LOSTNFOUND KMUTT: KEEP A CLOSE EYE ON YOUR ITEMS.

MR.PASSAKORN PUTTAMA

MR.VATCHARAMAI RODRING

MR.PATTHADOL RAKSAPRAM

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR

THE DEGREE OF BACHELOR OF SCIENCE (COMPUTER SCIENCE)

SCHOOL OF INFORMATION TECHNOLOGY

KING MONGKUT'S UNIVERSITY OF TECHNOLOGY THONBURI

2023

LOSTNFOUND KMUTT: KEEP A CLOSE EYE ON YOUR ITEMS.

MR.PASSAKORN PUTTAMA

MR.VATCHARAMAI RODRING

MR.PATTHADOL RAKSAPRAM

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR

THE DEGREE OF BACHELOR OF SCIENCE (COMPUTER SCIENCE)

SCHOOL OF INFORMATION TECHNOLOGY

KING MONGKUT'S UNIVERSITY OF TECHNOLOGY THONBURI

2023

LOSTNFOUND KMUTT: KEEP A CLOSE EYE ON YOUR ITEMS.

MR.PASSAKORN PUTTAMA

MR.VATCHARAMAI RODRING

MR.PATTHADOL RAKSAPRAM

A Thesis Submitted in Partial Fulfillment of the Requirements for

The Degree of Bachelor of Science (Computer Science)

School of Information Technology

King Mongkut's University of Technology Thonburi

Academic Year 2023

Thesis Committee

.....	Chairman of Thesis Committee
(Assoc.Prof.Dr. Vajirasak Vanijja Ph.D)	
.....	Member and Thesis Advisor
(Asst.Prof. Worarat Krathu Ph.D)	
.....	Member and Thesis Co-Advisor
(Dr. Watanyoo Suksa-ngaim Ph.D)	
.....	Member
(Dr. Vithida Chongsuphajaisiddhi Ph.D)	

Copyright reserved

Thesis Title	LOSTNFOUND KMUTT: KEEP A CLOSE EYE ON YOUR ITEMS.
Thesis Credits	6
Candidate	MR.PASSAKORN PUTTAMA
	MR.VATCHARAMAI RODRING
	MR.PATTHADOL RAKSAPRAM
Thesis Advisors	Asst.Prof. Worarat Krathu Ph.D
	Dr. Watanyoo Suksa-ngaim Ph.D
Program	Bachelor of Science
Field of Study	Computer Science
Faculty	School of Information Technology
Academic Year	2023

Abstract

TravelKit is a mobile application that assists individuals in navigating through public transportation options like buses, sky trains, subways, boats, and mini trucks to reach their desired destinations. Users can access information such as pricing, travel duration, and the number of transfers involved in their chosen route. Additionally, the application includes a feature enabling users to share their routes with one another. TravelKit provides a simple way to commute around Bangkok with a personal preferences, while enhancing travel experience and collecting user data to improve the system. The result is a working mobile application for users to search for travel routes and visualization of route detail on a map, algorithm for suggesting routes to users, and the uses of external frameworks.

Keywords : travel, commute, mobile application, google map

Acknowledgement

We received assistance and guidance from several respected persons, who deserve our deepest gratitude. This project would not have been possible without the support of Asst. Prof. Dr. Worarat Krathu and Dr. Watanyoo Suksa-ngaim contributed invaluable insight and experience.

Contents

CHAPTER PAGE

ABSTRACT iii

ACKNOWLEDGEMENT iv

CONTENTS v

LIST OF TABLES vii

LIST OF FIGURES viii

Chapter 1 Introduction 1

1.1 Background 1

1.2 Objective 1

1.3 Scope 1

1.3.1 Target 1

1.3.2 Scope of Work 1

1.4 Expected Benefits 2

Chapter 2 Feasibility 3

2.1 Introduction 3

2.2 Problem Statement 3

2.3 Related Research Projects 4

2.3.1 Google Maps 4

2.3.2 Apple Maps 4

2.3.3 ViaBus 5

2.3.4 Existing Functions 5

2.4 Requirement Specifications 6

2.4.1 System requirements 6

2.4.2 Mobile application requirements 6

2.5 Implementation Technique 6

2.6 Deliverables 7

2.6.1 Application 7

2.6.2 Documentation 7

Contents(CONT.)

CHAPTER PAGE

2.7 Implementation Plan 8

Chapter 3 Analysis and Design 9

3.1 Introduction 9

3.2 User definition 9

3.2.1 User persona 9

3.2.2 User requirement analysis 10

3.3 Use case diagram 11

3.4 Context diagram 12

3.5 Activity diagram 12

3.5.1 Travel process 12

3.5.2 Community process 13

3.6 Database design 14

3.6.1 Relational Database 14

3.6.2 non-relational database 15

3.6.3 Graph Database 16

Chapter 4 System functionality 20

4.1 Introduction 20

4.2 System architecture 20

4.3 Main function 21

4.4 Test plan and test result 32

Chapter 5 Summary and suggestions 33

5.1 Introduction 33

5.2 Project summary 33

5.3 Problems encountered and solutions 33

5.4 Suggestions for further development 34

5.4.1 Integration of AI in our system 34

5.4.2 Responsive design and implementation 34

5.4.3 High coupling of the data 34

List of Tables(CONT.)

Table PAGE

REFERRENCES..... 34

List of Tables

Table **PAGE**

Table 2.1 Existing functions of related research application and TravelKit..... 5

Table 4.1 Test plan and result 32

List of Figures

Figure PAGE

Figure 2.1 Problem diagram 3
Figure 2.2 The Google Maps application 4
Figure 2.3 The Apple Maps application 4
Figure 2.4 The ViaBus application 5
Figure 2.5 Gantt Chart of the implementation plan 8
Figure 3.1 User one persona 9
Figure 3.2 User two persona 10
Figure 3.3 User three persona 10
Figure 3.4 Use case diagram 11
Figure 3.5 Context diagram 12
Figure 3.6 Travel process 12
Figure 3.7 Community process 13
Figure 3.8 Relational Database Schema 14
Figure 3.9 Non-Relational Database Schema 15
Figure 3.10 GOTO relationship 17
Figure 3.11 WALKTO relationship 18
Figure 3.12 STOPAT/STOPBY relationship 19
Figure 4.1 System architecture 20
Figure 4.2 Welcome screen 21
Figure 4.3 Home screen 22
Figure 4.4 Choosing destination screen 23
Figure 4.5 Choosing route screen 24
Figure 4.6 Route options 25
Figure 4.7 Arrival alert options 26
Figure 4.8 Selected route screen 27
Figure 4.9 Selected route detail screen 28
Figure 4.10 Real-time navigation screen 29

List of Figures(CONT.)

Figure **PAGE**

Figure 4.11 Share route screen..... 30

Figure 4.12 Community screen..... 31

Chapter 1

Introduction

1.1 Background

Losing personal belongings within a university environment is a common challenge, and KMUTT is no exception. Currently, the recovery process involves contacting security personnel in different areas, which can be inefficient and inconvenient for both students and staff. This lack of a centralized system often leads to prolonged delays in retrieving lost items and potential frustrations for the involved parties.

To address these issues, the LostNFound KMUTT platform aims to streamline and enhance the lost-and-found process through a centralized web-based solution. By utilizing modern web technologies and intelligent text-matching models, this platform simplifies reporting, cataloging, and locating lost items.

1.2 Objective

To create a user-friendly and efficient platform that facilitates the reporting and retrieval of lost items. By integrating modern technology into the existing process, we aim to reduce the complexities associated with lost and found items, making it easier for both users and officers to manage and locate belongings promptly.

1.3 Scope

1.3.1 Target

King Mongkut's University of Technology Thonburi (KMUTT).

1.3.2 Scope of Work

1. Define Project Features:

    - **Distinct Features:**

        (a) Item Found & Lost Form: Allow users who found or lost items to report them, providing the time and location where it was found and describing the item to enable matching.

            - Found Item Form.
            - Lost Item Form.

        (b) Claim Process: Allow users to claim found items after the form is matched.

            - Text comparison using Natural Language Processing (NLP).
            - Notification system.

        (c) Secure Items Ownership: Prevent unauthorized claims using university email

authentication.

- **Support Features:**
  (a) Authentication using OAuth with Microsoft authentication.
  (b) History of Claimed Items.

2. Architecture and Technology Stack:
   - **Frontend:** Next.js 14.
   - **Backend:** Next.js 14 and FastAPI.
   - **Database:** PostgreSQL.
   - **Authentication:** NextAuth.js with Microsoft as the provider.
   - **Infrastructure:** Google Cloud (GCP), Docker.
   - **Model:** Public text similarity and NER model from Hugging Face.
   - **Testing:** Jest (Unit testing), Postman (API testing).

1.4 Expected Benefits

1. **Efficiency Improvement:** Reduced processing time for reporting, cataloging, and retrieving lost items.
2. **User Convenience:** Accessible information through a user-friendly interface.
3. **Enhanced Security:** Robust identity verification to ensure rightful ownership.
4. **Reduced Frustration:** Simplified processes for users and staff.
5. **Resource Optimization:** Improved productivity for staff by automating processes.
6. **Timely Notifications:** Notifications to remind users to pick up their items.
7. **Documentation and Accountability:** Comprehensive records of claimed items for auditing.

Chapter 2

Feasibility Study

2.1 Introduction

The LostNFound platform is a centralized web application designed to address the challenges associated with managing lost-and-found items at King Mongkut's University of Technology Thonburi (KMUTT). The platform aims to provide students, staff, and university personnel with an efficient, accessible, and secure solution for reporting and reclaiming lost items.

Unlike the current manual and on-site processes, the LostNFound platform digitizes the entire lost-and-found workflow. Users can report lost items, search for them, and claim ownership through the platform, reducing time and effort. It also introduces features such as real-time item status updates, secure ownership verification, and centralized database management. This initiative aligns with KMUTT's commitment to modernizing student services and fostering a more convenient university experience.

2.2 Problem Statement

Currently, KMUTT's lost-and-found system relies on physical reporting to either nearby security personnel or the Student Affairs Division. This approach is inefficient, time-consuming, and prone to security risks such as unauthorized claims. The lack of a unified system makes tracking and managing lost items cumbersome for both users and staff. Furthermore, the absence of a centralized database means there is no effective way to ensure transparency or accessibility for all stakeholders.

The LostNFound platform aims to resolve these issues by providing a digital solution. The platform will centralize all lost-and-found operations, improve communication between stakeholders, and reduce the time required to locate and reclaim lost items. By introducing advanced features such as user profiles, secure claims verification, and automated notifications, the system will create a seamless experience for KMUTT's community.

2.3 Related Research and Projects

2.3.1 Lifeguard Lost & Found

A system designed to handle lost and found items in public spaces like airports and transport hubs, which shares similarities with the university setting. It allows users to report, search for, and claim items, ensuring secure ownership verification.

2.3.2 MissingX

A centralized platform for lost and found items, focusing on automatic matching of lost and found property. The platform also supports user notifications when a match is made, aligning

with this system's goal of real-time updates and seamless matching.

2.3.3 ItemFinder

Designed specifically for institutions like universities, ItemFinder handles lost and found items by providing a centralized platform for reporting, cataloging, and verifying items.

2.3.4 Comparison of Existing Functions

Table 2.1: Existing Functions of Related Applications

Applications	Item Reporting	Matching (NLP)	Notifications	History of Claims	User Pr
Lifeguard Lost & Found	Yes	No	No	No	No
MissingX	Yes	Yes	Yes	No	No
ItemFinder	Yes	No	No	No	Ye
LostNFound KMUTT	Yes	Yes	Yes	Yes	Ye

2.4 Requirement Specifications

2.4.1 Functional Requirements

1. Report lost and found items by filling out a form with details such as item description, location, and time.
2. Provide real-time updates regarding the status of items.
3. Use Natural Language Processing (NLP) to match descriptions of lost and found items.
4. Notify users when a match is found for their lost item or when their claimed item is ready for pickup.
5. Maintain a history of claimed items, including photos and verification details.
6. Integrate with the university's email system (OAuth with Microsoft) for authentication and ownership verification.

2.4.2 Data Requirements

1. **User Information:** Collect and store user details (e.g., phone number, university email, profile information, claim history) for authentication and personalized user experience.
2. **Item Information:** Store details of each lost and found item, including item type, description, location, date, and images.

2.5 Implementation Techniques

2.5.1 Frontend

- Programming Language: TypeScript.
- Framework: Next.js.

2.5.2 Backend

- Programming Language: TypeScript, Python.
- Frameworks: Next.js, FastAPI.

2.5.3 Infrastructure

- OS: Debian Linux.
- Cloud Provider: Google Cloud Platform (GCP).
- Containerization: Docker.
- CI/CD: Google Cloud Build.
- Deployment Platform: Google Cloud Run.

2.5.4 Database

- Database Type: Relational (PostgreSQL).

2.5.5 Testing

- Unit Testing: Jest.
- API Testing: Postman.

Chapter 3

Analysis and Design

3.1 Introduction

This chapters covers three topics which are user requirement, business process designs, and system design. The user requirement part discusses the business needs for what users require from the system, The business process design part discusses the activities diagram, use case diagram, and context diagram of the application. Lastly, the system design consists of database design, and system architecture.

3.2 User definition

- People who aged around 12 - 40 years old and familiar with English language
- Lived in Bangkok and in the city crowded with technology-accessible people
- People who trying to travel into newly visited place without known directions
- People who want to browse alternative transportation method to reach their destination with their own condition like cost savings, faster route, or less transfer

3.2.1 User persona

Somchai Suriyawong

He is a student and just graduated Grade 12 from high school at his hometown, Petchaburi. He is the one who familiar with modern technology, he usually browse for new application to tryout, so he have sense of application basis and knows common application interaction so he can try new application without needs of reading manual or any tutorial. Recently, he has moved to Bangkok to attend bachelor degree in the university. While he is in Bangkok, he usually browse for second-hand deals on the internet. So, he needs to make an appointment with the seller which the places are varies. As he has only one income channel which is family supporting, so he needs to save money as much as he can. However, as just moved form provincial, so he do not know much public transport system and route in Bangkok and the existing routing applications still not show the complete cost for him to choose whether which way is most suitable for him based on time and cost and it would be better if it notifies him before transferring.

GOALS
Reasons to use our service

- Ease of information access
- Orderly arranged transport information displaying
- Shorten time for choosing or sorting the route based on trip criteria

PERSONAL

Age	19
Occupation	Student
Status	Single
Hometown	Petchaburi, Thailand
Income	8,000 THB
Education	Bachelor Degree

INFLUENCERS
What products or services are influencing

PAIN POINTS
The problems

- Existing application not have cost information for each of public transports
- No way to notify when about to reach transfer point or destination if he fall asleep.

Figure 3.1: User one persona

Sasithorn Chandranapa

She is an employee of a consultant company, so she regularly travel to customer company which located both inside Bangkok and its perimeter. As the company continuously grow, new customer and their branches are increasing. So she always need to find the way to visit each of new places but she does not have personal car, so she needs to use public transport for traveling. As some days, she have to visit 2 company branches which located far away. So, speed of travel to get into the destination is the most important factor for her. However, existing navigation application does not have public transport information outside Bangkok and it still have only government-provided transport system information. So, we might miss some transport system that can help her to reach destination faster, like Van or Minibus.

GOALS
Reasons to use our service

- Able to find the fastest possible route to reach destination
- Need crowdsourced information to inform the traffic situation of a time in each area

PERSONAL

Age	27
Occupation	Marketing Consultant
Status	Married
Hometown	Bangkok, Thailand
Income	34,000 THB
Education	Bachelor Degree

INFLUENCERS
What products or services are influencing

PAIN POINTS
The problems

- Existing application have only government-provided information
- Some applications does not provide public transport information, including local transport method.

Figure 3.2: User two persona

Niran Pruksamanee

He is a fresh graduated university student who recently get a job as an software engineer at Centara which located in the center of Bangkok. As he has chosen not to rent an apartment near his workplace because the rent is very high and he want to save up that money. So, he needs to travel by public transportations every morning as he does not own personal car. Usually, he will take the motorcycle taxi and skytrain to his workplace. However, sometimes he is not in rush and has time prepared for travelling in the morning, he is now seeking other ways to travel to his workplace, in order to save up more money as the skytrain is quite expensive. Unfortunately, during his university days, he usually spent his free time for travelling around his university, so he does not familiar with the places and other public transportations, for example, sometimes he does not know which bus line he needs to take.

GOALS
Reasons to use our service

- Able to find the different routes apart from the same route he is familiar with
- Able to find the route that can save him some money for transportation
- Explore other travel routes in Bangkok

PERSONAL

Age	22
Occupation	Software Engineer
Status	Single
Hometown	Chiang Mai, Thailand
Income	25,000 THB
Education	Bachelor Degree

INFLUENCERS
What products or services are influencing

PAIN POINTS
The problems

- Existing application not have cost information for each of public transports
- Some applications does not provide public transport information, including local transport method.

Figure 3.3: User three persona

3.2.2 User requirement analysis

- This application helps users to find the best route for traveling.
- This application uses map service provider and our public transportation data such as

vans, and mini truck to suggest the route for users.

- This application provides the community to share the routes of traveling of users.
- This application will provide a mobile application for users that use smartphones.

3.3 Use case diagram

Figure 3.4: Use case diagram

The actor who is user in the routing system. The user must create an account to login to the system. After the user finds routes to the destination, the system will calculate the cost of traveling and display alternative routes using public transport data, such as vans and mini trucks, so that the user can make informed decisions. In addition, the user can record their trip after reaching the destination. Lastly, the user can share their trip to the community, along with the details of their travel, as well as comment and recommend other users.

3.4 Context diagram

Figure 3.5: Context diagram

The context diagram shows the overall of the TravelKit system that required the name, password, user's location, destination. Then, the system will create choices of the routes, directions and calculate the cost of traveling to the destination.

3.5 Activity diagram

3.5.1 Travel process

Figure 3.6: Travel process

The travel process for the TravelKit system starts when the user searches for the designated destination. Then, the user will be presented with the default starting point, which the user can either use their current location or select it again. After that, the user will choose their route options, which include the travel cost, time, number of transfers, and the arrival alert. If the user desires to have an arrival alert, the system will enable travel notification. Subsequently, the user can see the provided routes details, choose the route and then start the trip. Meanwhile, the system will start recording the trip and send the notification to the user when they are almost arriving at the destination. Finally, the user can choose either to share their travel route or not. If they choose to do so, the system will save the travel information to improve our system data and user can share to other people in community.

3.5.2 Community process

Figure 3.7: Community process

The community processes for the TravelKit system can be divided into two sub-processes. The first process is for the users who desire to search for routes, which can enter the code to get exact route detail. The second process is for users who wish to find for routes or places recommendations. The process starts when the user select the destination. Afterwards, the user can view the recommended routes and then select the route they preferred.

3.6 Database design

3.6.1 Relational Database

Figure 3.8: Relational Database Schema

Our database relies on MySQL to store user data efficiently. It is intricately connected with MongoDB, particularly in managing community-related information. This dual-database approach enhances functionality, ensuring seamless integration between individual user data and community features, contributing to a robust and scalable system.

3.6.2 non-relational database

Figure 3.9: Non-Relational Database Schema

MongoDB is used for storing transport data, stops, and each stop related to each route. It includes:

- **Transports** for storing all data about transport types and transports
- **Stops** for storing stop data, such as bus stops, train stations, and ports
- **Transport Stops** for storing all stops that relate to each transport sequentially
- **Community Routes** for storing shared routes from each users
- **Suggestion** for storing the most frequently selected destinations by users

3.6.3 Graph Database

The graph database used for storing routes and stops relationship for using the algorithm to find shortest path based on user queries. The graph consist of nodes and relations which is the following node types:

- **STOP** For storing stop or station of all transportation type (e.g. National Theatre, Sanam Luang Bus Terminal, Hua Lamphong, Opposite Tha Phra Chan)
- **ARL** For each line of Airport Rail Link system
- **BRT** For each line of Bus Rapid Transit system
- **BTS** For each line of BTS Skytrain system
- **BUS** For each route of BMTA bus line
- **CHE** For each route of Chao Phraya Express Boat
- **KPS** For each route of minibus and local mini truck
- **MRT** For each line of Metropolitan Rapid Transit
- **SRT** For each line of train

Concerning relationships, the project incorporates various types tailored for tasks such as shortest path finding [**?**], performance optimization, and conditional querying. These relationships are categorized as follows:

3.6.3.1 GOTO relationship

Figure 3.10: GOTO relationship

The relation of each two stops that are next to each other and have at least one transportation that pass from one to another (for example, from Sanam Luang stop to National Theatre stop) consider as one-way direction of each side of the road. This relation used for specifying the stop that are available for each bus line to transfer to another

3.6.3.2 WALKTO relationship

Figure 3.11: WALKTO relationship

As GOTO relation might not enough for finding the best suitable route, since some stops may not next to each other but user can transfer, walk or reroute between bus line that are in opposite direction for shorter distance. So we create WALKTO relation for all pair of stops that are walkable within 500 meters for querying the route which user can transfer to the nearby stop which are not directly next to each other.

3.6.3.3 STOPAT/STOPBY relationship

Figure 3.12: STOPAT/STOPBY relationship

The relationship represent that each transportation line (e.g., bus line, BTS line) are stop at each stop/stations. Each stop are stoppable from many transportation line and each transportation line can pass trough and stoppable at many stop as well.

Chapter 4

System functionality

4.1 Introduction

This chapter describes the core aspects of the system's functionality, covering its system architecture, primary functions, planning, and testing results that defined the application's capabilities. The first part covers the architecture of the system in this project and the second part covers the main functionality of the system.

4.2 System architecture

Figure 4.1: System architecture

The system will consist of two main components. The first component is frontend, the user-interaction part written as Flutter application which has functionalities of login, using Google OAuth API and has the map preview for user to search for places and navigate through routes. It will have to retrieve raw location data from the device GPS and call to backend using REST API. The second component is backend, which consists of the data manipulation and algorithmic part, which written using Golang. The backend use for retrieve route request from frontend, then All the connections between frontend and backend are proxies trough Cloudflare and internal NGINX reverse proxy. For the backend process, we use MariaDB for relational data store, Neo4j [?] for graph data store, The maps library, Google Maps API and self-hosted OSRM API, is used gather places, routes and any geographic information.

4.3 Main function

Figure 4.2: Welcome screen

This figure shows the welcome message and login button, Users can log in by Google to register and log into our application.

Figure 4.3: Home screen

This figure shows the map, current location, the name the current location after logging in and the user can tap the location icon to update the map screen to show their current location. The user can switch between the home screen and the community screen using the bottom navigation bar below.

Figure 4.4: Choosing destination screen

Users can search routes by specifying both the starting location and the destination to find routes for users.

Figure 4.5: Choosing route screen

This figure shows the search route results from the starting location to the destination. All routes are sorted by estimated time of arrival by default and users can see multiple routes to go to the destination.

Figure 4.6: Route options

The recommended routes can be sorted according to the user's preferences, such as the fastest arrival time, the cheapest price, the shortest transfer, and the avoid ferry option.

Figure 4.7: Arrival alert options

There are alerts that can be set when users are nearing the transfer point or the destination which help them reach their destination accurately.

Figure 4.8: Selected route screen

Users will see the details of the selected route on the map once they select a route from the search route results. From the starting point to the destination, the route polyline is color-coded separately.

Figure 4.9: Selected route detail screen

After swiping up the bottom sheet, users will see the details of the selected route including cost, estimated time of arrival, and passed stop of each transport type. Users can tap the Start button for starting to real-time navigation or the Share icon to share the selected route with others.

Figure 4.10: Real-time navigation screen

This screen shows the details of the route in real-time including the current location, estimated arrival time, and the navigation overlay showing the current step with time and the next step at the top of the screen.

Figure 4.11: Share route screen

When users tap the Share button, they will receive the generated code and be able to share the code with others without having to search for the route again.

Figure 4.12: Community screen

This figure shows the suggested places, which are sorted by how often they are searched, and users can search routes based on the code that is generated by others that will navigate them to the shared routes.

4.4 Test plan and test result

Module	Test expectation	Expected result	Result
Authentication	Login with Google	Successful login to the system with user information provided	Success
Home page	Able to navigate between home page and community page	Can change tab between two page	Success
	Able to see the map with the current user location	See the map and correct current location	Success
	Able to logout from profile	User is logged out	Success
	Able to navigate to search page/ choose destination	Navigate to search page correctly	Success
Routing	Able to search for places	User sees the list of places from Google API	Success
	Able to sort the route options	User can change the sort by choices to change the travel preferences based on time, cost, and number of transfer	Success
	Able to see the suggested routes when entering start and end location	User sees the list of suggested routes correctly with the details	Success
	Able to navigate to see each route detail	User sees the route detail on the map and modal with travel time and cost	Success
	Able to share the route	User gets a 6-digit code for sharing the route with other users	Success
	Able to start the trip	User navigates to the routing screen with a map and directions provided	Success
	Able to receive notification when an event happens	The system alerts the correct notification	Success
Community	Able to get the exact route when searching by code	User navigates to the route detail page correctly	Success
	Able to see the suggested places	List of suggested places displays correctly when entering the community page	Success
	Able to navigate to suggested routes page from suggested places	User navigates to the routes selection page correctly	Success

Table 4.1: Test plan and result

Chapter 5

Summary and suggestions

5.1 Introduction

This chapter provides a summary of the project, which comprises three parts: project summary, problems encountered and solutions, and suggestions for further development. The first section summarizes and provides the overall results of the project. The second section outlines the problems encountered and proposes solutions to the limitations of the project. Finally, suggestions are provided for the future improvement of the project.

5.2 Project summary

TravelKit is a mobile application that assists individuals in navigating through public transportation options like buses, sky trains, subways, boats, and mini trucks to reach their desired destinations. Users can access information such as pricing, travel duration, and the number of transfers involved in their chosen route. Additionally, the application includes a feature enabling users to share their routes with one another. From the project's objective and scope, we can create a software that can calculate travel routes, distances, and estimate travel costs with optimizing the suggested routes based on user personal choices.

5.3 Problems encountered and solutions

There are two main problems that we encountered during the project. First problem that we find difficult to cope with is the process of developing mobile application that uses Dart with Flutter as a programming language and framework that we are not quite familiar with. So, we need to learn some new programming styles for the development. The second challenge involves incorporating Neo4j as a graph database, a tool with which we lack prior implementation experience. Consequently, we must familiarize ourselves with the database's query language in order to effectively build our system.

5.4 Suggestions for further development

5.4.1 Integration of AI in our system

Currently, our application has a feature that suggest routes based on the users preferences but we can use artificial intelligence to enhance this feature by collecting data from user. We can use that data to predict what sorting preferences user is likely to choose. Furthermore, we can suggest more accurate recommended places to user if we have a proper ai model.

5.4.2 Responsive design and implementation

Currently, our application is working correctly with only certain devices. To support wide variety of mobile and desktop screens, it need revisions in design and implementation to support the responsive design.

5.4.3 High coupling of the data

Currently, our service has accurate data that we can return route data correctly, but to modify route data, we need to update both in the MongoDB and Neo4j separately which is consequence of manually inserting data to all databases and all relation in Neo4j needs to changed for the updated route.