**Emergency Information on Mobile**

Software Design Document

By

Putchakarn Jaikon 542115031

Sawatdiporn Kitirot 542115065

Department of Software Engineering

College of Arts, Media and Technology

Chiang Mai University

Project Advisor

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Aj.Chartchai Doungsa-ard**

**Document History**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Document Name** | **Details** | **Status** | **Date** | **Viewable** | **Reviewer** | **Responsible** |
| **Documents** | | | | | | |
| **EIOM-SDD-V.0.1.docx** | **Chapter 1**      Introduction | Draft | 1/4/2014 | PJ, SK, CD | PJ, SK | PJ, SK |
| **EIOM-SDD-V.0.2.docx** | **Chapter 2**      System Architecture  **Chapter 3**      Detailed Design | Draft | 17/4/2014 | PJ, SK, CD | PJ, SK | PJ, SK |
| **EIOM-SDD-V.0.3.docx** | **Modify Chapter 3**      Class Diagram      Class Diagram Description      Sequence Diagram | Draft | 23/4/2014 | PJ, SK, CD | PJ, SK | PJ, SK |
| **EIOM-SDD-V.0.4.docx** | **Chapter 4**      User Interface Design | Draft | 28/4/2014 | PJ, SK, CD | PJ, SK | PJ, SK |
| **EIOM-SDD-V.0.5.docx** | **Modify Chapter 4**      User Interface Design | Draft | 30/4/2014 | PJ, SK, CD | PJ, SK | PJ, SK |
| **EIOM-SDD-V.0.6.docx** | **Modify Chapter 3**      Class Diagram Description | Draft | 1/5/2014 | PJ, SK, CD | PJ, SK | PJ, SK |
| **EIOM-SDD-V.0.7.docx** |     Add Table of content and cover page | Draft | 4/5/2014 | PJ, SK, CD | PJ, SK | PJ, SK |
| **EIOM-SDD-V.1.0.docx** | **Modify Chapter 3**      Class Diagram      Class Diagram Description      Sequence Diagram  **Modify Chapter 4**      User Interface Design | Release | 10/5/2014 | PJ, SK, CD | PJ, SK | PJ, SK |

PJ – Putchakarn Jaikon, SK – Sawatdiporn Kitirot, CD – Chartchai Doungsa-ard

Table of Contents

[Chapter One | Introduction 4](#_Toc394405803)

[1.1 Objective 4](#_Toc394405804)

[1.2 Project Scope 4](#_Toc394405805)

[1.3 Purpose 4](#_Toc394405806)

[1.4 Acronyms and Definitions 4](#_Toc394405807)

[Chapter Two | System architecture 6](#_Toc394405808)

[Chapter Three | Detailed Design 8](#_Toc394405809)

[3.1 Class Diagram 8](#_Toc394405810)

[3.2 Class Diagram Description 13](#_Toc394405811)

[3.3 Sequence Diagram 57](#_Toc394405812)

[Chapter Four | User Interface Design 72](#_Toc394405813)

# Chapter One | Introduction

## 1.1 Objective

The purpose of the software design document (SDD) for Emergency Information on Mobile project is to design the detailed structure of the system accordance with the software requirement specification (SRS). This SDD also making the members in the project team understand the work in the detailed design of the system using the class diagram, sequence diagram, entity relationship diagram and user interface design.

## 1.2 Project Scope

Emergency Information on Mobile is composed of two parts that are server and mobile part. The server part uses to manage the information of help place. The mobile application part runs on Android OS. Emergency Information on Mobile will provide the necessary information of help places to the user.

## 1.3 Purpose

This software design document consists of progress report I. So the stakeholder of Emergency Information on Mobile can review software design in this progress.

## 1.4 Acronyms and Definitions

1.4.1 Acronyms

EIOM Emergency Information on Mobile

SDD Software Design Document

UI User Interface

CDs Class Diagram Server

CDm Class Diagram Mobile

MDs Method Description Server

MDm Method Description Mobile

1.4.2 Definitions

|  |  |
| --- | --- |
| Sequence diagram | A sequence diagram in a [Unified Modeling Language](http://en.wikipedia.org/wiki/Unified_Modeling_Language) (UML) is a kind of [interaction diagram](http://en.wikipedia.org/wiki/Interaction_diagram) that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. A sequence diagram shows object interactions arranged in time sequence. It depicts the objects and classes involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario. |
| Feature | Transformation of input parameters to output parameters based on a specified algorithm. It describes the functionality of a product in the language of the product. Used for requirements analysis, design, coding, testing or maintenance.[IEEE90] |
| IEEE | Institute for Electrical and Electronics Engineers. Biggest global interest group for engineers of different branches and for computer scientists. [IEEE90] |
| User Interface | The portion of a computer program with which the user interacts, i.e., the interface between a user and a computer program. There are command-line interfaces, menu-driven interfaces, and graphical user interfaces (GUIs). |
| UML | Unified Modeling Languages. Standardized notation for Modeling design descriptions, architectures or scenarios. Not depending on a specific method. Issued and maintained byte object Management Group (OMG).[IEEE90] |

# Chapter Two | System architecture

****

**Figure 1 Architecture overview of Emergency Information on Mobile system**

Figure 1 shows overall of architecture such as map and help information system, search information system and manage information system.

|  |  |  |  |
| --- | --- | --- | --- |
| **Feature** | **Function name** | **Online** | **Offline** |
| **Feature1:** Map and help information system | View map and help places | √ | √ |
| View information of help places | √ | √ |
| Make emergency call | √ | √ |
| **Feature2:** Search information system | Search the help place by keyword | √ |  |
| Find the nearest help place | √ | √ |
| **Feature3:** Rating location | Rate the help place | √ |  |
| View average rating score | √ | √ |
| **Feature4:** Automatic collecting data system | Download data of help place automatically | √ |  |
| Set the scope for downloading data | √ |  |
| **Feature5:** Manage information system | Add help place | √ |  |
| Edit help place | √ |  |
| Remove help place | √ |  |
| View information of the help place | √ |  |
| Browse the help place by category | √ |  |
| Browse the help place by province | √ |  |
| Browse the help place by help place’s category and help place’s province | √ |  |

**Mobile Part**

**Feature1: Map and help information system**

In this feature, the help place will show on the map with their information such as address and phone number. Moreover, the phone number can be called directly on the application.

**Feature2: Search information system**

Feature 2 provides search help place by keyword or name of help place. Furthermore, the application can show the nearest help place in many categories such as a police station, hospital, and garage.

**Feature3: Rating location**

In this feature, the user can use the rate function to rate each help place. One user will be count at one for rating each place. The rating location collects the rate and provides the average rate to the user. Furthermore, the rate function will help the user to compose their decision to go among many help places.

**Feature4: Automatic collecting data system**

Feature 4 will download data of help place around the user automatically and save into a mobile device. So, the information can show without the internet connection. In addition, the user can set the scope of download data.

**Server part**

**Feature5: Manage information system**

Feature 5 furnishes manage information system to admin. The administrator can add, edit, remove, view information of the help place. Moreover, the administrator can browse the help place by selecting category or province.

# Chapter Three | Detailed Design

## 3.1 Class Diagram

3.1.1 Server Part

Controller class diagram



**Figure 1 Controller class diagram**

Service class diagram



**Figure 2 Service class diagram**

DAO class diagram



**Figure 3 DAO class diagram**

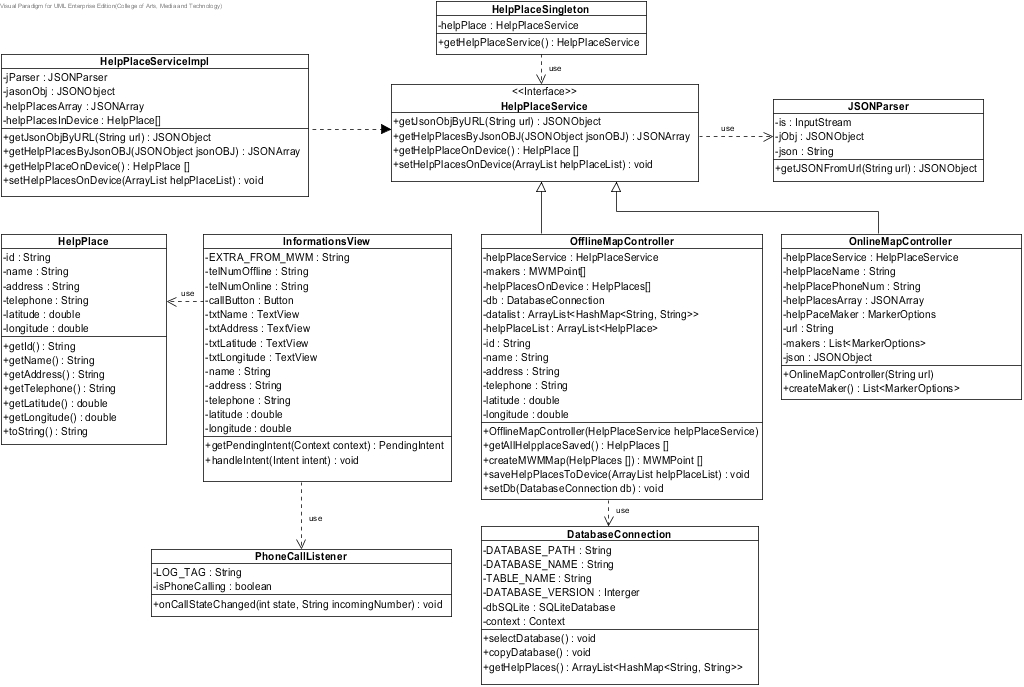
Entity class diagram



**Figure 4 Entity class diagram**

3.1.2 Mobile Part

Emergency Information on Mobile class diagram



**Figure 5 Emergency Information on Mobile class diagram**

## 3.2 Class Diagram Description

3.2.1 Server Part

1. HelpPlace



**Figure 6 HelpPlace class**

**Entities**

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Name** | **Description** | **Type** |
| 1 | id | stored id of the help place | Integer |
| 2 | name | stored name of the help place | string |
| 3 | district | stored district of the help place | string |
| 4 | province | stored province object of the help place | Province |
| 5 | zipCode | stored zipcode of the help place | string |
| 6 | phoneNumber | stored phone number of the help place | string |
| 7 | latitude | stored latitude of the help place | Double |
| 8 | longitude | stored longitude of the help place | Double |
| 9 | category | stored category object of the help place | Category |

**Method details**

1. HelpPlaces

public HelpPlace(String name, String address, String district, String zipcode, Province province, String phoneNumber, Double latitude, Double longitude, Category category)

**Description:** A constructor method is used to set the attributes of help place object.

1. HelpPlaceDAO



**Figure 7 HelpPlaceDAO class**

**Method details**

1. getHelpPlaces

public List<HelpPlace> getHelpPlaces()

**Description:** Get a list of help places from the database.

**Return:** Lists of the help places in the database.

1. updateHelpPlace

public HelpPlace updateHelpPlace(HelpPlace helpPlace)

**Description:** Update information of the help place to the database.

**Parameter:** helpPlace-HelpPlace uses for update information of the help place to the database.

**Return:** Help place object that needs to update.

1. deleteHelpPlace

public boolean deleteHelpPlace(HelpPlace helpPlace)

**Description:** Delete the help place from database.

**Parameter:** helpPlace-HelpPlace uses for delete the help place from database.

**Return:** Boolean, if the help place has been deleted, will return true.

1. findById

public HelpPlace findById (Integer id)

**Description:** Get help place by id from the database.

**Parameter:** id-Integer uses for getting the help place.

**Return:** The help place from database.

1. getHelpPlacesByCategory

public List<HelpPlace> getHelpPlacesByCategory(Integer categoryId)

**Description:** Get help places from the database by the selected category id.

**Parameter:** categoryId-Integer uses for getting the help place from database.

**Return:** Lists of the help places that contain the selected category, from database.

1. getHelpPlacesByProvince

public List<HelpPlace> getHelpPlacesByProvince(Integer provinceId)

**Description:** Get help places from the database by the selected province id.

**Parameter:** provinceId -Integer uses for getting the help place from database.

**Return:** Lists of the help places that contain the selected province, from database.

1. getHelpPlacesByCategoryAndProvince

public List<HelpPlace> getHelpPlacesByCategoryAndProvince (Integer categoryId, Integer provinceId)

**Description:** Get help places from the database by the selected category id and province id.

**Parameter:** provinceId–Integer and categoryId-Integer use for getting the help place from database.

**Return:** Lists of the help places that contain the selected category and province, from database.

1. HelpPlaceDAOImpl



**Figure 8** HelpPlaceDAOImpl **class**

**Entities**

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Name** | **Description** | **Type** |
| 1 | sessionFactory | A thread-safe, immutable cache of compiled mappings for a single database | SessionFactory |

**Method details**

1. getHelpPlaces

public List<HelpPlace> getHelpPlaces()

**Description:** Get help places from the database by using session to query data.

**Flow:**

1. Create list of help place to store data.
2. Create query to get all help places.
3. Store the queried data in list.
4. Return the list.

**Return:** List of help places in database.

1. updateHelpPlace

public HelpPlace updateHelpPlace(HelpPlace helpPlace)

**Description:** Update information of the help place to database by using session to query data.

**Parameter:** helpPlace-HelpPlace uses for update information of the help place to database.

**Flow:**

1. Create query to update the help place where help place equal to helpPlace.
2. Return helpPlace.

**Return:** Help place object that needs to update.

1. deleteHelpPlace

public boolean deleteHelpPlace(HelpPlace helpPlace)

**Description:** Delete the help place from database by using session to query data.

**Parameter:** helpPlace-HelpPlace uses for delete the help place from database.

**Flow:**

1. Create Boolean equal to false.
2. If helpPlace-HelpPlace not equal to null.
3. Create id-Integer to store ID of helpPlace-HelpPlace, which comes from parameter.
4. Create helpPlace-HelpPlace to store data.
5. Create query to get the help place where help place’s id equal to id-Integer.
6. Store the queried data in helpPlace-HelpPlace object.
7. Create query to delete help place where help place in database equal to helpPlace-HelpPlace.
8. Update Boolean equal to true.
9. Return Boolean.

**Return:** Boolean, if the help place has been deleted, will return true.

1. findById

public HelpPlace findById (Integer id)

**Description:** Get help place from the database by using session to query data.

**Parameter:** id-Integer uses for getting the help place.

**Flow:**

1. Create query to get help place where help place’s id in database equal to id-Integer, which is received in parameter.
2. Return the queried data.

**Return:** The help place that contains the inputted id, from database.

1. getHelpPlacesByCategory

public List<HelpPlace> getHelpPlacesByCategory(Integer categoryId)

**Description:** Get help places from the database by the selected category id and using session to query data.

**Parameter:** categoryId -Integer uses for getting the help place from database.

**Flow:**

1. Create list of help place to store data.
2. Create query to get help place where category’s id equal to categoryId.
3. Store the queried data in list.
4. Return lists of help places.

**Return:** List of the help places that contain the selected category, in database.

1. getHelpPlacesByProvince

public List<HelpPlace> getHelpPlacesByProvince(Integer provinceId)

**Description:** Get help places from the database by the selected province id and using session to query data.

**Parameter:** provinceId -Integer uses for getting the help place from database.

**Flow:**

1. Create list of help place to store data.
2. Create query to get help place where province’s id equal to provinceId.
3. Store the queried data in list.
4. Return the list of help places.

**Return:** List of help places that contain the selected province, in database.

1. getHelpPlacesByCategoryAndProvince

public List<HelpPlace> getHelpPlacesByCategoryAndProvince (Integer categoryId, Integer provinceId)

**Description:** Get help places from the database by selected category id and province id and using session to query data.

**Parameter:** provinceId –Integer and categoryId-Integer use for getting the help place from database.

**Flow:**

1. Create list of help place to store data.
2. Create query to get help place where province’s id equal to provinceId and category’s id equal to categoryId.
3. Store the queried data in list.
4. Return list of help places.

**Return:** List of help places that contain the selected category and province, from database.

1. HelpPlaceService



**Figure 9** HelpPlaceService **class**

**Method details**

1. getHelpPlaces

public List<HelpPlace> getHelpPlaces ()

**Description:** Get list of all help places from the database.

**Return:** List of all help places in database.

1. findById

public HelpPlace findById (Integer id)

**Description:** Get help place by id from the database.

**Parameter:** id-Integer uses for getting the help place.

**Return:** The help place in database.

1. updateHelpPlace

public HelpPlace updateHelpPlace(HelpPlace helpPlace)

**Description:** Update information of the help place to database.

**Parameter:** helpPlace-HelpPlace uses for update information of the help place to database.

**Return:** Help place object that needs to update.

1. deleteHelpPlace

public boolean deleteHelpPlace(HelpPlace helpPlace)

**Description:** Delete the help place in the database.

**Parameter:** helpPlace-HelpPlace uses for delete the help place from database.

**Return:** Boolean, if the help place has been deleted, will return true.

1. getHelpPlacesByCategory

public List<HelpPlace> getHelpPlacesByCategory(Integer categoryId)

**Description:** Get help places from the database by the selected category id.

**Parameter:** categoryId-Integer uses for getting the help place from database.

**Return:** List of help places that contain the selected category, in the database.

1. getHelpPlacesByProvince

public List<HelpPlace> getHelpPlacesByProvince(Integer provinceId)

**Description:** Get help places from the database by the selected province id.

**Parameter:** provinceId-Integer uses for getting the help place from database.

**Return:** List of help places that contain the selected province, in the database.

1. getHelpPlacesByCategoryAndProvince

public List<HelpPlace> getHelpPlacesByCategoryAndProvince (Integer categoryId, Integer provinceId)

**Description:** Get help places from the database by the selected category id and province id.

**Parameter:** provinceId–Integer and categoryId-Integer use for getting the help place from database.

**Return:** List of help places that contain the selected category and province, in database.

1. setHelpPlaceDAO

public void setHelpPlaceDAO(HelpPlaceDAO helpPlaceDAO)

**Description:** Set HelpPlaceDAO value into help place’s attribute.

**Parameter:** helpPlaceDAO–HelpPlaceDAO use for setting help place DAO.

1. HelpPlaceServiceImpl



**Figure 10** HelpPlaceServiceImpl **class**

**Entities**

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Name** | **Description** | **Type** |
| 1 | helpPlaceDAO | Contained HelpPlaceDAO object | HelpPlaceDAO |

**Method details**

1. getHelpPlaces

public List<HelpPlace> getHelpPlaces()

**Description:** Get help places from the database through getHelpPlaces method of HelpPlaceDAO class.

**Flow:**

1. Use method getHelpPlaces()of HelpPlaceDAO class to get all help places.
2. Return list of all help places.

**Return:** List all help places in the database.

1. updateHelpPlace

public HelpPlace updateHelpPlace(HelpPlace helpPlace)

**Description:** Update information of the help place to the database through updateHelpPlace method of HelpPlaceDAO class.

**Parameter:** helpPlace-HelpPlace uses for update information of the help place to the database.

**Flow:**

1. Use method updateHelpPlace(helpPlace)of HelpPlaceDAO class to update help place.

**Return:** Help place object that needs to update.

1. deleteHelpPlace

public boolean deleteHelpPlace(HelpPlace helpPlace)

**Description:** Delete the help place from database through deleteHelpPlace method of HelpPlaceDAO class.

**Parameter:** helpPlace-HelpPlace uses for delete the help place from database.

**Flow:**

1. Use method deleteHelpPlace(helpPlace)of HelpPlaceDAO class to delete help place.

**Return:** Boolean, if the help place has been deleted, will return true.

1. findById

public HelpPlace findById(Integer id)

**Description:** Get help place from the database through findById method of HelpPlaceDAO class.

**Parameter:** id-Integer uses for getting the help place.

**Flow:**

1. Use method findById(id)of HelpPlaceDAO class to find the help place by using its id.

**Return:** The help place in the database.

1. getHelpPlacesByCategory

public List<HelpPlace> getHelpPlacesByCategory(Integer categoryId)

**Description:** Get help places from the database by selected category id through getHelpPlacesByCategory method of HelpPlaceDAO class.

**Parameter:** categoryId-Integer uses for getting the help place from database.

**Flow:**

1. Use method getHelpPlacesByCategory(categoryId)of HelpPlaceDAO class to get list of help places.
2. Return list of help places.

**Return:** List of help places in the database.

1. getHelpPlacesByProvince

public List<HelpPlace> getHelpPlacesByProvince(Integer provinceId)

**Description:** Get help places from the database by selected province id through getHelpPlacesByProvince method of HelpPlaceDAO class.

**Parameter:** provinceId-Integer uses for getting the help place from database.

**Flow:**

1. Use method getHelpPlacesByProvince(provinceId) of HelpPlaceDAO class to get list of help places.
2. Return the list of help places.

**Return:** List of help places in the database.

1. getHelpPlacesByCategoryAndProvince

public List<HelpPlace> getHelpPlacesByCategoryAndProvince (Integer categoryId,Integer provinceId)

**Description:** Get help places from the database by selected category id and province id through getHelpPlacesByCategoryAndProvince method of HelpPlaceDAO class.

**Parameter:** provinceId–Integer and categoryId-Integer uses for getting the help place from database.

**Flow:**

1. Use method getHelpPlacesByCategoryAndProvince(categoryId,provinceId)of HelpPlaceDAO class to get list of help places.
2. Return the list of help places.

**Return:** List of help places in the database.

1. setHelpPlaceDAO

public void setHelpPlaceDAO(HelpPlaceDAO helpPlaceDAO)

**Description:** Set HelpPlaceDAO value into help place’s attribute.

**Flow:**

1. Get parameter to set value of helpPlaceDAO attribute.

**Parameter:** helpPlaceDAO–HelpPlaceDAO use for setting help place DAO.

1. Category



**Figure 11** Category **class**

**Entities**

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Name** | **Description** | **Type** |
| 1 | id | stored id of the category | Integer |
| 2 | name | stored name of the category | string |

**Method details**

1. Category

public Category(String name)

**Description:** A constructor method is used to set name of the category.

1. CategoryDAO



**Figure 12** CategoryDAO **class**

**Method details**

1. getCategories

public List<Category> getCategories()

**Description:** Get all categories from the database.

**Return:** List of all categories in the database.

1. getCategoryById

public Category getCategoryById(Integer id)

**Description:** Get the category, which category’s id equal to id-Integer, from the database.

**Parameter:** id-Integer uses for getting the category from the database.

**Return:** The category in the database.

1. CategoryDAOImpl



**Figure 13** CategoryDAOImpl **class**

**Entities**

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Name** | **Description** | **Type** |
| 1 | sessionFactory | A thread-safe, immutable cache of compiled mappings for a single database | SessionFactory |

**Method details**

1. getCategories

public List<Category> getCategories()

**Description:** Get all categories from the database by using session to query data.

**Flow:**

1. Create query to get all categories in the database.
2. Return the queried data

**Return:** List of categories in the database.

1. getCategoryById

public Category getCategoryById(Integer id)

**Description:** Get the category from the database by using session to query data.

**Parameter:** id-Integer uses for getting the category from the database.

**Flow:**

1. Create query to get the category, where category’s id equal to id-Integer.
2. Return the queried data.

**Return:** The category in the database.

1. CategoryService



**Figure 14** CategoryService **class**

**Method details**

1. getCategories

public List<Category> getCategories()

**Description:** Get all categories from the database.

**Return:** List of all categories in the database.

1. getCategoryById

public Category getCategoryById(Integer id)

**Description:** Get the category from the database by using id-Integer.

**Parameter:** id-Integer uses for getting the category from the database.

**Return:** The category in the database.

1. setCategoryDAO

public void setCategoryDAO(CategoryDAO categoryDAO)

**Description:** Set categoryDAO value into category’s attribute.

**Parameter:** categoryDAO–CategoryDAO use for setting category DAO.

1. CategoryServiceImpl



**Figure 15** CategoryServiceImpl **class**

**Method details**

1. getCategories

public List<Category> getCategories()

**Description:** Get all categories from the database through getCategories method of CategoryDAO class.

**Flow:**

1. Use method getCategories()of CategoryDAO class to get a list of all categories.
2. Return the list of all categories.

**Return:** List of all categories in the database.

1. getCategoryById

public Category getCategoryById(Integer id)

**Description:** Get the category from the database through getCategoryById method of CategoryDAO class.

**Parameter:** id-Integer uses for getting the category from the database.

**Flow:**

1. Use method getCategoryById(Integer id)of CategoryDAO class to get the category.
2. Return the category.

**Return:** The category in the database.

1. setCategoryDAO

public void setCategoryDAO(CategoryDAO categoryDAO)

**Description:** Set categoryDAO value into category’s attribute.

**Flow:**

1. Use parameter to set categoryDAO.

**Parameter:** categoryDAO–CategoryDAO use for setting category DAO.

1. Province



**Figure 16** Province **class**

**Entities**

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Name** | **Description** | **Type** |
| 1 | id | stored id of the province | Integer |
| 2 | name | stored name of the province | string |

**Method details**

1. Province

public Province(String name)

**Description:** A constructor method is used to set name of the province.

1. ProvinceDAO



**Figure 17** ProvinceDAO **class**

**Method details**

1. getProvinces

public List<Province> getProvinces()

**Description:** Get all provinces from the database.

**Return:** List of all provinces in database.

1. getProvinceById

public Province getProvinceById(Integer id)

**Description:** Get province from the database.

**Parameter:** id-Integer uses for getting the province from database.

**Return:** The province in database.

1. ProvinceDAOImpl



**Figure 18** ProvinceDAOImpl **class**

**Entities**

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Name** | **Description** | **Type** |
| 1 | sessionFactory | A thread-safe, immutable cache of compiled mappings for a single database | SessionFactory |

**Method details**

1. getProvinces

public List<Province> getProvinces();

**Description:** Get all provinces from the database by using session to query data.

**Flow:**

1. Create query to get all provinces in database.
2. Return the queried data.

**Return:** List of all provinces in database.

1. getProvinceById

public Province getProvinceById(Integer id)

**Description:** Get province from the database by using session to query data.

**Parameter:** id-Integer uses for getting the province from database.

**Flow:**

1. Create query to get the province, where province’s id equal to id-Integer.
2. Return the queried data.

**Return:** The province in database.

1. ProvinceService



**Figure 19** ProvinceService **class**

**Method details**

1. getProvinces

public List<Province> getProvinces();

**Description:** Get all provinces from the database

**Return:** List of all provinces in database.

1. getProvinceById

public Province getProvinceById(Integer id)

**Description:** Get province from the database.

**Parameter:** id-Integer uses for getting the province from database.

**Return:** The province in database.

1. setProvinceDAO

public void setProvinceDAO(ProvinceDAO provinceDAO)

**Description:** Set provinceDAO value into province’s attribute.

**Parameter:** provinceDAO–ProvinceDAO use for setting province DAO.

1. ProvinceServiceImpl



**Figure 20** ProvinceServiceImpl **class**

**Method details**

1. getProvinces

public List<Province> getProvinces()

**Description:** Get provinces from the database through getProvinces method of ProvinceDAO class.

**Flow:**

1. Use method getProvinces()of ProvinceDAO class to get a list of all provinces.
2. Return the list of all provinces.

**Return:** List of provinces in database.

1. getProvinceById

public Province getProvinceById(Integer id)

**Description:** Get province from the database through getProvinceById method of ProvinceDAO class.

**Parameter:** id-Integer uses for getting the province from database.

**Flow:**

1. Use method getProvinceById(Integer id)of ProvinceDAO class to get the province.
2. Return the province.

**Return:** The province in database.

1. setProvinceDAO

public void setProvinceDAO(ProvinceDAO provinceDAO)

**Description:** Set provinceDAO value into province’s attribute.

**Flow:**

1. Use parameter to set provinceDAO.

**Parameter:** provinceDAO–ProvinceDAO use for setting province DAO.

1. HelpPlaceController



**Figure 21** HelpPlaceController **class**

**Entities**

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Name** | **Description** | **Type** |
| 1 | helpPlaceService | Contained HelpPlaceService object | HelpPlaceService |
| 2 | provinceService | Contained ProvinceService object | ProvinceService |
| 3 | categoryService | Contained CategoryService object | CategoryService |

**Method details**

1. listHelpPlaces

public String listHelpPlaces(Model model)

**Description:** Show a list of help places in the home page.

**Parameter:** model-Model uses for getting the model.

**Return:** Home\_page.jsp page

1. getHelpPlaceByCategoryAndProvince

public String getHelpPlaceByCategoryAndProvince

(@PathVariable("input")Integer catId,@PathVariable("input2")Integer proId, ModelMap model)

**Description:** Show a list of help places by its category, province, or category and province.

**Parameter:** model-ModelMap uses for getting the model.

proId-Integerusesfor getting help places by province’s id.

catId-Integerusesfor getting help places by category’s id.

**Return:** Home\_page.jsp page

1. getHelpPlaceById

public String getHelpPlaceById

(@PathVariable("input")Integer id,ModelMap model)

**Description:** Show information of the selected help place on view information page.

**Parameter:** model-ModelMap uses for getting the model.

id-Integerusesfor getting help place by its id.

**Return:** viewInformation.jsp page

1. addHelpPlace

public String addHelpPlace(Model model)

**Description:** Show add help place page with empty boxes to input information of new help place**.**

**Parameter:** model-Model uses for getting the model.

**Return:** updateInfo.jsp page

1. updateHelpPlace

public String updateHelpPlace(@PathVariable("id") Integer id, Model model)

**Description:** Show updates information page with boxes for inputting new information.

**Parameter:** model-Model uses for getting the model.

id-Integerusesfor getting help place by its id.

**Return:** updateInfo.jsp page

1. removeHelpPlace

public String removeHelpPlace(@PathVariable("id") Integer id,RedirectAttributes redirectAttributes)

**Description:** Create command to delete the help place and create the confirmation message for removing.

**Parameter:** model-Model uses for getting the model.

id-Integerusesfor deleting help place by its id.

**Return:** Home\_page.jsp page.

1. addValidHelpPlace

public String addValidHelpPlace(@Valid HelpPlace helpPlace, BindingResult bindingResult,Model model,RedirectAttributes redirectAttributes)

**Description:** Check the validation of the help place before add or update into the database.

**Parameter:** model-Model uses for getting the model.

helpPlace-HelpPlaceuses for updating the help place.

bindingResult-BindingResultuses for keeping the invalid data.

redirectAttributes-RedirectAttributesuses for store the error message.

**Return:** If there are invalid information, go to updateInfo.jsp page, otherwise go to Home\_page.jsp page.

3.2.2 Mobile Part

1. HelpPlace



**Figure 22 HelpPlace class**

**Entities**

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Name** | **Description** | **Type** |
| 1 | id | Store id of help place | String |
| 2 | name | Store name of help place | String |
| 3 | address | Store address of help place | String |
| 4 | telephone | Store telephone of help place | String |
| 5 | latitude | Store latitude of help place | double |
| 6 | longitude | Store longitude of help place | double |

1. HelpPlaceService



**Figure 23 HelpPlaceService class**

**Method details**

1. getJsonObjByURL

public JSONObject getJsonObjByURL (String url)

**Description:** Get Json object by use url of server.

**Parameters:** url-string uses to get Json object file from server.

**Return:** Json object, if invalid input returns null

1. getHelpPlacesByJsonOBJ

public JSONArray getHelpPlacesByJsonOBJ (JSONObject jsonOBJ)

**Description:** Get Json array by use Json object that receive from getJsonObjByURL method.

**Parameters:** jsonOBJ-JSONObject uses to get Json array from getJsonObjByURL method.

**Return:** Json array, if invalid input returns null

1. getHelpPlacesOnDevice

public HelpPlace[]getHelpPlacesFromDevice ()

**Description:** Get help places from the user’s device that saved in file sqlite.

**Return:** HelpPlace array, if invalid input returns default sqlite file.

1. setHelpPlacesOnDevice

public void setHelpPlacesToDB (ArrayList <HelpPlace> helpPlaceList)

**Description:** Set help places into database of user’s device by input ArrayList for setting value of help place in offline map.

**Parameters:** helpPlaceList-ArrayList uses to set help place saved into sqlite file on database of user’s device.

1. HelpPlaceServiceImpl



**Figure 24 HelpPlaceServiceImpl class**

**Entities**

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Name** | **Description** | **Type** |
| 1 | jParser | Store JSON parser | JSONParser |
| 2 | jasonObj | Store JSON Object | JSONObject |
| 3 | helpPlacesArray | Store help places as JSON array | JSONArray |
| 4 | helpPlacesInDevice | Store help places to array | HelpPlace [ ] |

**Method details**

1. getJsonObjByURL

public JSONObject getJsonObjByURL (String url)

**Description:** Get Json object by use url of server.

**Parameters:** url-string use to get Json object file from server.

**Flow:**

1. Set JSONObject equal to jParser to call getJSONFromUrl method and use url parameter.
2. Return jsonObj.

**Return:** Json object, if invalid input returns null

1. getHelpPlacesByJsonOBJ

public JSONArray getHelpPlacesByJsonOBJ (JSONObject jsonOBJ)

**Description:** Get Json array by use Json object that receive from getJsonObjByURL method.

**Parameters:** jsonOBJ-JSONObject use to get Json array from getJsonObjByURL method.

**Flow:**

1. Set helpPlaceArray variable equal to jsonOBJ to call getJSONArray method.
2. Return helpPlaceArray.

**Return:** Json array, if invalid input returns null

1. getHelpPlacesOnDevice

public HelpPlace[]getHelpPlacesFromDevice ()

**Description:** Get help places from the user’s device that saved in file sqlite.

**Flow:**

1. Set for returning value of helpPlaceInDevice.

**Return:** HelpPlace array, if invalid input returns default sqlite file.

1. setHelpPlacesOnDevice

public void setHelpPlacesToDB (ArrayList <HelpPlace> helpPlaceList)

**Description:** Set help places into database of user’s device by input ArrayList for setting value of help place in offline map.

**Parameters:** helpPlaceList-ArrayList use to set help place saved into sqlite file on database of user’s device.

**Flow:**

1. Set for condition to set value into an array by use parameter array list.

1. HelpPlaceSingleton



**Figure 25 HelpPlaceSingleton class**

**Entities**

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Name** | **Description** | **Type** |
| 1 | helpPlace | Store HelpPlaceService object | HelpPlaceService |

**Method details**

1. getHelpPlaceService

public HelpPlaceService getHelpPlaceService ()

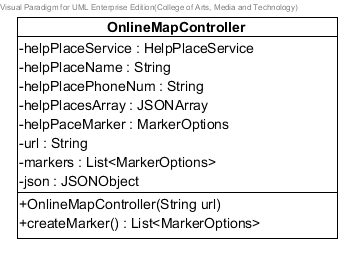
**Description:** Get HelpPlaceService with static value.

**Flow:**

1. Set helpPlace attribute to equal new HelpPlaceServiceImpl ( )
2. Return helpPlace.

**Return:** helpPlace object, if invalid input returns null.

1. OnlineMapController



**Figure 26 OnlineMapController class**

**Entities**

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Name** | **Description** | **Type** |
| 1 | helpPlaceService | Create help place service object | HelpPlaceService |
| 2 | helpPlaceName | Store name of help place | String |
| 3 | helpPlacePhoneNum | Store telephone number of help place | String |
| 4 | helpPlacesArray | Store help place in Json array | JSONArray |
| 5 | helpPlaceMarker | Create markerOptions object | MarkerOptions |
| 6 | url | Store url of server | String |
| 7 | markers | Store markers in List of MarkerOptions | List<MarkerOptions> |
| 8 | json | Store Json object | JSONObject |

**Method details**

1. OnlineMapController

public OnlineMapController(String url)

**Description:** a constructor method uses to set url attribute.

**Parameters:** url-String uses for getting the url for setting into service.

**Flow:**

1. A constructor method gets url from parameter.
2. Create object helpPlaceService.
3. Define value to json attribute equal to helpPlaceService.getJsonObjByURL(url)
4. createMarker

public List<MarkerOptions> createMarker ()

**Description:** Create marker with information of help place.

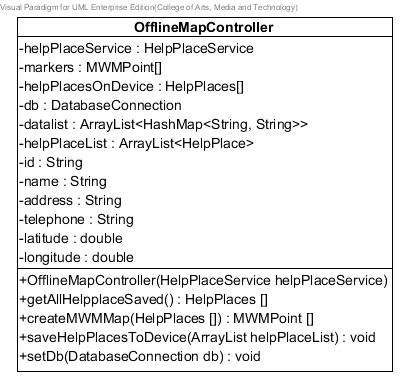
**Parameters:** None

**Flow:**

1. Set value into helpPlacesArray variable by use getHelpPlacesByJsonOBJ method of HelpPlaceServiceImpl
2. Set string json variable to equal helpPlacesArray to use getJSONObject method
3. Set value to string helpPlaceName, helpPlacePhoneNum by use json variable to getString method.
4. Create createMarker method type List<MarkerOptions> to set information of help place into each marker.
5. Add helpPaceMarker variable that saved information into the list of MarkerOptions.

**Return:** MarkerOption in an array list, if invalid input returns null.

1. OffllineMapController



**Figure 27 OfflineMapController class**

**Entities**

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Name** | **Description** | **Type** |
| 1 | helpplaceService | Create help place service ohject | HelpPlaceService |
| 2 | markers | Store MWMPoint object | MWMPoint [ ] |
| 3 | helpPlacesOnDevice | Store help place array | HelpPlaces [ ] |
| 4 | db | Store object class DatabaseConnection | DatabaseConnection |
| 5 | datalist | Store list of data in ArrayList | ArrayList<HashMap<String, String>> |
| 6 | helpPlaceList | Store list of help place in ArrayList | ArrayList<HelpPlace> |
| 7 | id | Store id of help place | String |
| 8 | name | Store name of help place | String |
| 9 | address | Store address of help place | String |
| 10 | telephone | Store telephone of help place | String |
| 11 | latitude | Store latitude of help place | double |
| 12 | longitude | Store longitude of help place | double |

**Method details**

1. OfflineMapController

public OfflineMapController (HelpPlaceService helpPlaceService)

**Description:** a constructor method uses to set helpPlaceService attribute.

**Parameters:** hPlaces-HelpPlaces use to set value to an array.

**Flow:**

1. Use this.helpPlaceService attribute equal to helpPlaceService parameter.
2. getAllHelpplaceSaved

public HelpPlaces[] getAllHelpplaceSaved ()

**Description:** Get help places from database of user’s device.

**Parameters:** None

**Flow:**

1. Initial value of helpPlacesOnDevice variable by use method getHelpplaceFromDevice of HelpPlaceServiceImpl.
2. Return help place array.

**Return:** HelpPlaces array, if invalid value returns null.

1. createMWMMap

public MWMPoint[]createMWMMap(HelpPlaces []hPlaces)

**Description:** Set help place marker to array.

**Parameters:** hPlaces-HelpPlaces use to set value to an array.

**Flow:**

1. Initial value of marker variable by new object of MWMPoint. ]
2. Return marker of help place in array.

**Return:** MWMPoint array, if invalid input returns null.

1. saveHelpPlacesToDevice

public MWMPoint[]saveHelpPlacesToDevice(ArrayList <HelpPlace> helpPlaceList)

**Description:** Set help place marker to array.

**Parameters:** helpPlaceList-ArrayList<HelpPlace> uses to set value into setHelpPlacesToDB method of HelpPlaceServiceImpl.

**Flow:**

1. Initial value of marker variable by new object of MWMPoint.
2. Return marker of help place in array.
3. setDb

public void setDb(DatabaseConnection db)

**Description:** Set database in OfflineMapController class.

**Parameters:** db-DatabaseConnection uses to set value to equal the attribute of DatabaseConnection object.

**Flow:**

1. Receive object in parameter
2. Set value to equal the attribute of DatabaseConnection object.
3. InformationView



**Figure 28 InformationView class**

**Entities**

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Name** | **Description** | **Type** |
| 1 | EXTRA\_FROM\_MWM | Store string “from-maps-with-me” | String |
| 2 | telNumOffline | Store telephone number of offline map | String |
| 3 | telNumOnline | Store telephone number of online map | String |
| 4 | callButton | Store value from telephone number to phone call button | Button |
| 5 | txtName | Store name of help place | TextView |
| 6 | txtAddress | Store address of help place | TextView |
| 7 | txtLatitude | Store latitude of help place | TextView |
| 8 | txtLongitude | Store longitude of help place | TextView |
| 9 | name | Store name of help place | String |
| 10 | address | Store address of help place | String |
| 11 | telephone | Store telephone of help place | String |
| 12 | latitude | Store latitude of help place | double |
| 13 | longitude | Store longitude of help place | double |

**Method details**

1. getPendingIntent

public PendingIntent getPendingIntent (Context context)

**Description:** Set pending intent value.

**Parameters:** Context-Context uses for input as a parameter to create Intent object.

**Flow:**

1. Create Intent object to use InforamtionController.class as parameter.
2. Set intent to use putExtra method and input *EXTRA\_FROM\_MWM* as parameter.
3. Return PendingIntent to use getActivity method.

**Return:** PendingIntent, if invalid input returns null.

1. handleIntent

public void handleIntent(Intent intent)

**Description:** Set information of help place that selected by handle.

**Parameters:** intent-Intent use to get and set value.

**Flow:**

1. Use intent to getBooleanExtra method in if condition
2. Create MWMResponse object for using extractFromIntent method.
3. Set helpPlace variable to equal HelpPlaceServiceImpl.*fromMWMPoint*(response.getPoint());
4. Check helpPlace not null to set text into Text View.

1. JSONParser



**Figure 29 JSONParser class**

**Entities**

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Name** | **Description** | **Type** |
| 1 | is | Store value form data source | InputStream |
| 2 | jObj | Store Json object | JSONObject |
| 3 | json | Use for creating new object of JSONObject | String |

**Method details**

1. getJSONFromUrl

public JSONObject getJSONFromUrl (String url)

**Description:** Get Json object by receive url of server.

**Parameters:** url-string uses to get url of server.

**Flow:**

1. Create object of DefaultHttpClient, HttpPost, HttpResponse, HttpEntity and set value to attribute “is” to use HttpEntity to get Content.
2. Use BufferReader to read InputStreamReader.
3. Keep value from readLine into StringBuilder append.

**Return:** Json object, if invalid input returns null.

1. PhoneCallListener



**Figure 30 PhoneCallListener class**

**Entities**

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Name** | **Description** | **Type** |
| 1 | LOG\_TAG | Use for test Log status | String |
| 2 | isPhoneCalling | Use to check state of TelephonyManager | boolean |

**Method details**

1. onCallStateChanged

public void onCallStateChanged (int state,String incomingNumber)

**Description:** Set state of the phone call.

**Parameters:** state-integer uses check compare with TelephonyManager state.

incomingNumber-String uses to set telephone number.

**Flow:**

1. Defind condition to check TelephonyManager state (CALL\_STATE\_RINGING, CALL\_STATE\_OFFHOOK, CALL\_STATE\_IDLE)

1. DatabaseConnection



**Figure 31 DatabaseConnection class**

**Entities**

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Name** | **Description** | **Type** |
| 1 | DATABASE\_PATH | Store path of file SQLite | String |
| 2 | DATABASE\_NAME | Store name of file SQLite | String |
| 3 | TABLE\_NAME | Store name of table in database | String |
| 4 | DATABASE\_VERSION | Store number version of database | Integer |
| 5 | dbSQLite | Store command to use database | SQLiteDatabase |
| 6 | context | Store context to call getAssets() | Context |

**Method details**

1. selectDatabase

public void selectDatabase ()

**Description:** Select database to use getWritableDatabase method to set dbSQLite.

**Parameters:** None

**Flow:**

1. Set value into dbSQLite variable equal to this.getWritableDatabase method.
2. copyDatabase

public void copyDatabase ()

**Description:** Copying database from resource SQLite.

**Parameters:** None

**Flow:**

1. Create variable InptStream and OutputStream are null.
2. Use context to call getAsset method to open database.
3. Use while condition for InputStream reading buffer
4. getHelpPlaces

public ArrayList<HashMap<String,String>> getHelpPlaces ()

**Description:** Copying database from resource SQLite.

**Parameters:** None

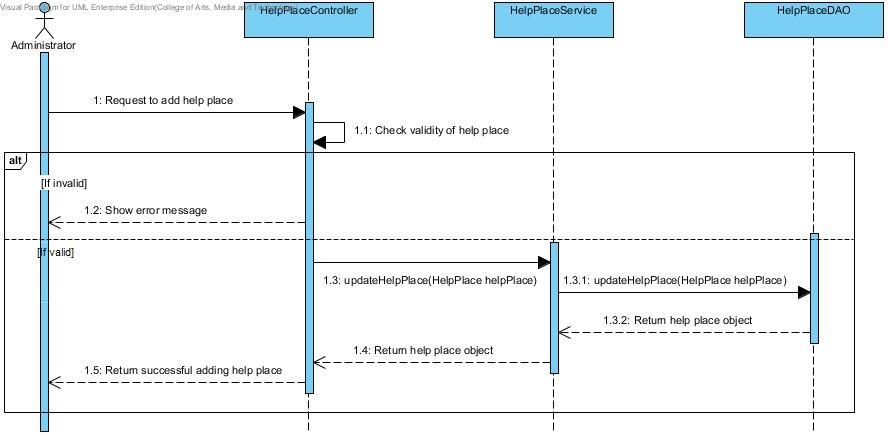
**Flow:**

1. Write command to get information from database “Select \* from (name of database)”
2. Set value to some variable by use cursor getString method for putting into HashMap.
3. Put HashMap into an array list.
4. Return an array list.

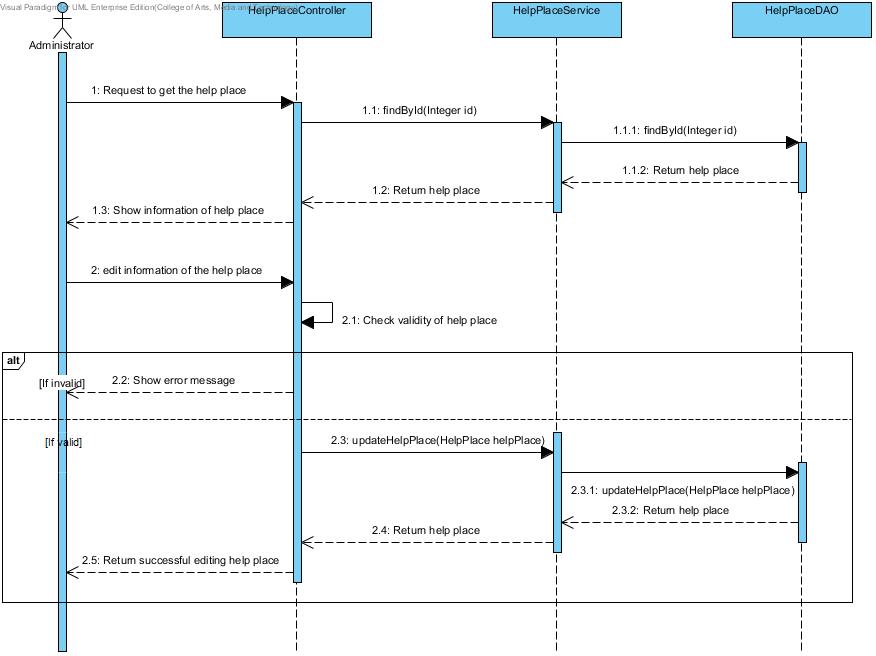
**Return:** Array list, if invalid value returns null.

## 3.3 Sequence Diagram

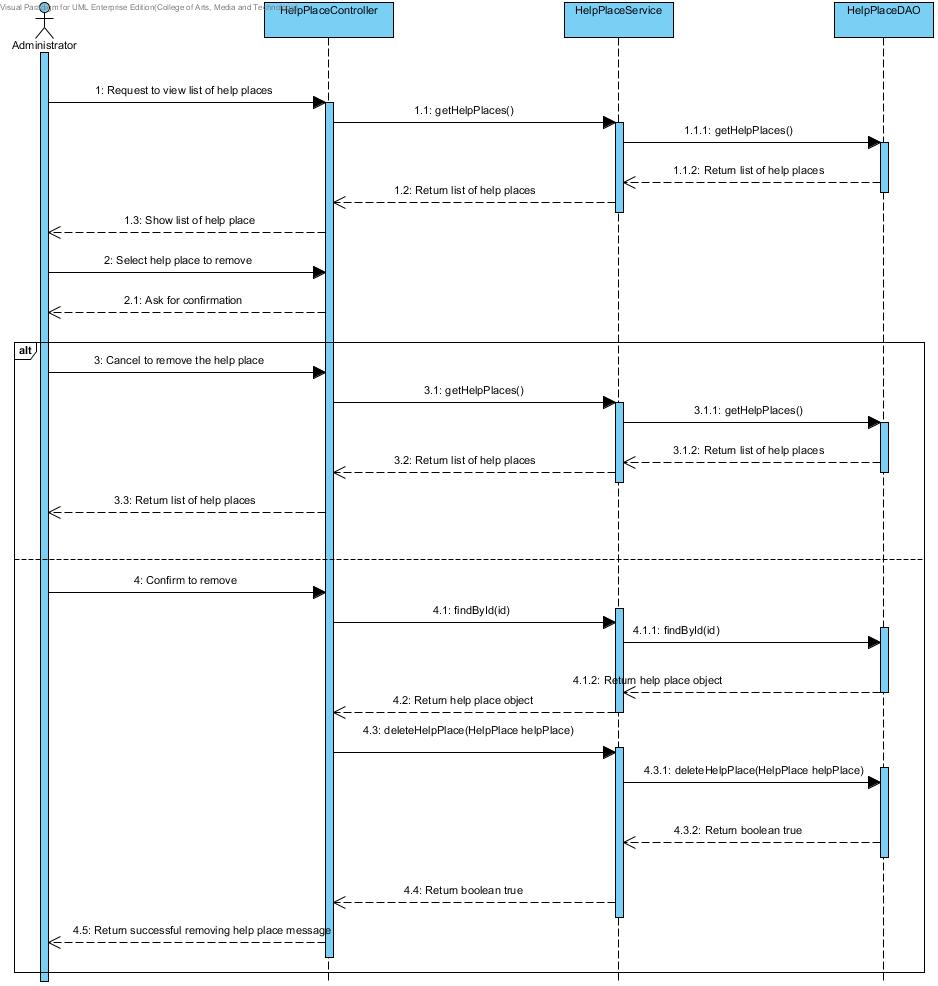
3.3.1 Server Part

****

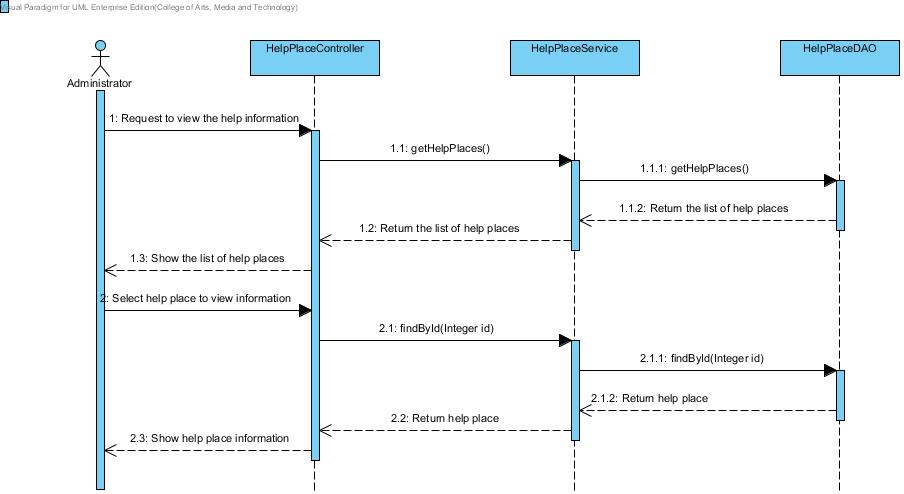
**Figure 32 Add help place sequence diagram (SD-01)**

****

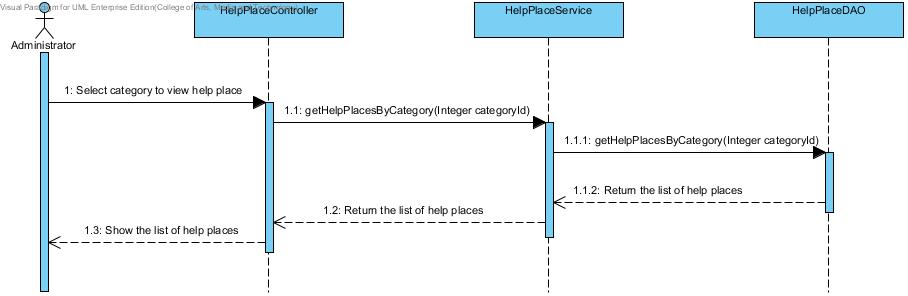
**Figure 33 Edit help place sequence diagram (SD-02)**

****

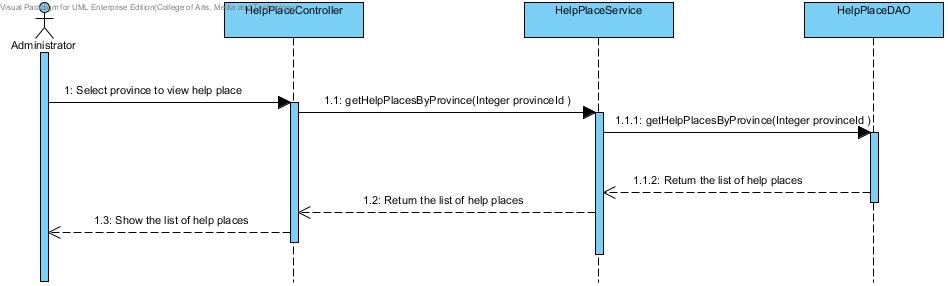
**Figure 34 Remove help place sequence diagram (SD-03)**

****

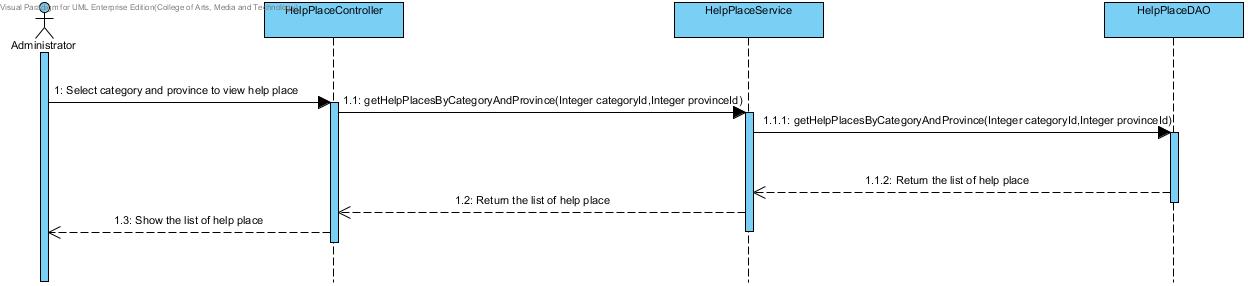
**Figure 35 View information sequence diagram (SD-04)**

****

**Figure 36 Search by category sequence diagram (SD-05)**

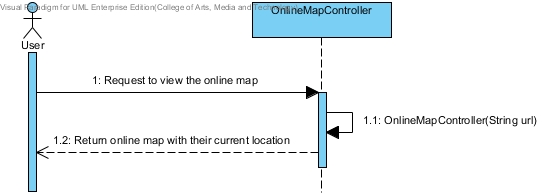
****

**Figure 37 Search by province sequence diagram (SD-06)**

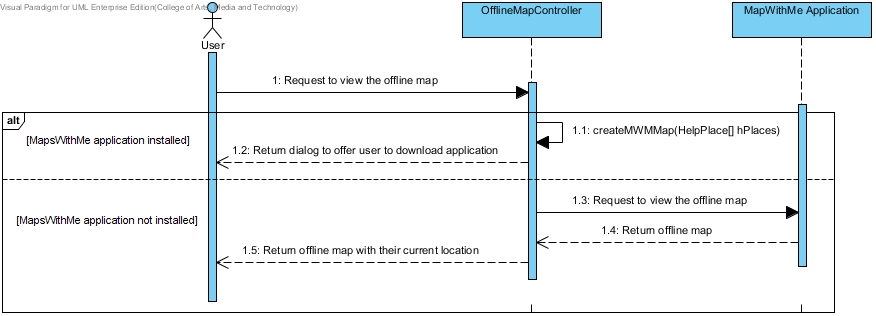
****

**Figure 38 Search by category and province sequence diagram (SD-07)**

**3.3.2 Mobile Part**

****

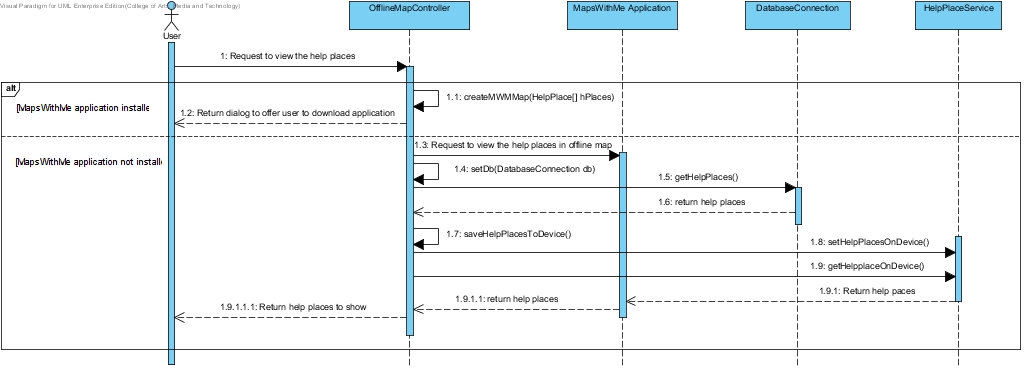
**Figure 39 View online map diagram (SD-08)**



**Figure 40 View offline map diagram (SD-09)**

****

**Figure 41 View the help places in online map diagram (SD-10)**

****

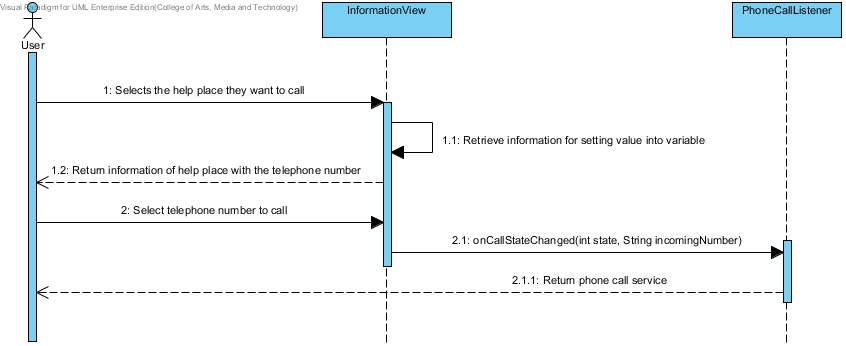
**Figure 42 View the help places in offline map diagram (SD-11)**

****

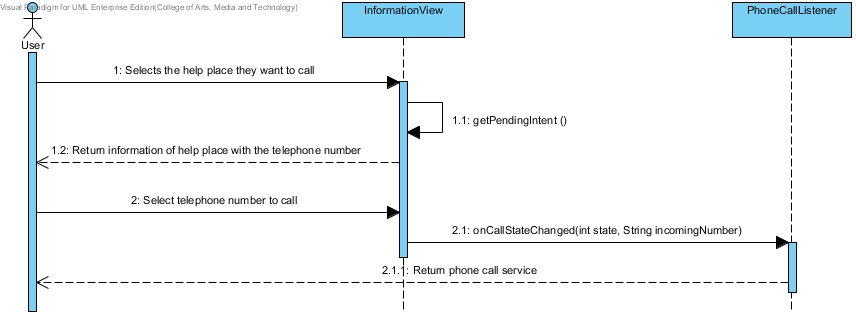
**Figure 43 View information of each help place in online map diagram (SD-12)**

****

**Figure 44: View information of each help place in offline map diagram (SD-13)**

****

**Figure 45 Make emergency call to each help place in online map diagram (SD-14)**

****

**Figure 46 Make emergency call to each help place in offline map diagram (SD-15)**

# Chapter Four | User Interface Design

**UI name:** Home Page(UI-01)

Include Requirement: UC-03, UC-05, UC-06, UC-07



**Figure 47 Home Page (UI-01)**

The home page includes 3 components. The top of the page is a logo of web page. Next, “Add Help Place” button for adding new help place, and “Search” button for searching by selected category and province. The center of the page is a list of help places with “Edit” and “Remove” button. The help place will link with its information page.

**UI name:** Remove Confirm Dialog(UI-02)

Include Requirement: UC-03



**Figure 48 Remove Confirm Dialog (UI-02)**

The messages dialog shows the confirmation message when administrator try to remove a help place.

**UI name:** SuccessfullyRemove Dialog(UI-03)

Include Requirement: UC-03



**Figure 49 Remove Confirm Dialog (UI-03)**

The messages dialog shows the message after administrator removes help place successfully.

**UI name:** Update Information Page(UI-04)

Include Requirement: UC-01, UC-02



**Figure 50 Edit Information Error Message (UI-04)**

The update information page includes 3 components. The top of the page is a logo of the web page. The center of the page is map and search box which are used for getting latitude and longitude. The last component, the bottom of the page is a text box for receiving new information from administer with the “Add” button.An error message will be shown when information are not passing the condition.

**UI name:** Successfully Add Dialog(UI-05)

Include Requirement: UC-01, UC-02



**Figure 51 Remove Confirm Dialog (UI-05)**

The messages dialog shows the message after administrator adds or updates help place successfully.

**UI name:** View Information Page(UI-06)

Include Requirement: UC-04



**Figure 52 View Information Page (UI-06)**

The view information page includes 2 components. The top of the page is a logo of the web page. The center of the page shows information of a help place.

**UI name:** Start Page (UI-07)

Include Requirement: UC-08, UC-09, UC-10



**Figure 53 Start Page (UI-07)**

The start page shows the online map automatic and buttons for user to switching map.

• Online map

• Offline map

**UI name:** Show Online map (UI-08)

Include Requirement: UC-08, UC-10



**Figure 54 Show Online map (UI-08)**

The switcher tab online button shows the online map for automatic on Start page. The online map will provide search bar to user search help place on the online map. The button allows users to access the online map.

**UI name:** Connect Offline map (UI-09)

Include Requirement: UC-09



**Figure 55 Connect Offline map (UI-09)**

The switcher tab offline button shows the offline map for Start page. The button allows users to access the offline map.

**UI name:** Show Offline map (UI-10)

Include Requirement: UC-09, UC-11



**Figure 56 Show Offline map (UI-10)**

MapsWithMe application will show Thailand offline map. The offline map will show automatic when the application lost internet connection.

**UI name:** MapsWithMe application installed (UI-11)

Include Requirement: UC-09, UC-11



**Figure 57 MapsWithMe application installed (UI-11)**

What will happen if the user call for offline map but MapsWithMe application is not installed? The API library will show a simple dialog with a gentle offer to download MapsWithMe.

**UI name:** MapsWithMe application (UI-12)

Include Requirement: UC-11, UC-13



**Figure 58 MapsWithMe application (UI-12)**

MapsWithMe application will start with show offline map. The user selects the help place on offline map to view information. MapsWithMe provides latitude, longitude, name and show more info button.

**UI name:** Show Information Page (UI-13)

Include Requirement: UC-12,UC-13, UC-14, UC-15



**Figure 59 Show Information Page (UI-13)**

The Show Information page will show latitude, longitude, address, province, zip code and call button. On show information page, the user can call to help place directly from the application.