MY CARBON FOOTPRINT APPLICATION

Natnicha Thonket

Tanasak Ngerniam

Pattama Kaewsopa

Bachelor of Engineering Program in Software Engineering

International College

King Mongkut's Institute of Technology Ladkrabang

2015

Thesis – Academic Year 2015

B.Eng. in Software Engineering

International College, King Mongkut's Institute of Technology Ladkrabang

Title MY CARBON FOOTPRINT APPLICATION

Authors

1. Ms. Natnicha Thonket Student ID 55090018
2. Mr. Tanasak Ngerniam Student ID 55090023
3. Ms. Pattama Kaewsopa Student ID 55090030

Approved for submission

_____ Advisor

Dr. Ronnachai Tiyarattanachai

_____ Co-Advisor

Dr. Isara Anantavrasilp

Date 24/5/16

My Carbon Footprint Application

Ms. Natnicha Thonket Student ID 55090018

Mr. Tanasak Ngerniam Student ID 55090023

Ms. Pattama Kaewsopa Student ID 55090030

Dr. Ronnachai Tiyarattanachai Advisor

Dr. Isara Anantavrasilp Co-Advisor

Academic Year 2015

ABSTRACT

The major purpose of the project is to raise awareness of climate change problem on this global by tracking their carbon footprint from people's activities daily and convince people to reduce carbon dioxide emission. My Carbon application is developed as an application based on hybrid mobile technology which uses web technology with a plug-in to access mobile device capabilities and it can run on both operation systems which are iOS and Android. The application primarily tracks two main sources of CO_2 emission which are from transportation and household electricity usage. The amount of CO_2 emission will be calculated from the activities and presented the data compared with the others in the form of graphs. Also, the application also provides leaderboard for ranking the number of CO_2 emission between friends and other users. In addition, the application allows users to interact and share the total emission on a social network such as Facebook. This application cooperates with some business in order to reward some prize to users. Besides, to broaden the issue, it provides forum feature for users to share an idea and discuss a relevant topic together. With these features, it will be very useful for people to be aware of this global impact with an easy-to-use interface. This project could help resolve the global warming problem.

Acknowledgements

First and foremost, we greatly appreciate the kindness to our project advisor, Dr. Ronnachai Tiyarattanachai. Along with the positive learning environment and the lessons he provided us, it has been very insightful and fun. We are grateful for his encouragement, not only the project consultation but also many morals and recommendation in real life. Many thanks for his assistance and guidance in advancing our presentations and this excellent thesis. We have learned so much. Thank you for being patient and helping us improve.

Accompanying, we proudly say thank you to our co-advisor, Dr. Isara Anantavrasilp for spending time and taking the trouble to help us on this project. He inspires and encourages us with his valuable experiences and wisdom to complete this project powerfully. We wholeheartedly appreciate everything that both advisor and co-advisor have done for us.

Ultimately, we admiringly acknowledge our parents all professors from the faculty of International College, KMITL who always encourage and strengthen us to study and do all the projects since we were freshmen. Without their supports and worthy recommendation, we would not have been succeeded so far.

Table of Contents

Table of Figures

Chapter 1

Introduction

Carbon dioxide is one of the key greenhouse gases. It poses the greatest impact on greenhouse gas emission as presented as Figure 1.1. Carbon dioxide enters the atmosphere through the use of fossil fuels, terse and wood products. It is removed from the atmosphere when it is absorbed by plants as part of the biological carbon cycle [1].

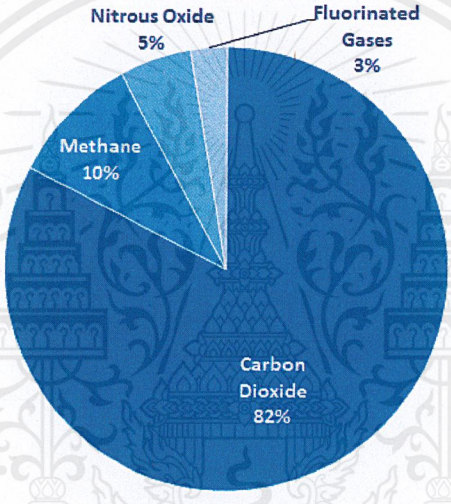


Figure 1.1 Greenhouse Gases Emission

Carbon Dioxide Emission

Global warming is primarily a problem of too much carbon dioxide (CO_2) in the atmosphere and this gas survives for a long time — up to many centuries, so its heat-trapping effects are compounded over time [2]. Carbon dioxide is emitted from many different sources which are electricity, transportation, industry, residential and commercial activities. The two greatest sources that emit lots of CO_2 are electricity and transportation which are 37% and 31%, respectively [3]. Thereby, this application focuses on these sources in order to track people's carbon footprint in their daily lives.

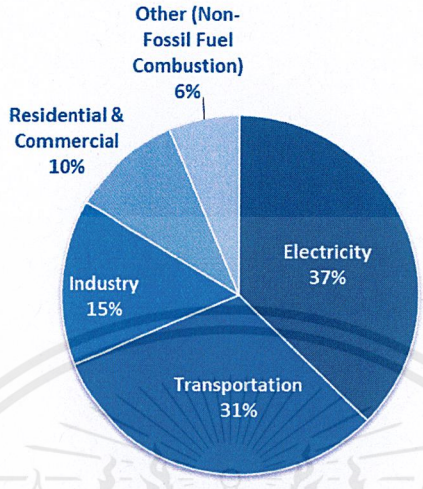


Figure 1.2 Carbon Dioxide Emission by Sources

Effects of Carbon Dioxide

It is well known that carbon dioxide is one of greenhouse gases, which cause global warming. This phenomenon also causes several subsequent problems such as sea level rise, impacts on agriculture, reduction of the ozone layer and increase extreme weather [4].

1.1 Motivation

Global warming is one of the most important environmental problems. Global warming is linked to the amount of greenhouse gases being emitted into the atmosphere. There are several main greenhouse gases which are carbon dioxide, methane, nitrous oxide and fluorinated gases. These gases are emitted from many activities likewise burning fossil fuels, agricultural and industrial activities, but a greenhouse gas which has the most significant impact is carbon dioxide (CO_2) [1]. Each of these gases remains in the atmosphere for a long time, so its heat trapping effects have accumulated over time. This problem poses many negative impacts to the world and its environment such as world's temperature which has been gradually increasing for years. A warming world also has the potential to change rainfall, severe storms, reduce lake ice cover and increase sea levels [2].

This global impact is mostly caused by human activities such as using motor vehicles, turning on electronic devices and deforestation. The main human activity that emits CO_2 is the combustion of fossil fuels like coal, natural gas and oil for energy and transportation [3]. Electricity is significantly used to power home and industry, and transportation is also used in people's daily lives. Therefore, the amount of CO_2 emission could be controlled and reduced little by little from human individual activities. Every small difference that has changed would add up to a big difference, so even the minor things are worthwhile [4].

1.2 Objective

This project aims to raise awareness of people about this global impact by tracking their carbon footprint and showing them the information. This project develops a mobile application on both operation systems which are iOS and Android, so that people can use the application in their daily lives. Due to transportation and electricity are main sources of CO_2 emission [3], the application will track these two activities of people. By convincing people to reduce their CO_2 emission through the application, this project could help resolve the global warming problem.

1.3 Scope of Work

This application is based on hybrid mobile technology that use web technology with plug-in to access mobile device capabilities. By using hybrid mobile technology, the application is developed using AngularJS, Ionic, CSS, HTML and Python programming language. To access to mobile capabilities, it uses NgCordova as a plug-in. Also, Google SDK library is included for map services.

The application provides users to track their CO_2 emission in transportation and electricity used. The amount of CO_2 emission that is emitted will be ranked and also compared with other users. In addition, it supports users to interact with other users and their friends in social network such as Facebook. This application also entails business partnership by cooperating with some businesses in order to reward some prize to users for their achievements.

1.4 Procedure

My Carbon application will be developed following these phases as below:

Phase 1: Research, Identify and define system features

- List all features in application
- Draw complete wireframe including mockup and workflow
- Identify requirement and use cases

Phase 2: Design system components

- Design interface
- Design system architecture
- Design database

Phase 3: Implement minimum viable product and testing

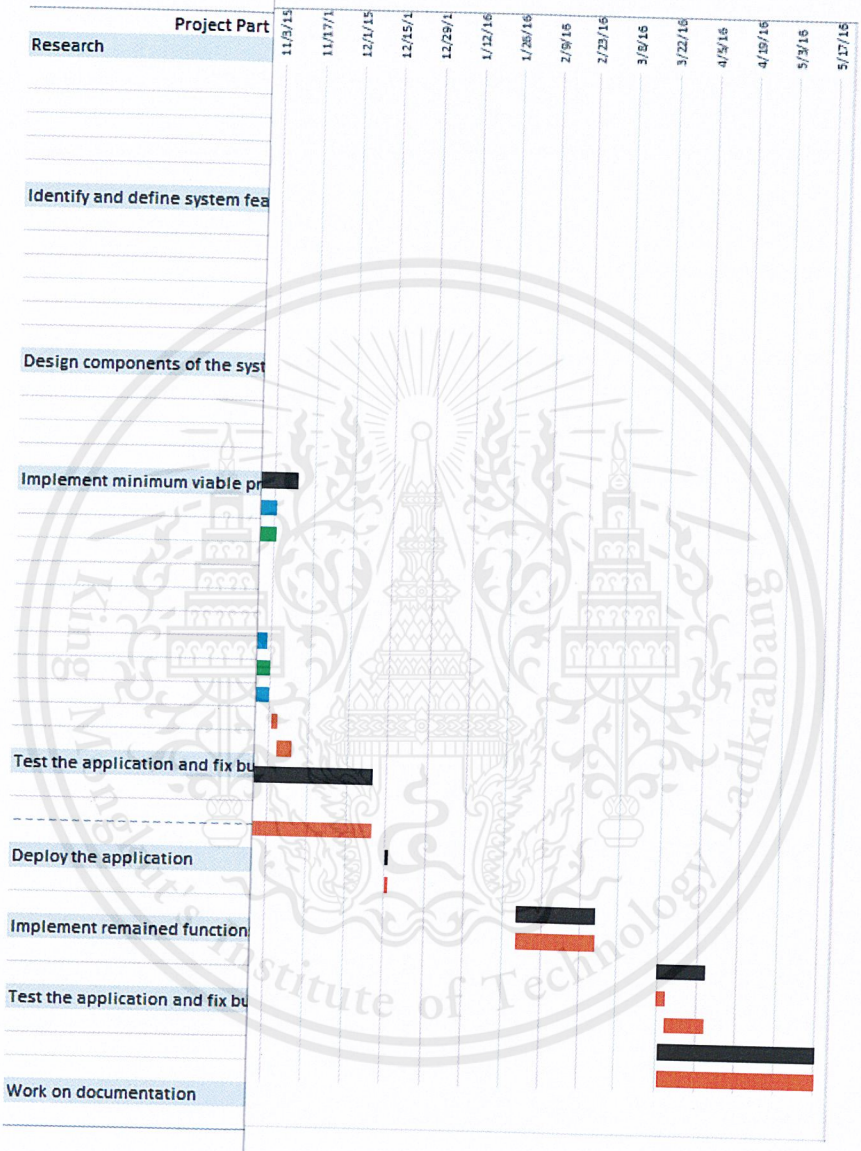
- Implement account system
- Implement database system
- Implement CO_2 tracking system
- Implement leaderboard
- Do integration testing and system testing

Phase 4: First deployment and fixed bug after deploy

Phase 5: Implement remained function and testing

- Implement forum

Phase 6: Deploy complete application and fixed bug after deploy



1.5 Structure of Thesis

The rest of thesis is organized as follows

Chapter 2 describes features in carbon footprint application and existing mobile application.

Chapter 3 describes technical background which is used in this application.

Chapter 4 describes software architecture in details which are requirements, use case, interface, system architecture and database design.

Chapter 5 describes what are happened after the application is published.

Chapter 6 summaries the result and suggestion.

Chapter 2

Literature Review

This chapter shows existing related works in details which explain about features and limitations in Section 2.1. After that, Section 2.2 presents review of existing Carbon footprint mobile application which describes integrated features. Lastly, there is a comparison table of features between My Carbon application and other related applications.

2.1 Related Work

2.1.1 Ecolife

Ecolife is an application on Android and iOS which focuses on commuting. It provides many kinds of commuting such as cycling, driving, and taking public transport. The application calculates the number of saved carbon-dioxide and how many trees are saved. It also calculates the calories burning for each commuting. The more users save carbon-dioxide, the more they gain Ecopoints. Another way to get Ecopoints is to succeed an achievement. Users can redeem their Ecopoints for discounts at some restaurants or shops which are their partners. Besides, the application has a ranking system that shows users' rank weekly.

This application provides a special menu likewise member card as Figure 2.1. Users will get the member card after they register to the application. This member card in Figure 2.2 can be identified users' status and their points will be kept in this card as well.

Figure 2.1 Ecolife home page

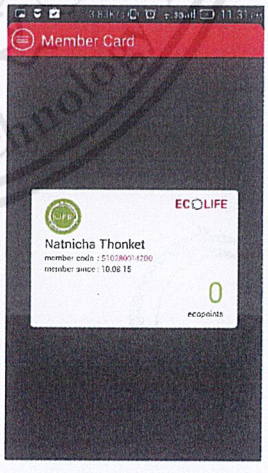


Figure 2.2 Ecolife member card

Figure 2.3 Ecolife Badge

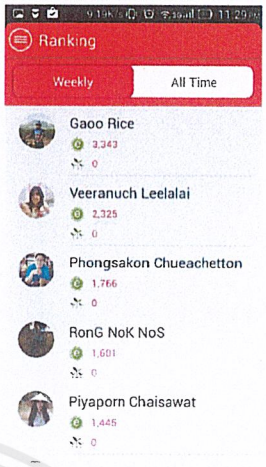


Figure 2.4 Ecolife Ranking

A badge feature of this application is an additional way for users to gain more points by completing the conditions of the badged. Their points will be used to classify their rank weekly. A special feature is that this application has a lot of partnership as Ecoshop, Silpakorn Green Bike, MFU Grows Green and Green Wave Green bike.

2.1.2 Changer

Changer is an application on mobile devices which calculates how much CO_2 is saved from people's transportation. The application provides several ways of transportation which are walking, cycling, flying, and driving. It tracks the transportation using mobile's GPS to get a total distance and speed to calculate the number of CO_2 emission and estimate ReCoins they should get from each activity.

Figure 2.5 Changer Home Page

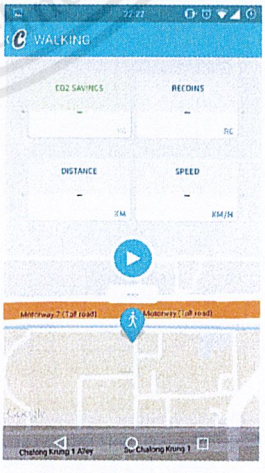


Figure 2.6 Changer Transport Section

As presented on Figure 2.7, there are missions for users to be completed. If a user achieves a mission, the award will be unlocked and the user will get some Recoins from the mission. Otherwise, users can purchase the amount of CO_2 using their Recoins as Figure 2.8 shown.

Figure 2.7 Changer Mission

Figure 2.8 Changer Store

The total of their CO_2 balance (kg) will be ranked in three leaderboards which are between users in Figure 2.9, cities in Figure 2.10 and countries.

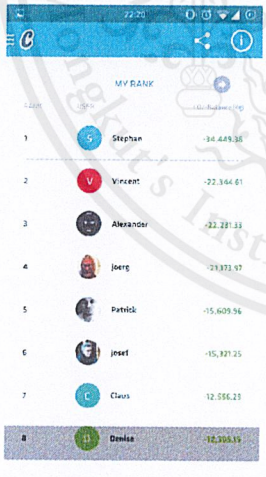


Figure 2.9 Changer User Ranking

Figure 2.10 Changer City Ranking

2.1.3 Lotus Greens

Lotus Greens application separates data resources into five sections which mainly focus on daily activities such as energy consumption, commuting in a city, and other lifestyle shown as Figure 2.11, Figure 2.12 and Figure 2.13. The application will analyze and calculate the total of CO_2 emission from the input data and present the finalized data as pie chart as in Figure 2.14 below. Furthermore, the tonnes of CO_2 emission can be shared to social network as Facebook.

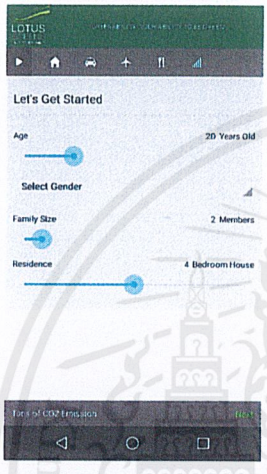


Figure 2.11 Lotus Greens Start Information

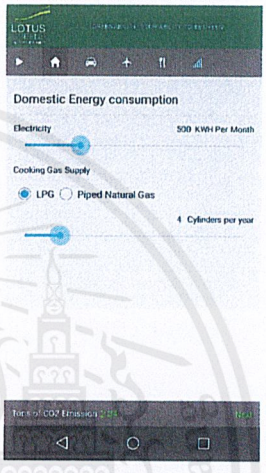


Figure 2.12 Lotus Greens Energy Consumptions

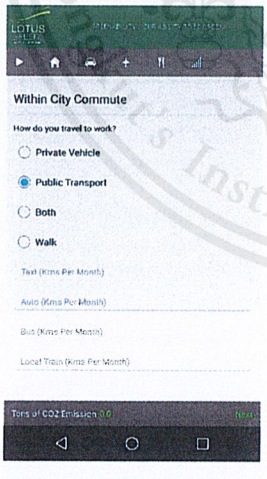


Figure 2.13 Lotus Greens City Commute

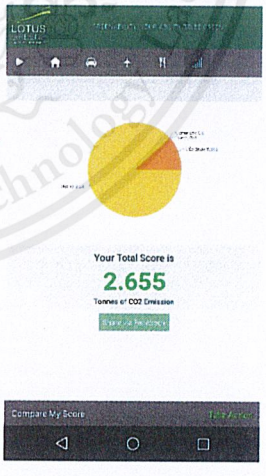


Figure 2.14 Lotus Greens CO_2 Emission

2.2 Integrated Carbon Footprint Features

Features of My Carbon footprint mobile application can be divided into eight main sections which are Account, CO_2 system, Transportation, Electricity, Leaderboard, Reward, Friends and Forum section.

2.2.1 Account Section

This section is about the management system for all accounts which are used to access the application.

- **Sign up a new account**

  User can create a new account directly with the application using an E-mail address.

- **Sign in to the application**

  The application can be accessed by using either the directly registered account with the application or using a Facebook account.

- **Retrieve password**

  The application will send a new generated password to the requested E-mail address.

2.2.2 Carbon Dioxide System Section

This section mainly focuses on numbers of CO_2 emission and reduction. It also shows analyzed data in graphical format, which is easier to understand by users.

- **Calculate CO_2 emission**

  The system will calculate the amount of CO_2 emission from both household electricity usage each month and a data from each traveling.

- **Define comparison graphs of CO_2 emission**

  After analyzing, the system will show comparison graph of CO_2 emission between a user and the average number of all all users. There are three types of graph which are electricity usage, transportation and all consumption of both systems.

2.2.3 Transportation Section

This section is one of CO_2 tracking systems of this application. It provides many different kinds of transportation — i.e. walking, cycling and driving. The system tracks routes and distances, then uses these data in CO_2 system section.

- **Using GPS**

  The system uses GPS to track a journey. The route will be shown on a map as a path from the started location to current location.

- **Calculate distance**

  The system gets a total distances from the started location to final location using GPS.

2.2.4 Electricity Section

This section is another system of the tracking system which calculate the amount of CO_2 emission from household electricity usages. The application has to be provided a amount of electricity usage for each address in order to calculate the emission in each month. Every data can be edited with in.

2.2.5 Leaderboard Section

Leaderboard section provides ranks of users in the application sorted by the lowest amount of CO_2 emission. There are three modes of ranking system which are only for transportation, electricity and lastly all consumption. Each modes are divided to two ranking which are in local ranking or global ranking.

- **Local ranking**

  This ranking compares the emission with only accounts which are friends together.
- **Global ranking**

  The global ranking compares the emission between the user and everyone.
- **Share the rank**

  Users can share their ranks to social networks.

2.2.6 Reward Section

The application provides daily missions and achievements with some conditions. Users will get points when they completed a mission or achieve any achievement.

- **Daily mission**

  The application will provide a randomly mission everyday.
- **Achievements**

  Achievements will be unlocked when users complete all the conditions, then they will get the points.
- **Redeem awards**

  The points can be used to redeem some rewards with our partnerships.

2.2.7 Friends Section

This section is focused on friend circle of an account. The account can be searched and added to be friends with other people to compete together on the leaderboard.

- **Search friends**

  An account can be looked up by its username.

- **Add friends**

  This feature allows users to add other people in an application as friends.

2.2.8 Forum Section

This section provides forum for all users to communicate and discuss together based on the related topics. Users can create their own forums and also they can comment or reply to other forum.

- **Create a forum**

  This feature is to create a new topic to discuss with other people.

- **Edit forum**

  Any forum can be edited by the account which created the forum.

- **Delete forum**

  Any forum can be deleted only by the owner of the forum.

- **Comment**

  Users can comment to any forum and topics.

- **Reply**

  Each comments can be replied from other people.

- **Favorite forums**

  This feature allows users to mark any forum as their favorite, so that they can re-read the forum easier.

2.3 Comparison of Applications' Features

Features	CO_2 Footprint Applications			
	My Carbon	Ecolife	Changer	Lotus Greens
Sign up with application	x	x	x	
Sign in with social network	x	x	x	x
Retrieve password	x	x	x	
Calculate CO_2 emission	x			x
Calculate CO_2 saved		x	x	
Compare CO_2 emission with objects	x	x		x
Share result to social network	x	x	x	x
Define comparison graph of CO_2 emission	x			x
Using GPS	x	x	x	
Using map	x		x	
Calculate distance	x	x	x	
Calculate velocity			x	
Calculate calories burn		x		
Calculate CO_2 from house bill	x			x
Friend circle ranking	x			
All users ranking (Global)	x	x	x	
City ranking			x	
Country ranking			x	
Achieve badges and get points	x	x	x	
Achieve daily mission to earn points	x		x	

Table 2.1 Comparison among My Carbon and Related Application

Features	CO_2 Footprint Applications			
	My Carbon	Ecolife	Changer	Lotus Greens
Redeem score with some prize	x	x	x	
Share award to social network	x			x
Add friends	x			
Search users	x			
Create forum	x			
Edit forum	x			
Delete forum	x			
Comment in forum	x			
Reply in forum	x			

Table 2.2 Comparison among My Carbon and Related Application (Continued)

Chapter 3

Background Knowledge

3.1 Calculating Carbon Dioxide Emission

3.1.1 Transportation

Transportation is part of people's daily life. There are many modes of transportation such as bicycle, car, train or plane. The combustion of fuel such as gasoline and diesel in transportation is the numerous source of CO_2 emission. For a car, the amount of CO_2 emission for each trip can be calculated in numerical data using CO_2 emission rate from each type of transportation multiply by the distances as shown in the formula below.

CO_2 emission (g-km) = CO_2 emission rate (g) x Distances (km)

Each car's model provides different amount of CO_2 emission from manufacturing. CO_2 emission data in this application were retrieved from publicly available databases. In similar way, the CO_2 emission rates from other kinds of transportation likewise motorcycle, electric train, and bus are 94, 69 and 65 gram per kilometers [6].

3.1.2 Electricity

Electricity is a significant source and used to power homes, business and industries. The type of fossil fuel used to generate electricity will emit different amount of CO_2, burning coal will produce more CO_2 than oil or natural gas.

Based on data from Energy Policy and Planning Office [5] reports on CO_2 emission from energy sector in first six months (January to June) in 2015. The report includes percentage on power generation which is 63 percent from natural gas, 36 percent from coal and lignite and 2 percents from oil. Therefore, number on average of CO_2 emission per kilowatt hour is 0.526 kg CO_2/kWh. CO_2 emission from electricity used can be calculated from the formula below.

CO_2 Emission (g) = Electricity used (kWh) x 526 ($^{g\,CO_2}/_{kWh}$)

3.2 Mobile Application

My Carbon is developed as a mobile application. Mobile application can be run on various platforms such as the popular platforms, iOS and Android. Each platform has its characteristics depending on each company. Difference of construction makes them numerous implementations. Therefore, hybrid mobile application can solved problems about time implementation on cross-platforms mobile application.

Hybrid mobile application uses web technologies for creating interface instead of native development languages. Normally, native mobile application may use Native UI as user interface classes but hybrid version had changed to WebView representing web technologies developed UI.

3.2.1 Ionic

Ionic is a software development kit which can be used to develop mobile application by using web technologies. This is used to design and organize the application to be more powerful and beautiful in user interface view. Ionic also provides a Creator to create a real Ionic application with simple drag-and-drop prototyping tool both beginner and advanced developers. Furthermore, Ionic gives a guide which covers all the basics of creating an application or website, e.g. CSS components, JavaScript, Ionicons and HTML5.

3.2.2 NgCordova

NgCordova is a tool for developing hybrid mobile application and access the full native functionality of the device using. NgCordova is wrapped up from Cordova and a numerous collection of AngularJS which make it easier to get a mobile application done.

- **AngularJS**

  AngularJS is a framework that has models similar with Javascript objects. It acts as a HTML vocabulary extension for dynamic views of HTML. Hence, Angular code is easy to test, maintain and reuse based on these following concepts.

  - 2-ways data blinding provides model that contains values. Implementation of application will use only that model for blinding inside data.
  - Model-View-Whatever design pattern which allows variables data sharing between layers.
  - Custom-directive for reusable components.

Furthermore, This application use alternative source called **Angular Chart.** Angular Chart is a reactive, responsive and beautiful chart containing a set of native AngularJS. This is a better way to describe and show the application's data easily in many different ways even in comparison and summary. It nevertheless provides several templates of charts likewise pie chart, line chart and bar chart.

- **Cordova**

  Cordova is a free open-source framework, which allows developers to use web technologies including HTML, CSS and Javascript. Its objective is a multiple platforms application with only one code base.

- **Cordova plugin**

  Hybrid mobile application has advantage on web technologies but somehow web technologies cannot replace all functions. Cordova provides plugin as accessor to mobile capabilities, frameworks also different processes.

  Google maps: Cordova-plugin-googlemaps is providing developers to use google maps mobile sdk instead of google maps web sdk.

  InAppBrowser: Cordova-plugin-inappbrowser is providing developers to run browser in mobile application.

3.3 Server

3.3.1 Django

Django is a high-level Python web framework which intends to serve as a support or guide for the building of program that expands the structure more usefully. The Django ORM handle creation of a database and some advance querying and also supports multiple databases such as MySQL, PostgreSQL, Oracle and SQLite. Its template creates the HTML, adds an error message and cleans the data. Besides it generates and updates the database from database model that has been created. Python, which is a powerful language and have a lot of library and tutorials for self-study, is used to develop in Django. This application uses two additional things to implement called REST framework and Django-allauth.

- **REST framework**

  This is a powerful and flexible toolkit for building Web APIs.

- **Django-allauth**

  A set of Django applications which is addressing authentication, registration, account management as well as the third party likewise social account authentication.

3.3.2 Database

Database is used to store all data of the application included the number of CO_2 emission, user accounts, and ranking system. The number of CO_2 emission is essential for analyzing to visualize the tendency of the data. This application uses MySQL and PhpMyAdmin to manage the database on server.

- **MySQL**

  MySQL is an open source relational database management system and based on the structure quest language, which is used to add, remove and modify information in the database. It runs on virtually all platform including Linux, UNIX and windows. Also it can be used in wide range of applications.

- **PhpMyAdmin**

  An application for MySQL database management via a web browser written in PHP intended to handle the administration of MySQL. It can create, alter, drop, import and export MySQL database tables. PhpMyAdmin runs MySQL queries, optimize, repair & check tables and change & execute other database management commands.

Chapter 4
Software Architecture

4.1 Methodology

This project uses a scrum method for developing which is provided user stories to clarify the requirements. Each user story has a number which tells a level of its complexity and developing time. The tasks will be organized to be done from the most necessary to the less important every week in a meeting in order to find and figure out a problem that occurred during the developing. The project could be seen the progress every week, so the requirements would be changed to a better way to reach the best solution for the project's proposal.

4.2 System Requirements

4.2.1 Functional Requirements

The system can verify for account validation.
The application remembers user's account on the device.
Password can be retrieved by sending through email.
The application will calculate CO_2 emission from number of household electricity usage and traveling consumption.
The application will summarise daily CO_2 emission.
The system will keep track for daily CO_2 consumption and show as graphical interface.

4.2.2 Non-Functional Requirements

The system must encrypt password.
The application will be developed both iOS and Android.
Back-end system is based on Django and Python.
The system uses mySQL database.
The system manages database using SQL language.

4.3 User Requirements

4.3.1 Functional Requirements

Users can sign up with the application.
Users can sign in with either their local accounts or Facebook accounts.
User can sign out of the application.
Users can retrieve their passwords.
Users can track CO_2 emission from their traveling.
Users can input a quantity of energy usage from house bill.
Users can select a way to travel, i.e. walking, cycling and driving.
Users can view a map and route during traveling.
Users can view CO_2 consumption history.
Users can unlock achievements.
Users can achieve daily mission.
Users can view leaderboard which presents the rank of users.
Users can share their rank to social network.
Users can search friends by their username.
Users can add others to be friends.
Users get points from their daily activities, unlocked achievements and completed missions.
Users can redeem their points with rewards.
Users and administration can read any topic in Forum.
Users and administration can create a new topic for discussion.
Users and administration can edit or delete their posts.
Users and administration can comment to any topic.
Users can mark or unmark any topic as favorite.
Administrator can delete any post in forum.
Administrator can view analyzed data of CO_2 consumption.
Administrator can view data in database.

4.3.2 Non-Functional Requirements

Application has to be logged-in first.

Application supports only one user at a time.

4.4 Use Case Diagram

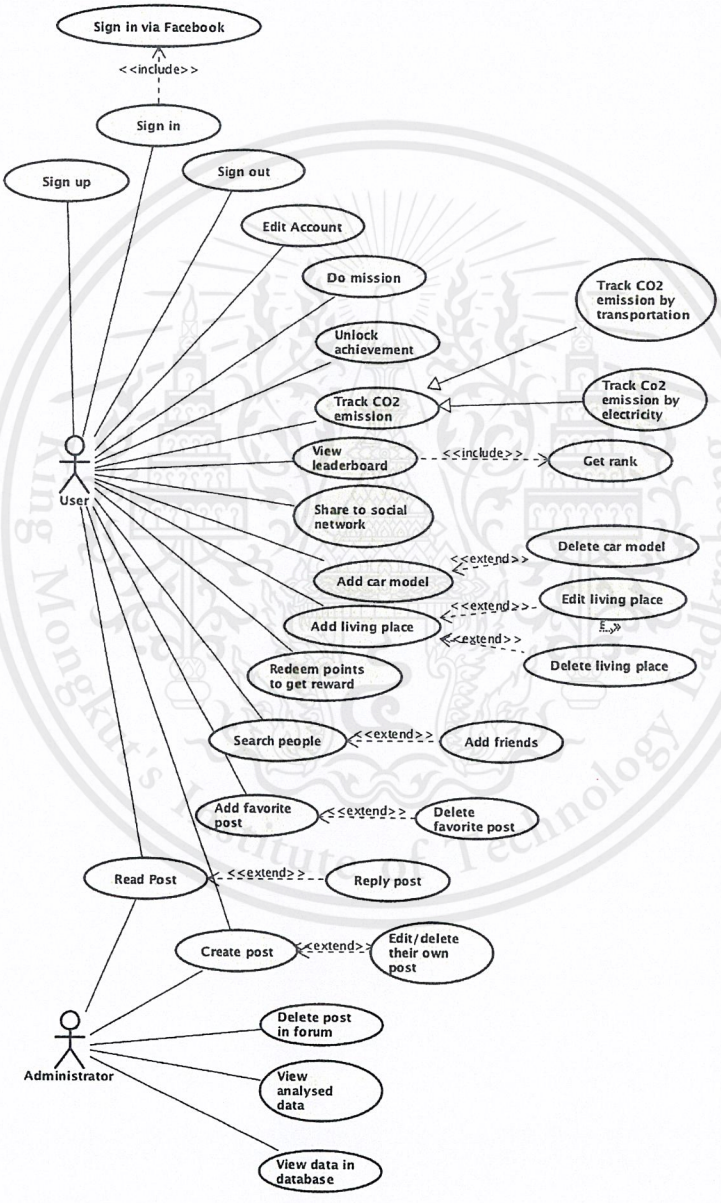


Figure 4.1 Use Case Diagram

4.5 Use Case Description

In this section describes actions between an actor and the system of this application. Each use case presents a primary actor, pre-condition and post condition. It also shows flow of the events for each action for more clearly understanding.

Sign Up		
Primary actor	User	
Pre-condition	-	
Post-condition	-	
Flow of events	**Actor actions**	**System actions**
	1) User inputs username, email and password.	
	2) User clicks sign up button.	
		3) System checks for duplicating username.
		4) System creates an account.
		5) System lunches home page as signed-in account.
Alternative Flow	When username is duplicated with another.	

Table 4.1 Sign Up Case Description

Sign In		
Primary actor	User	
Pre-condition	-	
Post-condition	-	
Flow of events	**Actor actions**	**System actions**
	1) User inputs username and password.	
		2) System checks for validation.
		3) System lunches home page.
Alternative Flow	When username or password is mismatch.	

Table 4.2 Sign In Case Description

Edit Account		
Primary actor	User	
Pre-condition	Sign in	
Post-condition	-	
Flow of events	**Actor actions**	**System actions**
		1) System authenticates user before changing.
	2) User puts password.	
		3) System checks for password validation.
	4) User edits account details.	
	5) User submits the data.	
		6) System updates the data.
Alternative Flow	When password is incorrect, system will reject the request.	

Table 4.3 Edit Account Case Description

Sign In via Facebook		
Primary actor	User	
Pre-condition	-	
Post-condition	-	
Flow of events	**Actor actions**	**System actions**
	1) User clicks at Facebook login button.	
		2) System generates Facebook login page.
	3) User inputs username and password.	
		4) System lunches home page.
Alternative Flow	When username or password is mismatch.	

Table 4.4 Sign In via Facebook Case Description

Track CO_2 Emission by Electricity		
Primary actor	User	
Pre-condition	Sign in, Add living place	
Post-condition	-	
Flow of events	**Actor actions**	**System actions**
	1) User chooses an address to record data.	
		2) System shows summarize graph of the chosen address and a record form.
	3) User selects month and input the data.	
	4) User submits data.	
		5) System updates the data and graph.
Alternative Flow	When resident number is less than zero or duplicated of name.	

Table 4.5 Track CO_2 Emission by Electricity Case Description

Track CO_2 Emission by Transportation		
Primary actor	User	
Pre-condition	Sign in, Add car model	
Post-condition	-	
Flow of events	**Actor actions**	**System actions**
	1) User chooses a vehicle to travel.	
		2) System gets its rate of CO_2 emission.
		3) System lunched a map on travel page.
	4) User clicks a start button to travel.	
		4) System starts tracking the location.
	5) User stops the travel.	
		6) System freezes for a while and goes back to home page.
Alternative Flow	When resident number is less than zero or duplicated of name.	

Table 4.6 Track CO_2 Emission by Transportation Case Description

View Leaderboard		
Primary actor	User	
Pre-condition	Sign in	
Post-condition	-	
Flow of events	**Actor actions**	**System actions**
	1) User clicks at leaderboard tab.	
		2) System goes to leaderboard initiated with friend range and travel mode.
Alternative Flow	When resident number is less than zero or duplicated of name.	

Table 4.7 View Leaderboard Case Description

Do Mission		
Primary actor	User	
Pre-condition	Sign in	
Post-condition	-	
Flow of events	**Actor actions**	**System actions**
	1) User completed the daily mission.	
		2) System notices the completed mission.
		3) System adds reward to the account.
Alternative Flow	-	

Table 4.8 Do Mission Case Description

Unlock Achievements		
Primary actor	User	
Pre-condition	Sign in	
Post-condition	-	
Flow of events	**Actor actions**	**System actions**
	1) User completed an achievement.	
		2) System notices the completed mission.
		3) System adds reward to the account.
Alternative Flow	-	

Table 4.9 Unlock Achievements Case Description

Add Car Model		
Primary actor	User	
Pre-condition	Sign in	
Post-condition	-	
Flow of events	**Actor actions**	**System actions**
	1) User chooses brand and model. 2) User submits the data.	
		3) System updates the data.
Alternative Flow	When there are no desired models, user can choose from engine size.	

Table 4.10 Add Car Model Case Description

Delete Car Model		
Primary actor	User	
Pre-condition	Sign in	
Post-condition	-	
Flow of events	**Actor actions**	**System actions**
	1) User clicks at delete button at each vehicle.	
		2) System updates the data.
Alternative Flow	-	

Table 4.11 Delete Car Model Case Description

Add Living Place		
Primary actor	User	
Pre-condition	Sign in	
Post-condition	-	
Flow of events	**Actor actions**	**System actions**
	1) User puts place name, type and number of resident.	
	2) User submits the data.	
		3) System updates the data.
Alternative Flow	When resident number is less than zero or duplicated of name.	

Table 4.12 Add Living Place Case Description

Edit Living Place		
Primary actor	User	
Pre-condition	Sign in	
Post-condition	-	
Flow of events	**Actor actions**	**System actions**
	1) User clicks at the address.	
	1) User edits address data.	
	2) User submits the data.	
		3) System updates the data.
Alternative Flow	When resident number is less than zero or duplicated of name.	

Table 4.13 Edit Living Place Case Description

Delete Living Place		
Primary actor	User	
Pre-condition	Sign in	
Post-condition	-	
Flow of events	**Actor actions**	**System actions**
	1) User clicks at the address.	
	2) User clicks at delete button.	
		3) System updates the data.
Alternative Flow	-	

Table 4.14 Delete Living Place Case Description

Redeem Points		
Primary actor	User	
Pre-condition	Sign in	
Post-condition	-	
Flow of events	**Actor actions**	**System actions**
	1) User goes to achievement page.	
	2) User clicks at redeem reward button.	
		3) System shows a list of rewards.
	4) User chooses the reward.	
		5) System pops up succeeded message and update data.
Alternative Flow	When user already redeemed the reward past 7 days or not enough points to redeem.	

Table 4.15 Redeem Points Case Description

Search People		
Primary actor	User	
Pre-condition	Sign in	
Post-condition	-	
Flow of events	**Actor actions**	**System actions**
	1) User inputs a username in search box.	
	2) User clicks at search button.	
		3) System shows found account.
Alternative Flow	There is no searching account.	

Table 4.16 Search People Case Description

Add Friend		
Primary actor	User	
Pre-condition	Sign in, Search people	
Post-condition	-	
Flow of events	**Actor actions**	**System actions**
	1) User clicks at add friend button.	
		2) System updates data on database.
		3) System shows the account as friend.
Alternative Flow	-	

Table 4.17 Add Friend Case Description

Share to Social Network		
Primary actor	User	
Pre-condition	Sign in	
Post-condition	-	
Flow of events	**Actor actions**	**System actions**
	1) User clicks at share button.	
		2) System shows a list of applications that can be share to.
	3) User chooses the application.	
		4) System links to the application.
	5) User submits sharing message.	
		5) System shares and returns to My Carbon application.
Alternative Flow	When resident number is less than zero or duplicated of name.	

Table 4.18 Share to Social Network Case Description

Read Post		
Primary actor	User, Administrator	
Pre-condition	Sign in	
Post-condition	-	
Flow of events	**Actor actions**	**System actions**
	1) User clicks at desired topic.	
		2) System shows the post information.
Alternative Flow	-	

Table 4.19 Read Post Case Description

Reply Post		
Primary actor	User, Administrator	
Pre-condition	Sign in, Read post	
Post-condition	-	
Flow of events	**Actor actions**	**System actions**
	1) User inputs message in comment box.	
	2) User clicks button to send message.	
		3) System refreshes the post.
Alternative Flow	When comment box is empty.	

Table 4.20 Reply Post Case Description

Create Post		
Primary actor	User, Administrator	
Pre-condition	Sign in	
Post-condition	-	
Flow of events	**Actor actions**	**System actions**
	1) User clicks at create post button.	
		2) System goes to create post page.
	3) User inputs topic and body message then submit.	
		4) System shows created post.
Alternative Flow	When either topic title box or body message box is empty.	

Table 4.21 Create Post Case Description

Edit Their Own Post		
Primary actor	User, Administrator	
Pre-condition	Sign in	
Post-condition	-	
Flow of events	**Actor actions**	**System actions**
	1) User goes to my post page.	
	2) User clicks at edit button.	
		3) System lunches the edit post page.
	4) User inputs data to edit then submit.	
		5) System updates data and refresh page.
Alternative Flow	-	

Table 4.22 Edit Their Own Post Case Description

Delete Their Own Post		
Primary actor	User, Administrator	
Pre-condition	Sign in	
Post-condition	-	
Flow of events	**Actor actions**	**System actions**
	1) User goes to my post page.	
	2) User clicks at delete button at the post.	
		3) System asks to confirm.
	4) User confirms to delete post.	
		5) System updates data and refresh page.
Alternative Flow	-	

Table 4.23 Delete Their Own Post Case Description

Delete Post		
Primary actor	Administrator	
Pre-condition	Sign in	
Post-condition	-	
Flow of events	**Actor actions**	**System actions**
	1) Administrator clicks at delete button at the post.	
		3) System asks administrator to confirm.
	4) Administrator confirms to delete post.	
		5) System updates data and refresh to my post page.
Alternative Flow	-	

Table 4.24 Delete Post Case Description

View data in database		
Primary actor	Admin	
Pre-condition	Sign in as an admin	
Post-condition	-	
Flow of events	**Actor actions**	**System actions**
	1) Admin puts username and password.	
		2) System checks for username and password validation.
	3) Admin chooses database table to view.	
		4) System returns data to admin.
Alternative Flow	-	

Table 4.25 View data in database Case Description

View analysed data		
Primary actor	Admin	
Pre-condition	Sign in as an admin	
Post-condition	-	
Flow of events	**Actor actions**	**System actions**
	1) Admin puts username and password.	
		2) System checks for username and password validation.
	3) Admin choose type of analyze data.	
		4) System analyze data.
		5) System return analyze data to admin.
Alternative Flow	-	

Table 4.26 View Analyzed data Case Description

4.6 Program Design

4.6.1 Overall Design

The application has been designed into three layers which are MVC or Model-View-Controller architecture pattern. In MVC pattern, users see program through view layer but interact with controller layer. However, controller will manipulate information to model then update the view layer in ordered. Figure 4.2 shows objects included in the model layer.

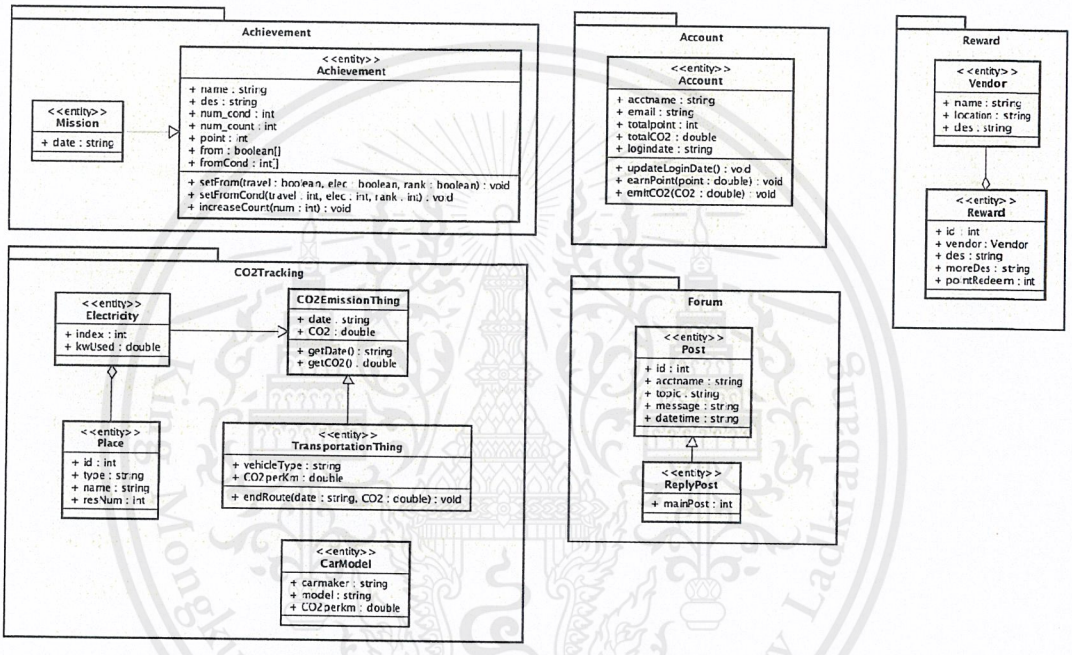


Figure 4.2 Model layer of MVC pattern

Objects in model layer categorizes into five main packages by their duty. These packages are `Account`, `CO2Tracking`, `Achievement`, `Reward` and `Forum`.

`Account` package (Figure 4.3) contains only `Account` class which has duty to keep current user information. `Account` class has five attributes, `acctname` for account name, `email`, `totalpoint` for all points user has at that time, `totalCO2` for amount of carbon dioxide use been emitted from started, and `logindate` hold recent login date of user. `Account` class has three operations. The first operation is `updateLoginDate` that will be called automatically when the application start. It has duty to save current date as last login data to `Account` object.

The `earnPoint` operation will be called when the application wants to update account current point, either increase or decrease. Last operation is `emitCO2` that called when user add activity to the application.

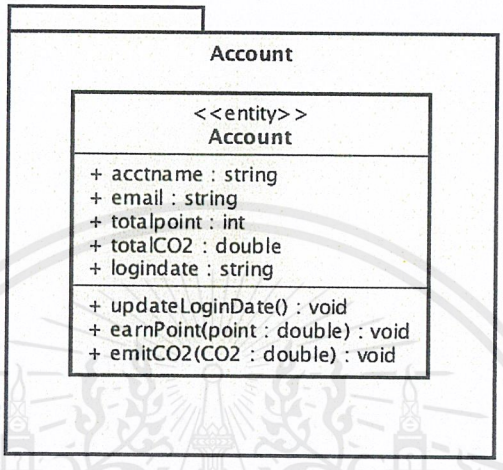


Figure 4.3 `Account` Package

`CO2Tracking` package (Figure 4.4) contains five classes which are `CO2EmissionThing`, `Electricity`, `Place`, `TransportationThing` and `CarModel`. `CO2EmissionThing` class is a super class of `Electricity` class and `TransportationThing` class which contains two attributes and two operations. The `CO2EmissionThing` class has `date` object to define when that thing was used and `CO2` to hold among of carbon dioxide that thing was emitted. Other objects can access to date attribute through operation `getDate` and CO2 attribute through operation `getCO2`. `Electricity` class extends `CO2EmissionThing` class, and each object acts as an electricity bill. It has two additional attributes which are `index` to define unique of object, and `kwUsed` to define how much electricity emitted in kilowatt-hour unit. Moreover, `Electricity` class has a `Place` class as where electricity been emitted. The Place class includes unique identification number as `id` attribute, type of place such as house and condominium as `type` attribute, name to define place as `name` attribute, and number of resident living in the place as `resNum` attribute. Another inheritance of `CO2EmissionThing` is `TransportationThing` class. It has `vehicleType` attribute to define which type of transportation vehicle used and `CO2perKm` attribute to define that vehicle CO_2 emission in gram per one kilometre. Also, it has an operation named `endRoute`. The `endRoute` operation will summarise travel data after it done.

CarModel class is using personal car been chosen for traveling. It contains carmaker as brand of the car, model as car model, and CO2perkm as carbon dioxide emission in gram per kilometre.

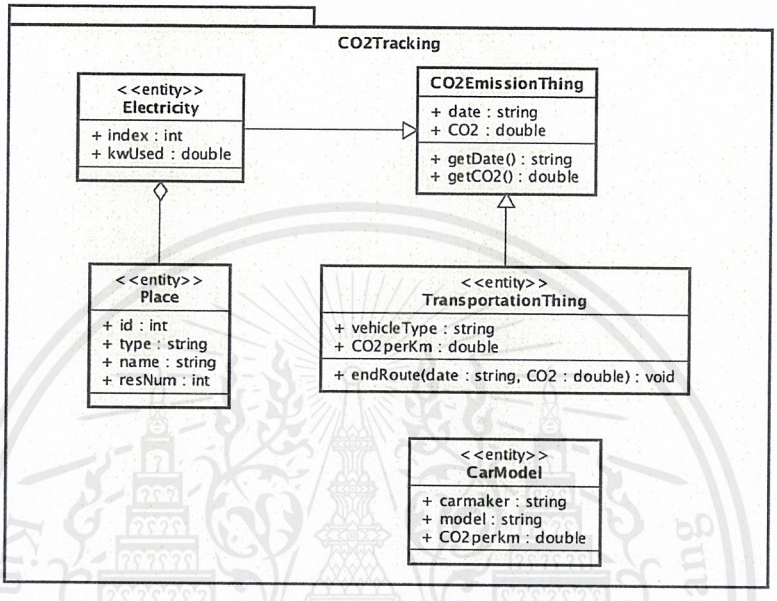


Figure 4.4 CO2Tracking Package

Achievement package (Figure 4.5) has two classes including Achievement and Mission. Achievement class is a super class of Mission class. Achievement class contains seven attributes and three operations that will be describe after. The name attribute defines name of that achievement to show in the application, and also des attribution defines description of the achievement. The num_cond attribute is a static number being as a condition to complete the achievement. The num_count attribute is using with num_cond but for counting. If num_count exceeds num_cond, the achievement is completed. The point attribute is a reward when achievement been accomplished. The from attribute defines the ways achievement can be done which contains transportation, electricity bill, and rank in leaderboard in ordered. The fromCond is a condition for each way to complete. It can be said user has to complete fromCond attribute in from attribute way, then it counts as one in num_count attribute to complete num_cond attribute. The setFrom and setFromCond operations are used to construct an achievement condition. After it pass fromCond, inCreaseCount operation will be called to increase number in num_count attribute.

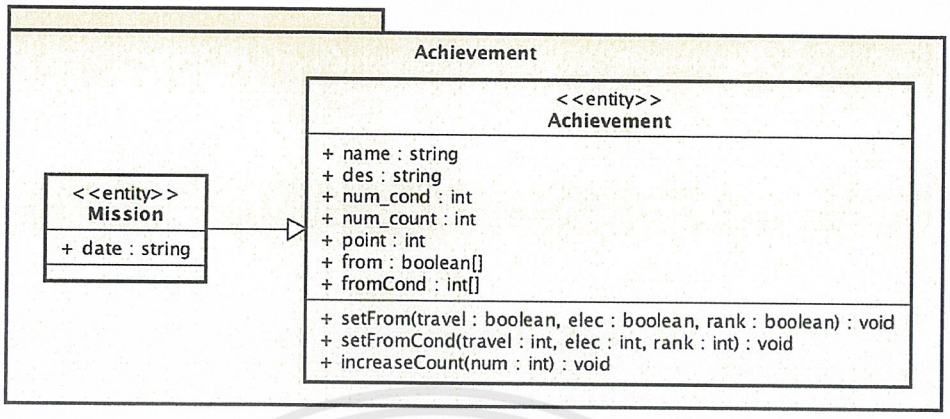


Figure 4.5 `Achievement` Package

`Reward` package (Figure 4.6) has two classes which are `Vendor` and `Reward`. `Vendor` class, representing partners of the application, contains `name` attribute for partner name, `location` for places to redeem, and `des` for partner description. `Reward` class contains `Vendor` inside. It has five attributes which are `id` for unique identity, `vendor` for the brand who offer reward, `des` for short description, `moreDes` for detail of reward, and `pointRedeem` for point transferring.

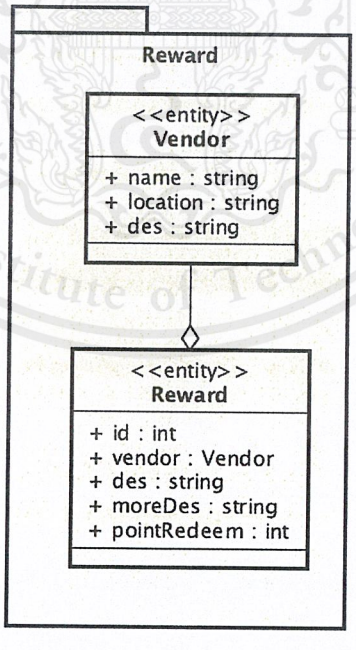


Figure 4.6 `Reward` Package

Reward package (Figure 4.7) has two classes which are Post and ReplyPost. Post class contains id attribute for unique identity, acctname attribute for the owner of post, topic attribute as title, message attribute as body of post, and datetime attribute for date and time post created. ReplyPost class is inherited from Post class. There are only attribute adding which is mainPost. mainPost attribute is acting as a link to its topic.

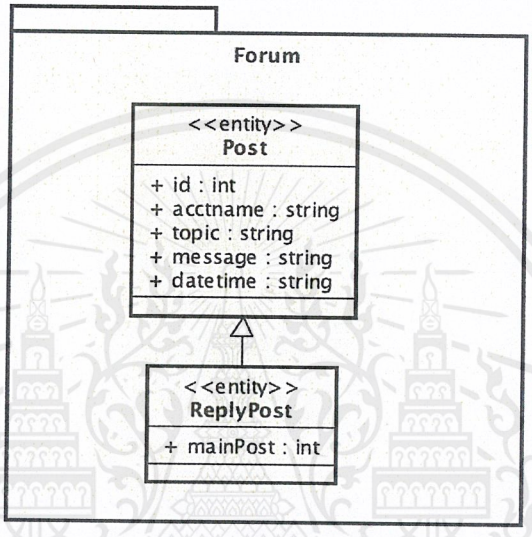


Figure 4.7 Forum Package

The information above contains only model layer. Next part will describe more in applied model layer with controller layer.

The Account class is used for its main duty in three controllers. SignUpCtrl is a controller class for sign up process as its name. It contains accessAcc as attribute to hold input data which are username, email and password. SignUpCtrl class has accessed with local storage through saveToLocal operations that acting as setter to save account data as cache. The signUpClicked operation is a main function in this controller. All processes to sign up for an account must do in this function. SignInCtrl is used for sign in process. It has accessAcc attribute which is the same as in SignUpCtrl. SignInCtrl also save account data to local storage as operate by saveToLocal function. However, there are two ways to log in, through application account or Facebook account. Each way has its function to operate, signInClicked operation for local account and signInFBClicked operation for Facebook account. Moreover, SignInCtrl has retrieve password function handle by forgetPwdClicked and confirmEmail operations.

Finally, `AccessCtrl` class which the first class to be processed in the application. It has duty to check for cache account inside the application. If the account is existed, the application will ignore sign in and sign up step. Otherwise, it still be the bound controller between `SignUpCtrl` and `SignInCtrl`. The `accessAcc` attribute is exactly the same object as two above and been constructed in this class. `AccessCtrl` also has to contact with local storage, so it has `getFromLocal` operation to check for account cache. The `checkLocalAcc` function is operating on checking account in local storage. The remaining operations, `buttonClicked` `setMode` and `closeModal`, are the function to carry switching between sign up and sign in process.

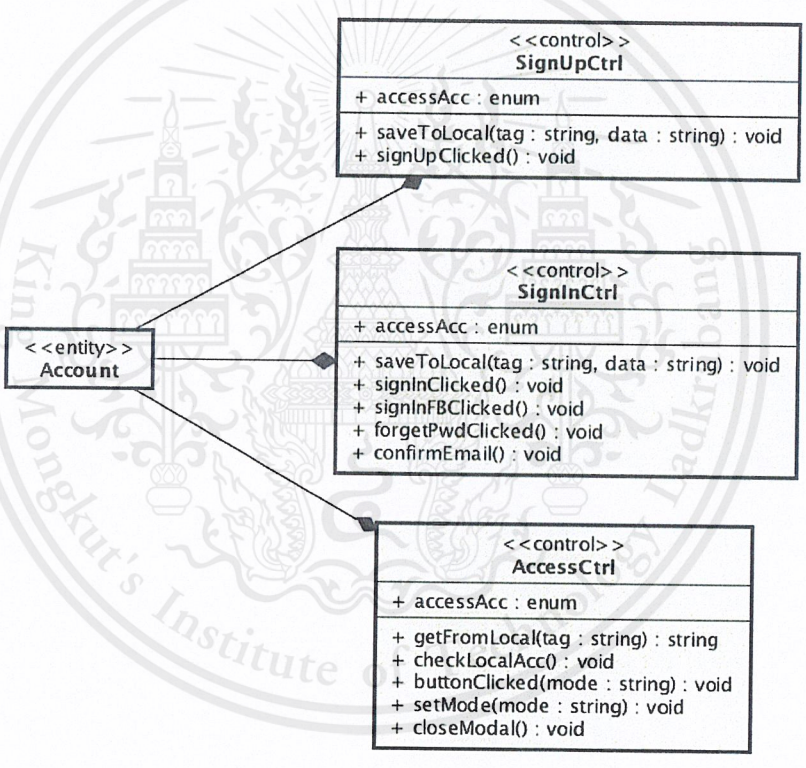


Figure 4.8 `Account` class with its controllers

The Place class is used by HouseAddsCtrl. HouseAddsCtrl controller handles all living place of account myAcc. It has adds and editAdds attribute to hold input fields for adding and editing place. The addsList attribute contains all places of the account. All Operations also be separated by its role into four groups. The getAddsList operation get all places stored to addsList attribute. The saveAddsClicked operation is used to add new living place. The showEditForm and editAddsForm operations perform in each place to display edit form, consequences of them are updateAddsClicked and delAddsClicked operation. The updateAddsClick is used for editing living place and the delAddsClicked is used for remove living place. The Electricity class is used by HouseBillCtrl, similar to Electricity class and Place class relationship. HouseBillCtrl class is a consequence of HouseAddsCtrl class, it is used to add and edit bills of each living place. The date, month, year and monthList attribute are represent ordered previous twelve months in graph of electronic bills in a place. The agraph and series attributes are also the graph components. The addsList attribute is storing all living place as same as in HouseAddsCtrl. By the way, addsList attribute has been proceed into addsBill attribute which adding latest bill inside. The restBill attribute separates unknown bill of latest month from addsBill. The editAccountData operation updates changes on account data which are CO_2 emission and points for this case. getMonthList operation orders months in the monthList. getAddsList, getAddsBillList, and getRestBillList operations produces data in addsList, addsBillList and restBillList attributes. getBillYear operation is called to get data of a place displaying in the graph. MapCtrl controller uses TransportationThing class to define the vehicle using. VehicleSettingCtrl class hold all operations of personal car. It has all brands and models stored in vehicleBrans attribute and vehicleModel attributes. carList attribute stores all personal cars owned by the account. request is an attribute hold requested message for additional car brand and model. getCarList operation performs on carList attribute to get personal car data. selectModel operation was performing after user choose car brand, it will get all models inside the selected brand. saveVehicle and delVehicle operations are used for adding and removing a personal car. MapCtrl class contains buttonStatus attribute used for checking travel state. Operations of MapCtrl are running step by step, starting from prepareMap. It will prepare map fragment and call getStartPoint operation as the beginning of transportation. Then user performs startTravel operation, it will loop calling getCurrentPosition and findDistance operation as updating of travel route. Finally, user performs finishTravel operation, the application will calculate all data and update by editAccountData operation and call

quitMap operation to close the state. TransportCtrl controller performs before map start to define selected vehicle from user. It hold carList attribute as personal car list which been stored by getCarList operation. If user choose personal car as vehicle, user has to choose car model in consequence. selectCarModel operation performs in this step. Otherwise, walkClicked, carClicked, taxiClicked, trainClicked, motorbikeClicked, and busClicked operations will be used depending on selected vehicle.

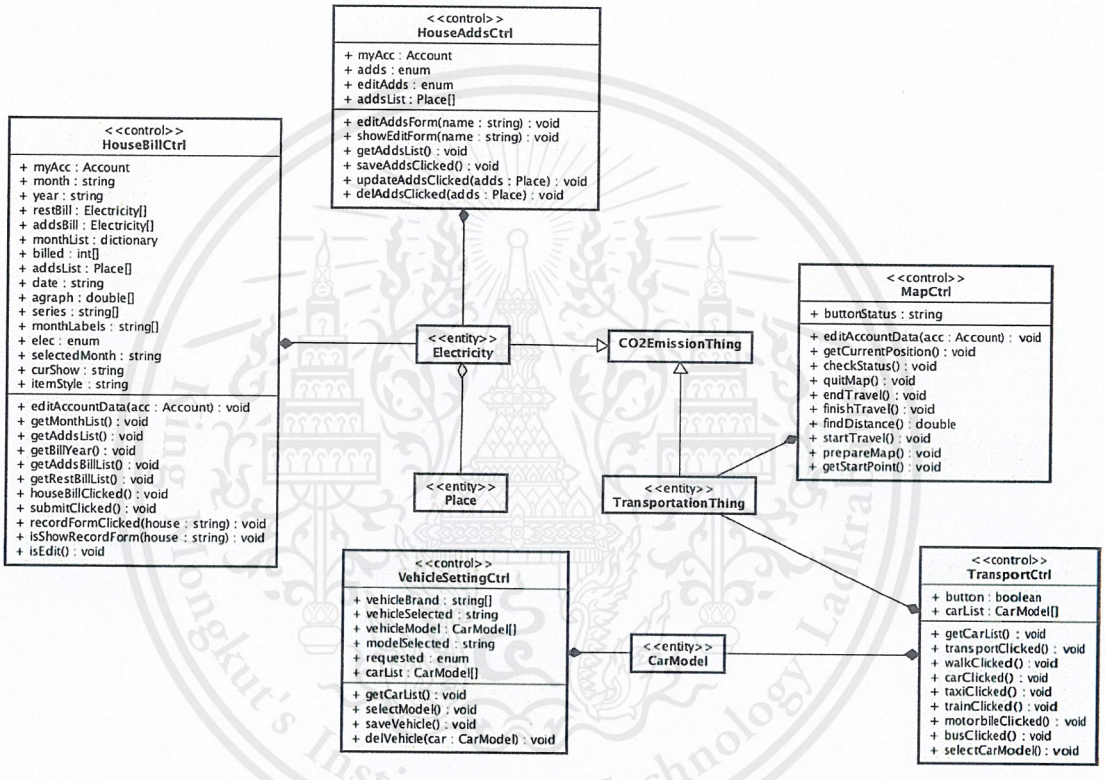


Figure 4.9 CO2Tracking package with its controllers

`Mission` class is used by `MissionCtrl` controller which has a static number represented condition to complete the mission, zero for transportation, one for electricity bill and two for rank from leaderboard. `getMission` operation is called every time user starting the application. It will check for today mission. If there is no mission, it will random a new once through `randNum` operation. When user complete the mission's conditions, `changeOnCO2`, `changeOnDist` and `changeOnRank` operations will be called depending on the mission conditions. After three mentioned operation done, `checkCondCompleted` operation will be used to check is mission done or not. Counting number completed on condition is stored in local storage used `saveToLocal` operation. After mission is completed, points will be update by `editAccountData` operation.

`AchievementCtrl` controller is similar to `MissionCtrl` controller, but change from using `Mission` class to `Achievement` class. `AchievementCtrl` has `achList` attribute to store all achievement to be perform in background. Achievement is not random choose everyday, so it has no operation liked `getMission` operation in `MissionCtrl`. By the way, `changeOnCO2`, `changeOnDist`, `changeOnRank`, `checkCondCompleted` and `editAccountData` are exactly the same as in `MissionCtrl`.

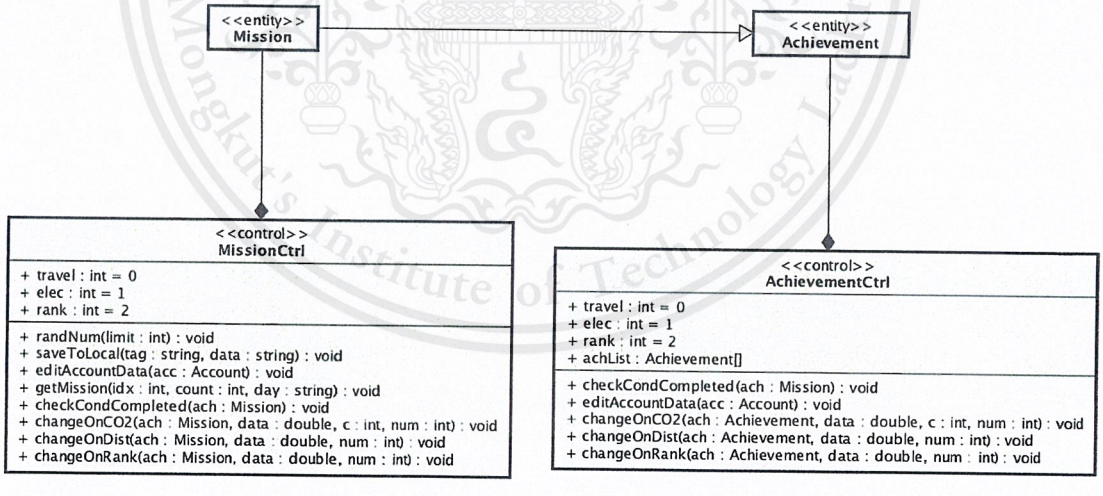


Figure 4.10 `Achievement` package with its controllers

`Vendor` class and `Reward` class are used only by `RewardCtrl` controller which contains `vendorList` and `RewardList` attribute to display rewards in the application. `rewardWrap` and `curReward` attributes are used when switching of reward performed. `getVendorList` and `getRewardList` operations are using to get data vendor and reward then stored them to `vendorList` and `rewardList` attributes. The `redeemReward` operation is called when user want to get the reward. It will call `editAccountData` to update decreasing point.

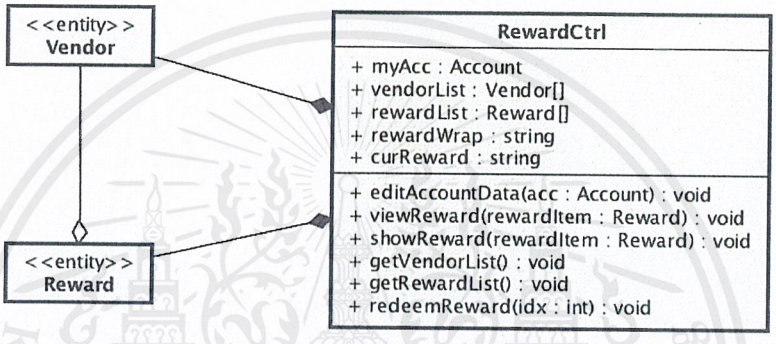


Figure 4.11 `Reward` package with its controllers

The `Post` class is used by three controller class including `ForumCtrl`, `MyFavForumCtrl` and `MyForumCtrl`. `ForumCtrl` controller is likely a combination of `MyFavForumCtrl` and `MyForumCtrl`, but `MyFavForumCtrl` has `getMyFavForum` operation to get favourite forum data to store in `myFavList` attribute and also `MyForumCtrl` has `getMyForum` operation to get own forum list stored to `myPostList` attribute. `ForumCtrl` has `forumFields` attribute to hold data when user want to create a new topic and `allPostList` attribute storing all topics in database which getting data from `getAllForum` operation. `createTopicClicked` and `createClicked` operations are using for create a new topic. When user want to read a post directly, `postClicked` operation will be performed. In the post, user can add the post as favourite through `favPostClicked` operation or leave a message through `commentPostClicked` operation. Although, users can edit their own post through `editClicked` and `editMyForumClicked` operation.

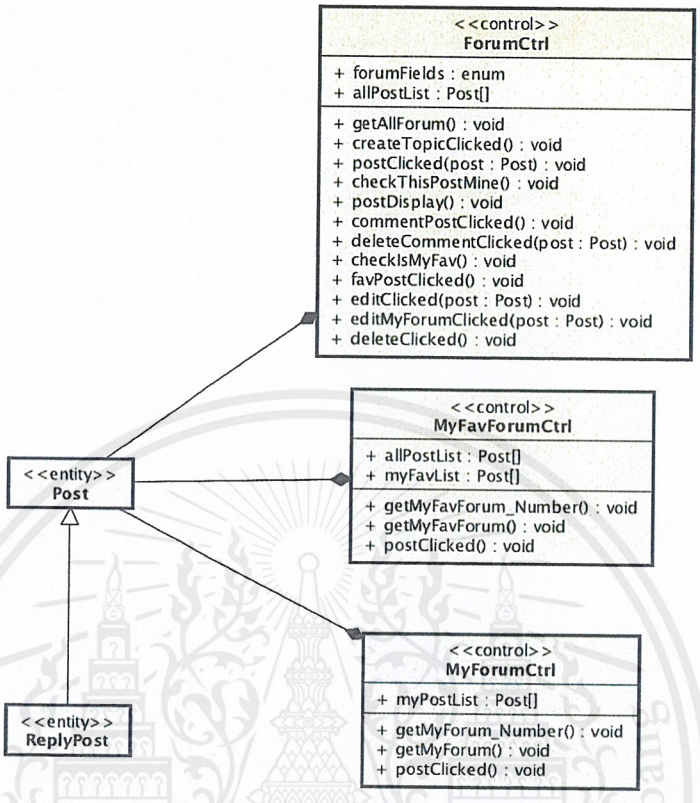


Figure 4.12 `Forum` package with its controllers

4.6.2 Functional Design

4.6.2.1 Sign Up

Signing up uses two classes for implementation which are `SignUpCtrl` as controller and `Account` as model shown in Figure 4.13 and Figure 4.14 respectively.

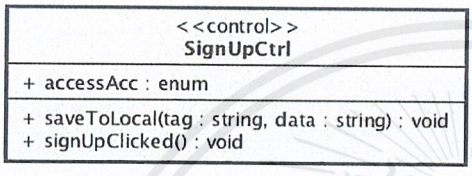


Figure 4.13 `SignUpCtrl` Class Diagram

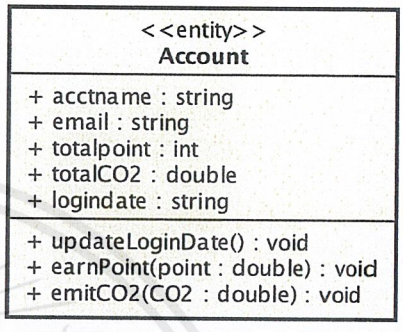


Figure 4.14 `Account` Class Diagram

To sign up an account, user has to select choice to sign up in `welcome` page then input account name, email and password into form and submit data. After that the page will call `SignUpCtrl` class which has duty to validate inputed data. If there are no duplication on account name, `Account` will be created. In addition, `SignUpCtrl` saves account name, email, points, and CO_2 which point, CO_2 initiating at 0. In the `welcome` page will show succeed message then go to `guide` page. Otherwise, `welcome` page will show failure message. The sequence is shown as diagram in Figure 4.15 below.

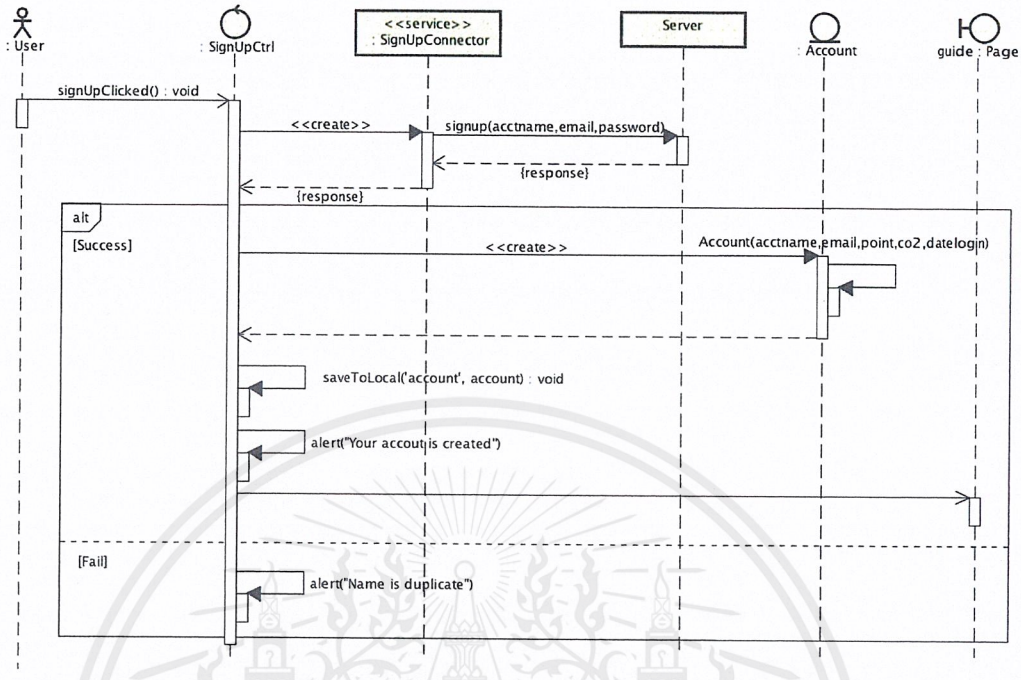


Figure 4.15 Sign Up Sequence Diagram

By the way, consequence physical actions after account was created are same as sign in. The application saves account data into `localStorage` as cache which application can pull these information even the application was closed as describe in Section 4.6.2.2.

4.6.2.2 Sign In

Signing in uses two classes for implementation which are `Account` as model and `SignInCtrl` as controller shown in Figure 4.14 and Figure 4.16 respectively.

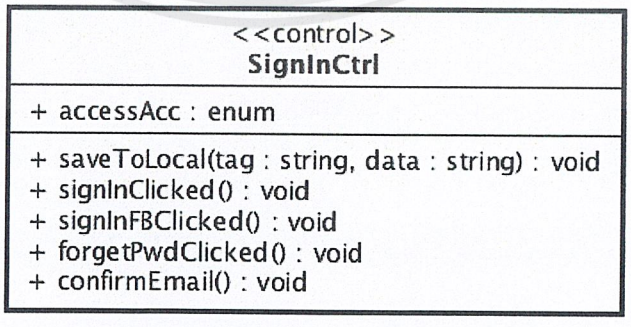


Figure 4.16 `SignInCtrl` Class Diagram

In Figure 4.6 shows a sequence in order to sign in to the account, user has to select choice to sign-in in `welcome` page then input account name and password into form and submit data. After that the page will call `SignInCtrl` that will check for validation of inputed data. If nothing is invalid, `Account` will be created from retuned data in database. In addition, `SignInCtrl` saves account name, email, point, and CO_2. For invalid inputted data, `welcome` page will alert failure message.

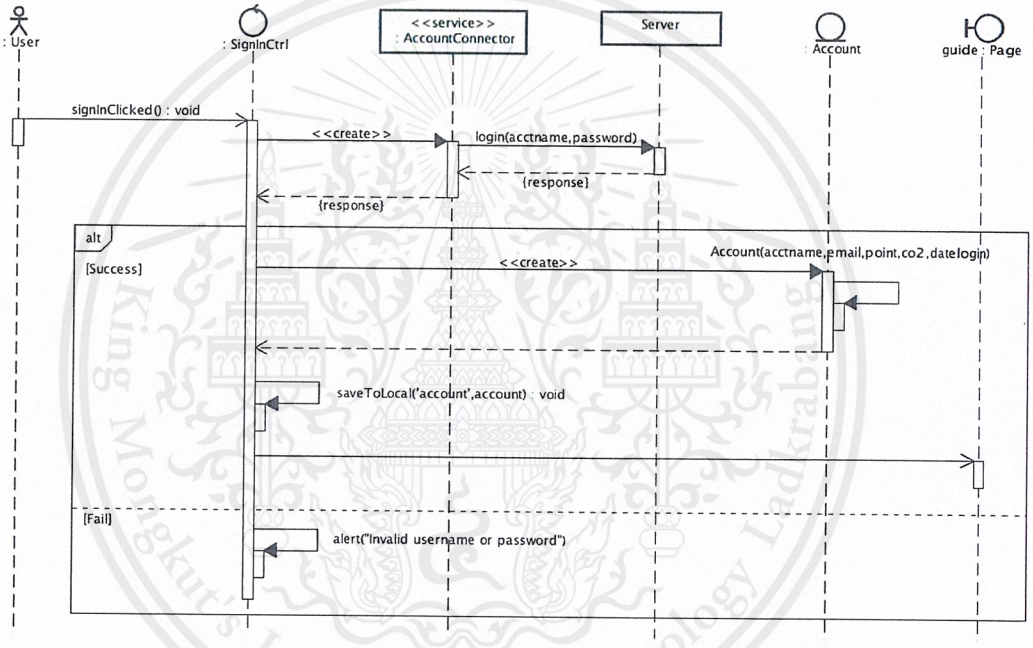


Figure 4.17 Sign In Sequence Diagram

Similar to application account sign in, Facebook account sign in calls different function named `SignInFBClicked`. It will send a signal to server for Facebook token, after that token will be used as a ticket for Facebook account inputted. When the application is done with Facebook API, it will check that is the account existed in database or not. If account was found, the application will have processes as normal sign in. Otherwise, it will automatically sign up for a new application account for Facebook account. These processes are representing in Figure 4.18 and Figure 4.19.

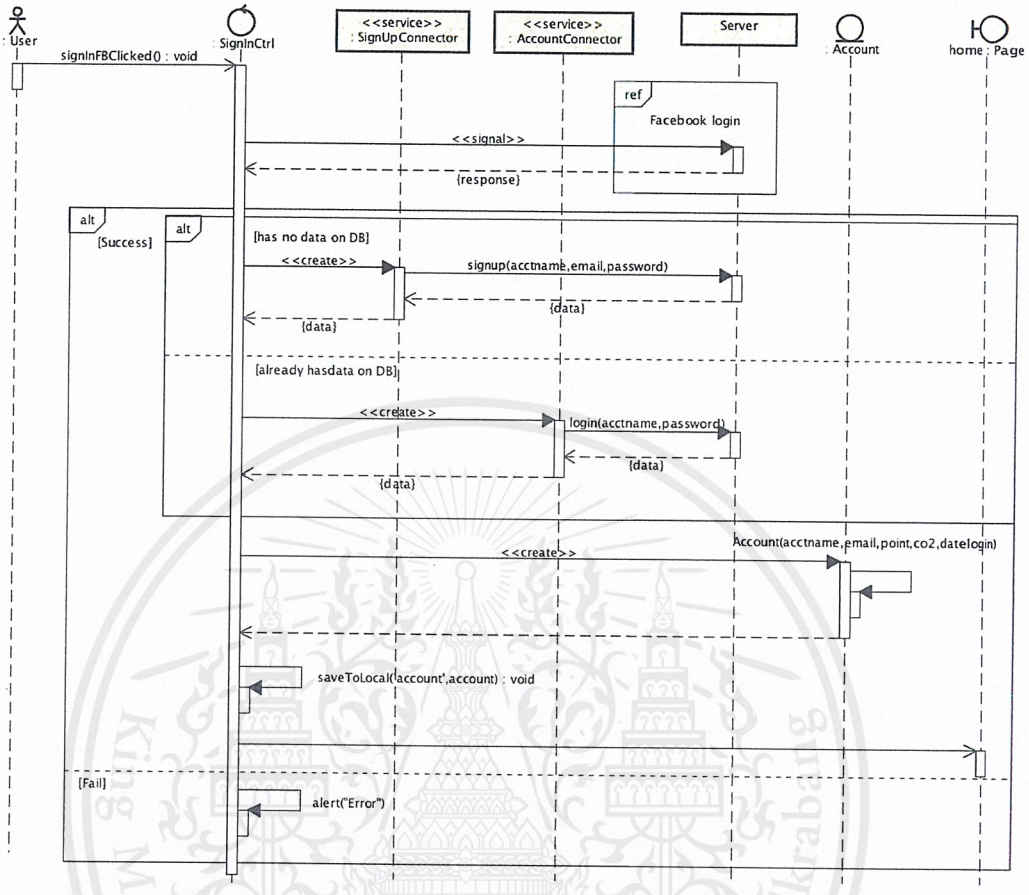


Figure 4.18 Sign In via Facebook Sequence Diagram

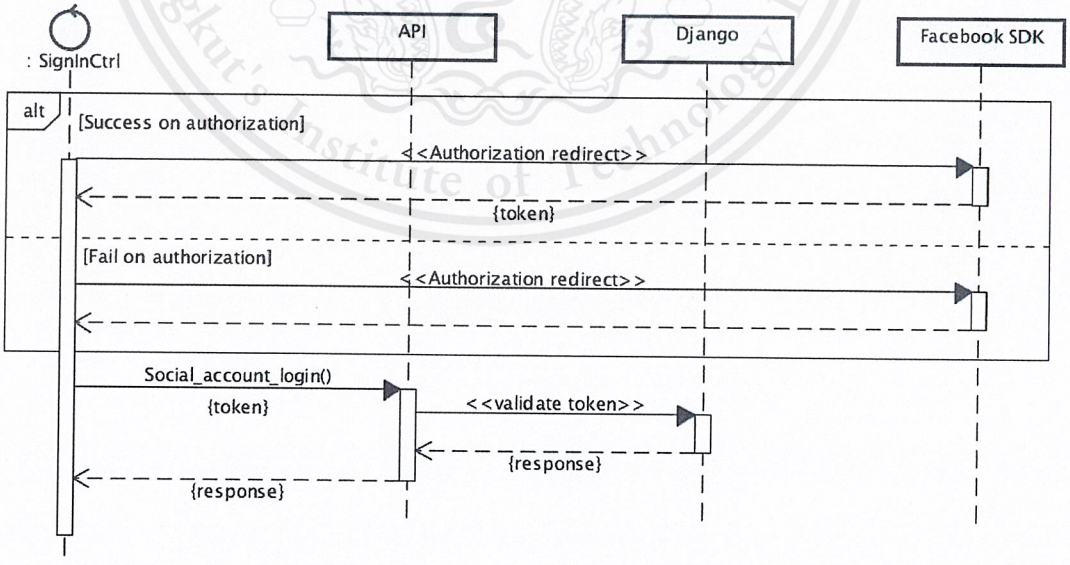


Figure 4.19 Facebook login on Server reference Sequence Diagram

About saving information into local storage, those data will be called as cache that saved on device. Individual data saved on device make the application can get those data without connection to database. For this case, account information makes user had no need to re-log in to the application.

```
//save data to local storage
$scope.savedToLocal = function(tag, data) {
    window.localStorage[tag] = data;
}

//get data from local storage
$scope.getFromLocal = function(tag) {
    return window.localStorage[tag];
}
```

Figure 4.20 `SignInCtrl` with Local Storage Saved

4.6.2.3 Sign Out

Signing out uses `SignOutCtrl` for implementation as shown in Figure 4.21.

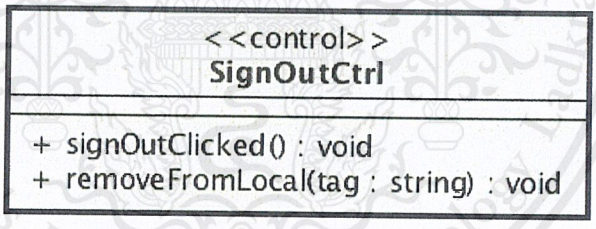


Figure 4.21 `SignOutCtrl` Class Diagram

To sign out from an account, a user has to click at sign out button in setting page then `SignOutCtrl` will clear all data in local storage and bring user to welcome page.

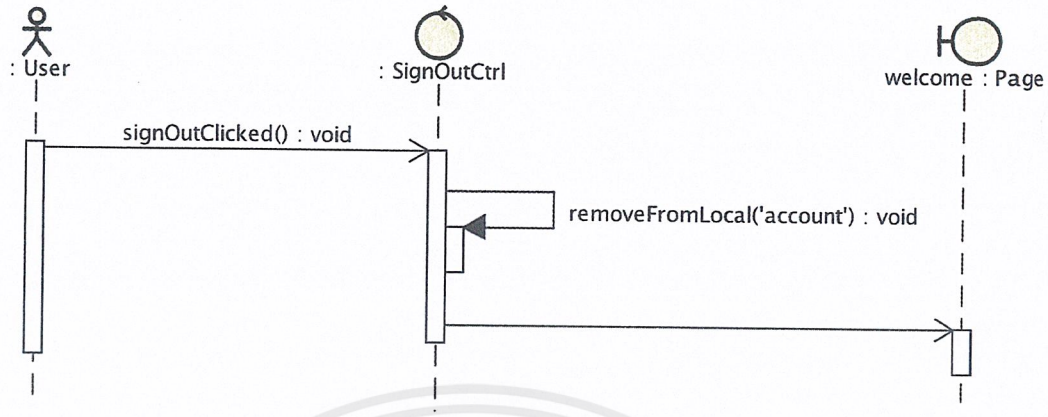


Figure 4.22 Sign Out Sequence Diagram

To removing information in local storage, application has to remove item by item by defining its name as use command as shown in Figure 4.23.

```
//remove data to local storage
$scope.removeFromLocal = function(tag) {
    window.localStorage.removeItem(tag);
}
```

Figure 4.23 `SignOutCtrl` with local Storage Removed Items

4.6.2.4 Edit Account

Editing account uses two controller classes which is `AccountCtrl` and `IdentifyCtrl` as shown in Figure 4.24 and Figure 4.25 respectively.

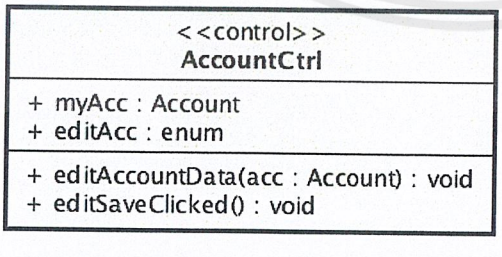


Figure 4.24 `AccountCtrl` Class Diagram

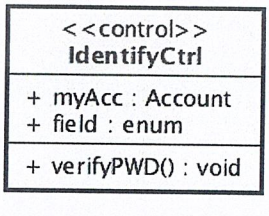


Figure 4.25 `IdentifyCtrl` Class Diagram

To edit the account, user will begin at setting page then click at edit account tab. Next, users have to identify themselves first which provided by `IdentifyCtrl`. If success , `IdentifyCtrl` will send user to `editAccount` page. After that user will fill in changed data then submit. `AccountCtrl` class check for validation in the form. If they have nothing wrong, data will be edited. Otherwise, the application will alert failure message. However, changed on email is affect to kept account data so email in local storage has to be changed too. The sequence is shown in Figure 4.26.

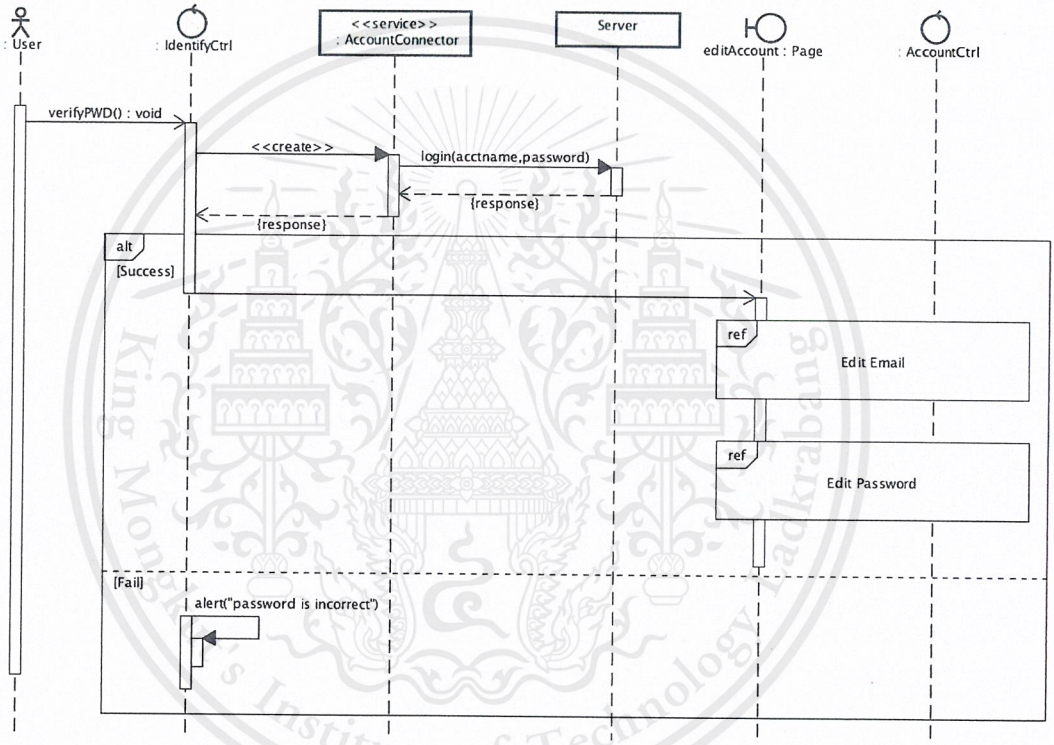


Figure 4.26 Edit Account Sequence Diagram

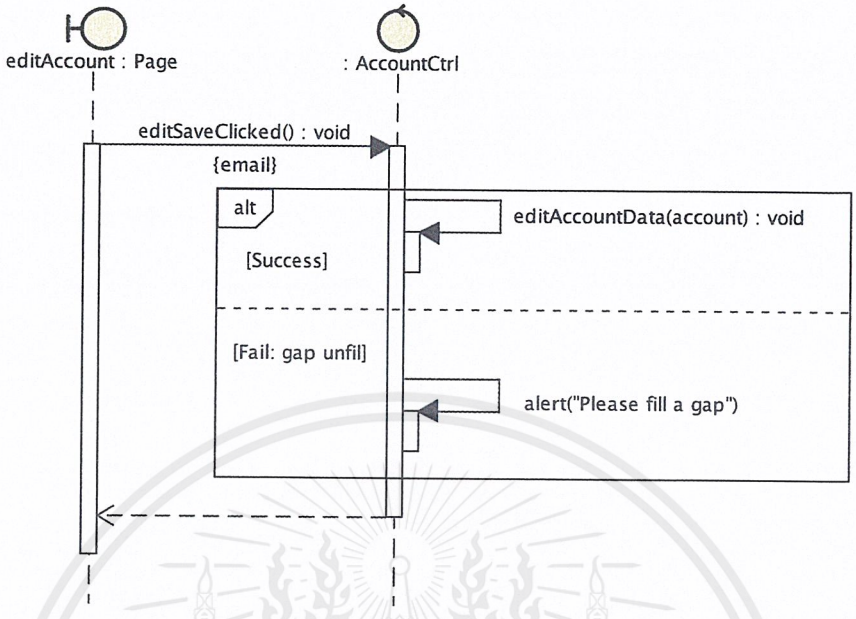


Figure 4.27 Edit Email Reference Sequence Diagram

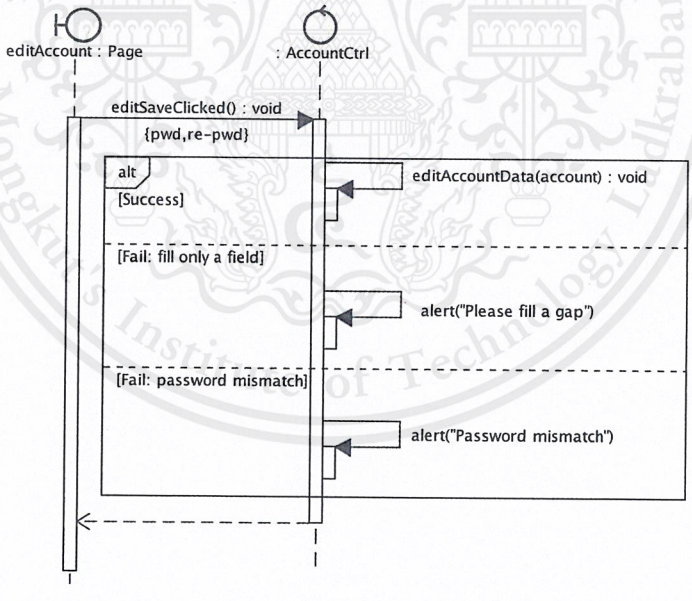


Figure 4.28 Edit Password Reference Sequence Diagram

4.6.2.5 Daily Mission and Achievement

Daily mission and achievement uses five classes for implementation which are `Achievement`,`Mission`,`MissionCtrl`, `AchievementCtrl` and `AchievementViewCtrl` as shown in Figure 4.29, Figure 4.30, Figure 4.31 and Figure 4.32. `Mission` is derivative from `Achievement`.

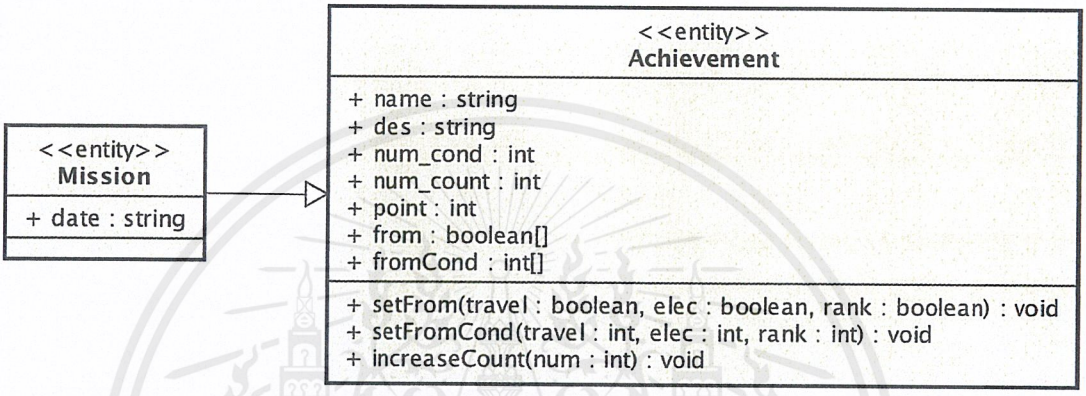


Figure 4.29 `Achievement` and `Mission` Class Diagram

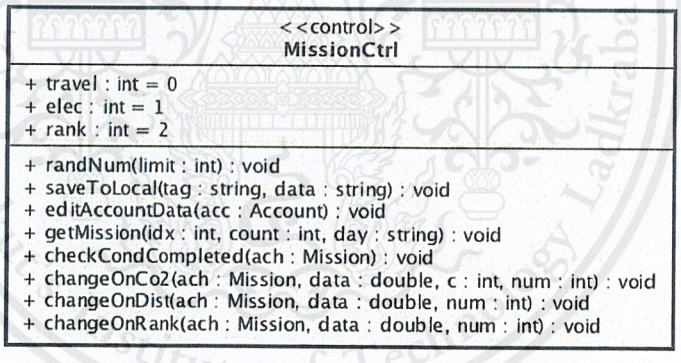


Figure 4.30 `MissionCtrl` Class Diagram

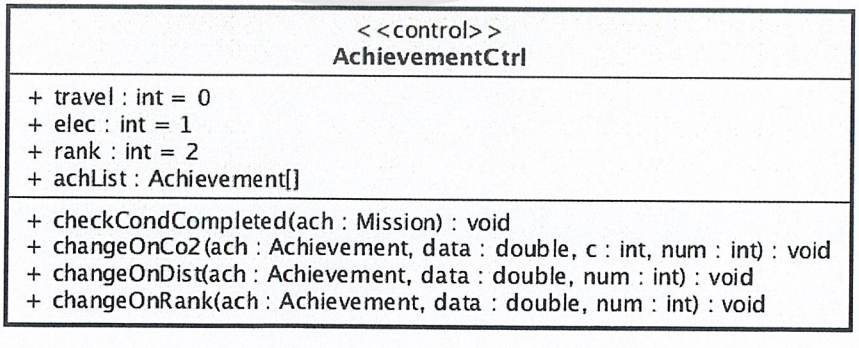


Figure 4.31 `AchievementCtrl` Class Diagram

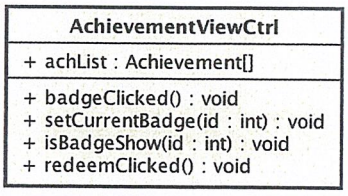


Figure 4.32 `AchievementViewCtrl` Class Diagram

The application will random daily mission when user is in `home` page for first time application opening. The `home` page will send command to `MissionCtrl` to check that is any mission assigned today. If there is no mission found, `MissionCtrl` will random a number as picked index of mission. The Random Daily Mission Sequence Diagram is shown as sequence diagram in Figure 4.33.

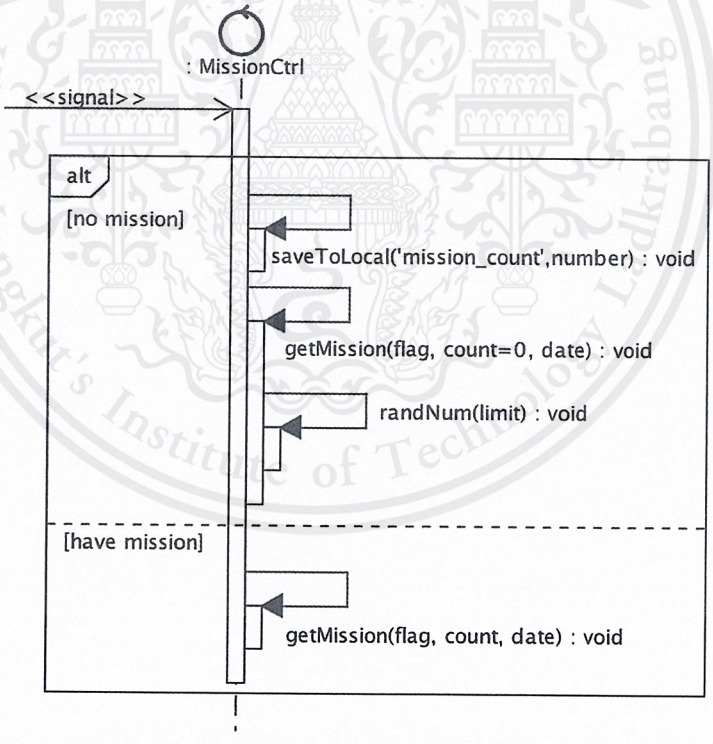


Figure 4.33 Random Daily Mission Sequence Diagram

After user done any activity which contained in missions' condition, system will check for completed of mission. There are three cases of missions' condition including amount of CO_2 emission, travel distance and rank in leaderboard. By the way, it is the same processes on daily mission and achievement only class used is different, `MissionCtrl` for daily mission and `AchievementCtrl` for achievement.

User can check for completed achievement by goto achievement tab. It will show completed achievement badges in colors. Otherwise, grayscale image. This part provided by `AchievementViewCtrl`.

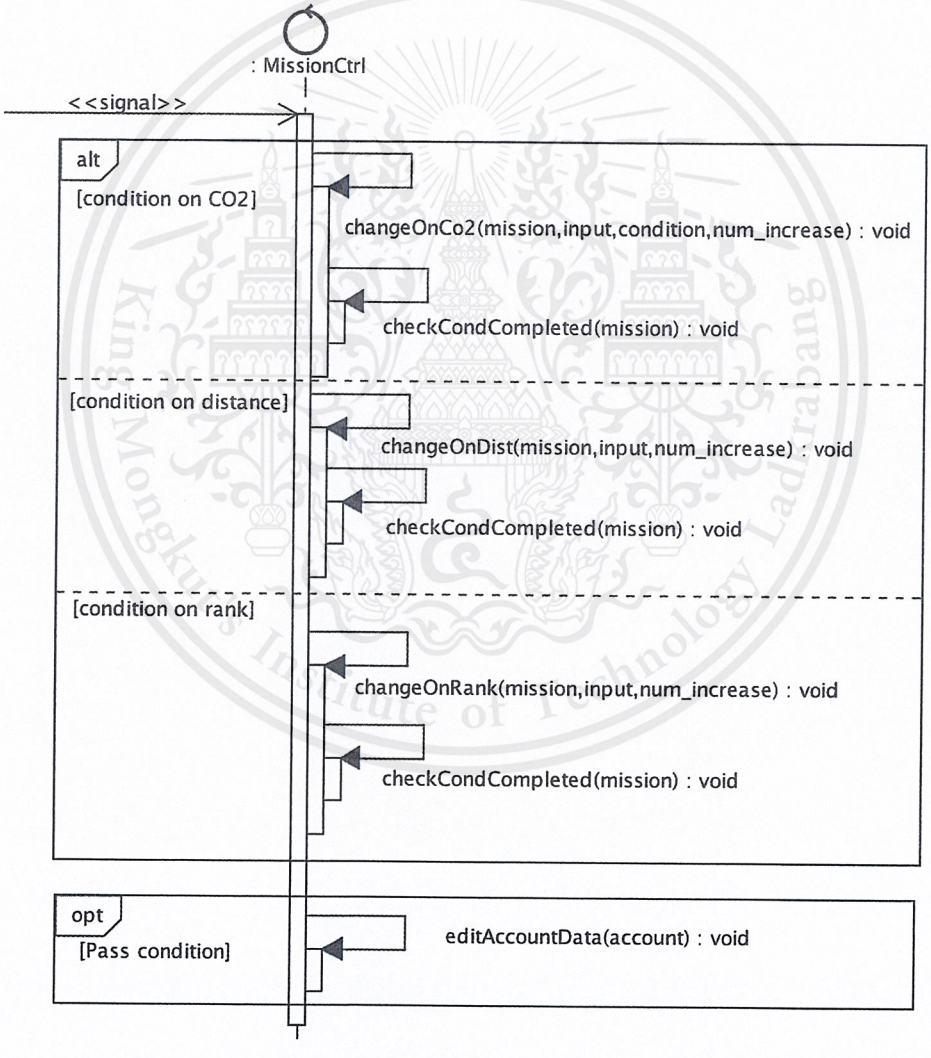


Figure 4.34 Complete Mission Sequence Diagram

4.6.2.6 Track The Emission by Electricity Bills

CO_2 emission tracking by electricity bills uses five classes which are `Co2EmissionThing`, `Electricity`, `Place`, `HouseAddsCtrl` and `HouseBillCtrl` for implementation as shown in Figure 4.35, Figure 4.36 and Figure 4.37. `Electricity` is used `Co2EmissionThing` as super class and has `Place` contained.

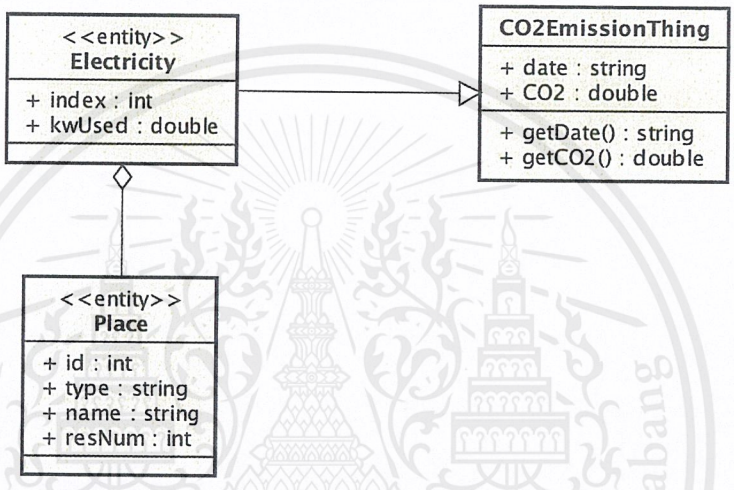


Figure 4.35 `Co2EmissionThing`, `Electricity` and `Place` Diagram

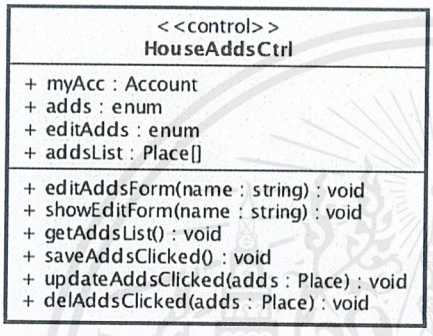


Figure 4.36 `HouseAddsCtrl` Class Diagram

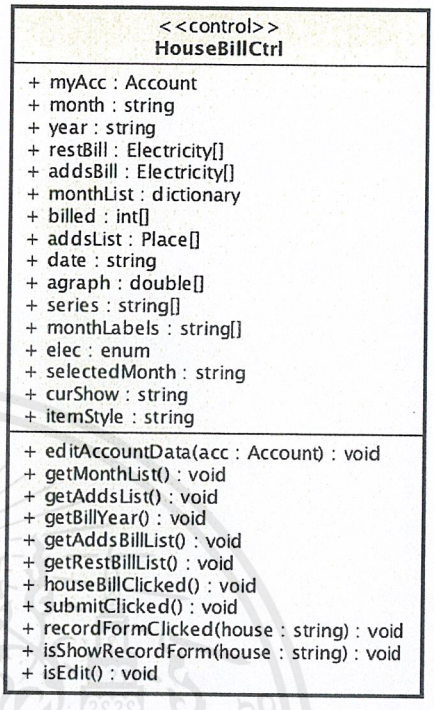


Figure 4.37 `HouseBillCtrl` Class Diagram

Before users add electricity bills, they have to add their address details first by going to setting page and click Address button. `HouseAddsCtrl` will provide all about address list, a list of `Place`, including add, edit and delete. First, the `HouseAddsCtrl` will prepare the page by getting all addresses that have been assigned before. Then, user can add new address by inputting data into adding form also edit and delete by clicking at desired address.

After that users can put bill by choosing a location which have been initiated by `HouseBillCtrl`. Location details including graph of bills in past twelve months will be push down then also adding form. Then, user fills the form and submit data. `HouseBillCtrl` will immediately update data and represent them in graph.

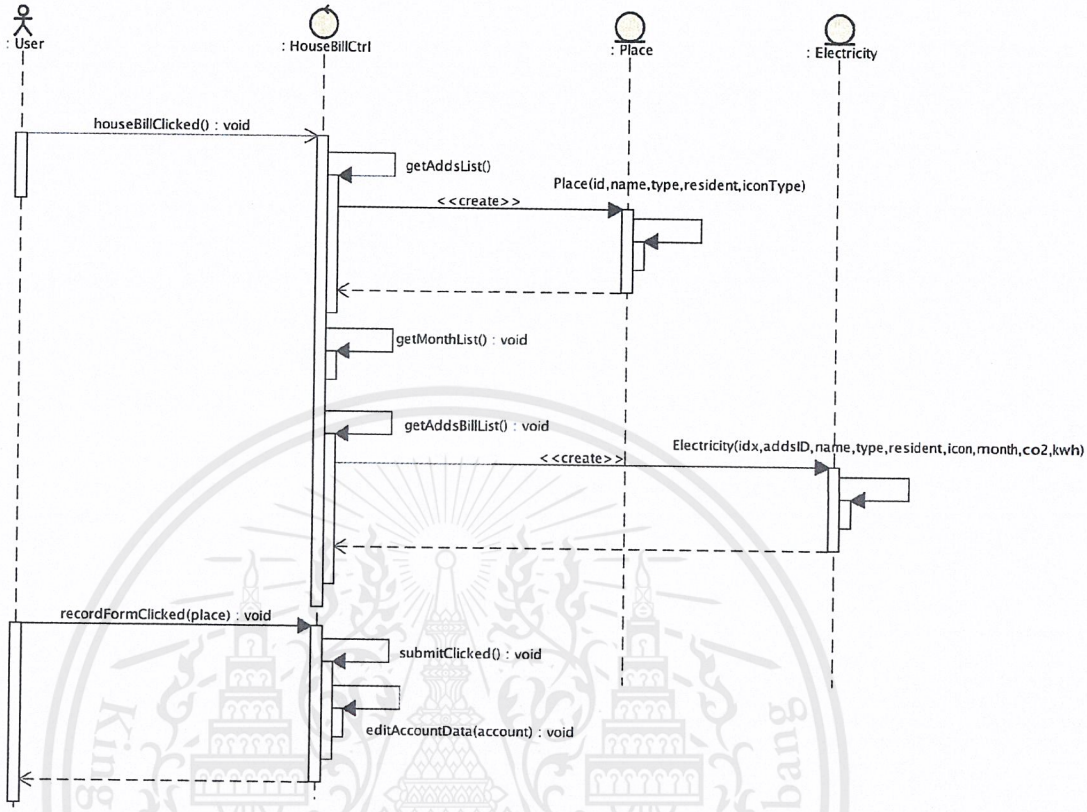


Figure 4.38 Add Electricity Usage Sequence Diagram

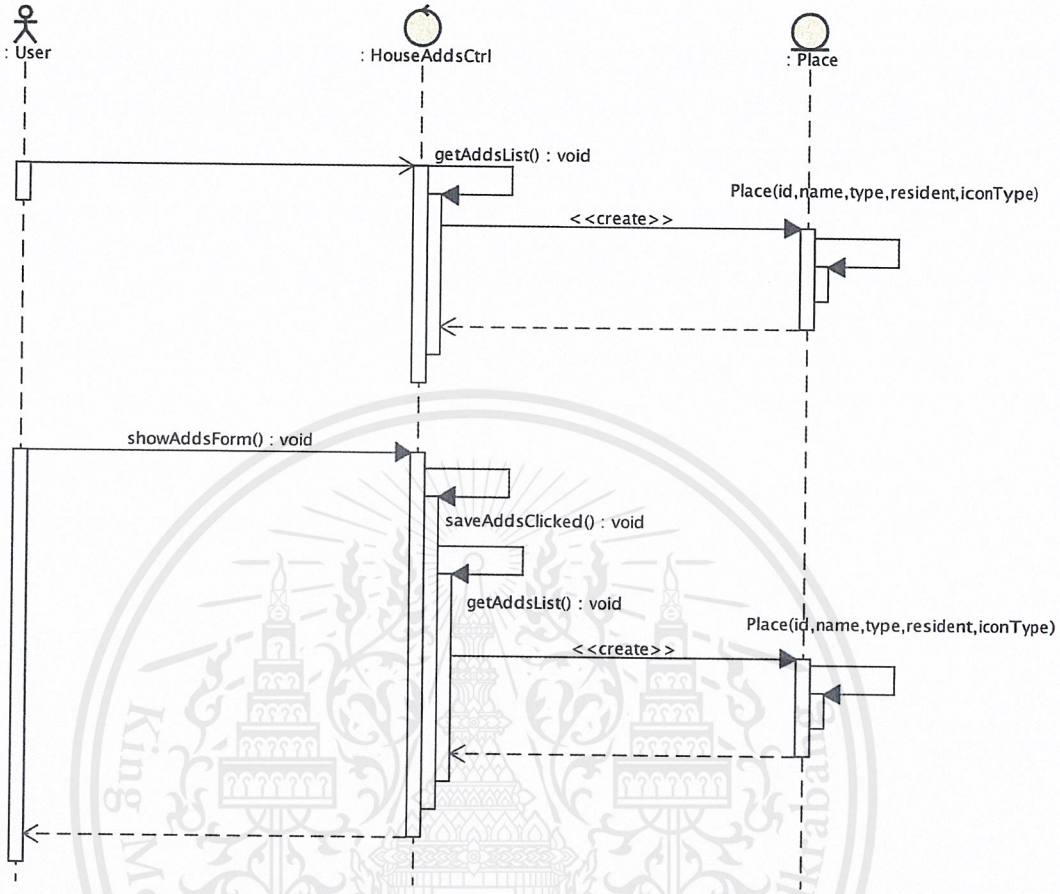


Figure 4.39 Adding Places Sequence Diagram

4.6.2.7 Track The Emission by Traveling

CO_2 emission tracking by traveling uses these classes for implementation which are `Co2EmissionThing`, `TransportationThing`, `CarModel`, `VehicleSettingCtrl`, `TransportCtrl` and `MapCtrl` as shown in Figure 4.40, Figure 4.41, Figure 4.42, Figure 4.43 and Figure 4.44. `TransportationThing` is a subclass of `Co2EmissionThing`.

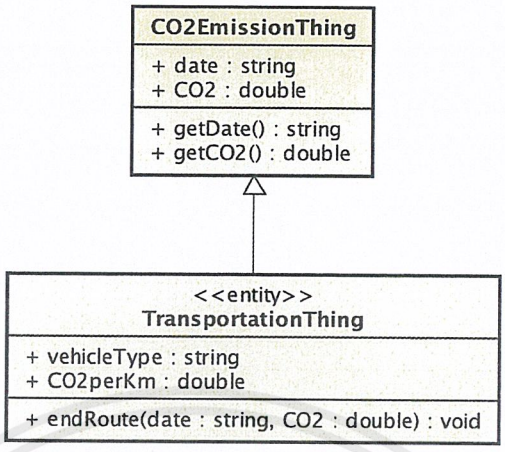


Figure 4.40 `Co2EmissionThing` and `TransportationThing` Class Diagram

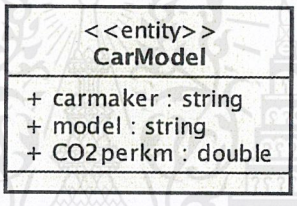


Figure 4.41 `CarModel` Class Diagram

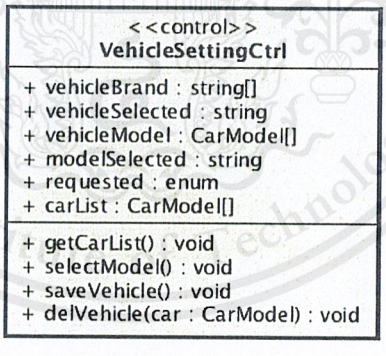


Figure 4.42 `VehicleSettingCtrl` Class Diagram

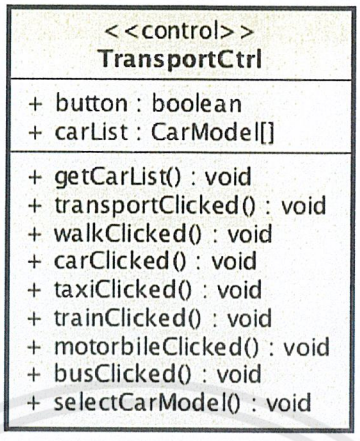


Figure 4.43 `TransportCtrl` Class Diagram

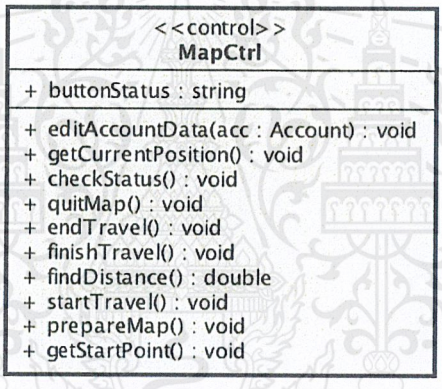


Figure 4.44 `MapCtrl` Class Diagram

There are several ways to make a transportation including train, taxi, bus, motorbike, car and walk/bike. By the way, individual cars have to assign before users start traveling by car. Users can add their car models by going to setting page and click Transportation button. `VehicleSettingCtrl` will initiate list of car that have been add before. Next, users have to choose car maker and car model then submit information. `VehicleSettingCtrl` will update car model list. Users also can delete added car model by click at minus button at the end of car model in added car model list.

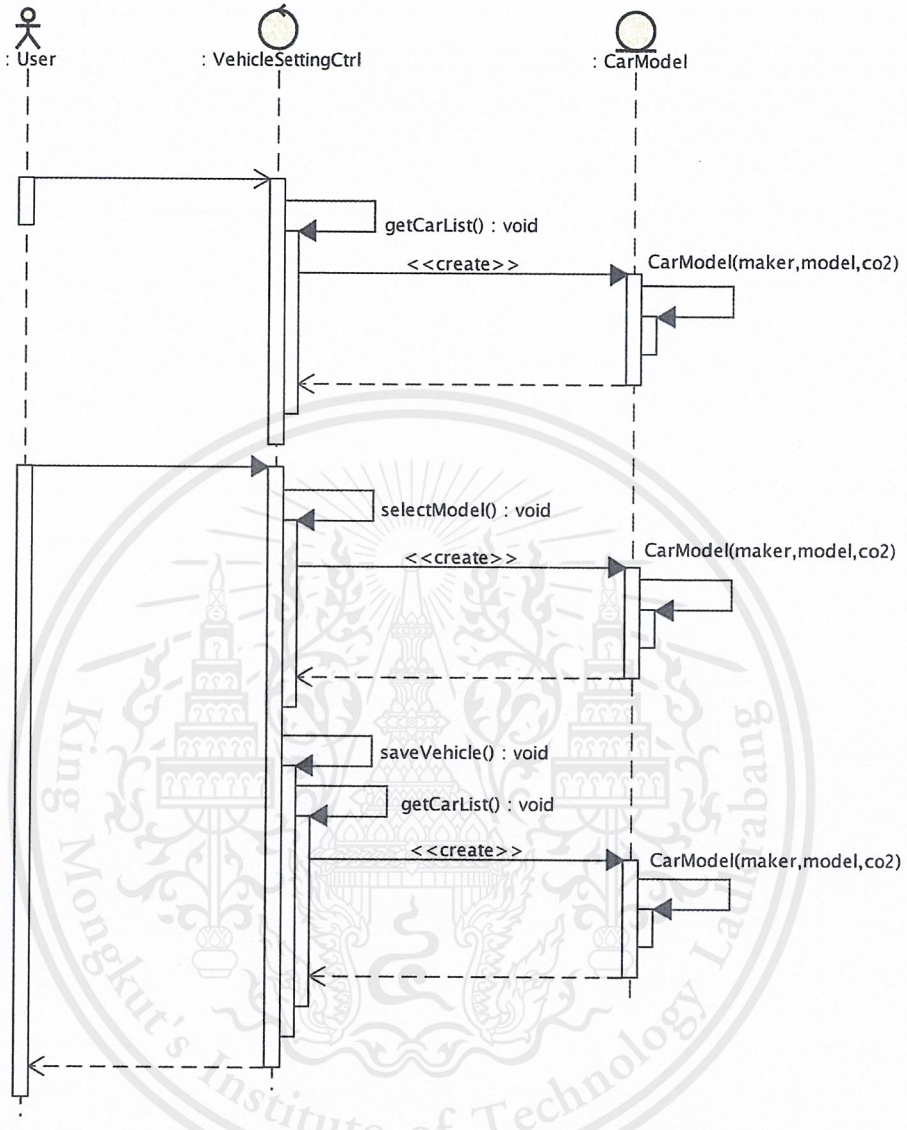


Figure 4.45 Adding Car Model Sequence Diagram

To start travel, user has to choose vehicle type first. If user travels using car, there is a car list popping up for choosing car model. `TransportCtrl` has duty on vehicle type selected. After vehicle was chosen `TransportCtrl` will create a `TransportationThing` object and send it to `MapCtrl`. `MapCtrl` will start with preparing map object which is provided by Cordova Plugin Googlemaps. When user clicks start travel button, `MapCtrl` will start tracking changed on location. For this case, the application need ten meters on differential to notice on location change.

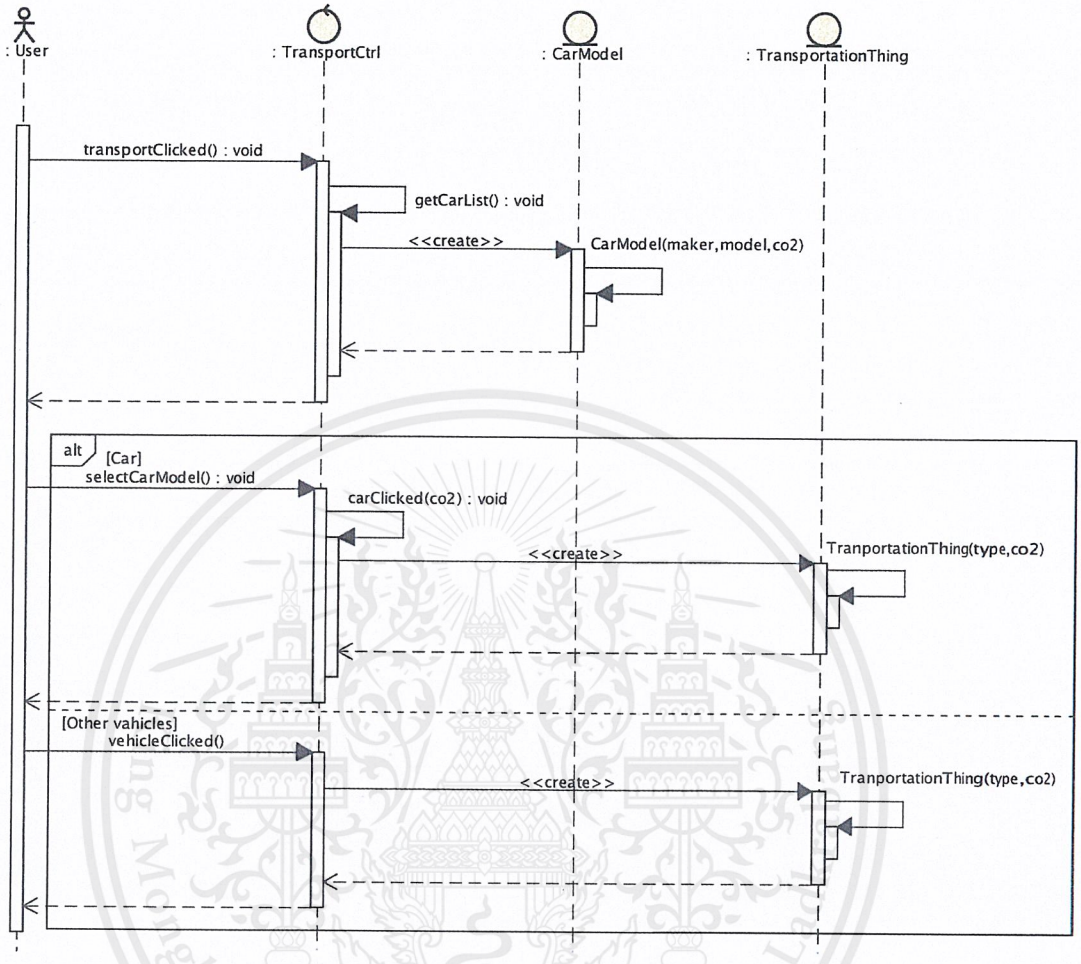


Figure 4.46 Preparing Vehicle for Traveling Sequence Diagram

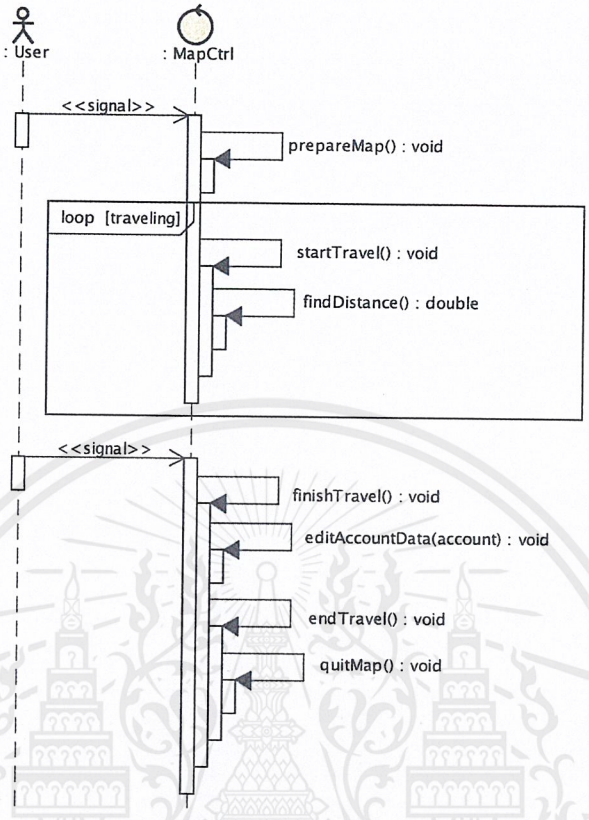


Figure 4.47 Traveling Sequence Diagram

Also, any changed on location makes application calculated a distance by finding a scale for latitude and longitude on two locations as in Figure 4.48. When user want to end the travel and user clicks at stop travel button, the application will be froze for a while then the map will be terminated.

```
$scope.findDistance = function(loc1, loc2) {
    var R = 6371;
    var roundNum = 1000000;

    // //3
    var dlng = Math.abs( Math.round((loc2.lng - loc1.lng)*roundNum) / roundNum) * 111.23;
    var dlat = Math.abs( Math.round((loc2.lat - loc1.lat)*roundNum) / roundNum) * 111.23;
    var d = Math.sqrt((dlng * dlng) + (dlat * dlat));
    console.log(' dlng:'+dlng+' dlat:'+dlat);

    return d;
}
```

Figure 4.48 `findDistance` Function inside `MapCtrl` Class

4.6.2.8 Leaderboard

Leaderboard uses only `RankCtrl` for implementation as shown in Figure 4.49.

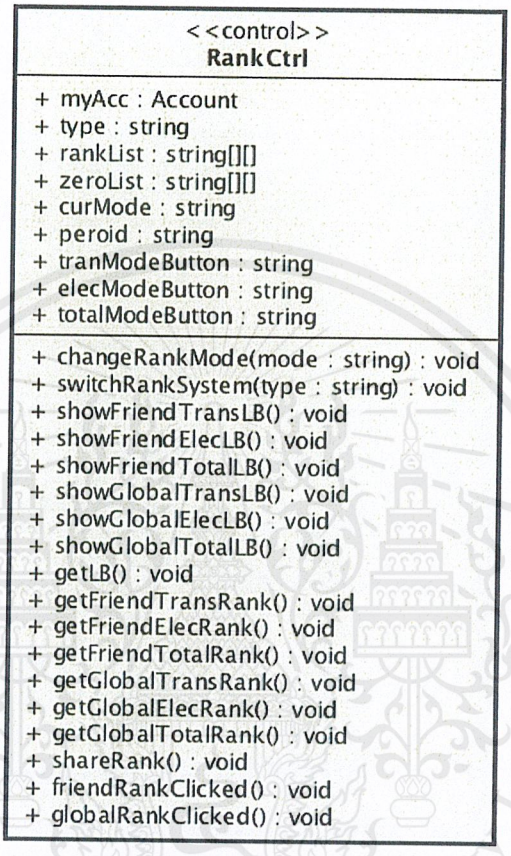


Figure 4.49 `RankCtrl` Class Diagram

There are six categories on leaderboard which been defined by range of users and source where CO_2 emission. Those categories contains friend scope and CO_2 emission from transportation period on a week, friend scope and CO_2 emission from electricity period on a month, friend scope and CO_2 emission from transportation and electricity period on a month, all users scope and CO_2 emission from transportation period on a week, all users scope and CO_2 emission from electricity period on a month, and all users scope and CO_2 emission from transportation and electricity period on a month. When conditions were changes, `RankCtrl` will call functions to update leaderboard and rank.

4.6.2.9 Share to Social Network

Sharing function has no class. It uses only a function on `RankCtrl` as shown in Figure 4.49. The Figure 4.50 is a construction of `shareRank` function.

```
$scope.shareRank = function() {
  message = "My Rank is " + $scope.myRank + " from CO2 emission "+ $scope.co2 + " g";
  //$cordovaSocialSharing
  $cordovaSocialSharing
    .share(message) // Share via native share sheet
    .then(function(result) {
      // Success!
    }, function(err) {
      // An error occured. Show a message to the user
      alert("Sharing fail!");
  });

}
```

Figure 4.50 `shareRank` Function inside `RankCtrl` Class

To sharing something to social network, user just only click on share button then the `ShareRank` will be called. In addition, sharing function uses a plug-in called Cordova Social Sharing.

4.6.2.10 Rewards

Redeeming rewards uses three classes for implementation which are `Vendor`, `Reward` and `RewardCtrl` as shown in Figure 4.51 and Figure 4.52.

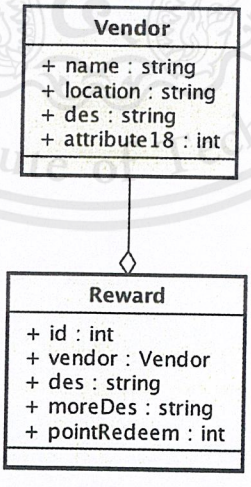


Figure 4.51 `Vendor` and `Reward` Class Diagram

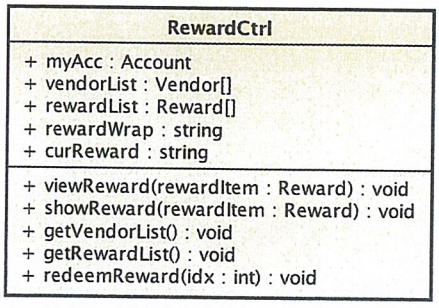


Figure 4.52 `RewardCtrl` Class Diagram

To redeem a reward, user has to go to redeem page continued from achievement page. Then, `RewardCtrl` will get all vendors and use those vendors to get a list of rewards. Next, users click at reward they wanted to redeem and click redeem button. By the way, a reward can redeem once a week.

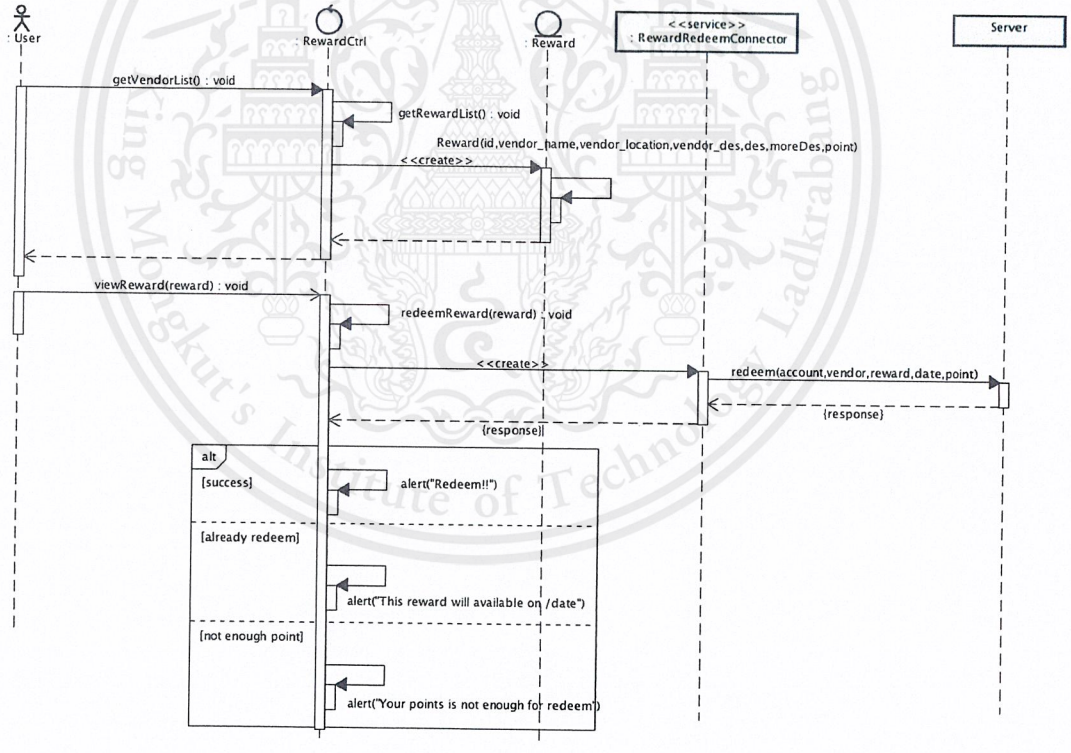


Figure 4.53 Redeem Reward Sequence Diagram

4.6.2.11 Search Account

Adding friend uses only `SearchAccountCtrl` shown in Figure 4.54.

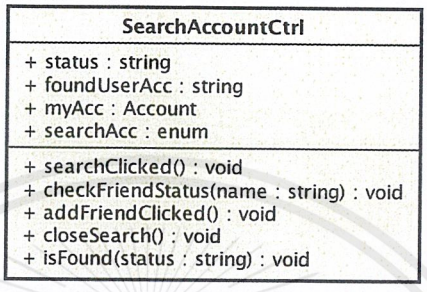


Figure 4.54 `SearchAccountCtrl` Class Diagram

To add friend, users have to search for the account first. If the account was found, users can add that account to their friends list. `SearchAccountCtrl` provides every steps from search account, check friend status and add friend.

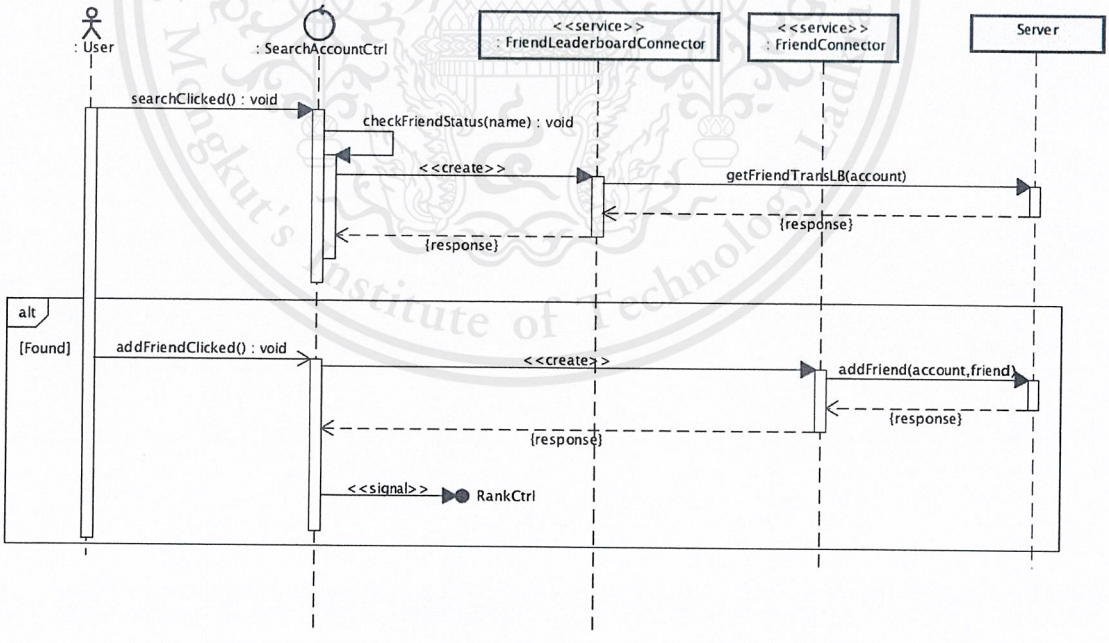


Figure 4.55 Adding a Friend Sequence Diagram

4.6.2.12 Create a Post

Creating post topic uses two classes for implementation which are `Post` and `ForumCtrl` showing in Figure 4.56 and Figure 4.57.

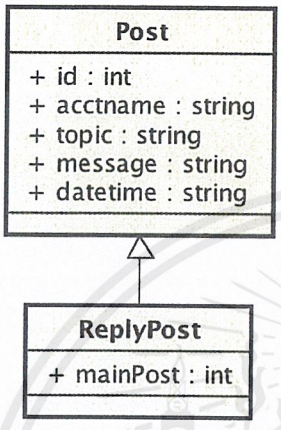


Figure 4.56 `Post` and `ReplyPost` Class Diagram

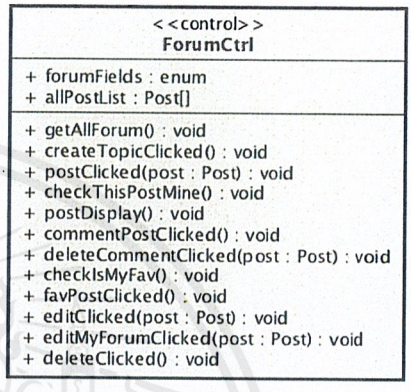


Figure 4.57 `ForumCtrl` Class Diagram

Users have to access to `forum_create` page for creating a new topic. Then, they put forum topic and forum message in the boxes showing. After submit the forum, application will bring users to the topic page.

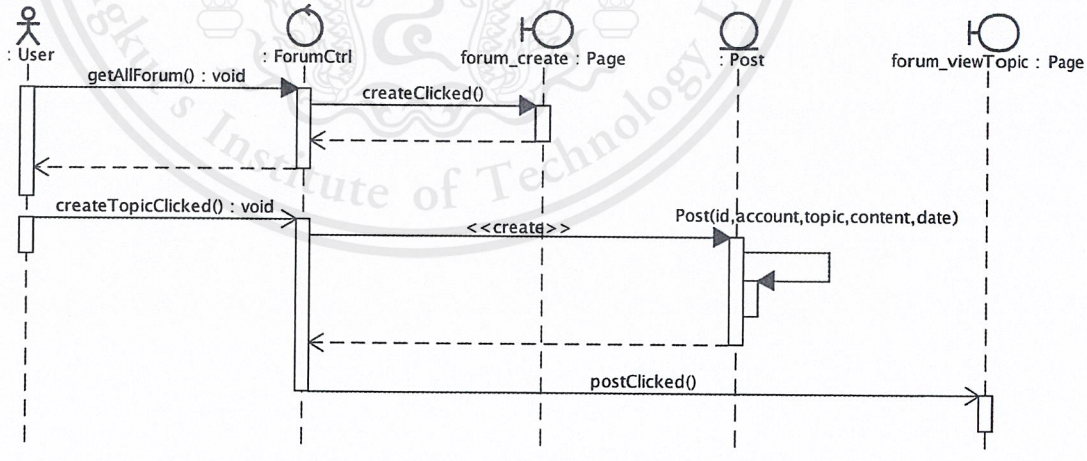


Figure 4.58 Create a Post Sequence Diagram

4.6.2.13 Read and Reply a Post

To read and reply a post, there are three classes being used for implementation which are `Post`, `ReplyPost` and `ForumCtrl` as shown in Figure 4.56 and Figure 4.57 previously.

Users can click at the post to read its content and reply comments. After `ForumCtrl` get all topics in form of `Post` objects. Users choose a post from the list, then the post send command `postClicked` to `ForumCtrl`. In case users want to leave a message, they put that message in text box and submit. `ForumCtrl` will call function `commentPostClicked` as sending data to database and refresh the topic to show recent message.

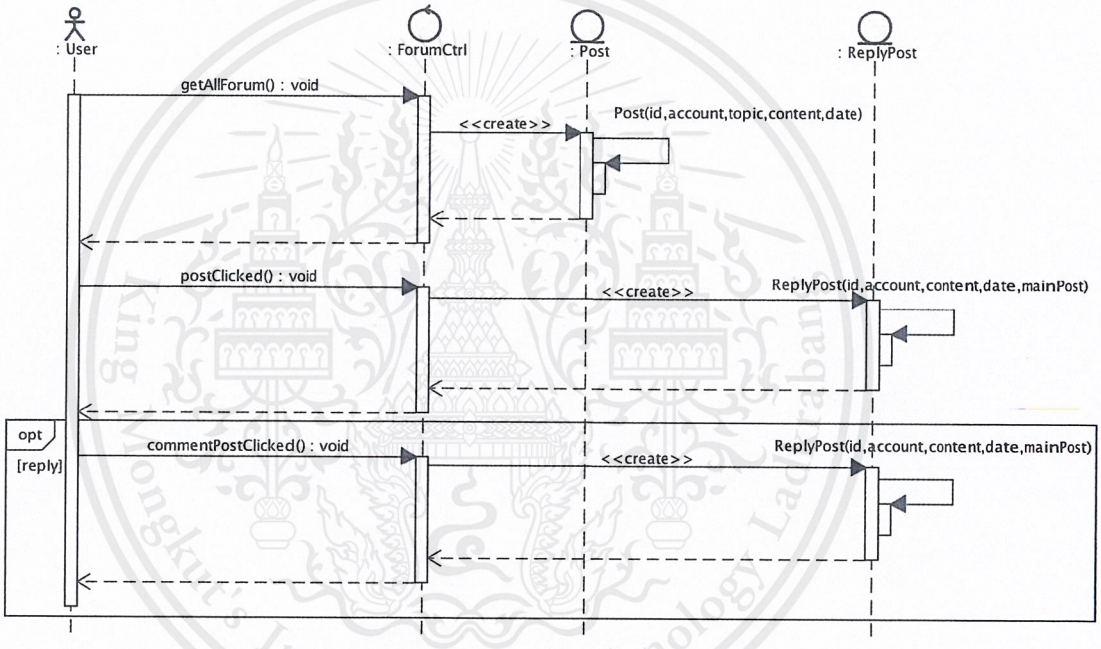


Figure 4.59 Reply Post Sequence Diagram

4.6.2.14 Add Favorite Post

Adding a post to my favorite uses two classes for implementation which are `Post` and `MyFavForumCtrl` as shown in Figure 4.56 and Figure 4.60.

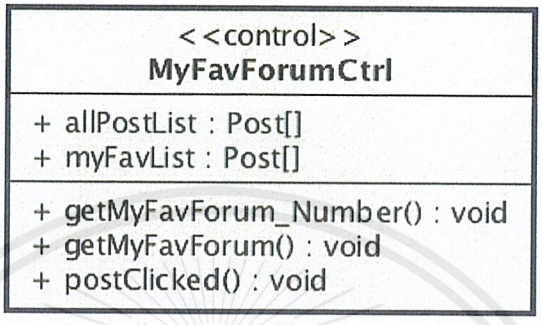


Figure 4.60 `MyFavForumCtrl` Class Diagram

After users get into a post page, they can add the topic in their favourite list. The process is simplest, just clicked at favourite button. The application will save the post into favourite list immediately. Users can open the page with their favourite posts also.

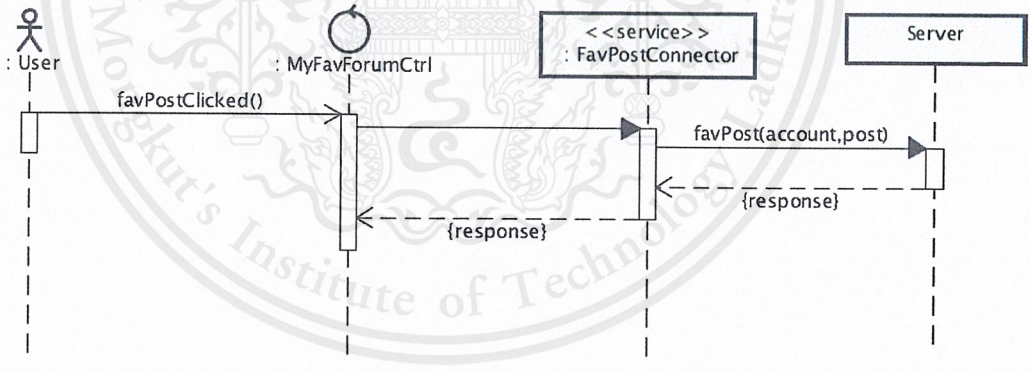
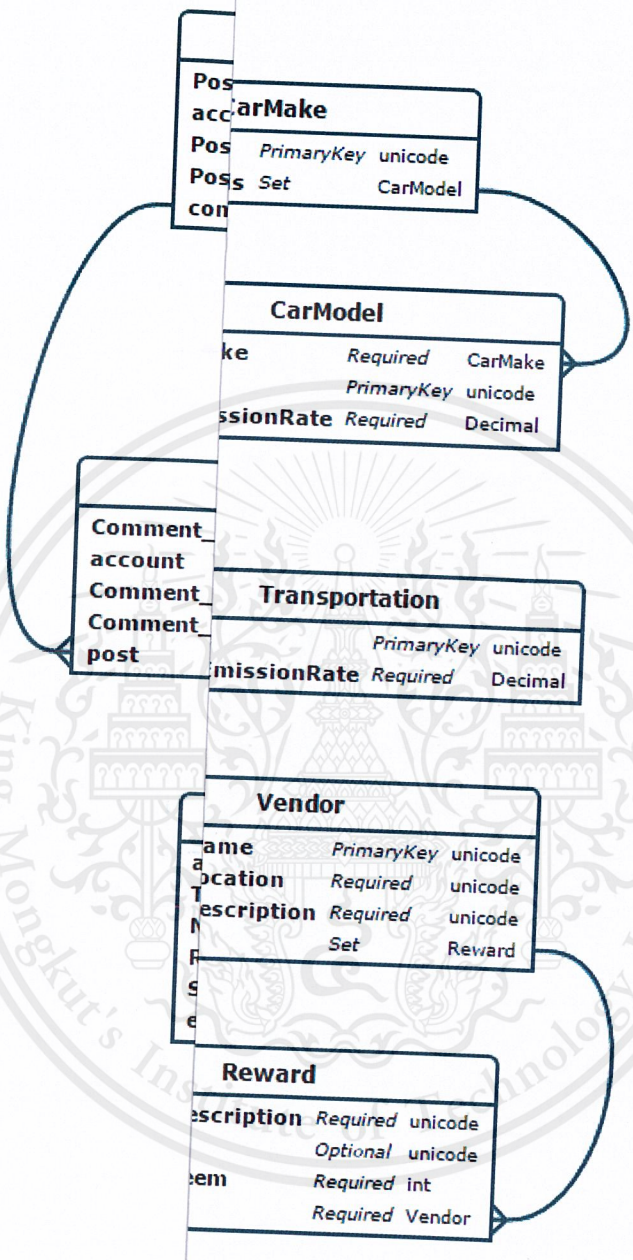


Figure 4.61 My Favorite Forum Sequence Diagram



- **Account**

  `Account` table is a table which keeps all information of a user.

  - `AcctName` field is a primary key of the table. It is using for keep account name.
  - `Email` field uses for keeping email of each account.
  - `Password` field uses for keeping password of each account.
  - `Token` field uses for keeping token of each account.
  - `TotalCO2` field uses for keeping all the CO_2 that each user emitted.
  - `Logindate` field uses for keeping user login date.

- **Friend**

  `Friend` table is a table which keeps friend information of each user.

  - `account` field is a foreign key between this table and `Account` table.
  - `FriendAccName` field uses for keeping friend list of each account.

- **Place**

  `Place` table is a table which keeps place data of each user.

  - `account` field is a foreign key between this table and `Account` table.
  - `Type` field uses for keeping type of each place liked House, Condo.
  - `Name` field uses for keeping name of each place.
  - `Resident` field uses for keeping number of resident who live in each place.
  - `Status` field uses for keeping status of each place (Active,Inactive).

- **Electricity**

  `Electricity` table is a table which keeps electricity used data of each user.

  - `place` field is a foreign key between this table and `Place` table.
  - `ElectricityUsed` field uses for keeping electricity data that user used.
  - `Month` field uses for keeping electricity month.
  - `EmitCO2` field uses for keeping CO_2 that user emit from electricity used.

- **Redeem**

`Redeem` table is a table which keeps redeem data of each user.

- `account` field is a foreign key between this table and `Account` table.
- `DateRedeem` field uses for keeping date that user redeemed reward.
- `VendorName` field uses for keeping name of vendor.
- `RewardDescription` field uses for keeping description of each reward.
- `PointRedeem` field uses for keeping point data of each reward.

- **Car**

`Car` table is a table which keeps car data of each user.

- `account` field is a foreign key between this table and `Account` table.
- `Maker` field uses for keeping maker name of each car.
- `Model` field uses for keeping model name of each car.
- `CO2EmissionRate` field uses for keeping rate of CO_2 emission of each car.
- `Status` field uses for keeping status if each car (Active,Inactive).

- **AccountTransportation**

`AccountTransportation` table is a table which keeps transportation data of each user.

- `account` field is a foreign key between this table and `Account` table.
- `TravelDistance` field uses for keeping distance data that user travelled.
- `Type` field uses for keeping vehicle type that user used.
- `Date` field uses for keeping date that user travelled.
- `EmitCO2` field uses for keeping CO_2 that user emitted.

- **AccountAchievement**

`AccountAchievement` table is a table which keeps achievement information of each account.

- `account` field is a foreign key between this table and `Account` table.
- `AchievementId` field uses for keeping id of each achievement.
- `Status` field uses for keeping status of each achievement.
- `Progress` field uses for keeping progress of each achievement.

- **AccountDailyMission**

  `AccountDailyMission` table is a table which keeps daily mission data of each user.

  - `account` field is a foreign key between this table and `Account` table.
  - `Date` field uses for keeping date of each mission.
  - `MissionId` field uses for keeping id of each mission.
  - `Status` field uses for keeping status of each mission.

- **CarMake**

  `CarMake` table is a table which keeps car maker data.

  - `Maker` field is a primary key of the table. This field uses for keeping name of each car maker.

- **CarModel**

  `CarModel` table is a table which keeps data of each car.

  - `car_make` is a foreign key between this table and `CarMaker` table.
  - `Model` field uses for keeping model name of each car.
  - `CO2EmissionRate` field uses for keeping the rate of emit CO_2 of each car.

- **Transportation**

  `Transportation` table is a table which keeps other vehicle data.

  - `Type` field is a primary key of the table. This field uses for keeping type of each vehicle.
  - `CO2EmissionRate` field uses for keeping the rate of emitted CO_2 in a vehicle.

- **Vendor**

  `Vendor` table is a table which keeps data of each vendor.

  - `VendorName` field is a primary key of the table. This field uses for keeping name of each vendor.
  - `VendorLocation` field uses for keeping location of each vendor.
  - `VendorDescription` field uses for keeping description of each vendor.

- **Reward**

  `Reward` table is a table which keeps reward data of each vendor.

  - `vendor` is a foreign key between this table and `Vendor` table.
  - `RewardDescription` field uses for keeping description of each reward.
  - `MoreDes` field uses for keeping more specific description of each reward.
  - `PointRedeem` field uses for keeping the number of points that used to redeem each reward.

- **Post**

  `Post` table is a table which keeps post data.

  - `Post_Id` field is primary key of the table. This field uses for keep id of each post.
  - `account` field is a foreign key between this table and Account table.
  - `Post_Content` field uses for keeping content of each post.
  - `Post_Date` field uses for keeping the date that post has been posted.

- **Comment**

  `Comment` table is a table which keeps comment data.

  - `Comment_Id` field is primary key of the table. This field uses for keep id of each comment.
  - `account` field is a foreign key between this table and Account table.
  - `post` field is a foreign key between this table and Post table.
  - `Comment_Content` field uses for keeping content of each comment.
  - `Comment_Date` field uses for keeping the date that comment has been commented.

4.8 User Interface Design

Before start using the application, a user is required to sign in to the application either with a local account or Facebook account as shown in Figure 4.63. Otherwise, the user has to create a local account first as shown in Figure 4.64. Additionally, the user can retrieve the password by touching on forget password button. A new generated password will be sent to the registered Email address. Also the password can be changed after signing in again with the new password.

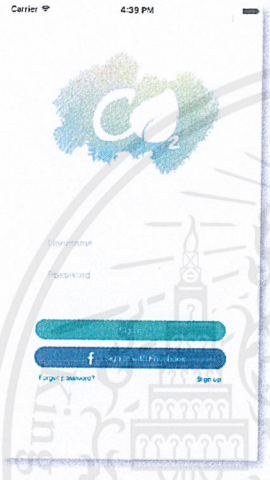


Figure 4.63 My Carbon Sign In Page

Figure 4.64 My Carbon Sign Up Page

For the first time using users, the application will briefly introduce main functions as slider pages which are tracking systems, leaderboard, summary graphs, rewards and forum as presented in Figure 4.65, Figure 4.66, Figure 4.67, Figure 4.68, Figure 4.69, Figure 4.70 below.

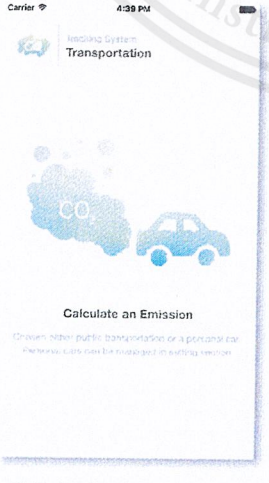


Figure 4.65 My Carbon Transportation Guide

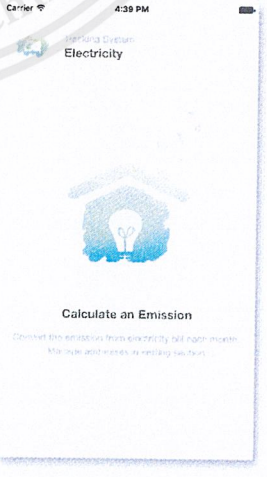


Figure 4.66 My Carbon Electricity Guide

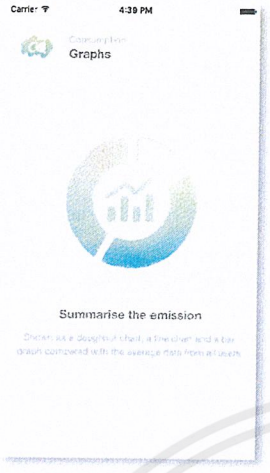


Figure 4.67 My Carbon Summary Graphs Guide

Figure 4.68 My Carbon Leaderboard Guide

Figure 4.69 My Carbon Reward Guide

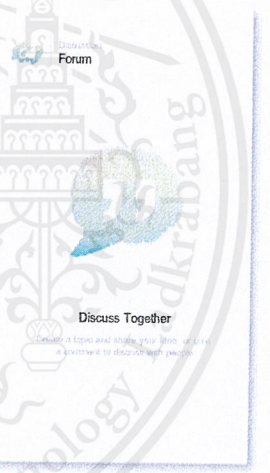


Figure 4.70 My Carbon Forum Guide

If the account has already signed in, the process will be skipped and directly lunched a main page of the application. The main page shows the summary of CO_2 emission from transportation each day presented as doughnut chart in Figure 4.71, also provides daily mission that user can complete and tracking system buttons of both transportation and electricity usage. User can choose a way to travel and the application will show a map while traveling as Figure 4.72.

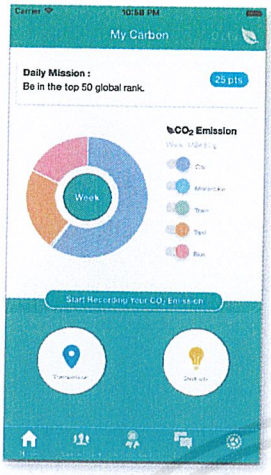


Figure 4.71 My Carbon Main Page

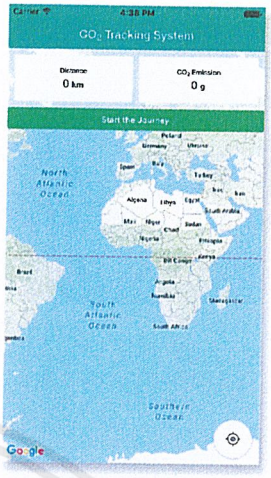


Figure 4.72 My Carbon Traveling Map

Another way to track CO_2 emission is from electricity usage. User will see all the addresses which are assigned in the setting mode of the application. Every month, user needs to input the amount of electricity usage for each addresses in kilowatt unit. The system will show the information as a bar graph in Figure 4.74.

Figure 4.73 My Carbon Electricity Tracking

Figure 4.74 My Carbon Electricity Record

In Figure 4.75 shows all carbon dioxide consumption including from transportation, electricity and the summary of both tracking shown as line graph compare with average data from other users. The CO_2 consumption will be used to rank in leaderboard with other users shown as Figure 4.76. Also, user can look up for friends by searching users' ID.

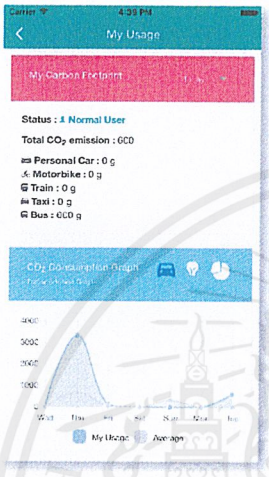


Figure 4.75 My Carbon My Usage page

Figure 4.76 My Carbon Leaderboard

User gets points from tracking systems and complete daily missions. Besides, user can also get point by complete achievements in Figure 4.77. All points that user got can be used to redeem a reward in Figure 4.78.

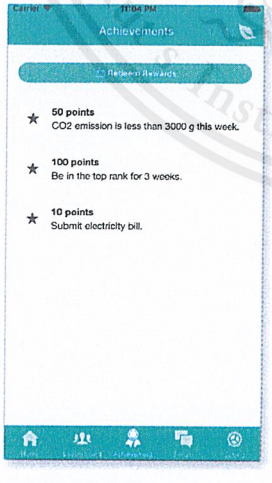


Figure 4.77 My Carbon Achievements

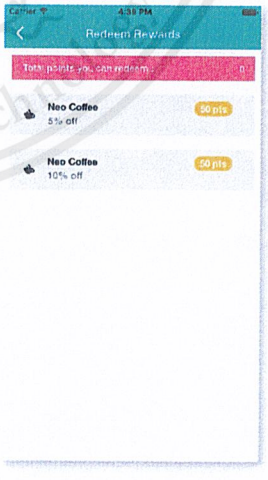


Figure 4.78 My Carbon Rewards

In setting page, there are three things to be set which are to edit account, manage personal addresses and manage personal cars in tracking system. User is able to change the Email address and password in Figure 4.80.

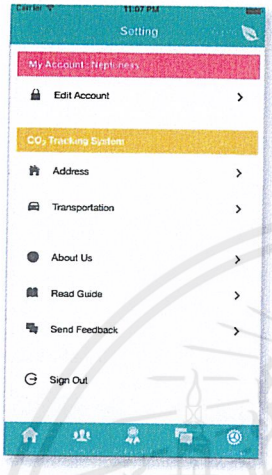


Figure 4.79 My Carbon Setting Page

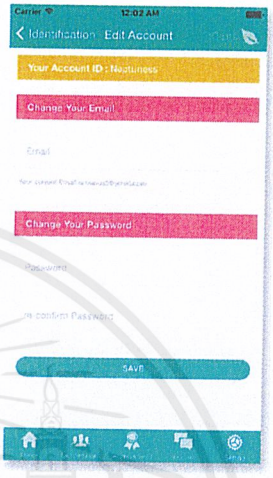


Figure 4.80 My Carbon Account Setting

User needs to add at least one address in Figure 4.81. Each addresses need to be input type, name and number of people living there. However, it can be edit the information or delete it. Same as Figure 4.82, user has to add at least one vehicle in order to start traveling.

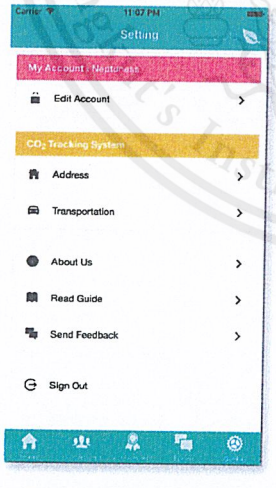


Figure 4.81 My Carbon Address Setting

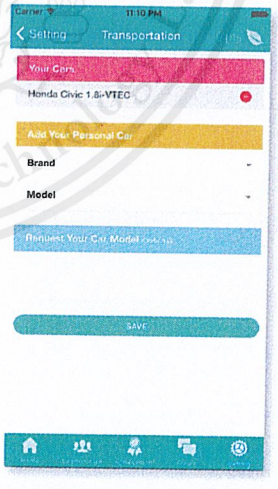


Figure 4.82 My Carbon Transportation Setting

This application is also provided a feedback feature in order to get some feedback and suggestion directly from a user as shown in Figure 4.83.

Figure 4.83 My Carbon Feedback

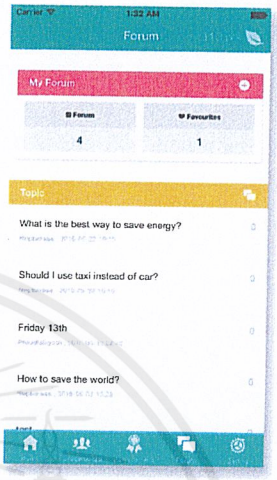


Figure 4.84 My Carbon Forum

In Figure 4.84 as shown above is another feature called Forum. This feature is for users to create relevant topic of this application to be shared and discussed together. Also, users can edit information or delete their own post anytime. Users can read and share their idea to any post as comments. Any post on this forum can be marked as favorite by tapping on a heart button in the top left corner of the post as shown in Figure 4.85. For their own post, post will show button to edit or remove the post as shown in Figure 4.86.

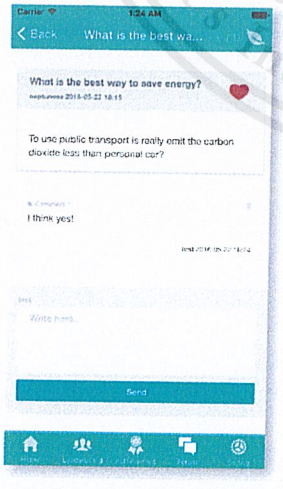


Figure 4.85 My Carbon Forum Post

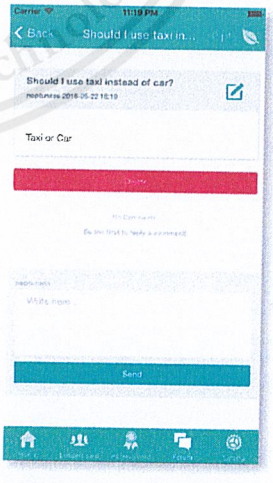


Figure 4.86 My Carbon Forum My Forum Post

The number of forum and favorite forum of each users will be presented on the Forum page as shown in Figure 4.63 previously, and it will be listed and shown after clicked at each section as in the Figure 4.66 and Figure 4.67.

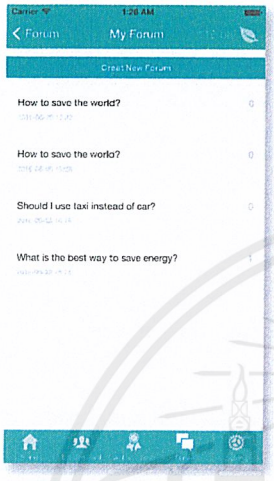


Figure 4.87 My Carbon My Forum

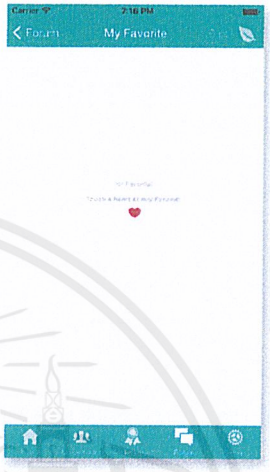


Figure 4.88 My Carbon My Favorite Forum

Chapter 5
Software Launch

5.1 Partnerships

Owing to the application's main objective in raising public awareness about climate change, it is important that the application be recognized and used by people in their daily life. This application has features which make it more interesting and fun such as daily mission, achievement and leaderboard. This application also provides some rewards to users by making a deal with some coffee shops nearby our college. Users can redeem points received by making achievements in the application for discounts at the coffee shops. In return, the coffee shops receive free advertisement from the application.

The coffee shops which became to be our partnership are Neo Cup Coffee Shop and Pavari Coffee Shop. Neo Cup Coffee Shop is located next to the KMITL dormitory.

5.2 Green Campus

This application was first officially released to be downloaded on store and advertised via several ways, for instance, posters, brochure and social network. Furthermore, the application was formally presented by the developers on the stage after opening the event. The audiences were invited to participate in the talk show and try to use the application at the time.

During the Green Campus Event, developers also set a booth to give more information included brochures about this application for people who were interested in. Moreover for people who downloaded the application during the event, a gift voucher for discount from our partnership was offered.

Figure 5.1 My Carbon Gift Voucher

Figure 5.2 My Carbon Poster

Figure 5.3 My Carbon Brochure

5.3 Website

In the long run, the application has not been popular and well-known as expected because most of people do not actually realized that the global warming or this climate change issue give an affect to people and animal's lives extraordinarily and also the application has not been announced or advertised enough to people. Therefore, a website represented about the information of this application and the climate change issue has been developed to broaden the knowledge to everybody efficiently and handily. The website is organized by Green Campus in form of a domain as *www.greenkmitl.com*. Hence, the website will be included all the information and softwares which are related to the organization.

Figure 5.4 Green Campus Website

Chapter 6

Conclusion and Suggestion

6.1 Conclusion

This application is based on hybrid mobile application which uses web technologies to access all devices capabilities. Its purpose is to raise awareness of people about this global impact by tracking their carbon footprint from their daily activities such as traveling. The application mainly track the amount of CO_2 emission from two sources which are from transportation and household electricity usage per capita. The data of all emissions will be presented as graphs for more understanding, and also compare the data with the average data of all users. Furthermore, this application provides some rewards to give back to all users in order to make the application more interesting.

The application has been published on both Play store and App store for the first version and intended to get some feedback or suggestion from the users either via the stores or directly through the application, so that it can be debugged and developed to a better version. Besides, the application was advertised and announced on the Green Campus event to persuade people and get more users, but there were not many people interested in this application because they did not realize of the climate changes and the world environment enough. Anyway, after officially releasing the application, we got both positive and negative feedbacks from the users. The compliment for this application was that it has a pretty good color and design. Also it was interesting because there were not so many people developed this kind of this application. On the other hand, some features did not work well on every different devices, for example, the interface on some pages were mess up on some devices or the user got confused and did not know what to communicate with the application. Those feedbacks made us aware of the different point of views between users and developers, and also realized the flaws of the application.

In summary, this application is successfully developed with all the features including tracking systems from daily activities and some features which connect people together and make it fun, but this application is not so successfully in persuading people to use this application and raise awareness of people about the climate changes on this global, so that there are not so many people using this application as we expected.

6.2 Suggestion

Even though the application covers all the requirement, there are still some part for improvement. For the tracking system in transportation, the location which is located by the GPS is not stable and compatible with the application, and also cannot run on background smoothly while traveling. For missions and achievements, there are less activities for users to achieve and complete. The application needs to use the Internet to access all the data between the application and server. This causes the application to be slow depended on the Internet. Lastly, the marketing of this application is low. It should be more productive and well-known for people to get to the objective of this application.

Reference

(1) USEPA (2015) Greenhouse Gas Emission | Climate Change [Online]
Available: (http://www3.epa.gov/climatechange/ghgemissions/gases.html)

(2) Union of Concerned Scientists (2015) The Cause of Global Warming [Online]
Available: (http://www.climatehotmap.org/about/global-warming-causes.html)

(3) USEPA (2015) Carbon Dioxide Emission | Climate Change [Online]
Available: (http://www3.epa.gov/climatechange/ghgemissions/gases/co2.html)

(4) Warwick University Carbon Footprint Project Group (2015) Effect of CO_2 [Online]
Available: (http://www.carboncalculator.co.uk/effects.php)

(5) Energy Policy and Planning Office (2015, September 17) CO_2 Emission [Online]
Available: (http://www.eppo.go.th/info/emission/2015_Q2.pdf)

(6) CO_2nnect (2015) Calculation of Emission [Online]
Available: (http://www.co2nnect.org/help_sheets/?op_id=602&opt_id=98)