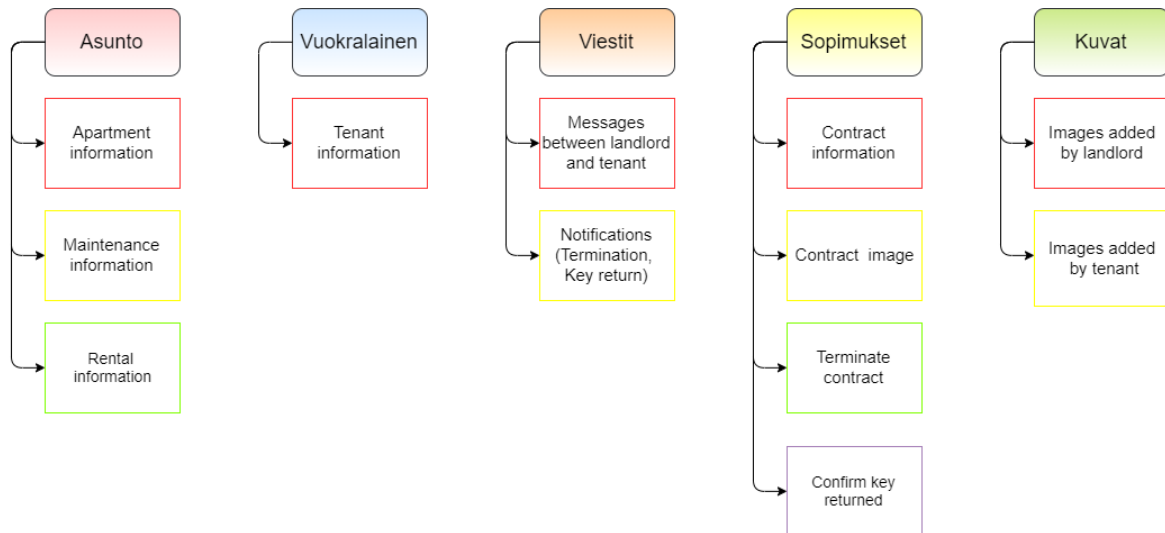


Figure 20 Application structure

The purpose of Osmi project was developing a fully functional application that meets the client's targets and requirements. The core features in five sections mentioned above associated with general functions of the app were:

- **Asunto:** Adding, editing and displaying avatar, data about address, location on map, apartment type and landlord's contacts of the apartment. In addition, the tenant could view information and guidance about rental the landlord added.
- **Vuokralainen:** Generating connection code, adding, modifying and presenting avatar, data about name and contact information of the renter.
- **Viestit:** Implementing chat function and notification of termination and returning key.
- **Sopimukset:** Adding contract information and PDF file, terminating contract (choosing date of termination and declaring reason) and returning key confirmation (including digital signature).
- **Kuvat:** The landlord and tenant could add photos of the apartment in two separate picture folders.
- **General features:** Login, registration, push notification.

#### 4.3 Two developer teams

To testify the advantages and convenience of Vue.js in enhancing the performance of different roles of people in the same project, the author compared her own cooperating development process with another team. Team 1 included Jason, a senior developer who was interested in back-end and the author, a junior developer who was good at front-end.



The author (An Pham) performed as a front-end developer in this project while Jason was responsible for all the back-end features and functionalities. Team 2 was a senior developer, Phan, who was capable of doing both front-end and back-end. He executed the whole project by himself. Both team developed Osmi application to experiment the utility of Vue.js in Team 1 which had two different developer roles.

#### 4.4 Case study plan

Two developer teams used Trello for task management and project organizing. Trello is a platform allowing user to generate boards, lists and cards for all events, duties and assignments contained in a project. These cards can be moved between lists, assigned to one or several user and marked with special colors for different purposes. (Trello 2019.)

For both teams, an application layout was already created by an UX designer of Super-App, therefore the designing phase was not included. As a result, the development of Osmi project was divided into four sprints: Planning, implementing basic role data, deploying detailed core features, testing and fixing bugs. Two teams created four sprints checklists with subordinate tasks. Each phase had precise agenda and due date which would be checked when completed. Smaller and more explicit features were displayed by cards in “To do” list which could be moved to “Done” once accomplished. Developers used four colors orange, blue, yellow and grey respectively to classify cards belongs to sprint 1, 2, 3 and 4. The Trello boards of Team 1 is presented in the two figures below.

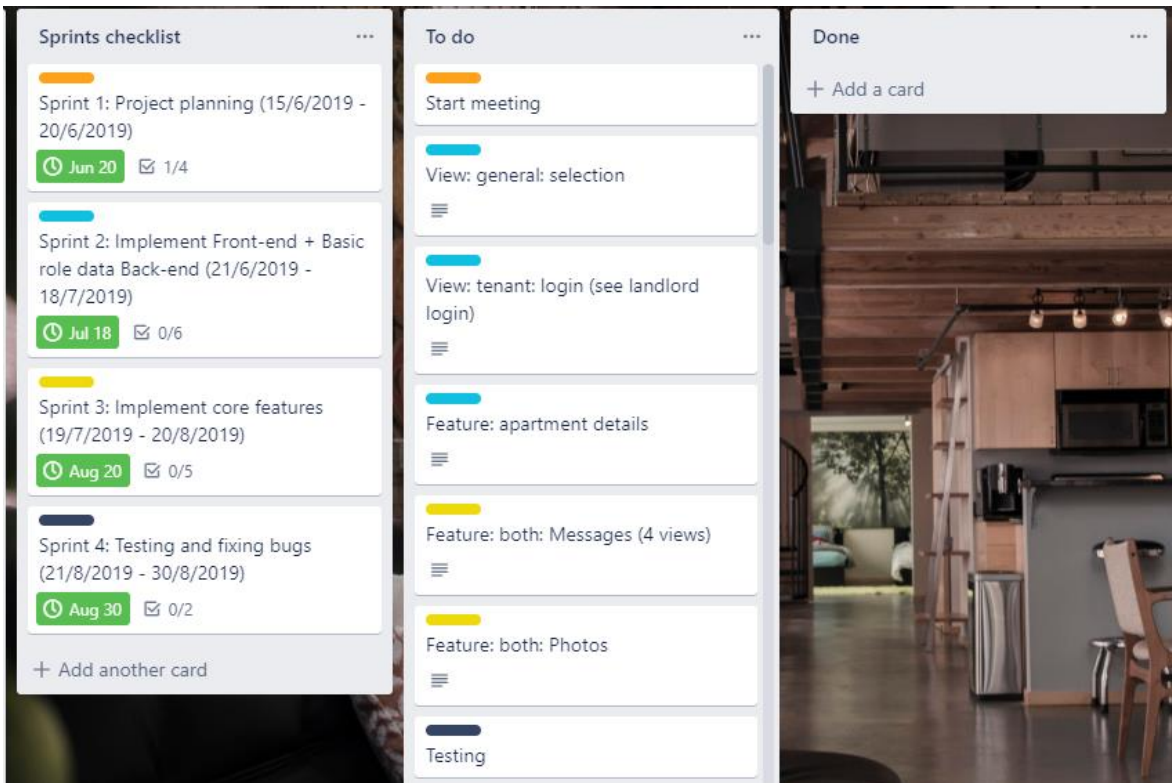


Figure 21 Four sprint checklists and “To do” list (Team 1)

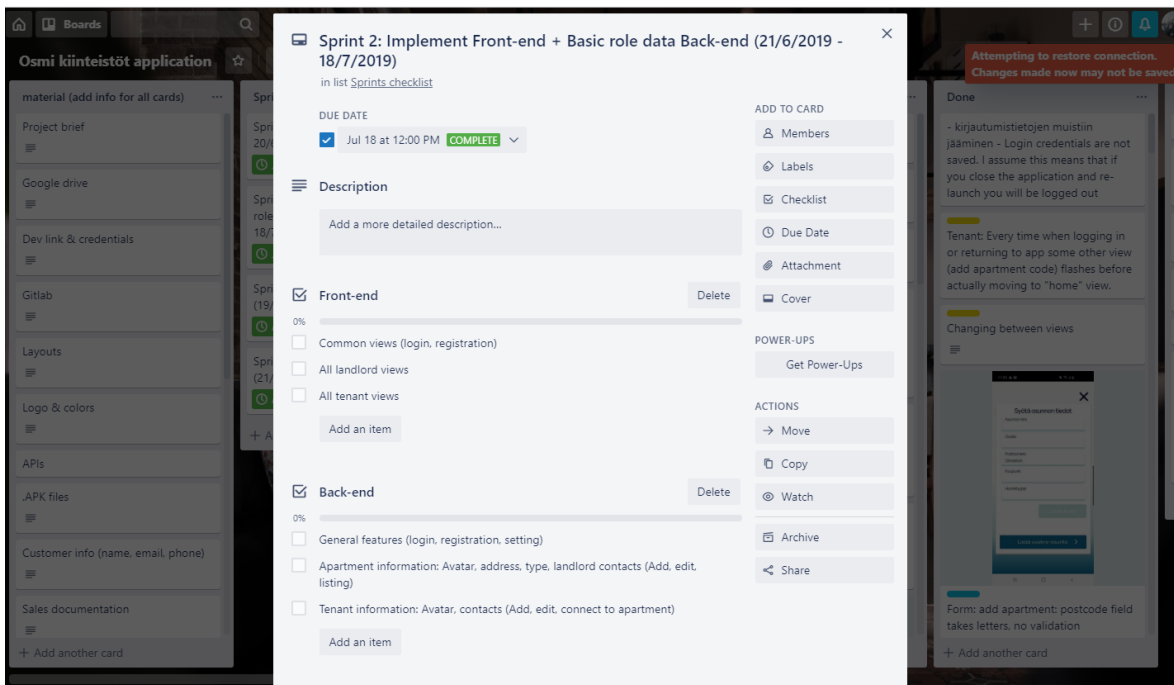


Figure 22 Example of a single sprint checklist (Team 1)

For Team 1, sprint 1 was a preparation stage in which developers had a start meeting to study about the layouts design, followed by setting up the coding environment (arranging development server and creating GitLab project to share code between two developers). The smaller tasks were then determined and assigned for team members and listed in “To do” list. Because the back-end functionalities were expected to be more time-consuming than building front-end layouts, Team 1 targeted that in the print 2 Jason would implement basic role data such as information of landlord, apartment and tenant while the author would completed all views of the application. Consequently, phase 3 fully concentrated on deploying other cores features. Testing and fixing bugs were the missions of the final stage.

Team 2 shared a quite similar schedule with four phases: planning, implementing basic role feature, executing detailed functionalities, testing and fixing bugs. However, the fact that front-end and back-end development were mixed in sprint 2 and sprint 3 differentiated the performance of two teams.

## 5 COLLECTION OF DATA

### 5.1 Data collecting process

The data on developing mobile application by Vue.js of two developer teams was collected in two stages: during and after the projects. The data on time consumption of each tasks was recorded during the development while evaluation results about difficulty levels was gathered after two teams had accomplished all development phases.

#### 5.1.1 Collecting data time consumption

For time recording purpose, Team 1 took advantage of SuperProject, the time-management system of SuperApp. This platform saved all the working records of developer including developer name, project name, task type, milestone, date, task description and duration in hours. The figure below shows how a new work record could be established.

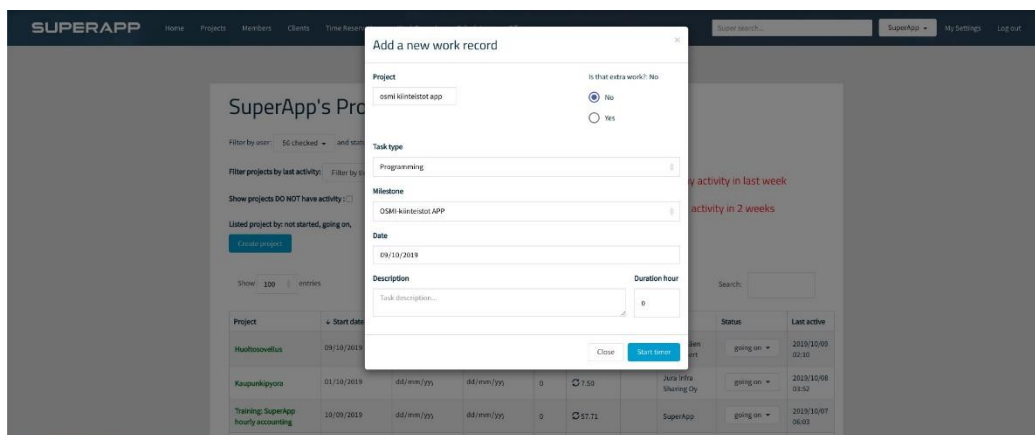


Figure 23 Creating a new work record on SuperProject

Every time each member of Team 1 started with a new task, he or she created an entry to keep track on the time consumption and date of implementation by clicking on the “Start timer” button. SuperProject also allowed developers to inspect all the hourly records, see the total hour and search for particular tasks of an individual. Figure 24 and 25 illustrate how those data could be observed in SuperProject’s dashboard.

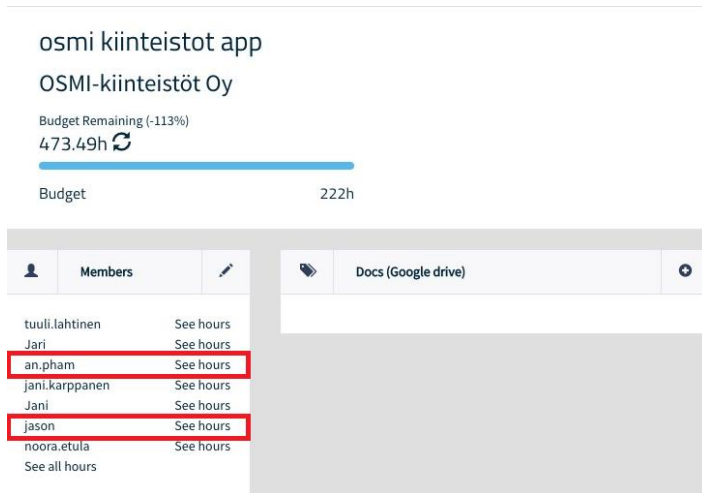


Figure 24 List of all members in Osmi project and links to see hourly records of them. (Two developers of Team 1 is marked with red boxes)

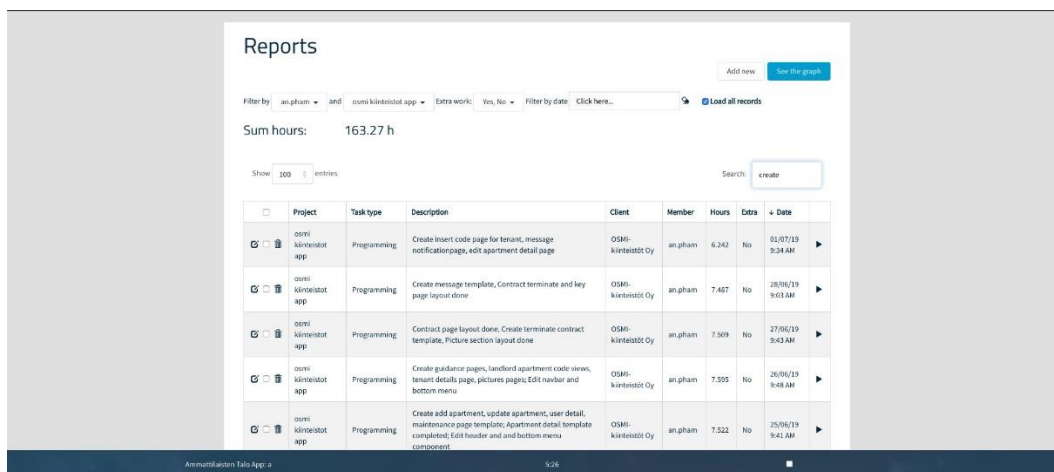


Figure 25 Hourly records of the author

Developers could easily start and stop the timer by the button with “play” symbol located at the end right of each task entry row. Time duration could also be added and modified manually.

In the case of Team 2, because developer Phan built Osmi application specifically for the researching purpose of this thesis and it was not a real project of SuperApp, SuperProject was not used. Instead, he used Clockify, an online time tracker and timesheet that worked utterly similar to SuperProject. At the beginning of each task, Phan input in the task and project name and activated the stopwatch. Once he stopped the timer, the cur-

rent entry was automatically added to his record dashboard. After the project accomplished, Team 2 exported the summary report displaying the sum hour of all development process as well as total working hour on separate feature as different PDF files for analysis. Two figures below present examples of the time tracker and the summary report ready-made for exporting.

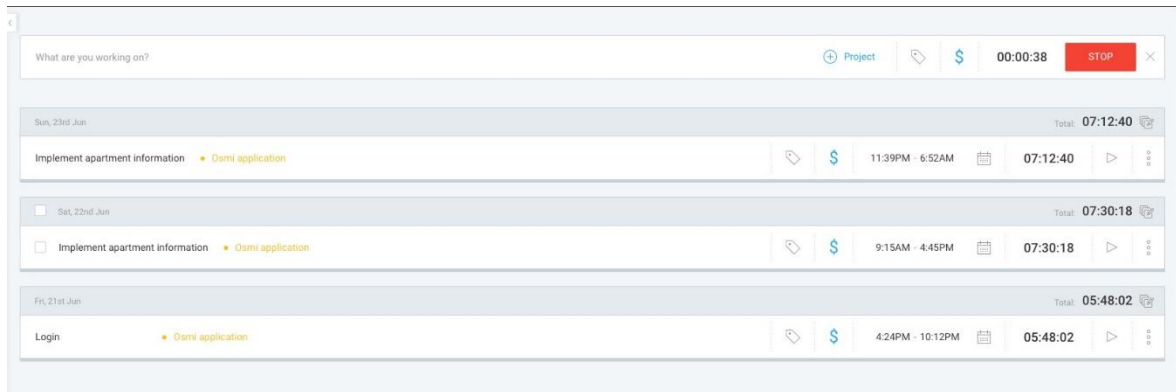


Figure 26 Clockify's time tracker dashboard of Team 2

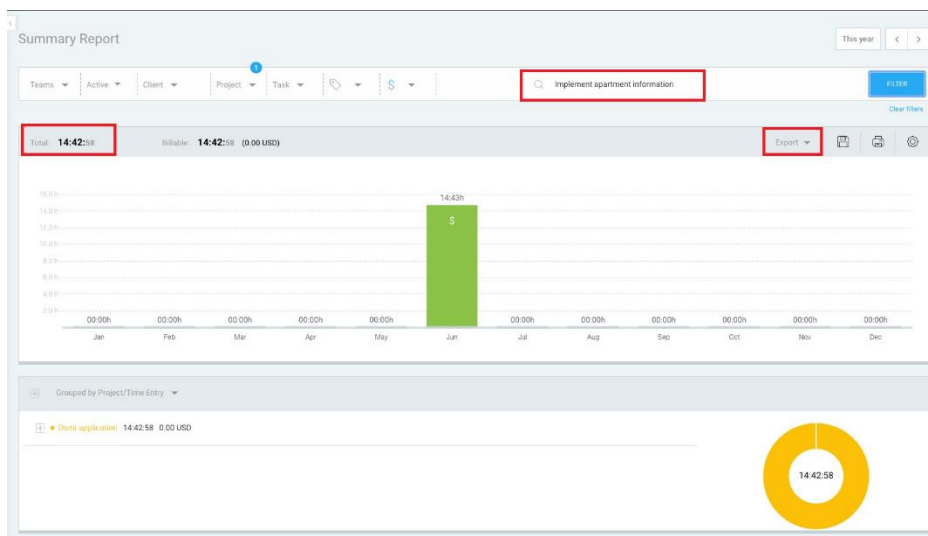


Figure 27 Clockify's summary report for "Implement apartment information" feature of Team 2

### 5.1.2 Collecting data on finalizing phases, difficulties and other supporting data

As previously stated, Team 1 and Team 2 both used Trello as the task management system and supervised the coding progress by moving cards from “To do” to “Done” list. In Team 1 there were two members so each developers was tagged in the card that he or she was responsible for. An example of marking duties is presented below.

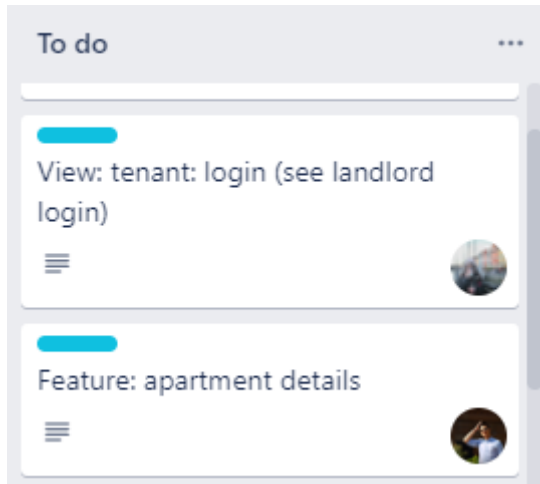


Figure 28 Example of managing task’s duties in Team 1

In Figure 28, the author (An Pham) was tagged in the first card (Creating login view layout) and her colleague (Jason) was responsible for the second one (Implementing apartment details). By this way, managing front-end and back-end workload was straightforward and transparent which made it more convenient to collect data of different developer roles.

Regarding final phases of development process, in sprint 4, all the bugs and fixes found were listed in “Bugs and fixes” list so the data on production quality was also gathered through Trello. Reports related to bug fixing time duration were collected by the two aforementioned time record methods, SuperProject and Clockify.

In both team, personal notes and developer dairies were maintained throughout the programming process to document all noticeable bottlenecks or privileges. At the very end of development process, the author gathered the most significant reviews of all developers through survey form. By this mean, all members of Team 1 and Team 2 committed a questionnaire regarding difficulties and obstacles of the projects. An additional survey on

the convenience level of separating workload with Vue.js was conducted only for two developers in Team 1.

## 5.2 Collected data results

This chapter present all the data of two teams gathered and summarized after entire development process accomplished.

### 5.2.1 Data on development time

After Team 1 and Team 2 successfully completed their projects, the author assembled data on Trello board, SuperProject and Clockify concerning the start and end date of four sprints mentioned in chapter 4.4 and hourly development time of both teams. Firstly, on the subject of sprint durations, all the activity logs on Trello board such as moving cards between “To do” and “Done” lists and marking subordinate tasks in sprint checklists as “Completed” were noted and considered as a contribution of time report. The durations of each development stages are demonstrated in the diagram below with four colors blue, orange, purple and red symbolizes sprint 1, 2, 3 and 4 respectively. The upper row is development time periods of Team 1 while the line under timeline represents durations of Team 2.

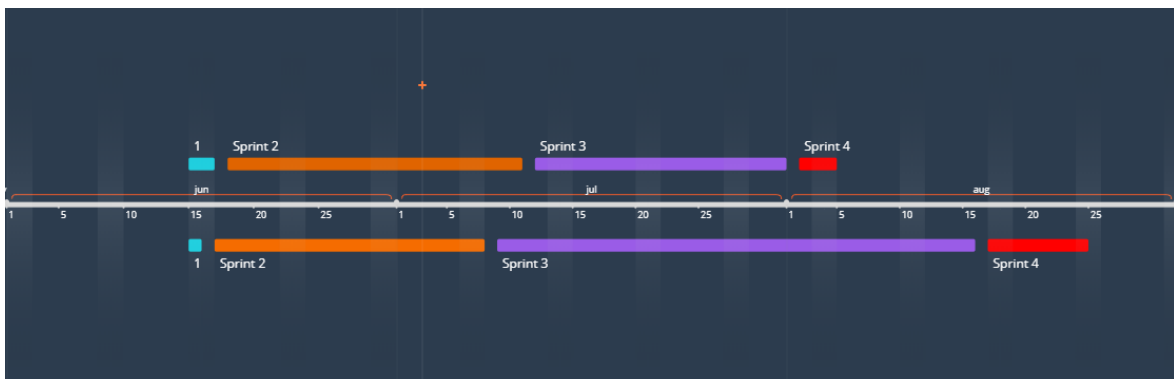


Figure 29 Development time periods of two teams in four sprints

Both developer teams conducted their projects from June 2019 to August 2019 started on June 15. However, Team 1 accomplished the Osmi application on the 5<sup>th</sup> of August while it was not until 25<sup>th</sup> that Team 2 successfully finished the development.

Furthermore, the hourly records that two team spent exclusively for programming with Vue.js in the second and third stage (regardless of time consumption for planning and fix-



ing bugs in the first and final phases) were also summarized. These statistics were illustrated in Table 4. There were nine main tasks to build all required features listed in the previous chapter. A special note was that because two members of Team 1 developed Osmi application concurrently, the development time of this team was not the sum of individual records but rather the actual time that a single task consumed.

Table 4 Programming time records of two developer team

No.	Task	Team 1	Team 2
1	General features (login, registration, setting)	15 hours	24.5 hours
2	Implementing apartment information	38 hours	51 hours
3	Implementing tenant information	33.5 hours	45.5 hours
4	Connecting apartment to tenant account by auto-generated code	16.5 hours	28 hours
5	Deploying maintenance and rental documents	29 hours	47.5 hours
6	Deploying contract functions	27 hours	39.5 hours
7	Adding chat function	46 hours	53 hours
8	Creating apartment's photo folders	23 hours	36.5 hours
9	Push notification	7.5 hours	6.5 hours

The total programming time of the first team not including bug fixing time was 235.5 hours, while the same data of the second team was 332 hours.

### 5.2.2 Data on finalizing phases and production quality

After developed all essential features and functions of Osmi application, two teams spent the fourth sprint for testing the app and fixing bugs. The author gathered information about these finalizing phases to study about production quality and how smooth and favorable

the bug fixing processes were. In this case, bugs were all the front-end and back-end elements that did not work appropriately as well as necessary fixes required in order to improve and perfect the app performance. At the end of the development procedure, there were 12 errors detected in the app version that Team 1 delivered. Two members of this team spent four days to solve these bugs. On the other hand, the application of Team 2 had 21 bugs resulted in nine days of bug fixing and retesting in sprint 4. In total, the first team consumed 22.5 hours on the last sprint while 58 hours was required for Team 2 to finish the same task. Some of the most noticeable bugs found in Team's 2 app is presented below.

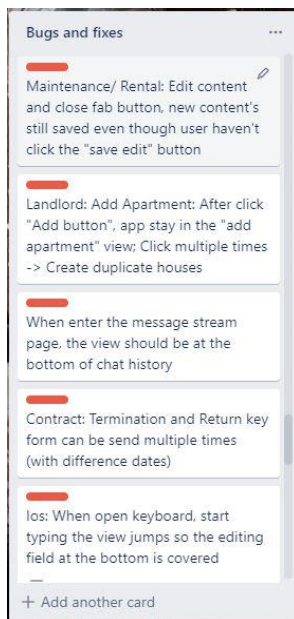


Figure 30 Examples of Team 2's bugs

### 5.2.3 Data on difficulty during development process

At the end of the fourth sprint, the data on obstructions regarding both front-end and back-end were collected by questionnaires. Any step that was challenging or caused a developer more time and effort than expected was considered as a bottleneck. The table below displays information about number obstacles interpreted through developer's responses. The amount of obstacles of Team 1 was a total number of both developer An Pham and Jason.

Table 5 Number of obstacles during development process

No.	Task	Team 1	Team 2
1	General features (login, registration, setting)	2	3
2	Implementing apartment information	3	5
3	Implementing tenant information	0	0
4	Connecting apartment to tenant account by auto-generated code	1	2
5	Deploying maintenance and rental documents	2	3
6	Deploying contract functions	3	6
7	Adding chat functions	3	5
8	Creating apartment's photo folders	2	4
9	Push notification	1	0

#### 5.2.4 Data on difficult levels of cooperation using Vue.js

Two members of Team 1 participated in a special survey together to clarify the convenience of cooperation between front-end and back-end developers utilizing Vue.js framework. There were five scales to evaluate difficulty level with the following characteristics:

- 1 – Very easy, can be executed smoothly without any obstruction
- 2 – Quite simple, may take a few effort to employ but doesn't cause much downtime
- 3 – Slightly complicated, require more time and attempt to resolve
- 4 – Challenging and possibly demands supports from other developer(s) to be overcome
- 5 – Substantially serious obstacle that really time-consuming to achieve adequate solution

The results is revealed in the table below.

Table 6 Difficulty levels of different aspects when working simultaneously with Vue.js

No.	Evaluation aspect	Difficulty level given by Team 1
1	Merging code to programming at the same time	1
2	Developing front-end without back-end (Build <template> layout)	1
3	Developing back-end without front-end (Employ <script> section and handling request by PHP back-end)	2
4	Connecting front-end to back-end using Vuex	2
5	Synchronization between front-end and back-end	2

## 6 STUDY AND ANALYSIS

In this section, all data collected in the previous chapter is analyzed to compare the development processes of two developer teams.

### 6.1 Comparing development time of two developer team

The figure below is the durations of the entire projects conducted by Team 1 and Team 2 with specific period of each sprint calculated in days. The duration was interpreted through the interval between start and end dates of a single stage mentioned in chapter 5.2.1.

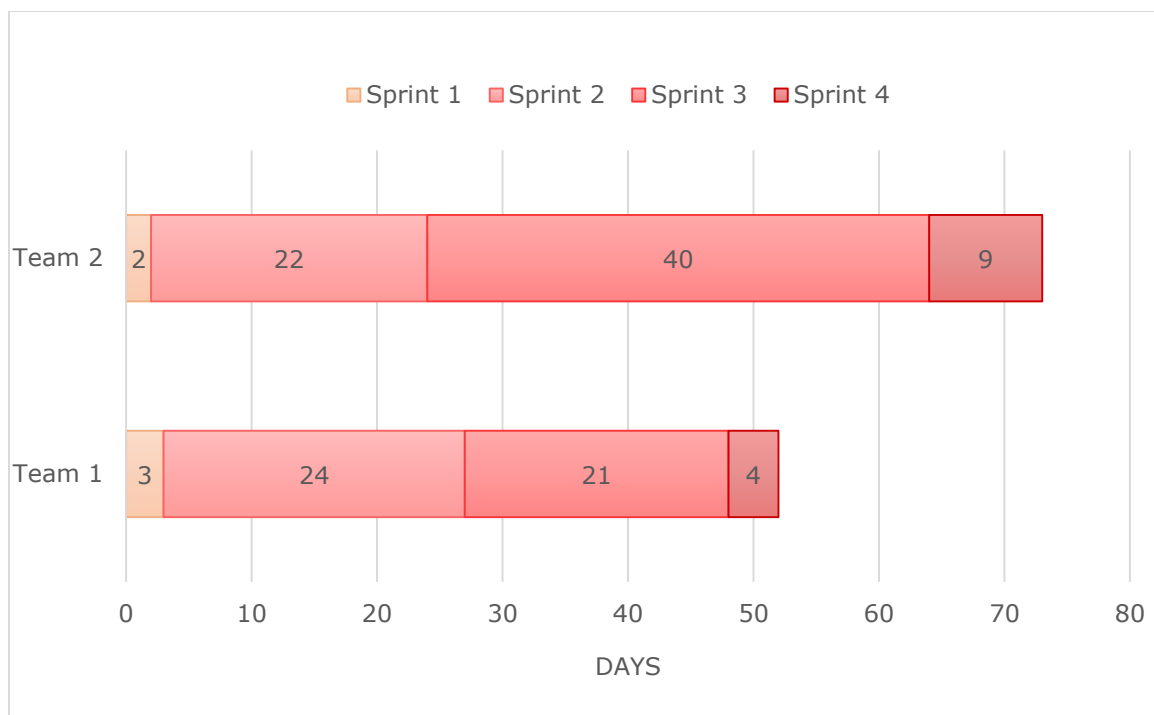


Figure 31 The durations of four sprints of Team 1 and Team 2

In total, Team 1 successfully delivered Osmi application after 52 days while it took Team 2 73 days to accomplish the development. This discrepancy did not exist from the beginning of the project timeline. As demonstrated in the figure, in the first sprint in which developers planned the workload and set up the coding environment, Team 2 was actually one day faster than Team 1. This was because Team 2 consisted of two developers doing separate jobs. Therefore, the planning phase consumed more time to dividing tasks and setting up common repository of code on GitLab. Developer Phan worked alone with the project so those preparations were slightly faster. When it came to sprint 2, Team 2 was

still in the lead regarding time with 22 days of development while the data of Team 1 was 24 days. This is rooted in the fact that although the milestone of this phase is to complete general features, apartment and tenant information, Team 2 targeted to complete all the front-end views at this point. Two days in difference was a part of the required time for developer An Pham to finish all the app views for future back-end implementations. Nevertheless, a substantial distinguish was recorded in two last sprints of the development processes. Team 2 spent approximately double the time period of team 1 in both sprint 3 and 4. The significant gap in the third stage was because developer Phan had to work on both front-end and back-end at the same time. This blend in workload demanded extra effort as well as generating distraction during development process that prevented developer from concentrating thoroughly on neither roles. By contrast, developer Jason of Team 1 already had the finished layouts implemented in the previous sprint so he only needed to focus comprehensively on deploying back-end handlings and connect them to front-end. Figure 32 describes the detailed comparisons of development time that nine main features mentioned in Table 4 consumed.

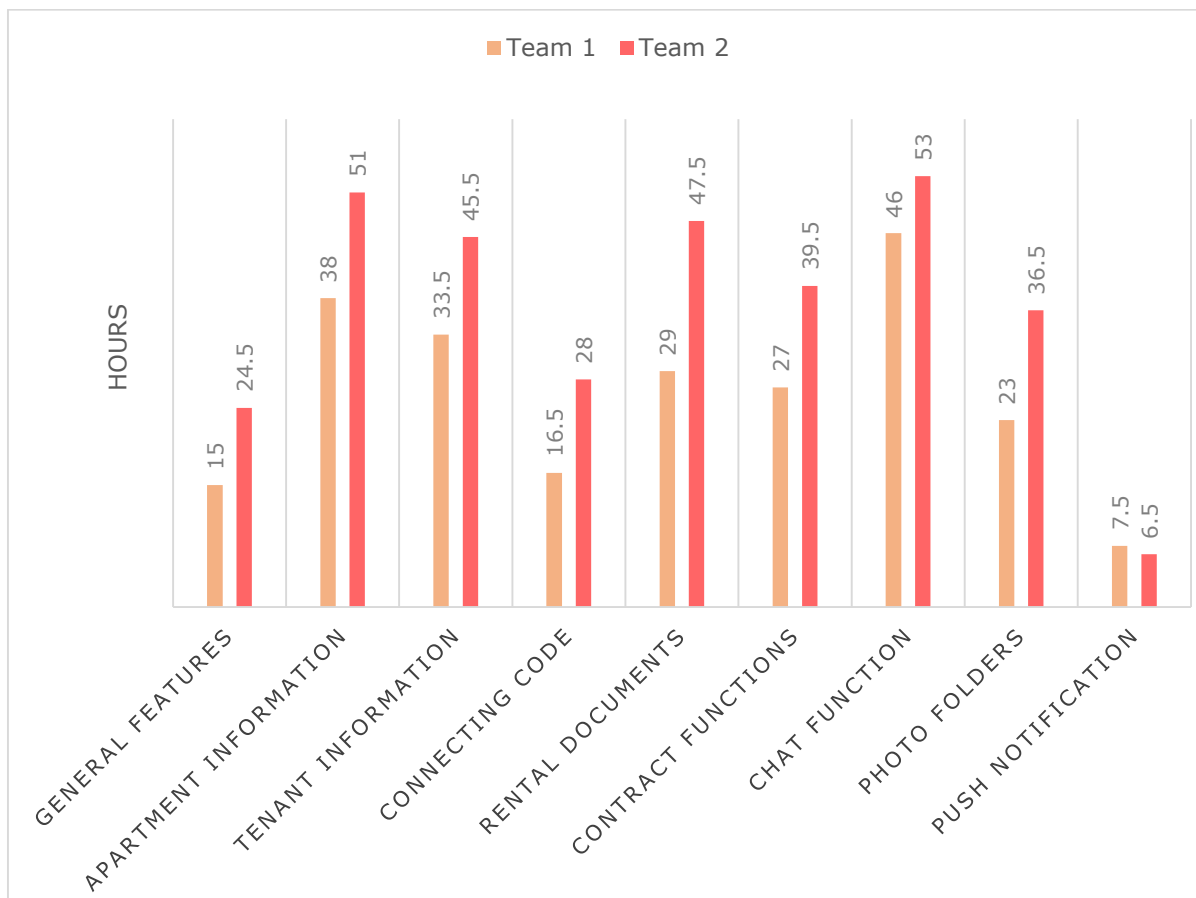


Figure 32 Comparisons of developing nine main features

## 6.2 Comparing quality of production

The table below demonstrates the differences between two teams in the finalizing phases.

Table 7 Comparisons about production quality

	<b>Team 1</b>	<b>Team 2</b>
<b>Number of bugs</b>	12 bugs	21 bugs
<b>Bug fixing hours</b>	22.5 hours	58 hours

The number of bugs detected in the app version that Team 2 delivered was nearly double that data of Team 1. In addition, the application of the first team preformed remarkably smoother than the second one's without delays and crashes. The reason behind this was because Team 1 had two members working closely in a supportive manner so a developer could check the app performance during programming process and inform his/her colleague if there was something performing improperly. The manifold points of view increased the possibility that bugs were found instantly in the time they were implementing. Furthermore, separating duties reduced workload for developers and therefore provided retrospective time in the end of each sprints for them to evaluate their work. Consequently, the time required for Team 1 to fix bugs was just roughly half of Team 2. Another factor that made modifying phase of An Pham and Jason faster was pair debugging in which front-end and back-end bugs were fixed by separate developers. On the contrary, developer Phan could only fix one bug at a time.

## 6.3 Comparing the difficulty levels of tasks

When evaluating difficulty level based on number of obstacles, the development process of Team 1 was apparently more pleasure than Team 2's. The number of obstacles in each tasks are compared in the figure below.

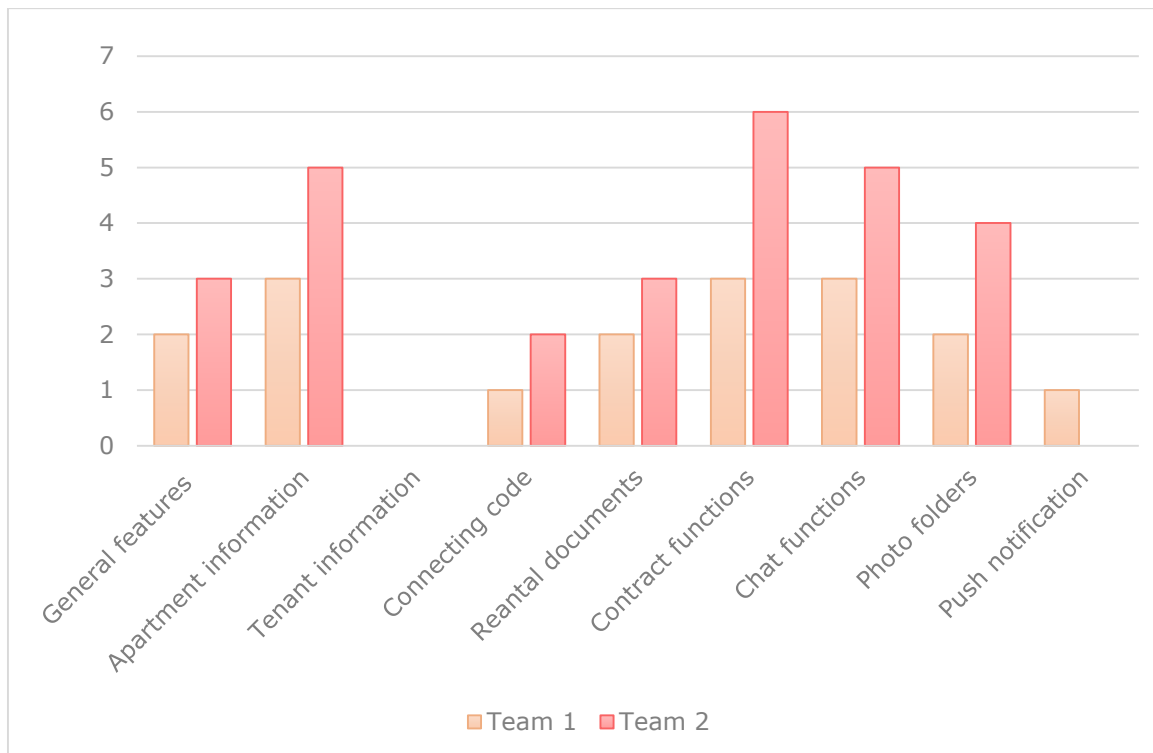


Figure 33 Comparing number of obstacles in each tasks

As can be observed from the graph, there are more obstacles that confronted the second team than the first one in almost every tasks. The most noticeable gaps were witnessed in the implementation of apartment information, contract functions, chat functions, deploying photo folders and debugging. This difference was because although developer Phan of Team 2 was a full-stack developer with high expertise, the skillsets and personal preferences of individual developer of Team 1 were more specific and centralized. To put it another way, members of Team 1 were able to focus on the particular aspect that they are good at and did not need to work on the field they were not confident of. Moreover, there was a higher chance that An Pham and Jason experienced the similar tasks when they were working with front-end or back-end of other projects because it was their focused proficiency. Therefore, it was easier for Team 1 to contemplate their problems. However, there were two tasks that the data of Team 1 did not lead: implementing tenant information and push notification. Taking the first one into account, because apartment information and tenant information were quite similar feature in both front-end and backend so after finished the pervious feature, there was no bottleneck for both team in implementing the next one. With push notification, developer Phan deployed this in several projects so he did not meet any obstacle while Jason who were responsibility of this feature in Team 1 had done this only once, resulted in one obstacle in Osmi project.



#### 6.4 Analysis on convenience level of separating workload with Vue.js

Based on Team 2's responses presented in Table 6, there was no difficulty in setting up a common coding environment. Git was a powerful and convenient tool that allow developers to create their own branches for version controlling and merge other's code by just one simple command line. There were also many available Git tutorial sources online that coders could seek for help. Vue.js itself was very supportive in separating development roles. The author easily constructed the front-end without back-end because she only work with *<template>* part and styling which was merely HTML and CSS combined with several prefixed syntaxes of Vue.

Developing back-end without front, however, was slightly more complex because at some points creating back-end requests without working with app layouts limited the vision. This resulted in the problem that sometimes developers needed to modify the call-back functions to meet the layout's demands because some required variables was missing. Connecting front-end to back-end and synchronization between these two parts were occasional confusing because two developers had the different ways of implementing some features in the perspective of their duties. For examples, An Pham intended to use a common template for all pages that shows same styled input fields but Jason suggested that separating these pages guaranteed the transparency and simplicity of *<script>* functions rather than mixing them. The team agreed to continue with the second solution. Nevertheless, connecting and synchronizing the front-end and back-end were overall simple tasks because the *<script>* section of Vue allowed developers to assign information received through back-end requests into local "data". Therefore, when the back-end of one feature was completed, Team 1 just needed to replace the hardcoded placeholder value in *<template>* (which An Pham had written when constructing app layout) with the official data. By this mean, two developer could work simultaneously without difficulties and combine their work later.

## 7 CONCLUSION

This chapter concludes answers for the research questions summarized after theoretical research and case study's outcome. The limitation, reliability and validation of the research are also discussed to give a decent and complete point of view to the results. The final section suggest ideas for further studies in the future.

### 7.1 Answering research questions

The main purpose of this research was to identify and examine the benefits of Vue.js in developing mobile application in general and more specific, in separating workload between different development roles. The theoretical background part provided fundamental concepts of three Javascript frameworks for comparisons as well as introducing technical knowledge base and methodologies related to Vue.js. A case study's process, results and analysis were then presented to compose a thorough reply to two research questions mentioned in chapter 2.1:

- How the advantages of Vue.js can be optimized in developing mobile application in general?
- How choosing Vue.js can benefit IT companies by enhancing the efficiency of different development roles in the same project?

Regarding the first question, Vue.js framework brings plenty of advantages to companies and programmers. The most beneficial characteristic of Vue is its small learning curve because of simple structure and syntaxes which are principally based on HTML, CSS and JavaScript. This saves a lots of learning time for developers as well as training resources for companies. Moreover, Vue.js is highly flexible that it allows developing in diverse way by providing many internal features to serve different coding purposes, for examples Vuex, Vue Router and Vue Server-Side Renderer. Last but not least, high adaptability is also a superiority of Vue because a developer can easily integrate Vue into an existing project thanks to its progressive nature. In addition, the developer himself can switch to use another JavaScript framework from Vue.

When it comes to encouraging different development roles' performances in the same project, the benefits of Vue.js is even more impressive. It allows not only two but three roles to work simultaneously: back-end developer, front-end developer and UI designer. These role can cooperate smoothly without big problems because of the clear and separate structure of Vue and the help of Vuex. This increases time efficiency, production qual-

ity and allow easier debugging. Moreover, utilizing Vue.js to divide workload reduces difficulties for individuals because each developer can focus on the aspect that they are confident of. Consequently, IT companies can accomplish more cases in a certain period with higher effectiveness.

## 7.2 Limitations

In the case study, despite the difference regarding development roles, both Team 1 and Team 2 used Vue.js. It would be more distinctive and persuasive if Team 2 also had front-end and back-end developers and used another framework, for example Angular or React. If so, the research could have compared the supportiveness of Vue with the other one. The second limitation of the thesis is although the original theory stated that a project could have up to three developer roles, Team 1 did not involve a UI designer in the programming process, but only a front-end developer and a back-end developer.

## 7.3 Reliability and Validation

The concept and methodologies of Vue.js presented in the theoretical part were based on researches from both official and well-known sources as well as the author's own knowledge. The author herself has a certain understanding of Vue.js because she had done four applications with this framework. Proper research methods and tools such as a stopwatch, a programming journal, and a questionnaire were used to collect the data in the case study. The data analyzed was from the observation of two developer teams' projects, so the reliability and validation of this are determined by the development process's results. The technologies used, coding styles, and developers' expertise could affect the research results.

## 7.4 Suggestions for further study

Some of the potential topics to continue after or related to this research can be:

- The benefits that Vue.js brings to IT companies compared to Angular
- The benefits that Vue.js brings to IT companies compared to React
- The advantages of Vue.js in enhancing different development roles in the same project compared to other JavaScript frameworks
- Utilizing Vue.js in building cross-platform mobile applications.

## 8 SUMMARY

Along with a surging number of mobile applications in the past few year, many frameworks are established to serve the application development industry. Choosing an adequate framework for implementation is an essential decision for tech companies. This paper is conducted to suggest and testify the advantages of Vue.js, one of the most popular JavaScript framework nowadays.

With theoretical base and case study providing practical evidence and analysis, the author archived the goal of this research and was able to answer two main research questions. The privileges that Vue.js brings to development process in general associated with its outstanding advantages in promoting the productivities of different developer roles were explained and examined. C, key benefits found included small learning curve, high flexibility and adaptability, great time efficiency and production quality, faster debugging and decreasing obstacles during development process. From these finding, it can be concluded that choosing Vue.js is a deliberate decision for IT companies, especially fast-growing startups that have restricted resources. Limitations, reliability, validations and suggestions for further study was also involved.

## 9 LIST OF REFERENCES

### Written References

Creswell, J.W 2003 Research Design Qualitative, Quantitative, and Mixed Methods Approaches. 2<sup>nd</sup> Edition. Thousand Oaks: Sage Publications, Inc.

Creswell, J.W. & Plano Clark, V.L. 2003. Designing and conducting mixed methods research. Thousand Oaks, CA: Sage Publications.

DeWalt, K. M. & DeWalt, B. R. 2002. Participant observation: a guide for fieldworkers. Walnut Creek, CA: AltaMira Press.

Hevner, A., 2004. Design Science in Information Systems Research. Minneapolis: MIS Quarterly.

Prince, M. & Felder M. 2006. Inductive Teaching and Learning Methods: Definitions, Comparisons, and Research Bases.) United States: American Society for Engineering Education.

Simon, H.A. 1996. The Sciences of the Artificial. London: MIT Press.

### Electronic Sources

Ahuvia, Y. 2019. React vs. Vue (vs. Angular). Medium [accessed 29 September 2019]. Available at: <https://medium.com/fundbox-engineering/react-vs-vue-vs-angular-163f1ae7be56>

Aldwin, N. 2019. What is Angular? – A Beginner’s Guide. Hostinger [accessed 27 September 2019]. Available at: <https://www.hostinger.com/tutorials/what-is-angular>

AWS, 2019. What is Mobile Application Development?. Amazon [accessed 8 September 2019]. Available at: <https://aws.amazon.com/mobile/mobile-application-development/>

Ball, K. 2018. Why Vuex Is The Perfect Interface Between Frontend and API [accessed 2 October 2019]. Available at: [https://zendev.com/2018/05/21/vuex-perfect-interface-frontend-backend.html?fbclid=IwAR3ZhN3HvUCrMTDquVP9jx5lUs5okMIXQhtK8trv-qprlfQgINw\\_MCMB6hiQ](https://zendev.com/2018/05/21/vuex-perfect-interface-frontend-backend.html?fbclid=IwAR3ZhN3HvUCrMTDquVP9jx5lUs5okMIXQhtK8trv-qprlfQgINw_MCMB6hiQ)

Blair, I. 2019. Mobile App Download and Usage Statistics (2019) [accessed 8 September 2019]. Available at: <https://buildfire.com/app-statistics/>

Bojanowska, I. 2018. Pros and Cons of the Vue.js Framework [accessed 17 October 2019]. Available at: <https://naturally.com/blog/pros-cons-vue-js>

Education Ecosystem. 2019. Introduction to ReactJS JavaScript Library [accessed 27 September 2019]. Available at: <https://www.education-ecosystem.com/guides/programming/react-js/history>

Evan You, 2019. GitHub Customer story [accessed 20 September 2019]. Available at: <https://github.com/customer-stories/yyx990803>

Ferguson, N. 2019. What's The Difference Between Frontend And Backend Web Development? [accessed 25 September 2019]. Available at: <https://career-foundry.com/en/blog/web-development/whats-the-difference-between-frontend-and-backend/>

Fernandez, M. 2019. 7 Qualitative Research Methods for High-Impact Marketing [accessed 15 September 2019]. Available at: <https://optinmonster.com/qualitative-research-methods-for-understanding-your-user/>

Guru99, 2019. What is Backend Developer? Skills to become a Web Developer [accessed 21 September 2019]. Available at: <https://vuejs.org/v2/guide/syntax.html>

Hussain, A. 2019. Intro to TypeScript [accessed 27 September 2019]. Available at: <https://codecraft.tv/courses/angular/quickstart/typescript-intro/>

Lindley, C. 2019. Front-end Developer Handbook 2019 [accessed 25 September 2019]. Available at: <https://frontendmasters.com/books/front-end-handbook/2019/>

Malhotra, M. 2019. Vue.js is good, but is it better than Angular or React? [accessed 29 September 2019]. Available at: <https://www.valuecoders.com/blog/technology-and-apps/vue-js-comparison-angular-react/>

Martin, S. 2019. Top Mobile App Development Trends of the Year 2019 [accessed 9 September 2019]. Available at: <https://hackernoon.com/top-mobile-application-development-trends-in-2019-5bc1ba19188>

Mroczkowska, A. 2018. Stage 3. Designing UX & UI – Mobile & Web App Development Process [accessed 21 September 2019]. Available at: <https://www.thedroidsonroids.com/blog/stage-3-designing-ux-ui-mobile-web-app-development-process#ux-ui>

Pluralsight. 2015. What's the Difference Between the Front-End and Back-End? [accessed 25 September 2019]. Available at: <https://www.pluralsight.com/blog/film-games/whats-difference-front-end-back-end>

React. 2019. Introducing JSX [accessed 28 September 2019]. Available at: <https://reactjs.org/docs/introducing-jsx.html>

Rouse, M. 2015. Mobile UI (mobile user interface) definition [accessed 21 September 2019]. Available at: <https://searchmobilecomputing.techtarget.com/definition/mobile-UI-mobile-user-interface>

Rouse, M. 2019. Mobile application development definition [accessed 10 September 2019]. Available at: <https://searcharchitecture.techtarget.com/definition/mobile-application-development>

Rybachuk, T. 2016. Front-End Developer vs UI Developer – Who Is Who? [accessed 23 September 2019]. Available at: <https://vintage.agency/blog/front-end-developer-vs-ui-developer-who-is-who/>

Schade, A. 2014. Responsive Web Design (RWD) and User Experience [accessed 26 September 2019]. Available at: <https://www.nngroup.com/articles/responsive-web-design-definition/>

Sidorenko, I. 2019. What Are The Pros And Cons Of Using Vue.js [accessed 30 September 2019]. Available at: <https://towardsdatascience.com/what-are-the-pros-and-cons-of-using-vue-js-3689d00d87b0>

Soiferman, K. 2010. Compare and Contrast Inductive and Deductive Research Approaches [accessed 15 September 2019]. Available at: <https://files.eric.ed.gov/fulltext/ED542066.pdf>

TechMagic. 2018. React vs Angular vs Vue.js — What to choose in 2019? [accessed 30 September 2019]. Available at: <https://medium.com/@TechMagic/reactjs-vs-angular5-vs-vue-js-what-to-choose-in-2018-b91e028fa91d>

The State of JavaScript Survey, 2018. Vue.js [accessed 29 September 2019]. Available at: <https://2018.stateofjs.com/front-end-frameworks/vuejs/>

Topics. GitHub 2019 [accessed 17 October 2019]. Available at: <https://github.com/topics>

Trello. 2019 [accessed 26 September 2019]. Available at: <https://trello.com/en>

Vue.js. 2019a. Introduction [accessed 26 September 2019]. Available at: <https://vuejs.org/v2/guide/>

Vue.js. 2019b. Directives [accessed 26 September 2019]. Available at: <https://012.vuejs.org/guide/directives.html>

Vue.js. 2019c. Template Syntax [accessed 21 September 2019]. Available at: <https://vuejs.org/v2/guide/syntax.html>

Vuex. 2019. What is Vuex? [Accessed 30 September 2019]. Available at: <https://vuex.vuejs.org/>

Wodehouse, C. 2019. The Role of a Front-End Web Developer: Creating User Experience & Interactivity. Upwork [accessed 25 September 2019]. Available at: <https://www.upwork.com/hiring/development/front-end-developer/>

Wohlgethan, E. 2018. Supporting Web Development Decisions by Comparing Three Major JavaScript Frameworks: Angular, React and Vue.js [accessed 20 September 2019]. Available at: [http://edoc.sub.uni-hamburg.de/haw/volltexte/2018/4350/pdf/BA\\_Wohlgethan\\_2176410.pdf?fbclid=IwAR0R8hVyOnVEx-qGvoJtJ2wp46FHuyiDaamcWkJovBnSsK4MuYmMf1WaDPAc](http://edoc.sub.uni-hamburg.de/haw/volltexte/2018/4350/pdf/BA_Wohlgethan_2176410.pdf?fbclid=IwAR0R8hVyOnVEx-qGvoJtJ2wp46FHuyiDaamcWkJovBnSsK4MuYmMf1WaDPAc)

Xing, Y., Huang, J. & Lai, Y. 2019. Research and Analysis of the Front-end Frameworks and Libraries in E-Business Development [accessed 12 September 2019]. Available at: [https://www.researchgate.net/publication/332456776\\_Research\\_and\\_Analysis\\_of\\_the\\_Front-end\\_Frameworks\\_and\\_Libraries\\_in\\_E-Business\\_Development](https://www.researchgate.net/publication/332456776_Research_and_Analysis_of_the_Front-end_Frameworks_and_Libraries_in_E-Business_Development)

### **Oral References**

Nikula, J. 2019. CTO. SuperApp Oy. Interview 10th September 2019




## APPENDICES

## Appendix 1 GitLab branch of An Pham

SuperApp > projects > Osmi > Repository

dev-an osmi / + History Find file Web IDE


 Fix bugs: Message topnav, Photo topnav, placeholder picture for landlord and...  
An authored 1 month ago fa767af1

Name	Last commit	Last update
mobile	Fix bugs: Message topnav, Photo topnav, placeholder pi...	1 month ago
tools	Initial commit	4 months ago
.DS_Store	icon and splash screen updated	1 month ago
.gitignore	Initial commit	4 months ago
README.md	Initial commit	4 months ago
package-lock.json	Initial commit	4 months ago

## Appendix 2 GitLab branch of Jason

SuperApp > projects > Osmi > Repository

dev-jason osmi / + History Find file Web IDE

 **read me update**  
Jason authored 1 week ago 39a86cd6

Name	Last commit	Last update
mobile	ad banner	1 week ago
tools	Initial commit	4 months ago
.DS_Store	Added camera icon to renter picture folder	1 month ago
.gitignore	Initial commit	4 months ago
README.md	read me update	1 week ago
package-lock.json	Initial commit	4 months ago