Chapter 4

Software Design

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 • CD 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.1.0.docx	**Modify Chapter 3** • Class Diagram • Class Diagram Description • Sequence Diagram **Modify Chapter 4** • UI Design	Release	10/5/2014	PJ, SK, CD	PJ, SK	PJ, SK
EIOM-SDD-V.1.1.docx	• Modify Chapter 1-4	Release	30/7/2014	PJ, SK, CD	PJ, SK	PJ, SK
EIOM-SDD-V.2.0.docx	• Modify feature5,CDs-04,05,16 • Add MDs-61 – MDs-73 • Add MDm-24 – MDm-51	Release	21/10/2014	PJ, SK, CD	PJ, SK	PJ, SK
EIOM-SDD-V.2.1.docx	• Modify system architecture(Add feature 6)	Release	11/11/2014	PJ, SK, CD	PJ, SK	PJ, SK
EIOM-SDD-V.3.0.docx	• Add MDs, MDm, UI and update class diagram	Release	11/12/2014	PJ, SK, CD	PJ, SK	PJ, SK

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

Table of Contents

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, II, and III. 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
SRSs	Software Requirement Specification Server
SRSm	Software Requirement Specification Mobile
URSs	User Requirement Specification Server
URSm	User Requirement Specification Mobile
UCs	Use Case Server
UCm	Use Case Mobile
UIs	User Interface Server
UIm	User Interface Mobile
UTCs	Unit Test Case Server
UTCm	Unit Test Case Mobile

1.4.2 Definitions

Sequence diagram	A sequence diagram in a Unified Modeling Language (UML) is a kind of 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

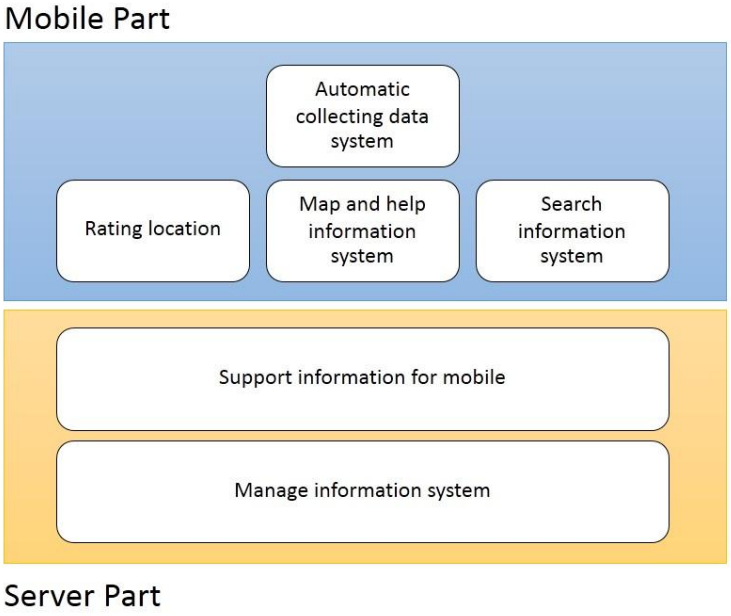

Figure 1 Architecture overview of Emergency Information on Mobile system

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
Feature 1: Map and help information system	View map with their current location.	√	√
	View the help places	√	√
	View help information of each help place	√	√
	Make emergency call	√	√
	View the route of distance between the current locations of user to the destination	√	
	view details of each point on routing the direction	√	
Feature 2: Search	Search help place's name by keyword	√	

information system	Find the nearest help place by selection the category	√	√
Feature 3: Rating location	Rate the help place	√	
	View average rating score of each help place	√	√
Feature 4: Automatic collecting data system	Download data of help place automatically	√	
	Set the scope for downloading data	√	
Feature 5: 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	√	
	Login to the system	√	
	Logout of the system	√	
	Update account's password	√	
Feature 6: Support information for mobile	Sent nearest help place in JSON form	√	
	Sent list of all help places in JSON form	√	
	Sent list of all help places in setting scope in JSON form	√	
	Retrieve new average rating score	√	

Mobile Part

Feature 1: 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.

Feature 2: 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.

Feature 3: 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.

Feature 4: 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

Feature 5: 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. In addition, the server supports information of help place in JSON form to the mobile application.

Feature 6: Support information for mobile

Feature 6 will build information in form of JSON to support the mobile application.

Chapter Three | Detailed Design

3.1 Class Diagram

3.1.1 Mobile Part

Emergency Information on Mobile class diagram

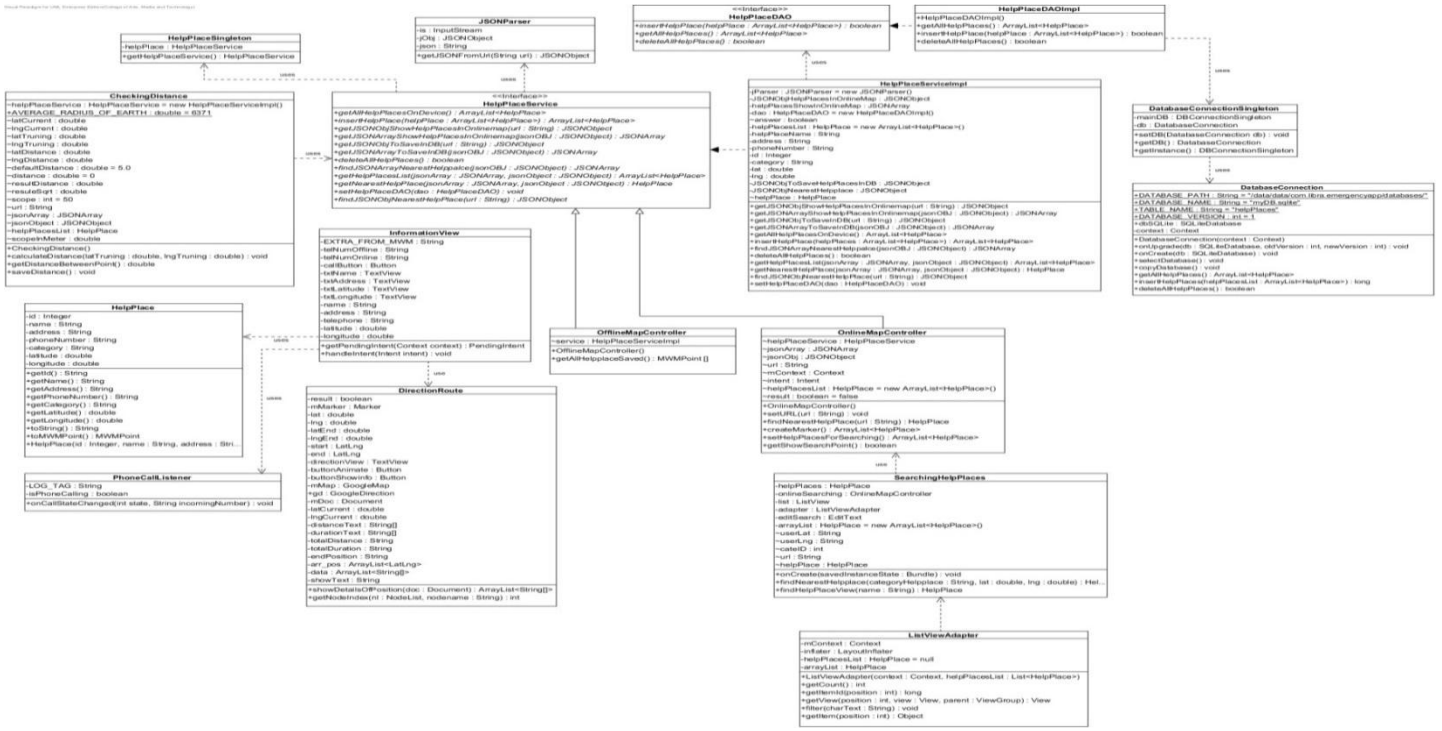

Figure 2 Emergency Information on Mobile class diagram

3.1.2 Server Part

Controller class diagram

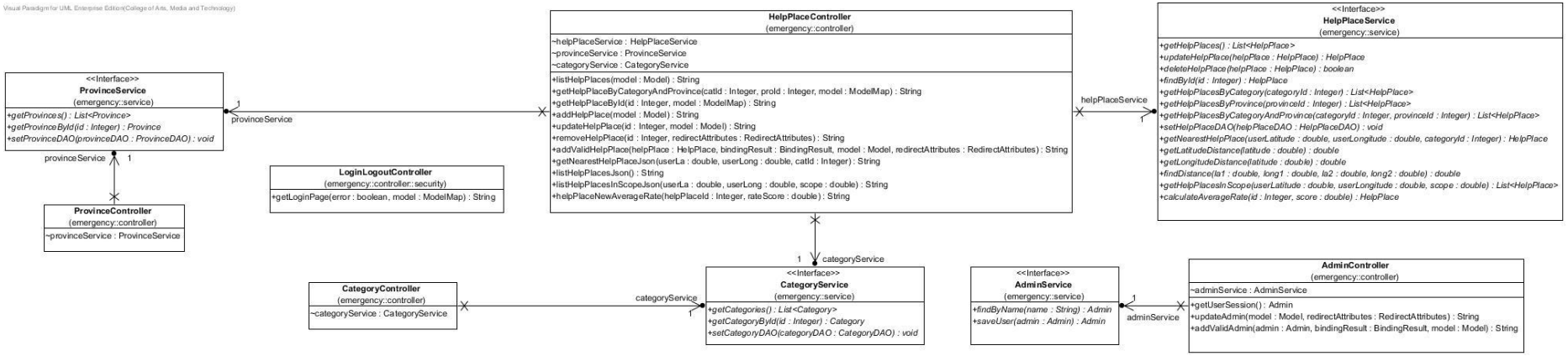

Figure 3 Controller class diagram

Service class diagram

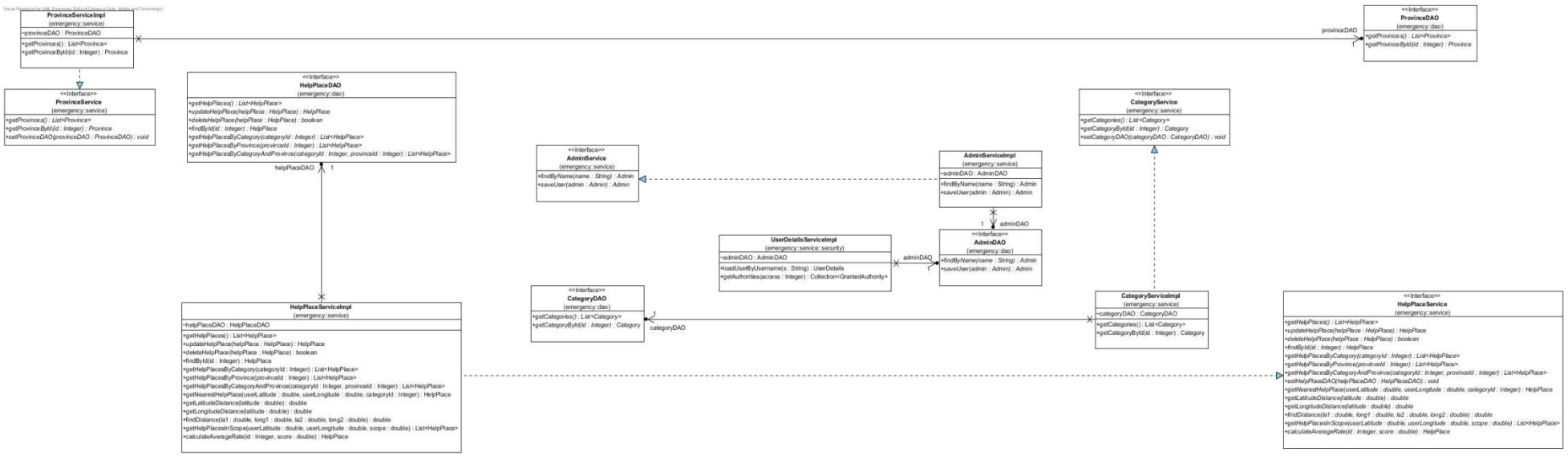

Figure 4 Service class diagram

DAO class diagram

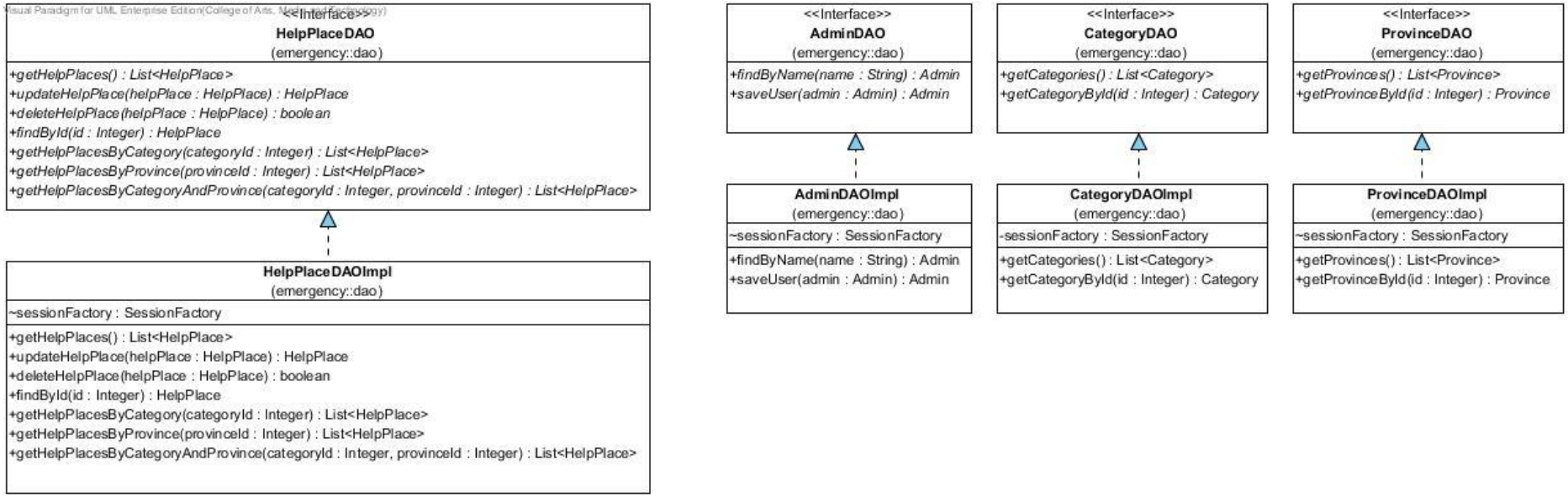

Figure 5 DAO class diagram

Entity class diagram

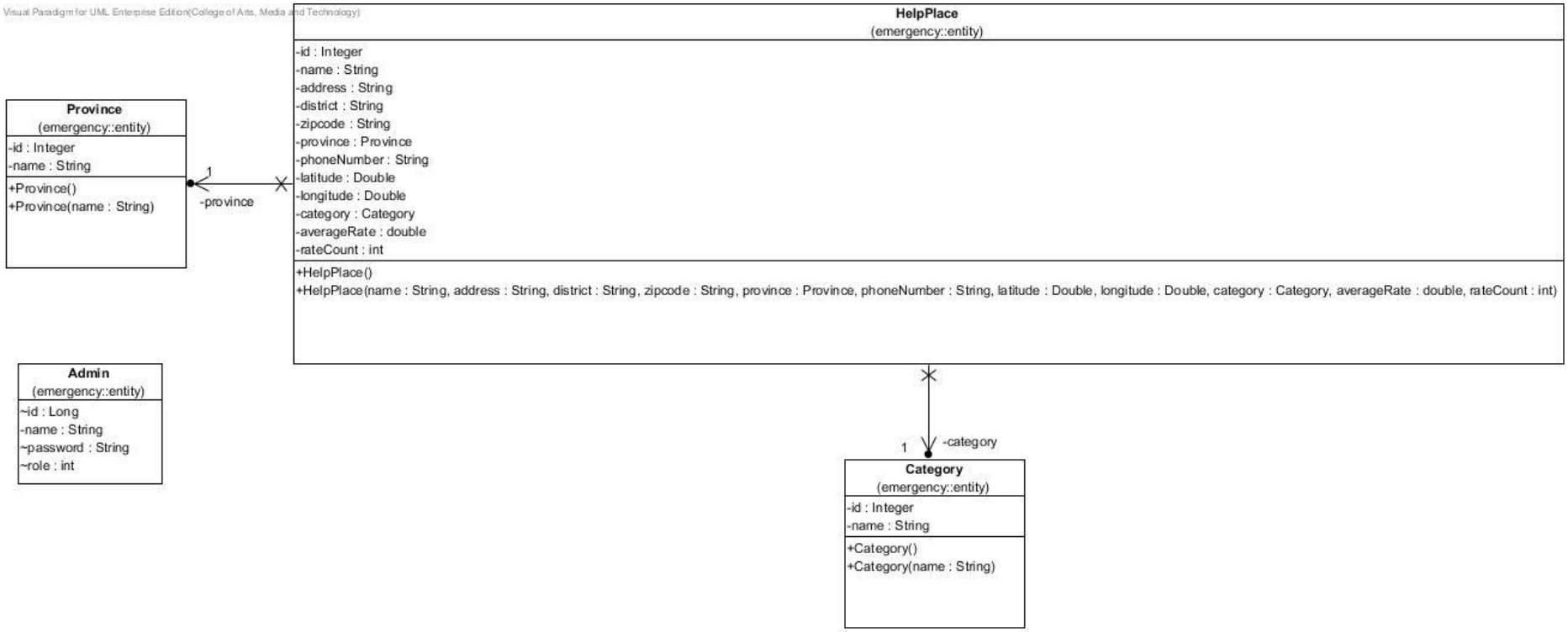

Figure 6 Entity class diagram

3.2 Class Diagram Description

3.3.1 Mobile Part

CDm-01: HelpPlace

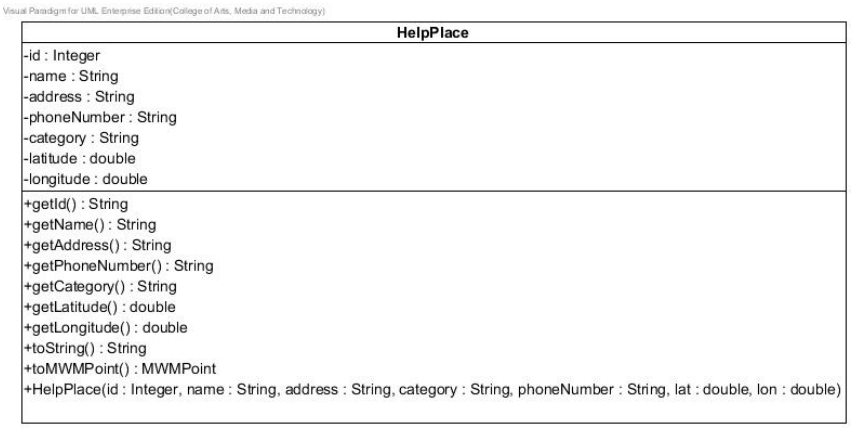

Figure 7 HelpPlace class

Entities

ID	Name	Description	Type
1	id	Store id of help place	Integer
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	category	Store category of help place	String
6	latitude	Store latitude of help place	double
7	longitude	Store longitude of help place	double

CDm-02: HelpPlaceService

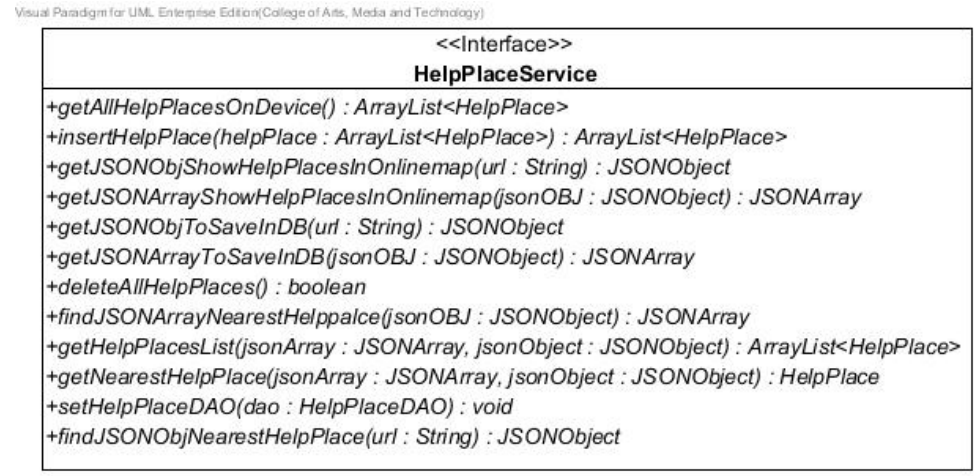

Figure 8 HelpPlaceService class

Method details

MDm-01: getJSONObjShowHelpPlacesInOnlineMap

```
public JSONObject
getJSONObjShowHelpPlacesInOnlineMap (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

MDm-02: getJSONArrayShowHelpPlacesInOnlineMap

```
public JSONArray
getJSONArrayShowHelpPlacesInOnlineMap (JSONObject
jsonOBJ)
```

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

Parameters: `jsonOBJ-JSONObject` uses to get Json array from getJsonObjShowHelpPlacesInOnlineMap method.

Return: Json array, if invalid input returns null

MDm-03: getJSONObjToSaveInDB

```
public JSONObject getJSONObjToSaveInDB (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

MDm-04: getJSONArrayToSaveInDB

```
public JSONArray getJSONArrayToSaveInDB (JSONObject
jsonOBJ)
```

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

Parameters: `jsonOBJ-JSONObject` uses to get Json array from getJSONObjToSaveInDB method.

Return: Json array, if invalid input returns null

MDm-24: findJSONObjNearestHelpPlace

```
public JSONObject findJSONObjNearestHelpPlace(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

MDm-25: findJSONArrayNearestHelpPlace

```
public JSONArray
findJSONArrayNearestHelpPlace(JSONObject jsonOBJ)
```

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

Parameters: `jsonOBJ-JSONObject` uses to get Json array from findJSONArrayNearestHelpPlace method.

Return: Json array, if invalid input returns null

MDm-26: getHelpPlacesList

```
public ArrayList<HelpPlace>
getHelpPlacesList(JSONArray jsonArray,JSONObject
jsonObject)
```

Description: Get Array list of help place object by inputting JSONArray and JSONObj

Parameters : `jsonArray - JSONArray, JSONObject - jsonObject`

Return: Array List of help place object, if invalid input returns null

MDm-27: getNearestHelpPlace

```
public HelpPlace getNearestHelpPlace(JSONArray
jsonArray,JSONObject jsonObject)
```

Description: Get help place object by inputting JSONArray and JSONObj.

Parameters: `jsonArray - JSONArray`, `JSONObject - jsonObject`.

Return: Help place object, if invalid input returns null

MDm-28: setHelpPlaceDAO

```
public void setHelpPlaceDAO(HelpPlaceDAO dao)
```

Description: Set HelpPlaceDAO value into dao attribute.

Parameters: `HelpPlaceDAO dao`.

MDm-29: deleteAllHelpPlaces

```
public boolean deleteAllHelpPlaces ()
```

Description: Delete all help places in database.

Return: Boolean, if all help places in database that deleted, will return true.

MDm-30: getAllHelpPlacesOnDevice

```
public ArrayList<HelpPlace> getAllHelpPlacesOnDevice()
```

Description: Get all help places from database on the mobile device.

Return: Array List of help place object, if invalid input returns null

MDm-31: insertHelpPlace

```
public ArrayList<HelpPlace>
insertHelpPlace(ArrayList<HelpPlace> helpPlace)
```

Description: Insert help place into database by inputting array list of help place.

Parameters: `helpPlace - ArrayList<HelpPlace>`

Return: Array List of help place object, if invalid input returns null

CDm-03: HelpPlaceServiceImpl

Figure 9 HelpPlaceServiceImpl class

Entities

ID	Name	Description	Type
1	jParser	Store JSON parser	JSONParser
2	JSONObjectHelpPlacesInOnline Map	Store JSON Object	JSONObject
3	helpPlacesShowInOnlineMap	Store help places as JSON array	JSONArray
4	dao	Store HelpPlaceDAO object	HelpPlaceDAO
5	answer	Store value of Boolean (true or false)	boolean
6	helpplaceList	Store HelpPlace object in array list	ArrayList<HelpPlace>
7	helpPlaceName	Store help place name	String
8	address	Store help place address	String
9	phoneNumber	Store help place phone number	String
10	id	Store help place ID	Integer
11	category	Store help place category	String
12	lat	Store help place latitude	Double
13	lng	Store help place longitude	Double
14	JSONObjToSaveHelpPlacesInDB	Store JSONObject	JSONObject
15	JSONObjNearestHelpplace	Store JSONObject	JSONObject
16	helpPlace	Store help place object	HelpPlace

Method details

MDm-05: getJSONObjShowHelpPlacesInOnlineMap

```
public JSONObject
getJSONObjShowHelpPlacesInOnlineMap (String url)
```

Description: Get Json object by use url of server.

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

Flow:

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

Return: Json object, if invalid input returns null

MDm-06: getJSONArrayShowHelpPlacesInOnlineMap

```
public JSONArray
getJSONArrayShowHelpPlacesInOnlineMap (JSONObject
jsonOBJ)
```

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

Parameters: `jsonOBJ-JSONObject` uses to get Json array from getJsonObjShowHelpPlacesInOnlineMap method.

Flow:

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

Return: Json array, if invalid input returns null

MDm-07: getJSONObjToSaveInDB

```
public JSONObject getJSONObjToSaveInDB (String url)
```

Description: Get Json object by use url of server.

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

Flow:

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

Return: Json object, if invalid input returns null

MDm-08: getJSONArrayToSaveInDB

`public JSONArray getJSONArrayToSaveInDB (JSONObject jsonOBJ)`

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

Parameters: `jsonOBJ-JSONObject` uses to get Json array from getJSONObjToSaveInDB method.

Flow:

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

Return: Json array, if invalid input returns null

MDm-32: findJSONObjNearestHelpPlace

`public JSONObject findJSONObjNearestHelpPlace(String url)`

Description: Get Json object by use url of server.

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

Flow:

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

Return: Json object, if invalid input returns null

MDm-33: findJSONArrayNearestHelpPlace

`public JSONArray findJSONArrayNearestHelpPlace(JSONObject jsonOBJ)`

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

Parameters: `jsonOBJ-JSONObject` uses to get Json array from findJSONArrayNearestHelpPlace method.

Flow:

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

Return: Json array, if invalid input returns null

MDm-34: getHelpPlacesList

```
public ArrayList<HelpPlace>
getHelpPlacesList(JSONArray jsonArray,JSONObject
jsonObject)
```

Description: Get Array list of help place object by inputting JSONArray and JSONObj

Parameters: `jsonArray - JSONArray, JSONObject - jsonObject`

Flow:

1. Set JSONArray and JSONObject variable to add value as HelpPlace in array list.
2. Return Array list of HelpPlace.

Return: Array List of help place object, if invalid input returns null

MDm-35: getNearestHelpPlace

```
public HelpPlace getNearestHelpPlace(JSONArray
jsonArray,JSONObject jsonObject)
```

Description: Get help place object by inputting JSONArray and JSONObj.

Parameters: `jsonArray - JSONArray, JSONObject - jsonObject`

Flow:

1. Set JSONArray and JSONObject variable to add value as HelpPlace in array list.
2. Return HelpPlace.

Return: Help place object, if invalid input returns null

MDm-36: setHelpPlaceDAO

```
public void setHelpPlaceDAO(HelpPlaceDAO dao)
```

Description: Set HelpPlaceDAO values into dao attribute.

Parameters: `HelpPlaceDAO dao`

Flow:

1. Get parameter to set value of helpPlaceDAO attribute.

MDm-37: deleteAllHelpPlaces

```
public boolean deleteAllHelpPlaces ()
```

Description: Delete all help places in database.

Flow:

1. Use method deleteAllHelpPlaces( ) of HelpPlaceDAO class to delete help place.
2. Return Boolean true.

Return: Boolean, if all help places in database that deleted, will return true.

MDm-38: getAllHelpPlacesOnDevice

```
public ArrayList<HelpPlace> getAllHelpPlacesOnDevice()
```

Description: Get all help places from database on the mobile device.

Flow:

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

Return: Array List of help place object, if invalid input returns null

MDm-39: insertHelpPlace

```
public ArrayList<HelpPlace>
insertHelpPlace(ArrayList<HelpPlace> helpPlace)
```

Description: Insert help place into database by inputting array list of help place.

Parameters: `helpPlace - ArrayList<HelpPlace>`

Flow:

1. Use method insertHelpPlace (helpPlace) of HelpPlaceDAO class to insert array list of help places.
2. Return help place object.

Return: Array List of help place object that contained, if invalid input returns null

CDm-04: HelpPlaceSingleton

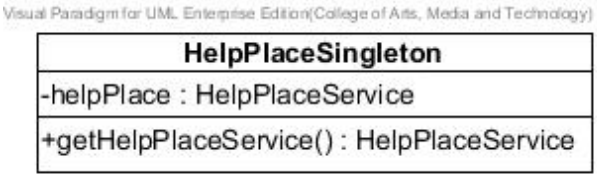

Figure 9 HelpPlaceSingleton class

Entities

ID	Name	Description	Type
1	helpPlace	Store HelpPlaceService object	HelpPlaceService

Method details

MDm-09: getHelpPlaceService

```
public HelpPlaceService getHelpPlaceService ()
```

Description: Get HelpPlaceService with static value.

Flow:

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

Return: Help place object, if invalid input returns null.

CDm-05: OnlineMapController

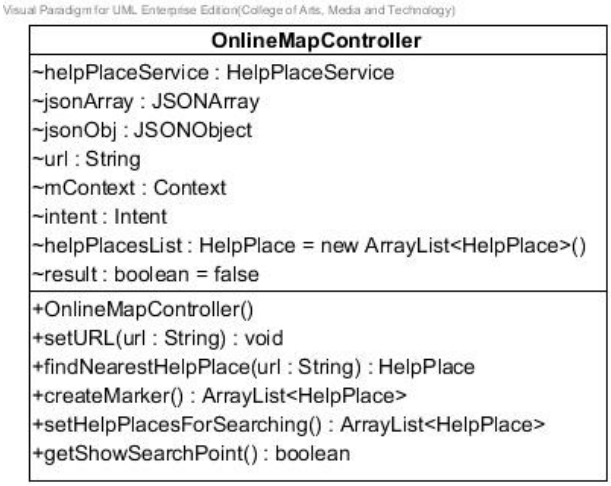

Figure 10 OnlineMapController class

Entities

ID	Name	Description	Type
1	helpPlaceService	Create help place service object	HelpPlaceService
2	mContext	Store context object	Context
3	intent	Store intent	Intent
4	jsonArray	Store help place in Json array	JSONArray
5	result	Store Boolean (true or false)	boolean
6	url	Store url of server	String
7	helpPlaceList	Store HelpPlace in array list	ArrayList<HelpPlace>
8	jsonObj	Store Json object	JSONObject

Method details

MDm-10: 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)

MDm-11: createMarker

```
public ArrayList<HelpPlace> createMarker ()
```

Description: Create marker with information of help place.

Parameters: None

Flow:

1. Input url to get JSONObject by using method findJSONObjShowHelpPlacesInOnlineMap of HelpPlaceService class.
2. Input JSONObject to get JSONArray in method findJSONArrayShowHelpPlacesInOnlineMap of HelpPlaceService class.
3. Input JSONObject and JSONArray as parameter in method getHelpPlacesList of HelpPlaceService class
4. Return Array list of Help place object.

Return: Array list of Help place object , if invalid input returns null.

MDm-40: setURL

```
public void setURL(String url)
```

Description: Set url to connect with the server

Parameters: `url-String` uses for setting url of server

Flow:

1. Set valiable url with the parameter that received.
2. Input url in to method getJSONObjShowHelpPlacesInOnlinemap of HelpPlaceService class.

MDm-41: findNearestHelpPlace

```
public HelpPlace findNearestHelpPlace(String url)
```

Description: Set url to connect with the server to find the nearest help place

Parameters: `url-String` uses for setting url of server

Flow:

1. Set valiable url with the parameter that received.
2. Input url to get JSONObject by using method findJSONObjNearestHelpPlace of HelpPlaceService class.
3. Input JSONObject to get JSONArray in method findJSONArrayNearestHelppalce of HelpPlaceService class.
4. Input JSONObject and JSONArray as parameter in method getNearestHelpPlace of HelpPlaceService class
5. Return Help place object.

Return: Help place object, if invalid input returns null.

MDm-42: setHelpPlacesForSearching

```
public ArrayList<HelpPlace>
setHelpPlacesForSearching ()
```

Description: Set help places into array list.

Flow:

1. Input url to get JSONObject by using method findJSONObjShowHelpPlacesInOnlineMap of HelpPlaceService class.
2. Input JSONObject to get JSONArray in method findJSONArrayShowHelpPlacesInOnlineMap of HelpPlaceService class.
3. Input JSONObject and JSONArray as parameter in method getHelpPlacesList of HelpPlaceService class
4. Return Array list of Help place object.

Return: Array list of Help place object, if invalid input returns null

MDm-43: getShowSearchPoint

```
public boolean getShowSearchPoint()
```

Description: Get result of searching point.

Flow:

1. Set condition if result : Boolean equal true.
2. mContext variable call method startActivity by inputting intent
3. return result as Boolean value.

Return: Boolean, if all help places in database that deleted, will return true.

CDm-06: OfflineMapController

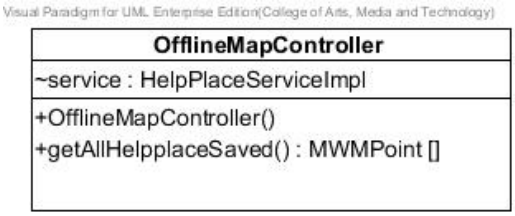

Figure 11 OfflineMapController class

Entities

ID	Name	Description	Type
1	service	Create help place service ohject	HelpPlaceServiceImpl

Method details

MDm-12: OfflineMapController

```
public OfflineMapController()
```

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.

MDm-13: getAllHelpplaceSaved

```
public HelpPlaces[] getAllHelpplaceSaved ()
```

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

Parameters: None

Flow:

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

Return: HelpPlaces array, if invalid value returns null.

CDm-07: InformationView

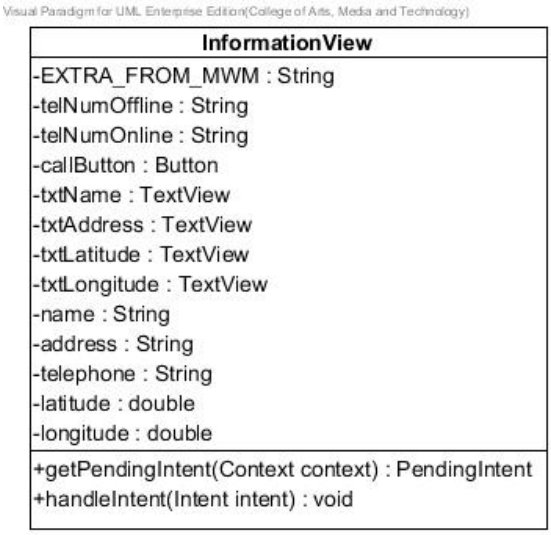

Figure 12 InformationView class

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

MDm-17: 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.

MDm-18: 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.

CDm-08: JSONParser

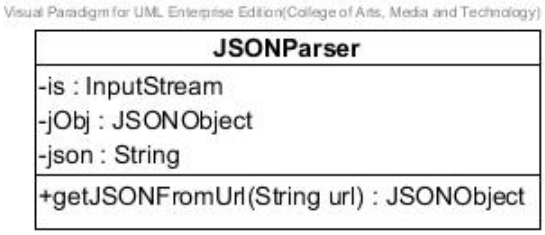

Figure 13 JSONParser class

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

MDm-19: 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.

CDm-09: PhoneCallListener

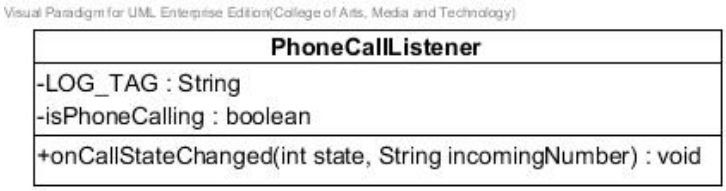

Figure 14 PhoneCallListener class

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

MDm-20: 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)

CDm-10: DatabaseConnection

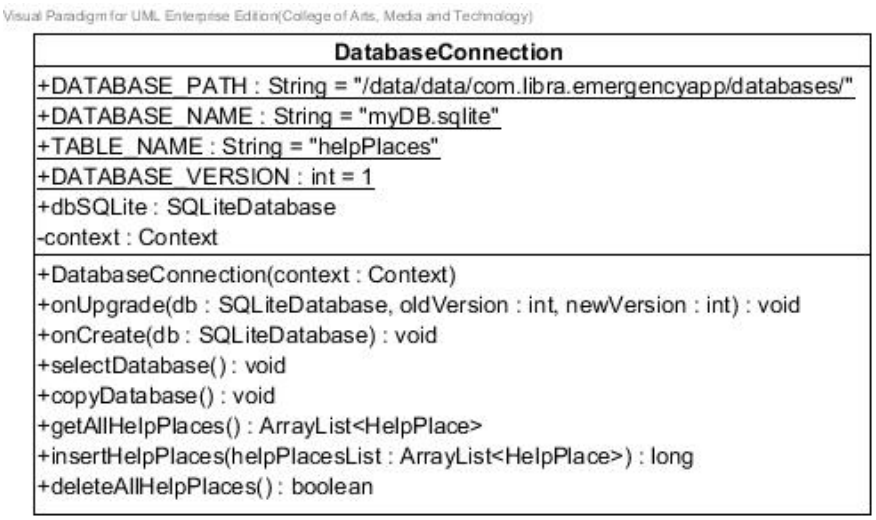

Figure 15 DatabaseConnection class

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

MDm-21: 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.

MDm-22: 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

MDm-23: geAllHelpPlaces

```
public ArrayList <HelpPlace> getAllHelpPlaces ()
```

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 , getInteger, getString and getDouble method for putting value into HelpPlace.
3. Put HelpPlace object into an array list.
4. Return an array list.

Return: Array list, if invalid value returns null.

MDm-14: DatabaseConnection

```
public DatabaseConnection(Context context)
```

Description: Set context, database name and database version

Parameters: `Context context`

Flow:

1. Set context, database name and database as parameter of super class.

MDm-15: insertHelpPlaces

```
public long insertHelpPlaces(ArrayList<HelpPlace>
```

helpPlacesList)

Description: Insert array list of HelpPlace into database

Parameters: `ArrayList<HelpPlace> helpPlaceList`

Flow:

1. Use for each to set value of HelpPlace object into database.
2. Insert command by using method insert of SQLiteDatabase.
3. Return rows of help place.

Return: rows of help places in database, if invalid value returns null.

MDm-16: deleteAllHelpPlaces

```
public boolean deleteAllHelpPlaces()
```

Description: delete all help places from database

Flow:

1. Use for each to set value of HelpPlace object into database.
2. Insert command by using method insert of SQLiteDatabase.
3. Return rows of help place.

Return: Boolean, if all help places in database that deleted, will return true.

CDm-11: CheckingDistance

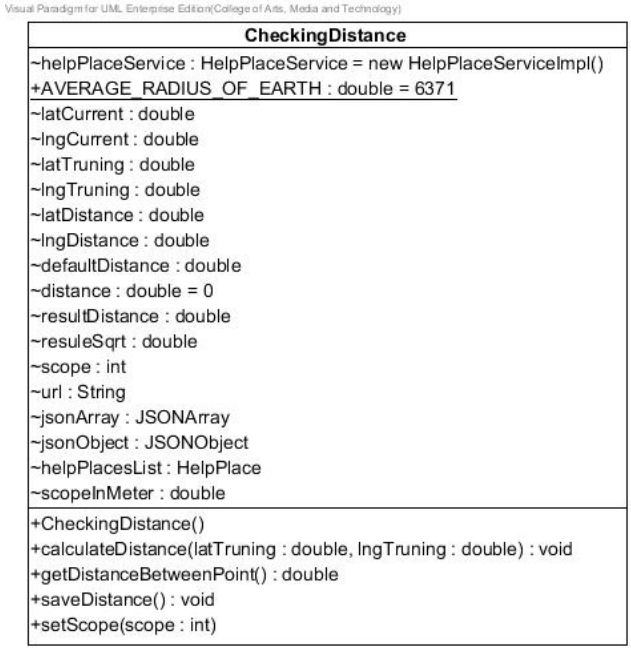

Figure 16 CheckingDistance class

Entities

ID	Name	Description	Type
1	helpPlaceService	Store HelpPlaceService object	HelpPlaceService
2	*AVERAGE_RADIUS_OF_EARTH*	Store final value of average radius of earth	Double
3	latCurrent	Store latitude in last position	Double
4	lngCurrent	Store longitude in last position	Double
5	latTruning	Store latitude in new position	Double
6	lngTruning	Store longitude in new position	Double
7	latDistance	Store latitude different distance	Double
8	lngDistance	Store longitude different distance	Double
9	defaultDistance	Store default distance	Double

10	distance	Store distance	Double
11	resultDistance	Store value calculating in sin, cos, tan	Double
12	resultSqrt	Store value calculating in sqrt	Double
13	scope	Store scope of saving HelpPlace	int
14	url	Store URL of server	String
15	helpPlacesList	Store array list of HelpPlace	ArrayList<HelpPlace>
16	jsonArray	Store JSON Array	JSONArray
17	jsonObject	Store JSON Object	JSONObject
18	scopeInMeter	Store number of scope in meter	Double

MDm-44: setScope

```
public void setScope (int scope)
```

Description: Setting scope by inputting scope as parameter

Parameters: `scope – int`

Flow:

1. Set number of scope that receive in parameter

MDm-45: calculateDistance

```
public void calculateDistance(double latTruning,
double lngTruning)
```

Description: Calculate distance by receiving new position as latTruning and lngTruning in parameter

Parameters: `latTruning – double, lngTruning - double`

Flow:

1. Set variable equal with parameter
2. Set value to calculate with class math toRadians
3. Calculate in math class with sin, cos and tan.
4. Average distance
5. Check condition to save new information into database.

MDm-46: getDistanceBetweenPoint

```
public double getDistanceBetweenPoint()
```

Description: Get distance from calculateDistance method

Flow:

1. Return distance of method calculateDistance

Return: distance, if all help places in database that deleted, will return 0.

MDm-47: saveDistance

```
public void saveDistance()
```

Description: Get distance from calculateDistance method

Flow:

1. Use deleteAllHelpPlaces method of helpPlaceService class.
2. input URL to get JSONObject and JSONArray
3. Use getHelpPlacesList method of HelpPlaceService class by inputting JSONObject and JSONArray as parameter
4. Get help place list
5. Insert list of help place by use insertHelpPlace method of HelpPlaceService class.

CDm-12: SearchingHelpPlaces

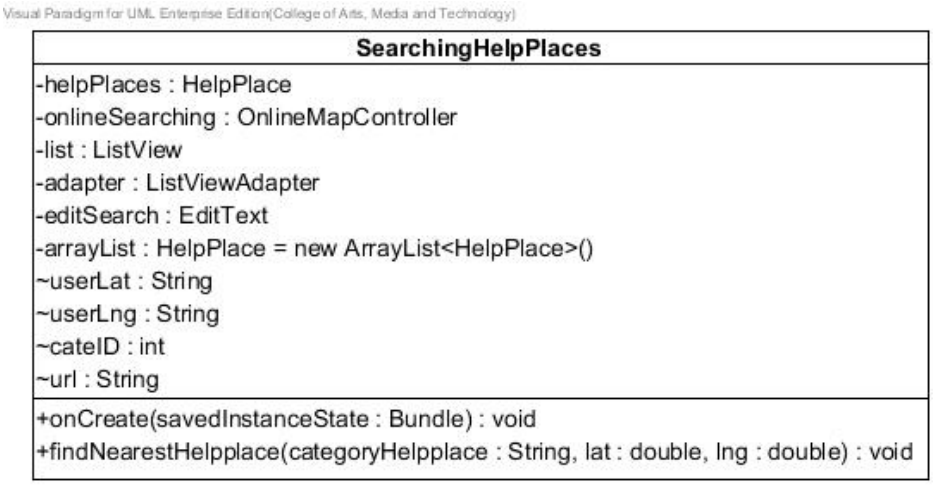

Figure 17 SearchingHelpPlace class

Entities

ID	Name	Description	Type
1	helpPlace	Store HelpPlace object	HelpPlace
2	onlineSearching	Store OnlineMapController object	OnlineMapController
3	list	Store ListView	ListView
4	adapter	Store ListViewAdapter	ListViewAdapter
5	editSearch	Store EditText	EditText
6	arrayList	Store array list of HelpPlace	ArrayList <HelpPlace>
7	userLat	Store latitude of user	Double
8	userLng	Store longitude of user	Double
8	cateID	Store category's ID	int
9	url	Store URL of server	String

MDm-48: findNearestHelpplace

`public void findNearestHelpplace(String categoryHelpplace, double lat, double lng)`

Description: Find nearest help place by inputting name of category, latitude and longitude.

Parameters: `categoryHelpplace – String, lat – double, lng – double`

Flow:

1. Generate URL by inputting latitude and longitude
2. Check condition and set category id to input as URL
3. Use findNearestHelpPlace method of OnlineMapController class.

CDm-13: DatabaseConnectionSingleton

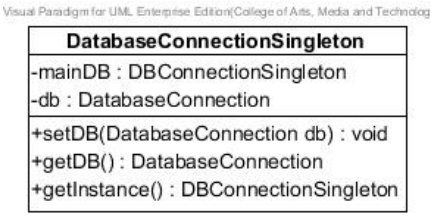

Figure 18 DatabaseConnectionSingleton class

Entities

ID	Name	Description	Type
1	mainDB	Store static DBConnectionSingleton	DBConnectionSingleton
2	db	Store static DatabaseConnection	DatabaseConnection

MDm-49: setDB

`public void setDB(DatabaseConnection db)`

Description: Set database connection.

Parameters: `db – DatabaseConnection`

Flow:

1. Set db variable is equal receiving parameter

MDm-50: getDB

`public DatabaseConnection getDB()`

Description: Get database connection.

Flow:

1. Set db variable is equal receiving parameter

Return: database Connection, if all help places in database that deleted, will return null.

MDm-51: getInstance

```
public DBConnectionSingleton getInstance()
```

Description: Get instance from DBconnectionSingleton.

Flow:

1. Get instance of DBconnectionSingleton

Return: database Connection Singleton, if all help places in database that deleted, will return null.

CDm-14: DirectionRoute

Figure 19 DatabaseConnection class

Entities

ID	Name	Description	Type
1	result	The variable to use for checking status	Boolean
2	mMarker	Adding marker and data of marker into map	Marker
3	lat	Set value of latitude	Double
4	lng	Set value of longitude	Double
5	latEnd	Set value of destination's latitude	Double
6	lngEnd	Set value of destination's longitude	Double
7	start	Set value for start point	LatLng

8	end	Set value for end point	LatLng
9	directionView	Text filed for view direction	TextView
10	buttonAnimate	Button for click to make animation	Button
11	buttonShowinfo	Button for click to view information	Button
12	mMap	Variable use for showing Google Maps	GoogleMap
13	gd	Variable to use Google Direction	GoogleDirection
14	mDoc	Document file	Document
15	latCurrent	Latitude of user's current location	Double
16	lngCurrent	Longitude of user's current location	Double
17	distanceText	Get list of distance from document	String [ ]
18	durationText	Get list of duration time from document	String [ ]
19	totalDistance	Get total distance from start to end position	String
20	totalDuration	Get total duration from start to end position	String
21	endPosition	Get end position name	String
22	arr_pos	Get list of position's section	ArrayList<LatLng>
23	data	Set data from Google document	ArrayList<String []>
24	showText	Use for show information about routing	String

Method details

MDm-52: showDetailsOfPosition

```
public ArrayList<String[]>
showDetailsOfPosition(Document doc)
```

Description: Get information from document for defining data of each node. Includes, maneuver, duration and distance

Parameters: `doc - Document` uses define document to use

Flow:

1. Set node list to equal doc.getElementsByTagName("step")
2. Use for loop to get item from node list
3. Checking in for loop by using "if statement" to fine keyword "maneuver"
4. If found "maneuver" next node will getIndex and set value of "duration", "distance" and "maneuver"
5. Adding data into array list

Return: Array list of string array, if invalid input returns null.

MDm-53: getNodeIndex

```
public int getNodeIndex(NodeList nl, String
nodename)
```

Description: Get number of the index node

Parameters: `nl-NodeList` uses to get size get details
`nodename-String` uses to compare node

Flow:

1. Set for loop by length of node.
2. Use if statement for check item equal the name
3. If valid Return index of node or not valid return -1

Return: Integer of index, if invalid value returns null.

3.3.2 Server Part

CDs-01: HelpPlace

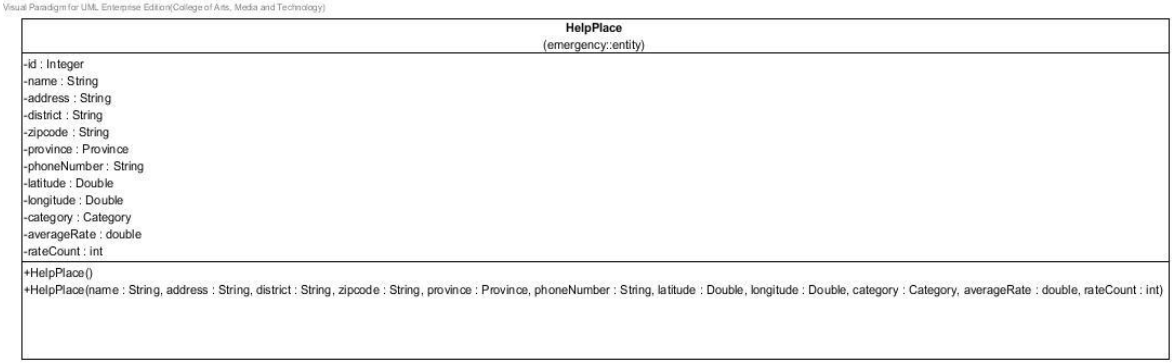

Figure 20 HelpPlace class

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
10	averageRate	stored average rating score	Double
11	rateCount	stored number count time of rating	int

Method details

MDs-01: HelpPlaces

```
public HelpPlace(String name, String address, String district, String zipcode, Province province, String phoneNumber, Double latitude, Double longitude, Category category, double averageRate, int rateCount)
```

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

CDs-02: HelpPlaceDAO

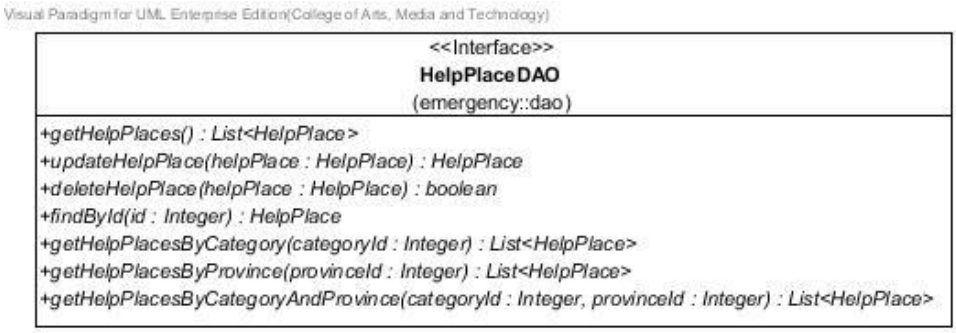

Figure 21 HelpPlaceDAO class

Method details

MDs-02: getHelpPlaces

```
public List<HelpPlace> getHelpPlaces()
```

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

Return: Lists of the help places in the database.

MDs-03: 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.

MDs-04: 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.

MDs-05: 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.

MDs-06: 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.

MDs-07: 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.

MDs-08: 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.

CDs-03: HelpPlaceDAOImpl

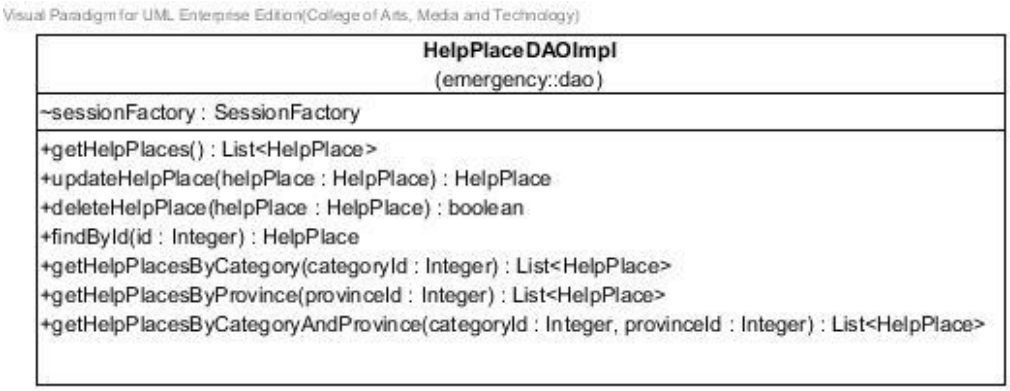

Figure 22 HelpPlaceDAOImpl class

Entities

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

Method details

MDs-09: 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.

MDs-10: 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.

MDs-11: 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.

MDs-12: 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.

MDs-13: 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.

MDs-14: 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.

MDs-15: 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.

CDs-04: HelpPlaceService

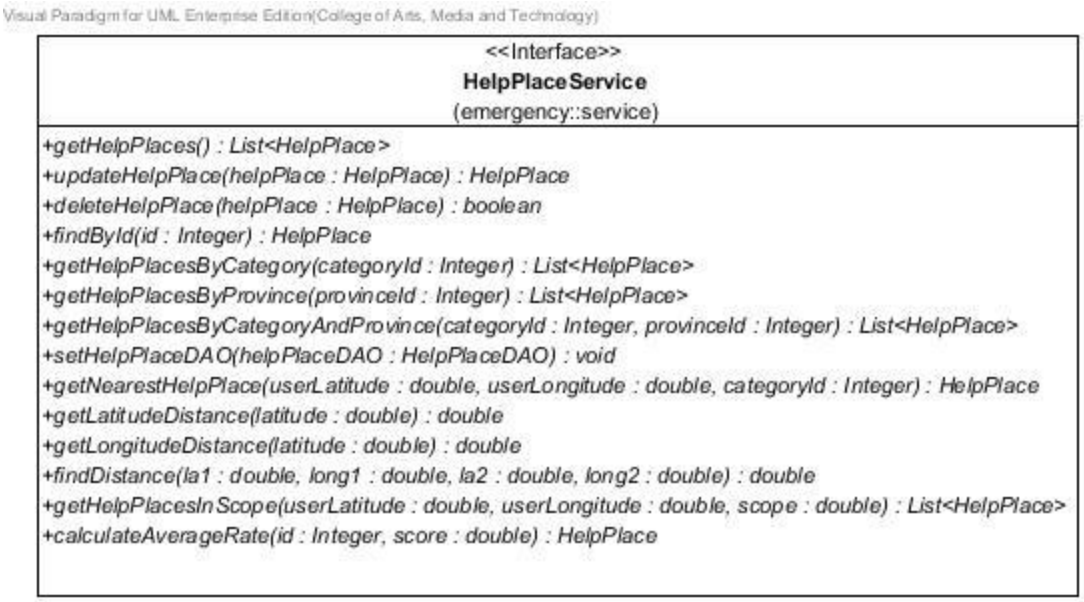

Figure 23 HelpPlaceService class

Method details

MDs-16: getHelpPlaces

```
public List<HelpPlace> getHelpPlaces ()
```

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

Return: List of all help places in database.

MDs-17: 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.

MDs-18: 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.

MDs-19: 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.

MDs-20: 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.

MDs-21: 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.

MDs-22: 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

MDs-23: setHelpPlaceDAO

```
public void setHelpPlaceDAO(HelpPlaceDAO
helpPlaceDAO)
```

Description: Set `HelpPlaceDAO` value into help place's attribute.

Parameter: `helpPlaceDAO-HelpPlaceDAO` use for setting help place DAO.

MDs-61: getNearestHelpPlace

```
public HelpPlace getNearestHelpPlace(double
userLatitude, double userLongitude, Integer categoryId)
```

Description: Get nearest help place from the database by the selected category id.

Parameter: `userLatitude-double` use for finding nearest help place.

`userLongitude-double` use for finding nearest help place.

`categoryId-Integer` use for getting the help place from database.

Return: Nearest help place by the selected category.

MDs-62: getLatitudeDistance

```
public double getLatitudeDistance(double latitude)
```

Description: Get surface distance per 1 degree change in latitude (meter).

Parameter: `latitude-double` use for finding surface distance per 1 degree change in latitude .

Return: Surface distance per 1 degree change in latitude in unit meter.

MDs-63: getLongitudeDistance

```
public double getLongitudeDistance(double latitude)
```

Description: Get surface distance per 1 degree change in longitude (meter).

Parameter: `latitude-double` use for finding surface distance per 1 degree change in longitude .

Return: Surface distance per 1 degree change in longitude in unit meter.

MDs-64: findDistance

```
public double findDistance(double la1, double
long1,double la2, double long2)
```

Description: Get distance between two difference locations measures in unit meter.

Parameter: `la1-double` use for indicate latitude of location one.

`long1-double` use for indicate longitude of location one.

`la2-double` use for indicate latitude of location two.

`long2-double` use for indicate longitude of location two.

Return: Distance between two locations in unit meter.

MDs-65: getHelpPlacesInScope

```
public List<HelpPlace> getHelpPlacesInScope(double
userLatitude, double userLongitude, double scope)
```

Description: Get list of help places in the setting area scope.

Parameter: `userLatitude-double` use for indicate user current location.

`userLongitude-double` use for indicate user current location.

`scope-double` use to define the scope for getting help places.

Return: List of help places locate in the selected scope.

MDs-74: calculateAverageRate

```
public HelpPlace calculateAverageRate(Integer id,
double score)
```

Description: Calculate the new average rating score from new input score

Parameter: id-Integer use for indicate help place's ID.

score-double use for indicate a score for recalculating.

Return: Help place with new average rating score.

CDs-05: HelpPlaceServiceImpl

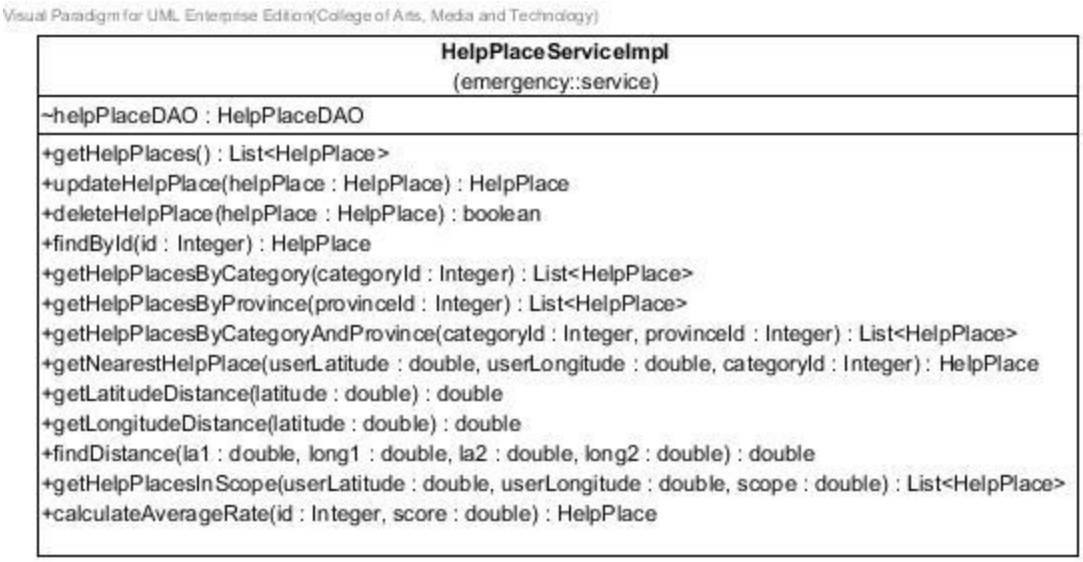

Figure 24 HelpPlaceServiceImpl class

Entities

ID	Name	Description	Type
1	helpPlaceDAO	Contained HelpPlaceDAO object	HelpPlaceDAO

Method details

MDs-24: 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.

MDs-25: 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.

MDs-26: 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.

MDs-27: 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.

MDs-28: 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.

MDs-29: 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.

MDs-30: 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.

MDs-31: 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.

MDs-66: getNearestHelpPlace

```
public HelpPlace getNearestHelpPlace(double userLatitude, double userLongitude, Integer categoryId)
```

Description: Get nearest help place from the database by the selected category id.

Parameter: `userLatitude-double` use for finding nearest help place.

`userLongitude-double` use for finding nearest help place.

`categoryId-Integer` use for getting the help place from database.

Flow:

1. Use method `getHelpPlacesByCategory(categoryId)` to get list of help places .
2. Check if the list is empty, then return null.
3. If not, get distance of all help places in list by use method `findDistance(double la1, double long1, double la2, double long2)`, by compare latitude and longitude in parameter with latitude and longitude of each help place in list.
4. Keep all distances in distance list.
5. Find the minimum distance from each value in distance list.
6. Return help place which have a minimum distance value.

Return: Nearest help place by the selected category.

MDs-67: getLatitudeDistance

```
public double getLatitudeDistance(double latitude)
```

Description: Get surface distance per 1 degree change in latitude (meter) which is a tolerance value.

Parameter: `latitude-double` use for finding surface distance per 1 degree change in latitude **.**

Flow:

1. Use latitude from parameter divided by 15 and convert result into integer, to find number of index in an array.
2. Get value “Surface distance per 1 degree change in latitude” in the array, multiply with 1000 to change unit into meter.
3. Return Surface distance per 1 degree change in latitude in unit meter.

Return: Surface distance per 1 degree change in latitude in unit meter.

MDs-68: getLongitudeDistance

```
public double getLongitudeDistance(double latitude)
```

Description: Get surface distance per 1 degree change in longitude (meter) which is a tolerance value.

Parameter: `latitude-double` use for finding surface distance per 1 degree change in longitude **.**

Flow:

1. Use latitude from parameter divided by 15 and convert result into integer, to find number of index in an array.
2. Get value “Surface distance per 1 degree change in longitude” in the array, multiply with 1000 to change unit into meter.
3. Return “Surface distance per 1 degree change in longitude” in unit meter.

Return: Surface distance per 1 degree change in longitude in unit meter.

MDs-69: findDistance

```
public double findDistance(double la1, double
long1,double la2, double long2)
```

Description: Get distance between two difference locations measures in unit meter.

Parameter: `la1-double` use for indicate latitude of location one.

`long1-double` use for indicate longitude of location one.

`la2-double` use for indicate latitude of location two.

`long2-double` use for indicate longitude of location two.

Flow:

1. Find "Surface distance per 1 degree change in latitude", and "Surface distance per 1 degree change in longitude" to use calculate distance.
2. Get parameter and calculate to get distance between two locations.
3. Return distance.

Return: Distance between two locations in unit meter.

MDs-70: getHelpPlacesInScope

```
public List<HelpPlace> getHelpPlacesInScope(double
userLatitude, double userLongitude, double scope)
```

Description: Get list of help places in the setting area scope.

Parameter: `userLatitude-double` use for indicate user current location.

`userLongitude-double` use for indicate user current location.

`scope-double` use to define the scope for getting help places.

Flow:

1. Get list of all help places by using method `getHelpPlaces()`.
2. Find a distance between each help place in the list, and `userLatitude`, `userLongitude`.
3. Check each distance is in the scope or not.
4. Keep the help place which its distance is in the setting scope.
5. Return list of help places where locate in the setting scope.

Return: List of help places locate in the selected scope.

MDs-75: calculateAverageRate

```
public HelpPlace calculateAverageRate(Integer id,
double score)
```

Description: Calculate the new average rating score from new input score

Parameter: id-Integer use for indicate help place's ID.

score-double use for indicate a score for recalculating.

Flow:

1. Get help place by using method findById(id)
2. Calculate new average rating score.
3. Set new rate count and average rate to help place object.
4. Update help place object into system database.
5. Return help place with new average rating score.

Return: Help place with new average rating score.

CDs-06: Category

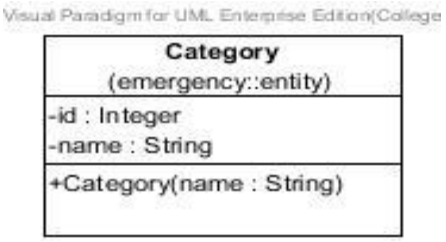

Figure 25 Category class

Entities

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

Method details

MDs-32: Category

```
public Category(String name)
```

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

CDs-07: CategoryDAO

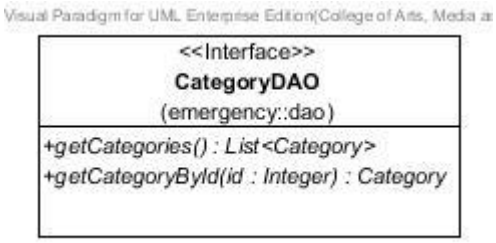

Figure 26 CategoryDAO class

Method details

MDs-33: getCategories

```
public List<Category> getCategories()
```

Description: Get all categories from the database.

Return: List of all categories in the database.

MDs-34: 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.

CDs-08: CategoryDAOImpl

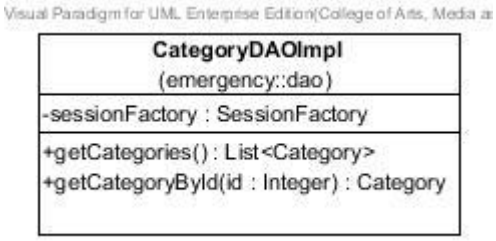

Figure 27 CategoryDAOImpl class

Entities

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

Method details

MDs-35: 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.

MDs-36: 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.

CDs-09: CategoryService

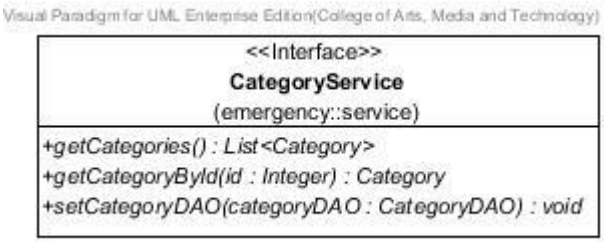

Figure 28 CategoryService class

Method details

MDs-37: getCategories

```
public List<Category> getCategories()
```

Description: Get all categories from the database.

Return: List of all categories in the database.

MDs-38: 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.

MDs-39: setCategoryDAO

```
public void setCategoryDAO(CategoryDAO
categoryDAO)
```

Description: Set `categoryDAO` value into category's attribute.

Parameter: `categoryDAO-CategoryDAO` use for setting category DAO.

CDs-10: CategoryServiceImpl

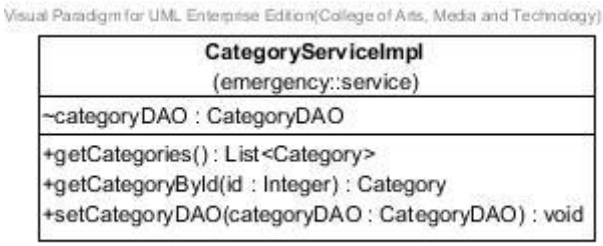

Figure 29 CategoryServiceImpl class

Method details

MDs-40: 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.

MDs-41: 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.

MDs-42: 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.

CDs-11: Province

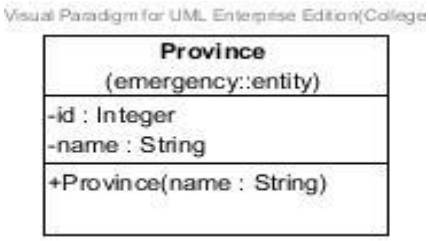

Figure 30 Province class

Entities

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

Method details

MDs-43: Province

```
public Province(String name)
```

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

CDs-12: ProvinceDAO

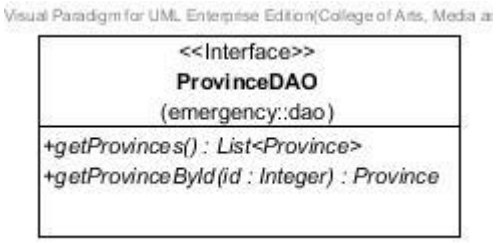

Figure 31 ProvinceDAO class

Method details

MDs-44: getProvinces

```
public List<Province> getProvinces()
```

Description: Get all provinces from the database.

Return: List of all provinces in database.

MDs-45: 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.

CDs-13: ProvinceDAOImpl

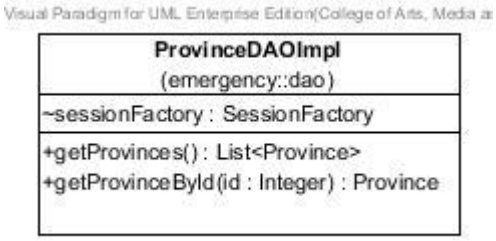

Figure 32 ProvinceDAOImpl class

Entities

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

Method details

MDs-46: 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.

MDs-47: 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.

CDs-14: ProvinceService

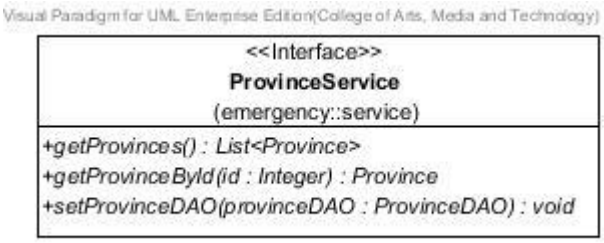

Figure 33 ProvinceService class

Method details

MDs-48: getProvinces

```
public List<Province> getProvinces();
```

Description: Get all provinces from the database

Return: List of all provinces in database.

MDs-49: 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.

MDs-50: setProvinceDAO

```
public void setProvinceDAO(ProvinceDAO provinceDAO)
```

Description: Set `provinceDAO` value into province's attribute.

Parameter: `provinceDAO-ProvinceDAO` use for setting province DAO.

CDs-15: ProvinceServiceImpl

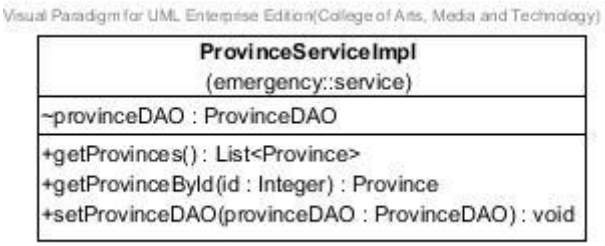

Figure 34 ProvinceServiceImpl class

Method details

MDs-51: 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.

MDs-52: 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.

MDs-53: 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.

CDs-16: HelpPlaceController

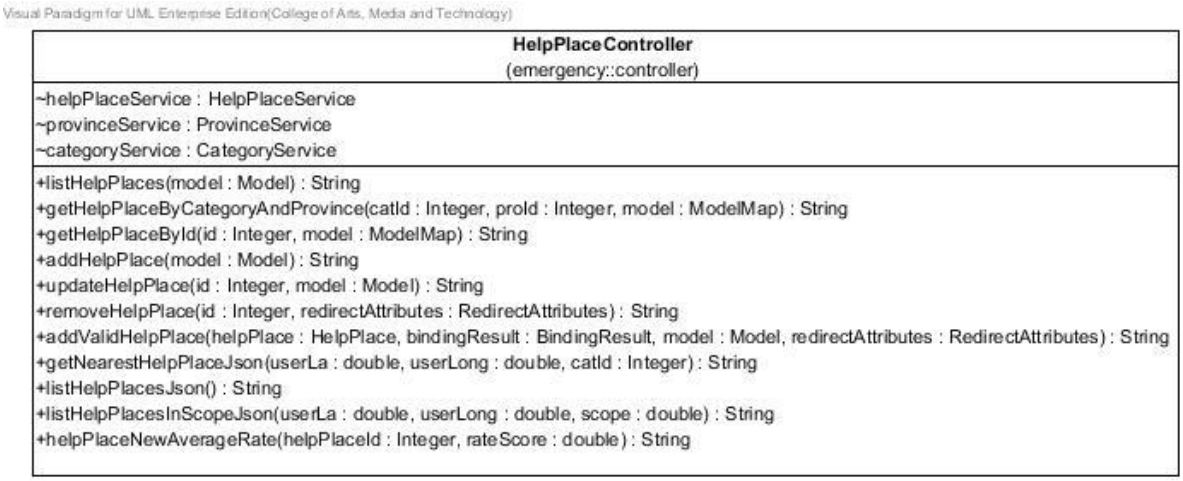

Figure 35 HelpPlaceController class

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

MDs-54: 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

MDs-55: 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-Integer` uses for getting help places by province's id.

`catId-Integer` uses for getting help places by category's id.

Return: Home_page.jsp page

MDs-56: 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-Integer` uses for getting help place by its id.

Return: viewInformation.jsp page

MDs-57: 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

MDs-58: 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-Integer` uses for getting help place by its id.

Return: updateInfo.jsp page

MDs-59: 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-Integer` uses for deleting help place by its id.

Return: Home_page.jsp page.

MDs-60: 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-HelpPlace` uses for updating the help place.

`bindingResult-BindingResult` uses for keeping the invalid data.

`redirectAttributes-RedirectAttributes` uses for store the error message.

Return: If there are invalid information, go to updateInfo.jsp page, otherwise go to Home_page.jsp page.

MDs-71: getNearestHelpPlaceJson

```
public @ResponseBody String
getNearestHelpPlaceJson(@PathVariable("userLa") double
userLa,@PathVariable("userLong") double
userLong,@PathVariable("catId") Integer catId)
```

Description: Get the nearest help place in JSON form.

Parameter: `userLa-double` use for get user latitude.

`userLong-double` use for get user longitude.

`catId-Integer` use for get help places by the selected category.

Return: Information of the nearest help place in form of Json.

MDs-72: listHelpPlacesJson

```
public @ResponseBody String listHelpPlacesJson()
```

Description: Get list of all help places in JSON form.

Parameter: –

Return: Information of all help places in form of Json.

MDs-73: listHelpPlacesInScopeJson

```
public @ResponseBody String
listHelpPlacesInScopeJson(@PathVariable("userLa") double
userLa,@PathVariable("userLong") double
userLong,@PathVariable("scope") double scope)
```

Description: Get list of all help places, where locate in the setting scope, in JSON form.

Parameter: `userLa-double` use for get user latitude.

`userLong-double` use for get user longitude.

`scope-double` use for define area to get help places.

Return: Information of help places, where locate in the setting scope, in form of Json.

MDs-76: helpPlaceNewAverageRate

```
  public @ResponseBody String
helpPlaceNewAverageRate(@PathVariable("helpplaceId")
Integer helpPlaceId,@PathVariable("rateScore") double
rateScore) throws JSONException
```

Description: Get help place with new average rating score in JSON form.

Parameter: helpPlaceId - Integer use for get help place.

rateScore -double use for calculate new average rating score.

Return: Information of help places with new average rating score in form of Json.

CDs-17: AdminController

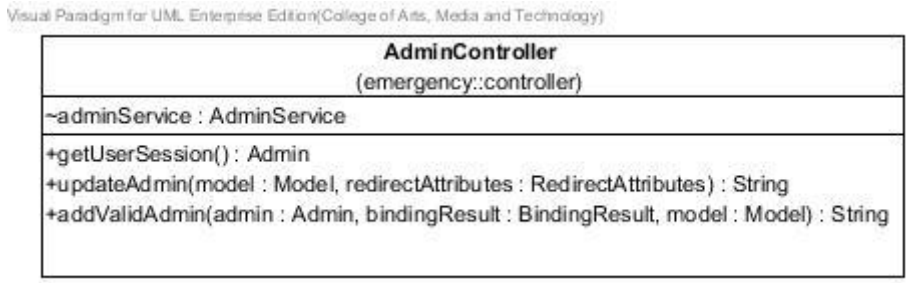

Figure 36 AdminController class

Entities

ID	Name	Description	Type
1	adminService	Contained AdminService object	AdminService

Method details

MDs-77: getUserSession

```
public Admin getUserSession()
```

Description: For getting the user in session.

Parameter: -

Return: Admin object that is in the session.

MDs-78: updateAdmin

```
  public String updateAdmin(Model model,final
RedirectAttributes redirectAttributes)
```

Description: Show update password page with box for inputting new password.

Parameter: model-Model uses for getting the model.

redirectAttributes-RedirectAttributes uses for store the error message.

Return: updateAdmin.jsp page

MDs-79: addValidAdmin

```
public String addValidAdmin(@Valid Admin admin,
BindingResult bindingResult,Model model)
```

Description: Check the validation of the administrator's information before update into the database.

Parameter: model-Model uses for getting the model.

Admin-Admin uses for updating the admin.

bindingResult-BindingResult uses for keeping the invalid data.

Return: If there are invalid information, go to updateAdmin.jsp page, otherwise go to Home_page.jsp page.

CDs-18: LoginLogoutController

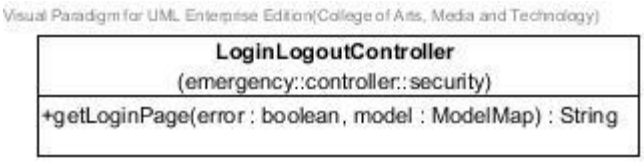

Figure 37 LoginLogoutController class

Method details

MDs-80: getLoginPage

```
public String
getLoginPage(@RequestParam(value="error", required=false)
boolean error, ModelMap model)
```

Description: For calling login page and check the error message.

Parameter:

error-boolean uses for check and show invalid login message.

model-ModelMap uses for getting the model.

Return: login.jsp page

CDs-19: Admin

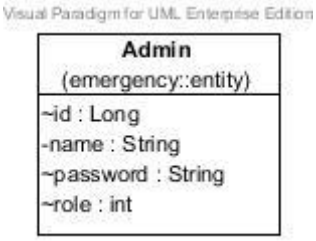

Figure 38 Admin class

Entities

ID	Name	Description	Type
1	id	Stored id	Long
2	name	Stored username	String
3	password	Stored password	String
4	role	Store user role	Int

CDs-20: AdminDAO

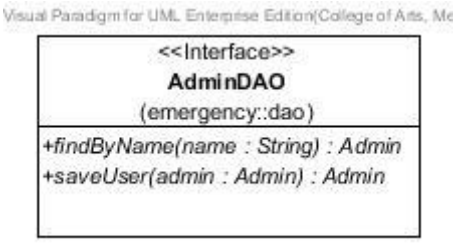

Figure 39 AdminDAO class

Method detail

MDs-81: findByName

```
public Admin findByName(String name)
```

Description: Find admin data from database by using name(Username)

Parameters: name-String use to search name(Username) in database

MDs-82: saveUser

```
public Admin saveUser(Admin admin)
```

Description: Add admin into the database

Parameters: admin-Admin object that will be add to database

CDs-21: AdminDAOImpl

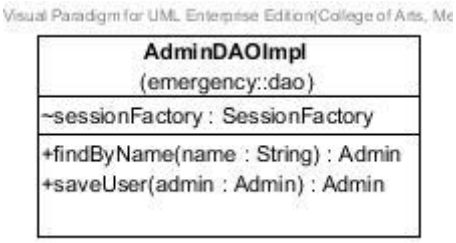

Figure 40 AdminDAOImpl class

Entities

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

Method detail

MDs-83: findByName

```
public Admin findByName(String name)
```

Description: Find admin data from the database by using name and session to query data.

Parameters: name-String use to search username to database

MDs-84: saveUser

```
public Admin saveUser(Admin admin)
```

Description: Add admin to database by using session to query data.

Parameters: admin-Admin object that will be add to database

CDs-22: AdminService

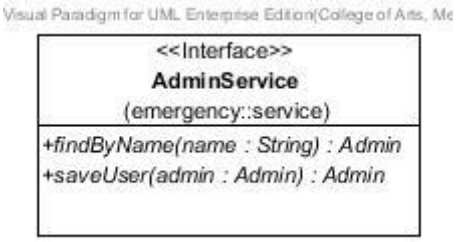

Figure 41 AdminService class

Method detail

MDs-85: findByName

```
public Admin findByName(String name)
```

Description: Find admin data

Parameters: name-String use to search name(username) in database

MDs-86: saveUser

```
public Admin saveUser(Admin admin)
```

Description: Add admin object to database

Parameters: admin-Admin object that will be add to database

CDs-23: AdminServiceImpl

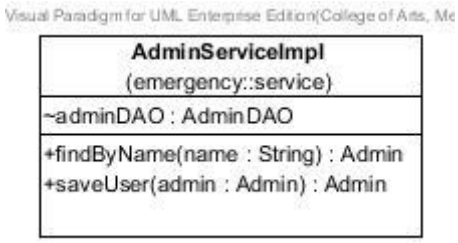

Figure 42 AdminServiceImpl

Entities

ID	Name	Description	Type
1.	dao	Contained AdminDAO classfor using method	AdminDAO

Method detail

MDs-87: findByName

```
public Admin findByName(String name)
```

Description: Find admin data through *findByName* method of AdminDAO class.

Parameters: name-String use to search name(username) to database

MDs-88: saveUser

```
public Admin saveUser(Admin admin)
```

Description: Add admin through *saveUser* method of AdminDAO class.

Parameters: admin-Admin object that will be add to database

CDs-24: UserDetailsServiceImpl

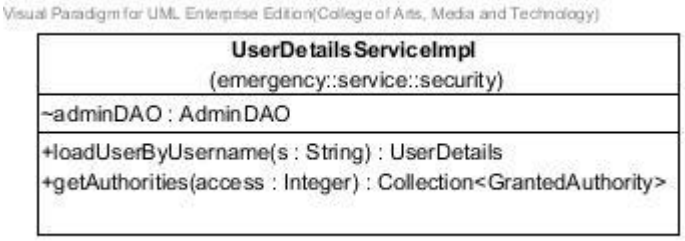

Figure 43 UserDetailsServiceImpl

Entities

ID	Name	Description	Type
1.	adminDAO	Contained AdminDAO classfor using method	AdminDAO

Method detail

MDs-89: loadByUsername

```
public UserDetails loadByUsername(String username)
```

Description: Check username and password in database

Parameters: username-String use to search username and password in database

MDs-90: getAuthorities

```
public Collection<GrantedAuthority>getAuthorities
(Integer access)
```

Description: Check role of user

Parameters: access-Integer use for check role of user

3.3 Database Design

3.3.1 Mobile Part

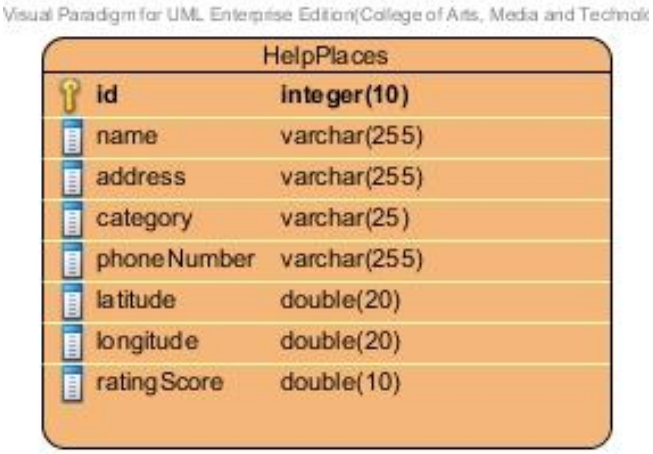

Figure 44 Database Design of Mobile Part

Figure 44 show database design for storing data in the mobile part

3.4 Sequence Diagram

1.3.1 Mobile Part

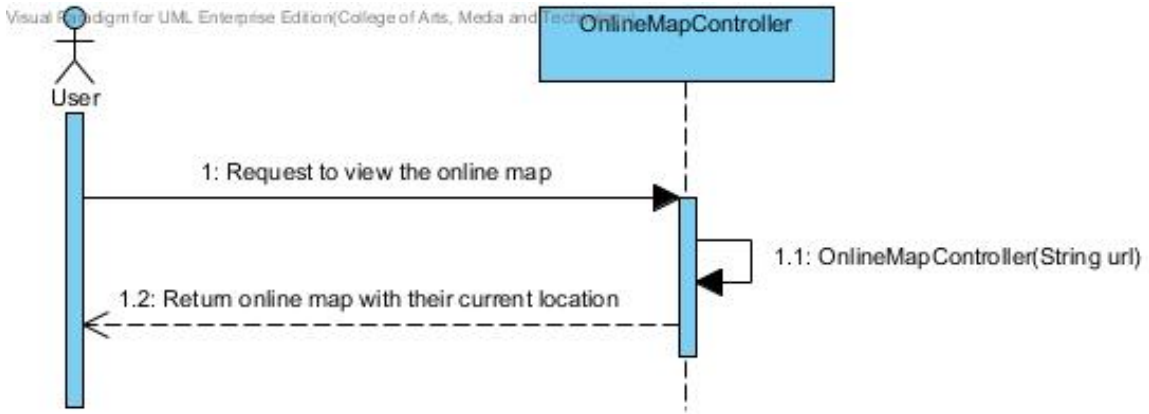

Figure 45 View online map diagram (SDm-01)

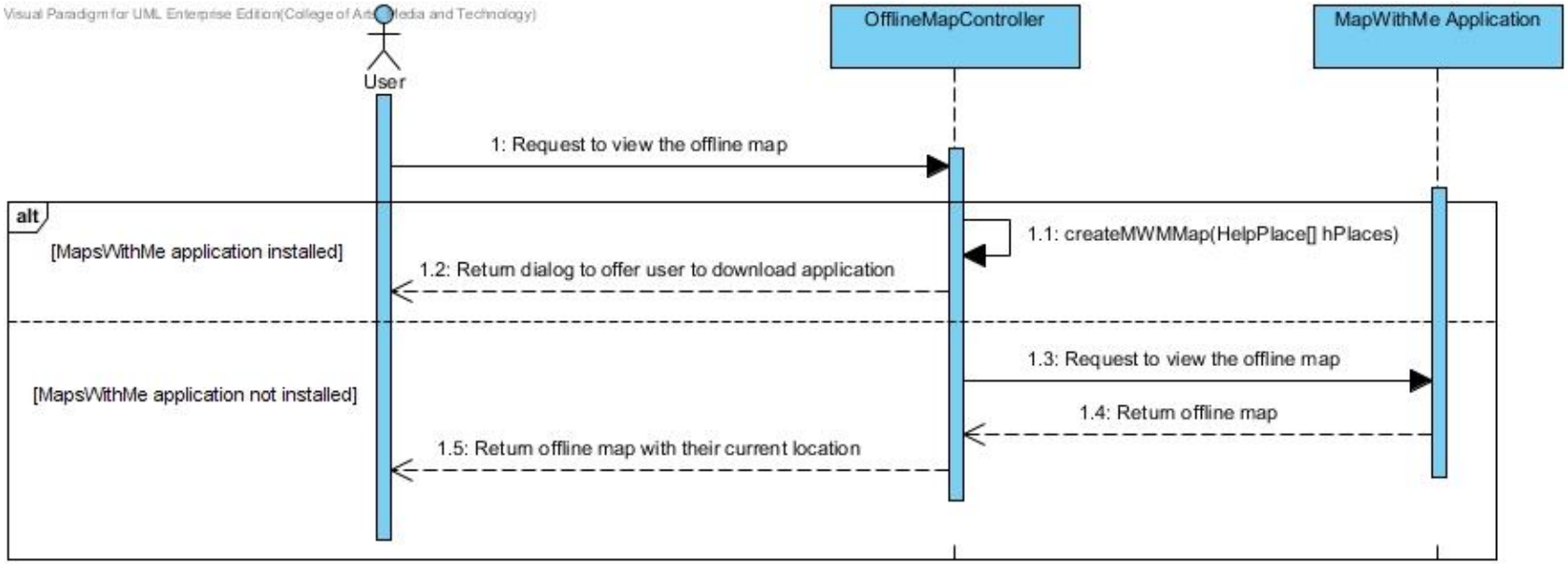

Figure 46 View offline map diagram (SDm-02)

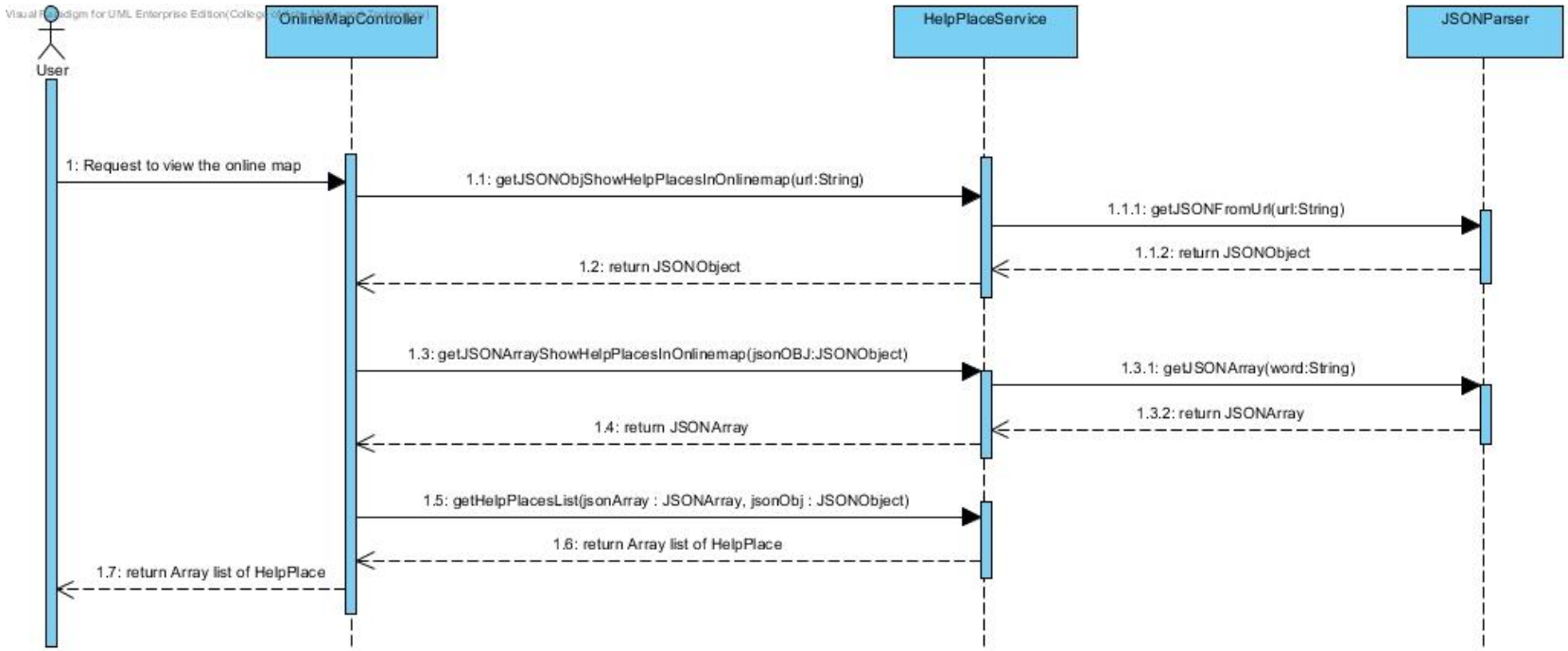

Figure 47 View the help places in online map diagram (SDm-03)

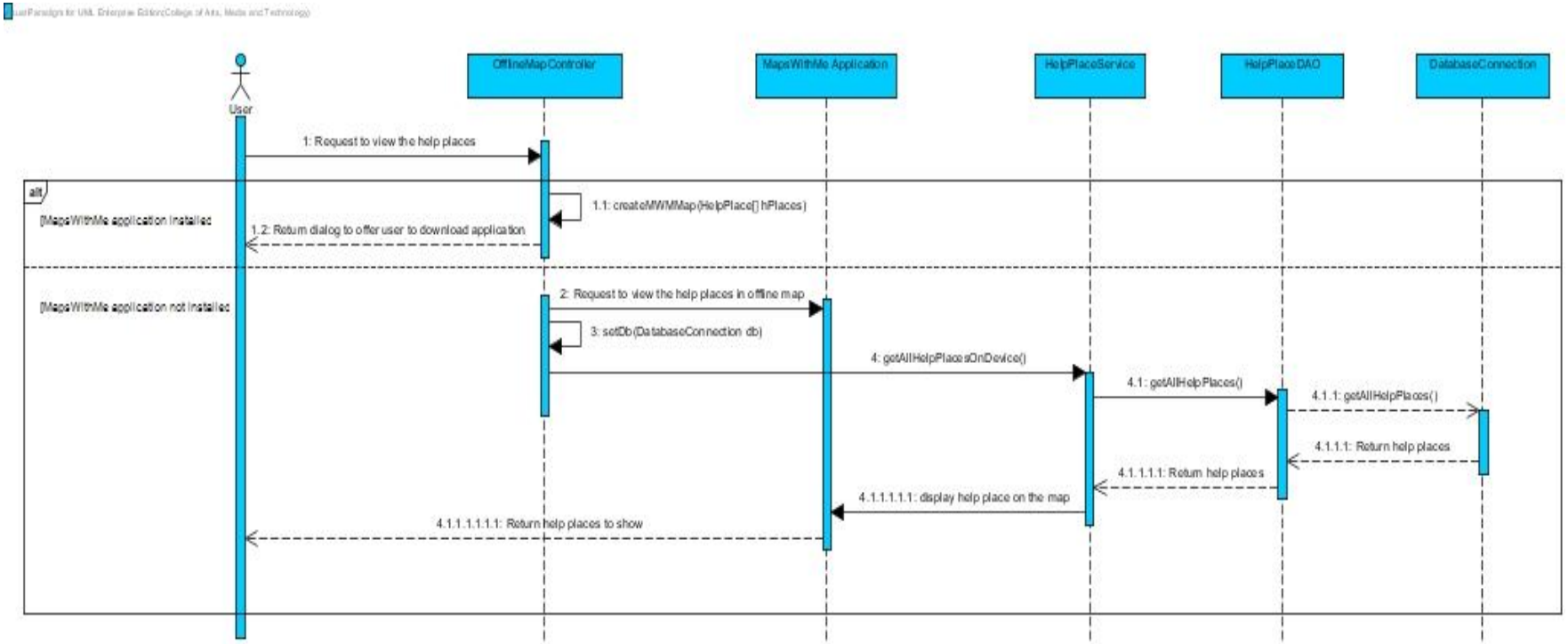

Figure 48 View the help places in offline map diagram (SDm-04)

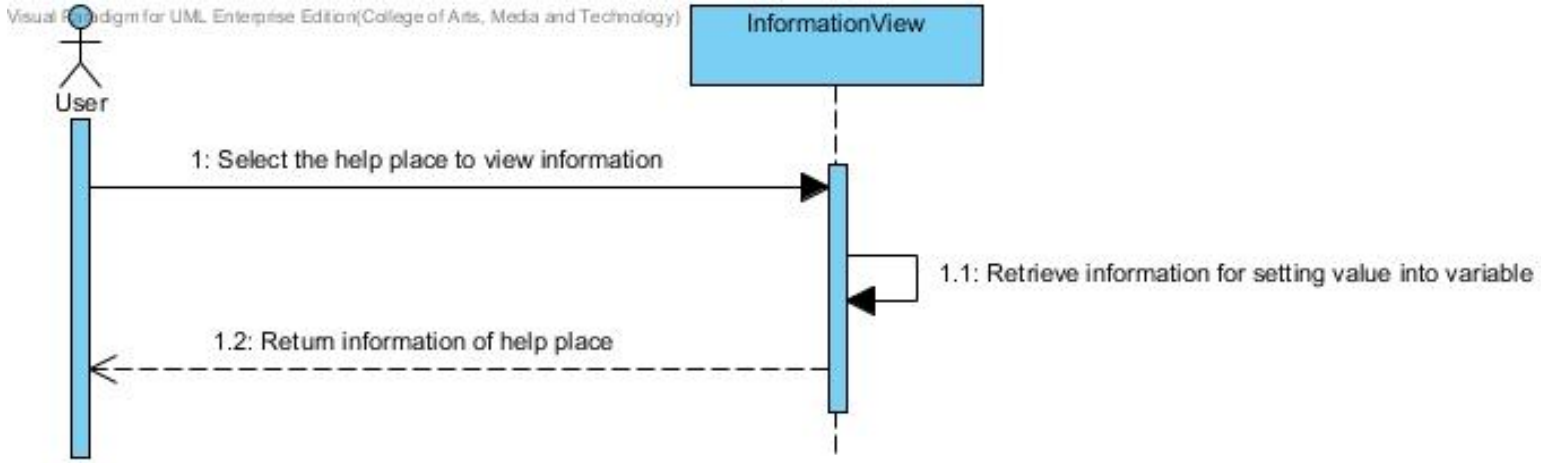

Figure 49 View information of each help place in online map diagram (SDm-05)

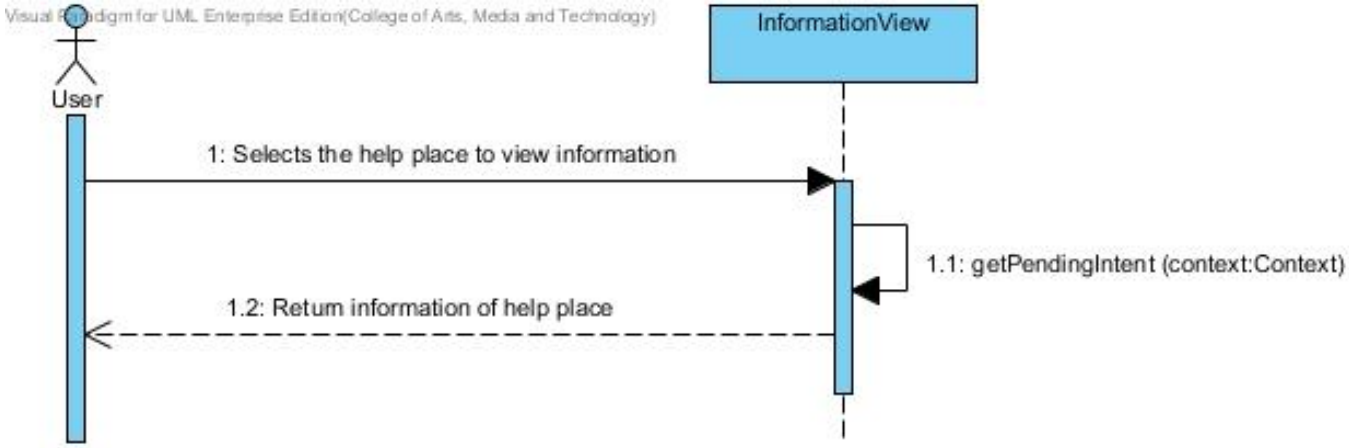

Figure 50 View information of each help place in offline map diagram (SDm-06)

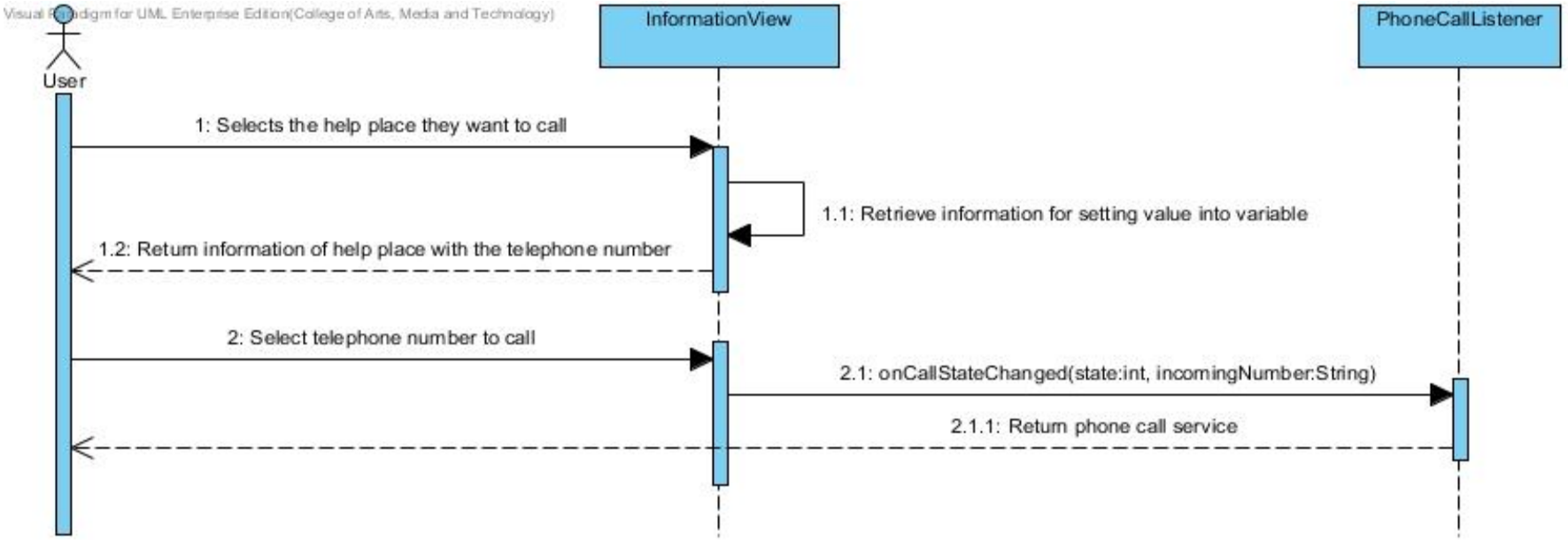

Figure 51 Make emergency call to each help place in online map diagram (SDm-07)

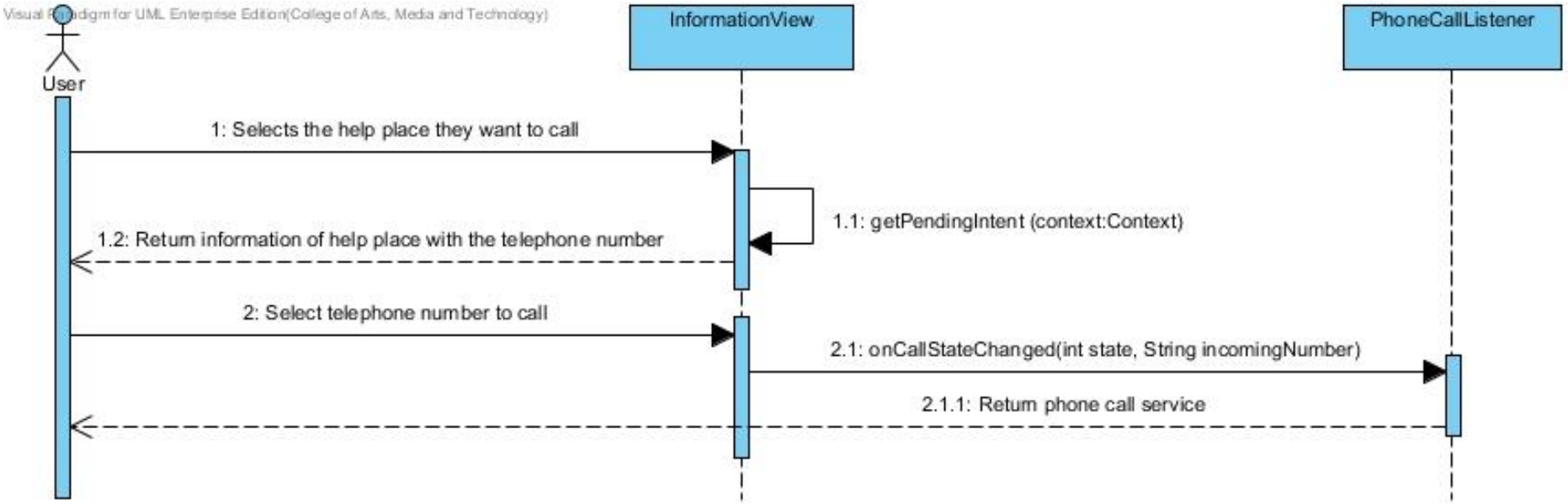

Figure 52 Make emergency calls to each help place in offline map diagram (SDm-08)

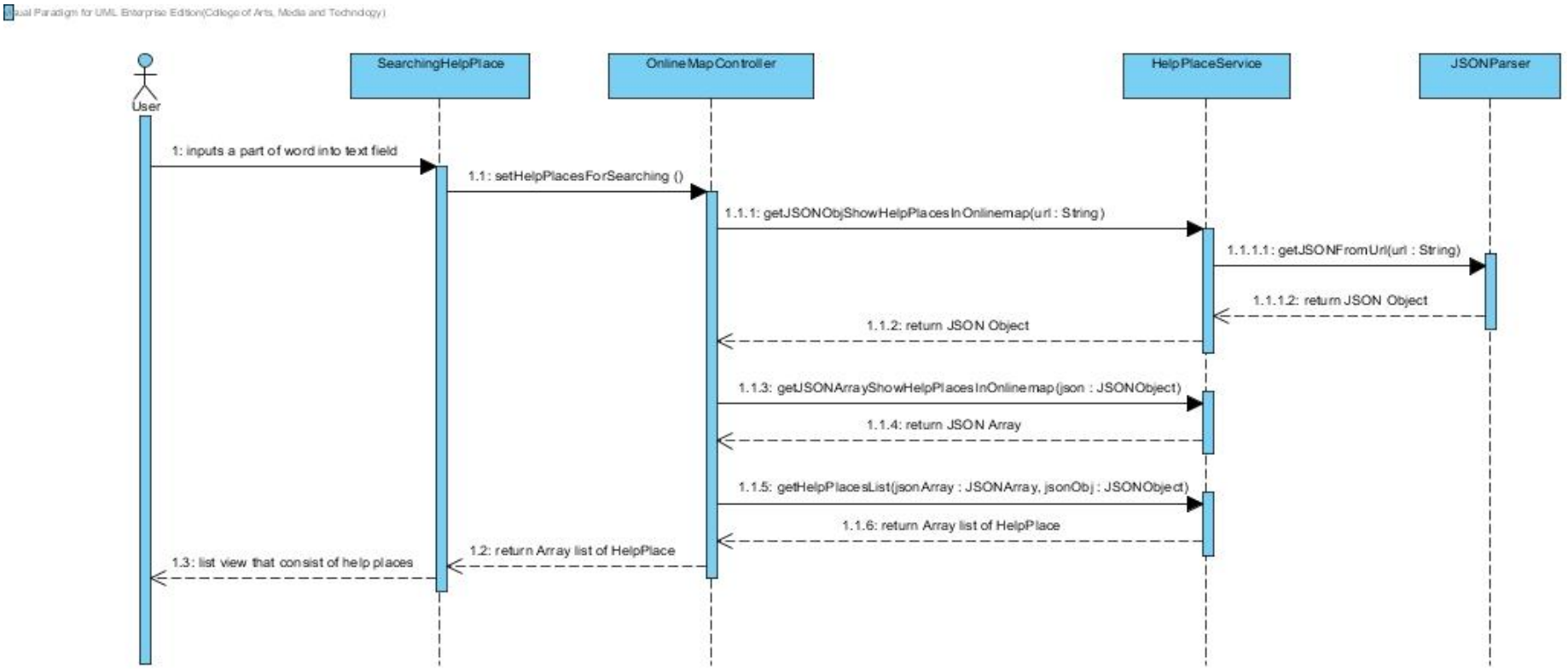

Figure 53 Search help place's name by keyword in online map diagram (SDm-09)

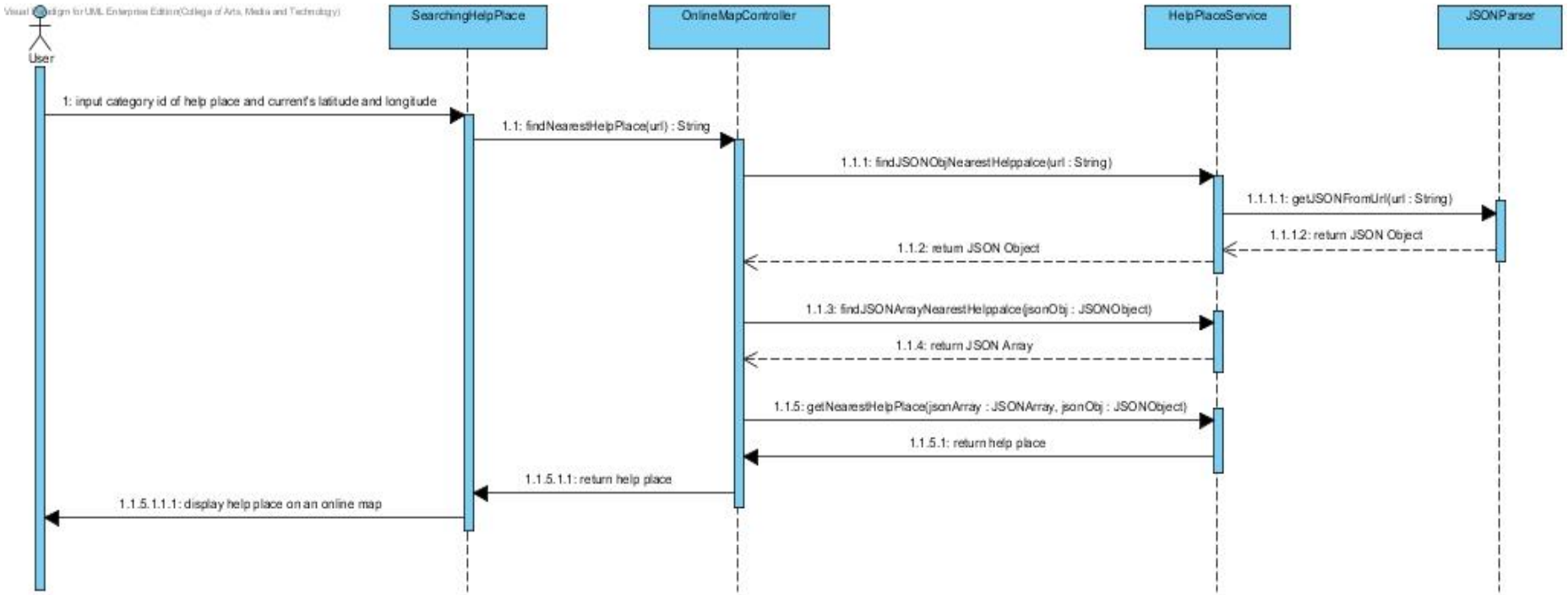

Figure 54 Find the nearest help place diagram (SDm-10)

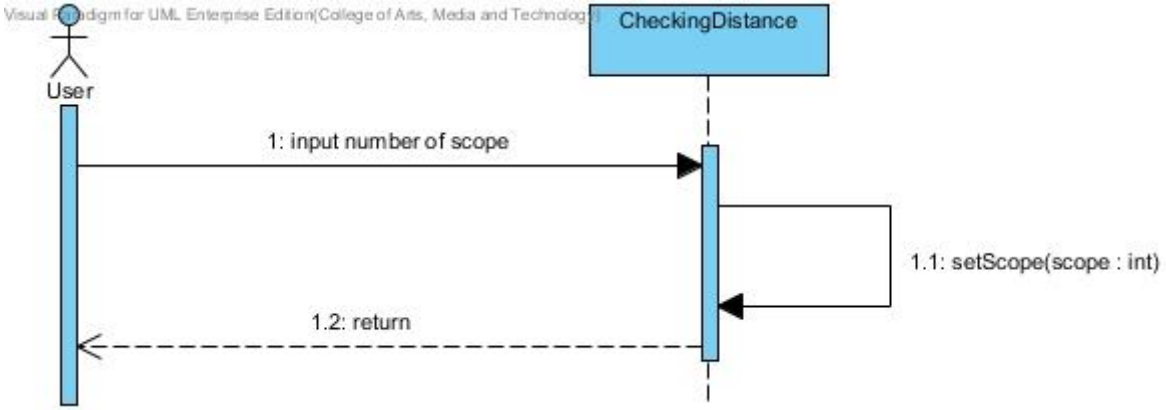

Figure 55 Set the scope for downloading data diagram (SDm-11)

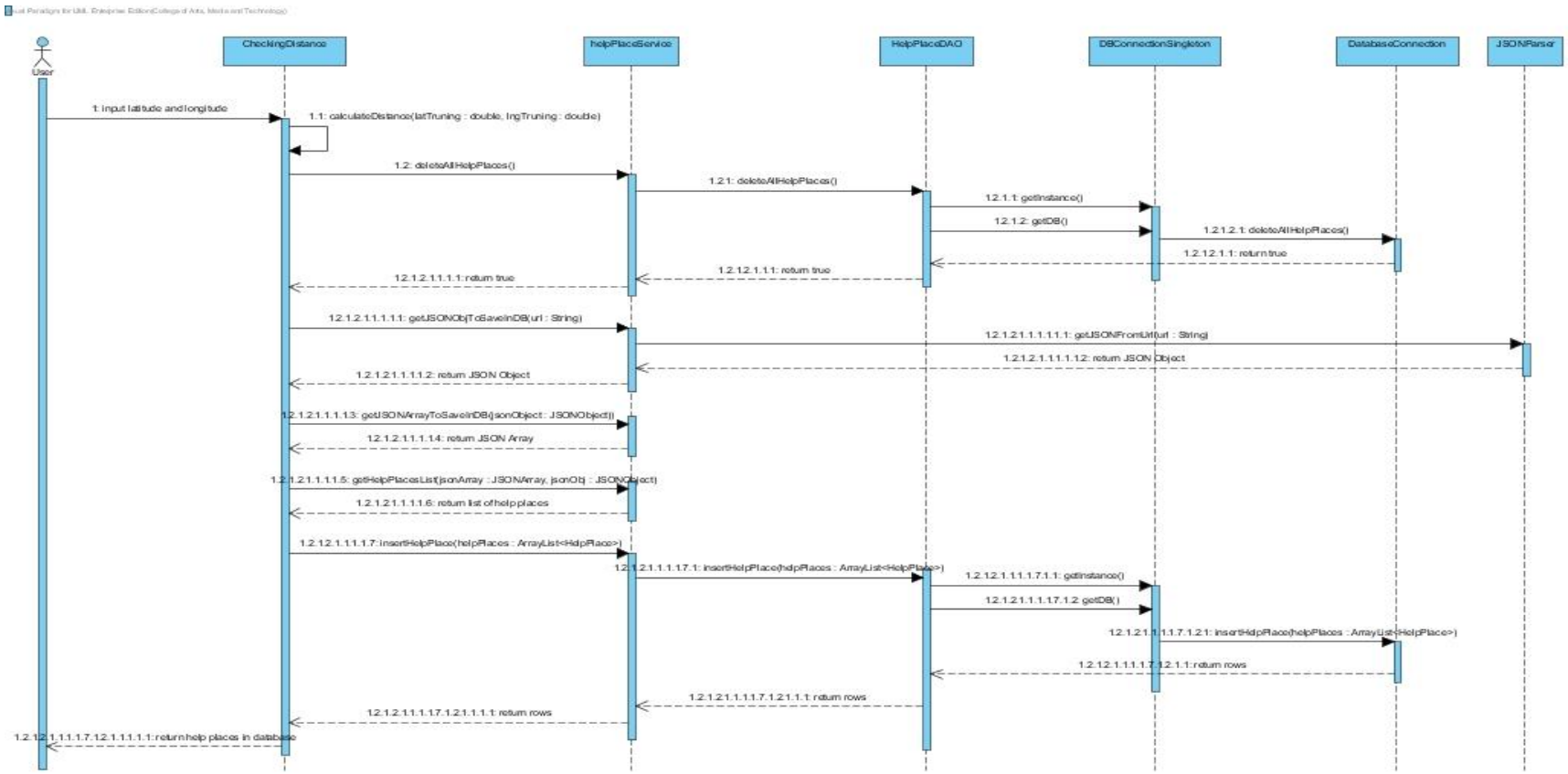

Figure 56 Collect help place information automatically diagram (SDm-12)

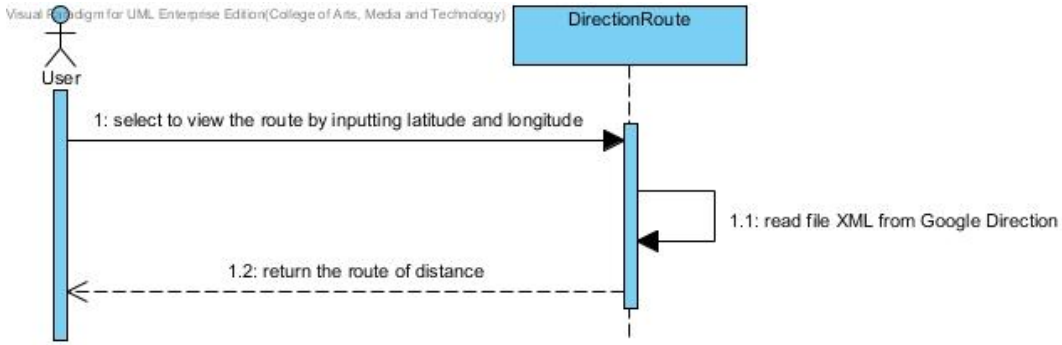

Figure 57 View the route of distance between the current locations of user to the destination (SDm-13)

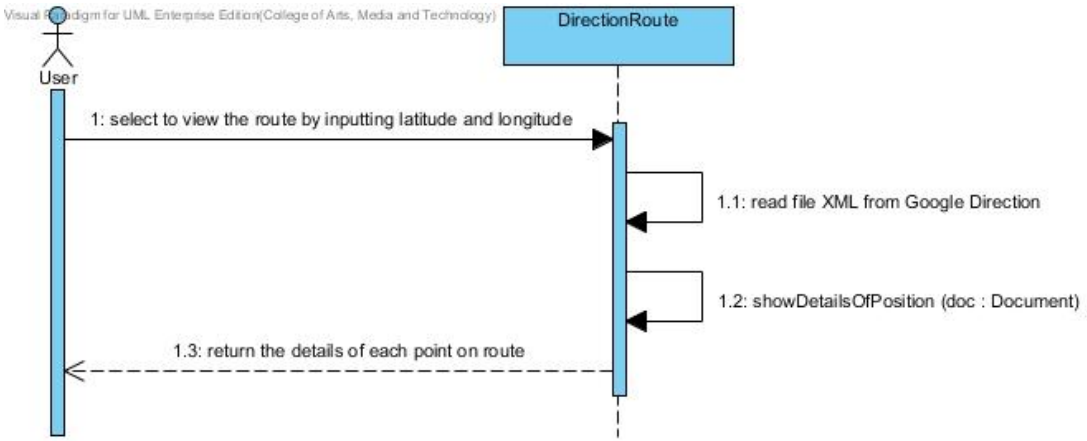

Figure 58 View details of each point on routing the direction (SDm-14)

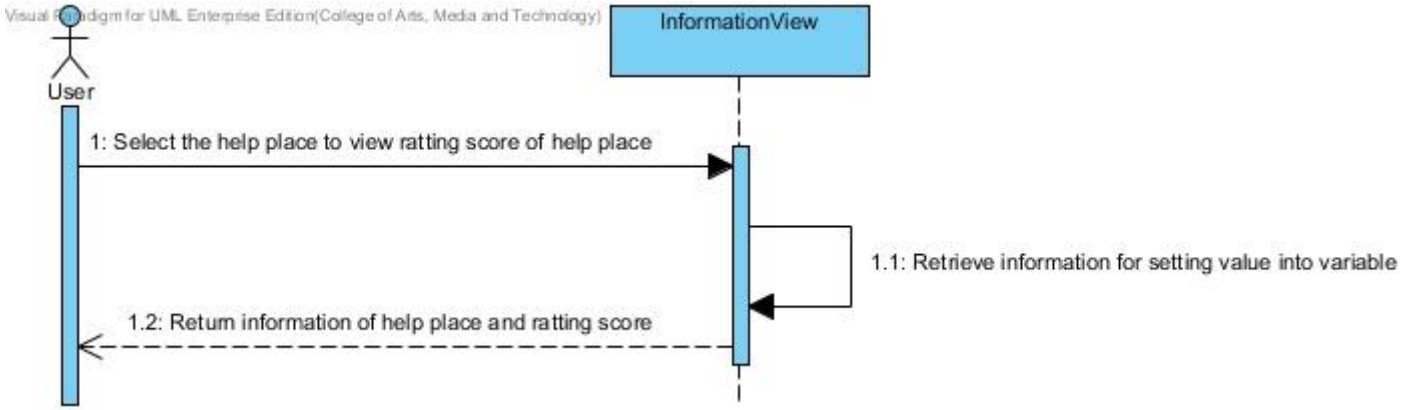

Figure 59 View average rating score of each help place in online map (SDm-15)

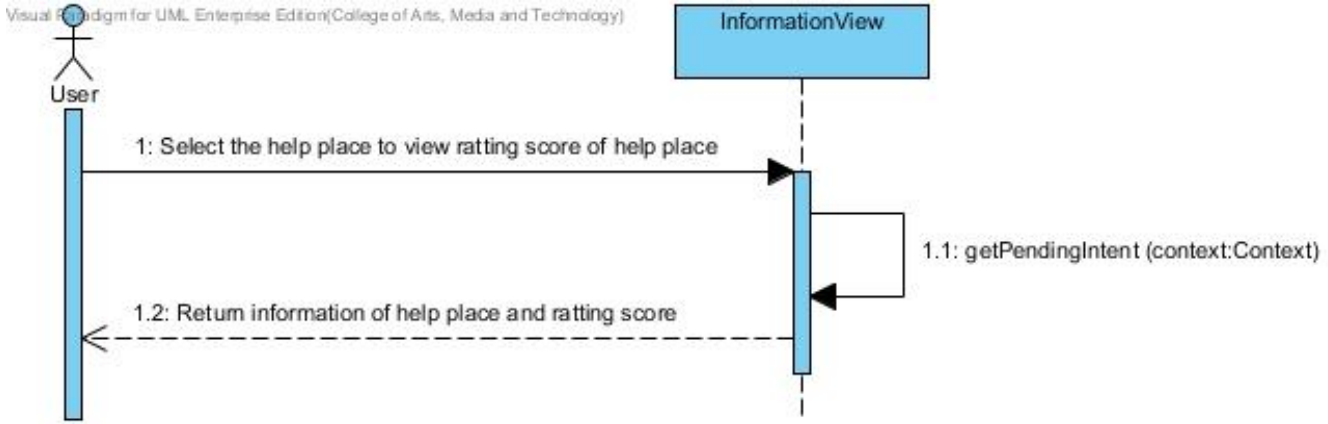

Figure 60 View average rating score of each help place in offline map (SDm-16)

231

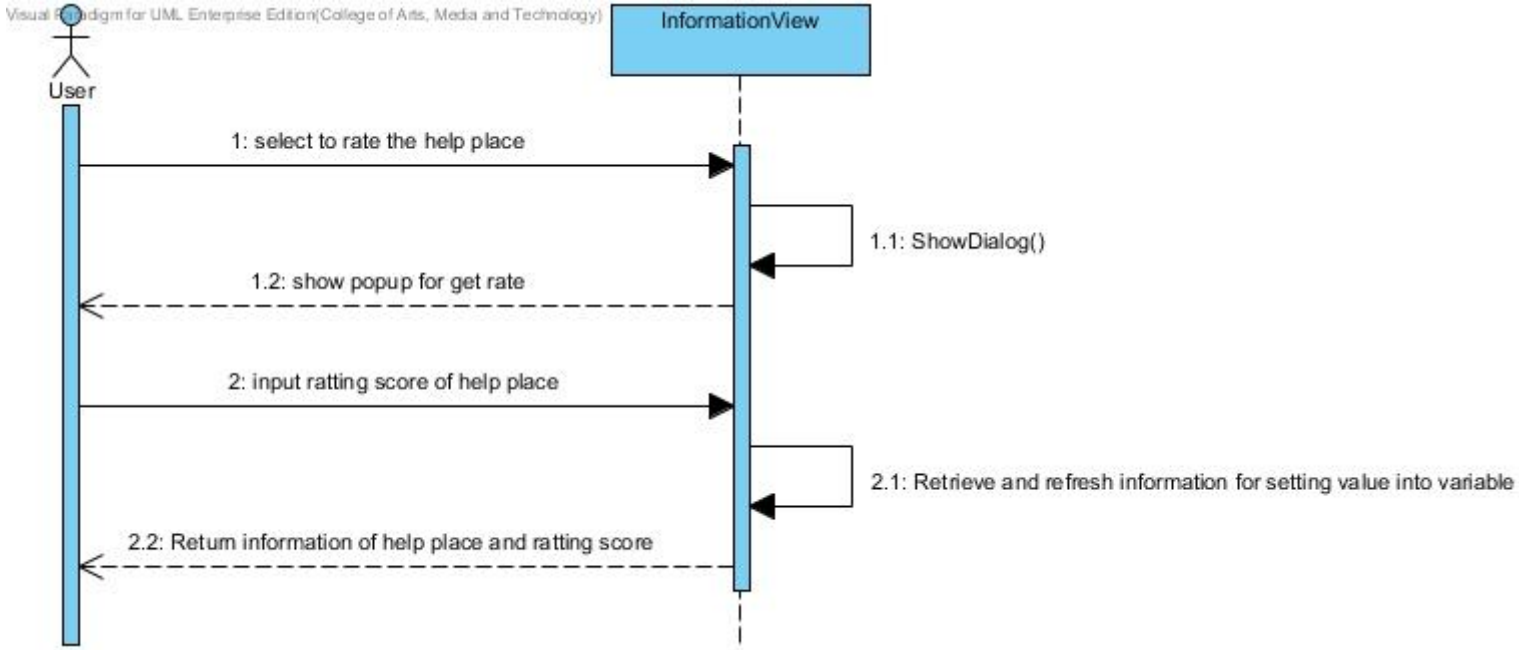

Figure 61 Rate the help place in online map (SDm-17)

232

3.3.2 Server Part

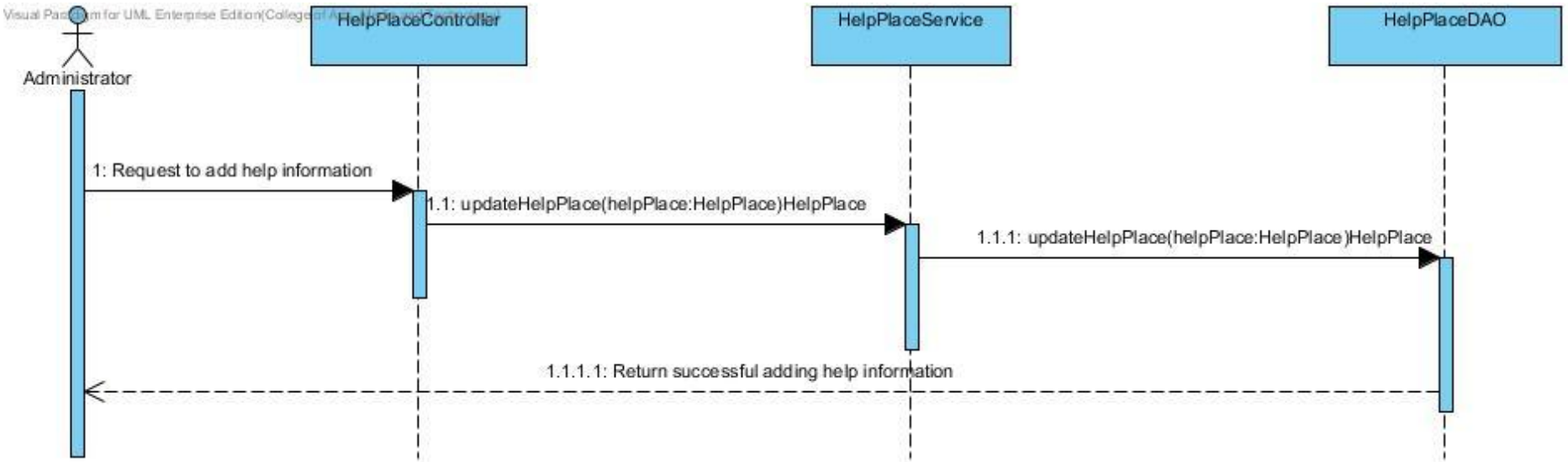

Figure 62 Add help place sequence diagram (SDs-01)

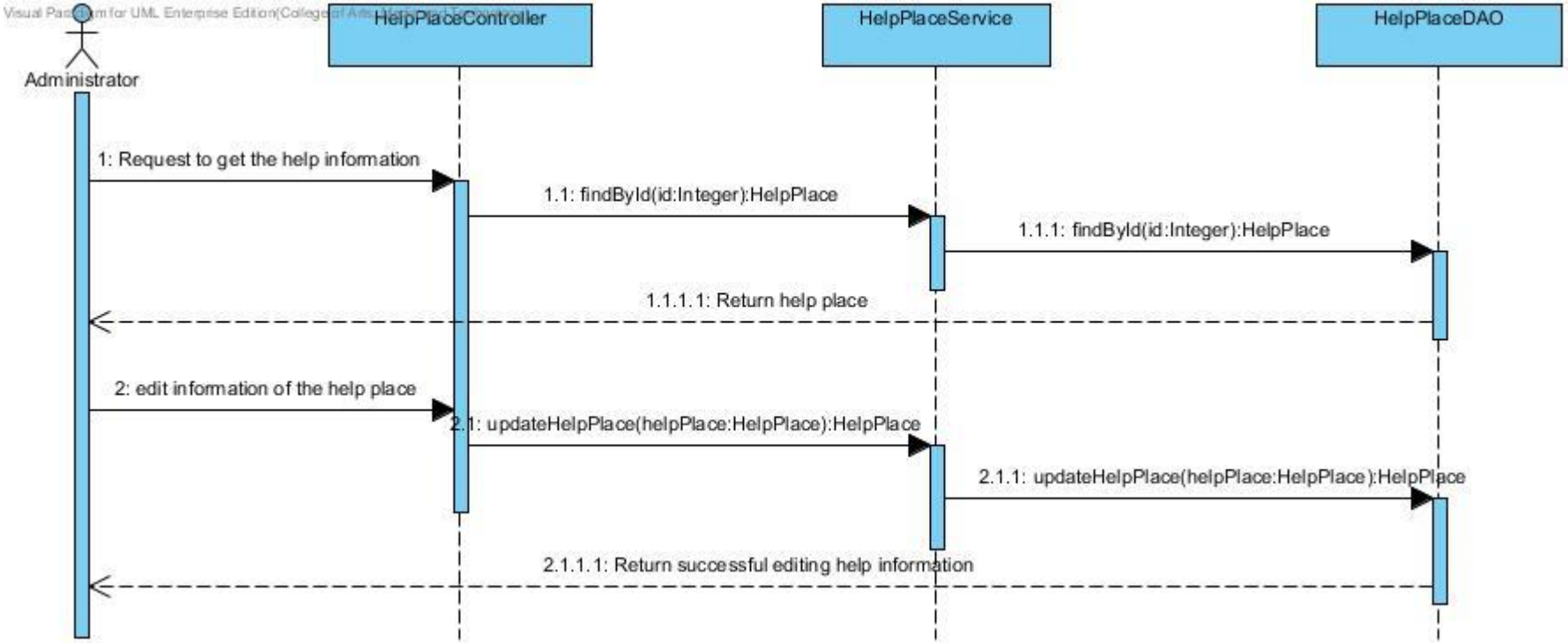

Figure 63 Edit help place sequence diagram (SDs-02)

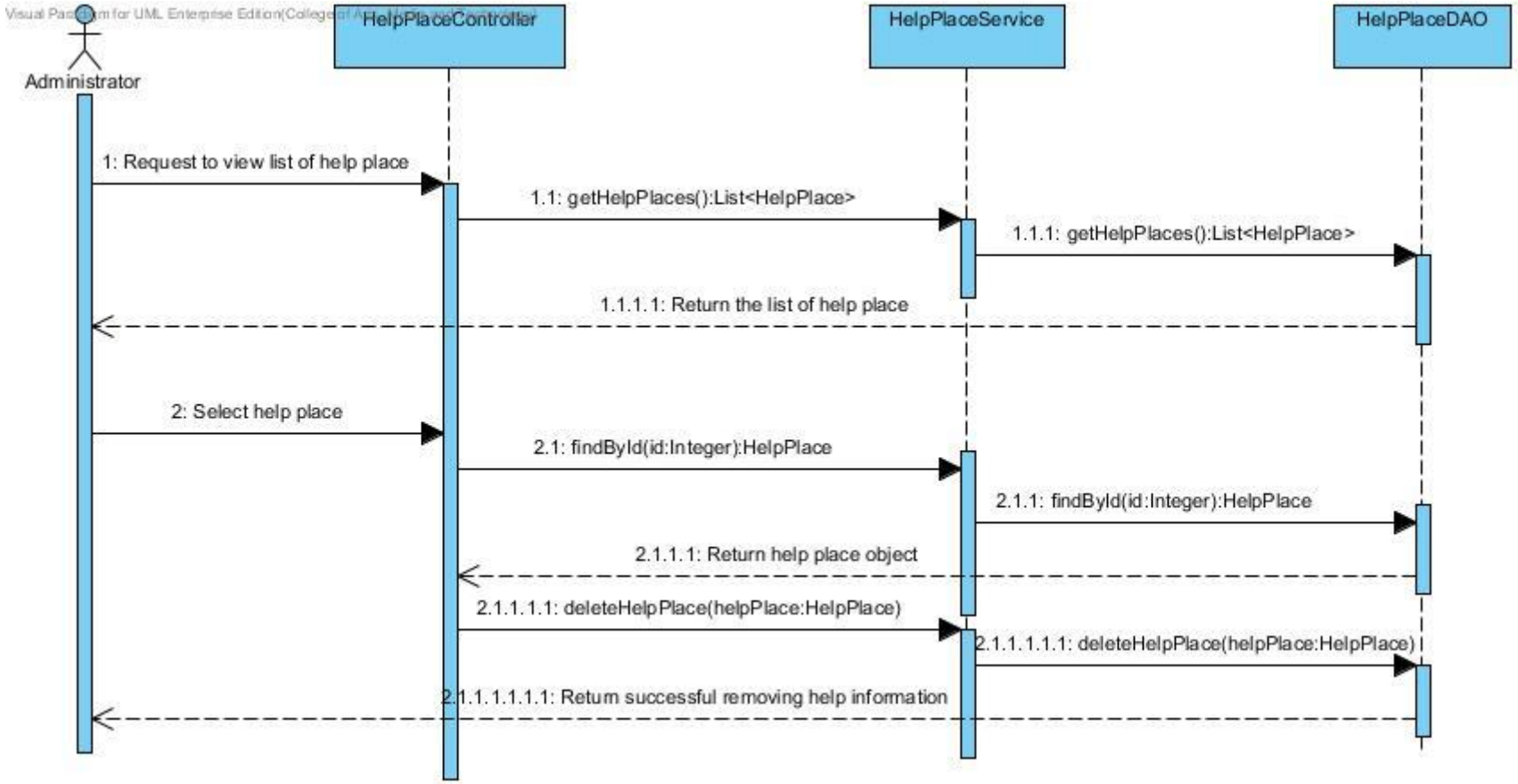

Figure 64 Remove help place sequence diagram (SDs-03)

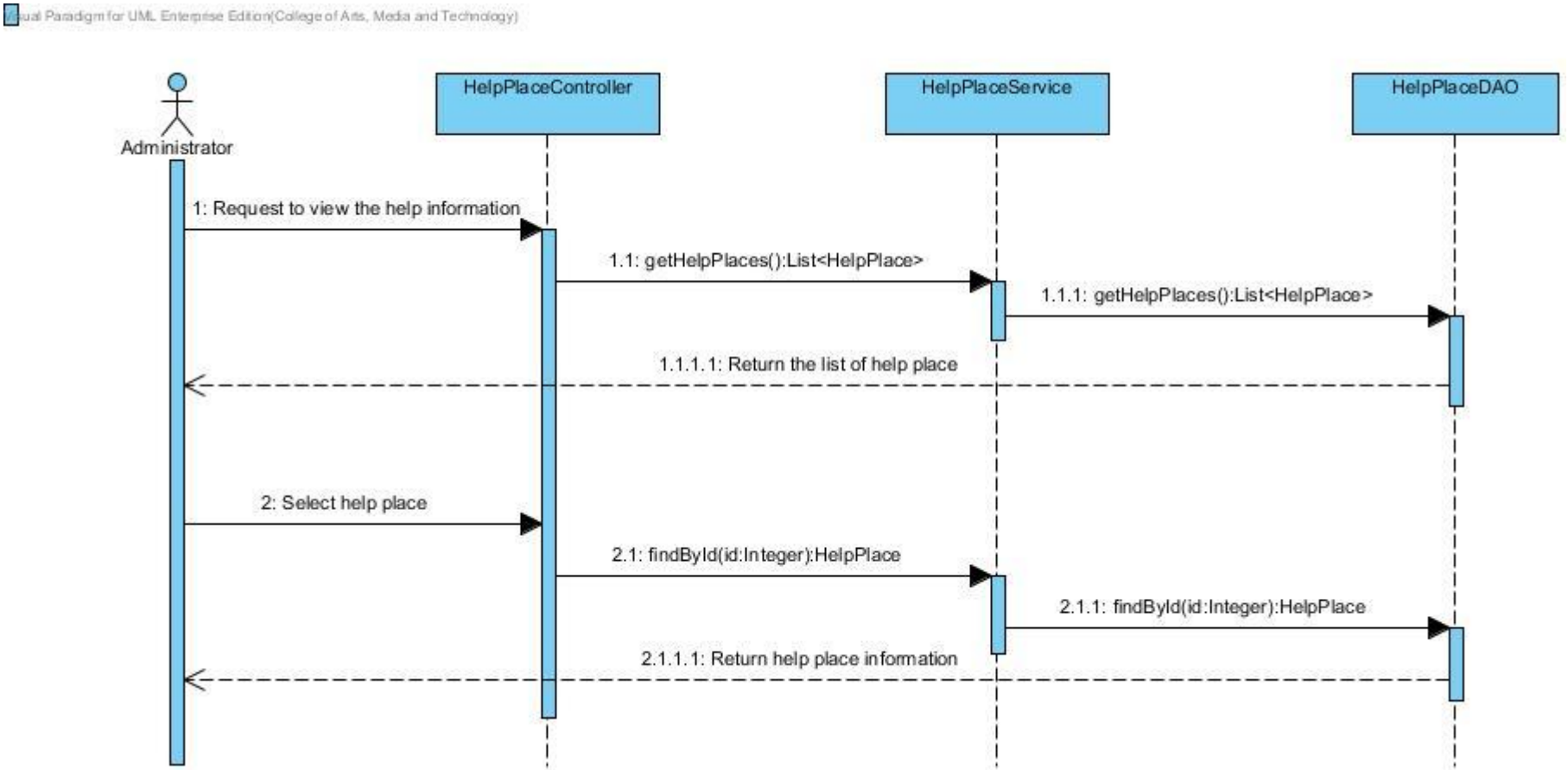

Figure 65 View information sequence diagram (SDs-04)

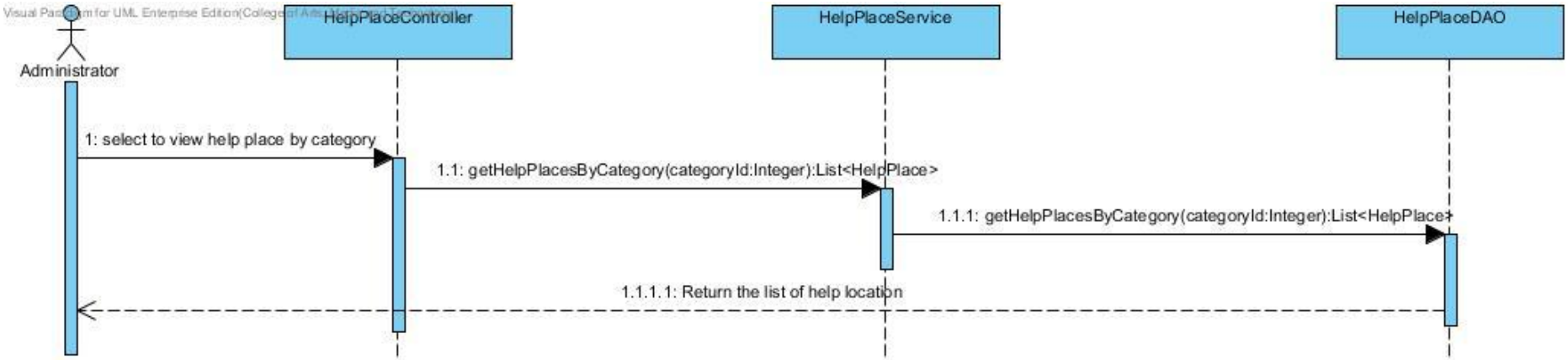

Figure 66 Search by category sequence diagram (SDs-05)

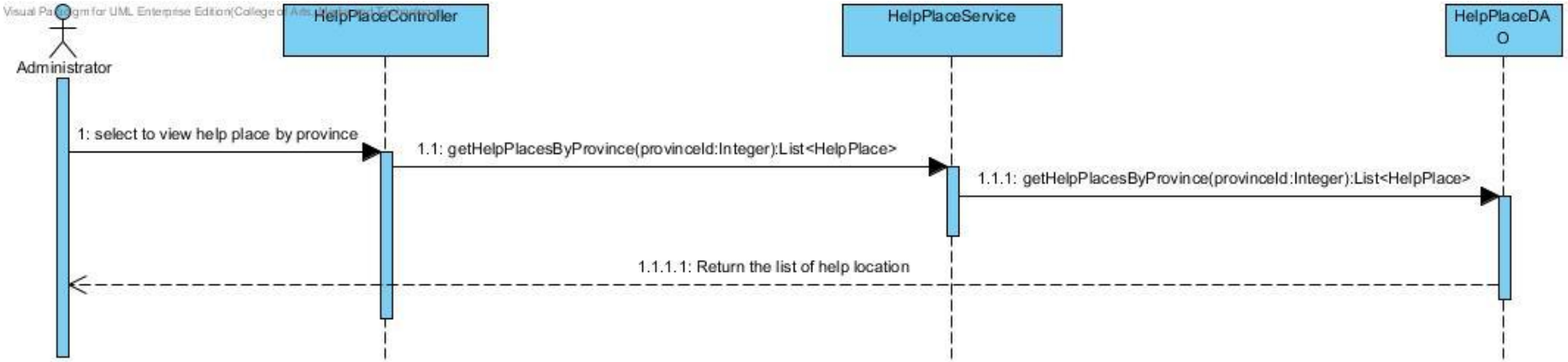

Figure 67 Search by province sequence diagram (SDs-06)

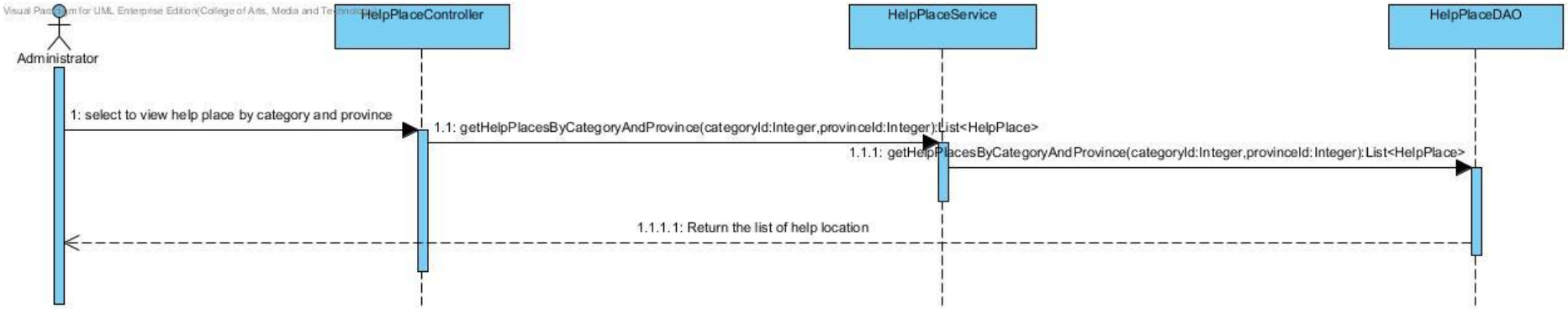

Figure 68 Search by category and province sequence diagram (SDs-07)

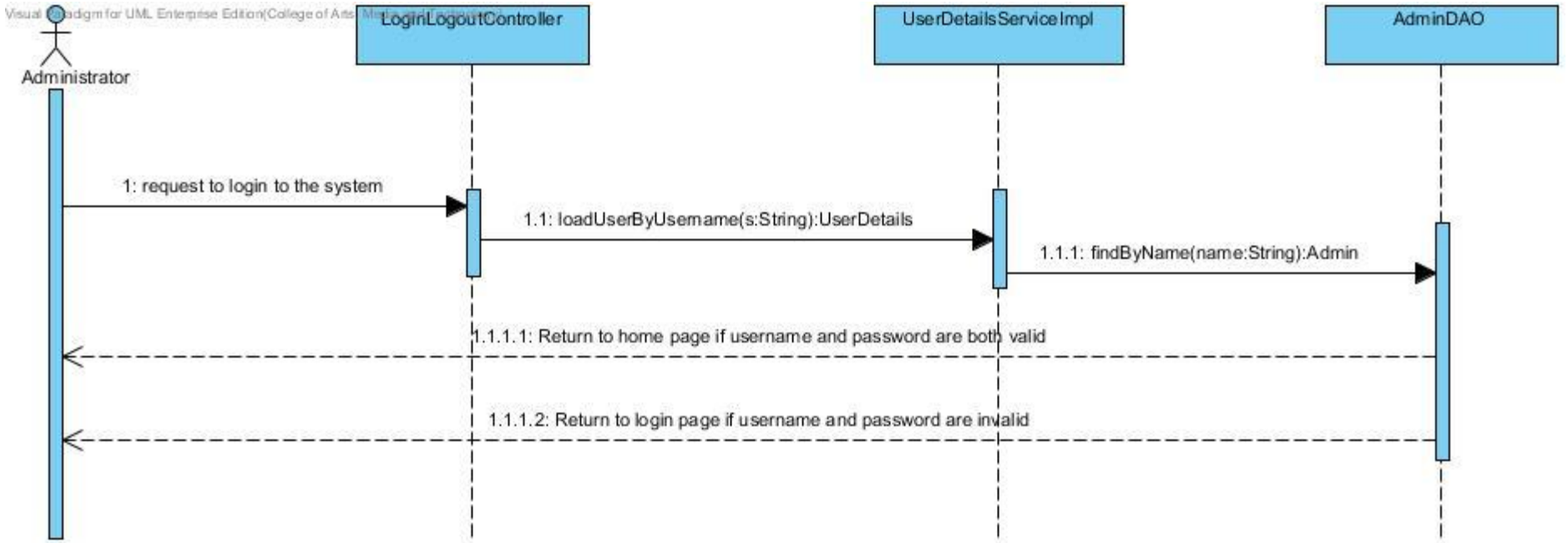

Figure 69 Login to the system (SDs-08)

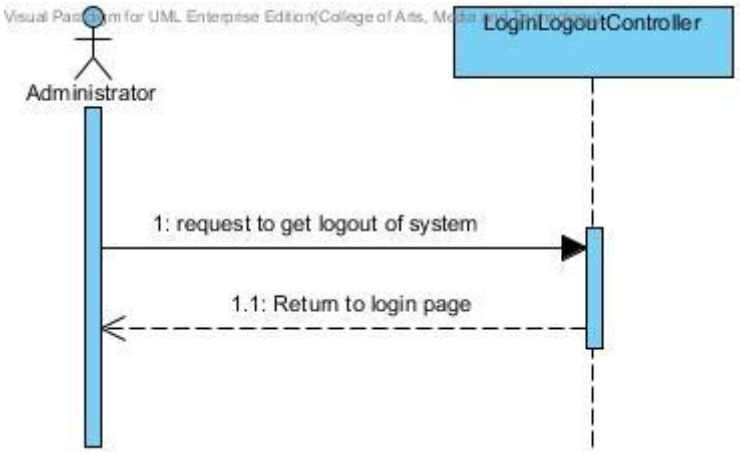

Figure 70 Logout of the system (SDs-09)

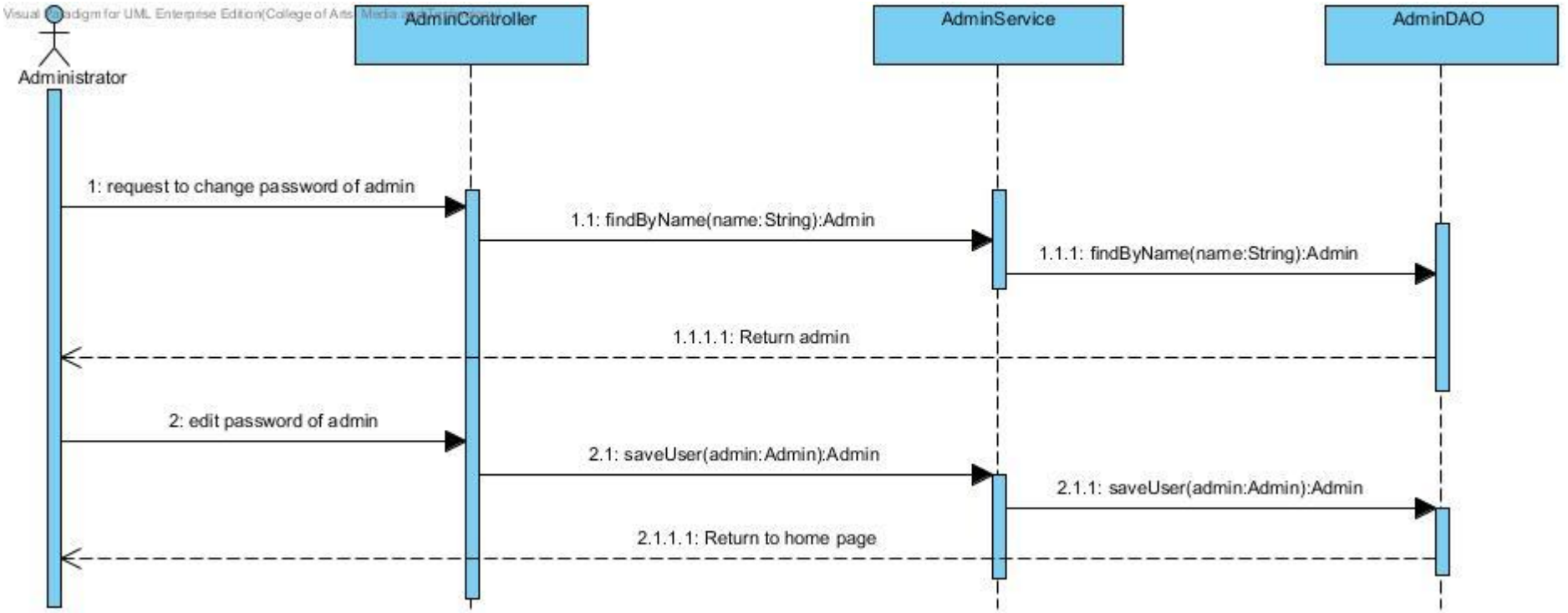

Figure 71 Update account's password (SDs-10)

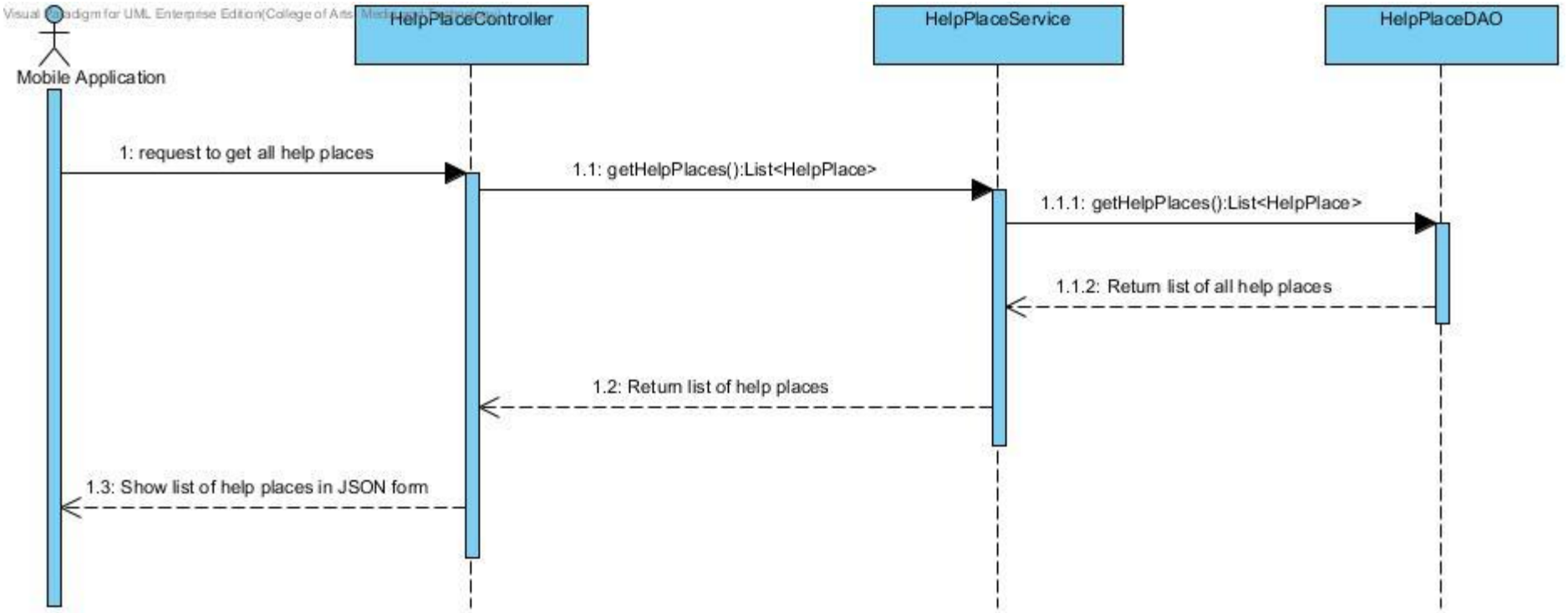

Figure 72 Get all help places diagram (SDs-11)

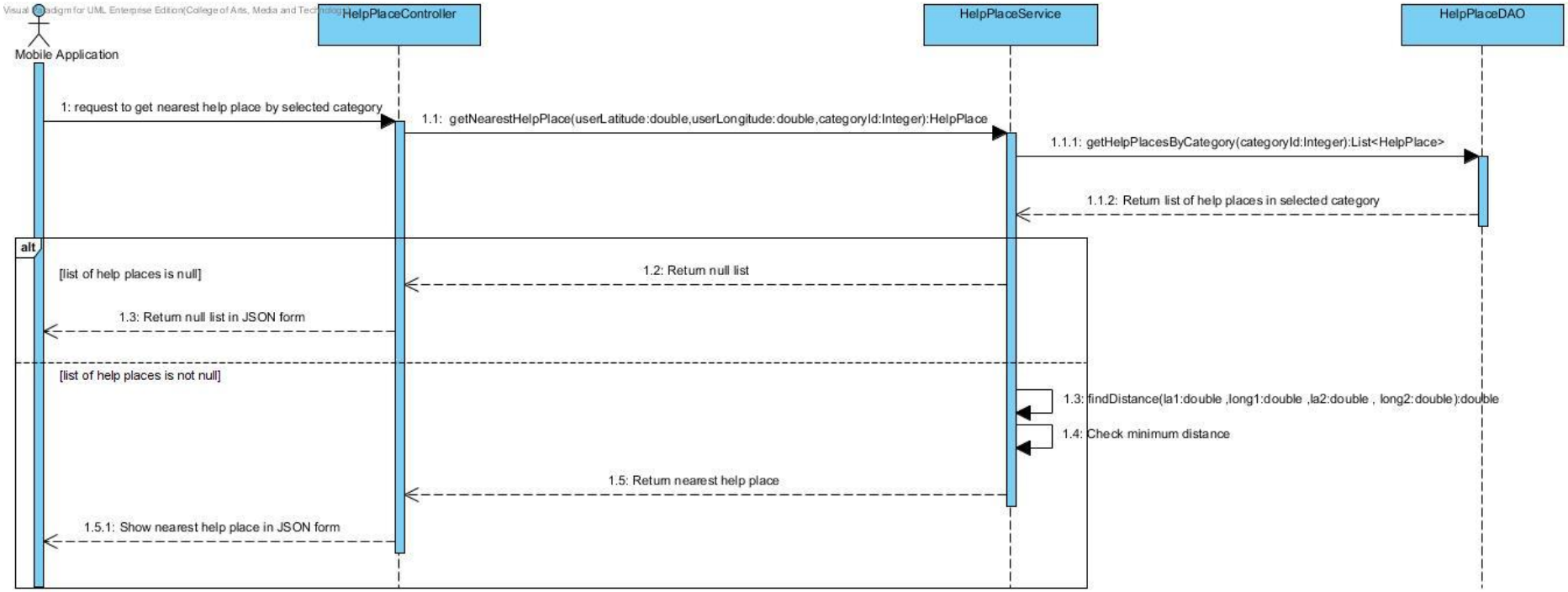

Figure 73 Get nearest help place diagram (SDs-12)

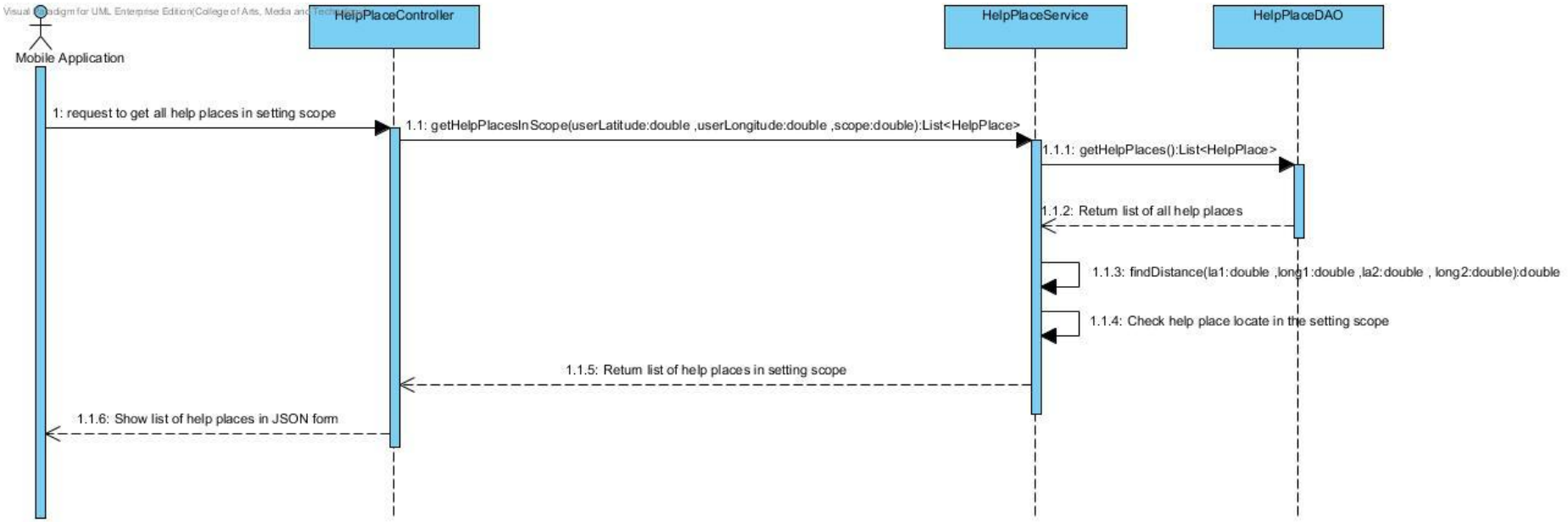

Figure 74 Get all help places in setting scope diagram (SDs-13)

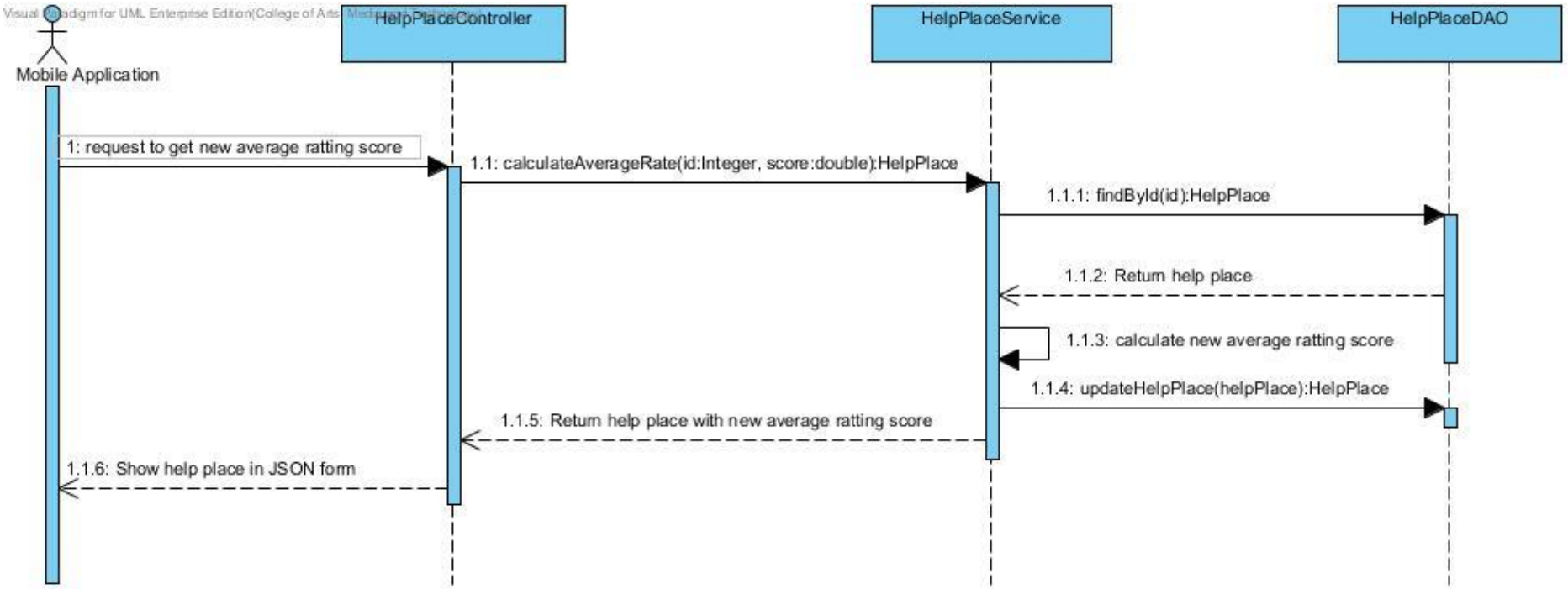

Figure 75 Retrieve new average rating score (SDs-14)

Chapter Four | User Interface Design

Mobile Part

UI name: Start Page (UI-01)

Include Requirement: UCm-01, UCm-02, UCm-03, UCm-09, UCm-10

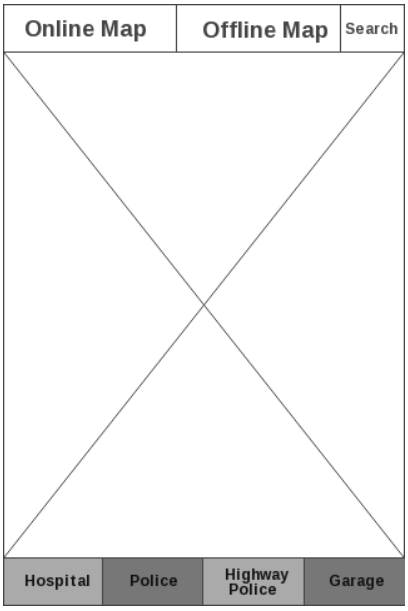

Figure 76 Start Page (UI-01)

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

- Online map
- Offline map
- Search

Buttons are used to find nearest help place by category hospital, police station, highway police and garage.

UI name: Show Online map (UI-02)

Include Requirement: UCm-01, UCm-03, UCm-09, UCm-10, UCm-12

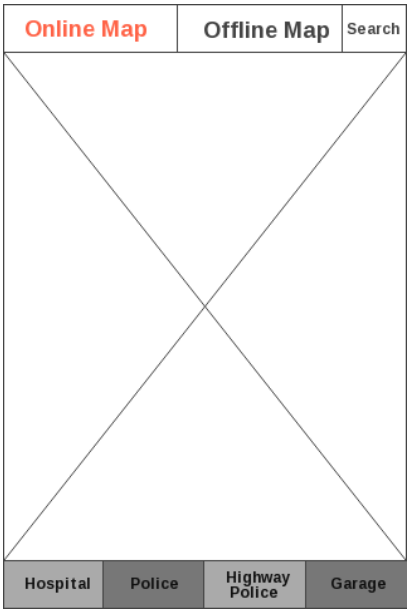

Figure 77 Show Online map (UI-02)

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-03)

Include Requirement: UCm-02

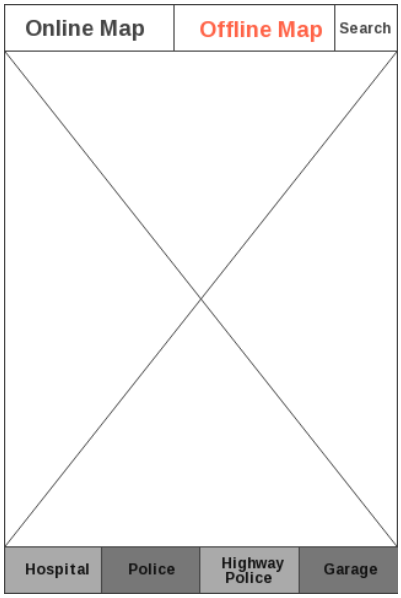

Figure 78 Connect Offline map (UI-03)

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-04)

Include Requirement: UCm-02, UCm-04, UCm-12

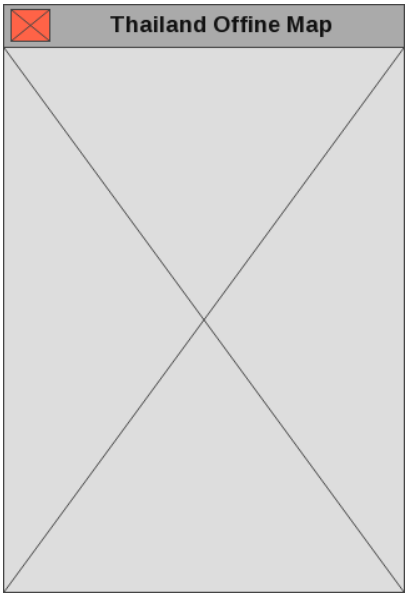

Figure 79 Show Offline map (UI-04)

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-05)

Include Requirement: UCm-02, UCm-04

Figure 80 MapsWithMe application installed (UI-05)

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-06)

Include Requirement: UCm-04, UC,-06

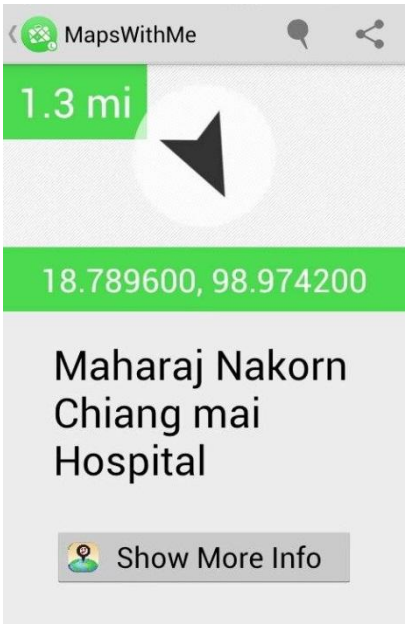

Figure 81 MapsWithMe application (UI-06)

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-07)

Include Requirement: UCm-05, UCm-06, UCm-07, UCm-08, UCm-15, UCm-16

Figure 82 Show Information Page (UI-07)

The Show Information page will show latitude, longitude, address, category, telephone number, average rating score and call button. On show information page, the user can call to help place directly from the application.

UI name: Search Help place Page (UI-08)

Include Requirement: UCm-09

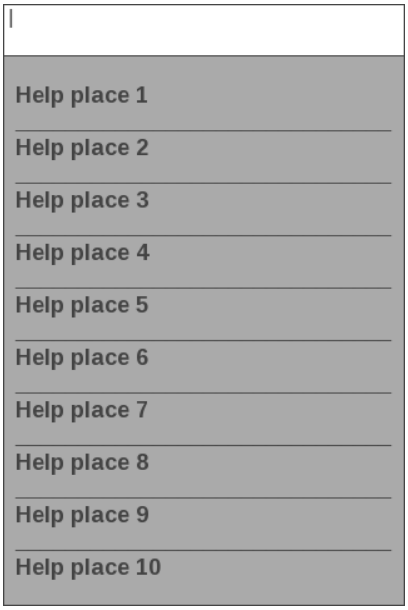

Figure 83 Search Help place Page (UI-08)

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.

UI name: Setting scope (UI-09)

Include Requirement: UCm-11

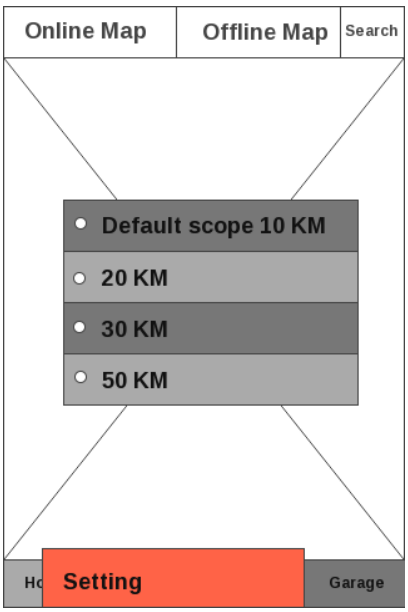

Figure 84 Setting Scope (UI-09)

Setting Scope that will provide menu bar for click setting menu before show the radio button with umber of scope in kilometer.

UI name: Routing direction (UI-10)

Include Requirement: UCm-13, UCm-14

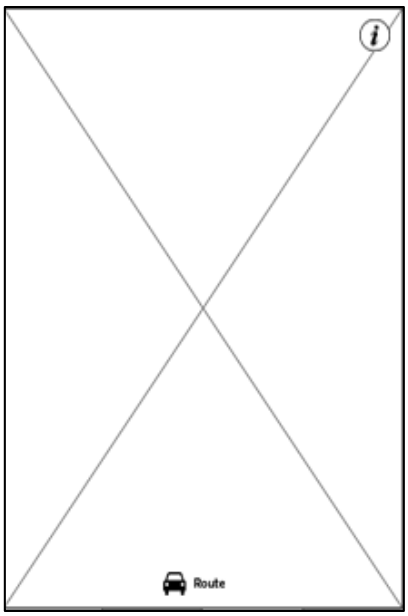

Figure 85 Routing directions (UI-10)

Routing scope direction that show the map with polyline from user's current location to the destination.

UI name: Give rating score (UI-11)

Include Requirement: UCm-17

Figure 86 Give rating score (UI-11)

The system provides dialog popup to get the rating score from the user. The user can give the rating that rating is less than the number of stars that are provided.

Server Part

UI name: Home Page (UI-12)

Include Requirement: UCs-03, UCs-05, UCs-06, UCs-07, UCs-09

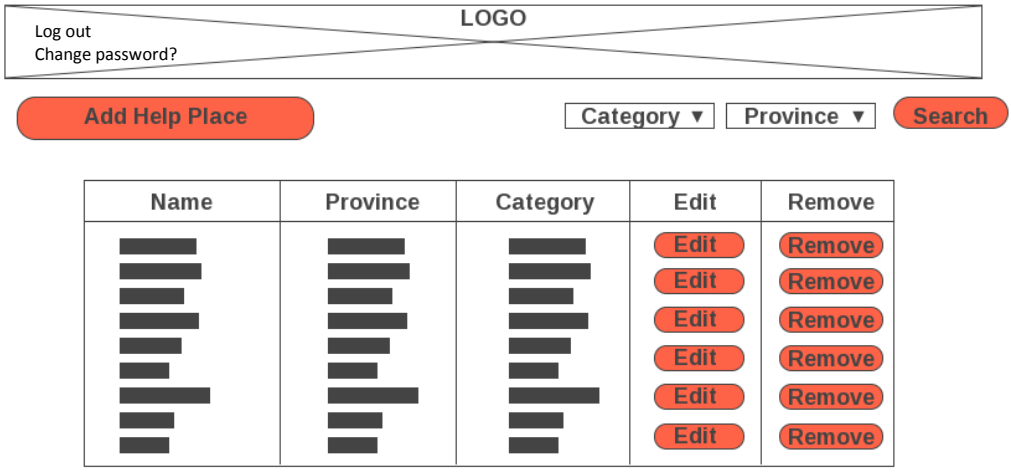

Figure 87 Home Page (UI-12)

The home page includes 3 components. The top of the page is a logo of web page, link to change password, and link to logout. 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-13)

Include Requirement: UCs-03

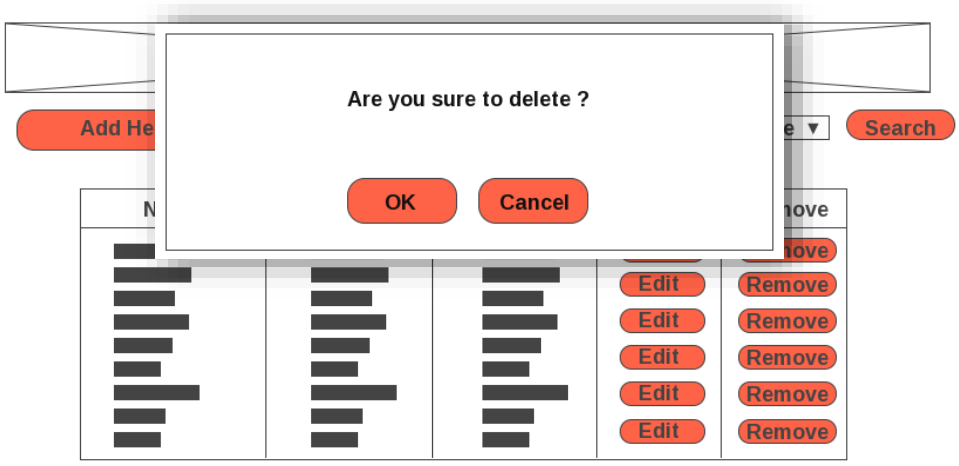

Figure 88 Remove Confirm Dialog (UI-13)

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

UI name: Successfully Remove Dialog (UI-14)

Include Requirement: UCs-03

Figure 89 Remove Confirm Dialog (UI-14)

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

UI name: Update Information Page (UI-15)

Include Requirement: UCs-01, UCs-02

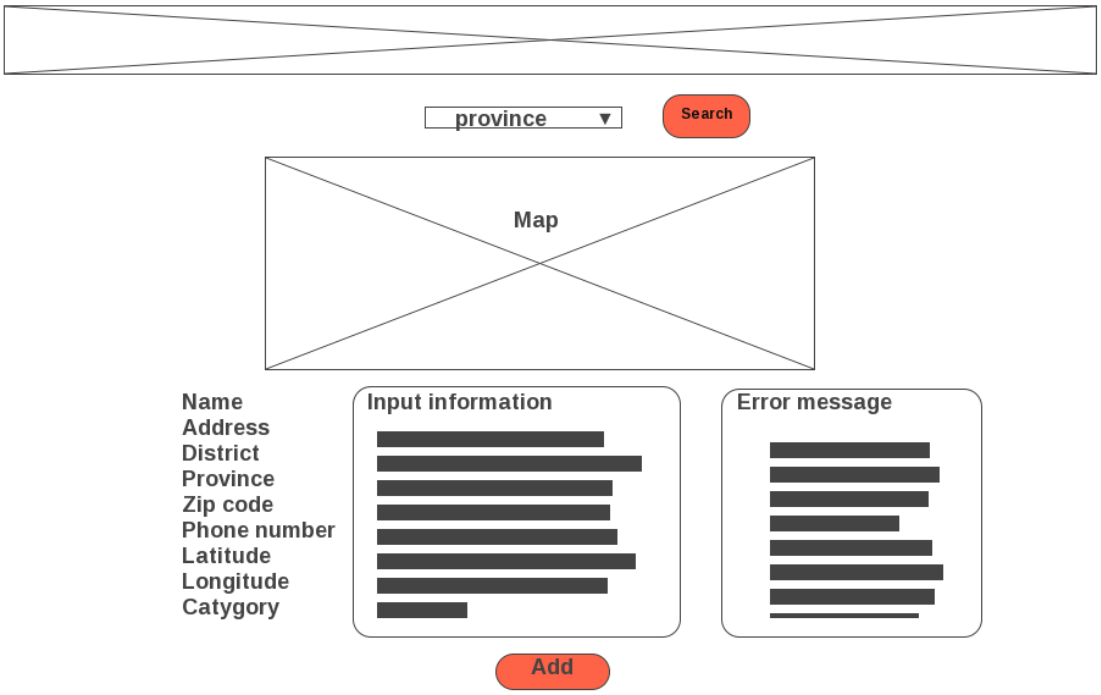

Figure 90 Edit Information Error Message (UI-15)

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-16)

Include Requirement: UCs-01, UCs-02

Figure 91 Remove Confirm Dialog (UI-16)

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

UI name: View Information Page (UI-17)

Include Requirement: UCs-04

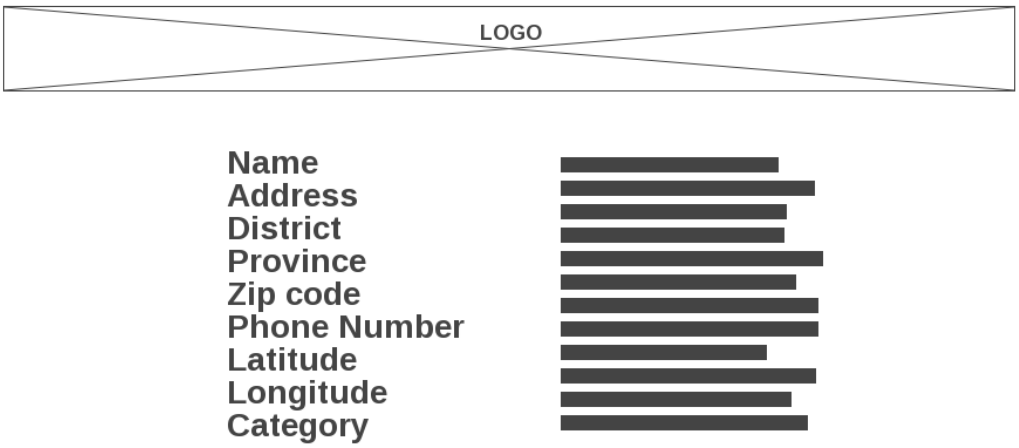

Figure 92 View Information Page (UI-17)

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: Login Page (UI-18)

Include Requirement: UCs-08

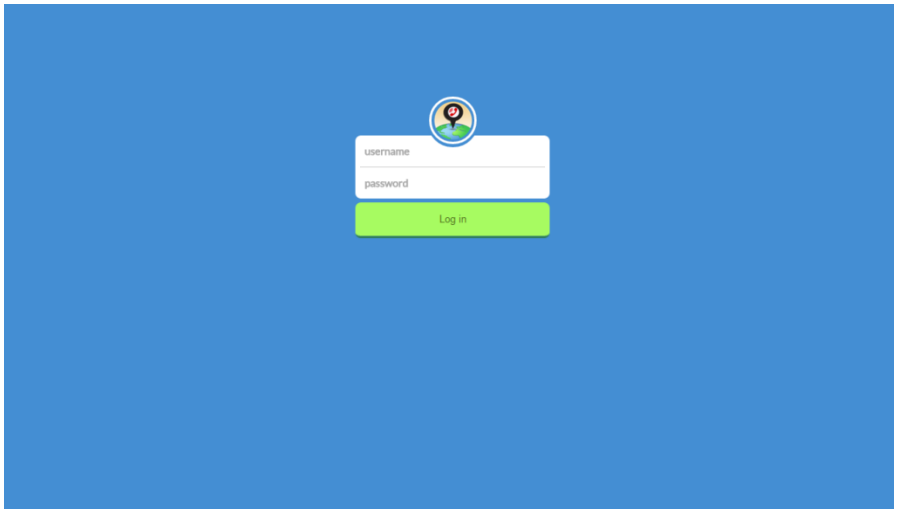

Figure 93 Login Page (UI-18)

The login page provides UI to input username and password.

UI name: Change Password Page (UI-19)

Include Requirement: UCs-10

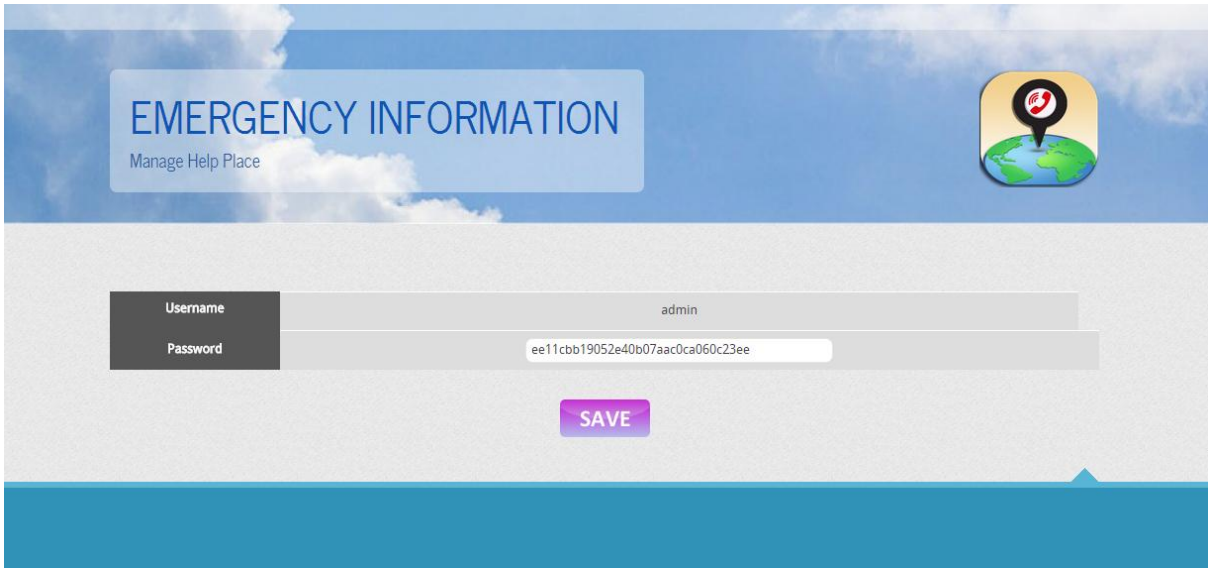

Figure 94 Login Page (UI-19)

The change password page provides UI to input password.