**Chiang Mai Red Taxi Service Assistant**

Software Design Document

By

**Miss. KanitteeHongron 542115003**

**Miss. PimchittraSukkasem 542115042**

Department of Software Engineering

College of Arts, Media and Technology

Chiang Mai University

Project Advisor  
  
**Ms. Pattama Longkanee**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Document Name** | **Version** | **Status** | **Date** | **Viewable** | **Reviewer** | **Responsible** |
| **RoseBeam\_Software Design \_18april2014** | CM Red Taxi Service Assistant\_SDD\_0.1 | Reviewed | 18-04-2014 | PM, SA, DEV, QA, Tester, Advisor | Kanittee,  Pimchittra | Kanittee,  Pimchittra |
| **RoseBeam\_Software Design \_4July2014** | CM Red Taxi Service Assistant\_SDD\_0.2 | Reviewed | 04-07-2014 | PM, SA, DEV, QA, Tester, Advisor | Kanittee,  Pimchittra | Kanittee,  Pimchittra |
| **RoseBeam\_Software Design \_15July2014\_** | CM Red Taxi Service Assistant\_SDD \_0.3 | Reviewed | 15-07-2014 | PM, SA, DEV, QA, Tester, Advisor | Kanittee,  Pimchittra | Kanittee,  Pimchittra |
| **RoseBeam\_Software Design \_24July2014\_** | CM Red Taxi Service Assistant\_ SDD \_0.4 | Reviewed | 24-07-2014 | PM, SA, DEV, QA, Tester, Advisor | Kanittee,  Pimchittra | Kanittee,  Pimchittra |
| **RoseBeam\_Software Design \_31July2014\_** | CM Red Taxi Service Assistant\_ SDD \_1.0 | Reviewed | 31-07-2014 | PM, SA, DEV, QA, Tester, Advisor | Kanittee,  Pimchittra | Kanittee,  Pimchittra |
| **RoseBeam\_Software Design \_27/08/2014** | CM Red Taxi Service Assistant\_ SDD \_1.1 | Reviewed | 31-07-2014 | PM, SA, DEV, QA, Tester, Advisor | Kanittee,  Pimchittra | Kanittee,  Pimchittra |

**Document History**

Table of Contents  
  
Chapter 1: Introduction6  
1.1 Purpose6  
1.2 Software Scope6  
1.3 Technology Review7  
 1.3.1 Android7  
 1.4.1Google Maps Android API8  
 1.5.1 Java9  
 1.6.1 JSON9  
 1.7.1 MySQL10  
 1.8.1 XAMPP10  
Chapter 2: System Architecture11

Chapter 3: Detail Design12  
3.1 Database Design12  
3.2 Class Diagram Design26  
 3.2.1 Driver’s Class Diagram Design 26  
 3.2.2 Passenger’s Class Diagram Design 27

3.3 Class Diagram Description 28  
 3.3.1 Driver’s Class Diagram 28  
 3.3.2 Passenger’s Class Diagram 32

3.4 Sequence Diagram 38

SD01 Passenger can register to the system38

SD02 Passenger can login to the system39 SD03 Passenger can logout of the system40 SD04 Passenger can search for taxi41

SD05 Passenger can send request for taxi,   
 Passenger can chat with driver 42 SD06 Driver can register into the system43 SD07 Driver can log in to the system 44 SD08 Driver can logout from the system 45 SD09 Driver can update driving information 46 SD10 Driver can respond to passenger’s request,   
 Driver can chat with passenger47 SD11 Administrator can login to the Administration system48 SD12 Administrator can logout49 SD13 Administrator can add destinations50 SD14 Administrator can browse the destination51 SD15 Administrator can edit destinations52 SD16 Administrator can delete destinations 53 SD17 Administrator can search destinations 54 SD18 Administrator can clear data 55

3.5 User Interface Design 57

3.5.1 User Interface for Passenger – Mobile Application 59

UI-01Passenger can register to the system 59  
UI-02 Passenger can login to the system 59

UI-03 Passenger can logout of the system 60

UI-04 Passenger can search for taxi 60  
UI-05 Passenger can send request for taxi 56  
UI-06 Passenger can chat with driver 56  
UI-07 Driver can register into the system57

3.5.2 User Interface for Driver – Mobile Application

UI-08Driver can log in to the system 57

UI-09Driver can logout from the system 58

UI-10Driver can update driving information 58

UI-11 Driver can respond to passenger’s request 59  
UI-12 Driver can chat with passenger 59

3.5.3 User Interface for Administrator – Web Application 59

UI-13 Administrator can login to the Administration system60

UI-14 Administrator can logout 60  
UI-15 Administrator can add destinations61

UI-16 Administrator can browse the destination61  
UI-17 Administrator can edit destinations 62  
UI-18 Administrator can delete destinations 62  
UI-19 Administrator can search destinations 63  
UI-20 Administrator can clear data63

**Chapter One | Introduction**

**1.1 Purpose**

This software design document (SDD) describes the system design and architecture of the Chiang Mai Red Taxi Service Assistant. The Chiang Mai Red Taxi Service Assistant is designed to create a web service that assists passengers to catch a red taxi and aims to reduce the transportation issues in Chiang Mai.

**1.2 Software Scope**

This document describes the implementation details of theChiang Mai Red Taxi Service Assistant. The software will consist of two major functions – the first function is to help red taxi driversmatch the destinations of passengers according to various conditions (eg: location and travel) and the second is for passengers to easily book the service of a red taxi from any location matching the conditions of the driver and also specifying the number of passengers.

First, the Driver reports his area and travel conditions to the server. Second, the Passenger retrieves his current location from Google Maps, and then the Passenger sends the location and travel conditions (e.g. the number of passengers and the destination) to the server. Thereafter, the web service finds taxi driverslocated near the Passenger. The web service will process the matching conditions and send the information of the available red taxis back to the Passenger. Afterwhich, the Passenger has the option to send a request to one of the available taxis. Finally, the Driver that receives the request can either accept or decline the request to the Passenger through the web service.

This document explains how each feature of the application is designedusing sequence diagrams, class diagrams, and also provides an overview of the system.

**1.3 Technology Review**

**1.3.1Android**



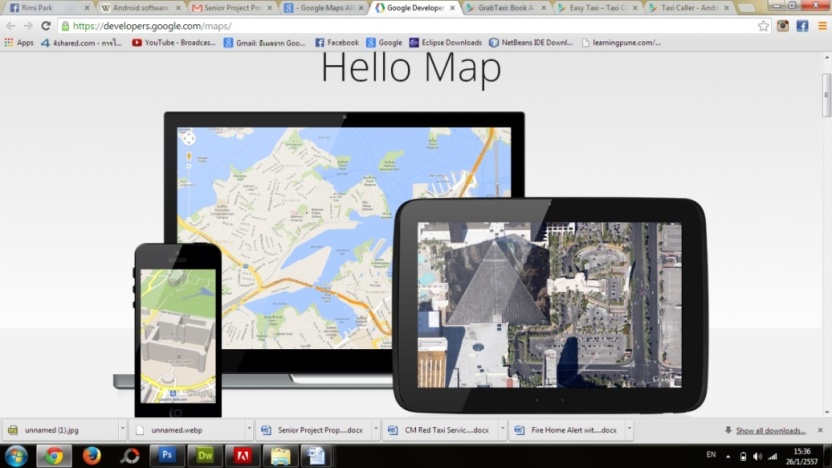
*Figure 1: Android Logo.*

**1.3.1.1 Technology Description**

Android is an open-source, Linux-based operating system (OS) for smartphones that Google developed. Android's user interface is purely using direct manipulation, which uses touch inputs that correlate to the user’s movement or actions, like tapping, pinching, reverse pinching and swiping to control objects on the screen. Response to the user's input is designed to be instantaneous, which, provides a smooth touch interface. The internal hardware supported by the Android OS includes proximity sensors, gyroscopes and accelerometers. The Android OS loads to the home screen typically made up of widgets and app icons. The widgets app displays content in real-time like emails or the weather outlook, whereas the app icons initiate execution of the applications linked to them.

**1.3.1.2Selection of this tool**The Android platform is selected because it is mostly built in Java, which is a popular programming language that we are also familiar with, and it is highly versatile, which allows developers to design applications for multipurpose. The Android OS would provide us with an open development platform, which allows us to enjoy the freedom of utilizing tools from third-party sources for the implemention of our system. This would allow us to explore the various functions of their applications, which may give us insight into how to design our own features. The Android OS also provides us with an exceptional testing platform and systematically arranged testing tools.

**1.4.1 Google Maps Android API**



*Figure 2: Image of Google Maps on different devices*.

**1.4.1.1 Technology Description**  
Google Maps APIs lets developers embed Google Maps on web pages using a JavaScript or Flash interface. The Google Maps Android API gives developers the freedom to customize their own maps. The API manages the entry to the servers of Google Maps, handles the display of maps, the response to gestures in the map and data downloads. Developers are able to adjust the user’s map view of an area. They are also able to utilize the API calls to add in polygons, overlays and markers to a simple map and these features allow users to interact with the application.

**1.4.1.2Selection of this tool**

The Google Maps Android API was selected because it allows us to include maps from the database of Google Maps into our own application. We are also able to customize our own maps. The maps would load faster with a lower bandwidth used and it has rotate, zoom gesture and intuitive tilt controls. The Google Maps Android API would allow us to customize popular destination spots that users might tell the drivers to go.

**1.5.1 Java**



*Figure 3: Image of Java logo.*

**1.5.1.1 Technology Description**Java is a simplified object-oriented language, which is used, to avoid language features that could be a cause of common programming issues. It is somewhat similar to C++ but less complicated. Usually most Java codes are able to function on most computers as Java runtime environments and Java interpreters are used in many operating systems like, Windows, Macintosh OS and UNIX.

**1.5.1.2Selection of this tool**

Java was selected because of its popular, open source ability and it is extremely user-friendly and there are many sources available online. We would be able to refer to the different material and sources available online. It can be used on various softwares such as Notepad, NetBeans and Eclipse. Java is something that we are familiar with and thus, it would be easier for us to manage.

**1.6.1 JSON  
1.6.1.1 Technology Description**JavaScript Object Notion (JSON) makes use of human-readable script in an open standard format, and it transfers data objects comprising of key-value pairs. JSON acts as a substitute to XML, and used mainly to transfer data between a web application and a server. JSON ,originally created from JavaScript uses two main data structures, value/name pairs known as ‘objects’ and ordered lists known as ‘arrays’.

**1.6.1.2 Selection of this tool**

The JSON format is language-independent, and the data structure uses both objects and arrays. These structures widely recognized by almost all developers are also more or less supported by almost all kinds of programming languages. JSON would make it the best possible format for the exchange of data online.

**1.7.1 MySQL  
1.7.1.1 Technology Description**

MySQL is a database system that operates on a server and is suitable for both large and small applications. MySQL operates on various platforms, and it is reliable, user-friendly and fast. The data produced in MySQL kept in tables consist of rows and columns. MySQL can reduce down in size to support embedded database applications.

**1.7.1.2 Selection of this tool**

The MySQL database is an open source tool that is free for all to use and this makes it more cost efficient for developers to use. The main advantage of MySQL is its ‘cross-platform operability’, which allows it to be installed in a variety of platforms such as Windows, Solaris, Linux etc. MySQL databases are extremely secure because all the stored passwords encrypted or converted into code prevent unapproved admittance to the database.

**1.8.1 XAMPP**

  
*Figure 4: Image of Xampp logo.*

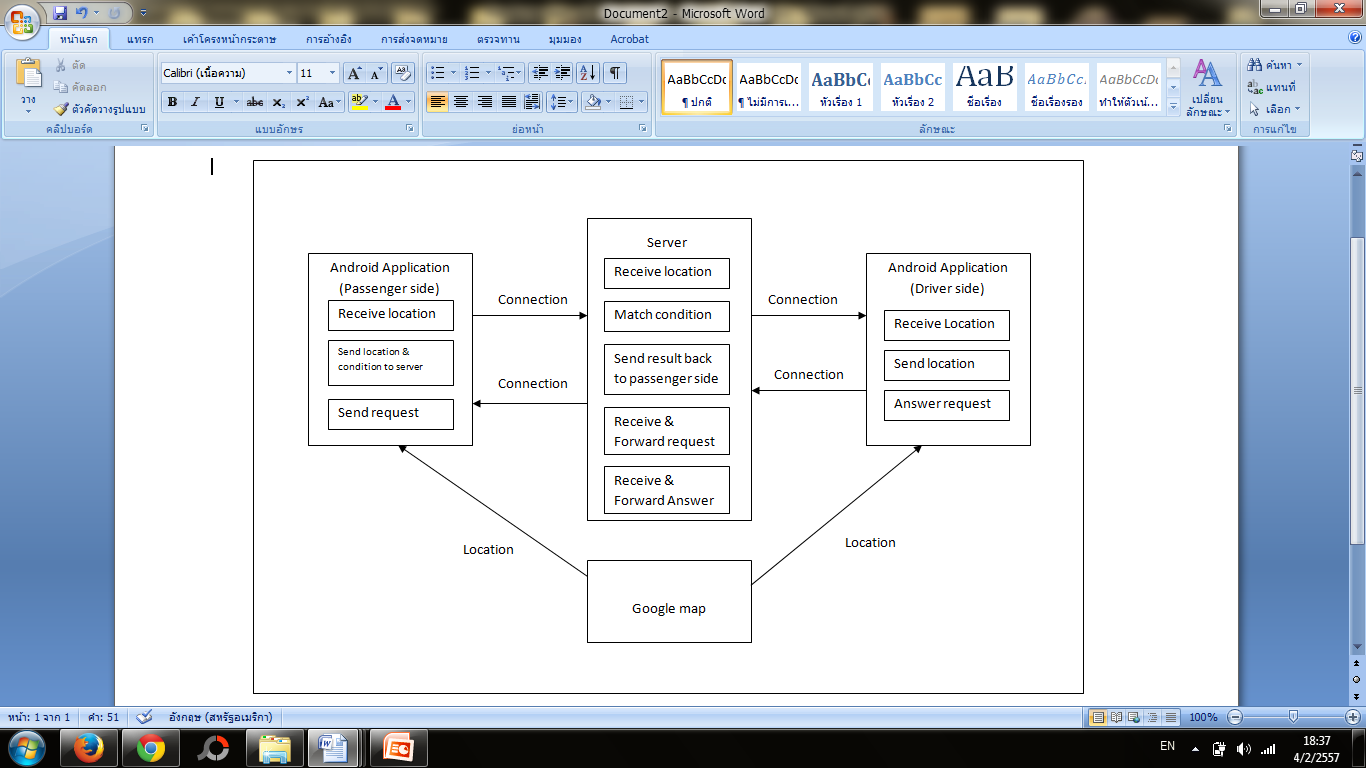
**1.8.1.1 Technology Description**

XAMPP is a web server package that is open-source, easy to install Apache distribution, which can function on a number of platforms. The name of this webserver XAMPP is an acronym which can be broken down into X which represents a “cross” platform, A stands for Apache HTTP server, M stands for MySQL, P represents PHP, and P stands for Perl. This webserver was specifically designed to assist programmers, designers and developers, in reviewing and checking their work on the computer while offline.

**1.8.1.2 Selection of this tool**

XAMPP was selected because of its ability to perform on multiple platforms which includes Mac OS X, Windows, Solaris and Linux. It is a freeware which does not require users to configure anything when setting up the system. The advantage of this application is that we would be able to edit, review and check our work even if we are not connected to the internet.

**Chapter Two | System Architecture**

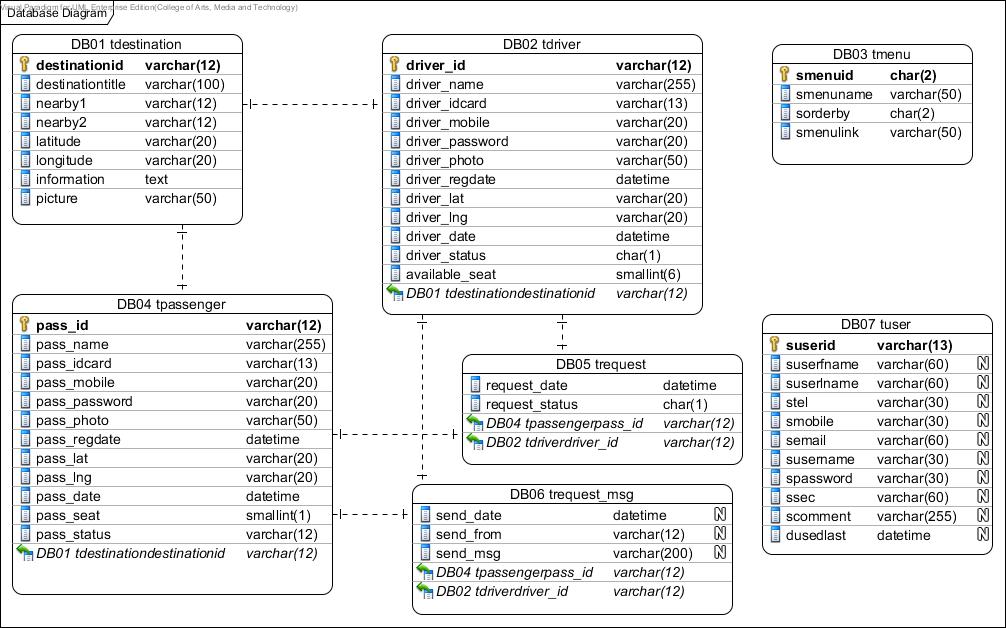
****

*Figure 5: System Architecture.*

The figure shows The Chiang Mai Red Taxi Sevice Assistant architecture which consist of 3 components. Andriod application side (Passenger) send the request to server by seat,destination and location. Android application side (Driver) will get request and respond to Passenger side via on server. The system will find the driver around Passenger’s lacate at default circle by use Google map and send the list of Driver back to Passenger.

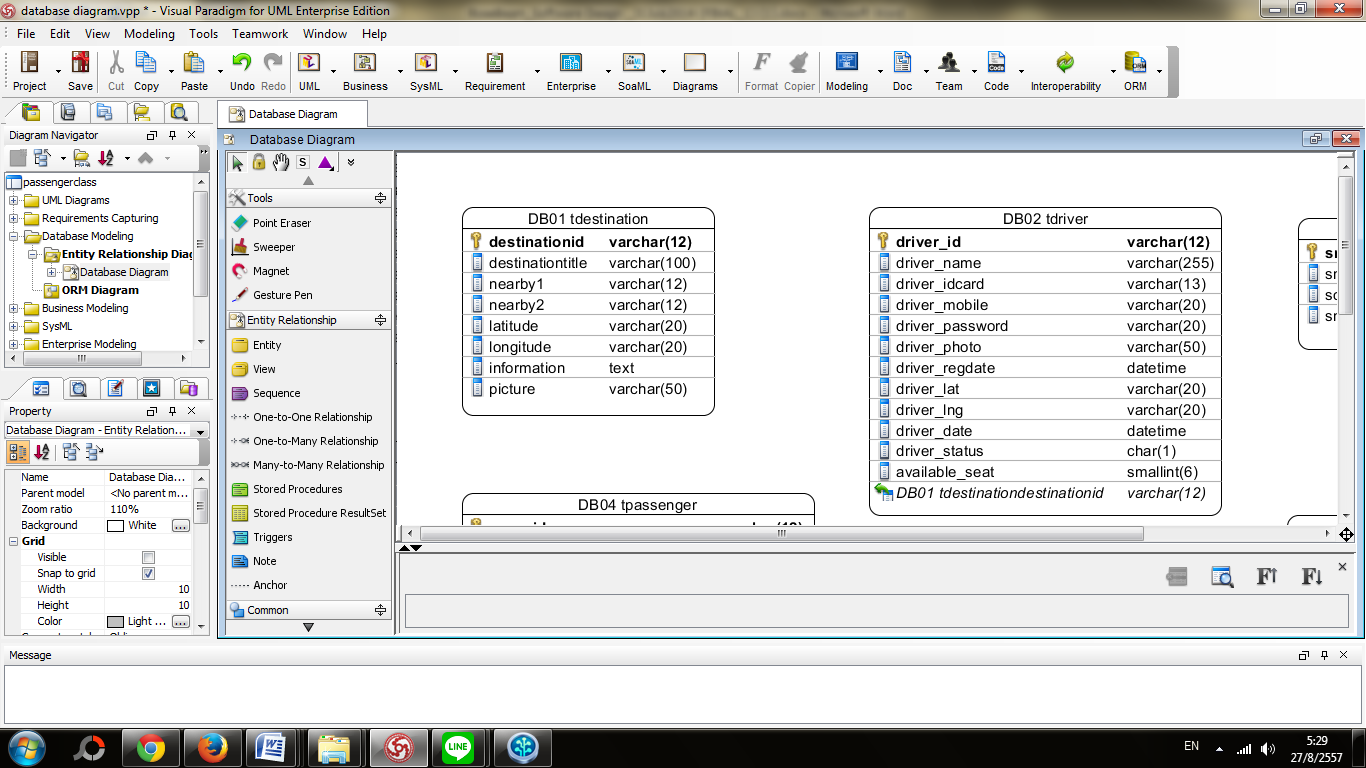
**Chapter Three | Detail Design**

**3.1 Database Design**

****

*Figure 6: Database design.*

**DB-01 tdestination**

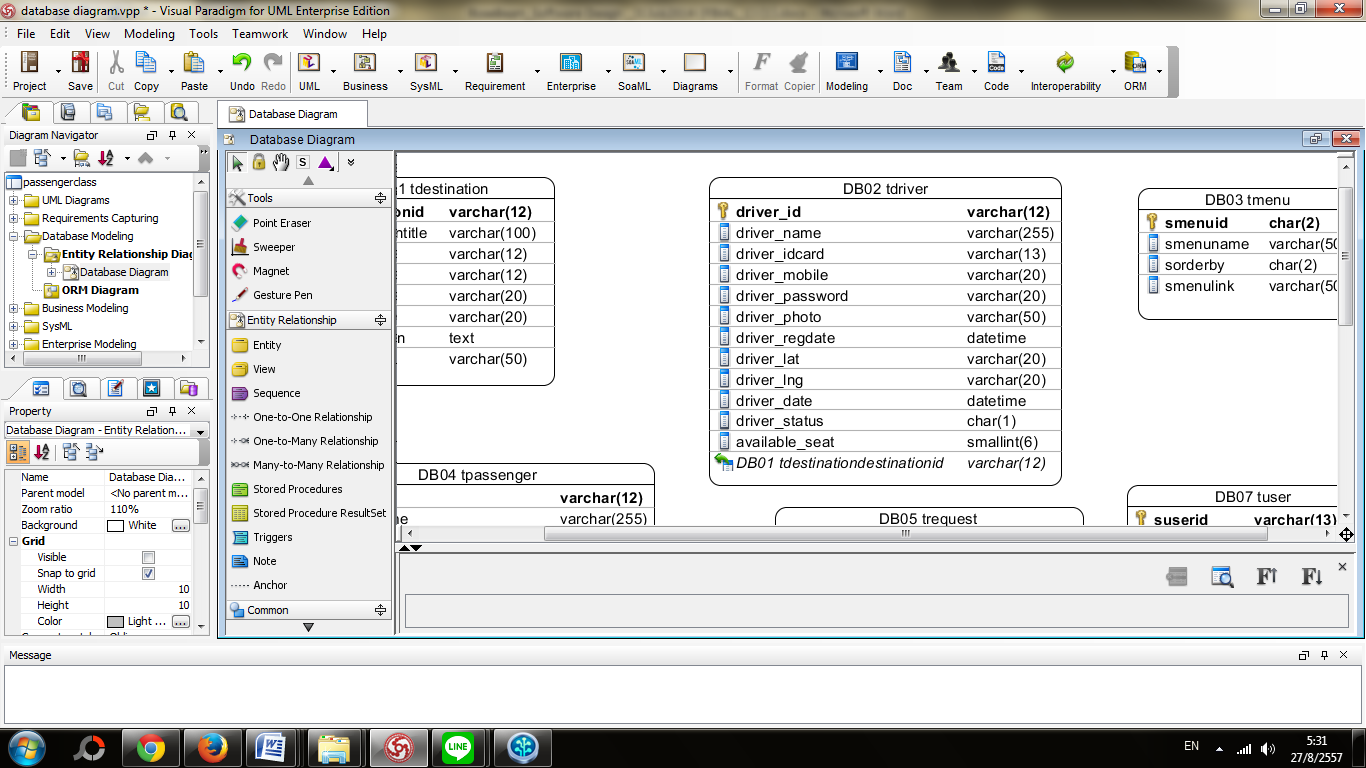
****

*Figure7-1: Database design DB01 tdestination*

*-store the infomation of the destination that will be provided in the system for maching condion of Driver-Passenger.*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Field Name** | **Data type** | **Length** | **PK/FK** | **Null able** | **Description** |
| **destinationid** | varchar | 12 | PK | Not Null | This variable is declared as an identification number of each destination.  (e.g. 140510043030) |
| **destinationtitle** | varchar | 100 | - | Not Null | This variable is for the title of the destination.  (e.g. Thaphae Gate) |
| **nearby1** | varchar | 12 | - | Not Null | This variable is for represent the destination ID of one of two places that located near by the destination.  (e.g. 140519040517) |
| **nearby2** | varchar | 12 | - | Not Null | This variable is used to represent the destination ID of one of two places that located near by the destination.  (e.g. 14051905032) |
| **latitude** | varchar | 20 | - | Not Null | This variable is used to describe the latitude value of a destination.  (e.g. 18.78801743704621) |
| **longtitude** | varchar | 20 | - | Not Null | This variable is used to describe the longitude value of a destination. (e.g. 98.9937686920166) |
| **information** | text | - | - | Not Null | This variable is used to describe the information about the destination. |
| **picture** | varchar | 50 | - | Not Null | This variable is used to name the image file. Numeric digits and alphabets can be used.  (e.g. 1309799402.jpg) |

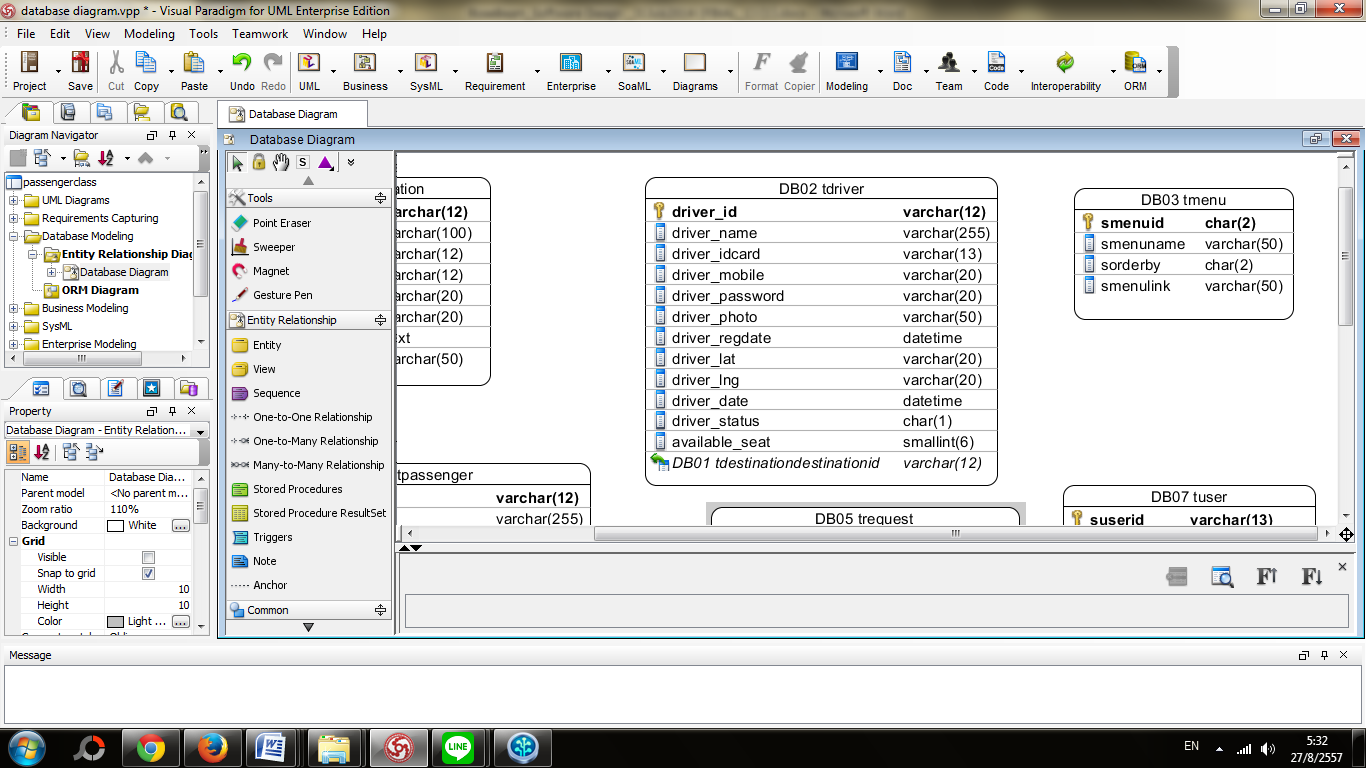
**DB-02 tdriver**

****

*Figure7-2: Database design DB2 trequest.*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Field Name** | **Data type** | **Length** | **PK/FK** | **Null able** | **Description** |
| **driver\_id** | varchar | 12 | PK | Not Null | This variable is used as an identification number for the driver.  (e.g. 140425101020) |
| **driver\_name** | varchar | 255 | - | Not Null | This variable is used to identify the driver’s name.  (e.g. somchaijaidee) |
| **driver\_idcard** | varchar | 13 | - | Not Null | This variable is used to represent the driver’s national identification number.  (e.g. 4589666636669) |
| **driver\_mobile** | varchar | 20 | - | Not Null | This variable is used to represent the driver’s mobile phone number.  (e.g. 0859696696) |
| **driver\_password** | varchar | 20 | - | Not Null | This variable is used to represent the driver’s password.  (e.g. drv01) |
| **driver\_photo** | varchar | 50 | - | Not Null | This variable is used to display the image of the driver. |
| **driver\_regdate** | datetime | - | - | Not Null | This variable is used to show the driver’s date and time of registration.  (e.g. 2014-04-25 12:47:39) |
| **driver\_lat** | varchar | 20 | - | Not Null | This variable is used to describe the latitude value of the driver’s current location. (e.g. 18.78801743704621) |
| **driver\_lng** | varchar | 20 | - | Not Null | This variable is used to describe the longitude value of the driver’s current location.  (e.g. 98.9937686920166) |
| **driver\_date** | datetime | - | - | Not Null | This variable is used to represent the driver’s last online login date. (e.g. 2014-05-15 08:22:07) |
| **driver\_status** | char | 1 | - | - | This variable is used to show whether the driver is currently available or not available.  (e.g. 1) |
| **available\_seat** | smallint | 6 | - | - | This variable is used to indicate the number of seats available  (e.g. 5) |
| **destinationid** | varchar | 12 |  | FK | This is variable is used to show the driver’s last destination.  (e.g. cmu) |

**DB-03 tmenu**

****

*Figure7-3: Database design DB03 tmenu.*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Field Name** | **Data type** | **Length** | **PK/FK** | **Null able** | **Description** |
| **smenuid** | char | 2 | PK | Not Null | This variable represents the id for each menu.  (e.g. 01) |
| **smenuname** | varchar | 50 | - | Not Null | This variable represents the name for each menu.  (e.g. destinations) |
| **sorderby** | char | 2 |  | Not Null | This variable sorts the menu according to an order.  (e.g. 01) |
| **smenulink** | varchar | 50 | - | Not Null | This variable links the menu to various pages.  (e.g. placelst.php) |

**DB-04 tpassenger**

****

*Figure7-4: Database design DB04 tpassenger.*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Field Name** | **Data type** | **Length** | **PK/FK** | **Null able** | **Description** |
| **pass\_id** | varchar | 12 | PK | Not Null | This variable is used as an identification number for the passenger. (e.g. 140425101020) |
| **pass\_name** | varchar | 255 | - | Not Null | This variable is used to identify the passenger’s name. (e.g. RoseappleArmia) |
| **pass\_idcard** | varchar | 13 | - | Not Null | This variable is used to represent the passenger’s national identification number. (e.g. 4589666636669) |
| **pass\_mobile** | varchar | 20 | - | Not Null | This variable is used to represent the passenger’s mobile phone number. (e.g. 0804589659) |
| **pass\_password** | varchar | 20 | - | - | This variable is used to represent the passenger’s password. (e.g. drv01) |
| **pass\_photo** | varchar | 50 | - | - | This variable is used to display the image of the passenger. |
| **pass\_regdate** | datetime | - | - | - | This variable is used to show the passenger’s date and time of registration. (e.g. 2014-04-25 12:47:39) |
| **pass\_lat** | varchar | 20 | - | Not Null | This variable is used to describe the latitude value of the passenger’s current location. (e.g. 18.78801743704621) |
| **pass\_lng** | varchar | 20 | - | Not Null | This variable is used to describe the longitude value of the passenger’s current location. (e.g. 98.9937686920166) |
| **pass\_date** | datetime | - | - | Not Null | This variable is used to represent the passenger’s last online login date. (e.g. 2014-05-15 08:22:07) |
| **pass\_seat** | smallint | 1 | - | Not Null | This variable is used to represent the number of seats the passenger selects. (e.g. 1) |
| **destinationid** | Varchar | 6 | FK | - | This is variable is used to show the destination that the passenger has selected.  (e.g. 140609032237) |
| **pass\_status** | varchar | 12 | - | - | This variable is uses numerical values to show whether the passenger is currently online or offline. (e.g. 1) |

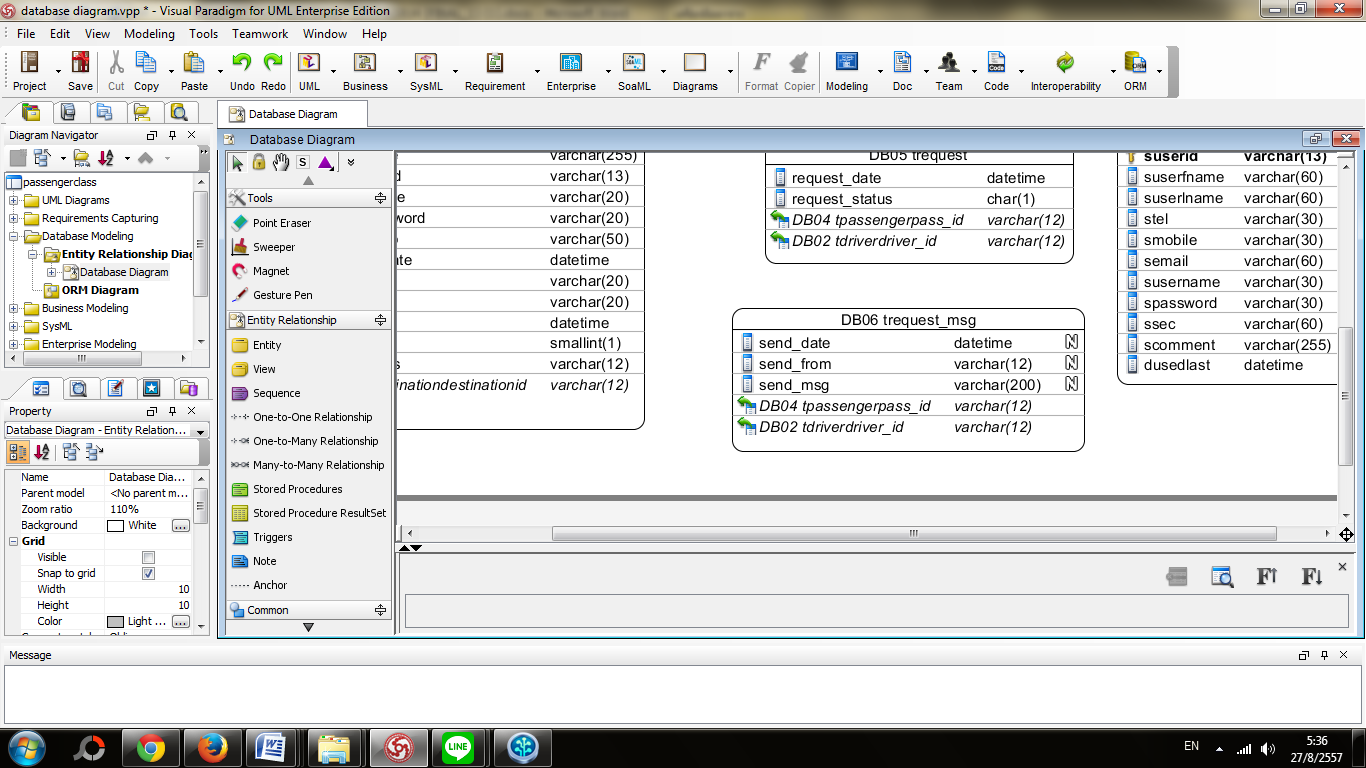
**DB-05**[**trequest**](http://theredtaxiservice.com:8880/domains/databases/phpMyAdmin/sql.php?server=1&db=theredta_db&table=tdriver_position&pos=0&token=a07dd9e2971e344c2200897200b6903a)

****

*Figure7-5: Database design DB05 trequest.*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Field Name** | **Data type** | **Length** | **PK/FK** | **Null able** | **Description** |
| **driver\_id** | varchar | 12 | FK | Not Null | This variable is used as an identification number for the driver.  (e.g. 140529051946) |
| **pass\_id** | varchar | 12 | FK | Not Null | This variable is used as an identification number for the passenger.  (e.g. 140602012236) |
| **request\_date** | datetime | - | - | Not Null | This variable is used to show the date and time of request.  (e.g. 2014-07-06 17:00:45) |
| **request\_status** | char | 1 | - | - | This variable uses numerical values to indicate the status of the request.   1=Request, 2=Accept, 3=Reject/Offline |

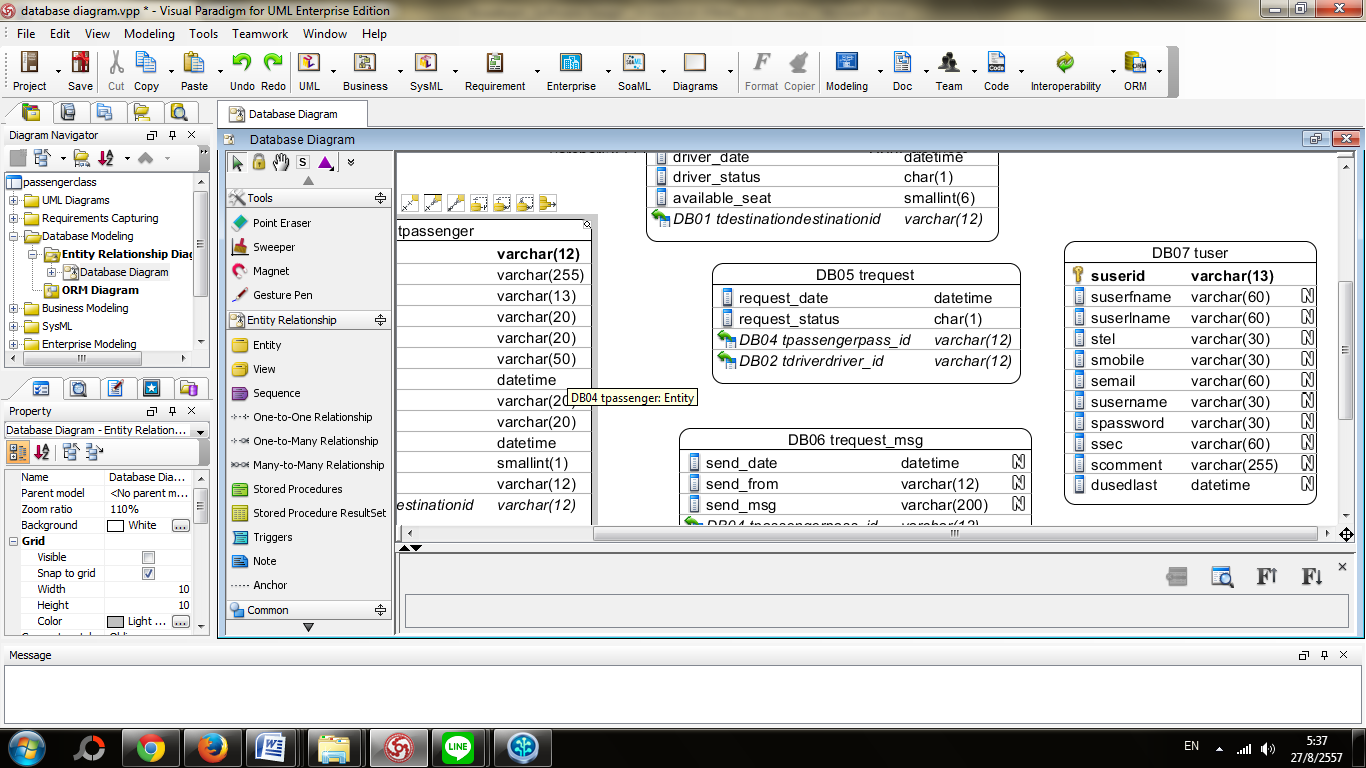
**DB-06 trequest\_msg**

****

*Figure7-6: Database design DB06trequest\_msg.*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Field Name** | **Data type** | **Length** | **PK/FK** | **Null able** | **Description** |
| **driver\_id** | varchar | 12 | PK | Not Null | This variable is used as an identification number for the driver. (e.g. 140529051946) |
| **pass\_id** | varchar | 12 | PK | Not Null | This variable is used as an identification number for the passenger. (e.g. 123456789456) |
| **send\_date** | datetime | - | - | Not Null | This variable is used to show the date and time of request. (e.g. 2014-07-06 22:17:12) |
| **send\_from** | varchar | 12 | - | Not Null | This variable uses numerical values to indicate the status of the request.   1=Request, 2=Accept, 3=Reject |
| **send\_msg** | Varchar | 200 |  | - | This variable is a chat message. |

**DB-07tuser**

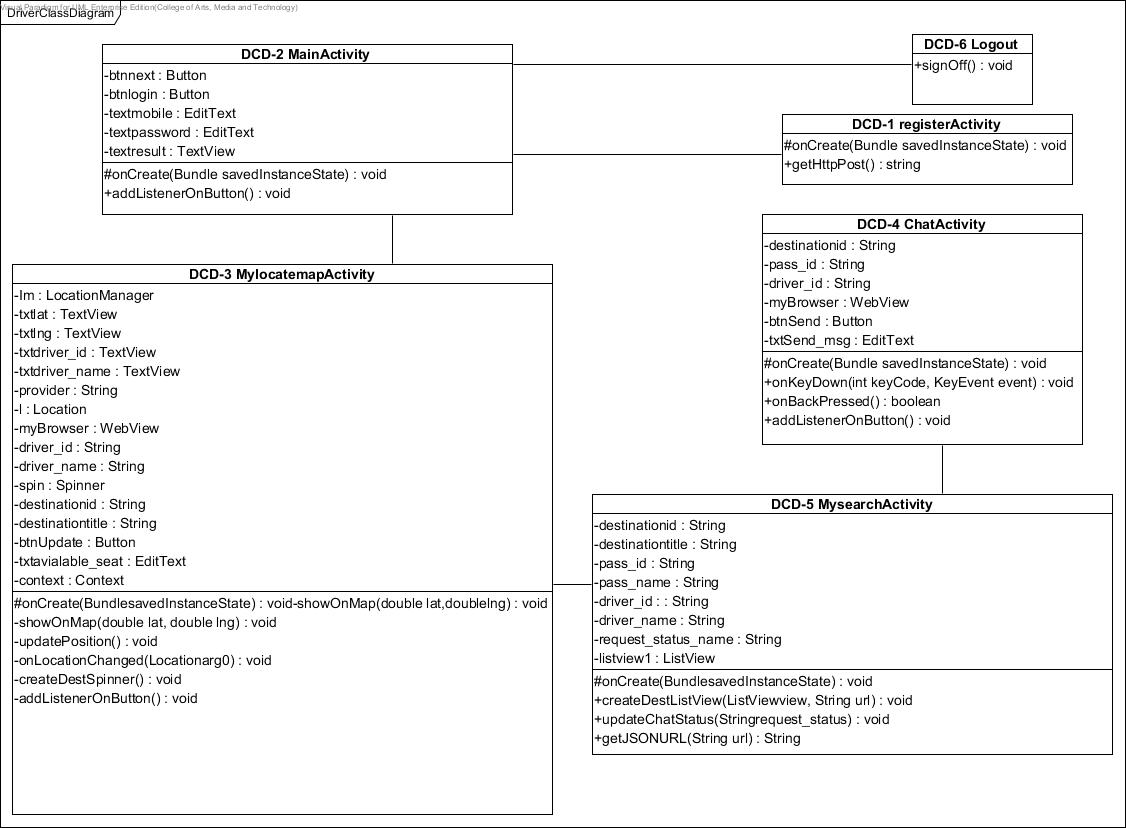
****

*Figure7-7: Database design DB07 tuser*.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Field Name** | **Data type** | **Length** | **PK/FK** | **Null able** | **Description** |
| **suserid** | varchar | 13 | PK | Not Null | This variable is used as an identification number for the Administrator. (e.g. 1) |
| **suserfname** | varchar | 60 | - | Not Null | This variable is used for the Administrator’s first name. (e.g. sompop) |
| **suserlname** | varchar | 60 | - | Not Null | This variable is used for the Administrator’s last name. (e.g. suwatt) |
| **stel** | varchar | 30 | - | Not Null | This variable is used for the Administrator’s home number. (e.g. 053-000000) |
| **smobile** | varchar | 30 |  | Not Null | This variable is used for the Administrator’s mobile number. (e.g. 059-xxx-xxx) |
| **semail** | **varchar** | **60** |  | Not Null | This variable is used for the Administrator’s email address. (e.g. maumiv@hotmail.com) |
| **susername** | **varchar** | **30** |  | Not Null | This variable is used for the Administrator’s username. (e.g. admin) |
| **spassword** | **varchar** | **30** |  | Not Null | This variable is used for the Administrator’s password. (e.g. admin) |
| **ssec** | **varchar** | **60** |  | Not Null | This variable is used for the Administrator’s rights to edit/make changes. (e.g. 111) |
| **scomment** | **varchar** | **255** |  | Not Null | This variable is used for the Administrator to make comments. |
| **dusedlast** | **datetime** | **-** |  | Not Null | This is variable is used to show the Administrator’s last access to the site. (e.g. 2014-07-06 21:02:36) |

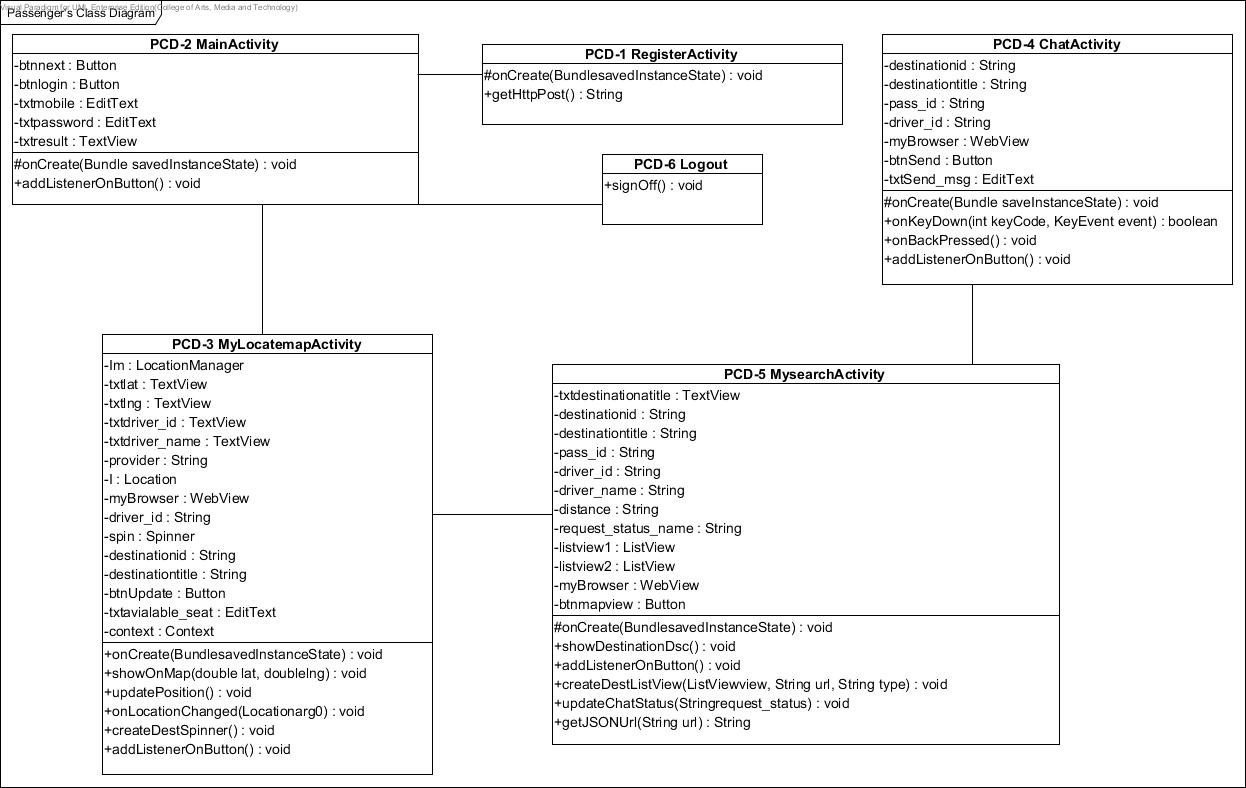
**3.2 Class Diagram Design**

**3.2.1 Driver’s ClassDiagram Design**



*Figure8: Driver’s ClassDiagram Design.*

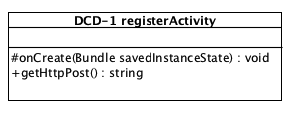
**3.2.2 Passenger’s Class Diagram Design**

****

*Figure9: Passenger’s ClassDiagram Design.*

**3.3 Class Diagram Description**

**3.3.1 Driver’s Class Diagram**

****

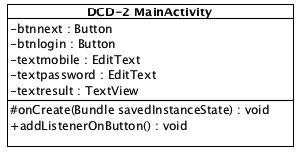
*Figure8-1: DCD-1 registerActivity.*

**Attributes**

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Name** | **Description** | **Data Type** |
| - | - | - | - |

**Methods**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ID** | **Name** | **Description** | **Parameters** | **Return** |
| 1 | #onCreate | Open action on android | Bundle savedInstanceState | void |
| 2 | +getHttpPost | Connect with server | String url,  List<NameValuePair>params | String |



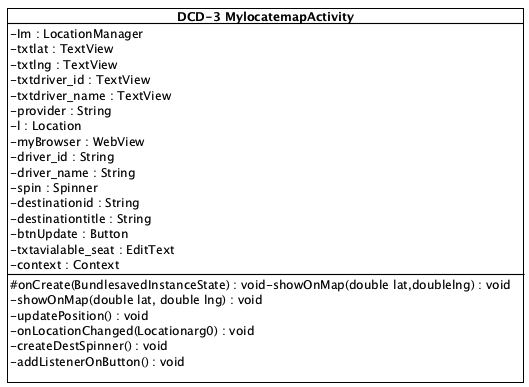
*Figure8-2: DCD-2 MainActivity.*

**Attributes**

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Name** | **Description** | **Data Type** |
| 1 | btnnext | Button which contains next actions after login | Button |
| 2 | btnlogin | Button which contains login action | Button |
| 3 | txtmobile | Text field to put mobile number | EditText |
| 4 | txtpassword | Text field to put password | EditText |
| 5 | txtresult | Field that display login result | TextView |

**Methods**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ID** | **Name** | **Description** | **Parameters** | **Return** |
| 1 | #onCreate | Open action on android | Bundle savedInstanceState | void |
| 2 | addListenerOnButton | Run an action in andriod | - | void |

****

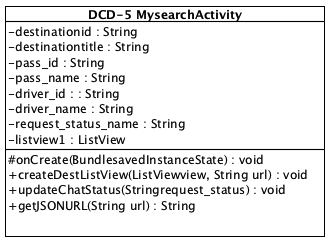
*Figure8-3: DCD-3 MylocatemapActivity.*

**Attributes**

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Name** | **Description** | **Data Type** |
| 1 | Lm | Set the location and connect to google map | LocationManager |
| 2 | txtlat | Set latitude | TextView |
| 3 | txtlng | Set longutude | TextView |
| 4 | txtdriver\_id | Textfield to put driver’s id | TextView |
| 5 | txtdriver\_name | Textfield to put driver’s name | TextView |
| 6 | provider | Set and get map | String |
| 7 | L | Location | Location |
| 8 | myBrowser | Find map | Webview |
| 9 | driver\_id | Driver’s id | String |
| 10 | driver\_name | Driver’s name | String |
| 11 | Spin | Special attribute for spinner method | Spinner |
| 12 | destinationid | Set destination id | String |
| 13 | destinationtitle | Set destination target | String |
| 14 | btnUpdate | Button action update destination | Button |
| 15 | txtavialable\_seat | Textfield set available seat | EditText |
| 16 | context | Set context | Context |

**Methods**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ID** | **Name** | **Description** | **Parameters** | **Return** |
| 1 | #onCreate | Open action on android | Bundle savedInstanceState | void |
| 2 | showOnMap | Show point on map | double lat,  double lng | void |
| 3 | updatePosition | Map update | - | void |
| 4 | onLocationChanged | Map point location update | Location arg0 | void |
| 5 | createDestSpinner | Method with show map destination on android | - | void |
| 6 | addListenerOnButton | Run an action in andriod | - | void |

****

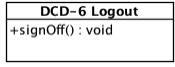
*Figure8-5: DCD-5 MysearchActivity.*

**Attributes**

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Name** | **Description** | **Data Type** |
| 1 | destinationid | Set destination id | String |
| 2 | Pass\_id | Set passenger id | String |
| 3 | driver\_id | Set driver id | String |
| 4 | myBrowser | Find map | WebView |
| 5 | btnSend | Button with send action | Button |
| 6 | txtSend\_msg | Textfield message | EditText |

**Methods**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ID** | **Name** | **Description** | **Parameters** | **Return** |
| 1 | #onCreate | Open action on android | Bundle savedInstanceState | void |
| 2 | onKeyDown | Action that put chat event | intkeyCode, KeyEvent event | void |
| 3 | onBackPressed | Action that send button | - | Boolean |
| 4 | addListenerOnButton | Run an action in andriod | - | void |

****

*Figure8-6: DCD-6 Logout.*

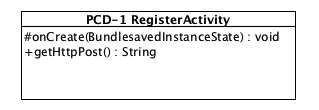
**Attributes**

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Name** | **Description** | **Data Type** |
| - | - | - | - |

**Methods**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ID** | **Name** | **Description** | **Parameters** | **Return** |
| 1 | +signOff | Sign off from the system and display the login page. | - | void |

**3.3.2 Passenger’s Class Diagram**

****

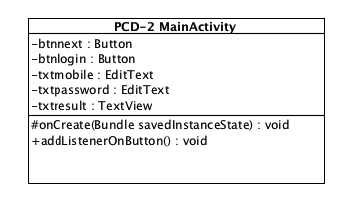
*Figure9-1: PCD-1 RegisterActivity.*

**Attributes**

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Name** | **Description** | **Data Type** |
| - | - | - | - |

**Methods**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ID** | **Name** | **Description** |  | **Return** |
| 1 | #onCreate | Open action on android | Bundle savedInstanceState | void |
| 2 | +getHttpPost | Connect with server | String url,  List<NameValuePair>params | String |

****

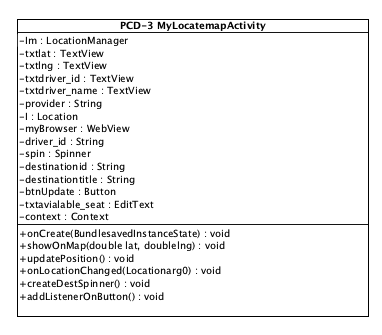
*Figure9-2: PCD-2 MainActivity.*

**Attributes**

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Name** | **Description** | **Data Type** |
| 1 | btnnext | Button which contains next actions after login | Button |
| 2 | btnlogin | Button which contains login action | Button |
| 3 | txtmobile | text field to put mobile number | EditText |
| 4 | txtpassword | text field to put password | EditText |
| 5 | txtresult | field that display login result | TextView |

**Methods**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ID** | **Name** | **Description** | **Parameters** | **Return** |
| 1 | #onCreate | Open action on android | Bundle savedInstanceState | void |
| 2 | addListenerOnButton | Run an action in andriod | - | void |

****

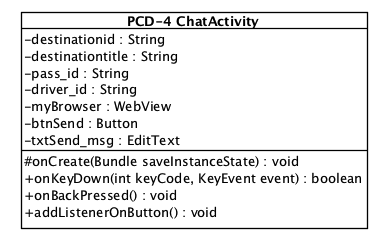
*Figure9-3: PCD-3 MylocatemapActivity.*

**Attributes**

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Name** | **Description** | **Data Type** |
| 1 | Lm | Set the location and connect to google map | LocationManager |
| 2 | txtlat | Set latitude | TextView |
| 3 | txtlng | Set longutude | TextView |
| 4 | txtdriver\_id | Textfield to put driver’s id | TextView |
| 5 | txtdriver\_name | Textfield to put driver’s name | TextView |
| 6 | provider | Set and get map | String |
| 7 | L | Location | Location |
| 8 | myBrowser | Find map | WebView |
| 9 | driver\_id | Driver’s id | String |
| 10 | driver\_name | Driver’s name | String |
| 11 | Spin | Special attribute for spinner method | Spinner |
| 12 | destinationid | Set destination id | String |
| 13 | destinationtitle | Set destination target | String |
| 14 | btnUpdate | Button action update destination | Button |
| 15 | txtavialable\_seat | Textfield set available seat | Edit Text |
| 16 | context | Set context | context |

**Methods**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ID** | **Name** | **Description** | **Parameters** | **Return** |
| 1 | #onCreate | Open action on android | Bundle savedInstanceState | void |
| 2 | showOnMap | Show point on map | double lat,  double lng | void |
| 3 | updatePosition | Map update | - | void |
| 4 | onLocationChanged | Map point location update | Location arg0 | void |
| 5 | createDestSpinner | For displaying destination select list on android | - | void |
| 6 | addListenerOnButton | Run an action in andriod | - | void |

****

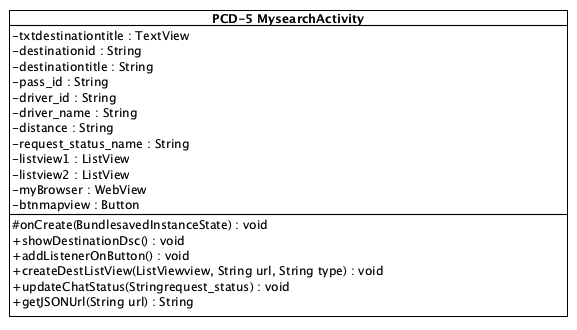
*Figure9-4: PCD-4 ChatActivity.*

**Attributes**

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Name** | **Description** | **Data Type** |
| 1 | destinationid | Set destination id | String |
| 2 | destinationtitle | Set destination title | String |
| 3 | Pass\_id | Set passenger id | String |
| 4 | driver\_id | Set driver id | String |
| 5 | myBrowser | Find map | WebView |
| 6 | btnSend | Button with send action | Button |
| 7 | txtSend\_msg | Textfield message | EditText |

**Methods**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ID** | **Name** | **Description** | **Parameters** | **Return** |
| 1 | #onCreate | Open action on android | Bundle savedInstanceState | Void |
| 2 | onKeyDown | Action that put chat event | intkeyCode,  KeyEvent event | Boolean |
| 3 | onBackPressed | Action that send button |  | Void |
| 4 | addListenerOnButton | Run an action in android |  | Void |

****

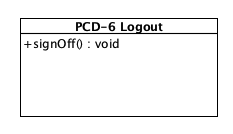
*Figure9-5: PCD-5 MySearchActivity.*

**Attributes**

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Name** | **Description** | **Data Type** |
| 1 | txtdestinationtitle | Set destination title | TextView |
| 2 | destinationid | Set destination id | String |
| 3 | destinationtitle | Set passenger id | String |
| 4 | Pass\_id | Set Pssenger id | String |
| 5 | driver\_id | Set driver id | String |
| 6 | driver\_name | Set driver name | String |
| 7 | distance | Set distance | String |
| 8 | request\_status\_name | Set name on the list | String |
| 9 | listview1 | Set in list1 | ListView |
| 10 | listview2 | Set in list2 | ListView |
| 11 | myBrowser | Find map | WebView |
| 12 | btnmapview | Button show map view | Button |

**Methods**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ID** | **Name** | **Description** | **Parameters** | **Return** |
| 1 | #onCreate | Open action on android | Bundle savedInstanceState | Void |
| 2 | showDestinationDsc | Show destination |  | Void |
| 3 | addListenerOnButton | Run an action in andriod | - | Void |
| 4 | createDestListView | Set destination on list | ListView view,  String url,  String type | Void |
| 5 | updateChatStatus | Update chat status | String request\_status | Void |
| 6 | getJSONUrl | Connect with JSON | String url | Void |

****

*Figure9-6: PCD-6 Logout.*

**Attributes**

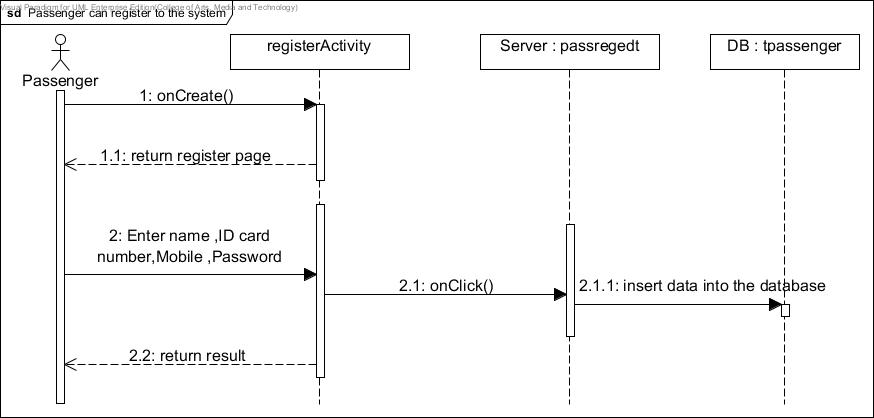
|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Name** | **Description** | **Data Type** |
| - | - | - | - |

**Methods**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ID** | **Name** | **Description** | **Parameters** | **Return** |
| 1 | +signOff | Sign off from the system and display the login page. | - | void |

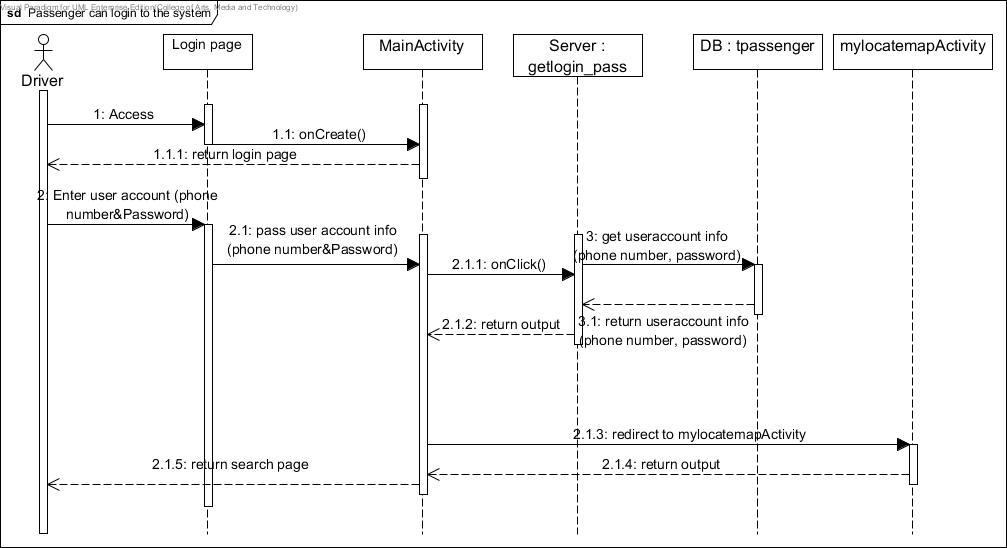
**3.4 Sequence Diagram**

**SD01 Passenger can register to the system (URS-01)**

****

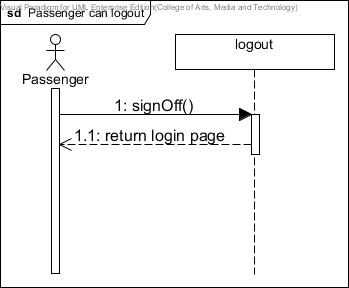
*Figure10-1: SD01 Passenger can register to the system.*

**SD02 Passenger can login to the system (URS-02)**

****

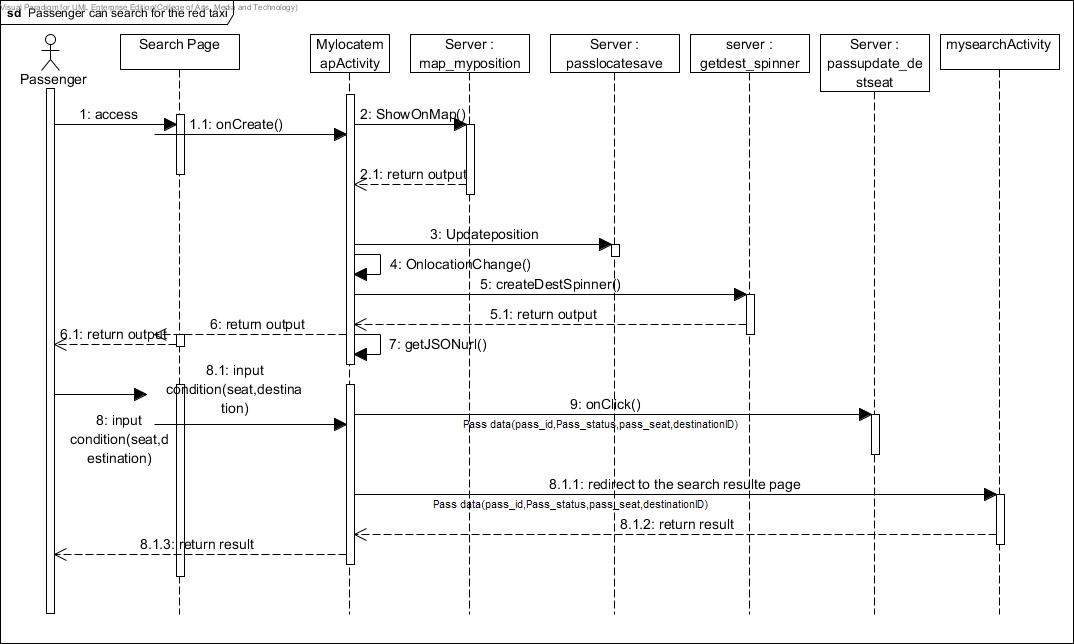
*Figure10-2: SD02 Passenger can login to the system (URS-02)*

**SD03 Passenger can logout of the system (URS-03)**

****

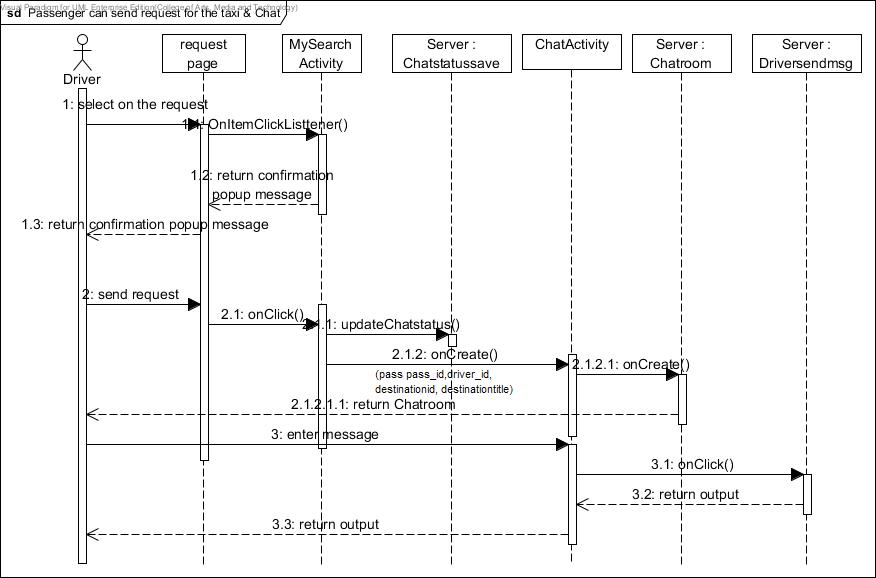
*Figure10-3: SD03 Passenger can logout of the system (URS-03)*

**SD04 Passenger can search for taxi (URS-04)**

****

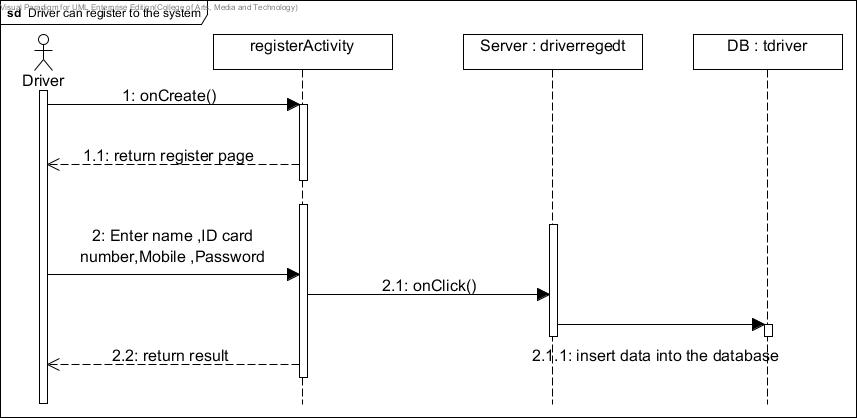
*Figure10-4: SD04 Passenger can search for taxi (URS-04)*

**SD05 Passenger can send request for taxi (URS-05), Passenger can chat with driver (URS-06)**

****

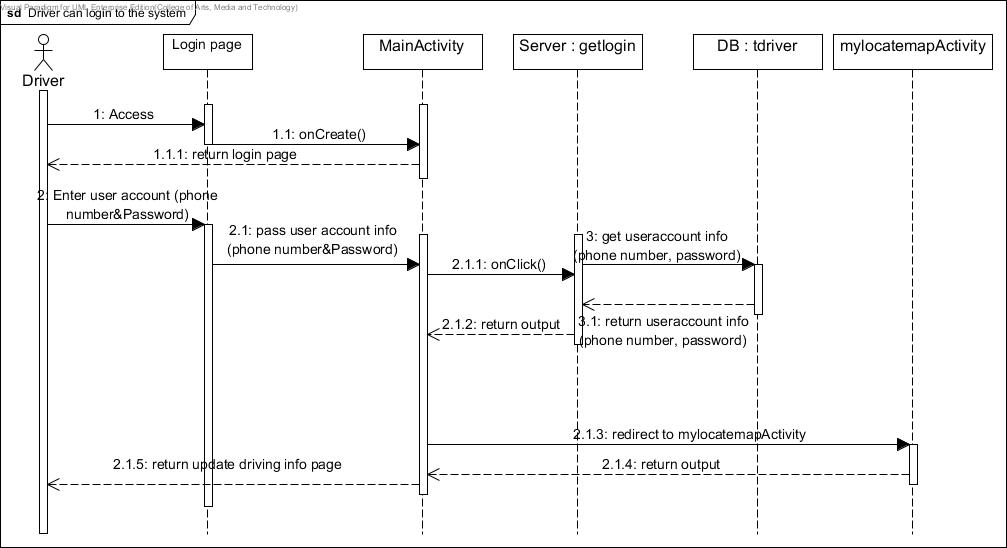
*Figure10-5: SD05 Passenger can send request for taxi (URS-05), Passenger can chat with driver (URS-06)*

**SD06 Driver can register into the system (URS-07)**

****

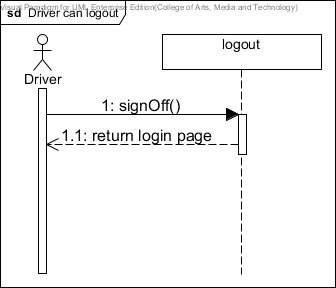
*Figure10-6: SD06 Driver can register into the system (URS-07)*

**SD07 Driver can log in to the system (URS-08)**

****

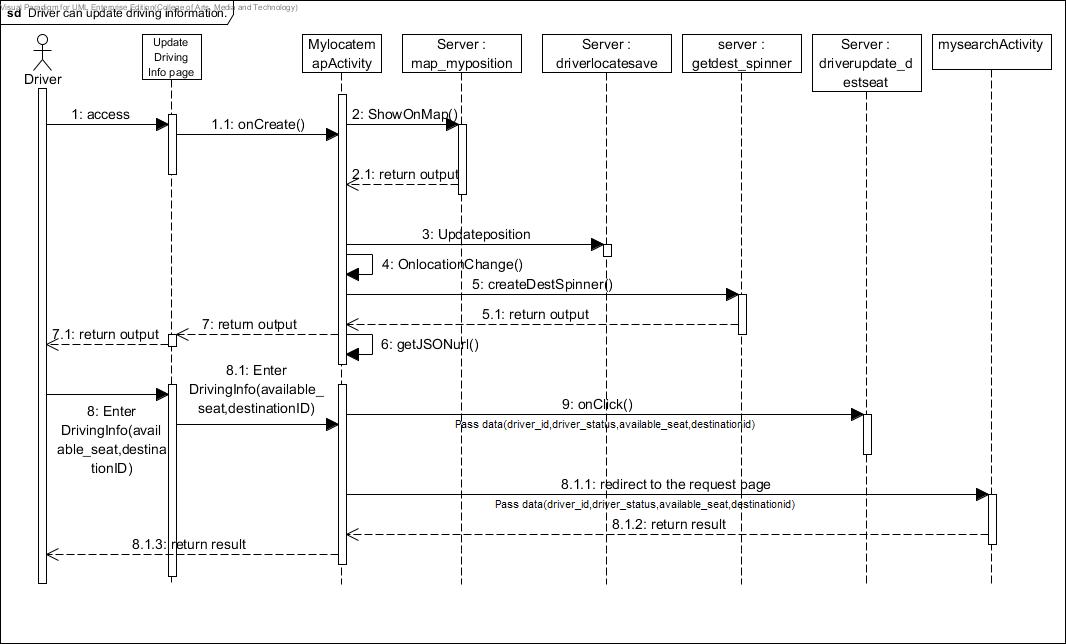
*Figure10-7: SD07 Driver can log in to the system (URS-08)*

**SD08 Driver can logout from the system (URS-09)**

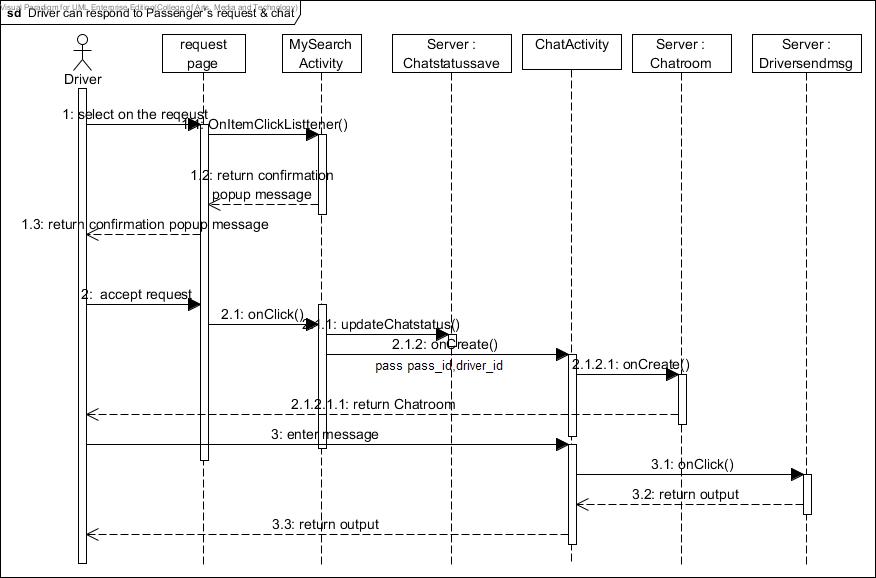
****

*Figure10-8: SD08 Driver can logout from the system (URS-09)*

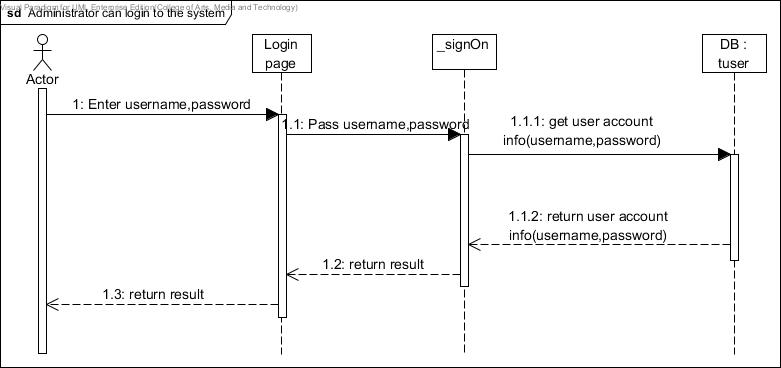
**SD09 Driver can update driving information (URS-10)**

*****Figure10-9: SD09 Driver can update driving information (URS-10)*

**SD10 Driver can respond to passenger’s request (URS-11), Driver can chat with passenger (URS-012)**

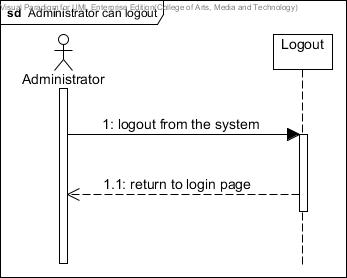
*****Figure10-10: SD10 Driver can respond to passenger’s request (URS-11), Driver can chat with passenger (URS-012)*

**SD11 Administrator can login to the Administration system (URS-013)**

****

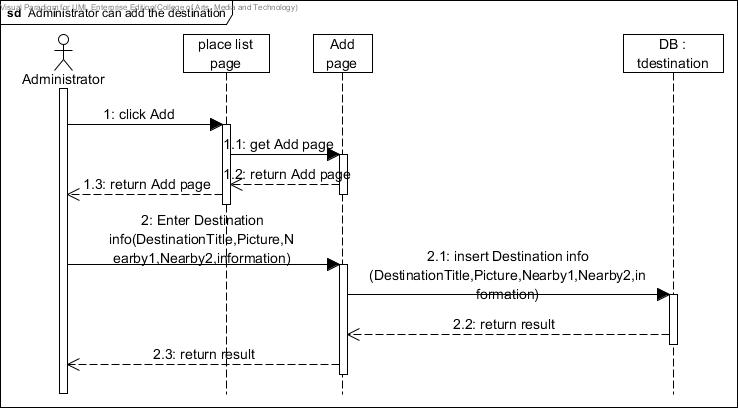
*Figure10-11: SD11 Administrator can login to the Administration system (URS-013)*

**SD12 Administrator can logout (URS-14)**

****

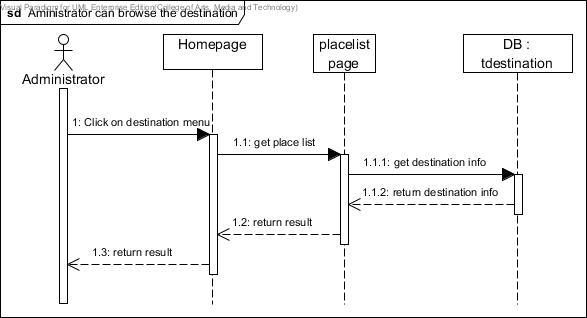
*Figure10-12: SD12 Administrator can logout (URS-14)*

**SD13 Administrator can add destinations (URS-15)**

****

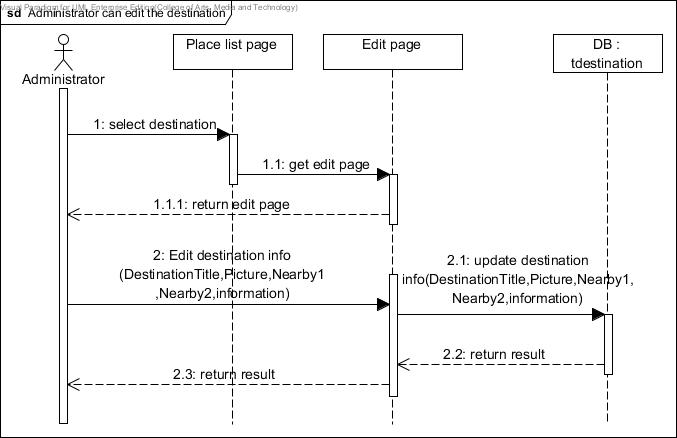
*Figure10-13: SD13 Administrator can add destinations (URS-15)*

**SD14 Administrator can browse the destination (URS-16)**

****

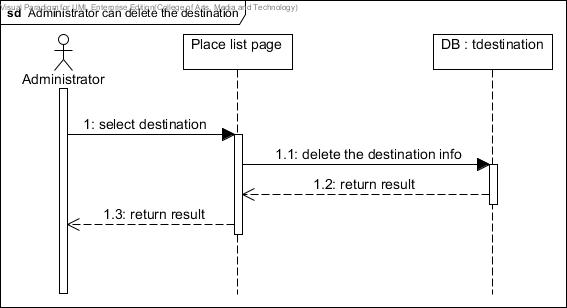
*Figure10-14: SD14 Administrator can browse destinations (URS-16)*

**SD15 Administrator can edit destinations (URS-17)**

****

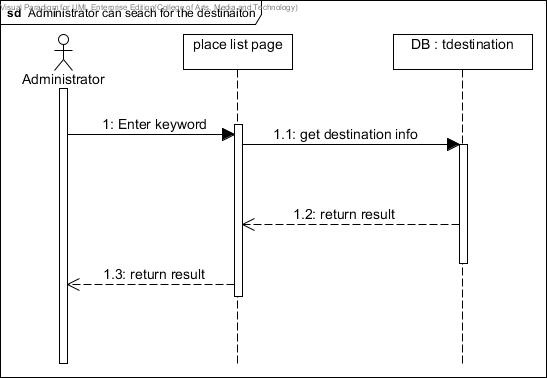
*Figure10-15: SD15 Administrator can edit destinations (URS-17)*

**SD16 Administrator can delete destinations (URS-18)**

****

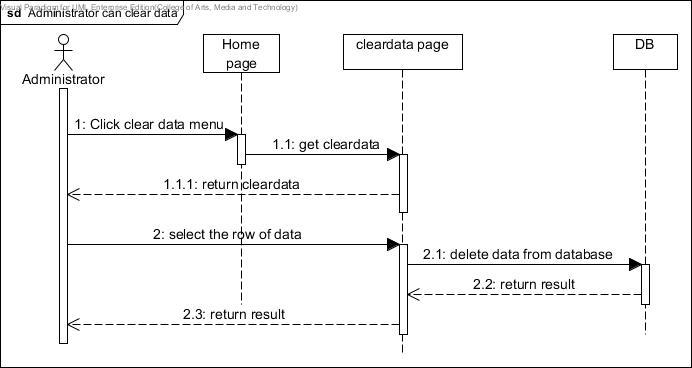
*Figure10-16: SD16 Administrator can edit destinations (URS-18)*

**SD17 Administrator can search destinations (URS-19)**

****

*Figure10-17: SD17 Administrator can edit destinations (URS-19)*

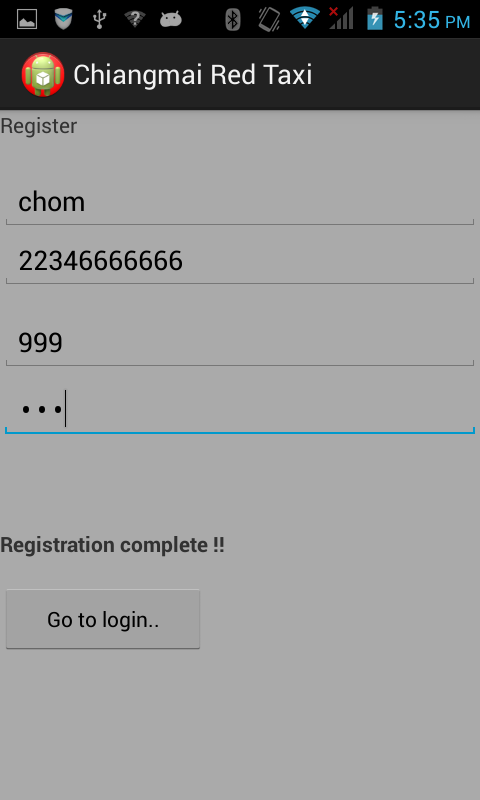
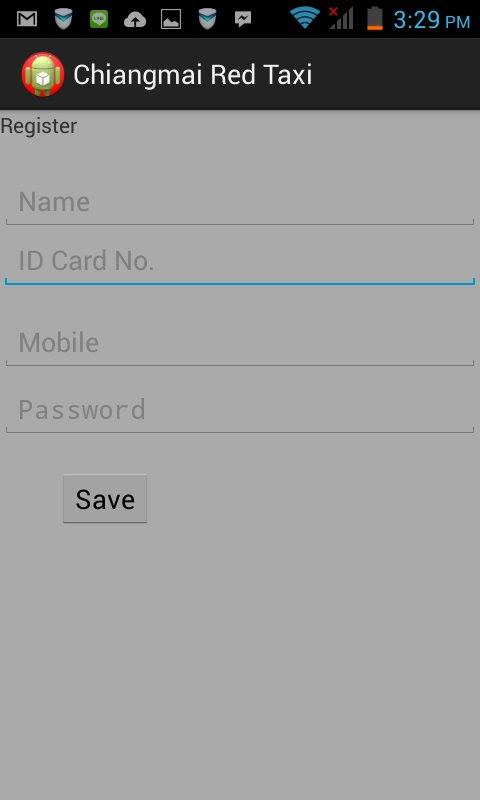
**SD18 Administrator can clear data (URS-20)**

*****Figure10-18: SD18 Administrator can clear data (URS-20)*

**3.5 User Interface Design**

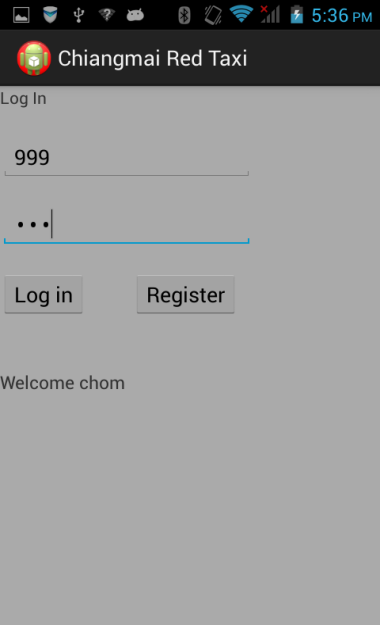
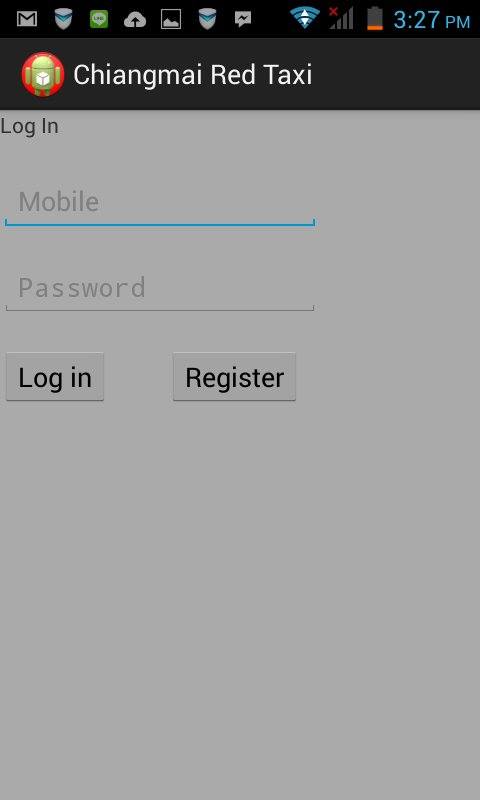
**3.5.1 User Interface for Passenger – Mobile Application**

UI-01Passenger can register to the system (URS-01)



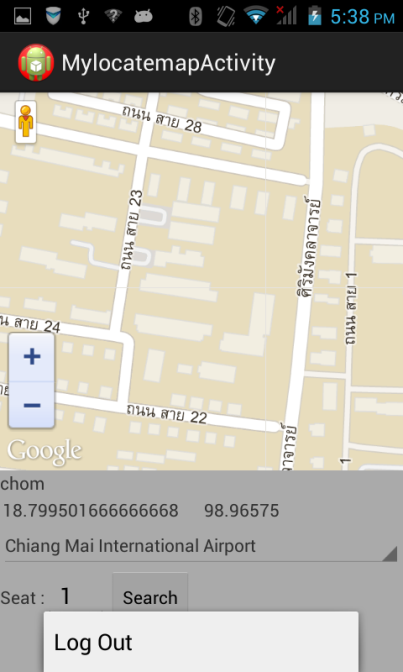
*Figure 11-1:left-The user interface of passenger’s registration page,  
right – registration completed.*

UI-02 Passenger can login to the system (URS-02)



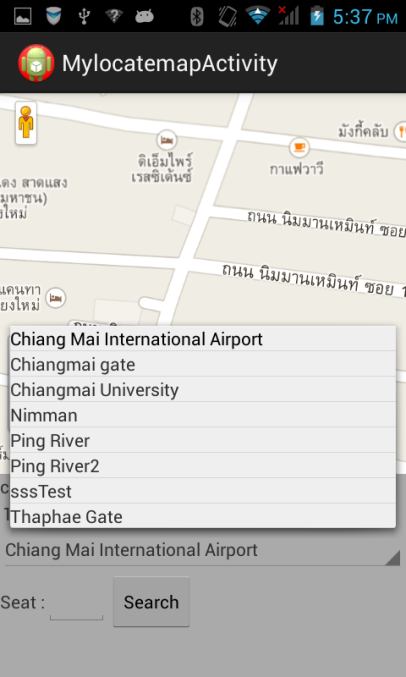
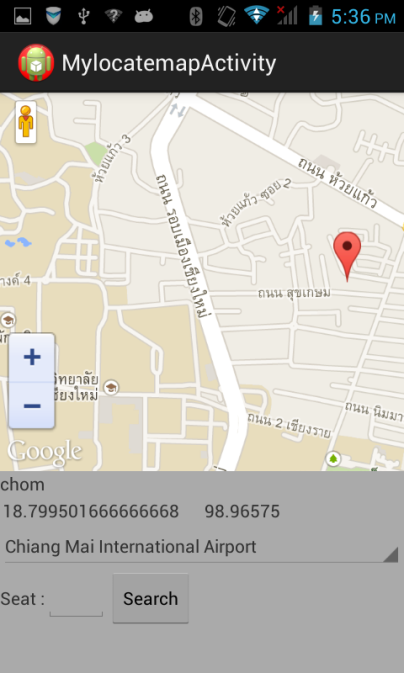
*Figure 11-2: left-The user interface of passenger’s login page,  
right – login completed.*

UI-03 Passenger can logout of the system (URS-03)

****

*Figure 11-3: User interface when passenger clicks on menu to log out.*

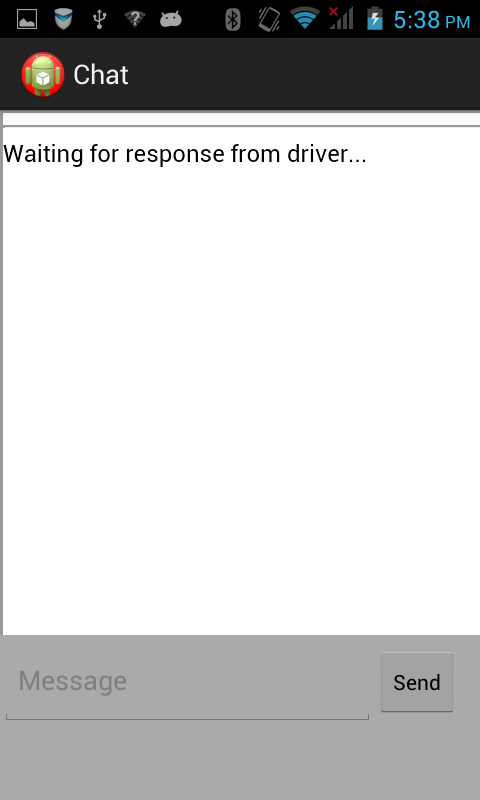
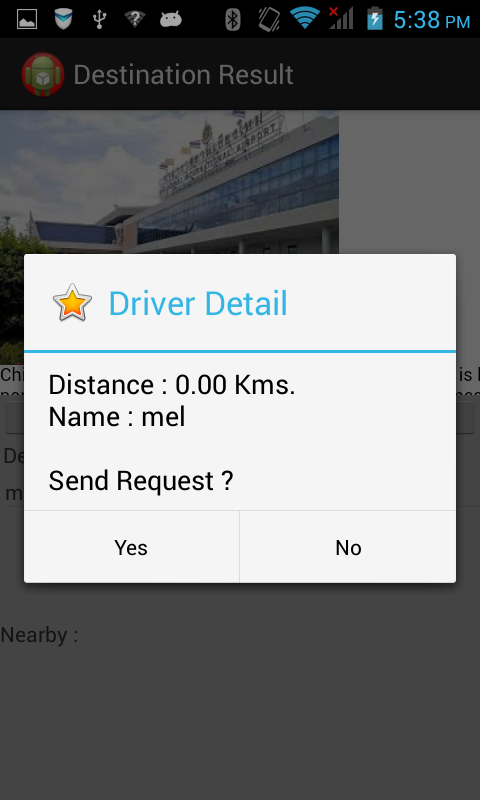
UI-04 Passenger can search for taxi (URS-04)

****

*Figure 11-4:left-The user interface of passenger’s search page,  
center – selecting destination from the provided list,*

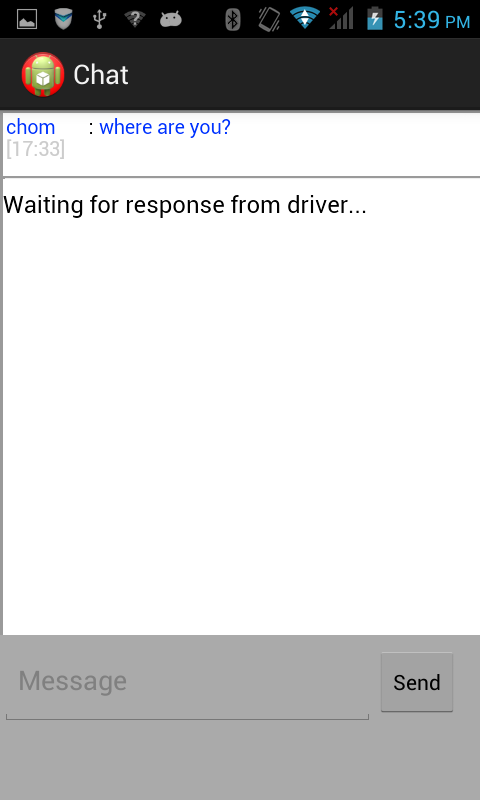
*right – search results.*

UI-05 Passenger can send request for taxi (URS-05)



*Figure 11-5:left-The pop-up message asking for confirmation of sending request to Driver,  
right - Request information and chat box displayed after request sent.*

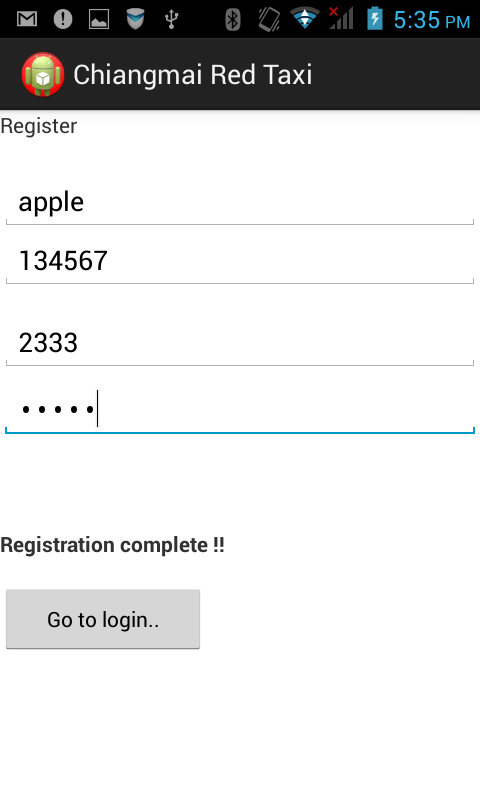
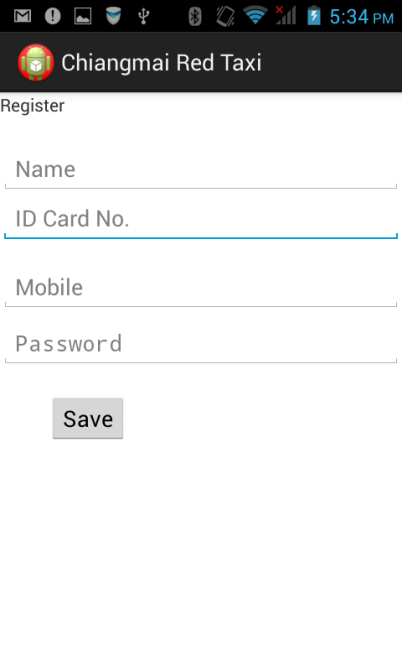
UI-06 Passenger can chat with driver (URS-06)



*Figure 11-6: The user interface of passenger’s chat with Driver*

**3.5.2 User Interface for Driver – Mobile Application**

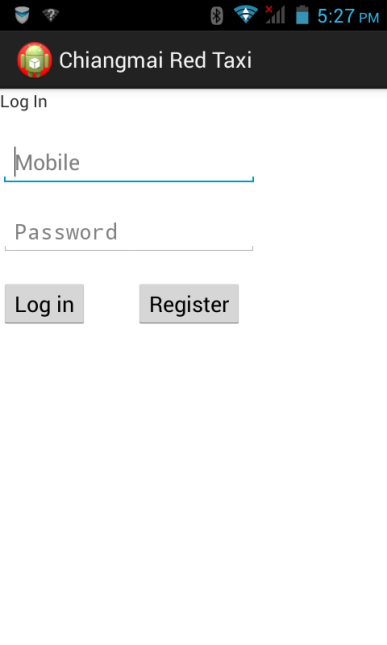
UI-07 Driver can register into the system (URS-07)



*Figure 11-7: left-The user interface of Driver’s registration page,*

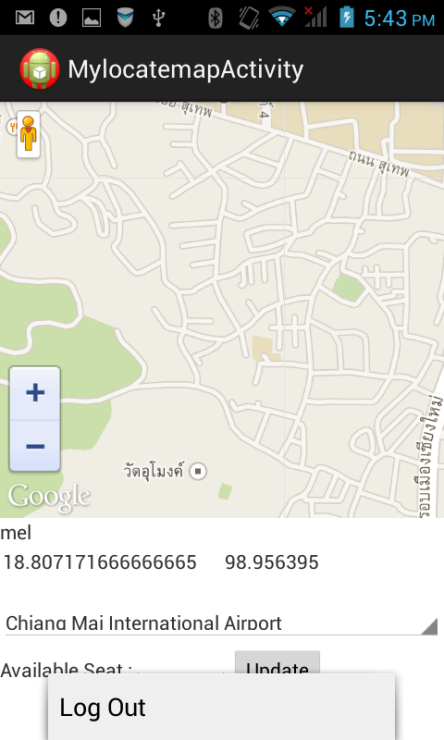
*right- registration completed.*

UI-08Driver can log in to the system (URS-08)



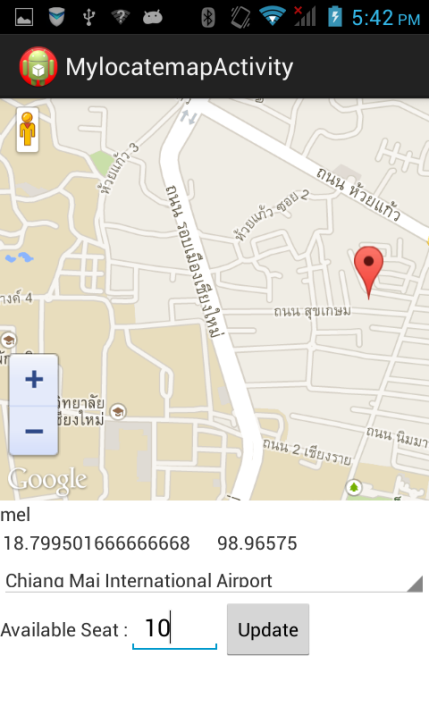
*Figure 11-8 :The user interface of Driver’s login page*.

UI-09Driver can logout from the system (URS-09)



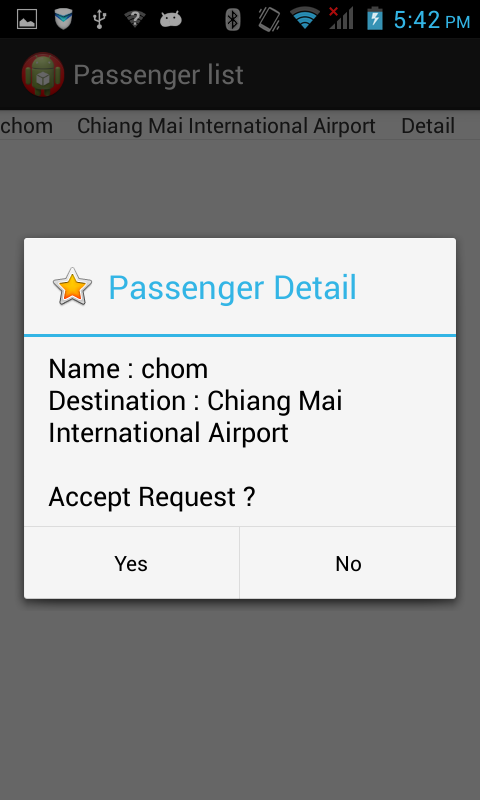
*Figure 11-9:The user interface when driver clicks on menu to log out.*

UI-10Driver can update driving information (URS-10)



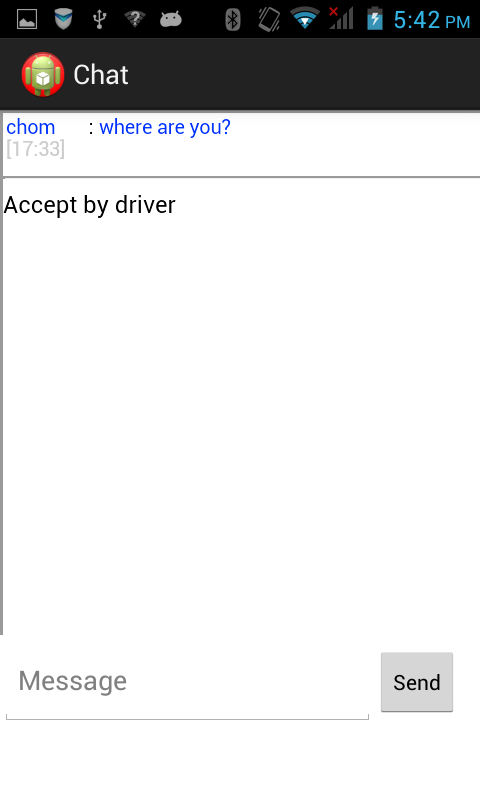
*Figure 11-10 :left-The user interface of Driver’s updating driving information page,  
right- Driver can select the destination from the destination list provided.*

UI-11 Driver can respond to passenger’s request (URS-11)



*Figure 11-11: left-The user interface of Passenger request list,  
right –Passenger request pop-up  
.*

UI-12 Driver can chat with passenger (URS-012)



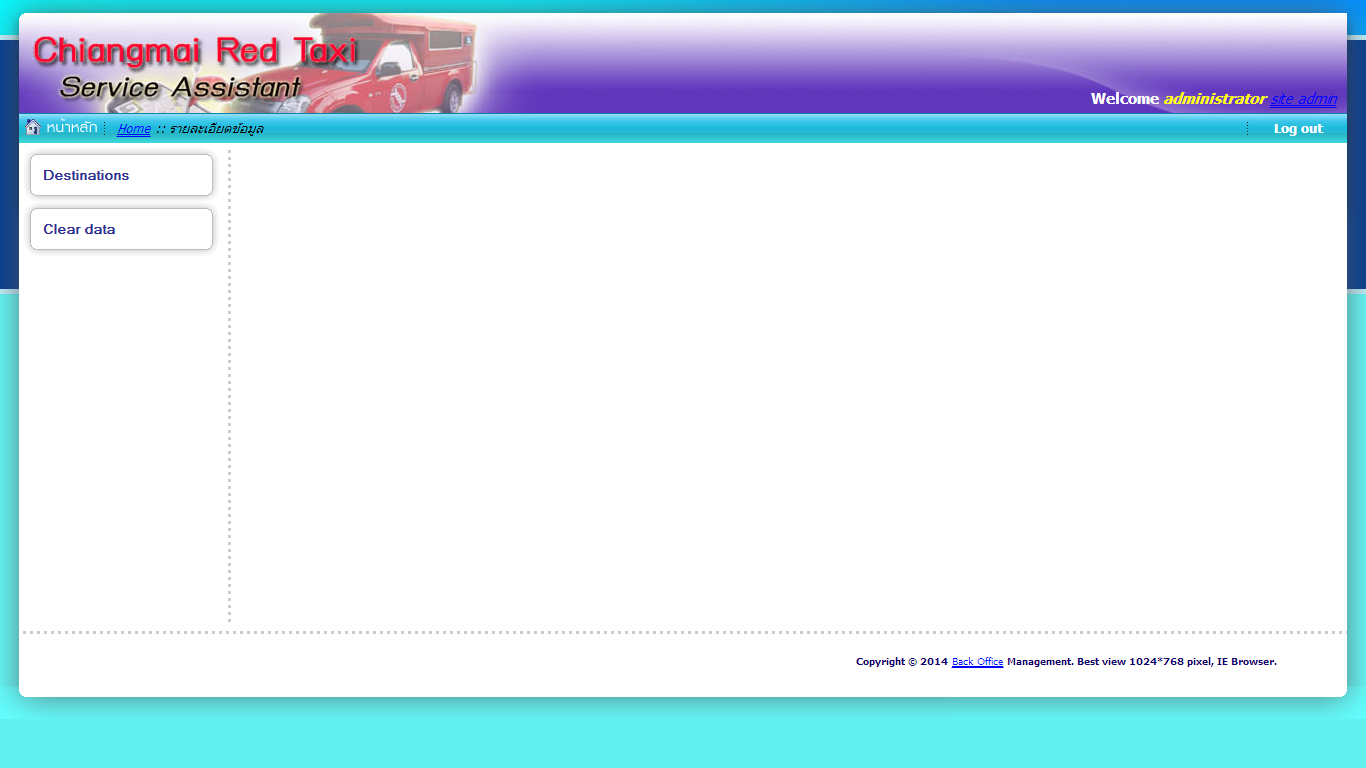
*Figure 11-12:The user interface of Passenger’s chat conversation with the Driver.*

**3.5.3 User Interface for Administrator – Web Application**

UI-13 Administrator can login to the Administration system (URS-013)

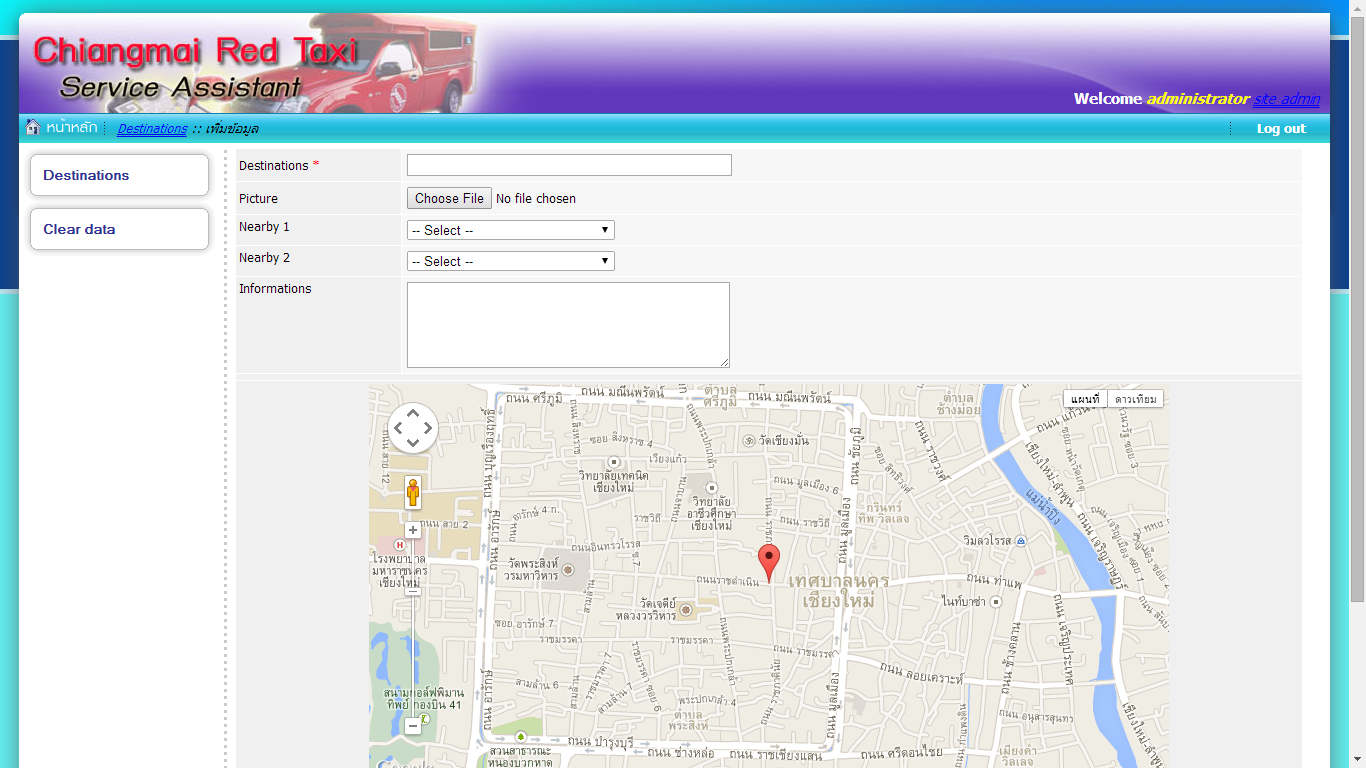


*Figure 12-1: The user interface of Administrator’s login page.*

UI-14 Administrator can logout (URS-14) ****

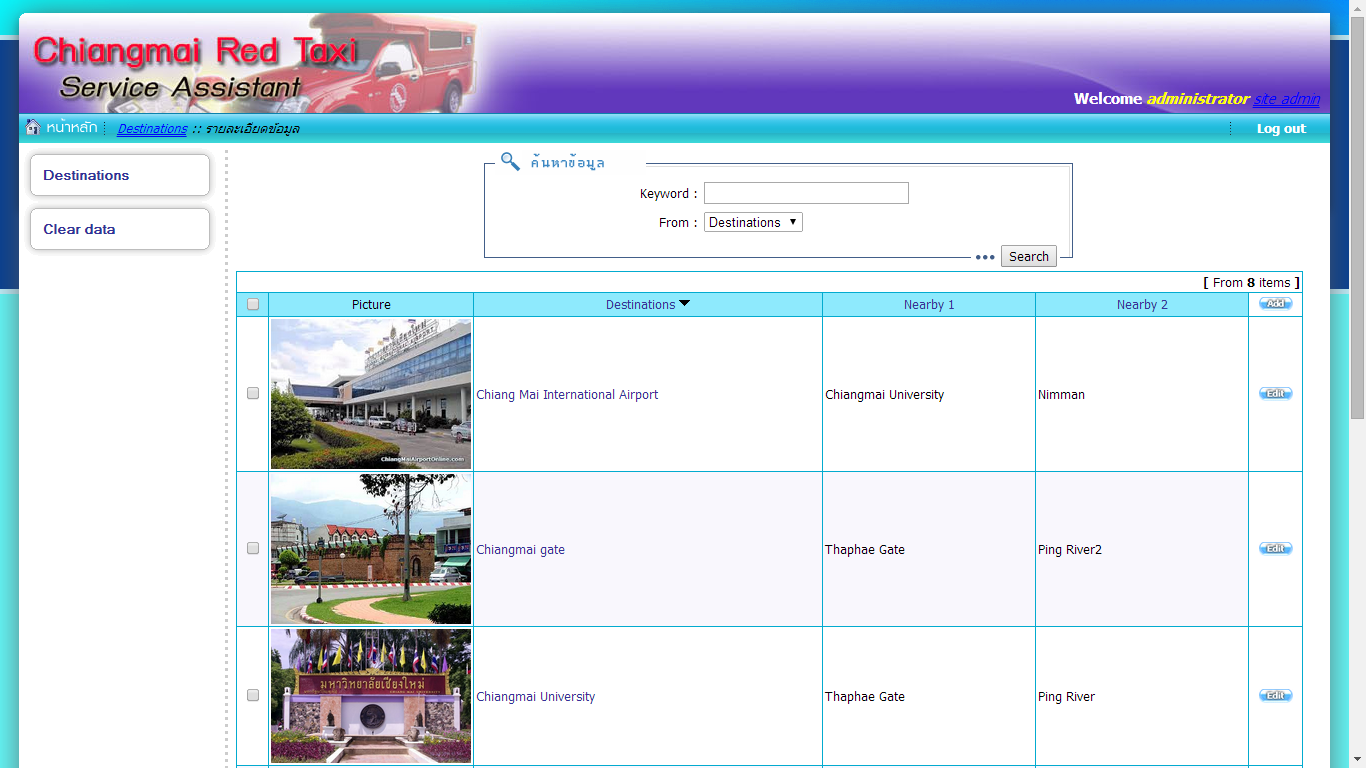
*Figure 12-2: The user interface of Administrator’s Homepage with the logout button on the top-right of the page .*

UI-15 Administrator can add destinations (URS-15)

****

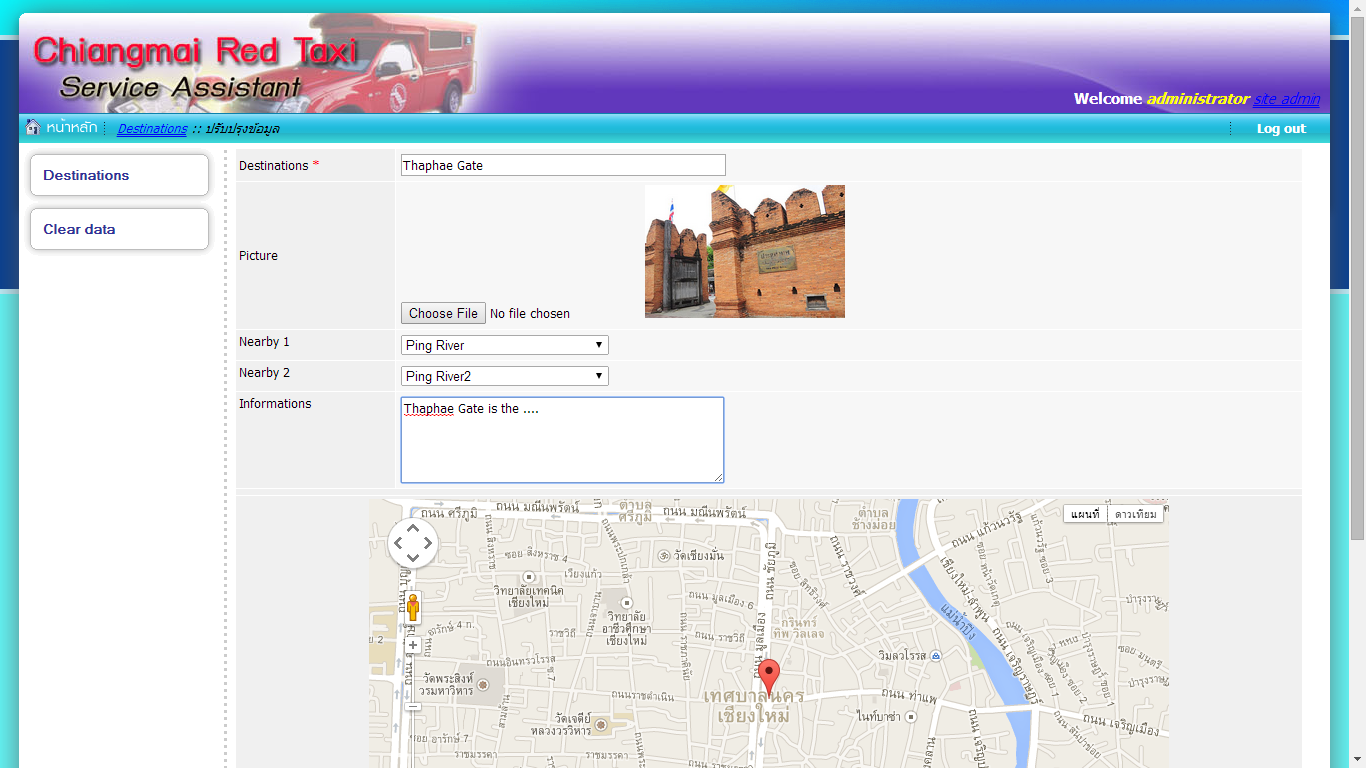
*Figure 12-3 :The user interface for the Administrator’s to add destination.*

UI-16 Administrator can browse the destination (URS-16)

****

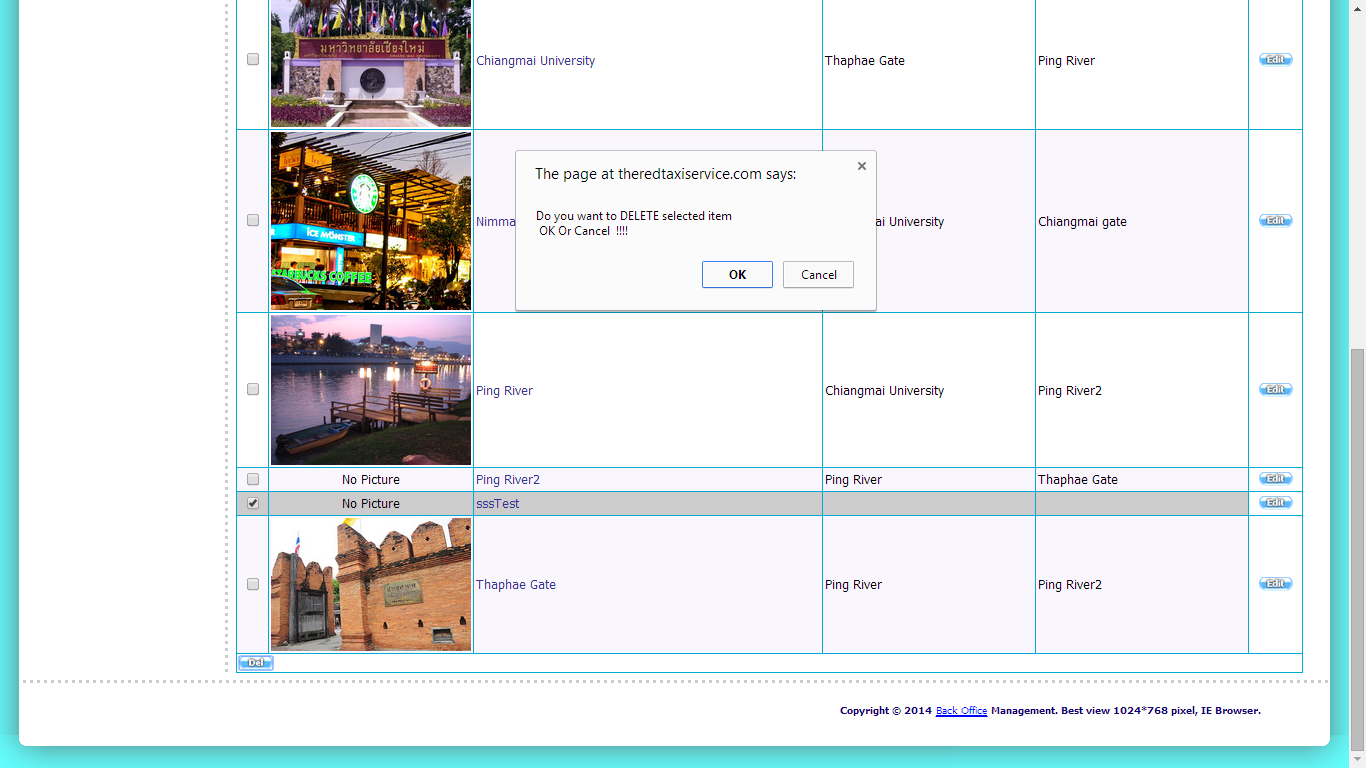
*Figure 12-4: The user interface whenAdministrator wants to browse the destinations.*

UI-17 Administrator can edit destinations (URS-17)

****

*Figure 12-5: The user interface when administrator clicks to edit each destination*.

UI-18 Administrator can delete destinations (URS-18)

*****Figure 12-6: The pop up message that appearswhen Administratorclicks to delete each destination*.

UI-19 Administrator can search destinations (URS-19)

****

*Figure 12-7:The result page after the Administrator searches for a destination*.

UI-20 Administrator can clear data (URS-20)

****

*Figure 12-8: The pop-up message that appearswhen Administrator clicks to clear data   
in each table*.