TRANSPORT MANAGEMENT SYSTEM

A PROJECT REPORT

Submitted By

AKSHAY.PK	**310813205010**
AJAY.M	**310813205005**
AKASH.M	**310813205007**

In the partial fulfilment for the award of the degree
Of

BACHELOR OF TECHNOLOGY

In

INFORMATION TECHNOLOGY

JEPPIAAR ENGINEERING COLLEGE

ANNA UNIVERSITY

CHENNAI - 600 025

APRIL 2017

JEPPIAAR ENGINEERING COLLEGE

DEPARTMENT OF INFORMATION TECHNOLOGY

JEPPIAAR NAGAR, RAJIV GANDHI ROAD, CHENNAI-119

This is to certify that this Project Report "**TRANSPORT MANAGEMENT SYSTEM"** is the bonafide work of **"AKSHAY.PK ,AJAY.M** and **AKASH. M"** who carried out the project under my supervision.

SUPERVISOR	**HEAD OF THE DEPARTMENT**
Mr.GLADWIN, M.E.,	**Dr.R.Sabitha, M.E., Ph.D.,**
Assistant Professor,	Professor and Head,
Department of Information Technology	Department of Information Technology
Jeppiaar Engineering College	Jeppiaar Engineering College
Chennai	Chennai

Submitted for the examination held on ……………………

INTERNAL EXAMINER **EXTERNAL EXAMINER**

ACKNOWLEDGEMENT

We are very much indebted to **(Late) Hon'ble Colonel DR. JEPPIAAR, M.A., B.L., Ph.D.**, Our Chairman **Honarable MRS. REMIBAI JEPPIAAR,** Our Managing Director **Dr. M. REGEENA JEPPIAAR, B. Tech., M.B.A., Ph.D.**, and Our Principal **Dr. G.VENUGOPALAKRISHNAN, M.Tech. F.I.E, Ph.D.**

We would like to express our deep SENSE of gratitude to **DR.R.SABITHA, ME,Ph.D, Head of the Department** and also to our guide **MR.A.GLADWIN, M.E.,** for giving valuable suggestions for making this project a grand success.

We also thank the teaching and non teaching staff members of the department of Information Technology for their constant support.

TABLE OF CONTENT

ABSTRACT

The bus routes are totally around 42 in number and is constantly changing every day. Hence we tend to deploy this application over a server based mechanism where the routes are entered onto the server any time of the day and once the student logs in to the application can see all the new refreshed routes for that day. Therefore the requirements are simple. A front end described in JSP and HTML code which gives the ADMIN access to changing the routes which is constantly in sync with the ANDROID Application so that once the routes are changed by the ADMIN it gets reflected in the mobile application. The routes are described along with GOOGLE MAPS so that the student gets real time look at where the bus would be going for that day. Along with the routes we tend to describe the exact roads on which the bus would be travelling on. The start and end points can also change accordingly to the college timings. If this is deployed then the students will have no problem in going on time to their buses as well as the buses leaving the campus on time. Every student must have this app on his phone or this app can be deployed on many IPads or TV's placed on campus for the student's convenience. This app can also be displayed on TV's placed on every floor fifteen minutes before the students leave to their respective buses. From this project, we wish to build a student friendly application that helps them in many ways other from what is described in this abstract.

LIST OF FIGURES

SCREENSHOTS

CHAPTER 1
INTRODUCTION

In colleges, students and staff face a problem in finding their daily route that is displayed in the notice board . They get trapped in the stampede in the course of finding their daily route. So to overcome this we have developed a web application to maintain a good transportation system.

1.1 OBJECTIVE

Objective of the project is to develop a new web application for maintaining a good transportation the college campus. Thus, making the process of finding the daily routes much more easier to all the students and staffs in the college campus the daily routes can be accessed by all students and staffs this may reduce the amount of effort put by them to know there daily routes in a regular basis. The proposed project is a web application to maintain a good transportation in the campus. The main objective is to make the daily routes easily accessible to all students and staffs in the campus.

1.2 MOTIVATION

As per getting the feed backs that are obtained from the students ,the current scenario of the college transportation is evaluated and we came to the idea of developing a web application to manage a good transportation in our college. And to make the process of getting the information about the daily available routes in the college can reach the students and faculties in a easy way.

We are developing this application to give the students and faculties to experience a new way of representation of daily routes in the college also the main purpose is to minimize the consumption of papers on a daily basis. The representation of routes in web application consume less amount of time.

Our motive is to make the entire transport management process much easier and

CHAPTER 2
SYSTEM ANALYSIS

2.1 SURVEY ON EXISTING SYSTEM

The study on the existing system is made by collecting the feed backs from students on what are the difficulties they face in knowing their respective routes.In the current system students are finding it very difficult in knowing the routes.They tend to suffer a lot by getting trapped in the stampede. In the existing system the amount of time consumed in knowing the daily routes is more than in the proposed system.

The existing system is more tedious for the students to know there routes in this scenario we tend to waste more number of papers in displaying the daily routes to minimize the paper consumption we tend to display the routes via web pages this consume less time as all the details are entered in the database and we have a common admin to maintain the routes in the system. Any modification in the daily routes can be made earlier and not at the neck of the moment.

In the existing system the students and staffs are on a hurry to know their respective routes since the route details are displayed in the neck of the moment by the management this creates a lot of confusion among the students in knowing their respective routes.This is a very ,tedious process for students and faculties as they get trapped in the stampede as they are searching for their routes at the last minute.

To overcome this issue we are developing a web application to display the daily routes well in advance and also to prevent the students and faculties from getting trapped in the stampede. The routes that are displayed in the web page can also be shown in mobile phones by using the responsive design .So that students can get to

2.1.1 DRAWBACK

They tend to consume more time for the students to know their routes. In this system the consumption of papers is also high this process may consume a lot of effort from students and faculties in knowing their respective routes in daily basis

2.2 PROPOSED SYSTEM

In the proposed system we tend to overcome all the drawbacks in the existing system. In this project we are displaying the daily routes in the web application this application can be accessed in android phones since we are using thee responsive design to bring the web application in android and other type of operating system this makes a easy access to the route details for all kind of users in the system. The routes can be updated by the admin much earlier than in the existing system.

2.2.1 ADVANTAGE

The advantage of using this system is that they consume less amount of time in displaying the route and also reduces the effort of students and faculties in knowing there respective routes that are available in the college. The proposed system is much more easier use than that of the existing system.

2.3 FEASIBILITY REPORT

The feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential.
Three key considerations involved in the feasibility analysis are

- **ECONOMICAL FEASIBILITY**
- **TECHNICAL FEASIBILITY**
- **SOCIAL FEASIBILITY**

2.3.1 ECONOMICAL FEASIBILITY

This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

2.3.2 TECHNICAL FEASIBILITY

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

2.3.3 SOCIAL FEASIBILITY

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system. The original and reference implementation Java compilers, virtual machines, and class libraries were developed by Sun from 1995. As of May 2007, in compliance with the specifications of the Java Community Process, Sun relicensed most of their Java technologies under the GNU General Public License. Others have also developed alternative implementations of these Sun technologies, such as the GNU Compiler for Java and GNU Class path.

CHAPTER 3
APPLICATION REQUIREMENTS

This application runs on also android handheld device starting with the minimum OS platform of 4.2. The requirements in depth are discussed below.

3.1 HARDWARE REQUIREMENTS

The android running handheld devices furnishes the basic needs of the application.

HARDWARE REQUIREMENT

Minimum System Requirements to run the software :

OS : Windows 7 or higher versions.

RAM : 2GB

Browser : Chrome v38 or higher.

HDD : 50Gb

Maximum System Requirements to run the software :

OS : Windows 7 or higher versions.

RAM : 4GB

Browser : Chrome v38 or higher.

HDD : 50Gb

3.2 SOFTWARE REQUIREMENTS

The basic operating platform of the application is android that runs on the above mentioned hardware additionally Java Run time Environment (JRE) will be required to support various classes in android.

CHAPTER 4
SOFTWARE DESCRIPTION

4.1 FRONT END

JAVA

Java is a computer programming language that is concurrent, class-based, object-oriented, and specifically designed to have as few implementation dependencies as possible. It is intended to let application developers "write once, run anywhere" (WORA), meaning that code that runs on one platform does not need to be recompiled to run on another. Java applications are typically compiled to byte code (class file) that can run on any Java virtual machine (JVM) regardless of computer architecture. Java is, as of 2014, one of the most popular programming languages in use, particularly for client-server web applications, with a reported 9 million developers. Java was originally developed by James Gosling at Sun Microsystems (which has since merged into Oracle Corporation) and released in 1995 as a core component of Sun Microsystems' Java platform. The language derives much of its syntax from C and C++, but it has fewer low-level facilities than either of them. The original and reference implementation Java compilers, virtual machines, and class libraries were developed by Sun from 1991 and first released in 1995. As of May 2007, in compliance with the specifications of the Java Community Process, Sun relicensed most of its Java technologies under the GNU General Public License. Others have also developed alternative implementations of these Sun technologies, such as the GNU Compiler for Java (byte code compiler), GNU Class path (standard libraries), and Iced Tea-Web (browser plug-in for applets).

MYECLIPSE

My Eclipse is a java EE and Ajax IDE created and maintained any the company Genuine , a founding member of the Eclipse Foundation. My Eclipse is built upon the Eclipse platform, and integrates both proprietary and open source solutions into the development environment. My Eclipse has two primary versions a professional and a standard edition. The standard edition adds database tools, a visual web designer, persistence tools, spring tools, Struts and JSF tooling, and a number of other features to the basic Eclipse Java Developer profile. It competes with the Web Tools Project, which is a part of Eclipse itself, but My Eclipse is a separate project entirely and offers a different feature set.

My Eclipse has also been made available via Pulse (ALM), a provisioning tool that maintains Eclipse software profiles, including those that use My Eclipse. Additionally, My Eclipse is offering a customized version for IBM products, "My Eclipse Blue Edition", that adds specific support for Rational Software and Web Sphere development. Currently, My Eclipse Blue Edition is available for Windows and Linux, though Mac is unsupported.

SPRING

The Spring Framework is an application framework and inversion of control container for the Java platform. The framework's core features can be used by any Java application, but there are extensions for building web applications on top of the Java EE platform. Although the framework does not impose any specific programming model, it has become popular in the Java community as an alternative to, replacement for, or even addition to the Enterprise JavaBeans (EJB) model. The Spring Framework is open source. The Spring Framework includes several modules that provide a range of services:

Spring Core Container: this is the base module of Spring and provides spring containers (Bean Factory and Application Context).

Aspect-oriented programming: enables implementing cross-cutting concerns. Authentication and authorization: configurable security processes that support a range of standards, protocols, tools and practices via the Spring Security sub-project (formerly Security System for Spring).

Convention over configuration: a rapid application development solution for Spring-based enterprise applications is offered in the Spring module

Data access: working with relational database management systems on the Java platform using JDBC and object-relational mapping tools and with No SQL databases

Inversion of control container: configuration of application components and lifecycle management of Java objects, done mainly via dependency injection

Messaging: configurative registration of message listener objects for transparent message-consumption from message queues via JMS, improvement of message sending over standard JMS APIs

Model–view–controller: an HTTP- and servlet-based framework providing hooks for extension and customization for web applications and RESTful Web services.

Remote access framework: configurative RPC-style marshalling of Java objects over networks supporting RMI, CORBA and HTTP-based protocols including Web services (SOAP)

Transaction management: unifies several transaction management APIs and coordinates transactions for Java objects

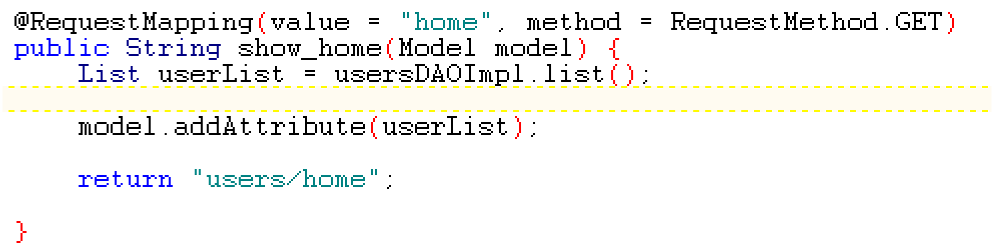
Remote management: configurative exposure and management of Java objects for local or remote configuration via JMX

Testing: support classes for writing unit tests and integration tests

4.2 BACKEND

SPRING CONTROLLER:

In Spring framework, all the requests sent by the dispatcher servlet normally directed to a controller class. This controller class is mapping those requests to each process & execute the requested inputs. In a project there can be multiple controllers defined for different purposes. All these controllers refers to the same dispatcher servlet. As I mentioned in the previous post, @Request Mapping keyword is used to map the dispatcher servlet with the controller class. In a Controller mapping there are two types of mapping as GET & POST. Normally there can be many GET methods in a controller while one POST method is employed. GET request method is used to get the requests from the user do the desired work & output results into a view(jsp pages). A GET request is shown below.



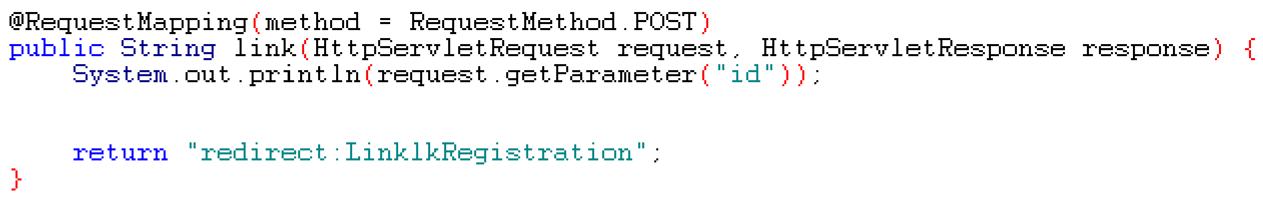
```
@RequestMapping(value = "home", method = RequestMethod.GET)
public String show_home(Model model) {
    List userList = usersDAOImpl.list();

    model.addAttribute(userList);

    return "users/home";

}
```

A POST request is used to post the results in to a page as usual. A POST request is shown below.



```
@RequestMapping(method = RequestMethod.POST)
public String link(HttpServletRequest request, HttpServletResponse response) {
    System.out.println(request.getParameter("id"));


    return "redirect:LinklkRegistration";
}
```

SPRING MVC

Spring MVC helps in building flexible and loosely coupled web applications. The Model-view-controller design pattern helps in separating the business logic, presentation logic and navigation logic. Models are responsible for encapsulating the application data. The Views render response to the user with the help of the model object . Controllers are responsible for receiving the request from the user and calling the back-end services.

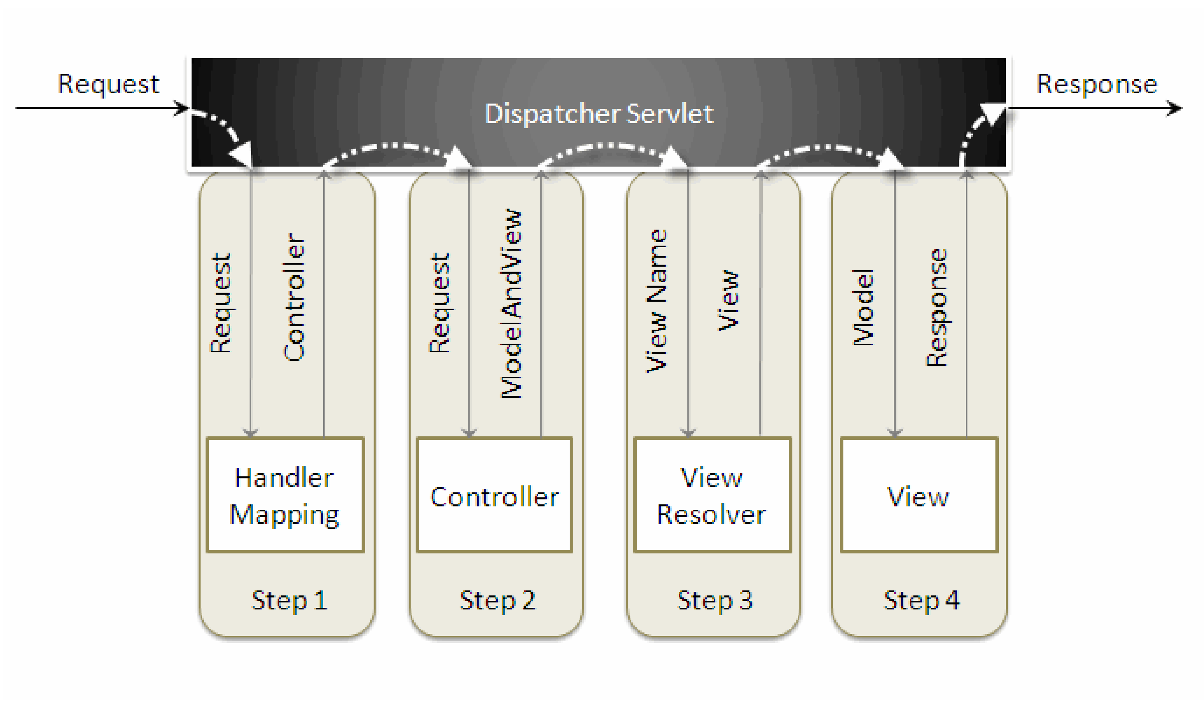


Fig 4.1 string mvc framework

When a request is sent to the Spring MVC Framework the following sequence of events happen. The Dispatcher Servlet first receives the request. The Dispatcher Servlet consults the Handler Mapping and invokes the Controller associated with the request.

The Controller process the request by calling the appropriate service methods and returns a Mode And View object to the Dispatcher Servlet. The Mode And View object contains the model data and the view name. The Dispatcher Servlet sends the view name to a View Resolver to find the actual View to invoke. Now the Dispatcher Servlet will pass the model object to the View to render the result. The View with the help of the model data will render the result back to the user.

4.3 FEATURES

4.3.1SIMPLE

The software is easy to write and understand. The software programmer doesn't have to work with pointers and doesn’t need to manage memory explicitly. The software eliminated the use of pointers for security reasons. Its virtual machine is able to handle the memory management and so removes the occupied memory automatically when it is no longer referenced. Programmer can now concentrate on the required application logic rather than wasting time and logic with these managements.

4.3.2 THE SOFTWARE IS OBJECT-ORIENTED

There are several languages like C and C++ which are not pure object oriented but The software is fully object oriented language which follows all the rules of OOPs like Inheritance, Encapsulation, Polymorphism etc and everything can be treated in the form of objects. Even primitive data types can also be treated as objects using wrapper classes to make it fully object oriented. Class is the basic unit in the software and objects are entities following the prototype defined by the class.

4.3.3 THE SOFTWARE IS ROBUST AND SECURE

The software applications are reliable in various ways. It offers compile time checking to detect early the causes of bugs, run time checking, eliminates the use of pointers which can cause memory corruption or unwanted access of memory, garbage collection management to free the unused memories automatically, and exception handling to handle the situation at the time of occurrence of any error and a lot more.

CHAPTER 5
PROJECT DESIGN

5.1 ARCHITECTURE DESIGN

Architecture diagram explains briefly about the working of our proposed system. The system works under two cases and it is shown below.

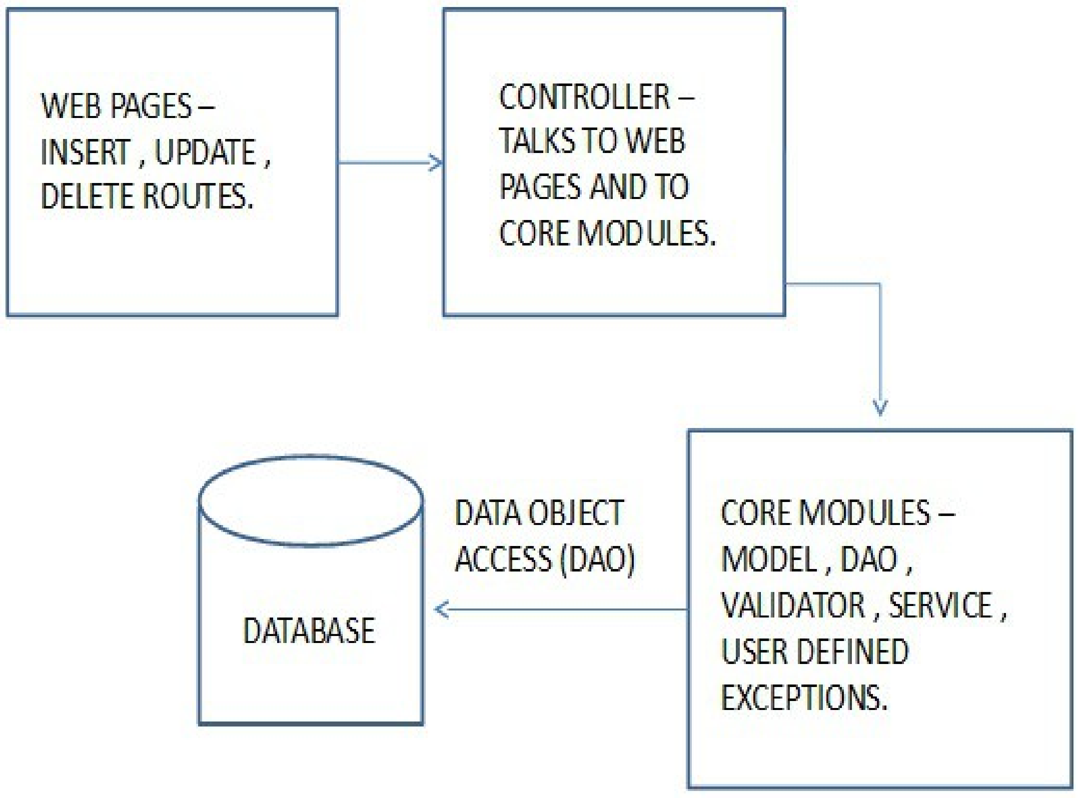

WEB PAGES – INSERT , UPDATE , DELETE ROUTES.
CONTROLLER – TALKS TO WEB PAGES AND TO CORE MODULES.
CORE MODULES – MODEL , DAO , VALIDATOR , SERVICE , USER DEFINED EXCEPTIONS.
DATA OBJECT ACCESS (DAO)
DATABASE

Fig 5.1 architecture design

The architecture design of the transport management system is done to give a frame work of what are the internal process that are done in the system to develop

web page in a system in the web page we will insert update and delete the daily routes in the system. The insert update and delete process is done only by the admin of the transport department.

INSERT

In this module the bus routes are inserted in the web page as the available route details are obtained from the transport in charge of the college and the details are entered in the webpage and the arrival time of the bus to the college campus is also entered in the system and the departure time is also given for the students to verify the daily available routes the students and staffs can only verify the daily routes and they cannot change any routes as a separate login and password is given to the admin in the system.

UPDATE

The daily routes are updated in the system by the admin on the basis of the available count of the students on exam time and also in special class because some exams are scheduled in afternoon for students and some students are made to stay back to attend the special classes in the college this would be an issue for admin to combine or update the routes in the basis of count of the students in the college campus. In regular days all routes are allowed to drop the students in there respective boarding points in the system.

DELETE

The route details are monitored in the system by the system admin the unused routes are removed from the database to make the task much more easier for the admin in the system.

CONTROLLER

The controller takes the admin to the webpage and administer the routes available in the system the controller also administers the core modules used in the system the core modules are managed by the controller in the system they can be managed easily by the admin in the system the evaluation process in the system .

DATA OBJECT ACCESS

The data object access is one of the core models used in the system to manage a good transport management system in the college the data is accessed by the admin from the controller to modify the data in the system.

VALIDATOR

The validator evaluates each and every modules and its functions in the system the validator evaluates each and every sub process in the system the modules are evaluated in a particular manner to evaluate the modules execution of small tasks

SERVICE

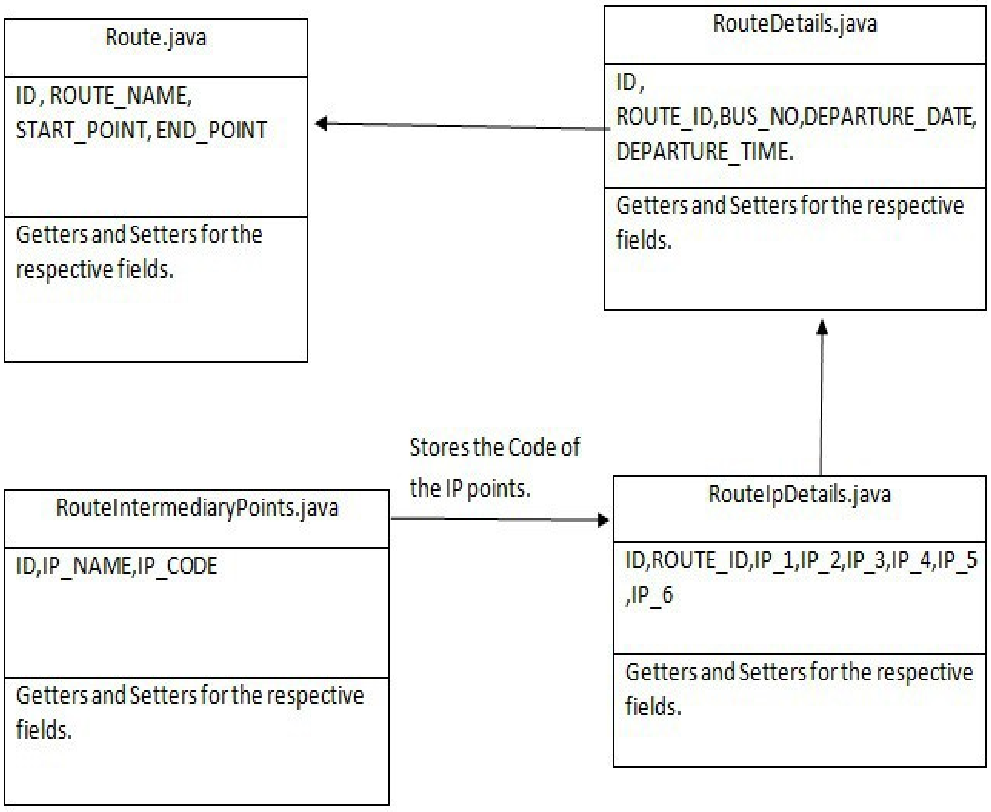
This module tells the admin each and every small tasks that are provided in the system to manage a particular webpage in the system the evaluation process is done in a short span of time in a system.

DATABASE

The database is a another type of module used to store all data in the system and those data can be accessed later when it is required to be seen in the system the database stores all kinds of data that are relevant to the project in the system.

5.2 DATA OBJECT DIAGRAM

The data flow diagram clearly explains the flow of the process and also explains how the system works



Route.java
ID, ROUTE_NAME, START_POINT, END_POINT
Getters and Setters for the respective fields.
RouteDetails.java
ID, ROUTE_ID,BUS_NO,DEPARTURE_DATE, DEPARTURE_TIME.
Getters and Setters for the respective fields.
Stores the Code of the IP points.
RouteIntermediaryPoints.java
ID,IP_NAME,IP_CODE
Getters and Setters for the respective fields.
RouteIpDetails.java
ID,ROUTE_ID,IP_1,IP_2,IP_3,IP_4,IP_5 ,IP_6
Getters and Setters for the respective fields.

Fig 5.2 data object diagram

5.3 MODULES DESCRIPTION

5.3.1 DATABASE DESIGN MODULE

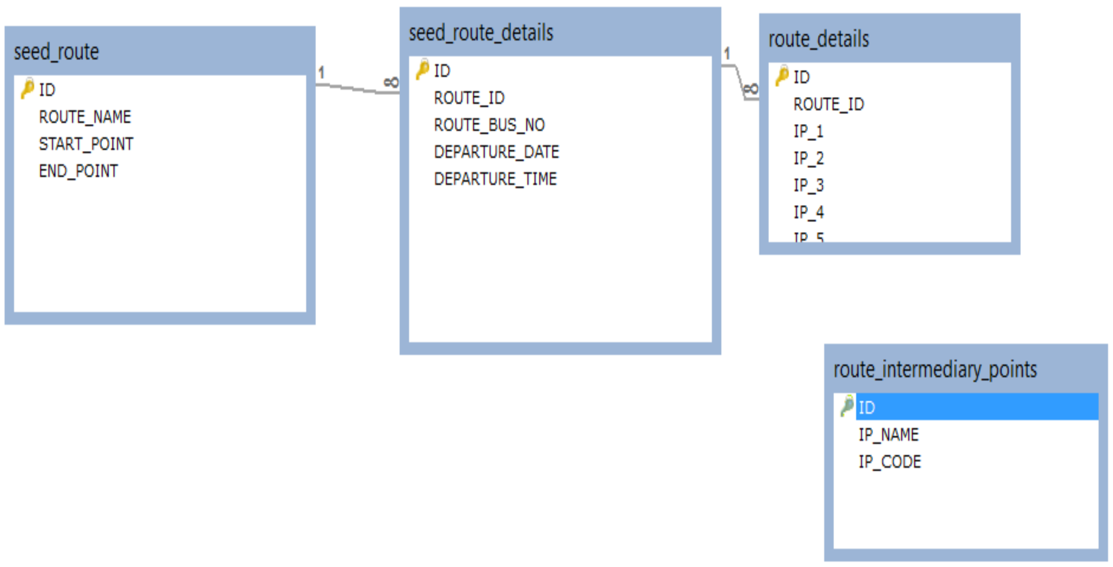


Fig 5.3 database model

5.4 CLASS DIAGRAM

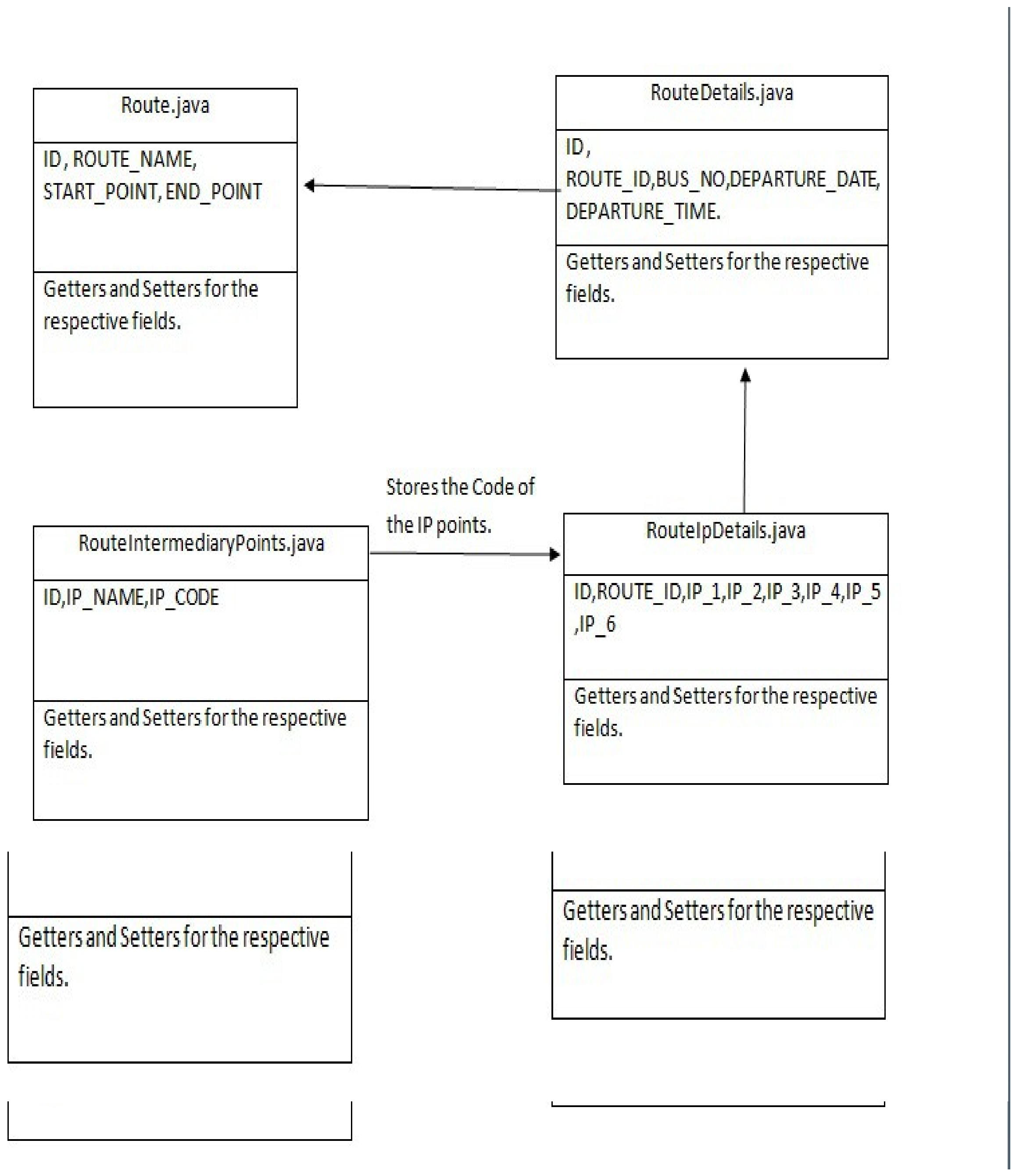

Route.java
ID, ROUTE_NAME, START_POINT, END_POINT
Getters and Setters for the respective fields.
RouteDetails.java
ID, ROUTE_ID,BUS_NO,DEPARTURE_DATE, DEPARTURE_TIME.
Getters and Setters for the respective fields.
Stores the Code of the IP points.
RouteIntermediaryPoints.java
ID,IP_NAME,IP_CODE
Getters and Setters for the respective fields.
RouteIpDetails.java
ID,ROUTE_ID,IP_1,IP_2,IP_3,IP_4,IP_5 ,IP_6
Getters and Setters for the respective fields.
Getters and Setters for the respective fields.
Getters and Setters for the respective fields.

Fig 5.4 class diagram

CHAPTER 6
SYSTEM IMPLEMENTATION

Our system is used to develop a good transport management system in our college campus. We mainly propose this system to minimize the consumption of papers. And to prevent people from getting trapped in the stampede on the course of finding there daily routes. We impliment this project on web application and can also be brought on android application using responsive design in the system.

Our application consists of an admin login with a user id and password when the admin enters the user name and password it automatically takes the admin to the server where he has to enter the route on the daily basis and on exam time he has to allocate buses for two schedule one for afternoon and another one for evening in the system the buses are allocated based on the available count of the students on that particular schedule in the system.

The students and staffs are only allowed to access the daily route and they are not allowed to modify the existing routes in the system. The admin is the main authority to modify the routes and not the students or staffs in the system. The routes are displayed in the LED display to provide a new manner of providing those routes to the students and faculties in the campus and also to make the daily routes easily accessible for the faculties and students.

In the system implementation we display the routes in map using markers by specifying the starting point and destination point using the markers with different colours and also tells some of the intermediate points by using different colour in the system. We are using the search option for the students and staffs to easily access their location in the system. In the web page when we specify the particular route all the intermediate routes are displayed in the system.

CHAPTER 7
TESTING

7.1 UNIT TESTING

Unit testing is a method by which individual units of source code, sets of one or more computer program modules together with associated control data, usage procedures, and operating procedures are tested to determine if they are fit for use.] Intuitively, one can view a unit as the smallest testable part of an application. In procedural programming, a unit could be an entire module, but it is more commonly an individual function or procedure.

In object-oriented programming, a unit is often an entire interface, such as a class, but could be an individual method.] Unit tests are short code fragments created by programmers or occasionally by white box testers during the development process. Ideally, each test case is independent from the others. Substitutes such as method stubs, mock objects, fakes, and test harnesses can be used to assist testing a module in isolation. Unit tests are typically written and run by software developers to ensure that code meets its design and behaves as intended.

Unit testing is the testing of an individual unit or group of related units. It falls under the class of white box testing. It is often done by the programmer to test that the unit he/she has implemented is producing expected output against given input.

7.2 TEST CASE

ONE TIME REGISTRATION

The user should get registered once when installing the application.

All field should be filled to register in server. If any field is left unfilled, “FILL ALL THE FIELD" message will be displayed.

CHAPER 8
APPENDICES

APPENDIX 1
SOURCE CODE

ConnectionUtil.java :

```
package com.jeppiaar.util;
import javax.sql.DataSource;
import org.apache.commons.dbcp2.BasicDataSource;
import org.springframework.jdbc.core.JdbcTemplate;

public class ConnectionUtil {
        public static DataSource getDataSource() {
                final BasicDataSource ds = new BasicDataSource();
                ds.setDriverClassName("com.mysql.cj.jdbc.Driver");
                ds.setUsername("root");
                ds.setPassword("machadon");
                ds.setUrl("jdbc:mysql://localhost:3306/jeppiaartransport");
                return ds;
        }

        public static JdbcTemplate getJdbcTemplate()

        {
                final JdbcTemplate jt = new JdbcTemplate();
                jt.setDataSource(getDataSource());
                return jt;
        }
}
```

Model Code :

```
package com.jeppiaar.model;

import lombok.Data;
@Data
public class Route {
private int Id;
private String routeName;
private String startPoint;
```

```
import com.jeppiaar.model.AdminLogin;
import com.jeppiaar.model.Route;
import com.jeppiaar.util.ConnectionUtil;
public class AdminLoginDAO {
        JdbcTemplate jdbcTemplate=ConnectionUtil.getJdbcTemplate();

    public void save(AdminLogin adminLogin)
    {
        String sql="insert into admin_login(username,password,emailid,phone)
values(?,?,?,?)";
        Object[]
params={adminLogin.getUserName(),adminLogin.getPassword(),adminLogin.
getEmailId(),adminLogin.getPhoneNumber()};
        int rows=jdbcTemplate.update(sql,params);
        System.out.println(rows);
    }                {
        String sql="update admin_login set password=?,emailid=? where
username=?";
        Object[]
params={adminLogin.getPassword(),adminLogin.getEmailId(),adminLogin.get
UserName()};
        int rows=jdbcTemplate.update(sql,params);
        System.out.println(rows);
    }
    public void delete(int id)
    {
        String sql="delete from admin_login where ID=?";

        int rows=jdbcTemplate.update(sql,id);
        System.out.println(rows);
    }
        public List<AdminLogin> list() {
                final String sql = "select
USERNAME,PASSWORD,EMAILID,PHONE from admin_login";
                return jdbcTemplate.query(sql, (rs, rowNum) -> {
                        AdminLogin adminLogin=convert(rs);
                        return adminLogin;
                });

        }
```

```
/**
 * Converts Object to ResultSet
 * @param rs
 * @return
 * @throws SQLException
 */
	static AdminLogin convert(final ResultSet rs) throws SQLException {
		AdminLogin adminLogin = new AdminLogin();
		adminLogin.setUserName(rs.getString("USERNAME"));
		adminLogin.setPassword(rs.getString("PASSWORD"));
		adminLogin.setEmailId(rs.getString("EMAILID"));
		adminLogin.setPhoneNumber(rs.getLong("PHONE"));
		return adminLogin;
	}
	public boolean functionValidLogin(String username,String password) {
		String sql = "select fn_is_valid_login(?,?)";
		return	jdbcTemplate.queryForObject(sql,	new	Object[]
{username,password}, Boolean.class);
	}
}


package com.jeppiaar.dao;

import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.List;

import org.springframework.jdbc.core.JdbcTemplate;

import com.jeppiaar.model.Route;
import com.jeppiaar.model.RouteDetails;
import com.jeppiaar.model.RouteIpDetails;
import com.jeppiaar.util.ConnectionUtil;

public class RouteDAO {

JdbcTemplate jdbcTemplate=ConnectionUtil.getJdbcTemplate();

    public void save(Route route)
```

```
        String                         sql="insert                              into
seed_route(ROUTE_NAME,START_POINT,END_POINT) values(?,?,?,?)";
        Object[]
params={route.getRouteName(),route.getStartPoint(),route.getEndPoint()};
        int rows=jdbcTemplate.update(sql,params);
        System.out.println(rows);
    }
    public void update(Route route)
    {
        String sql="update seed_route set ROUTE_NAME=? where ID=?";
        Object[] params={route.getRouteName(),route.getId()};
        int rows=jdbcTemplate.update(sql,params);
        System.out.println(rows);
    }

    {
        String sql="delete from seed_route where ID=?";

        int rows=jdbcTemplate.update(sql,id);
        System.out.println(rows);
    }
        public List<Route> list() {
                final            String            sql            =            "select
ID,ROUTE_NAME,START_POINT,END_POINT from seed_route";
                return jdbcTemplate.query(sql, (rs, rowNum) -> {
                        Route route=convert(rs);
                        return route;
                });

        }
 /**
  * Converts Object to ResultSet
  * @param rs
  * @return
  * @throws SQLException
  */
        static Route convert(final ResultSet rs) throws SQLException {
                Route route=new Route();
                route.setId(rs.getInt("ID"));
                route.setRouteName(rs.getString("ROUTE_NAME"));
                route.setStartPoint(rs.getString("START_POINT"));
```

```
return route;
	}
}
package com.jeppiaar.dao;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Types;
import java.time.LocalDate;
import java.time.LocalTime;
import java.util.List;
import java.util.Map;
import org.springframework.jdbc.core.SqlOutParameter;
import org.springframework.jdbc.core.SqlParameter;
import org.springframework.jdbc.core.namedparam.MapSqlParameterSource;
import org.springframework.jdbc.core.namedparam.SqlParameterSource;
import org.springframework.jdbc.core.simple.SimpleJdbcCall;

import com.jeppiaar.model.Route;
import com.jeppiaar.model.RouteDetails;
import com.jeppiaar.model.RouteIpDetails;
import com.jeppiaar.util.ConnectionUtil;

public class RouteDetailsDAO {

	JdbcTemplate jdbcTemplate=ConnectionUtil.getJdbcTemplate();

	public void save(RouteDetails routeDetails)
	{
		String                    sql="insert                    into
seed_route_details(ROUTE_ID,ROUTE_BUS_NO,DEPARTURE_DATE,DEP
ARTURE_TIME) values(?,?,?,?,?)";
		Object[]
params={routeDetails.getRouteId().getId(),routeDetails.getBusNo(),routeDetail
s.getDepartureDate(),routeDetails.getDepartureTime()};
		int rows=jdbcTemplate.update(sql,params);
		System.out.println(rows);
	}
	{
		String sql="update seed_route_details set ROUTE_BUS_NO=?
where ROUTE_ID=?";
		Object[]
params={routeDetails.getBusNo(),routeDetails.getRouteId().getId()};
		int rows=jdbcTemplate.update(sql,params);
		System.out.println(rows);
	}
	public void delete(int id)
	{
		String sql="delete from seed_route_details where ID=?";
		int rows=jdbcTemplate.update(sql,id);
		System.out.println(rows);
	}

	public List<RouteDetails> list() {
		final String sql = "select
ID,ROUTE_ID,ROUTE_BUS_NO,DEPARTURE_DATE,DEPARTURE_TIM
E from seed_route_details";
		return jdbcTemplate.query(sql, (rs, rowNum) -> {
			RouteDetails routeDetails=convert(rs);
			return routeDetails;
		});

	}
 /**
  * Converts Object to ResultSet
  * @param rs
  * @return
  * @throws SQLException
  */
	static RouteDetails convert(final ResultSet rs) throws SQLException {
		RouteDetails routeDetails=new RouteDetails();
		routeDetails.setId(rs.getInt("ID"));
		Route route=new Route();
		route.setId(rs.getInt("ROUTE_ID"));
		routeDetails.setRouteId(route);
		routeDetails.setBusNo(rs.getInt("ROUTE_BUS_NO"));

routeDetails.setDepartureDate(rs.getDate("DEPARTURE_DATE").toLocalDate
());
routeDetails.setDepartureTime(rs.getTime("DEPARTURE_TIME").toLocalTim
e());
            return routeDetails;
        }

        public String addRoute(String routeName, String endPoint, int busNo,
LocalDate depDate, LocalTime depTime , String ip1 , String ip2 ,String
ip3,String ip4,String ip5,String ip6,String errmsg) {
new SqlParameter("i_routename", Types.VARCHAR), new
SqlParameter("i_endpoint", Types.VARCHAR),
                        new SqlParameter("i_busno", Types.INTEGER), new
SqlParameter("i_depdate", Types.DATE), new SqlParameter("i_deptime",
Types.TIME), new SqlParameter("i_ip1", Types.VARCHAR), new
SqlParameter("i_ip2", Types.VARCHAR),
                        new SqlParameter("i_ip3", Types.VARCHAR), new
SqlParameter("i_ip4", Types.VARCHAR),new SqlParameter("i_ip5",
Types.VARCHAR),new SqlParameter("i_ip6", Types.VARCHAR),
                        new SqlOutParameter("errmsg", Types.VARCHAR));
            call.setAccessCallParameterMetaData(false);
            SqlParameterSource in = new
MapSqlParameterSource().addValue("i_routename",
routeName).addValue("i_endpoint", endPoint)
                        .addValue("i_busno",busNo ).addValue("i_depdate",
depDate).addValue("i_deptime", depTime).addValue("i_ip1",
ip1).addValue("i_ip2", ip2).addValue("i_ip3", ip3).addValue("i_ip4",
ip4).addValue("i_ip5", ip5).addValue("i_ip6", ip6).addValue("errmsg",errmsg );
            Map<String, Object> execute = call.execute(in);
            return (String) execute.get(errmsg);
}
        public String deleteRoute(Integer busno,String errmsg)
        {
            SimpleJdbcCall call = new
SimpleJdbcCall(jdbcTemplate).withProcedureName("PR_DELETE_ROUTE").
declareParameters(
                        new SqlParameter("i_busno", Types.INTEGER),new
SqlParameter("errmsg",Types.VARCHAR));
            call.setAccessCallParameterMetaData(false);
            SqlParameterSource in = new
MapSqlParameterSource().addValue("i_busno",
busno).addValue("errmsg",errmsg);
```

```
            return (String) execute.get(errmsg);

        }
        public String updateRouteName(Integer busno,String routeName,String
errmsg)
        {
SimpleJdbcCall                    call                    =                    new
SimpleJdbcCall(jdbcTemplate).withProcedureName("PR_UPDATE_ROUTE_
NAME").declareParameters(
                        new  SqlParameter("i_busno",  Types.INTEGER),new
SqlParameter("i_routename",Types.VARCHAR),new  SqlParameter("errmsg",
Types.VARCHAR));
            call.setAccessCallParameterMetaData(false);
            SqlParameterSource             in             =             new
MapSqlParameterSource().addValue("i_busno",
busno).addValue("i_routename",routeName).addValue("errmsg",errmsg);
            Map<String,Object> execute=call.execute(in);
            return (String) execute.get(errmsg);

        }
        public String updateRouteBusNo(String routeName,Integer busNo,String
errmsg)
        {
            SimpleJdbcCall             call             =             new
SimpleJdbcCall(jdbcTemplate).withProcedureName("PR_UPDATE_BUS_NO"
).declareParameters(
                        new                SqlParameter("i_routename",
Types.VARCHAR),new      SqlParameter("i_busno",Types.INTEGER),new
SqlOutParameter("errmsg", Types.VARCHAR));
            call.setAccessCallParameterMetaData(false);
            SqlParameterSource             in             =             new
MapSqlParameterSource().addValue("i_routename",
routeName).addValue("i_busno",busNo).addValue("errmsg",errmsg);
            Map<String,Object> execute=call.execute(in);
            return (String) execute.get(errmsg);

        }
        public String updateRouteIntermediary(Integer busNo,String ip1,String
ip2,String ip3,String ip4,String ip5,String ip6,String errmsg)
```

```
		SimpleJdbcCall call = new SimpleJdbcCall(jdbcTemplate).withProcedureName("PR_UPDATE_INTERMEDIARY").declareParameters(
				new SqlParameter("i_busno",Types.INTEGER),new SqlParameter("i_ip1",Types.VARCHAR),new SqlParameter("i_ip2",Types.VARCHAR)
```
27
```
				,new SqlParameter("i_ip3",Types.VARCHAR),new SqlParameter("i_ip4",Types.VARCHAR),new SqlParameter("i_ip5",Types.VARCHAR),new SqlParameter("i_ip6",Types.VARCHAR),new SqlOutParameter("errmsg", Types.VARCHAR));
		call.setAccessCallParameterMetaData(false);
		SqlParameterSource in = new MapSqlParameterSource().addValue("i_busno",busNo).addValue("i_ip1", ip1).addValue("i_ip2", ip2)
					.addValue("i_ip3", ip3).addValue("i_ip4", ip4).addValue("i_ip5", ip5).addValue("i_ip6", ip6).addValue("errmsg",errmsg);
		Map<String,Object> execute=call.execute(in);
		return (String) execute.get(errmsg);

	}
}


package com.jeppiaar.dao;

import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.List;
import org.springframework.jdbc.core.JdbcTemplate;
import com.jeppiaar.model.RouteIntermediaryPoints;
import com.jeppiaar.util.ConnectionUtil;

public class RouteIntermediaryPointsDAO {


	JdbcTemplate jdbcTemplate=ConnectionUtil.getJdbcTemplate();

	public void save(RouteIntermediaryPoints routeIp)
```

```
		String			sql="insert			into
route_intermediary_points(IP_NAME,IP_CODE) values(?,?)";
		Object[] params={routeIp.getIpName(),routeIp.getIpCode()};
		int rows=jdbcTemplate.update(sql,params);
		System.out.println(rows);
	}

public void update(RouteIntermediaryPoints routeIp)
	{
		String sql="update route_intermediary_points set IP_CODE=?
where IP_NAME=? ";
		Object[] params={routeIp.getIpCode(),routeIp.getIpName()};
		int rows=jdbcTemplate.update(sql,params);
		System.out.println(rows);
	}
	public void delete(int id)
	{
		String sql="delete from route_intermediary_points where ID=?";
		int rows=jdbcTemplate.update(sql,id);
		System.out.println(rows);
	}
	public List<RouteIntermediaryPoints> list() {
		final String sql = "select ID,IP_NAME,IP_CODE from
route_intermediary_points";
		return jdbcTemplate.query(sql, (rs, rowNum) -> {
			RouteIntermediaryPoints routeIp=convert(rs);
			return routeIp;
		});

	}
 /**
  * Converts Object to ResultSet
  * @param rs
  * @return
  * @throws SQLException
  */
	static RouteIntermediaryPoints convert(final ResultSet rs) throws
SQLException {
		RouteIntermediaryPoints routeIp = new RouteIntermediaryPoints();
		routeIp.setId(rs.getInt("ID"));
		routeIp.setIpName(rs.getString("IP_NAME"));
```

```
Object[] params={routeIpDetails.getIp2(),routeIpDetails.getRouteId().getId()};
		int rows=jdbcTemplate.update(sql,params);

	}
	public void updateIP3(RouteIpDetails routeIpDetails)
	{
		String sql="update route_details set IP_3=? where ROUTE_ID=?";
		Object[]
params={routeIpDetails.getIp3(),routeIpDetails.getRouteId().getId()};
		int rows=jdbcTemplate.update(sql,params);
		System.out.println(rows);
	}
	public void updateIP4(RouteIpDetails routeIpDetails)
	{
		String sql="update route_details set IP_4=? where ROUTE_ID=?";
		Object[]
params={routeIpDetails.getIp4(),routeIpDetails.getRouteId().getId()};
		int rows=jdbcTemplate.update(sql,params);
		System.out.println(rows);
	}
	public void updateIP5(RouteIpDetails routeIpDetails)
	{
		String sql="update route_details set IP_5=? where ROUTE_ID=?";
		Object[]
params={routeIpDetails.getIp5(),routeIpDetails.getRouteId().getId()};
		int rows=jdbcTemplate.update(sql,params);
		System.out.println(rows);
	}
	public void updateIP6(RouteIpDetails routeIpDetails)
	{
		String sql="update route_details set IP_6=? where ROUTE_ID=?";
		Object[]
params={routeIpDetails.getIp6(),routeIpDetails.getRouteId().getId()};
		int rows=jdbcTemplate.update(sql,params);
		System.out.println(rows);
	}
	public void delete(int id)
	{
		String sql="delete from route_details where ID=?";
		int rows=jdbcTemplate.update(sql,id);
```

```
            final              String              sql              =              "select
ID,ROUTE_ID,IP_1,IP_2,IP_3,IP_4,IP_5,IP_6 from route_details";
            return jdbcTemplate.query(sql, (rs, rowNum) -> {
                  RouteIpDetails routeIpDetails=convert(rs);
                  return routeIpDetails;
            });

      }
/**
 * Converts Object to ResultSet
 * @param rs
 * @return
 * @throws SQLException
 */
       static RouteIpDetails convert(final ResultSet rs) throws SQLException {
      RouteIpDetails routeIpDetails=new RouteIpDetails();
      routeIpDetails.setId(rs.getInt("ID"));
      RouteDetails routeDetails=new RouteDetails();
      routeDetails.setId(rs.getInt("ROUTE_ID"));
      routeIpDetails.setRouteId(routeDetails);
      routeIpDetails.setRouteId(routeDetails);
      routeIpDetails.setIp1(rs.getString("IP_1"));
      routeIpDetails.setIp2(rs.getString("IP_2"));
      routeIpDetails.setIp3(rs.getString("IP_3"));
      routeIpDetails.setIp4(rs.getString("IP_4"));
      routeIpDetails.setIp5(rs.getString("IP_5"));
      routeIpDetails.setIp6(rs.getString("IP_6"));
            return routeIpDetails;
       }

       public List<RouteIpDetails> listRoutes()
       {
            final                    String                    sql="SELECT
seed_route.ROUTE_NAME,`seed_route_details`.`ROUTE_BUS_NO`,`seed_ro
ute_details`.`DEPARTURE_DATE`,`seed_route_details`.`DEPARTURE_TIME
`,"+
"`route_details`.`IP_1`,`route_details`.`IP_2`,`route_details`.`IP_3`,`route_detai
ls`.`IP_4`,`route_details`.`IP_5`,`route_details`.`IP_6` FROM"+
"`seed_route`          JOIN          `seed_route_details`          ON
`seed_route`.`ID`=`seed_route_details`.`ROUTE_ID` JOIN `route_details` ON
`seed_route_details`.`ID`=`route_details`.`ID`";
```

```
                    RouteIpDetails routeIpDetails=convertRoutes(rs);
                    return routeIpDetails;
            });
    }
    static RouteIpDetails convertRoutes(final ResultSet rs) throws
SQLException {
            Route route=new Route();
            route.setRouteName(rs.getString("ROUTE_NAME"));


            RouteDetails routeDetails=new RouteDetails();
            routeDetails.setRouteId(route);

            routeDetails.setBusNo(rs.getInt("ROUTE_BUS_NO"));
routeDetails.setDepartureDate(rs.getDate("DEPARTURE_DATE").toLocalDate
());


routeDetails.setDepartureTime(rs.getTime("DEPARTURE_TIME").toLocalTim
e());

            RouteIpDetails routeIpDetails=new RouteIpDetails();
            routeIpDetails.setRouteId(routeDetails);

            routeIpDetails.setIp1(rs.getString("IP_1"));
            routeIpDetails.setIp2(rs.getString("IP_2"));
            routeIpDetails.setIp3(rs.getString("IP_3"));
            routeIpDetails.setIp4(rs.getString("IP_4"));
            routeIpDetails.setIp5(rs.getString("IP_5"));
            routeIpDetails.setIp6(rs.getString("IP_6"));



            return routeIpDetails;

    }
    public List<RouteIpDetails> listByRouteName(Route route)
    {
                        final String sql="SELECT
seed_route.ROUTE_NAME,`seed_route_details`.`ROUTE_BUS_NO`,`seed_ro
ute_details`.`DEPARTURE_DATE`,`seed_route_details`.`DEPARTURE_TIME
```

```
"`route_details`.`IP_1`,`route_details`.`IP_2`,`route_details`.`IP_3`,`route_detai
ls`.`IP_4`,`route_details`.`IP_5`,`route_details`.`IP_6` FROM"+
"`seed_route`          JOIN          `seed_route_details`          ON
`seed_route`.`ID`=`seed_route_details`.`ROUTE_ID` JOIN `route_details` ON
`seed_route_details`.`ID`=`route_details`.`ID`"+
"WHERE `seed_route`.`ROUTE_NAME`=?";
		Object[] params={route.getRouteName()};
		return jdbcTemplate.query(sql,params, (rs, rowNum) -> {
			RouteIpDetails routeIpDetails=convertRoutes(rs);
			return routeIpDetails;
		});
	}


}
```

Web Application Coding :

Application.java

```
package com.jeppiaar;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.context.annotation.ComponentScan;
@SpringBootApplication
@ComponentScan(basePackages = "com.jeppiaar")
public class Application {
	public static void main(String[] args) {
		SpringApplication.run(Application.class);
	}
}
```

Controller Classes Code :

Admin Login Controller :

```
package com.jeppiaar.controller;

import java.time.LocalDate;
import java.time.LocalTime;
import java.util.List;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpSession;

import org.springframework.stereotype.Controller;
import org.springframework.ui.ModelMap;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestParam;

import com.jeppiaar.dao.AdminLoginDAO;
import com.jeppiaar.dao.RouteDetailsDAO;
import com.jeppiaar.dao.RouteIpDetailsDAO;
import com.jeppiaar.model.AdminLogin;
import com.jeppiaar.model.RouteIpDetails;

@Controller
@RequestMapping("/admin")
public class AdminLoginController {

@GetMapping
          public    String    listAllRoutes(ModelMap    modelMap,HttpSession
httpSession)
       {
    AdminLogin            admin            =            (AdminLogin)
httpSession.getAttribute("LOGGED_USER");
    if(admin!=null){
                        RouteIpDetailsDAO      routeIpDetailsDAO=new
RouteIpDetailsDAO();
List<RouteIpDetails> routes=routeIpDetailsDAO.listRoutes();
              modelMap.addAttribute("ROUTE_LIST",routes);
```

```
                    return "adminviewroutes.jsp";
            }
        else {
                return "redirect:/";
        }
            }
@PostMapping("/login")
        public              String              login(@RequestParam("username")String
username,@RequestParam("password")String password,HttpSession session)
        {   AdminLogin admin=new AdminLogin();
           admin.setUserName(username);
                AdminLoginDAO adminLoginDAO=new AdminLoginDAO();
                boolean       valid=adminLoginDAO.functionValidLogin(username,
password);
                if(valid)
                        {
                        session.setAttribute("LOGGED_USER",admin);
                        return "redirect:../admin";
               }

                return "redirect:../home.jsp";

        }
@PostMapping("/addroute")
public         String         addRoute(@RequestParam("routename")         String
routename,@RequestParam("endpoint")String
endpoint,@RequestParam("busno")  Integer  busno,@RequestParam("depdate")
String depdate,@RequestParam("deptime") String deptime,
                @RequestParam("ip1")  String  ip1,@RequestParam("ip2")  String
ip2,@RequestParam("ip3")      String      ip3,@RequestParam("ip4")      String
ip4,@RequestParam("ip5")      String      ip5,@RequestParam("ip6")      String
ip6,HttpSession httpSession)
{
        AdminLogin
admin=(AdminLogin)httpSession.getAttribute("LOGGED_USER");
if (admin !=null){
        RouteDetailsDAO dao = new RouteDetailsDAO();
        String errmsg = null;
        System.out.println(depdate);
        System.out.println(deptime);
        LocalDate departureDate = LocalDate.parse(depdate);
        LocalTime departureTime=LocalTime.parse(deptime);


        dao.addRoute(routename,endpoint , busno, departureDate, departureTime,
ip1, ip2, ip3, ip4, ip5, ip6, errmsg);
        return "redirect:../admin";
}
else
{
        return "redirect:/";
}
}
@PostMapping("/deleteroute")
public String deleteRoute(@RequestParam("busno") Integer busno,HttpSession
httpSession)
{
        AdminLogin
admin=(AdminLogin)httpSession.getAttribute("LOGGED_USER");
        if (admin != null){
        RouteDetailsDAO dao=new RouteDetailsDAO();
        String errmsg=null;
        dao.deleteRoute(busno, errmsg);
        return "redirect:../admin";
}
        else
        {
                return "redirect:/";
        }
}

@PostMapping("/updateroutename")
public String updateRouteNam(@RequestParam("busno") Integer
busno,@RequestParam("routename") String routeName,HttpSession
httpSession)
{
        AdminLogin
admin=(AdminLogin)httpSession.getAttribute("LOGGED_USER");
if(admin != null){
        RouteDetailsDAO dao=new RouteDetailsDAO();
        String errmsg=null;
        dao.updateRouteName(busno, routeName, errmsg);
        return "redirect:../admin";
}
else
{
        return "redirect:/";
}
}
@GetMapping("/updateroutename")
public String updateRouteName(HttpSession httpSession)
{
        AdminLogin
admin=(AdminLogin)httpSession.getAttribute("LOGGED_USER");
if(admin !=null){
        return "redirect:/updateroutename.jsp";

}
else
{
        return "redirect:/";
}
}
@PostMapping("/updateroutebusno")
public    String    updateRouteBusNo(@RequestParam("routename")    String
routeName,@RequestParam("busno") Integer busno,HttpSession httpSession)
{
        AdminLogin
admin=(AdminLogin)httpSession.getAttribute("LOGGED_USER");
if (admin != null){
        RouteDetailsDAO dao=new RouteDetailsDAO();
        String errmsg=null;
        dao.updateRouteBusNo(routeName, busno, errmsg);
        return "redirect:../admin";
}
else
{
        return "redirect:/";


}
}
```

```
public String updateRouteIntemediary(@RequestParam("busno") Integer busno,@RequestParam("ip1") String ip1,@RequestParam("ip2") String ip2
        ,@RequestParam("ip3") String ip3,@RequestParam("ip4") String ip4,@RequestParam("ip5") String ip5,@RequestParam("ip6") String ip6,HttpSession httpSession)
{
        AdminLogin
admin=(AdminLogin)httpSession.getAttribute("LOGGED_USER");
if (admin !=null)
{


        RouteDetailsDAO dao=new RouteDetailsDAO();
        String errmsg=null;
        dao.updateRouteIntermediary(busno, ip1, ip2, ip3, ip4, ip5, ip6, errmsg);
        return "redirect:../admin";
}
else
{
        return "redirect:/";

}
}
@GetMapping("/logout")
public String logout(HttpServletRequest request) {
        HttpSession httpSession = request.getSession(false);
        Object userSession = httpSession.getAttribute("LOGGED_USER");
        if (userSession != null) {
                httpSession.invalidate();
                return "redirect:/";
        } else {
                return "redirect:/";
        }
}
}
```

HomeController.java :

```
package com.jeppiaar.controller;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RequestMapping;
```

```
@Controller
@RequestMapping("/")
public class HomeController {
	@GetMapping
	public String home() {

		return "home.jsp";
	}
}
```

RouteIpDetailsController.java :

```
package com.jeppiaar.controller;
import java.util.List;
import org.springframework.stereotype.Controller;
import org.springframework.ui.ModelMap;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestParam;

import com.jeppiaar.dao.RouteIpDetailsDAO;
import com.jeppiaar.model.Route;
import com.jeppiaar.model.RouteIpDetails;

@Controller
@RequestMapping("/routes")
public class RouteIpDetailsController {

RouteIpDetailsDAO routeIpDetailsDAO=new RouteIpDetailsDAO();

@GetMapping
	public String index(ModelMap modelMap)
	{

		List<RouteIpDetails> routes=routeIpDetailsDAO.listRoutes();
		modelMap.addAttribute("ROUTE_LIST",routes);

		return "viewroutes.jsp";
	}
```

```
                        public        String        searchByRouteName(ModelMap
modelMap,@RequestParam("routename") String routeName)
   {
       Route route=new Route();
       route.setRouteName(routeName);
       List<RouteIpDetails>
routes=routeIpDetailsDAO.listByRouteName(route);
       modelMap.addAttribute("ROUTE_LIST_BY_NAME",routes);

       return "../searchroutes.jsp";

   }
}

 {
      height: 100%;
     }
     /* Optional: Makes the sample page fill the window. */
     html, body {
      height: 100%;
      margin: 0;
      padding: 0;
     }
   </style>


  </head>
  <body>
 <jsp:include page="header.jsp"></jsp:include>
   <div id="map"></div>
   <script>

     function initMap() {
      var myLatLng = {lat: 12.8686, lng:80.2211};
      var thiruvallurlatlong ={lat : 13.1444 , lng:79.8940};
      var manavalannagar ={lat : 12.9331 , lng: 80.1579};
      var kallarai={lat:13.049514,lng:80.104778};
      var vellavudu={lat:13.0675 , lng:80.0356};
      var flightPlanCoordinates = [
         {lat: 12.8686, lng:80.2211},
         {lat :12.9331 , lng: 80.1579},
    {lat: 13.0675 , lng:80.0356},
    {lat: 13.1444 , lng:79.8940}
   ];
  var flightPath = new google.maps.Polyline({
    path: flightPlanCoordinates,
    geodesic: true,
    strokeColor: '#FF0000',
    strokeOpacity: 1.0,
    strokeWeight: 2
   });

  var startImage = 'http://maps.google.com/mapfiles/ms/icons/red-dot.png';
    var  intermediary  =  'http://maps.google.com/mapfiles/ms/icons/yellow-
dot.png';
  var endHome = 'http://maps.google.com/mapfiles/ms/icons/green-dot.png';
  var map = new google.maps.Map(document.getElementById('map'), {
   zoom: 9,
   center: myLatLng
  });

  var marker = new google.maps.Marker({
   position: myLatLng,
   map: map,
   title: 'Jeppiaar Engineering College Start',
   icon: startImage
  });
  var marker = new google.maps.Marker({
   position: vellavudu,
    map: map,
    title: 'Vellavudu',
    icon : intermediary
   });
  var marker = new google.maps.Marker({
   position: thiruvallurlatlong,
    map: map,
    title: 'Thiruvallur Home',
    icon : endHome
   });
  var marker = new google.maps.Marker({
      position: manavalannagar,
      map: map,
      title: 'Manavalan Nagar',
      icon : intermediary
    });
  var marker = new google.maps.Marker({
      position: kallarai,
      map: map,
      title: 'Kallarai',
      icon : intermediary
    });
  flightPath.setMap(map);

}
```

8.2 APPENDIX 2
SCREENSHOTS

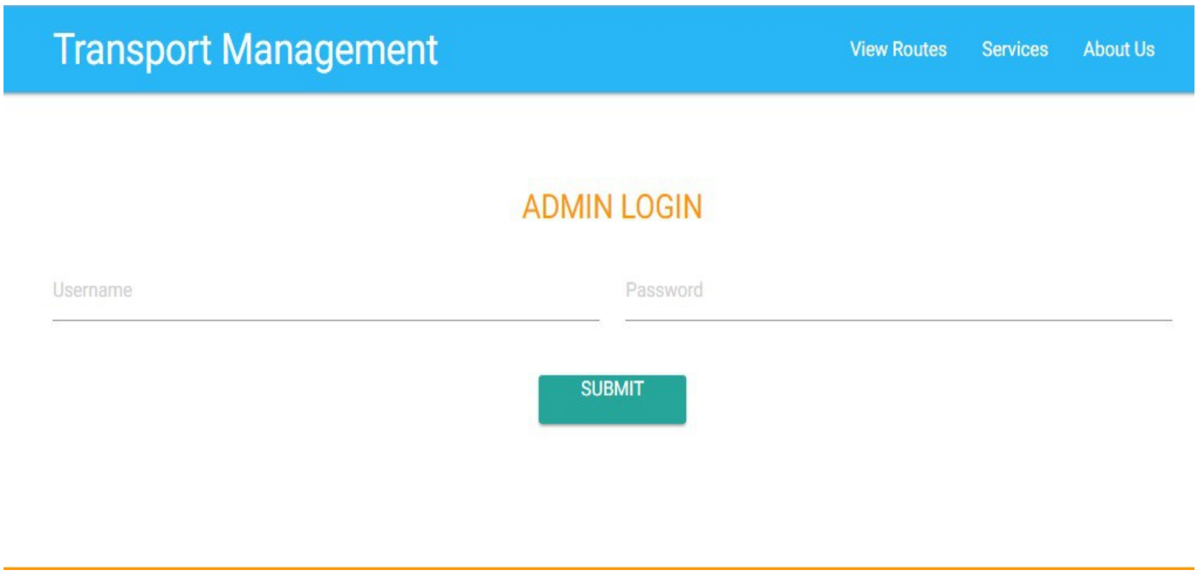
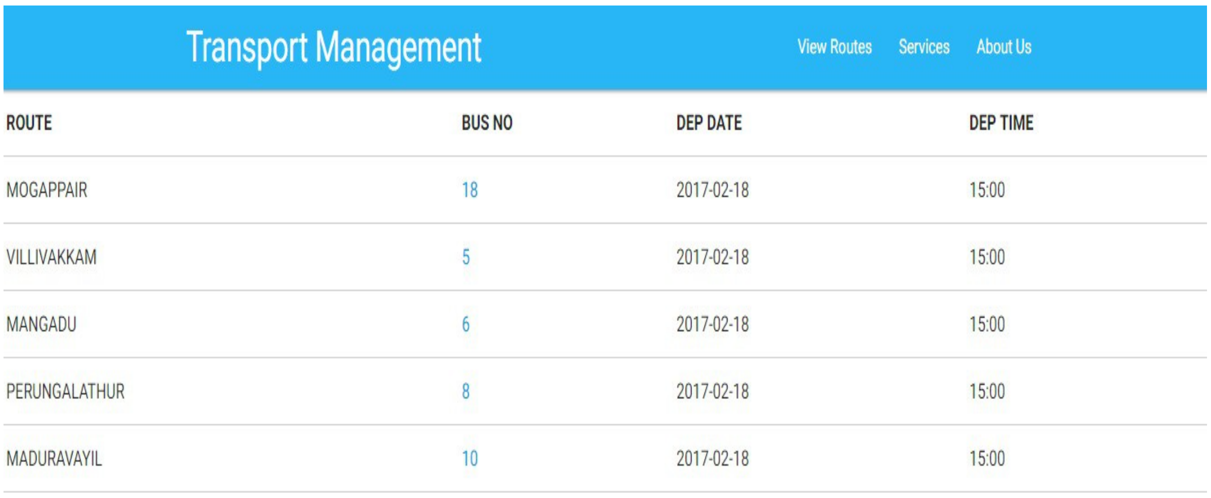
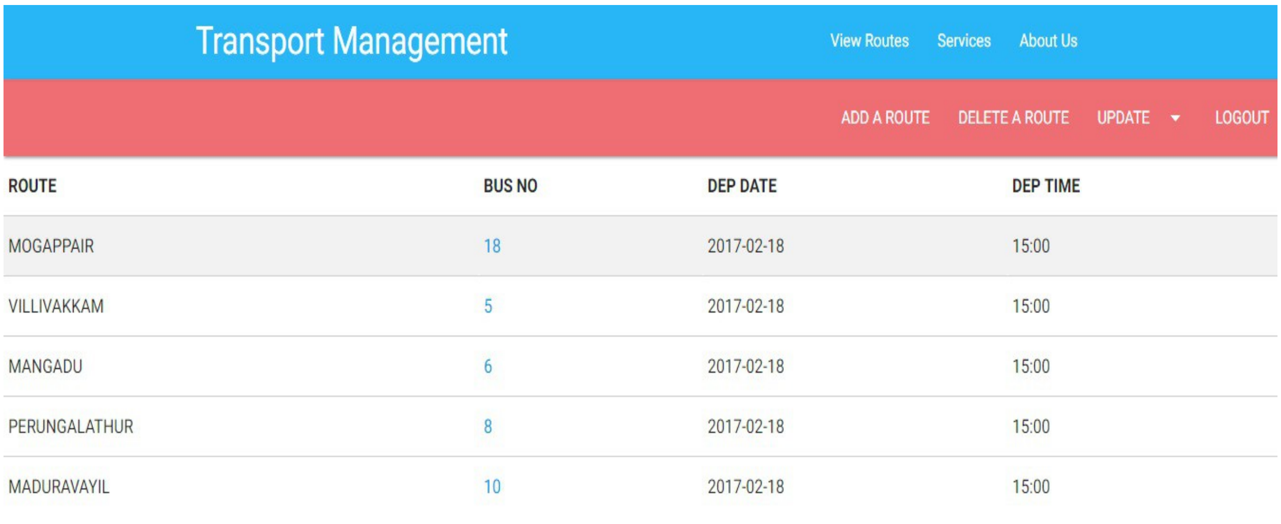


Fig 8.2.1 Login page for the admin.



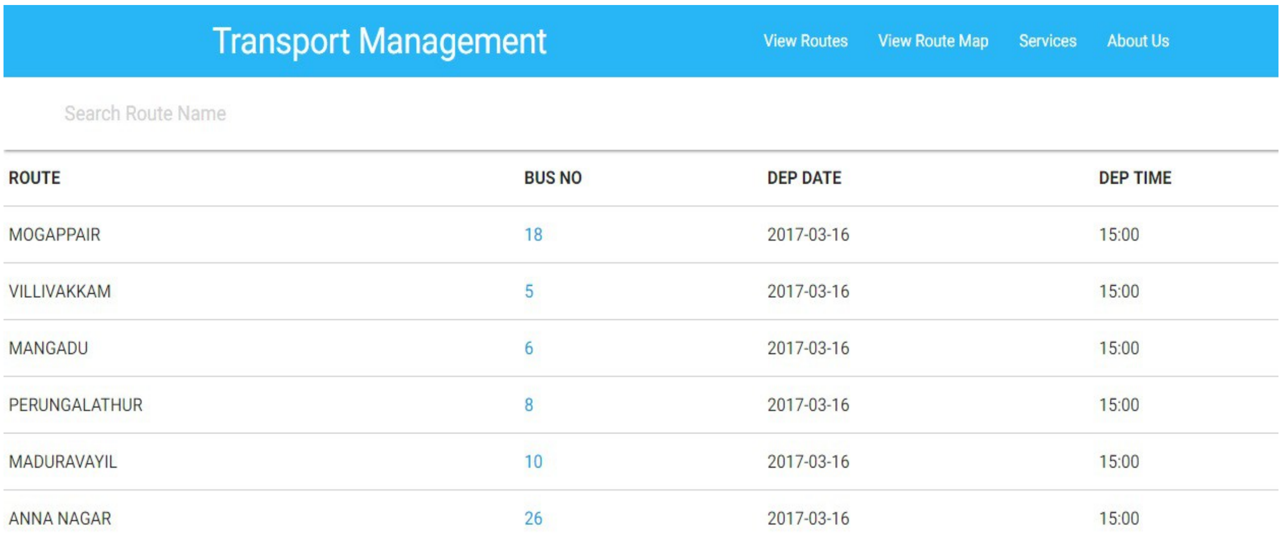
ROUTE	BUS NO	DEP DATE	DEP TIME
MOGAPPAIR	18	2017-02-18	15:00
VILLIVAKKAM	5	2017-02-18	15:00
MANGADU	6	2017-02-18	15:00
PERUNGALATHUR	8	2017-02-18	15:00
MADURAVAYIL	10	2017-02-18	15:00

Fig 8.2.2 View routes for the particular day for the students.



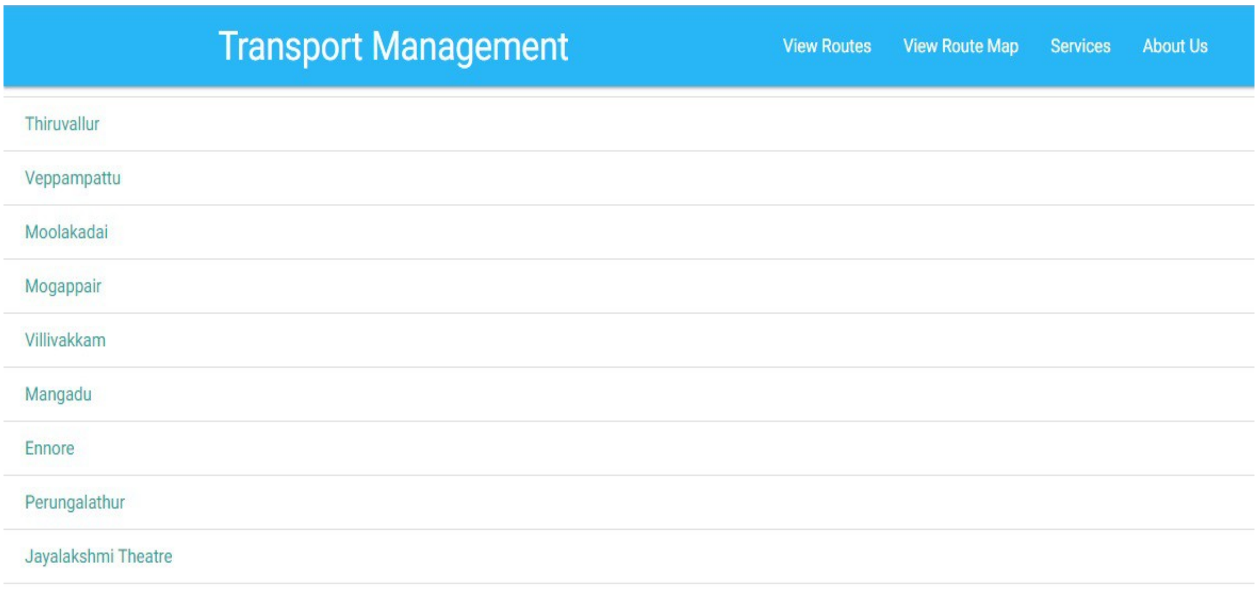
ROUTE	BUS NO	DEP DATE	DEP TIME
MOGAPPAIR	18	2017-02-18	15:00
VILLIVAKKAM	5	2017-02-18	15:00
MANGADU	6	2017-02-18	15:00
PERUNGALATHUR	8	2017-02-18	15:00
MADURAVAYIL	10	2017-02-18	15:00

Fig8.2.3 View routes for the particular day for the admin and options to add , update , delete a route.



ROUTE	BUS NO	DEP DATE	DEP TIME
MOGAPPAIR	18	2017-03-16	15:00
VILLIVAKKAM	5	2017-03-16	15:00
MANGADU	6	2017-03-16	15:00
PERUNGALATHUR	8	2017-03-16	15:00
MADURAVAYIL	10	2017-03-16	15:00
ANNA NAGAR	26	2017-03-16	15:00

Fig 8.2.4 View routes for the particular day for the student and search option by rouetname.



Thiruvallur
Veppampattu
Moolakadai
Mogappair
Villivakkam
Mangadu
Ennore
Perungalathur
Jayalakshmi Theatre

Fig 8.2.5 Google Map for every route is displayed onClick.

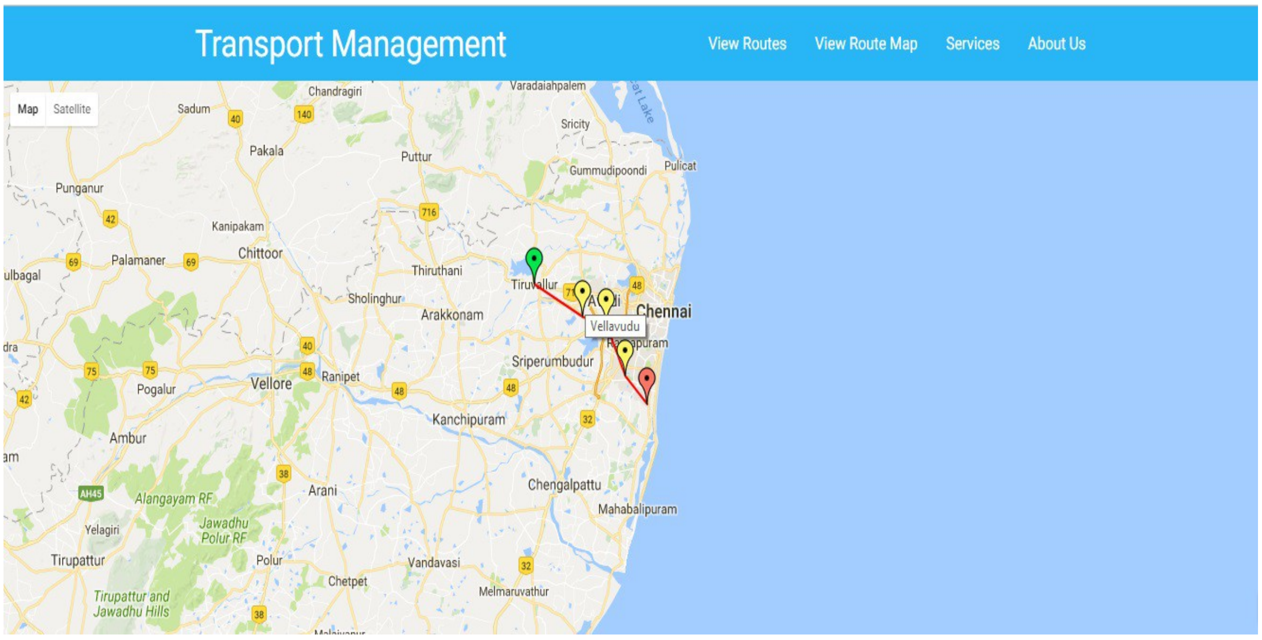


Fig 8.2.6 Google Map for a particular route is displayed.

CHAPTER 8
CONCLUSION

The web application was created in a motive to maintain a good transport management system in our college through this web application all routes can be accessed by students and staffs to know their respective routes in a daily basis.

This process of accessing the daily routes may consume less amount of paper and prevent the students and staffs from getting trapped in the stamp paid in the course of finding their daily routes in the system and also it will reduce the amount of effort taken by the students to find their daily routes.

Also this gives a new experience for them in knowing the routes since we display the routes in each floor of the college campus and the routes can also be seen using the mobile phones since we are using the responsive design to bring it in mobiles.

8.1FUTURE ENHANCEMENT

In future the web application can be used by all college in chennai and they can also get to know there routes from this application and can make the accessing of routes much easier to all students from other college .

REFERENCES

[1] "Bus Management System based on Zigbee and GSM/GPRS" by L.V.Zhian , Hu-Han.

[2] Spring Boot Documentation , https://docs.spring.io/spring-boot/docs/current-SNAPSHOT/reference/htmlsingle/

[3] Spring Web MVC Documentation , https://docs.spring.io/spring/docs/current/spring-framework reference/html/mvc.html

[4] Front End Design , Bootstrap Documentation , http://getbootstrap.com/getting-started/

[5] Programming Tutorials , https://www.w3schools.com/html/

[6] MySql Server 5.7 Reference Docs , https://dev.mysql.com/doc/refman/5.7/en/