



VAASAN AMMATTIKORKEAKOULU  
VASA YRKESHÖGSKOLA  
UNIVERSITY OF APPLIED SCIENCES

Tinbit Belay

# GPS TRACKER DATA ANALYSIS PLATFORM

Technology and Telecommunication

2015

## ABSTRACT

Author	Tinbit Belay
Title	GPS Tracker Data Analysis Platform
Year	2015
Language	English
Pages	58
Name of Supervisor	Dr. Ghodrat Moghadampour, Johan Dams

---

This thesis work offers GPS tracking solutions which provides detailed reporting on the tracked assets while significantly reducing costs. Despite the fact that GPS trackers are used as a security mechanism.

The objective of this thesis was to develop a web platform for gathering the behavioral aspects of the tracked assets. The thesis work was designed and implemented using Groovy-based web application framework, which integrates smoothly with the JVM. The user interface was implemented using CSS3 and HTML5 and JavaScript for responsive structure. Further technologies used in the application are MySQL to persist the data, jQuery to simplify the client side scripting and JSON for data exchange.

The application was designed for WRD Systems Ltd which specializes in developing a GPS tracking platform that offers development of bespoke tracking hardware and software to suit any application.

All features mentioned above were achieved; the application has been demonstrated and reviewed by the company representative. Regarding the performance and completeness of data reporting, positive feedback has been given to the project.

## CONTENTS

### ABSTRACT

1	INTRODUCTION .....	7
1.1	WRD Systems .....	7
1.1.1	Intelligent Asset Managements .....	8
1.2	Project Objectives and Outcomes .....	8
2	RELEVANT TECHNOLOGIES.....	10
2.1	HTML5 .....	10
2.2	CSS .....	10
2.3	JavaScript.....	11
2.4	JQuery .....	11
2.5	SQL statement.....	12
2.6	Groovy .....	12
2.7	Grails and the MVC development model .....	13
2.7.1	Grails Architectural description .....	14
2.8	Groovy/ Grails Tool Suite.....	15
2.9	Git .....	16
2.10	MySQL .....	17
3	APPLICATION DESCRIPTION .....	18
3.1	General Description .....	18
3.2	Quality Function Deployment.....	18
3.3	Application Architectural design .....	19
3.4	Functional Requirements .....	21
3.4.1	Login .....	22
3.4.2	View Statistics of Tracked Assets.....	22
3.4.3	Search Tracker .....	23
3.4.4	View Tracked Assets Reports .....	24
3.4.5	Social integration .....	24
3.4.6	Register users .....	25
3.4.7	Search user .....	26
3.5	Class diagram.....	26

3.6	Sequence diagram .....	27
4	DATABASE AND GUI DESIGN .....	31
4.1	Design of the database .....	31
4.2	GUI design .....	32
4.2.1	Login view .....	32
4.2.2	Home page view.....	33
4.2.3	Map view.....	34
4.2.4	Reports view .....	34
4.2.5	Statistics view .....	35
4.2.6	Social view .....	36
4.2.7	Admin view .....	37
5	IMPLEMENTATION AND DEPLOYMENT.....	39
5.1	Setup Windows Development Environment.....	39
5.2	Database configuration .....	40
5.3	Data Response Structure .....	42
5.4	Reports .....	43
5.5	SQL Server stored procedure.....	45
5.6	Application Deployment.....	53
6	TEST CASES .....	55
6.1	Application Testing.....	55
6.2	Completeness of data reporting .....	55
6.3	Performance Testing .....	55
7	CONCLUSION .....	57
7.1	Future work.....	57
8	REFERENCES .....	58

**LIST OF FIGURES, TABLES AND CODE SNIPPETS**

<b>Figure 1.</b>	Grails Architectural design	p. 15
<b>Figure 2.</b>	Git workflow	p. 16
<b>Figure 3.</b>	Application architecture	p. 20
<b>Figure 4.</b>	Use case diagram of IAM GPS Tracker	p. 21
<b>Figure 5.</b>	Class diagram screenshot	p. 27
<b>Figure 6.</b>	Login process sequence diagram	p. 28
<b>Figure 7.</b>	User registration sequence diagram	p. 28
<b>Figure 8.</b>	Search tracker sequence diagram	p. 29
<b>Figure 9.</b>	Reports sequence diagram	p. 29
<b>Figure 10.</b>	Status sequence diagram	p. 30
<b>Figure 11.</b>	Entity-relationship diagram	p. 32
<b>Figure 12.</b>	Login view screenshot	p. 32
<b>Figure 13.</b>	Home page view screenshot	p. 32
<b>Figure 14.</b>	Map view screenshot	p. 33
<b>Figure 15.</b>	Reports view screenshot	p. 34
<b>Figure 16.</b>	Statistics view screenshot	p. 35
<b>Figure 17.</b>	Social view screenshot	p. 36
<b>Figure 18.</b>	Admin view screenshot	p. 37

<b>Table 1.</b>	Use case “Login”	p. 22
<b>Table 2.</b>	Use case “View Statistics of Tracked Asset”	p. 23
<b>Table 3.</b>	Use case “Search Tracker”	p. 23
<b>Table 4.</b>	Use case “View Tracked Assets Report”	p. 24
<b>Table 5.</b>	Use case “Social integration”	p. 25
<b>Table 6.</b>	Use case “Register users”	p. 25
<b>Table 7.</b>	Use case “Search users”	p. 26
<b>Code snippet 1.</b>	Grails installation and Setup environment test	p. 38
<b>Code snippet 2.</b>	Sample DataSource.groovy file	p. 40
<b>Code snippet 3.</b>	Sample BuildConfig.groovy	p. 41
<b>Code snippet 4.</b>	A Data table in a database	p. 41
<b>Code snippet 5.</b>	String parsing	p. 42
<b>Code snippet 6.</b>	Reports table	p. 43
<b>Code snippet 7.</b>	PROCEDURE sp_DistanceCalculator	p. 44
<b>Code snippet 8.</b>	get_distance_in_miles_geo_locations function	p. 45
<b>Code snippet 9.</b>	PROCEDURE sp_GetLastCalculatedTime2	p. 46
<b>Code snippet 10.</b>	Search Tracker	p. 46
<b>Code snippet 11.</b>	Getchart method	p. 47
<b>Code snippet 12.</b>	GSP page using visualization	p. 48
<b>Code snippet 13.</b>	Google visualization plugin	p. 48

<b>Code snippet 14.</b> Status model	p. 49
<b>Code snippet 15.</b> Security service	p. 49
<b>Code snippet 16.</b> lookupUser method	p. 49
<b>Code snippet 17.</b> currentUserTimeline method	p. 50
<b>Code snippet 18.</b> Status index method	p. 50
<b>Code snippet 19.</b> Update the status	p. 50
<b>Code snippet 20.</b> Manage user information	p. 51
<b>Code snippet 21.</b> Show method	p. 51
<b>Code snippet 22.</b> User GSP page	p. 52
<b>Code snippet 23.</b> Source code revision management	p. 53

**LIST OF ABBREVIATIONS**

API	Application Programming Interface
CSS	Cascading Style Sheets
HTML	HyperText Markup Language
MVC	Model View Controller
GUI	Graphical User Interface
URI	Uniform Resource Identifier
WRD	World Systems
GPS	Global Positioning System
IAM	Intellegente Asset Management
RMC	Recommended minimum data for gps
ERD	Entity-Relationship Diagram
SQL	Structured Query Language
IDE	Integrated Development Environment
ORM	Object Relational Model
JVM	Java Virtual Machine
IT	Information Technology
UAV	Unmanned aerial vehicle
GGTS	Groovy/ Grails Tool Suite
RDMS	Relational database management system

# 1 INTRODUCTION

GPS tracking devices are used for tracking cars, pets, bikes and other valuable assets. Tracking devices deliver real time information on location. These trackers provide valuable protection in the event of asset theft and recovery.

WRD Systems develops one of the smallest GPS trackers available in the market today. These systems are available as standalone units, or integrated into other products and projects. Real-time GPS updates with 24/7 access offer convenient tracking solution allows to monitor the status of assets at all times. While these trackers are used as a security mechanism in many of these; this thesis work provide an efficient, effective asset tracking system management to products and projects.

The primary focus of this thesis work is to use GPS trackers to gather behavioral aspects of the tracked assets such as bikes, cars, pets and other valuable assets. One particular example is when using the tracker to track bikes as a part of a bike rental project. It is not only useful to know where each bike (asset) is at any time, but also aspects such as total distance the bike has been driven, speed data, calories being burned, carbon offset as well as disuse of assets can be tracked. This data can offer insights and facts about how successful a project is at an assessment phase, what the actual impact of the project is, as well as marketing data with social network integration.

## 1.1 WRD Systems

This application was developed for WRD Systems Ltd which is a global technology company established in 2007. The company develops one of the smallest GPS trackers available in the market today. These systems are available as stand-alone units, or integrated into other products and projects.

A Vast number of clients of the company come from well-known companies such as BADGE, FlatTurtle, MIRKA, NordicAurum, NORDITEK, SEDC, UKAV, Vaa-saETT, and VENTUS. /8/

The company employs over 14 IT experts, and is committed for communities to operate in is one of the founding main beliefs, from North Africa to South America and across Europe. WRD works in partnership with local communities to create jobs and develop sustainable business models. /8/

### **1.1.1 Intelligent Asset Managements**

The company provides Intelligent Asset Management (IAM) GPS tracking platform that offers development of bespoke tracking hardware and software to suit any application. It can locate the position and speed of an asset anywhere in the world with an accuracy of up to 5 meters. This product is currently in use in countries such as Algeria and USA.

The project ARGUS which aims to deliver inexpensive unmanned aerial vehicle (UAV) platform is currently under development in Finland. /8/

## **1.2 Project Objectives and Outcomes**

The company is looking to design and implement a new web platform which offers insights and hard facts about the completeness of data reporting, extensible architecture, and ability to scale as datasets grow in size. It is not uncommon to be doing data analysis on more than 4000 trackers in a project, each one sending data updates on location, speed, heading, etc. at five minute intervals. This requires some design decisions to be made on what data can be accessed in real time, and what data needs to be updated on hourly, daily, weekly and monthly basis. It also means that performance analysis is a part of the implementation. The application is fully deployable on Linux based servers. The generated web interface should naturally be cross platform, meaning the program may run on as many as all existing platforms such as on Microsoft Windows on the x86 architecture, Mac OS X on either the Power PC or x86 based Apple Macintosh systems and Linux on the x86 architecture.

The features of this application are:

- A map with markers indicating tracker location with pop-up info-windows showing speed, heading, date and time, battery status, and signal strength.

- Ability to define geofence for individual trackers.
- Ability to generate heat maps for tracker or tracker groups.
- Graphs to show data such as top x most used assets, historical distance (Distance in Miles the last seven days and Distance in Miles the last seven months) / speed data, speed vs distance graphs.
- Statistics such as top speed, average speed, active trackers, amount of incoming data points.
- Advanced data analysis using custom database queries for data mining purposes.
- User account and access management.

## **2 RELEVANT TECHNOLOGIES**

The application functionality will be implemented using Groovy programming language and JavaScript. Grails, which is an open source web application framework, will be used. Groovy is an agile and dynamic language built to run on the Java Virtual Machine (JVM).

To style the layout and appearance of the application, the user interface will be implemented using HTML and CSS.

After parsing the RMC GPS sentence sent by the GPS tracker, the data will be managed by using procedures. This will be performed in MySQL database.

The database system that will be used in this application is MySQL. Stored procedures are implemented in SQL programming language.

### **2.1 HTML5**

Hyper Text Markup Language (HTML) is a markup language in use for creating web documents. HTML describes the structure and layout of a web pages by using defined elements and attributes, such as tables, paragraphs, headings, links etc.

The current most commonly used version is HTML5, with new behaviors and strong multimedia support for computers and mobile devices. Furthermore, HTML5 has features such as the ability to handle data when the application is no longer connected to the internet, geolocation, the ability to detect and work with the user location, as well as offline storage. /11/

### **2.2 CSS**

Cascading Style Sheets (CSS) is a language for describing the layout on a web page written in a markup language.

CSS is used to format web documents that can be embedded in a web document as an internal component or external as a .css file extension.

### 2.3 JavaScript

JavaScript is an object oriented (OO) dynamic language that has methods, standard built-in objects, and types and operators. It is designed to run as a scripting language in a host environment for communicating with the outside world. /2/

Many structures and syntax applied to JavaScript come from the Java and C languages. JavaScript is Prototype-based programming that does not have classes, instead the class functionality is accomplished by object prototypes (equivalent to inheritance in class-based languages). Its functions are objects which give functions the capacity to hold executable code and be passed around like any other object.

Objects created in JavaScript can be utilized and reused for different purposes. JavaScript provides support to user interfaces and dynamic websites. Furthermore, it supports different programming styles such as object-oriented, imperative and, functional styles. The significant capabilities of JavaScript are variable parameter lists, function variables, runtime object construction, object introspection and dynamic script creation. /9/

### 2.4 JQuery

JQuery is built on top JavaScript, a rich and expressive language in its own right. As quoted on the official website, “It is a fast and concise JavaScript Library that simplifies HTML document traversing, animating, event handling and Ajax interactions for web development”. /5/

JQuery increases the productivity of the developer by enabling them to achieve critical User Interface (UI) functionality without writing hundreds of lines of codes, to improve the performance of the application, to develop most browser compatible web pages, and to implement UI related critical functionality.

## 2.5 SQL statement

Stored Procedures are a batch of SQL statements that can be executed in a variety of ways. It is used to centralize data access logic into a single place that is then easy for Database to optimize. The advantages of using stored procedures are:

- Maintainability – Since scripts are in one location, updates and tracking of dependencies based on schema changes becomes easier.
- Testing – They can be tested independent of the application.
- Isolation of Business Rules – Having Stored Procedures in one location means that there is no confusion of having business rules spread over potentially disparate code files in the application.
- Optimization or Speed – Stored procedures are cached on the server. Execution plans for the process are easily reviewable without having to run the application.
- Utilization of Set-based Processing – The power of SQL is its ability to quickly and efficiently perform set-based processing on large amounts of data; the coding equivalent is using iterative looping, which is generally much slower.
- Security – Limit direct access to tables via defined roles in the database. It provides an “interface” to the underlying data structure so that all implementation and even the data itself is shielded. /6/

Securing just the data and the code that accesses it is easier than applying that security within the application code itself.

## 2.6 Groovy

Groovy is a superset of Java, which means it uses the strength of Java and new structures inspired by other dynamic languages such as Ruby, Python and Perl. This gives Grails a strong base to stand on. It is compiled to Java bytecode. Similarly, Grails projects that are written in Groovy can be deployed to Java application servers such as Tomcat, Glassfish and JBoss. /2/

Groovy is designed to be both source-compatible and binary compatible with Java, in which a Groovy class could extend a Java class or implement a Java interface, and that a Java class could extend a Groovy class or Implement a Groovy interface.

Since Groovy is a pure OO language, it supports all object-oriented (OO) programming concepts that are well-known to java developers: classes, objects, interfaces, polymorphism, and others. It is possible to fully organize the code into classes and packages and still mix scripts with classes, because the Groovy compiler can convert such scripts into Java classes.

## **2.7 Grails and the MVC development model**

Grails is the leading next-generation Groovy-based framework, is built on widely-held Spring and Hibernate frameworks.

Furthermore, Grails is an open source web development framework that packages best practices (such as convention over configuration and unit testing) with the best-of-the-best open source application frameworks (such as Spring, and Hibernate). Together with the productivity of the Groovy scripting language, everything runs on top of the robust Java and Java EE platform. /13/

Grails integrates smoothly with the Java Virtual Machine, allowing to be immediately productive whilst providing powerful features, including integrated ORM, Domain-Specific Languages, runtime and compile-time, meta programming and Asynchronous programing. There is a command line interface available for Grails, which can be used for e.g. installing plug-ins or generating an application structure. It can also be used to run self-made scripts. /13/

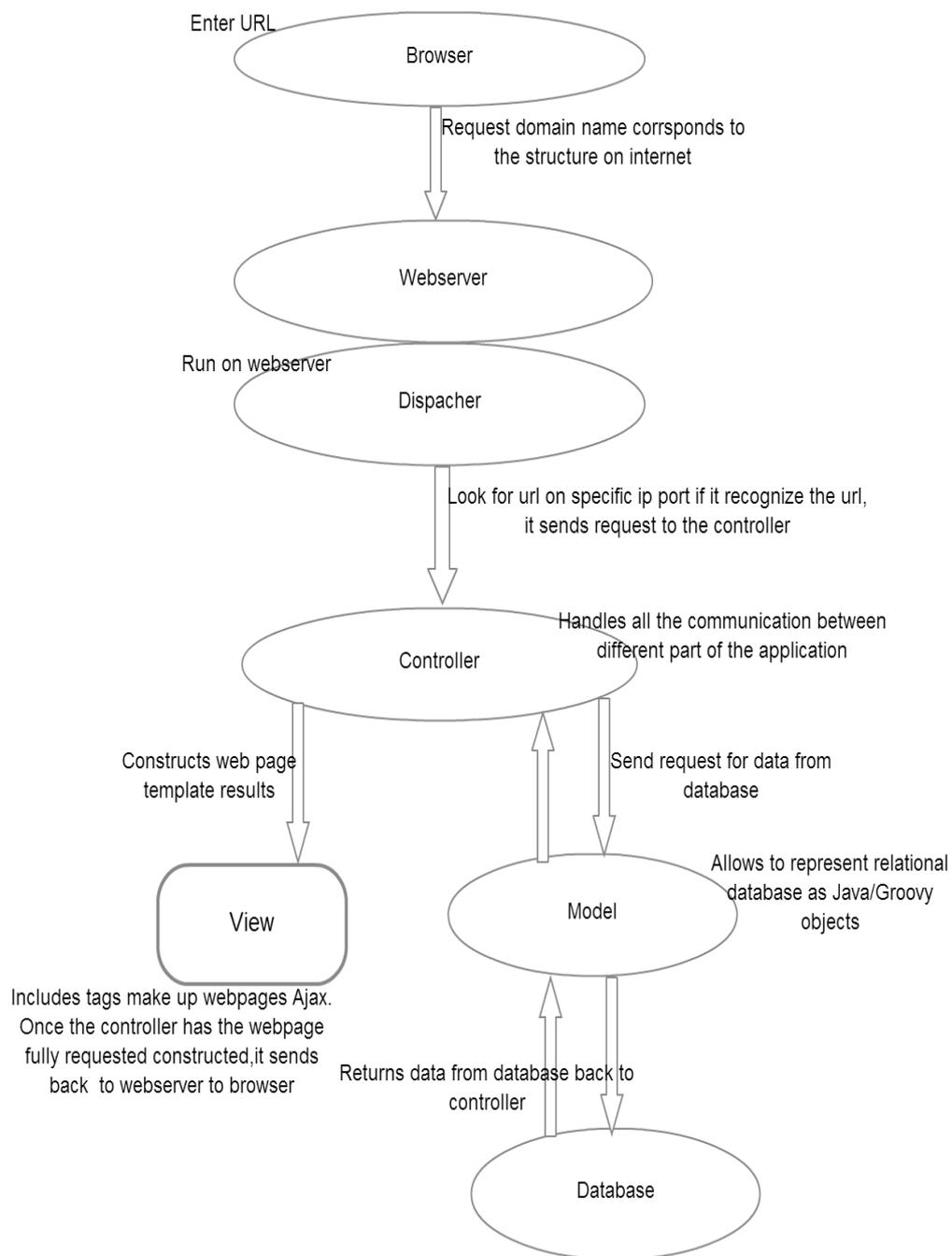
Grails comes with the Grails console and is an extended version of the regular Groovy console. The Grails console lets one create projects, domain models, controllers and views based on the domain model, and much more. /13/

Moreover, Grails allows the use of the MVC pattern software architecture. The MVC development model simply encourages separation of concerns in three functionality are as:

- Model – handles data representation and operations. Classes that represents entities and business logic or functionalities, collectively represents the model.
- View – handles how data is viewed or presented by the user.
- Controller – handles the code that links a user to the system. It receives a request from the user, prepares data and invokes proper business logic, and then returns to the view that represents the result of user actions.

### **2.7.1 Grails Architectural description**

Grails architectural design and description is shown below in Figure 1.



**Figure 1.** Grails Architectural design.

## 2.8 Groovy/ Grails Tool Suite

The Groovy/ Grails Tool Suite (GGTS) is the best development tool for Groovy and Grails. It provides the full Eclipse-powered development for building applications; commonly used components are pre-installed. /17/

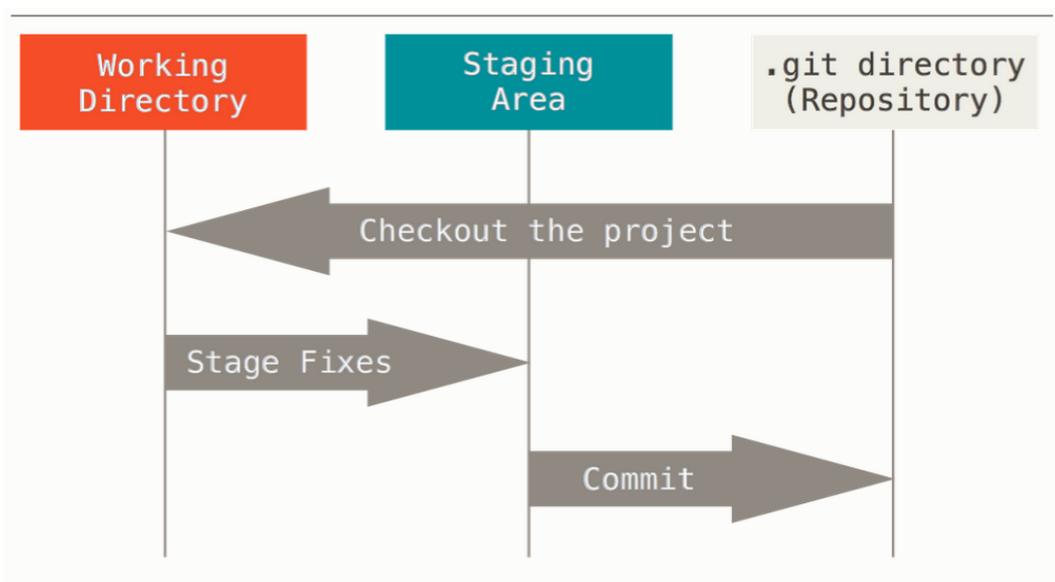
Furthermore, GGTS supports all the most common Java EE application servers such as vFabric tc Server. It provides a ready to use Integrated Development Environment (IDE) integration for Cloud Foundry, including real debugging in the cloud.

GGTS supports an applications pursuing to local, virtual and cloud-based servers and provides support for all the most common Java application servers such as vFabric tc Server for the latest versions of Groovy and Grails. /2/

## 2.9 Git

Version control is a system that records changes to a file. Git is an open source distributed version control system designed to handle from small to a very large projects with speed and efficiency.

Git is used as source code revision management for the entire project. It has three important states that the project can reside in: commit, where the data is safely stored in a database; modified, where the data has been edited but have not committed it to the database yet, a stage in which the modified file in its current version to go in to the next commit. The workflow of Git is shown below in Figure 2.



**Figure 2.** Git workflow. /14/

The working directory is where files are pulled out of the compressed database and placed on disk for modification.

The Git directory is where Git stores an object database and the metadata which will be copied and clone a repository from another computer.

The staging area is a file that stores data which are not committed yet.

## **2.10 MySQL**

Relational Database Management System (RDMS) provide relational operators to operate the data stored into the database tables.

MySQL is a RDMS based on Structured Query Language. It is easy to use and operates very fast. It runs on virtually all major platforms and; it can run on many different operating systems such as Windows, Linux, UNIX and Solaris.

### **3 APPLICATION DESCRIPTION**

The current chapter describes the system design decisions to be made, resulting from the requirements gathering and analysis process.

Before implementing a software application its requirements must be gathered and analyzed and then it should be designed. This process provides the client and the developers a clear understanding of the project to be developed.

#### **3.1 General Description**

The main objectives are completeness of data reporting, extensible architecture, and the ability to scale as data sets grow in size. This data analysis is done on more than 4000 trackers, each one sending data updates on location, speed, heading, etc. at 5 minute intervals. This requires some design decisions on what data can be processed in real time, and what data needs to be updated on hourly, daily, weekly and monthly basis. It also mean that the performance analysis is part of the implementation.

#### **3.2 Quality Function Deployment**

The quality function deployment provides a functional definition for the application. The IAM GPS Tracker's project requirements are divided in to three groups based on their priority level. The quality function deployment is as follows:

Normal requirements (must-haves): The most basic features the application must implements are:

- A map with markers indicating tracker location with pop-up information-windows
- Ability to define geofence for individual trackers
- Ability to generate heat maps for tracker
- Graphs to show tracker's data
- The user is able to search the tracked asset

Expected requirements (should-haves) describe important features that should be present for the user:

- The web interface should naturally be cross platform, but should not be built with explicit Internet Explorer compatibility
- Should be fully deployable on Linux based servers
- Statistics of the tracker data
- Social media integration
- Changing heat map gradient, heat map radius and heat map opacity should be user's preference.

Exciting requirements (nice-to-haves): Extra requirements which are not compulsory to be implemented:

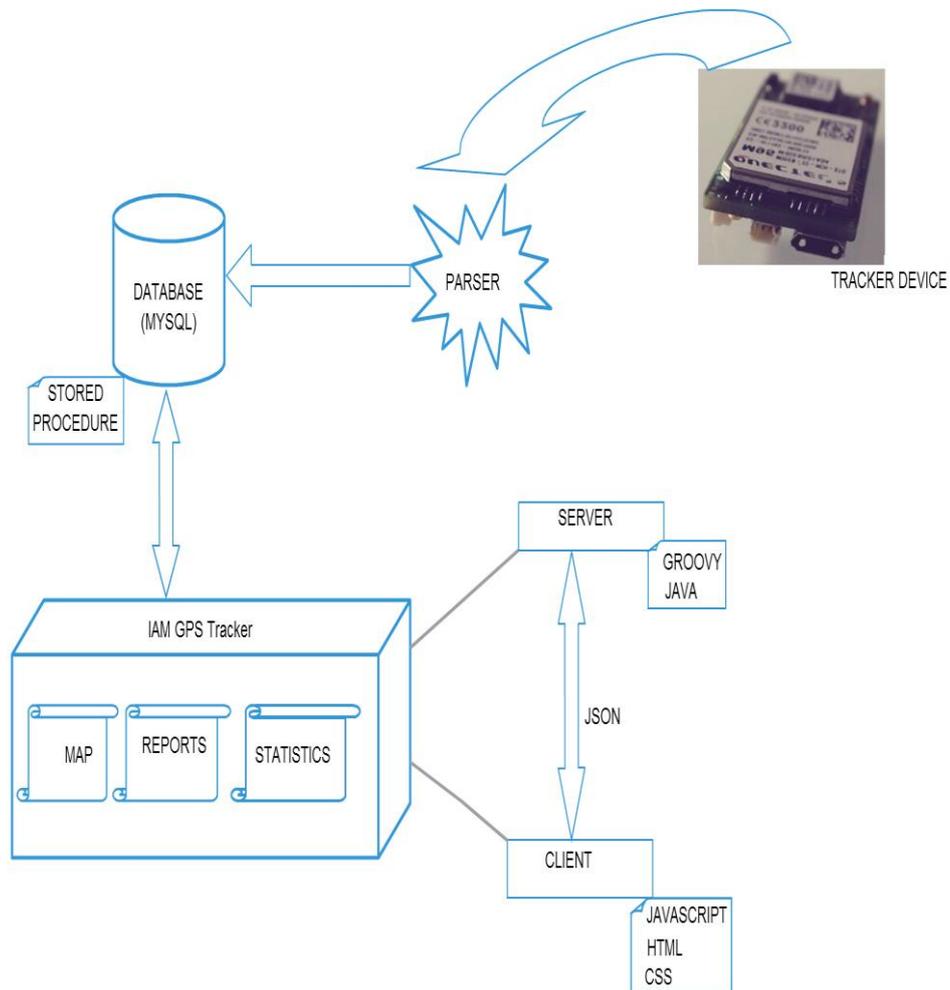
- User account and access management; the user is able to log in to use the IAM GPS Tracker application.
- At the top of map and reporting page there are; total distance of the tracker of the week, top speed, data points from the last 24 hours, active units
- An administrator can register users
- An administrator can view user lists
- An administrator can search user information by username or tracker name

### **3.3 Application Architectural design**

The Recommended Minimum Sentence C, is the National Marine Electronic Association (NMEA) version of the essential GPS pvt (position, velocity, time) data. The data received from the tracker contain the GPS location which include the latitude and longitude, speed and heading of the asset being tracked encoded in a modified NMEA RMC format. /16/

Application architecture diagram can be seen below in Figure 3.

Architectural Diagram

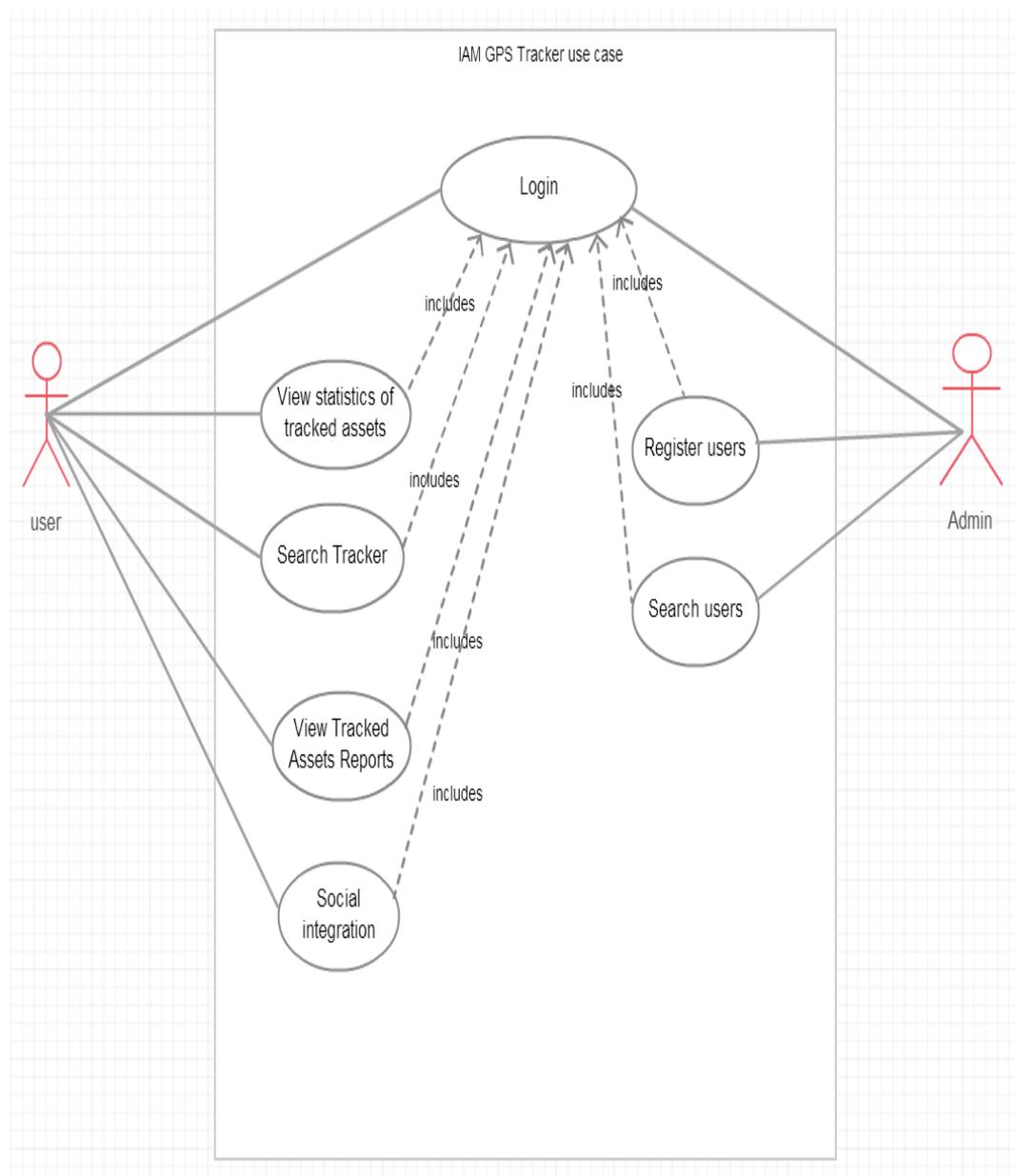


**Figure 3.** Application Architecture.

The tracker sends data with different field representations. A complete data is referred to as a sentence; the detail is described in the implementation section of data response structure. This data will be parsed as string and saved in to a database. Hence, this will be the data that will be used to generate the marker on the map, reports and statistics.

### 3.4 Functional Requirements

The most important required functions can be displayed using the use case diagram. Use case diagrams are used to present a graphical overview of the functionality of a system. The use case diagram of the whole application can be seen in Figure 4.



**Figure 4.** Use case diagram of IAM GPS Tracker.

Figure 4 indicates the functionalities of the user and administrator. In order to access the application for the user and administrator a login process is required. Once

the login process is authenticated, users have the access to view the statistics of the tracked asset, search the tracker based on the criteria (by the tracker name, the start and end date and time) , view reports of the tracked asset total distance in charts, and the access of social view which provides an option to post data automatically.

### 3.4.1 Login

A user can only log in if valid credentials have previously been obtained through the registration process.

In Table 1 the main features of the use case “Login” are described.

**Table 1:** Use case “Login”.

Pre-Conditions	The user must be registered
Input	Username and Password
Description	The user logs in to use IAM GPS Tracker application
Exceptions and errors	Username and/or password not valid
Result or outputs	Access to the functionality of IAM GPS Tracker application

### 3.4.2 View Statistics of Tracked Assets

In Table 2 the main features of the use case “View Statistics of Tracked Assets” are described.

**Table 2:** Use case “View Statistics of Tracked Assets”.

Pre-Conditions	User logged in
Input	-
Description	User can view statistics of tracked asset such as top speed, average speed, amount of incoming data points and total distance of the assets traveled
Exceptions and errors	-
Result or outputs	Information about tracker statistics is displayed

### 3.4.3 Search Tracker

In Table 3 the main features of the use case “Search Tracker” are described.

**Table 3:** Use case “Search Tracker”.

Pre-Conditions	User logged in
Input	Tracker name , date and time and button click to perform the search functionality
Description	User makes a search for owned tracked asset to view information on map; To be able to see the tracked asset information(location), ability to define geofence, to generate heat map (to be able to change heat map opacity, radius and gradient)
Exceptions and errors	Invalid tracker name or missing data

Result or outputs	Search functionality is performed: map with marker indicating tracker with info-windows showing speed, heading, latitude, longitude etc. will be displayed
-------------------	--

#### 3.4.4 View Tracked Assets Reports

In Table 4 the main features of the use case “View Tracked Assets Reports” are described.

**Table 4:** Use case “View Tracked Assets Report”.

Pre-Conditions	The user logged in
Inputs	A Mouse hovers on graphs for detailed information
Description	Graphs showing tracked asset total distance for past 7 days of a week and total distance in the past 7 months
Errors and Exceptions	Cannot retrieve data

#### 3.4.5 Social integration

In Table 5 the main features of the use case “Social integration” are described.

**Table 5:** Use case “Social integration”.

Pre-Conditions	User logged in
Inputs	Write message
Description	Option to automatically post data on Twitter; i.e. “Biked x Miles Today!”
Errors and Exceptions	-
Results and outputs	Graphs showing tracked asset historical distance

### 3.4.6 Register users

In Table 6 the main features of the use case “Register users” are described.

**Table 6:** Use case “Register users”.

Pre-Conditions	Administrator must login in to the system
Input	All the necessary user data: tracker name, username, password, city and state
Description	Users registers to be able to access IAM GPS Tracker application
Errors and Exceptions	Missing information or Registration must be completed or Duplicate user registration
Results and outputs	Login page is loaded; user can login with the registered username and password

### 3.4.7 Search user

In Table 7 the main features of the use case “Search user” are described.

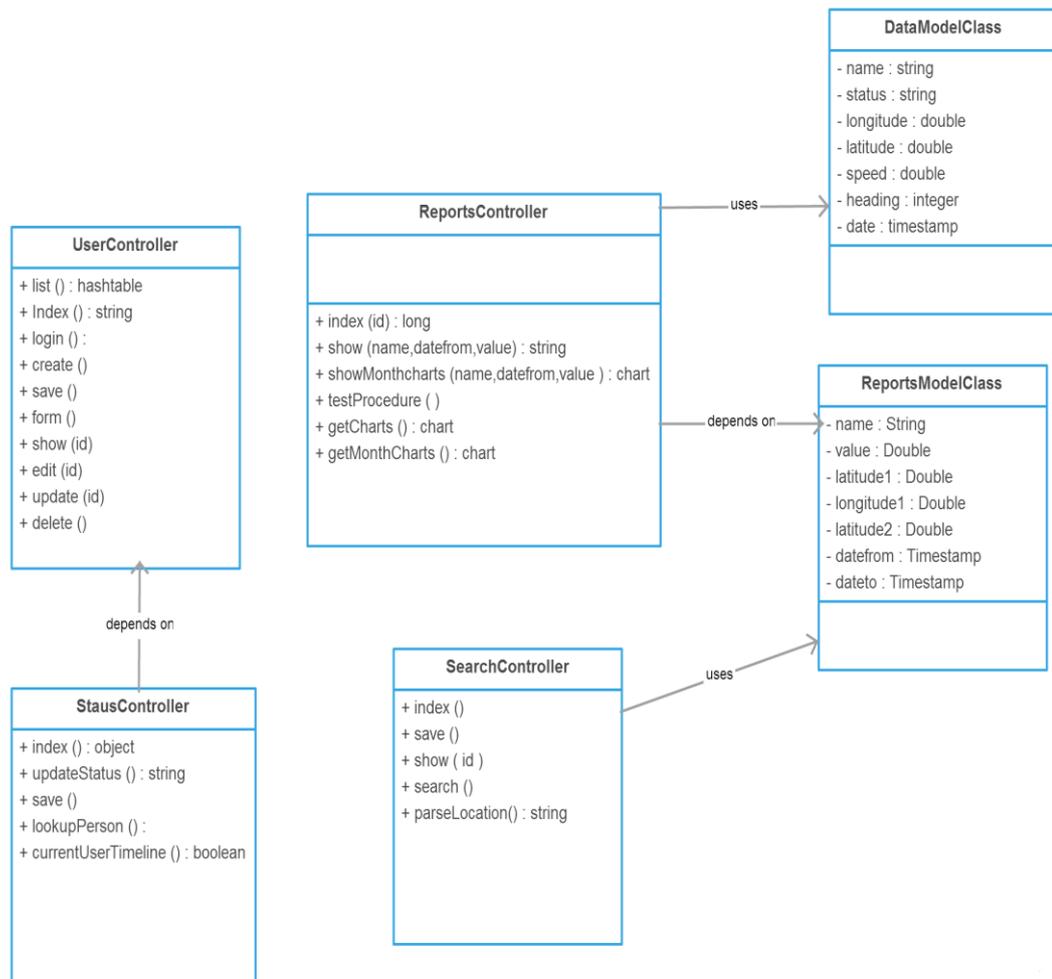
**Table 7:** Use case “Search users”.

Pre-Conditions	Administrator logged in
Input	Users name or Tracker name
Description	To view individual user information
Errors and Exceptions	Not found
Results and outputs	All the necessary user information is displayed

The tracker sends data with different field representations. This data will be parsed as a string and saved in to the database. Hence, this will be the data that will be used to generate the marker on the map, reports and the statistics.

### 3.5 Class diagram

A class diagram describes the types of objects in the system and the various kinds of static relationships that exist among them. Class diagrams also show the attributes and operations of a class and the constraints that apply to the way objects are connected. The class diagram of the application can be seen in Figure 5.

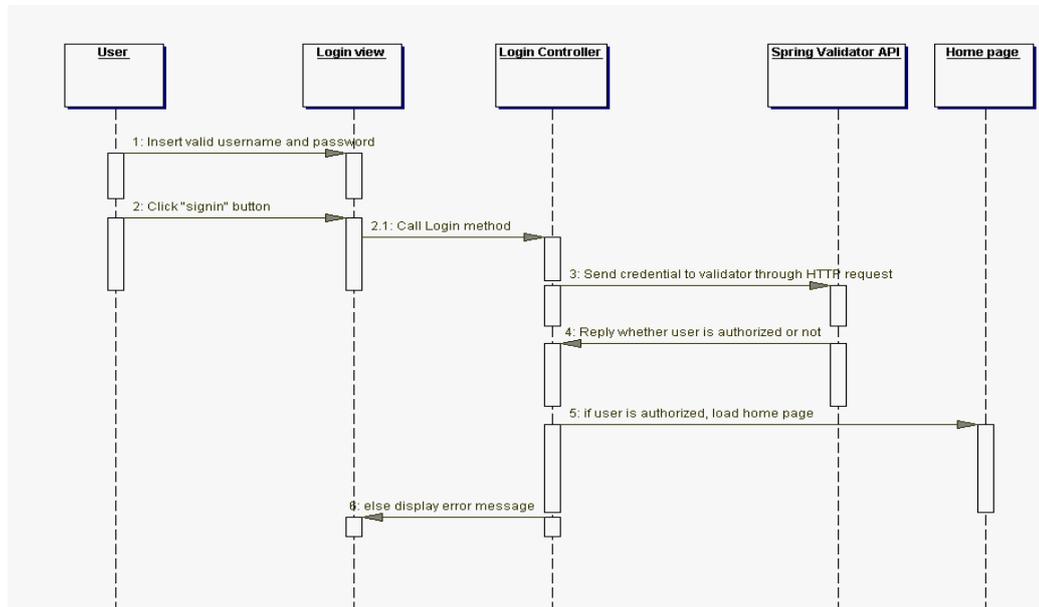


**Figure 5.** Class diagram screenshot.

### 3.6 Sequence diagram

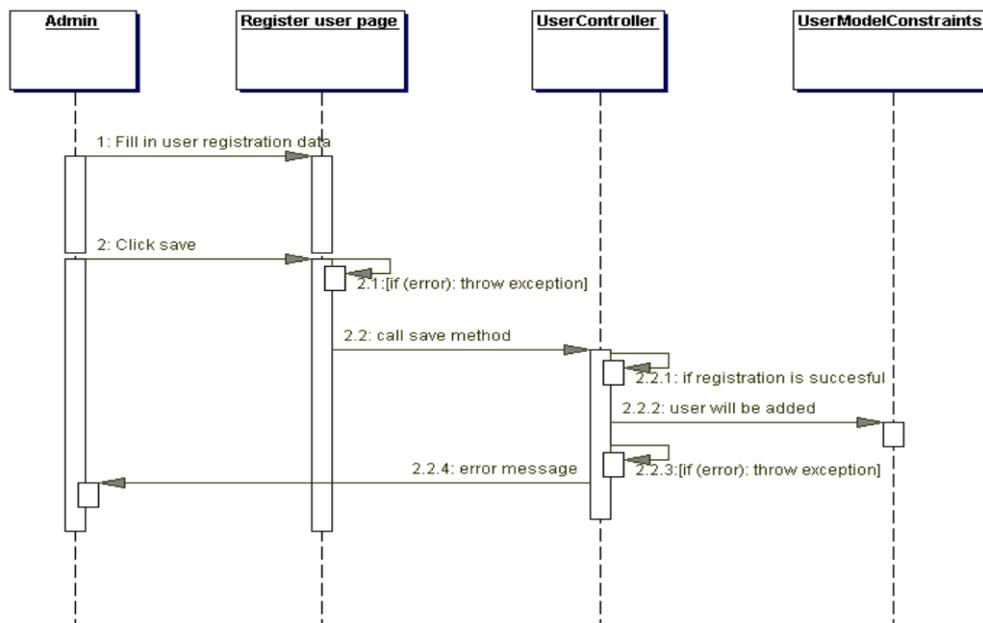
A sequence diagram is a graphical way to illustrate scenarios. Sequence diagrams are used to describe the sequence of events. Sequence diagrams are used in conjunction with class diagrams to capture most of the information about a system.

Figure 6 presents the login sequence diagram of IAM Tracker application. It can be seen that the username and password are validated through Spring validator API.



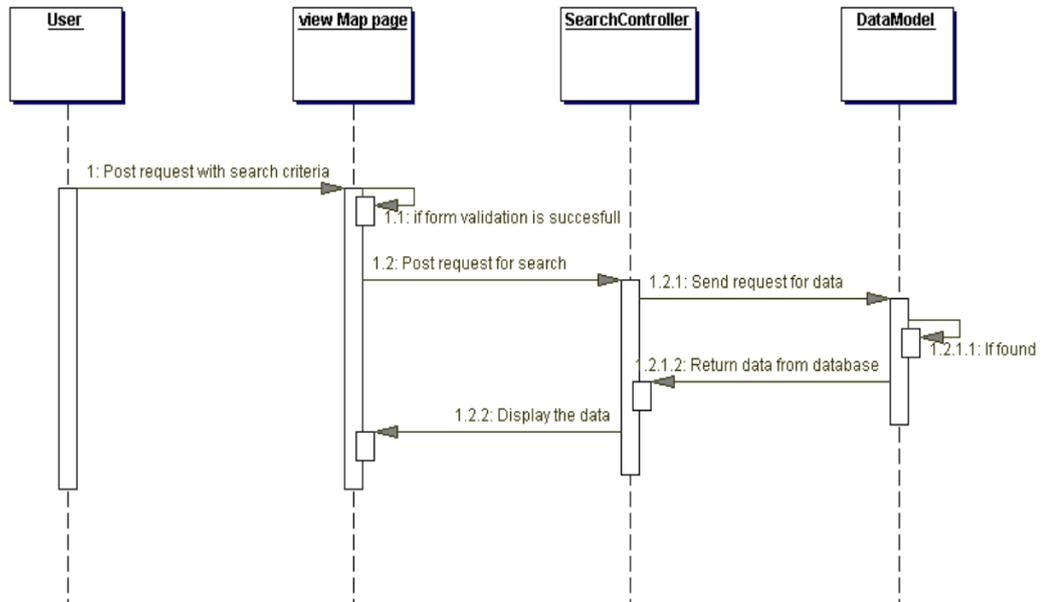
**Figure 6.** Login process sequence diagram.

Figure 7 represents the user registration sequence diagram. It can be seen that the administrator needs to fill all the required user data and save it to a database.



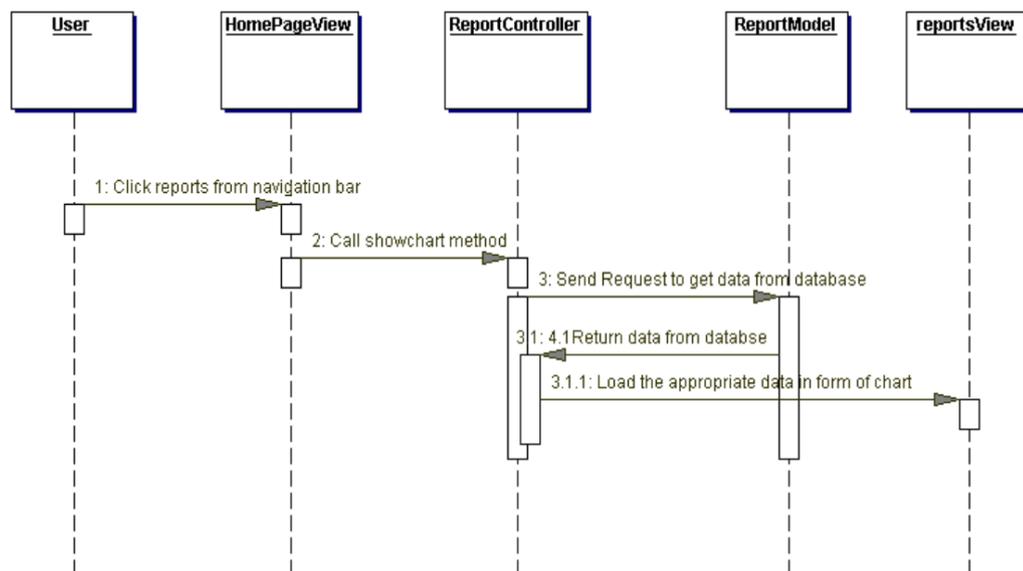
**Figure 7.** User registration sequence diagram.

Figure 8 represents search tracker sequence diagram. It can be seen that tracker data is look at by the tracker name and between the date and time from a database.



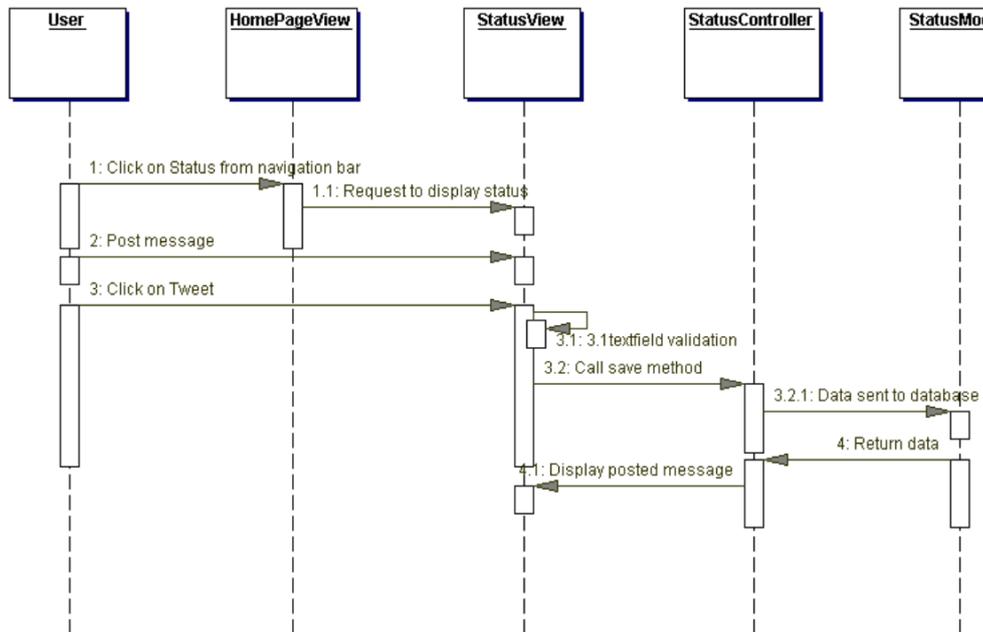
**Figure 8.** Search tracker sequence diagram.

Figure 9 represents Reports sequence diagram. It can be seen that report model handles the communication between a database and to the report controller.



**Figure 9.** Reports view sequence diagram.

Figure 10 represents status sequence diagram. It can be seen that once a user logged in to the system all post will be saved in to the status model and displayed to a user.



**Figure 10.** Status sequence diagram.

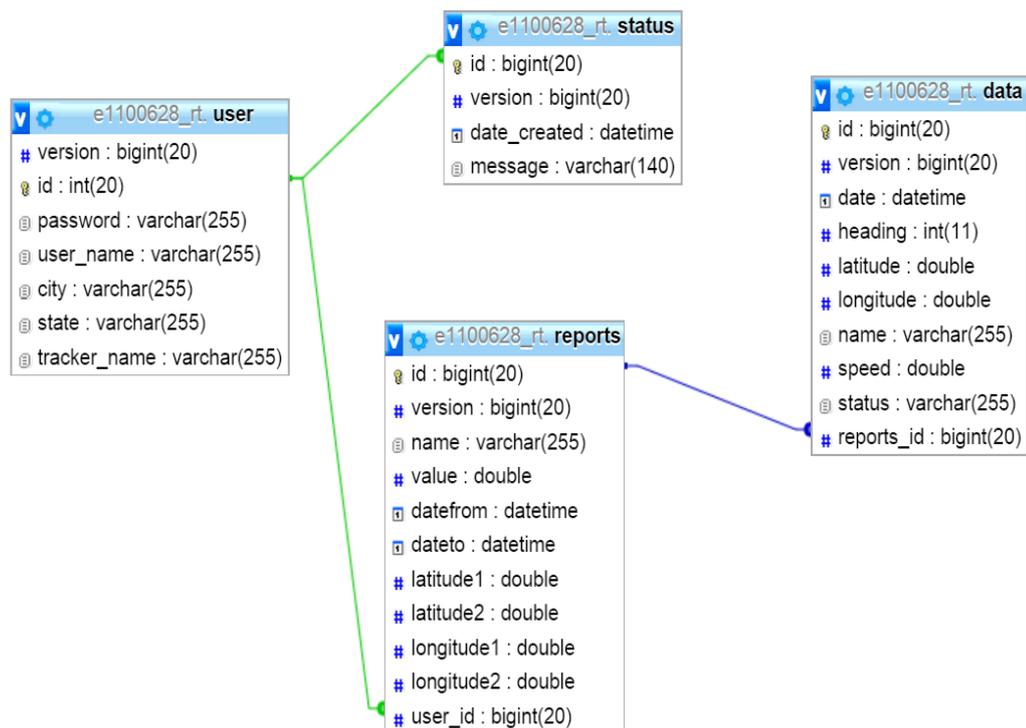
## **4 DATABASE AND GUI DESIGN**

For storing and retrieving data, MySQL database is used in this web application. In this database each table has a list of attributes (or columns) which define Schema (metadata) and contain a set of rows (or records), which assigns values to each attribute within a table. Similarly, duplicate records are not allowed.

### **4.1 Design of the database**

An entity-relationship diagram (ERD) is a graphical representation of an information system to see the overall design of database. ERD separates the information needed for each table and shows how the tables stick together.

Three major components of an entity diagrams are the entities, which represent an object or concepts that can have data stored about them, the relationship between the entities, which describes how one or more entities interact with each other, and the cardinality (minimum or maximum), which defines the count of instances that are allowed or are necessary between entity relationships.



**Figure 11.** Entity-relationship diagram

Four tables were created: namely data, reports, user and status. The first two are used to log data sent by the application. The last two tables are used in persisting data related to the location data. Each table has an id primary key.

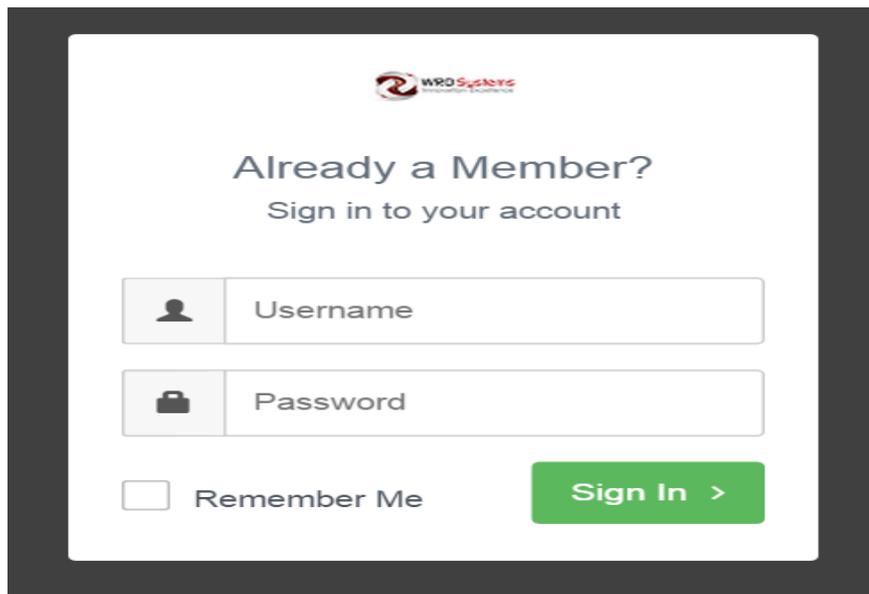
## 4.2 GUI design

### 4.2.1 Login view

Firstly, the login view is what gets to be displayed to the end user in a browser upon starting browsing IAM GPS Tracker application. It contains fields for username and password and a button for submitting the credentials.

In the case of the user submitting correct credentials, the home page will be loaded; if not the login view displays an error message.

A screenshot of the login view can be seen in Figure 12.

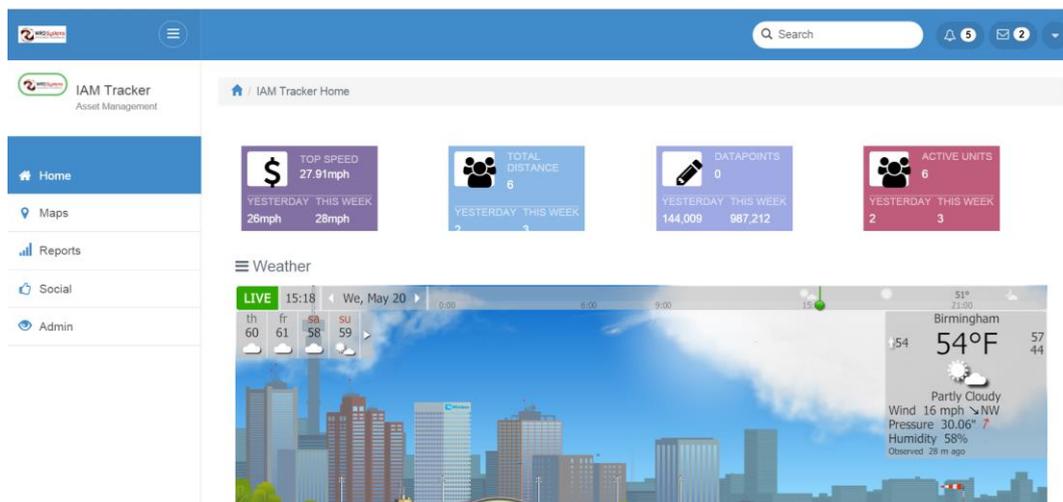


**Figure 12.** Login view screenshot.

#### 4.2.2 Home page view

The Home page view shows real time information for the current city. It gives information on how the application can be used and provides information if a user encounters any technical problems.

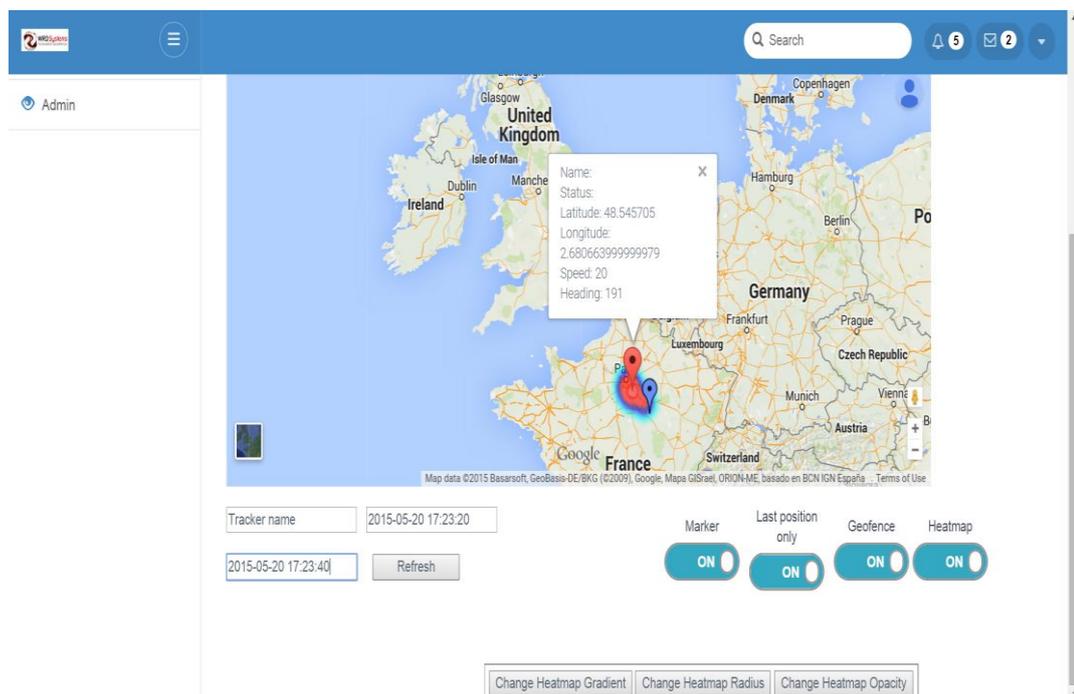
A screenshot of the home page view can be seen in Figure 13.



**Figure 13.** Home page view screenshot.

### 4.2.3 Map view

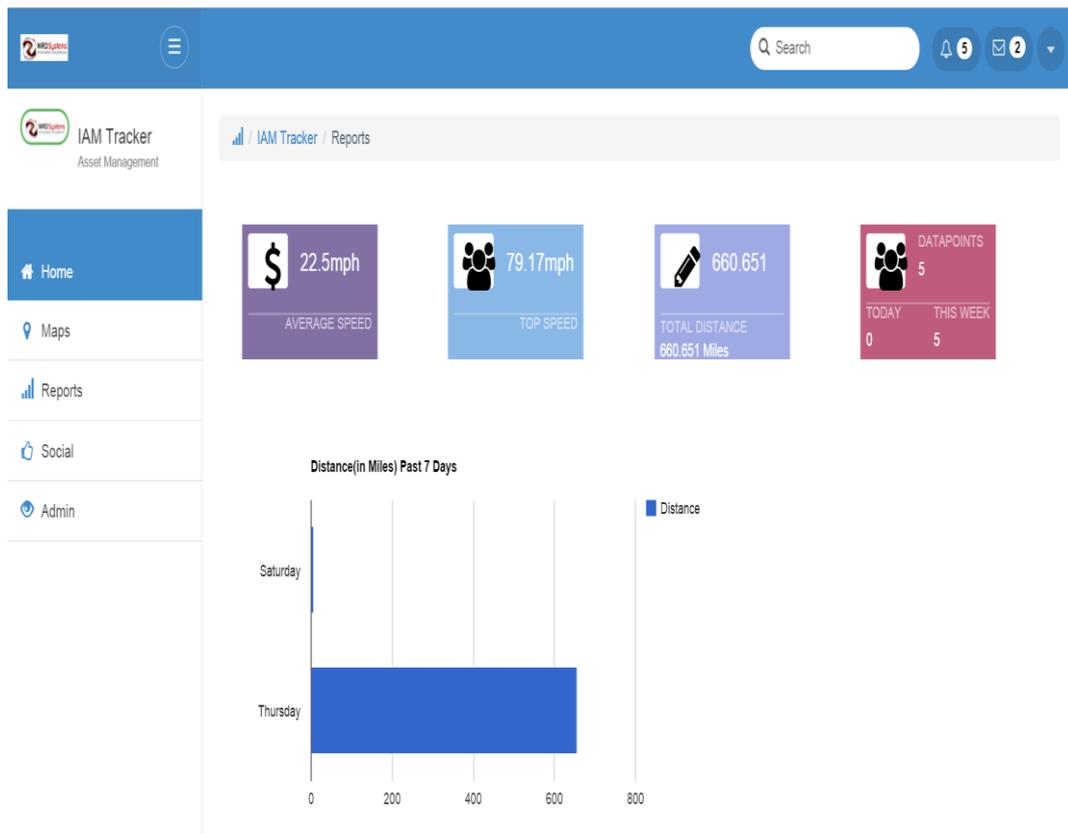
A map view allows the user to view a map with markers indicating their asset's location with pop-up information windows showing speed, heading, longitude and latitude. Furthermore, it allows the user to change the generated heat map radius, heat map opacity and heat map gradient. It also enable the user to turn on and off the marker on the map, geofence and heat map. A screenshot of the map view can be seen in Figure 14.



**Figure 14.** Map view screenshot.

### 4.2.4 Reports view

Presenting information in charts and graphs is easier to understand and interpret than the data. In turn this makes the reporting process much easier and meaningful. Such graphical reports assist users in easily pinpointing the (historical) distance the asset has been driven. A screenshot of reports can be seen in Figure 15.

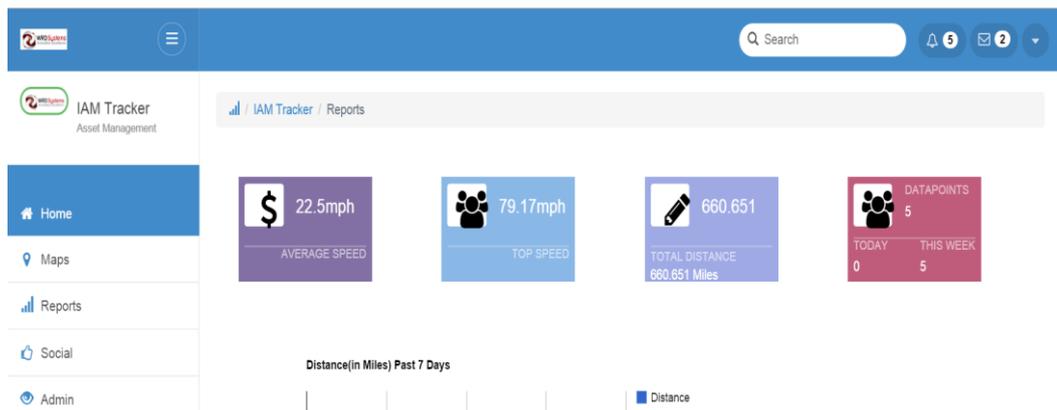


**Figure 15.** Reports view screenshot.

Reports view allows the user to view the results of the tracked assets in graphs. As can be seen in Figure 10 a user can view two graphs demonstrating the total distance of seven days of a week and total distance of the past seven months the asset has been driven.

#### 4.2.5 Statistics view

Statistics provide hard data on performance and output. When summarizing numerical data using statistical methods, enhancing the understanding of the user's tracked asset actual conditions is conceivable. The statistics view can be seen in Figure 16.



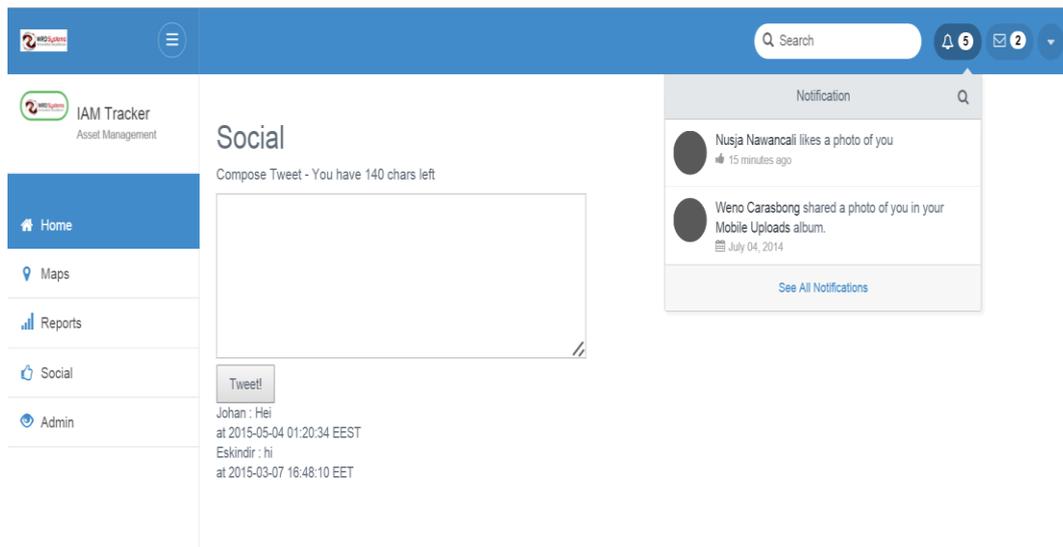
**Figure 16.** Statistics view screenshot.

The four boxes above including the map, reports and home page show the statistics of the tracked asset in at 5 minute intervals. The average speed in miles per hour, top speed in miles per hour, total distance in miles, and the amount of incoming data can be seen in Figure 16.

#### 4.2.6 Social view

Social view is a place where social media integration takes place. It contains a text field and a button for posting data. It offers an option to automatically post data (post user drives or rides). For instance, “Biked x miles today!”.

The social network integration allows a user to post a minimum size of two and a maximum size of 140 writings. The social view displays the messages posted previously up to ten different posts. A social view can be seen in Figure 17.



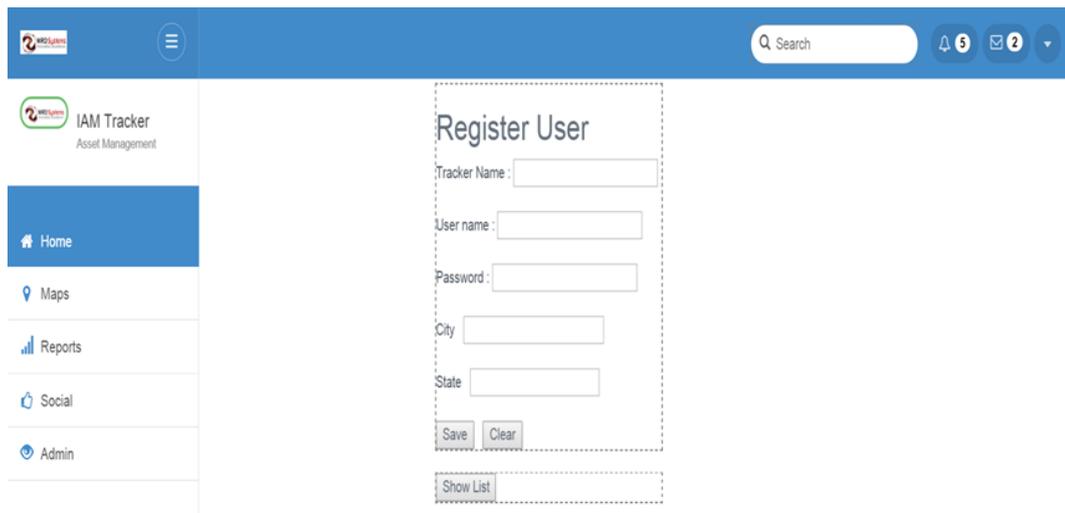
**Figure 17.** Social view screenshot.

If the user is submitting blank messages, the social view will not be loaded or make any changes to the previous post area. The user cannot see the blank message post.

#### 4.2.7 Admin view

The admin view is loaded when an admin follows the “admin” hyperlink. The admin view contains user registration form fields requiring crucial information for user data; a button to submit the registration, to clear wrong text input in case of a typing error and a button to view a list of users who are already registered.

Upon clicking the “save” button, the user information will be sent to the database. Upon clicking the “show list” button, all the user information will be retrieved from the database and displayed on admins view. The social view can be seen in Figure 18.



**Figure 18.** Admin view screenshot.

## 5 IMPLEMENTATION AND DEPLOYMENT

For early development, Grails uses the Java based H2 database, which runs in memory. Configuring Grails to persist the data for test purposes and for production deployment is a requirement.

The application is developed on an Integrated Development Environment (IDE), namely Spring Source Tool Suite.

### 5.1 Setup Windows Development Environment

The stages to set up the application development environment are described below.

1. Since Grails run on top of the Java platform, installation of Java Development Kit (JDK) version 1.6 in the system is a requirement.
2. Install Grails version and setup environment variables by specifying JAVA\_HOME (the directory where JDK is installed), GRAILS\_HOME (the directory where the extracted Grails is placed), and the path to the executable (bin) directory. Testing the installation and setting up environment at this stage is shown below in Code Snippet 1.

```
C:\Users>java -version
java version "1.6.0_45"
Java(TM) SE Runtime Environment (build 1.6.0_45-b06)
Java HotSpot(TM) 64-Bit Server VM (build 20.45-b01, mixed mode)

C:\Users>grails -version
May 26, 2015 12:40:14 AM org.codehaus.groovy.runtime.m12n.MetaIn
newModule
WARNING: Module [groovy-all] - Unable to load extension class [c
vy.runtime.NioGroovyMethods]
May 26, 2015 12:40:15 AM org.codehaus.groovy.runtime.m12n.MetaIn
newModule
WARNING: Module [groovy-all] - Unable to load extension class [c
vy.runtime.NioGroovyMethods]
Grails version: 2.4.3

C:\Users>
```

**Code Snippet 1.** Grails installation and Setup environment test.

This is done by going to the command line and issue “java -version” and “grails -version”.

3. Install free IDE Groovy/Grails Tool Suite version 3.6.2.
4. Configure Groovy/Grails Tool Suite to make sure Java run time of the previous installation of IDE points to a JDK instead of a JRE.
5. In order to validate everything is properly working, a sample project was created and run. This can be done by following the path (create Grails project -> open Grails Console -> type “run-app” -> press Enter Key).

## 5.2 Database configuration

Once Grails is installed, by default Grails uses the Java based (an embedded) H2 database, which runs in memory. For this reason, any databases that are created are not persisted (saved permanently).

MySQL is the most used database system in the world. It is compatible with virtually for every operating system, and is almost a commerce standard. To persist the data for test purposes and for production deployment of the application, configuring Grails to use MySQL is a constraint. Using Grails to create the database sets the application’s SQL login as “Owner”.

There are three different MySQL drivers for connecting to MySQL server. The drivers are Open Database connectivity (ODBC) or Java Database Connectivity (JDBC) connection method or the native driver.

A sample configuration file (DataSource.groovy) from the development environment can be seen below in Code Snippet 2, which specifies database connection information, server ports, url and the JDBC driver for configuring serverUrl in the development environment.

```

dataSource {
    pooled = true
    jmxExport = true
    driverClassName = "com.mysql.jdbc.Driver"
    username = "username" //MySQL database username
    password = "password" //MySQL database password
    url="jdbc:mysql://mysql.cc.puv.fi/databaseName"
}
hibernate {
    cache.use_second_level_cache = true
    cache.use_query_cache = false
    cache.region.factory_class = 'org.hibernate.cache.ehcache.EhCacheRegionFactory' // Hibernate 4
    singleSession = true // configure OSIV singleSession mode
    flush.mode = 'manual' // OSIV session flush mode outside of transactional context
}
environments { // environment specific settings
    development {
        dataSource {
            dbCreate = "update" // 'create', 'create-drop', 'update', 'validate'
            pooled = true
            jmxExport = true
            driverClassName = "com.mysql.jdbc.Driver" // JDBC driver class
            username = "username" //MySQL database username
            password = "password" //MySQL database password
            url="jdbc:mysql://mysql.cc.puv.fi/databaseName"
        }
    }
}
}

```

### Code Snippet 2. Sample DataSource.groovy file.

The modified config file in DataSource.groovy describe the configuration of the development environment data source to use MySQL. The com.mysql.jdbc.Driver is the JDBC driver, so that Grails “recognize” the driver class has changed for the development environment. The username and password are for the application’s SQL Server login. The url contain server name mysql.cc.puv.fi where the SQL Server instance is located, and the databaseName is the name of the database.

When running the application after adding mysql-connector-java:5.1.29 to the dependencies section, MySQL driver jar will be loaded. This can be seen below in Code Snippet 3.

```

// specify dependencies
dependencies {
    runtime 'mysql:mysql-connector-java:5.1.29' //MySQL driver for connecting to MySQL
    test "org.grails:grails-datastore-test-support:1.0-grails-2.4"
}

```

### Code Snippet 3. Sample BuildConfig.groovy file.

### 5.3 Data Response Structure

GORM is Grails' object relational mapping (ORM) implementation, which sits on top of the very popular Hibernate framework. It is used to store and retrieve information.

Upon starting the application, Grails automatically creates a database table for each domain class and adds columns id to be the primary key for all tables. For optimistic locking it adds column version as well.

Here is a code snippet 4 on domain class named Data that will mapped onto a database table data.

```
class Data {
    String name;
    String status;
    Double latitude;
    Double longitude;
    Double speed;
    Integer heading;
    Timestamp date;
```

**Code Snippet 4.** A Data table in a database.

Start up the application and the corresponding table “data” will be created with columns name, status, latitude, longitude, speed, heading, and date. The correct sql data type (VARCHAR for String, Timestamp for TIMESTAMP, BIGINT for Integer and Double for Double) will be applied as well.

The tracker (device) sends data with different field representations. The responses with location data are formatted as follows:

```
String location = "%$3E5T;210824.000,A,6304.7046,N,02143.2818,E,0.30,198.02,150914;
```

In this string, the following fields are delimited by semicolons (;):

\$\$: This is a data identifier used to distinguish normal data from tracking data.

3E5T: This is a four character unique asset identifier.

Fields delimited by semicolons (;):

210824.000,A,6304.7046,N,02143.2818,E,0.30,198.02,150914: RMC GPS data with fix taken time stamp, status (Availability), latitude, longitude, speed (speed over the ground knots), heading (Track angle in degrees) and date respectively. A code of string parsing is shown below in Code Snippet 5.

```
def parseLocation(String location){
    String[] informations = location.split(";");
    String[] coordinates = informations[1].split(",");
    Data[] data = new Data[1];
    data[0] = new Data()
    //data[0].status=coordinates[1]
    data[0].name = informations[0]
    data[0].latitude=Double.parseDouble(coordinates[2])
    data[0].longitude=Double.parseDouble(coordinates[4])
    data[0].speed=Double.parseDouble(coordinates[6])

    render(view:"index", model:[data: data]);
}
```

#### Code Snippet 5. String parsing.

This parseLocation function gets the location parameter and split fields' representation according to the delimiter semicolon (;) and comma (,), put on variable information and coordinates. Then, this data is parsed as a string and saved in to table data in the database. Later, this will be the data that will be used to generate the marker on the map, reports and the statistics.

## 5.4 Reports

Reports is the entity that is mapped onto database table reports. As mentioned in the data response structure discussion, when starting the application the corresponding reports table is created automatically for the entire domain attribute, which is table "reports" with the column's name, value, latitude1, longitude1, latitude2, longitude2, datefrom, and dateto will be created in MySQL database. This can be seen below in Code Snippet 6.

```
class Reports {  
  
    String name  
    Double value  
    Double latitude1  
    Double longitude1  
    Double latitude2  
    Double longitude2  
    Timestamp datefrom  
    Timestamp dateto  
  
    static hasMany = [data: Data]
```

**Code Snippet 6.** Reports table.

The right sql datatype is applied as well (e.g. varchar for String). The hasMany defines a one-to-many relationship, which indicates Reports has many instances of Data class.

## 5.5 SQL Server stored procedure

```

BEGIN
  DECLARE datetimefrom Timestamp;
  DECLARE trackername VARCHAR(4);
  DECLARE lat decimal(10,6);
  DECLARE lng decimal(10,6);

  DECLARE datetimefrom2 Timestamp;
  DECLARE trackername2 VARCHAR(4);
  DECLARE lat2 decimal(10,6);
  DECLARE lng2 decimal(10,6);

  DECLARE distance float;
  DECLARE done INT DEFAULT FALSE;
  DECLARE cursor_i CURSOR FOR SELECT name, latitude, longitude, date FROM data WHERE
  DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;

  OPEN cursor_i;
  FETCH cursor_i INTO trackername, lat, lng, datetimefrom;

  read_loop: LOOP
    FETCH cursor_i INTO trackername2, lat2, lng2, datetimefrom2;

  IF done THEN
    LEAVE read_loop;
  END IF;

  INSERT INTO reports(name,datefrom,dateto, value,latitude1,longitude1,latitude2,longitude2)
  (trackername,get_distance_in_miles_between_geo_locations(lat,lng,lat2,lng2));

  SET trackername=trackername2;
  SET lat=lat2;
  SET lng=lng2;
  SET datetimefrom=datetimefrom2;

```

**Code Snippet 7.** PROCEDURE sp\_DistanceCalculator.

The distance calculator procedure gets tracker data from the table data and invokes the function `get_distance_in_miles_between_geo_locations`. This function is shown below in Code Snippet 8.

```
BEGIN
  return ((ACOS(SIN(geo1_latitude * PI() / 180) * SIN(geo2_latitude * PI() / 180)
+ COS(geo1_latitude * PI() / 180) * COS(geo2_latitude * PI() / 180)
* COS((geo1_longitude - geo2_longitude) * PI() / 180)) * 180 / PI())
|* 60 * 1.1515);
END
```

**Code Snippet 8.** `get_distance_in_miles_between_geo_locations` function.

This function calculates distance in miles between two geo locations and return results. Then, the procedure inserts the data (name, value (distance), latitude1, longitude1, latitude2, longitude2, datefrom, and dateto) in to the table report in the database.

Since there is much data coming from the device at five minute interval, the `sp_Get-LastCalculatedTime2` procedure defines the last run date and time. In the case of the last run date and time not being found it sets the previous date and time as the starting point. This is shown below in Code Snippet 9.

```

BEGIN

DECLARE last_run_time DATETIME DEFAULT NULL;
DECLARE current_run_time DATETIME DEFAULT NOW();

-- Define the last run time
SET last_run_time := (SELECT dateto FROM reports WHERE name=p_name order by dateto DESC LIMIT 1);

-- if there is no last run time found then use yesterday as starting point
IF(last_run_time IS NULL) THEN
    SET last_run_time := (SELECT date FROM data WHERE name=p_name order by id ASC LIMIT 1);
END IF;

SET Value1=last_run_time ;
END

```

#### **Code Snippet 9.** PROCEDURE sp\_GetLastCalculatedTime2.

With the help of procedure sp\_GetLastCalculatedTime2, the search function handles tracker search between the start and end date and time. This is shown below in Code Snippet 10.

```

def search(){
    SimpleDateFormat df = new SimpleDateFormat("yyyy-mm-dd hh:mm:ss");
    String keyword = params.id;

    Timestamp from = df.parse(params.date_from).toTimestamp();
    Timestamp to = df.parse(params.date_to).toTimestamp();
    def data = Data.findAllByNameAndDateBetween(keyword,from,to)

    render(view:"index", model:[data: data]);
}

```

#### **Code Snippet 10.** Search Tracker.

This function parse the start and end date and time. Then, it gets the tracker data by the name of the tracker and between the start and end date and time. Later, it sends back to view and display tracker information on a map.

The get chart method creates a new instance of groovy.sql.sql with the database of the Grails and gets each data as name, day name and distance from the table reports. With this data it generates charts. The getchart method code snippet is shown below in Code Snippet 11.

```
def getCharts() {

    def returnableCharts = [:]
    def sql = new Sql(dataSource) // Create a new instance of groovy.sql.Sql with the DB of the Grails
    def result

    result = sql.rows("SELECT name, DAYNAME( Datefrom ) AS MyDay, SUM( value ) AS Distance FROM reports")

    result.each { q->
        def chart = [:]
        def column = [['string', 'MyDay'],['number', 'Distance']]
        def data = []

        result.each{ r->
            data << [r.MyDay,r.Distance]
        }

        chart << [name:q.MyDay]
        chart << [Distance:q.Distance]
        chart << [column:column]
        chart << [data:data]

        returnableCharts << [(q.MyDay):chart]
    }
    return returnableCharts

}
```

### Code Snippet 11. Getchart method.

The GSP page to use the visualization, the tag lib <gvisualization: apiImport /> is used to import the google visualization API JavaScript library. In the tag lib gvisualization is the name space for visualization.

This Grails Server Page (GSP) passes the following essential parameters: elementId, columns, and data to the view. This indicates which element (div) to draw the chart in, the name of the axes, and the data points for those axes respectively. The

width, height, and title attributes are all part of the chart's configuration options. This is shown below in Code Snippet 12.

```
<html>
  <head>
    <meta name="layout" content="themeforest" />
    <link rel="stylesheet" type="text/css" href="{request.contextPath}/themeforest/css/boxes.css">
    <title><g:message code="default.show.Label" args="[entityName]"/></title>

    <gvisualization:apiImport/>
  </head>
  <body>
    <div class="media-body">
      <div class="contentpanel">

        <g:each in="{charts.values()}" var="item">
          <gvisualization:barCoreChart
            elementId="charts"
            title="Distance(in Miles) Past 7 Days"
            width="{600}" height="{340}"
            columns="{item.column}"
            data="{item.data}"/>

        </g:each>
```

**Code Snippet 12.** GSP page using visualization.

The Google Visualization plugin convert the appropriate JavaScript Object that the Google Charts Tools API expects. A vertical bar chart is rendered in the browser, the column charts display tooltips when the user hovers over the data.

```
compile ":google-visualization:1.0.1"
```

**Code Snippet 13.** Google visualization plugin.

The domain class status is where the status updates of all the existing users are. When the application starts table status with the corresponding column message and datecreated will be created automatically in the database. This can be seen below in Code Snippet 14.

```

class Status {
  String message
  Date dateCreated

  static constraints = {
    message size:2..140, blank:false
  }
}

```

**Code Snippet 14.** Status model.

The constraints specify that the message has to have a minimum size of two and a maximum size of 140, and does not allow blank messages to be tweeted.

```
@Secured('IS_AUTHENTICATED_FULLY')
```

**Code Snippet 15.** Security service.

@Secured ('IS\_AUTHENTICATED\_FULLY') is for the page to be secured, only registered users to be able to access the page.

The look up user method creates a user object and uses the id of the current logged in user using spring security service. The lookupUser method is shown below in Code Snippet 16.

```

private lookupUser() {
  User.get(springSecurityService)
}

```

**Code Snippet 16.** lookupUser method.

The current user timeline method returns a collection of Statuses whose writer is the current logged in user. The currentUserTimeline method is shown below in Snippet Code 17.

```

private currentUserTimeline() {
  def query = Status.whereAny {
  }.order 'dateCreated', 'desc'
  def messages = query.list(max: 10)

  return messages
}

```

**Code Snippet 17.** currentUserTimeline method.

The maximum number of records is 10 (It displays the last 10 entries sorted by date and time) and these records are sorted in descending order.

When the index page loads, it will load all the messages and display them. This is shown below in Code Snippet 18.

```
def index() {
  def msgs = currentUserTimeline()
  return [messages: msgs] // key=>value
}
```

**Code Snippet 18.** Status index method.

Here Code Snippet 19 shows that the update status creates a new Status and uses the message from the form submission as its value and saves the status. The messages define a collection of existing messages that will be used for displaying. Then it will render (display) all the messages provided before the last line of code. This is shown below in Code Snippet 19.

```
def updateStatus = {
  def status = new Status(message: params.message) //creates a new Status and uses the
  status.save() //saves the status object.
  def messages = currentUserTimeline() //defines a collection of existing messages t
  render template: 'messages', collection: messages, var: 'message' //renders all the
}
```

**Code Snippet 19.** Update the status.

A table user is created with the column's tracker name, username, password, city and state with data type (varchar for String) in the database. This helps to store the user's information in the database. Defining constraints help to validate instances in the user domain class. This can be seen in Code Snippet 20.

```

class User {
    String trackerName
    String userName
    String password
    String city
    String state

    // constraints are defined as static
    static constraints = {

        trackerName blank: false
        userName(blank: false)
        password(password: true)
        city blank: false
        state blank: false
    }
}

```

**Code Snippet 20.** Manage user information.

In order to get detailed information on the users for the administrator a code which display a user list is shown below in Code Snippet 21.

```

def show(Long id){
    def userInstance = User.list()
    [allUsers: userInstance]
}

```

**Code Snippet 21.** Show method.

The code `def userInstance = User.list()` will fetch all the records in the user table and store them in `userInstance` variable. The `show()` method returns a map of values. The object with name “userInstance” and the value are the list of allusers read from the database. This map will be passed to admins view (GSP).This is shown below in Code Snippet 22.

```

<!DOCTYPE html>
<html>
  <head>
    <meta name="layout" content="themeforest" />
    <title>Users List</title>

  </head>
  <body>
    <g:each in="{userInstance}" status="i" var="allUsers">
      <tr>
        <td>${allUsers.trackerName}</td>
        <td>${allUsers.userName}</td>
        <td>${allUsers.password}</td>
        <td>${allUsers.city}</td>
        <td>${allUsers.state}</td>
      </tr>
    </g:each>

    <g:form controller="adminsResp" action="searchById">
      <input type="submit" value="Back"/>
    </g:form>
  </body>
</html>

```

**Code Snippet 22.** User GSP page.

This GSP HTML code displays a list of users. The tag field values are used to get the value of user's property. The content inside `{ }` is evaluated and rendered.

## 5.6 Application Deployment

Source code revision is done using Git. The project uses a Git server provided by WRD Systems. For this reason, Git for Windows were installed to provide the Git graphical user interface which is a powerful alternative to Git BASH. The Git BASH offers a graphical version of every Git command line function, which behaves like the "git" command in LINUX and UNIX environments.

As mentioned above in the Git discussion, the entire project resides in: committed, modified and staged. This can be seen below in Code Snippet 23.

The screenshot shows a Git GUI window titled 'gitk: tinbit'. The main area displays a commit diff for the 'Final IAM Tracker project' by Tinbit Bekele. The commit message is: 'Final IAM Tracker project'. The diff shows changes to 'Allapps/grails-app/conf/BuildConfig.groovy' and 'Allapps/grails-app/controllers/adminsactivity/AdminsRespController.groovy'. The 'BuildConfig.groovy' file has several lines added, including dependencies for 'google-visualization' and 'spring-security-core'. The 'AdminsRespController.groovy' file has a class definition added. The right sidebar shows a list of files in the commit, including 'Allapps/grails-app/conf/BuildConfig.groovy', 'Allapps/grails-app/controllers/adminsactivity/AdminsRespController.groovy', and 'Allapps/grails-app/controllers/stringparsing/ReportsController.groovy'. The bottom of the window shows the commit details, including the author 'Tinbit Bekele', the parent commit '39b6c35d6b68bd4bc4711f7baacfb498521f0d1', and the branches 'master' and 'remotes/origin/master'.

### Code Snippet 23. Source code revision management.

Furthermore, using Git for source code revision management allows having the same tools available in case of issues.

## **6 TEST CASES**

Testing is a major part of the software development process done to ensure the correct aspect and functionality of an application.

The agile methodology used in the development of the entire project ensures that testing is done at each life cycle of the development.

### **6.1 Application Testing**

For each of the functionalities implemented, the application is tested and shown to the company representative (sent to the server provided by the company). From the feedback obtained, bugs are fixed and changes are made before proceeding to the next feature.

Debugging is performed in the development phase while fixing the reported bugs and/or while conducting in Unit Testing.

### **6.2 Completeness of data reporting**

Since each tracker sending data updates on location, speed and heading at five minute intervals for doing the data analysis design decisions are made on what data can be processed in real time and what data needs to be updated on hourly, daily, weekly, and monthly basis. This has been extensively tested during all the stages of the development.

### **6.3 Performance Testing**

Performance analysis was a part of the implementation. The application performance was tested for eight conditions:

- Scalability as datasets grow in size.
- Extensible architecture; developers can expand or add to its capabilities.
- Completeness of data reporting.

- Quality and correctness during the design phase and implementation phase.
- The generated web interface being cross platform.
- The use of external libraries is limited, especially on client side.
- Proper testing procedures.
- Fully deployable on Linux based servers.

## 7 CONCLUSION

While trackers are used as a security mechanism, the main objective of this project is to use these trackers to gather behavioral aspects of the tracked assets. In particular it is not only useful to know where each asset is at any time with a typical dot on a map, but also aspects such as total distance the assets has been driven, speed data and disuses of assets are valued. This project offers insight and facts about the tracked assets for customers.

The project has been reviewed by the WRD Systems Ltd representative for acceptance tests and deployments. The goal of the project was achieved and the company representative was satisfied. Regarding the performance and completeness of data reporting, positive feedback was given to the project.

One of the biggest challenges that came across in this project was to setup the developing environment. As a beginner, learning a new technology, Grails, was a challenge as well.

To conclude, an opportunity to work with this project did not only expand my skills set, but also opened up a new window to look further on my career options. A wide range of reading, but most importantly understanding and problem solving were a new and valuable experience as well.

### 7.1 Future work

The project is fully deployable on Linux based servers. This allows having the same tools available in case of issues and further development. Furthermore, the Virtual Machine provided by WRD Systems can be used by others to continue the work, or to have a starting point with all the tools and libraries added during the project.

## 8 REFERENCES

- /1/ Installation requirements. Accessed on 02.02.2015.  
<http://grails.github.io/grails-doc/2.4.3/guide/gettingStarted.html#requirements>
- /2/ Grails. Accessd on 15.01.2015.  
<http://grails.github.io/grails-doc/2.4.3/guide/introduction.html#whatsNew24>
- /3/ Introduction to Object-Oriented Java Script. Accessed on 09.02.2015.  
[https://developer.mozilla.org/en-US/docs/Web/JavaScript/Introduction\\_to\\_Object-Oriented\\_JavaScript](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Introduction_to_Object-Oriented_JavaScript)
- /4/ GORM query reference. Accessed on 06.04.2015.  
[https://www.safaribooksonline.com/library/view/grails-in-action/9781617290961/kindle\\_split\\_034.html](https://www.safaribooksonline.com/library/view/grails-in-action/9781617290961/kindle_split_034.html)
- /5/ jQuery website. Accessed on 07.03.2015.  
<http://jquery.com/> .
- /6/ PL/SQL- Procedures. Accessed on 13.03.2015.  
[http://www.tutorialspoint.com/plsql/plsql\\_procedures.htm](http://www.tutorialspoint.com/plsql/plsql_procedures.htm)
- /7/ Configuring Grails to use an SQL Server Database. Accessed on 12.02.2015.  
<https://corketech.wordpress.com/2013/06/23/configuring-grails-to-use-an-sql-server-database/>
- /8/ Company website. Last Accessed 03.05.2015.  
<http://www.wrdsystems.com/> .
- /9/ Javascript: Advantange and Disadvantage. Accessed on 03.02.2015.  
<http://www.jsripters.com/javascript-advantages-and-disadvantages/>
- /10/ Setup your windows development environment. Last Accessed 12.02.2015.  
<http://grails.asia/grails-tutorial-for-beginners-setup-your-windows-development-environment>
- /11/ Planet of the web. Accessed on 08.03.2015.

[http://planetoftheweb.com/components/pro-mos.php?id=556%5b10:52:25%20PM%5d%20#@\\$\\$!:%20http://www.fiercewireless.com/offer/kony\\_wirelesswebinar\[10:53:12%20PM\]%20%23@\\$\\$!:%20http://www.html5report.com/topics/html5/articles/356596-html5-adoption-pros-far-outweigh-cons.htm](http://planetoftheweb.com/components/pro-mos.php?id=556%5b10:52:25%20PM%5d%20#@$$!:%20http://www.fiercewireless.com/offer/kony_wirelesswebinar[10:53:12%20PM]%20%23@$$!:%20http://www.html5report.com/topics/html5/articles/356596-html5-adoption-pros-far-outweigh-cons.htm)

/12/ Google visualization API Reference. Accessed on 4.04.2015.

<https://developers.google.com/chart/interactive/docs/reference>

/13/ By Glen Smith and Petter Ledbrook. P. June 2014. Grails in Action. Second Edition. Manning Publications.

<https://www.safaribooksonline.com/library/view/grails-in-action/9781617290961/>

/14/ Section of Git project. Accessed on 20.02.2015.

<https://git-scm.com/book/en/v2/Getting-Started-Git-Basics>

/15/ Groovy 2.4 and Grails 3 To Be Last Major Releases. Accessed on 02.02.2015

<http://blog.pivotal.io/pivotal/news-2/groovy-2-4-and-grails-3-0-to-be-last-major-releases-under-pivotal-sponsorship>

/16/ Sentence description in NMEA. Accessed on 15.02.2015.

<http://www.gpsinformation.org/dale/nmea.htm#RMC>

/17/ Groovy/ Grails Tool Suite (GGTS) .Accessed on 1.02.2015.

<http://grails.org/products/ggts /4/>

