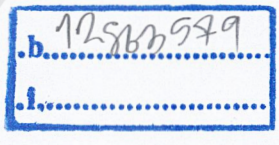
Smart Itinerary Planner Using Collaborative Filtering

Chanon Deeprasertkul
Tanapat Jiamsawas
Jarukit Phattanaporngul



Bachelor of Engineering in Software Engineering
International College
King Mongkut's Institute of Technology Ladkrabang
Academic Year 2016
KMITL-2017-IC-B-003-003

Thesis — Academic Year 2016

Bachelor of Engineering in Software Engineering

International College

King Mongkut's Institute of Technology Ladkrabang

Title: Smart Itinerary Planner Using Collaborative Filtering

Authors:

1. Mr. Chanon Deeprasertkul — Student ID 56090003
2. Mr. Tanapat Jiamsawas — Student ID 56090027
3. Mr. Jarukit Phattanaporngul — Student ID 56090037

Approved for submission

(Assistant Professor Dr. Chaiwat Nuthong)

Project Adviser

Date 24 / 6 / 2017

Abstract

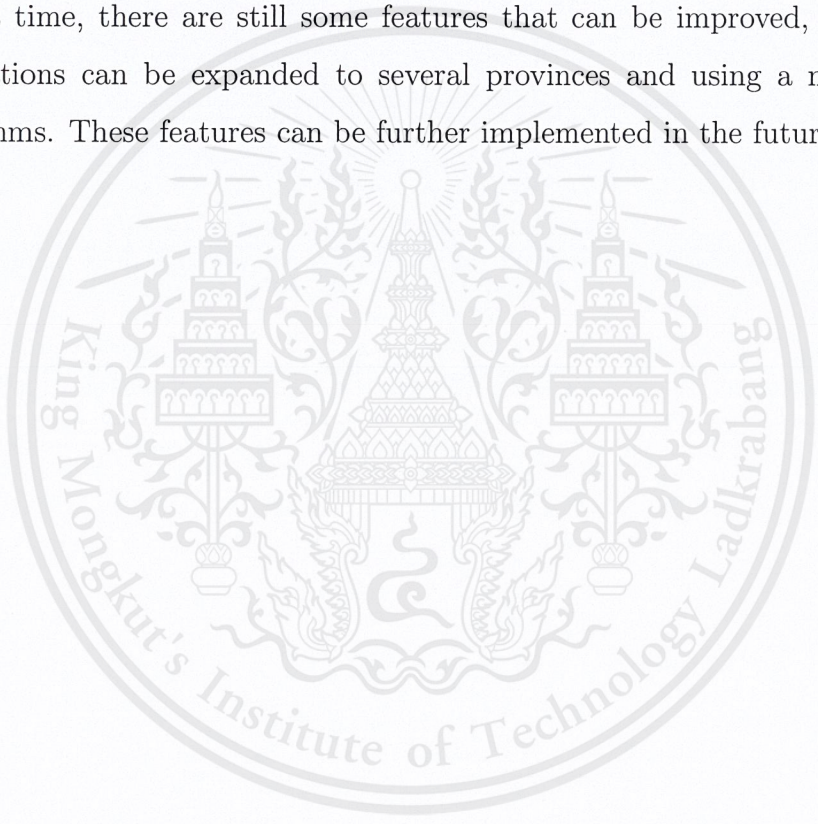
Nowadays, a tourism business in Thailand is a very popular business and becomes a major income of the country. One of the reasons many tourists decide to visit Thailand is that they want to experience and discover unseen attractions. However, in order to visit these unfamiliar destinations, travelers need to have a plan. Currently, there are many ways to plan a trip. However, in general, the trip planning process consumes lots of efforts and times. This is because it requires lots of necessary information such as, prices, places, directions, etc. Moreover, it might even become more challenge because of the differences in cultures and languages.

At the present, planning a trip contains several steps. It usually starts with travelers searching for an information from many websites, comparing it to each other and planing the route manually. This process usually takes days to complete. For these reasons, this project aims to develop a web application which can plan the trip automatically, yet can be manually customized later.

The web application proposed by this project is designed to be able to plan the trip which is preferable, well organized, and reasonable for an individual user. For this web application, the planed trip should satisfy the constraints of the given time and budget. The proposed system can be split into three main modules, i.e. web application, collaborative filtering, and filtering system. The first module is a web application. This part works as a user interface which collects the user information including budget and time constraints as well as user's travel preferences. The second part is a collaborative filtering. It is used to obtain the recommended places based on the input information from the user. Lastly, the third part is a filtering system. This is the module which search the best combination of the

destinations in order to plan the desired trip. In the proposed web application, the planning process starts with the collaborative filtering after receiving user information from web application. The recommended places will be generated and sent to the filtering system, which filters them according to the constraints. The obtained result, a planned trip, is returned and displayed to the user in the web application.

It is found from the experiments that the proposed system works well under the given scope of works. The trip is planned according to the constraints and user's preferences with less than ten seconds. Although the trip is generally planned in a short time, there are still some features that can be improved, for examples, destinations can be expanded to several provinces and using a more powerful algorithms. These features can be further implemented in the future works.



Acknowledgments

First of all, this thesis could not be completed if we do not have very great advices and dedicated time from our advisor, Asst.Prof.Dr. Chaiwat Nuthong. We would not have achieved this far without all the supports that we always received from him.

In addition, thanks for knowledges, suggestion, and comments from Dr. Isara Anantavrasilp that helps us through the hard time. Also we really appreciate the help and supports from our friends.

Finally, we most gratefully acknowledge our parents for all their supports throughout the period of this project.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Planning a trip usually requires searching on numerous websites and consulting several friends or guidebooks which often become outdated. People have to prepare and plan everything by themselves e.g., finding many tourist attractions and planning a trip for each day on a spreadsheet. According to the study from Expedia Media Solutions, it states that "travelers visit 38 websites over 21 days when planning a trip" [1]. Skimming through hundreds of websites while planning an upcoming trip is time consuming and inconvenient. This project proposes a solution which can significantly reduce planing time by applying an appropriate machine learning algorithm to the task at hand.

There are a few applications that already available. But several problems have been found after using these applications. Some applications are inflexible, users cannot create or change a plan by themselves. Instead, they need to pay for a plan that the applications already provided, while other applications just provide a huge amount of information of each place.

Because of these reasons, this project proposes a better solution which is to create a web-based application that be able to generate the trip based on user's information and preferences. It provides a plan with details such as what time they should visit each places and route plan for each day. Moreover, user can

customize the trip to exactly match their styles.

1.2 Objectives

The smart itinerary planner is developed to help travelers plan their trips at ease. The system provides all the necessary information in the trip. For example, time to visit a place, duration to spend, and activities to do for each places in the trip. Moreover, the system provides route navigation for travelers to use during the trip. The system provides many benefit to travelers. For example, reducing costs to plan the trip so that travelers don't need to book a tour and reducing time in such a way that travelers do not need to search for an overwhelmed information from many websites and decide which places they should go.

1.3 Scope of Problems

- Study and research tourist attractions in Phitsanulok. Find and analyze the detail for each place such as open and close time, entrance fee and related activities
- Develop a web-based application that obtains user's information, i.e. the destination, travel date, budget, preferred activities, preferences of tourist attractions and starting place in the trip. The system would generate the trip based on those information and allows users to customize the trip
- Study and apply an appropriate machine learning algorithm to use in the web application in order to generate the trip

1.4 Thesis Structure

The content of this thesis can be described as follows. Chapter 2 provides a summary of literature reviews and related application that similar to this project.

Chapter 3 provides a brief information of the algorithm and tools used to implement the system. Chapter 4 provides a software design of the system. Chapter 5 provides a brief information of the implementation process that has been experimented so far. Chapter 6 provides a summary of this project.

Chapter 2

Literature Review

Before the system was designed, works and applications that are similar to the system had been reviewed. The followings are research work and applications that have similar functionalities to the system.

2.1 Sygic Travel

Sygic travel [3] is a part of Sygic system which are automotive navigation systems for mobile phone. Sygic travel allows user to plan and organize the trips, and navigates them using their navigation systems. Sygic claims that the major advantages of this system are the following features: intuitive user interface, fast rendering and route calculation, low demand on processors and memory, automatic screen resolution adaption, smart recognition and self-setting to keyboard or touch screen operation and above all the multi-platform engine, all this to make the application starts quickly and runs smoothly on any device. Sygic travel, with these features, not only planning a trip, but also navigating the user using their navigation system. This makes Sygic travel is one of the best trip planning system.

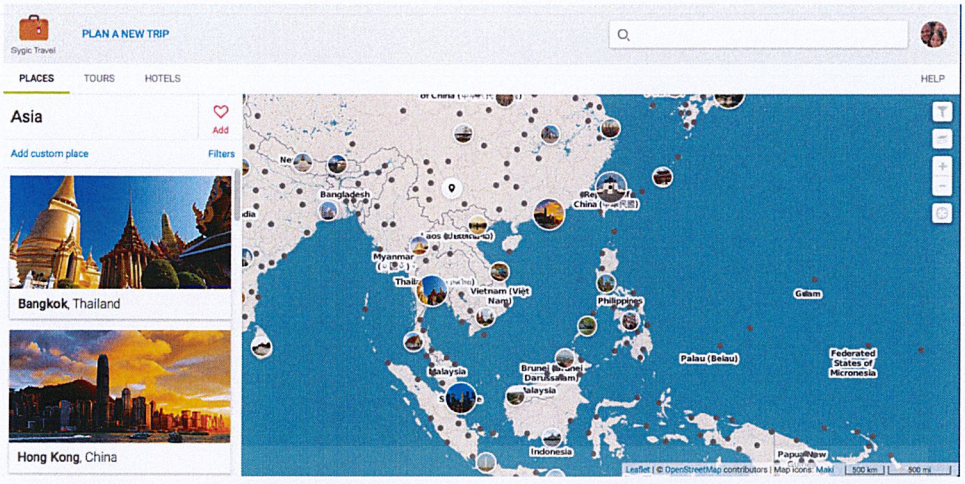


Figure 2-1: Sygic Travel: Trip Planner

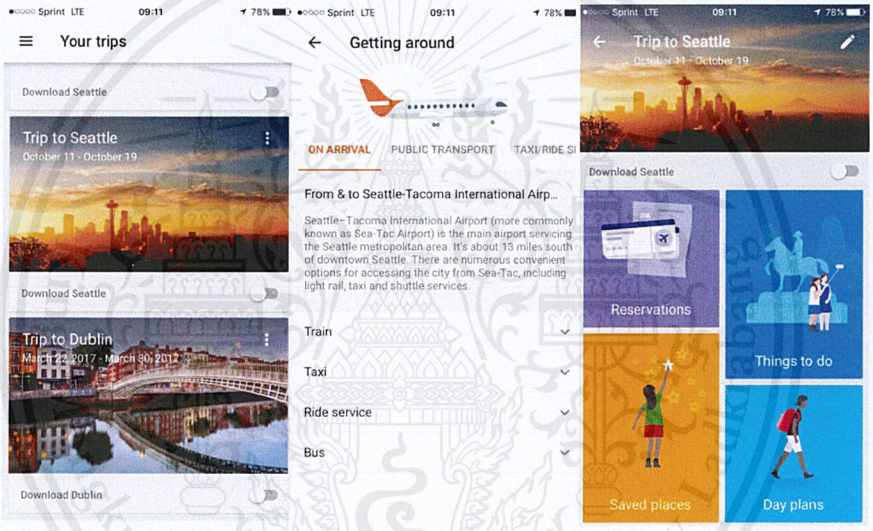


Figure 2-2: Google Trip Application

2.2 Google Trip

Google trip [2] is an application that allows user to plan and organize the trips. It automatically maps out a half day or a full day with suggestions for thing to see and do. User can also refresh the suggestions to see more nearby places and nearby attractions. Google trip provides the information of all the places in the trip such as hotel reservation, museum working time, etc. Moreover, Google trip is available offline, so user can see the info wherever they are.

2.3 Content-Based Recommendation Systems

Content-Based Recommendation Systems [4] is a research work of Michael J. Pazzani and Daniel Billsus. It presents a way to display the content based on user preferences. In this paper, the authors give an example of how to build the user profile using user preferences and represent contents based on that profile.

Figure 2-3: User customization in Amazon.com

Figure 2-3 is an example of user customization in Amazon.com. There are several methods and algorithms to filter the content-based on user's profile. The followings are example of methods and algorithms that were used in content-based recommendation system [5].

- Probabilistic Methods and Naive Bayes [6]
- Decision Tree and Rule Induction [6] [7]
- Nearest Neighbor Methods [7]
- Relevance Feedback and Rocchio's Algorithm [8]

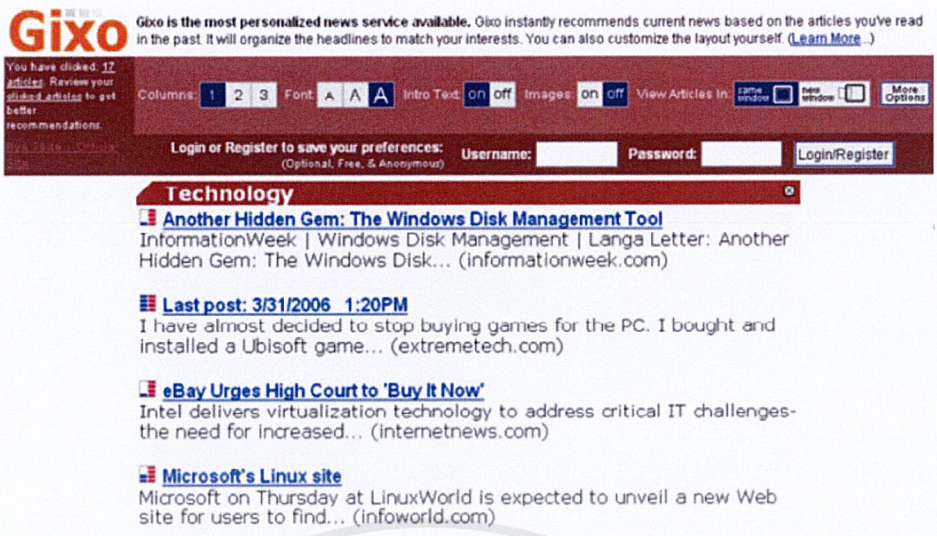


Figure 2-4: Gixo presents personalized news based on similarity to articles that have previously been read

Although there are several methods and algorithms to make a user's profile and using with content-based recommendation system, no content-based recommendation system can provide a good recommendations if the contents do not contain enough information to classify items that user likes and does not like.

After we had reviewed several applications, we still detect some improvement. For example, some applications require too much users' information and some applications are inflexible. Because of those reasons, we decided to develop a web application. The trips from the application must be flexible and the application itself does not require much users' information.

Chapter 3

Methodology

In order to achieve the objectives, several methods have been researched. The followings are knowledge and methods which help achieve our goals.

3.1 Collaborative Filtering

Collaborative filtering [9] is a method that uses a given rating dataset that contains many items as the basis for predicting and/or creating a recommendation list for a given user. Usually, rating dataset will be stored in a form of $M \times N$ rating matrix R, where M is a set of users $U = \{u_1, u_2, u_3,...,u_M\}$ and N is a set of items $I = \{i_1, i_2, i_3,...,i_N\}$.

Set of users	Set of items							
	i_1	i_2	i_3	i_4	i_5	i_6	i_7	i_8
u_1	5.0	?	4.0	?	2.0	1.0	?	3.0
u_2	?	1.0	?	4.0	4.0	3.0	5.0	2.0
u_3	4.0	?	4.0	1.0	3.0	2.0	5.0	4.0
u_4	1.0	?	2.0	?	3.0	1.0	?	2.0
u_5	3.0	3.0	?	4.0	5.0	?	3.0	1.0
u_6	2.0	4.0	?	4.0	?	5.0	4.0	?
Given user a	Set of items rated by given user a							
u_a	2.0	5.0	?	?	3.0	1.0	?	?

Table 3.1: Example of real rating matrix R where u_a is a given user

This rating matrix will be used to calculate the similarity for a given user u_a. The similarity can be defined by using k nearest neighbors algorithm. Popular similarity measures for Collaborative filtering [10] are the Pearson correlation coefficient [11] and the Cosine similarity [12]. These similarities measured are defined between two users, u_x and u_y as

$$\text{sim}_{\text{Pearson}}(\vec{x}, \vec{y}) = \frac{\sum_{i \in I}(\vec{x}_i \vec{x})(\vec{y}_i \vec{y})}{(|I| - 1)sd(\vec{x})sd(\vec{y})} \tag{3.1}$$

and

$$\text{sim}_{\text{Cosine}}(\vec{x}, \vec{y}) = \frac{\vec{x} \cdot \vec{y}}{\|\vec{x}\| \|\vec{y}\|} \tag{3.2}$$

where $\vec{x} = r_x$ and $\vec{y} = r_y$ represent the row vectors in Matrix R with the two users' profile vectors. $sd(\cdot)$ is the standard deviation and $||\cdot||$ is the l^2 - norm of a vector.

According to Table 3.1, using k nearest neighbor, where $k = 3$, the user u_4, u_5, and u_2 are selected. This data will be used to predict the rating for a given user u_a where $\hat{r}_a$ represents the rating that user u_a misses. The easiest form is to average the ratings in the neighborhood.

Set of users	Set of items							
	i_1	i_2	i_3	i_4	i_5	i_6	i_7	i_8
u_1	5.0	?	4.0	?	2.0	1.0	?	3.0
u_2	?	1.0	?	4.0	4.0	3.0	5.0	2.0
u_3	4.0	?	4.0	1.0	3.0	2.0	5.0	4.0
u_4	1.0	?	2.0	?	3.0	1.0	?	2.0
u_5	3.0	3.0	?	4.0	5.0	?	3.0	1.0
u_6	2.0	4.0	?	4.0	?	5.0	4.0	?
Given user a	Set of items rated by given user a							
u_a	2.0	5.0	?	?	3.0	1.0	?	?
$\hat{r}_a$			2.0	4.0			4.0	1.66

Table 3.2: Example of real rating matrix R where u_a is a given user and $\hat{r}_a$ is the missing rating of the given user

According to Table 3.2, the item i_3, i_4, i_7 and i_8 are recommended to the user respectively. However, in real life application, the system should not recommend i_3 and i_8 due to their low ratings compare to i_4 and i_7.

3.2 Filtering system

Filtering system is divided into 3 parts. The system is implemented in Python and the initial list is the list of every possible places. Figure 3-1 shows the work flow of the system.

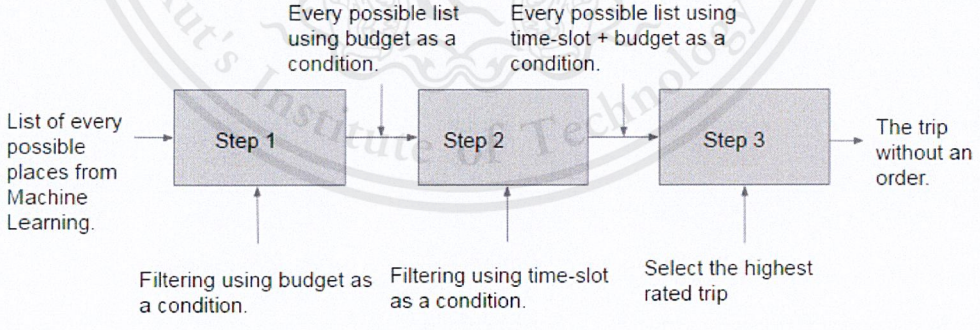


Figure 3-1: The diagram shows each parts of the filtering system

First, filtering with the cost of the trip. The system is going to collect user's requirements about the budget from the web page. Any trip costs higher than this budget will be filtered out from the list.

Secondly, filtering with time in the trip. In this part, the system already gets the list of the trip that is satisfied the budget requirement. The system uses user's requirements about the time from the web page. Any trip that doesn't satisfy the time constraint will be filtered out from the list in this part.

The last process is selection. The system already gets the list of the trip that satisfies both budget and time constraints. Thus, the system has to select the best trip to represent to the user. Each place has its own rating so the combination that has the highest rating will be selected.

3.3 Trip planning system

Trip planning system has a purpose to decide which place should be visited first by calculating from the distances. The system has to get the user's requirements about starting point from the web page. The system uses google-map application programming interface to help with the planning system. System has to measure the real distance between each places in the trip. To achieve this, the system has to send a request to google-map application programming interface to get the distance between each places. Then, the system selects the place that has the least distance from the starting point. The system's going to uses that place as a next starting point. The system would recursively do this process until it gets the completed plan.

Chapter 4

Software Design

4.1 System Requirements

4.1.1 Functional Requirements

The functional requirements of our smart itinerary planner can be summarized below:

- The system allows a user to generate the trip based on user's preferences.
- The system allows a user to change the starting point in the trip.
- The system allows a user to change and remove places in the trip.
- The system allows a user to export the trip as a PDF file.

4.1.2 Non-Functional Requirements

The non-functional requirements of our smart itinerary planner can be summarized below:

- The system uses HTML,CSS and JavaScript to develop web application.
- The system uses AngularJS as a client-side framework and Node.JS as a server-side framework.

- The system uses Google Maps API as a service to show the map, plan the route and calculate duration and distance between places.
- The system uses MySQL to manage database.
- The system uses Python as a connector between web application, recommending system, filtering and planning system .
- The system uses R language to create the recommendation engine using collaborative filtering algorithm.

4.2 Use cases

There are three main functions within the use case diagram as shown in Figure 4-1, namely, generate a trip, customize trip and export trip to PDF file.

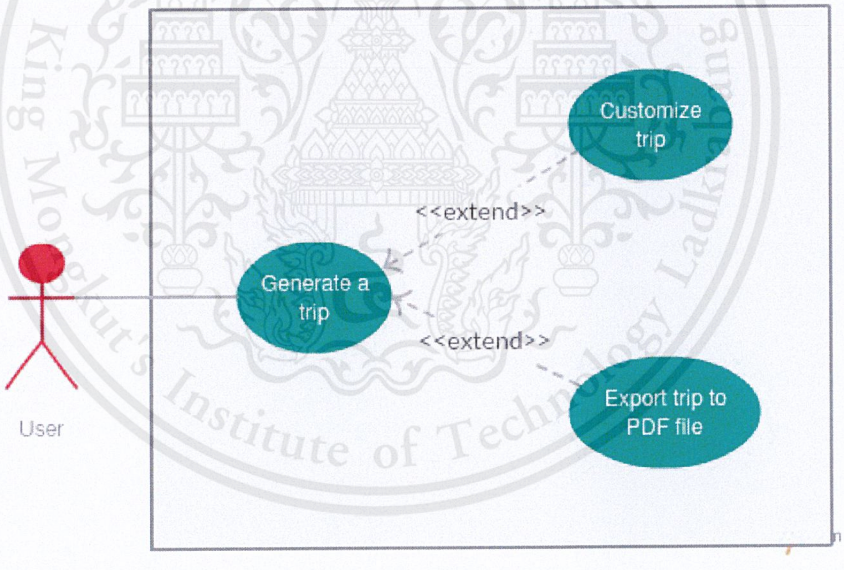


Figure 4-1: Use case diagram for Smart Itinerary Planner

4.2.1 Use case description

- **Expanded description of "Generate a trip"**

  Previous case: -

  Use case: Generate a trip

  Actor: User

  Goal: To obtain the generated trip based on user information and preferences.

  Overview: This is a step that user has to input information and preferences which are destination, budget, date, prefered activities and places and starting place in order to get a trip result.

  Typical course of events:

User	System
1. User inputs information, preferences and starting place	
	2. The system generates and shows the trip based on user's information and preferences.

- **Expanded description of "Customize a trip"**

  Previous case: Generate a trip

  Use case: Customize a trip

  Actor: User

  Goal: To customize the generated trip by adding, changing or removing place from the trip.

  Overview: This is a step that allows user to customize the trip to exactly match user's style by adding, changing or removing place from the trip.

  Typical course of events:

User	System
	1. The system shows the result trip
2. User customizes trip by adding, changing or removing place	
	3. The system updates estimated time, distance and route of the place that has been modified.

- **Expanded description of "Export trip to PDF file"**

  Previous case: Generate a trip

  Use case: Export trip to PDF file

  Actor: User

  Goal: To export the trip as a PDF file.

  Overview: This is a step that allows user to export the trip as a PDF file in order to use without opening a web application.

  Typical course of events:

User	System
1. User clicks "Export" button	
	2. The system exports the whole trip into a table and allows user to download.

4.3 System overview architecture

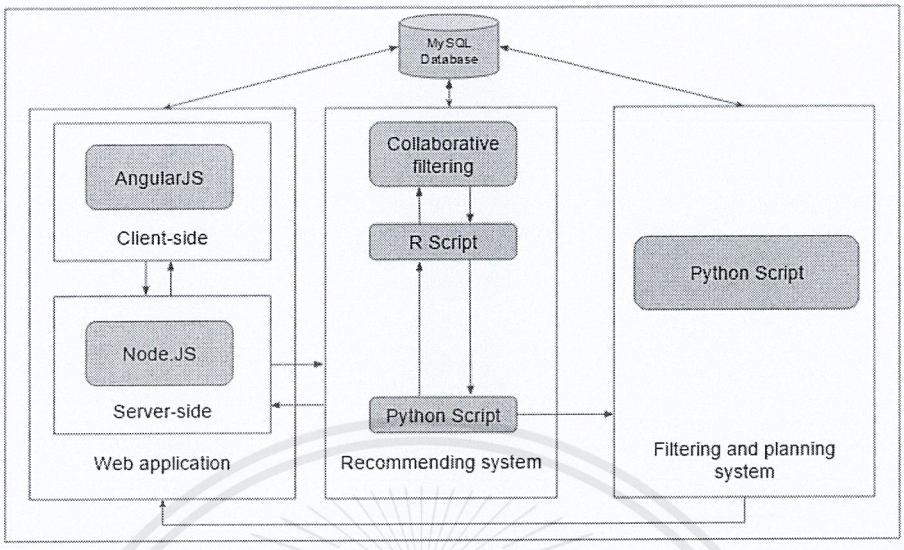


Figure 4-2: System overview architecture

The system consists of the 3 main parts which are web application, recommending system and filtering and planning system. Figure 4-2 describes the entire system.

4.3.1 Web Application

Web application is a part that obtains information from user i.e. destination, date, budget, preferred activities, user's preferences of tourist attractions and starting place. Then, it sends those information to recommendation part. After filtering and planning system get the trip, the server-side would obtain result and sends to client-side in order to display to user.

4.3.2 Recommending System

Recommending system is a part where the system recommends the places to user according to user's preferences. It obtains the user's input from the web application i.e. user's preferences, user's set of rating places and proceeds. The system uses the followings data to find which users have similar preferences. Lastly, the system

will generate a set of recommended places as an output and passes it to the filtering and trip planning system.

4.3.3 Filtering and trip planning System

Filtering and planning system is a part that obtains set of recommended places from the recommending system. After the filtering and planning system are executed, the result will be sent back to the web application to present a completed trip to the user.

Filtering system contains three sub parts. Each part has its own purpose. The first part is used to filter the places out using the budget as a constraint and returns the combination of places. The combinations will be used as an input for the second part. The second part is used to filter the places out using the time as a constraints and returns the combination of the places. The last part will select the combination that has the highest rate. The result will be used as an input of the trip planning system. Trip planning system orders the combination of the places by measuring the distances between each place.

Chapter 5

Experimental

In this chapter, the implementation and experimental of this project will be explained. As mentioned earlier, this project is divided into three parts, web application, recommendation, filtering and routing. Web application is implemented using HTML,CSS and JavaScript. Recommendation system is implemented using Python and R programming and the filtering and routing are implemented using Python. Finally, these three parts are integrated together.

5.1 Web Application

Web application allows user to interact with the system by giving information and preferences, view the trip result,customize and export the trip. Client-side of the website is implemented in HTML,CSS and JavaScript framework which is AngularJS. For the server-side, it is implemented using JavaScript framework which is Node.JS. There are two main pages in the web application, index page and trip result page.

-

5.1.1 Index Page

Figure 5.1 is the first page in the system. It allows user to input information and preferences i.e. destination, budget, date, preferred activities, preferences of tourist attractions and starting place. Furthermore, the system randomly selects 6 places from the database as shown in Figure 5.2. User has to give the score for at least 3 places based on user's preferences. Finally, user needs to choose the starting point in the trip which is usually a hotel where user stays as shown in Figure 5.3.

Figure 5-1: Index Page

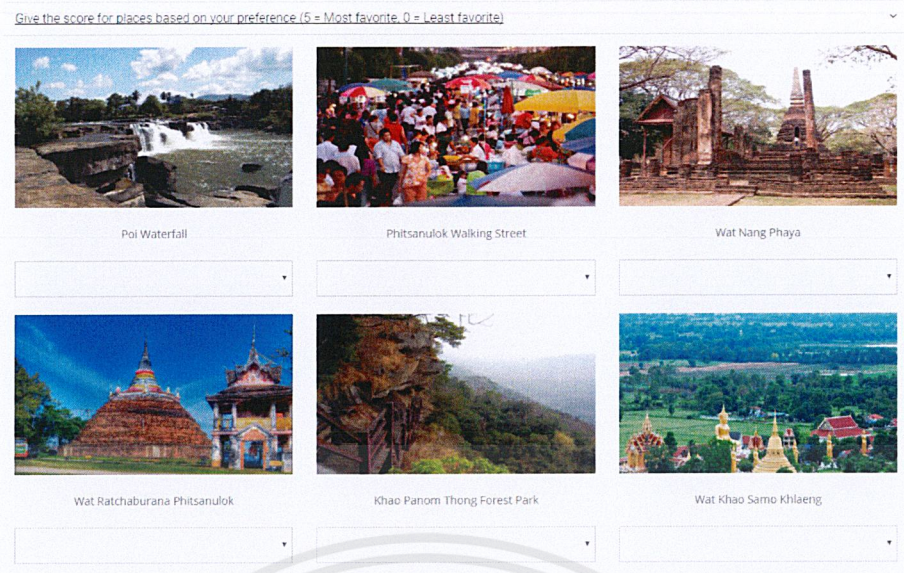


Figure 5-2: Rating place part in index page

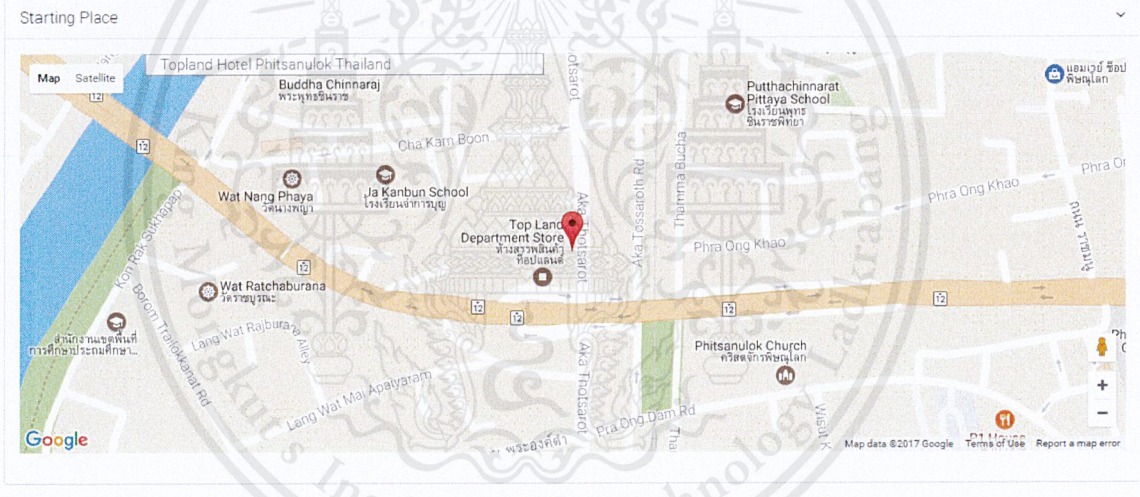


Figure 5-3: Starting place part in index page

5.1.2 Trip Result Page

After user submit information and preferences, the system will show the trip result page that contains the information of the trip (Part A), plan for each day (Part B) and provides additional features (Part C) as shown in Figure 5.4. At the top of the page, it shows information of the trip i.e. duration, destination and date as shown in Figure 5.5. In Figure 5.6, it shows a plan for each day in the trip. There are 4 main places i.e. starting place, place to visit in the morning, afternoon and evening. The system provides all necessary information for each destination i.e. address, telephone number, cost and category. It also provides time to visit, duration and distances to travel between each places. User is able to customize the trip by adding,changing or removing the place but the maximum places in the plan is 4 places and the starting place is required. Moreover, the system provides the route of a plan as shown in Figure 5.7. Furthermore, the system provides several additional features on the left side of the page as shown in Figure 5.8. It allows user to update information i.e. destination, budget and date of the trip in order to generate a new trip. Lastly, user is able to export the generated trip as a PDF file to use later on. The system also allows user to give feedback back to the system whether user like the trip or not.

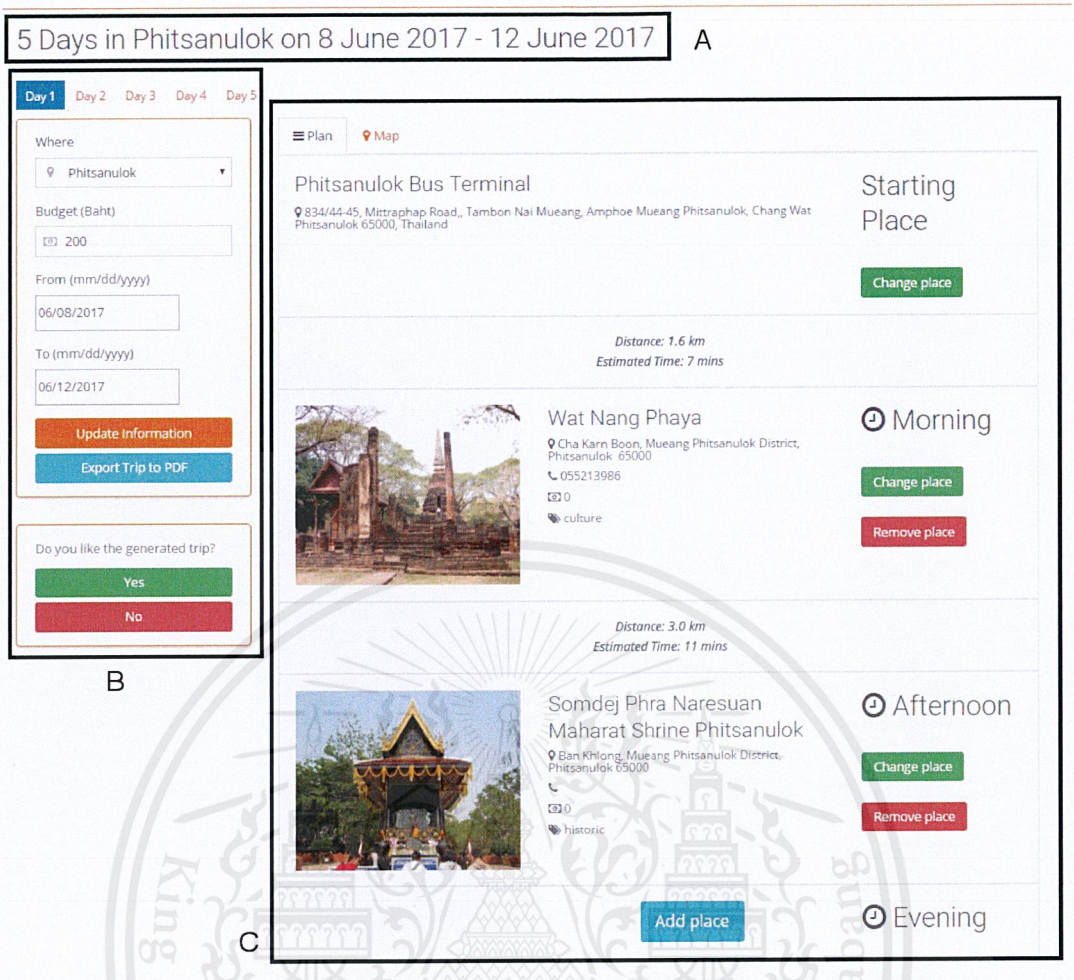


Figure 5-4: Trip result page

5 Days in Phitsanulok on 8 June 2017 - 12 June 2017

Figure 5-5: Information of the trip

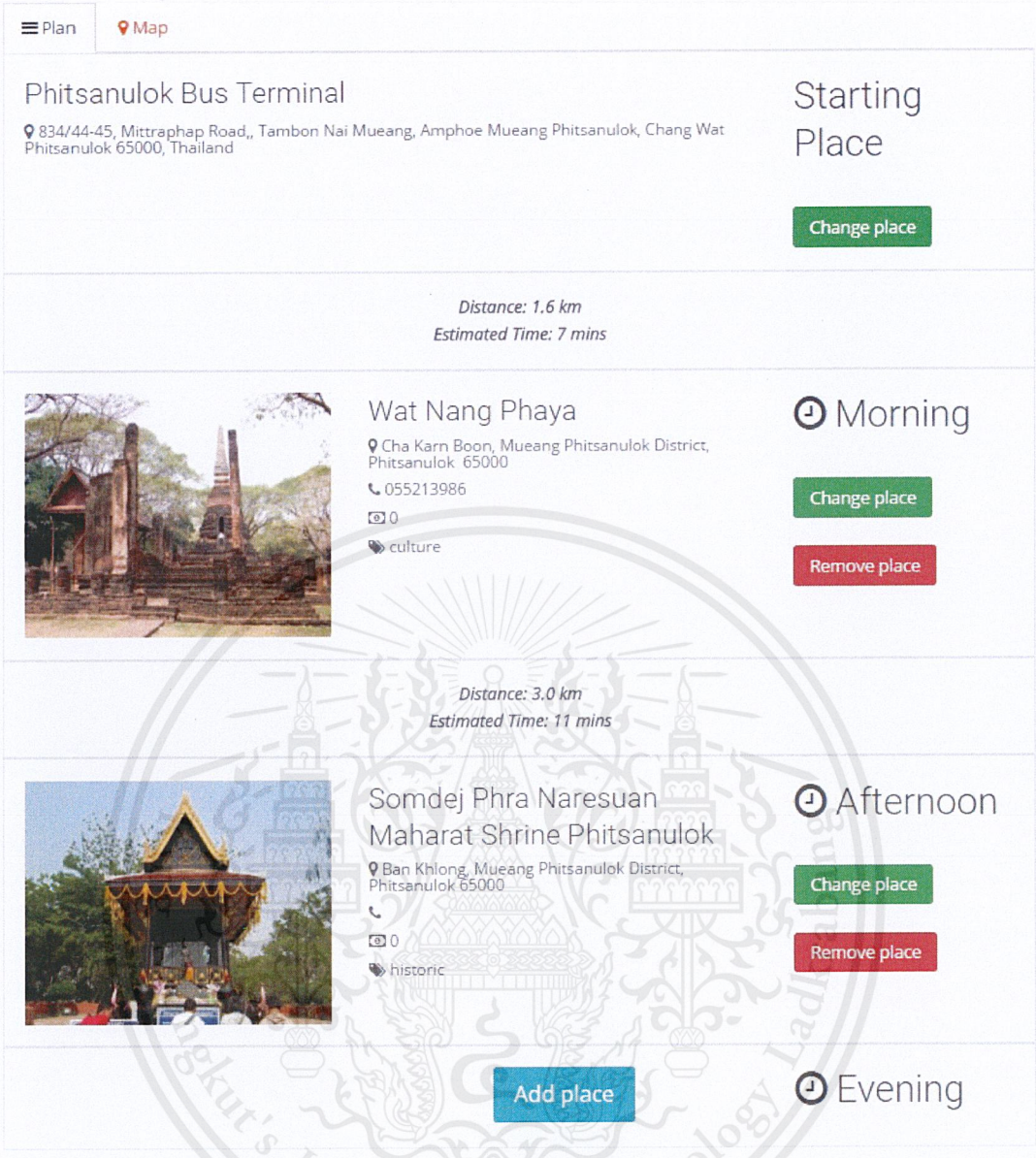


Figure 5-6: Plan for each day

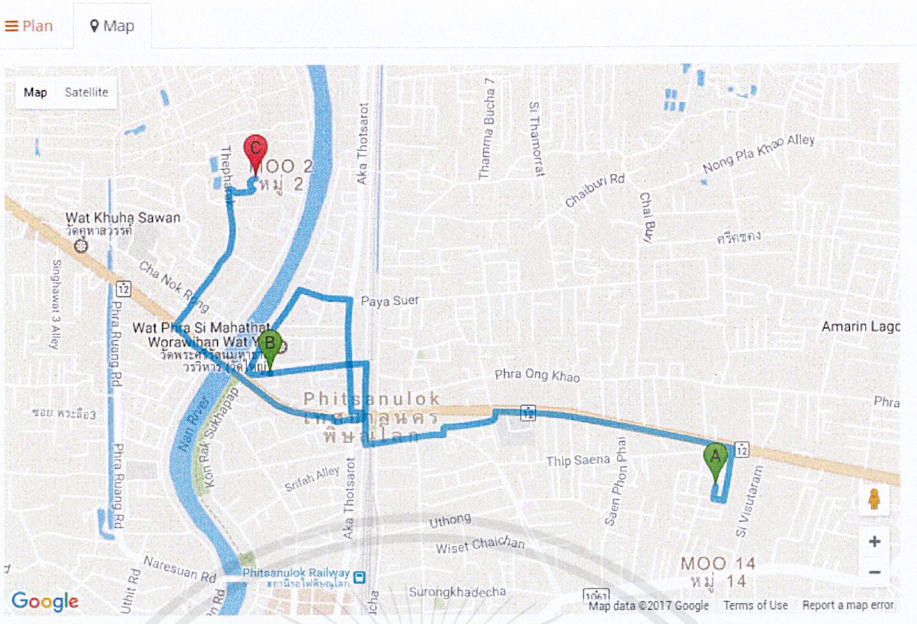


Figure 5-7: Route of a plan

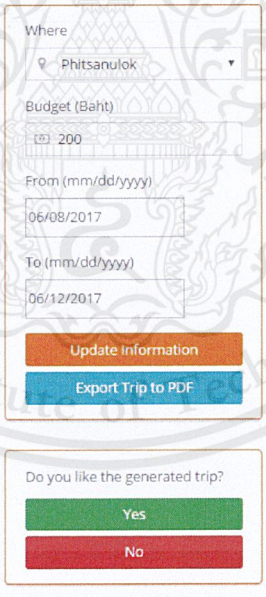
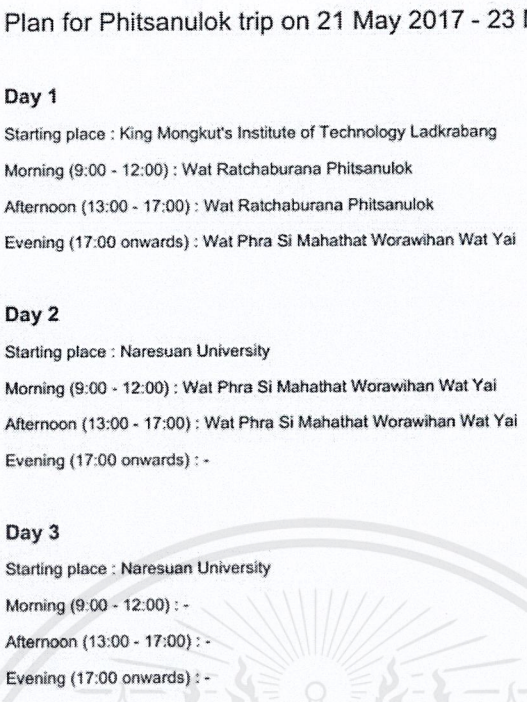


Figure 5-8: Additional features


Plan for Phitsanulok trip on 21 May 2017 - 23 May 2017

Day 1

Starting place : King Mongkut's Institute of Technology Ladkrabang

Morning (9:00 - 12:00) : Wat Ratchaburana Phitsanulok

Afternoon (13:00 - 17:00) : Wat Ratchaburana Phitsanulok

Evening (17:00 onwards) : Wat Phra Si Mahathat Worawihan Wat Yai

Day 2

Starting place : Naresuan University

Morning (9:00 - 12:00) : Wat Phra Si Mahathat Worawihan Wat Yai

Afternoon (13:00 - 17:00) : Wat Phra Si Mahathat Worawihan Wat Yai

Evening (17:00 onwards) : -

Day 3

Starting place : Naresuan University

Morning (9:00 - 12:00) : -

Afternoon (13:00 - 17:00) : -

Evening (17:00 onwards) : -

Figure 5-9: PDF file after export the trip

5.2 Recommendation

Recommendation is the part that recommends places to user based on user's preferences. It uses machine learning technique to achieve this. The recommendation engine is implemented using Python and R. R is used for creating the recommendation engine. Whereas, Python is used to connect every part of the system together. To develop and test recommending algorithms, "recommenderlab" package in R is used. This project uses 2 datasets along with the collaborative filtering method to recommend the places to the user.

5.2.1 Collaborative Filtering in Recommendation Engine

Collaborative filtering method in this project uses scores of places that user gives and user's preferences from the web application as the information to create the recommendation engine.

User Number	Places							
	A	B	C	D	E	F	G	H
1	5	?	4	?	?	4	?	?
2	4	5	4	5	4	5	4	5
3	5	5	5	5	5	1	1	5
4	5	5	5	5	4	5	5	5
.				.				.
.				.				.
77	5	4	3	5	4	3	5	5
78	4	5	2	1	3	2	5	5
79	3	4	3	3	2	2	2	5
80	5	5	5	5	3	3	3	4

Remarks

A : Wat Phra Si Mahathat Worawihan Wat Yai

B : Thung Salaeng Luang National Park

C : Wat Khao Samo Khlaeng

D : Phu Hin Rong Kla National Park

E : Wat Rachaburana Phitsanulok

F : Somdej Phra Naresuan Maharat Shrine

G : Kaeng Sopa Waterfall

H : Phu Soi Dao National Park

Table 5.1: Dataset of rating places that used to create recommendation engine

According to Table 5.1, the dataset for testing the algorithm is gathered from 80 persons with 18 places. The different values in the matrix mean that the user has rated the place on a scale of 1-5, where 1 is the least satisfaction and 5 is the most satisfaction, while a missing ratings mean that the user has not visited that place or does not know about it.

Table 5.2 illustrates the dataset of user's preferences that gathered from 80 persons. The value 1 in the matrix means that the user has that preference, while 0 means that the user does not have that preference.

5.3 Filtering System

As explained in section 3.3, the authors divide the filtering system into 3 parts.

The first part is to filter out the combination which the overall cost is higher than the budget constraint. The result is the combination of the places which passes the budget constraint. The results are shown in the Figure 5-10.

User Number	User's Preferences				
	Culture	Adventure	Natural	History	Art
1	0	0	1	1	0
2	0	0	1	0	0
3	1	1	1	1	1
4	1	1	0	0	0
.	.	.			.
.	.	.			.
77	1	1	0	0	0
78	1	1	1	1	1
79	1	0	0	1	0
80	0	0	1	0	1

Table 5.2: Dataset of rating preferences that used to create recommendation engine

Second part is to filter out the combination which does not pass the time constraint. The system divides the time frame into three slots. There are morning, afternoon and evening. Each place can belong to only one period of time. The result will be the combination of the places which passes the time constraint. In other word, the result is the combination of places that user can visit within the time constraint. Figure 5-11 shows the result after applied this function into the result from the filtering_with_time function

There are two combinations which pass the time constraint. Thus, there are two trips for the user at this state. First trip is to visit Waddee Temple in the morning and visit Nightfall Street in the afternoon. Second trip is to visit Night Waterfall in the morning and visit Nightfall Street in the afternoon. The last part of the filtering system is going to decide which combination is the best trip for user.

```
----------------------Filtering with budget----------------------
Trip : 0
Waddee Temple
Trip : 1
Night Waterfall
Trip : 2
Rama's Garden
Trip : 3
Nightfall Street
Trip : 4
Waddee Temple
Nightfall Street
Trip : 5
Night Waterfall
Nightfall Street
-----------------------------------------------------------------
```

Figure 5-10: The result after apply the filtering_with_cost function

```
----------------------Filtering with time-----------------------
Trip : 0
Waddee Temple
Nightfall Street
Trip : 1
Night Waterfall
Nightfall Street
-----------------------------------------------------------------
```

Figure 5-11: The result after apply the filtering_with_time function

The last part is the selection system. The system selects the combination which has the highest overall rating. The system is going to query rating of each places from the recommendation part. Then, it calculates the rating of each combination and selects the highest combination. The selected combination will be the result from the filtering system and sent to trip planning system. The Figure 5-12 shows the result of the selection system.

5.4 Trip planning System

The purpose of trip planning system is to order the combination of places in a suitable order. Trip planning system calculates the distance between each places and finds the nearest place. Trip planning system uses the real distance using google-map application programming interface (API). In case there are more than one day in the combination, the system would restart the starting point to the initial point at the end of the time frame. The initial starting point is collected from the website. After the system founds the first place, the starting point will be changed to the coordinate of that place. The system will recursively executes this process until there is no place in the combination.

```
---------------Select the trip by rating system-----------------
Trip : 0
Night Waterfall
Nightfall Street
----------------------------------------------------------------
```

Figure 5-12: The result with the highest rating

```
---------------Rearrange_Trip using distance--------------------
Starting Cordinations : 13.738357,100.5323
Nightfall Street
Night Waterfall
----------------------------------------------------------------
```

Figure 5-13: The result after apply the Planning function with the result from filtering system

The result is the combination of the places with a suitable order. The result will be sent back to the web application to represent to user.

5.5 Running Time

This experiment has been tested on the machine which specifications are as follows: Windows 10, 3rd Generation Intel® Core™ i7 Processor, 8GB RAM. The experiment has been done using different users' information. The duration on the planning time are 1 day to 5 days. Each day are tested for 10 times.

According to the Table 5.3, the running time of the application is gathered from 10 experiments. The average running time per planning duration are as follows: 3.63 seconds for 1 day, 5.56 seconds for 2 days, 7.73 seconds for 3 days, 9.42 second for 4 days and 9.58 seconds for 5 days.

Experiment	Running time per planning duration (sec)				
Number	1 Day	2 Days	3 Days	4 Days	5 Days
1	3.38	5.34	7.70	8.67	9.61
2	3.71	5.22	6.45	9.76	9.74
3	3.85	4.86	8.57	9.93	9.50
4	3.77	5.40	7.29	9.87	9.62
5	3.75	6.13	7.91	8.78	9.51
6	3.47	6.35	7.78	8.95	9.33
7	3.83	5.75	8.67	9.42	9.42
8	3.72	5.45	8.51	9.93	9.57
9	3.62	5.87	7.42	9.78	9.63
10	3.18	6.08	6.95	9.12	9.88
Average	3.63	5.65	7.73	9.42	9.58

Table 5.3: Running time per planning duration

Chapter 6

Conclusion and Discussion

6.1 Conclusion

In this project, several methods have been studied in order to developed the web application that be able to plan the trip based on the user's information i.e. budget, preferred activities. Many algorithms i.e. decision tree and k-nearest neighbour has been studied. However, the problem is our data is limited and those algorithms required a different data structure to process. Collaborative filtering is another algorithms that has been studied and used to solve this problems. Finally, the authors successfully developed a web-based application that help users plan a trip based on their information. The web-based application has reduced the planning time to less than 10 seconds

6.2 Discussion

Nowadays, the travelling business grows widely in Thailand. There are lots of tourist attractions. Each place has different purposes to satisfy the visitors. In order to get the best experience from travelling, tourists need to plan a trip to guide them while visiting. However, the planning steps consume lots of time and may be inconvenient to some people.

Because of these reasons, the authors wanted to implement the web-based ap-

plication that can be used to generate the travelling trip for the visitors. The system is designed to has three main modules, web application, recommending system and filtering and trip planning system. The web application allows user to interact with with the system by giving information and preferences, customizing and exporting the trip. Recommending system is used to generate the recommended places based on user's preferences. Filtering and planning system module is used to filter the trip, generate the best choice depends on the constraints from the user, and routing the trip.

After these three modules are integrated, the authors have tested and collected the user's feedback. We come to the conclusion that we have satisfied the goals we started in the beginning of this thesis, the system can provide a reasonable trip that satisfy the users according to the user's feedback.

6.3 Problems and obstacles

6.3.1 Dataset

Dataset is the first and biggest problem in this project project, since the machine learning field involves data, large amount of data is needed in order to train and test the algorithms. However, the authors do not have data about travelling and it took a lot of time to gather by making the questionnaire to gather the data.

6.3.2 Machine Learning

Machine learning is our new field in study, the authors do not have any knowledge about machine learning at the start and spent one semester to figure how machine learning could apply to the project.

6.3.3 Algorithm

There are many algorithms that can be used to create the recommendation engine, but in this project, the authors decided to use collaborative filtering after

studied and considered some other algorithms. In the future, we can apply other algorithms that faster and more powerful to create the recommendation engine and get a better result.

6.4 Future Work

- Improve the algorithm used to find related user to a more powerful one
- Improve the algorithm used to filtering and planning the trip for user to a more powerful one
- Expand the scope to many provinces
- Make the web application more user friendly
- Make the web application handle many requests at the same time

References

[1] Barbara Peterson, 'Consumers Visit 38 Sites Before Booking, Expedia Says', Tarvel Market Report, 2015. [Online]. Available: http://www.travelmarketreport.com/articles/Consumers-Visit-38-Sites-Before-Booking-Expedia-Says

[2] Jessica Plautz, 'Google Launches Trips App to Be Your Guide Around the World', Travel and Leisure, 2016. [Online]. Available: http://www.travelandleisure.com/travel-tips/mobile-apps/google-trips-app

[3] Jb Macatulad, 'Sygic Travel: Travel Planning Made Easier (Formerly Known as Tripomatic)', Willy fly for food, 2016. [Online]. Available: http://www.willflyforfood.net/2016/05/26/sygic-travel-the-awesome-travel-planning-app-formerly-known-as-tripomatic/

[4] Michael J. Pazzani and Daniel Billsus. *Content-Based Recommendation System.* Rutgets University, New Brunswick, New Jersy, 2007

[5] Ivens Portugal, Paulo Alencar, Donald Cowan. *The User of Machine Learning Algorithms in Recommender Systems: A Systematic Review.* University of Waterloo, ON, Canada, 2015

[6] Felden, C., Chamoni, P. *Recommender systems based on an active data warehouse with text documents.* In System Sciences, 2007. HICSS 2007. 40th Annual Hawaii International Conference on (pp. 168a-168a). IEEE.

[7] Marović, M., Mihoković, M., Mikša, M., Pribil, S., Tus, A. *Automatic movie ratings prediction using machine learning.* In MIPRO, 2011 Proceedings of the 34th International Convention (pp. 1640-1645). IEEE.

[8] Pazzani M., Billsus, D. *Learning and Revising User Profiles: The Identification of Interesting Web Sites.* Machine Learning 27(3) (1997) 313-331

[9] Michael Hashler. *recommenderlab - A Framework for Developing and Testing Recommendation Algorithms.* Southern Methodist University, Texas, USA, 2017

[10] Haifeng Liu , Zheng Hu, Ahmad Mian, Hui Tian, Xuzhen Zhu *A new user similarity model to improve the accuracy of collaborative filtering.* Beijing University of Posts and Telecommunications, China, 2013

[11] P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, J. Riedl, *GroupLens: an open architecture for collaborative filtering of netnews*, in: Proceeding of the ACM Conference on Computer Supported Cooperative Work, 1994, pp. 175–186.

[12] G. Adomavicius, A. Tuzhilin *Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions* IEEE Trans. Knowl. Data Eng., 17 (6) (2005), pp. 734–749

Appendices

A Questionnaire

In order to build the recommendation engine, the authors created questionnaire and gathered responses. There are 2 types of data that we used. First is users' preferences and second is users' rating of places.

A.1 Users' preferences

Figure A.1 shows the result of the users' preferences in summary. The data was gathered from 80 persons and each person can have more than one preferences.

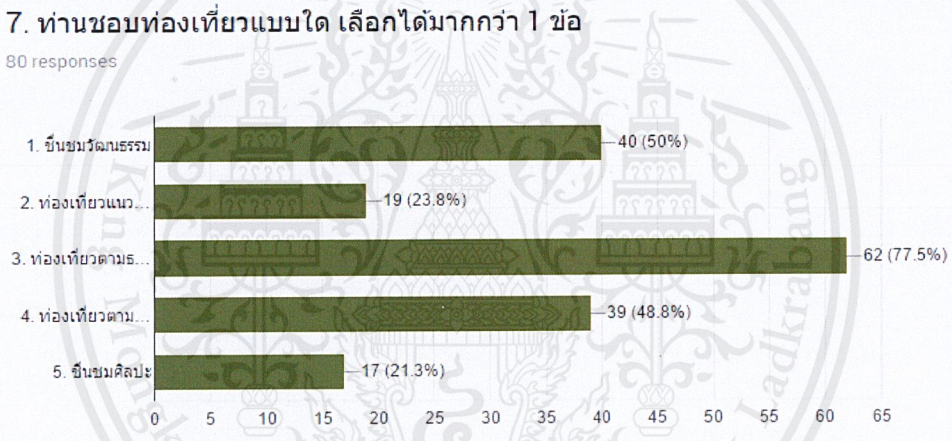


Figure A.1: Users' Preferences

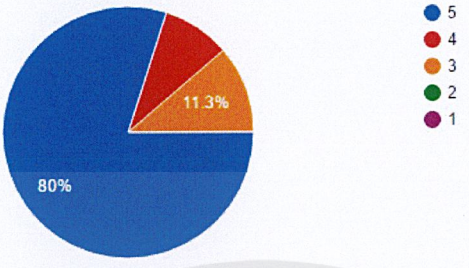
A.2 Users' rating of places

Figure A.2 shows the result of the users' rating of places in summary. The data was gathered from 80 persons. There are 18 places in the questionaire. Only 2 places, Wat Phra Si Mahathat Worawihan Wat Yai and Thung Salaeng Luang National, are shown in Figure A.2.

ส่วนที่ 3 ความพึงพอใจเกี่ยวกับสถานที่ท่องเที่ยวในจังหวัดพิษณุโลก

1. วัดพระศรีรัตนมหาธาตุวรมหาวิหาร

80 responses



2. ทุ่งแสลงหลวง

74 responses

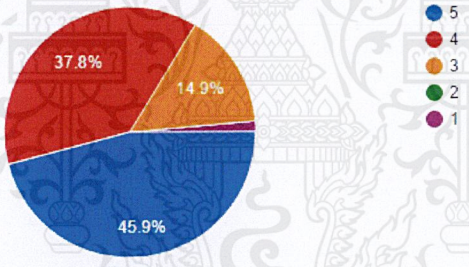


Figure A.2: Users' rating of places

B Filtering system

The proposed filtering system can be divided into 3 parts, i.e. filtering with cost, filtering with time and selection. The following sections shows the code applied in each part of the program.

B.1 Filtering with cost

The proposed function of filtering with cost will get all possible combinations of the attractions as an input. The function will filter out the combinations of the attractions that do not pass the budget requirement. The remaining combinations of the attractions will be sent to the filtering with time function which will be explained in the next section. The following code is applied in order to obtain such result.

```
#create all possible combinations of the attractions
def powerSet(items):
    n = len(items)
    for i in range(2**len(items)):
        combination = []
        for j in range(len(items)):
            if (i >> j) % 2 == 1:
                combination.append(items[j])
        yield combination
#filtering with cost
def filtering_with_cost(self,condition):
        result = []
        temp = []
        for i in powerSet(self.places):
           if( valid(i,condition)) and i != []:
               result.append(i)
        self.places = result
```

```
18 #validate with the budget requirement
19 def valid(x,condition):
20     temp = 0
21     for i in range(0,len(x)):
22         temp += x[i].getCost()
23     if temp <= condition:
24         return True
```

B.2 Filtering with time

After the filtering with time function get the input from filtering with cost function, this function will evaluate the input using time constraints from user. At the end of the procress, the result will be sent to the selection function to select the highest rated combination. The following code is applied in order to obtain such result.

```
1  #filtering with time
2  def filtering_with_time(self,condition):
3          #condition depends on user's requirement
4          night = condition
5          morning = condition
6          evening = condition
7          #both Morning and Evening
8          both_time = 0
9          result = []
10         for i in range (0,len(self.places)):
11             for j in range(0,len(self.places[i])):
12                 if(self.places[i][j].getTime() == ["
                       Morning"] and morning != 0):
13                 morning = morning-1
14                 elif(self.places[i][j].getTime() ==
                       ["Evening"] and evening != 0):
```

```
15                  evening = evening-1
16                  elif(self.places[i][j].getTime() ==
                        ["Morning","Evening"]):
17                  both_time = both_time+1
18                  elif(self.places[i][j].getTime() ==
                        ["Night"] and night != 0):
19                  night = night-1
20              while(both_time != 0):
21                  if(morning==0 and evening==0):
22                      morning = morning-1
23                  elif(morning==0 and evening!=0):
24                      evening = evening-1
25                  elif(morning!=0 and evening ==0):
26                      morning = morning-1
27                  elif(morning!=0):
28                      morning = morning-1
29                  elif(evening!=0):
30                      evening = evening-1
31                  both_time = both_time-1
32              #meet the requirement
33              if(morning ==0 and evening ==0 and night
                    >= 0):
34                  result.append(self.places[i])
35                  morning = condition
36                  evening = condition
37                  night = condition
38              else:
39                  morning = condition
40                  evening = condition
41                  night = condition
```

```
42          self.places = result
```

B.3 Selection

The input of selection is the result from filtering with time function. Selection function will choose the combination the has highest rating. The result is the combination of attractions that has the highest rating. It will be sent to trip planning system. The following code is applied in order to obtain such result.

```
1  #selection function
2  def selection(self):
3          temp = 0
4          temp2 = 0
5          result = []
6          for i in range(0,len(self.places)):
7              for j in range(0,len(self.places[i])):
8                  temp += self.places[i][j].getRating()
9                  #search rating from ratings.txt
10                 with open("ratings.txt") as openfile:
11                     for line in openfile.readlines():
12                         if self.places[i][j].getName
                               () in line:
13                             temp += float(line.strip
                                   ().split(":")[1])*10
14
15             if(temp>temp2 and len(result)==0):
16                 result.append(self.places[i])
17                 temp2 = temp
18             elif(temp>temp2 and len(result)!=0):
19                 result.pop()
20                 result.append(self.places[i])
```

```
21                    temp2 = temp
22                temp = 0
23            self.places = result
```

This appendix shows the program used in the proposed filtering system. The system is divided into 3 parts. There are filtering with cost, filtering with time and selection. Each part has its own purpose but the goal of this system is to select the best combination of the attractions. After the inputs which are the list of recommended places have been evaluated. The obtained results are the combination of recommended places which has the highest rating. The result will be sent to the trip planning system to evaluate further.

C Trip planning system

The proposed trip planning system is used to order the result of the filtering system. The result of trip planning system will be displayed on the web application. The following code is applied in order to obtain such result.

```
#Navigation function
def Navigation(x,origin,time,ans):
    if(len(x[0])==1):
        ans.append(x[0][0])
        return ans
    destination_cord = []
    origin_cord = origin
    temp = 0
    result_ans = ans
    #morning_list,evening_list,night_list,morning
      and evening list
    m_list = []
    e_list = []
    n_list = []
    me_list = []

    for i in range(0,len(x[0])):
        if(x[0][i].getTime() == ["Morning"]):
                    m_list.append(x[0][i])
        elif(x[0][i].getTime() == ["Evening"]):
                    e_list.append(x[0][i])
        elif(x[0][i].getTime() == ["Morning","
           Evening"]):
                    me_list.append(x[0][i])
        elif(x[0][i].getTime() == ["Night"]):
```

```
24                      n_list.append(x[0][i])
25
26      #time's counter is morning
27      if(time=="Morning"):
28          #there is a place that can be visited in the
                morning
29          if(len(m_list)!=0):
30              for i in range(0,len(m_list)):
31                  destination_cord = m_list[i].getCord
                        ()
32                  url = "https://maps.googleapis.com/
                        maps/api/distancematrix/json?
                        units=imperial&origins="+str(
                        origin_cord[0])+","+str(
                        origin_cord[1])+"&destinations="+
                        str(destination_cord[0])+","+str(
                        destination_cord[1])+"&key="+
                        API_KEY
33                  result= simplejson.load(urllib.
                        request.urlopen(url))
34                  distance = result['rows'][0]['
                        elements'][0]['distance']['value
                        ']
35                  #select nearest place
36                  if(temp == 0):
37                      temp = float(distance)
38                      result_ans.append(m_list[i])
39                  elif(distance < temp):
40                      temp = float(distance)
41                      result_ans.pop()
```

```
42                  result_ans.append(m_list[i])
43          target = result_ans[len(result_ans)-1]
44          #print("Target: "+target.getName())
45          x[0].remove(target)
46          next_cord = target.getCord()
47          #increase time's counter
48          time = "Evening"
49          Navigation(x,next_cord,time,result_ans)
50      #there is a place that can be both visited
          in the monring and evening
51      elif(len(me_list)!=0):
52          for i in range(0,len(me_list)):
53              destination_cord = me_list[i].
                  getCord()
54              url = "https://maps.googleapis.com/
                  maps/api/distancematrix/json?
                  units=imperial&origins="+str(
                  origin_cord[0])+","+str(
                  origin_cord[1])+"&destinations="+
                  str(destination_cord[0])+","+str(
                  destination_cord[1])+"&key="+
                  API_KEY
55              result= simplejson.load(urllib.
                  request.urlopen(url))
56              distance = result['rows'][0]['
                  elements'][0]['distance']['value
                  ']
57              #select nearest place
58              if(temp == 0):
59                  temp = float(distance)
```

```
60                      result_ans.append(me_list[i])
61                  elif(distance < temp):
62                      temp = float(distance)
63                      result_ans.pop()
64                      result_ans.append(me_list[i])
65              target = result_ans[len(result_ans)-1]
66              x[0].remove(target)
67              next_cord = target.getCord()
68              #increase time's counter
69              time = "Evening"
70              Navigation(x,next_cord,time,result_ans)
71      #there is a place that can be both visited in
            the evening
72      elif(time == "Evening"):
73          if(len(e_list)!=0):
74              for i in range(0,len(e_list)):
75                  destination_cord = e_list[i].getCord
                        ()
76                  url = "https://maps.googleapis.com/
                        maps/api/distancematrix/json?
                        units=imperial&origins="+str(
                        origin_cord[0])+","+str(
                        origin_cord[1])+"&destinations="+
                        str(destination_cord[0])+","+str(
                        destination_cord[1])+"&key="+
                        API_KEY
77                  result= simplejson.load(urllib.
                        request.urlopen(url))
78                  distance = result['rows'][0]['
                        elements'][0]['distance']['value
```

```
                ']
79              if(temp == 0):
80                  temp = float(distance)
81                  result_ans.append(e_list[i])
82              elif(distance < temp):
83                  temp = float(distance)
84                  result_ans.pop()
85                  result_ans.append(e_list[i])
86          target = result_ans[len(result_ans)-1]
87          #pop out from result
88          x[0].remove(target)
89          #increase time's counter
90          time = "Night"
91          Navigation(x,start_cord,time,result_ans)
92      #there is a place that can be both visited
          in the monring and evening
93      elif(len(me_list)!=0):
94          for i in range(0,len(me_list)):
95              destination_cord = me_list[i].
                  getCord()
96              url = "https://maps.googleapis.com/
                  maps/api/distancematrix/json?
                  units=imperial&origins="+str(
                  origin_cord[0])+","+str(
                  origin_cord[1])+"&destinations="+
                  str(destination_cord[0])+","+str(
                  destination_cord[1])+"&key="+
                  API_KEY
97              result= simplejson.load(urllib.
                  request.urlopen(url))
```

```
98                  distance = result['rows'][0]['
                        elements'][0]['distance']['value
                        ']
99                  #select nearest place
100                 if(temp == 0):
101                     temp = float(distance)
102                     result_ans.append(me_list[i])
103                 elif(distance < temp):
104                     temp = float(distance)
105                     result_ans.pop()
106                     result_ans.append(me_list[i])
107             target = result_ans[len(result_ans)-1]
108             x[0].remove(target)
109             #increase time's counter
110             time = "Night"
111             Navigation(x,start_cord,time,result_ans)
112     #there is a place that can be visited in the
        night
113     elif(time == "Night"):
114         #no more night place
115         if(len(n_list) == 0):
116             #reset time's counter to morning
117             time = "Morning"
118             Navigation(x,start_cord,time,ans)
119         elif(len(n_list)!=0):
120             for i in range(0,len(n_list)):
121                 destination_cord = n_list[i].getCord
                        ()
122                 url = "https://maps.googleapis.com/
                        maps/api/distancematrix/json?
```

```
                units=imperial&origins="+str(
                origin_cord[0])+","+str(
                origin_cord[1])+"&destinations="+
                str(destination_cord[0])+","+str(
                destination_cord[1])+"&key="+
                API_KEY
123             result= simplejson.load(urllib.
                request.urlopen(url))
124             distance = result['rows'][0]['
                elements'][0]['distance']['value
                ']
125             if(temp == 0):
126                 temp = float(distance)
127                 result_ans.append(n_list[i])
128             elif(distance < temp):
129                 temp = float(distance)
130                 result_ans.pop()
131                 result_ans.append(n_list[i])
132         target = result_ans[len(result_ans)-1]
133         x[0].remove(target)
134         #reset time's counter to morning
135         time = "Morning"
136         Navigation(x,start_cord,time,result_ans)
```

This appendix shows the program used in the proposed trip planning system. The goal of this system is to correctly sort the combinations of the attractions. The result of this system will be displayed on the web application.