**WEB APPLICATION FOR EFFICIENT TRAVEL DECISION MAKING USING MVC ARCHITECTURE**

A PROJECT REPORT

Presented to the Department of Computer Science

California State University, Long Beach

In Partial Fulfilment

of the Requirements for the Degree

Master of Science in Computer Science

Committee Members:

Frank Murgolo, Ph.D.

College Designee:

Antonella. Sciortino, Ph.D.

By Keval Soni

B.E, 2011, University of Mumbai, India

May 2017

ABSTRACT

**WEB APPLICATION FOR EFFICIENT TRAVEL DECISION MAKING USING MVC ARCHITECTURE**

By

Keval Soni

May 2017

**TABLE OF CONTENTS**

ABSTRACT

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

1. INTRODUCTION

2. SYSTEM ANALYSIS

3. CONCEPT BREAKDOWN

4. METHODOLOGY

5. RESULTANT SYSTEM

6. CONCLUSION

APPENDIX: USE CASES

REFERENCES

**LIST OF TABLES**

**LIST OF FIGURES**

**ABSTRACT**

Software Engineering is an extremely dynamic field which can be used to solve many real-world problems. Software engineering principles and design patterns provide abstract ways and approaches to be implemented while developing new software or resolving core engineering problem. Travel Helper is a web application built on Model-View-Controller (MVC) architecture.

Travel Helper is a web application developed using Spring MVC framework which provides information for deciding best option to travel from source to destination. Application uses REST API to fetch data from Google, Uber and Lyft API. The heterogeneous data is processed to make it structured for data processing and comparison. The processed data helps users to decide on best option for user to travel. Another feature of the application allows user to save the future travel and notify user best time to start the travel using Push Notification. Dashboard allows user to reflect upon travel history.

Model-View-Controller (MVC) is an architectural pattern which facilitates independent development and maintenance of view, model and controller.

Spring framework is used for application development. A framework is an outline, structure, built using design pattern and facilitates development of enterprise applications. Spring framework provides modules like Dependency Injection, Inversion of Control which helps in development of modules with high cohesion and loose coupling. Spring MVC module is used in development of web applications. Other major framework functionality integrated in

REST API are lightweight http protocol based web service used to exchanging data between applications.

Database operation is performed using Object Relational Mapping technology (ORM) along with Spring Transaction Manager.

Spring security module is implemented to handle user profiling and session management.

In this document, Object oriented analysis and design methodology and software design patterns are discussed which are used for development of travel helper web application.

**CHAPTER 1**

**INTRODUCTION**

The evolution of the computers in the last decade has been absolutely amazing. The pace at which the power and size of the computers have changed is immense. The concept of communicating information between computers using computer networks and protocols led to rise of World Wide Web famously known as Internet. Internet has grown at an enormous speed in a short span of time. Today, majority of the world population is connected by the Internet. Growth of internet has been popular because it is full of services: search engines,

online stores, weblogs, wikis, calculators, and games. Rather than installing all this data

and all these programs on your own computer, you install one program—a web browser

—and access the data and services through it.

Web based application are Internet based software which can be accessed using web browser. Web applications are developed based on client-server model where the application is deployed on server and it can be accessed using client like web browser. Web applications became immensely popular and have taken over the traditional desktop software applications. The reason is the simplicity and ease in application development, accessible anywhere, platform independence and ease in support and maintenance.

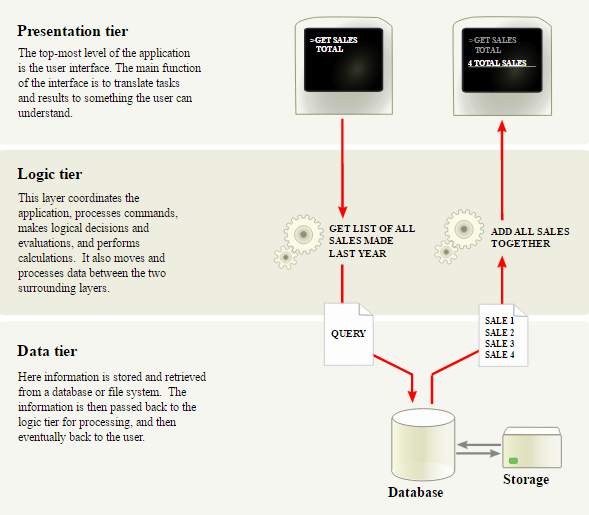
Today, every business and non-profit organization owns a web site used for business or informational purpose. It is easy to conduct business online in term of online shopping or trading etc. Web applications can be customized and designed with respect to business requirements. Therefore, web applications are services available to users over the Internet.

Web Applications can be of 2-tier or 3-tier architecture. 2-tier architecture consists of presentation and logic tier. 3-tier application consists of presentation, logic and data tier.

Presentation tier – This is the client facing layer consisting of user interface. User Interface displays information to users for reading and perform user actions on interface for future action.

Logic tier – Application logic is deployed on this tier. It acts as a coordinator between presentation and data tier. Requests from presentation layer are processed, perform computations based on user requests and interacts with database to store or retrieve data. Response is forwarded back to user interface.

Data tier – This tier consists of a data store containing database management system for storing data. The information is stored and retrieved from database. The processed information is passed back to logic tier and result is displayed to user interface.



Model View Controller or MVC as it is known popularly, is a software design pattern for developing web applications. A Model View Controller pattern is made up of the following three parts:

Model - The lowest level of the pattern which is responsible for maintaining data.

View – This is the client facing user interface responsible for displaying information to user.

Controller – It acts as an interface between model and view.

Spring is a Java based framework containing Spring MVC module which provides model-view-controller architecture that can be used for development of web applications.

Travel Helper is a web application developed using Spring MVC framework. Various modules of travel helper are user profiling, efficient travel decision making and schedule future travel. User can do analysis of past travel history using dashboard. Travel Helper is a web application that will demonstrate the benefits of developing web application using Spring MVC architecture. Travel Helper also showcase the ease of implementation and the power of REST API. This document also reflects upon the development of software using Object Oriented Analysis and Design methodology.

**CHAPTER 2**

**OBJECT ORIENTED ANALYSIS AND DESIGN – OLD**

**SYSTEM ANALYSIS – NEW**

**2.1 Goal**

The application should enable the user to select best and preferred travel option available. This will relieve users from doing multiple search across multiple applications to find cheapest or fastest travel option available. Features of pre-scheduling travel and real-time traffic congestion information will help users in saving time and money.

**2.2 Object Oriented Analysis and Design**

Object–Oriented Analysis and Design (OOAD) is the procedure of identifying software engineering requirements and developing software specifications in terms of a software system’s object model, which comprises of interacting objects. It is a popular technical approach for analysing, designing an application, system, or business by applying the object-oriented paradigm and visual modelling throughout the development life cycles to foster better stakeholder communication and product quality. The development of Travel Helper was done following the life cycle proposed by this model.

The Unified Modeling Language (UML) is a graphical language for OOAD that gives a standard way to write a software system’s blueprint. It helps to visualize, specify, construct, and document the artifacts of an object-oriented system. It is used to depict the structures and the relationships in a complex system.

2.2.1 **Functional and Non Functional Requirement**

2.2.2 **Use Case**

Use case is a behavioural diagram. It depicts the sequence of actions a user of a system can perform to achieve functionality of system. User has a role in the system which needs to be satisfied. Use Case describes complete details of the role, condition and action corresponding to functional requirement of the system.

Below are the various section of use case:

1. Use Case Name: Describes the name of the action.
2. Actors: User responsible for performing specified action.
3. Pre-Conditions: Conditions that need to be satisfied before the action specified in the use case can be performed.
4. Flow of Control: It lists the actions that needs to be performed for successful execution of use case.
5. Post Conditions: Conditions that should be executed successfully after the actor has completed actions required for the use case.
6. Error Conditions: Unsuccessful execution of the action specified by the use case due to possible errors or issues that can occur during execution of the use case.

Below list contains brief description of the use case designed to describe various functionalities for the Travel Helper application. All use cases have been included in Appendix A.

1. New Customer registration: This use case depicts the actions need performed by user to register new profile in application.
2. Search economical travel option:

**1.2 Scope of System**

**USE CASE and parts of use case**

**1.3 Functional Objectives**

**What functional objectives are and how they are diff from non-functional**

**Listing of Functional req**

**1.4 Non-Functional Objectives**

**What are non functional objectives**

**Listing of Non functional req**

**4.2 Class Diagram**

**Definition of class diagram**

**1 class diagram (appendix)**

**4.3 Sequence Diagram sequence diagram for each use case**

**Definition of sequence diag**

**Seq diag in appendix**

**CHAPTER 3**

**ARCHITECTURE AND FRAMEWORKS**

**https://www.youtube.com/watch?v=Q3yStECBuAg - spring security arch**

**CHAPTER 4**

**IMPLEMENTATION’S USE OF TECHNOLOGIES**

**4.1**

**4.4 Database**

**Definition of relational database and orm**

Database Design appendix

**USER\_PROFILE**

|  |  |  |
| --- | --- | --- |
| **Column Name** | **Datatype** | **Remark** |
| USER\_ID | Integer (20) | Primary key |
| FIRST\_NAME | Varchar (20) |  |
| LAST\_NAME | Varchar (20) |  |
| ADDR\_STREETNAME | Varchar (20) |  |
| ADDR\_APTNO | Varchar (20) |  |
| ADDR\_CITY | Varchar (20) |  |
| ADDR\_STATE | Varchar (20) |  |
| ADDR\_ZIP | Integer (10) |  |
| CONTACT\_NUMBER | Integer (20) |  |
| EMAIL | Varchar (50) |  |
| PASSWORD | Varchar (100) |  |

**TRAVEL\_REQUEST**

|  |  |  |
| --- | --- | --- |
| **Column Name** | **Datatype** | **Remark** |
| TRAVEL\_REQUEST\_ID | Integer (20) | Primary Key |
| USER\_ID | Integer (20) | Foreign Key |
| START\_POINT\_X\_COORDINATE | float (9,6) | **Google Co-ordinates** |
| START\_POINT\_Y\_COORDINATE | float (9,6) |  |
| END\_POINT\_X\_COORDINATE | float (9,6) |  |
| END\_POINT\_X\_COORDINATE | float (9,6) |  |
| REQUEST\_TIMESTAMP | Datetime |  |
| TRAVEL\_START\_TIMESTAMP | datetime |  |
| TRAVEL\_END\_TIMESTAMP | Datetime |  |
| TRAVEL\_DISTANCE\_IN\_MILES | Integer (10) |  |
| TRAVEL\_MODE\_SELECTED | tinyint (3) | References “TRAVEL\_MODE\_ID” from Table “TRAVEL\_MODE” |
| FINAL\_TRAVEL\_MODE | tinyint (3) | References “TRAVEL\_MODE\_ID” from Table “TRAVEL\_MODE” |
| TRAVEL\_DRIVE\_SELECTED | tinyint (3) | References “DRIVE\_ID” from Table “TRAVEL DRIVE” |
| FINAL\_TRAVEL\_DRIVE | tinyint (3) | References “DRIVE\_ID” from Table “TRAVEL DRIVE” |
| TRAVEL\_STATUS | Varchar (20) |  |

**FUTURE\_SCHEDULED\_TRAVEL**

|  |  |  |  |
| --- | --- | --- | --- |
| **Column Name** | **Datatype** | | **Remark** |
| RECORD\_ID | Autoincrement | | Primary Key |
| USER\_ID | Integer (20) | Foreign Key | |
| START\_POINT\_X\_COORDINATE | float (9,6) |  | |
| START\_POINT\_Y\_COORDINATE | float (9,6) |  | |
| END\_POINT\_X\_COORDINATE | float (9,6) |  | |
| END\_POINT\_X\_COORDINATE | float (9,6) |  | |
| PRE\_NOTIFICATION\_TIME\_IN\_MINUTES | Integer (20) |  | |
| TRAVEL\_MODE\_SELECTED | tinyint (3) | References “TRAVEL\_MODE\_ID” from Table “TRAVEL\_MODE” | |

**USER\_CLOUD\_NOTIFICATION\_ID**

|  |  |  |
| --- | --- | --- |
| **Column Name** | **Datatype** | **Remark** |
| USER\_ID | Integer (20) | Primary Key , Foreign Key |
| GCM\_REGISTERATION\_ID | Varchar (100) | Primary Key |

**TRAVEL\_DRIVE**

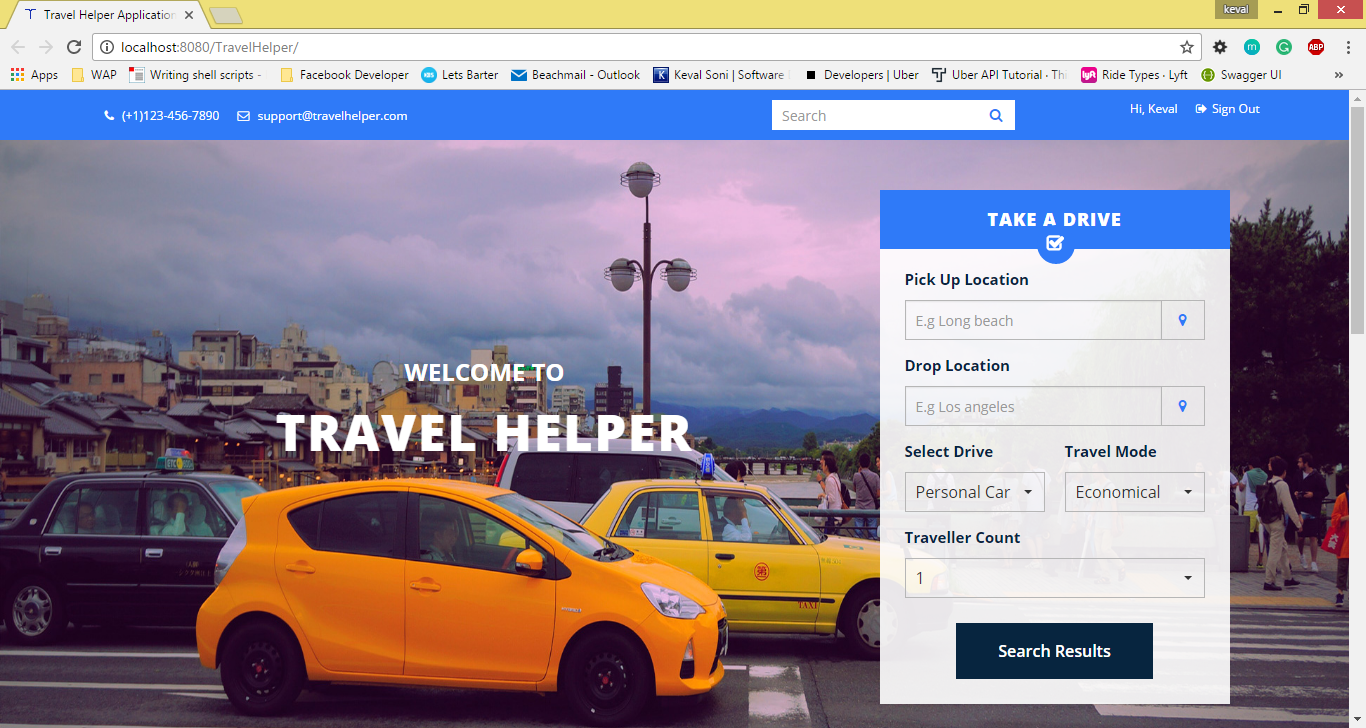
|  |  |  |
| --- | --- | --- |
| **Column Name** | **Datatype** | **Remark** |
| DRIVE\_ID | tinyint (3) | Primary Key |
| DRIVE\_NAME | Varchar (20) | Unique |

**TRAVEL\_MODE**

|  |  |  |
| --- | --- | --- |
| **Column Name** | **Datatype** | **Remark** |
| TRAVEL\_MODE\_ID | tinyint (3) | Primary Key |
| TRAVEL\_MODE\_NAME | Varchar (20) | Unique |

**CHAPTER 5**

**RESULTANT SYSTEM**



**Figure xx. Home page**

**CHAPTER 6**

**CONCLUSION**

**APPENDIX**

**USE CASE**

**TABLE 1:**

|  |  |
| --- | --- |
| Use-case Name: | 1. New Customer registration |
| Actors: | Customer |
| Pre-Conditions: | 1.   The application is running.  2.   Database connection established. |
| Flow of Control: | 1.   Customer clicks on ‘Register’ button on the system.  2.   System acknowledges and presents registration form to the customer.  3. Customer inputs information into appropriate fields.  4. Customer submits the form.  5. System saves the form. |
| Post-Conditions: | 1.   New customer created.  2.   New customer information stored in database. |
| Error-Conditions: | 1. Invalid characters or more than max characters allowed – system prompts to correct error inputs. 2. Fields left incomplete at time user submits form - system prompts user to complete input fields. |
| Non-Functional Requirements: | 1. Availability – Web application must be available 95% of time in a span of 24 hours. 2. Performance - Page load time should be less than or equal to 4 sec.   Application should be capable to handle minimum load of 100 users at a particular moment.   1. Security - Maximum failed login attempt allowed must be 5. 2. Compatibility – User Interface should be rendered as expected in browsers like Internet Explorer (above version 8), Mozilla Firefox, Chrome and Safari |

**TABLE 2:**

|  |  |
| --- | --- |
| Use-case Name: | 1. Customer browses for travelling from destination A to destination B in economical way |
| Actors: | Customer |
| Pre-Conditions: | 1. The application is running.  2. Database connection established.  3. Customer is registered with application.  4. API is running. |
| Flow of Control: | 1.   Customer clicks on ‘Economical Travel’ button on the system.  2.   System prompts for source and destination location from customer.  3.   System calls API to find travel cost using Uber, Lyft, LB Transit or Metro at current time.  4.   System shows all possible options with less cost option at top.  5.   Customer selects travel option.  5.1 If customer selects booking option, see use case no. 4  6. System saves travel preference selected by customer. |
| Post-Conditions: | 1.   New customer travel entry is saved into database. |
| Error-Conditions: | 1. Invalid characters or more than max characters allowed – system prompts to correct error inputs. 2. Fields left incomplete at time user submits form - system prompts user to complete input fields. |
| Non-Functional Requirements: | 1. Interoperability – Web application should be able to connect and request/respond data to Travel API using REST API 2. Availability: Connectivity with Uber, Lyft, LB Transit and Metro should be available for 90%. |

**TABLE 3:**

|  |  |
| --- | --- |
| Use-case Name: | 1. Customer browses for travelling from destination A to destination B in fastest way |
| Actors: | Customer |
| Pre-Conditions: | 1. The application is running.  2. Database connection established.  3. Customer is registered with application.  4. API is running. |
| Flow of Control: | 1.   Customer clicks on ‘Faster Travel’ button on the system.  2.   System prompts for source and destination location from customer.  3.   System calls API to find travel cost using Uber, Lyft, LB Transit or Metro at current time.  4.   System shows all possible options with fastest travel option at the top.  5.   Customer selects travel option.  5.1 If customer selects booking option, see use case no. 4  6. System saves travel preference selected by customer. |
| Post-Conditions: | 1.   New customer travel entry is saved into database. |
| Error-Conditions: | 1. Invalid characters or more than max characters allowed – system prompts to correct error inputs. 2. Fields left incomplete at time user submits form - system prompts user to complete input fields. |
| Non-Functional Requirements: | 1. Interoperability – Web application should be able to connect and request/respond data to Travel API using REST API 2. Availability: Connectivity with Uber, Lyft, LB Transit and Metro should be available for 90%. |

**TABLE 4:**

|  |  |
| --- | --- |
| Use-case Name: | 1. Customer books Uber or Lyft |
| Actors: | Customer |
| Pre-Conditions: | 1.   The application is running.  2.   Database connection established.  3.   Customer is registered with application. |
| Flow of Control: | 1.   Customer clicks on ‘Book Now’ button on the system.  2.   System calls API for booking.  3.   Uber or Lyft car is booked and details are sent to customer. |
| Post-Conditions: | 1.   Details regarding booking are stored in database. |
| Error-Conditions: | 1. Time out in API call |
| Non-Functional Requirements: | Usability: See Non Functional Requirement Section 1 |

**TABLE 5:**

|  |  |
| --- | --- |
| Use-case Name: | 1. Save scheduled travel in application |
| Actors: | Customer |
| Pre-Conditions: | 1.   The application is running.  2.   Database connection established. |
| Flow of Control: | 1. Customer clicks on ‘Schedule travel’ button on the system.  2. System acknowledges and presents travel schedule form to the customer.  3. Customer enters destination and expected time to reach destination.  4. Customer submits the form.  5. System saves the form. |
| Post-Conditions: | 1. Travel details are stored. |
| Error-Conditions: | 1. Invalid characters or more than max characters allowed – system prompts to correct error inputs.  2. Fields left incomplete at time user submits form - system prompts user to complete input fields. |
| Non-Functional Requirements: | Performance: Response time to complete the save should be less than 1 minute. |

**TABLE 6:**

|  |  |
| --- | --- |
| Use-case Name: | 1. System prompts user about travel start time |
| Actors: | Travel Scheduler (TS) |
| Pre-Conditions: | 1. The application is running.  2. Database connection established.  3. API is running.  4. Google cloud connectivity is established.  5. Customer Internet is working. |
| Flow of Control: | 1. TS sends notification to customer regarding start of travel to reach destination in time. 2. Customer clicks on notification to view notification. 3. Based on customer preferences, Go to use case 2 or 3. |
| Post-Conditions: | See Use case 2 or 3 |
| Error-Conditions: | NA |
| Non-Functional Requirements: | Interoperability: Data exchange with google cloud messaging API should be done using REST.  Availability: Connectivity with google cloud should be established 98% of time. |

**TABLE 7:**

|  |  |
| --- | --- |
| Use-case Name: | 1. User selects Personal Car travel to reach destination |
| Actors: | Customer |
| Pre-Conditions: | 1. The application is running.  2. Database connection established.  3. API is running. |
| Flow of Control: | 1. Customer clicks on ‘Traveling by my car’ button on the system.  2. System presents best route to travel to reach destination. |
| Post-Conditions: | 1. Travel details are stored. 2. Travel is monitored to notify congestion or risk on route. |
| Error-Conditions: | 1. Request Time-out. |
| Non-Functional Requirements: | Usability: See Non Functional Requirement Section 1 |

**TABLE 8:**

|  |  |
| --- | --- |
| Use-case Name: | 1. System notifies user about change in route due to accident or congestion. |
| Actors: | System |
| Pre-Conditions: | 1. The application is running.  2. Database connection established.  3. API is running.  4. Google cloud connectivity is established.  5. Customer Internet is working. |
| Flow of Control: | 1. System sends notification to customer regarding option to change route during travel with details of problem.  2. Customer clicks on “Alternate Travel route”.  3. Alternate route displayed to customer |
| Post-Conditions: | NA |
| Error-Conditions: | NA |
| Non-Functional Requirements: | Interoperability: Data exchange with google cloud messaging API should be done using REST.  Availability: Connectivity with google cloud should be established 98% of time. |

**TABLE 9:**

|  |  |
| --- | --- |
| Use-case Name: | 1. Customer views travel dashboard |
| Actors: | Customer |
| Pre-Conditions: | 1.   The application is running.  2.   Database connection established.  3.   Customer is registered with application. |
| Flow of Control: | 1. Customer clicks on ‘Travel Dashboard’ button on the system.  2. System queries database for fetching details.  3. Travel summary is displayed to customer. |
| Post-Conditions: | NA |
| Error-Conditions: | NA |
| Non-Functional Requirements: | Usability: See Non Functional Requirement Section 1 |

**TABLE 10:**

|  |  |
| --- | --- |
| Use-case Name: | 1. Customer filters details obtained on dashboard based on date range |
| Actors: | Customer |
| Pre-Conditions: | 1.   The application is running.  2.   Database connection established.  3.   Customer is registered with application. |
| Flow of Control: | 1. Customer clicks on ‘Filter Results’ button on the system.  2. System queries database based on filter for fetching details.  3. Travel details are displayed to customer. |
| Post-Conditions: | NA |
| Error-Conditions: | NA |
| Non-Functional Requirements: | Usability: See Non Functional Requirement Section 1 |

**TABLE 11:**

|  |  |
| --- | --- |
| Use-case Name: | 1. Customer wants to download report |
| Actors: | Customer |
| Pre-Conditions: | 1.   The application is running.  2.   Database connection established.  3.   Customer is registered with application. |
| Flow of Control: | 1. Customer clicks on ‘Download report’ button on the system.  2. System generates pdf format report.  3. Customer saves report in his computer. |
| Post-Conditions: | NA |
| Error-Conditions: | NA |
| Non-Functional Requirements: | Accessibility: Document should be accessible within 5 minutes.  Usability: See Non Functional Requirement Section 1 |