HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY
SCHOOL OF INFORMATION AND COMMUNICATION TECHNOLOGY

_____ * _____

THESIS

SUBMITTED FOR PARTIAL FULFILLMENT OF
THE REQUIREMENTS FOR THE DEGREE OF

ENGINEER

IN

INFORMATION TECHNOLOGY

VADI ITRITHUC –

MOBILE APPLICATION OF AUDIO NEWS AND MAPS FOR COMMUNITY

Author: **An Nguyen Quynh Anh**
Class ICT 1 K59

Supervisor: **Dr. Nguyen Thi Thu Trang**

Hanoi, May 2019

HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY
SCHOOL OF INFORMATION AND COMMUNICATION TECHNOLOGY

_____ * _____

THESIS

SUBMITTED FOR PARTIAL FULFILLMENT OF
THE REQUIREMENTS FOR THE DEGREE OF

ENGINEER

IN

INFORMATION TECHNOLOGY

VADI ITRITHUC –

MOBILE APPLICATION OF AUDIO NEWS AND MAPS FOR COMMUNITY

Author: **An Nguyen Quynh Anh**
Class ICT 1 K59

Supervisor: **Dr. Nguyen Thi Thu Trang**

Hanoi, May 2019

Declaration

Full name: Ân Nguyễn Quỳnh Anh

Phone number: +84969048043 Email: anhanqq@gmail.com

Class: ICT1 K59 Program: ICT

I – An Nguyen Quynh Anh – hereby declare that the graduation thesis entitled « Vadi Itrithuc » submitted by me to School of Information and Communication Technology, Hanoi University of Science and Technology for partial fulfillment of the requirements for the degree of Engineer in Information Technology was carried by me under the guidance and supervision of Dr. Nguyen Thi Thu Trang. I further declare that all results presented in this thesis are truthful and are not copied from any other work.

Hanoi, 25th May, 2019

Author

An Nguyen Quynh Anh

Attestation of the supervisor on the fulfillment of the requirements of the thesis

Hanoi, 25th May, 2019

Supervisor

Dr. Nguyen Thi Thu Trang

Acknowledgement

First and foremost, I would like to express my sincere gratitude to my family: my parents, my little brother and my little sister, who have been always supporting me financially and spiritually during the last years. Even though we have been through many ups and downs, they are still there loving and encouraging me unconditionally. Without their unwavering and unselfish love and support given to me at all timesB, I would not be here at the moment to proudly write these words.

In my five-year journey, I have found a teacher, an inspiration and a role model, Dr. Nguyen Thi Thu Trang, lecturer of School of Information and Communication Technology. Not only guiding and encouring me to complete this thesis, she has set an example for me for her serious working attitude, her enthusiasm and dedication in everything she does.

I have great pleasure in acknowledging my gratitude to all the lecturers in School of Information and Communication Technology for the knowledge they have given to me. I will always respect them as the most inspirational and dedicated teachers ever.

It would be inappropriate if I omit to mention the names of my dearest friends: Thuy, Nguyen, Ngoc, who have always been beside me to share all my joys and sorrows, kept me going on my path in whatever manner possible.

My acknowledgement would not be complete without thanking my ICT-K59 classmates. We have been though five years together and created the most memorable days that I will never forget.

Finally, I would like to send my thank you to my boyfriend, Le Trong Hung. Being so gentle, so caring, so smart, he has brightened my life. Without his support everyday, it would not be possible for me to complete my journey happily and delightfully.

And thank you, Hanoi University of Science and Technology, for giving me a chance to pursue my dream. During the last five years, I have grown up a lot. The days I was here would be the most beautiful days I have ever had. It gave me many good friends, enthusiastic mentors, inspirational figures who I always respect.

An Nguyen Quynh Anh

Abstract

Knowledge is the key to not only an individual's success but also a whole country's. Vietnam can only prosper once the desire and creativity are ignited, all citizens are well-educated and the potential of science and technology is recognized and emphasized. Each and every individual in this society, regardless of background knowledge, is capable of working together in order to dedicate to the community in their own ways. To develop Vietnamese Digital Knowledge System is one of the most efficient ways to play our part in the Industrial Revolution 4.0. The act of creating, sharing and exploiting this knowledge system widely among the society is the first step in raising the level of education and awareness for people of all social status, as well as the capability of innovation in every fields.

Being a part of Vietnamese Digital Knowledge System, Vadi Itrithuc is an application providing users with audio news and map services, both of which are developed based on Vietnamese technologies by Vietnamese people. Audio news function utilizes text-to-speech technology, constructed by Dr. Nguyen Thi Thu Trang from Hanoi University of Science and Technology, while map services are provided by a group of researchers from Hanoi National University. Beside using the application, users now can contribute to the knowledge system by easily send and edit voice commands to the system in order to build a rich and realistic dataset for training more accurate and stronger Vietnamese smart dialog systems. For those functionalities, Vadi Itrithuc got the second prize in Startup Competition for Vietnamese Students – SV.Startup in 2018 and Most Favorite Product in Angelhack Hackathon Hanoi 2018.

Tóm tắt

Tri thức chính là chìa khóa tới thành công, không chỉ đối với một cá nhân mà còn đối với cả một quốc gia. Nước chỉ có thể mạnh một khi khơi dậy được khát vọng và sáng tạo, nâng cao được dân trí, tăng cường được tiềm lực khoa học công nghệ. Mỗi người dù trình độ, hiểu biết ở mức nào cũng đều cần và đều có thể cùng nhau, giúp nhau tìm hiểu, tạo ra và cống hiến tri thức bằng nhiều cách. Phát triển Hệ tri thức Việt số hóa là một cách hiệu quả trong thời đại cách mạng công nghiệp 4.0. Việc tạo lập, chia sẻ, khai thác và sử dụng Hệ tri thức Việt số hóa một cách sâu rộng trong xã hội là tiền đề để nâng cao trí tuệ của mọi tầng lớp nhân dân, năng lực đổi mới sáng tạo ở các ngành, các lĩnh vực trong bối cảnh của cuộc Cách mạng công nghiệp lần thứ 4.

Là một dự án trong đề án "Phát triển Hệ tri thức Việt số hóa" do Thủ tướng Chính phủ phê duyệt, Vadi Itrithuc là một ứng dụng di động cung cấp dịch vụ báo nói và bản đồ giao thông cho người dung. Hai dịch vụ trên đều được phát triển bởi những nhà khoa học của Việt Nam. Dịch vụ báo nói sử dụng công nghệ chuyển đổi văn bản thành giọng nói (TTS) của Tiến sĩ Nguyễn Thị Thu Trang, giảng viên trường Đại học Bách khoa Hà Nội. Dịch vụ bản đồ được cung cấp bởi một nhóm các nhà nghiên cứu và phát triển của trường Đại học Quốc gia Hà Nội. Sử dụng hoàn toàn những thành tựu công nghệ của Việt Nam, Vadi Itrithuc còn cho phép người dùng đóng góp dữ liệu ra lệnh bằng giọng nói trong quá trình sử dụng, đồng thời cũng là đóng góp cho Hệ tri thức Việt số hóa với mục tiêu thu thập dữ liệu mang tính thực tế cao để tạo ra những hệ thống chat bot tiếng Việt mạnh mẽ và chính xác hơn. Nhờ những cải tiến và chức năng mới này, Vadi Itrithuc đã đạt giải nhì trong cuộc thi Học sinh, sinh viên với ý tưởng khởi nghiệp – SV.Startup năm 2018 và giải Sản phẩm được yêu thích nhất trong cuộc thi Angelhack Hackathon tại Hà Nội năm 2018.

Table of contents

List of figures

List of tables

List of acronyms

API	Application Programming Interface
NER	Named Entity Recognition
HTML	HyperText Markup Language
CSS	Cascading Style Sheets
OSM	Open Street Map

Chapter 1 Introduction

1.1 Motivation

Industry 4.0, also known as the fourth industrial revolution, is a name given to the current trend of automation and data exchange in manufacturing technologies. Machines will be able to work independently, or corporate with humans in creating a customer-oriented production field that constantly works on maintaining itself. They will become an independent entity that is able to collect data, analyse it and advise upon it. Manufacturers will communicate with computers rather than operates them. Along with this development, people are required to learn to adapt with life that changes day by day.

In Vietnam, the term "Industry 4.0" can be heard everywhere at every time. However, Vietnam is still a developing country with a large number of physical labours who even do not truly understand the definition of "Industry 4.0". There is a need for popularizing intellectuality in all fields for all citizens. Innovation should be conducted in fields that directly affect daily life and well-being of people such as civil laws, medical, engineering. It is a must to creating a friendly environment for all citizens and corporations to join hands as both providers and users to enrich the resources of Itrithuc – Vietnamese Digital Knowledge System [7].

Traffic has been a serious problem that affects our daily life in a bad way. The number of vehicles is increasing dramatically day by day, which results in severe congestions. Moreover, more vehicles mean that it is more dangerous to drive. It is necessary to create a mobile application that supports drivers when moving on roads but reduces hand interactions to ensure their safety. Besides, because transport infrastructure in Vietnam changes every day, many new bridges and roads are built, maps need to be updated frequently to enhance searching and routing accuracy, give users better experiences. In addition to map services, drivers also need to update news every day

although they are on roads most of the time. Reading news traditionally is not a suitable choice in this situation since they are required to focus on driving to avoid dangers to themselves as well as others.

For these reasons, in this thesis, I have developed an application with audio news functions to keep drivers stay tuned; it uses Vietnamese map services to quickly update any transport infrastructure changes. Besides, virtual assistance is a trend these days. We can apply virtual assistants in plenty of fields with undeniable advantages. However, creating virtual assistants is not an easy job. Not only hardware requirements, it needs a huge amount of data in training process, especially Vietnamese data for Vietnamese chat bots. In order to build a strong and realistic dataset, data should be collected from users in their daily lives. Due to those reasons, I decided to equip the application with the ability to listen to voice commands. Although it is not a truly virtual assistant, it still helps users in some basic activities. Moreover, users can contribute data under the form of voice commands.

1.2 Objectives and scopes

For the motivation mentioned in the above section, the objectives of conducting this thesis is to developing an application for community with following functions:

- Instead of reading news traditionally, drivers can listen to audio news. The application is supposed to provide audio news service for drivers to listen no matter where they are and what they are doing. The voice of audio news should be fluent and emotional like human being's to give users the best experience.
- The application can support voice commands to control basic functionalities. It should be easily accessed from main screens.
- Instead of using maps services provided by foreign agencies, the application will use Vietnamese services to quickly update with road and traffic changes.
- The application is supposed to allow users to contribute voice commands (add new ones and edit existing ones).

1.3 General solution

Nowadays there are two main approaches when developing applications: web applications and mobile applications. Web apps have some undeniable advantages. They can run on every device which has a browser supporting HTML/CSS/JavaScript. It is not necessary for user to bother updating for web applications are able to update themselves without any user intervention. However, when a web app runs on a mobile device's web browser, the problem that arises is that each mobile device has unique features, which cause unique troubles web apps cannot fully resolve. Multiple mobile platforms require higher costs for development and maintenance. Moreover, web apps performance varies in different platforms; in other words, their performance is unstable and unpredictable. Compared with web applications, mobile ones are the better approach. They fully take advantages of devices' hardwares, ensure their performance to be high and stable. Mobile devices are now equipped with huge memories, high-quality hardwares, but prices are getting lower and lower. Additionally, more and more frameworks supporting multi-platform application development are created, which makes mobile app development less resource-consuming and more convenient. I decided to choose React Native for its reliable creator – Facebook, and for its huge user community.

Vadi Itrithuc is developed based on an available application – Vadi. It is a complicated application with many functions and services. Independent services should be separated in different servers to avoid unexpected changes, simplify error detection and maintenance processes. Microservice architecture is the best choice for this project. Different services communicate with others or with UI layer via RESTful APIs, which accepts many data formats, has better performance when using less bandwidth.

1.4 Structure of thesis

The structure of the rest of the thesis is as follows.

Chapter 2 presents requirement analysis of the project. It is divided into three parts: similar applications, functionality overview and use case specifications.

In chapter 3, I will list and explain the technologies and frameworks adopted in the project. Specifically, I divide them into two groups: front-end and back-end. In each part, I will give an overview about the technology or framework, then explain why I apply them in my application.

Chapter 4 will describe the process of developing and deploying the application. I will explain the architecture of the whole application, then its architecture and detail design. The construction and testing steps will also be mentioned.

Chapter 5 presents my main contributions to the application. Difficult problems arising during developing process and how I have solved them will be specified.

Finally, in chapter 6, there will be a conclusion of the thesis, which would be about the result after this project. Moreover, I would like to tell about the orientation of the future works for this system.

Chapter 2 Requirements analysis

The first chapter has presented why I conducted this project, the scopes and approaches that I took to fulfill the requirements and the structure of the thesis. In this chapter, I will discuss some similar applications in the market and their pros and cons, then give an overview of the application's functionalities. Some major use cases will also be specified.

2.1 Similar applications

Nowadays, there are many mobile applications that supports drivers when moving on roads such as Google Maps and Waze. According to a survey conducted in the United States, Google Maps is the most popular navigation app, while Waze takes the second place.

Google Maps is known as an application with wealth of information, multiple of transport mode and can even be shared through Google or Facebook account. However, information in Google Maps may have errors. Occasionally, ambiguities and flaws in location data may produce a route that does not take users to the destination they expect. It does not have real-time information or unusual conditions such as roads damaged by weather, blocked by recent construction works. Some remote locations may not be in Google Maps either.

Waze is a community-driven map application that uses data from users to provide quicker navigation routes. Different from Google Maps, data contributed from the Waze community is real-time and can be used to create traffic alerts.

Despite their great advantages and popularity, they still have some disadvantages. First, either of them supports Vietnamese. This results in difficulties for users who do not know English. Furthermore, users are not allowed to use voice commands to

control the application. Traffic information in Vietnam could not be update frequently.

There are some applications providing Vietnamese audio news. However, only some of them generate audio from tradition news automatically. Their voices are not realistic and emotional like human.

Vadi Itrithuc is developed based on the application Vadi, a mobile application of audio news and maps. The audio news functions of two applications are similar. However, Vadi uses Google Maps Service and does not support voice commands. Different functions between the two will be specified in 2.2.

2.2 Functionality overview

Specifically, for audio news function, I have restructured Vadi to manage application state and data flow better. Besides, I made some changes to libraries react-native-maps and react-native-voice to support VNMap and Vais services. Voice commands are also supported to control some basic functions in the application, and users can contribute their own voice commands to the system.

2.2.1 General use case diagram

The two main actors of the system are guests (who do not log in) and users (who log in by Vadi / Google / Facebook accounts) which is shown in Figure 1. Guests are anonymous users, anyone who opens the application. They are only allowed to use basic functionalities of the application, specifically, manage audio news, search for places and directions, and manage layers on the map. Guests can register an account on the system or login by Google or Facebook account to have access to more functions.

Users are guests who have logged into the system. They have access to all functions that guests can do. Besides, they can manage personal playlists, download audio news to play offline and contribute for voice commands. When finishing a session, users may log out of the system to use the application as a guest.

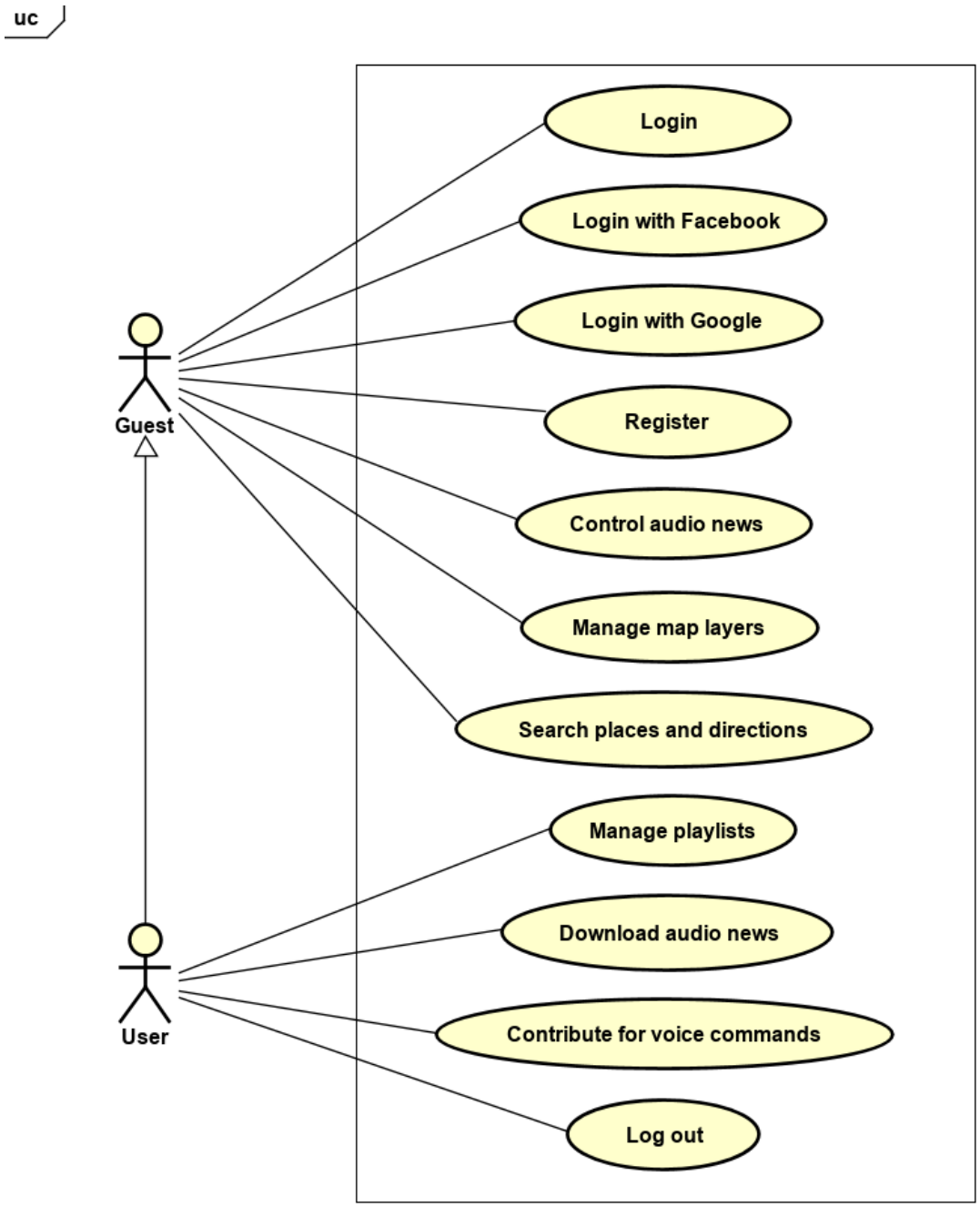

Figure 1 General use case diagram.

2.2.2 Use case diagram for "Control audio news"

The use case "Control audio news" is decomposed into smaller use cases in Figure 2. Users can (i) play particular audio news. In addition, they can (ii) play audio news, (iii) pause audio news, (iv) play next audio news and (v) play previous audio news, each of which may be executed in two ways: by touching and by voice.

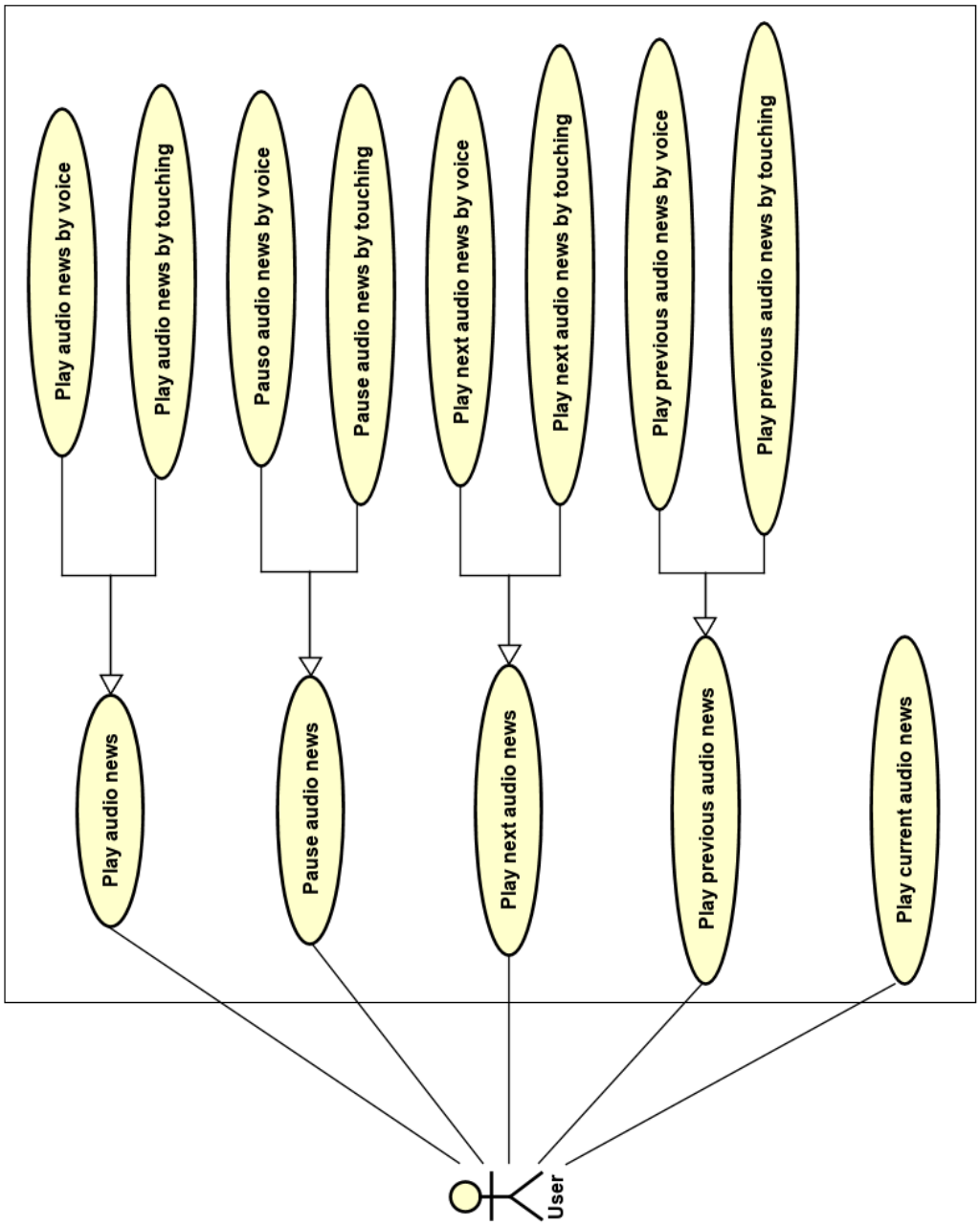

Figure 2 Use case diagram for “Control audio news”.

2.2.3 Use case diagram for "Search for places and directions"

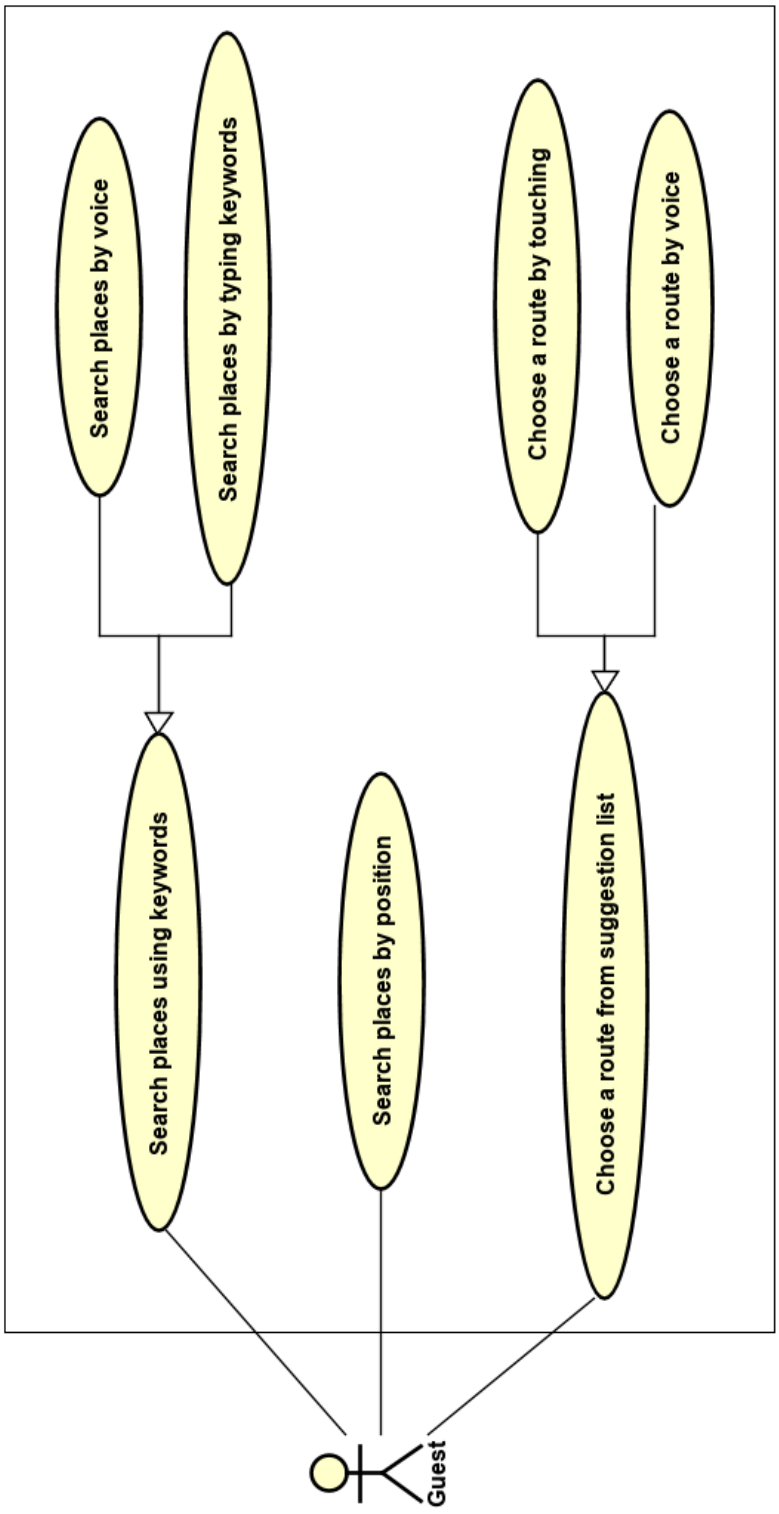

Figure 3 Use case diagram for "Search for places and directions".

The use case “Search for places or directions” is illustrated in Figure 3. This function allows both guests and users to search for places and directions in three ways: (i) type the key words into the search box and pick the expected place from the suggestion list, (ii) speak out the expected place and the voice command function will consider it as a key word, then pick the first place in the suggestion list, and (iii) drag the maps to some certain region in order for the expected place to be in the middle of the screen.

2.2.4 Use case diagram for “Contribute for voice commands”

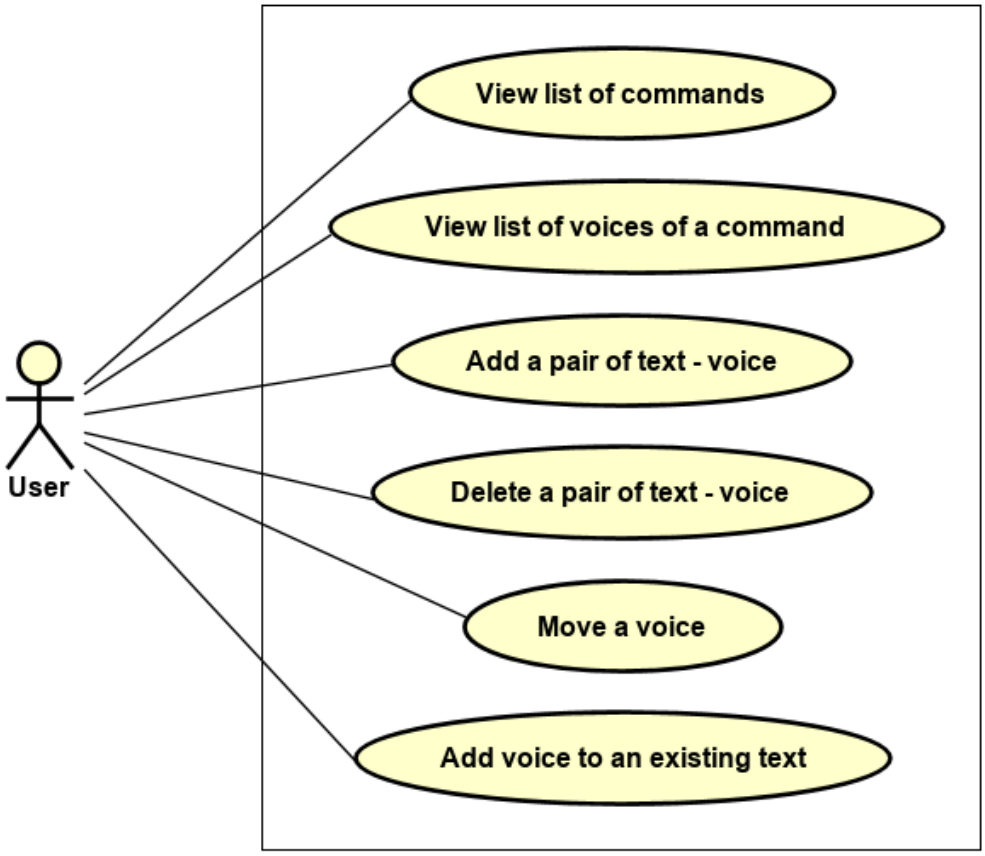

Figure 4 Use case diagram for “Contribute for voice commands”.

The use case “Contribute to voice commands” is described in Figure 4. Only users who have logged into the system are allowed to use this function.

Users can view list of commands that are programmed to execute by voices, then press on a particular command to view its list of texts. On this screen, users can press the add button to contribute pairs of text – voice, or contribute more voices to an

existing text. Additionally, if users detect errors in the text of a voice, they can edit it. They also can delete some pairs of text – voice if they are wrong. In case that a pair of text – voice is correct but is arranged in wrong command, users can move it to the right one.

2.2.5 Use case diagram for "Manage map layers"

The use case "Manage map layers" is described in Figure 5. Users can select layers to display on the maps, deselect them one by one or deselect them all. Users can also click on a point to get its detail information.

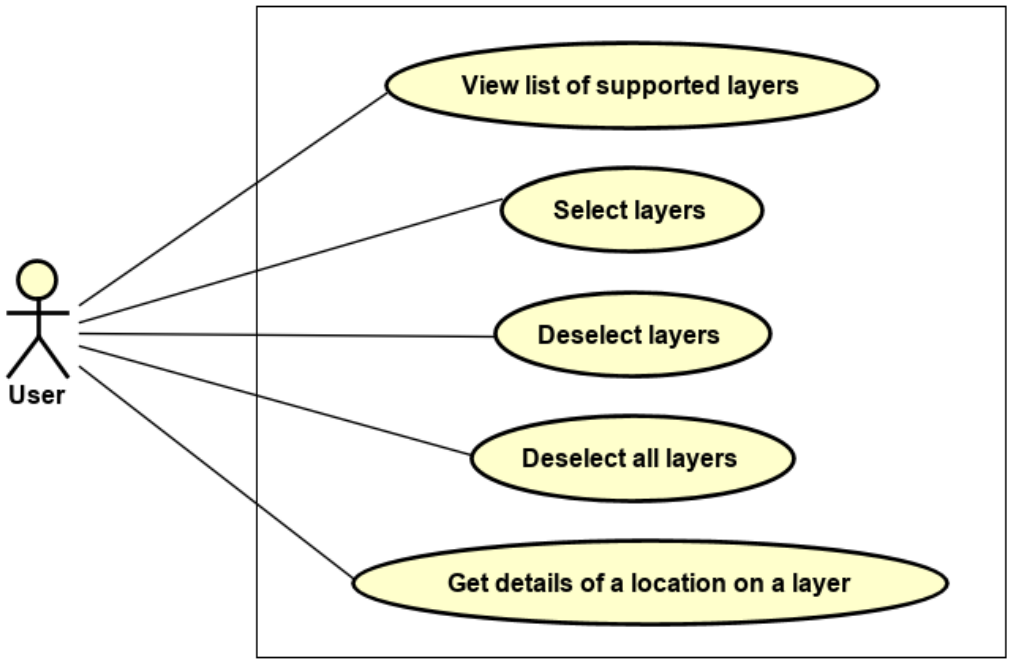

Figure 5 Use case diagram for "Manage layers".

2.3 Use case specifications

There are totally 33 use cases which are listed in Table 1.

Table 1 List of use cases

Use case ID	Name	Use case ID	Name
UC01	Login	UC02	Login by Facebook
UC03	Login by Google	UC04	Register new account
UC05	Log out	UC06	Play audio news by touching
UC07	Play audio news by voice	UC08	Pause audio news by touching
UC09	Pause audio news by voice	UC10	Play next audio news by touching
UC11	Play next audio news by voice	UC12	Play previous audio news by touching
UC13	Play previous audio news by voice	UC14	Play current audio news
UC15	Manage playlists	UC16	Download audio news
UC17	Search places by voice	UC18	Search places by typing keywords
UC19	Search places by position	UC20	Choose a route by touching
UC21	Choose a route by voice	UC22	View list of commands
UC23	View list of voices of a command	UC24	Add a pair of text – voice
UC25	Delete a pair of text – voice	UC26	Move a pair of text – voice
UC27	Add voice to an existing text	UC28	Edit text of a voice
UC29	View list of supported layers	UC30	Select layers
UC31	Deselect layers	UC32	Deselect all layers

Use case ID	Name	Use case ID	Name
UC33	Get details of a location on a layer		

For the thesis' length is limited, I will only specify some main use cases in this section. They are "Control audio news", "Search places by voice", "Edit text of a voice" and "Get details of a location on a layer".

2.3.1 Use case specification for "Control audio news"

Table 2 Use case specification for "Control audio news"

<table>
<tr><td>Use case ID</td><td>UC06-14</td><td>Use case</td><td>Control audio news</td></tr>
<tr><td>Actor</td><td colspan="3">Guest</td></tr>
<tr><td>Pre-conditions</td><td colspan="3">No</td></tr>
<tr><td colspan="4"><u>Playing audio news by touching:</u>
<table>
<tr><td>Pre-conditions</td><td colspan="3">No</td></tr>
<tr><td>Basic flow of events</td><td>#</td><td>Doer</td><td>Action</td></tr>
<tr><td></td><td>1.</td><td>Actor</td><td>Click the play button</td></tr>
<tr><td></td><td>2.</td><td>System</td><td>Get the audio news' url address</td></tr>
<tr><td></td><td>3.</td><td>System</td><td>Play the autio news</td></tr>
<tr><td>Alternative flow</td><td>#</td><td>Doer</td><td>Action</td></tr>
</table></td></tr>
</table>

	3a.	System	Stop the media player since the url address can not be accessed

Pause audio news by touching:

Pre-conditions	Audio news is being played		
Basic flow of events	**#**	**Doer**	**Action**
	1.	Action	Click the pause button
	2.	System	Pause the media player
Alternative flow	No		

Play next audio news by touching:

Pre-conditions	The current news is not at the end of the list		
Basic flow of events	**#**	**Doer**	**Action**
	1.	Actor	Click the next button
	2.	System	Get the id of the next news in the playlist, update the current news to be already listened.
	3.	System	Get the next audio news url address
	4.	System	Play the audio
Alternative flow	**#**	**Doer**	**Action**

	3a.	System	Load more audio news if all in the current playlist are already played
	4a.	System	Pause the media player if no audio url exists

Play previous audio news by touching:

Pre-conditions	The current news is not at the beginning of the list		
Basic flow of events	**#**	**Doer**	**Action**
	1.	Actor	Click the previous button
	2.	System	Get the id of the previous news in the playlist, update the current news to be already listened.
	3.	System	Play the audio
Alternative flow	**#**	**Doer**	**Action**
	3a.	System	Pause the media player if no audio url exists

Play current audio news

Pre-conditions	No		
Basic flow of events	**#**	**Doer**	**Action**

	1	Actor	Press the play button on ArticleItem component or in news detail screen
	2	System	Determine position of the news in the playlist
	3	System	Get the audio url
	4	System	Play the audio
Alternative flow	4a	System	Pause the media player if no audio url exists
Post-conditions	No		

2.3.2 Use case specification for "Search places by voice"

Table 3 Use case specification: "Search places and directions by keywords"

Use Case ID	UC11	**Use Case**	Search places and directions by keywords
Actor	User		
Preconditions	Internet connection		

Basic flow of events	**#**	**Doer**	**Action**
	1	Actor	Click on the search box
	2	System	Check for GPS permission
	3	Actor	Type in keyword
	4	System	Call API to get suggestion list
	5	System	Display the first five places in the suggestion list.

	6	Actor	Choose the desired place
	8	System	Save the search history
Alternative flow	**#**	**Doer**	**Action**
	2a	System	Open device's setting
	6b	System	Allow access to GPS
Post-conditions	No		

2.3.3 Use case specification for "Edit text of a voice"

Table 4 Use case specification for "Edit voice commands"

Use Case ID	UC28	**Use Case**	Edit voice commands
Actor	User		
Preconditions	No		
Basic flow of events	**#**	**Doer**	**Action**
	1	Actor	Open voice command management screen
	2	System	Check Internet connection
	3	System	Call API to get the list of commands
	4	System	Display the list of commands
	5	Actor	Choose a command
	6	System	Display list of texts and voices for the command
	7	Actor	Choose a pair of voice – text to edit
	8	Actor	Click to play the voice
	9	Actor	Click the text box to edit the text
	10	Actor	Edit the text

	11	Actor	Click save button to save the changes
	12	System	Call the API to update the changes
Alternative flow	**#**	**Doer**	**Action**
	2a	Actor	Close voice command management screen
	5a	Actor	Close voice command management screen without saving
Post-conditions	No		

2.3.4 Use case specification for "Get details of a location on a layer"

Table 5 Use case specification for "Get details of a location on a layer"

Use Case ID	UC33	**Use Case**	Get details of a location on a layer
Actor	User		
Preconditions	No		
Basic flow of events	**#**	**Doer**	**Action**
	1	Actor	Open Map screen
	2	Actor	Click on the layer button
	3	System	Prompt a list of supported layers
	4	Actor	Choose layers to display
	5	System	Call the API to get images of selected layers and display on the screen
	6	Actor	Click on a point of the layer to get details
	7	System	Call the API to get the details
	8	System	Display the details
Alternative flow	**#**	**Doer**	**Action**

	4a	Actor	Close the layer selection screen
Post-conditions	No		

Conclusion

This chapter analysed requirements for the application and clarified all specifications the application has to fulfill. The next chapter will present and explain technologies and frameworks adopted in the project in two categories: front-end and back-end.

Chapter 3 Adopted Technologies and Frameworks

In the last chapter, I have mentioned the requirements for the application and specified them in diagrams. I will explain the technologies and frameworks adopted in the process of developing this application in this chapter. Within this thesis, they can be classified into two groups: front-end and back-end technologies.

3.1 Front-end Technologies

3.1.1 React Native and its ecosystem

3.1.1.1 React Native

React Native is a framework released by Facebook in March 2015 with a view to allowing developers to write native applications for iOS and Android using Javascript [1]. It uses the same declarative approach to construct user interfaces as React for the web. These days there are a huge number of mobile applications developed with this framework which provide good performance and have received good feedbacks. Some of them which are published by Facebook are listed below in Figure 6.

The smallest logical unit in a React application is the component: a function that transforms input into a nested set of views rendered based on a set of parameters [2]. Components are composable and reusable. A component can consist of other components or be of primitive.

The data inside React components is managed by state and props. Props are used to customize components when they are created. They are set by the parent and can not be changed throughout the lifetime of the component, so props are read-only and immutable. However, in many cases, in addition to properties passed via props, components also need to keep track of some internal states. State can only be read by

the component, and can be changed to update visual display options generally; for example, a state parameter is needed to control whether the component is displayed or collapsed. Different from props, state is mutable.

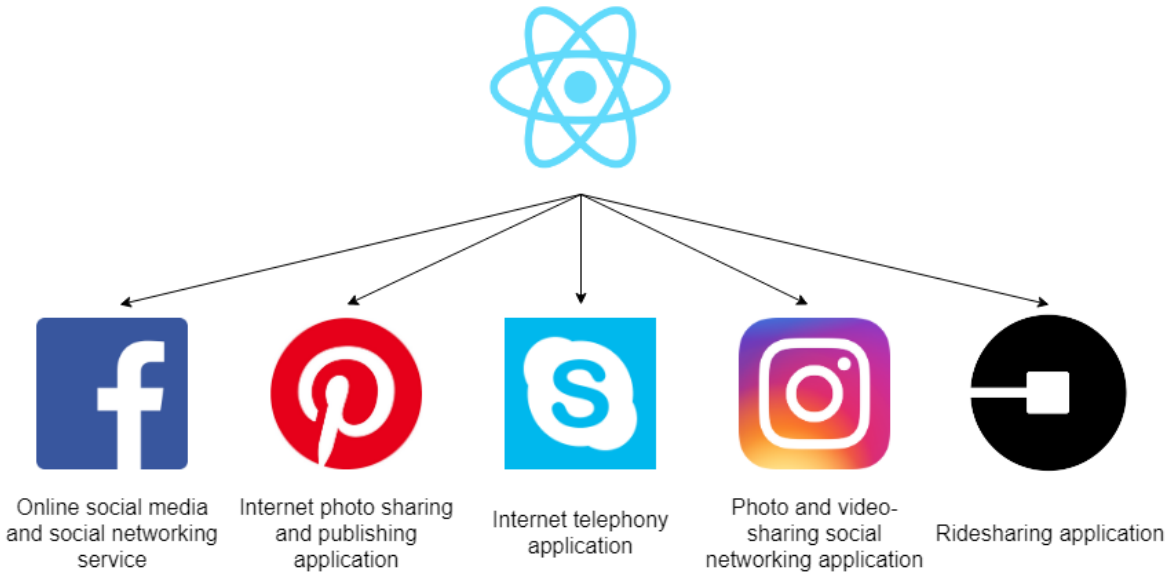

Figure 6 Widely-used applications developed with React Native.

Normally there are two important threads running in a React Native application. The first one is the main thread, which runs in every native application and in charge of displaying elements of the user interface and handle users' gestures. The other thread is specific to React Native. It is responsible for executing Javascript code in a separate Javascript engine, in other words, it deals with the business logic of the application. These two threads never communicate with each other directly; they do via a so-called bridge. It is asynchronous to assure that the two threads do not block each other.

Compared with other frameworks, React Native does stand out for many reasons. First, up to 90% of the codebase can be shared between Android and iOS; moreover, business logic code can also be shared with Web applications. It does not require developers to master native languages to create native applications. This helps saving a lot of developing time and human resources. Besides, the performance of applications developed with React Native is comparatively as good as native ones. These advantages will be explained in details below.

- Multi-platform applications

A native application is software program that is developed for a particulat platform in a certain language. Specifically, native applications for Android devices are written in Java, while native applications for iOS devices are developed in Objective-C/Swift. The most significant advantage of native apps is that they are capable of using device-specific hardware and software and latest technology for that platform. Therefore, their performance and flexibility can be optimized dramatically. Nevertheless, native applications have some undeniable disadvantages. Developing them is obviously costly and time-consuming. The same code can not be deployed on different platforms, and it requires skillful programmers in each platform and requires more time to rewrite the code. A certain application, furthermore, can be inconsistent among different platforms, which results in bad user experience and difficulty in maintenance.

To solve the cons of native applications, many frameworks for hybrid application development are released. The three frameworks that are predicted to become more widely used in 2019 are Ionic, Xamarin and React Native. The differences between them are listed in Table 6.

Table 6 Comparison among Ionic, Xamarin and React Native

Framework	**Ionic**	**Xamarin**	**React Native**
Language	TypeScript	C#	JavaScript
UI Rendering	HTML, CSS	Native UI Controllers	Native UI Controllers
Performance	Medium	High	High
Development time	Fast	Slow	Fast
Price	Opensource	Opensource	Opensource
Github Stars	35.3k	5k	69.3k
Community	Large	Medium	Large

Although rendering native UI, Xamarin framework takes more time for debugging and releasing applications. Those applications are largers, requires more storage than

native ones. Developers using Xamarin must use C#, which is considered to be rather complicated. For Ionic, the UI rendered on screens are created with HTML and CSS, which leads to the performance gap between Ionic apps and native apps with no doubt. Moreover, some native functions may not be available, and programmers have to develop them on their own. React Native was born overcoming the disadvatages of both Xamarin and Ionic.

Similar to Ionic and Xamarin, React Native only needs one codebase to run properly on both iOS and Android. Building blocks of React Native are reusable native components and compiled to native platforms, so it is not mandatory to use WebView. As a result, applications developed with React Native run as smoothly as native ones. Additionally, it has a huge community of developers as well as a large number of libraries, which can support almost any developers' requirement, making development process more convenient and less time-consuming.

- Development process

Not only in performance and user experience, React Native outweighs other frameworks in development process. Normally, the process of developing an application includes modifying codes, compiling and testing repeatedly, which takes a lot of time. React Native has been equipped with a special feature – hot reload. It allows developers to refresh the applications automatically while developing.

- Native Modules

React Native is a young framework, so there are some functions that have not been fully developed. However, since it is outsourcing, the developer community will contribute to complete them in a short time.

In addition, React Native is designed to allow developers to access mobile devices' hardware by native codes. When applications need to access APIs of device operating systems or optimizing performances and there are no React Native module to serve that, or when developers want to reuse some pieces of code written in Objective-C/Swift or Java/Kotlin within executing in Javascript, Native Modules is an advanced feature that can help developers do their job flexibly.

3.1.1.2 Redux

One of the most popular libraries used by React Native developers is Redux, a predictable state container for JavaScript apps. It helps writing applications that behave consistently, run properly in different environments. It is mostly used in React, but it can also be used in any other JavaScript frameworks or libraries without making applications' asset size bigger. As mentioned in 3.1.1.1, the key unit of a React Native application is the component, managed by props and state. In applications with a few components, React Native can control their state easily and simply without external support. However, when the application grows bigger and more complicated, managing states shared by component is a huge challenge. In React Native application, a state must live in the parent component to be shared among sibling components. A method for updating this state is provided by the parent and passed down to sibling components as props. It is a simple job if the parent and sibling components do not live far apart in the component tree. If they do, the state has to be passed from a component to another until it gets to the expected one. Moreover, in many cases, some components between the parent and that sibling do not need to use that state, which wastes time and memory. That is the main reason why Redux was born.

Redux can be described in three fundamental principles:

- Single source of truth.

The state of the whole application is stored in an object tree within a single store [5]. Any component has the right to access the store to get necessary state. That the state is stored in only one source avoids difficulties in tracing down any changes that cause errors and to maintain.

- State is read-only.

The state can only be changed by emiting an action. Actions are objects which describe what happens in the application. They are triggered by user interactions, API calls or form submission.

- Changes are made with pure functions.

To specify how the application's state is modified by actions, it is a must to use pure reducers. Reducers are pure functions that take the current state and an action, then return the next state. They do not change the previous state but return a new object – the next state. This can avoid unexpected changes in state.

Components connect with the store by method mapPropsToState, which allows they to access the state via their props. When users interact with the application, or an API is called, or a form is submitted, an action is initiated. It can contain a type and a payload. Once dispatched, it is received by a reducer. Depending on the action type, the reducer returns the next state of the application. Redux workflow is illustrated in Figure 7.

Supposed that each circle in Figure 8 represents a component in an application. The relationship among them is represented by arrows as in the figure. When a user interaction happens in component A in order to change the state of component B, without Redux, the data has to be passed from A back to D, then from D to B.

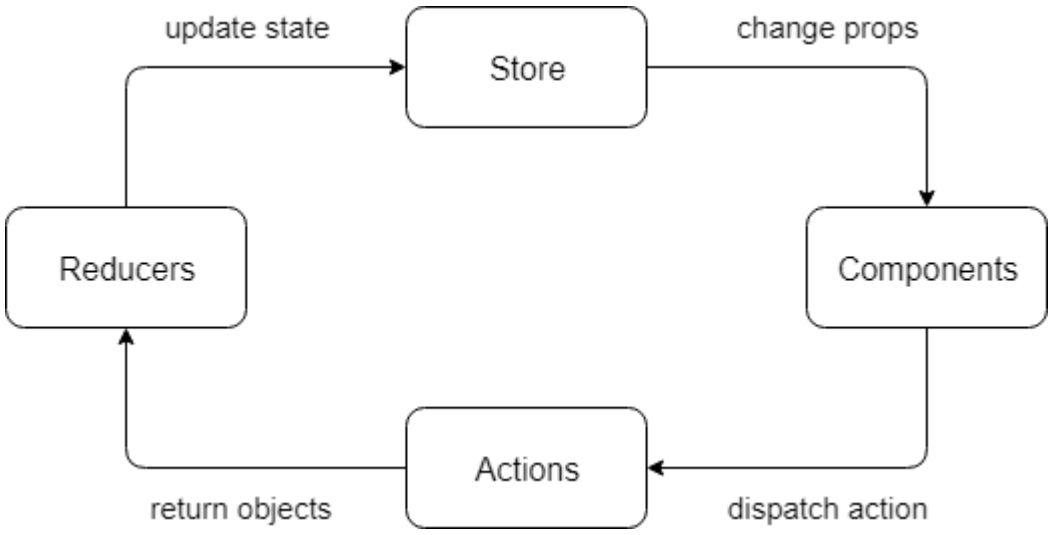

Figure 7 Redux workflow.

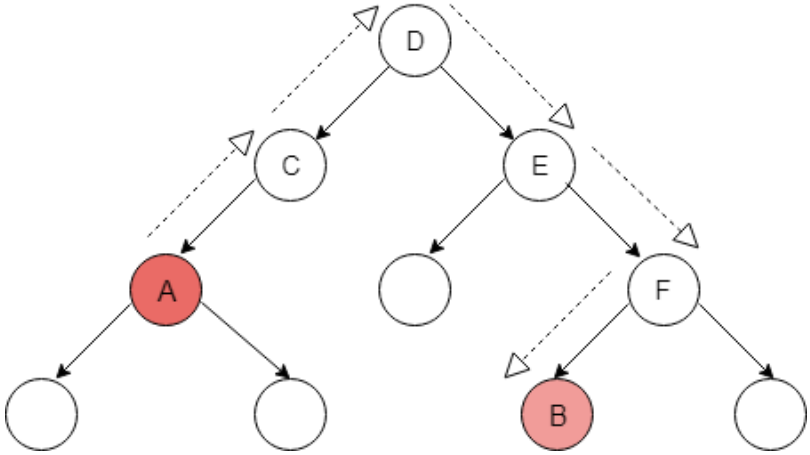

Figure 8 State management without Redux.

But thanks to Redux, component A and component B only need to connect to the store. An action is dispatched when the interaction happens at A, then a reducer is in charge of updating the state of the application. Component B can easily get the new state from the store and do its own job. Whenever the state changes, B detects it automatically and updates itself. Redux workflow in this example is illustrated in Figure 9.

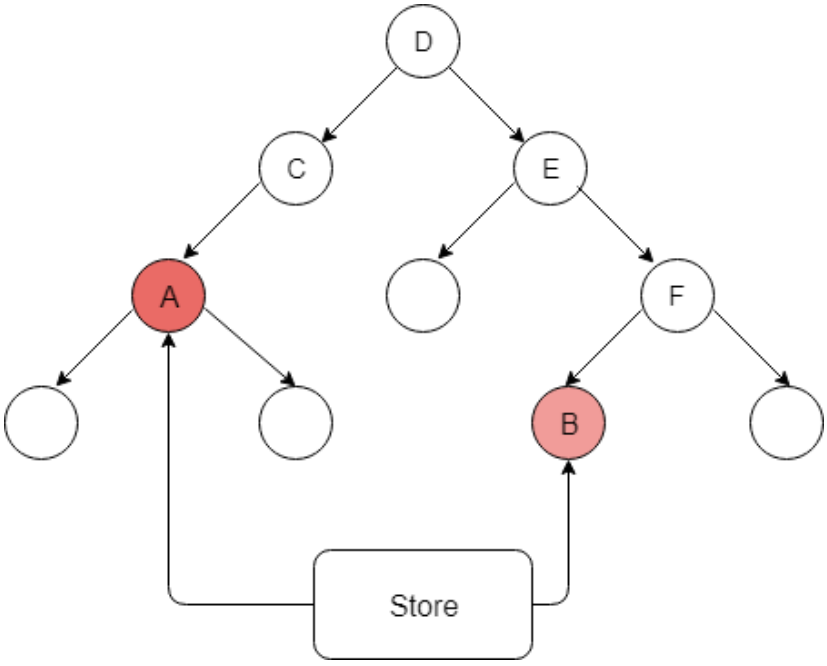

Figure 9 State management with Redux.

3.1.2 Web Map Services

Web Map Services (WMS) is a standard protocol that describes how to serve any georeferenced map images over the Internet. These images are typically produced by a map server that uses data from a geographic information system database [6]. The WMS protocol uses HTTP interface to make requests. A WMS request defines the geographic layers and area of interest to be processed. The response to the request is one or more geo-registered map images that can be displayed in a browser application without any special process. The interface also supports the ability to specify whether the returned images should be transparent so that layers from multiple servers can be combined to create overlayed maps that display more information. WMS in particular defines following operations:

- GetMap: request and provide a map as a picture or set of features.
- GetFeatureInfo: get and provide information about the content of a map such as the value of a feature at a location.
- GetCapabilities: get and provide information about what types of maps a server can deliver.

In Vadi Itrithuc, WMS is used to request and display layers on the maps. Thanks to the support of the library react-native-maps, it is much easier to work with WMS.

3.1.3 VNMap APIs

With a view to creating and sharing platform data address to each house number, each address nationally including administrative, school, cultural, medical and educational agencies, at the end of 2018, Vietnam Post Corporation was assigned to cooperate with the Central Ho Chi Minh Communist Youth Union and Hanoi National University to build and implement the project “Vietnam Digital Map Platform” [7].

This project is a part of the project “Vietnamese Digital Knowledge System” approved by the Prime Minister in order to create a comprehensive ecosystem for everyone, especially the young generation of Vietnam, to develop advanced technologies on the basis of big data, IoT, artificial intelligence.

"Vietnam Digital Map Platform" is created by Vietnamese, which is the basis for businesses to develop applications in many fields such as education, culture and tourism. Thanks to this, users can easily find and get specific directions to addresses in each lane, alley, village or commune.

3.2 Back-end Technologies

3.2.1 Node.js and Express.js

Javascript used to be known to only live inside web browsers. It started as a simple scripting language used to modify small details of web pages, then it grew into a complex language with a lot of applications and libraries. In 2009, Node.js was created by Ryan Dahl with a view to taking V8 – Google Chrome's powerful Javascript engine – out of the browser and enabling it to run on servers [3]. Node.js has simplified the process of developing applications for its ability to share code between browser and server, allowing developers to use the same code and coding paradigms instead of doing any kind of context switch. Furthermore, thanks to the fast V8 Javascript engine, Node.js encourages an asynchronous coding style, making for faster code while avoiding multithreaded disadvantages. However, Node.js only provides a bevy of low-level features needed to build an application, which led to the birth of Express framework.

Express is a framework that acts as a light layer atop the Node.js web server, making it more pleasant to develop Node.js web applications [3]. Some of its core features are to allow to set up middlewares to respond to HTTP requests, to allow to dynamically render HTML pages based on passing arguments to templates, and to define a routing table which is used to perform different actions based on HTTP method and url. Owing to Express, it is much easier and faster to create APIs.

3.2.2 REST and RESTful API

REST is acronym for Representational State Transfer. It is architectural style for distributed hypermedia systems and was first presented by Roy Fielding in 2000. It defines six constraints that all interfaces must follow to be referred as RESTful:

client-server, stateless, cacheable, uniform interface, layered system, code on demand [4].

RESTful APIs are obviously APIs that are satisfied the mentioned six constraints, which are widely used whenever it comes to communication between front-end and back-end. One advantage of them is the scalability thanks to the separation between client and server. Furthermore, HTTP requests may include data and parameters in its param, in query strings or in requests' body.

3.2.3 MongoDB

MongoDB is an open source document-oriented NoSQL database. Instead of storing data in a relational type format, it stores the data in documents, which makes it possible to represent complex hierarchical relationships easily. MongoDB outweighs other types of database for its high performance, high availability and automatic scaling.

Conclusion

In this chapter, I have explained all the technologies and frameworks adopted in this project. For front-end layer, I used React Native to develop multi-platform mobile application and Redux to manage the application state. Additionally, I used Web Map Services to communicate with the layer managing server developed by VNMap, and VNMap APIs to control user interactions with maps. In back-end layer, Node.js and Express.js were used to quickly and easily develop server in Javascript. Services communicate with others via RESTful APIs, and data of the application is stored in with MongoDB.

Chapter 4 Application Development and Deployment

In the third chapter, I have explained the technologies and frameworks adopted in the application. In this chapter, the process of developing and deploying the application will be described in details. It is divided into five parts: Architectures, Design, Development, Testing and Deployment.

4.1 General architecture

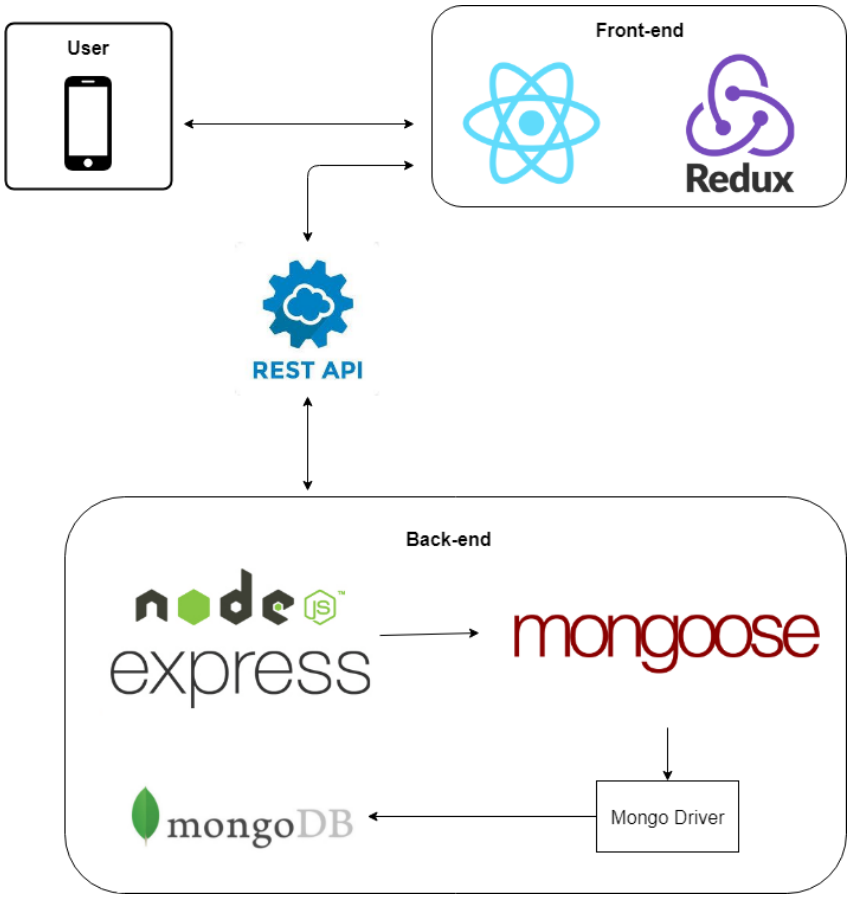

Figure 10 General architecture.

The general architecture of the application is illustrated in Figure 10. It can be seen from the figure that the system is divided into two parts: front-end (client-side) and back-end (server-side).

4.1.1 Front-end general architecture

Front-end is the interface of the system, where users interact with the system. It is developed with React Native, a framework for mobile application development as mentioned in 3.1.1.1. Redux is also utilized to manage the application state in a simple and effective way.

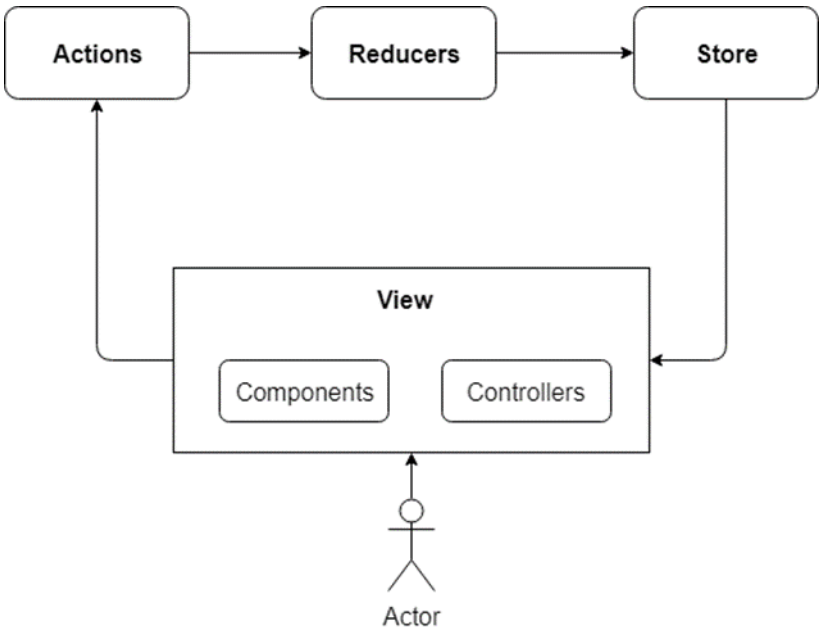

Figure 11 Front-end general archirecture.

Figure 11 presents the general architecture of front-end layer. It can be seen from the figure that there are four main components: views, actions, reducers and store. The relationships between them are illustrated by arrows in the figure.

4.1.2 Back-end general architecture

Back-end is organized in microservice architecture, which will be discussed further in next section. As described in the figure, the server is developed with Node.js and Express, and the data is saved in MongoDB by using Mongoose module to communicate with Mongo driver.

Many applications nowadays are built in monolithic architecture or in other words, they are built as a single unit. These applications generally have three main parts: the user interface, the data access layer, and the data store. The first and second ones are usually combined together into a single program on a single platform, tightly coupled together and depend on each other in order to run properly. In spite of the fact that this architecture is simple to develop and test, it is challenging for developers to maintain, scale vertically and adopt new technologies. Meanwhile, microservice architecture allows applications to have a suite of loosely coupled services, each supports a specific business goal and use a simple, well-defined interface to communicate with other services. An example for monolithic and microservice architecture is illustrated in Figure 12.

The most significant advantage of microservice architecture is that it is easier to detect any errors and maintain the whole system since failure in one service hardly impact other services. Furthermore, scalability would be better because demanding services can be deployed in other servers to enhance performance, which is a real challenge in monolithic architecture. It is also easier for developers to deploy, adopt new technologies as well as modify any built-in functions.

In Vadi Itrithuc, back-end blocks have been developed in microservice architecture. There are four main services as specified in Figure 13: (i) user service providing registration and login functions, (ii) news service providing news-related functions, (iii) maps service providing traffic warning and user contribution functions, (iv) smartdialog service supporting voice commands in the application.

Each of those four services is capable of running independently in different servers. They can communicate with front-end layer or with others easily via APIs.

Thanks to microservice architecture, the maintenance, error detection or scaling processes become much more predictable and simpler.

The elements that were developed within this thesis is emphasized in the following figure.

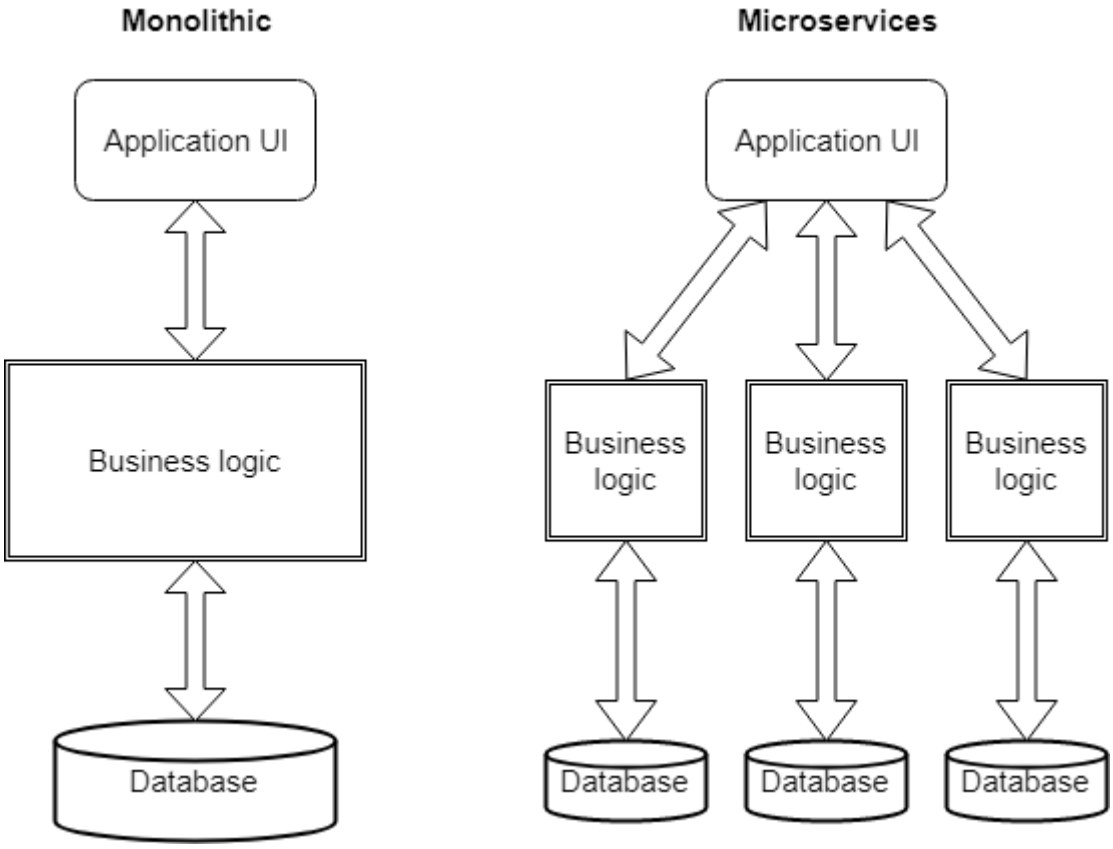

Figure 12 Monolithic architecture vs. Microservice architecture.

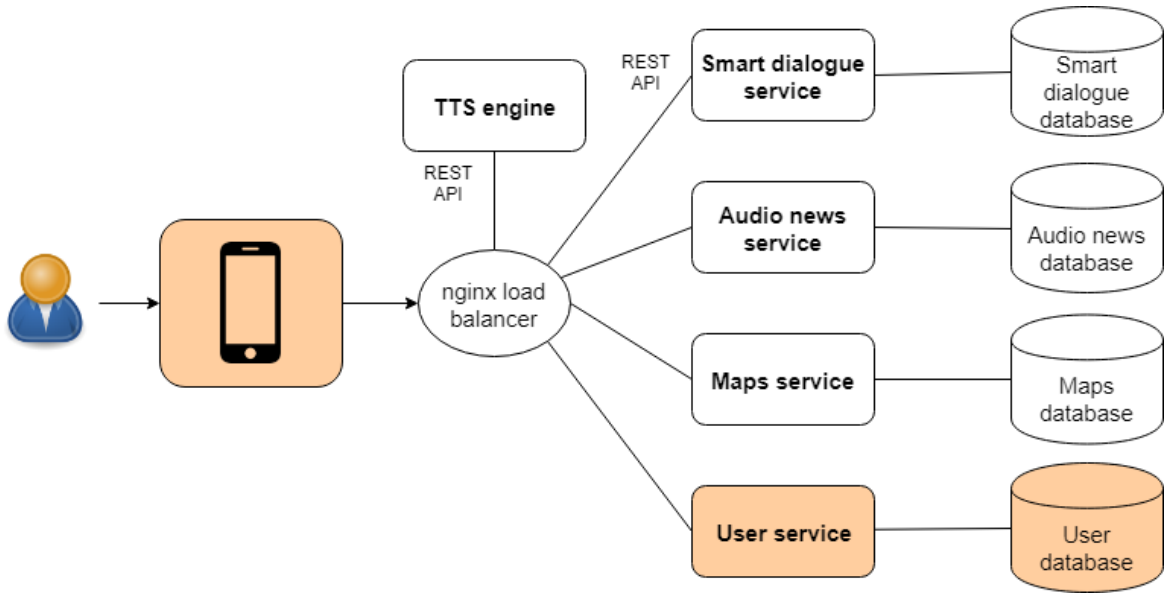

Figure 13 Microservice architecture in Vadi Itrithuc.

4.2 Architecture design

4.2.1 Use case design

4.2.1.1 Use case "Add a pair of text - voice"

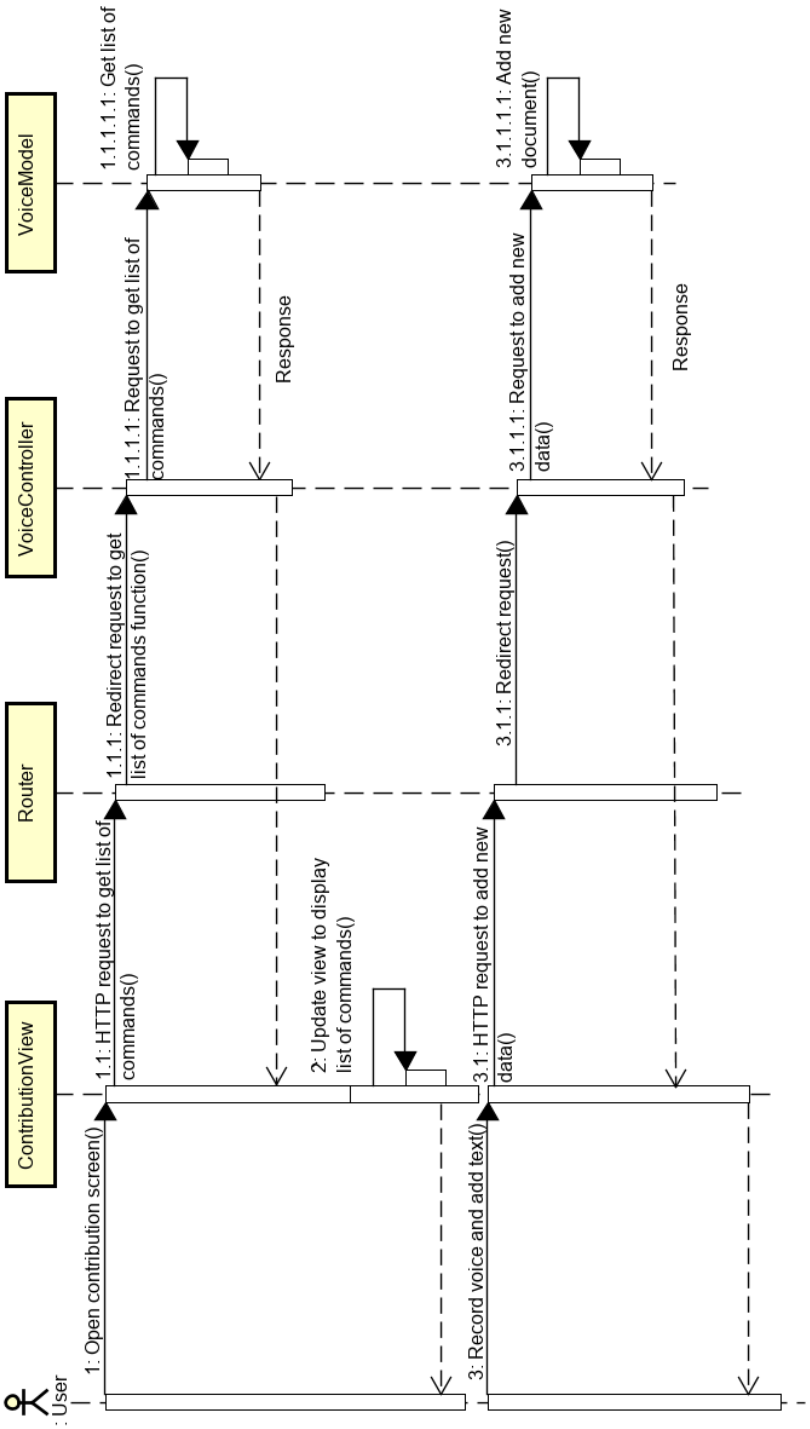

Figure 14 Sequence diagram for "Add a pair of text - voice".

This use case is one of the main functions of the system. It is illustrated by a sequence diagram in Figure 14.

Firstly, user opens the contribution screen by expanding menu tab. This view will send a HTTP request to get list of commands. The router will redirect the request to the corresponding controller, specifically VoiceController. This controller is in charge of communicating with VoiceModal to get list of commands and return it to the View, which will display the list to user.

After that, user will contribute by recording their voice and edit the text if necessary. Contribution View will send a HTTP request again, but to add a new document. The rest of the process is similar to the first step, but the VoiceModal will add a new document instead.

4.2.1.2 Use case "Search places by voice"

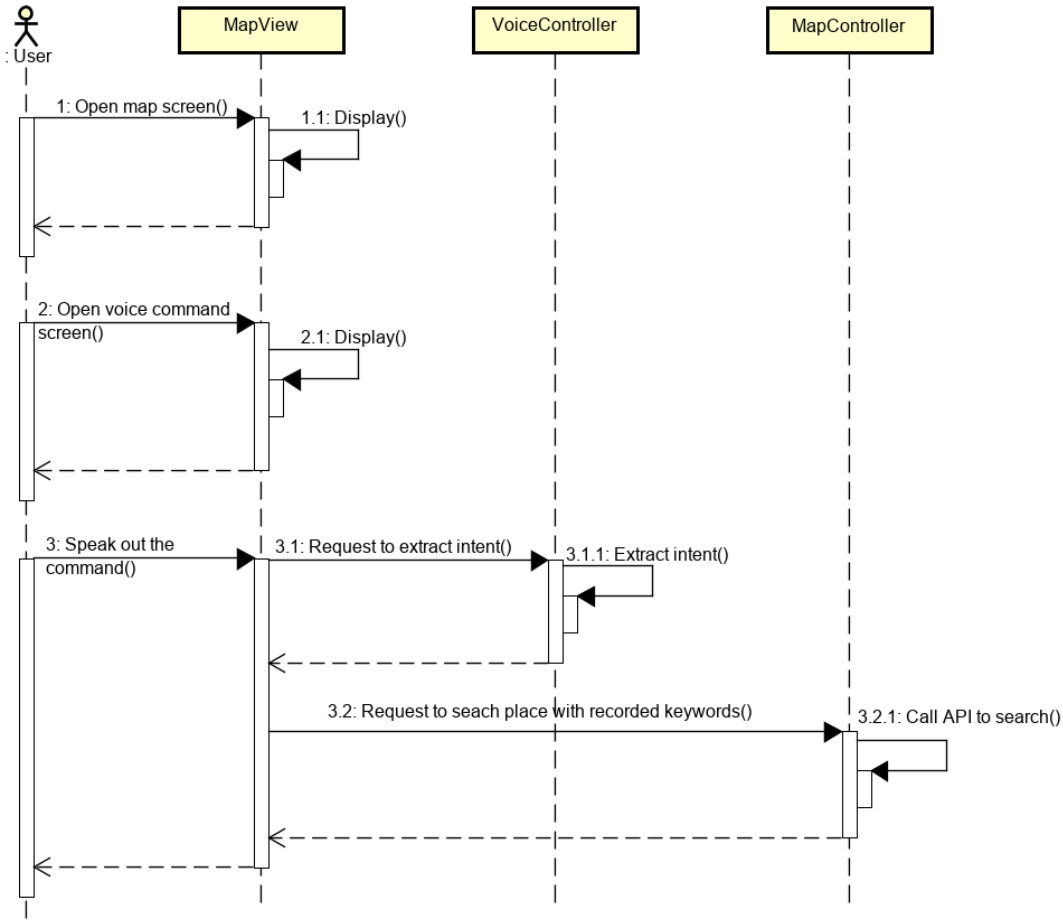

Figure 15 Sequence diagram for "Search places by voice".

Figure 15 illustrates use case “Search places by voice” by a sequence diagram. It can be seen from the figure that there are three big steps.

First, user opens the map screens, which is in charge of displaying maps.

Next, user opens voice command screen and it will be displayed.

Then user speaks out the command, it is recorded and sent to VoiceController to extract the user intent. The intent is returned to MapView, which will request MapController to search places by keywords extracted from the recorded voice. MapController invokes some API and sends the result back to MapView and back to user.

4.2.1.3 Use case “Play previous audio news by touching”

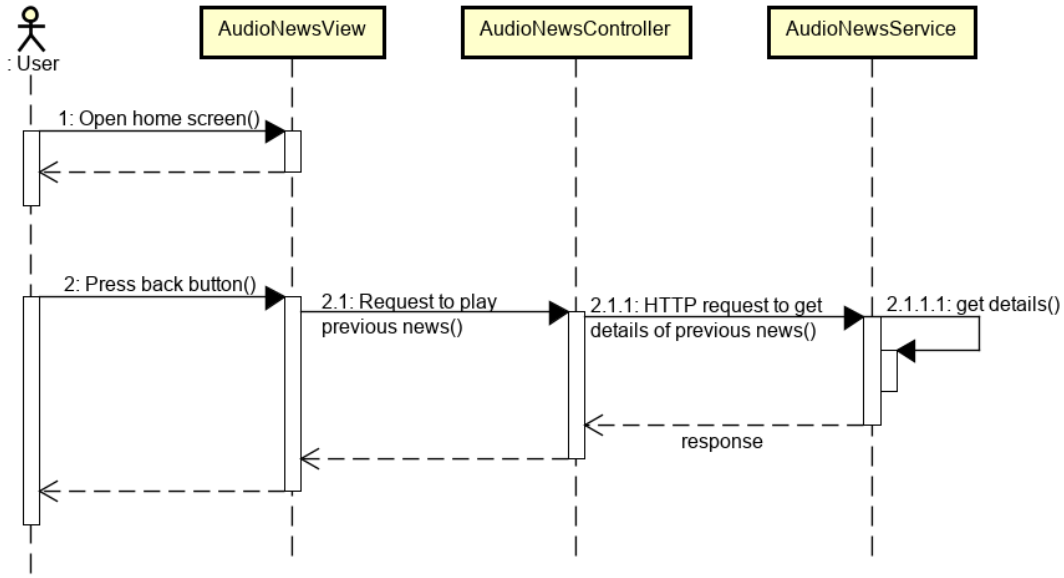

Figure 16 Sequence diagram for “Play previous audio news by touching”.

Use case “Play previous audio news by touching” is specified by a sequence diagram in **Figure 16**. Firstly, user opens home screen where the media controller is located. Then user presses the back button. AudioNewsView will send a request to play previous news to AudioNewsController, which will communicate with AudioNewsService via HTTP requests to get details of the previous new, then return it back to AudioNewsView. The audio will be played as a response to user.

4.2.2 Component design

4.2.2.1 Front-end component design

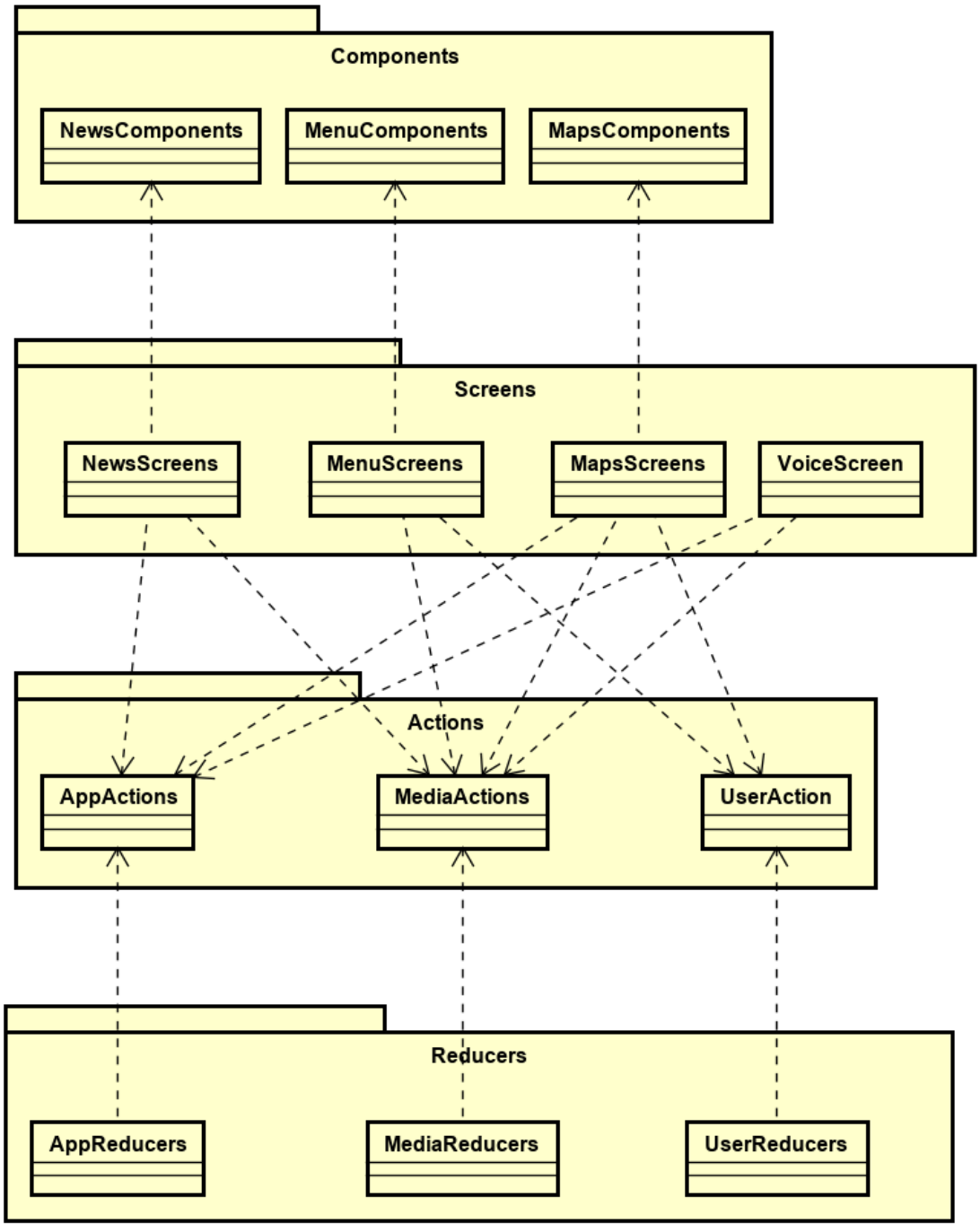

Figure 17 Front-end component diagram.

4.2.2.2 Back-end component design

As I mentioned in Architecture design section, the application is organized as microservices. Since I only developed User Service, I will explain it in this section.

When receiving an HTTP request from front-end layer, the router will redirect it to the corresponding controller. The controller will communicate with the model to execute expected actions.

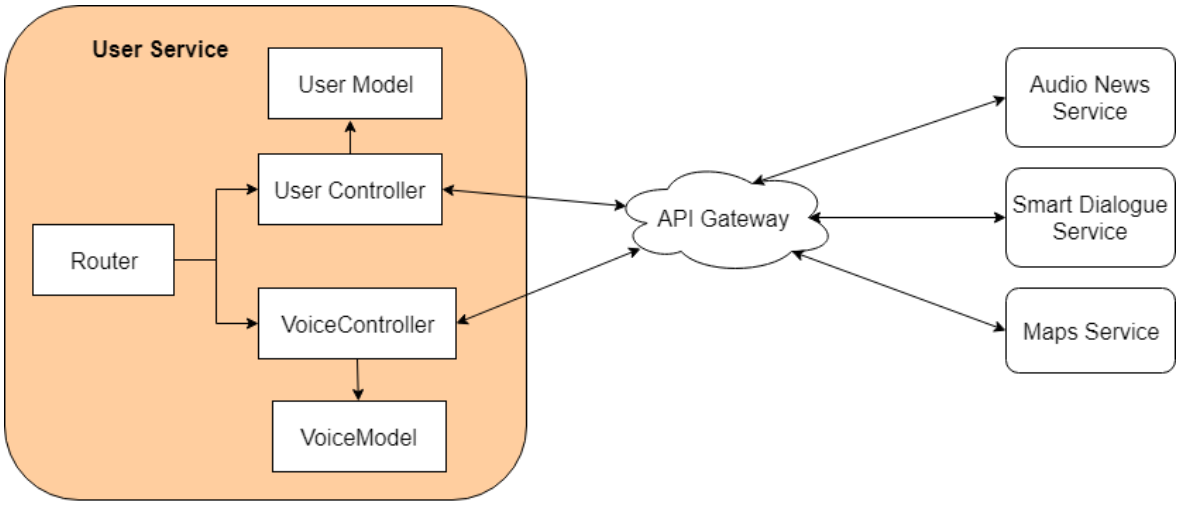

Figure 18 Component diagram of back-end layer.

4.3 Detail design

4.3.1 User interface design

The application is design with a view to optimizing user experience, avoid hand interactions, especially when users are driving. The document W3C UAAG 2.0 was used for references. Specifically, transitions between screens are simplified by wiping, placeholders are used to optimize the display area on screens, input components change their colours for users to notice easily.

Figure 19 is the screen transition diagram of the application. When user first open the app, the introduction screen is prompted, then they are directed to Home Screen, where users can access Menu, Map and Voice Command Screen. Other screens and their relationships are described with types of arrows as in the figure.

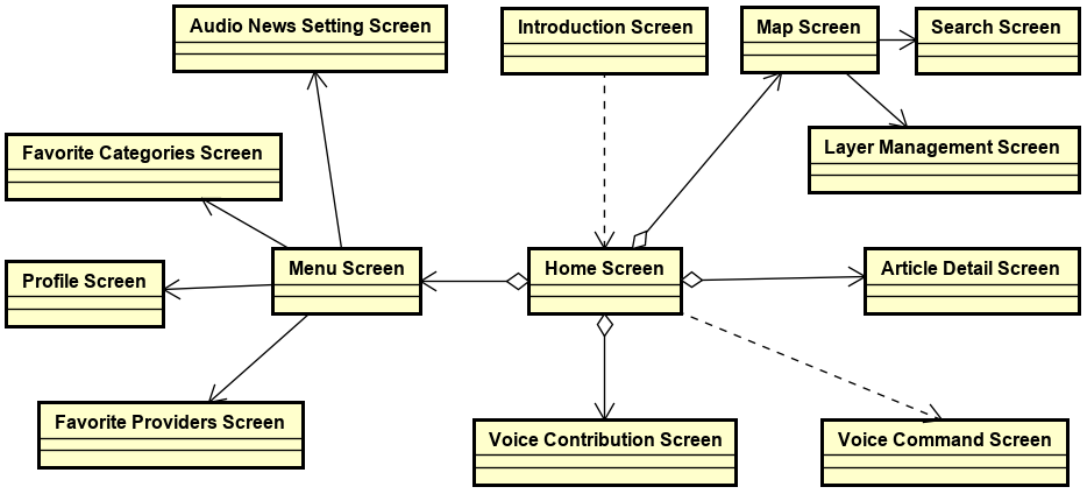

Figure 19 Screen transition diagram.

Figure 20 illustrates the component flow of HomeScreen components. Header and Footer can be modularized into components with data passed via props. MediaComponent is inside TabFooter to be always visible to users.

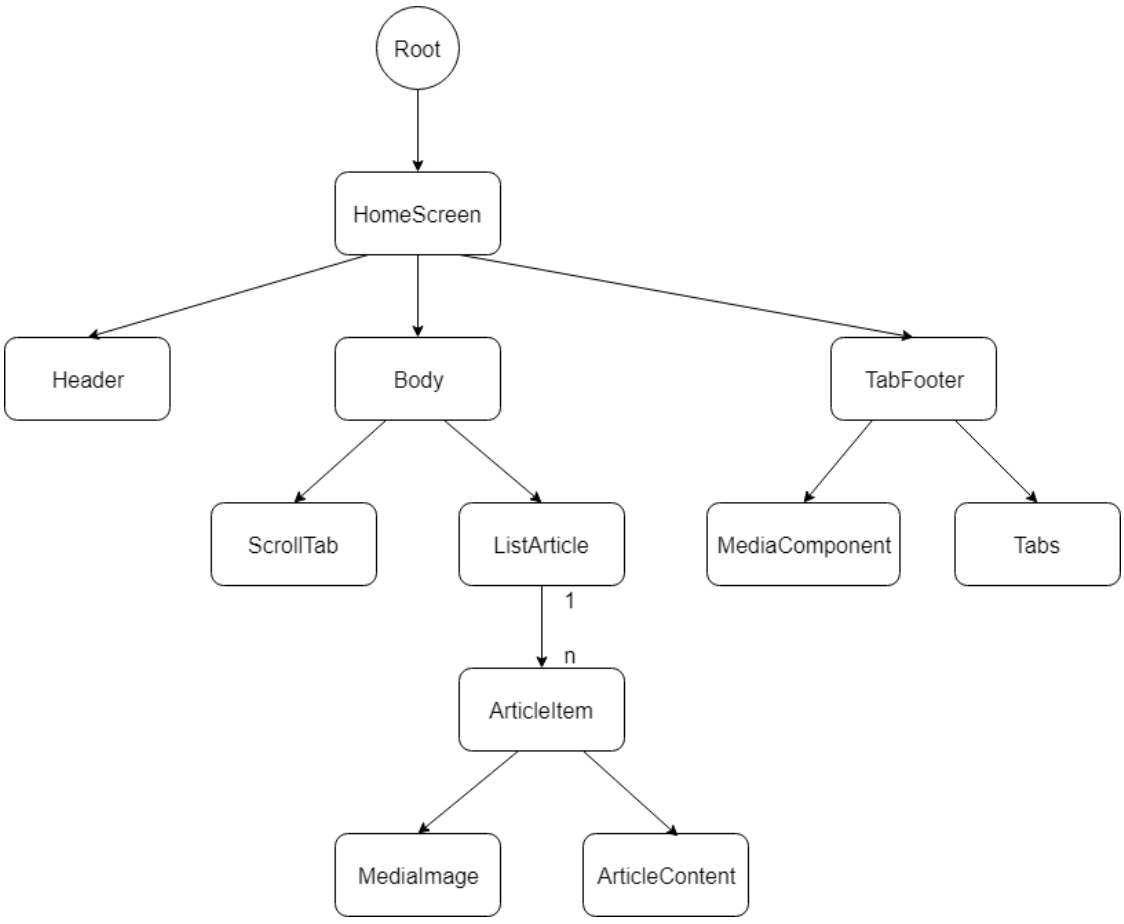

Figure 20 Component flow diagram of HomeScreen.

Component Body is divided into two sub-components: ScrollTab to display list of news categories and ArticleItem to display basic information of each article.

Additionally, to avoid eye strain for users, the application has two display modes: night and day as in Figure 21.

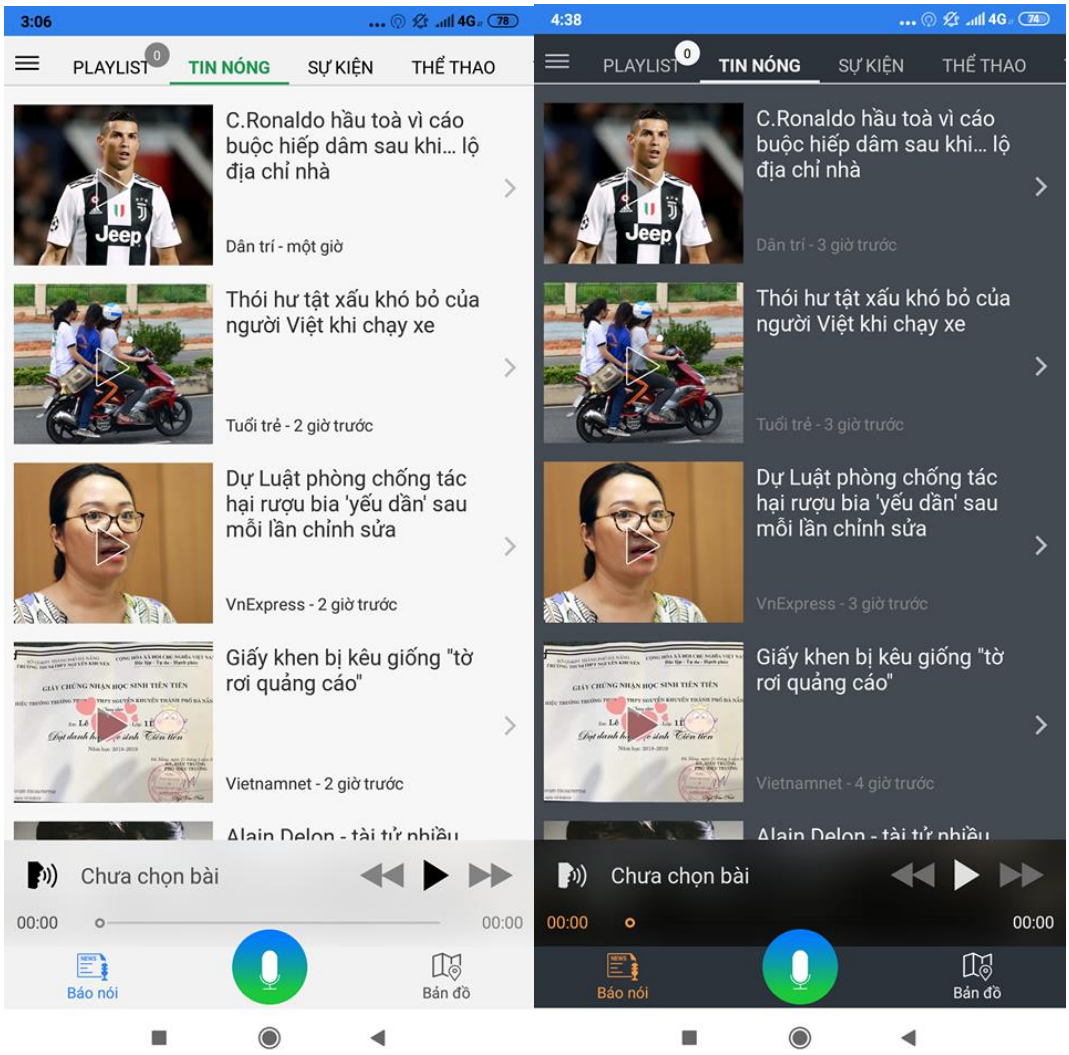

Figure 21 Day and night display modes.

The application is for all the community, so it is necessary for it to be responsive to a large number of mobile devices and to two main mobile operating systems iOS and Android. However, devices with large screens such as tablets have not been supported.

4.3.2 Database design

4.3.2.1 Front-end database

In front-end layer, data is stored in devices' storage via React Native's AsyncStorage API. AsyncStorage is a simple, unencrypted, consistent, asynchronous key-value storage system that allows access from all parts of applications. Place search histories storing design is described in **Table 7**.

Table 7 Place search history storing design

Key	**Value**		
Histories	MapHistory		
	Attribute	**Type**	**Description**
	place_id	Number	Identifier of the place
	osm_id	Number	Identifier of the place in OSM system
	osm_type	String	Type of the place in OSM system (can be "N", "W", "R")
	title	String	Name of the place
	description	String	Description of the place in details
	coord	Object	Coordinates of the place (consists of two attributes: longitude, latitude)
	type	String	Can be "SEARCH" – searched by text, or "GEO" – search by a point on the map.

4.3.2.2 Back-end database

As mentioned in above sections, the back-end layer of the application has four services: audio news, smart dialogue, maps and user services. However, I have only been in charge of developing user services, so I will discuss about user service in this section.

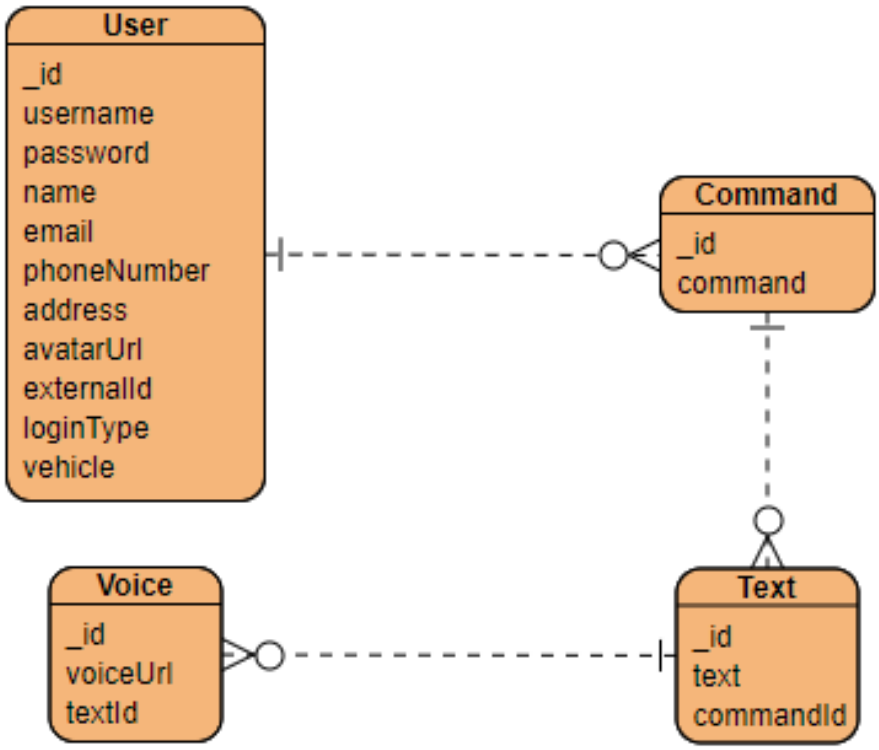

Figure 22 E-R Diagram.

In back-end layer, data is stored in MongoDB as mentioned in 3.2.3. The structure of the database is illustrated in Figure 22.

There are four main collections: User, Command, Text and Voice. Each user can have many commands. Each command is specified by many texts that can be contributed by users. Each of those text can be transformed to many voices.

The user collection is specified in Table 8. It can be seen from the table that a user document contains 11 attributes. Besides attributes to store basic information of the user such as username and password, loginType is used to save the way user login to the system: with Facebook, Google or Vadi account. User's vehicle is also stored to produce the best routing algorithm for that kind of vehicle.

Table 8 User collection specification

Attribute	Type	Description
_id	String	Id of the user
username	String	Username of the user. Must be unique.
password	String	Password of the user
name	String	Display name of the user
email	String	Email of the user
phoneNumber	String	Phone number of the user
address	String	Address of the user
avatarUrl	String	Url of the profile picture to display in the app
externalId	String	Id of the user logging in by Facebook or Google
loginType	String	Login type of the user. Can be “facebook”, “google” or “personal”
vehicle	String	Vehicle of the user

4.4 Construction

4.4.1 Libraries and tools

In the process of developing Vadi Itrithuc, I have taken advantages of many different tools, libraries, APIs, languages and debuggers. They will be listed specifically in Table 9.

Table 9 List of adopted tools and libraries

Purpose	Tool	URL
IDE	Visual Studio Code 1.33.1	https://code.visualstudio.com/
Debugger	React Native Debugger 0.9.6	https://github.com/jhen0409/react-native-debugger
Build/Deploy tool	Android Studio 3.0	https://developer.android.com/studio

Programming language	React Native 0.57.8	https://facebook.github.io/react-native/
	React 16.6.0	https://reactjs.org/
	Node.js 8.12.0	https://nodejs.org/en/

In addition to tools, I have utilized some libraries which are listed in Table 10.

Table 10 Third-party libraries

Library	Version	Purpose
react-native-maps	0.24.2	Display and control maps view
react-native-call-detection	1.6.2	Detect incoming and outcoming calls on devices
react-native-fbsdk	0.8.0	Provide Facebook functions such as Login and Share
react-native-google-signin	0.12.0	Log in with Google Services
react-native-music-control	0.7.3	Manage and control audio player when running in background
react-native-video	4.0.0	Manage video and audio playing on devices
react-native-vector-icons	4.5.0	Provide icons under the form of vectors
react-navigation	1.6.1	Manage screens and screen transitions
redux	3.7.2	Manage application state
react-redux	5.0.7	Connect React components with Redux store
react-native-scrollable-tab-view	0.10.0	Provide tab scrollable components

Table 11 presents the list of mapp APIs provided by VNMap.

Table 11 List of VNMap APIs

Purpose	Method	URL
Search places by keywords	GET	http://map.itrithuc.vn/geocode/search?q=${keywords}&format=json
Generates an address from a latitude and longitude	GET	http://map.itrithuc.vn/geocode/reverse?lat=${latitude}&lon=${longitude}&format=json
Look up details about a single place by its osm id and osm type	GET	http://map.itrithuc.vn/geocode/details?osmid=${osmId}&osmtype=${osmType}&format=json
Routing	GET	http://map.itrithuc.vn/routing/route?point=${startingPointCoords}&point=${destinationCoords}

Additionally, there are some APIs to handle user login and registration, and to get information from audio news service. Table 12 introduces some APIs utilized in Vadi Itrithuc.

Table 12 User and audio news APIs

Purpose	Method	URL
Login	POST	api/v1/users/signin
Register	POST	api/v1/users/signup
Get hot news list	GET	api/v1/users/${userID}/articles/hotnews?page=${page}&size=${size}&fields=${fields}
Get news details by id	GET	api/v1/users/${userID}/articles/${articleID}?fields=$${fields}
Get news data by category id	GET	api/v1/users/${userID}/categories/${categoryID}/articles?page=${page}&size=${size}&fields=${fields}

4.4.2 Source code structure

Vadi Itrithuc is developed with React Native framework and Redux library, so the project's sourcecode structure is organized as in Figure 23.

- Actions

  [1] This folder contains action declarations and middlewares to handle data asynchronously before passing to reducers.

- Components

  [2] This folder contains components that can be reused many times within the application (Header, Button, …).

- Config

  [3] It contains configuration files used to configure the parameters of the application such as API urls, color codes.

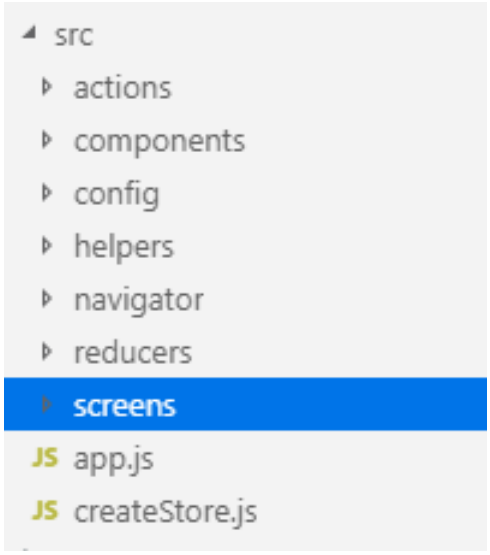

Figure 23 Souce code structure of the project.

- Helpers

  [4] Functions which support data handling and calculation are defined in this folder.

- Navigator

  [5] Files in this folder are in charged of managing navigation between screens.

- Reducers

[6] It contains Redux reducers, taking an action and the current state, returning the next state of the application.

- Screens

  [7] Main screens of the application are defined here. It includes five subfolders: article, login, maps, menu, voice.

The deployed application contains source codes and assets. For Android operating system, the application is under the form of .apk file. For iOS operating system, it is under the form of .ipa. The application specifications are described in Table 13.

Table 13 Application specifications

Specification	Value
Source code size	350 MB
Application size	~ 16 MB

4.4.3 Main screens and features

4.4.3.1 Audio news

Figure 24 is screenshots of audio news home screen and news detail screen.

Home screen is designed as a top scrollable tab view, each tab presents audio news in different categories such as Hot News, Sports, Events, Economy. Audio news is arranged in a vertical list. Each article contains an image, a title, its source and release time. For example, in the figure, the audio news being played is in Hot News tab, posted on Dân Trí website 8 minutes ago.

The news detail screen is a web view that displays exactly how the news is displayed on its source website. Users can swipe right or left to read next or previous news, and press the big play button on the screen to play the news they are reading. They can also share news to their Facebook timelines.

About the bottom tab bar, it has three main buttons: one for audio news, one for maps and the middle one for voice commands. It is designed to stand out for users to easily notice and interact with.

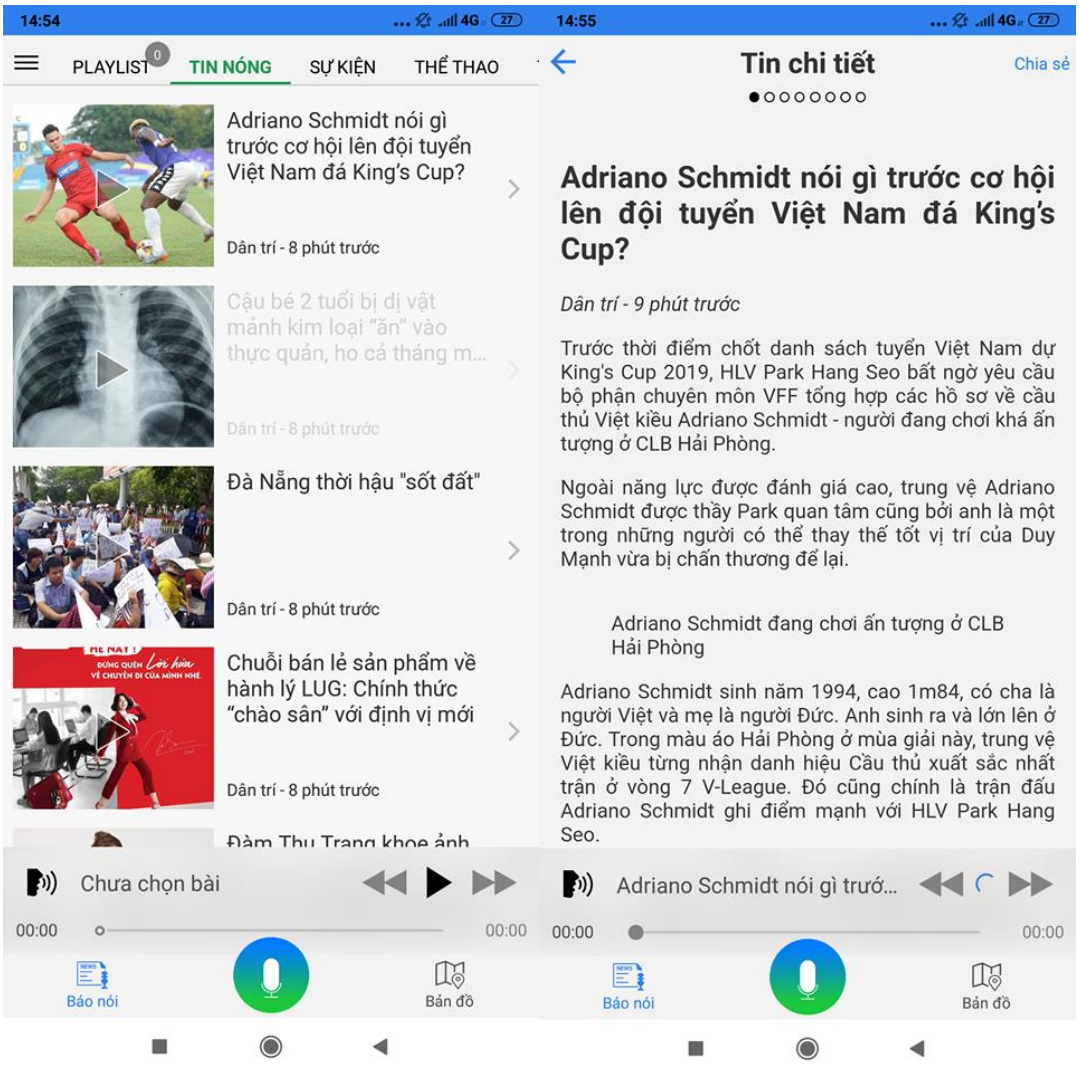

Figure 24 Audio news screens.

4.4.3.2 Maps

Figure 25 is two pictures of map screen.

The first picture is what users see when they press the "Bản đồ" button. The screen contains VNMap base maps, a button to manage layers of the maps and a button to centralize the users' current position. At the top of the screen, there is a search box where users type in keywords to search places and routes.

The second figure is what displays when users press the search box. Once keywords are typed in, a suggestion list will be prompted. Users can choose the places they want by press on it, or press “Chọn vị trí trên bản đồ” to search places by positions.

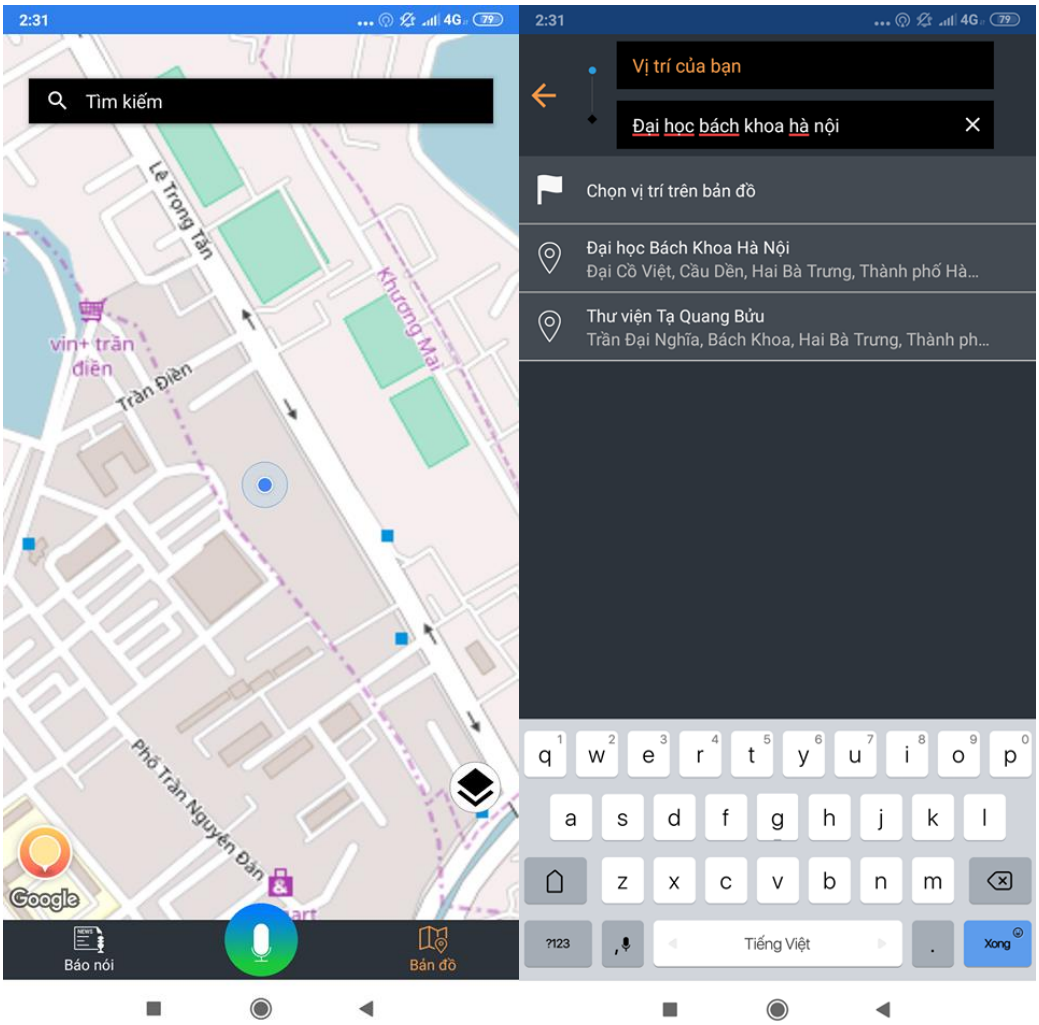

Figure 25 Maps screens.

Another function of the application is to manage layers on the maps. As mentioned above, there is a layer button on top of the maps where users can click to control layers. Once the button is pressed, a modal is prompted, where a list of supported layers is displayed. The modal has two buttons: “Bỏ chọn tất cả” to deselect all listed layers, and “Hoàn tất” to send requests to VNMap server and present expected layers.

The right figure is what users see when they finish choosing layers. Places in the requested layers are marked with big red icons. This helps avoid confusing users

between markers and places. Only places within the screen are loaded. Only when users drag the maps are places in the new region is loaded.

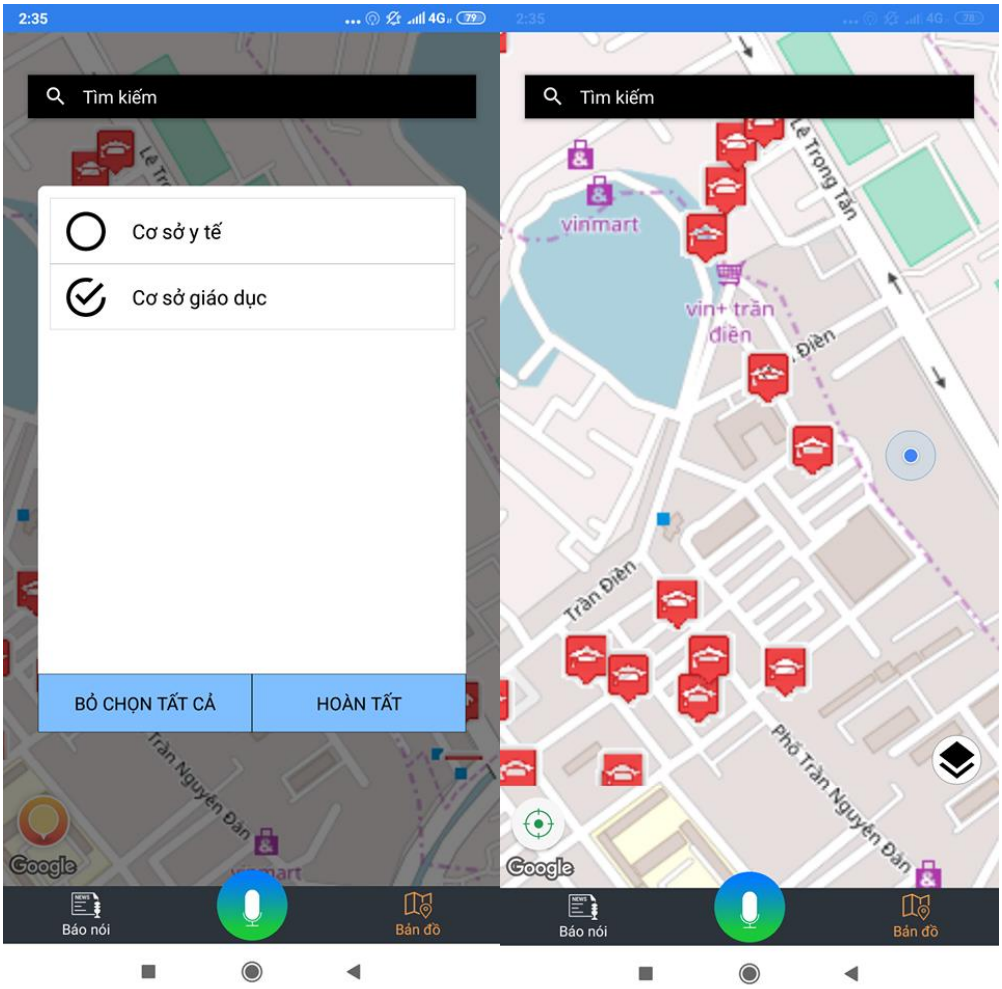

Figure 26 Layer screens.

4.4.3.3 Voice commands

Figure 27 presents the voice command screen. This screen is accessed by press the big middle button on the tab bar. It has black background and white text to assure that users can see clearly what they have ordered the application, especially for driving users. When users speak out their commands, the text is sent to a service to extract users' intent. For example, in the figure, user asked the application to search for routes to Presidential Palace. After receiving the intent from NER service, the system used some API to get the suggestion list and considered the first place in the list is the expected one. Then routes are prompted on the screen as in the figure.

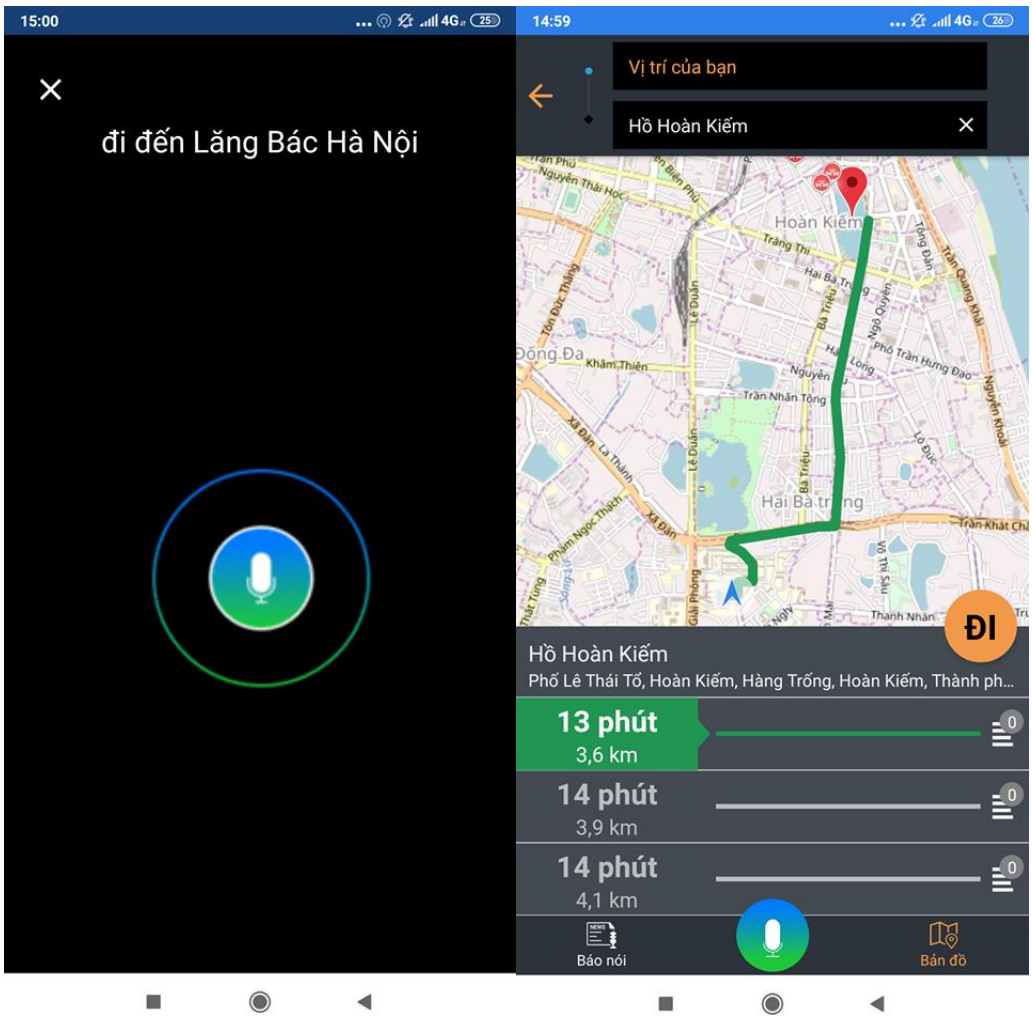

Figure 27 Voice command screens.

4.5 Testing

4.5.1 Test for “Play audio news by voice” feature

Table 14 Test for “Play audio news by voice” feature

Test case	Steps	Expected result	Test result

Play next audio news by voice	Open the application Log in with Facebook account Open voice command screen Speak out “Bắt đầu phát báo”	Display Introduction screen Display news home screen Display voice command screen Start playing the first news in tab Hot News	PASS
	Open the application Open voice command screen	Display Introduction screen Prompt “Bạn cần đăng nhập để sử dụng tính năng này”	PASS

4.5.2 Test for “Control app by voices” feature

Table 15 Test for “Control app by voices” feature

Test case	Steps	Expected result	Test result
Control app by voice	Open the application Log in with Google account Open voice command screen Speak out “Bắt đầu phát báo”	Display Introduction screen Display news home screen Display voice command screen Start playing the first news in tab Hot News	PASS

	Open the application Open voice command screen	Display Introduction screen Prompt “Bạn cần đăng nhập để sử dụng tính năng này”	PASS
	Open the application Log in with Google account Open voice command screen Speak out “Hôm nay trời nắng”	Display Introduction screen Display news home screen Display voice command screen Prompt “Không nhận dạng được lệnh”	PASS

4.5.3 Test for “Manage map layers” feature

Table 16 Test for “Manage map layers” feature

Test case	Steps	Expected result	Test result
Manage map layers	Open the application Log in with Google account Press maps tab Press layer button Select one arbitrary layer and press “Hoàn tất” Press layer button Select another arbitrary layer and press “Hoàn tất” Press layer button Deselect all layers and press “Hoàn tất”	Display Introduction screen Display news home screen Display maps screen Prompt a modal with list of supported layers Display base maps with the corresponding layer Prompt a modal with list of supported layers Display base maps with another layer Prompt a modal with list of supported layers Display only base maps	PASS

	Open the application Press maps tab Press layer button	Display Introduction screen Display news home screen Display maps screen Prompt “Bạn cần phải đăng nhập để sử dụng tính năng này”	PASS
	Open the application Log in with Google account Press maps tab Press layer button Select one arbitrary layer and press “Hoàn tất” Disconnect to the Internet Press layer button	Display Introduction screen Display news home screen Display maps screen Prompt a modal with list of supported layers Display base maps with the corresponding layer Prompt “Bạn cần phải kết nối với Internet”	PASS

4.6 Deployment

After development time, Vadi Itrithuc has been only deployed in Google Play for Android devices. It will soon be deployed in App Store. Users can access the store or access https://play.google.com/store/apps/details?id=com.vadifreeitrithuc&hl=en to install and experience the application.

Chapter 5 Main Contributions

In chapter 4, I introduced the process of developing and deployment the application. I explained its architecture in both front-end and back-end layers. Use case and database design were also clarified. In this chapter, I will present my main contributions to the application development process.

5.1 Voice command

5.1.1 Problem

Virtual assistants are becoming a trend for mobile devices such as Siri on iOS and Google Assistant on Android. They obviously have undeniable advantages compared with hand interaction, especially in mobile applications for drivers. The highest priority for them is to ensure safety while driving, but traditional interaction requires drivers to look at the screen and touch by hands. Virtual assistants allow users to control applications by voice commands and get responses under the form of voice. This can decrease hand interaction significantly and improve user experience. Vadi Itrithuc is not a complete virtual assistant in the present since it has not been equipped to respond to users in voices. However, it supports some basic commands for more convenient and easier usage.

Because Vadi Itrithuc has a large codebase, it was rather challenging to integrating voice command into the application. Each supported command is related to a different component. Passing data between components traditionally may cause the data flow in the application to get messy. Therefore, it is necessary to make data consistent and accessible among all components.

5.1.2 Solution

The process of record and handle voice commands is illustrated in Figure 28. When user opens voice command screen, audio record permission is checked. If it is not accessible, setting screen is prompted and the user will turn on the permission. Then, the system checks if there is Internet connection. If not, voice command screen will close. Otherwise, the user is allowed to speak out the command. Module react-native-voice will recognize the text, which is displayed on the screen immediately. The text

will then be sent to NER service where the intent of the user is extracted. A response including that intent will be sent back to the system, then the desired command is executed. However, how commands are executed is a challenge.

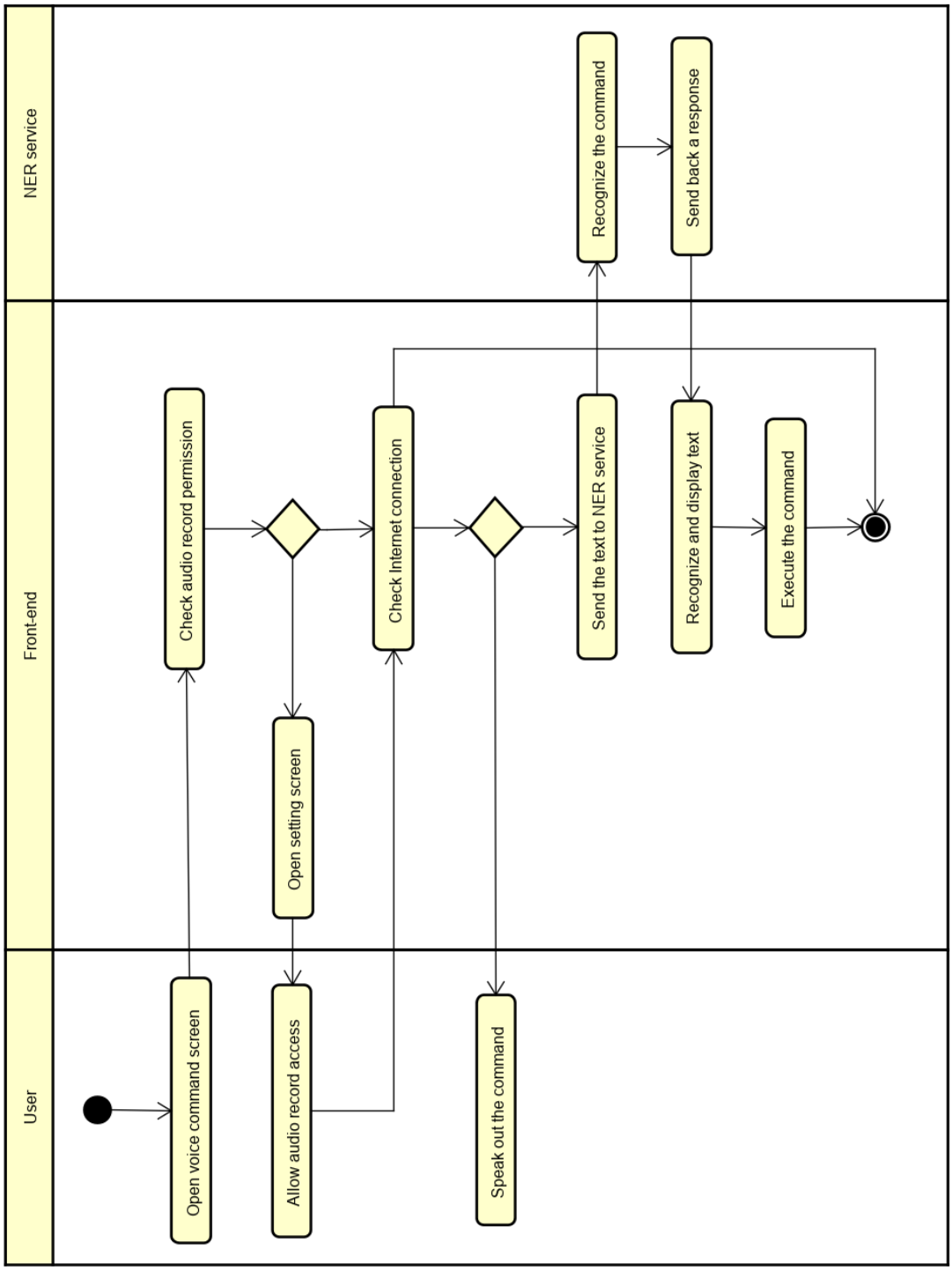

Figure 28 Activity diagram for "Control app by voices".

Redux is a strong and widely-used tool for ReactJs and React Native developers. As mentioned in 3.1.1.2, Redux helps manage state in the application. In Redux, there is

a central store that holds the entire state of the application. Each component can access the stored state without having to send down props from one to another. There are three building parts: actions, store and reducers. Actions are simply events; they are the only way to send data from the application to Redux store. Reducers are pure functions that take the current state of an application, perform an action and returns a new state. These states are stored as objects and they specify how the state of an application changes in response to an action sent to the store. Obviously, the store holds the application state. There is only one store in any Redux application.

Actions in the application are described in **Table 1**. Most of them require no parameters. Only action PLAY_IMMEDIATELY must contain the url address of the audio about to be played. In Vadi Itrithuc, the application state includes (i) isPlaying to control whether any audio news is being played, (ii) playlists to specify the list of audio news to execute action PLAY_NEXT and PLAY_PREVIOUS, (iii) displayMode to specify which mode the application is in to execute action CHANGE_DISPLAY_MODE.

However, due to the complication of related components, voice commands to control maps are not handled with Redux in the present. Application state is still managed traditionally. In the future, Redux should be applied to assure the consistency as well as the performance of the application.

Table 17 Redux Actions

Action	Parameters	Description
PAUSE	No	Pause the media player
PLAY	No	Start media player
PLAY_IMMEDIATELY	audio_url	Play particular audio news
PLAY_NEXT	No	Play next audio news in the playlist
PLAY_PREVIOUS	no	Play previous audio news in the playlist
CHANGE_DISPLAY_MODE	no	Change display mode from day to night or vice versa

5.1.3 Result

Organizing application state with Redux assures the consistency of the state. Controlling audio news playing with Redux also helps simplify and improve the performance compared with the traditional way.

After developing, the application now can execute some basic functions ordered by voice commands. The performance is generally acceptable and smooth enough for users to have good experiences. However, for map related functions, it sometimes takes a rather long period to response. In the future, it is essential to organize and manage map state with Redux.

Figure 29 presents an example of voice command to change display mode from night to day. The action was executed nearly immediately.

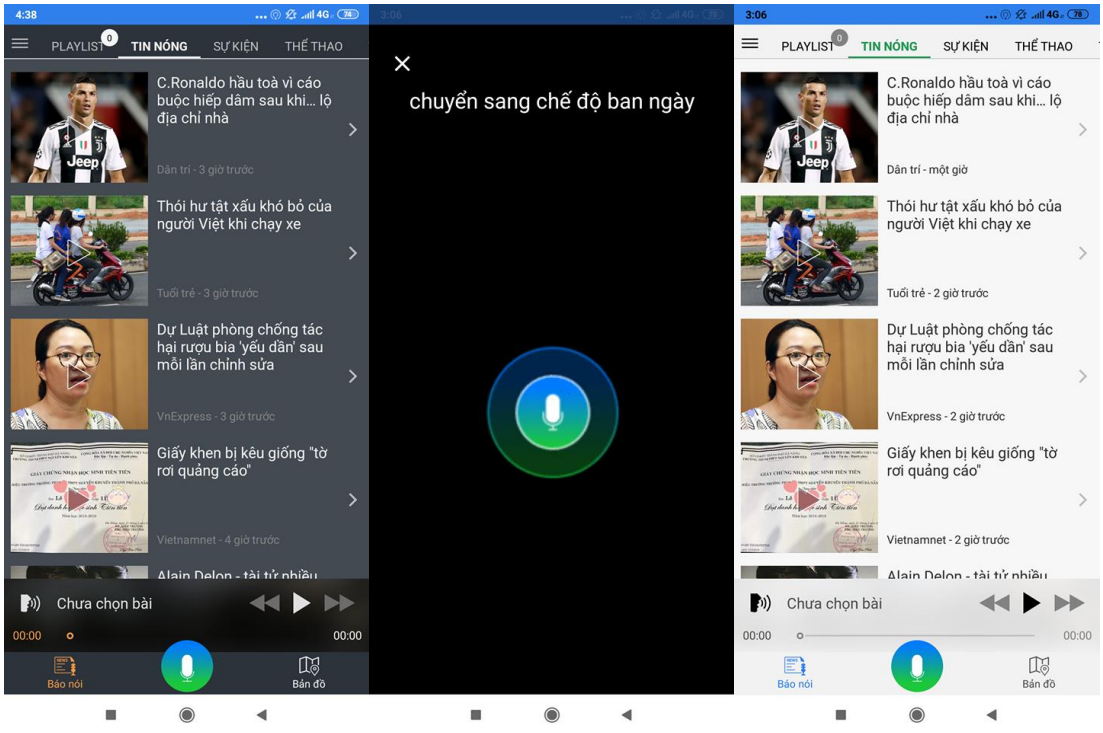

Figure 29 Example of voice command.

5.2 Native modules

As I mentioned in Chapter 3, React Native is a pretty new framework. Although it has a large user community, developers may encounter problems requiring native intervention in some particular cases. They may need to change native codes to enhance applications' performance or fulfill a certain function.

In Vadi Itrithuc, the library react-native-maps is used to display and control interactions with the maps. It helps rendering MapView, originally of Google Maps. However, in this application, instead of Google Maps, VNMaps MapView needs rendering. For this library interferes devices' native layer, I was required to modify and write native codes.

Similar to react-native-maps, react-native-voice is a library that supports voice recognition. It uses Google Cloud Speech API to recognize voice input. But in the application, Vais is utilized instead of Google's service. To achieve this goal, I did modify native codes of the library.

The process of modifying those libraries is rather difficults. It was confusing to find a suitable library because the React Native community is very large, which leads the the releases of many different libraries. It was a must to take a look at some popular libraries, consider their pros and cons to pick the most suitable one for the application.

Additionally, modifying the libraries' code requires basic knowledge about native language programming. Normally, the code base of widely-used libraries such as react-native-maps is very large; it takes time to read and understand them.

A common situation that happens to React Native developers is that the application performance is not good enough, especially in functions that require third-party libraries. Those libraries sometimes are not optimized for their problems.

In this application, when modifying the library react-native-maps to support VNMap, the maps performace is lowered dramatically. After debugging process, what worsens the performance is that the Google base maps are still under the VNMap. I had to turn off that layer to ensure the maps run smoothly enough.

In general, changing a third-party library to solve a particular problem is not an easy job. Developers have to search carefully, master the native language to edit it correctly and be patient enough when debugging.

Chapter 6 Conclusions and Perspectives

Chapter 5 has presented difficulties as well as constructive solutions to solve some complex problems in the application development process. Chapter 6 will summarize the results as well as lessons learned during the application development process in general and graduation project in particular, as well as the orientation of future application development.

6.1 Conclusion

After months of developing, Vadi Itrithuc has become a complete application with many outstanding features. It will soon be deployed in all mobile platforms and expected to get popular in the community. Since all the technologies in the application are provided by Vietnamese agencies, I hope that users will install and experience it and send feedbacks as a way of encourage creativity and dedication among the young generation.

In the process of developing the application, there have been many difficulties. However, thanks to them, I have learnt many new things and improved problem-solving as well as programming skills.

Fortunately, Vadi Itrithuc got the second prize in Startup Competition for Vietnamese Students – SV.Startup in 2018 and Most Favorite Product in Angelhack Hackathon Hanoi 2018.

6.2 Perspectives

Vadi Itrithuc is now a free application that allows users to listen to audio news, use map services and contribute knowledge under the form of voice commands. However, its performance is not stable, especially in map tab. In the future, it is necessary to improve the performance by optimizing map-related functions to avoid interruption and ensures that users have the best experiences with this application.

Additionally, to encourage users to contribute more to the community, they should be charged to listen to audio news, not with money but with points which they can earn by contributing more voice commands. For example, a contributed voice

command is equivalent to twenty points that is used to listen to audio news four times. This method requires users to contribute before they are allowed to use the application.

The dataset collected from users should be utilized to train Vietnamese chatbots, directly improve the application's voice command function. Vadi Itrithuc now only supports some basic commands with medium accuracy. In near future, thanks to improved smart dialogue system, this application will be able to communicate and help users as a true virtual assistant. Not only control within the application, users then can also ask about whatever fields they want such as weather and currency conversion.

In the future, Vadi Itrithuc is expected to become a widely-used application for not only drivers but all the community as well.

Referrences

[1] Facebook, React Native, https://facebook.github.io/react-native/, last visited May 2019.

[2] Jonathan Lebensold, React Native Cookbook, O'Reilly Media, 2018.

[3] Evan M. Hahn, Express in Action, Manning Publications Co., 2016.

[4] RESTfulAPI.net, https://restfulapi.net/, last visited May 2019.

[5] Dan Abramov, Redux, https://redux.js.org/, last visited May 2019.

[6] Open Geospatial Consortium, https://www.opengeospatial.org/standards/wms, last visited May 2019.

[7] Vietnamese Ministry of Science and Technology, https://itrithuc.vn/, last visited May 2019.