



Electric

al and Computer Engineering Department

**Communication and Electronics Engineering Program
Computer Engineering program**

Bachelor Thesis

Graduation Project

GPS Car Tracking System

Project Team

Samah Rateb AL-badawi

Ola Mohammad Rashed

Project Supervisor

Eng. Ayman Wazwaz

**Hebron – Palestine
june,2012**

GPS Car Tracking System

Project Team:

SamahRatebAL-badawi

Ola Mohammed Rashed

Project Supervisor:

Eng.Ayman Wazwaz

This under-graduate project report submitted to computer and electrical engineering department in college of engineering and technology.

Palestine Polytechnic University

For accomplishment the requirements of the bachelor degree in computer system, and telecommunication engineering fields.

Palestine Polytechnic University
Hebron-Palestine
May,2012

Dedication

To Palestine Polytechnic University, and all our teachers.

To our parents, brothers, and sisters

To our all friends and colleagues

To all, whom raising us to be the persons we are today, whom being with us in every step of the way, through good times and bad, thanks you for all the support that you have always give us, helping us to succeed and instilling in us the confidence that we are capable of doing anything we put in our minds.

Thank you for everything.

Acknowledgements

This graduation project has been supported by the Deanship of Graduate Studies and Scientific Research through "Distinguished Graduation Projects Fund", special thanks for them.

Special thanks to Palestine Polytechnic university for its attention, support, and chances that give us to be graduated students, and improve our selves and abilities to be engineering.

Thanks to our supervisor Eng .AymanWazwaz for his support and advice during our works, thanks to our teachers who help us in the project, especially Eng. MajdiZalloum, and thanks to all teachers in electrical and computer engineering department.

Thanks to Palearth organization, especially Mr.AliTaha, fore provide us with some project resources .

Special thanks to our parents, families, friends, and anyone help us, and trusted deeply that we can success.

To each of the above, we extend our deepest appreciation.

Abstract

Depending on the high usage of mobile phones, and the large services that it provides, and the needs to always be connected to others, and share information with them, we built a mobile based car tracking system.

Car tracking system, will give users the ability to determine their locations, and other cars locations, during any travel interval; by providing an obvious and clear observation, to the movement of group of cars in different places, and reflecting their paths on a map. Also it provides the ability of continuously checking of speed of the moving car.

It uses mobiles built in GPS Receiver, to determine the coordinates and speed of the moving cars, then sends them over GSM network, by GPRS services based on connection technique, frequently to a central server. In order to process and plot on the map, with complete and clear system interface, and provide the user with accurate history for cars movements and speeds in the past period, which can be retrieved at any time.

We built our system, using all needed hardware components and software programs, and we created a way to interface them with each other, to completely achieve the system objectives in efficient way.

بفعل الاستخدام المتزايد للهواتف المتنقلة ، وما تقدمه من خدمات متزايدة، وبفعل الحاجة المستمرة الى التواصل مع الآخرين والمشاركة بالمعلومات المختلفة، فقد قمنا ببناء نظام تتبع حركة السيارات بالاعتماد على الهواتف المتنقلة.

نظام تتبع السيارات، هذا النظام الذي سيتيح للأفراد القدرة على تحديد مساراتهم، وكذلك تحديد مسارات حركة الآخرين، ضمن فترات مختلفة، وتمثيلها برسم توضيحي على خرائط جغرافيه، ومعرفة سرعة السيارات خلال تعقبها.

سيستخدم النظام، نظام تحديد المواقع العالمي الموجود ضمن الهواتف المتنقلة، ليتمكن من تحديد إحداثيات لموقع الحركة وسرعة السيارة المتحركة، ومن ثم إرسالها عبر شبكة الاتصالات اللاسلكية إلى الخادم المركزي بطريقة اتصال معينه وبشكل متكرر بعد كل فتره محدده ، حيث سيتم تمثيل هذه الإحداثيات كمسارات متتابعة من الحركة، على الخرائط الجغرافية الممثلة للمنطقة المتابعة.

لقد قمنا باستخدام جميع الاجزاء والبرمجيات اللازمة لإتمام المشروع ، وقد قمنا بعمل ربط بين هذه الأجزاء بطريقه متكاملة، لتحقيق أفضل كفاءة وعمل للمشروع.

CHAPTER ONE : INTRODUCTION

1

1.1	Overview	2
1.2	Aim of the project	2
1.3	Requirements	2
1.3.1	Functional requirements	2
1.3.2	Non-Functional requirements	3
1.4	Assumption and dependencies	3
1.5	Literature Review(related projects)	4
1.5.1	GPS and GSM based vehicle tracking system	4
1.5.2	Vehicle tracking and control systems	5
1.5.3	OpenGTS™ - Open GPS Tracking System	5
1.5.4	Vehicle Tracking System using GPS and GSM modem	6
1.6	Project Schedule	7
1.7	General system description	9
1.8	Report Content	9

CHAPTER TWO : THEORITICAL BACKGROUND

11

2.1	Overview	12
2.2	GPS technology	12
2.2.1	Definition	12
2.2.2	GPS components	13
2.2.3	How it works?	14
2.2.4	The GPS satellite system	14
2.2.5	GPS in our project	14
2.3	Coordinate systems	15
2.3.1	World Geodetic System 1984 (WGS84)	15
2.3.2	Earth Gravitational Model (EGM96)	15
2.3.3	Earth Gravitational Model (EGM2008)	15
2.3.4	North American Datum of 1927 (NAD27)	15
2.3.5	North American Datum of 1983 (NAD83)	15
2.3.6	Coordinate systems in our project:	16
2.4	GSM technology	16
2.4.1	Definition	16
2.4.2	GSM transmission and frequency allocation	16
2.4.3	GSM Network Architecture	17
2.4.3.1	The Mobile Station (MS)	17
2.4.3.2	The Base Station Subsystem (BSS).	18
2.4.3.3	The Network and Switching Subsystem (NSS).	18
2.4.3.4	Operation and Support Subsystem (OSS)	18
2.4.4	GSM In Our Project	19
2.5	GPRS service	19
2.5.1	Definition	19

2.5.2	GPRS and packet data in GSM	19
2.5.3	GPRS Network Elements	20
2.5.4	GPRS in our project	21
2.6	HTTP	21
2.6.1	Definition	21
2.6.2	Terminology	22
2.6.3	HTTP Client/Server Communication	23
2.6.4	HTTP Message	23
2.6.4.1	Message Types	23
2.6.4.2	Message Headers	24
2.6.4.3	Message Body	24
2.6.5	Methods	25
2.6.5.1	POST	25
2.6.6	HTTP in our project	26
2.7	Mobile phone (cellular phone)	26
2.7.1	Mobile Operating systems (mobile software platform)	27
2.7.1.1	Android operating system:	27
2.7.1.2	Android architecture	28
2.7.1.3	Android features	29
2.7.2	Mobile programming language	30
2.7.2.1	Java architecture	30
2.7.2.2	Java development environment (E_clips)	31
2.7.2.3	Java features	31
2.7.3	Mobile phone in our project	32
2.8	C# Programming language	32
2.8.1	C# architecture	33
2.8.2	C# features	34
2.8.3	C# in our project	34
2.9	Software design blocks	35

CHAPTER THREE :		36
CONCEPTUAL DESIGN		
3.1	Overview	37
3.2	General system block diagram	37
3.3	System main components	37
3.3.1	GPS Receiver	38
3.3.1.1	Android OS	38
3.3.1.2	Galaxy S	39
3.3.1.3	Why GPRS	40
3.3.1.4	Connection Options	40
3.3.1.4.1	GPRS mobile with PC interface.	40
3.3.1.4.2	HTTP Connection	41
3.3.2	Central server	41
3.3.2.1	Geographic maps	41
3.4	System flow chart	42
3.4.1	Fill registration data and get coordinates	43

3.4.1.1	Fill registration data	43
3.4.1.2	Get coordinates and speed	44
3.4.2	Send data and coordinates to main server	45
3.4.3	Receive data and view system interface	46
3.4.4	Register cars, and display movement on map	48
3.5	Data Flow Diagram	49
3.6	System Functions	50
3.6.1	Functional block diagram	50
3.6.2	Mobile Functions	51
3.6.3	Main server functions	51

CHAPTER FOUR: DETAILED DESIGN	53
--	-----------

4.1	Overview	54
4.2	System classes Model Diagram	54
4.2.1	Mobile Software UML diagram	54
4.2.1.1	classes implementation	54
4.2.2	Desktop Software UML diagram	58
4.2.2.1	classes implementation:	58
4.3	System use case	66
4.4	System Sequence diagram:	67
4.4.1	send registration request sequence diagram	67
4.4.2	receive new request sequence diagram	68
4.4.3	select mode sequence diagram	69
4.4.4	added car to system sequence diagram	70
4.4.5	Send GPS coordinates sequence diagram	72
4.4.6	plot received coordinates sequence diagram	73
4.4.7	Log In to system sequence diagram	73
4.4.8	Activate tracking sequence diagram	74
4.4.9	zoom sequence diagram	75
4.4.10	display cars information sequence diagram	75
4.4.11	display cars history sequence diagram	77

CHAPTER FIVE: SYSTEM TESTING	78
---	-----------

5.1	Overview	79
5.2	Sub-System Testing:	79
5.3	Installation and preparing the system	79
5.3.1	Mobile application	79
5.3.2	Server Application	84
5.4	Testing Scenarios	89
5.4.1	TEST 1	90
5.4.2	TEST 2	92

5.4.3	TEST 3	94
5.4.4	TEST 4	96
5.4.5	TEST 5	96

CHAPTER SIX: CONCLUSION AND RECOMMENDATION		98
6.1	Overview	99
6.2	Conclusion and Achievement	99
6.3	Challenges	99
6.4	Future work Recommendation	100
	References	101

FIGURES

Figure 1.1 system description	9
Figure 2.1 Earth-circling satellites orbits	13
Figure 2.2 GSM Band Spectrum	16
Figure 2.3 GSM Network's Components	18
Figure 2.4 GPRS network elements	20
Figure 2.5 architecture of Android OS	28
Figure 2.6 java main architecture	31
Figure 2.7 software design block	35
Figure 3.1: general system block diagram	37
Figure 3.2: system flow chart	43
Figure 3.3: get coordinates flow chart	44
Figure 3.4: Send data and coordinates flow chart	45
Figure 3.5: server received data, and coordinates	47
Figure 3.6: received data processing	48
Figure 3.7: Data Flow Diagram	49
Figure 3.8: Functional Block Diagram	50
Figure 4.1: Mobile UML model	55
Figure 4.2 Desktop UML model Diagram	59
Figure 4.3 System use case diagram	66
Figure 4.4 send registration request sequence diagram	68
Figure 4.5 receive new request sequence diagram	69
Figure 4.6 select mode sequence diagram	70
Figure 4.7 added car to system sequence diagram	71
Figure 4.8 Send GPS coordinates sequence diagram	72
Figure 4.9 plot received coordinates sequence diagram	73
Figure 4.10 Log In to system sequence diagram.	74
Figure 4.11 Activate tracking sequence diagram	75
Figure 4.12 zoom sequence diagram	76
Figure 4.13 display cars information sequence diagram	76
Figure 4.14 display cars history sequence diagram	77
Figure 5.1: Opening apk file	80
Figure 5.2: Installing application	80

Figure 5.3: Installation completed	81
Figure 5.4: Welcome Screen	82
Figure 5.5: Registration	82
Figure 5.6: Select mode	83
Figure 5.7: Walking mode/GPS off	83
Figure 5.8: Driving mode/ coordinates and speed	83
Figure 5.9 GPSCarTrackingSystem Project file	84
Figure 5.10 GPSCarTrackingSystem Welcome Screen	84
Figure 5.11 GPSCarTrackingSystem Log In Screen	85
Figure 5.12 GPSCarTrackingSystem Main Screen	85
Figure 5.13 system screen car's dynamic region	86
Figure 5.14 new connection message	86
Figure 5.15 System registration screen	87
Figure 5.16 System Car's Information form	87
Figure 5.17 System Car's history form	88
Figure 5.18 System Car's Information form	89
Figure 5.19 test1 result tracked paths	91
Figure 5.20 test2 result tracked paths	93
Figure 5.21 desktop test2 error	93
Figure 5.22 test3 result tracked paths	95
Figure 5.23 test4 result tracked paths	96
Figure 5.24 test5 result tracked paths	97

TABLES

Table 1.1: Timing plane for first semester	8
Table 1.2: Timing plane for second semester	8
Table 2.1 Data Rate for GPRS Software	20

1

CHAPTER ONE

INTRODUCTION

- 1.1 Overview**
- 1.2 Aim of the project**
- 1.3 Requirements**
- 1.4 Assumption and dependencies**
- 1.5 Literature Review**
- 1.6 Project schedule**
- 1.7 General system description**
- 1.8 Report contents**

1.9 Overview

In this chapter, we will introduce the main idea for our project, the aim we try to achieve by this system, the main requirement that will be needed in our work, some of the assumption and dependencies that we will rely on, we also made some literature review, and we mention some of these project that are related to our work and show how we will be different from there, we also show general block diagram for our system.

1.10 Aim of the project

The project main idea is to provide car tracking system, which gives a clear vision about car movements through it's travel.

Our system will enable the customer to easily track his cars, over country or even worldwide, and for companies with a large number of vehicles, knowing where customer's cars are at every moment in real time and know the speed of traveling at any interval.

1.11 Requirements

The requirements of our project will be described in two parts the functional and nonfunctional requirements as follow:

1.3.1 Functional requirements:

- The system must, track users cars in real time, over its travel along any time, this will require a hardware to receive the GPS coordinates, and send it through communication network to the server.
- The system must be able to track more than one car at the same time, and update each car movement separately, and this require server software that deal with coordinates and geographic maps.

- The system should be able to provide clear history for cars, and their travel over long period of times, and this require server software to deal with database and cars history.

1.3.2 Non-Functional requirements:

- Usability, the system will be used from car's owner, and company's owners, or even any customer need to track one of its valuable components, so it must be useful and easy to use.
- Availability, the system will be able for any person wants to use, and also it should be available at any place during car travel, so we rely on GPS system that have wide spread in our countries, and if some region does not have this we try to solve this limitation.
- Reliability, the system will be reliable, so it can give you a proper location of your car, with acceptable minimum error.

And for all these requirements to be achieved we will need the following hardware and software:

Hardware:

- Mobile phones with GPS services.
- GSM network with GPRS capability
- Main server.

Software :

- Mobile programming language (Android Programming).
- Programming software for map display.
- Graphical user interface (GUI).

- Database for cars and history.

1.12 Assumption and dependencies

The following assumptions are considered in our project:

- Each car that will be tracked has a mobile with GPS service to receive GPS data.
- Longitude and latitude coordinates received from the GPS satellite and speed of car determined using number of GPS satellites will be sent using GSM network service to the central server.

1.13 Literature Review (Related Projects)

Many projects around the world have similar points to this project, but we try to be different, and make it suitable for our country, with reasonable cost.

And here we will mention some similar projects, that related to our work, and show the difference between our works and their works.

1.13.1 GPS and GSM based vehicle tracking system

Description:

This Project presents a positioning system using GPS and GSM-SMS services. The system permits localization of the automobile and transmitting the position to the owner on his mobile phone as a short message (SMS) at his request.

The system can be interconnected with the car alarm system and alert the owner on his mobile phone. This tracking system is composed of a GPS receiver, Microcontroller and a GSM Modem. GPS Receiver gets the location information from satellites in the form of latitude and longitude.

The Microcontroller processes this information and this processed information is sent to the user/owner using GSM modem. The presented application is a low cost solution for

automobile position and status, very useful in case of car theft situations, for monitoring adolescent drivers by their parents as well as in car tracking system applications. The proposed solution can be used in other types of application, where the information needed is requested rarely and at irregular period of time, when requested. [1]

Advantage:

- This project can used in different types of applications.

Disadvantage:

- The use of sms service is considered to be highly cost compared to other technologies such as GPRS.
- Sending the location depends on the request of the owner only; so if the owner become busy or forget to request the position of the tracked vehicle , he will not get any information.

Differences:

- We want to use GPRS service instead of SMS.
- Tracked cars/vehicles continuously send their location frequently without a request from the owner.

1.13.2 Vehicle tracking and control systems

Description:

This Project Promote a Pre-competitive Research Project and demonstrate Telematics in Public Transportation. Provide buses time schedule with starting and arrival time, It enables communication of location data from the device fitted in the vehicle to a center using GSM/CDMA, GPS, SMS technology. Applied in 'Koyambedu' Asia's Largest Bus Terminus.[2]

Advantage:

- Handle more than 2500 bus trips per day.
- Control in real time
- Cost effective project, pay utilization , productivity and efficiency.

Disadvantage:

- The used technologies don't fit the existence technologies in Palestine.
- Complicated.

Differences:

- Our project will provide same service with lower cost and using the available technology which is limited compared to those used in 'Koyambedu' project.

1.13.3 OpenGTS™ - Open GPS Tracking System

Description:

This Project is the first available open source project designed specifically to provide web-based GPS tracking services for a "fleet" of vehicles.

Until now, OpenGTS™ has been downloaded and put to use in over 90 countries around the world to track many 1000's of vehicles, assets around all 7 Continents. The types of vehicles and assets tracked include taxis, delivery vans, trucks, trailers, farm equipment, personal vehicles, service vehicles, containers, ships, ATVs, personal tracking, cell phones, and more. OpenGTS™ is very highly configurable and scalable to large enterprises.[3]

Advantage:

- Use in over 90 countries.
- Operating system independent: written entirely in Java, using technologies such as Apache Tomcat for web service deployment, and MySQL for the data store. Cost effective project, pay utilization , productivity and efficiency.

Disadvantage:

- Support only determined types of devices.

Differences:

- Many cars will be tracked at the same time in real time manner, all locations will be send to the same direction (central server).

1.13.4 Vehicle Tracking System using GPS and GSM modem

Description:

In this Project, it is proposed to design an embedded system which is used for tracking and positioning of any vehicle by using Global Positioning System (GPS) and Global system for mobile communication (GSM).

Microcontroller is used for interfacing to various hardware peripherals. The current design is an embedded application, which will continuously monitor a moving Vehicle and report the status of the Vehicle on demand.

A GSM modem is used to send the position (Latitude and Longitude) of the vehicle from a remote place. The GPS modem will continuously give the data, the latitude and longitude indicating the position of the vehicle. The GPS modem also gives many parameters as the output, but only the NMEA data coming out is read and displayed on to the LCD. The same data is sent to the mobile at the other end from where the position of the vehicle is demanded. An EEPROM is used to store the mobile number.

When the request by user is sent to the number at the modem, the system automatically

sends a return reply to that mobile indicating the position of the vehicle in terms of latitude and longitude.[4]

Advantage:

- This project can be used in different types of applications.

Disadvantage:

- The driver has to do and repeat some steps in order to send its location, he has to press the Button to send the Location to a stored number.(Switch on the kit, give a call)

Differences:

- No effort from the drivers of vehicles or the owner while tracking, because
- Tracked cars/vehicles continuously send their location every 10 minutes without a request from the owner.

1.14 Project Schedule

- **Stag1: Select the idea**

Determine the idea of the project, the motivation, and the main objective we intend to achieve.

- **Stage2: Preparing for the project**

In this stage, more and deeper determination of the tasks, and steps we want to perform, is done.

- **Stage3: Project Analysis**

In this step, a study of the all possible design options to determine our own design.

- **Stage4: Determine the project requirement**

after determine our design scheme, we specify all the needed requirement for the user and the system, software and hardware. And try to bring them to be ready for the implementation stage.

- **Stage5: Studying the Principles**

This stage of the project is necessary to study the GSM, GPS, GPRS, mobile programming language, main server programming language.

- **Stage6: Documentation Writing**

Documenting the project will begin from the first stage to the last stage.

- **Stage7: make the hardware available**

In this stage, the needed hardware mobiles and main server's PC will be brought for the next step

- **Stage8: build up the software**

the programming of the project code is started and will be downloaded to the mobile and the central server. Make sure the needed maps are existing.

- **Stage9: testing the system**

In this stage cars will move, tracking will start, feedback will be provided.

- **Stage10: Writing Documentation**

The documentation will continue from the first stage to the last one in parallel.

Table 1.1: Timing plane for first semester

Task/ week	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
S1															
S2															
S3															
S4															
S5															
S6															

Table 1.2: Timing plane for second semester

Task/ week	16	17	18	19	20	21	22	23	24	25	26	27	28
S7													
S8													
S9													
S10													

1.15 General system description

From the general description for our project idea, and main requirements, we will need major parts to achieve our goals, the hardware which used to receive the GPS signals from the satellites, and this will be a mobile with GSM service, then these information and any other we want to add will be sent through the GSM base station to the central server through http connection , at the server side and after receiving transmitted data we will convert these coordinates to be suitable to reflect accurate values to our city coordinates, and then using software to display car movements on the map, and provide the system user with all available services in clear and complete interface. Figure 1.1 shows a main block diagram of the project parts.

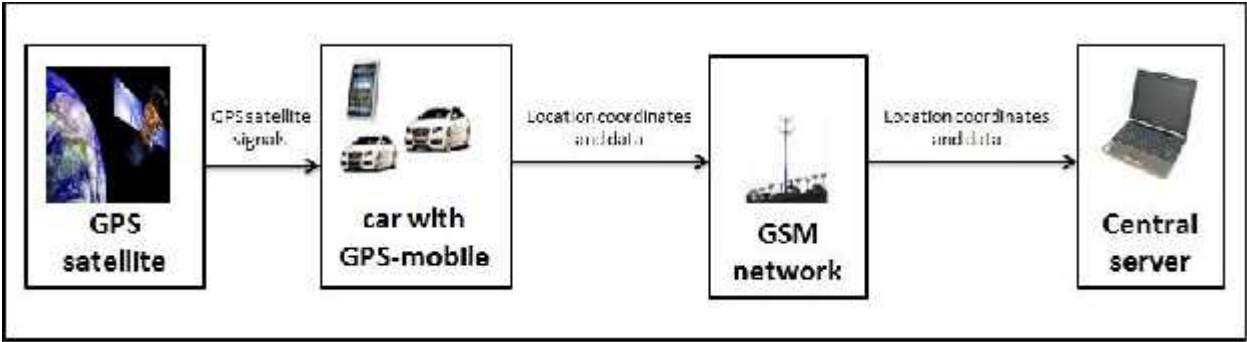


Figure 1.1 system description

1.16 Report contents

This report includes the following six chapters:

- **Chapter one: Introduction.**

In this chapter, we introduce a general overview about the project, the main idea of the project, requirements, assumption and dependences, the time schedule, and the estimated cost.

- **Chapter two: Theoretical Background.**

This chapter includes the Theoretical background related to the main idea of the project, and technologies, hardware, and software used in the project.

- **Chapter three: Project Conceptual Design.**

In this chapter we talk about the design concepts; the general block diagram that show how the system works, the system flow chart, the data flow diagram, and the functional block diagram that mention all the implemented functions in the system.

- **Chapter four: Detailed Design.**

In this chapter we talk about the detailed software engineering design including UML model diagram for both mobile and server application software classes. Then system use case diagram will be described, also the sequence diagram for use cases will be mentioned.

- **Chapter five: System Testing.**

In this chapter, the whole testing stage will be described, that's mean a test of an entire interconnected set of components and software for the purpose of determining proper functions and achieving the desired goals of the system. We will talk about testing of each part of the system, describe the scenarios, represent each test and it's errors, challenges and modifications, and also we will figure out the error rate.

- **Chapter six: Conclusion and Recommendations**

In this chapter, we will mention what we achieved in this project and the conclusion for all things that we have done, also we will talk about the challenges that we faced and ending with recommendation needed for the future work.

2

CHAPTER TWO

THEORITICAL BACKGROUND

- 2.1 Overview**
- 2.2 GPS Technology**
- 2.3 Coordinate Systems**
- 2.4 GSM Technology**
- 2.5 GPRS Service**
- 2.6 HTTP**
- 2.7 Mobile Phone**
- 2.8 C# Programming Language**

2.1 Overview

In this chapter we will mention the basic theoretical information, some technologies, and learn about some devices that will be used in our project. In order to use them in an appropriate way and take the desired advantages of them. So make it easy for the reader to understand and interact with the project.

We will talk about the GPS technology, GSM/GPRS technology; a collection of hardware includes the mobiles, GPS receivers, GPRS modems.

Software that will be used represented by a programming language on the mobile phone, software on the central server in order to receive locations coordinates and process them, and then plot on the geographic maps, and we will mention some of popular used systems coordinate.

And also we will talk about the connection technique, which represents important part in our system, where the data and coordinates will be transmitted from tracking mobile to the main server.

2.2 GPS Technology

2.2.1 Definition

The global positioning system (GPS) is a system for determining the geographic location of people with GPS receivers. Consists of constellation of 24 well-spaced satellites that makes a revolution every 12 hours.

- Global: Anywhere on earth but not inside buildings, underground, or anywhere else not having a direct view of a portion of the sky.
- Positioning: Give answers to the following questions, Where are you? How fast are you moving and in what direction? What direction should you go to get to some other specific location?
- System: A collection of components with connections between them.

The official U.S. Department of Defense name for GPS is NAVSTAR (Navigation System with Time And Ranging) which is a constellation of 24 satellites orbiting the earth, broadcasting data that allows users on or near the earth to determine their spatial positions.[5]

2.2.2 GPS components

The GPS technology consists of the following components:

- **The Earth:** it's the major component of GPS, it's mass and its surface, and the space immediately above. The mass of the Earth holds the satellites in orbit.
- **Earth-circling satellites:** as shown in figure 2-1, twenty four solar powered radio transmitters, forming a constellation such that several are visible from any point on Earth at any given time. The satellites are at middle altitude of 20200 kilometers(Km) , each is in a 12 hour orbit.



Figure 2.1 Earth-circling satellites orbits [6]

- **Ground based stations:** the GPS satellites are crammed with electronics, so they require monitoring. This is done by four ground based stations located on Ascension Island, at Diego Garcia, in Hawaii, and Kwajalein. Each satellite passes over at least one monitoring station twice a day. Information developed by the monitoring station is transmitted back to the satellite, which in turn re-broadcasts it to GPS receivers.

- **GPS Receivers:** it consists of
 - An antenna whose position the receiver reports
 - Electronics to receive the satellite signals
 - A microcomputer to process the data that determines the antenna position, and to record position values
 - Controls to provide user input to the receiver
 - Screen to display information.

- **The United States Department of Defense:** The U.S. DoD is developing and maintaining NAVSTAR.[5]

2.2.3 How it works?

GPS satellites circle the earth twice a day in a very precise orbit and transmit signal information to earth. GPS receivers take this information and use triangulation to calculate the user's exact location. Essentially, the GPS receiver compares the time a signal was transmitted by a satellite with the time it was received. The time difference tells the GPS receiver how far away the satellite is. Now, with distance measurements from a few more satellites, the receiver can determine the user's position.

To calculate a 2D position (latitude and longitude) and track movement, GPS receiver must be locked on to the signal of at least three satellites. With four or more satellites in view, the receiver can determine the user's 3D position (latitude, longitude and altitude). Once the user's position has been determined, the GPS unit can calculate other information, such as speed, bearing, track, trip distance, distance to destination, sunrise and sunset time and more.
[5]

2.2.4 The GPS satellite system

The first GPS satellite was launched in 1978, a full constellation of 24 satellites was achieved in 1994, and each satellite is built to last about 10 years. Replacements are constantly being built and launched into orbit. A GPS satellite weighs approximately 2000 pounds and is about 17feet across with the solar panels extended. Transmitter power is only 50 watts or less.[5]

AVASTAR uses radio waves instead of sound: The waves that are used to measure the distance are electromagnetic radiation (EM). They move a lot faster than sound. Regardless of frequency, in vacuum, EM moves at about 299,792.5Km/sec, which is roughly 186,282 statute miles per hour.[5]

2.2.5 GPS in our project

We will use GPS to determine the real time location of the moving tracked car, using GPS receiver built in the mobile phone. Mobile needs GPS receiver plus processing to calculate position, and read assistance data from network. The handset takes GPS measurements, and report these to the central server in the network.

2.3 Coordinate systems

There are many coordinates system, which are systems that used to represent coordinates position of a point or geometric elements that can appear in different formats and they are different from each other related to the reference, which used to create them.

Some of these common modern systems mentioned here:

2.3.1 World Geodetic System 1984 (WGS84):

The World Geodetic System, is one of the coordinates system used in GPS, where it consists of three-dimensional Cartesian coordinate system and an associated ellipsoid, so that WGS84 positions can be described as either XYZ Cartesian coordinates or latitude, longitude and ellipsoid height coordinates.[7]

2.3.2 Earth Gravitational Model (EGM96):

Earth Gravitational Model geoids 96, defines the nominal sea level surface by means of a spherical harmonics series of degree 360, which will provides about 100km horizontal resolution. And the deviations of the EGM96 geoids from the WGS 84 reference ellipsoid range from about 150m to about 85m .[7]

2.3.3 Earth Gravitational Model (EGM2008):

Earth Gravitational Model geoids 2008 is a new higher fidelity model for the original WGS 84, This new model will have a geoids with a resolution approaching 10km, requiring over 4.6 million terms in the spherical expansion, versus 130317, EGM96 and 32,757 in WGS 84.[7]

2.3.4 North American Datum of 1927 (NAD27) :

North American Datum of 1927, is a datum based on the Clarke ellipsoid of 1866. The reference or base station is located at Meades Ranch in Kansas, There are over 50,000 surveying monuments throughout the US and these have served as starting points for more local surveying and mapping efforts.[7]

2.3.5 North American Datum of 1983 (NAD83):

North American Datum of 1983, is an earth-centered datum based on the Geodetic Reference System of 1980. The size and shape of the earth was determined through measurements made by satellites and other sophisticated electronic equipment; the measurements accurately represent the earth to within two meters.[7]

2.3.6 Coordinate systems in our project:

The received coordinates will be in one of the common coordinates systems, that depend on the type of mobile being used to transmit the coordinates. In our system, we will depend on American coordinate systems, where our mobile operating system android, will support these types of coordinates in accurate manner.

2.4 GSM technology

2.4.1 Definition

GSM (Global System for Mobile communications) is a second-generation digital network, supporting voice and simple data services, including “dial-up” data and text messaging. Open, digital cellular technology used for transmitting mobile voice and data services. GSM supports voice calls and data transfer speeds of up to 9.6 kbps, together with the transmission of SMS (Short Message Service). It predates Code Division Multiple Access (CDMA), which is especially strong in Europe. Enhanced Data rates for GSM Evolution (EDGE) (also known as Enhanced GPRS (EGPRS) is faster than GSM and was built upon GSM.

The system built based on cell phone technology. GSM is the dominant digital mobile phone standard for most of the world. It determines the way in which mobile phones communicate with the land-based network of towers.[8]

2.4.2 GSM transmission and frequency allocation

Radio systems such as GSM can have hundreds of frequencies specified for use within a specified part of the frequency spectrum,. GSM needs multiple frequencies to allow different frequencies to be used, in different geographical parts of the network, to avoid radio interference within the communication channels.

GSM operates in many frequency, the 900MHz and 1.8GHz bands in Europe, and the 1.9GHz and 850MHz bands in the US. The 850MHz band is also used for GSM and 3G in Australia, Canada and many South American countries.

For GSM Frequency Allocation, the spectrum assigned for the various GSM bands, In each case, there is an uplink (from mobile phone to base station), and a downlink (from base station to mobile phone). It is standard practice for the uplink to use the lower frequencies in cellular systems, as the attenuation of this link will be marginally less than the corresponding downlink. Transmitting and receiving on different frequency bands is known as Frequency Division Duplex (FDD).

Each of the bands is divided into a number of radio By carriers. These carriers are separated in all cases by 200 kHz. The GSM 900 bands are each 25 MHz wide, and therefore each contains 124 carriers. The end carriers are positioned at 200 kHz inside the band edges. As shown in figure 2.2, the lowest carrier in the 900 uplink band is 890.2 MHz, and the highest is 914.8 MHz.

The 1900 bands (United States) contain 299 carriers. Carriers are assigned in pairs (uplink and downlink), and each pair is always separated by a fixed amount, equal 45 MHz at 900, and 95 MHz at 1800. A number called the Absolute Radio Frequency Channel Number (ARFCN) commonly refers to the pairs.

GSM also has the advantage of using SIM (Subscriber Identity Module) cards in. The SIM card, which acts as digital identity, is tied to the cell phone service carrier’s network rather than to the handset itself. This allows for easy exchange from one phone to another without new cell phone service activation.

The main purpose of the GSM network is to facilitate easier access to cellular and satellite platforms across international lines. Using digital technology, it employs both speech and data channels in its system. At minimum these channels operate on the second generation

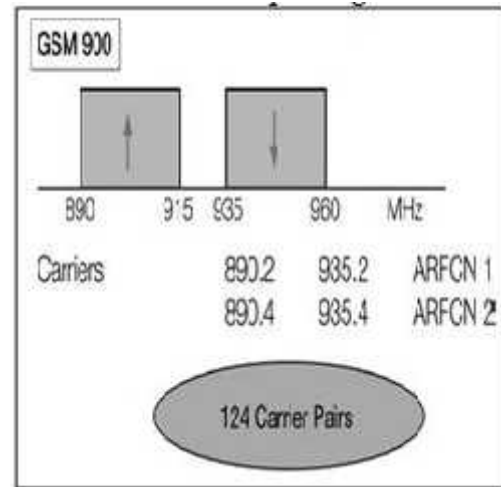


Figure 2.2 GSM Band Spectrum[8]

(2G) network, but many use the third generation (3G) system to offer these services to clients. This enables the exchange of information at high-speed data rates via satellites and mobile cellular towers across networks and company lines.[8]

2.4.3 GSM Network Architecture

The GSM network can be divided into four main parts:

2.4.3.1 The Mobile Station (MS)

Consists of following two main components: **The Subscriber Identity Module (SIM)** which protected by a four-digit Personal Identification Number (PIN). In order to identify the subscriber to the system, and **Mobile equipment/terminal (ME)** with different types of terminals distinguished principally by their power and application.

2.4.3.2 The Base Station Subsystem (BSS).

It may be divided into two parts: Base Station Controller (BSC) which controls a group of BTSs and manages their radio sources. And Base Transceiver Station (BTS) which usually placed in the center of a cell, its transmitting power defines the size of a cell.

2.4.3.3 The Network and Switching Subsystem (NSS).

Its main function is to manage the communications between the mobile users and other users. It also includes data bases needed in order to store information about the subscribers and to manage their mobility. The different components of the NSS are: Mobile Switching Center (MSC), Gateway MSC (GMSC), Home Location Register (HLR), Visitor Location Register (VLR), Authentication Center (AuC), Equipment Identity Register (EIR), GSM Interworking Unit (GIWU).

2.4.3.4 Operation and Support Subsystem (OSS)

It is connected to components of the NSS and the BSC, in order to control and monitor the GSM system. It is also in charge of controlling the traffic load of the BSS. It must be noted that as the number of BS increases with the scaling of the subscriber population some of the maintenance tasks are transferred to the BTS, allowing savings in the cost of ownership of the system.[8]

All previous networks' elements are shown in the following figure, that shows the relation between them.

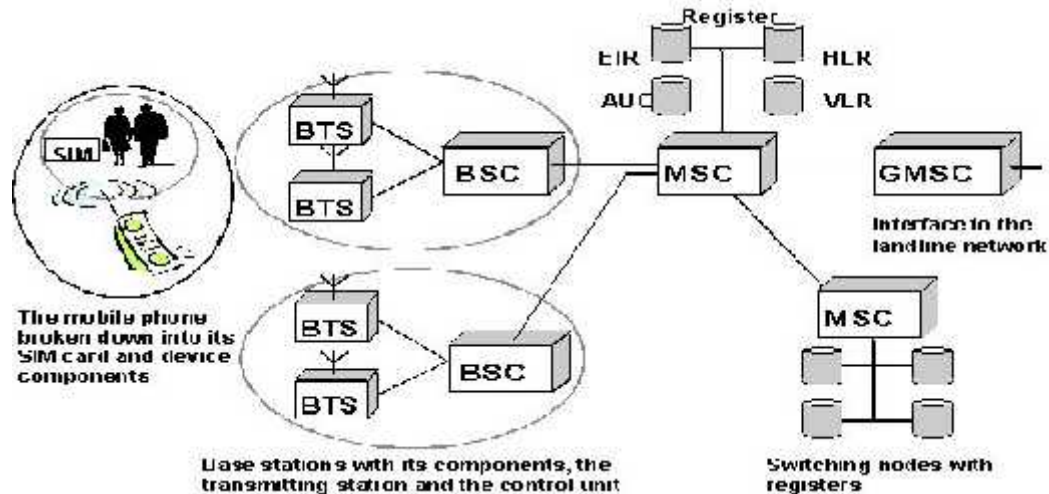


Figure 2.3 GSM Network's Components[9]

2.4.4 GSM In Our Project

The usage of the GSM technology in this project represented by transferring the data from the cars that contain the GPS receiver (or the mobile phone) to the central server. these data will contained position, and other information the driver want to send.

2.5 GPRS service

2.5.1 Definition

General Packet Radio Services (GPRS), is a packet-based wireless communication service that provides data rates from 56 up to 114 Kbps, and continuous connection to the Internet for mobile phone and computer users. The higher data rates allow users to take part in video conferences, and interact with multimedia Web sites, and similar applications using mobile handheld devices as well as notebook computers. GPRS is based on Global System for Mobile communication (GSM) and complements existing services such circuit-switched cellular phone connections and the Short Message Service (SMS).

In theory, GPRS packet-based services cost users less than circuit-switched services, since communication channels are being used on a shared-use, as-packets-are-needed basis rather than dedicated to only one user at a time. It is also easier to make applications available to mobile users, because the faster data rate means that middleware currently needed to adapt applications to the slower speed of wireless systems are no longer be needed.

GPRS also complements Bluetooth, a standard for replacing wired connections between devices with wireless radio connections. In addition to the Internet Protocol (IP), GPRS supports X.25, a packet-based protocol that is used mainly in Europe. GPRS is an evolutionary step toward Enhanced Data GSM Environment (EDGE) and Universal Mobile Telephone Service (UMTS).[10]

2.5.2 GPRS and packet data in GSM

GPRS adds a packet-based data capability to the GSM system, and brings IP networking into the GSM world. As well as modifications within the handsets and the radio interface, GPRS also involves the addition of new packet routers into the GSM core network, producing a core network with separate circuit- and packet switched domains. Using GPRS, each user only requires radio resources when a packet of information is sent or arrives, it does not require the channel for the entire duration of the data transfer. In this way, the single timeslot can be made available for more than one user, allowing radio channels to be shared, by up to seven users in GPRS.

The overall data rates in GPRS are specified for each timeslot, varying depending on what is known as the coding scheme. There are four coding schemes (CS) in GPRS (known as CS1, CS2, CS3, and CS4). The data rates for each coding scheme are shown in table 2.1. Note that each overall channel is actually designed to carry 22.8 kbps (as in standard GSM connections), and the extra capacity is actually used for error protection. Where there is a good error protection for CS1, but not so good for CS4. In practice, only coding schemes 1 and 2 are widely used.

CS1 and CS2 offer good error detection and correction with low throughput; in the first step of GPRS software only these two techniques may be used. CS3 and CS4 provide higher throughputs but have little or no error correction capabilities.

Due to the packet switched characteristics of GPRS software the allocation of the available timeslots may vary from one instant to the next. So, it may have 8 timeslots at one time and 4 later on. [11]

User Data Rate	CS1	CS2	CS3	CS4
1 Timeslot	9.05 kbps	13.4 kbps	15.6 kbps	21.4 kbps
8 Timeslot	72.4 kbps	107.2 kbps	124.8 kbps	171.2 kbps

Table 2.1 Data Rate for GPRS Software[11]

2.5.3 GPRS Network Elements

Additional elements are required to provide the General Packet Radio Services (GPRS), including:

- core network GPRS support nodes (GSNs), comprising:
 - Serving GSN (SGSN).
 - Gateway GSN (GGSN).

These elements use IP (Internet Protocol) technology to route information to and from the mobile handset. They are effectively IP routers with modifications to allow for managing subscriber access, mobility, and IP sessions. A GPRS data session would be handled by the GSN rather than the MSC, which is used for voice. In addition, GPRS operation requires the BTS to be GPRS-capable, and a packet control unit must be available, usually at or within the BSC. These two base station subsystem (BSS) elements allow the data packets to transfer in the right format on the correct GPRS radio channels.[8]

2.5.4 GPRS in our project

After determining the driver location coordinates by the GPS receiver in the mobile, this coordinates and other additional information will be sent using the GPRS technology from mobile, through the base stations, to another GPRS mobile near to the central server to be stored and processed.

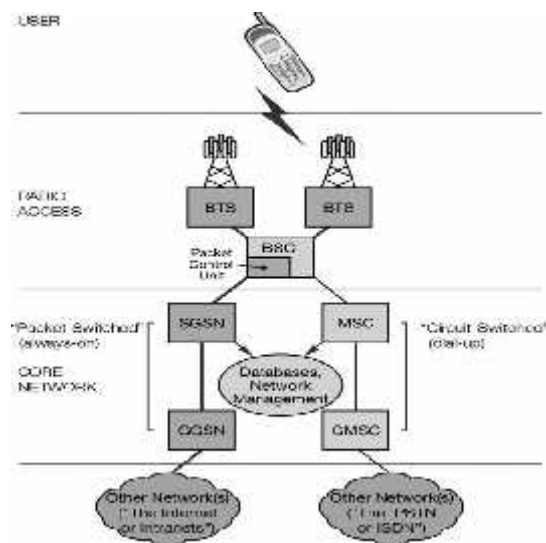


Figure 2.4 GPRS network elements.[8]

2.6 HTTP

2.6.1 Definition

HTTP stands for Hypertext Transfer Protocol. It's the network protocol used to deliver virtually all files and other data on the World Wide Web, whether they're HTML files, image files, query results, or anything else. Usually, HTTP takes place through TCP/IP sockets.

HTTP concepts include (as the Hypertext part of the name implies) the idea that files can contain references to other files whose selection will elicit additional transfer requests. Any Web server machine contains, in addition to the Web page files it can serve, an HTTP daemon, a program that is designed to wait for HTTP requests and handle them when they arrive. Your Web browser is an HTTP client, sending requests to server machines. When the browser user enters file requests by either "opening" a Web file (typing in a Uniform Resource Locator or URL) or clicking on a hypertext link, the browser builds an HTTP request and sends it to the Internet Protocol address (IP address) indicated by the URL. The HTTP daemon in the destination server machine receives the request and sends back the requested file or files associated with the request.

A browser is an HTTP client because it sends requests to an HTTP server (Web server), which then sends responses back to the client. The standard (and default) port for HTTP servers to listen on is 80, though they can use any port.[12]

2.6.2 Terminology

HTTP's specification uses a number of terms to refer to the roles played by participants in, and objects of, the HTTP communication.

- **Connection**

A transport layer virtual circuit established between two programs for the purpose of communication.

- **Message**

The basic unit of HTTP communication, consisting of a structured sequence of octets and transmitted via the connection.

- **Request**

An HTTP request message from a client to a server includes, within the first line of that message, the method to be applied to the resource, the identifier of the resource, and the protocol version in use.

- **Response**

An HTTP response message. Each request message sent by an HTTP client to a server prompts the server to send back a *response message*. Actually, in certain cases the server may in fact send two responses, a preliminary response followed by the real one. Usually though, one request yields one response, which indicates the results of the server's processing of the request, and often also carries an entity (file or resource) in the message body.

- **Resource**

A network data object or service that can be identified by a URI (Uniform Resource Identifiers). Resources may be available in multiple representations (multiple languages, data formats, size, and resolutions) or vary in other ways.

- **Entity**

The information transferred as the payload of a request or response. An entity consists of meta information in the form of entity-header fields and content in the form of an entity-body.

- **Representation**

An entity included with a response that is subject to content negotiation. There may exist multiple representations associated with a particular response status.

- **Variant**

A resource may have one, or more than one, representation(s) associated with it at any given instant. Each of these representations is termed a 'variant'. Use of the term 'variant' does not necessarily imply that the resource is subject to content negotiation.

- **Client**

A program that establishes connections, for the purpose of sending requests.

- **User agent**

The client which initiates a request. These are often browsers, editors, spiders (web-traversing robots), or other end user tools.

- **Server**

An application program that accepts connections in order to service requests by sending back responses. Any given program may be capable of being both a client and a server; our use of these terms refers only to the role being performed by the program for a particular connection, rather than to the program's capabilities in

general. Likewise, any server may act as an origin server, proxy, gateway, or tunnel, switching behavior based on the nature of each request.

- **Origin server**

The server on which a given resource resides or is to be created.[12]

2.6.3 HTTP Client/Server Communication

In its simplest form, the operation of HTTP involves only an HTTP client, usually a Web browser on a client machine, and an HTTP server, more commonly known as a Web server. After a TCP connection is created, the two steps in communication are as follows:

1. **Client Request:** The HTTP client sends a request message formatted according to the rules of the HTTP standard—an HTTP Request. This message specifies the resource that the client wishes to retrieve, or includes information to be provided to the server.
2. **Server Response:** The server reads and interprets the request. It takes action relevant to the request and creates an HTTP Response message, which it sends back to the client. The response message indicates whether the request was successful, and may also contain the content of the resource that the client requested, if appropriate. [12]

2.6.4 HTTP Message

2.6.4.1 Message Types

HTTP messages consist of requests from client to server and responses from server to client.

HTTP-message = Request | Response.

The format of the request and response messages are similar, and English-oriented. Both kinds of messages consist of:

- an initial line,
- zero or more header lines,
- a blank line (i.e. a CRLF by itself), and
- an optional message body (e.g. a file, or query data, or query output).

2.6.4.2 Message Headers

HTTP header fields, which include general-header, request-header, response-header, and entity-header fields. Each header field consists of a name followed by a colon (":") and the field value. Field names are case-insensitive. The field value MAY be preceded by any amount of LWS(linear white space).

The common forms:

message-header = field-name ":" [field-value]

field-name = token

field-value = *(field-content | LWS)

field-content = <the OCTETs making up the field-value and consisting of either *TEXT or combinations of token, separators, and quoted-string>

2.6.4.3 Message Body

The message-body (if any) of an HTTP message is used to carry the entity-body associated with the request or response. The message-body differs from the entity-body only when a transfer-coding has been applied, as indicated by the Transfer-Encoding header field.

message-body = entity-body

| <entity-body encoded as per Transfer-Encoding>

Transfer-Encoding MUST be used to indicate any transfer-codings applied by an application to ensure safe and proper transfer of the message. Transfer-Encoding is a property of the message, not of the entity, and thus MAY be added or removed by any application along the request/response chain. The rules for when a message-body is allowed in a message differ for requests and responses.

The presence of a message-body in a request is signaled by the inclusion of a Content-Length or Transfer-Encoding header field in the request's message-headers. A server SHOULD read and forward a message-body on any request; if the request method does not include defined semantics for an entity-body, then the message-body SHOULD be ignored when handling the request.

For response messages, whether or not a message-body is included with a message is dependent on both the request method and the response status code. All responses to the HEAD request method **MUST NOT** include a message-body, even though the presence of entity- header fields might lead one to believe they do. All 1xx (informational), 204 (no content), and 304 (not modified) responses **MUST NOT** include a message-body. All other responses do include a message-body, although it **MAY** be of zero length.[12]

2.6.5 Methods

There are many methods to send request from client to server, the set of common methods for HTTP/1.1 is defined below.

- "OPTIONS"
- "GET"
- "HEAD"
- "POST"
- "PUT"
- "DELETE"
- "TRACE"
- "CONNECT"

We will use the POST method in our project so we will explain it in details below.

2.6.5.1 POST

The POST method is used to request that the origin server accept the entity enclosed in the request as a new subordinate of the resource identified by the Request-URI in the Request-Line. POST is designed to allow a uniform method to cover the following functions:

- Annotation of existing resources;

- Posting a message to a bulletin board, newsgroup, mailing list, or similar group of articles;
- Providing a block of data, such as the result of submitting a form, to a data-handling process;
- Extending a database through an append operation.

The actual function performed by the POST method is determined by the server and is usually dependent on the Request-URI. The posted entity is subordinate to that URI in the same way that a file is subordinate to a directory containing it, a news article is subordinate to a newsgroup to which it is posted, or a record is subordinate to a database.

The request URI is not a resource to retrieve; it's usually a program to handle the data you're sending. And the responses to this method are not cacheable; the HTTP response is normally program output, not a static file.

The most common use of POST, by far, is to submit HTML form data to CGI scripts. In this case, the *Content-Type: header* is usually *application/x-www-form-urlencoded*, and the *Content-Length: header* gives the length of the URL-encoded form data. The CGI script receives the message body through STDIN, and decodes it. Here's a typical form submission, using POST:

```
POST /path/script.cgi HTTP/1.1
From: frog@jmarshall.com
User-Agent: HTTPTool/1.0
Content-Type: application/x-www-form-urlencoded
Content-Length: 32

"Message Body"
```

You can use a POST request to send whatever data you want, not just form submissions. Just make sure the sender and the receiving program agree on the format.[12]

2.6.6 HTTP in our project

After determining GPRS to transfer data between mobile in cars and main server, we choose the HTTP connection to represent the GPRS service, considering the phone as a HTTP client and PC as a HTTP server.

2.7 Mobile phone (cellular phone):

Mobile phone is one of the most sophisticated and modern techniques that spread dramatically in our societies, it transformed from a simple use to make calls, to be used nowadays in many complex and useful applications.

Back to the history of mobile phones, we found that the first mobile telephone call was made from a car in St. Louis, Missouri, USA on June 17, 1946, using the Bell System's Mobile Telephone Service.

Many updates and developments are quickly continued on mobile phones, and still updates until now, Mobile phones become nowadays a very smart, efficient, and small device that puts the whole world in your hands, and support large number of modern techniques included within this small device.[13]

2.7.1 Mobile Operating systems (mobile software platform):

Like Microsoft's Windows, Apple's Macintosh, and Linux, and many others desktop operating systems, the evolution that has occurred on the mobile phone also enable us to control a mobile devices , applications and services, with their special operating systems, as the case with computer operating systems.

The most common mobile operating systems are:

- Android from Google Inc. (open source, Apache).
- BlackBerry OS from RIM (closed source, proprietary).
- Symbian OS from the Symbian Foundation (open public license).
- Windows Phone from Microsoft (closed source, proprietary).

These operating systems and many other ones, represent the operating systems for our mobile phones, and by using the appropriate programming language for each operating system ,we enable to programming these phones, so we can build different applications and services we want from these mobiles.[13]

In our system Android is the operating system that we want to use in order to support our project objective requirements.

2.7.1.1 Android operating system:

In recent years Open Handset Alliance led by Google has deployed a neat, versatile, powerful, and elegant platform which was android. This platform spread widely and its market quickly growing, because it is supported by large set of hardware, software, and network carriers.

“Android is an open source platform and it is released under open source license. The Android operating system software stack consists of Java applications running on a Java based object oriented application framework on top of Java core libraries running on a Dalvik virtual machine featuring JIT compilation. Libraries written in C include the surface manager, Open Core media framework, SQLite relational database management system, OpenGL ES 2.0 3D graphics API, WebKit layout engine, SGL graphics engine, SSL.”^[13]

2.7.1.2 Android architecture

The architecture of Android operating system is described as following diagram with briefly explanation:

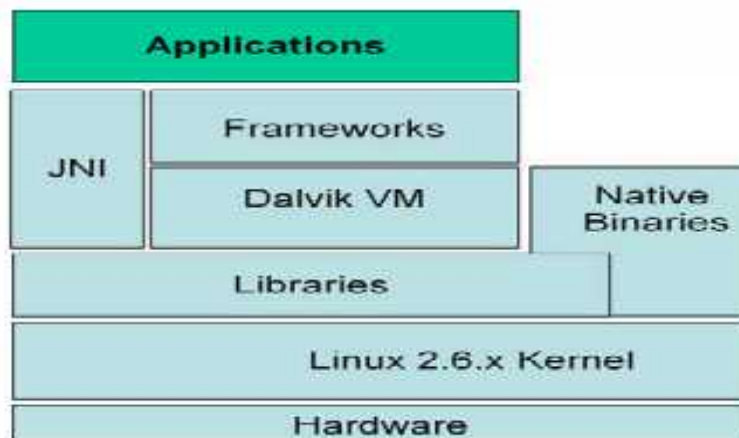


Figure 2.5 architecture of Android OS^[14]

- **Application** java is the programming language for most of applications written with android , using Android SDK.

These application when deployed is being a single file in a format (.apk). which is a modified Java Archive (JAR) file. this file contains the Java classes in a custom format (.dex) which make up the application, as well as an application manifest.

- **Application Frameworks** are also written in Java, and are based on the low level core libraries -which provide the basic subset of Java – java.io.*, java.util.*, etc.

And there is many level of manage and control the application, such as activity, window, package, telephony, and many other managers, that control the application you build.

- **Dalvik VM** Dalvik virtual machine which has been adapted to the specifics of mobile systems with limited CPU capabilities, low RAM and disk space, and limited battery life.

Dalvik is a register-based virtual machine, and it is neither fully J2SE nor J2ME compatible. Instead it works with its own version of Java Byte Code, pre-processing its input by using a utility called “dx” which produce “.dex” files from the corresponding Java “.class” files.

- **Native Binaries** this is a standard Linux binaries of ELF formatted, they can be created with the Android Native Development Kit after being coded in C or C++ usually, and then compiled directly to the target processor (usually, ARM).
- **JNI** Java Native Interface which used to achieve native-level functionality, mainly by enables a Java application to directly invoke a non-Java function. Which make the runtime environment is so rich. And a Good reasons to use JNI are, efficiency, and obfuscation.
- **Libraries:** when the Dalvik VM wants to execute an operation . it will call for services from these libraries which are a collection of many libraries, all open source, and implement the low level functionality provided by the runtime.

Android support advanced point to these libraries which is preloaded into memory, which allows for faster load times, and doesn't waste any memory.

- **Linux** : Android uses the open source Linux Kernel as its own, the kernel is similar, but not identical, to the standard Linux kernel distribution.[14]

2.7.1.3 Android features:

There are many features supported by android operating system, and here we mention some of them which give a helpful point for our system objectives:

- Android supports connectivity technologies including GSM, Bluetooth, and many other, but in our project mainly we need the GSM support, and we may need the Bluetooth.
- Android operating system is powerful OS that growing worldwide and widely supported by large number of hardware, software and network, so it will be useful in our project, and available to large number of customers.
- Android support some forms of messaging, like SMS and MMS, and in our project it may be needed for communication.
- While most Android applications are written in Java, there is no Java Virtual Machine in the platform and Java byte code is not executed. Java classes are compiled into Dalvik executable and run on Dalvik, a specialized virtual machine designed specifically for Android and optimized for battery-powered mobile devices with limited memory and CPU. J2ME support can be provided via third-party applications.[14]

2.7.2 Mobile programming language :

In our project the phones will being programmed using java programming language, where it is an important languages, that is created by James Gosling from Sun Microsystems in 1991, and nowadays owned by Oracle, it's one of a very popular, powerful, general-purpose, concurrent, class-based, and object-oriented language.

It uses to support many important applications and one of these are Android applications, which relies heavily on java fundamentals.

The Android SDK includes many standard Java libraries such as data structure libraries, math libraries, graphics libraries, and many other ones as you want.

2.7.2.1 Java architecture:

We will mention the main architecture for java programming language, where Java programming language consists mainly of four components; Java compiler, Java virtual machine, java run time environment, and the Java class libraries. and the relation between these three components can be described in brief during program execution as follow :

- You write your source code with the programming instructions.
- When you run your code it will be compiled using the language compiler that translates it from the source to Java class files.
- Run the class files on a Java virtual machine, where JVM is written specifically for a specific operating system.
- Java class libraries: that enables you to access system resources; by calling methods in the classes that implement the Java Application Programming Interface. As your program runs, it fulfills your program's Java API calls by invoking methods in class files that implement the Java API.
- Java runtime environment (JRE) that consists of the JVM and the Java class libraries.[15]

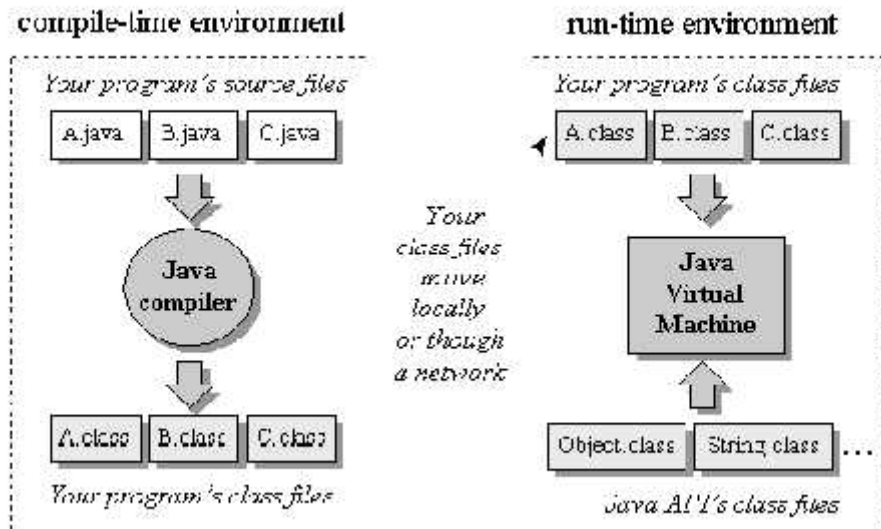


Figure 2.6 java main architecture[15]

2.7.2.2 Java development environment (E_clips):

Eclipse will be the development environment for our mobile phone software, where it is an integrated development environment (IDE) for Java. and also a multi-language software development environment comprising an integrated development environment (IDE) and an extensible plug-in system.

And nowadays you can build your applications on mobile, and one of your best choices is to get started on mobile Java development using Eclipse, where to build any application on mobile you will need three main software, first will be sun's java, then Eclipse, and Eclipse ME plug-in and the Sun Java Wireless Toolkit also needed.[15]

2.7.2.3 Java features:

- **Platform Independence:** Java compilers produce byte-code instructions for the Java Virtual Machine (JVM). and use the JVM as abstraction and do not access the operating system directly. so same compiled byte code can run unmodified on several platforms.

- **Objects Oriented:** everything in a Java program is an object and everything is descended from a root object class. except the primitive data types.
- **Strongly-typed programming language:** Java is strongly-typed, where the types of the used variables must be pre-defined and conversion to other objects is relatively strict
- **Familiar C++-like Syntax:** there is similarity between Java syntax and that of the popular C++ programming language. which enabling the rapid adoption of Java.
- **Garbage Collection:** this feature make your programming easier to write and less prone to memory errors, make memory use efficient and reduce the losses and unused space. where Java does not require programmers to explicitly free dynamically allocated memory.[15]

2.7.3 Mobile phone in our project:

We will use the mobile phone, as mainly device in our project; in order to transmit the GPS coordinates for each car, to another mobile adjacent to the main server, to finally reach the server.

2.8 C# Programming language:

The advancement of programming tools and consumer-electronic devices like, cell phones and PDAs created problems and new requirements. Also the integration of software components from various languages proved difficult, and installation problems were common because new versions of shared components were incompatible with old software.

Developers also discovered they needed Web-based applications that could be accessed and used via the internet. And also the need for software that was accessible to anyone and available via almost any type of device. and to address these needs, in 2000, Microsoft announced the C# programming language.

C# developed at Microsoft by a team led by Andres Hejlsberg and Scott Wiltamuth, was designed specifically for the .NET platform as a language that would enable programmers to migrate easily to .NET.

C# has roots in C, C++, Java, adapting the best features of each and adding new features of its own. Also it is object oriented and contains a powerful class library of prebuilt components, enabling programmers to develop applications quickly.

C# and .NET are set both to revolutionize the way that you write programs, and to make programming on Windows much easier than it has been, where c# provides a means for you to code up almost any type of software or component that you might need to write for the windows platform, not just a language for writing internet or network-aware applications.[16]

2.8.1 C# architecture:

C# (pronounced C-sharp) is a powerful programming language that support the power of C++ and simplicity of Visual Basic. And it is supposed to be the best language for Microsoft's .NET programming.

C# is an object-oriented language, and has support for component-oriented programming, and it has many features aid in the construction of robust and durable applications such as Garbage collection, exception handling, type-safe and many other features.

C# is significant mainly in two aspects:

First: it is designed specifically and targeted for use with Microsoft's .NET framework.

Second: it's a language that based on the modern object-oriented design methodology.

And here we will mention the main architecture components of c# programming language, where it's important to notice that c# is a language in its own right, and although it is designed to generate code that targets the .NET environment it is not itself part of .NET.

The main architecture can be described in :

- Common Language Runtime (CLR):
 - manages code execution
 - provides services
- Compiler

Always the source code you develop for any programming language, needs to be compiled before it can be executed by the CLR .

- C# compiler is part of .NET Framework SDK.

Where in .NET compilation occurs in two step first: compilation of source code to intermediate language(IL). Second: compilation of IL to platform-specific code by the CLR.

Microsoft intermediate language shares with java byte code the idea that it is a low level-language with a simple syntax, which can be very quickly translated into native machine code. Which also has significant advantages:

- Platform independence: mean that the same file containing byte code instruction can be placed on any platform, so by compiling to IL you obtain platform independence for .NET.
- Performance improvement: instead of compiling the entire application in one go, the JIT compiler simply compiles each portion of code as it is called, so when code has been compiled once, the resultant native executable is stored until the application exits so that it does not need to be recompiled the next time that portion of code is run.
- Language interoperability: the use of IL facilitates language interoperability, simply put you can compile to IL from one language, and this compiled code should then be interoperable with code that has been compiled to IL from another language.[16]

2.8.2 C# features:

C# is powerful programming language, and it has several features and advantages, here we mention some of general features that being helpful for any application or system you build, based on this language:

- Garbage collection relieves the programmer of the burden of manual memory management.
- Variables in C# are automatically initialized by the environment.

- Managed execution environment
- Variables are type-safe.
- Native support for the Component Object Model (COM) and Windows-based APIs.
- With C#, every object is automatically a COM object.
- Platform and language independent.
- Inside a specially marked code block, developers are allowed to use pointers and traditional C/C++ features such as manually managed memory and pointer arithmetic.
- Compiler allows use of initialised Variables only.
- Strong exception handling.
- Suited well for building Web Services.
- Array bounds checking.
- The language is intended for use in developing software components suitable for deployment in distributed environments.[16]

2.8.3 C# in our project:

This will be the programming language for the main server, where the interface to our system will be, deal with maps, process the coordinates, and display each car path on the map.

2.9 Software design blocks:

Here we design a block that shows the software parts, and the relation between them, in a clear way, for our system:

As we see in the figure (2.7) the mobile will be programmed with java, and the main server with c# and they will be connected through http client/server connection technique.

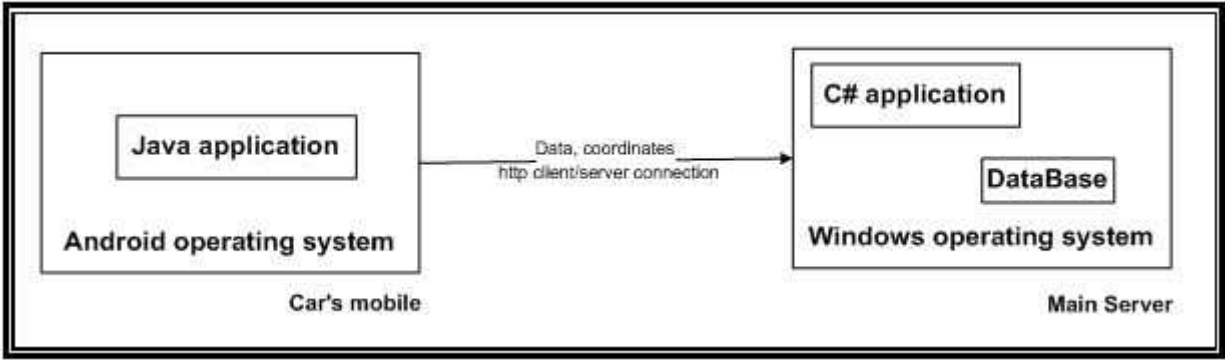


Figure 2.7 software design block

3

CHAPTER THREE

PROJECT CONCEPTUAL DESIGN

- 3.1 Overview**
- 3.2 General system block diagram**
- 3.3 System main components**
- 3.4 System flow charts**
- 3.5 Data flow diagram**
- 3.6 System functions**

3.1 Overview

In this chapter we will describe our system main parts and the design concepts in some details, we will talk about system general block diagram, the system main components and their related options, system main flow chart, system data flow diagram, and the system main functions.

3.2 General system block diagram:

As shown in figure(3.1)the general system block diagram; the main parts of our project are: GPS receiver, where data and locations coordinates will be created, GSM network which is the connection medium through which data will be transferred, and Central server where the data will be received and processed, and create complete system user interface.

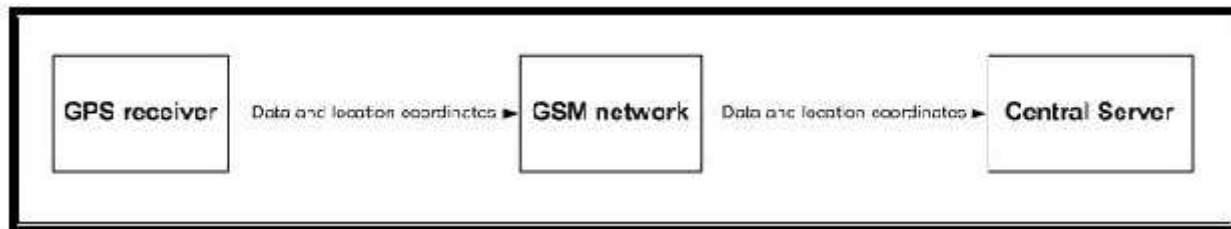


Figure 3.1: general system block diagram

3.3 System main components:

The main components of our system are the GPS Receiver, GSM network, and Central server, all of them in this section will be described in more details, and the options for these components will be also described as follow:

3.3.1 GPS Receiver:

The first component of our system is the GPS receiver, it's the main component in the tracked car that will receive the GPS coordinate from the GPS satellite. Many alternatives of GPS receivers are available to be used, and it can be one of the following types:

- **Type 1:** Separate GPS Receiver, come as a separate device, that can be one of the following:
 - Not-self-contained receivers (without screen), also known as RS232 receivers or also GPS mice. This type needs a computer (often a Pocket-PC or Palm PDA) and according program in order to visualize the actual position of the GPS receiver.
 - Self-contained receivers (with screen). Here the computer is integrated in the GPS receiver. This type sometimes has extra features as an electronic compass or even a radio transmitter/receiver.

This type of receiver will not be used, because of the need to many other components to perform the objectives of the system. Which mean combination of GPS receiver, Micro controller, GPRS modem, and a small screen may be needed. And accurate interfaces between them must be established, which increase the complexity and cost of the system.

- **Type 2:** Phones with incorporated GPS receiver.

This will provides a GPS receiver built inside the mobile phone to provide better outdoor GPS solution.

This type of receiver will be used in our system, due to its availability and widespread in our life. Besides that, it can be used alone without the need to other components such as modems, microcontrollers or screens.

Mobiles with GPS receiver available with different operating system. In our system, Android is the operating system that we want to use. [17]

3.3.1.1 Android OS

Google Android is an "open source software" and a competitive operating system, we choose it to be used between all other alternatives, because it is taking over other leading systems with many advantages, such as:

- Android Has a Better App Market, allows developers and programmers to develop apps (applications) in what is known as "application without borders".
- Updated user interface design.
- Support for extra-large screen sizes and resolutions.
- Improved power management and application control.
- “Google fully developed Android and make it into an Open Source. Now, any phone manufacturer can use Android without expensive license fee from Google. Because it is open, manufacturer can modify Android without restriction, allowing it to fit the device they are making total freedom. This makes it a big incentive for any device manufacturers to adopt Android. The ability to run tens of thousands of apps is another big incentive”.^[18]

6.3.1.2 Galaxy S

We choose galaxy S mobile phone to implement our system, it's one of many other mobiles phones that can be used, it has been chosen because it contains and supports all of the required technologies to perform the project objectives. It has the following features:

- Support Operating Frequency:
 - 2G Network: GSM 850 / 900 / 1800 / 1900.
 - 3G Network: HSDPA 900 / 1900 / 2100
- Operating System
 - Android OS, v2.1 (Eclair), upgradable to v2.3

- GPS : with A-GPS support
- Support for GSM voice communication, SMS, and MMS.
- Connectivity
 - Wi-Fi: 802.11 b/g/n
 - USB 2.0
 - Bluetooth 3.0 Connectivity
- Network
 - GPRS/EDGE

6.3.2 GSM network:

The second component of our system is the GSM network. It represents the service provider in the system, which we will perform the transmission of the coordinates from the cars to the central server, over GSM components.

The transmission procedure will be done through GPRS, so the used network must support GPRS, which adds a packet-based data capability to the GSM system, brings IP networking into the GSM world, and involves the addition of new packet routers into the GSM core network, producing a core network with separate circuit- and packet switched domains.

6.3.2.1 Why GPRS ?

- **Cost:** Communication via GPRS is cheaper than through the regular GSM network. Customers only pay for the amount of data transported, and not for the duration of the Internet connection. Also it's cheaper than transmitting messages in SMS (short message service). So, it's more suitable to our system in order to send coordinates continuously with acceptable cost.
- **Constant Connection:** Through GPRS technology, users are constantly connected to the Internet. As GPRS services are available wherever there is GSM coverage.
- **Mobility:** GPRS provides wireless access to the Internet from any location where there is a network signal. This enables you to surf the Internet on your laptop or phone, even in remote areas. That's what we need to track the moving cars.
- **Speed:** Although new, faster technology exists today, GPRS is still faster than the older WAP (Wireless Application Protocol) and regular GSM services. GPRS data is transferred at speeds ranging from 9.6 kilobytes per second up to 114kbps.
- **Simultaneous Use:** When you access the Internet through GPRS, it does not block incoming calls through the GSM network. This enables you to make or receive voice calls while you are using our application.

6.3.2.2 Connection Options

To implement a GPRS connection, many connection options are available. Here we will mention it and choose the most proper one:

- **GPRS mobile with PC interface.**

At this option, the user data and location coordinates will be sent from the first mobile inside the tracked car to next mobile beside the main server. So an interface is needed between the adjacent mobile and the PC, this interface can be:

- **Bluetooth connection (wireless interface):** which is efficient to be used here, because we want to transmit data over short range distance, (between GPRS mobile and the central server). Transmit and receive data between mobile and central server, as Bluetooth signals.

- **USB cable (wired interface):** A data cable is a long cord, which connects computer with the mobile in order to share data between them. The data cable transfers data on the form of electrical signals from mobile to central server and vice versa.

- **HTTP Connection**

At this option we will consider the mobile phone at each car as HTTP client, that has an application make a post directly to the PC, which will be considered as a HTTP server. The data will be posted directly as a parameter to the functions built on the server.

In our system we will use the second option, HTTP connection, because it more direct and practical, and it cancel the need of the PC mobile adjacent and the interface between them.

6.3.3 Central server:

The last main component of our system, will be the central server, where the data and location coordinates that transmitted by the mobile will be received, processed, plot cars movement as continuous paths on the geographic map, and provide the user with clear

system interface, and provide them with other available services like cars history and customers related information.

The server can be programmed using one of the common programming languages, like C#, Java, Visual Basic, and many other choices, that all has its own features, and in our system it will be programmed with C# language, by creating a complete program that process the received cars coordinates, then display it then update the paths each time the coordinates being received.

6.3.3.1 Geographic maps :

There are many sources where our geographic maps can be taken, to create our system tracked geographic area, and some of these sources are:

- **Google earth :**

Google earth is a map and geographic information software that Puts the whole world in your hands, this product released in 2005, and become nowadays available on all personal computers and mobiles, you only need to download free version of this software, then you can use it easily, which provide you with ,word maps and geographical information about any region on the earth, these maps and photos was taken via satellite imagery, aerial photography, and GIS 3D globe.[19]

- **Pal earth :**

Pal Earth is the first website to provide electronic Palestinian specialized geographic information services through the Internet, That provide us with infrastructure and frameworks for the provision of geographic information systems and services, PAL aims Earth through this effort to play a role in raising the level of use of geographic information systems and applications for individuals and institutions alike, and that by providing the possibility of using these systems and their applications at a cost commensurate with the goals to be achieved.[20]

- **Yahoo maps :**

Yahoo maps is a free online mapping portal provided by Yahoo, The street network and other vector data Yahoo maps uses is from Navteq Tele Atlas, and public domain sources. Detailed street network data is currently available for the United States, Canada, Puerto Rico, the Virgin Islands, and most European countries. while Country borders, cities, and water bodies are mapped for the rest of the world, moderate resolution satellite imagery is available worldwide. Where one to two meter resolution is available for most of the contiguous United States, and select cities worldwide.[21]

In our system, we will use PAL earth as reference for our regions map, because it give us the best resolution and appearance for Palestine cities, streets, and high locations details, more than what Google earth or yahoo maps provide, because it focus especially on Palestine region, and try to process it in the best vision.

6.4 System flow chart:

To achieve the objectives of our system, many steps will be done, and these can be described in general, as shown in the system main flow chart in figure (3.2) where these main steps are :

1. Fill the required registration data by customer, and determine the car's location, using GPS Technology.
2. Develop mobile application, First to send registration information, and then get the active car's location, and send them periodically, via GPRS.
3. Data send via mobile will be received by the main server, and the whole system interface will be created for server side's user.
4. Data that received will be processed, register the new cars, and display their locations, as continuous path of movements on map.

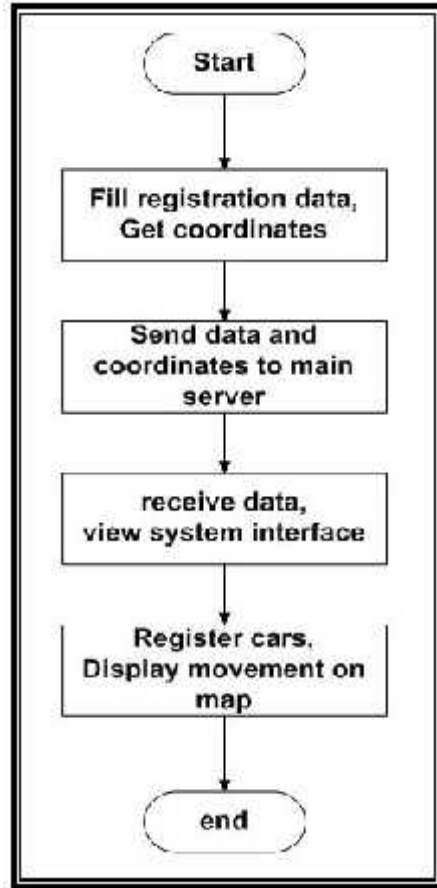


Figure 3.2: system flow chart

Each one of these steps , will be described in detail as follow:

6.4.1 Fill registration data and get coordinates :

6.4.1.1 Fill registration data

The first step in our system is to make a registration for a new car, this is done by letting the user to enter some required information, in order to be sent to the main server, as a request for joining the whole system. These data sent only one time before the start of sending coordinates and speed.

6.4.1.2 Get coordinates and speed

After registration completed, the next step is to determine the accurate – real time – GPS coordinates and speed of the tracked car. The used mechanism of getting these coordinates is described as shown in figure (3.3) get coordinates flow chart:

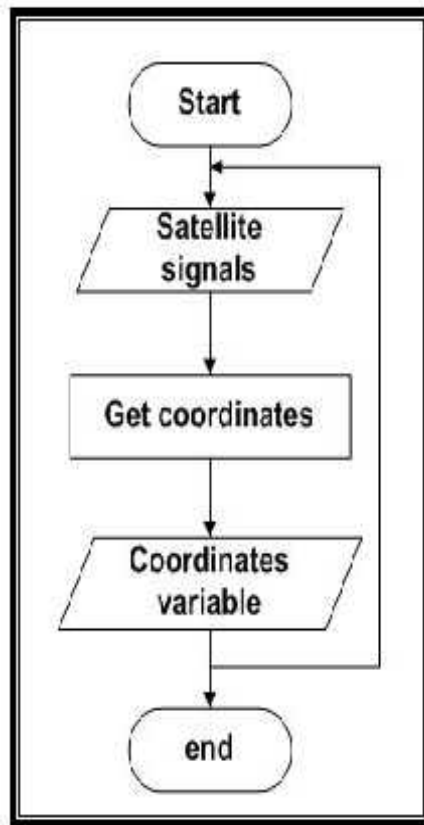


Figure 3.3: get coordinates flow chart

- The received signal from the GPS satellite are processed, at the mobile's built-in receiver, to determine the longitude and latitude.
- A code will be added to the mobile, to provide access to the built-in receiver, and store the longitude and latitude parameters, at the defined coordinates variables.
- The speed of the car is determined by the GPS system relative to the rounding satellites, received also by the GPS receiver and inserted into speed variable updated with every new coordinates.
- Now, longitude and latitude of the current location and speed are available to be sent in the next step.

6.4.2 Send data and coordinates to main server :

At this step, the registration information and coordinates variables will be transmitted using the described mechanism as shown in figure (3.4) send data and coordinates flow chart:

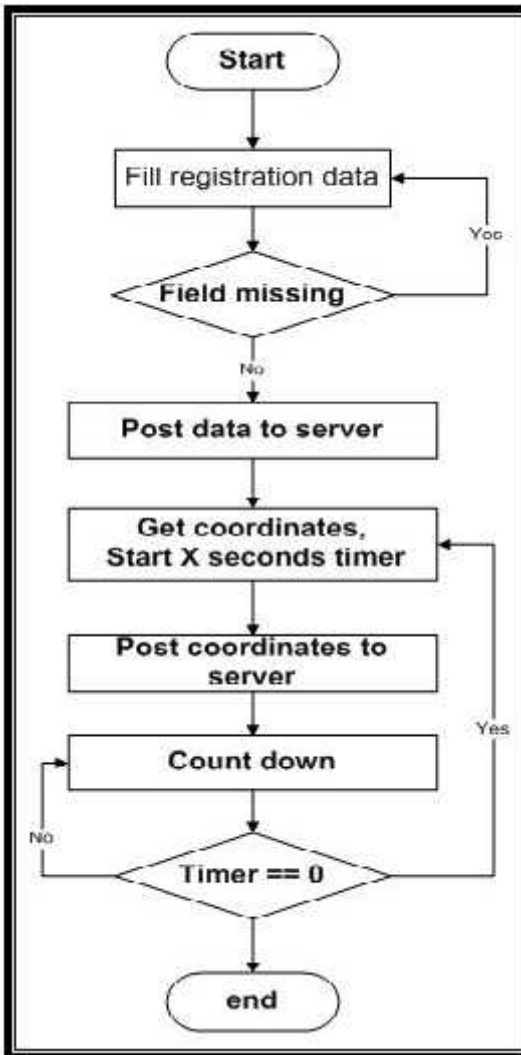


Figure 3.4: Send data and coordinates flow chart

- As shown in the figure, as well as the required registration information is available; they will be transferred immediately to the main server as a posted data.
- The success of post process is a sign that the system accept this user and it can continue to send its coordinates and speed.

- The received coordinates and the calculated speed which already stored at the defined variables will be prepared to be posted to the main server frequently.
- A countdown timer will be started -at the same time when the coordinates ready to be posted and will be checked continuously.
- After performing the post process, the registration information and location coordinates and speed, are now at the server side, and ready to be processed and plotted.
- The received messages will be stored at coordinate's data base on the main server.
- The system continuously checks the timer, and has to wait until it reaches zero. When that happened, it repeats the whole process, with new coordinates, of the new car's location.

6.4.3 Receive data and view system interface :

In the other side of our system, where the server need to continue the system work, there are main steps, start with connection side that is related to the mobile posted data, these data will be received by the server to process it, and full clear system interface will be created to the system users, to provide them with all available services that our system support.

The connection server side is described as shown in figure (3.5) server received data and coordinates flow chart:

Where the main parts of the connection process are :

- First activate the ability to receive new cars, that want to connect to our system.

- Then for each new car try to connecting, we will give the server system side's user the ability to accept new cars or reject them.

- Then when the car accepted it will be able to start its coordinates and speed posting continuously to main server, which receives these coordinates for any tracked car, and at the same time still waiting for any new connection and sending of registration data.

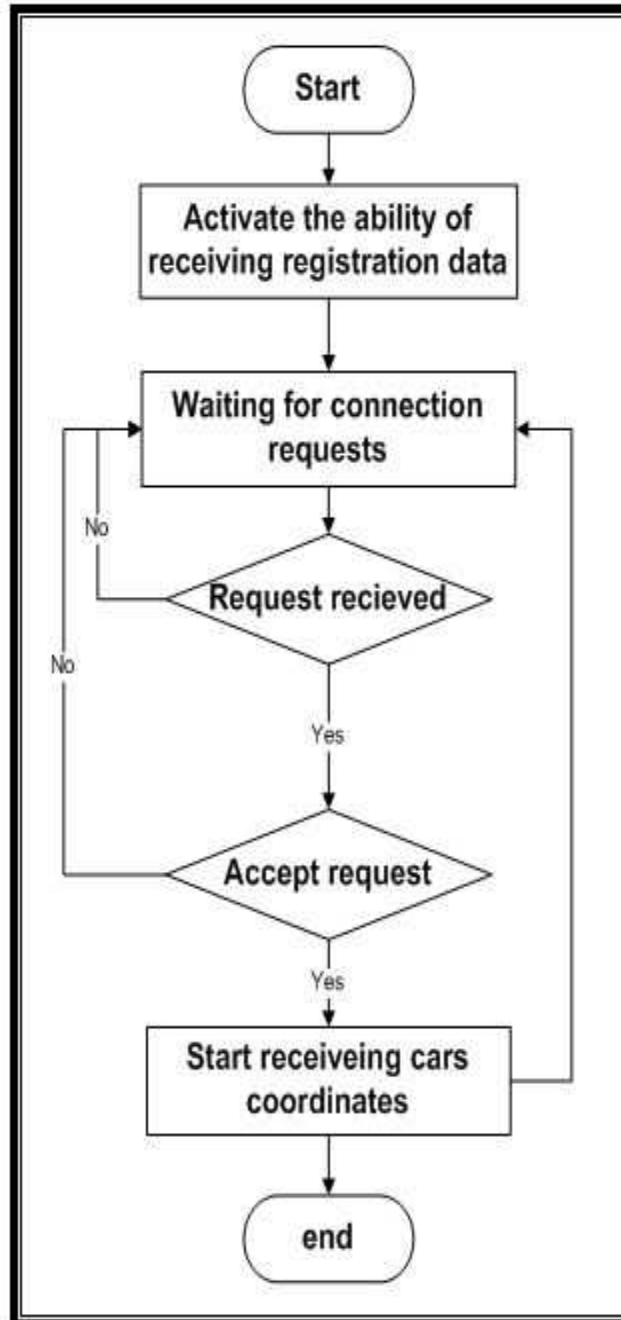


Figure 3.5: server received data, and coordinates

6.4.4 Register cars, and display movement on map :

Each time registration data or location coordinates and speed will be posted to the main server, and after received by the connection server side, it will be processed before add to the system as shown in the figure 3.6 received data processing flow chart:

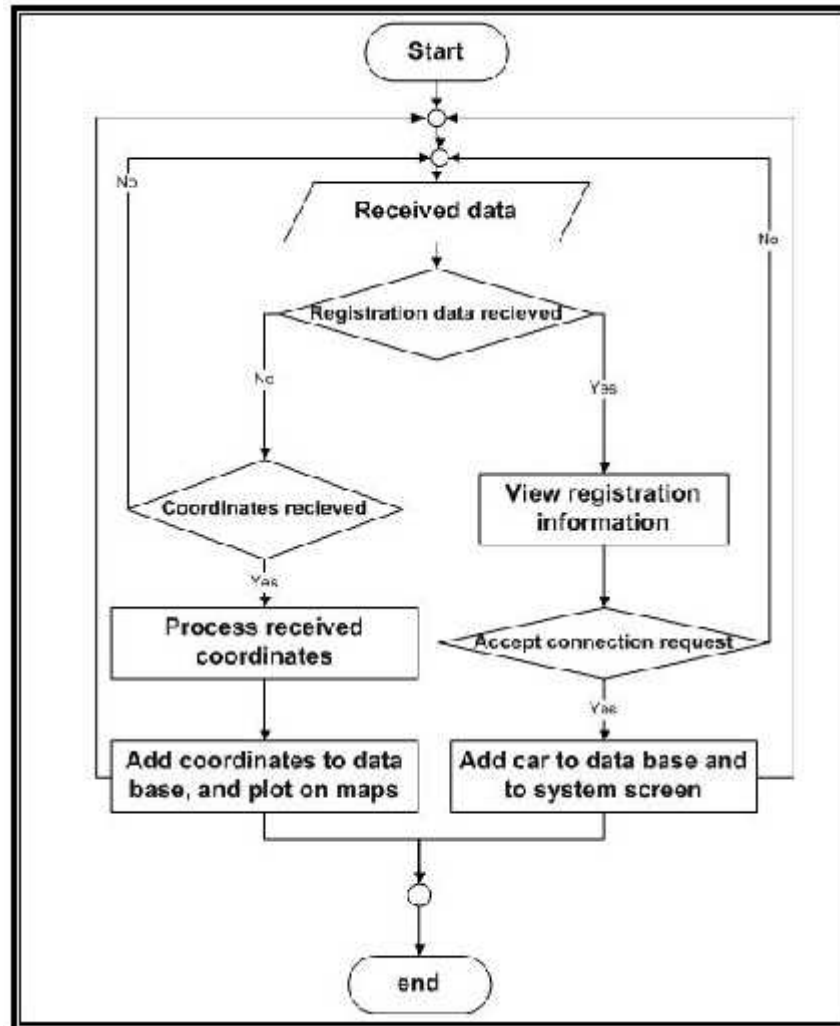


Figure 3.6: received data processing

As shown the received data process will be done in main steps as:

- Each time the data received by the server, it will be tested to see whether new car try connecting to your system or coordinates for one of active cars being received.
- When registration data received, the user will be able to see these information and accept the registration, so the car will be inserted to the system and add to data base , also he can reject the car that try connecting without add to the system.

- When coordinates being received, it will be processed in some way, according to the type of system coordinates being used, and then add to data base, and plot on the map. Also with each receiving coordinates speed will processed to be shown on the screen and stored at the database.
- Each time we receive data we still have the ability to receive new cars registration data and new locations coordinates.

6.5 Data Flow Diagram:

The following figure (3.7) Data flow diagram, describe how data will be flow in our system, from the GPS system to our mobile application, then through GSM network to the main server:

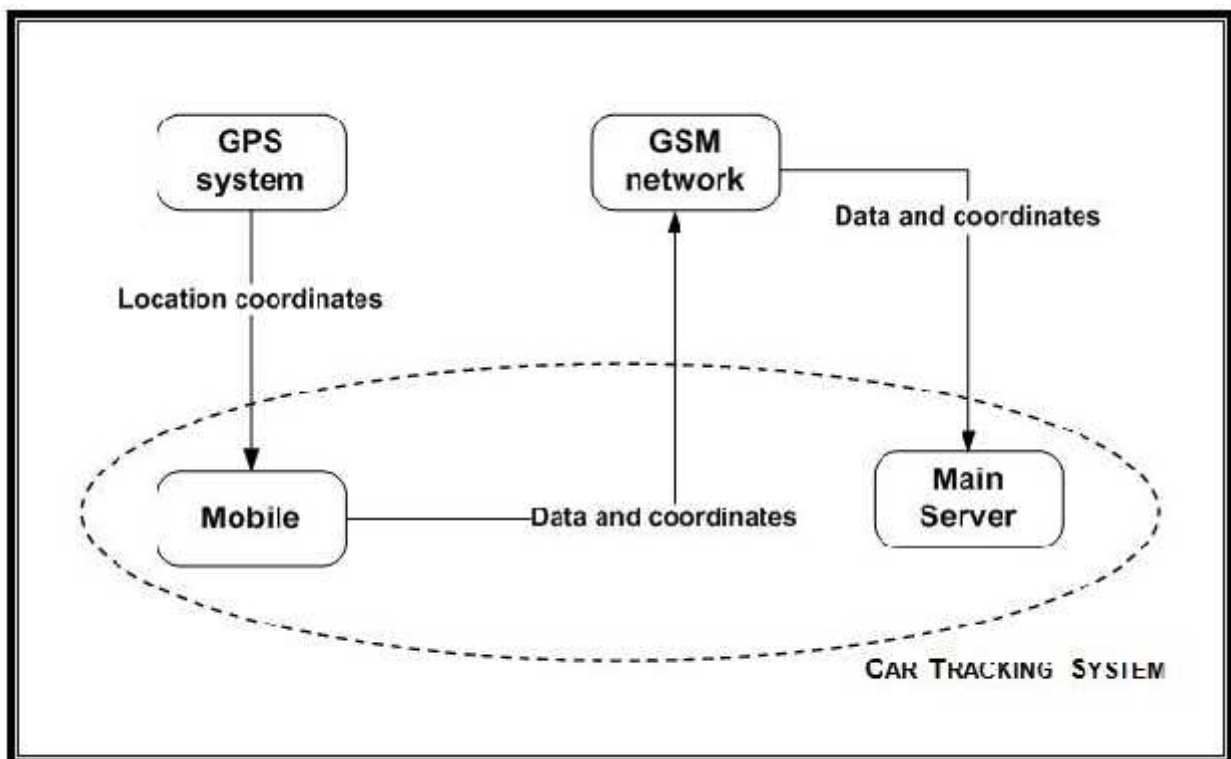


Figure 3.7: Data Flow Diagram

6.6 System Functions :

Here we mention more details about main parts in our project, and the main functions that will be implemented.

6.6.1 Functional block diagram:

In order to perform the main parts, many functions have to be done, the following figure describe these functions:

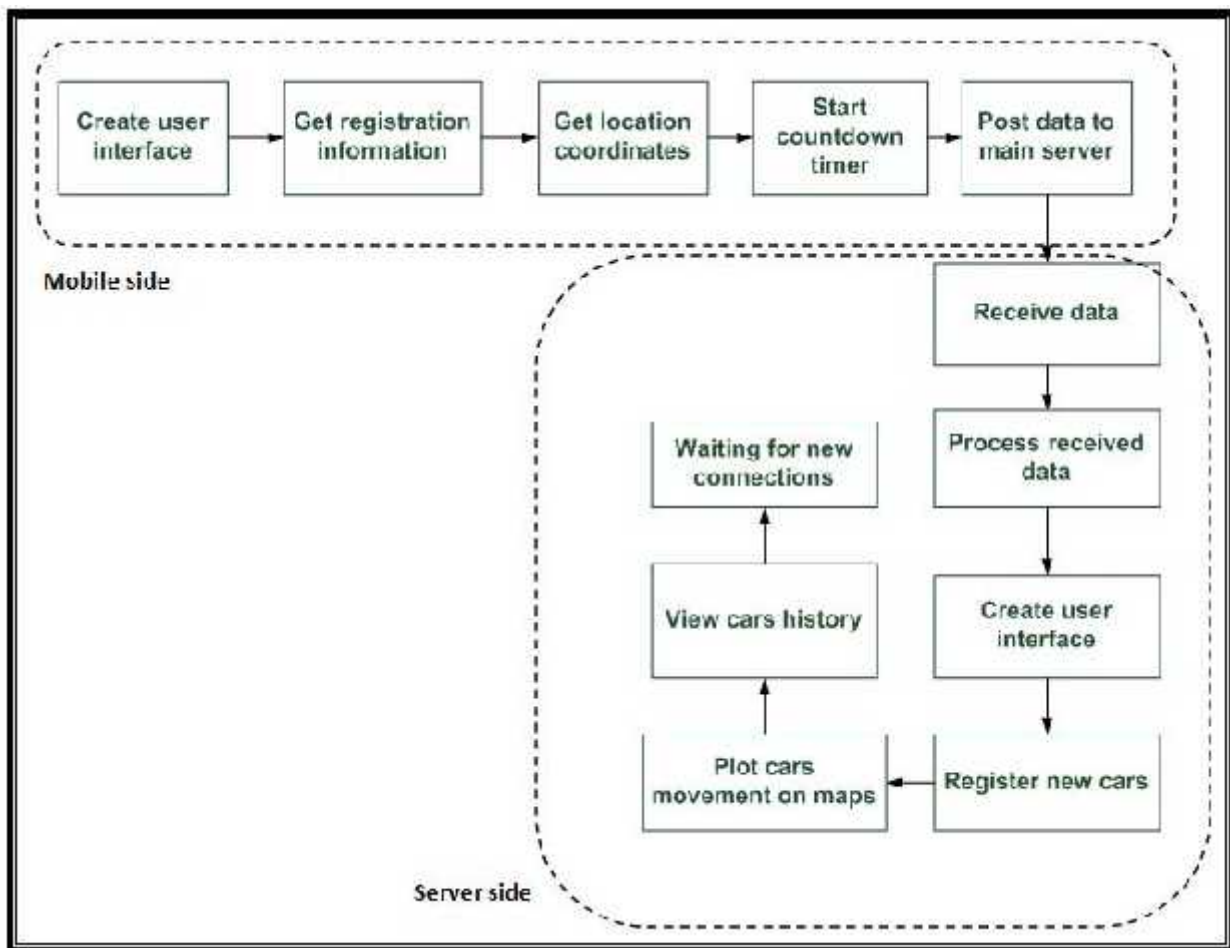


Figure 3.8: Functional Block Diagram

6.6.2 Mobile Functions:

There is a specific mechanism the mobile follows to determine the location of the tracked car specifically, and then transmit it. This mechanism consists of the following functions:

- **Create user interface:** We will built an application to be installed on the mobile; this application must be usable and easy for the users to interact with it.
- **Get registration information:** For any user, in order to register in the system; some information are necessary to be supplied, this information consist of: Car number, Mobile number, and Car driver.
- **Get coordinates:** This is done by activating the GPS receiver, to enable it to receive the coordinates, of the tracked car, from the GPS satellites. Then, read the stored coordinates, in the GPS receiver of the mobile.
- **Get speed:** This function also required the GPS to be activated in order to enable the GPS receiver to calculate the speed.
- **Start a countdown timer:** The determination of the car's coordinates will be done every 'X' seconds. So, we need a timer to determine the moment at which the coordinates must be read. The timer will be initiated by a value equals the 'X', then reduced and checked continuously.
- **Post data to main server:** after getting the data it must be posted to the main server directly by the http connection. A POST request sends additional data to the web server, specified after the URL, A header will be added, followed by a blank line to indicate the end of the headers, then the sent data. An example:

POST /login.jsp HTTP/1.1
Host: www.mysite.com
User-Agent: Mozilla/4.0
Content-Length: 27
Content-Type: application/x-www-form-urlencoded

" posted data".

6.6.3 Main server functions:

There is a specific mechanism the main server follows to continue the system build . And this mechanism consists of the following main functions:

- **Receive data:** each data or coordinates that post from the mobile will be received on the server side, through the http connection mechanism, which will be the server side of http connection where the client post the data.
- **Process received data:** the data that are received will be processed in some way, even it is new car registration data or location coordinates send by one of current active cars.
- **Create user interface:** to make our system usable and easy for the system server side user to interact with; a clear interface must be created, and all services that available by the system will be set. .
- **Register new cars:** any new car try connecting to the system, will be determined to be accepted or not by the system user, and when accepted it will be add to the system database, and to system active tracked cars.
- **Plot cars movement on map:** each time new coordinates are received, it will be processed to be represented accurately on tracked geographic map in our system, then update the current tracking car's path for each car

separately . So at any time of tracking you can see a continuous path of movements for cars on the map.

- **View cars history:** the system must be able to save, any cars that connected to it , store its registration data, and store the history for the cars over any time they previously connected to the system, and clear way to retrieve this history .
- **Waiting for new connection:** during all process of the system, it must still able to receive any new car that try to connect to the system, and receiving all active cars coordinates, process them, and plot on the map, without any reflection in the processes sequence or connections.

4

CHAPTER FOUR

DETAILED DESIGN

- 4.1 Overview**
- 4.2 System classes model diagram**
- 4.3 System use cases**
- 4.4 System sequence diagram**

4.1 Overview

In this chapter, the software engineering details will be mentioned and described, including UML model diagram for both mobile and server application software classes, with implementation for these classes, then the system use case diagram will be described, also the sequence diagram for use cases will be mentioned.

4.2 System Classes Model Diagram:

4.2.1 Mobile Software UML diagram:

A detailed description for our Mobile Application will be implemented using UML model diagram; that will describe the main classes, the attributes used in each class, methods that were defined, and a detailed description of these methods.

The mobile application software are described as shown in figure (4.1).

4.2.1.1 Classes Implementation:

Our system Mobile application software contains a group of classes that interact with each other, to make the whole system work properly, and complete the system objectives, and these classes are:

- **Hello class:**

This class will be used when starting the application, as greeting screen, it consist mainly of two parts:

- **Variables:**

mpstart: it's media player object plays music at the beginning of the application.

logtimer: a thread contains the run and sleep method.

timer: an integer contains the number of second represent the time interval for showing the hello class and playing its music.

RegisterActivity: Intent contains the new activity that will be shown when the welcome activity finished.

- **Methods:**

onCreate: Called when the activity is starting, this is where most initialization should go.

run: this is runnable thread method, that will be applied when the form being loaded, to start get the timer its interval value, and then start its down counting.

startActivity: Launch a new activity, this implementation overrides the current activity, clear the screen and start the next activity.

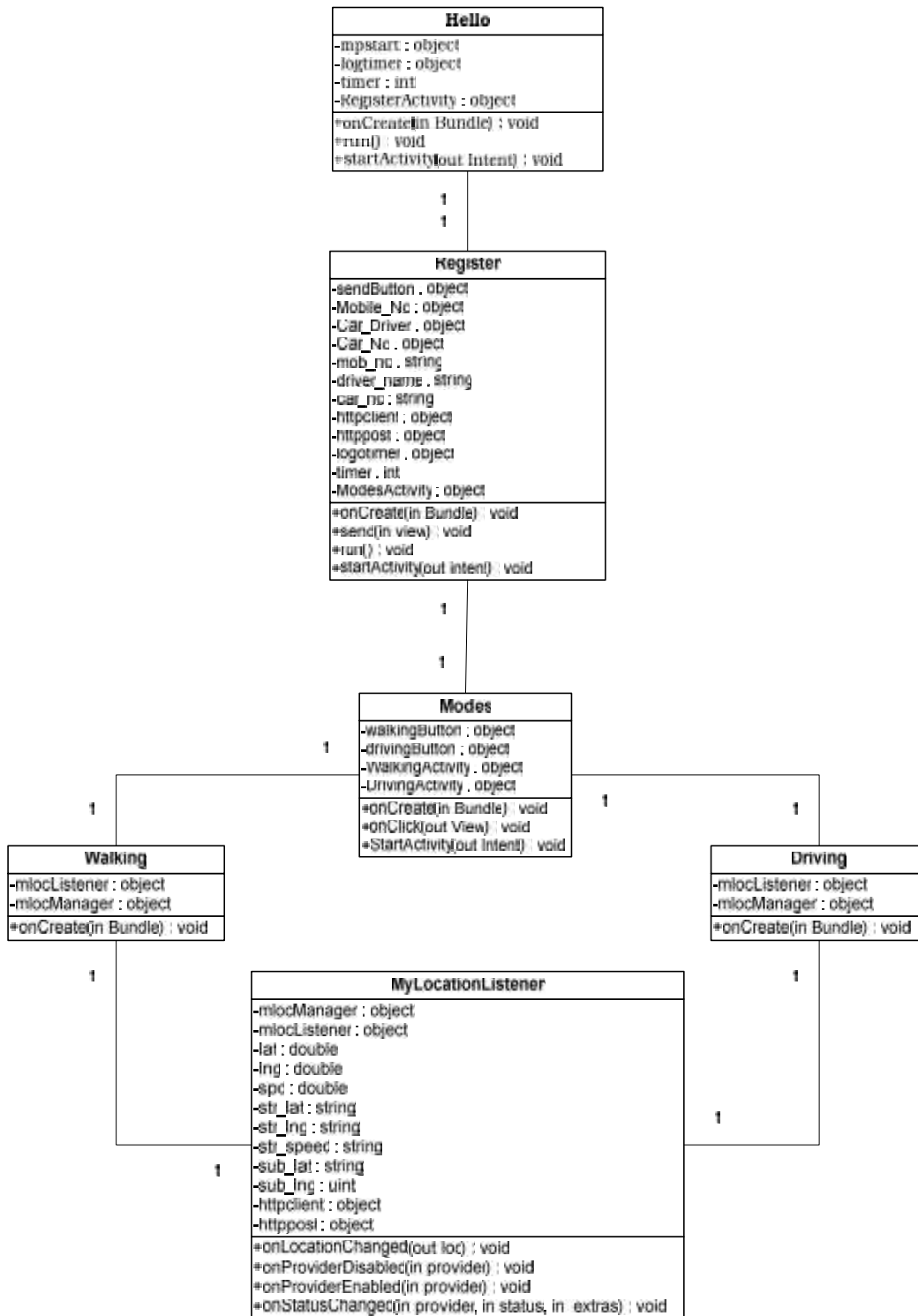


Figure 4.1: Mobile UML model

- **Register class:**

This class will be used to allow the user to enter it's information and register to system

- Variables:

Car_Driver, Car_No, mobile_No: An editable text view fields that will be filled by the user to hold the name of the driver, the car number and the mobile number of the user.

mob_no,car_no, driver_name: sting variables to hold the entered information by the user on the previously mentioned edit text.

sendButton: Push-buttons can be pressed, or clicked, by the user to perform an action, the action here is to post user information to main server.

httpClient: Interface for an HTTP client required to execute HTTP requests while handling cookies, authentication, connection management, and other features.

httppost: declare a POST method to request that the origin server accept the entity enclosed in the request as a new subordinate of the resource identified by the Request-URI in the Request-Line. POST is designed to allow posting the string variables entered by the user.

ModesActivity: Intent contains the new activity that will be shown when the Register activity finished.

- Methods:

onCreate: Called when the activity is starting, this is where the initialization for the text fields should go.

send: this method that gets called when you click the sendButton to get the message from the message text box, make sure the fields are not empty, prepare the data to be posted and finally perform the post process.

run: this is runnable thread method, that will be applied to make delay between the two activities.

startActivity: Launch a new activity, this implementation overrides the current activity, and moved to the next activity.

- **Modes class:**

This class will be used after completing the registration process in order to select the mode of travelling. It consists of:

- Variables:

walkingButton: a button when you click on it, open the walking mode activity.

drivingButton: a button when you click on it, open the driving mode activity.

WalkingActivity: The activity that will be opened by clicking the first button.

DrivingActivity: The activity that will be opened by clicking the second button.

➤ Methods:

onCreate: Called when the activity is starting, for making initialization for the buttons.

onClick: This is runnable thread method, Called when a view has been clicked.

startActivity: Launch a new activity, this implementation overrides the current activity, clear the screen and start the next activity, walking or driving depends on the clicked button.

- **Walking class:**

This class will be used to get the location's coordinates and speed for a walking user and post them to the server.

➤ Variables:

mlocManager: a location manager object needed to obtain GPS location.

mlocListener: Define a listener that responds to location updates.

➤ Methods:

onCreate: Called when the activity is starting, initialization for mlocManager and mlocListener is done, and calling the MyLocationListener class.

- **Driving class:**

This class will be used to get the location's coordinates and speed for a moving car and post them to the server. The variables and methods in this class is the same as the walking class with different assigned values.

- **MyLocationListener class:**

This class will be used to register the listener with the Location Manager to receive location updates.

➤ Variables:

Lat, lng, spd: double variable change continuously with the location, Returns the latitude and longitude of the location and the speed of travelling.

str_lat, str_lng, str_speed: string variables results from converting the coordinates and speed from double type into string type.

sub_lat, sub_lng: sting variables returns a string containing a subsequence of characters from str_lat, str_lng strings.

sendButton: Push-buttons can be pressed, or clicked, by the user to perform an action, the action here is to post user information to main server.

httpClient: Interface for an HTTP client required to execute HTTP requests while handling cookies, authentication, connection management, and other features.

httppost: declare a POST method to request that the origin server accept the entity enclosed in the request as a new subordinate of the resource identified by the Request-URI in the Request-Line. POST is designed to allow posting coordinates obtained by the application.

➤ Methods:

onLocationChanged: Called when the location has changed, contains the coordinates data and the posting process and the timer for posting. Parameter, the new location, as a Location object.

onProviderDisabled: Called when the provider is disabled by the user. If requestLocationUpdates is called on an already disabled provider, this method is called immediately. Its parameter, the name of the location provider associated with this update.

onProviderEnabled: Called when the provider is enabled by the user. Its parameter, the name of the location provider associated with this update.

onStatusChanged: Called when the provider status changes. This method is called when a provider is unable to fetch a location or if the provider has recently become available after a period of unavailability.

4.2.2 Desktop Software UML diagram:

A detailed description for our desktop software will be implemented using UML model diagram; that will describe our main classes in the built system, the attributes used in these classes, methods that were defined, and a detailed description of these methods and our system classes will interact with each other during system working, and also some points that needed to be clarified will be mentioned in this section.

Our system desktop software are described as shown in figure (4.2):

4.2.2.1 Classes Implementation:

Our system desktop software built a group of classes that interact with each other, to make the whole system work properly, and complete the system objectives, and these classes are:

- **FrmWelcome class:**

This class will be used at the beginning of your program, as greeting screen to our system, that consist mainly of two parts:

- **Variables:**

timer: that will be used as counter to display the welcome screen for short time.

LogIn_Form: that will be used to hold object from the following class that will appear after close welcome screen, which is of type FrmLogIn class.

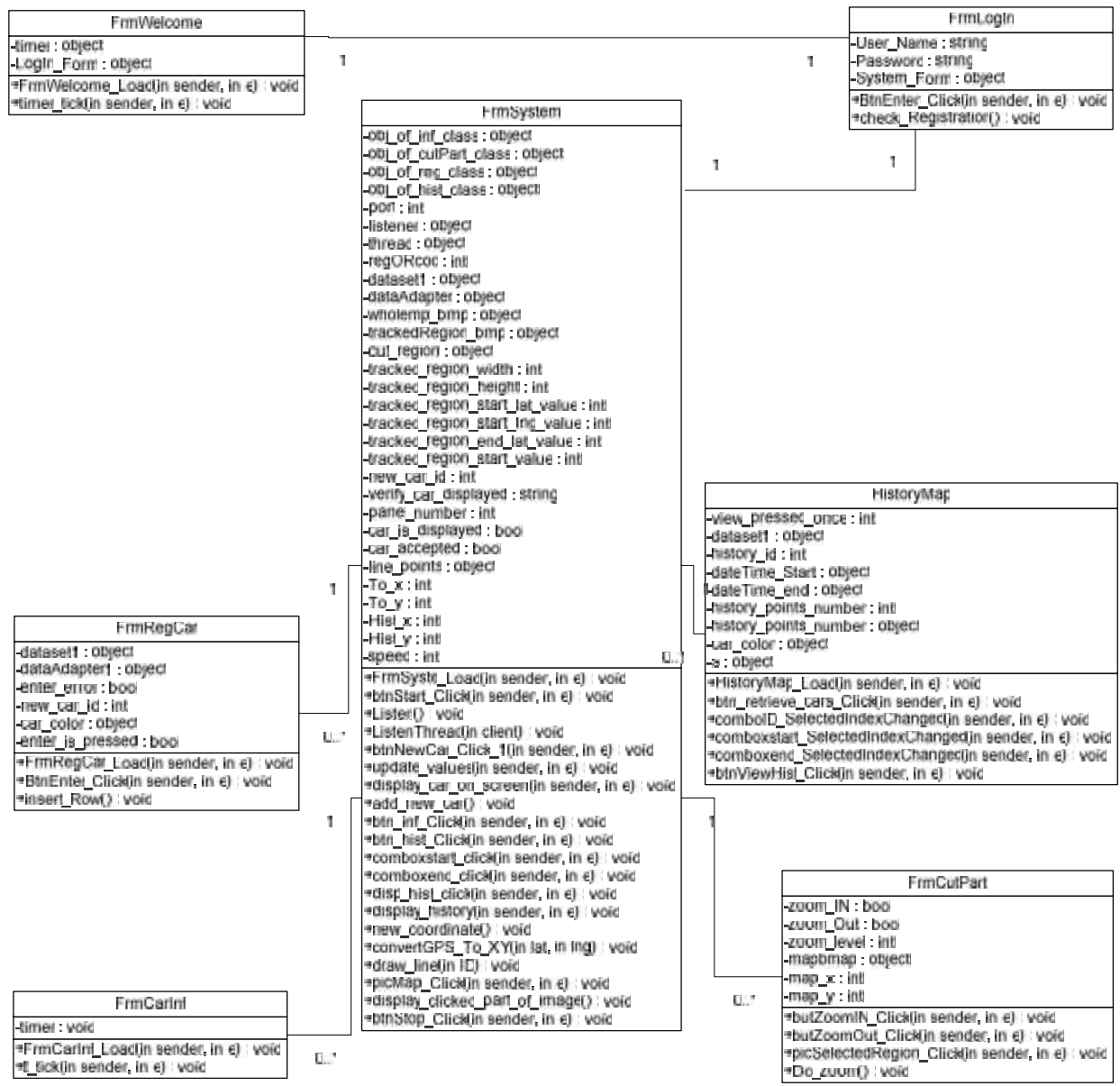


Figure 4.2 Desktop UML model Diagram



Methods:

FrmWelcome_Load: this is an event handler method, that will be applied when the form being loaded, to get the timer its interval value, and then start its down counting.

timer_tick: this is an event handler method, that will be applied when the timer stop, to hide current screen and open the next Log In screen.

- **FrmLogin class:**

This class will be used to allow the user to log in to the system by insert correct Username and password according to the code built in values:

- Variables:

User_Name: the user name variable that will be used during registration, it's value given to be "PPU".

Password: the password variable that will be used during registration, it's value given to be "12345".

System_Form: that will be used to hold object from the following class that will appear after insert correct username and password, which is of type FrmSystem class.

- Methods:

BtnEnter_Click: this is an event handler method, that will be applied when the enter button in the log in screen clicked, to call the check registration method.

check_Registration: this is a method with void return value, and without parameters, it used to open the next screen of the system form class, only when you enter correct user name and password, if not you will stay in the log in screen to try another times.

- **FrmSystem class:**

This is the main class in the system, where most of classes interact with, and a lot of operations will be performed in this class mainly the draw region of cars movement, and dynamic region for tracked cars and their related information and history ,it created directly after you log in correctly to the system.

This class contains a large number of variables and methods, the most important ones will be mentioned, and other ones will be grouped according to their usage in the code and the main ones will mentioned while leaving the other with less importance, and other ones will be left when we need to talk about it in other places.

- Variables:

obj_of_inf_class, obj_of_cutPart_class, obj_of_reg_class, obj_of_hist_class: these will be used to hold objects from the needed classes that will appear from the main system screen at run time.

Port, listener, thread, listening : these variables used for the http server side connection, where the port will set to "80".

regORcoo: this important variable used to determine the type of received data.

dataset1, dataAdapter1: and other like variables are needed to connect with database.

wholemap_bmp, tracked_region_bmp : and other like variables are needed in the system to hold images.

tracked_region_width, start_lat_value: and other like variables for the height and longitude values , that used to determine the tracked region borders and size.

new_car_id, car_color: these are variables use to store the ID and color of the active tracked car.

verify_car_displayed, car_is_displayed, car_accepted: variable that used to ensure that the car is already displayed on the dynamic region or not, and to test wither new car accepted or not.

Panel_number, btn_inf, panel_and_car_number, active_cars_id:these are variables related to the dynamic part of the system, where the car related regions, symbols, control buttons, and history control regions will be added.

line_points, lat,To_x, Hist_X, lat_d, lat_change, start_point: and other like variables are needed during the process of received data, and for draw of paths and movements of the cars.

Speed: this variable is used to hold the cars' received speed.



Methods:

FrmSystem_Load: this is an event handler method, which will be applied when the form being loaded, to open the connection with database, load the image of the geographic map, cut the tracked region from the whole map, initialize all the dynamic region controls, and their related event handlers.

btnStart_Click: this is an event handler method, which will be applied when the start tracking button in the screen clicked, to initialize the http listener and start Listen thread.

Listen: this is void method, without parameter, that created for the Listen thread to start listening, and to accept new clients, and for each of these new clients create another Listenthread.

ListenThread: this is an event handler method, that called each time new client accepted by the listener, that create network stream for the related client, t, to transmit the data according to the information format that established between the server and client .

There are two type of message format can be received as :

- ✓ " RegORcoo = 0,car_id="",car_number="", phone_number="",driver_name=""
- ✓ " RegORcoo = 1,car_id="",latitude="", longitude="" "

Each with the http header that is needed for determine http connection path.

btnNewCar_Click_1: this is an event handler method, that will be applied when accept new car button in the screen clicked, , then an object of registration form will created, and displayed where it's data filled from the received message.

update_values: this is an event handler method, that called when you pressed on the enter button of registration form, to retrieve the ID, color, verified of displayed string, and close the registration form, that call another event handler display_car_on_screen.

display_car_on_screen: if data entered from the registration form correctly without errors, then if it's already displayed on screen notify that with a message, else call another method add_new_car to display car dynamic region on screen.

add_new_car: this is void method, without parameter, that used to add dynamic related region, for each new active tracked car, and set the control size, name, color, and location, to create ordered region for cars, and store each panel ID that being set with, in order to make another processes easier.

btn_inf_Click: this is an event handler method, that will be applied when information button in the dynamic screen region clicked, it will retrieve the selected related car ID information from the database and fill it in the created object of the information class.

new_coordinate: this is void method, without parameter, that will be called when new coordinate being received, and if these values are within the tracked region call another methods, convertGPS_To_XY,draw_line.

convertGPS_To_XY: this is void method, with two parameters of type string that represent the latitude and longitude received values, in order to convert them into their equivalent X,Y values that will be used to draw on the map, and also the values that will be used to draw on the history map.



The built equation that will be used for X value is :

To_x = Convert.ToInt32(lat_step) + 243;

where:

(lat_step)=lat_diff / 0.000011429.

(0.000011429) related to width pixels with range of digrees.

(lat_diff)=lat_d-region_start_lat.

(243) related to left start border in the image.



The built equation that will be used for Y value is :

To_y = 6930 - Convert.ToInt32(lng_step) + 3560;

where:

(lng_step)=lng_diff / 0.000009707.

(0.000009707) related to height pixels with range of digrees.

(lng_diff)= lng_d-region_start_lng.

(3560) related to upper start border.

The same relations are created to convert the latitude and longitude received value to the X,Y values for history related points, with some changes in the constants, then after this conversion, the draw_line method will be called.

draw_line: this is void method, with one parameter, which is integer that represent the sender ID, it will added the data to database, and draw a line between the new received point and the last processed point for that car, and each time a new point drawn the map will be updated to reflect this change for the system user.

picMap_Click: this is an event handler method, that applied when click on the picture box region, so it will create New object of the cut part class, where the zoom can be performed.

display_clicked_part_of_image: this is void method, without parameter, that will be used to retrieve the related clicked point region on the map, from the whole map with size determined by the cut width and cut height.

btnHistory_Click: this is an event handler method, that applied when click on history button, that will open the history form.

btnStop_Click: this is an event handler method, that applied when click on stop tracking button, to close the system.

- **FrmRegCar class:**

This class will be used when cars try to register to the system, in order to view the cars information that try to connect with, give the user the ability to select the car tracked color on the screen, to change or modify information, and then to add this registration to the database if it's not set before :

- Variables:

Dataset1, dataadapter1: these are variables that needed to connect with database.

enter_error: this is Boolean variable that used to notify any error occurred during the insertion of data.

new_car_id: this is used to hold the value of the new car id.

car_color: this is used to hold the value of the new car color.

enter_is_pressed: this is Boolean variable that used to notify the case of close the form without accept the car, or the request to enter the data to system.

- Methods:

FrmRegCar_Load: this is an event handler method, that will be applied when the form being loaded, to open the connection with data base, initialize enter_error, and view the registration data.

BtnEnter_Click: this is an event handler method, that will be applied when enter button in the screen clicked, to get the registration car ID, then if it's found in data base before this time, show message for this case, if it's new car call the insert_Row method.

insert_Row: this is void method, without parameter, that called when the registration new and not registered in data base before, so insert it to the database.

- **FrmCarInf class:**

This class will be used to display the selected car information, that stored the first time it register to the system, any time the user want during the run of the system:

- Variables:

timer: that will be used as counter to display the information screen for specific time, so when you forget to close it will disappear automatically.

- Methods:

FrmCarInf_Load: this is an event handler method, that will be applied when the form being loaded, to get the timer its interval value, and then start its down counting, and view the information data.

t_tick: this is an event handler method, that will be applied when the timer stop, to hide information screens that opened for long time, and not closed yet.

- **FrmCutPart class:**

This class will be used to perform zoom in and zoom out on the selected region, that cut from the whole image when you click on it in the system form. In order to give clear level of view than the whole one:

- Variables:

zoomIN: this is Boolean variable used to indicate that we want to perform zoom in operation not zoom out.

zoomOUT: this is Boolean variable used to indicate that we want to perform zoom outoperation not zoom in.

zoom_level: this is used to determine the level of zoom that you currently perform, and it's value change by clicking on the image picture box, which has max value of "14".

Mapbmap: this is image object, that represent a copy of the whole map that the zoom will performed on, in order to hold all changes that occurs on the whole map during the run of the system.

map_x: this will retrieve the related x value for the clicked point on the whole map box to the whole map original bmp image, as it stored in its path.



The built equation that will be used are :

obj_of_cutPart_class.map_x = (tracked_region_width / picMap.Width) * point_x;

map_y: this will retrieve the related y value for the clicked point on the whole map picture box to the whole map original bmp image, as it stored in its path.



The built equation that will be used are :

obj_of_cutPart_class.map_y = (tracked_region_width/picMap.Width)*point_y+3560;



Methods:

butZoomIN_Click : this is an event handler method, that will be applied when the zoomIN button in the screen clicked, to set the Boolean zoomIN value to be true and zoomOUT false. **butZoomOut_Click:** this is an event handler method, that will be applied when the zoomOUT button in the screen clicked, to set the Boolean zoomOUT value to be true and zoomIN false.

picSelectedRegion_Click: this is an event handler method, that applied when click on the picture box region, determine zoomIN or OUT, ensure that zooming level not reach max, increase or decrease the zoom level , then call DO_zoom function.

DO_zoom: when we click on any region on the cut part to perform zoom we needed the following equations:



To find the value of clicked points related to the original whole map,

by the built equation:

x2=(x2*3)+(map_x-900);

Where:

(x2): at first equal the x value of clicked point on the zoom picture box.

(3): comes from "cut width/zoom Box width = 1800/600".

(900): we want to retrieve the clicked point at the center of the new image, so we subtract "1800/2= 900".

The same equation will be applied for the y value.



To determine the zoom size we need the built equation:

zoom_size = 2*(900-60*zoom_level);

Where:

(60): represents our factor of zoom each time.

The same equation will be applied for the y value.

Then and before retrieving the zoom region, we will determine the start x value and y value for the region, and perform testing for the cases of click near the borders of the image, finally we will draw the retrieved zoom region from the whole image in both zoomIN or out cases.

- **HistoryMap class:**

This class will be used to display the selected car history at specific interval, any time the user wants during the run of the system:

➤ Variables:

view_pressed_once: used to hold how many time you pressed on draw path button.

dataset1, dataAdapter1: and other like variables, that used for connection with data base.

history_id: used to store selected car id.

dateTime_Start,dateTime_end: used for select the history interval.

history_points_number, history_points: used to store the related interval and car history points.

car_color, Graphics s: used for draw the points on map.

➤ Methods:

HistoryMap_Load: this is an event handler method, that will be applied when the form being loaded, to initialize the view_pressed_once variable to zero.

btn_retrieve_cars_Click: used to retrieve all cars id from database.

comboID_SelectedIndexChanged: used to select one of cars by its id.

comboxstart_SelectedIndexChanged: used to select start time of interval.

comboxend_SelectedIndexChanged: used to select end time of interval.

btnViewHist_Click: this is an event handler method, which will be applied when the draw path button in the screen being clicked, to draw the path on map.

4.3 System use cases

A detailed of how the system outside users, and the system actions will interact with each other during the process of the system, will be described in the system use cases clearly in this section, as shown in figure (4.3) :



From the figure we can note that:

- Two actors mainly interact to our system, the mobile user, and desktop user.
- The mobile user who will start the mobile application.
- Then he will fill the required registration fields, and activate the action of send the registration request from mobile to the main server.
- This action will activate the process of selection the tracking mode, where mobile user needed to select mode, also activate the action of received these data at the server side.
- Then the other actor desktop user, will interact with this part by accept this car, and added it to the system.
- If car added to the system this will activate the action of reading GPS coordinates on the mobile side, and send it to main server, where also the selection of tracking mode is needed before start sending coordinates and speed.

- Also this action will activate the action of receiving data at the server side, and each time it received it will be processed and plot on the map.
- Then after plot in map, it will be used for history process, by adding it to data base where it can be retrieved any time you want.
- Other actions can be performed by the desktop user which are start the desktop application.
- The desktop user activate the action of Log in to the system to continue the processes.
- The desktop user also can activate the action of tracking, and the ability of receiving new cars.
- The desktop user can activate the action of zoom in and out operation when needed.
- The desktop user can activate the action of display cars information when needed.
- The desktop user activates the action of display cars history when needed.
- The desktop user can end the desktop user.
- Also the mobile user can end the mobile application.

4.4 System Sequence diagram:

For each of the use cases a sequence diagram will be created in this section, in order to show which objects are needed to perform this case, how they will interact with each other, and the time sequence of the process steps :

4.4.1 Send registration request sequence diagram:

As shown in figure (4.4); we can realize the time sequence for this use case, where the mobile user needed to fill the required registration data fields, then he click on the “send” button, this will open new httpclient, get the text from the fields, then it will execute http post that first will request the information data from the register form, then arrange them into an array list, then execute the post process to the server httpURL.

And the main methods in the system needed for this process are:

```
public void onCreate(Bundle savedInstanceState)
```

```

public void send(View v)
public void run()
startActivity(new Intent("gps.post.TRACKING"));

```

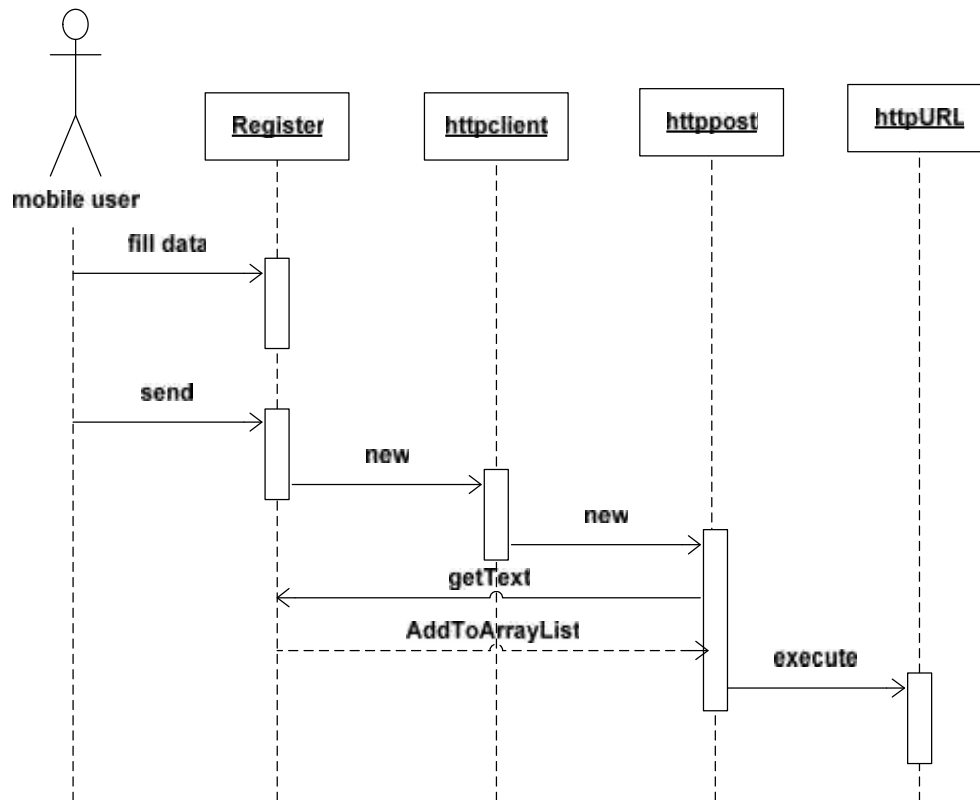


Figure 4.4 send registration request sequence diagram

4.4.2 Receive new request sequence diagram:

As shown in figure (4.5) ; we can realize the time sequence for this use case, where the server is built to accept any number of request, by multithreading way, where when the listener accept new client it create a thread for this new client to receive data with it, while other request allow to connect at the same time.

And the main methods in the system needed for this process are:

```

public void Listen().
public void ListenThread(Object client).

```

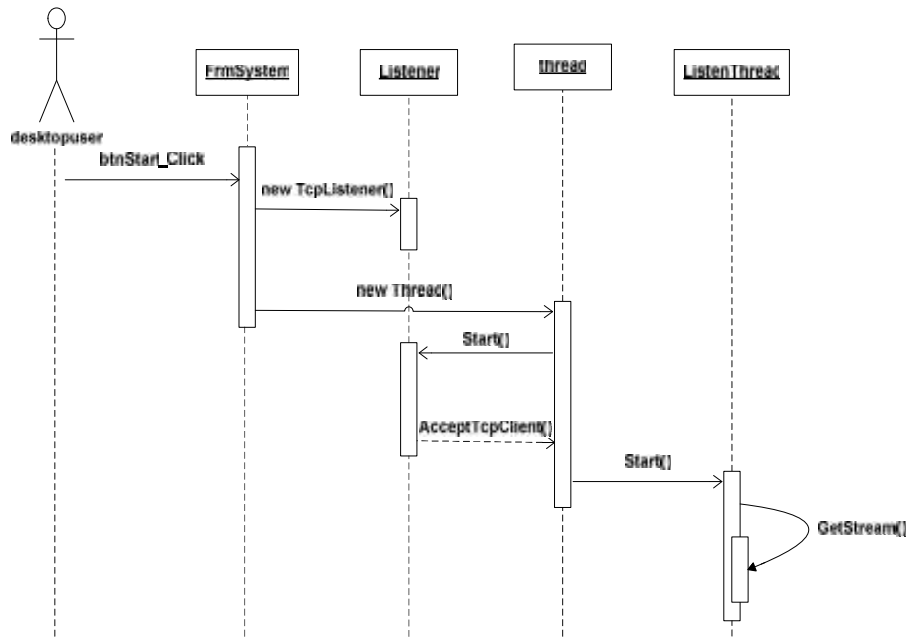



Figure 4.5 receive new request sequence diagram

4.4.3 Select mode sequence diagram:

As shown in figure(4.6); we can realize the time sequence for this use case, where after posting the registration data and accept them by the server the mobile application open new activity called the mode activity, where the user have to decide the mode of travelling, if he is walking or driving by clicking the proper button.

The two modes have variation in timer variables with similar methods, the difference appears here:

```

public void requestLocationUpdates (String provider, long minTime, float minDistance, LocationListener listener)
  
```

The **minTime** and **minDistance** variables differ from one mode to another, and they indicate the time interval or the distance at which the location is required to be sent.

And the main methods in the system needed for this process are:

```

protected void onCreate(Bundle savedInstanceState)
public void onClick(View v)
  
```

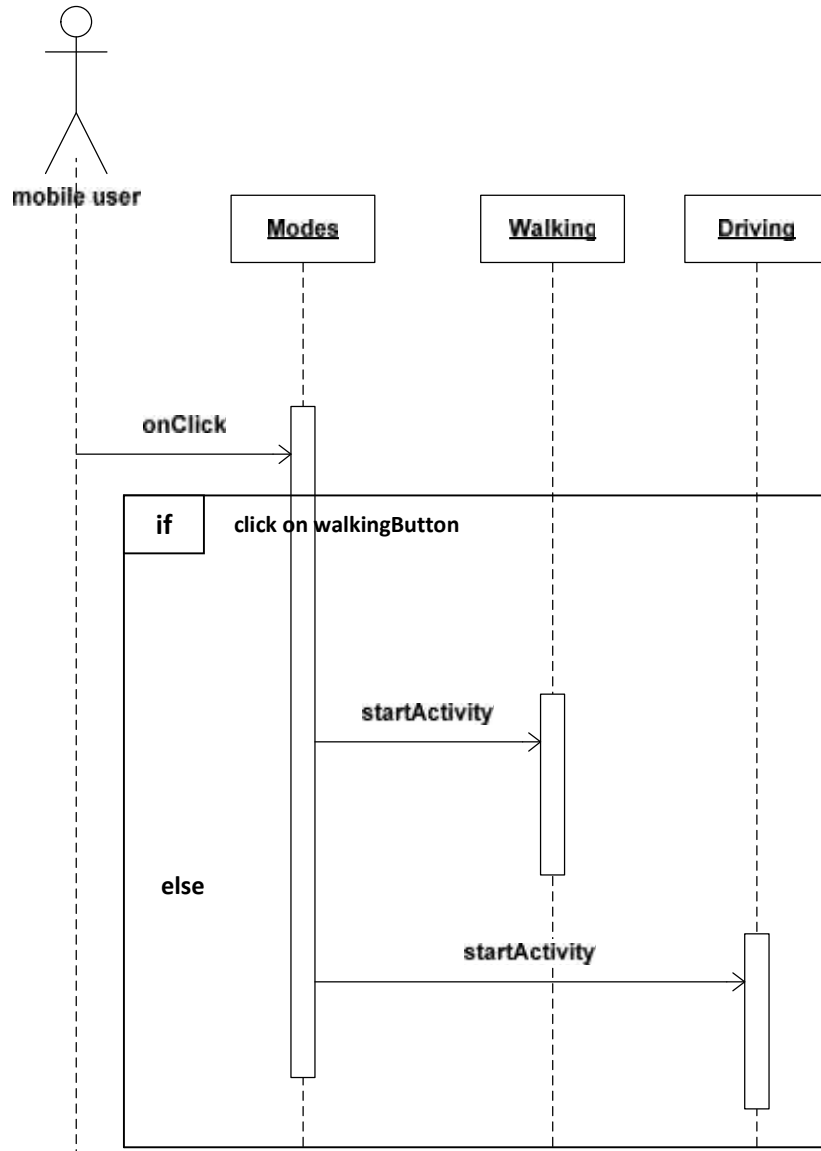


Figure 4.6 select mode sequence diagram

4.4.4 Adding car to system sequence diagram:

As shown in figure (4.7); we can realize the time sequence for this use case, where each time the system receive new request it display a message to notify that, then the desktop user needed to press on accept new car button to continue its registration, and select the tracked color, this will open the registration form with data, then when user press enter , if data are new then it added to database, else it's related information will be retrieved from database, then after press enter the car related symbols will added to the dynamic region on the screen, if it's already displayed a message will appear to notify that.

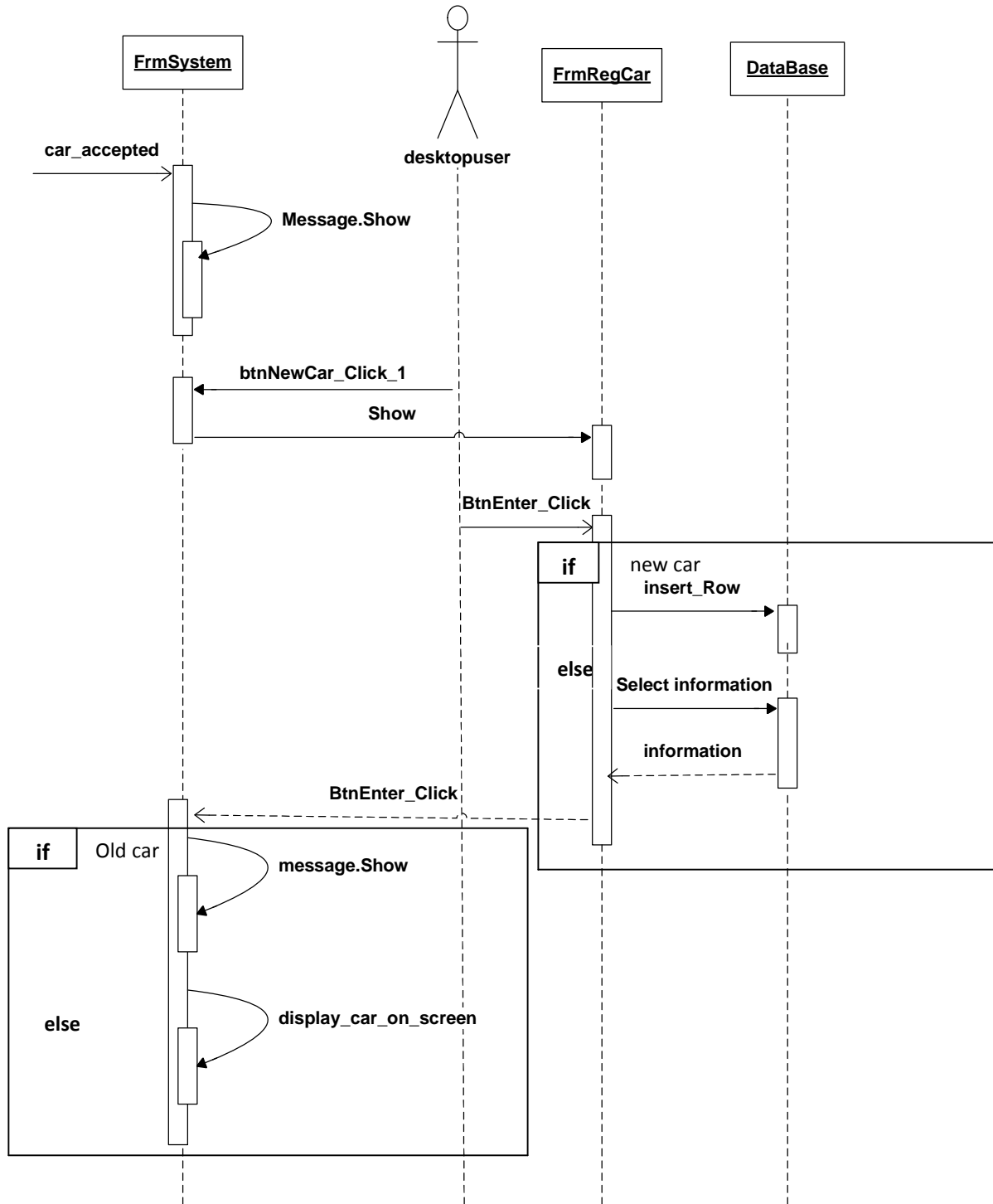
And the main methods in the system needed for this process are:

```
private void btnNewCar_Click_1(object sender, EventArgs e).
```

```

private void update_values(Object sender, System.EventArgs e).
private void display_car_on_screen(object sender, FormClosingEventArgs e).
void add_new_car().

```



4.4.5 **Figure 4.7 Adding car to system sequence diagram**
Send GPS coordinates sequence diagram:

Figure 4.7 added car to system sequence diagram

As shown in figure (4.9); we can realize the time sequence for this use case, where each time a ListenThread receive new coordinate, it will be processed with some equations as shown above in the classes implementation, then after process the system will draw these points at the map, and added it also to the data base, to be used in the history retrieve.

And the main methods in the system needed for this process are:

```
private void new_coordinate().
public void convertGPS_To_XY(string lat, string lng).
public void draw_line(int ID).
```

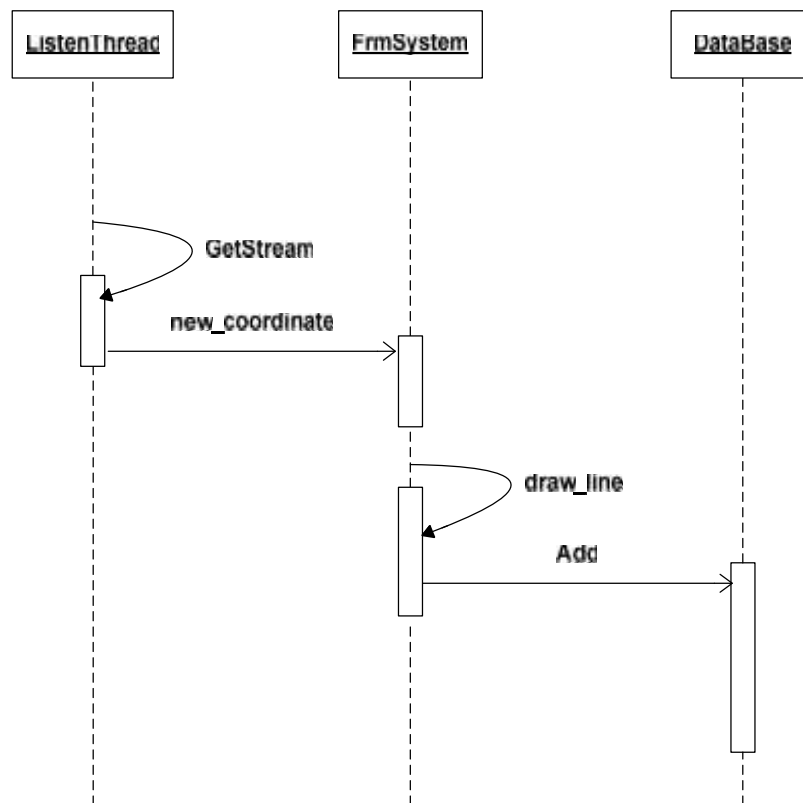


Figure 4.9 plot received coordinates sequence

4.4.7 Log In to system sequence diagram:

As shown in figure (4.10); we can realize the time sequence for this use case, where the desktop user need to log in to the system, by insert correct user name and password in order to go to the next form. But while insert wrong user name or password, he will stay in the same log in form.

And the main methods in the system needed for this process are:

```
private void BtnEnter_Click(object sender, EventArgs e).
private void check_Registration().
```

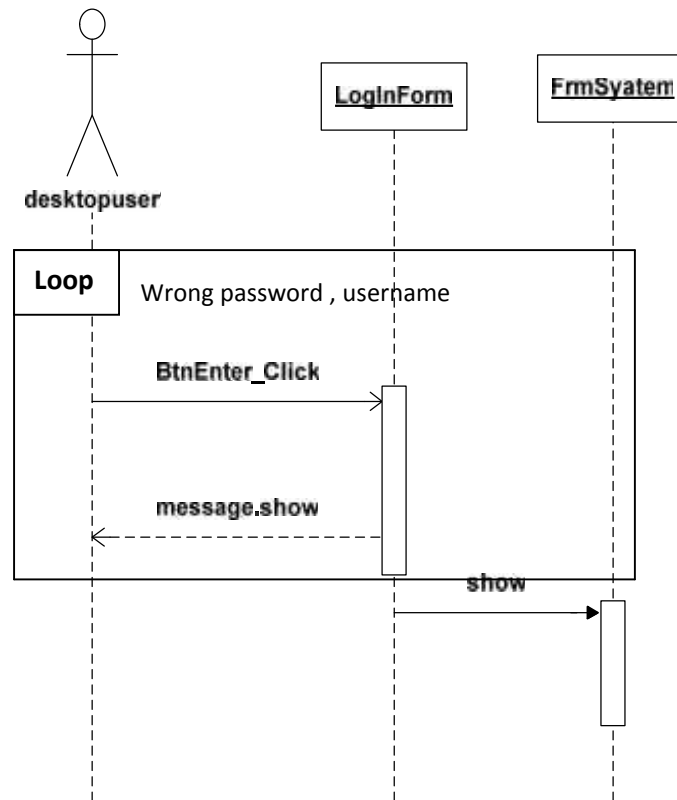


Figure 4.10 Log In to system sequence diagram.

4.4.8 Activate tracking sequence diagram:

As shown in figure (4.11); we can realize the time sequence for this use case, where the user needed to start tracking by click on start tracking button, this will initialize the listener, and create a thread that will start the listener, in order to accept new connections.

And the main methods in the system needed for this process are:

```

private void btnStart_Click(object sender, EventArgs e).
public void Listen().
public void ListenThread(Object client).
  
```

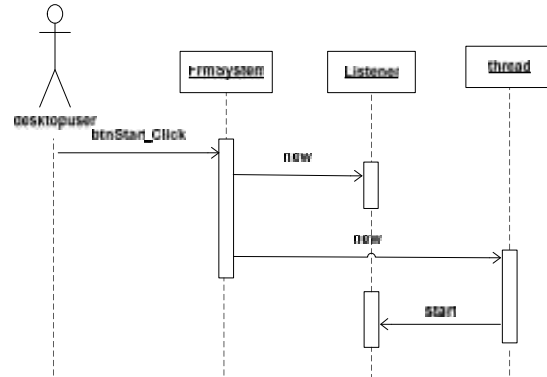


Figure 4.11 Activate tracking sequence diagram

4.4.9 Zoom sequence diagram:

As shown in figure (4.12); we can realize the time sequence for this use case, where the user needed to press on the whole map at the system form, this will cut a region around the clicked point, then to perform another levels of zoom in and out the user also will press on the cut map, then the zoom will done.

And the main methods in the system needed for this process are:

```

private void picMap_Click(object sender, EventArgs e).
private void display_clicked_part_of_image().
private void picSelectedRegion_Click(object sender, EventArgs e).
private void butZoomIN_Click(object sender, EventArgs e).
private void butZoomOut_Click(object sender, EventArgs e).
private void DO_zoom().
  
```

4.4.10 Display cars information sequence diagram:

As shown in figure (4.13); we can realize the time sequence for this use case, where the desktop user request the information form from the system by click on the information button, this will retrieve the request data from the database and display on the form and open it.

And the main methods in the system needed for this process are:

```

private void btn_inf_Click(Object sender, System.EventArgs e).
private void FrmCarInf_Load(object sender, EventArgs e).
  
```

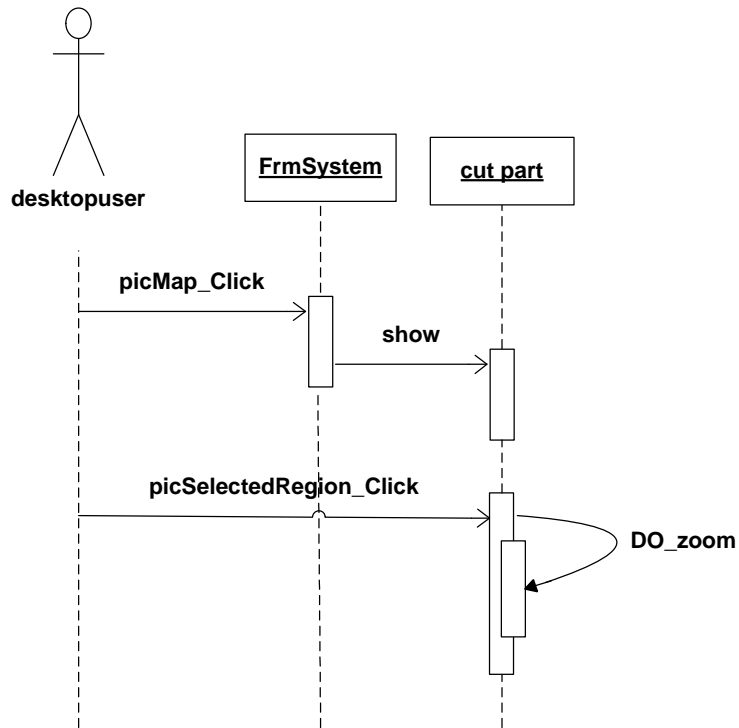


Figure 4.12 zoom sequence diagram

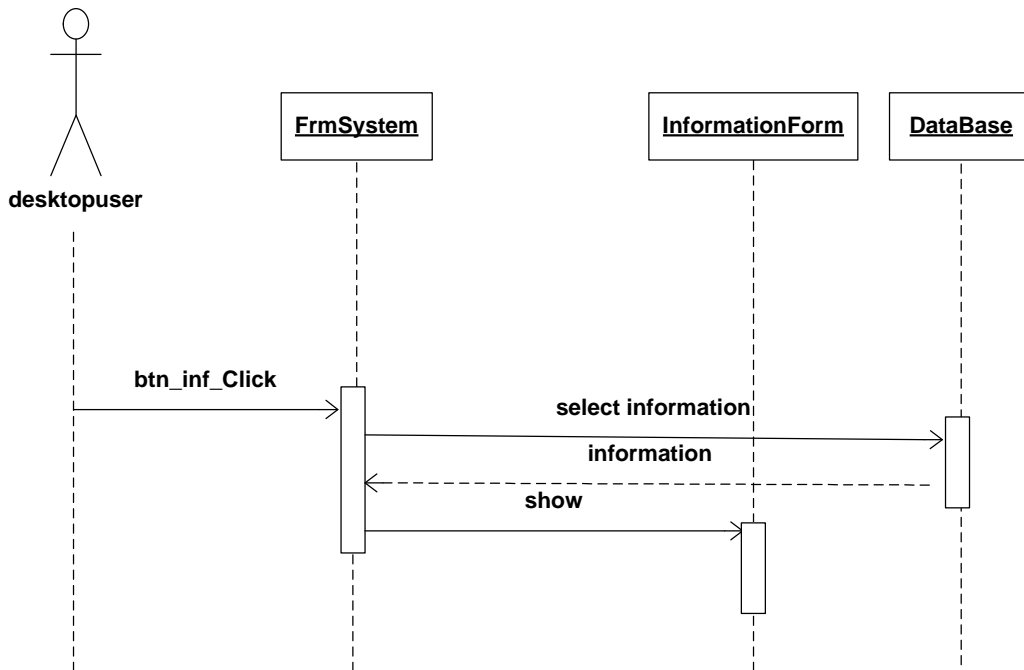


Figure 4.13 display cars information sequence diagram

4.4.11 Display cars history sequence diagram:

As shown in figure (4.14) display cars history sequence diagram; we can realize the time sequence for this use case, where the user first request the history dynamic region, this will added it's related symbols on the screen, then the user needed to select its history region, then also he must request the history map to appear, then at this form he will request the points to be drawn.

And the main methods in the system needed for this process are:

```
private void HistoryMap_Load(object sender, EventArgs e).  
private void btn_retrieve_cars_Click(object sender, EventArgs e).  
private void comboID_SelectedIndexChanged(object sender, EventArgs e).  
private void comboboxstart_SelectedIndexChanged(object sender, EventArgs e).  
private void comboboxend_SelectedIndexChanged(object sender, EventArgs e).  
private void btnViewHist_Click(object sender, EventArgs e).
```

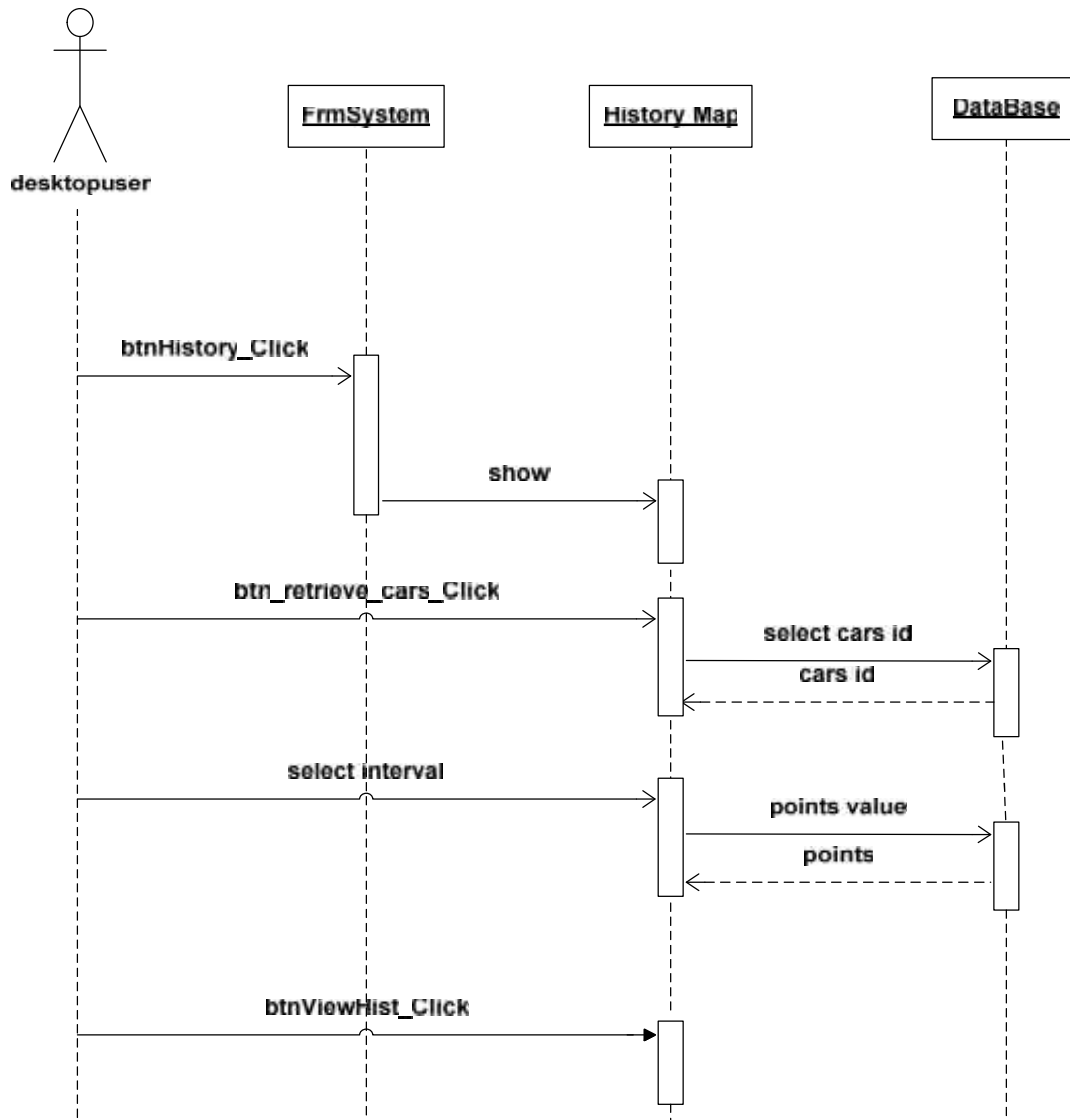


Figure 4.14 display cars history sequence diagram

5

CHAPTER FIVE

SYSTEM TESTING

- 5.1 Overview**
- 5.2 Sub-System Testing**
- 5.3 Installation and preparing the system**
- 5.4 Testing Scenarios**

5.1 Overview

In this chapter, the whole testing stage will be described, including a test of an entire interconnected set of components and software for the purpose of determining proper functions and achieving the desired goals of the system. We will talk about testing of each part of the system, describe the scenarios, represent each test and its errors, challenges and modifications, and also we will figure out the error rate.

5.2 Sub-System Testing:

The main aim of this testing part is to test the main operations in the system. There are five main operations in the system. The first one is the getting the correct coordinates and speed. The second operation is posting data to the main server. The third is receiving the data and processing it to register car and plot its coordinate to get its path of travelling. The fourth operation is accepting more than one car at the same time. The last one is allowing access to history data base at any time. All the operations mentioned before are tested and meet the expectations.

5.3 Installation and preparing the system

At this stage, all system parts must be ready to be tested at the two sides, mobile application at the car to be tracked, and the desktop application at the server.

5.3.1 Mobile application:

An application has already been built using the Eclipse environment, to use this application it must be installed to the mobile, and to perform this we can do one of the following ways:

- The first is to connect the mobile phone (Galaxy S) by the USB cable or Bluetooth connection to the PC at which the workspace of Eclipse is created. Then the following file must be copied to the mobile SD card.
C:\workspace\GpsCarTrackingSystem\bin\GPS_Car_Tracker.apk

Open the ".apk" file, a message will appear at the screen of the mobile as shown in figure(5.1), show the name of the application and the permissions needed for it to work probably. In our application we need to access the GPS location to get the location of the tracked car, and we need a full internet access for the communication between mobile(client) and PC(server) while posting data.

Click the install button to allow the installation of the application as shown at figure(5.2).

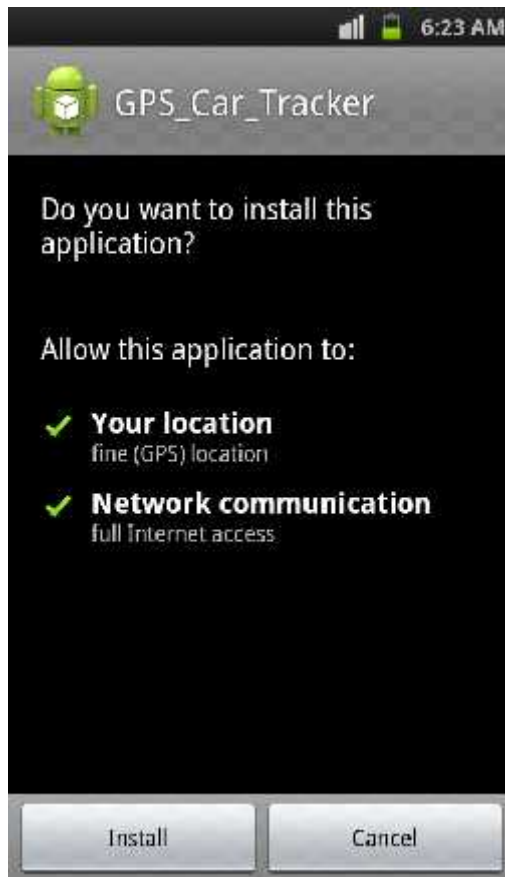


Figure 5.1: Opening apk file

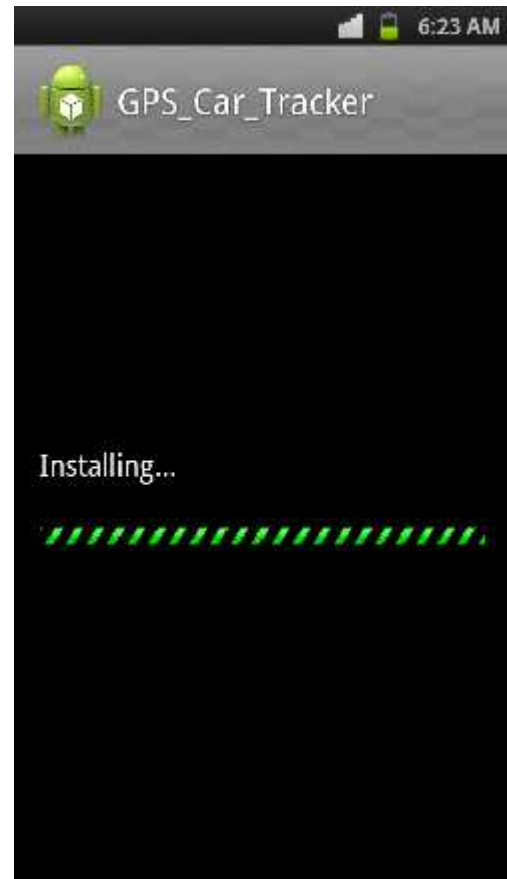


Figure 5.2: Installing application

When the installation is completed, as shown in figure (5.3), click the open button to start the application

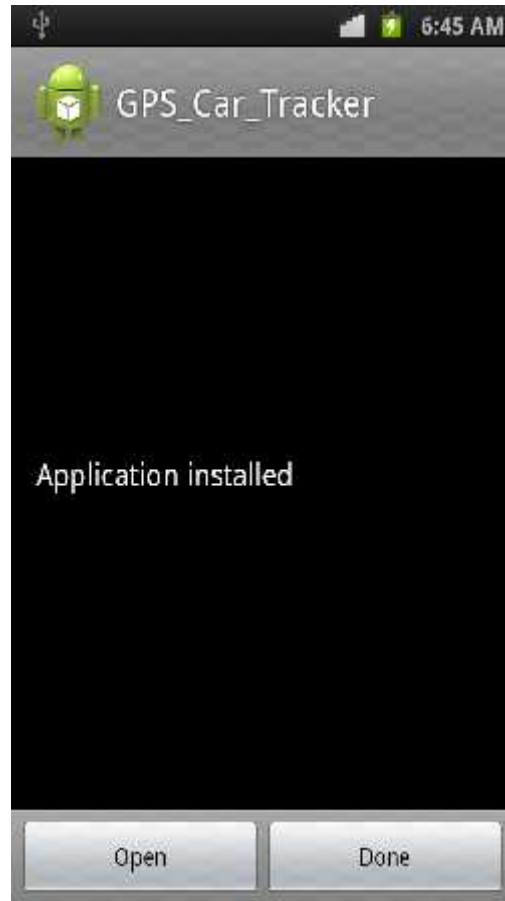


Figure 5.3: Installation completed

- The second way to install the application is to connect the mobile phone directly to the Eclipse and define it as one of the active running android device, choose it manually to run the application on.

After choosing one of the previous way to install the application, it will be opened and became ready to use, as shown in figure (5.4).

Now, the application on the mobile is ready to be used by the user. This welcome screen will last only 7 second then immediately a new module will be opened, as shown at figure(5.5), the user must enter it's information then click the "send car information" button.

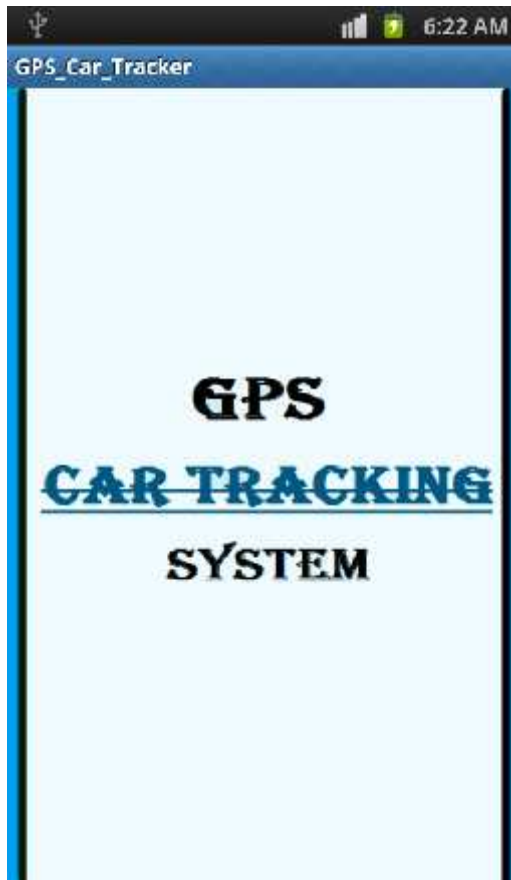


Figure 5.4: Welcome Screen

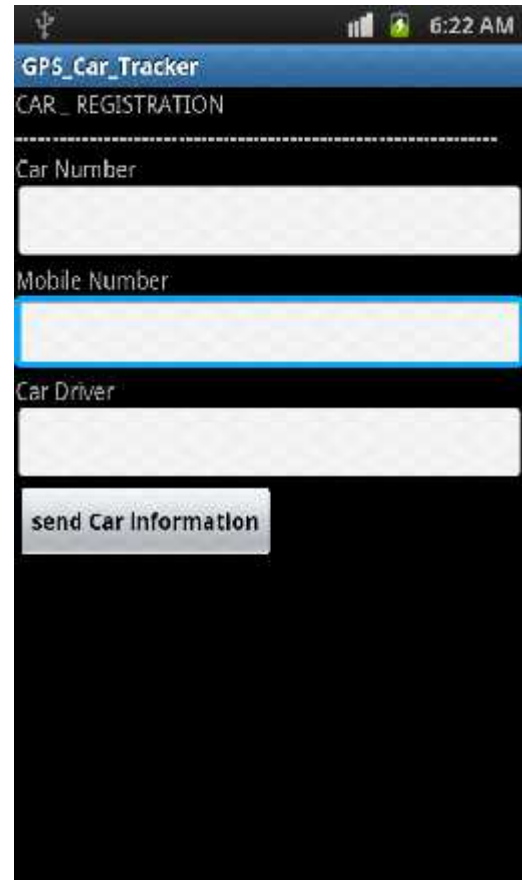


Figure 5.5: Registration

As shown, the required information is: the number of the car, the mobile number of the driver, and his name. This information will be needed at the server side. All the fields are required, so if the user tries to skip this step and click the send car information button without filling the fields a message will appear at the screen and tell him that: "All field are required, Fill it please".

Pressing the send car information button will post the data to the main server and move to the next module, select mode module, as shown at figure (5.6).

- For user how is walking without using a car he should click the " WALKING" button, to move immediately to walking module shown at figure(5.7).
- For user how is driving a car, he should click the "DRIVING" button to move immediately to driving module shown at figure (5.8).

At both of these modules, if the GPS is disabled immediately the following message will appear at the screen "GPS Disabled, turn on to start tracking your car", to tell the user that he has to turn GPS on in order to start receiving and sending the coordinates.



Figure 5.6: Select mode

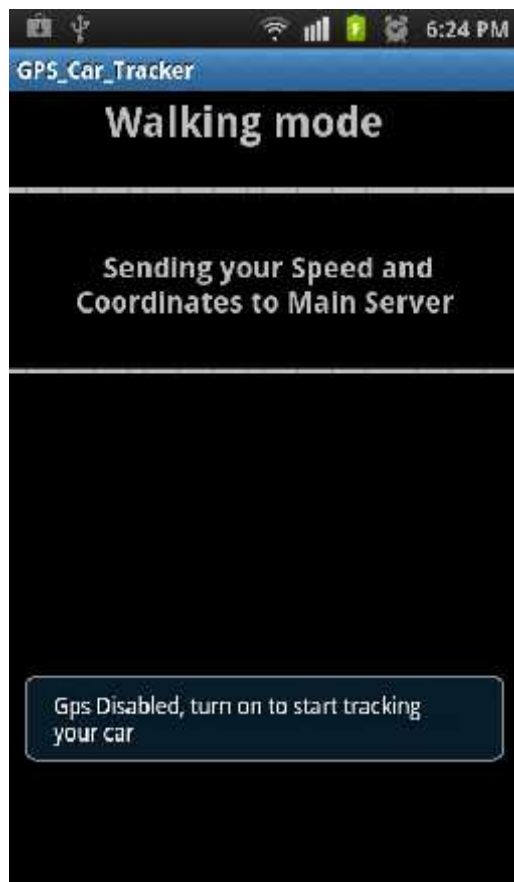


Figure 5.7: Walking mode/GPS off

module/Coordina

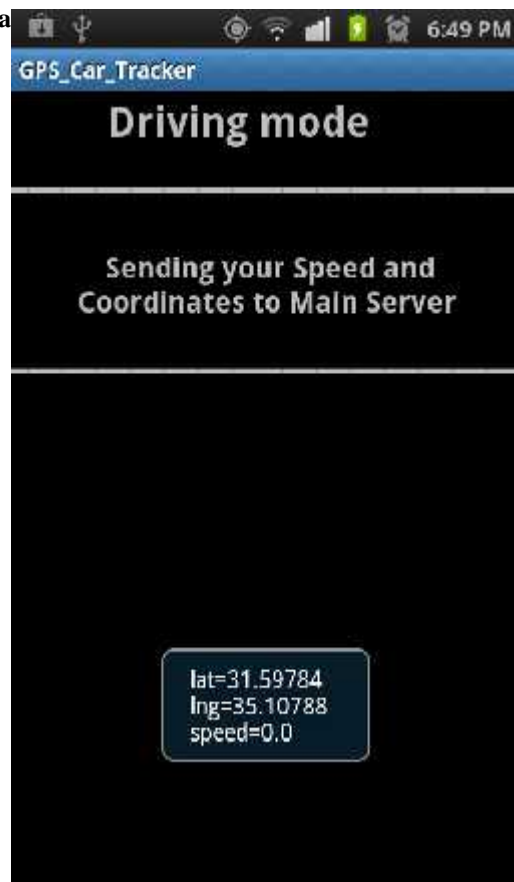


Figure 5.8: Driving mode/ coordinates and speed

While the car is moving, the coordinates and speed will appear at the screen and posted to the server at the same time. As shown at figure (5.8).

Now the driver can put the mobile in front of him at the car while moving, so the mobile application will work alone without any efforts from the driver.

5.3.2 Server Application:

An application has already been built using c# programming language, and to start up the application:

- we will open our built project "GPSCarTrackingSystem" From the project path, that was created using the Microsoft visual studio 2008 program. As shown in figure(5.9)

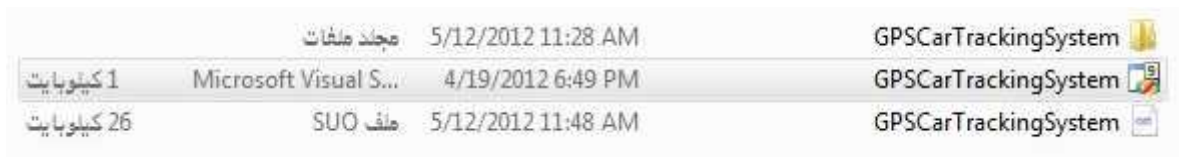
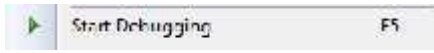


Figure 5.9: GPSCarTrackingSystem Project

- Then after start it, a screen for the c# 2008 environment will be open with the project solution explorer that contain our system, then we debug the system to start it by click on Debug then  , or press F5.
- Then our system will start its execution, where the first screen are the Welcome Screen as shown in figure (5.10)



- **Figure 5.10: GPSCarTrackingSystem Welcome Screen** name and password to sign in to the system as shown in figure(5.11)



Figure 5.11: GPSCarTrackingSystem Log In Screen

- Then we will reach the main system screen where all processes will created from, and the design for this screen is shown in figure (5.12)

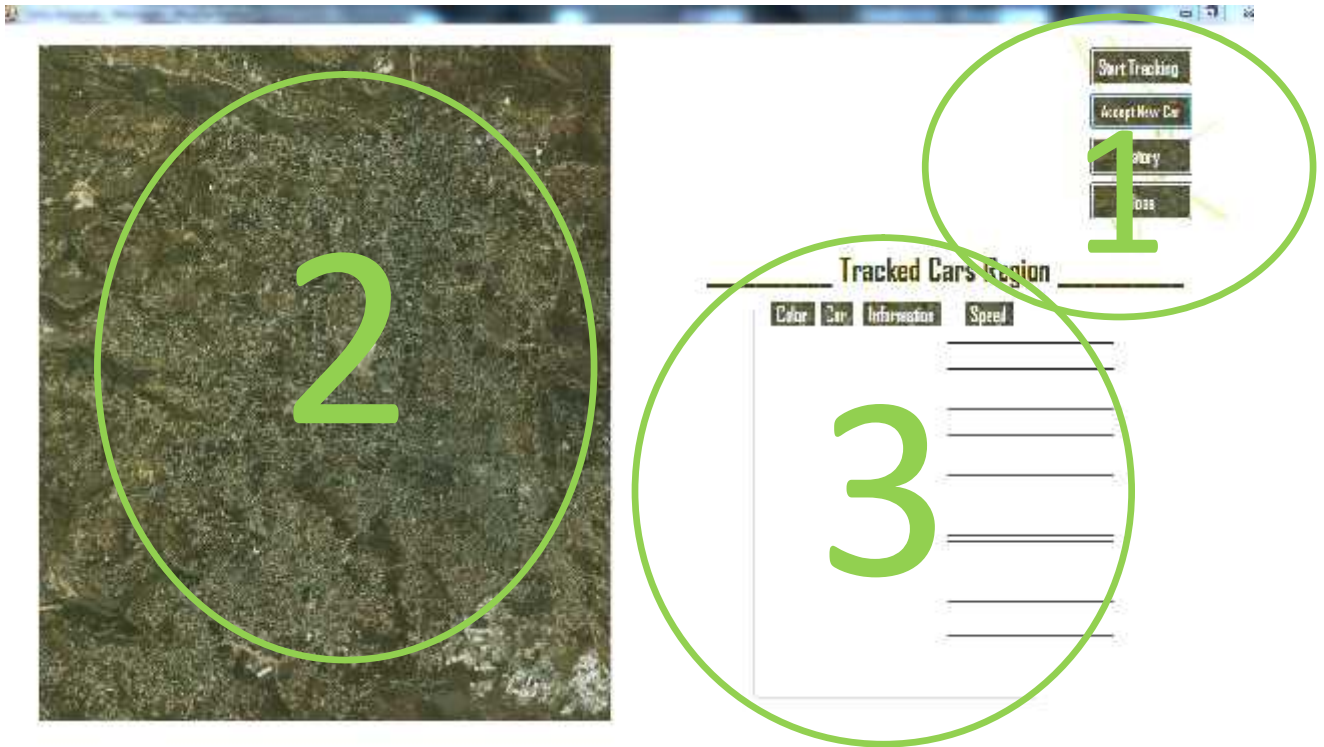


Figure 5.12: GPSCarTrackingSystem Main Screen

As show

➤ Region1:

It is the region for control buttons that used to start the tracking process, accept new car's try connecting to system, deal with history part, and to end the process of tracking.

➤ Region2:


This is the region where the whole tracked map being hold, which is represent a part of Hebron city, around PPU university.

➤ Region3:the dynamic region where car's will be added and their related symbols, as shown in figure (5.13) :



Figure 5.13: system screen car's dynamic region

And here we need to note that the test are done for maximum of 10 car's active, and tracked at the same time on the screen.

- Now to start tracking we need to press on the control button , then when new connection received a message will appear to notify that as shown in figure (5.14):

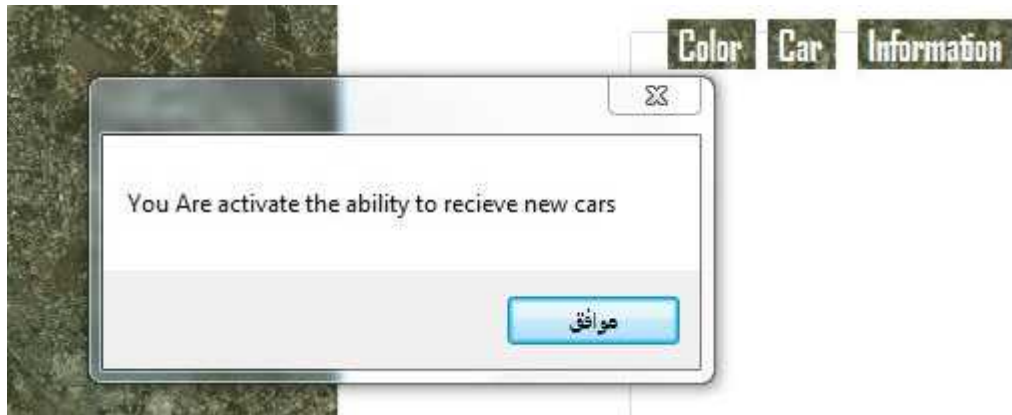


Figure 5.14: new connection message


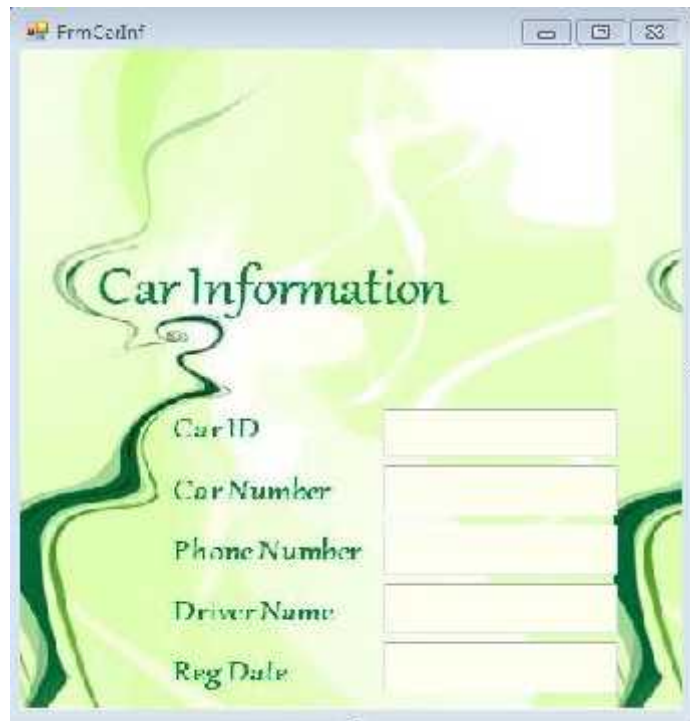
- Then when user must press on the control button  this will open the registration screen with the registration information, and this screen is as shown in figure (5.15):



Figure 5.15: System registration screen

- During the process there are other processes available, like view car's information that stored with it. And the form for this information is as shown in figure (5.16):



The image shows a screenshot of a Windows application window titled "FrmCarInf". The window contains a form titled "Car Information" with a light green background and a stylized car outline. The form has five input fields:

Car ID	<input type="text"/>
Car Number	<input type="text"/>
Phone Number	<input type="text"/>
Driver Name	<input type="text"/>
Reg Date	<input type="text"/>

- Also you figure (5.17):

Figure 5.16: System Car's Information form 1 in

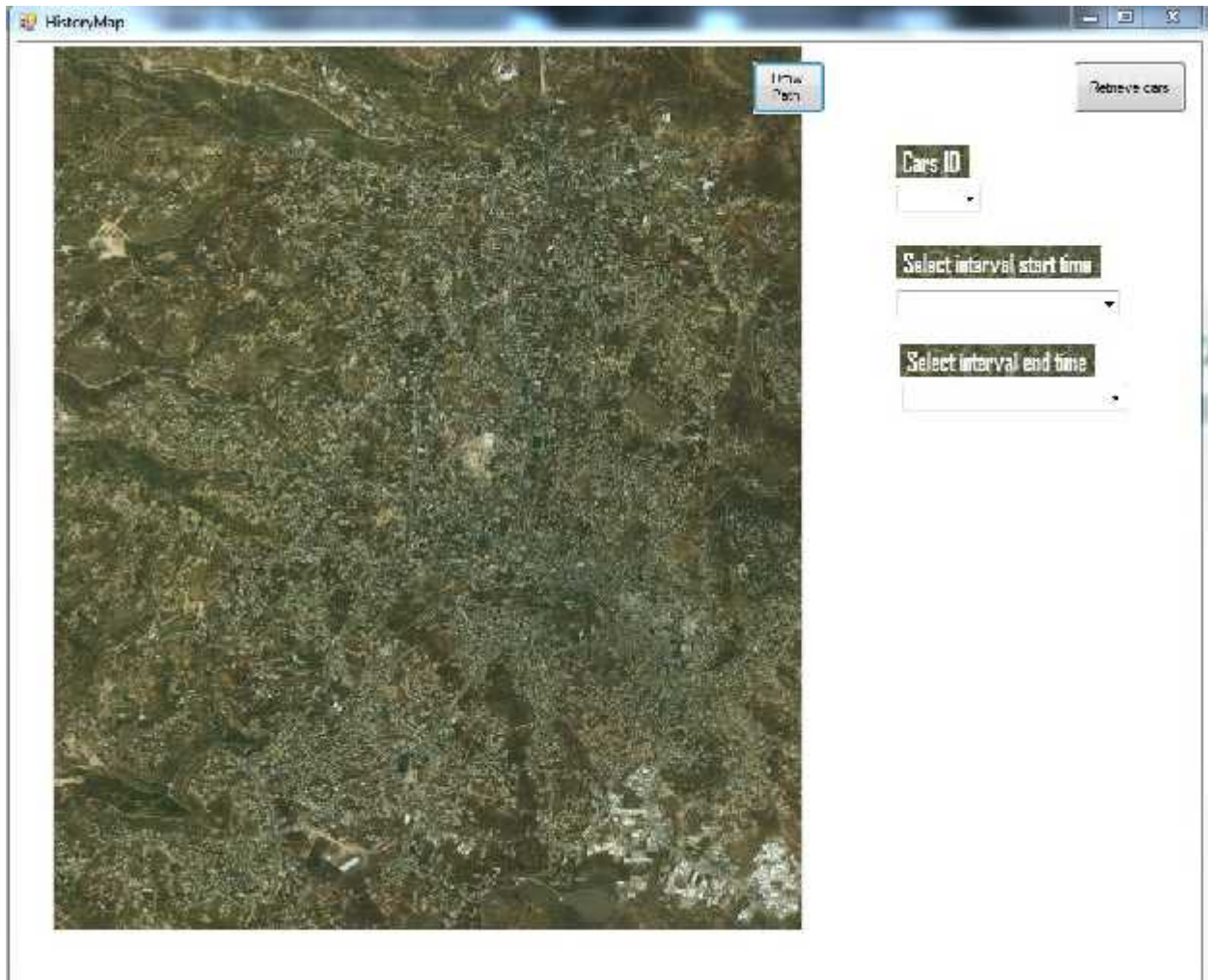


Figure 5.17: System Car's history form

- Also zooming can be performed by click on whole tracked region to give better resolution, for the image. And the form for that is as shown in figure (5.18):



Figure 5.18: System Car's Zoom In form

These are the main screens that will appear during the process of the system at the server side.

5.4 Testing Scenarios

In this section, the scenario of starting the system with all its parts will be mentioned, including all the necessary steps which include:

- Download the mobile application and open it, then give it to the car driver.
- Open the desktop application and login to it, then activate the listening for the new cars to register to system, by clicking on the " Start Tracking" button.
- At the mobile, fill the registration information and click the "send Car information" button.

- When the data reach the server a notification message will be displayed , it will be accepted by the user at the server side, and a color of the tracking path will be selected by selecting from system colors list, and then added to system by click enter.
- After the acceptance, the mobile application will start the tracking module; if the GPS receiver is off, the application will inform the driver to turn it on by a text appears at the screen bottom. So:
 - If GPS is off, Message= "GPS Disabled, turn on to start tracking your car".
 - If GPS is on, message="GPS Enabled".
- Enable the GPS, the mobile will start receiving GPS signals and calculate it's location's coordinates then show them on the screen and post them to the main server.
- At the same time a connected dots will appear continuously at the map on the main server screen.
- If more than one car is active, there will be more than one path at the map with different colors each with its' related car. And for more clarity. we can perform the operation zoom in and out to see the path; by click the "zoom in" button or “zoom out” as you want, then press on the map where you want.
- And during plot process also you can select information display and history display by click on their related buttons at the screen.

To make a test, the previous scenario including all of the previous steps is done in order. We have performed a number of tests to explore if the system works probably and to figure out the errors and challenges then make modification and corrections.

5.4.1 TEST 1

The first test is applied over a small distances without using a car, this is done at the university. A person carries the mobile, open the application and start walking with a medium speed around building B to building A, with one mobile.

Results

Number of reading = the mobile sends so many points approximately 50 with 6 only appear on the map.

The total cost= 1.00 NIS

The speed between 1 to 2 meter/ second.

Time: the time is approximately 3minutes.

The drawing path: shown at figure (5.19).

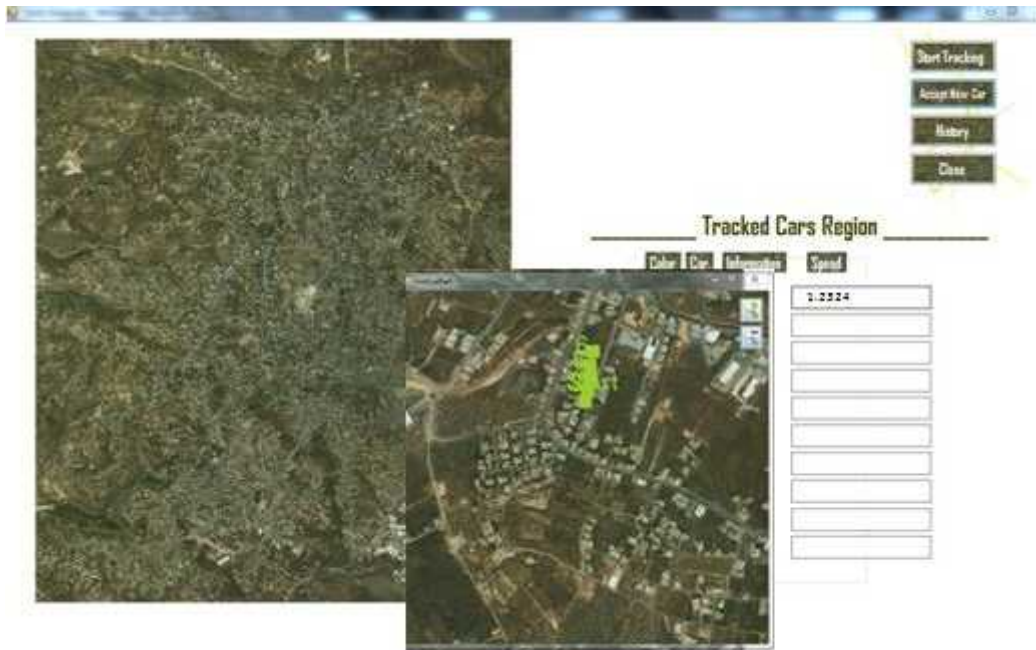


Figure 5.19 test1 result tracked paths

Errors

While testing the system the following errors happened:

- When the mobile sends the registration request to the server, it starts directly sending coordinates, before the server complete the accepting and registration of the new car, and the car information forum stay empty.
- At the server software there are error occurred after drawing a small number of points, that said there is out of memory exception.

Challenges

The mentioned errors bring the following challenges :

- Give the server the enough time to complete the registration and acceptance process.
- Dealing with image errors at server side.

Modifications

The following modification is done to solve the errors and meet the challenges:

- At the mobile side a delay between the Register activity and Tracking activity is added,

```
timer = 0;
while(timer < 9000){
    sleep(1000);
    timer=timer +1000;
}
```

so it gives the server the needed time to assign a color to the car and accept it.

- At the server side, the error is expected to be because each time we draw on the map we call it again and create object of bitmap class with large size of pixels, without being disposed after use, so to solve the problem we try to use the same object and overwrite it each time we use and define it as public variable, used once in the system with modify on the same object each time.

5.4.2 TEST 2

Another test is performed after applying the previous modifications, to check the system state after solving the errors. It's also over small distances without using cars yet, where two persons move between building A and B inside the university. one person carries the first mobile and start tracking, then after time the other mobile start working while the first also working at the same time.

Results

First mobile
Number of reading =30.
The total cost =0.6 NIS
Time: approximately 5 minutes.

Second mobile
Number of reading= 10.
The total cost = 0.2 NIS
Approximately3 minutes.

Speed for both are approximately between 1 and 2 meter per second.

The drawing path: shown at figure (5.20):

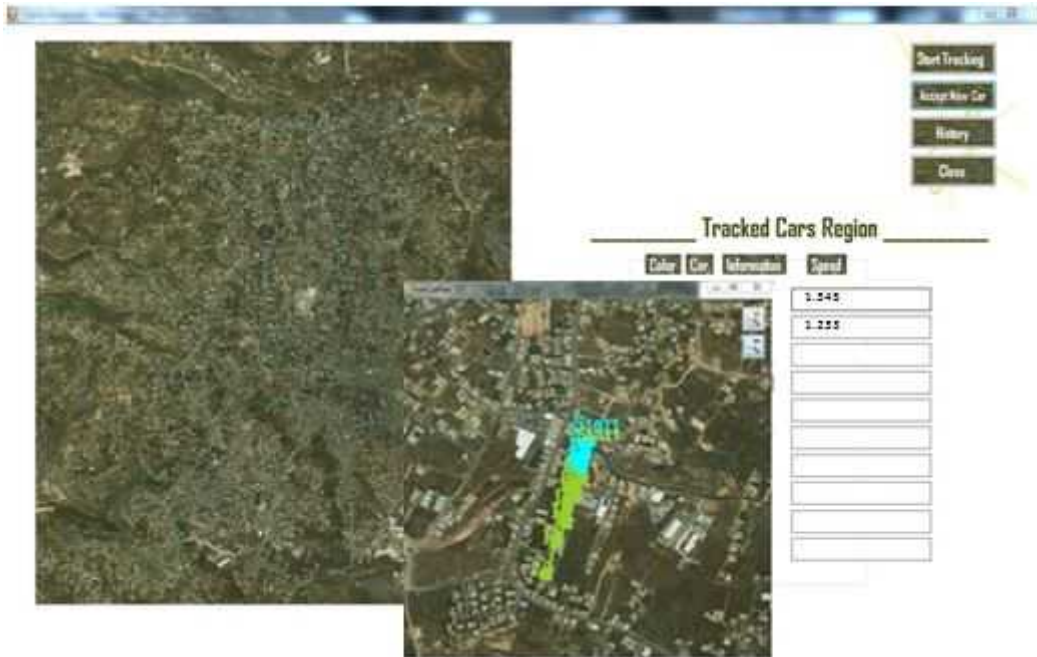


Figure 5.20: test2 result tracked paths

Errors

The errors that we faced previously are disappeared at this test and the results are acceptable and meet the expectations, but other errors were appeared

- The first mobile start sending readings continuously at very near places, appear approximately on the same point at the map, and with large number of coordinates during small interval, the same like first error on mobile side.
- The number of dots on the map related directly to the cost (how many time the application post data to the server?), this produce an unacceptable cost must be reduced.

- At the server side Error occur during process of two mobiles at the same time, and the application on the server meet errors and work incorrectly with two mobiles ,where following error message appear as shown in figure (5.21):



Figure 5.21 desktop test2 error

Challenges

All of these mentioned errors bring the following challenges:

- Control when to send coordinates.
- Reduce the cost.

Modifications

The following modification is done to solve the error and solve the challenges:

- Two timers is added to the mobile application to control the post process.
 - Timer # 1: Measures the travel distance, coordinates are sent every 'x' meters.
 - Timer # 2: Measures the time between readings, coordinates are sent every 'x' seconds or minutes.
- For reducing the cost, unnecessary data was deleted from the sent messages from mobile to server. Also we take only the first five digit after the decimal point from the longitude and latitude
- By putting these timers we try to solve the first two mentioned challenges; when to send data and reduce the cost

- At the server side, the error is expected to be because each time we draw on the map we use the object whole map_bmp, and when two reading reaches at the same time, they try to use the same object of map to draw on at the same time, so this cause the error, and to solve it we try to use a build in loop to create some delay between reaching coordinates, this time is small but also enough to draw one point on the object, without being interrupted with other reading.

The code used to solve this error is shown below:

```
int timer = 2099999999;  
  
while (timer > 0)  
{ timer--;  
}
```

Added to the beginning of Listen() function that receive the data.

5.4.3 TEST 3

This test is performed at larger distance using travelling cars. A car is moving from the out door of the university to Ain_sarah region .

Results

Number of reading= 49 reading

The total cost = 0.49 NIS

The sped is between 20 to 30 meter/second.

Time: approximately 30 minutes.

The drawing path: shown at figure (5.22).



Figure 5.22: test3 result tracked paths

Errors

The errors that we faced previously are disappeared at this test and the results are acceptable and meet the expectations, and more than one car can tracked at the same time, without previous exception.

- o But another error appear on the server which is occur during process of movement, especially when we try to perform zoom operation, and it's the same as error2 but in different location of code, in the cutForm at line:

```
g.DrawImage(mapbmap, 0, 0, region, GraphicsUnit.Pixel);
```

because when we try to perform zoom in operation, we use the same object we darw on.

Modifications

The following modification is done to solve the error and solve the challenges: at server side, and to solve its error of perform zooming at the same time we draw on the map, we solve it by using try and catch, for this case as shown in the following code that added to the DoZoom(function), where when such exception error occur it will be caught, to allow the point to be drawn, then do zoom without error :

```
try
{
    .
    .
    .
}
```

```
catch (Exception error2) { ;}
```

5.4.4 TEST 4

At this test two cars are active in the system at the same time, start moving from different points. The first car start moving from the outdoor of university to Ein_sara again, and the second car moving from Al_haras region to Dowar Al_sehah region.

The system accept the two cars successfully and draw their paths accurately as shown an figure(15.23).



Figure 5.23: test4 result tracked paths

Results

First car

Number of readings:40

Cost =0.40 NIS

speed between 20 and 25
meter/second

Time: approximately 25 minutes.

Second car

Number of readings:35

Cost= 0.35 NIS

speed between 20 and 30
meter/second

approximately 20 minutes.

5.4.5 TEST 5

At this test we use one car, to test the use of modes during tracking, where the user has the ability to select which mode of tracking he want. Where two types are available, one for walking and other for driving.

The car start with driving mode from region near university, then at the outdoor near building A, the mode converted to walking.

The paths of movements as shown in figure(15.24).



Figure 5.24: test5 result tracked paths

Results

Driving mode

Number of reading:26

Cost =0.26

Time: approximately 10minutes.

Walking mode

Number of reading:15

cost = 0.15

approximately 5minutes

Speed: between 20 and 25 meter/second
meter/second

Speed: between 1 and 2

6.1 Overview

In this chapter, we will mention what we achieved in this project and the conclusion for all things that we have done, also we will talk about the challenges that we faced and ending with recommendation needed for the future work.

6.2 Conclusion and Achievement

Almost all the goals of our system have been achieved. In this section the main achievements of the system are discussed and the ways of achieving it.

- We built a mobile application to give the accurate location, of the travelling car and send it to the main server.
- We built desktop software dealing with maps and received location and plotting the coordinates accurately on it.
- We built an accessible database for car drivers information and travelling history, that can be retrieved any time.
- We calculate the speed of the traveling cars, and display it on the screen according to any change, during the tracking.

6.3 Challenges

At this project we have faced some critical points that still need a solution for them. Such as:

- In the system implementation, we used a small map showing part of Hebron city , because we couldn't use a larger one with higher resolution.

- And it will be better if we can use a map with names of places, cities and streets added on it.
- Using the mobile application for a long time consumes large amount of power, so we get battery Restrictions.
- The approximation of meters related to the selected value, differ in some cases according to the state of GPS system.

6.4 Future work Recommendation

In this project, there are some ideas that could be done or added to improve its performance, or add some capabilities, some techniques that are efficient and meet user needs. Some of these ideas are mentioned below:

- System could be improved by adding a mechanical information about the state of the car by testing the internal structure and explore it. This improvements need a mechanical engineering knowledge.
- System could be improved to let all cars at the system know the locations of each other by sending them a photo.
- System can be improved by sending instructions to cars to go to a specific place.

REFERENCES:

[1] ECE Live Projects, "GPS and GSM based Vehicle Tracking System".

Available at

<http://www.mycollegeproject.com/GPS%20and%20GSM%20based%20Vehicle%20Tracking%20System.html>.

[2] Pallavan Transport Consulting Services, Ashok Leyland, Harita Infoserve Ltd, "Vehicle Tracking and Control Systems".

Available at

<http://www.simbaproject.org/download/india/Presentation%20and%20Feedback/TTS/IIT%20Bangalore.pdf>

[3] Source forge team, "Open GPS Tracking System".

Available at <http://opengts.sourceforge.net/>

[4] 8051 projects, Astra Telematics Limited telematics hardware supplier, "Vehicle Tracking System using GPS and GSM modem".

Available at <http://www.8051projects.info/content/projects/7-vehicle-tracking-system-using-gps-gsm-modem.html>

[5] M. Kennedy, , "The Global Positioning System and GIS An Introduction", fanrica, United State of America, 1996.

[6] Image from Top gps reviews.

Available at <http://topgpsreviews.net/wp-content/uploads/2010/08/gps-satellite-orbits.jpg>

[7] Wikipedia, the free encyclopedia, "World Geodetic System".

Available at http://en.wikipedia.org/wiki/World_Geodetic_System

[8] D. Bowler, A. Mayne, D. McNally, T. Wakefield, Introduction To Mobile Communication Technology Services Markets, Auerbach Publication Informa, New York, 2007.

[9] Image from Bundesamt.

Available at

https://www.bsi.bund.de/SharedDocs/Bilder/DE/BSI/Publikationen/GSM/gsm_e2_jpg.jpg?__blob=normal&v=2

[10] Search mobile computing, "GPRS".

Available at <http://searchmobilecomputing.techtarget.com/definition/GPRS>

[11] P. Calduwel Newton, DR. L. Arockiam, and Tai-hoon Kim, International Journal of Advanced Science and Technology, "A Quality of Service Strategy to Select Coding Schemes

in General Packet Radio Service System". Available at

<http://www.sersc.org/journals/IJAST/vol7/1.pdf>

[12] C. Kozierek, The TCP/IP guide, Available at

http://www.tcpipguide.com/free/t_TCPIPHypertextTransferProtocolHTTP.htm

[13] Wikipedia, the free encyclopedia, "Mobile operating system". Available at

http://en.wikipedia.org/wiki/Mobile_operating_system

[14] From Linux to Android, Android Internals for Linux Developers,

"Introduction to Android Architecture". Available at

<http://technologeeks.com/Courses/Android-Excerpt.pdf>

[15] B. Venners, artima developer, "Introduction to Java's Architecture".

Available at

<http://www.artima.com/insidejvm/ed2/introarchP.html>

[16] B. Evjen, J. Glynn, C. Nagel, M. Skinner, K. Watson, "C# 2008", Wiley Publishing Inc, 2008.

[16] GPS-practice-and-fun, "Different types of GPS receivers". Available at

<http://www.gps-practice-and-fun.com/gps-receivers.html>

[18] Android compare, "Why Android". Available at

<http://androidcompare.com/#ixzz1h91s3wrS>

[19] Wikipedia, the free encyclopedia, "Google Earth". Available at

http://en.wikipedia.org/wiki/Google_Earth

[20] Palearth, Available at

<http://www.palearth.ps/?q=ar/aboutus>

[21] Wikipedia, the free encyclopedia, "Yahoo Maps". Available at

http://en.wikipedia.org/wiki/Yahoo!_Maps
