|  |
| --- |
|  |

**Chapter 1.Introduction**

**1.1 Overview**

Indian Railways is the principal mode of transport in the country. It is one of the world’s largest rail networks under a single management. The route length is around 63,332 km[10] with more than 8000 stations[10]. As it is the backbone of nation’s transport system, IR owns more than 25,000 wagons[10], 45,000[10] different types of coaches and 8000 locomotives[10]. The system carries about 5,000 million[10] passengers generating a traffic output of 340 billion[10] passenger kms.

So many people travel daily in train and came across so many railway station and platforms. There is always problem to locate a book stall, canteen, bathroom, waiting room etc. on unknown station/platform. So we wanted to build an android app which will inform you which will be the next coming station and what is the layout of stations, like how many platforms, where is police station on map/own display.

**1.2 Problem Definition**

In railway guide system user can search all information about train, station, route, time required to reaching to destination. Apart from this through our system passenger access information like nearest tourist places, passenger can view the whole platform wise layout of a particular station. passenger can set a reminder of the particular station, which when arrives, the application will send an alert sound or vibrate. Also The passenger will be able to view the amount of time the train is going to wait at the particular station. Passenger can post complaints about the services in the train. The passenger can receive alerts and alarms for protection against thefts to take. The main aim of the project was to develop a application which would facilitate to access information about train, station layout, nearest tourist places etc through an effective and yet simple GUI for a normal passenger intending to travel in railways. Consequently, the higher number of passenger uses the train to travel from source to destination. so we proposed a system railway

**1.3 Motivation for the project**

The idea behind choosing to develop this application is that, day to day travelling via railways has become a tedious and complicated job which requires a lot of experience and knowledge about the place we visit. Hence to overcome this problem and tiring job of locating places on platforms like washrooms, restrooms, bookstalls etc we decided to provide an application which can provide us the features to make our journey easy and reliable. We wanted to develop a Indian Railway Commute Android App which will help user to locate some place/things quickly on unknown station platform and also give us the halt time of the station with the alert of the arrival of the next station. Instead of standing in line to wait for the booking of the lockers or waiting rooms or taxi services, we provide features to book it before hand while approaching the station sitting in your train.

**1.3 Motivation for the project**

The idea behind choosing to develop this application is that, day to day travelling via railways has become a tedious and complicated job which requires a lot of experience and knowledge about the place we visit. Hence to overcome this problem and tiring job of locating places on platforms like washrooms, restrooms, bookstalls etc we decided to provide an application which can provide us the features to make our journey easy and reliable. We wanted to develop a Indian Railway Commute Android App which will help user to locate some place/things quickly on unknown station platform and also give us the halt time of the station with the alert of the arrival of the next station. Instead of standing in line to wait for the booking of the lockers or waiting rooms or taxi services, we provide features to book it before hand while approaching the station sitting in your train.

**1.4 Design Idea**

The percentage of people trvelling by train is enormous.The inconvenience faced by passengers is to search various places on platform such as waiting room ,book stalls etc.So we are providing an android application that will give all the information to passenger about the various places on platform.Our application provides feature of locker booking and notification to passengers about the train arrival and departures time.our android application helps the user to provide the tourist places nearby.

1**.5 Achievements**

In railway guide system user can search all information about train, station, route, time required to reaching to destination. Apart from this through our system passenger access information like nearest tourist places, passenger can view the whole platform wise layout of a particular station.This application being user friendly helps in posting different messages and also getting notifications about the timing of train.

**Chapter 2**

1. **Software Requirement Specification (SRS)**
   1. **Introduction** 
      1. **Purpose of this document**

This document gives the functional and software requirements, and the order valid detailing of this application. This document also explains assumptions and limitations, other aspects of this application.

* + 1. **Scope of the development project**

**Passenger Module**

The passenger can register to use the app creating id, password. The passenger can login to the system with its own id and password created at the time of registration. The passenger will be receiving the notification of arrival of the next station. The passenger can view the whole platform wise layout of a particular station. The passenger can set a reminder of the particular station, which when arrives, the application will send an alert sound or vibrate. The passenger will be able to view the amount of time the train is going to wait at the particular station.

**Station Master**

The station master can login to his account of his station. The station master can feed in data of his station. The station master can send important notification to the passenger on the station. If baggage is reported lost the immediately notify the station staff and the other station masters, where ever the train is going to stop.

**Station layout module**

End user can search for next coming station and can see the layout of next station. It check all shops, tea stall, hotel on the coming station, number of platform, on which platform train will arrive etc. Station master add the station layout, information of station.

**Notification**

The passenger can set a reminder of the particular station, which when arrives, the application will send an alert sound or vibrate. The passenger will be able to view the amount of time the train is going to wait at the particular station. All information of arriving time of train, departure time of train, route add by station master.

**Station information:**

Displays the time table of the trains i.e arrival and departure time of the train. Cost of tickets according date, the train and the source and destination. Whenever the train has been delayed the passengers will be notified of this delay and the new arrival time.

**Complaints:**

Passenger can post complaints about the services in the train. The passenger can receive alerts and alarms for protection against thefts to take precaution. If baggage is reported lost then station master immediately notify the station staff and the other station masters, where ever the train is going to stop.

**Tours and travel guide:**

The passenger can access information about various tourist places at the particular station. Passenger can access all the information of nearest tourist places like how far from station, rickshaw, taxi available or not etc.

**Locker**

Number of locker, available locker, booked locker all information is add station master. Our user can access all this information and passenger can book a locker.

* 1. **General Description**
     1. **User personas and Characteristics**

The user of this product/system will be any citizen who is traveling through train and want information about trains and station.

* + 1. **Product perspective**

Indian Railway has so many people commuting via trains. People often find it hectic and time consuming to locate places on unknown station/platform. To save time and effort we want to build an android application which will inform you about the next coming station and what is the layout of stations(location of washroom, waiting rooms, book stalls, police station etc), like how many platforms, where is police station on map/own display. This will help user to get to know the next station before reaching the station and help to locate whatever we need on that station. Will would also like to add one more feature like app would show how much time train will stay on this station and remind the user 30 sec before. The scope does not involve the out of the station information except for the tourist places. Software has two major component one the server and the second one is the mobile application. The server will require Windows XP/Vista 7 machine with minimum 1GB RAM and 100 GB hard disk. The server machine also required WIFI devices sing which it can create Wireless Ad-hoc network.

* + 1. **Overview of Functional Requirements**

**2.2.3.1** **Overview of functions performed**

Station master:

* + - System should support Android handset
    - System should have internet on mobile
    - System should support to GPS on mobile
    - System should have support to register new user
    - System should have enter in their profile, when they login with their login credentials
    - System should support to add and update profile.
    - System should support to add new information.
    - System should support to add layout of station.
    - System should support to send notification message.
    - System should allow users to create/update their own profile.
    - System should give connectivity to client
    - System should have to maintain database.
    - System should allow multiple users at a time.
    - System should maintain update of user
    - System should maintain update of station master

User:

* System should support Android handset
* System should have internet on mobile
* System should support to GPS on mobile
* System should have support to register new user
* System should have enter in their profile, when they login with their login credentials
* System should able to get the layout of station
* System should able to receive notification message
* System should able to receive proper error message when system is fail to connect to internet.

**2.2.3.2 Deployment:**

Deployment would be done on android phone and apache tomcat as the web server.

* + 1. **Overview of Non-functional Requirements**

**2.2.4.1 Performance Requirements**:

* Performance is the main concern as our website should link to all the android phones of the clients at the back-end; store all the data in the database.
* The speed of system processing the data and mapping to the database should be as fast as the movement of the user
* The response time for refreshment and retrieving information must be low.

## 2.2.4.2 Safety Requirements:

To safely use of this system, the covering image should be large size seleted by end user, because of that may be loss of source data.

## 2.2.4.3 Security Requirements:

For using this system end user can enter username and password to login form in case of wrong information. account will be blocked if attempt is greater than 3 times.

* + 1. **Operating Environment**

Software has two major component one the server and the second one is the mobile application. The server will required Windows XP/Vista/7 machine with minimum 1GB RAM and 100 GB hard disk. The server machine also required WIFI devices sing which it can create Wireless Ad-hoc network. Mobile application will support Android phones so at least 2 Android devices required getting the output.

* + 1. **General Constraints, Assumptions, Dependencies, Guidelines**

The user is expected to have Android Mobile phones and should be able to send and receive data when connected to wifi range. First the user has to register to wifi network to use the service. Network gives the functionality to login and registration facility. The registered user uses this network to send and receive messages when connected to wifi network.

* 1. **Specific Requirements**
     1. **External Interface requirements**
        1. **User Interfaces**

#### Station master:

1. Registration Page
2. Login Page
3. Profile Page
4. Enter data of its station Page
5. Layout of its station Page
6. Send notification Page
7. Registered compliant Page

#### User:

1. Registration Page
2. Login Page
3. Searching next station Page
4. Station Layout Page
5. Searching tourist place near station Page
6. Message to user according to search Page
7. Message from station master Page
8. Compliant registration Page
   * + 1. **Hardware Interfaces**

Mobile application will get installed on mobile devices. These mobile devices should have WIFI device thorough which it will connect to server.

* + - 1. **Software Interfaces**
* Operating System: Windows XP/Windows Vista/Windows 7.
* Database: MySql 6.0.
* Android 2.2 supported mobile handset
* Tomcat 6
* JDK 1.6
* Eclipse 3.4
  + - 1. **Communication Interfaces**

Here we will be using WIFI network and going to create our own communication protocol. Software will also support BASE64 encryption logic while sending data to server. Server will support HTTP protocol for web based access

* + 1. **Detailed Description of Functional Requirements**

|  |  |
| --- | --- |
| Function: | Authentication |
| Precondition: | User has valid email-id and password. |
| Steps: | 1. Sign up for application 2. Enter valid email-id and password. 3. Get server access. 4. Successful log-in |
| Post-condition: | 1. Login Successful. 2. User can successfully access the features of the app. |
| Alternate Flow: | 1. If email-id and password is invalid check again. 2. If the user forgets the password click on the forgot password option. 3. If the user forgets user-id then system verifies and provides a new user-id and password. |

2>Update Application.

|  |  |
| --- | --- |
| Function: | Uploading database according to the passenger. |
| Precondtion: | Updating the data. |
| Steps: | 1. Server Authentication. 2. Check the database.   3>Update the changed data. |
| Post Condition: | Data Updated. |
| Alternate Flow: | NA |

3>Notification

|  |  |
| --- | --- |
| Function: | Notification |
| Precondtion: | User should be logged in and GPS should be activated. |
| Steps: | 1>Login to the app.  2>Check the GPS activation.  3>Server authentication.  4>Check the database for notification.  5>Notify the user with respective notification related to tickets reservation ,destination, timings for next railway stop , etc. |
| Post Condition: | User Notified |
| Alternate Flow: | 1>Check the GPS activation ,if not, activate and check again.  2> If GPS not available notification failed. |

4> Functional Requirement for Setting Reminder

|  |  |
| --- | --- |
| Functions: | Reminder |
| Precondition: | Account creation and user should be in the train. |
| Steps: | 1. Login. 2. Choose the arriving station. 3. Set alarm. |
| Post condition: | Alarm rings on arrival of the station |
| Alternate Flow: | If the User cannot login, alarm cannot be set hence reminder will not ring. |

5> Functional Requirement for Viewing Layout

|  |  |
| --- | --- |
| Function: | View Layout of the particular platform. |
| Precondition: | Account creation and user to be present on particular platform |
| Steps: | 1>Login to the app.  2>Select your platform  3>Check the layout according to requirement |
| Post condition: | Viewing the layout |
| Alternate Flow: | If user cannot login , then he cannot access layout. |

6> Functional Requirement for Booking a locker

|  |  |
| --- | --- |
| Function: | Book locker at a particular station. |
| Precondition: | Account creation and user to be travelling towards the station |
| Steps: | 1>Login to the app.  2>Select your platform  3>Check availability of locker  4>if locker available Book locker. |
| Post condition: | Locker booked |
| Alternate Flow: | 1>Login to the app.  2>Select your platform  3>Check availability of locker  4>if locker not available locker not booked. |

7> Functional Requirement for Booking a Taxi/Rickshaw

|  |  |
| --- | --- |
| Function: | Book Taxi/ Rickshaw at the station your destination |
| Precondition: | Account creation and user, destination |
| Steps: | 1>Login to the app.  2>Select your destination  3>Book taxi/rickshaw according to its rates. |
| Post condition: | Taxi/ Rickshaw booked |

7> Functional Requirement for sending Alert

|  |  |
| --- | --- |
| Function: | Alerting the passengers and sending notifications against thefts and danger one. |
| Precondition: | Station master should register station |
| Steps: | 1>Login to the app.  2>Broadcast notification to the passengers |
| Post condition: | Locker booked |
| Alternate Flow: | If not a user then cannot send alert on the particular station. |

8> Functional Requirement for Searching tourist places

|  |  |
| --- | --- |
| Function: | Searches for places of tourist interest in the area of that station |
| Precondition: | Station master should register station |
| Steps: | 1>Login to the app.  2>Broadcast notification on platform |
| Post condition: | Locker booked |
| Alternate Flow: | If not a user then cannot send alert on the particular station. |

9>Functional Requirement for registering a Complaint

|  |  |
| --- | --- |
| Function: | Registers complaints of the passenger against the services of the railway |
| Precondition: | Passenger should be registered as a user |
| Steps: | 1>Login to the app.  2>Write Complaint |
| Post condition: | Locker booked |

* + 1. **Quality Attributes**
* **Security:**

The security aspect of the system takes into consideration proper authentication when a user is trying to login.

* **Availability:**

The system is available to the user at any station platform and user can make use of the system at that time.

* **Reliability:**

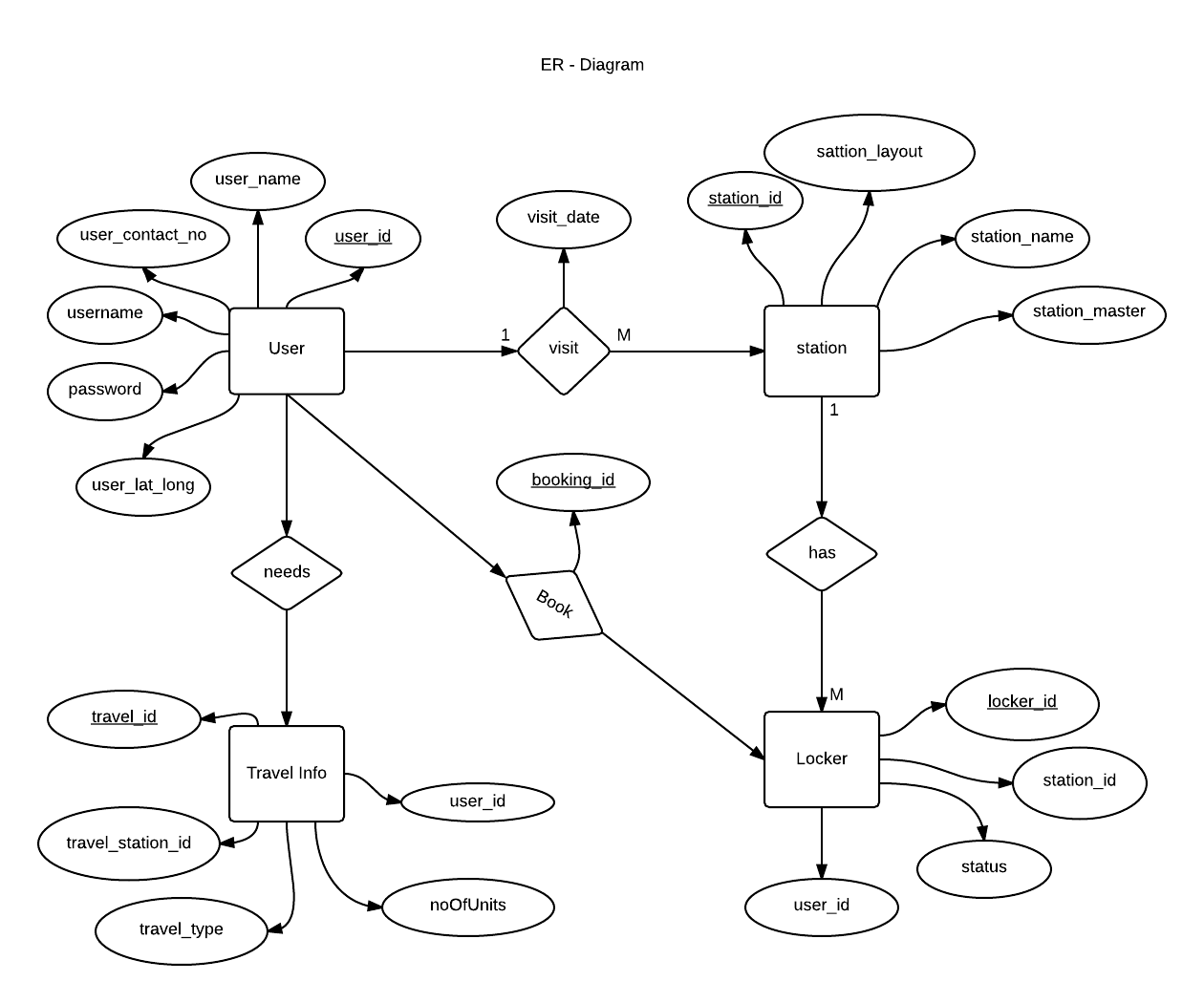
The system is reliable and meets the performance requirements for the access by user.

* **Maintainability:**

The system is modular and hence is flexible enough to allow modification, addition or deletion of layout of platform.The system maintains a database to store the information. Any modification, deletion or addition can be handled using script.



**Chapter 3**

1. **High Level Design**
   1. Entity Relationship Diagram

DESCRIPTION:

The Entity-Relationship diagram above presents a view of the database used in our Rsil commute system. The database mainly consists of the user information, station and station master information, locker information as well as the train information. This information is updated by the admin in the database. The database consists of variety of linked table that consists of data obtained by the station master about the station. The admin feeds in the other necessary data.

* 1. **Block diagram**

Post Complaint

View Station Layout

View Tourist information

Set Reminder for Destination Station

View Upcoming Station

View notifications

Send Important Notification

View and Update Complaints

Add Station Layout

USER INTERFACE

USER INTERFACE

USER REGISTRATION

PASSENGER

STATION MASTER

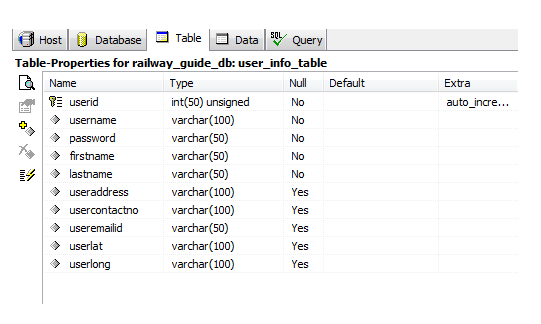
DESCRIPTION: The above block diagram presents the view of the features that out system consists. Each block has a feature represents by the other blocks as shown above. There are two kinds of registrations station master which is provided by the admin at the server side and the passenger who registers himself on his android application. The station master has the functionality of providing the station layout, send notifications and view and update the complaints of the passenger. The passenger on the other hand receives these notifications and provides the complaints regarding the train or station. Passenger can view station layout of the upcoming station and the tourist information.

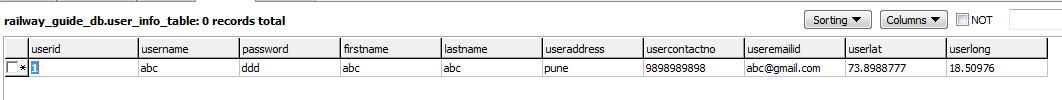
* 1. **Database schema**

This database schema provides the structure of the databases storing the information required.

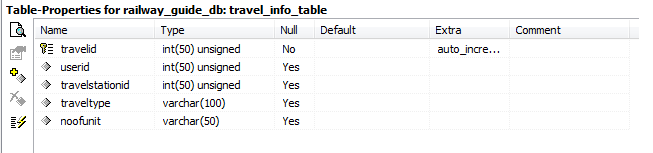
1. **User Database**

*Database schema and data*

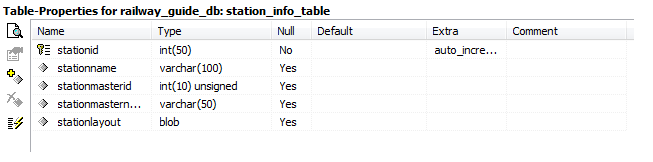




This table stores information about the user that is the passenger who enters these details during registration.

1. **Station Information Table**

*Database Schema and data*

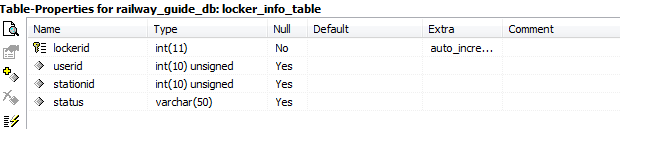


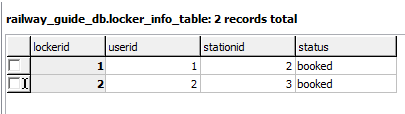


This Table stores the information regarding the station stored by the admin.

1. **Locker Information Table**

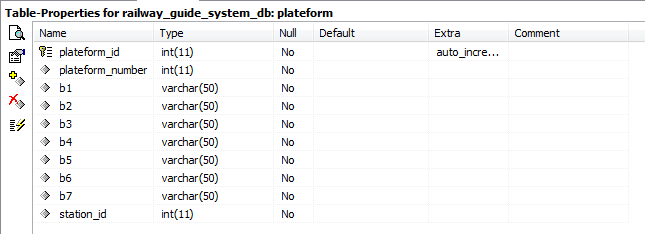
*Databse Schema and data*

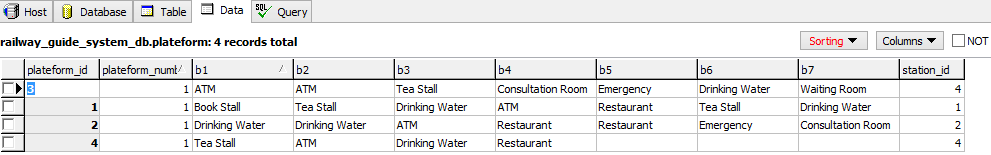




This table stores the record for the lockers booked and available at the particular stations

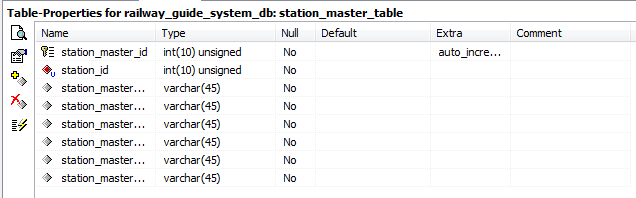
1. **Station Layout Information Table**

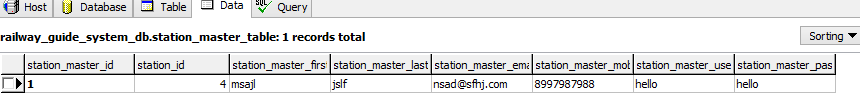
*Database schema and data*

****

This table is used to store the platform layout set according to the station master of that station.

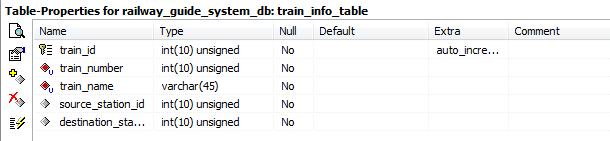
1. **Station Master Information table**

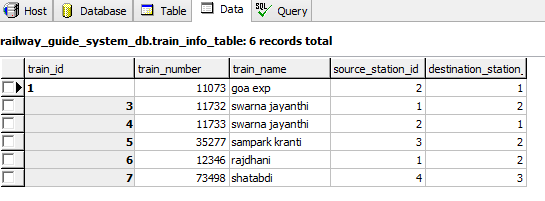
*Database Schema and Data*

****

This table stores information about the station master of the stations.

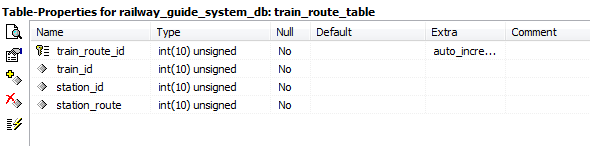
1. **Train Information table**

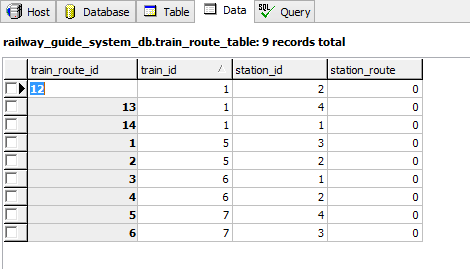
*Database Schema and Data*

****

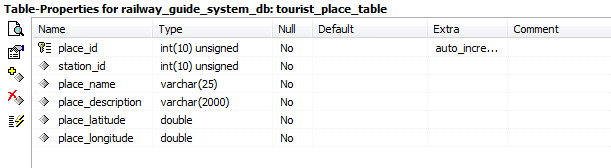
This table consists of the information about the train like the number source station destination station etc.

1. **Train Route Information table**

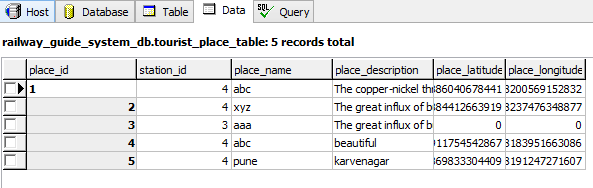
*Database and data*

****

This Table stores the route through which it will pass.it stores all the station ids it will pass through.

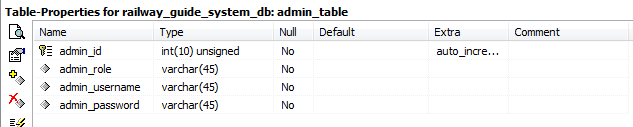
1. **Tourist Place table**

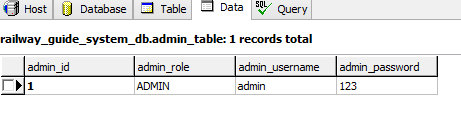
*Database Schema and Data*

**

This table stores information about the tourist places of the particular stations.

1. **Admin Table**

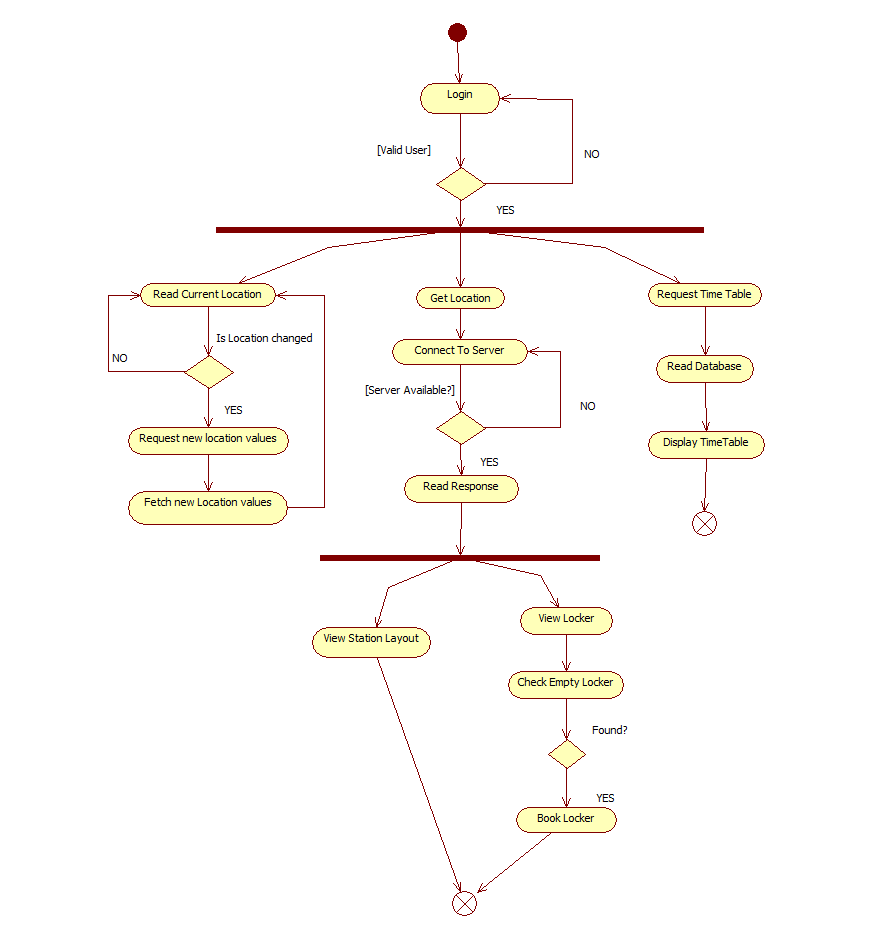
*Database Schema and Data*

****

This table stores the login details for the admin.

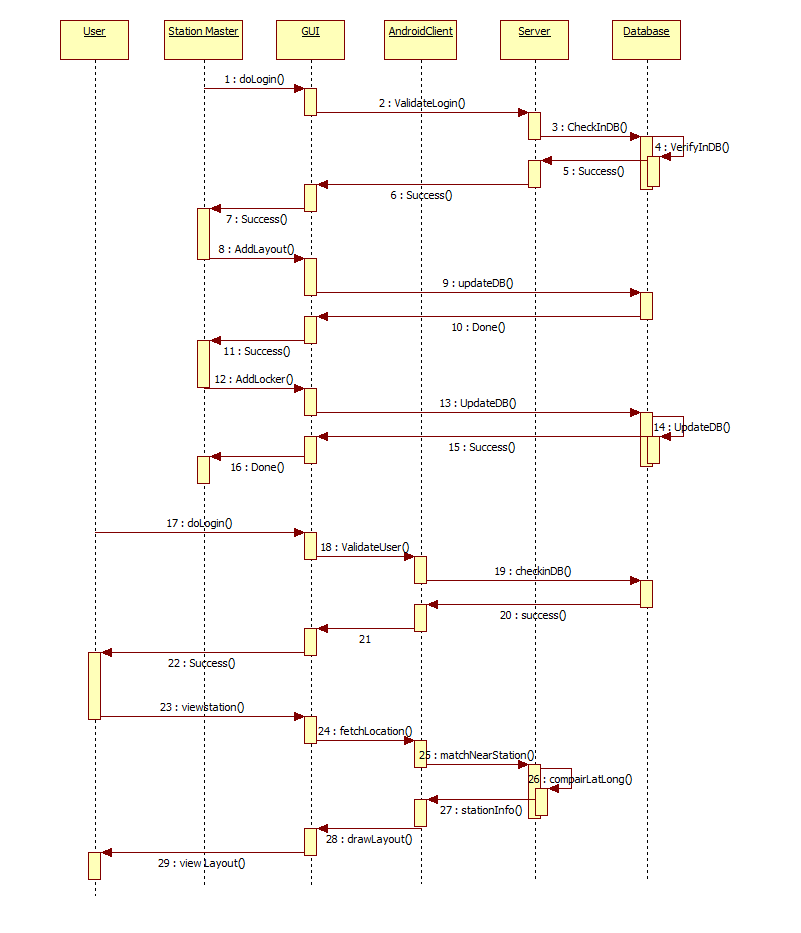
* 1. **UML diagrams**
     1. **Use Case diagram**

This is a use case diagram for Rail Commute System. The actors here are the admin, station master and the passenger. The admin has the functionality as shown above of registration verification and authentication. The passenger has been provided with the features of the application which he/she can access using the user interface. Station master has the functionality to provide the station layout notification and update the complaints of the passengers.

****

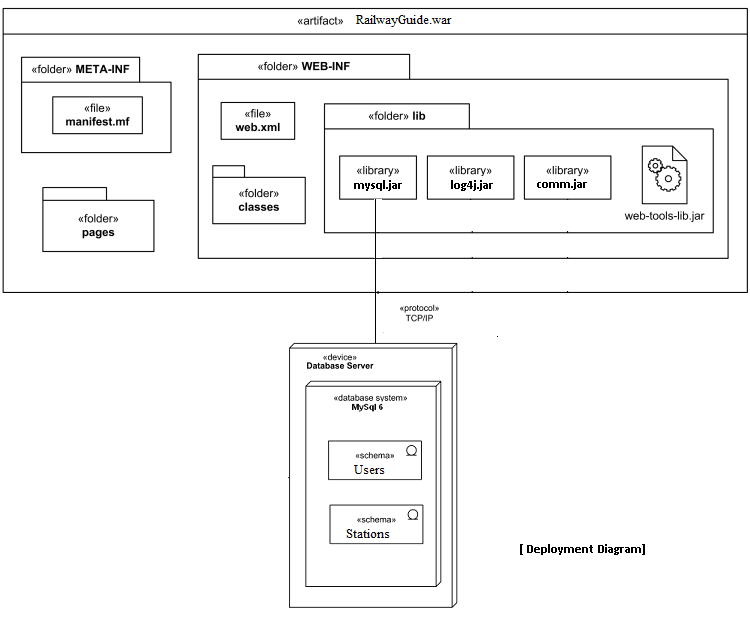
* + 1. **Activity diagram**

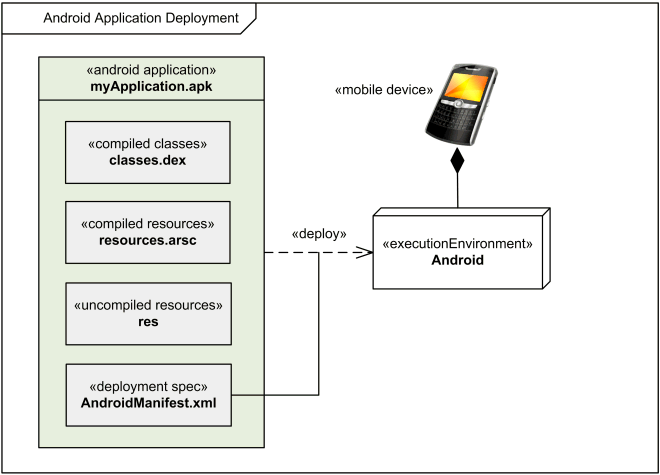
The above id an activity diagram of the railway commute system. Activity in our system starts with a login for user as well as admin and station master. After checkin the authentication of the user , the actors perform various functions. The passenger has to be connected to server at all times to access the data like station layout and to receive notifications. The station master and the admin at the server side add the data necessary for the passenger to view.

* + 1. **Sequence diagram**

This diagram presents the sequence of events in the system. The event start with the user login then on success() of that event passenger can view nearest station and access other features. The station master can provide the station layout and perform other functions. The admin stores the station by getting it coordinates through the google map api and on comparison to the passenger’s coordinate, the approaching station is notified to him/her and also adds other required informations.

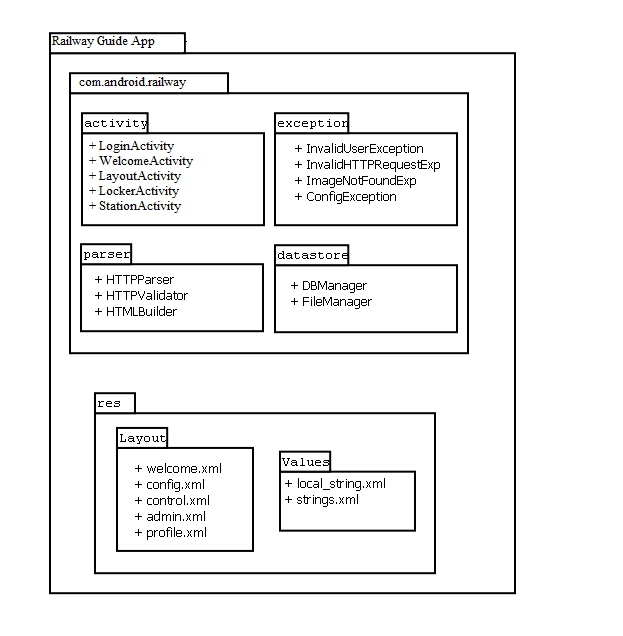
* + 1. **Deployment diagram**

****



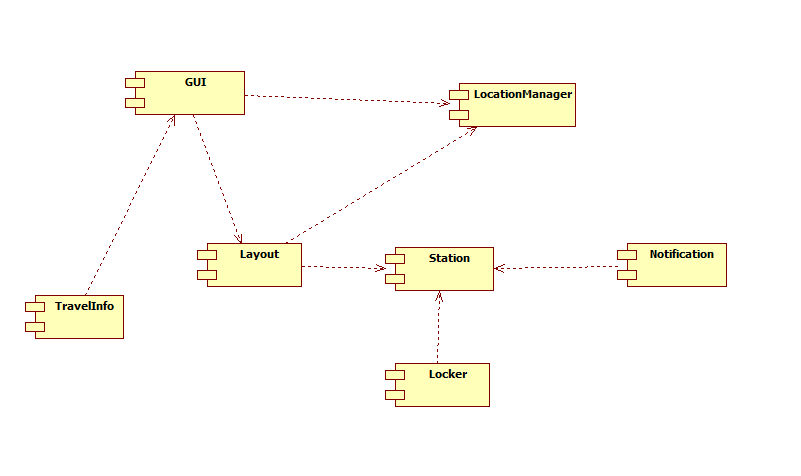
This diagram shows the deployment of our application on the android OS and the tomcat server. It shows the communication of the application with the server side where the admin and station master can access the website and add the required information. The passenger downloads the android APK to access the features of this system.

* + 1. **Package diagram**



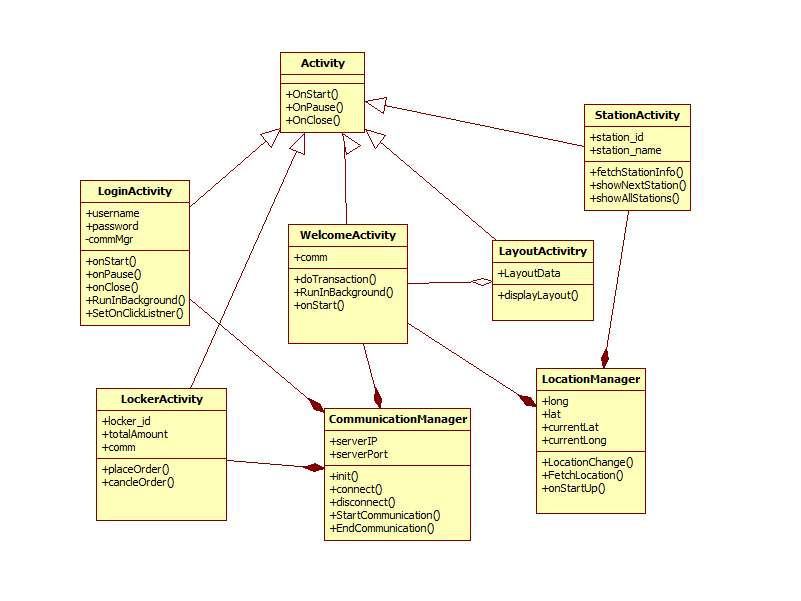
This diagram shows the packages involved in developing this application. The above diagram shows the packages created to store similar type of files together to avoid confusion, reduce time consumption and the code can be viewed easily. The android activity is the client side for the android application. The ‘res’ is the server side for the website to be accessed by the station master and the admin.

* + 1. **Component diagram**



The component diagram shows the components present in the system. GUI is required to access the features of the system by the user. Each Station has a layout, the number of lockers to be booked and the notifications to be sent by its station master.

**3.4.7 Class Diagram**



The class diagram shows the classes used in this system. The activity is the main class consisting of all the other activities which inherit its properties. The relationship with the communication and location manager with the particular activities is shown as above.

**Chapter 4**

**Technology**

**4.1** **JSP(Iava server pages)**

## What is JavaServer Pages?

JavaServer Pages (JSP) is a technology for developing web pages that support dynamic content which helps developers insert java code in HTML pages by making use of special JSP tags, most of which start with <% and end with %>.

## Why Use JSP?

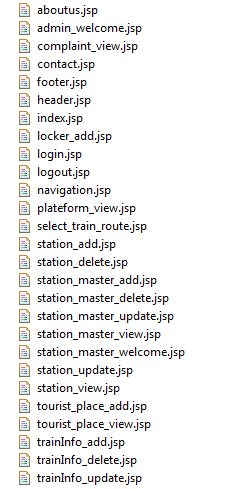
JavaServer Pages often serve the same purpose as programs implemented using the Common Gateway Interface (CGI). But JSP offer several advantages in comparison with the CGI.

* Performance is significantly better because JSP allows embedding Dynamic Elements in HTML Pages itself instead of having a separate CGI files.
* JSP are always compiled before it's processed by the server unlike CGI/Perl which requires the server to load an interpreter and the target script each time the page is requested.
* JavaServer Pages are built on top of the Java Servlets API, so like Servlets, JSP also has access to all the powerful Enterprise Java APIs, including JDBC, JNDI, EJB, JAXP etc.

**Advantages of JSP:**

Following is the list of other advantages of using JSP over other technologies:

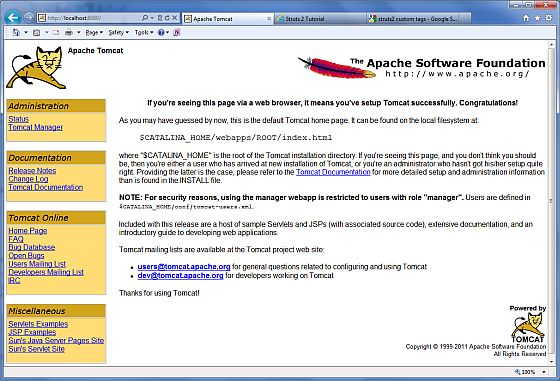
* **Active Server Pages (ASP):** The advantages of JSP are twofold. First, the dynamic part is written in Java, not Visual Basic or other MS specific language, so it is more powerful and easier to use. Second, it is portable to other operating systems and non-Microsoft Web servers.
* **Pure Servlets:** It is more convenient to write (and to modify!) regular HTML than to have plenty of println statements that generate the HTML.
* **Server-Side Includes (SSI):** SSI is really only intended for simple inclusions, not for "real" programs that use form data, make database connections, and the like.



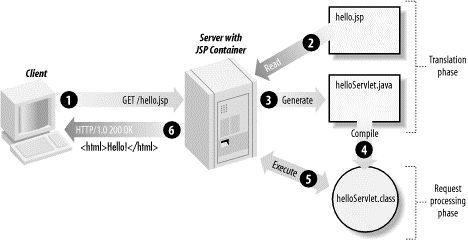
## Setting up Web Server: Tomcat

A number of Web Servers that support JavaServer Pages and Servlets development are available in the market. Some web servers are freely downloadable and Tomcat is one of them.

Apache Tomcat is an open source software implementation of the JavaServer Pages and Servlet technologies and can act as a standalone server for testing JSP and Servlets and can be integrated with the Apache Web Server.



Further information about configuring and running Tomcat can be found in the documentation included here, as well as on the Tomcat web site: http://tomcat.apache.org



## The Scriptlet:

A scriptlet can contain any number of JAVA language statements, variable or method declarations, or expressions that are valid in the page scripting language.

Following is the syntax of Scriptlet:

<% code fragment %>

You can write XML equivalent of the above syntax as follows:

<jsp:scriptlet>

code fragment

</jsp:scriptlet>

Any text, HTML tags, or JSP elements you write must be outside the scriptlet. Following is the simple and first example for JSP:

<html>

<head><title>Hello World</title></head>

<body>

Hello World!<br/>

<%

out.println("Your IP address is " + request.getRemoteAddr());

%>

</body>

</html>

**4.3Servlet**

## What are Servlets?

Java Servlets are programs that run on a Web or Application server and act as a middle layer between a request coming from a Web browser or other HTTP client and databases or applications on the HTTP server.

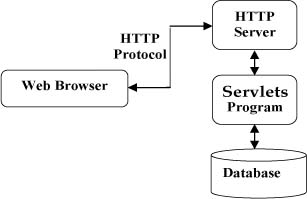
Using Servlets, you can collect input from users through web page forms, present records from a database or another source, and create web pages dynamically.

Java Servlets often serve the same purpose as programs implemented using the Common Gateway Interface (CGI). But Servlets offer several advantages in comparison with the CGI.

* Performance is significantly better.
* Servlets execute within the address space of a Web server. It is not necessary to create a separate process to handle each client request.
* Servlets are platform-independent because they are written in Java.
* Java security manager on the server enforces a set of restrictions to protect the resources on a server machine. So servlets are trusted.
* The full functionality of the Java class libraries is available to a servlet. It can communicate with applets, databases, or other software via the sockets and RMI mechanisms that you have seen already.

## Servlets Architecture:

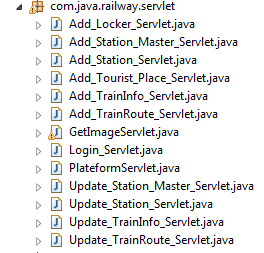
Following diagram shows the position of Servelts in a Web Application.



**Description:**

**Package name: com.java.railway.servlet**

In this package all servlet class is there. Our class extends with httpservlet class.

****

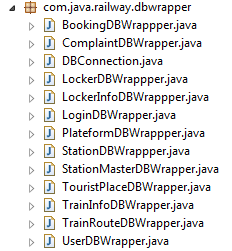
**Login Servlet.java**

This class is used to verify the admin

****

**Package name: com.java.railway.dbwrapper**

In this package all class is related to database operation. Means various class is used to add, update, view delete station info, train info etc

****

**4.4 Mysql database**

## MySQL Database:

MySQL is a fast, easy-to-use RDBMS being used for many small and big businesses. MySQL is developed, marketed, and supported by MySQL AB, which is a Swedish company. MySQL is becoming so popular because of many good reasons:

* MySQL is released under an open-source license. So you have nothing to pay to use it.
* MySQL is a very powerful program in its own right. It handles a large subset of the functionality of the most expensive and powerful database packages.
* MySQL uses a standard form of the well-known SQL data language.
* MySQL works on many operating systems and with many languages including PHP, PERL, C, C++, JAVA, etc.
* MySQL works very quickly and works well even with large data sets.
* MySQL is very friendly to PHP, the most appreciated language for web development.
* MySQL supports large databases, up to 50 million rows or more in a table. The default file size limit for a table is 4GB, but you can increase this (if your operating system can handle it) to a theoretical limit of 8 million terabytes (TB).
* MySQL is customizable. The open-source GPL license allows programmers to modify the MySQL software to fit their own specific environments.

**4.5 Android**

Android is a recently developed operating system designed for mobile devices. It was developed by Google and uses a Linux based kernel, Java compatible libraries along with the just-in-time compiler for development in the Java programming language. It supports many hardware components. Common hardware consists of cameras, a WiFi communications chip, cellular commutations chip, Bluetooth sender and receiver, and a color touch screen. The Android Application Program Interface (API) contains many functions and classes to control the cellular devices. This functionality is all available in a single device with at least a day worth battery life. For this project H.263 was used in development on the Android device. Android is a software stack for mobile devices that includes an operating system, middleware and key applications. The Android SDK provides the tools and APIs necessary to begin developing applications on the Android platform using the Java programming language.

Features

* Application framework enabling reuse and replacement of components
* Dalvik virtual machine optimized for mobile devices
* Integrated browser based on the open source WebKit engine
* Optimized graphics powered by a custom 2D graphics library; 3D graphics based on the OpenGL ES 1.0 specification (hardware acceleration optional)
* SQLite for structured data storage
* Media support for common audio, video, and still image formats (MPEG4, H.264, MP3, AAC, AMR, JPG, PNG, GIF)
* GSM Telephony (hardware dependent)
* Bluetooth, EDGE, 3G, and WiFi (hardware dependent)
* Camera, GPS, compass, and accelerometer (hardware dependent)
* Rich development environment including a device emulator, tools for debugging, memory and performance profiling, and a plugin for the Eclipse IDE

**4.6 SQLite:**

## What is SQLite?

SQLite is an in-process library that implements a self-contained, serverless, zero-configuration, transactional SQL database engine. It is the one database, which is zero-configured, that means like other database you do not need to configure it in your system.

SQLite engine is not a standalone process like other databases, you can link it statically or dynamically as per your requirement with your application. The SQLite accesses its storage files directly.

## Why SQLite?

* SQLite does not require a separate server process or system to operate.(serverless).
* SQLite comes with zero-configuration, which means no setup or administration needed.
* A complete SQLite database is stored in a single cross-platform disk file.
* SQLite is very small and light weight, less than 400KiB fully configured or less than 250KiB with optional features omitted.
* SQLite is self-contained, which means no external dependencies.
* SQLite transactions are fully ACID-compliant, allowing safe access from multiple processes or threads.
* SQLite supports most of the query language features found in the SQL92 (SQL2) standard.
* SQLite is written in ANSI-C and provides simple and easy-to-use API.
* SQLite is available on UNIX (Linux, Mac OS-X, Android, iOS) and Windows (Win32, WinCE, WinRT).

**4.7 Google Map Integration:**

### Google provides this facility using Google play services library which you have to download externally. After downloading, you have to integrate it with your project.In the end one have to integrate your application with Google via Google console

### GOOGLE MAP - ACTIVITY FILE

Google provides GoogleMap and MapFragmentapi to integrate map in your android application. In order to use GoogleMap , you have to create an object of GoogleMap and get the reference of map from the xml layout file.Its syntax is given below:

GoogleMapgoogleMap;

googleMap=((MapFragment)getFragmentManager().findFragmentById(R.id.map)).getMap()

### GOOGLE MAP - LAYOUT FILE

Now you have to add the map fragment into xml layout file. Its syntax is given below:

<fragment

android:id="@+id/map"

android:name="com.google.android.gms.maps.MapFragment"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"/>

### GOOGLE MAP - ANDROIDMANIFEST FILE

The next thing you need to do is to add some permissions along with the Google Map API key in the AndroidManifest.XML file. Its syntax is given below:

<!--Permissions-->

<uses-permissionandroid:name="android.permission.ACCESS\_NETWORK\_STATE"/>

<uses-permissionandroid:name="android.permission.INTERNET"/>

<uses-permissionandroid:name="com.google.android.providers.gsf.permission.READ\_GSERVICES"/>

<uses-permissionandroid:name="android.permission.WRITE\_EXTERNAL\_STORAGE"/>

<!--Google MAP API key-->

<meta-data

android:name="com.google.android.maps.v2.API\_KEY"

android:value="AIzaSyDKymeBXNeiFWY5jRUejv6zItpmr2MVyQ0"/>

## Customizing Google Map:

## You can easily customize google map from its default view , and change it according to your demand.

### ADDING MARKER

You can place a maker with some text over it displaying your location on the map. It can be done by via**addMarker()** method. Its syntax is given below:

finalLatLngTutorialsPoint=newLatLng(21,57);

Marker TP =googleMap.addMarker(newMarkerOptions().position(TutorialsPoint).title("TutorialsPoint"));

### CHANING MAP TYPE

You can also change the type of the MAP. There are four different types of map and each give different view of the map. These types are Normal,Hybrid,Satellite and terrain. You can use them as below

googleMap.setMapType(GoogleMap.MAP\_TYPE\_NORMAL);

googleMap.setMapType(GoogleMap.MAP\_TYPE\_HYBRID);

googleMap.setMapType(GoogleMap.MAP\_TYPE\_SATELLITE);

googleMap.setMapType(GoogleMap.MAP\_TYPE\_TERRAIN);

### ENABLE/DISABLE ZOOM

You can also enable or disable the zoom gestures in the map by calling the**setZoomControlsEnabled(boolean)** method. Its syntax is given below:

googleMap.getUiSettings().setZoomGesturesEnabled(true);

Integrating google maps in your application basically consists of these 4 steps.

1. Download and configure. Google Play Services SDK
2. Obtain API key from google console
3. Specify Android Manifest settings

## Download and configure. Google Play Services SDK

### INSTALL GOOGLE SERVICES SDK

Open your SDK manager in the eclipse by clicking the Window and then selecting the Android SDK manager.

Navigate to the extras tab and select the Google play services and click on install this package. It would be like this.

### IMPORT SDK TO ECLIPSE

After you download the SDK , click on file tab and select import option. Select existing android application code and press ok. Browse to your android folder and then sdk folder. In sdk folder expand extras folder. Expand google folder and select google play services.

### CONFIGURE YOUR PROJECT WITH SDK

After you import the SDK , you have to add it into your project. For this , right click on your eclipse project and select properties. Select android from left tab and then select add from right below panel and add the project. It would be like this

**Obtaining the API key**

This part is furthur divided into two steps. First you have to get an SHA1 fingerprint key from your pc and then you have to get map API key from google console.

### GETTING CERTIFICATE FROM KEYTOOL

You need to get a certificate key because you have to provide it to google console in order to get your API key for map.Open your command prompt and move to the path where your java jre has been placed. Now type this command.

keytool-list -v -aliasandroiddebugkey-keystore%%Your path%%-storepass android -keypass android

Replace the percentage part of the command with the path which you will copy from by selecting the window tab and selecting the preferences tab and then selectng the build option under android from left side.

Copy the default debug keystore path and replace it in the cmmand and hit enter. The following result would appear.

Copy the SHA1 key because you need it in the next step.

### GETTING KEY FROM GOOGLE CONSOLE

Open [Google Console](https://code.google.com/apis/console/) and sign in by clicking a new project.

Click on services from the left tab and then navigate to the Google Maps Android API v2. You have to turn them on like this.Now again go to the left tab and select API access. And click on create new android key. Now paste the key that you copied and put a semicolon and paste your project name and click create. It would be like this.Now copy the API key that has been given to your by android , because you have to paste it into your manifest file.

**Specify Android Manifest settings**

The final step is to add the API key to your application. Open your manifest file and place this code right before closing the application tag.

**Adding Google Maps to your application.**

Following is the content of the modifed main activity file**src/com.example.googlemaps/MainActivity.java**.

Let's try to run your GoogleMaps application. I assume you have connected your actual Android Mobile device with your computer. To run the app from Eclipse, open one of your project's activity files and click Run Eclipse Run Icon icon from the toolbar. Before starting your application, Eclipse will display following window to select an option where you want to run your Android application.



Now what you need to do is to tap on the ballon to see the text.

Now you can customize the google map according to your choice with the functions given in the GoogleMap API.

**Chapter 5**

**Implementation and Coding**

**5.1 MODULES**

**5.1.1 Passenger Module**

The passenger can register to use the app creating id, password. The passenger can login to the system with its own id and password created at the time of registration. The passenger will be receiving the notification of arrival of the next station. The passenger can view the whole platform wise layout of a particular station. The passenger can set a reminder of the particular station, which when arrives, the application will send an alert to the passenger. The passenger will be able to view the amount of time the train is going to wait at the particular station. The passenger will be provided with nearest tourist places of its respective destination.

**5.1.2 Station Master Module**

The station master can login to his account of his station. The station master can feed in data of his station. The station master can send important notification to the passenger on the station.

**5.1.3 Station layout module**

End user can search for next coming station and can see the layout of next station. It check all shops, tea stall, hotel on the coming station, number of platform, on which platform train will arrive etc. Station master add the station layout, information of station.

**5.1.4 Notification Module**

The passenger can set a reminder of the particular station, which when arrives, the application will send an alert. The passenger will be able to view the amount of time the train is going to wait at the particular station. All information related to the halt time of the train, train route from source to destination station is provided by the application. The passenger can receive alerts and alarms for protection against thefts to take precaution.

**5.1.5 Station information Module**

Whenever the train has been delayed the passengers will be notified of this delay and the new arrival time.

**5.1.6 Complaints Module**

Passenger can post complaints about the services in the train..

**5.1.7 Locker Module**

Number of locker, available locker, booked locker all information is added by the station master. Our user can access all this information and passenger can book the locker available.

**5.2 Flow Diagram:**

Fetch Current location

Give Train no as input

Registration and login

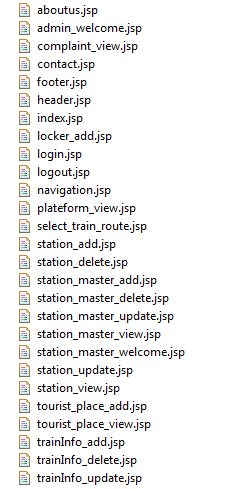
Any Upcoming Station?

Booking clock-room / view nearby places

View Notifications

Show Station layout

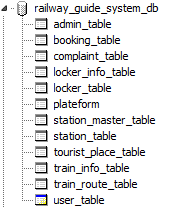
* 1. **Implementation Details**
     1. ServerSide
* **JSP Pages in our project:**



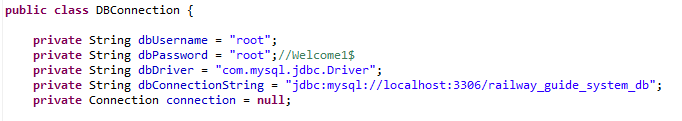
* **How database works in our project?**

1 database in our project

Database name: railway\_guide\_system\_db



First connect database through connect method in our project



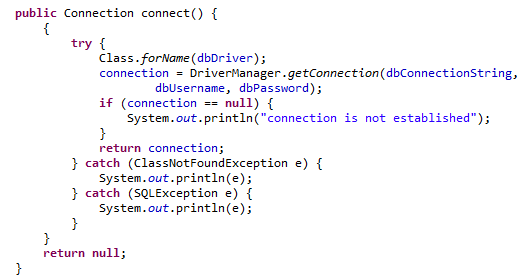
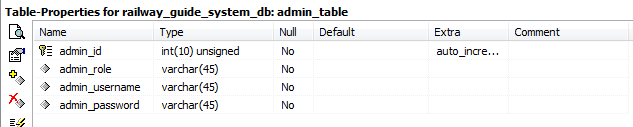
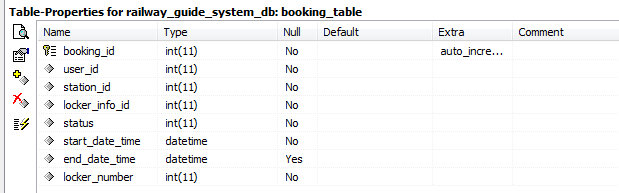
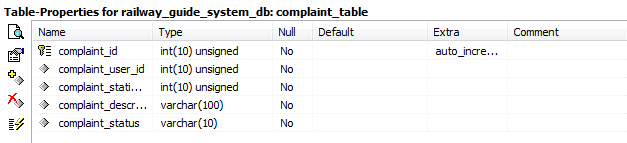
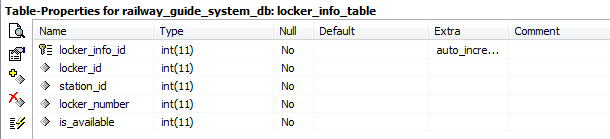


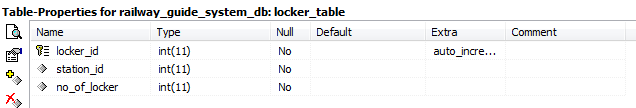
Table schema for railway\_guide\_system\_db database

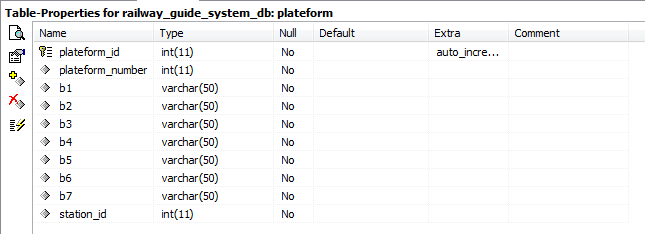


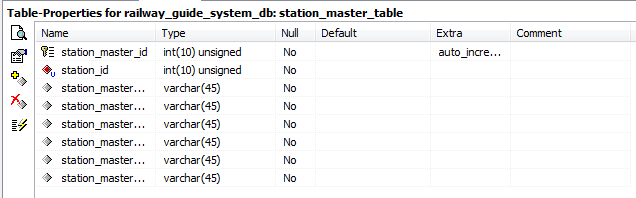




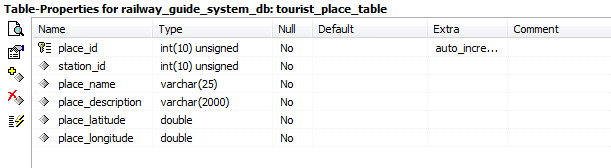


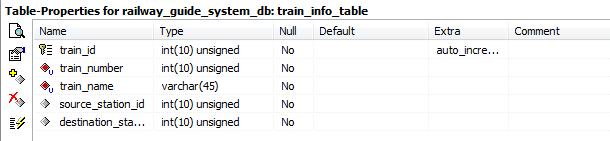


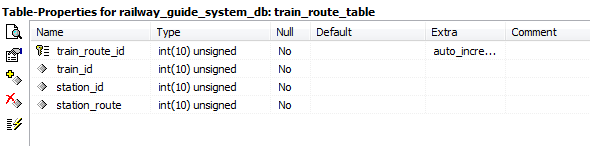


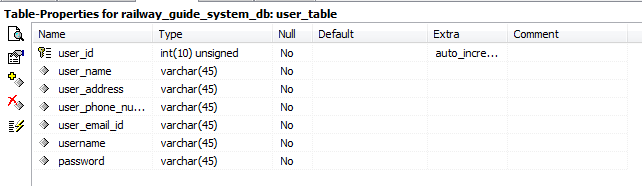




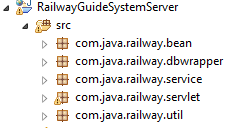








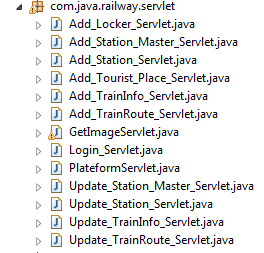
* **Class wise description**



* **Description:**

**Package name: com.java.railway.servlet**

In this package all servlet class is there. Our class extends with httpservlet class.

****

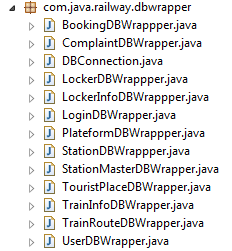
**Login Servlet.java**

This class is used to verify the admin

****

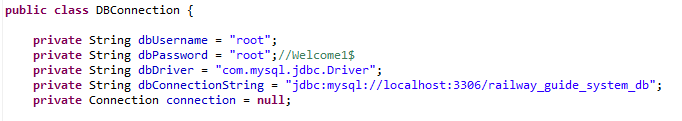
**Package name: com.java.railway.dbwrapper**

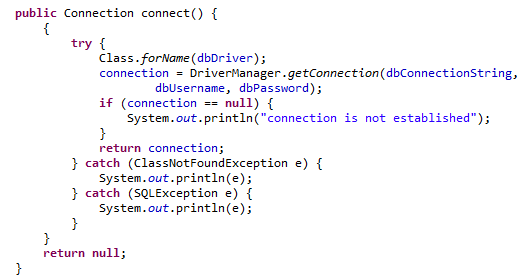
In this package all class is related to database operation. Means various class is used to add, update, view delete station info, train info etc

****

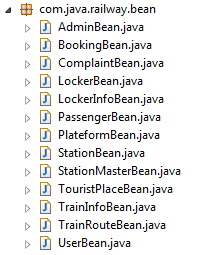
**DBConnection.java**

This is generic class and is used to make connection with Mysql database.





**Package name:com.java.railway.bean**

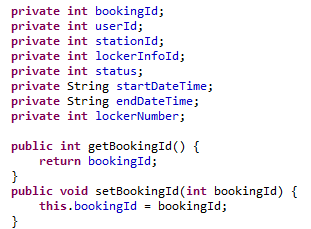


**Admin\_Bean.java**

This class is the place holder class and contains only getter setter methods to access values

**BookingBean.java**

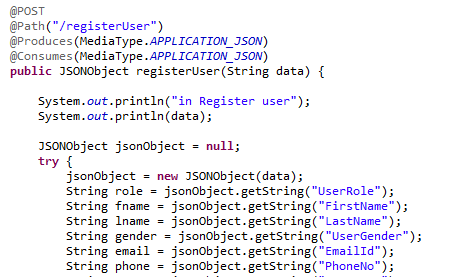
This class is the place holder class and contains only getter setter methods to access values

****

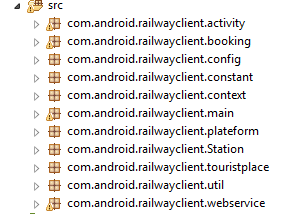
****

**Webservice.java**

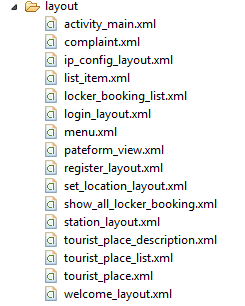
This class is responsible for client server Communication. When client sent request to server then it connect from this class



* + 1. **Client Side**
* **Class wise Description Client Side**

****

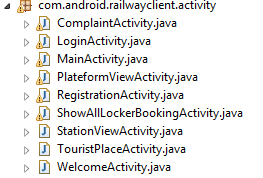
* **Layout**

In layout all user interface pages are given

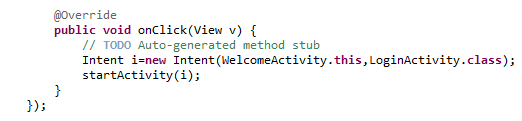
* **Description:**

Among the many activity which is the first one is given in manifest file as shown below

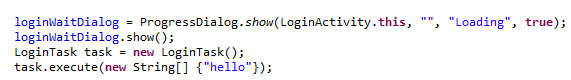
****

****

* **WelcomeActivty.java:**

This class is first time called. In this layout we gave one buttons for start. And option for entering ip address.

* **LoginActivity.java**

In this class we validate the user from server side. First we took data from login page then we use Login Task to communicate with server.

* 1. **Code Snippet**

**5.4.1 MAP ACTIVITY**

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_map);

stationBeanList = ((ApplicationContext) this.getApplicationContext()).getStationBeanList();

try {

MapsInitializer.initialize(this);

} catch (GooglePlayServicesNotAvailableException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

seekerMapFragment = (MapFragment) getFragmentManager()

.findFragmentById(R.id.mapView);

seekerMap = seekerMapFragment.getMap();

LatLng latlong = new LatLng(18.4, 73.8);

CameraPosition cameraPosition = new CameraPosition.Builder().target(latlong).zoom(6).build();

seekerMap.animateCamera(CameraUpdateFactory.newCameraPosition(cameraPosition));

**5.4.2WebServiceParser**

public ArrayList<AnnouncementBean> getAnnouncementInfo(Map<String, String> params) {

String response = wSCommunictor.invokeMethod("getAnnouncementInfo",params);

ArrayList<AnnouncementBean> announcementBeanList = new ArrayList<AnnouncementBean>();

try {

if (response != null) {

System.out.println(response);

Log.d("getAnnouncementInfo\_webServiceParser", response);

JSONArray jsonArray = new JSONArray(response);

for (int i = 0; i < jsonArray.length(); i++) {

AnnouncementBean announcementBean = null;

JSONObject jsonObject = new JSONObject(jsonArray.getString(i));

if (jsonObject != null) {

announcementBean = new AnnouncementBean(); announcementBean.setAnnouncementId(jsonObject.getInt("AnnouncementId"));

announcementBean.setTrainId(jsonObject.getInt("TrainId"));

announcementBean.setStationId(jsonObject.getInt("StationId"));

announcementBean.setAnnouncementName(jsonObject.getString("AnnouncementName"));

announcementBean.setAnnouncementDescription(jsonObject.getString("AnnouncementDescription")); announcementBean.setAnnouncementDate(jsonObject.getString("AnnouncementDate")); announcementBeanList.add(announcementBean);

}

}

}

} catch (Exception e) {

e.printStackTrace();

**5.4.3 DBConnection**

public Connection connect() {

{

try {

Class.forName(dbDriver);

connection = DriverManager.getConnection(dbConnectionString,

dbUsername, dbPassword);

if (connection == null) {

System.out.println("connection is not established");

}

return connection;

} catch (ClassNotFoundException e) {

System.out.println(e);

} catch (SQLException e) {

System.out.println(e);

}

}

return null;

}

**5.4.4 UserBean**

public int getUserId() {

return userId;

}

public void setUserId(int userId) {

this.userId = userId;

}

public String getUserName() {

return userName;

}

public void setUserName(String userName) {

this.userName = userName;

}

public String getUserAddress() {

return userAddress;

}

public void setUserAddress(String userAddress) {

this.userAddress = userAddress;

}

public String getUserPhoneNumber() {

return userPhoneNumber;

}

public void setUserPhoneNumber(String userPhoneNumber) {

this.userPhoneNumber = userPhoneNumber;

}

public String getUserMailId() {

return userMailId;

}

public void setUserMailId(String userMailId) {

this.userMailId = userMailId;

}

public String getUsername() {

return username;

}

public void setUsername(String username) {

this.username = username;

}

public String getPassword() {

return password;

}

public void setPassword(String password) {

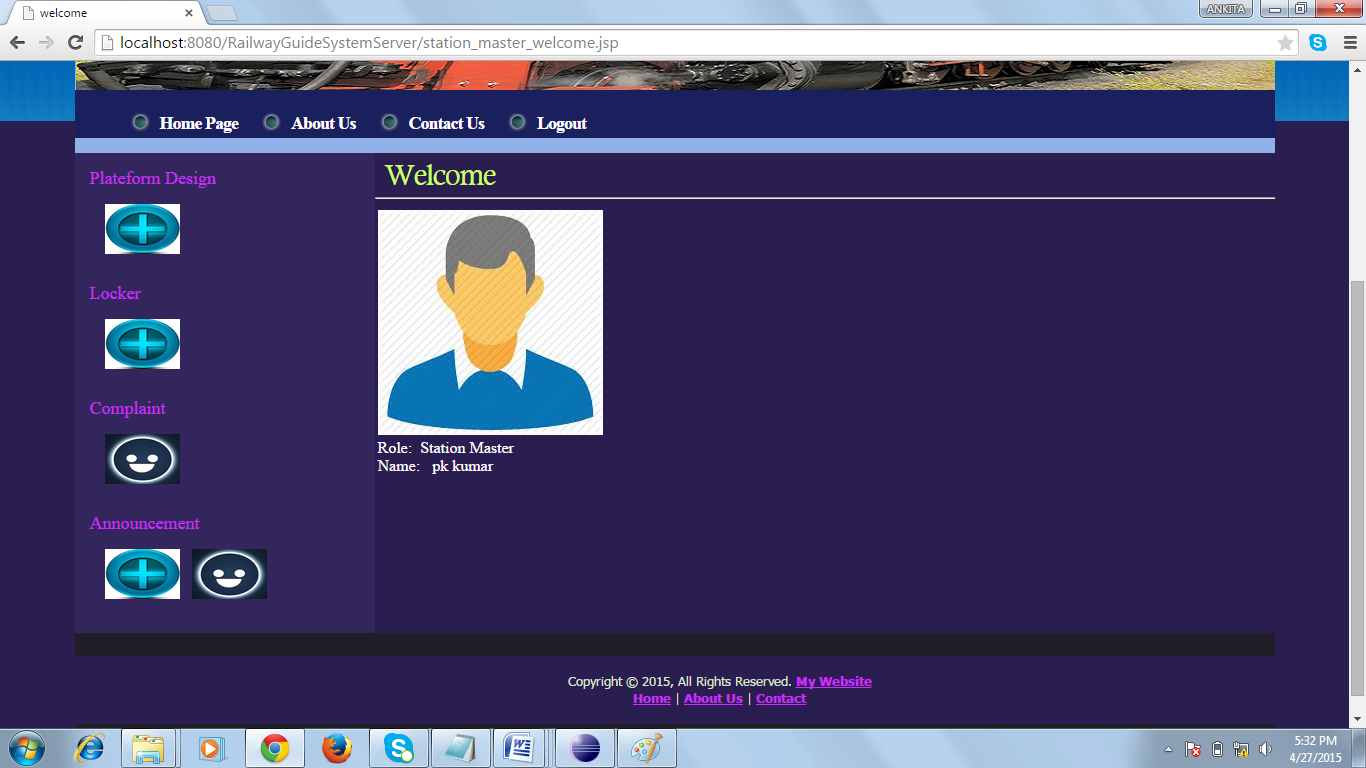
this.password = password;

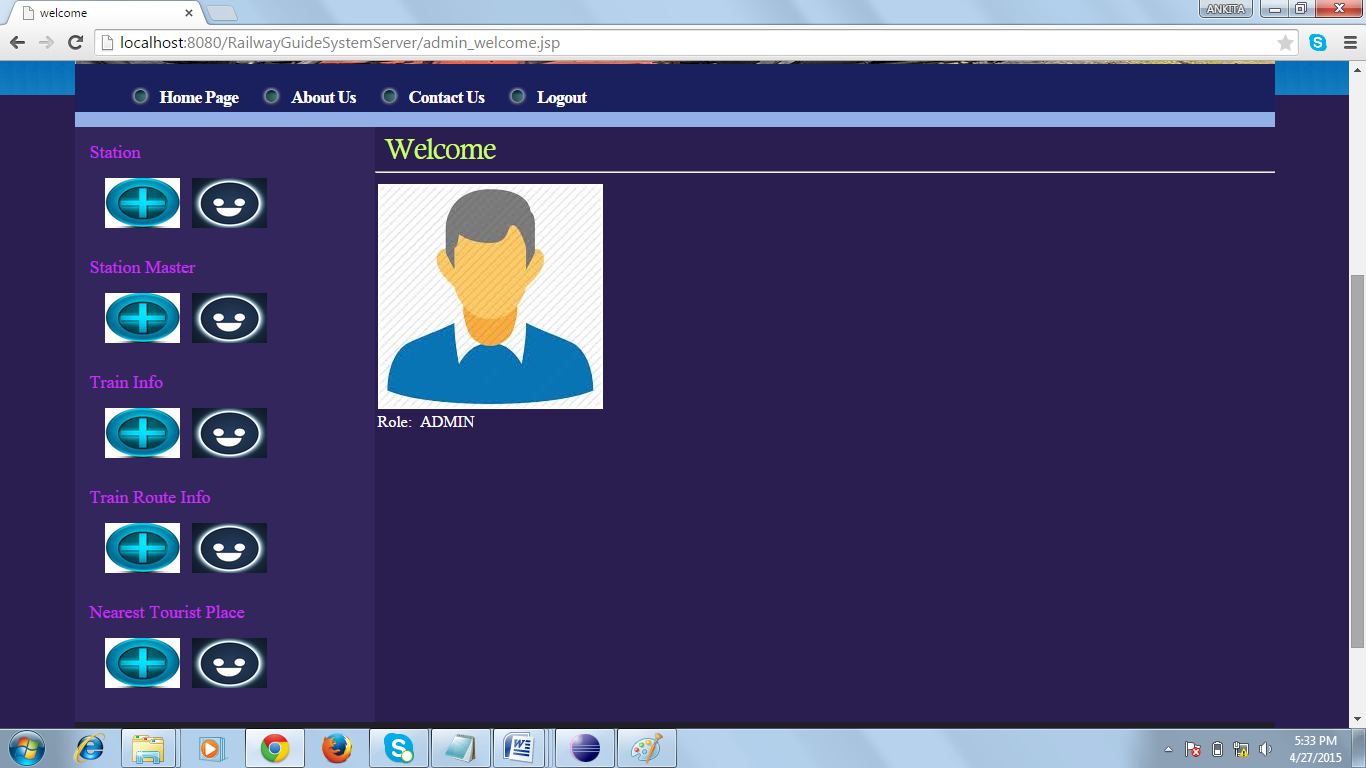
}

* 1. **Screenshots**

**Server side**

****

****



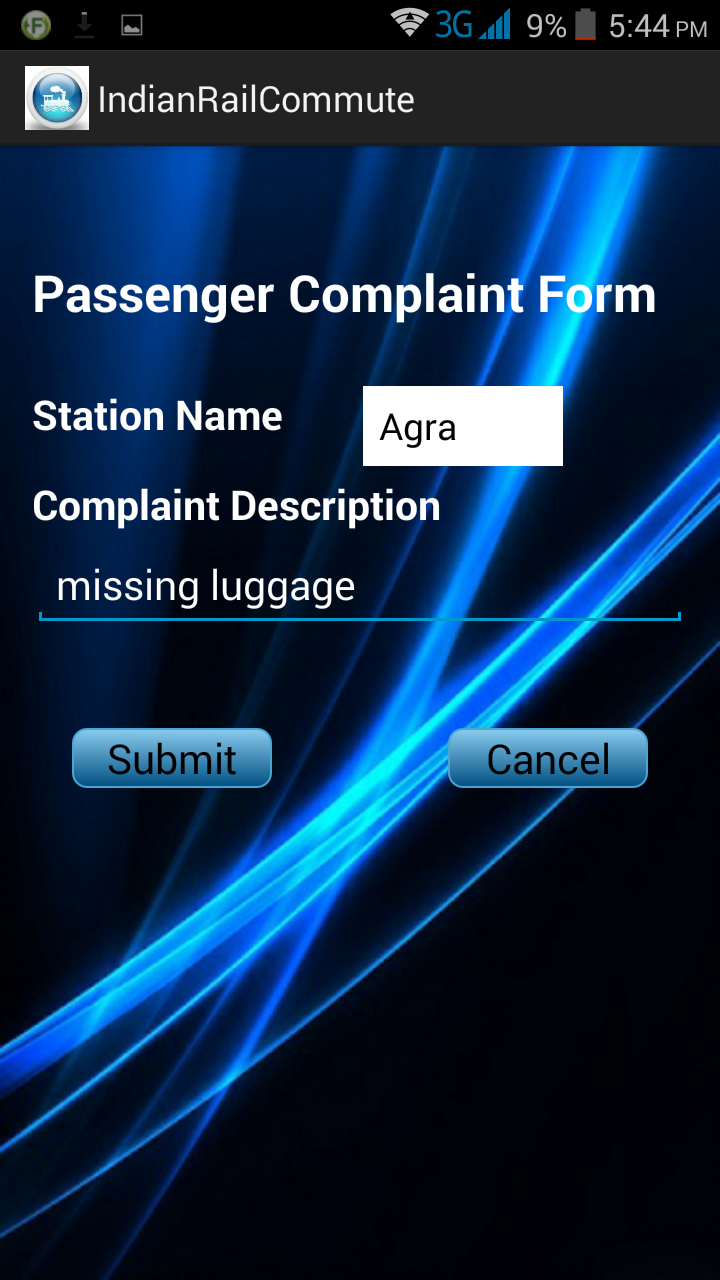
**Client Side:**

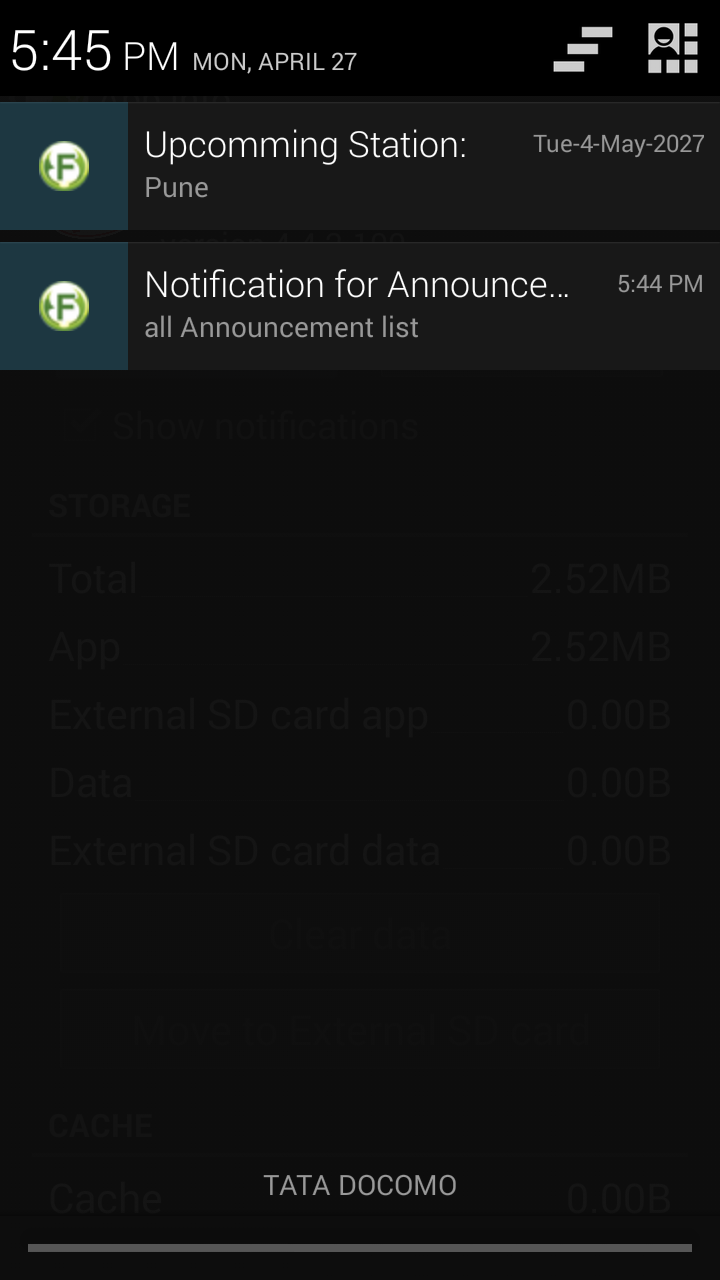










****

**Chapter 6: Software Testing**

**TEST CASES:**

**Unit test:**

|  |  |
| --- | --- |
| Test Case ID | 1 |
| Test Case Description | War file should get installed properly. |
| Steps | 1.Install the application inside Tomcat.  2. Run the application. |
| Test Case Result | Application should be successful installed. |
| Action Result | Application successfully started. |
| Status | Pass |

|  |  |
| --- | --- |
| Test Case ID | 2 |
| Test Case Description | After running application web page should be open on browser |
| Steps | 1.Install the application.  2. Open Mozilla browser  3. Type URL address |
| Test Case Result | Application should be successfully started. |
| Action Result | Application successfully started and displayed welcome page |
| Status | Pass |

|  |  |
| --- | --- |
| Test Case ID | 3 |
| Test Case Description | Application should provide login to admin only |
| Steps | 1. Install and Run the application  2. Login with admin |
| Test Case Result | Click on login insert valid username and password of admin, Then successfully login to next page |
| Action Result | Click on login insert valid username and password of admin, Then successfully login to next page |
| Status | Pass |

|  |  |
| --- | --- |
| Test Case ID | 4 |
| Test Case Description | System should not provide login to other than admin |
| Steps | 1.Install the application.  2. Run the application  3. Click on login link |
| Test Case Result | Insert invalid username and password, then application stay on same page and give proper error message |
| Action Result | Same as expected |
| Status | Pass |

|  |  |
| --- | --- |
| Test Case ID | 5 |
| Test Case Description | Functionality on admin page. |
| Steps | 1. Run the application  2.Login as admin |
| Test Case Result | For admin all options are given through link |
| Action Result | For admin all options are given through link |
| Status | Pass |

|  |  |
| --- | --- |
| Test Case ID | 6 |
| Test Case Description | Check functionality for different link |
| Steps | 1. Run the application  2.Login as admin |
| Test Case Result | Click on the link, then according to link next page is opened |
| Action Result | Click on the link, then according to link next page is opened |
| Status | Pass |

|  |  |
| --- | --- |
| Test Case ID | 7 |
| Test Case Description | Admin can insert new station |
| Steps | 1. Run the application  2.Login as admin  3.Click on add option |
| Test Case Result | Click on stations option add link, then admin add new station |
| Action Result | Click on stations option add link, then admin add new station |
| Status | Pass |

|  |  |
| --- | --- |
| Test Case ID | 8 |
| Test Case Description | Admin can add new Station Master. |
| Steps | 1. Run the application  2.Login as admin  3.Click on add option |
| Test Case Result | Click on Station Master’s option add link, then admin add new Station Master |
| Action Result | Click on Station Master’s option add link, then admin add new Station Master |
| Status | Pass |

|  |  |
| --- | --- |
| Test Case ID | 9 |
| Test Case Description | Admin can add Train Info |
| Steps | 1. Run the application  2.Login as admin  3.Click on add option |
| Test Case Result | Click on Train Info’s option add link, then admin add Train Info |
| Action Result | Same as expected |
| Status | Pass |

|  |  |
| --- | --- |
| Test Case ID | 10 |
| Test Case Description | Admin can add train root |
| Steps | 1. Run the application  2.Login as admin  3.Click on add option |
| Test Case Result | Click on train root’s option add link, then admin add new train root |
| Action Result | Click on train root’s option add link, then admin add new train root |
| Status | Pass |

|  |  |
| --- | --- |
| Test Case ID | 11 |
| Test Case Description | Admin can add nearest tourist place. |
| Steps | 1. Run the application  2.Login as admin  3.Click on add option |
| Test Case Result | Click on nearest tourist place’s option add link, then admin add new nearest tourist place |
| Action Result | Click on nearest tourist place’s option add link, then admin add nearest tourist place |
| Status | Pass |

|  |  |
| --- | --- |
| Test Case ID | 12 |
| Test Case Description | Admin can view station |
| Steps | 1. Run the application  2.Login as admin  3.Click on view option |
| Test Case Result | Click on stations option view link, then admin view all registered station |
| Action Result | Click on stations option view link, then admin view all registered station |
| Status | Pass |

|  |  |
| --- | --- |
| Test Case ID | 13 |
| Test Case Description | Admin can view Station Master information |
| Steps | 1. Run the application  2.Login as admin  3.Click on view option |
| Test Case Result | Click on Station Master option view link, then admin view Station Master information |
| Action Result | Click on users option view link, then admin view Station Master information |
| Status | Pass |

|  |  |
| --- | --- |
| Test Case ID | 14 |
| Test Case Description | Admin can view registered Train Info |
| Steps | 1. Run the application  2.Login as admin  3.Click on view option |
| Test Case Result | Click on Train Info option view link, then admin view Train Info |
| Action Result | Click on Train Info option view link, then admin view Train Info |
| Status | Pass |

|  |  |
| --- | --- |
| Test Case ID | 15 |
| Test Case Description | Admin can update station information |
| Steps | 1. Run the application  2.Login as admin  3.Click on update option |
| Test Case Result | Click on stations option update link, then admin can update station information |
| Action Result | Click on stations option update link, then admin can update station information |
| Status | Pass |

|  |  |
| --- | --- |
| Test Case ID | 16 |
| Test Case Description | Admin can update registered Station Master information |
| Steps | 1. Run the application  2.Login as admin  3.Click on update option |
| Test Case Result | Click on Station Master option update link, then admin update Station Master information |
| Action Result | Click on Station Master option update link, then admin update Station Master information |
| Status | Pass |

|  |  |
| --- | --- |
| Test Case ID | 17 |
| Test Case Description | Admin can update Train Info information |
| Steps | 1. Run the application  2.Login as admin  3.Click on update option |
| Test Case Result | Click on Train Info option update link, then admin update Train Info information |
| Action Result | Click on Train Info option update link, then admin update Train Info information |
| Status | Pass |

|  |  |
| --- | --- |
| Test Case ID | 18 |
| Test Case Description | Admin can delete station information. |
| Steps | 1. Run the application  2.Login as admin  3.Click on delete option |
| Test Case Result | Click on station option delete link, then admin can delete station |
| Action Result | Click on station option delete link, then admin can delete station |
| Status | Pass |

|  |  |
| --- | --- |
| Test Case ID | 19 |
| Test Case Description | Admin can delete Station Master |
| Steps | 1. Run the application  2.Login as admin  3.Click on delete option |
| Test Case Result | Click on users option delete link, then admin can delete Station Master |
| Action Result | Click on users option delete link, then admin can delete Station Master |
| Status | Pass |
|  |  |

|  |  |
| --- | --- |
| Test Case ID | 20 |
| Test Case Description | Admin can delete Train Info. |
| Steps | 1. Run the application  2.Login as admin  3.Click on delete option |
| Test Case Result | Click on Train Info option delete link, then admin can delete Train Info |
| Action Result | Click on Train Info option delete link, then admin can delete Train Info |
| Status | Pass |

|  |  |
| --- | --- |
| Test Case Result | Click on station option delete link, then admin can delete station |
| Action Result | Click on station option delete link, then admin can delete station |
| Status | Pass |

|  |  |
| --- | --- |
| Test Case ID | 21 |
| Test Case Description | Admin can delete Station Master |
| Steps | 1. Run the application  2.Login as admin  3.Click on delete option |
| Test Case Result | Click on users option delete link, then admin can delete Station Master |
| Action Result | Click on users option delete link, then admin can delete Station Master |
| Status | Pass |

|  |  |
| --- | --- |
| Test Case ID | 22 |
| Test Case Description | Apk should get installed properly. |
| Steps | 1.Install the application.  2.Open the application. |
| Test Case Result | Application should be successful installed. |
| Action Result | Application successfully started. |
| Status | Pass |

|  |  |
| --- | --- |
| Test Case ID | 23 |
| Test Case Description | After clicking on the android icon, application should be started. |
| Steps | 1.Install the application.  2. First click on the menu  3. Search the application  4.Run application as android application. |
| Test Case Result | Application should be successfully started. |
| Action Result | Application successfully started. |
| Status | Pass |

|  |  |
| --- | --- |
| Test Case ID | 24 |
| Test Case Description | System should provide the menu from which user can select the option for enter ip address. |
| Steps | 1.Click on the menu button.  2. Option menu should get open . |
| Test Case Result | After clicking on the menu system should perform the respective operation. |
| Action Result | After clicking on the menu system should perform the respective operation. |
| Status | Pass |

|  |  |
| --- | --- |
| Test Case ID | 25 |
| Test Case Description | System should go to login page after click on button start, After filling Ip address. |
| Steps | 1.Run application as android application.  2.Insert IP addrees  3. Click the start button. |
| Test Case Results | After click on start button application shows next page i.e login page. |
| Action Result | Same as expected |
| Status | Pass |

|  |  |
| --- | --- |
| Test Case ID | 26 |
| Test Case Description | System should provide the login to the validate users only |
| Steps | 1. Install the application.  2. First click on the menu  3. Search the application  4. Run application as android application.  5. Go to login page. |
| Test Case Result | System should provide the username password fields. |
| Action Result | Insert valid username and password, and submit. System logged in successfully |
| Status | Pass |

|  |  |
| --- | --- |
| Test Case ID | 27 |
| Test Case Description | System should provide different buttons for searching train, nearest tourist place, lockers etc |
| Steps | 1. Install the application.  2. First click on the menu  3. Search the application  4. Run application as android application.  5. Go to login page. |
| Test Case Result | After login as validate user, next page is opened with various button options |
| Action Result | After login as admin, next page is opened with various button options |
| Status | Pass |

|  |  |
| --- | --- |
| Test Case ID | 28 |
| Test Case Description | System should not put the phone  On flight mode while running. |
| Steps | 1.Run application as android application.  2.Click on start button. |
| Test Case Result | Application should run and phone should be in on mode. |
| Action Result | Running properly. |
| Status | Pass |

**Performance Testing**

|  |  |
| --- | --- |
| Test Case ID | 29 |
| Test Case Description | Phone should not get hang due to application. |
| Steps | 1.Run application as android application.  2.Click on start button. |
| Test Case Result | Application should run. |
| Action Result | Application running and phone working properly. |
| Status | Pass |

|  |  |
| --- | --- |
| Test Case ID | 30 |
| Test Case Description | System should not disturb telephony activity. |
| Steps | 1.Run application as android application.  2.Click on start button.  3.User can receive incoming calls. |
| Test Case Result | Application should not block incoming calls. |
| Action Result | User can receive calls. |
| Status | Pass |

|  |  |
| --- | --- |
| Test Case ID | 31 |
| Test Case Description | System should not block incoming massages. |
| Steps | 1.Run application as android application.  2.Click on start button.  3.User can receive incoming massages. |
| Test Case Result | Application should not block incoming massages. |
| Action Result | User can receive massages. |
| Status | Pass |

|  |  |
| --- | --- |
| Test Case ID | 32 |
| Test Case Description | Application should not run as background service. |
| Steps | 1.Run application as android application.  2.Click on start button. |
| Test Case Result | Application should not run as background service. |
| Action Result | Application is not running as background service. |
| Status | Pass |

|  |  |
| --- | --- |
| Test Case ID | 33 |
| Test Case Description | Application should use minimum battery. |
| Steps | 1.Run application as android application.  2.Click on start button. |
| Test Case Result | Application should use minimum battery. |
| Action Result | Application battery usage is minimum. |
| Status | Pass |

**Appendix A- Mathematical Model**

1. Set Theory
2. Let ‘S’ be the ‘Railway Guide Application’.

S= {…………….}

Set S is divided into 6 modules

S= {S1, S2, S3, S4, S5, S6}

S1= GUI Handler (GH)

S2= Location Manager (LM)

S3= Station Layout Logic (SLL)

S4= Locker Module (LM)

S5= Communication Manager (CM)

S6= Travel Guide (TG)

1. Identify the inputs.

Inputs = {X1, X2, X3, ……..Xn}

X1= Location Information

X2= Locker Information

1. Identify the output as O.

Outputs = {Y1, Y2, Y3, ……..Yn}

Y1= Station Layout

Y2= Locker Room

Y3= Travel Info

**SET THEORY**

|  |  |  |
| --- | --- | --- |
| **Sr No.** | **Description** | **UML Design observations.** |
| **1** | **Problem Description** |  |
|  | Let S be a system which do Railway Guide Application; suchthat S = {S1,S2,S3,S4,S5, S6} where S1 represents GUI Handler (GH); S2 represents Location Manager (LM); S3 represents Station Layout Logic (SLL); S4 represents Locker Module (LM); S5 represents Communication Manager (CM); S6 represents Travel Guide (TG) | S holds list of modules in the system |
| **2** | **Activities** |  |
|  | **2.1 Activity I**  **User Login Process.**  Let S1 be a set of User’s parameters for login.  S1= {userid, password}  Where,  userid – User id of the user  password – password of the user   |  |  | | --- | --- | | Condition/Parameter | Operation/Function | | If user == valid user | f1:Proceed() | | Else.. | Discard user | | If userid/password of the user is valid then proceed  Else discard the user |
|  | **2.2 Activity II**  **View Railway Station Layout**  Lets S2 be a set of station layout parameters:  S2={userid, destination\_station, layout\_id}  Where,  userid – userid of the user  destination\_station – destination station name  layout\_id – layout of that station   |  |  | | --- | --- | | Condition/Parameters | Operation/Function | | Layout | f1:Search(); | | If (layout is available)  Show Layout  Else  Throw error | f2:CheckLayout() ;  f3:error() | | Search the required layout in database of the destination station  If layout available then show to user, otherwise throw error  Else throw error |
|  | **2.3 Activity III**  **Locker Checking Process**  Let S3 be the set of parameters to validate ticket.  S3:{user\_id, locker\_id, station\_id}  Where,  user\_id – userid of the user  station\_id – station id of the station  locker\_id – Locker id of the locker   |  |  | | --- | --- | | Condition/Parameter | Operation/Function | | Locker\_id | F1:Validate() | | If(locker present)  Book Locker  Else  Throw Error | F2:CheckLocker()  F3: Invalidate() | | Check if the locker is available on the destination station,  If yes book the locker for the user or else throw error. |
| **3.** | **Venn Diagram**  As described above in entire Process: Railway Guide  Input(Location Info) Output(Layout, Locker, travel guide)  Y1  Y2  Y3  Y4  Y5  Y6  X1  X2  X3  X4  X5  :  : |  |

1. OCL(Object Constraint Language)
   * + 1. User Registration is compulsory for accessing application features::

Self.username -> not empty

Self.usermobileno> 10

Self.userpass> 6

Self.user\_address -> not empty

Self.user\_email\_id -> not empty

Self.user\_gender ->M/F

* + - 1. User location information is required for further processing:

Self.lattitude -> not empty

Self.longitude-> not empty

* + - 1. Destination station should be selected:
      2. Self.destination\_station\_name -> not empty
      3. Train should be selected :
      4. Self.train\_no->validnumber

**Appendix B : FEASIBILITY ANALYSES**

Rail Commute Application*:* Given a failure case viz. Q, invalid location fetch by users mobile phone, we devise an algorithm for this problem as follows:

For a Problem P1 to be NP-Hard, Satisfiability problem (SAT) must be reducible to P1;

SAT ≤ P ;

Let the propositional formula be: G = X1^X2

Where

X1: True if Latitude/Longitude are invalid

X2: True if destination station name is invalid.

**Algo sati()**

**{**

For i: 1 to 2

xi = Choice(True,False);

if G(x1,x2) then

Success();

else

failure();

}

Therefore, since the problem becomes a decision problem, it is **NP.**

**Satisfiability and Reducibility:**

3 SAT problem is NP Complete. The system can be reduced to 3SAT problem. A 3SAT problem takes a Boolean formula S that is in CNF in which each clause has exactly three literals. 3SAT is a restricted form of CNF-SAT problem.

x1--Location manager  
x2 - Station Layout View  
x3 - Locker Mgmt

S = (x1 ^ x2 ^ x3)

Algo sat()

{

For i= 1 to 3

Xi=Choice(true, false)

If(S(x1,x2,x3)= true)

Success()

Else

Failure()

}

As it is polynomial time. It is NP-Complete.

**APPENDIX D: USER MANUAL**

**Installation Steps**

We can use the application on any phone, in railway station.

The basic installation steps are:

1. Download the Indian Railway Commute application from the playstore.
2. Click on the configuration button to get the IP address of the server.
3. Register if you are a new user.
4. Login with your username and password if you are already an existing user.
5. Enjoy all the different features on the mobile application while you are travelling.

**LIST OF FIGURES**

|  |  |  |
| --- | --- | --- |
| TABLE NO. | NAME | PAGE NUMBER |
| 1 | Architecture Diagram | 2 |
| 3.1 | Entity Relationship Diagram | 20 |
| 3.2 | Block diagram | 21 |
| 3.5.1 | Use Case Diagram | 28 |
| 3.5.2 | Activity Diagram | 29 |
| 3.5.3 | Sequence Diagram | 30 |
| 3.5.4 | Deployment Diagram | 31 |
| 3.5.5 | Package Diagram | 32 |
| 3.5.6 | Component Diagram | 33 |
| 3.5.7 | Class Diagram | 33 |

**References**

[1] IEEE 2013 - Developing an Android based learning application for mobile devices

<http://ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber=6218028&url=http%3A%2F%2Fieeexplore.ieee.org%2Fxpls%2Fabs_all.jsp%3Farnumber%3D6218028>

[2] IEEE 2011 - Android: Changing the Mobile Landscape

<http://ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber=5676144&url=http%3A%2F%2Fieeexplore.ieee.org%2Fxpls%2Fabs_all.jsp%3Farnumber%3D5676144>

[3] IEEE 2012 - Voice enabled Android application for vehicular complaint system: Using GPS and GSM-SMS technology

<http://ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber=6409133&url=http%3A%2F%2Fieeexplore.ieee.org%2Fxpls%2Fabs_all.jsp%3Farnumber%3D6409133>

[4]IEEE 2013 - Hajj crowd management and navigation system: People tracking and location based services via integrated mobile and RFID systems

<http://ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber=6522008&url=http%3A%2F%2Fieeexplore.ieee.org%2Fxpls%2Fabs_all.jsp%3Farnumber%3D6522008>

[5] IEEE 2012 - Web based remote exploration and control system using android mobile phone

<http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=6317481>

[6]Raman, K.S. and Wig, S., ASCI Journal of Management, 2010, vol 39, page 33-54.

[7] Agarwal S., Singhal R., Mittal R., Freight Operations Information System (FOIS) – advanced architecture & its Future. IJLRST 2013, Vol 2 Issue 2.

[8] Agarwal, V.K., Indian Railways in Full steam, www.pib.nic.in

[9]  IEEE 2012 - Android city tour guide system based on Web service

<http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=6201621>

[10] http://borjournals.com/Research\_papers/Jun\_2013/1356IT.pdf