

KEMI-TORNIO UNIVERSITY OF APPLIED SCIENCES

Research and Implementation of Bus Locating System on Mobile  
Phone

Wang Yun

Thesis of Information Technology Degree Programme

KEMI 2011

## **PREFACE**

I really appreciate my instructor Mr. Thai Bui for his guidance and help step by step, and those who helped me out during my thesis project.

## ABSTRACT

Kemi-Tornio University of Applied Sciences, Technology	
Degree Programme	Information Technology
Name	Wang Yun
Title	Research and Implementation of Bus Locating System on Mobile Phone
Type of Study	Bachelor's Thesis
Date	18 December 2011
Pages	33 pages
Instructor	Thai Bui

This project aims to program a bus timetable system which is based on mobile device.

Some research reports of the recently mobile technologies are written such as LWUIT, GPS, JDBC and Servlet. Seven classes were programmed to compose the mobile application. One more class was made to finalize the server part function.

Some mobile technologies were used in the project, such as J2ME MIDP, TCP/IP network, Global Positioning System, JDBC and MySQL, to achieve the functionalities of the system.

The basic objectives of the project were accomplished successfully. A lot about the procedures of developing software and mobile technologies was learnt. Some improvement needs to be done in the future.

Keywords: J2ME, Servlet, GPS, JDBC, MySQL.

## TABLE OF CONTENTS

PREFACE .....	I 错误！未找到引用源。
ABSTRACT .....	II 错误！未找到引用源。错误！未找到引用源。
TABLE OF CONTENTS .....	III 错误！未找到引用源。
EXPLANATION OF CHARACTERS AND ABBREVIATIONS	IV 错误！未找到引用源。
1. INTRODUCTION .....	1
1.1. Motivation .....	1
1.2. Objectives .....	1
1.3. Expected results .....	2
2. TECHNOLOGY PREVIEW .....	3
2.1. Software involved .....	3
2.2. Technical method involved .....	3
2.2.1. J2ME .....	3
2.2.2. Light Weight User Interface Toolkit .....	5
2.2.3. Servlet .....	5
2.2.4. JDBC .....	7
2.2.5. Global Positioning System (GPS) .....	8
2.2.6. Google Maps .....	8
2.2.7. MidMaps library .....	9
3. DESIGN .....	11
3.1. User Requirements .....	11
3.2. Business goals .....	11
3.3. Use-case diagram .....	11
3.4. Sequence diagram .....	12
3.5. Database E-R model .....	16
3.6. User interface .....	18
4. IMPLEMENTATION .....	20
4.1. Client part .....	20
4.1.1. Class diagram and classes' implementation .....	20
4.1.2. Flow charts .....	21
4.1.3. Sending information to the server .....	23
4.1.4. Receiving information from the server .....	24
4.1.5. Splitting the inquiry result .....	25
4.1.6. Deal with invalid input data .....	26
4.2. Server part .....	29
4.2.1. Servlet structure and organization .....	29
4.2.2. Receiving and responding information from / to the client .....	31
4.2.3. Servlet and Database connection .....	31
5. EVALUATION .....	33
5.1. Achieved Results .....	33
5.2. Drawbacks .....	33
5.3. Future works .....	33
6. CONCLUSIONS .....	34
7. REFERENCES .....	35
8. Appendix .....	37

错误！未找到引用源。

## EXPLANATION OF CHARCTERS AND ABBREVIATIONS

J2ME	Java 2 platform, Micro Edition
CDC	Connected Device Configuration
CLDC	Connected Limited Device Configuration
MID	Mobile Information Device
MIDP	Mobile Information Device Profile
PDA	Personal Digital Assistant
PDAP	Personal Digital Assistant Profile
VM	Virtual Machines
JVM	Java Virtual Machines
UI	User Interface
GUI	Graphic User Interface
API	Application programming interface
GPS	Global Positioning System
JDBC	Java Database Connectivity

# 1. INTRODUCTION

This project aimed to develop a Bus Locating System (BLS), which is based on a mobile device. In this part, a general description will be given of the BLS project.

## 1.1. Motivation

Nowadays, with the development of network and technology of wireless communication, a mobile phone is no longer what it was. It may involve many kinds of functions like internet safari, personal information management, mobile business, entertainment and so on.

The mobile wireless communication devices are becoming more and more multifunctional and intelligentized. The cost of data transmission recently present an extent of debase because of the improvement of network, the requirement to all kinds of mobile technology becomes larger and larger.

In some cases of people's daily life, people may be planning travelling where they have not been before, so the people should first go to the bus stop and find a schedule, look into the every bus line and pick a certain stop (which they want to go) from the schedule of the bus. Sometimes the people even do not know where the bus stop is or the bus stop is too tousy to have a schedule. Maybe the people are not going right now but they need to know the schedule to make up the travelling plan. With the BLS system, they do not need to go to the bus stop and check.

Therefore, this mobile application is developed for those who want to go some places by bus to look up all possible methods and routes. Using BLS application could bring the users more convenient experience.

This application is free for users. Bus company could upload their bus line file (this file should following a certain standard). The customers should be able to purchase authority to download the file, so they can use the bus line data file on their own device. There are a number of such applications for different brands of smart phones, these applications provide a basic idea for developing this project.

## 1.2. Objectives

To study and build a mobile application, the user could benefit from searching useful information from the BLS system.

The development of a Client / Server system allows users to query bus information (e.g.

locate a bus stop, search for a bus schedule, find a particular bus, etc.). The client part is developed based on J2ME platform which means it can run on mobile devices with MIDP support. The server side is built based on server application technologies such as SQL Database.

There are several latest mobile technologies which applied in this project. For example, GPS is used for locating the user and related information of a bus station. Light Weight User Interface Toolkit (LWUIT) is used for building friendly GUI of client part.

### **1.3. Expected results**

The Client part: A J2ME application with all required requirements. The application can run on mobile devices supporting MIDP 2.0. GPS technology is used to locate the user's location and related information of a bus station. The UI of the client application is developed mainly based on LWUIT, in order to build an elegant and friendly user interface.

The Server part: storing all information of a bus system and allowing the client application to query the necessary data. A relational database (e.g. MySQL) is used to store all related information of the system. Some web technologies (e.g. SERVLET ) are used to develop the server applications.

The connection between the client application and the server part is based on TCP/IP network (e.g. Socket connection or HTTP connection). Data transmission is developed by using the J2ME library (e.g. DataInput and DataOutput).

The final release must be tested correctly on local network and the MIDP simulator.



## **2. TECHNOLOGY PREVIEW**

### **2.1. Software involved**

In this project, Netbeans IDE, Tomcat server, MySQL and MySQL Workbench were used to finish a large amount of work. The most important advantage of the software is that are open and free for use.

Netbeans IDE is an open source and a useful product for software development which supports several programming languages like PHP, JavaFX, C/C++, JavaScript, and frameworks, etc. It was used for UI design, coding of client and servlet.

Tomcat is also an open source product, and it is known as one of the most popular web servers. It provides a platform to run Java Servlet and the JavaServer Pages (JSP) on a web server.

MySQL is a free and a useful database. It is commonly used in co-operating with PHP scripts for building powerful and dynamic server-side applications. It was used for storing bus data in this project. MySQL workbench is an official released client to provide a visible and graphical method to operate MySQL server and database.

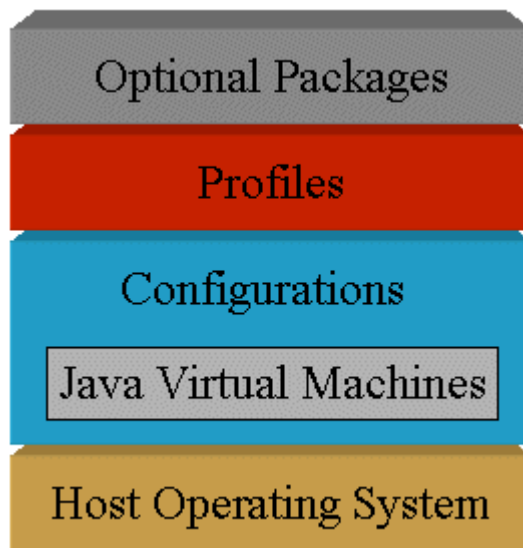
Besides those mentioned above, some other open software were also used for small part of work like: Photoshop for the UI design and Dia for the system analyses.

### **2.2. Technical method involved**

#### **2.2.1. J2ME**

J2ME (Java 2 Platform, Micro Edition) is released by SUN company /15/. It is used to provide an environment of running Java application for mobile phone, PDA (personal digital assistant), television top-set and other low-end and resource-limited embedded devices. It also provides a public, open and fixable environment for application developers to use the Java programming language and related tools.

J2ME platform is composed by a series of technical specifications, such as CDC (JSR-36), CLDC1.1 (JSR-139), MIDP2.0 (JSR-118), Mobile 3D Graphics API for J2ME (JSR-184), Mobile Media API (JSR-135), Wireless Messaging API (JSR-120) and so on /6/. These specifications are defined for the different types of embedded devices or functions. Many of the devices manufacture companies and software developping companies had also involved during the process of specification definition, such as Nokia, Motorola, Sun Microsystem, and Symbian.



**Fig . 1. J2ME architecture /17/**

J2ME is designed as a 3 layers architecture as the Fig . 1. It is constituted by configuration layer, profile layer and optional package layer.

**Configuration:** A configuration is the core in J2ME's architecture. In this layer, Java virtual machine features (JVM) and minimum class libraries have been defined. Due to the computational capabilities for the different types of embedded devices, the libraries and virtual machines are divided into two classes: one is the connected limited devices. This type refers to small memory, small bandwidth (128-256KB) and intermittent net connection device (e.g. Mobile phone, PDA). The VM and libraries for these limited devices are defined in CLDC (connected limited device configuration). The other class is the connected devices which refers to larger memory (2-4MB), wider bandwidth and durative net connection devices (e.g. Television topset, GPS navigator). The specific VM and libraries are defined in CDC (connected device configuration). /7/

**Profile:** A J2ME profile is another layer built up on top of a configuration. The main goal of a profile is to divide embedded devices into different classes as a device family or domain by defining a standard Java platform according to the applied capability of the device, and provide the device family with corresponding Java libraries support. For example, the specifications of a java library for mobile information device are defined in mobile information device profile (MIDP), and the specifications of a library for personal digital assistant are defined in personal digital assistant profile (PDAP). /7/

**Optional package:** A J2ME optional package is a layer placed on top of a profile and it extends the functionalities of a profile. It provides support for the corresponding Java libraries of the functionalities which is independent of a device family. It is used for defining APIs which can be programmed flexibly over a profile. /7/ For example, JSR 120 which refers to Wireless Messaging API, it lets the J2ME application send and receive messages. JSR 66 which refers to RMI API provides the possibility for one Java object to invoke methods of Java objects which runs in a different VM/14/.

### 2.2.2. Light Weight User Interface Toolkit

A brief introduction of LWUIT has been given in this part. This library is supposed to be used for user interface developing in the project. The schedule is under the influence of some factors, such as the limitation of time and the unknown errors from the LWUIT UI designer. After the discussion with the instructor, this function is allowed to be achieved in the future.

#### 1) LWUIT introduction:

A group of developers at java.net created the LIGHT WEIGHT USER INTERFACE TOOLKIT project to make it easier for creating more friendly and consistent mobile Java applications.

LWUIT is a library for developing user interface. It provides the developers with title bars, forms and status bars and also for other visual components and other UI features such as theming, transitions, animation and many useful Swing-like features. /3/

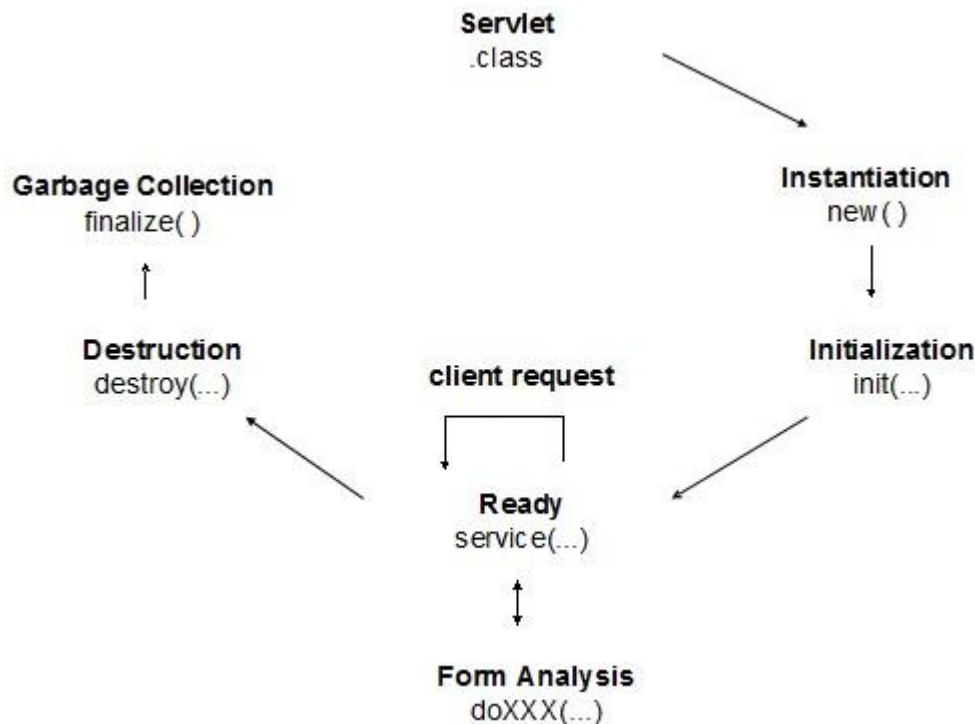
#### 2) Advantage of LWUIT:

LWUIT has some advantages comparing with other user interface programming libraries (SVG, LCDUI), for instance, it can be easily bundled with the application for its small footprint library. It has some capabilities for modern interface such as animations, transitions, layouts, UI controls, 3D capabilities, etc. It provides the possibility to modify component appearance, including theming and pluggable look-and-feel. The controls such as forms and lists can be displayed consistently in most devices which support Java ME. It requires only CLDC 1.1 and MIDP 2.0. The HTML component which is provided by LWUIT can render the mobile web content seamlessly. /3/

### 2.2.3. Servlet

A servlet is a class of Java programming language which runs on the server side. It is used to extend the capabilities of server and host application accessed which uses "request/response" programming method via http protocol. It can be thought of a Java application which has the same functionality with the JSP or PHP web pages for processing data on server side.

#### 1) How does servlet work?



**Fig . 2. Life cycle of Servlet /6/**

Here is a life cycle of servlet (as Fig . 2. Life cycle of Servlet). It begins when the Web container called the init() method, and it gets destroyed when the container process ended.

When client sends a request to server, the server transmits the request to web container (i.e. Tomcat). Tomcat finds the specific servlet which is comprised in the url from the client. The container creates two objects, HttpServletRequest and HttpServletResponse, then passes them to servlet. The container checks if the servlet instance is available, if the instance already exists then calls the services method according to the request method type (get, post or head). But if the servlet instance had not be created, the container loads the servlet class and creates an instance of this servlet, then calls init() method for initialization, once these have been done, does the same procedures as the case of instance existed. After that, service() method sends the response back to the client via HTTP protocol as well. If the servlet needs to be removed (for example, the servlet is being shut down), then the container calls the destroy() method to finalize the servlet./4/

## 2) Setting up environment

There are some programming tools including JDK, Tomcat and Netbeans. JDK for runtime environment of Java application and developer, Netbeans for implementation of source code and Tomcat is the server container which allows a servlet runs in it.

- Download "jdk-7-windows-i586", "netbeans-7.0.1-ml-windows" "apache-tomcat-7.0.22.jar".
- Click "jdk-7-windows-i586" to install Java SDK and copy the installation path. Right click on MY COMPUTER, click PROPERTIES, go to advanced tap then click

ENVIRONMENT VARIABLES. Add the installation path as JAVA\_HOME, ".;JAVA\_HOME\lib" as CLASSPATH and "; JAVA\_HOME\bin" to PATH variable.

- Unzipped "apache-tomcat-7.0.22. jar" does the same operations as STEP 2. Add a new system variable "CATALINA\_HOME" value of "c:\tomcat", a new variable "CATALINA\_BASE" value of "c:\tomcat" and a new variable "TOMCAT\_HOME" value of "C:\Tomcat" into CLASSPATH. One IMPROTANT thing is add "servlet-api.jar" which in lib folder of tomcat to CLASSPATH also. So the developers can locate the file "startup.bat" under the directory (the path: "/tomcat/bin") and execute it by double click. Open a browser type <http://localhost:8080> into address field to test if the tomcat has been configured correctly.

## 2.2.4. JDBC

### 1) Instruction of JDBC

The JDBC is a Java API with interfaces and classes which defines for Java application to create connection with relational databases.

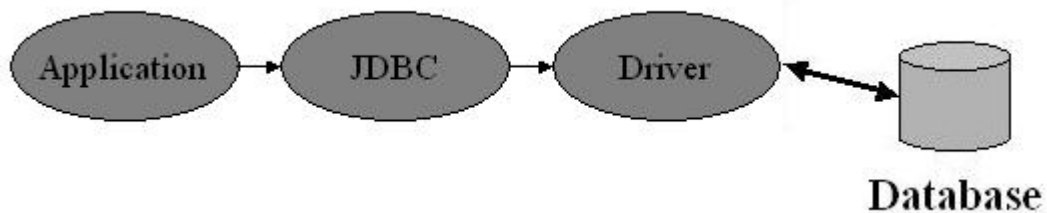


Fig . 3. JDBC architecture /8/

The figure above, Fig .3, shows the architecture of JDBC, when the JDBC called by a Java application, it first loads the driver which is defined by user, then make connection to communicate with database.

### 2) Preparation for JDBC

The JDBC connector is downloaded from the web site /13/, MySQL was chosen as database because firstly it is open and free to use, secondly it is smaller than other databases, such as Oracle and a very large size of database is not suitable in the project.

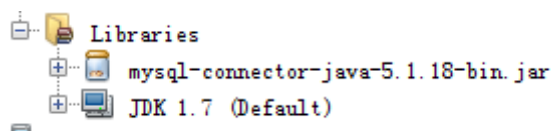


Fig . 4. Location of JDBC connector

"mysql-connector-java-5.1.18-bin.jar" is downloaded from the official web site of SUN company and import it into the project library as the Fig .4.

Properties of the computer is selected by right click on the desktop, the path of jar file is added to CLASSPATH variable under SYSTEM VARIABLE window, the value looks like: "D:\ProgramFiles\MySQL\mysql-connector-java-5.1.18\mysql-connector-java-5.1.18-bin.jar;".

### **2.2.5. Global Positioning System (GPS)**

#### 1) GPS introduction

The Global Positioning System (GPS) is a space-based global navigation satellite system. In the 1980s, the government provided the system at the service of civilian use. The GPS works very well without any limitation from time or weather, and what is more important it is totally free to use GPS which means no subscription fees or setup charges/12/.

If the GPS is tested on a mobile device which supports GPS module then it is very simple, the user can just run the GPS application. However, if the user tests it on a computer, there are some differences from what be done on the mobile device. The procedures of locating a place through the particular latitude and longitude data are indicated in appendix (GPS example).

#### 2) Setting up environment

In this part, some steps are given to explain how to set up the environment. It is used for developing a J2ME application which supports GPS technology.

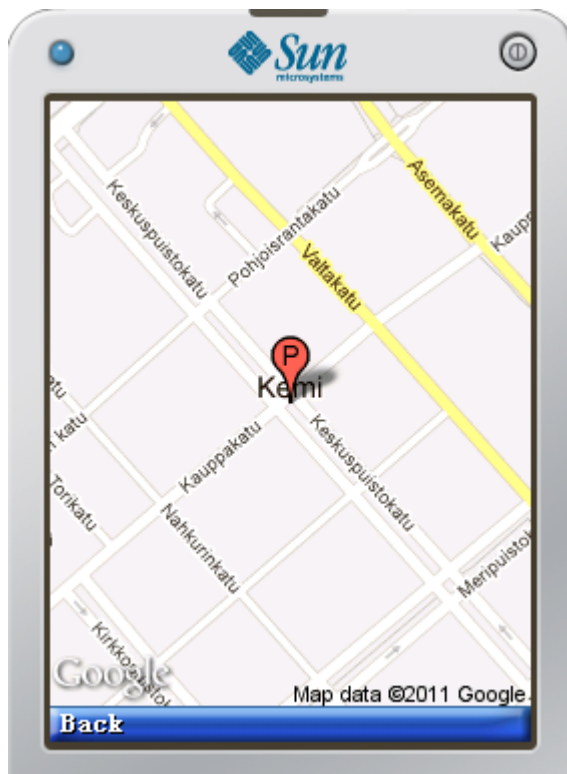
- Click "jdk-7-windows-i586" to install Java SDK and copy the installation path.
- Right click on MY COMPUTER, click PROPERTIES, go to advanced tap then click ENVIRONMENT VARIABLES. Add the installation path as JAVA\_HOME, ".;JAVA\_HOME\lib" as CLASSPATH and "; JAVA\_HOME\bin" to PATH variable.
- Click "netbeans-7.0.1-ml-windows" to install Netbeans, no special configuration for IDE.

### **2.2.6. Google Maps**

The Google Maps APIs is supposed to be used for locating in the project in the beginning. During the process of Map API pre-study, another library called MidMaps is found /1/ which is also based on Google Maps. It is very easy to integrate Google Map in J2ME application. Therefore, the decision of using this library for this project was made. However, MidMaps also has some limitations, which will be explained later, after Google Maps sector.

### 1) Google static maps introduction

Google Static Maps API released by Google company /2/ used to display a Google static map on the emulator or on a real device that has a GPS module, according to the coordinates which is got from querying the mobile device GPS module. These services are open and free, which need a mobile device that supports the Location API for J2ME under JSR-179. Fig . 5 shows an example of Google map which captured from the application. The location is Kemi with latitude of 65.736277 and longitude of 24.564398.



**Fig . 5. Example of Google static map**

### 2) Setting up environment

The environment required for Google Maps is exactly the same as used in GPS case. Download "jdk-7-windows-i586" and "netbeans-7.0.1-ml-windows" and install them. Add the class path of jdk to system environment variables after installed.

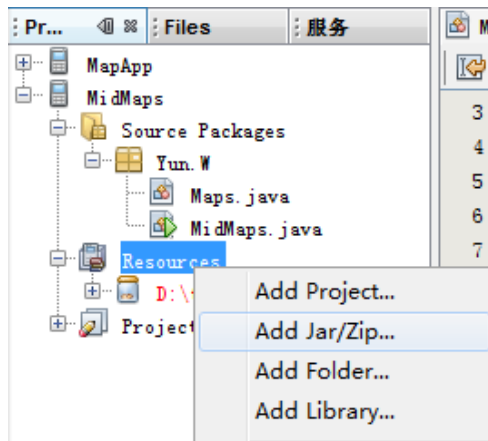
## 2.2.7. MidMaps library

### 1) MidMaps introduction

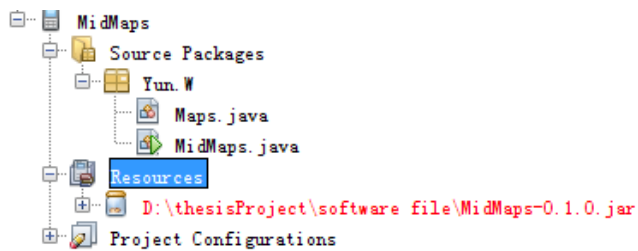
MidMaps /1/ is a tiny Google Maps library for Java ME applications which can integrate Google Map into J2ME application with in a few steps. It is totally open and free for use. To work with MidMaps the users do not need to care about threads and connections.

## 2) Setting up environment

The JAR file of MidMaps-0.1.0.jar is downloaded from <http://www.jappit.com/blog/midmaps-google-maps-java-me-library/> and “Add JAR/ZIP...” is clicked in project view window. Fig . 6 shows below. MidMaps. Jar is included in the project. The current version is compatible with devices supporting MIDP 2.0 and CLDC 1.1. The jar file is in red as Fig . 7 shows below, but it does not matter, it can work well.



**Fig . 6. add MidMaps library\_1**



**Fig . 7. add MidMaps library\_2**



### 3. SYSTEM ANALYSIS & DESIGN

#### 3.1. User Requirements

The purpose of this part is to define the requirements for system BLS (Bus Locating System). The topic is found in a textbook. After discussed with my instructor, the finally functionalities are decided as the table below.

Group of user:	Functionality of system:
Users or customers	Locating user/station on a static map
Users or customers	Input a bus or station name to be queried
Users or customers	Retrieve data from database

#### 3.2. Business goals

The BLS system is open and free for use. Bus companies could upload their bus line file which contains the details of bus information and schedule (this file should follows certain standard). Customers should be able to purchase authority to download the file, save the data file and use on their own device. This feature is not implemented in this version, it can be considered in future work.

#### 3.3. Use-case diagram

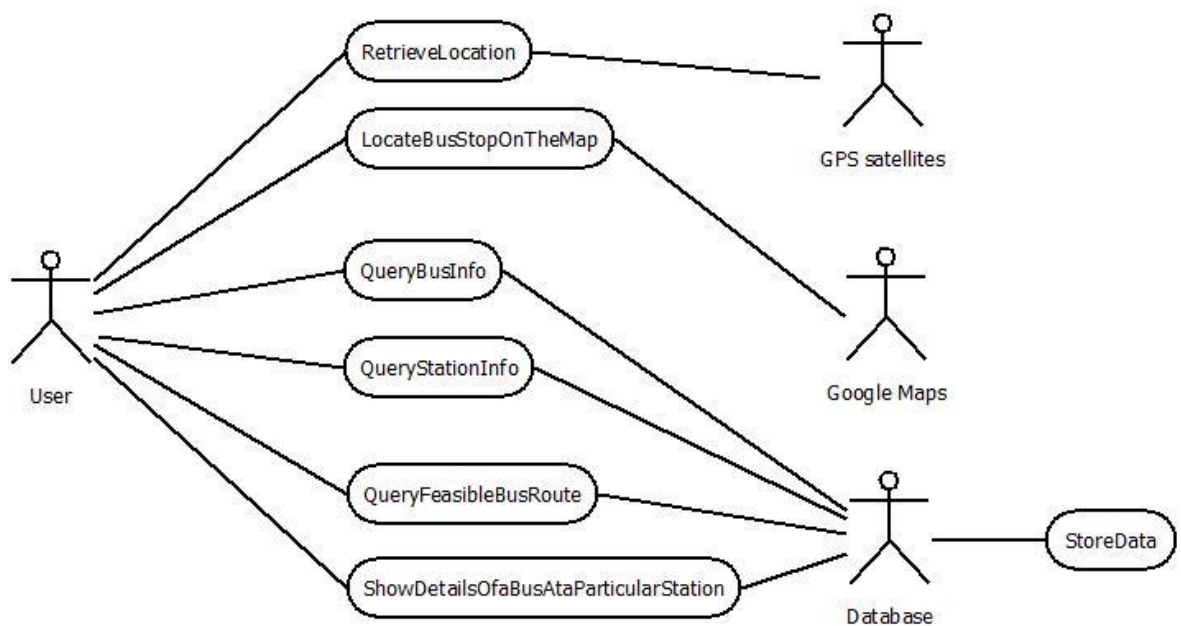


Fig . 8. Use case of BLS system.

This Fig . 8 is about the use case definition of the system. There are four actors in the system the User, GPS satellites, Google Maps and Database. The use case of store data is quite simple, it is used for storing data. Several use cases for user. When the user runs the application, there are about four functions. The first use case `RetriveLocation` stands for the first function. When the function is called by the user, it sends the request to another actor GPS satellite to get the user's location back and then mark the location on a static map. `QueryBusInfo` use case and `ShowDetailsOfaBusAtaParticularStation` use case indicate the second function, which requires the user to input a bus name and then return back the data related the bus. The use cases of `QueryStationInfo`, `LocateBusStopOnTheMap` and `ShowDetailsOfaBusAtaParticularStation` compose the third function, with which the user is able to retrieve station data and station location when give the station name to application. `QueryFeasibleBusRoute` describes the last function, which provides the user the possible bus names (which contain the departure station and destination station in their route). The departure and destination stations are input by user. Then the user can also get the timetables of the possible buses.

### 3.4. Sequence diagram

In this part, some diagram for each use case will be given.

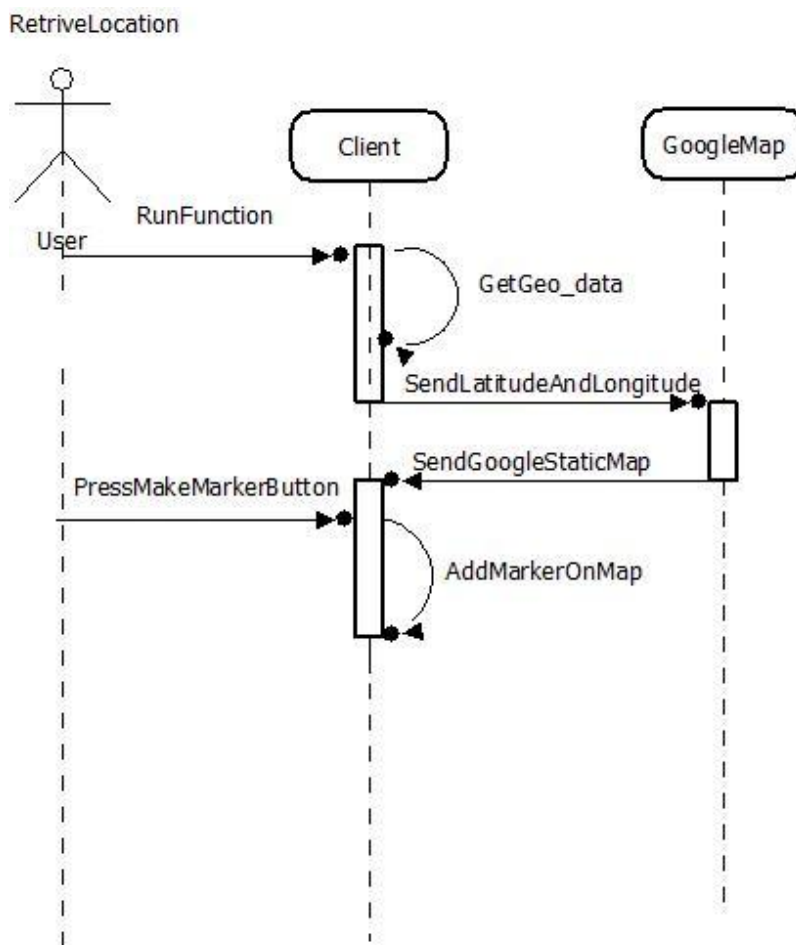
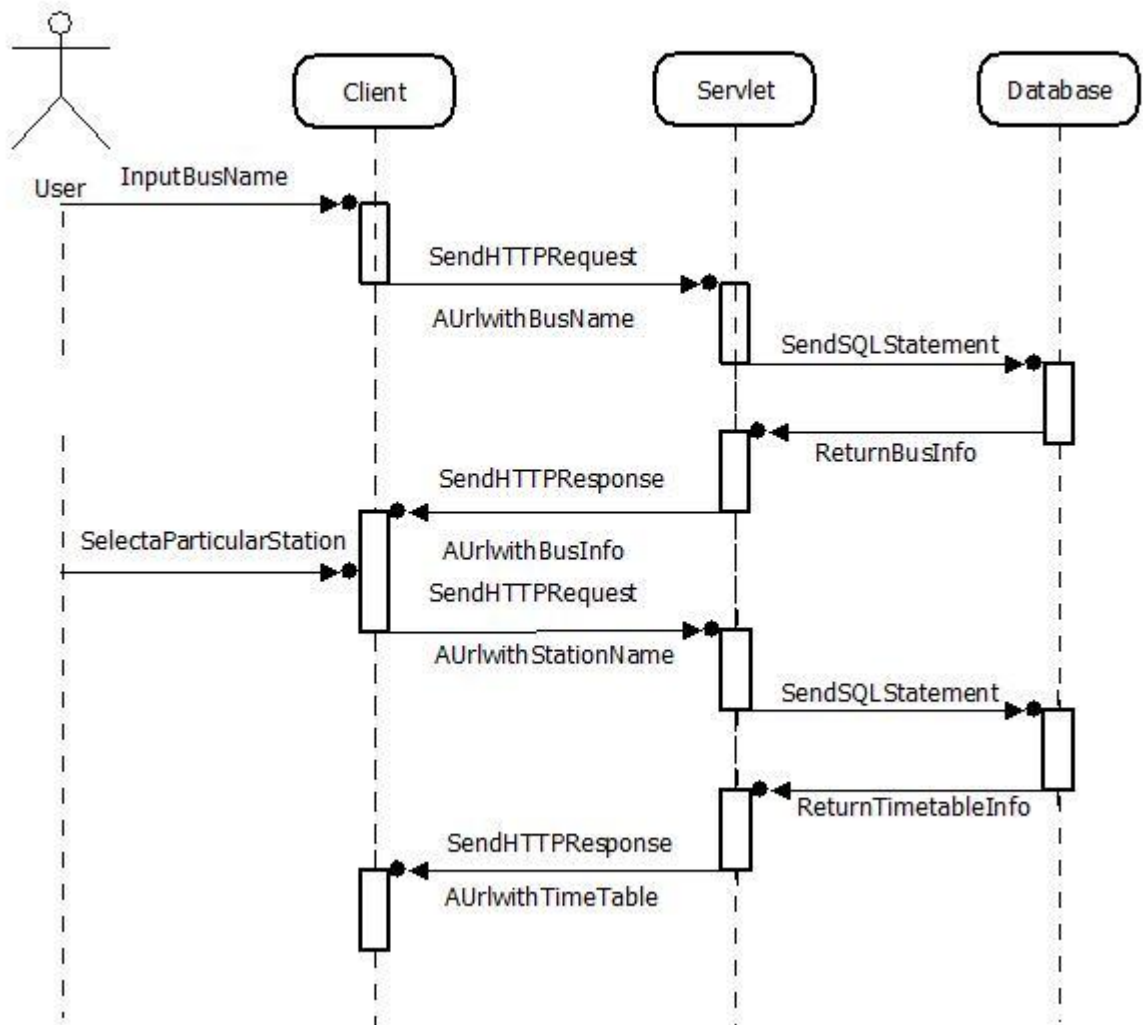


Fig . 9. RetriveLocation Sequence Diagram

Fig . 9 expresses the process of RetriveLocation use case. When the function is called by the user, application gets the user's latitude and longitude by using GPS (Global Position System) technology. The client sends a url (which composed by GoogleMap address and user's location and other basic configuration of map) to GoogleMap to retrieve a static map to display user's location.

#### QueryBusInfo



**Fig . 10. QueryBusInfo Sequence Diagram**

In this use case Fig . 10, there are four objects: user, client, servlet and database. At the beginning, the user is asked to input a bus name, the application sends a http request to servlet with the bus name. By processing the request from client, the servlet gets the parameter which is used to query database, sends SQL statement to database by using JDBC technology. After the SQL statement executed, the result returns to servlet, servlet sends a response to client with the data of the result from the database.

QueryStationInfo

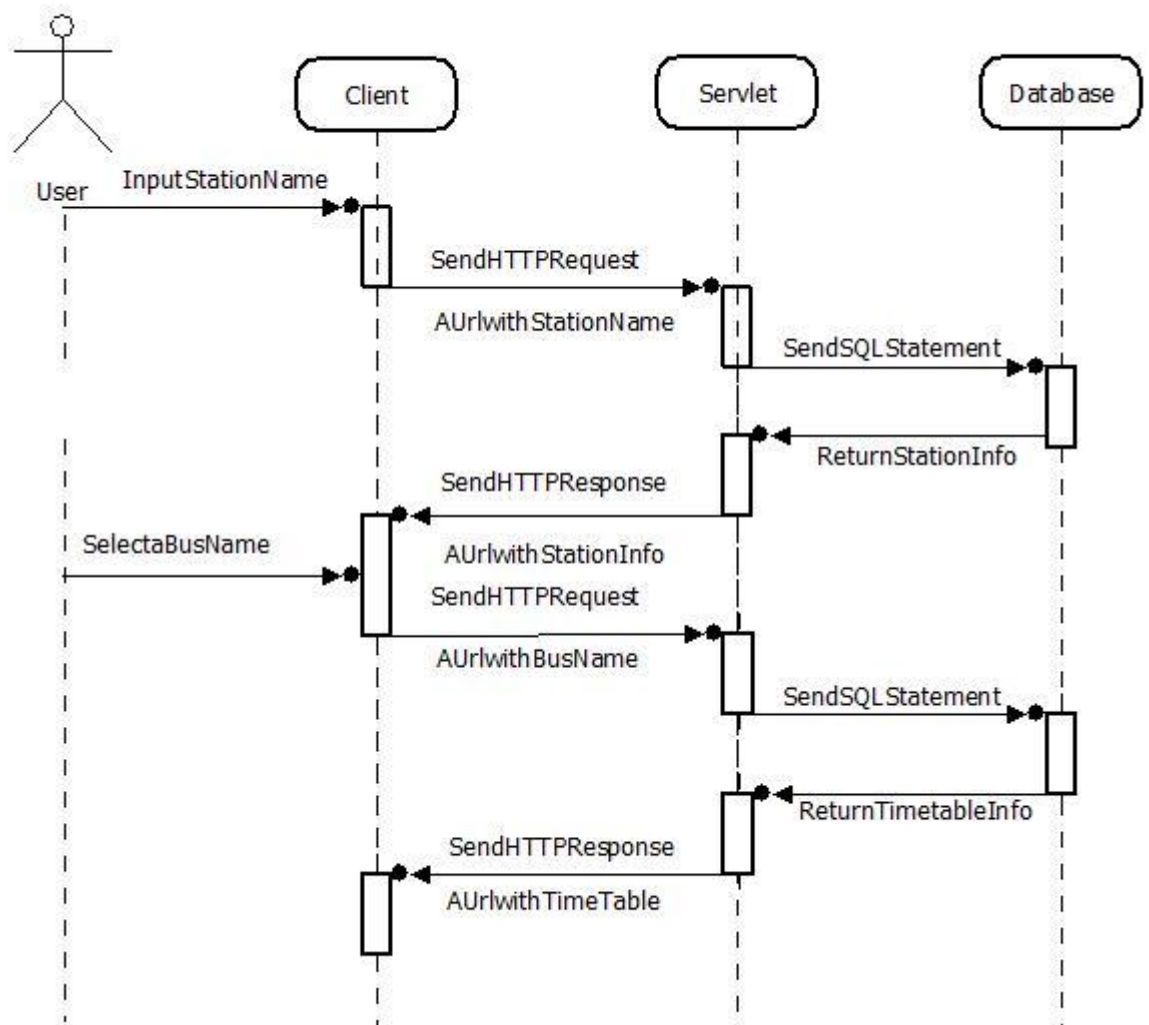
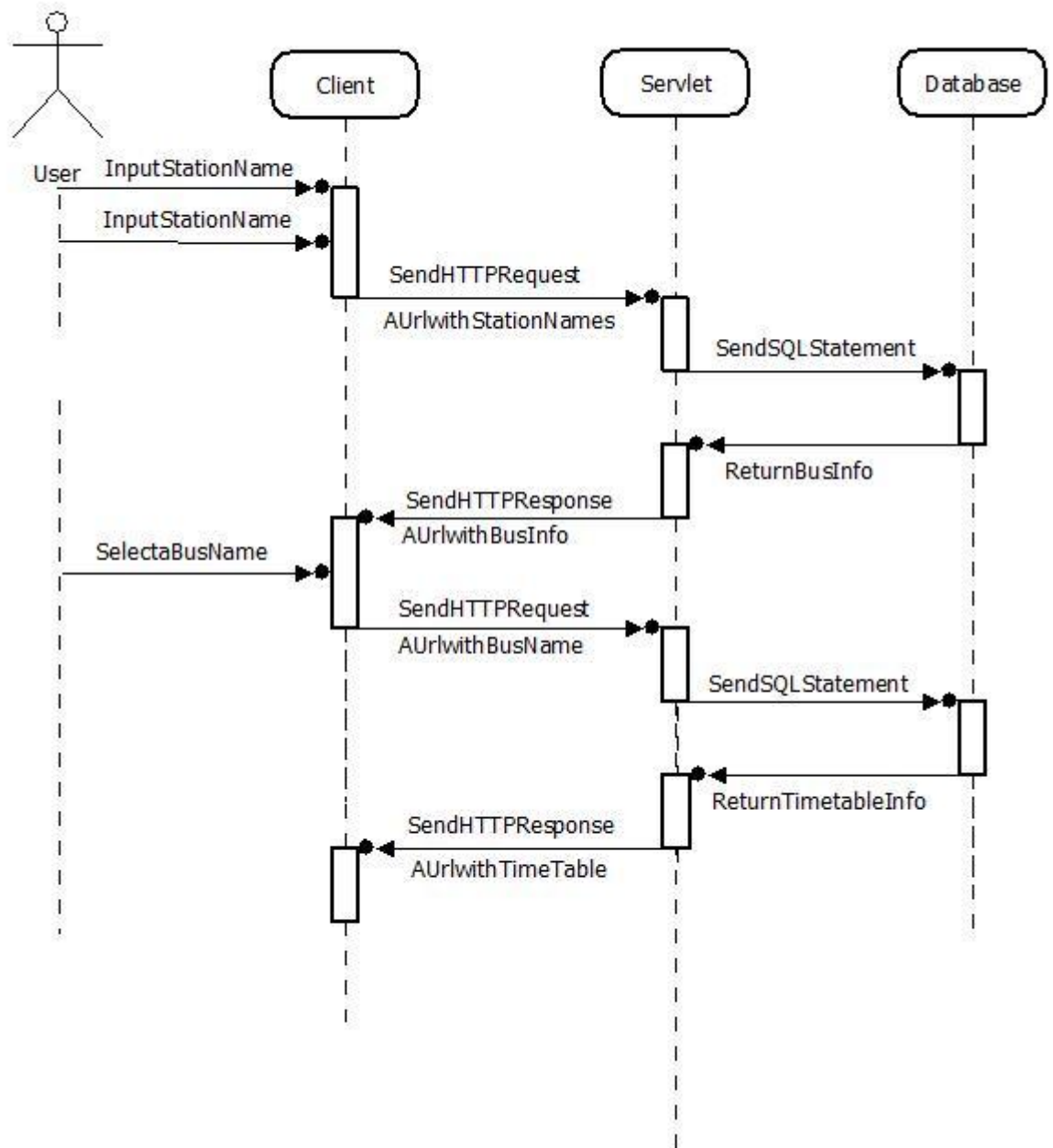
**Fig . 11. QueryStationInfo Sequence Diagram**

Fig . 11 above describes the process of use case of QueryStationInfo, and it is quite similar to QueryBusInfo sequence diagram. The only difference is that it requires a station name instead of a bus name at the beginning.

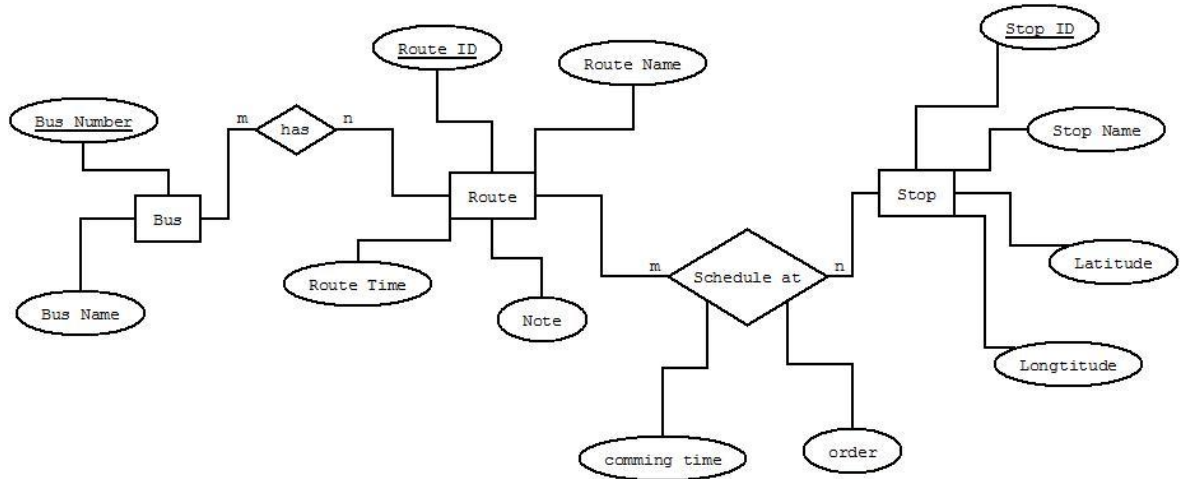
## QueryStationInfo



**Fig . 12. QueryFeasibleBusRoute Sequence Diagram**

For this use case of QueryFeasibleBusRoute as Fig . 12, the function works same with QueryBusInfo and QueryStationInfo use case. The difference is that user has to provide the origin station and destination station's name as the query condition.

### 3.5. Database E-R model



**Fig . 13. E-R model**

The database that used in the project is called busQueryDB\_v1. There are three objects in the system which are bus, route, and stop, so first of all the system has three tables in the database, there are two more tables “has” and “scheduleAt” which generated from the relationships of "bus has route" and "route schedule at stop".

The first one is bus table. It contains two attributes: Bus Number (primary key) and Bus Name.

**Bus Number:** this is the primary key for bus table, and it is NOT NULL, AUTO INCREASE and INTEGER type.

**Bus Name:** the identify name for each bus, there are some examples from HSL web site ( a bus company located in Helsinki ), like 11, 14, 15B.

The second is route table, the four attributes for this table are: Route ID (primary key), Route Name, Route Time, and Note. Route means a serial of stations with a certain order which locate on the way of a bus. For example, #1 there are five stations on its way and the order is: A-B-C-D-E. So that is called a route for bus #1. After the #1 bus arrived at E station then it will get back in an order like: E-D-C-B-A, and this is another route for this bus #1.

**Route ID:** the primary key for route table, used for identifying each route. Set as NOT NULL, AUTO INCREASE and INTEGER type.

**Route Name:** the name for each route, usually storing the first station name and the last station name in the route, here is an example: Eira-Viiskulma.

**Route Time:** it could give you an idea about how long will the bus take to complete a full route.

**Note:** This attribute is defined for storing notification of change

One bus can at least have one route or more and one route can be shared with one or more buses. Another table derives from this relationship, which is called “has” table in the

project. There are no attributes for “has” table. It only has two foreign keys (bus\_busId from bus table and route\_routeId from route table). It is used to associate bus table with route table.

The third is the stop table. The attributes are stopId, stopName, latitude and longitude.

Stop ID: the same as busId and routeId, it is the primary key of stop table, NOT NULL and AUTO INCREASE and INTEGER type.

Stop Name: the same as busName. Examples for Stop Name: Eira, Viiskulma.

Latitude and Longitude is used for store the geographical data for Google Map function.

The last table is named “scheduleAt”, it describes the relationship between route and stop table. It has two attributes: “commingTime” and “order”. One route could have around 10 stations or even more and one station also could belongs to more routes. So their relation is m:n. There are also two foreign keys in this table, route\_routeId which is inherited from route, stopId which is inherited from stop.

CommingTime: it indicates the estimated time for a bus arriving at a particular station in a route.

Order: order is used to specify the position of the station in a particular route.

The E-R model is converted to relational tables. Here are some tables for the database as below:

```
bus ( busId(INT); busName(VARCHAR) ),
```

```
has      ( routeId(INT); busId(INT) ),
```

```
route ( routeId(INT); routeName(VARCHAR); routeTime(VARCHAR);
note(VARCHAR) ),
```

```
scheduleAt ( routeId(INT);           stopId(INT); commTime(VARCHAR);
order(INT) ),
```

```
Stop ( stopId(INT); stopName(VARCHAR); latitude(DOUBLE);
longitude(DOUBLE) ).
```

### 3.6. User interface

In this part, some screen shots are presented which are made by photoshop for indicating the activity flow. These are not captured from the real product so there may be some differences with the final ones.



Fig . 14. GPS UI screen

Fig . 14, there are four functions in the application. When the GPS function is executed, it displays a static map. The red dot in the second screen indicates the user's location in the map and it is always displayed at the center of the map.

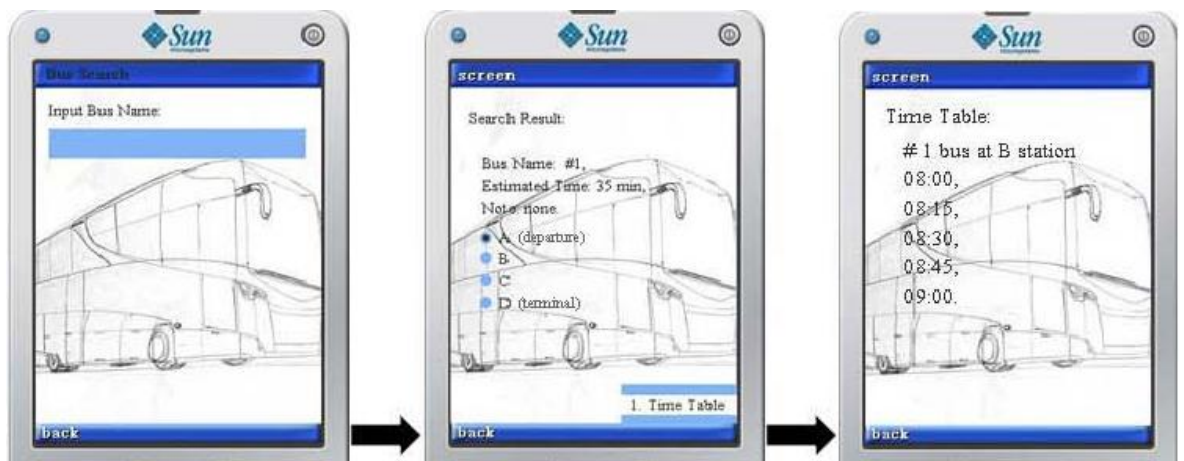
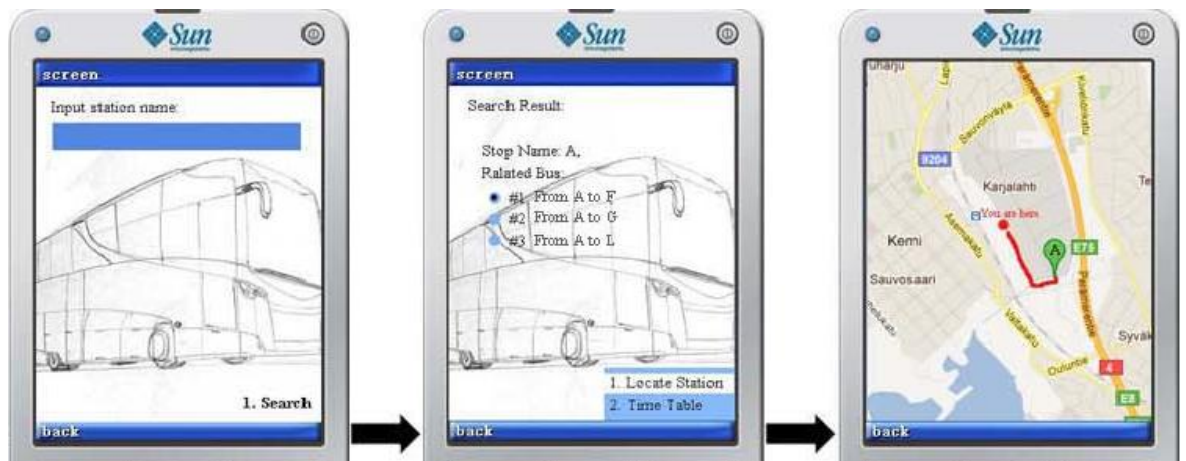


Fig . 15. Query bus UI screen

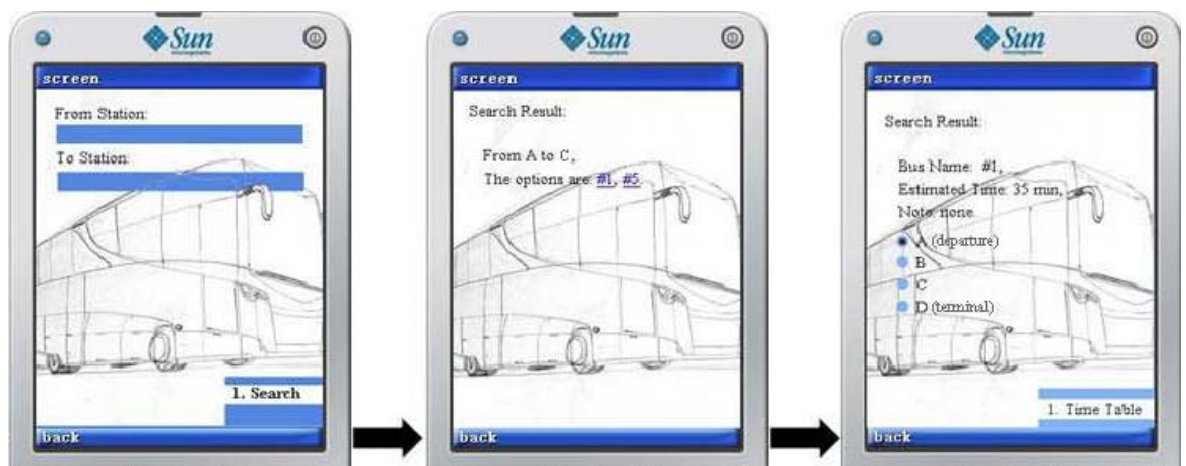
Fig . 15 shows bus query function. The user is required to give the bus name. The result screen is shown as the screen shot in the mid of the figure, it contains bus name, estimated time, note, and all the bus stations in its route. The user can go ahead to select one of the stations and then he can get the timetable of this bus to the station which is selected.





**Fig . 16. Search station UI screen**

In Fig . 16, the user is able to input a station name, and the same operations like that in bus query function, but it returns the station data such as station name and a bus name list which passes by this station on its way. If the user calls locate station function, the screen turns to a canvas to show a static map with a marker of the station which is selected and shows the path from the user to the station. The user can also choose a bus name from the bus name list and then gets the information about the timetable.



**Fig . 17. Search route UI screen**

Fig . 17 indicates the last function, if the user just knows the origination and the destination station name he/she is going to, the user can input the names. The application returns a list if any bus passes these two stations in its route. If so the user can choose one bus to get timetable of the bus schedule at these stations.

## 4. IMPLEMENTATION

### 4.1. Client part

#### 4.1.1. Class diagram and classes' implementation

The project is implemented within seven classes for the client part and one class for server part, according to the use cases diagram and function of the system.

##### BLSMid:

This class extends from a Midlet. It is the main class for the system. Not much function should be implemented in this class, only the function for getting the geographical location, then pass the parameters to Map class. This class just needs to define some items and commands here with which the users can get start for all other functions.

##### Map:

All functions about Google map are programmed in this class, in another words, all functions for the RetriveLocation use case are implemented in this class. Functions like displaying a map, making map markers and showing the paths between two markers.

##### Bus:

The function of this class is to build the interface for user to input the parameter which is used for searching bus information in the database and then parsing the data when received search result from search class.

##### Station:

The function of this class is almost the same as Bus class. It is about building interface of station related function, like: QueryStationInfo and ShowDetailsOfaBusAtaParticularStation use cases. Beside this, it has another function for locating the certain station on a map.

##### Route:

This class is to be created as the user interface of a flow for user to find out which bus he/she could take from the origination to his destination. It only creates user interface which contains items and commands used in the function, and some simple code for calling corresponding function when related commands are active.

##### Search:

This class is used for communicating between client and server. It extends from thread, no user interface for this class. It is used for sending request to server and also receiving response.

##### Time Table:

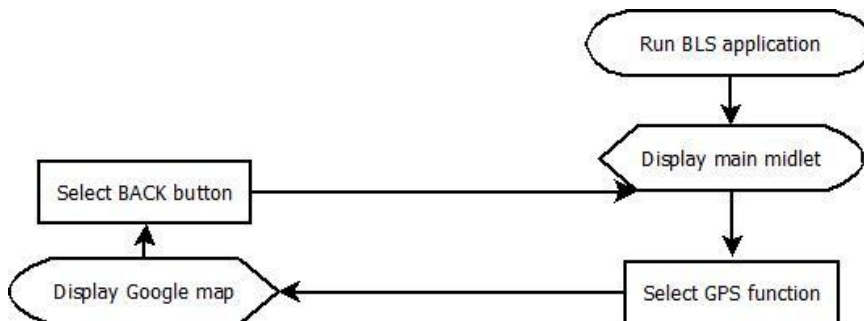
Because most functions (bus query, station query and route query) require timetable query function as their last step. So one more class for querying timetable is needed and showing the data in a form.

The last class left belongs to server side. It has two main functions: one is the communication with client, the other is accessing the database and fetching useful data.

#### 4.1.2. Flow charts

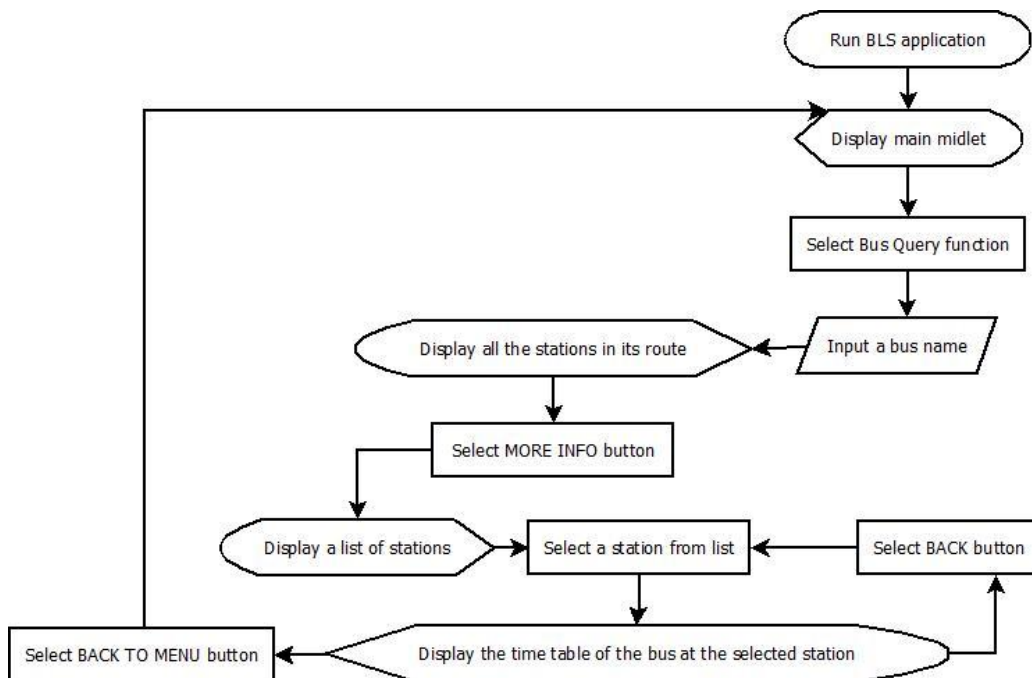
The whole figure is too big to place in the document so it is cut into four small parts according to the four main functions.

First is the flow chart for GPS function as Fig . 18:



**Fig . 18. GPS flow chart**

Next is bus query function as Fig . 19:



**Fig . 19. Query bus flow chart**

The third chart is station query function as Fig . 20:

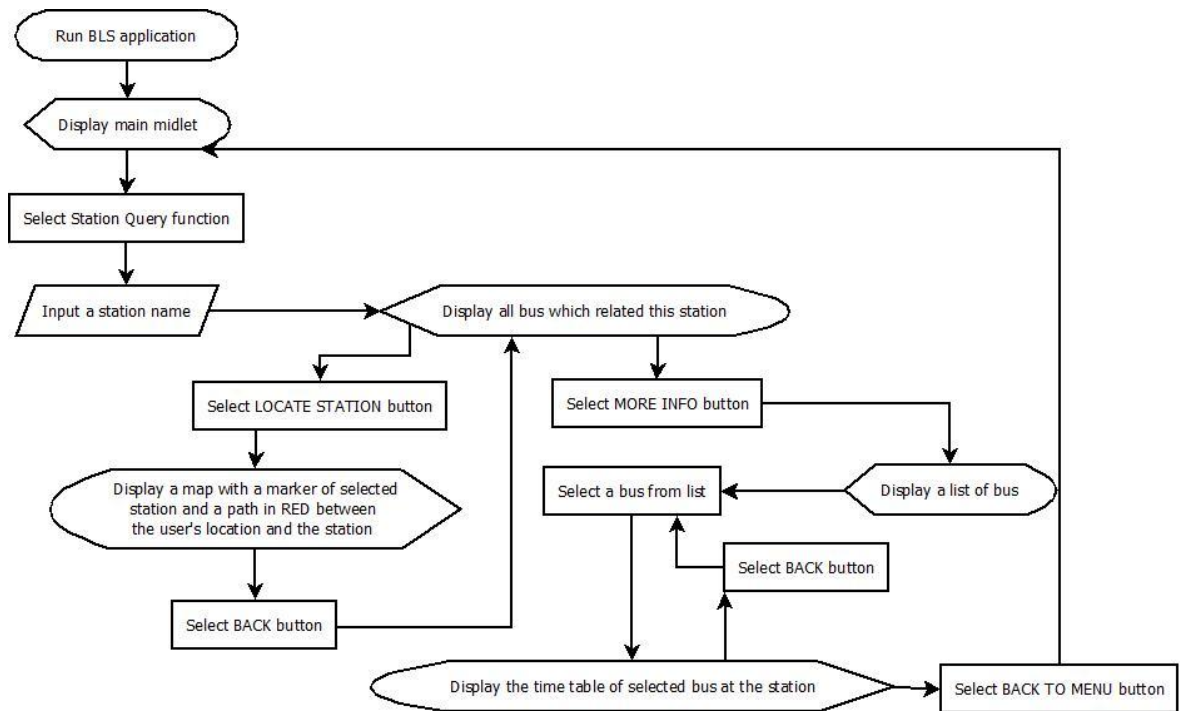


Fig . 20. Query station flow chart

The last chart is route query function as Fig . 21:

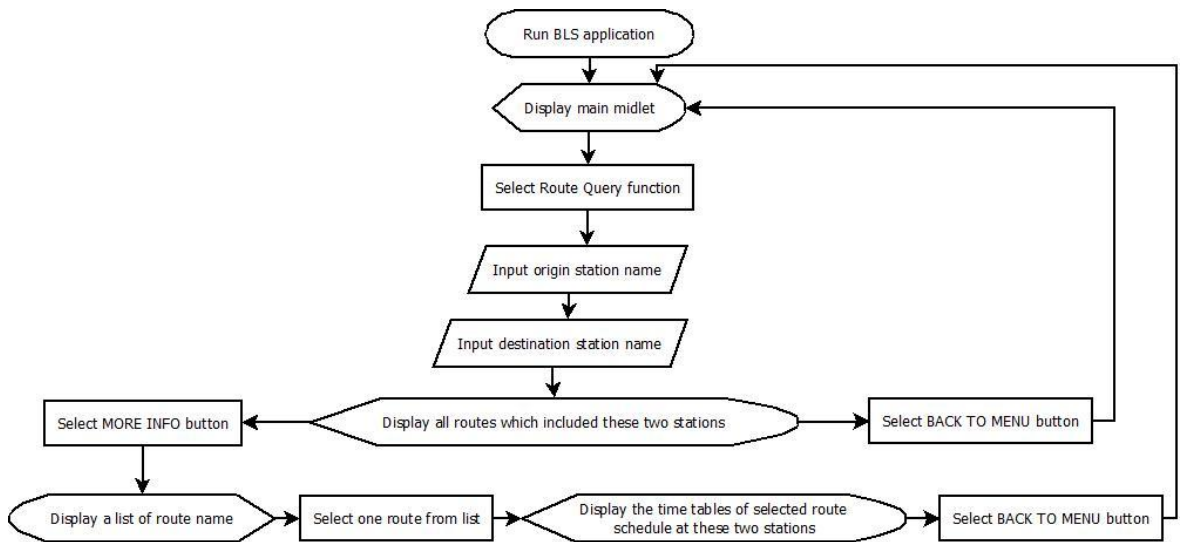


Fig . 21. Query route flow chart

### 4.1.3. Sending information to the server

There is a specific class "search" which used for sending and receiving information to/from server. This class extends from thread which means the application do not wait the information from the server all the time (in case of the server do not send response back for some reason). It can go on with other operations, this is more logical and practical.

In this part, the function of sending information is talked about first. When the application needs to query the database, it calls the search class with passing three parameters. Here is an example: In bus query function, when user inputs a bus name and presses "Bus Query" button, the variable of "type" is set to "b". It is used for letting the servlet to know which query function needs to be executed, as the code:

```
if(c == cmdSbt_B){
    bls.type = "b";
}
```

Then application creates an object (srhBus) of the search class and gives three parameters to the object. The first one is query type, which is mentioned above, the second one is obtained from the value of text field item which is input by user. For example, in the bus query function the second parameter refers to the bus name. These two parameters are enough for bus query function so the last parameter is set as null. But if the route query function is executed, the user needs to provide two station names as origination and destination so that all parameters are used: first for query type, second for origination station name and third for destination station name. An example of code is given below:

```
Search srhBus = new search(bls.type, tfBus.getString(), "");
srhBus.start();
```

When the object is created, it calls start() function, the parameters is given to local variables in class constructor and then search class creates a new thread and executes run() method which contains the code of sending information to server.

The process of the communication between the client and the server are divided into four parts.

- Define a url (which is the address of the servlet) for accessing servlet:

```
url = "http://localhost:8080/bls";
```

- Create an object of HttpURLConnection and call open() method to access to servlet via http protocol:

```
HttpConnection hc = null;
try{
    hc = (HttpConnection)Connector.open(url);
}catch(Exception ex){}
```

- Set request method as POST and get value of condition. The condition is a string variable which is passed to the servlet by POST method. The condition variable should follow the format as below, "condition=" can be a clue which specifies the value of the "condition" variable when servlet reads the request data.

```
hc.setRequestMethod(HttpConnection.POST);
hc.setRequestProperty("Content-Type","application/x-www-form-urlencoded");
condition = "condition = " + tfCon.getString();
```

- Create an object for OutputStream and call its write() method to send data to servlet, close output stream object when sending finished. The source code example is:

```
OutputStream os = null;
os = hc.openOutputStream();
os.write(condition.getBytes());
os.flush();
os.close();
```

#### 4.1.4. Receiving information from the server

After the application sent a request to the servlet, the object of connector checks if any response by using `getResponseCode()` method, if the value of response is `HTTP_OK` (Static variable of http connection). The object opens the input stream and gets ready for receiving information. Code like:

```
if(hc.getResponseCode() == HttpURLConnection.HTTP_OK){
    is = hc.openInputStream();
}
```

By calling the `read()` method, the object can get the response data and then store them in a string variable "result". The size of the input stream is defined by another integer variable "len". The letters in the response is read and stored byte by byte until `read()` method gets the value of -1 which means the end of the stream. So by these codes the response information from server can be stored in a string buffer "sb". The system converts "sb" to a byte array, at last gives the values to string variable "result" and displays the result on the screen. A piece of source code is given below for illustrating this function:

```
int len;
stringBuffer sb = new StringBuffer();
while((len = is.read()) != -1){
    sb.append((char)len);
}
byte a[] = new String(sb).getBytes();
String result = new String(a);
```

#### 4.1.5. Splitting the query result

When the client receives a response from the server, the result is not suitable to be shown on the screen directly. It needs to be split into a few of sub-strings and arranges them to an adaptable format. A pieces of code of an open source code is given below which is found on internet /9/ to illustrate how it works, the source code needs to be modified (setting separator, passing the original string) to meet the requirements of the project.

```
private String[] SplitString(String original) {
    Vector v = new Vector();
    String separator = "--";
    System.out.println("split start.....");
    // Parse nodes into vector
    int index = original.indexOf(separator);
    while(index>=0) {
        v.addElement(original.substring(0, index));
        original = original.substring(index+separator.length());
        index = original.indexOf(separator);
    }
    // Get the last node
    v.addElement(original);
    // Create splitted string array
    splitStr_B = new String[ v.size() ];
    if(v.size(>0) {
        for(int loop=0; loop<v.size(); loop++){
            splitStr_B[loop] = (String)v.elementAt(loop);
            System.out.println(loop+"-"+splitStr_B[loop]);
        }
    }
    return splitStr_B;
}
```

The response from the server is passed to the function as a parameter. The function creates a vector to store the sub-strings and specifies the symbol of separator, in this project "--" is used as separator. The function goes through the original string and counts how many separators in it, and then returns an integer variable which is named "index". If the value of index equal to 0, it means there is no separator exists so the whole original string is stored into the vector as one element. If the value of index more than 0, the function splits the sub-string from the first letter to the position which just before next separator's position and then save the sub-string into the vector, the function keeps doing throughout the whole original string by using "for()" loop. This function returns a string array at last.

Here is an example, a query for bus information is executed, the system gets response from server and prints out the result in the output console of Netbeans IDE as Fig . 22.

```

BUS NAME: 20
ESTIMATED TIME: 30 minutes
NOTE: none
STATIONS IN ROUTE:
--3--erottaja--vanha--Fredrikinkatu--Hietalahdentori--Hietalahti--Itamerenkatu --Eira

```

**Fig . 22. Splitting string example\_1**

The system passes the result to parameter of SplitString() function and starts splitting.

Step 1, After goes through the original string by indexOf(separator) method, the function returns the number of the location at where the separator appears for first time in the string, reads the letters before the separator position and adds them into a vector as the first element. So the first element is:

```

BUS NAME: 20
ESTIMATED TIME: 30 minutes
NOTE: NONE
STATIONS IN ROUTE:

```

Step 2, the rest part of the original string makes a new string without those letters which are already read including the separator, so the new string is obtained (3--erottaja--vanha--Fredrikinkatu--Hietalahdentori--Hietalahti--Itamerenkatu --Eira). The function repeats step 1.

Repeating step 2 until indexOf() gets the value -1, it means no separator exists anymore, so the rest letters composes the last element of the vector. The system defines a new string array and saves the elements of the vector into this array one by one by using elementAt() function. Fig . 23 is the string array which is got after splitting. It has 9 elements in the array. Elements from the second to eighth are added to a list of station names for advanced function.

```

0-
BUS NAME: 20
ESTIMATED TIME: 30 minutes
NOTE: none
STATIONS IN ROUTE:

1-3
2-erottaja
3-vanha
4-Fredrikinkatu
5-Hietalahdentori
6-Hietalahti
7-Itamerenkatu
8-Eira

```

**Fig . 23. Splitting string example\_2**



#### 4.1.6. Deal with invalid input data

The flow of all query functions have explained above, however, they are based on an ideal condition that the input data are usable, but what if the input data are invalid such as the user inputs a wrong bus name or an inaccurate station name or even the user does not input anything.

When the application calls query function, it requires the parameter which input by the user, the code is:

```
if(tfBus.getString().equals("")){  
    si2_B.setText("Input a bus name!");  
}
```

So if the function gets null value, then it does not establish the connection with server but shows a sentence for asking the user to input.

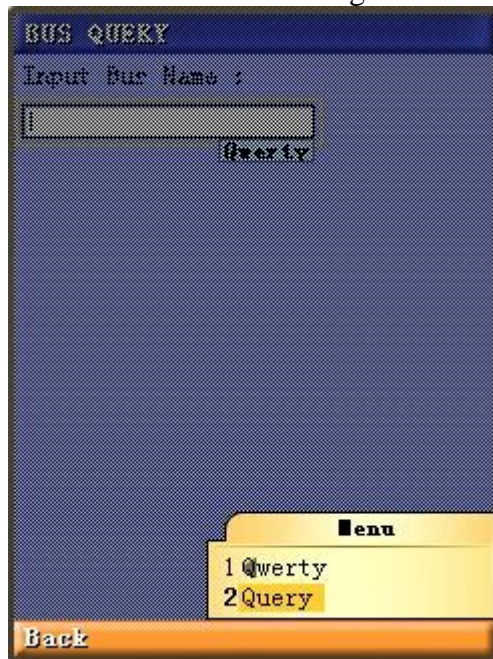


Fig . 24. Invalid input example\_1

Here is an example, the bus query function is called and the user leaves the input form in blank, in which he is supposed to put a bus name. Click Query button to execute the function as in Fig . 24.

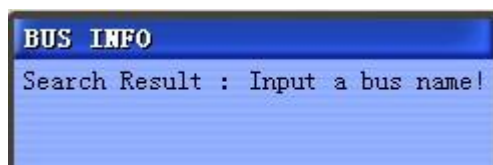


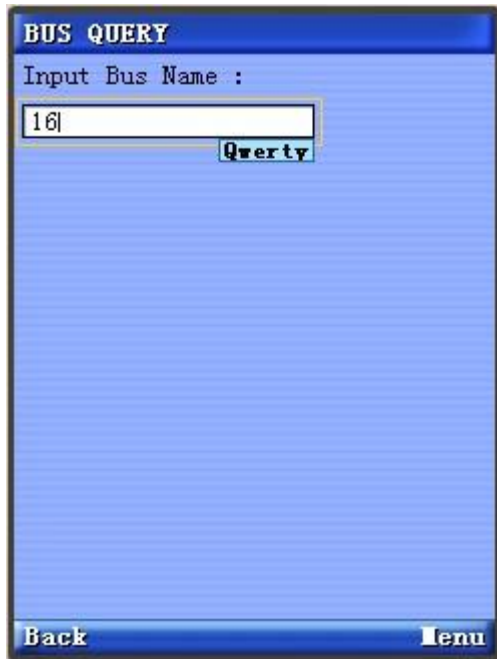
Fig . 25. Invalid input example\_2

An error sentence shows to the user as Fig . 25.

If the data is wrong, the client gets nothing in response, so if the size of response is smaller than 1, the system sets text to inform the user that the input data could be wrong.

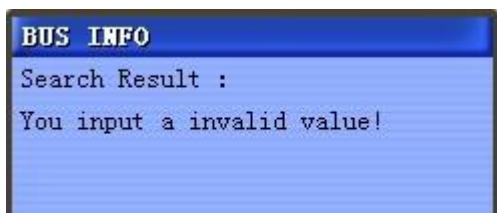
The code is:

```
if(srhBus.result.length()<1){  
    si2_B.setText("You have input a invalid value!");  
}
```



**Fig . 26. Invalid input example\_3**

Here is another example in Fig . 26, the user inputs a bus name "16" in input form, but actually it does not have such a bus name in the database.



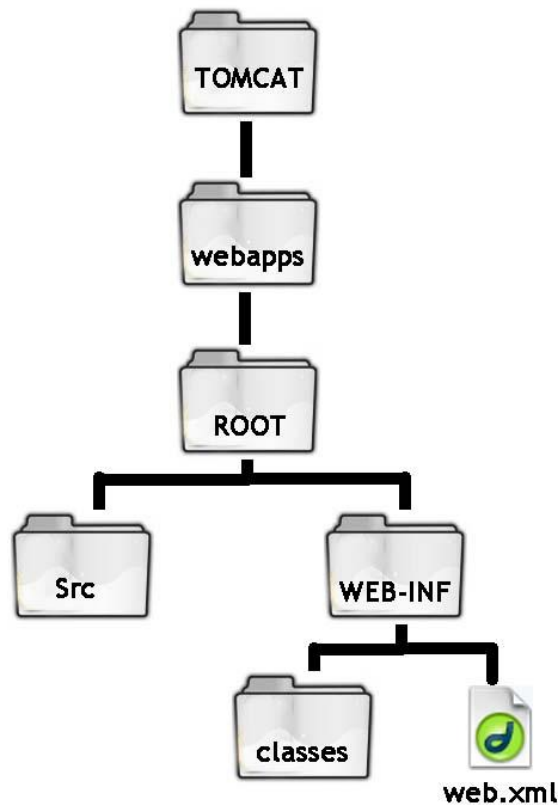
**Fig . 27. Invalid input example\_4**

Then the user gets the result as Fig . 27.

## 4.2. Server part

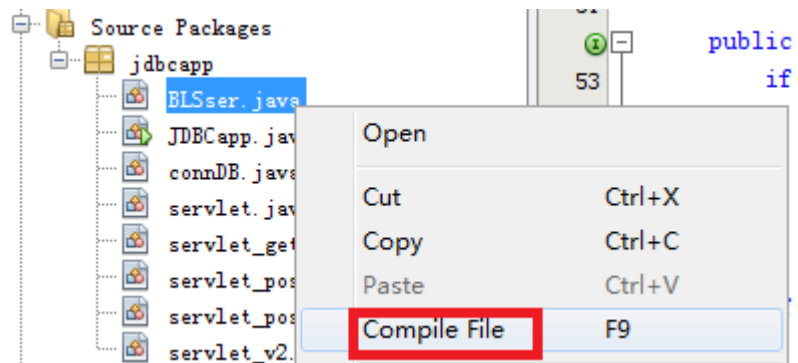
### 4.2.1. Servlet structure and organization

In this part, the procedures of deployment a servlet file are illustrated. The operations are divided into 5 steps for deploying a servlet under Windows OS /10/:



**Fig . 28. Tomcat deployment structure**

- 1) Builds the folder path for the servlet project as above in Fig . 28. Navigates to the Webapps folder of Tomcat, creates project folder in which creates another folder named "Src" (this folder is not an integrant element which is used for storing the \*.java source code file) and "WEB-INF" folder (the letters of WEB-INF folder MUST in capital letter), also creates "classes" folder and a web.xml file under WEB-INF. Puts the "\*. java" source code in Src folder and "\*. class" binary code in classes folder. The web.xml file specifies the relation between URL and servlet. /16/
- 2) Programs JAVA source code "\*. java", puts it into Src folder. Netbeans IDE is used for servlet programming.



**Fig . 29. Example of Compile File**

- 3) After the implementation of server source code, right clicks on the source file in project window on the left side and clicks Compile File as Fig . 29 shows. The \*.class file (which runs on server side as called servlet) can be found in “build” folder of the project. Copies this "\*.class" file then negative the Classes folder which is made under tomcat/Webapps/project/WEB-INF and paste it there.

```
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.
-->
<web-app xmlns="http://java.sun.com/xml/ns/javaee"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd"
version="3.0"
metadata-complete="true">
<display-name>Welcome to Tomcat</display-name>
<description>
Welcome to Tomcat
</description>
<servlet>
<servlet-name>jdbc</servlet-name>
<servlet-class>BLSser</servlet-class>
</servlet>
<servlet-mapping>
<servlet-name>jdbc</servlet-name>
<url-pattern>/bls</url-pattern>
</servlet-mapping>
</web-app>
```

**Fig . 30. Sample of XML file**

- 4) Creates deployment descriptor "web. xml" file in the "WEB-INF" folder. It is not necessary to remember the whole code in the xml file, they can be copied from an existing file from any example project and then modifies the following elements servlet name, servlet-class and url-pattern as Fig .30. There are two main elements <servlet> and <servlet-mapping> for each element has two child tags:

In <servlet> has: <servlet-name> and <servlet-class> tag. The name is symbolic only, it does not have to match the name of the java class. The purpose of this tag is to provides a way to match the same tag which in the <servlet-mapping> part.

In `<servlet-mapping>` has: `<servlet-name>` and `<url-pattern>`, `url-pattern` is used to specify which servlet to be called for an incoming URL.

```

信息: Server startup in 1825 ms
dbT =null, dbC1 =null, dbC2 =null, in JDBC.
url = jdbc:mysql://localhost:3306/busquerydb_v1?user=root&password=root

```

**Fig . 31. Outcome of servlet**

- 5) Clicks "startup. bat" under bin folder which is under tomcat server folder to start the server. Inputs the address, in this case the address is <http://localhost:8080/bls>. The tomcat server client gets result as above Fig . 31. There are three parameters in the figure: dbT, dbC1, dbC2, which values are null. Because there are not any data passed to the servlet, but it is enough to substantiate the deployment is successful.

### 4.2.2. Receiving and responding information from / to the client

In this chapter, the explanation about how servlet receives request from client and sends response back is given.

In server side, when there comes a request from client, the constructor gives the value of parameters to local variable by using `getParameter()`, which is the static function of request object:

```

String type = request.getParameter("type");
String cdt1 = request.getParameter("cdt1");
String cdt2 = request.getParameter("cdt2");

```

The server sends response to the client, first creates an object for `OutputStream`:

```
private OutputStream os = null;
```

Second, sets response type:

```
response.setContentType("application/octet-stream");
```

Then third, calls `write()` method to send data:

```
os.write(resp.getBytes()); flush the memory: "os.flush();
```

At the last, closes output stream:

```
os.close;
```

### 4.2.3. Servlet and Database connection

In the front of `jdbc()` function, some variables need to be declared which are used in every JDBC case, such as

```

String driveName = "com.mysql.jdbc.Driver";
String userName_db = "root";
String userPasswd_db= "root";
String database = "busQueryDB_v1";

```

```
String table = "schedule";
```

The process of communication can be divided into 6 steps between servlet and database.

1<sup>st</sup> step, Installs database drive:

```
DriverManager.registerDriver(new com.mysql.jdbc.Driver());
```

2<sup>th</sup> step, creates connection:

```
Connection conn = DriverManager.getConnection(
    "jdbc:mysql://localhost:3306/busDB","root","wxin2817");
```

3<sup>th</sup> step, creates statement variable:

```
Statement st = conn.createStatement();
```

4<sup>th</sup> step, executes statement:

```
ResultSet rs = st.executeQuery("Select * from schedule");
```

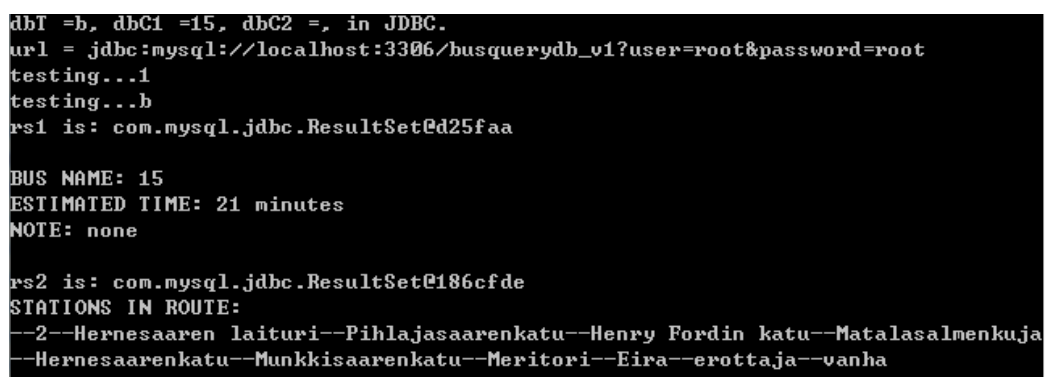
By the way, SQL query statement need to be put in double quotation marks.

5<sup>th</sup> step, disposes result for displaying in result console:

```
while (rs.next()){
    System.out.println(rs.getObject(1)+"\t"+rs.getObject(2)+"\t"+rs.getObject(3));
}
```

6<sup>th</sup> step, releases resource:

```
rs.close();
st.close();
conn.close();
```



```
dbf =b, dbC1 =15, dbC2 =, in JDBC.
url = jdbc:mysql://localhost:3306/busquerydb_v1?user=root&password=root
testing...1
testing...b
rs1 is: com.mysql.jdbc.ResultSet@d25faa

BUS NAME: 15
ESTIMATED TIME: 21 minutes
NOTE: none

rs2 is: com.mysql.jdbc.ResultSet@186cfde
STATIONS IN ROUTE:
--2--Hernesaaren laituri--Pihlajasaarenkatu--Henry Fordin katu--Matalasalmenkuja
--Hernesaarenkatu--Munkkisaarenkatu--Meritori--Eira--erottaja--vanha
```

**Fig . 32. Sample of query result**

A screen shot of the result in server window is shown in Fig . 32.

## **5. EVALUATION**

### **5.1. Achieved Results**

It is about to the end of the project, the main functionalities were accomplished successfully. The application allows the users to get their geographical location and then display on a Google static map, and some related functions like set markers or add paths on the map. The users are able to easily get start with the application and accomplish querying operations. There are some screen shots which are captured when the application is tested and shot explanation in appendix.

### **5.2. Drawbacks**

Because of the limitation of the ability and the time resource, it is still far away from the result that was expected at the beginning. Some disadvantages of the product are shared in this part. Firstly, the GUI is not that friendly as defined in requirements. Each screen should have background in the project plan, but J2ME do not support to do so to those displayable items, it is possible to insert a picture into these items but not background. The picture may take up those items' space. The solution is to convert each screen to canvas item and then program some classes manually which have same functionality with the items which are used in the project. Draw these classes in the application to replace those items. Secondly, for Google map, the name of marker has a limitation of only one character. The path between two terminals is shown as a straight line instead of following the roads, it is because the library of MidMaps (which is used in the project to call google map function) does not support for drawing polylines. The most serious imperfection is that the application gives nothing help if the route which user required need to exchange different bus from user's origination to destination.

### **5.3. Future work**

According to the disadvantages mentioned above, there is some work to be done to improve the application, such as GUI improvement with Light Weight User Interface Toolkit, replace the map function with original Google Map APIs, learn to implement drawing folded path following roads and city's geography, add function to let user move and change the zoom level of the map. The route search function improvement as well, this may take much time, and the modification may involve not only the client but also database.

## **6. CONCLUSIONS**

The project was carried out during more than two months. The basic objectives of the project were accomplished successfully. It is not good enough but a lot of programming knowledge had been learnt during this period, such as LWUIT for UI design, Google Map APIs, Client/Server system development, JDBC technology and so on. What is most important besides these is that the project provided a full experience of software application development (which included requirements specification, system design and analysis and code implementation) from the very beginning and in every detail.



## 7. REFERENCES

- /1/ Alessandro La Rosa, MidMaps: Google Maps Java ME library, [WWW-Document], <<http://www.jappit.com/blog/midmaps-google-maps-java-me-library/>>, 28.11.2011.
- /2/ it&c solutions.com, How to use GPS location services and Google Static Map API in J2ME MIDlets, [WWW-Document], <<http://www.itcsolutions.eu/2011/04/27/how-to-use-gps-location-services-and-google-static-map-api-in-j2me-midlets/>>, 27.11.2011.
- /3/ sun.com, LIGHTWEIGHT USER INTERFACE TOOLKIT, [WWW-PDF], <<http://projetomobilerss.googlecode.com/svn-history/r3/trunk/pdf/plugin-LWUIT-M3DD-Tutorial-1.0.pdf>>, 12.11.2011.
- /4/ SUN.com, Servlet Life Cycle, [WWW-Document], <[http://java.sun.com/j2ee/tutorial/1\\_3-fcs/doc/Servlets4.html](http://java.sun.com/j2ee/tutorial/1_3-fcs/doc/Servlets4.html)>, 1.12.2011.
- /5/ IBM.com, Servlet lifecycle, [WWW-Document], <<http://publib.boulder.ibm.com/infocenter/iserics/v5r3/index.jsp?topic=%2Frzatz%2F51%2Fprogram%2Fservlife.htm>>, 30.11.2011.
- /6/ Sharat, Explain servlet life cycle, [WWW-Document], <<http://sharat.wordpress.com/2006/09/06/31-explain-servlet-life-cycle/>>, 20.12.2011.
- /7/ James, Keogh, J2ME: the complete reference, 1<sup>st</sup> edition, 15.12.2011.
- /8/ jdbc-tutorial.com, Java JDBC Tutorial, [WWW-Document], <<http://www.jdbc-tutorial.com/>>, 25.12.2011.
- /9/ Qasitouch, How to Split String in J2ME, [WWW-Document], <<http://www.developer.nokia.com/Community/Discussion/showthread.php?97769-How-to-Split-String-in-J2ME.>>, 22.12.2011.
- /10/ intrinsyc.com, Deploying a Servlet in TomCat, [WWW-Document], <[http://j-integra.intrinsyc.com/support/com/doc/servlet\\_com/deployingservlettoTomCat.html](http://j-integra.intrinsyc.com/support/com/doc/servlet_com/deployingservlettoTomCat.html)>, 23.12.2011.
- /11/ Aaltio, Vanhanen, User Requirements Document, [WWW-Document], <<http://www.soberit.hut.fi/T-76.115/05-06/ohjeet/template/requirements.html>>, 23.12.2011.
- /12/ Wikipedia, Global Positioning System, [WWW-Document], <<http://en.wikipedia.org/wiki/GPS>>, 23.12.2011.

/13/ MySQL.com, Download Connector/J, [WWW-Document],  
<<http://dev.mysql.com/downloads/connector/j/>>, 21.12.2011.

/14/ sun.com, J2ME Optional Packages, [WWW-Document],  
<<http://developers.sun.com/mobility/midp/articles/optional/>>, 21.12.2011.

/15/ java.com, What is J2ME or Java ME?, [WWW-Document],  
<[http://www.java.com/en/download/faq/whatis\\_j2me.xml](http://www.java.com/en/download/faq/whatis_j2me.xml)>, 19.12.2011.

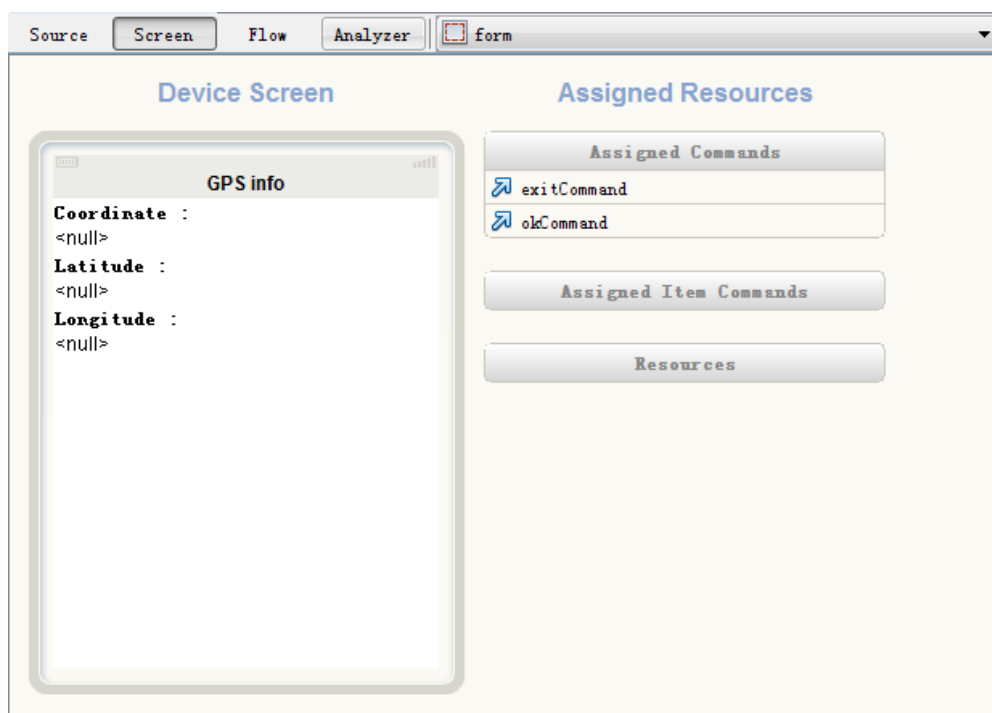
/16/ cs.calstatela.edu, Running a Servlet under Tomcat, [WWW-Document],  
<[http://cs.calstatela.edu/~abbott/Courses/CS\\_320b/Running%20a%20Servlet%20under%20Tomcat.html](http://cs.calstatela.edu/~abbott/Courses/CS_320b/Running%20a%20Servlet%20under%20Tomcat.html)>, 22.12.2011.

/17/ sun.com, J2ME for Home Appliances and Consumer Electronic Devices, [WWW-Document], <<http://developers.sun.com/mobility/configurations/articles/cdc/>>, 22.12.2011.

## APPENDIX

### GPS EXAMPLE

- Here is a small Java application for testing. A new project is established in Netbeans IDE, and create a new MIDlet as well in the project folder. Add three string items in Screen view, named: COORDINATE, LATITUDE and LONGITUDE. COORDINATE are create as a label so no more further value for it, but the geo-signal are receiving from the satellite will distribute to LATITUDY and LONGITUDE separately. And there is one more command is needed to run the function, named as RUN as the figure shows below.



- Define three variables: provider, latitude and longitude.

```
LocationProvider provider = null;  
double latitude;  
double longitude;
```

- Create a `javax.microedition.location.Criteria` instance under `okCommand` in `COMMANDACTION`, used to select the location provide.

```
Criteria crt = new Criteria();  
crt.setCostAllowed(true);  
crt.setPreferredPowerConsumption(Criteria.NO_REQUIREMENT);
```

- Make a LocationProvider instance, using previous defined criteria and use it to query the GPS module for the current location.

```
provider = LocationProvider.getInstance(crt);
Location location = provider.getLocation(60);
Coordinates cdn = location.getQualifiedCoordinates();
```

- Distribute the geographical data to LATITUDE and LONGITUDE variables, then display them on the screen.

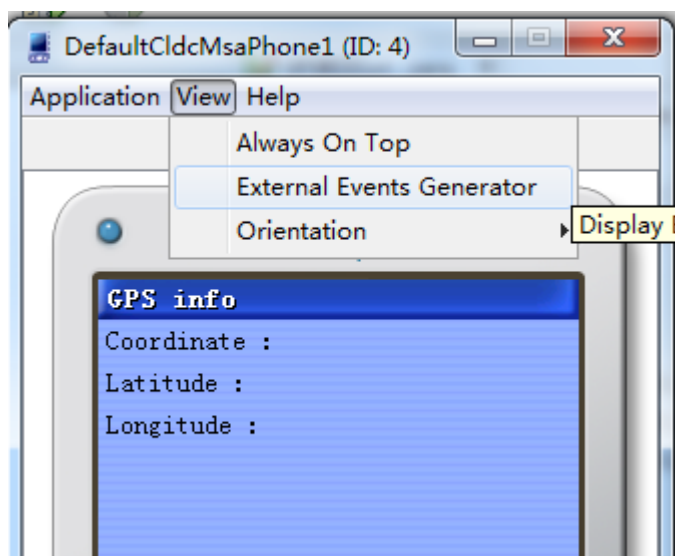
```
latitude = cdn.getLatitude();
Longitude = cdn.getLongitude();
stringItem1.setText(new String(Double.toString(latitude)));
stringItem2.setText(new String(Double.toString(longitude)));
```

Now the application is ready for testing GPS:

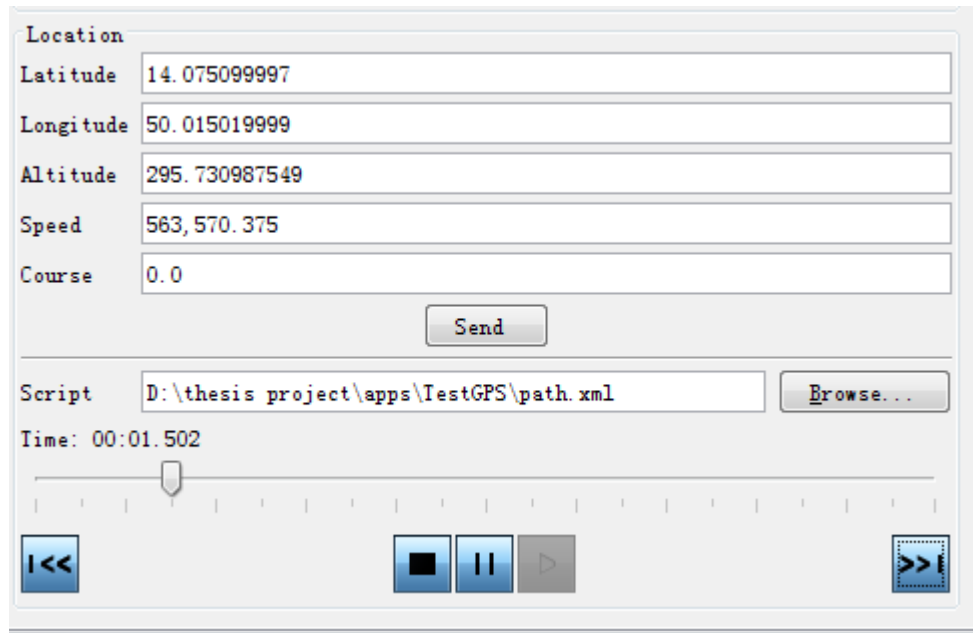
1. Use the text editor to make a simple location script(XML file) that specifies a starting point (time="0") and moves to a new point in ten seconds:

```
<waypoints>
  <waypoint time="0"
    latitude="14" longitude="50" altitude="310" />
  <waypoint time="10000"
    latitude="14.5" longitude="50.1" altitude="215" />
</waypoints>
```

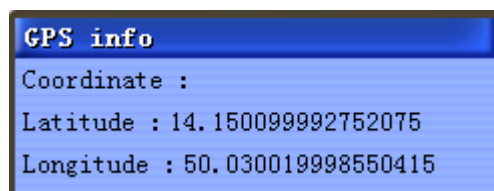
2. Run the application, and load the script into external events generator by clicking view tap in emulator window.



3. Click browse button in external event generator control panel, and navigate to the location of the script file, then click play button(right arrow). Then the latitude and longitude will be changing over time.



4. Back to the emulator and press RUN button to get the coordinate.



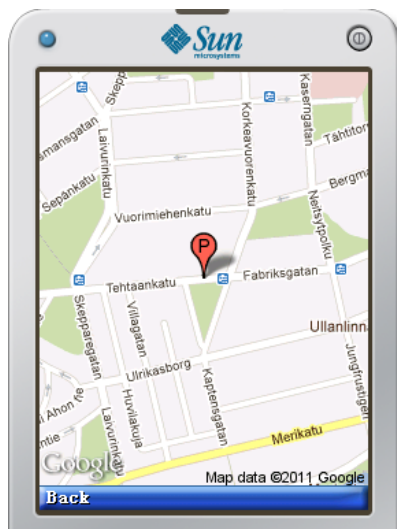
## TESTING

In this part, the testing results of the four functions are given by some screen shots.

First, GPS function:



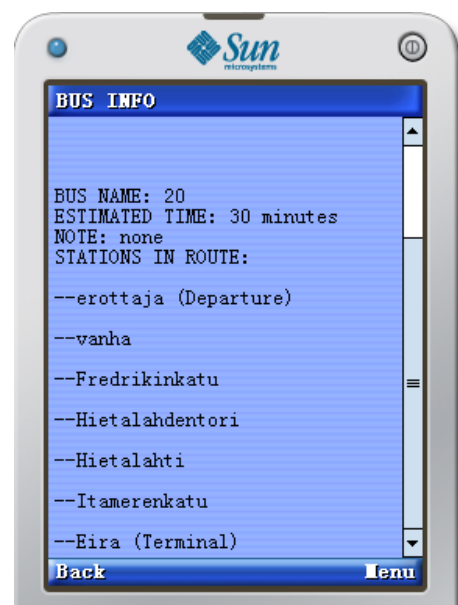
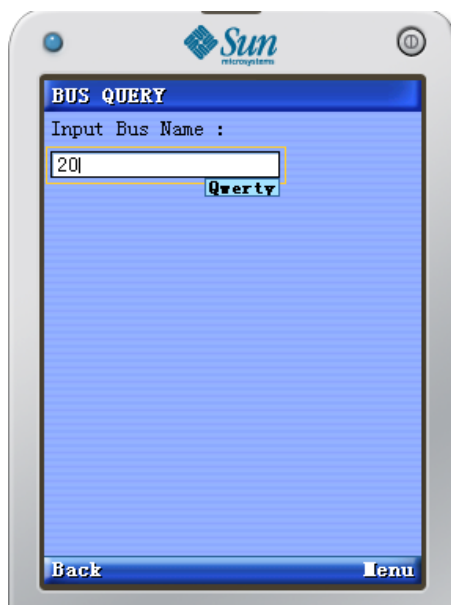
Run GPS Locating. The value of (60.158229828, 24.945129395) is supplied to application (according to GPS example before to pass the data to application), this is a particular station's geographical value which is grabbed from the database of stop table, the station is somewhere in Helsinki. Then the user can get the result as below :



Second, bus query function:



Run Bus Search function, here the user is asked to input a bus name, the value of (20) is given as an example, this is also get from database of bus table. Then the user can get some information of this bus, if a name which do not excited in database has been input, then an error message will be displayed as mentioned in Deal With Invalid Input part.



Then the user can press "More Info" command, the application will put these stations' name in a list, then selects one from them to obtain the time table of this station.

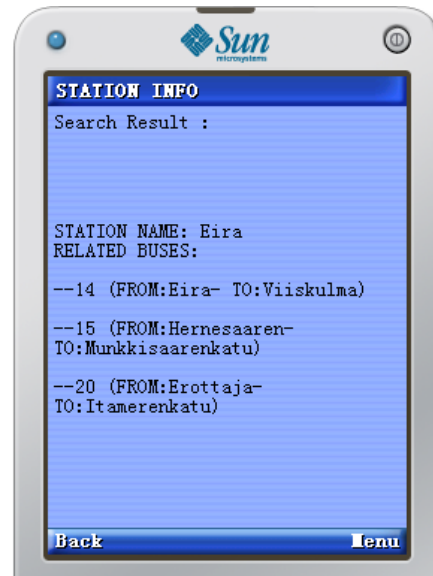


Third, station query function:

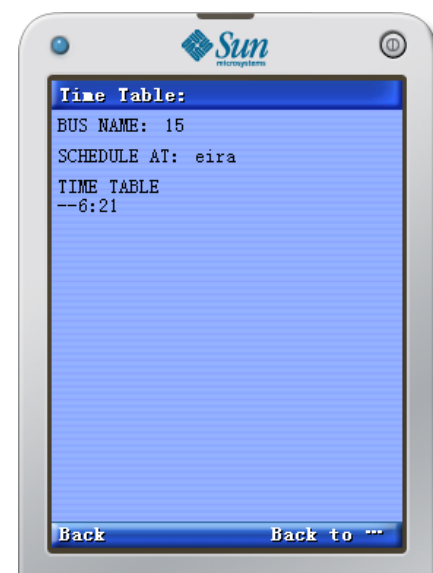
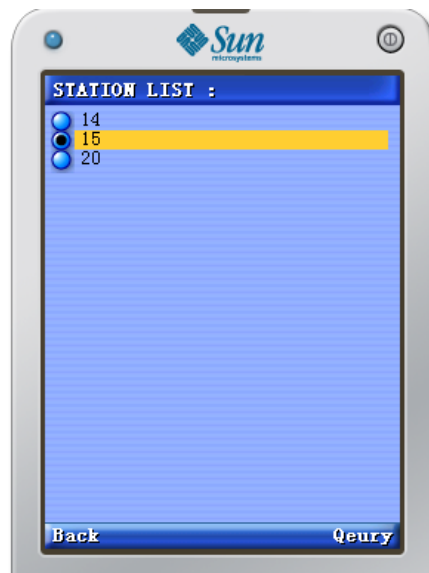




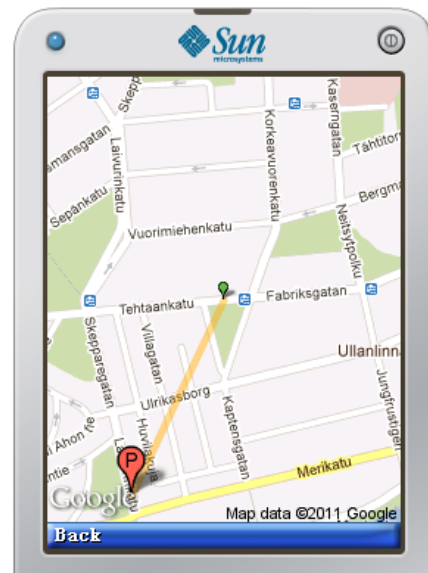
Run Station search function, it also requires a input of station name. type a station name (Eira station) into the text field, then press "Query" command. The data of station are displayed on screen.



Go ahead select "More Info" the user can get the list and choose one station to see its time table like it been done in bus query function.



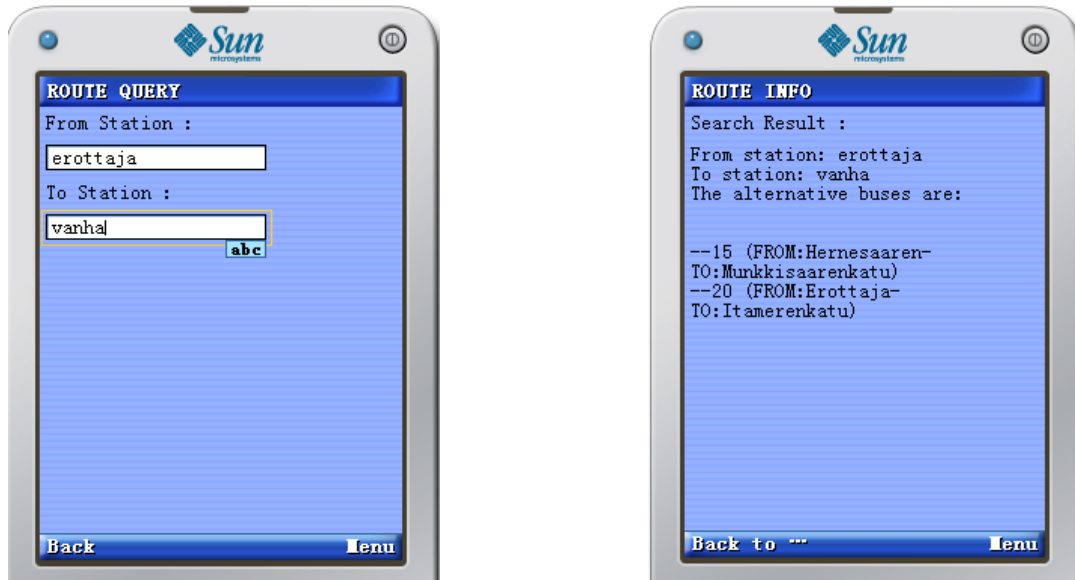
Beside this, the user can also locate the station on the Google static map and show the path from user location to station. To do so, turning back to Station Info screen and press "Station Locating" command. The center with a small green marker is where user stands, another marker refers to the station.



Fourth, route query function:



Run Route Search, it requires two station names as taking off station and destination station. Input the names then press "Query" command. It will list all possible routes between these two stations.



There are two routes in the figure. Press "More Info" to get list of route names, and choose one bus of them to get the time table.



So this is the product the author did so far, it is not good enough to put into use, there are some bugs happen while using the application sometimes. These will be done in future.