**Project**

**On**

**Tour Management System**

****

Dawa Tenzing Lama

00172886

Email: tenzing0070@gmail.com

Computing Project

Level 5 Diploma in Computing

Softwarica College of IT & E-Commerce

Kathmandu, Nepal

*April 2019*

Submitted to: Kiran Rana

**Abstract**

This report documents all the activities executed in each of the phases of software development lifecycle titled “Tour Management System”. All the requirements, use cases with scenario description, system architecture, the static and dynamic models and implementation details along with testing report and user manual is included in this report.

The project is a data management and analysis web-based application that designed to be used in travel booking activity. It allows for proper data recording, booking of the destination with proper information. It helps to save time as well as gives travelers to know the local information, price of the surroundings.

**Acknowledgement**

I am extremely grateful to all my supporters whose input, feedback and backing went onto making my envisaged project into a reality. This includes my college, Softwarica providing me the ecosystem with its amenities to carry out my academic undertakings in particular this project.

I am thankful to my module leader, Kiran Rana for diligently supervising my project and providing me the guidelines for software analysis, modelling, implementation and documentation methods.

Rekha kapali and Sapana Rajbansi have always been courteous to me, responding favorably to my email request and helping with Turnitin account, verifications and submission of project materials.

Lastly, my heartfelt gratitude goes to my family and friends who were there to understand and supportive in different ways.

Contents

[**Table of Figures** 6](#_Toc5663803)

[**Chapter 1 Introduction** 7](#_Toc5663804)

[**Introduction** 8](#_Toc5663805)

[**Background of the project** 8](#_Toc5663806)

[**Aims** 9](#_Toc5663807)

[**Objectives** 9](#_Toc5663808)

[**Over view of the project design** 10](#_Toc5663809)

[**Chapter 2 ANALYSIS** 10](#_Toc5663810)

[**Introduction to Analysis** 11](#_Toc5663811)

[**Need for Analysis** 11](#_Toc5663812)

[**Objected oriented Analysis** 11](#_Toc5663813)

[**Pitfalls and Merits** 12](#_Toc5663814)

[**Requirement Analysis** 13](#_Toc5663815)

[**Functional and Non-Functional** 13](#_Toc5663816)

[**Functional requirements:** 14](#_Toc5663817)

[**Non-Functional Requirements:** 15](#_Toc5663818)

[**NLA** 16](#_Toc5663819)

[**Initial Class Diagram** 16](#_Toc5663820)

[**Use-Case Diagram** 19](#_Toc5663821)

[**System Architecture** 21](#_Toc5663822)

[**Chapter 3 DESIGN** 22](#_Toc5663823)

[**Design Introduction** 23](#_Toc5663824)

[**Design Pattern** 23](#_Toc5663825)

[**Dynamic modelling** 25](#_Toc5663826)

[**Sequence Diagram** 25](#_Toc5663827)

[**Login Sequence Diagram** 26](#_Toc5663828)

[**Admin Sequence Diagram** 28](#_Toc5663829)

[**User Sequence Diagram** 30](#_Toc5663830)

[**Activity Diagram** 32](#_Toc5663831)

[**Admin Activity Diagram** 33](#_Toc5663832)

[**Generating receipt activity diagram** 35](#_Toc5663833)

[**Package reservation activity diagram** 37](#_Toc5663834)

[**Structural Modelling** 39](#_Toc5663835)

[**Class Diagram** 39](#_Toc5663836)

[**Class Diagram** 40](#_Toc5663837)

[**Database Modelling** 41](#_Toc5663838)

[**Entity Relationship Diagram** 41](#_Toc5663839)

[**Data Dictionary** 44](#_Toc5663840)

[**Meta – Data** 44](#_Toc5663841)

[**UI Modelling** 46](#_Toc5663842)

[**Registration** 46](#_Toc5663843)

[**Login Form** 47](#_Toc5663844)

[**Home Page** 48](#_Toc5663845)

[**Gallery** 49](#_Toc5663846)

[**Package** 50](#_Toc5663847)

[**Booking** 51](#_Toc5663848)

[**Contact** 52](#_Toc5663849)

[**Admin Dashboard** 53](#_Toc5663850)

[**Package Booked Information** 54](#_Toc5663851)

[**Admin package CRUD** 54](#_Toc5663852)

[**Admin Gallery Function** 55](#_Toc5663853)

[**User Information** 55](#_Toc5663854)

[**User Comment** 56](#_Toc5663855)

[**Chapter 4 IMPLEMENTATION** 57](#_Toc5663856)

[**Introduction** 58](#_Toc5663857)

[**Chapter 5 TESTING** 59](#_Toc5663858)

[**Introduction** 60](#_Toc5663859)

[**Principles of Testing** 60](#_Toc5663860)

[**Black box Testing** 60](#_Toc5663861)

[**Unit Testing** 69](#_Toc5663862)

[**Chapter 6 Other Project Issues** 79](#_Toc5663863)

[**Risk Management** 80](#_Toc5663864)

[**Scheduling** 89](#_Toc5663865)

[**Work Breakdown Structure (WBS) / Scheduling** 89](#_Toc5663866)

[**Work Breakdown Structure** 89](#_Toc5663867)

[**Milestones** 91](#_Toc5663868)

[**Description of Milestones:** 92](#_Toc5663869)

[**Scheduling / Gantt Chart** 93](#_Toc5663870)

[**Gantt chart** 94](#_Toc5663871)

[**Future Works** 95](#_Toc5663872)

[**User Manual** 95](#_Toc5663873)

[**Limitation** 96](#_Toc5663874)

[**Chapter 7 Conclusion** 97](#_Toc5663875)

[**Conclusion** 98](#_Toc5663876)

[**Chapter 8 Reference & Bibliography** 99](#_Toc5663877)

[**Chapter 9 Appendix** 101](#_Toc5663878)

# **Table of Figures**

[Figure 1Usecase Diagram 13](#_Toc5145115)

[Figure 2Initial class diagram 15](#_Toc5145116)

[Figure 3Class diagram 19](#_Toc5145117)

[Figure 4Admin activity. 21](#_Toc5145118)

[Figure 5Generating receipt 23](#_Toc5145119)

[Figure 6Package reservation 25](#_Toc5145120)

[Figure 7Login sequence diagram 27](#_Toc5145121)

[Figure 8Admin sequence diagram 29](#_Toc5145122)

[Figure 9User sequence diagram 30](#_Toc5145123)

[Figure 10ER diagram 32](#_Toc5145124)

[Figure 11Registration 35](#_Toc5145125)

[Figure 12Login 36](#_Toc5145126)

[Figure 13Homepage 37](#_Toc5145127)

[Figure 14Gallery 38](#_Toc5145128)

[Figure 15Package 39](#_Toc5145129)

[Figure 16Package details 39](#_Toc5145130)

[Figure 17Book package 40](#_Toc5145131)

[Figure 18Comment box 41](#_Toc5145132)

[Figure 19Admin page 42](#_Toc5145133)

[Figure 20Admin package booked information. 43](#_Toc5145134)

[Figure 21Admin Package CRUD 43](#_Toc5145135)

[Figure 22Admin Gallery Function. 44](#_Toc5145136)

[Figure 23User information. 44](#_Toc5145137)

# **Chapter 1 Introduction**

## **Introduction**

Tour management is web-based application which is designed to automate the travel process to the customers through online facilities for travelling, easy access to the relevant information and necessary travel booking and arrangement.

The complete information about the project has been highlighted in this project proposal. This application is going to add easiness for those users using such type of application and meet their satisfaction. This application will have various features for fulfilling the user requirement.

## **Background of the project**

Touring is a part of a life as which can sometimes lead to uncertain situation. So, in order to overcome that problem, this project will help users to get all the required information about the schedule and safe routes for their trip.

Solving this problem can uplift the range of tourism to go higher and increase the revenue too. Through this, customers can be able to utilized various benefits and resources such as safe routes, booking, destination information, reviews about the places, environments etc.

The project will be developed using Object Oriented PHP. Different techniques and procedures will be implemented that will enhance the quality of application.

## **Aims**

* To help the organization to build a proper system to record the travelling information.
* To provide proper information of the travelling routes, destination, locals etc.
* To make desktop application and online web application to create communication and sharing of information.
* To develop database system to store the information of the travelers, places etc.

## **Objectives**

* To collect the exact number of data of the travelers of different parts of the world.
* To give a brief information of the different beautiful places.
* To make user friendly GUI system which can also be used by non-technical users (finding user experiences)

## **Over view of the project design**

* A web-based application that maintains a centralized repository of all related information.
* To develop a system that automates the processes and activities of travel.
* Main aim is to implemented system using which one can perform all operations related to travelling, sight-seeing.
* To provide information to the travelers in detail.
* Provide reservation facility to the customers for travelling.

# **Chapter 2 ANALYSIS**

## **Introduction to Analysis**

Analyzing is the method of identifying and documenting of requirement of the proposed system. In the analyzing phrase, the first phase is to feasibility study after that we model the system in use case diagram and class diagram which is also known as system modelling.

## **Need for Analysis**

It is important because it helps an organization determine the gaps that are preventing it from reaching its desired goals. ... Knowing what is working well and what needs to be changed is crucial to progressing effectively towards those goals and making an organization successful.

## **Objected oriented Analysis**

The use of modeling to define and analyze the requirements necessary for success of a system. Object-oriented analysis is a process that groups items that interact with one another, typically by class, data or behavior, to create a model that accurately represents the intended purpose of the system as a whole. Object-oriented analysis does not factor implementation limitations into the model.

## **Pitfalls and Merits**

**Merits**

1. Operation and distribution costs are decreased.
2. Technology allows high-quality and effective market research.
3. Medium of promoting Tourism and sales.
4. Visualization of Tourism services and products through Multimedia Technology leaves a lasting impression on potential customers.
5. Increased efficiency of Travel Agencies and Tour Operators by speeding up communication.
6. SATISFACTION
7. COMMUNICATION – **Any time – any place**
8. GLOBALIZATION – **Bringing the world closer together**

**Pitfalls**

1. **24X7** support facility.
2. Online payment.

## **Requirement Analysis**

## **Functional and Non-Functional**

Functional requirements simplify defines what a system is supposed to do. It specifies a particular behavior of function of the system when certain conditions are met.

Non-Functional requirements define how a system is supposed to be. It specifies criteria that can be used to judge the operation of a system, rather than specific behaviors.

In the given table, the functional and non-functional requirements are identified and then they are prioritized with MoSCoW prioritization.

Index used in the table below are:

**F**=Functional requirements

**NF**=Non-Functional requirements

**M**=Must Have

**S**=Should Have

**C**=Could Have

## **Functional requirements:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Functional** | **Title** | **MoSCoW** | **Description** |
| **F(R1)** | Registration | **M** | User must registration to system. |
| **F(R2)** | Login | **M** | User must Login to System for booking packages. |
| **F(R3)** | Tour package option | **S** | Able to select different package option. |
| **F(R4)** | Package booking | **M** | Able to book package. |
| **F(R5)** | Routes information | **S** | To get the destination information. |
| **F(R6)** | Search package | **M** | Able to search package. |
| **F(R7)** | Generate and print receipt | **M** | Receipt must be generated after booking. |
| **F(R8)** | Change password | **M** | User can change password accordingly. |
| **F(R9)** | Online payment | **W** | Online transaction cannot be made. |
| **F(R10)** | Profile add, update, delete, view | **M** | Able to add, update, delete personal information. |
| **F(R11)** | Show package booked details | **M** | Can see package booking details |
| **F(R12)** | Package add, update, delete, view | **M** | Admin is able to add, delete, update, view package data. |
| **F(R13)** | Viewers Feedback | **M** | Can check viewers feedback |
| **F(R14)** | Booking History | **M** | Admin can check the booking history. |

## **Non-Functional Requirements:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Non - Functional** | **Requirements** | **MoSCoW** | **Description** |
| **NF 1** | Performance | **M** | System should be performed smoothly |
| **NF 2** | Usability | **M** | System should navigation easily and easy to use. |
| **NF 3** | Reliability | **C** | Precise and accurate information should give to system |
| **NF 4** | Interface | **C** | User interface should be easy and friendly. |
| **NF 5** | Portability | **S** | System should be run in every platform. |
| **NF 6** | Scalability | **C** | System should store more and more information easily |
| **NF 7** | Security | **M** | Make sure that System must secure and protected from unauthorized access. |
| **NF 8** | Website Backup | **S** | System should backup accordingly. |
| **NF 9** | Maintainability | **C** | Maintenance should be done in regular basic |
| **NF 10** | Privacy | **M** | System should keep user information private |

## **NLA**

Natural language analysis is defined by the Consortium on Cognitive Science instruction as “The use of ability of systems to process sentences in a natural language such as English, rather than in a specialized artificial computer language such as C++.”

## **Initial Class Diagram**

A class diagram is explanation/report of relationship in between different multiple classes. It is a static diagram that displays a static view of system and shows a collection of interface, classes and association. It also defines the attributes and operations of class.

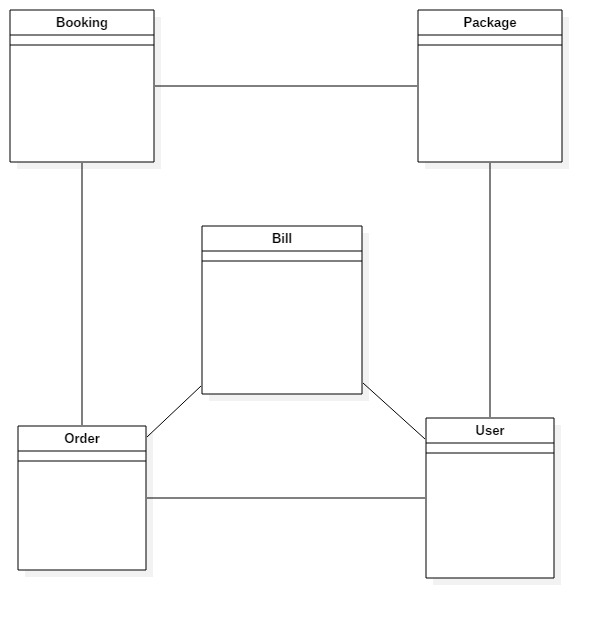


Figure 1Initial class diagram

**Justification**

* In the class diagram, classes are represented with boxes that contain three compartments:
* The top compartment contains the name of the class, printed in bold and centered, and the first letter is capitalized.
* The middle compartment contains the attributes of the class. They are left-aligned and the first letter is lowercase.
* The bottom compartment contains the operations, the class can execute. They are also left-aligned and the first letter is lowercase
* Class diagram are simple and fast to read. Forward and reverse engineering.
* To model the static view of an application, giving a sense of orientation.
* They provide detailed insight into the structure of our systems, describing responsibilities of a system.

## **Use-Case Diagram**

Use-case is a software and system engineering term that define how user can interact with system in order to accomplish a particular operation. A use case acts as a software modelling technique that implies function to be implemented. (Techopedia, 2018)

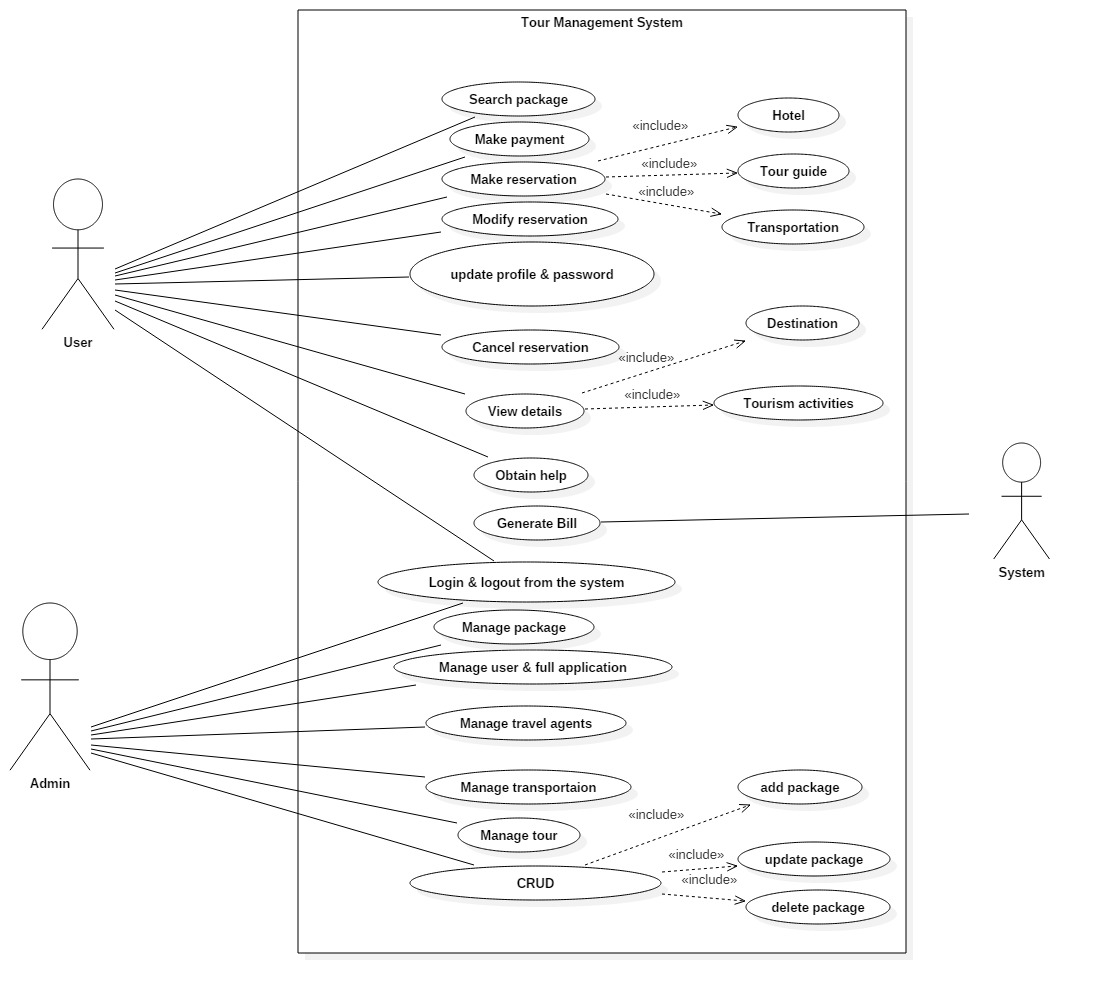
****

Figure 2Usecase Diagram

**Justification**

* For gathering the system requirements and actors and to collect the functional requirements of a system.
* To define the set of actions that system can execute by collaborating with one or more external users.
* To specify the events of a system and their flows.
* In above given diagram, three different actors i.e. (User, Admin, System) has been displayed. User is unregistering customer for the system, second one is admin who manages whole system and the third is the system.
* First actor i.e. user who can register into the system by filling up registration form become the registered customer. After registration, he/she can do all things that registered customer can do to the system.
* Second actor i.e. admin who is able to control and access whole system. Admin is able to manage, add, delete, update, view the customer and package information.
* Third actor i.e. system which helps to generate bill.

## **System Architecture**

It is known as the art of constructing and design. In the organization there are various types of architecture for the development of the system. Some of them are N-tier, Two-tier and three- tier architecture.

So, among them I have chosen the three- tier architecture.

Three-tier architecture is a client-server architecture in which the functional process logic, data access, computer data storage and user interface are developed and maintain independent modules on separate platform. The three tiers in this architecture are:

* Presentation tier:
* Application tier:
* Data tier:

I have decided to use three-tier architecture in my project because of the following reasons:

* Easy to apply object-oriented concept
* Easy to update data provider queries.
* Easy to maintain and understand large project and complex project.
* Database Security can be provided at application layer.

## **Methodology used**

It is a framework which is used for planning, structure and control the process of developing an information system. For my project development, I will use waterfall methodology. Since it is easy to manage because of the rigidity of model and sequential design approach.

In order to develop, organize and maintain the information system there are few steps to be taken that will lead the development process efficiently.



It is very simple methodology of software development. There are few sequential steps which are advanced after the accomplishment of one due to which there is no overlapping during the process. The steps involved in this methodology are:

* **Requirement gathering and analysis**

The first phase involves understanding what need to be design and what is its function, purpose etc.

* **System Design**

The requirement specifications from first phase are studied in this phase and system design is prepared. System Design helps in specifying hardware and system requirements and helps in defining overall system architecture.

* **Implementation**

 With inputs from system design, the system is first established in small programs called units, which are then integrated in the next phase. Each unit is developed and tested for its functionality which is referred to as Unit Testing.

* **Testing**

Testing is performed in order to avoid the problems. All the units developed in the implementation phase are integrated into a system after testing of each unit.

* **Deployment of System**

 Once the functional and non-functional testing is done, the product is deployed in the customer environment or released into the market.

* **Maintenance**

It involves modifications in the system to improve the performance. These modifications arise due to change that are requested by the customer. Client is provided with regular maintenance and support for the developed software.

# **Chapter 3 DESIGN**

## **Design Introduction**

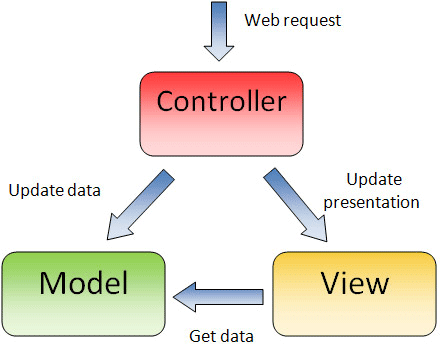
It is a detailed document that provides full information about a going to be developed product of process. Its main aims are to ensure that the development product must meet the users’ requirements. There are three different types of design phrase i.e. structural design, behavioral design and database design.

## **Design Pattern**

Among the various types of design pattern like Prototype pattern, factory pattern, Model View Controller, Singleton pattern etc. I have chosen the Model View Controller (MVC) pattern.

It is an architectural pattern widely used to develop user interfaces that divides an application into three different interconnected parts. This pattern is used to detached application’s concerns. In this pattern the model is known as the central component of the pattern which directly manages the data, logic and rules the application. A view can represent the information of chart diagram, bar chart etc. and the Controller accept the input that convert it to command for view and mode. Following are the reasons to select this pattern:

* Fast development process.
* Change/modification won’t affect the entire model system.
* Supports asynchronous technique.
* Supports multiple views.



**Controller**: It interacts with model and view.

**Model**: It takes data form the user.

**View**: Provides UI logic for the data.

## **Dynamic modelling**

It refers to represent the object interactions during runtime. A dynamic model represents the behavior of an object over time. It is used where the object's behavior is best described as a set of states that occur in a defined sequence.

## **Sequence Diagram**

A sequence diagram shows object interactions arranged in time sequence. It depicts the objects and classes involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario. Sequence diagrams are typically associated with use case realizations in the Logical View of the system under development. Sequence diagrams are sometimes called event diagrams or event scenarios.

## **Login Sequence Diagram**

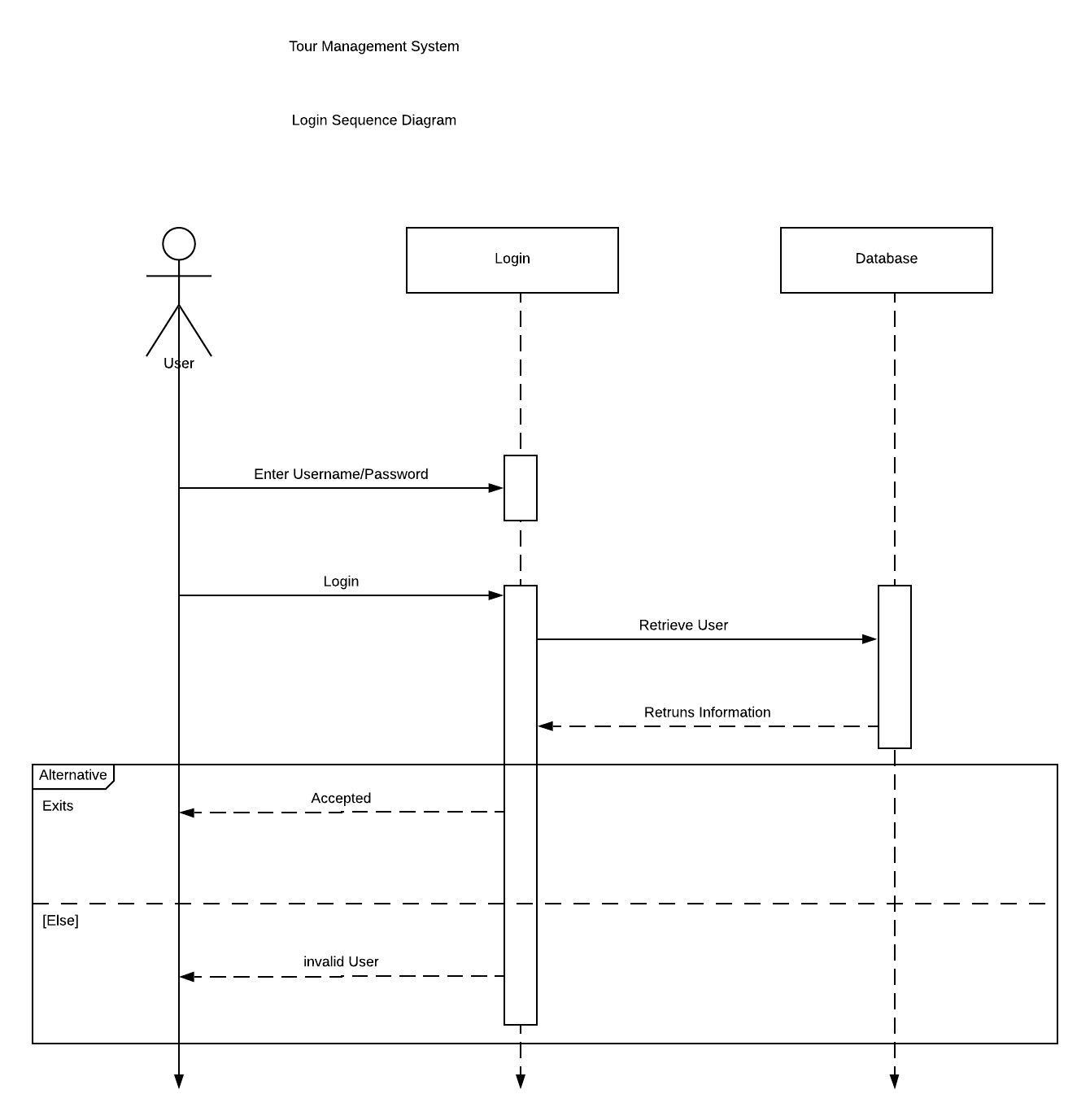


Figure 3Login sequence diagram.

**Justification**

In the above diagram, login sequence diagram has been displayed. User enters the username and password where the database checks the information and give authorization of login the system accordingly.

* The sequence diagram is used primarily to show the interactions between objects in the sequential order that those interactions occur.
* Easier to understand.
* UML specification is more sequence diagram centric.
* Allows reverse engineering.
* Help developers and business analyst to get a common understanding.

## **Admin Sequence Diagram**

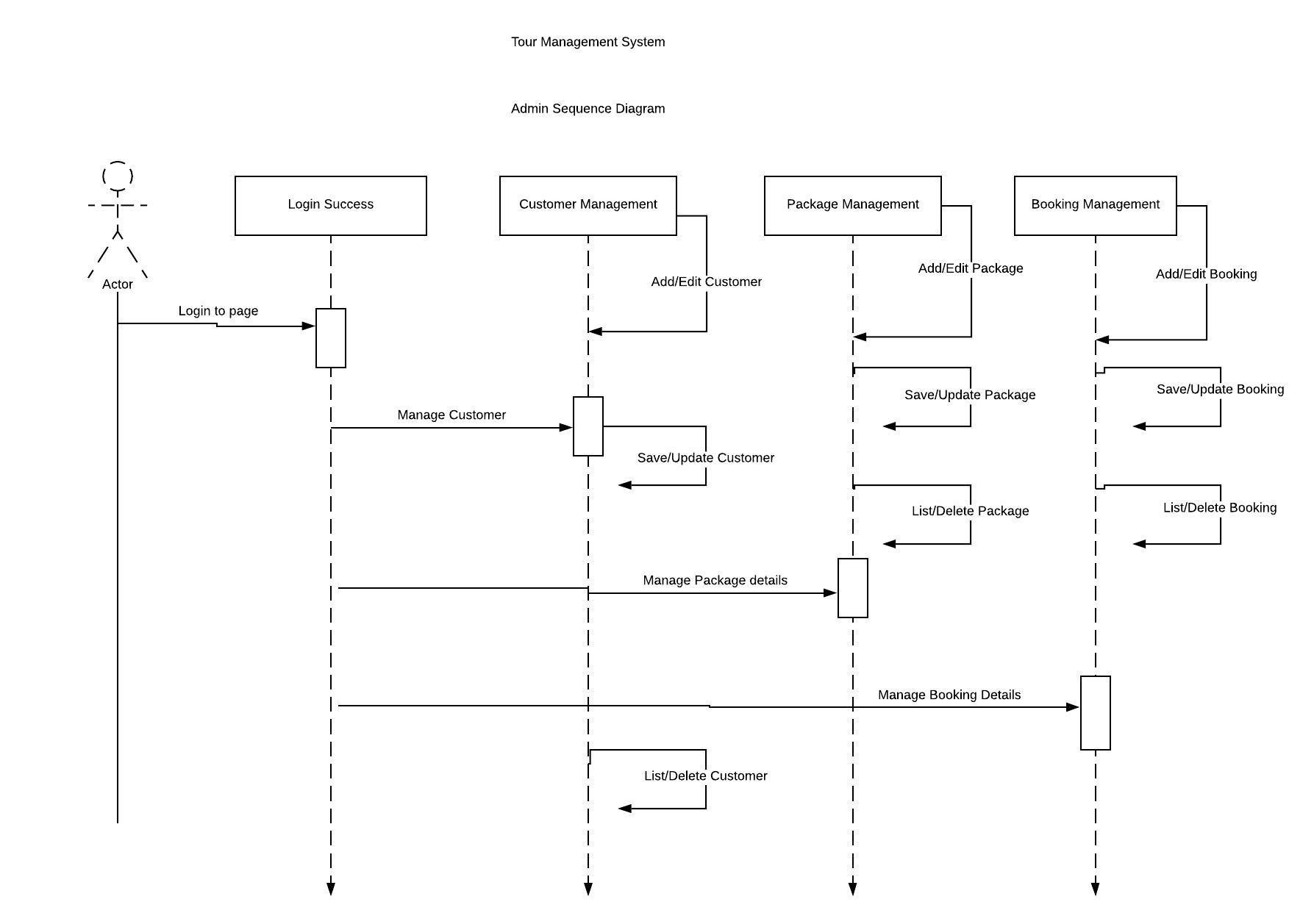
****

Figure 4Admin sequence diagram.

**Justification**

In the above sequence diagram, admin CRUD function has been displayed.

* The sequence diagram is used primarily to show the interactions between objects in the sequential order that those interactions occur.
* Easier to understand.
* UML specification is more sequence diagram centric.
* Allows reverse engineering.
* Help developers and business analyst to get a common understanding.

## **User Sequence Diagram**

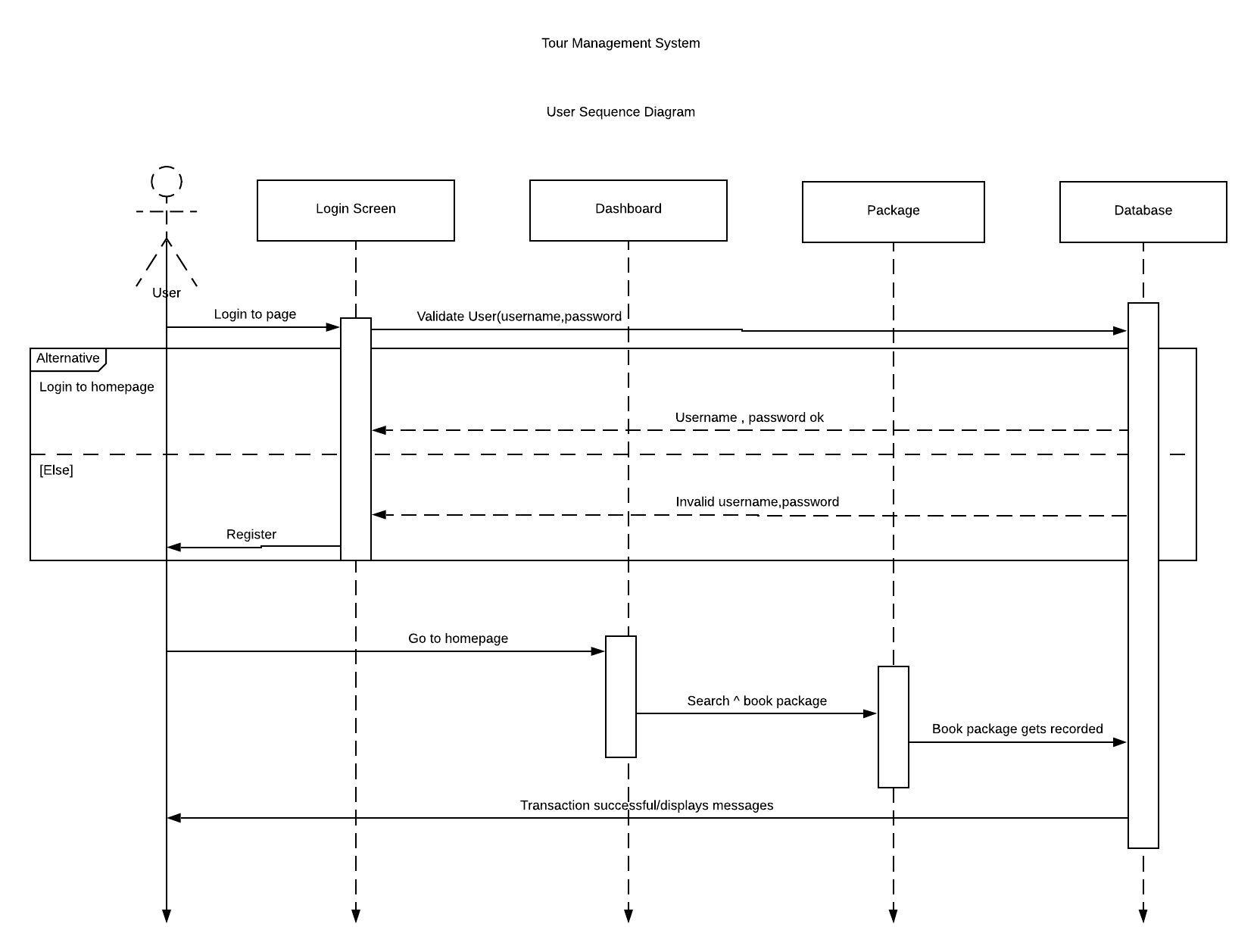


Figure 5User sequence diagram

**Justification**

The above sequence diagram shows the user booking activity. The user can login in to the system and able to select the package accordingly and book the package.

* The sequence diagram is used primarily to show the interactions between objects in the sequential order that those interactions occur.
* Easier to understand.
* UML specification is more sequence diagram centric.
* Allows reverse engineering.
* Help developers and business analyst to get a common understanding.

## **Activity Diagram**

It is a UML diagram that describes the dynamic aspect as well as advanced version of flow chart which show logic of flow of one activity to another activity of the system. The activity can be described as an operation of the system. The flow can be sequential, branched or concurrent. It deals with all type of flow control by using different elements like fork, join etc.

## **Admin Activity Diagram**

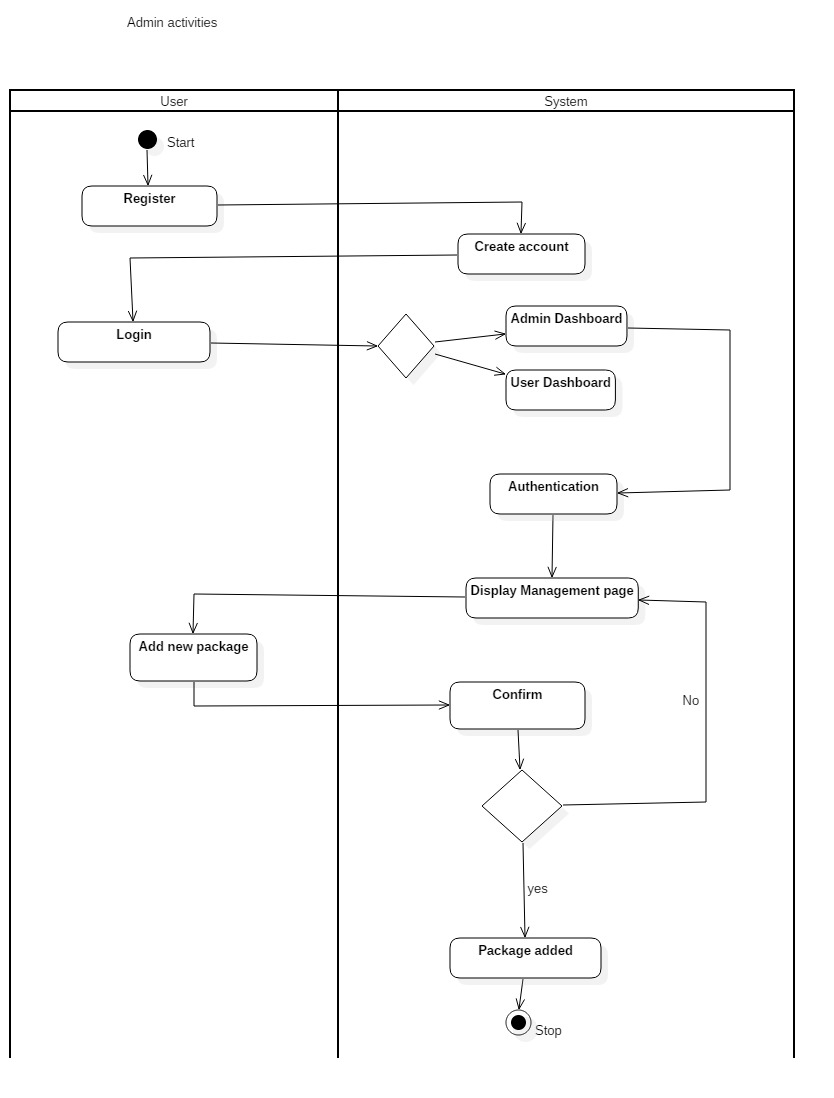


Figure 6Admin activity.

**Justification**

In the above given activity diagram, admin activities have been displayed. At first the user can register to create an account and login to the system. If the logged in user is valid as Admin then then the system will direct to Admin dashboard. In order to add new package in the system, the admin sends the command and system confirms the request. If the request is accepted, it creates the new package else it goes back to Admin dashboard.

* Activity diagram is basically a flow chart to represent the flow from one activity to another activity
* To describe the sequence from one activity to another.
* To describe the parallel, branched and concurrent flow of the system.
* The flow can be sequential, branched or concurrent. It deals with all type of flow control by using different elements like fork, join etc.
* Displays multiple conditions and actors within a work flow through the use of swim lanes.

## **Generating receipt activity diagram**

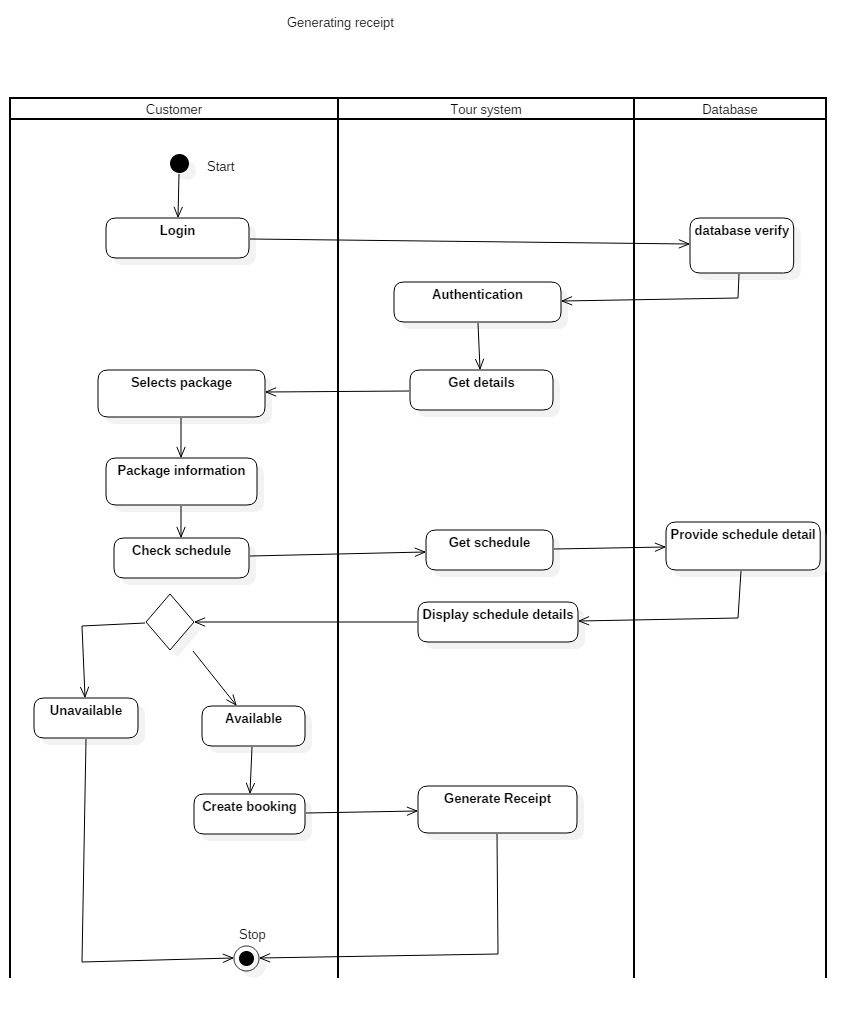


Figure 7Generating receipt.

**Justification**

In the above given activity diagram, generating receipt activity has been displayed. The user login to the system where database verifies the registered user and enters into the page and able to retrieve details. The users select the appropriate package, checks schedule for the availability through the database system, where the information will be displayed in the page. If the package if unavailable it cancels the booking and if the package schedule is available then it creates booking for the package and the system generates the receipt of the appropriate package.

* The activity can be described as an operation of the system.
* Activity diagram is basically a flow chart to represent the flow from one activity to another activity
* To describe the sequence from one activity to another.
* To describe the parallel, branched and concurrent flow of the system.
* The flow can be sequential, branched or concurrent. It deals with all type of flow control by using different elements like fork, join etc.
* Displays multiple conditions and actors within a work flow through the use of swim lanes.

## **Package reservation activity diagram**

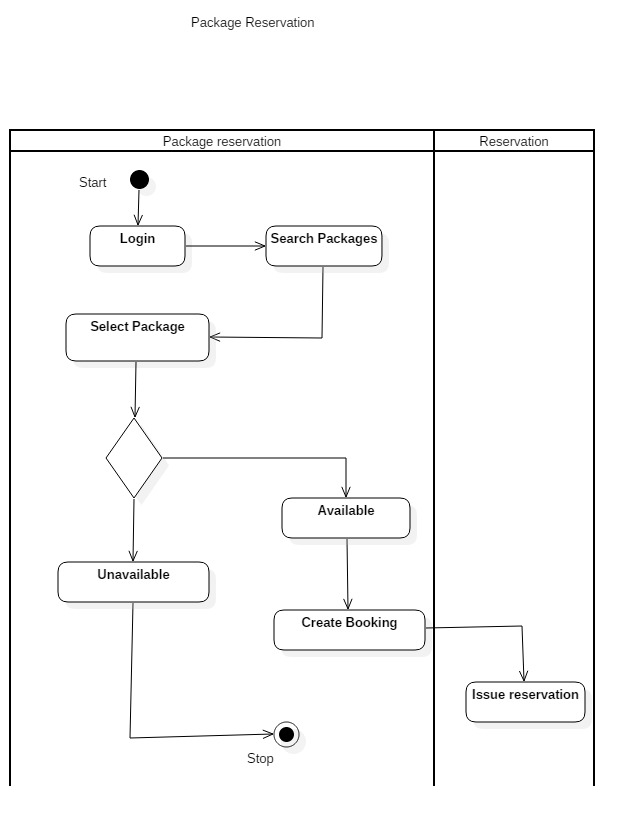


Figure 8Package reservation

**Justification**

In the above given activity diagram, package reservation activity has been displayed. User can login in to the system and search for packages accordingly. User can select the package and checks for availability. If the package is unavailable, the user gets the messages of unavailability and if the package is available, user can book the package and complete the reservation accordingly.

* The activity can be described as an operation of the system.
* Activity diagram is basically a flow chart to represent the flow from one activity to another activity
* To describe the sequence from one activity to another.
* To describe the parallel, branched and concurrent flow of the system.
* The flow can be sequential, branched or concurrent. It deals with all type of flow control by using different elements like fork, join etc.
* Displays multiple conditions and actors within a work flow through the use of swim lanes.

## **Structural Modelling**

**A** structural, or conceptual, model describes the structure of the objects that supports the business processes in an organization. During analysis, the structural model presents the logical organization of the objects without indicating how they are stored, created, or manipulated so that analysts can focus on the business, without being distracted by technical details. Later during design, the structural model is updated to reflect exactly how the objects will be stored in databases and files.

## **Class Diagram**

It is a type of static view of the system that describes the whole structure by showing classes, attributes, method or operation and relationship between classes of the system. (Visual-Paradigm, 2019)

It is used to illustrate and create a functional diagram of the system classes and serves as a system development resource within the software development life cycle.

## **Class Diagram**

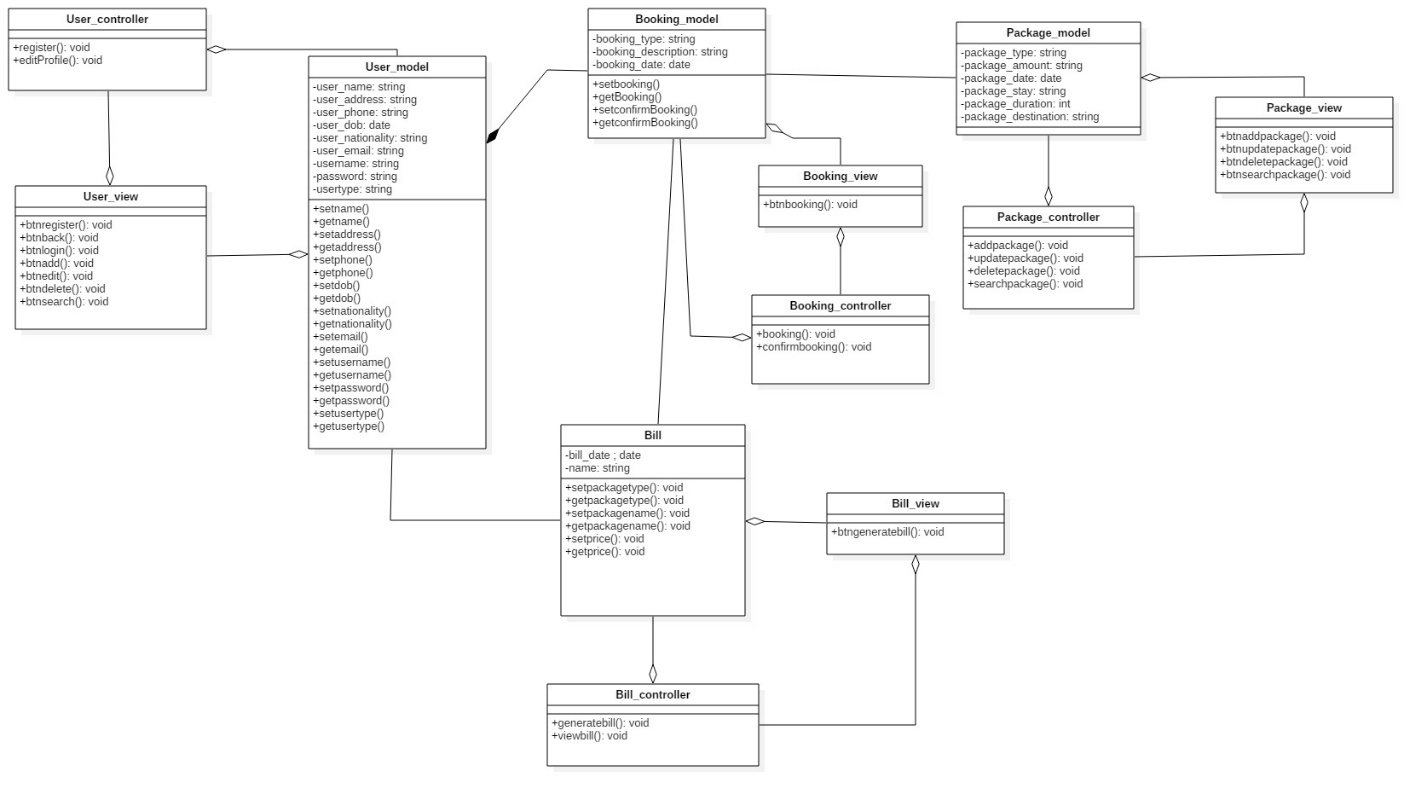


Figure 9Class diagram

**Justification**

* Displaying each class has each controller and the controller are directly linked with the database.
* Simple and easy to understand.
* It helps to model the static view of an application and describes the responsibilities of the system.
* It gives the detail insight into the structure of our systems but shows only the collaboration in between elements of the static view.
* For visualizing, describing, and documenting different aspects of a system but also for constructing executable code of the software application.
* Base for component and deployment diagrams.

## **Database Modelling**

It is the process of designing, developing and implementing the data for proposed system. Its main aim is to produce a logical and physical model for any system. Data modelling is the first step in the process of database design. This step is sometimes considered to be a high-level and abstract design phase, also referred to as conceptual design.

## **Entity Relationship Diagram**

An Entity Relationship Diagram also called as ERD is a type of flowchart that illustrates how entities relate with each other within system. It is a graphical representation of entities and their relationships to each other, typically used in computing in regard to the organization of data within databases or information system.

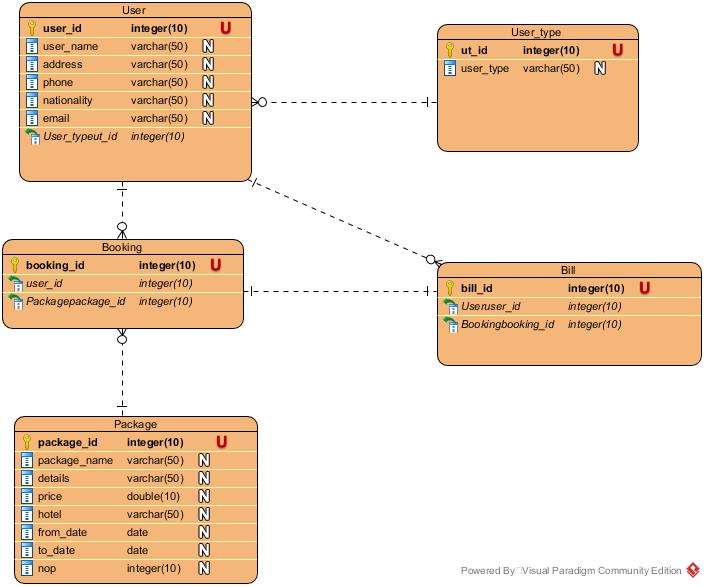


Figure 10ER diagram

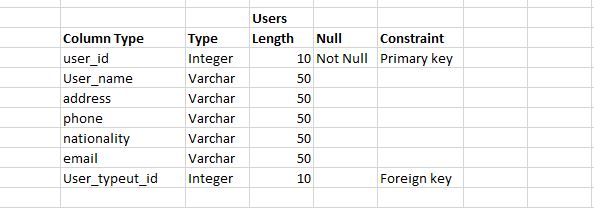
**Justification**

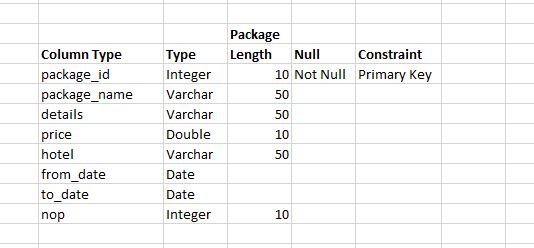
* It helps to find out the mistakes and flaws of the design, and able to correct before executing the changes in database.
* Easily the entities can be located by visualizing the database schema, attributes can be viewed and helps to identify the relationship.
* Better visual representation and simple if we know relationship between entities and attributes.
* An effective communication tool for database designer.
* Conversion from E-R diagram to a network or hierarchical data model can· easily be accomplished.
* Clear understanding of the data structure and minimizes redundancy and other problems

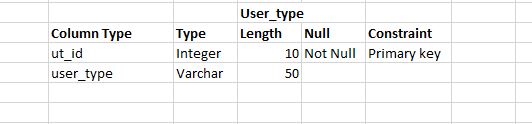
## **Data Dictionary**

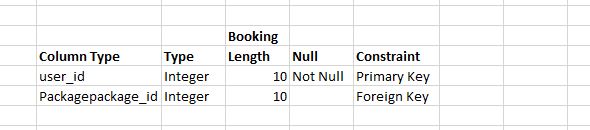
A set of information describing the contents, format, and structure of a database and the relationship between its elements, used to control access to and manipulation of the database.

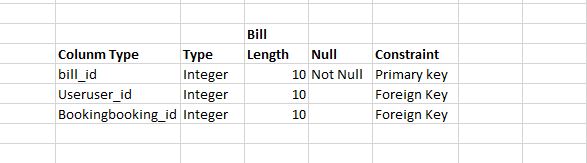
## **Meta – Data**











## **UI Modelling**

User interface modeling is a development technique used by computer application programmers. Today's user interfaces (UIs) are complex software components, which play an essential role in the usability of an application.

## **Registration**



Figure 11Registration

## **Login Form**



Figure 12Login

## **Home Page**



Figure 13Homepage

## **Gallery**

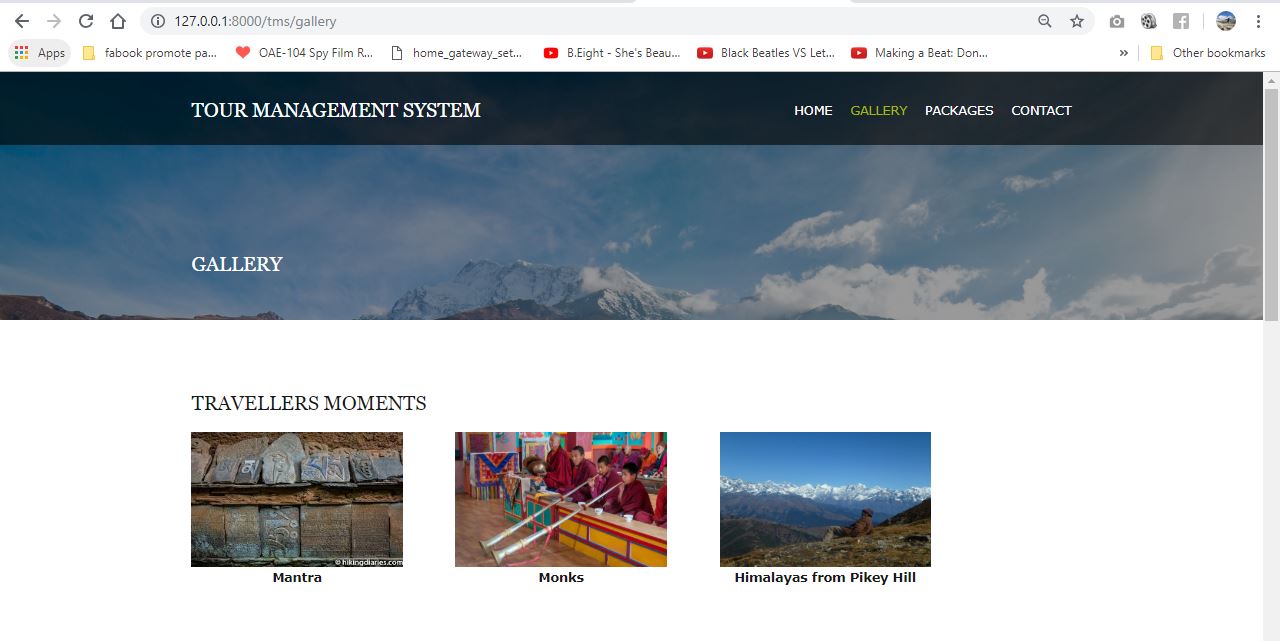


Figure 14Gallery

## **Package**

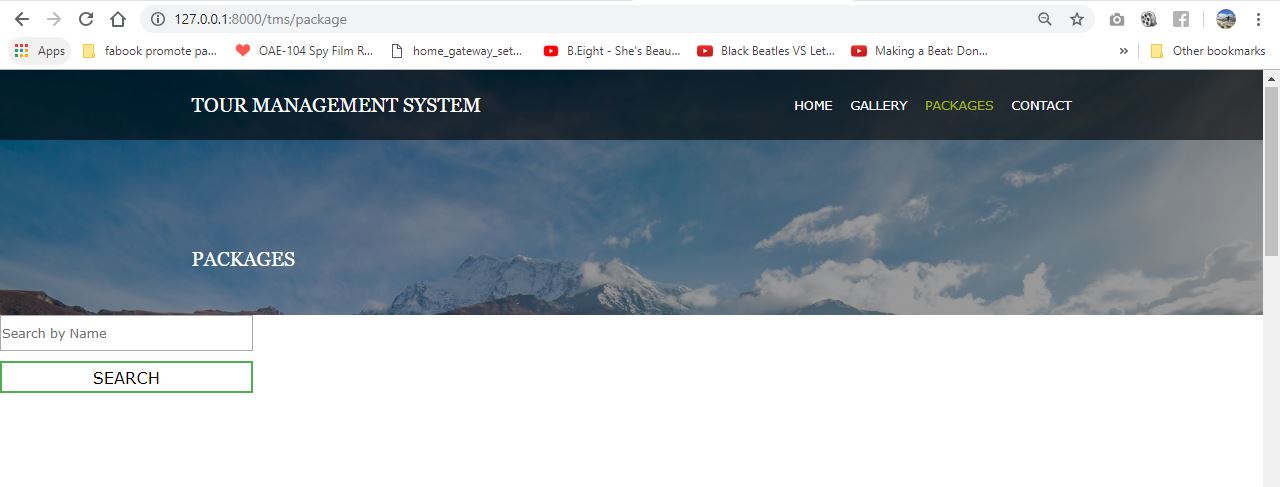


Figure 15Package

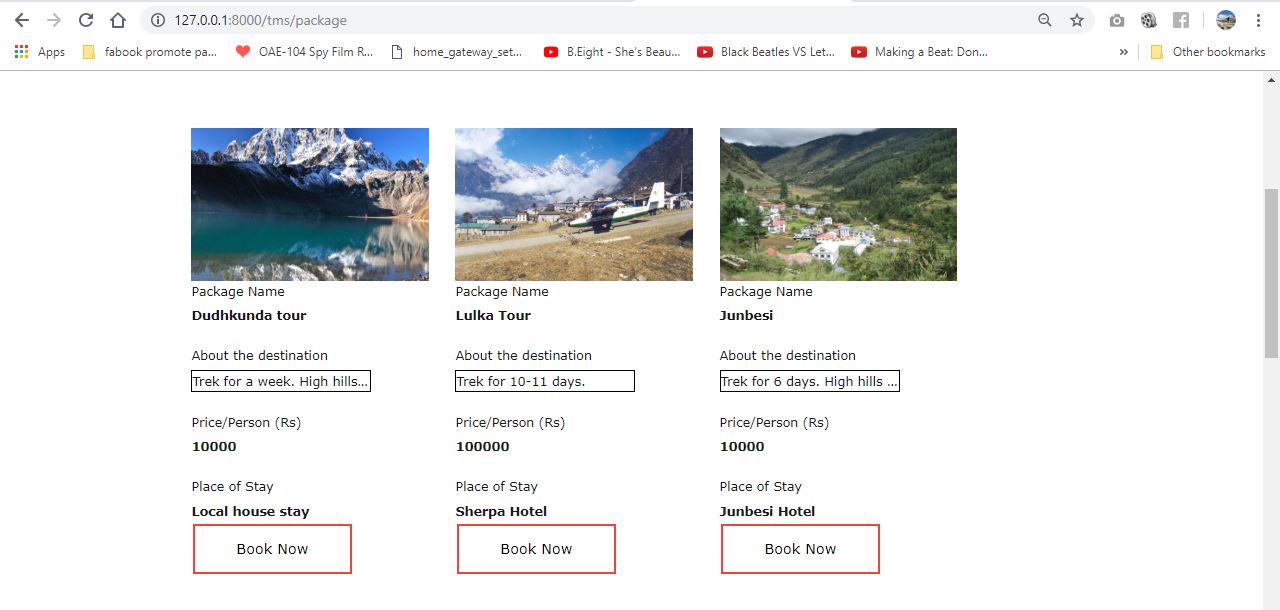


Figure 16Package details

## **Booking**

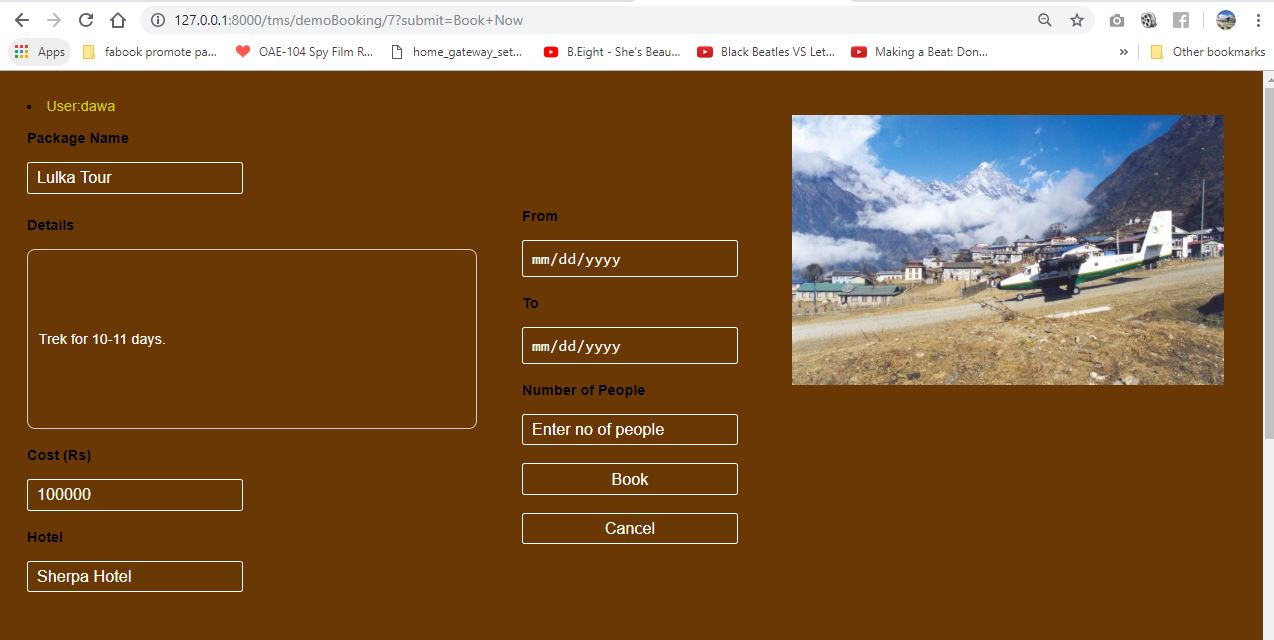


Figure 17Book package

## **Contact**



Figure 18Comment box

## **Admin Dashboard**

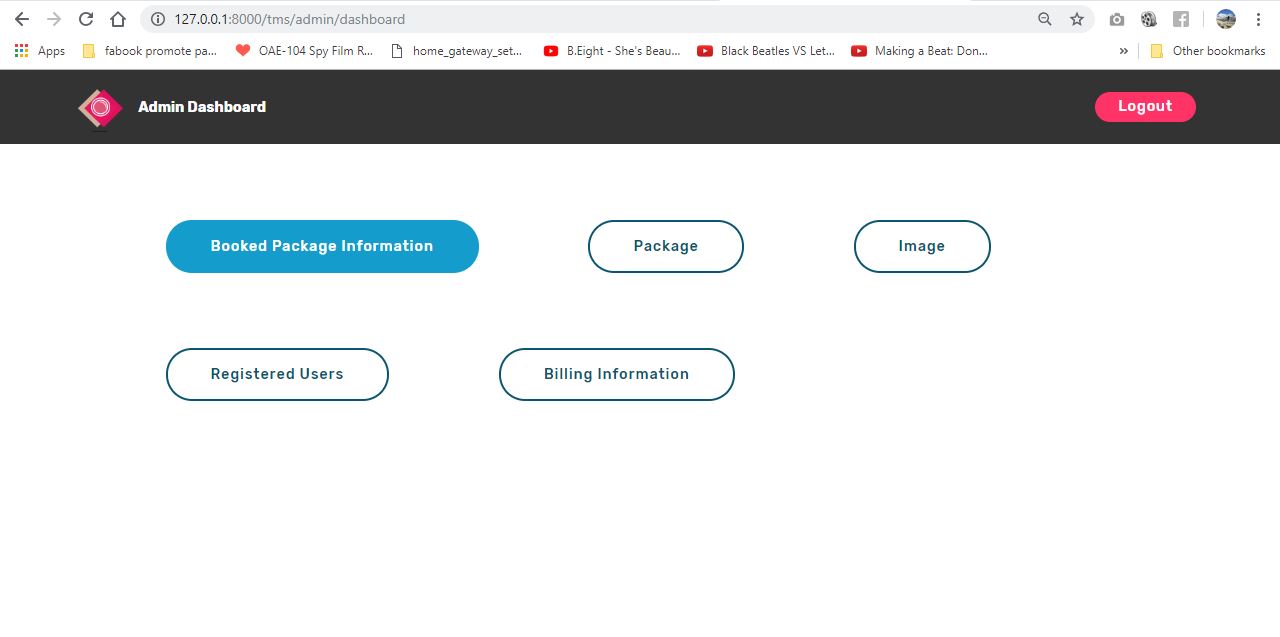


Figure 19Admin page

## **Package Booked Information**

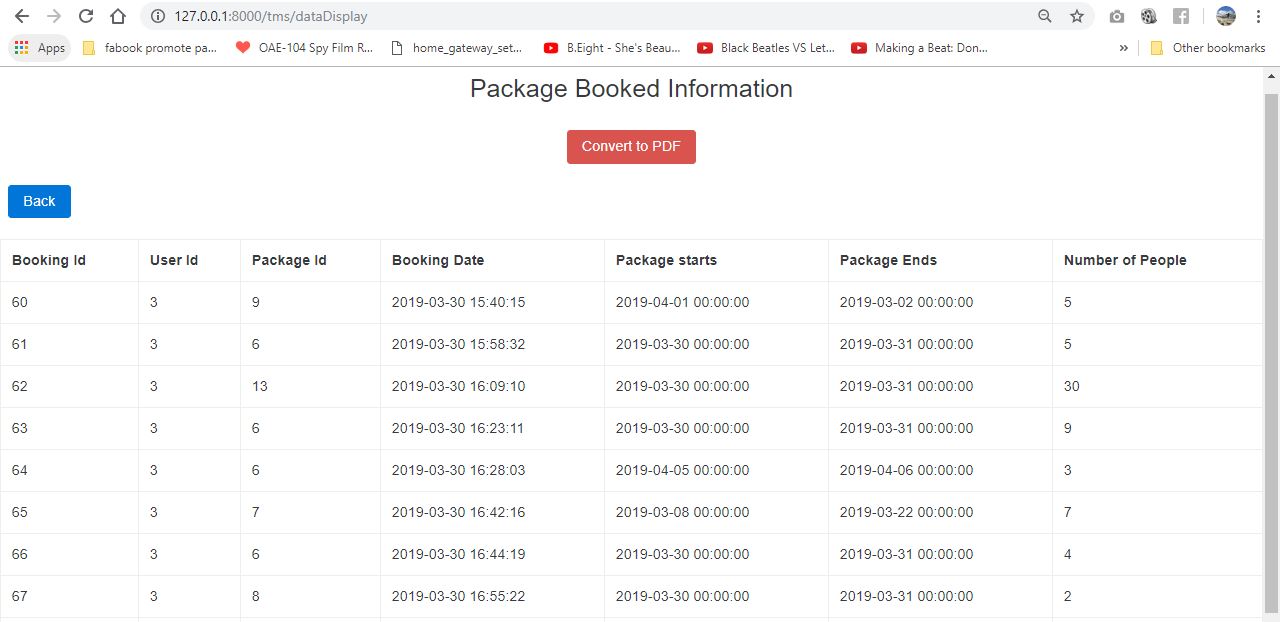


Figure 20Admin package booked information.

## **Admin package CRUD**

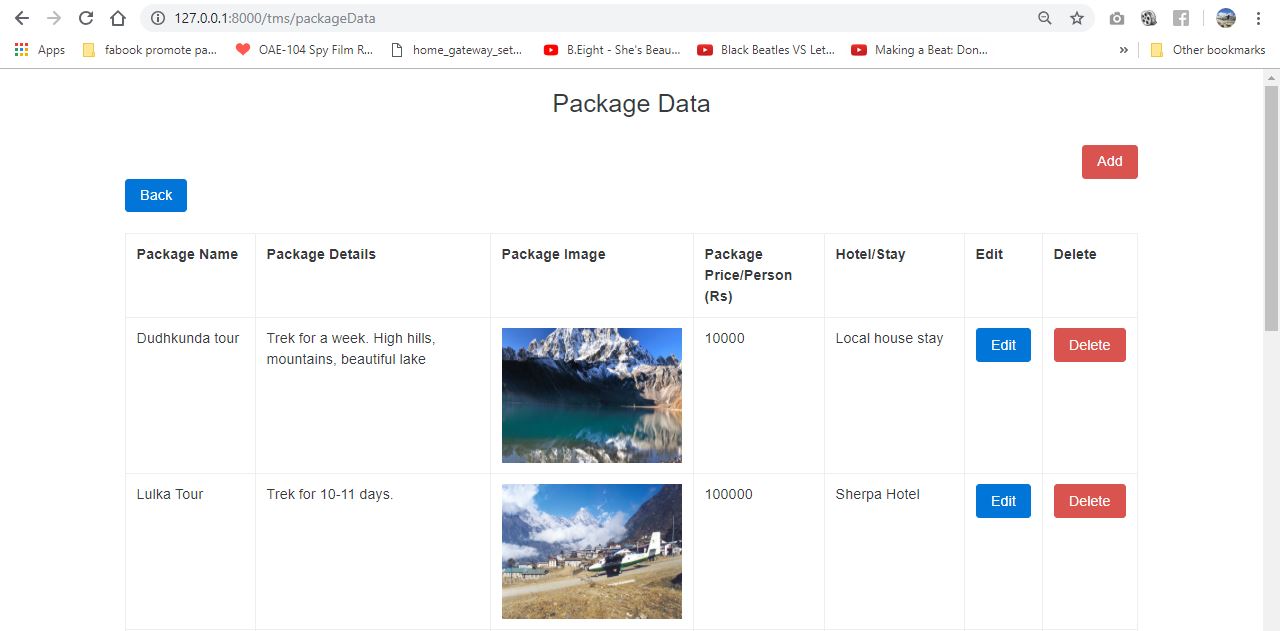


Figure 21Admin Package CRUD

## **Admin Gallery Function**

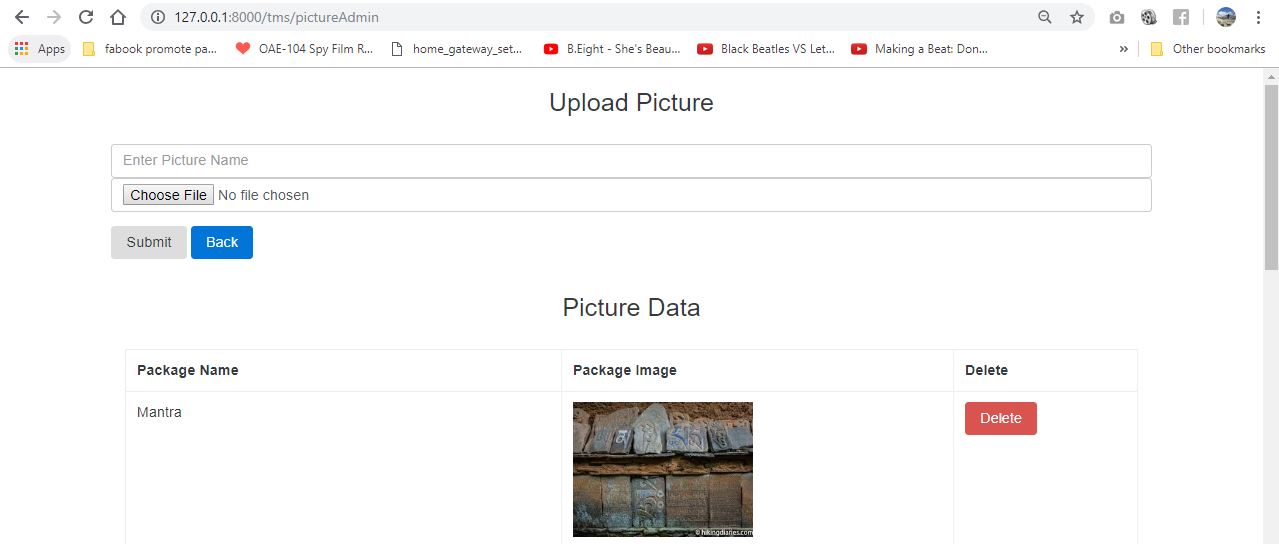
****

Figure 22Admin Gallery Function.

## **User Information**

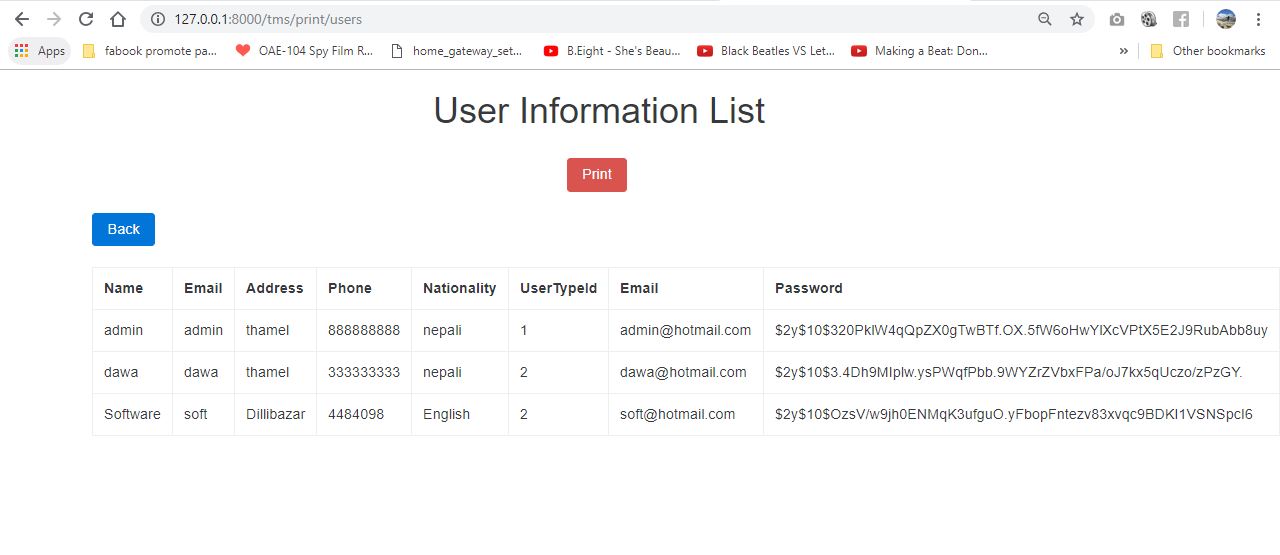
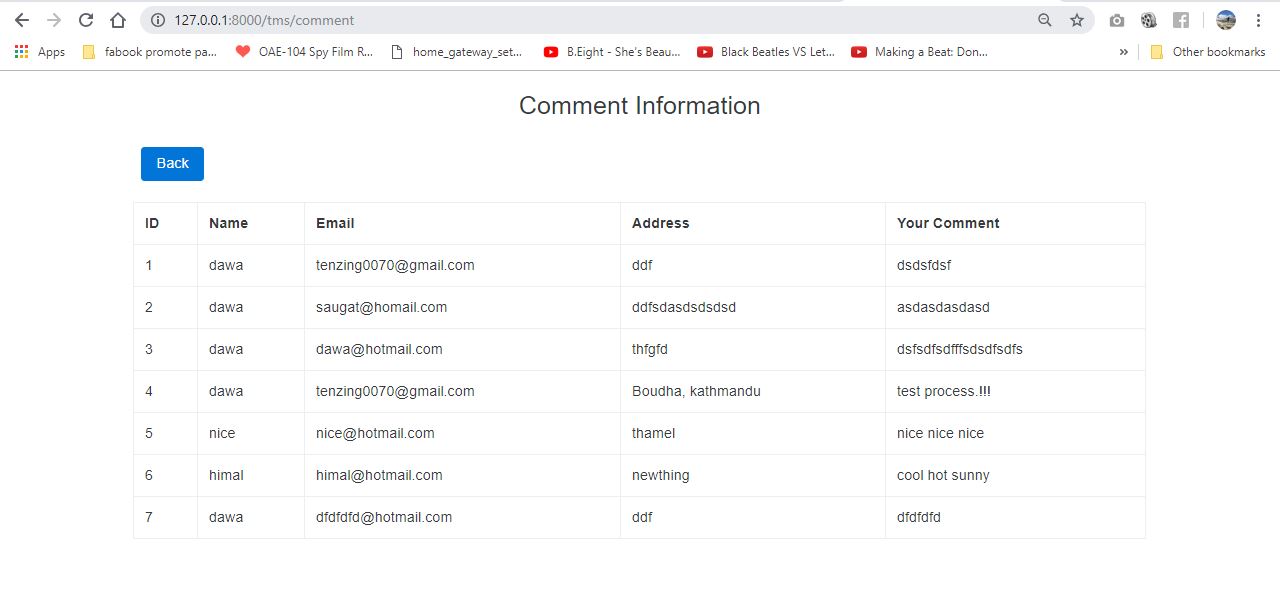


Figure 23User information.

## **User Comment**



# **Chapter 4 IMPLEMENTATION**

## **Introduction**

It is the phase after the design where the plan of project will be started to work according to those plans. All design of the project is now change in coding. The planning of Accommodation for all project will be coded using PHP programming language.

**PHP** stands for Hypertext Preprocessor (no, the acronym doesn't follow the name). It's an open source, server-side, scripting language **used** for the development of web applications. By scripting language, we mean a program that is script-based (lines of code) written for the automation of tasks.

The tools used to implement the project is sublime text, a proletary cross platform source code editor which supports many programming languages and is free software licenses.

Framework used for this project is Laravel. **Laravel** is a free, open-source **PHP** web **framework**, created by Taylor Otwell and intended for the development of web applications following the model–view–controller (MVC) architectural pattern and based on Symfony.

Laravel is a web application framework with expressive, elegant syntax. Laravel is accessible, yet powerful, providing powerful tools needed for large, robust applications.

# **Chapter 5 TESTING**

## **Introduction**

Testing is a process of executing a program with the aim of finding error. To make our software perform well it should be error free. If testing is done successfully it will remove all the errors from the software.

## **Principles of Testing**

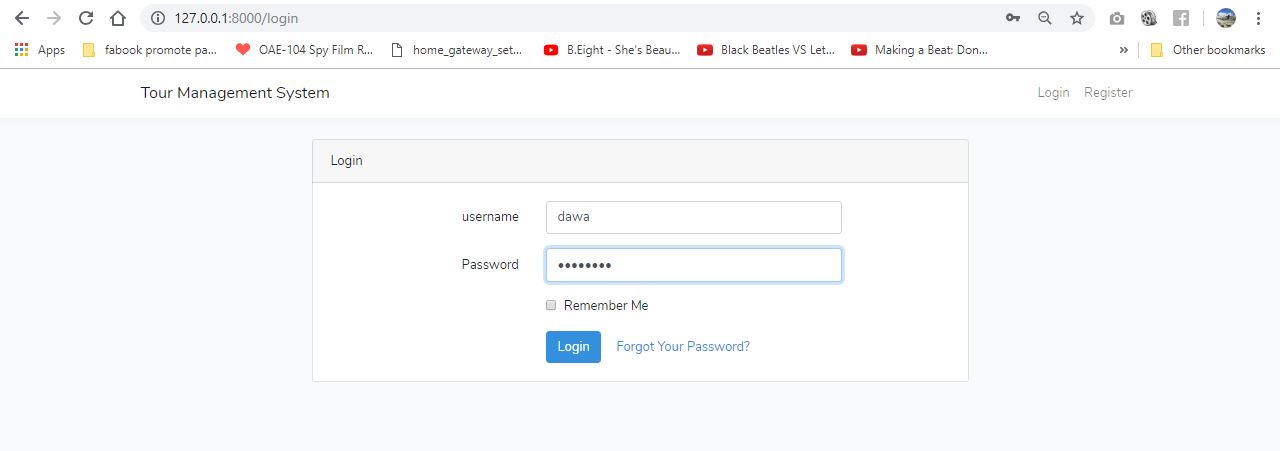
* All the test should meet the customer requirements.
* To make our software testing should be performed by third party.
* Exhaustive testing is not possible. As we need the optimal amount of testing based on the risk assessment of the application.
* All the test to be conducted should be planned before implementing it.
* It follows pareto rule (80/20 rule) which states that 80% of errors comes from 20% of the program components.
* Starts test with small parts and extend it to large parts.

## **Black box Testing**

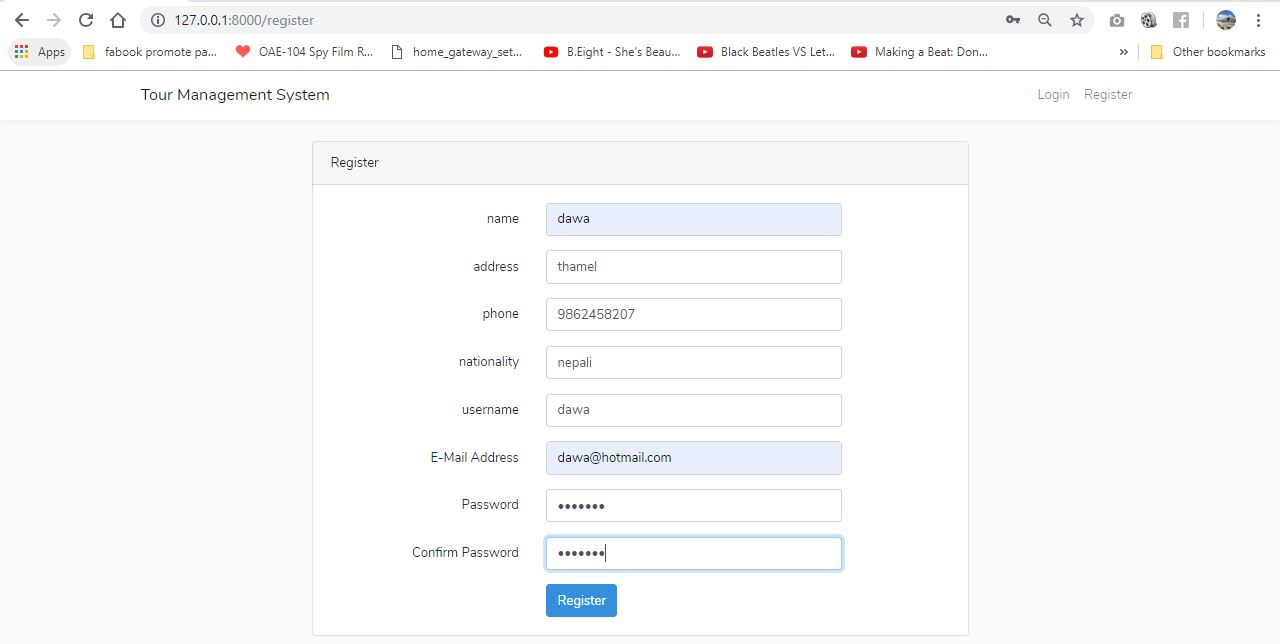
It is a [software testing method](http://softwaretestingfundamentals.com/software-testing-methods/) in which the internal structure/design/implementation of the item being tested is not known to the tester. These tests can be functional or non-functional, though usually functional.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **S.N No.** | **Testcase** | **Expected result** | **Actual result** | **Pass/Fail** | **Action** | **Conclusion** |

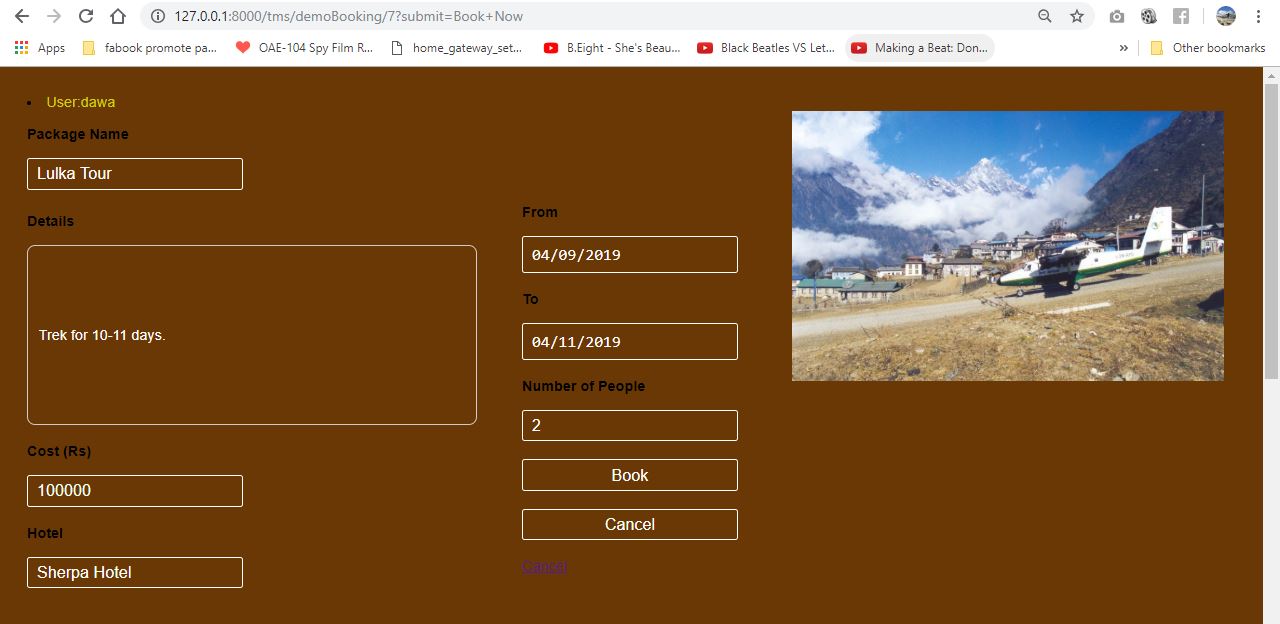
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 1 | Login | By entering username and password, user can access the system | System accessed | Pass | No action | Pass |



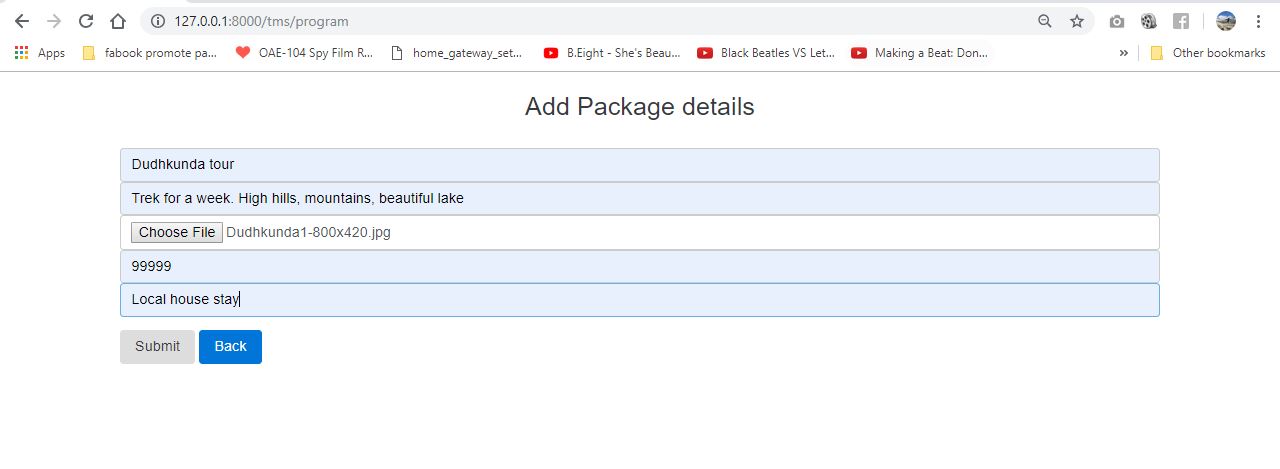
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **S.N No.** | **Testcase** | **Expected result** | **Actual result** | **Pass/Fail** | **Action** | **Conclusion** |
| 2 | Register | User can redirect to login after filling details on register page | Login form viewed | Pass | No action | Pass |



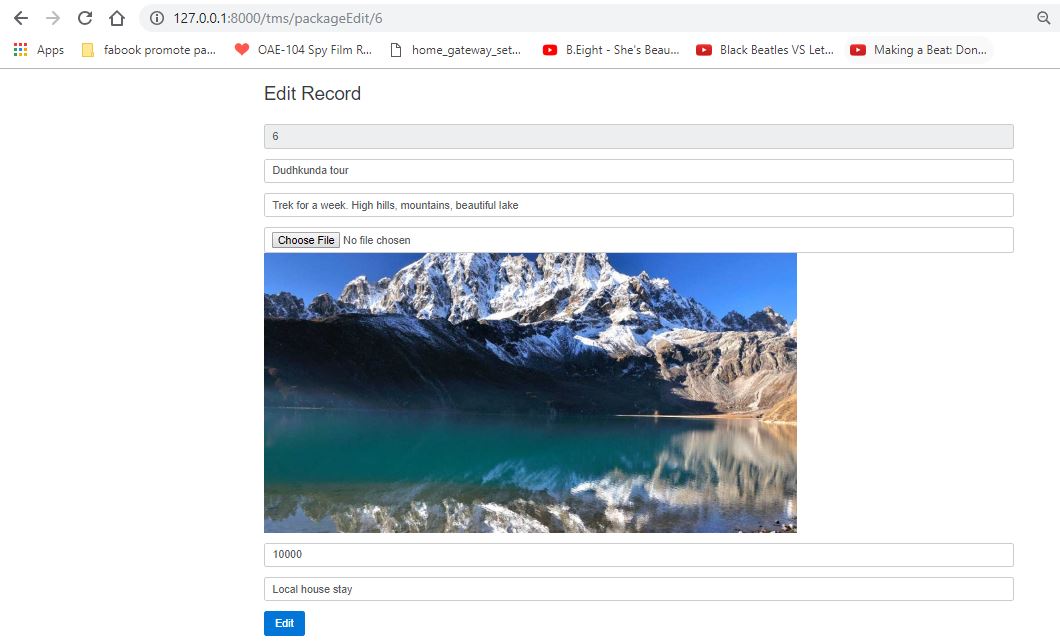
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **S.N No.** | **Testcase** | **Expected result** | **Actual result** | **Pass/Fail** | **Action** | **Conclusion** |
| 3 | Book Package | User can book the package | Successfully booked | Pass | No action | Pass |



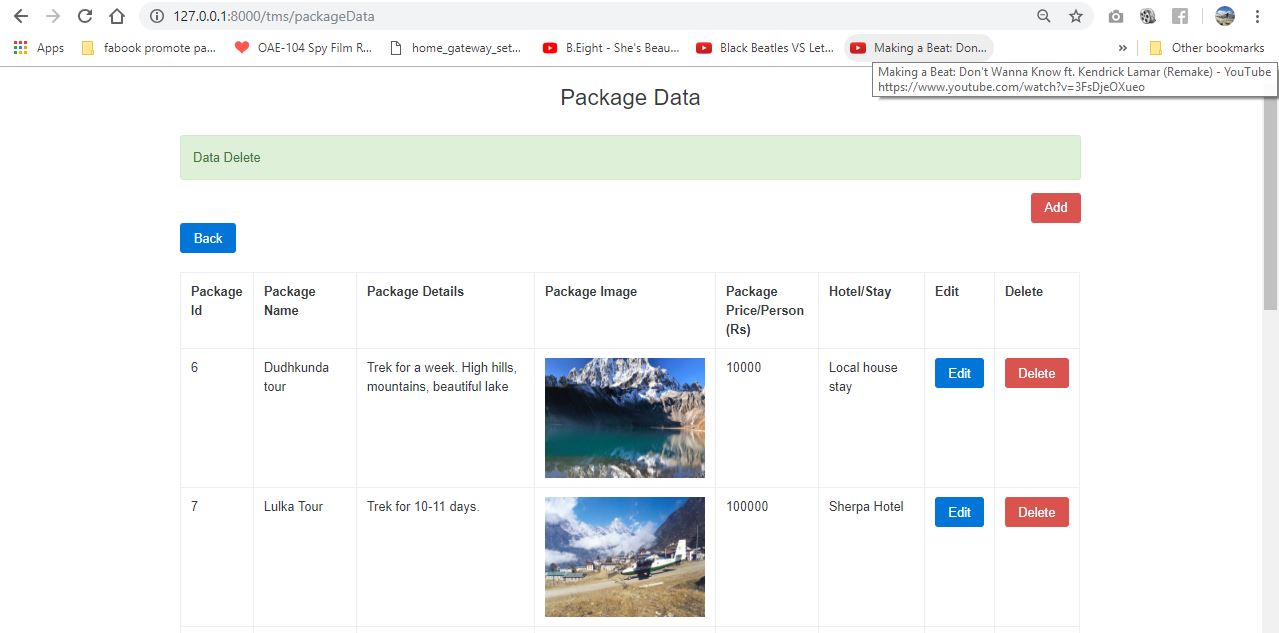
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **S.N No.** | **Testcase** | **Expected result** | **Actual result** | **Pass/Fail** | **Action** | **Conclusion** |
| 4 | Add Package | Package will be added in database | Package added in database | Pass | No action | Pass |



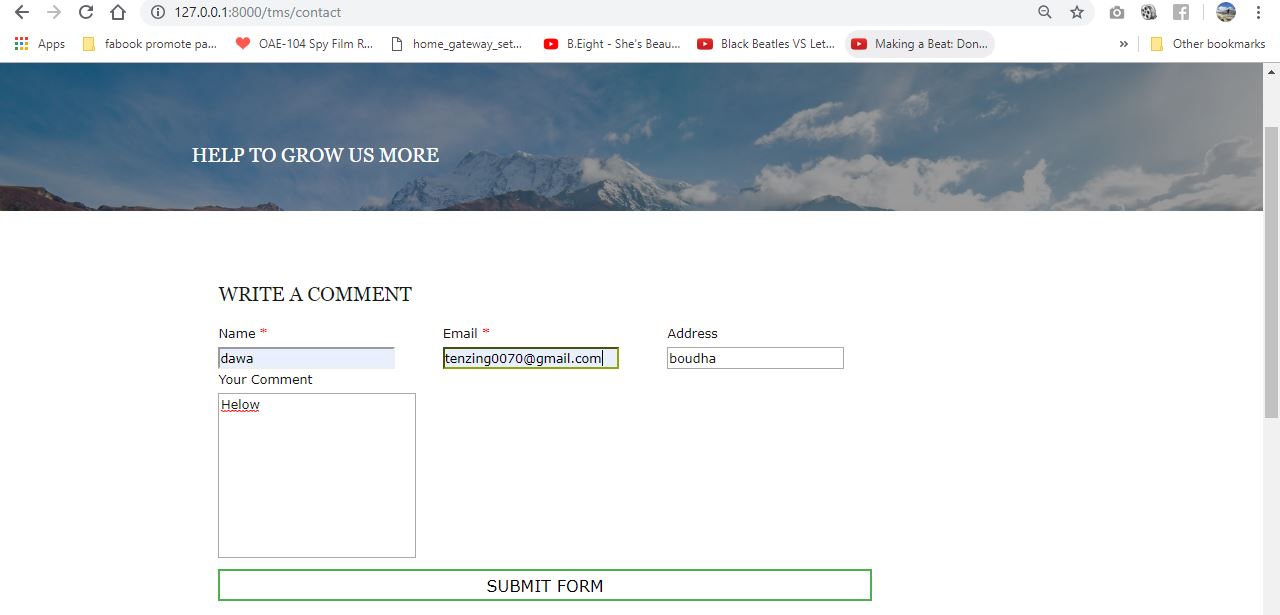
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **S.N No.** | **Testcase** | **Expected result** | **Actual result** | **Pass/Fail** | **Action** | **Conclusion** |
| 5 | Edit Package | Package should be Edited in database | Package edited in database | Pass | No action | Pass |



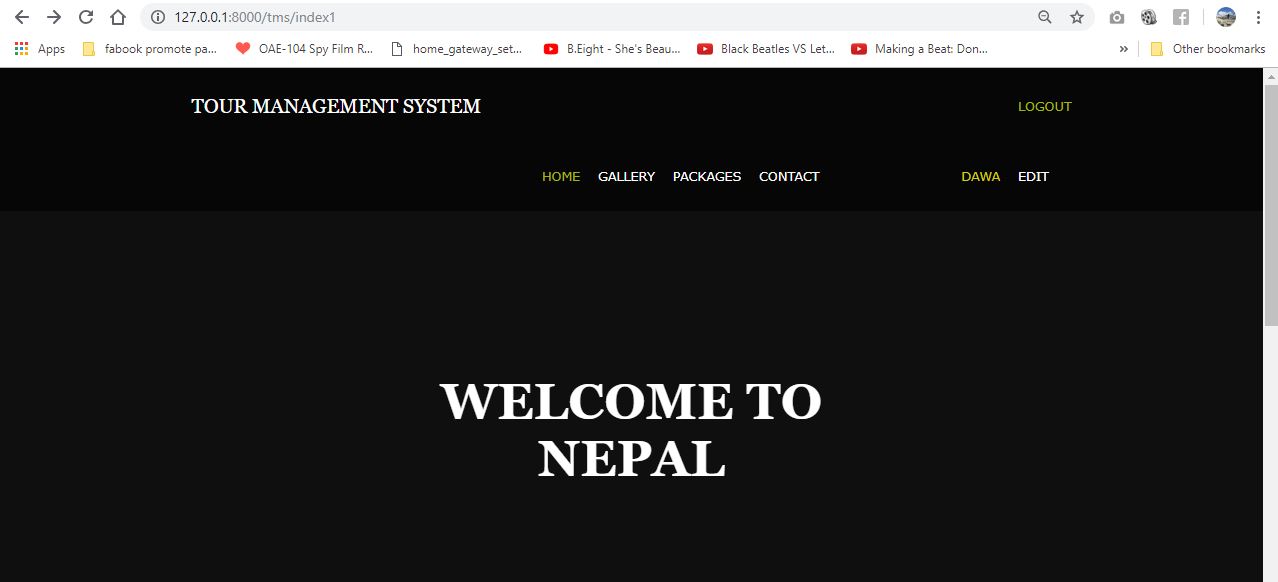
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **S.N No.** | **Testcase** | **Expected result** | **Actual result** | **Pass/Fail** | **Action** | **Conclusion** |
| 6 | Delete Package | Package should be deleted from database. | Package deleted from database | Pass | No action | Pass |



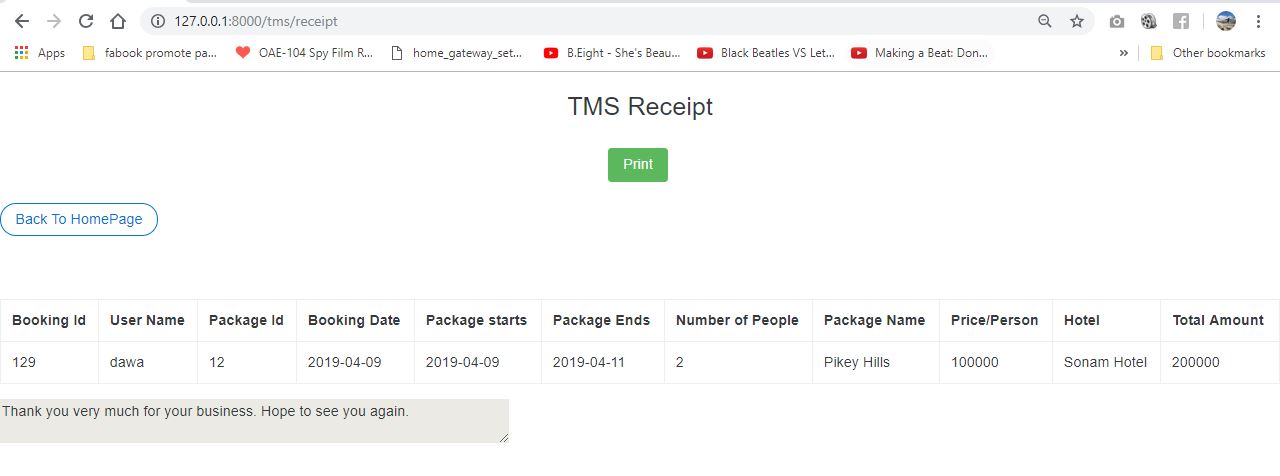
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **S.N No.** | **Testcase** | **Expected result** | **Actual result** | **Pass/Fail** | **Action** | **Conclusion** |
| 7 | Add Comments | Comments should be added to database | Display error message | Fail | Fix error message | Fail |



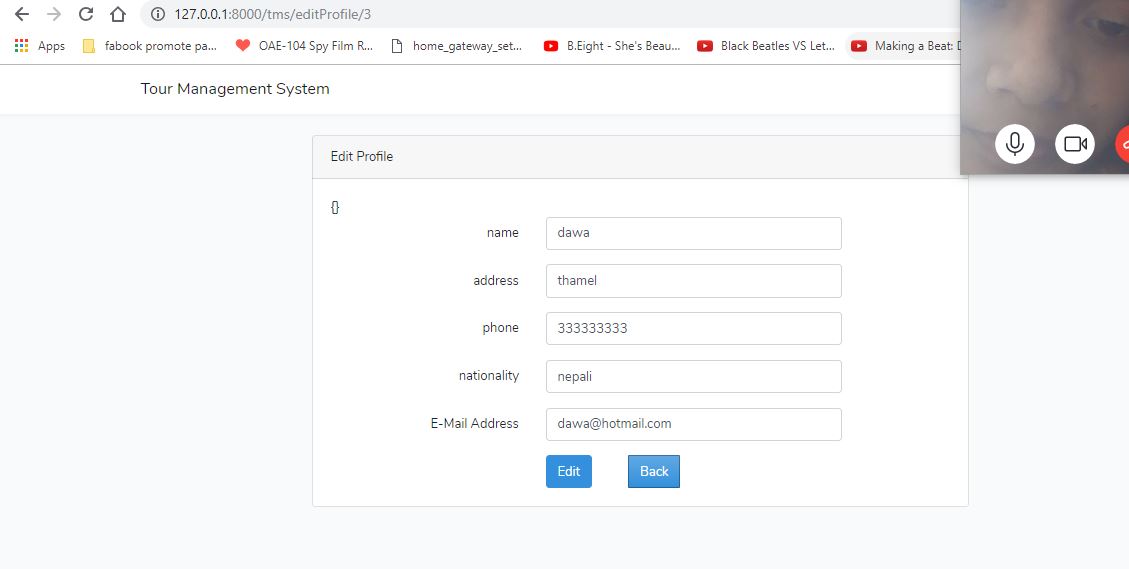
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **S.N No.** | **Testcase** | **Expected result** | **Actual result** | **Pass/Fail** | **Action** | **Conclusion** |
| 8 | Logout | User should be logout from the system | Successfully logout | Pass | No action | Pass |



|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **S.N No.** | **Testcase** | **Expected result** | **Actual result** | **Pass/Fail** | **Action** | **Conclusion** |
| 9 | Receipt Generate | User should get receipt after booking package | Receipt successfully received | Pass | No action | Pass |



|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **S.N No.** | **Testcase** | **Expected result** | **Actual result** | **Pass/Fail** | **Action** | **Conclusion** |
| 10 | Edit Profile | User should be able to edit profile. | Profile successfully edited | Pass | No action | Pass |



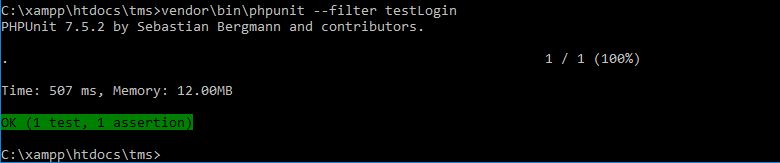
## **Unit Testing**

It is a level of software testing where individual units/ components of a software are tested. A unit is the smallest testable part of any software. It usually has one or a few inputs and usually a single output. In procedural programming, a unit may be an individual program, function, procedure, etc.

1. **Login test**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Test Case No.** | **Testcase** | **Purpose of the test** | **Test Data** | **Function Name** | **Expected result** | **Actual result** | **Outcome as expected** |
| 1 | Login Test | To login the system | Username=”john”  Password=”john1234” | testLogin | User should login | User Logged in | Expected result matched with actual result. |

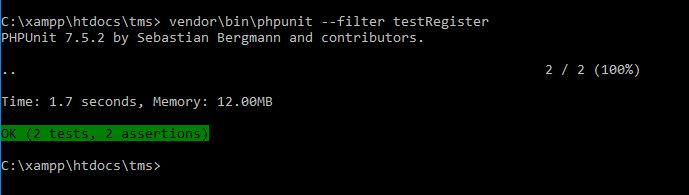
****

****

1. **Registration**

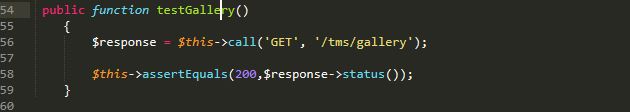
|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Test Case No.** | **Testcase** | **Purpose of the test** | **Test Data** | **Function Name** | **Expected result** | **Actual result** | **Outcome as expected** |
| 2 | Registration | To register in database | name=john  address=newyork  phone=999999999  nationality=nepali  username=john  [email=john@hotmail.com](mailto:email=john@hotmail.com)  password=john1234 | testRegister | User should be registered. | User is registered. | Expected result matched with actual result. |

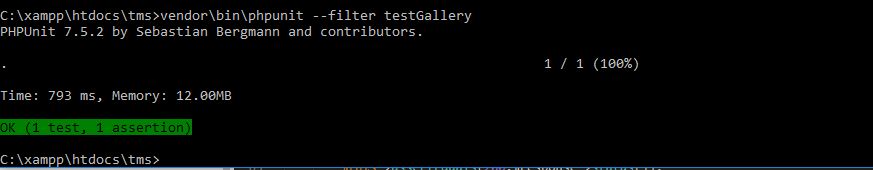
****

****

1. **Gallery**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Test Case No.** | **Testcase** | **Purpose of the test** | **Test Data** | **Function Name** | **Expected result** | **Actual result** | **Outcome as expected** |
| 3 | Gallery open. | To open gallery page |  | testGallery | User should open the gallery page. | User opened the gallery page. | Expected result matched with actual result. |

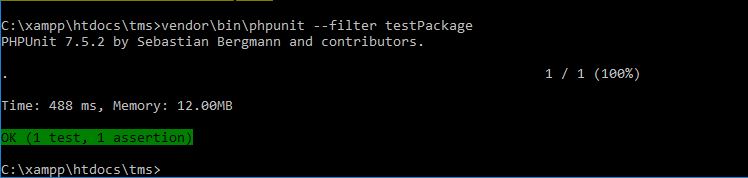




1. **Package**

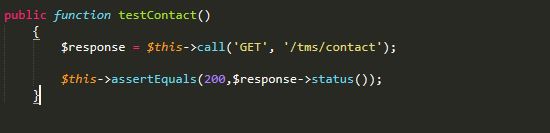
|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Test Case No.** | **Testcase** | **Purpose of the test** | **Test Data** | **Function Name** | **Expected result** | **Actual result** | **Outcome as expected** |
| 4 | Package Open | To open Package page. |  | testPackage | User should open the package page. | User opened the package page. | Expected result matched with actual result. |

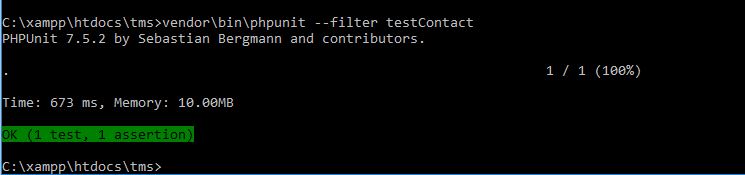




1. **Contact**

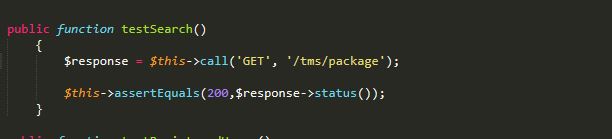
|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Test Case No.** | **Testcase** | **Purpose of the test** | **Test Data** | **Function Name** | **Expected result** | **Actual result** | **Outcome as expected** |
| 5 | Contact page open. | To open Contact page. |  | testContact | User should open the contact page. | User opened the contact page. | Expected result matched with actual result. |

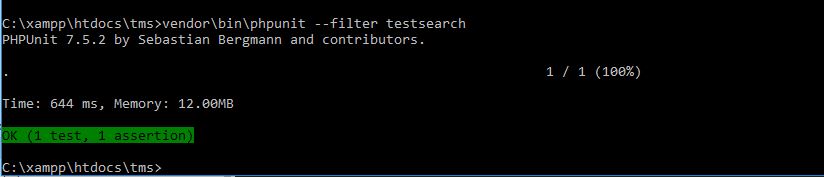




1. **Search**

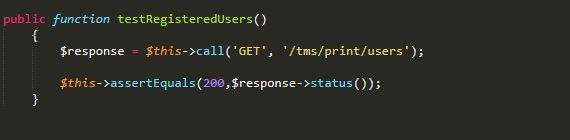
|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Test Case No.** | **Testcase** | **Purpose of the test** | **Test Data** | **Function Name** | **Expected result** | **Actual result** | **Outcome as expected** |
| 6 | Search package | To search package |  | testSearch | User should be able to search package | User searched the package. | Expected result matched with actual result. |

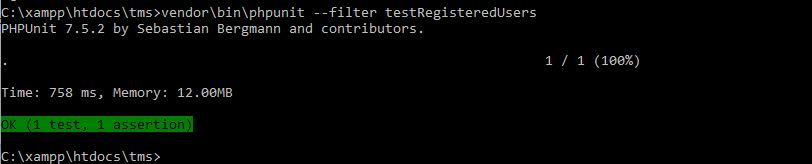




1. **Registered Users**

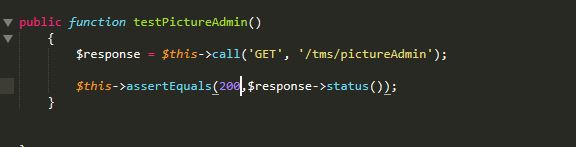
|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Test Case No.** | **Testcase** | **Purpose of the test** | **Test Data** | **Function Name** | **Expected result** | **Actual result** | **Outcome as expected** |
| 7 | Registered user open | To open registered users by admin |  | testRegisteredUsers | Admin should be able to open the page. | Admin opened the page. | Expected result matched with actual result. |

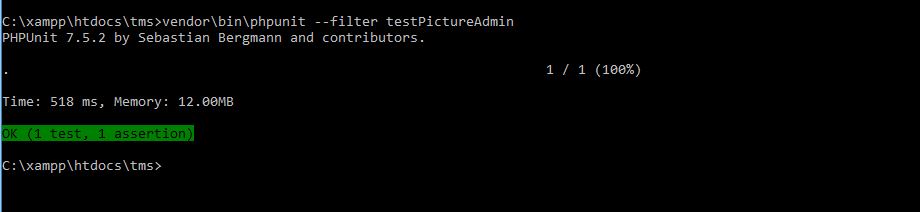
****

****

1. **Picture Admin**

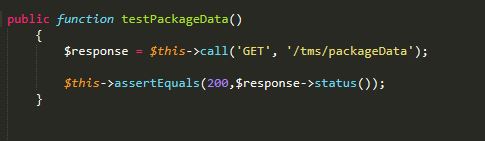
|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Test Case No.** | **Testcase** | **Purpose of the test** | **Test Data** | **Function Name** | **Expected result** | **Actual result** | **Outcome as expected** |
| 8 | Picture Admin Display | To open picture activity page by admin. |  | testPictureAdmin | Admin should be able to open the picture activity page. | Admin opened the picture activity page. | Expected result matched with actual result. |





1. **Package Data Admin**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Test Case No.** | **Testcase** | **Purpose of the test** | **Test Data** | **Function Name** | **Expected result** | **Actual result** | **Outcome as expected** |
| 9 | Package Data of admin page display. | To open package date of admin. |  | test | Admin should be able to open the picture activity page. | Admin opened the picture activity page. | Expected result matched with actual result. |

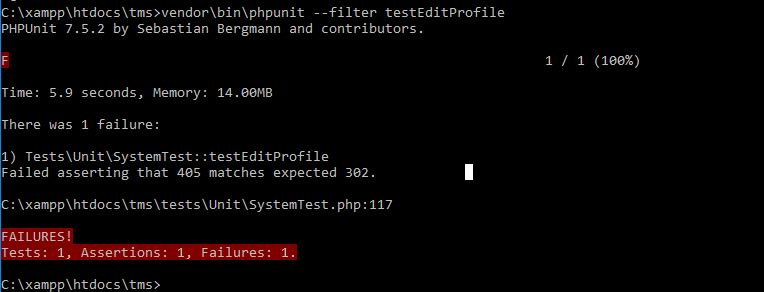
****

****

1. **Edit Profile**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Test Case No.** | **Testcase** | **Purpose of the test** | **Test Data** | **Function Name** | **Expected result** | **Actual result** | **Outcome as expected** |
| 10 | Editing profile. | To edit the profile. |  | testEditProfile | User should be able to edit profile | Not able to edit profile. | Expected result didn’t matched with actual result. |

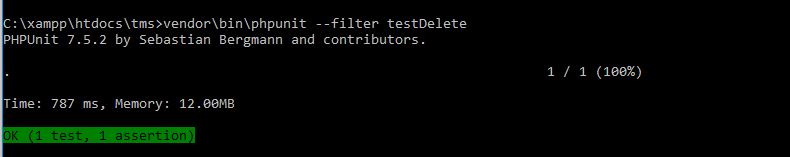
****

****

1. **Delete Data**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Test Case No.** | **Testcase** | **Purpose of the test** | **Test Data** | **Function Name** | **Expected result** | **Actual result** | **Outcome as expected** |
| 11 | Delete package data. | To delete the package data by admin. | Id=15 | testDelete | Admin should be able to delete data. | Admin was able to delete the data. | Expected result matched with actual result. |





# **Chapter 6 Other Project Issues**

## **Risk Management**

In order to identify and avoid the possible risk which may occur during the development of our project is risk management. It will help us to tackle the problem in real time when project should be implemented.

Risks that my project Tour Management System might face while development process are shown below with the values to define its level of attacking the system:

**Impact = Likelihood X Consequence**

|  |  |
| --- | --- |
| Likelihood | Value |
| Low | 1 |
| Medium | 2 |
| High | 3 |

Table: Risk likelihood values table

**Risk Consequence values are shown below:**

|  |  |
| --- | --- |
| Consequence | Value |
| Very low | 1 |
| Low | 2 |
| Medium | 3 |
| High | 4 |
| Very High | 5 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Risks** | **Likelihood** | **Consequence** | **Impact** | **Actions** |
| System failure | 1 | 4 | 4 | Prepare backup system and maintenance of a system |
| Hard disk crash | 2 | 4 | 8 | Proper backup multiple server. |
| Impractical budget | 2 | 4 | 8 | Proper planning of the budget required. |
| Hacking | 2 | 2 | 4 | Increase the system security. |
| Virus | 1 | 5 | 5 | Installation of Antivirus software. |
| Change in requirements | 3 | 4 | 12 | Keep Contracts and proof safely |
| Natural disaster | 3 | 4 | 12 | Proper backup and security of the data accordingly. |
| Spam | 2 | 1 | 2 | Block IP addresses for unauthorized users |

Table: Risk management

**Configuration Management**

It is the process of developing and maintaining reliability of the performance, functions as well as the physical attributes of any product with its necessities, enterprise and functioning information during its entire life.

It helps to track and keep the detail of the data systematically so that it can be accessed easily when required.

The given figure shows the configuration management of my project.

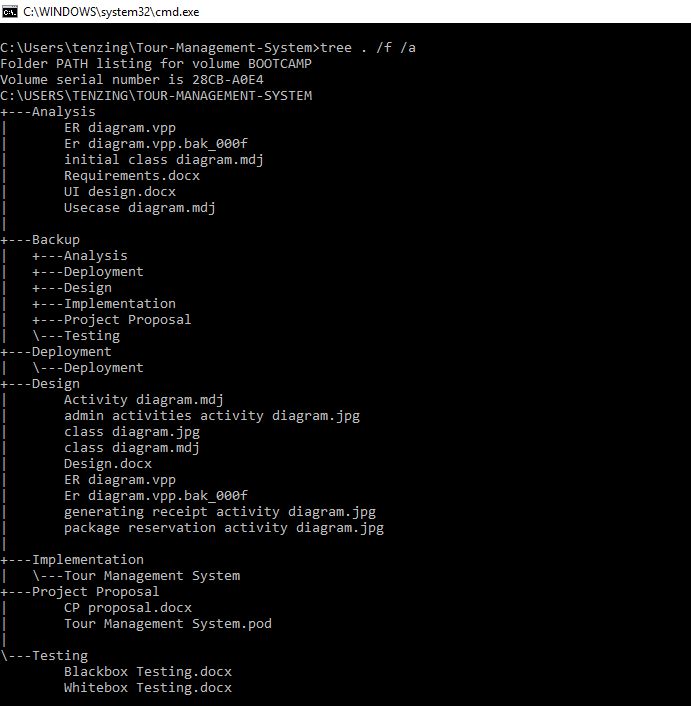


Figure 24Directory structure.

* **Main Directory**

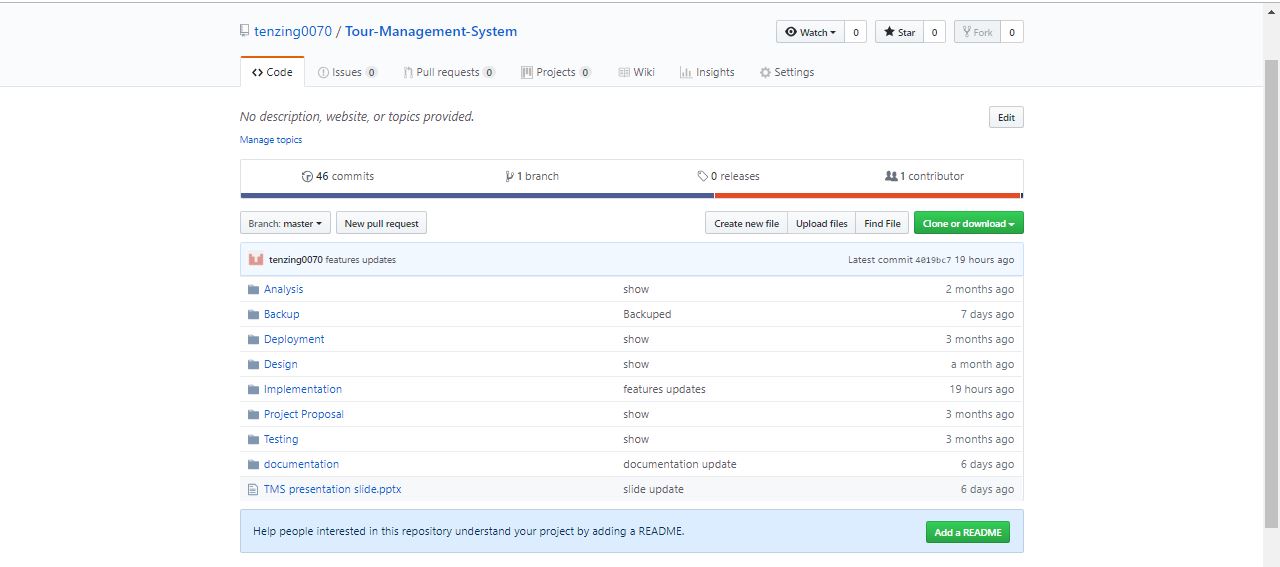
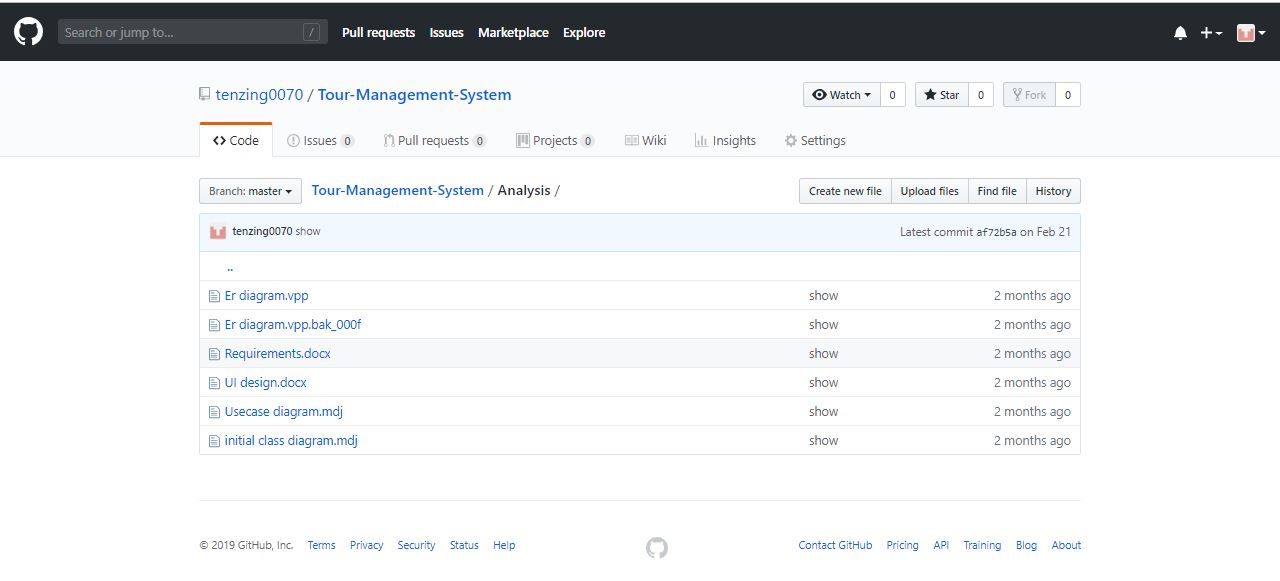
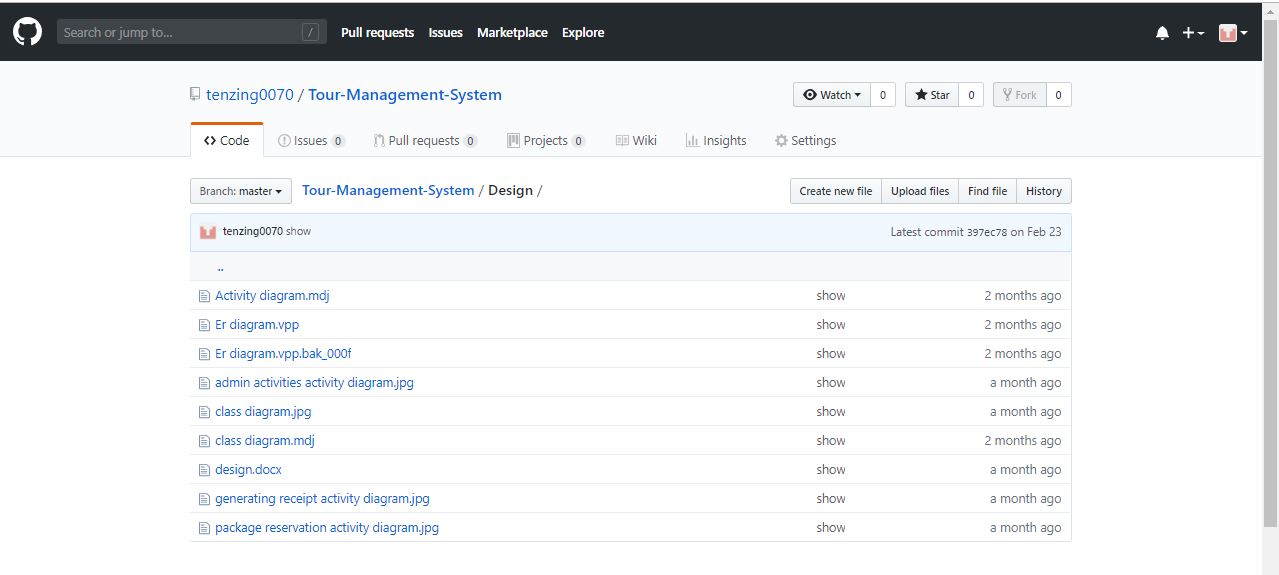
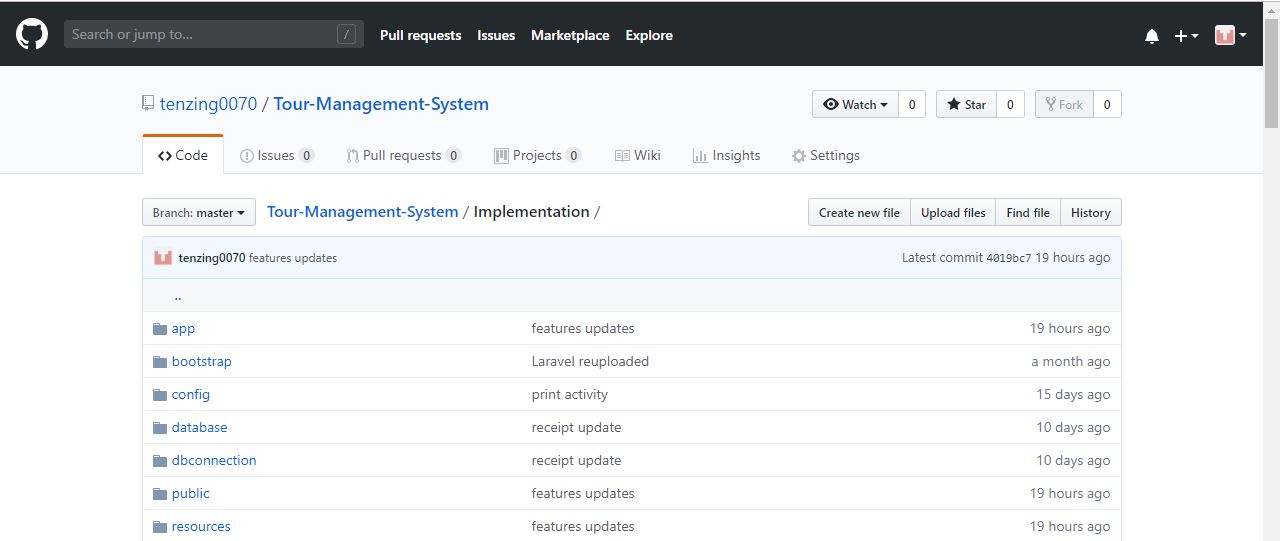
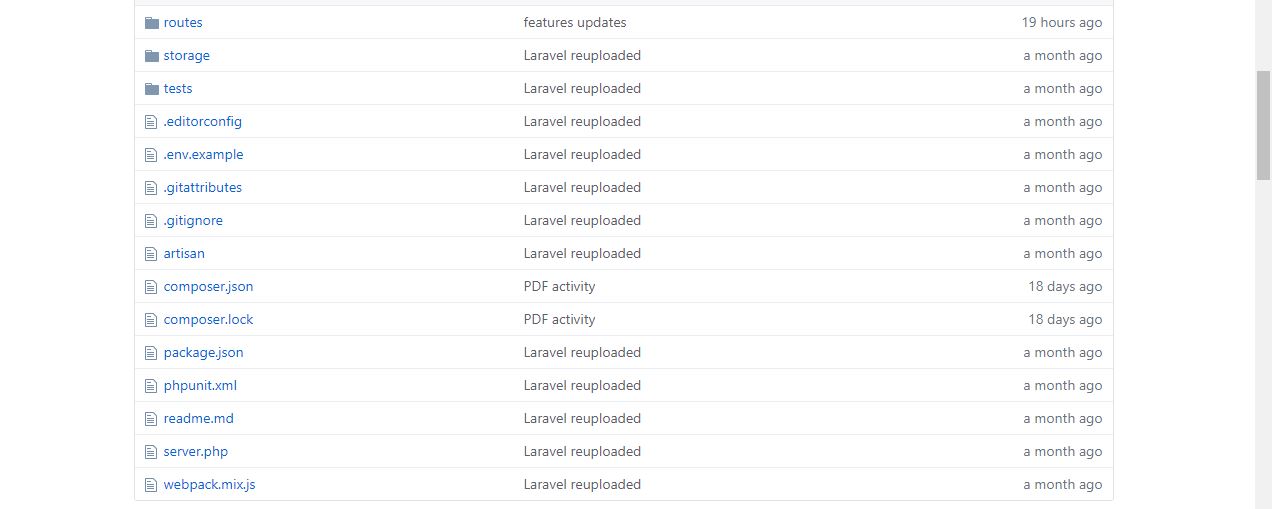
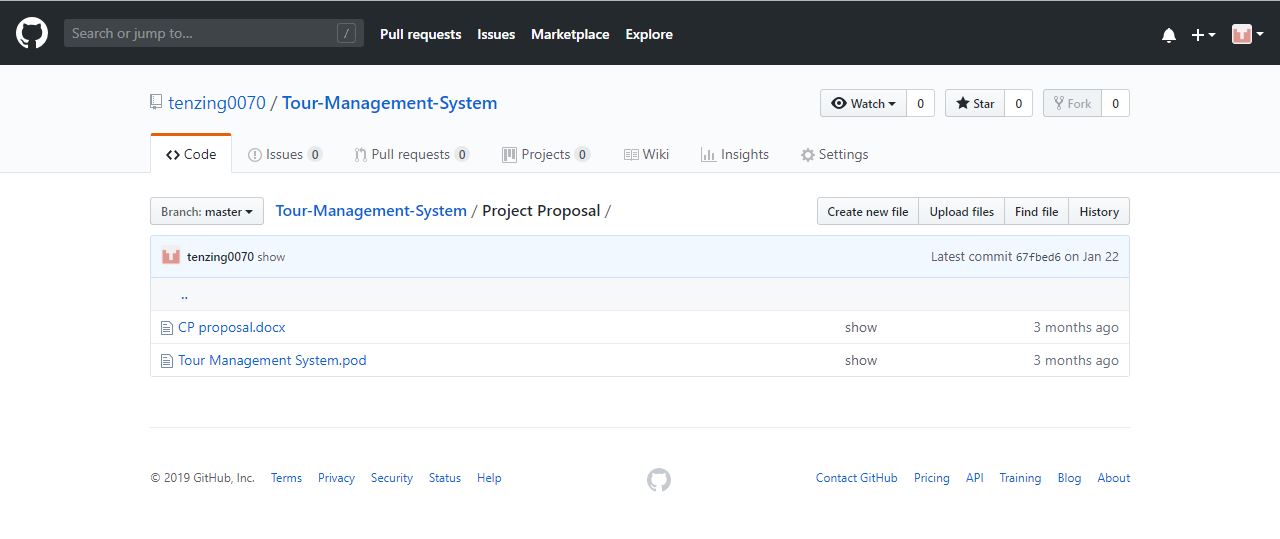


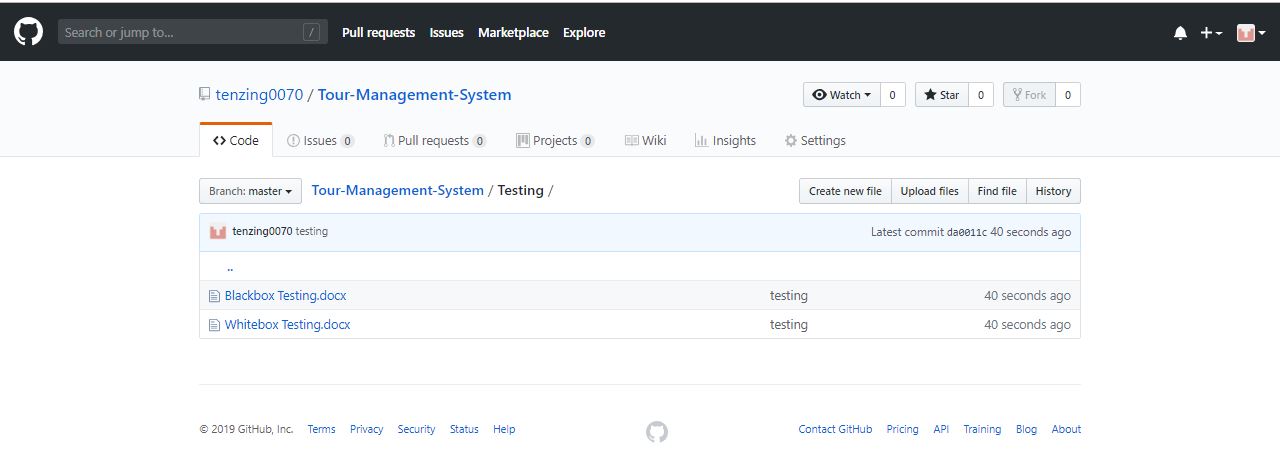
Figure 25 Main Directory folders.

* **Analysis**
* **Design**
* **Implementation**

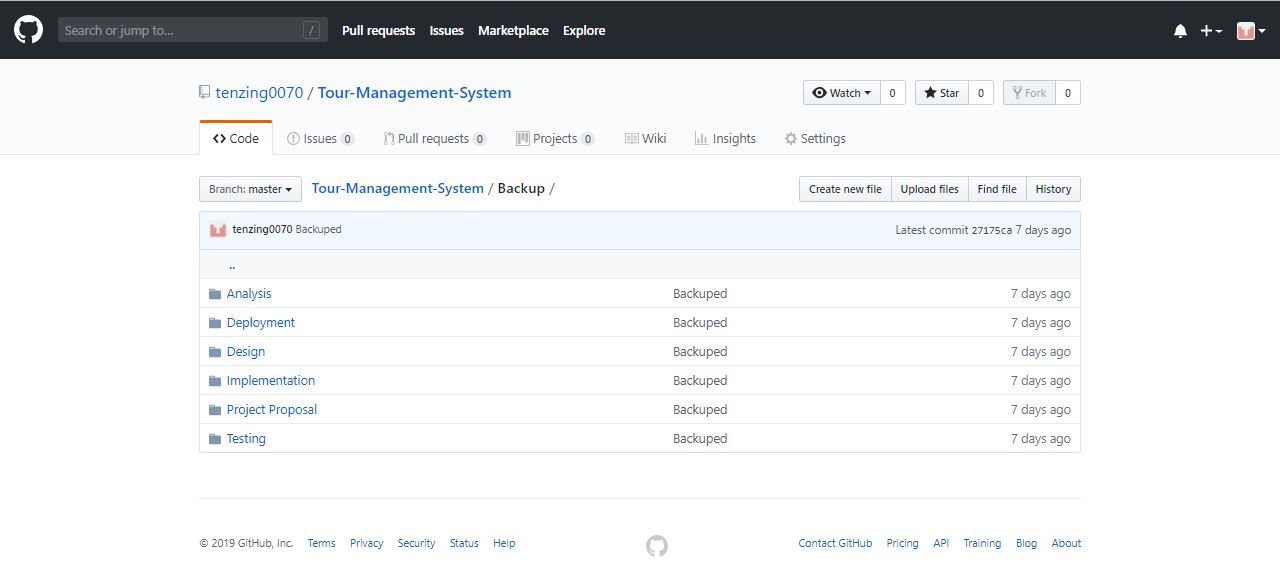




* **Project Proposal**
* **Deployment**
* **Testing**

****

* **Backup**

****

## **Scheduling**

## **Work Breakdown Structure (WBS) / Scheduling**

## **Work Breakdown Structure**

The WBS is the systematic breakdown of project into small parts where each part is related and followed by another. It organizes the teams into manageable so that teams can understand each level of the project. It makes any complex project more manageable. It's used because:

* It assists with accurate project organization.
* It helps in assigning responsibilities.
* It shows the control points and project milestones.
* It allows for more accurate estimation of cost, risk and time.

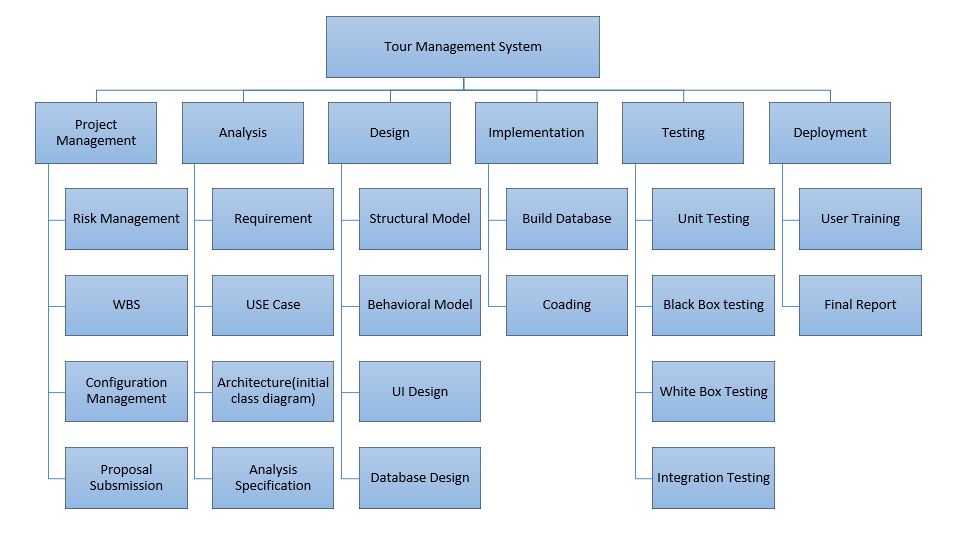


Figure 3 WBS

## **Milestones**

|  |  |
| --- | --- |
| **Milestones** | **Date (MM/DD/YY)** |
| **Project Management**  Risk Management  WBS  Configuration Management  Proposal Submission | **12/21/18-1/3/19**  12/21/18-12/24/18  12/25/18-12/27/18  12/28/18-1/1/19  1/2/19-1/3/19 |
| **Analysis**  Requirement  Use Case  Architecture (Initial Class Diagram)  Analysis specification | **1/4/19-1/27/19**  1/4/19-1/10/19  1/11/19-1/15/19  1/16/19-1/20/19  1/21/19-1/27/19 |
| **Design**  Structural Model  Behavioral Model  UI Design  Database Design | **1/28/19-2/26/19**  1/28/19-2/3/19  2/4/19-2/9/19  2/10/19-2/17/19  2/18/19-2/26/19 |
| **Implementation**  Building Database  Coding | **2/27/19-3/28/19**  2/27/19-3/8/19  3/9/19-3/28/19 |
| **Testing**  Unit Testing  Integration Testing  Blackbox Testing  Whitebox Testing | **3/29/19-4/8/19**  3/29/19-3/30/19  3/31/19-4/2/19  4/3/19-4/5/19  4/6/19-4/8/19 |
| **Deployment**  User Training  Final Report | **4/9/19-4/18/19**  4/9/19-4/14/19  4/15/19-4/18/19 |

## **Description of Milestones:**

1. **Project Management (14 days)**

* **Risk Management (4 days)**
* **WBS (3 days)**
* **Configuration Management (5 days)**
* **Proposal Submission (2 days)**

1. **Analysis (24 days)**

* **Requirement (7 days)**
* **Use Case (5 days)**
* **Architecture (Initial Class Diagram) (5 days)**
* **Analysis Specification (7 days)**

1. **Design (30 days)**

* **Structural Model (7 days)**
* **Behavioral Model (6 days)**
* **Structural Behavioral (8 days)**
* **Database Design (9 days)**

1. **Implementation (30 days)**

* **Build Database (10 days)**
* **Coding (20 days)**

1. **Testing (11 days)**

* **Unit Testing (2 days)**
* **Black Testing (3 days)**
* **White Testing (3 days)**
* **Integration (3 days)**

1. **Deployment (10 days)**

* **User Training (6 days)**
* **Final Report (4 days)**

## **Scheduling / Gantt Chart**

The time estimation is done for this project as per the days needed to complete each task. I have estimated total time of 119 days.

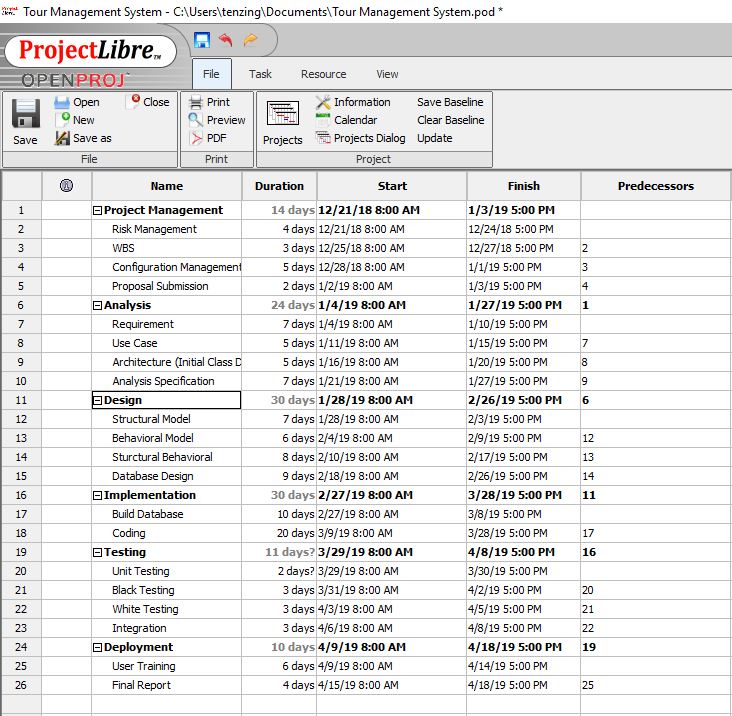


Figure 4 Scheduling

## **Gantt chart**

A Gantt chart is a graphical tool also known as visual presentation of project which shows activities or tasks performed against time the activities are broken down and displayed on a chart which makes it easy to understand and interpret.

The Gantt chart for my project is shown below:

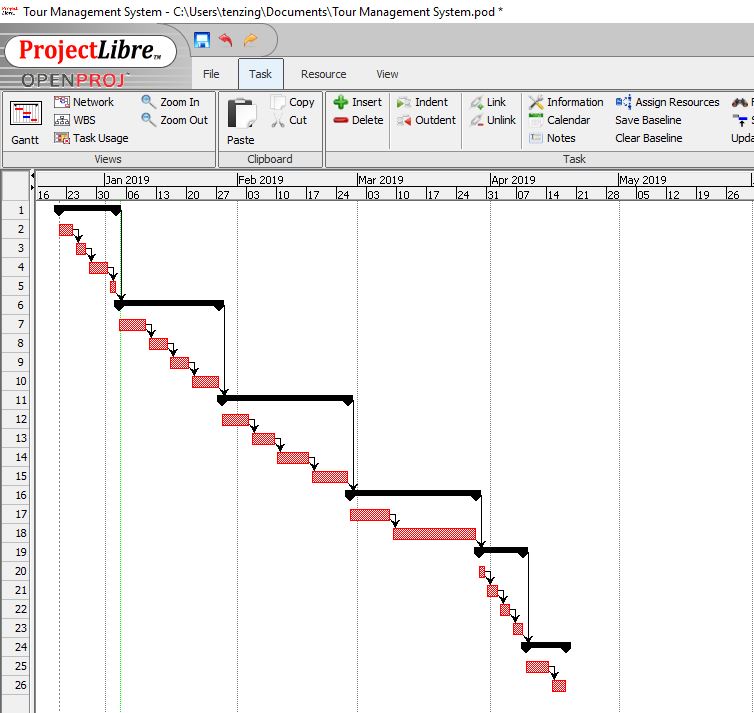


Figure 5 Gantt chart

## **Future Works**

* Online Payment.
* Real time tracking.
* Chat Box.
* Language options.
* Ability to track booking status
* Live seat availability

## **User Manual**

## **Limitation**

* Sometimes very high data interaction, so server gets down.
* Very high security needed to the system.
* Online payment cannot be made.
* Step by step procedure, so takes time.

# **Chapter 7 Conclusion**

# **Conclusion**

I have successfully completed the analysis phase of the developing software. In the completion of this phase, we have outcome the requirements, database model and user interface.

# **Chapter 8 Reference & Bibliography**

* Wikipedia.(n.d.).Retrieved from <https://en.wikipedia.org/wiki/Work_breakdown_structure>.
* <http://www.apiumhub.com/tech-blog-barcelona/benefits-of-agile-project-management/>

[Accessed on Dec 20, 2018]

* Agile Business Consortium. (2018). DSDM Atern Handbook (2008).

[online] Available at: <https://www.agilebusiness.org/content/moscow-prioritisation>

[Accessed 27 Feb. 2019]

* Anon, (2019). [online] Available at:

<https://developers.google.com/maps/documentation/javascript/tutorial>

[Accessed 17 Mar. 2019]

# **Chapter 9 Appendix**