



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

2024





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

2024

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 2024

---

#### Thesis Committee

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

Copyright reserved

<b>Thesis Title</b>	LOSTNFOUND KMUTT: KEEP A CLOSE EYE ON YOUR ITEMS.
<b>Thesis Credits</b>	6
<b>Candidate</b>	MR.PASSAKORN PUTTAMA MR.VATCHARAMAI RODRING MR.PATTHADOL RAKSAPRAM
<b>Thesis Advisors</b>	Asst.Prof. Worarat Krathu Ph.D
<b>Program</b>	Bachelor of Science
<b>Field of Study</b>	Computer Science
<b>Faculty</b>	School of Information Technology
<b>Academic Year</b>	2024

---

## Abstract

LostNFound KMUTT is a web-based platform designed to assist KMUTT students and university personnel in efficiently managing lost and found items within the university. The platform allows users to report lost or found items through dedicated forms, providing item descriptions, locations, and timestamps for precise matching. It utilizes advanced technologies such as Natural Language Processing (NLP) for text similarity matching and secure authentication via university email to ensure rightful ownership. Additionally, it features a history of claimed items for accountability and a notification system to remind users to retrieve their belongings promptly.

The development process demonstrates the feasibility of creating an efficient lost-and-found management system using existing university resources and modern web technologies. By streamlining the process, the platform minimizes user frustration and enhances operational productivity for KMUTT staff.

**Keywords :** Lost and Found, KMUTT, Web Platform, Item Matching, Secure Claims

## Acknowledgement

We would like to express our heartfelt gratitude to the individuals whose guidance and support made this project possible. Our deepest appreciation goes to our generous advisor Asst. Prof. Dr. Worarat Krathu, provided invaluable insights and mentorship throughout the development of the LostNFound KMUTT platform.

We also extend our special thanks to our special supporters, including Mr.Ukrit Ruckcharti, Mr.Sukanya Chinwicha, and Mr. Thanatat Wongabut for sacrificing their time for additional consulting. Lastly, we appreciate the support of the KMUTT community, whose feedback and encouragement have been instrumental in shaping this project.

## Contents

CHAPTER	PAGE
ABSTRACT .....	iii
ACKNOWLEDGEMENT.....	iv
CONTENTS.....	v
LIST OF TABLES .....	viii
LIST OF FIGURES .....	ix
<b>Chapter 1 Introduction .....</b>	<b>1</b>
1.1 Background.....	1
1.2 Objective.....	1
1.3 Expected Benefits .....	1
1.4 Scope.....	1
1.4.1 Target.....	1
1.4.2 Scope of Work .....	2
<b>Chapter 2 Feasibility Study .....</b>	<b>3</b>
2.1 Introduction.....	3
2.2 Problem Statement.....	3
2.3 Related Research and Projects.....	3
2.3.1 Lifeguard Lost & Found .....	3
2.3.2 MissingX.....	3
2.3.3 ItemFinder .....	4
2.3.4 Comparison of Existing Functions .....	4
2.4 Requirement Specifications.....	4
2.4.1 Functional Requirements.....	4
2.4.2 Data Requirements.....	4
2.5 Implementation Techniques .....	4
2.5.1 Frontend .....	4
2.5.2 Backend.....	5
2.5.3 Infrastructure .....	5

## Contents(CONT.)

CHAPTER	PAGE
2.5.4 Database .....	5
2.5.5 Testing.....	5
<b>Chapter 3 Analysis and Design.....</b>	<b>6</b>
3.1 Introduction.....	6
3.2 User definition .....	6
3.2.1 User persona .....	6
3.2.2 User requirement analysis.....	7
3.3 Use case diagram .....	8
3.4 Context diagram.....	9
3.5 Activity diagram.....	9
3.5.1 Travel process.....	9
3.5.2 Community process .....	10
3.6 Database design.....	11
3.6.1 Relational Database.....	11
3.6.2 non-relational database.....	12
3.6.3 Graph Database.....	13
<b>Chapter 4 System Functionality.....</b>	<b>17</b>
4.1 Introduction.....	17
4.2 System Architecture .....	17
4.3 Test Plan and Results.....	18
<b>Chapter 5 Summary and suggestions.....</b>	<b>19</b>
5.1 Introduction.....	19
5.2 Project summary.....	19
5.3 Problems encountered and solutions.....	19
5.4 Suggestions for further development .....	20
5.4.1 Integration of AI in our system .....	20
5.4.2 Responsive design and implementation.....	20
5.4.3 High coupling of the data .....	20



## List of Tables(CONT.)

Table	PAGE
REFERENCES.....	20

## List of Tables

Table	PAGE
Table 2.1 Existing Functions of Related Applications .....	4
Table 4.1 Test Plan and Results .....	18

## List of Figures

Figure	PAGE
Figure 3.1 User one persona .....	6
Figure 3.2 User two persona .....	7
Figure 3.3 User three persona .....	7
Figure 3.4 Use case diagram.....	8
Figure 3.5 Context diagram .....	9
Figure 3.6 Travel process .....	9
Figure 3.7 Community process.....	10
Figure 3.8 Relational Database Schema .....	11
Figure 3.9 Non-Relational Database Schema .....	12
Figure 3.10 GOTO relationship .....	14
Figure 3.11 WALKTO relationship.....	15
Figure 3.12 STOPAT/STOPBY relationship .....	16
Figure 4.1 System Architecture.....	17

# 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 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.

### 1.4 Scope

#### 1.4.1 Target

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

### 1.4.2 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).

## 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

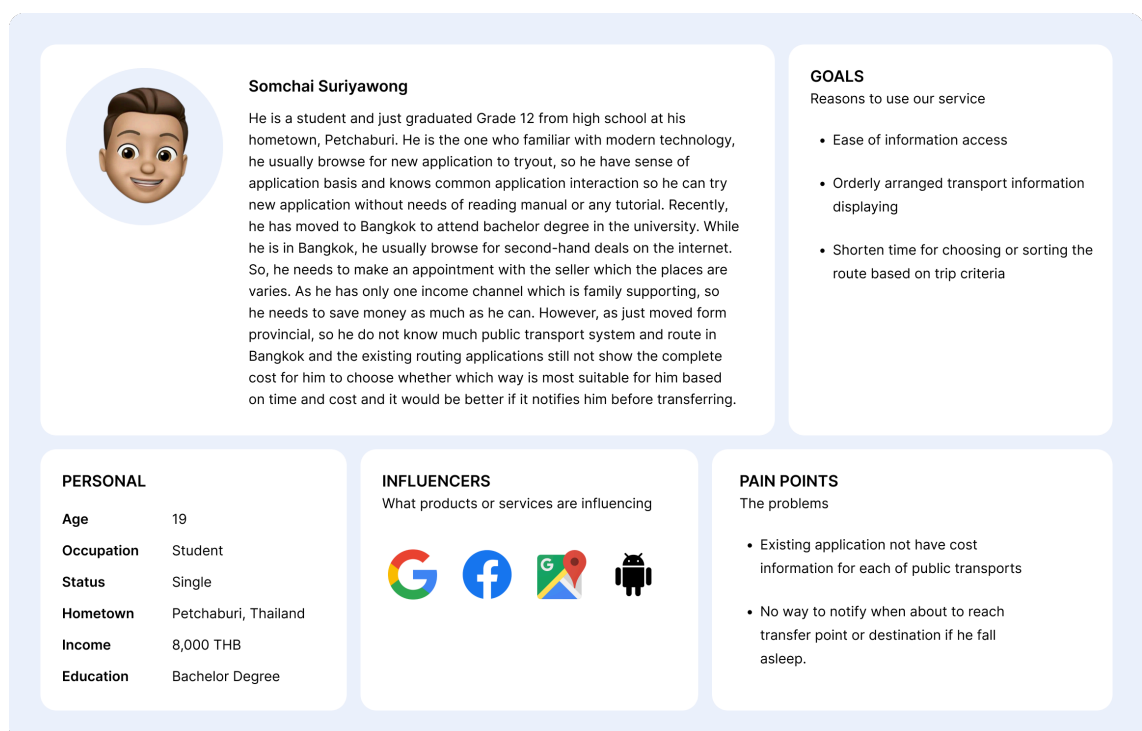
#### 3.1 Introduction

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

Google, Facebook, Google Maps, Android

**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

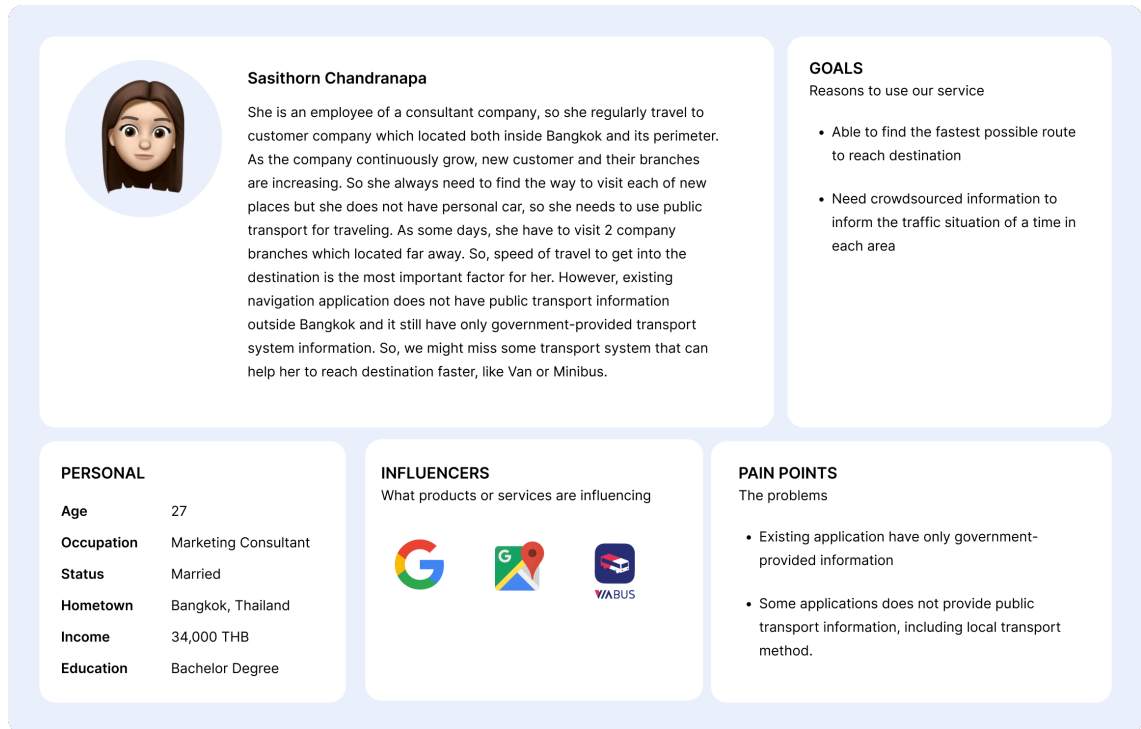


Figure 3.2: User two persona

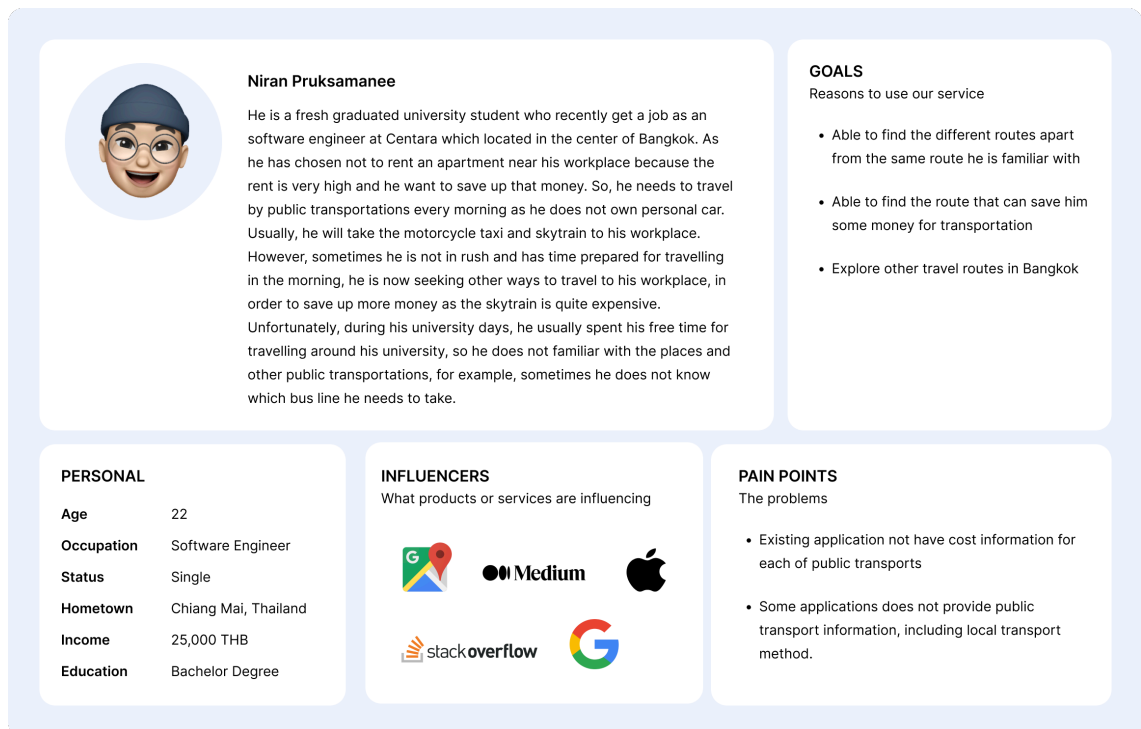


Figure 3.3: User three persona

### 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

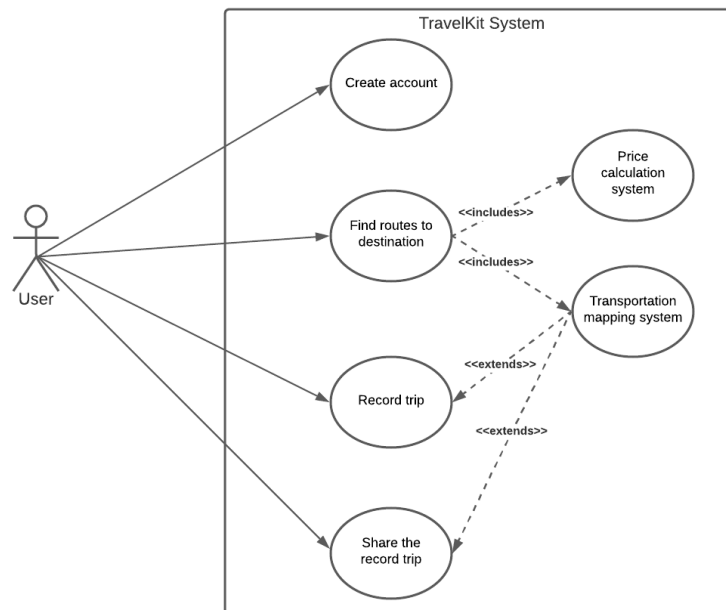


Figure 3.4: Use case diagram

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

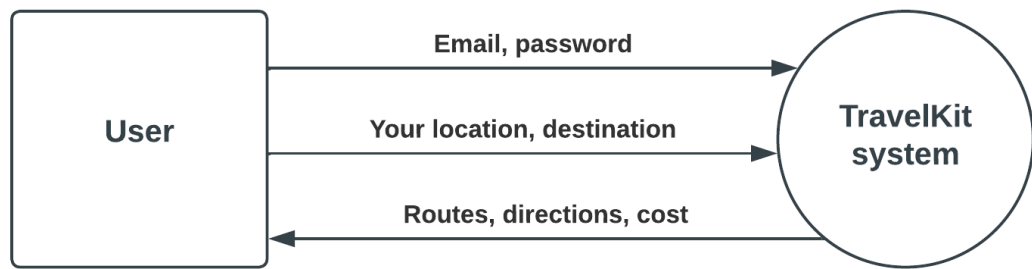


Figure 3.5: Context diagram

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

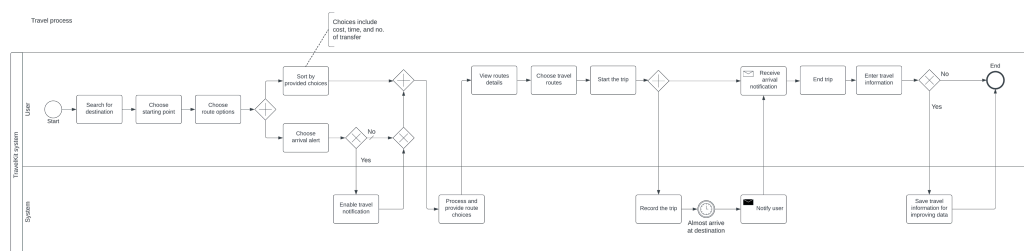


Figure 3.6: Travel process

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

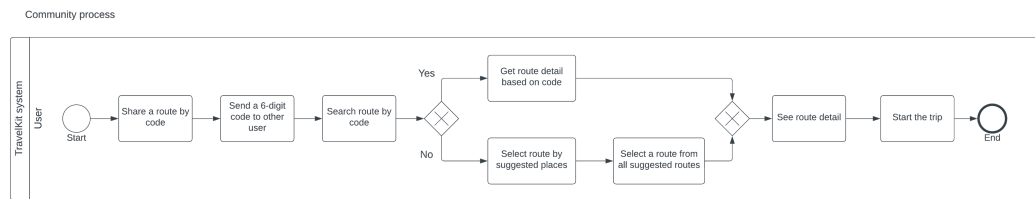
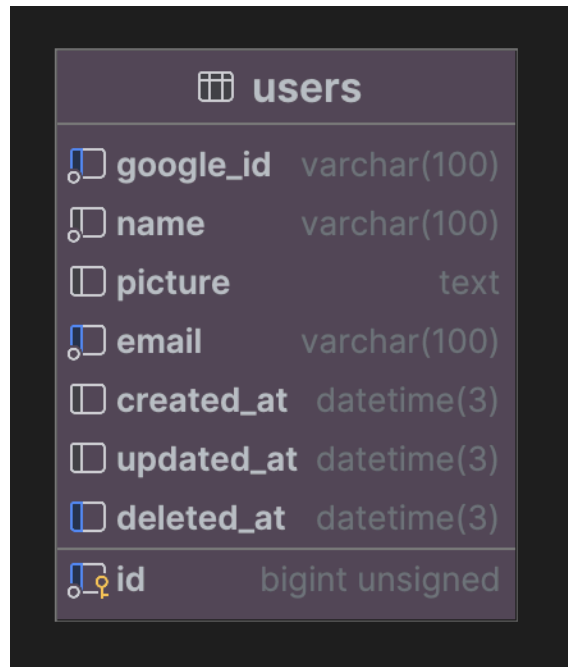


Figure 3.7: Community process

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



The image shows a database schema for a table named 'users'. The table has the following columns and data types:

users	
google_id	varchar(100)
name	varchar(100)
picture	text
email	varchar(100)
created_at	datetime(3)
updated_at	datetime(3)
deleted_at	datetime(3)
id	bigint unsigned

The 'id' column is marked as the primary key with a key icon. The 'google\_id', 'email', and 'deleted\_at' columns are marked with a blue icon, likely indicating they are indexed or have unique constraints.

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








































































community_routes	transport_stops	suggestions
<div><div></div><div><div>_id</div><div>objectid</div></div></div> <div><div></div><div><div>code</div><div>string</div></div></div> <div><div></div><div><div>created_at</div><div>isodate</div></div></div> <div><div></div><div><div>route</div><div>object</div></div></div> <div><div></div><div><div>updated_at</div><div>isodate</div></div></div> <div><div></div><div><div>user_id</div><div>string</div></div></div> <div><div></div><div><div>route.itinerary</div><div>object</div></div></div> <div><div></div><div><div>route.itinerary.steps</div><div>list</div></div></div> <div><div></div><div><div>route.itinerary.steps.full_name</div><div>string</div></div></div> <div><div></div><div><div>route.itinerary.steps.name</div><div>string</div></div></div> <div><div></div><div><div>route.itinerary.steps.price</div><div>int32</div></div></div> <div><div></div><div><div>route.itinerary.steps.time</div><div>int64</div></div></div> <div><div></div><div><div>route.itinerary.steps.transport_type</div><div>string</div></div></div> <div><div></div><div><div>route.itinerary.total_price</div><div>int32</div></div></div> <div><div></div><div><div>route.itinerary.total_stops</div><div>int32</div></div></div> <div><div></div><div><div>route.itinerary.total_time</div><div>int64</div></div></div> <div><div></div><div><div>route.step_detail</div><div>list</div></div></div> <div><div></div><div><div>route.step_detail.depart_stop_id</div><div>string</div></div></div> <div><div></div><div><div>route.step_detail.depart_stop_latitude</div><div>double</div></div></div> <div><div></div><div><div>route.step_detail.depart_stop_longitude</div><div>double</div></div></div> <div><div></div><div><div>route.step_detail.depart_stop_name</div><div>string</div></div></div> <div><div></div><div><div>route.step_detail.nodes</div><div>list</div></div></div> <div><div></div><div><div>route.step_detail.nodes.stop_id</div><div>string</div></div></div> <div><div></div><div><div>route.step_detail.nodes.stop_latitude</div><div>double</div></div></div> <div><div></div><div><div>route.step_detail.nodes.stop_longitude</div><div>double</div></div></div> <div><div></div><div><div>route.step_detail.nodes.stop_name</div><div>string</div></div></div> <div><div></div><div><div>route.step_detail.polyline</div><div>array</div></div></div> <div><div></div><div><div>route.step_detail.polyline_color</div><div>string</div></div></div> <div><div></div><div><div>route.step_detail.price</div><div>int32</div></div></div> <div><div></div><div><div>route.step_detail.time</div><div>int64</div></div></div> <div><div></div><div><div>route.step_detail.transport_full_name</div><div>string</div></div></div> <div><div></div><div><div>route.step_detail.transport_name</div><div>string</div></div></div> <div><div></div><div><div>route.step_detail.transport_type</div><div>string</div></div></div>	<div><div></div><div><div>_id</div><div>objectid</div></div></div> <div><div></div><div><div>created_at</div><div>isodate</div></div></div> <div><div></div><div><div>stops</div><div>list</div></div></div> <div><div></div><div><div>transport_level</div><div>int32</div></div></div> <div><div></div><div><div>updated_at</div><div>isodate</div></div></div> <div><div></div><div><div>stops.stop_id</div><div>int64</div></div></div> <div><div></div><div><div>stops.stop_level</div><div>int32</div></div></div> <div><div></div><div><div>stops.stop_polyline</div><div>list</div></div></div> <div><div></div><div><div>stops.stop_sequence</div><div>int64</div></div></div> <div><div></div><div><div>transport_id</div><div>int64</div></div></div>	<div><div></div><div><div>_id</div><div>objectid</div></div></div> <div><div></div><div><div>count</div><div>int32</div></div></div> <div><div></div><div><div>created_at</div><div>isodate</div></div></div> <div><div></div><div><div>image_reference</div><div>string</div></div></div> <div><div></div><div><div>latitude</div><div>double</div></div></div> <div><div></div><div><div>longitude</div><div>double</div></div></div> <div><div></div><div><div>name</div><div>string</div></div></div> <div><div></div><div><div>updated_at</div><div>isodate</div></div></div> <div><div></div><div><div>place_id</div><div>string</div></div></div>
	<div><div></div><div><div>stops</div></div></div> <div><div></div><div><div>_id</div><div>objectid</div></div></div> <div><div></div><div><div>created_at</div><div>isodate</div></div></div> <div><div></div><div><div>icon_url</div><div>string</div></div></div> <div><div></div><div><div>latitude</div><div>double</div></div></div> <div><div></div><div><div>stop_id</div><div>int64</div></div></div> <div><div></div><div><div>updated_at</div><div>isodate</div></div></div> <div><div></div><div><div>level</div><div>int32</div></div></div> <div><div></div><div><div>name</div><div>string</div></div></div> <div><div></div><div><div>longitude</div><div>double</div></div></div>	<div><div></div><div><div>transports</div></div></div> <div><div></div><div><div>_id</div><div>objectid</div></div></div> <div><div></div><div><div>ac_level</div><div>int32</div></div></div> <div><div></div><div><div>created_at</div><div>isodate</div></div></div> <div><div></div><div><div>icon_url</div><div>string</div></div></div> <div><div></div><div><div>name</div><div>string</div></div></div> <div><div></div><div><div>transport_id</div><div>int64</div></div></div> <div><div></div><div><div>transport_level</div><div>int32</div></div></div> <div><div></div><div><div>updated_at</div><div>isodate</div></div></div> <div><div></div><div><div>info</div><div>string</div></div></div>

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

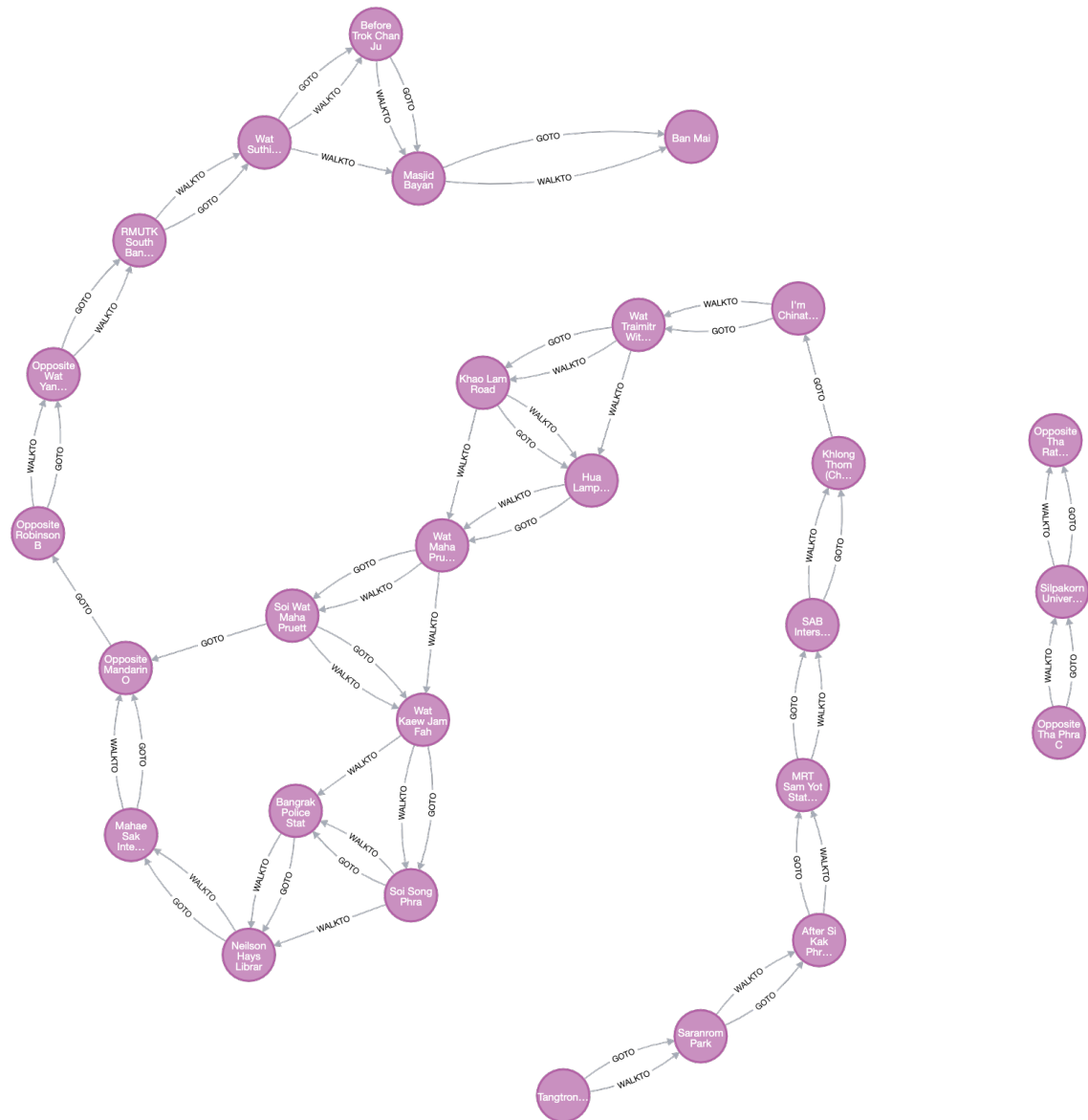


Figure 3.10: GOTO relationship

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

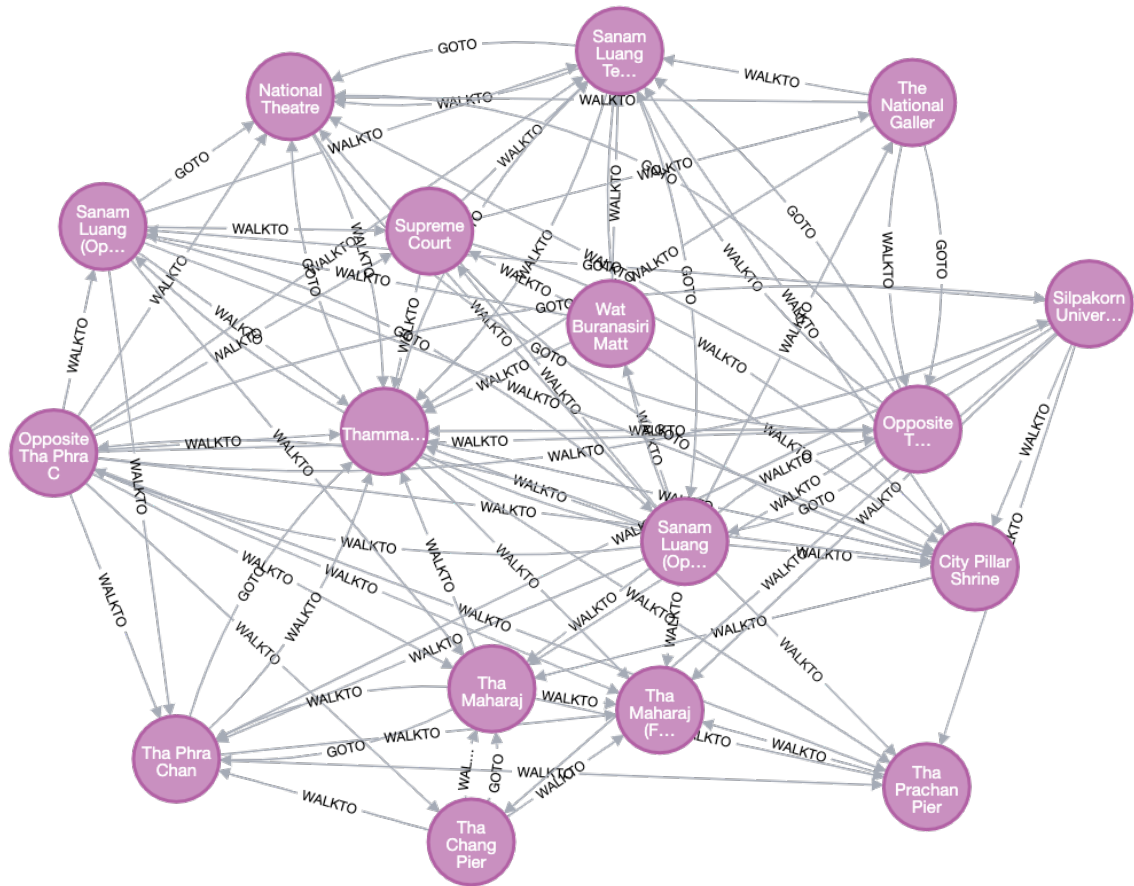


Figure 3.11: WALKTO relationship

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.



## Chapter 4

### System Functionality

#### 4.1 Introduction

This chapter describes the core aspects of the system's functionality, covering its architecture, primary functions, planning, and testing results that define the application's capabilities. The first part explains the system's architecture, and the second part details the main functionalities.

#### 4.2 System Architecture

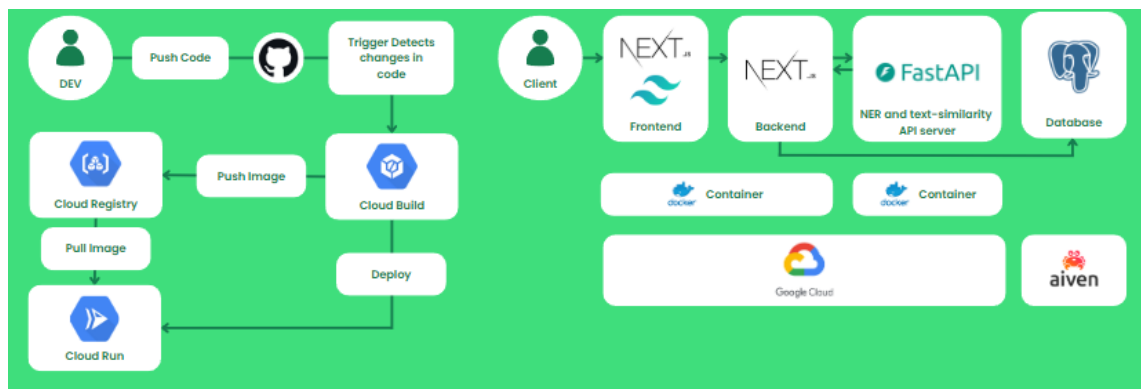


Figure 4.1: System Architecture

The system architecture consists of two main components designed to deliver an efficient lost-and-found platform. The front end, developed using Next.js, serves as the user interface for tasks like reporting lost items and browsing found items. It communicates with the backend via REST APIs and is containerized using Docker, deployed on Google Cloud Run for scalability. The backend, also built with Next.js and integrated with FastAPI, handles advanced functionalities like Natural Entity Recognition (NER) and text similarity algorithms to automate item matching. Data is managed using a PostgreSQL database hosted on Aiven, ensuring reliable and secure storage. The deployment process is streamlined with a CI/CD pipeline using Google Cloud Build, which automatically builds and deploys container images to Cloud Run, enabling an agile and scalable system.

### 4.3 Test Plan and Results

Module	Test Description	Expectation	Result
Authentication	Login with Microsoft Azure	Can log in with the KMUTT account	Success
Home Page	Navigate to Lost Report and Found Report forms	Can navigate to both forms	Success
Home Page	Navigate to the user profile page	Correct navigation to profile page	Success
Home Page	Log out from the profile	User logged out successfully	Success
Create Report Form	Fill in the form	Form is successfully filled and submitted	Success
Create Report Form	Validate required fields	Shows error for missing mandatory fields	Success
Create Report Form	Submit the report form	Report saved and reflected in the database	Success
Matching Process	Match lost items to found items	Matches displayed correctly with similarity score	Success
Matching Process	Display contact information of matched user	Contact information displayed correctly	Success
History Log	View previous reports filed by the user	Complete list of past reports displayed	Success
User Profile	Update user information (e.g., phone number)	User information updated successfully	Success

Table 4.1: Test Plan and Results

## 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.