[**Acknowledgement** 2](#_Toc525975850)

[**Abstract** 3](#_Toc525975851)

[**Chapter 1: Introduction** 4](#_Toc525975852)

[**1.1 Introduction** 4](#_Toc525975853)

[**1.2 Problem Statement** 4](#_Toc525975854)

[**1.3 Scope and Objectives** 5](#_Toc525975855)

[**1.4 Project Features** 5](#_Toc525975856)

[**1.5 Summary** 6](#_Toc525975857)

[**Chapter 2: Literature Review** 7](#_Toc525975858)

[**2.1 Overview of Existing System** 7](#_Toc525975859)

[**2.2 Limitation of Existing System** 7](#_Toc525975860)

[**2.2.1 The Causes** 8](#_Toc525975861)

[**2.2.2 Conclusion** 8](#_Toc525975862)

[**Chapter 3: System Analysis** 9](#_Toc525975863)

[**3.1 Requirement Specification** 9](#_Toc525975864)

[**3.1.1 Functional Requirement** 9](#_Toc525975865)

[**3.1.2 Non-Functional Requirement** 12](#_Toc525975866)

[**Chapter 4: System Design** 14](#_Toc525975867)

[**4.1 Context Diagram** 14](#_Toc525975868)

[**4.2 Data Flow Diagram** 15](#_Toc525975869)

[**4.3 Entity Relationship Diagram** 16](#_Toc525975870)

[**4.4 Class Diagram** 17](#_Toc525975871)

[**4.5 Use Case Diagram** 19](#_Toc525975872)

[**Chapter 5: System Development and Implementation** 20](#_Toc525975873)

[**5.1 Programming Platform** 20](#_Toc525975874)

[**5.2 Tools & Technologies** 21](#_Toc525975875)

[**5.2.1 Tools** 21](#_Toc525975876)

[**5.3 Testing & Debugging** 22](#_Toc525975877)

[**5.3.1 Test Cases** 22](#_Toc525975878)

[**5.4 Implementation & Result** 43](#_Toc525975879)

[**5.4.1 Implementation** 43](#_Toc525975880)

[**5.4.2 Result** 43](#_Toc525975881)

[**5.5 Maintenance** 43](#_Toc525975882)

[**Chapter 6: Conclusion and Future Enhancement** 45](#_Toc525975883)

[**6.1 Conclusion** 45](#_Toc525975884)

[**6.2 Future Enhancement** 45](#_Toc525975885)

[**References** 46](#_Toc525975886)

[**Appendices** 48](#_Toc525975887)

[**Screenshots** 48](#_Toc525975888)

# **Acknowledgement**

In completing this graduate project we have been fortunate to have help, support and encouragement from many people. I would like to acknowledge them for their cooperation.

First, I would like to thank Mr. Kusal Niraula, my project supervisior, for guiding me through each and every step of the process with his knowledge and support. Thank you for your advice, guidance and assistance.

I would also like to thank Mr. Raju Kattel, our Head of Department, who showed immense patience and understanding throughout the project and provided suggestions.

Finally, I would like to thank all of my friends who helped throughout the project by providing supporting materials and friendly advice to make this project a success.

# **Abstract**

Bus reservation is an online ticket booking platform that allows different travel agency to sell their bus tickets online. It utilizes business-to-consumer aspect of electronic commerce (e-commerce) which is the most visible business use of the World Wide Web. In this system, a user has the ability to search for specific route and choose affordable bus that travel through that route. In other words, a user can basically search for bus for a given route and travel date, choose a bus from list of available buses and book it. They will also have the ability to choose the seat they want for their journey.

This project deals with developing an e-commerce based website for online bus ticket reservation. It provides a user with access to different bus travel agency whose bus is available for booking. In order to facilitate payment for booked seats, a user can directly contact the travel agency where that bus belongs to. The system is developed using a 3-tier approach in mind, with a backend database (MySQL), a middle tier of Apache Server and PHP, and a web browser as the front end client. The application development framework used here is CodeIgniter.

This is a project with the objective to develop a basic website where a user is allowed to book seats on a bus of their choice. This document provides information about the technologies that has been used to develop such a system.

# 

# **Chapter 1: Introduction**

## **1.1 Introduction**

There are a lot of times when people are unable to buy a proper seat for their journey right before festival seasons. People don't want get into a huge crowd just to get a ticket for their destination. On top of that most people don’t have time to stay on a queue to get their ticket. Also, these days everyone is looking for a better and easier ways to book a ticket for such occasions.

So, to address this problem, a new system has been developed. This system is called the “Bus Reservation”. In this system, a user has the ability to search buses for a given route on a specific date. In other words, a user can search, choose bus, pick a seat and finally pay for it using online payment portals. Basically, this allows a user to book and buy tickets without reaching to a ticket counter. Also, this system aims to allow all bus operators to come under a single platform so there won't a scarcity of buses online for booking. So in short, this system also allows a user to reserve a seat in a bus. Hence, the term “Bus Reservation” since people are allowed to secure a seat for their journey.

The major concept of this system is to allow business to consumer transaction. So, no central governing body has been implemented in this system which controls the transaction among users. But there will be a body responsible for technically maintaining servers and other infrastructure for this system to be functional.

## **1.2 Problem Statement**

Generally, ticket booking was not done online (a few years back) mostly because due to lack of internet availability among many people. But even though we have access to the internet, still, there are not many portals available that facilitate bus ticket booking operations. Moreover, there are even few portals that allowed actual payment transactions while bus ticket booking. But the major problem is the service they provide is not of a satisfactory level as they didn't include a wide range of travel routes. On top of what is available, half of them were mostly broken in one way or the other such as broken links, insecure online transactions, lack of wide range of travel routes and so on. Most of the available tickets booking sites are owned by a travel agency who rarely update their sites and keep track of the available routes. So, people still have to contact travel agencies to get their ticket booked.

Also people these days are quite busy to go and buy a ticket for their journey. And most of the time, even in off seasons, people cannot pick a bus of their choice due to lack of available options. To matter even worst, people have to sit on a seat that is provided by the agents of that particular bus agency. This particular case becomes an actual problem when a passenger has motion sickness and requires traveling next to a window seat.

Thus, the proposed system aims to address such issue by providing an online portal where one can easily search a bus, choose a seat of their choice and pay for it.

## **1.3 Scope and Objectives**

* To allow users to book bus seats of their choice.
* To bring different bus agencies into single platform.
* To make bus booking easier.

## **1.4 Project Features**

The following are list of features that has been implemented on this system:

* Bus searching facility
* Choose a seat for booking
* Search for ticket that was purchased by a customer
* Ability for an agency admins to keep track of their staff (requires agency admin role)
* Ability to set travel schedule (requires agency user role)
* Ability to add or remove bus and its schedules.

## **1.5 Summary**

Bus Reservation is an online platform that focuses on making bus ticket reservation easier. It has tried to implement business to consumer transaction. That means, there is no presence of “middle man” who will gain profit upon sale of a ticket. It has tried to achieve this by allowing user to directly search a bus, choose a seat and pay the price for that choice. The booked ticket can later be retrieved using unique ticket id provided during payment.

In this, everyone can book a seat on an available bus without the need of logging into the system. Also, they will have a choice to choose between many other bus operators and different price for same route. Thus, bus reservation has tried to take ticket booking to another level by providing user with such facilities.

# **Chapter 2: Literature Review**

## **2.1 Overview of Existing System**

There are quite a few existing portals that deal with online bus ticket booking. Two different system that has been considered as an existing system for this project. They are:-

* Westnepalbus.com
  + The features provided by this system are:
    - Print ticket facility
    - Contains wide range of travel routes
    - List of available bus agency
    - Belongs to an organization
* Busnepal.com
  + The features provided by this system are:
    - Print ticket facility
    - Contains wide range of travel routes
    - Support for payment gateway like esewa and ipay

## **2.2 Limitation of Existing System**

Like everything else, the system for this review phase also has some limitations. They are:

* Westnepalbus.com
  + Broken links
  + No result upon submitting the inputs
  + Need to call a provide phone number to check availability of seat
* Busnepal.com
  + Broken links
  + No result upon submitting the inputs
  + Popped error page every now and then
  + Need to call agency office to check bus availability

### **2.2.1 The Causes**

Upon doing further analysis, by visiting, similar website owners, I found that one of the major causes of such problems was lack of technical support. Once bus reservation system is up and running, many software developers and software development firms stops providing support and hands over the website to their owner.

Another issue I found was the website owner themselves. They said they have a lot of customers during festival seasons so booking tickets via ticket counter only is not enough. So they only make their website up and running during festival seasons. And rest of the time, they leave the website unattended since no one cares about online book rather people call the owner of the travel agency and simply book the ticket via phone.

### **2.2.2 Conclusion**

In summary, many of the websites have no availability of bus for the route and date provided as an input. This indicates that the sites are not maintained, updated, checked and verified properly in a regular interval.

# **Chapter 3: System Analysis**

## **3.1 Requirement Specification**

### **3.1.1 Functional Requirement**

This section includes the requirements that specify all the fundamental actions of the software system. The lists of all functional requirements are given below:

**3.1.1.1 Functional Requirement 1.1**

ID: FR1

TITLE: Bus Searching

DESC: A customer should be able to search buses that are posted on the website. User should be able to search bus based on the date and route provided.

DEP: None

**3.1.1.2 Functional Requirement 1.2**

ID: FR2

TITLE: Check Seat Availability

DESC: A customer should be able to choose seats from the chosen bus.

DEP: FR1

**3.1.1.3 Functional Requirement 1.3**

ID: FR3

TITLE: Ticket Booking

DESC: A customer should be able to get the details of the booked ticket.

DEP: FR1

**3.1.1.4 Functional Requirement 1.4**

ID: FR4

TITLE: Ticket Printing

DESC: The customer should be able to print or download booked tickets through the website.

DEP: FR 3

**3.1.1.5 Functional Requirement 1.5**

ID: FR5

TITLE: User log-in

DESC: Given that a user has been registered, then the user should be able to log in to the system.

DEP: None

**3.1.1.6 Functional Requirement 1.6**

ID: FR6

TITLE: Buses (Bus Management) - Agency Member

DESC: Given that a user has successfully logged into the system, the user should be able to add new bus, remove old bus and update existing buses.

DEP: FR5

**3.1.1.7 Functional Requirement 2.1**

ID: FR7

TITLE: Schedules (Schedule Management) - Agency Member

DESC: Given that a user has successfully logged into the system, then the user should be able to add new schedule, remove old schedule and update schedule.

DEP: FR5, FR6

**3.1.1.8 Functional Requirement 2.2**

ID: FR8

TITLE: Tickets (Ticket Management) - Agency Member

DESC: Given that a user has successfully logged into the system, then the user should be able to update the tickets of customer and remove the ticket if the booking needs cancellation.

DEP: FR5, FR3

**3.1.1.9 Functional Requirement 2.3**

ID: FR9

TITLE: Updating credentials - Agency Member

DESC: Given that a user has successfully logged into the system, then the user should be able to update their credentials. This includes username and password change.

DEP: FR5

**3.1.1.10 Functional Requirement 2.4**

ID: FR10

TITLE: Users (User management) - Agency Member Admin

DESC: Given that a user has successfully logged into the system, then the agency admin should be able to add, update and delete its agency member.

DEP: FR5

**3.1.1.11 Functional Requirement 3.1**

ID: FR11

TITLE: Managing Picture Slideshow - Admin Module

DESC: Given that admin has successfully logged into the system, then the admin should be able to add new picture to the picture slider, update it and remove it. The admin should be able to view the currently active slide show picture with the system.

DEP: FR5

**3.1.1.12 Functional Requirement 3.2**

ID: FR12

TITLE: Places (Place Management)- Admin Module

DESC: Given that a admin has successfully logged into the system, then the admin should be able to add new places, update them and remove them.

DEP: FR5

**3.1.1.13 Functional Requirement 3.3**

ID: FR13

TITLE: Travel Agencies - Admin Module

DESC: Given that admin has successfully logged into the system, then the admin should be able to add new agency, update it and remove it.

DEP: FR5

**3.1.1.14 Functional Requirement 3.4**

ID: FR14

TITLE: Updating credentials - Admin Module

DESC: Given that a admin has successfully logged into the system, then the admin should be able to update their own credentials. This includes username and password change.

DEP: FR5

**3.1.1.15 Functional Requirement 3.5**

ID: FR15

TITLE: Agency Admin Management - Admin Module

DESC: Given that a admin has successfully logged into the system, then the admin should be able to add new agency admin, update their credential and delete it.

DEP: FR5

### **3.1.2 Non-Functional Requirement**

This section includes the requirements that specify all the fundamental actions of the software system. The lists of all functional requirements are given below:

**3.1.2.1 Non- Functional Requirement 1.1**

ID: NFR1

TITLE: Webpage Loading

DESC: The webpage should load faster so that user can quickly browse through the website to find what they are looking for.

DEP: none

**3.1.2.2 Non- Functional Requirement 1.2**

ID: NFR2

TITLE: Error Handling

DESC: The webpage should not show any major errors that could reveal its programming language. Instead, the website should redirect to a generic error page indicating some generic error information.

DEP: none

**3.1.2.3 Non- Functional Requirement 1.3**

ID: NFR3

TITLE: User Friendly Interface

DESC: The user interface should be easy to use. Website navigation should follow a proper pattern. It should also contain links at proper places so that user won't get lost within the website.

DEP: none

**3.1.2.4 Non- Functional Requirement 1.4**

ID: NFR4

TITLE: Security

DESC: The website should not be prone to attacks. It should be secure enough to handle attacks like cross site scripting and SQL injection.

DEP: none

**3.1.2.5 Non- Functional Requirement 1.5**

ID: NFR5

TITLE: Availability

DESC: The website should be hosted in a public network. The user should be able to access from anywhere around the world.

DEP: none

**3.1.2.6 Non- Functional Requirement 1.6**

ID: NFR6

TITLE: Reliability

DESC: The website should be reliable enough to perform its tasks without crashing. But for any input from user that is beyond its predefined functionality, the website should show an error page indicating the error.

# **Chapter 4: System Design**

## **4.1 Context Diagram**

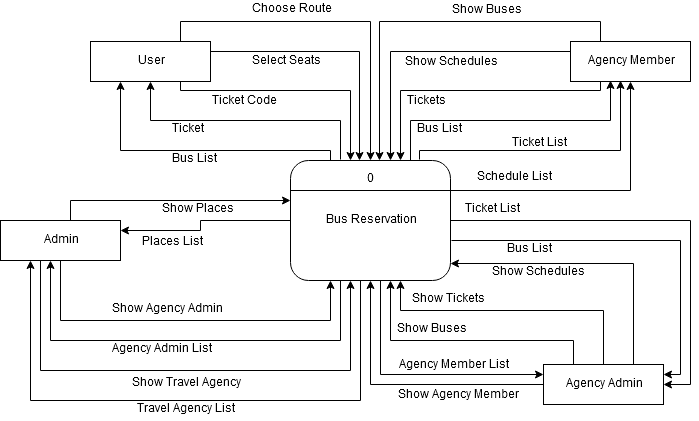


Fig 4.1: Context Diagram

## **4.2 Data Flow Diagram**

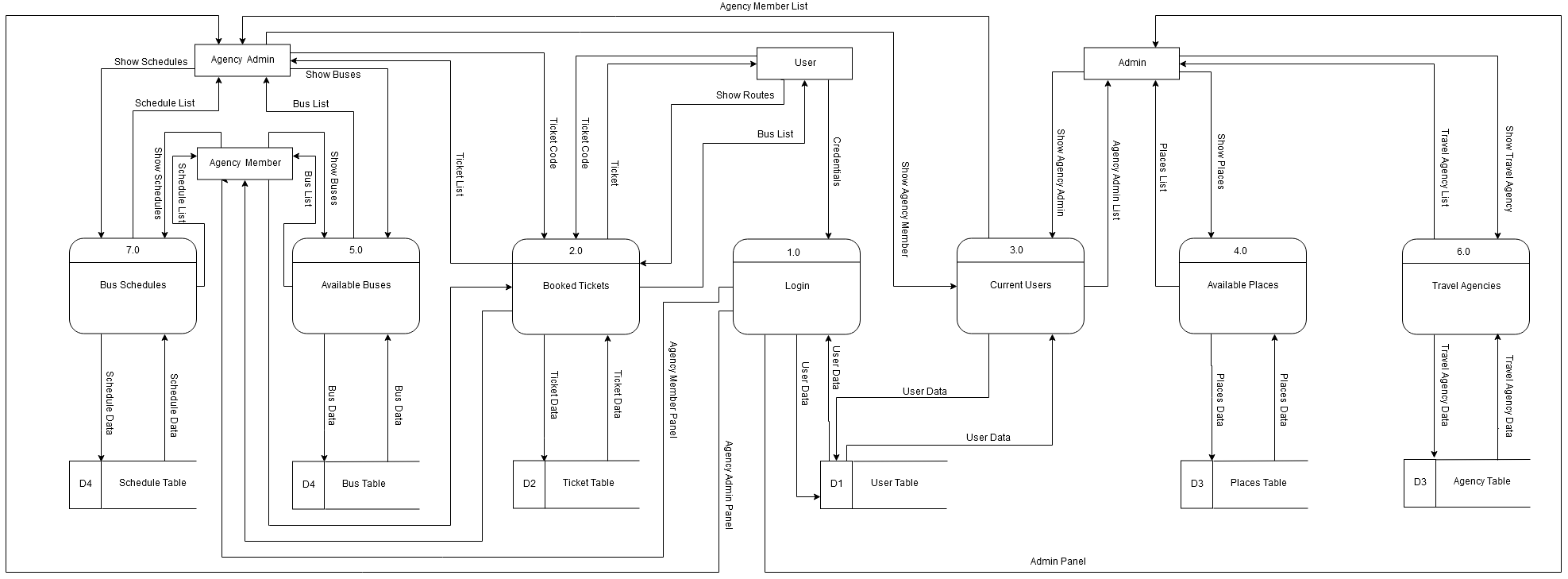


Fig 4.2: Data Flow Diagram

# 

## **4.3 Entity Relationship Diagram**

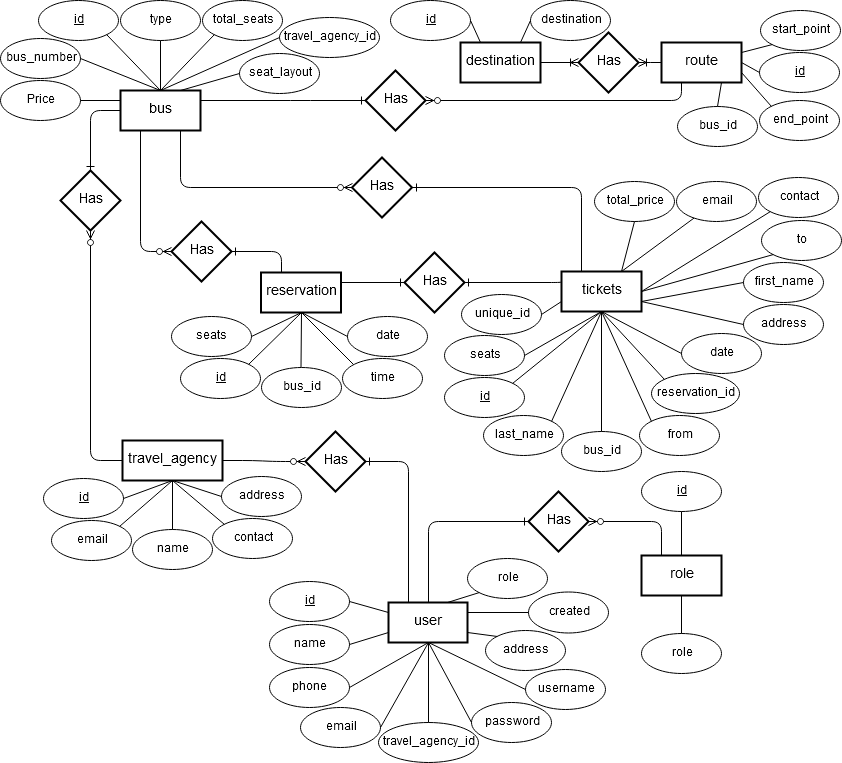


Fig 4.3: Entity Relationship Diagram

## **4.4 Class Diagram**

## Class Diagram - Controller.png

Fig 4.4.1 Class Diagram - Controller

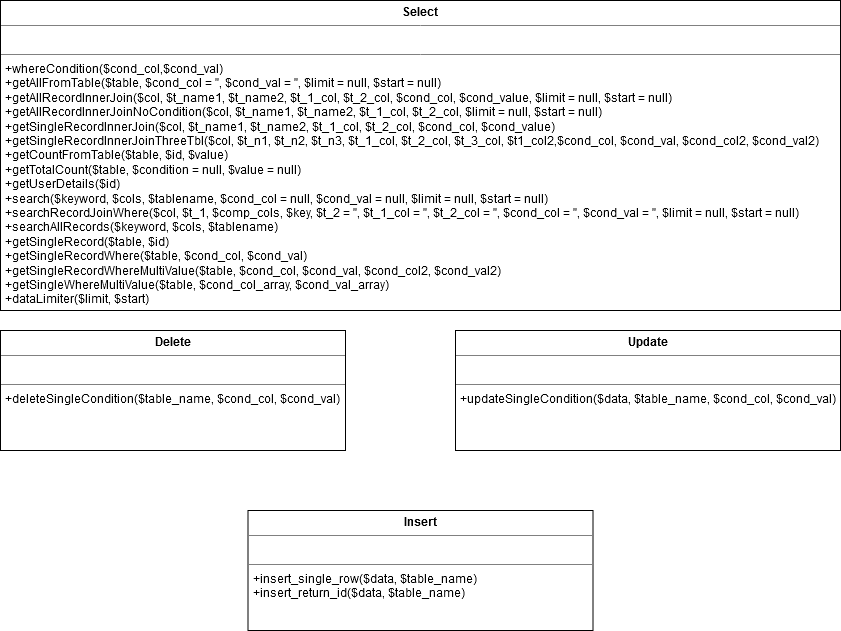


Fig 4.4.2: Class Diagram - Models

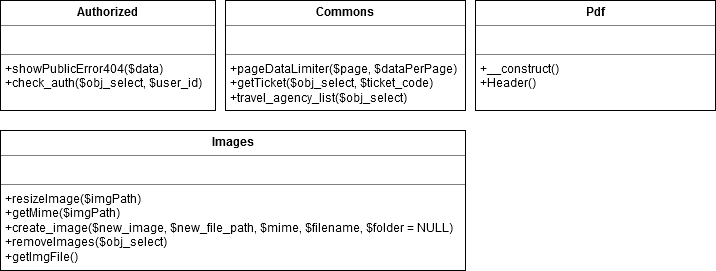


Fig 4.4.3: Class Diagram - Libraries

## **4.5 Use Case Diagram**

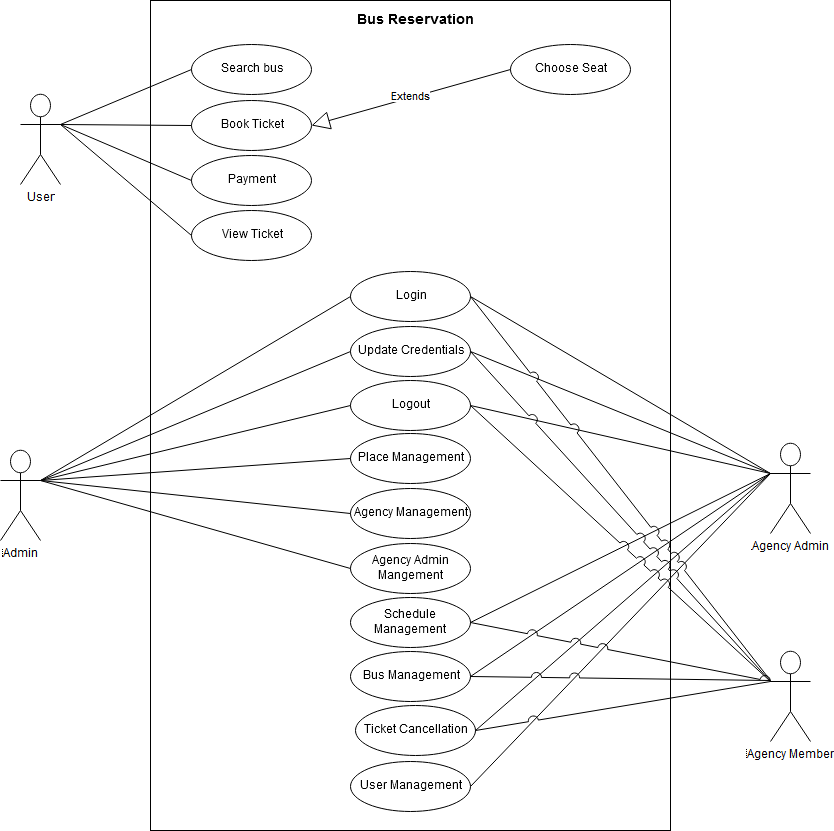


Fig 4.5: Use Case Diagram

# **Chapter 5: System Development and Implementation**

## **5.1 Programming Platform**

This system has been built using PHP (hypertext preprocessor) language. It is a widely-used open source general-purpose scripting language that is especially suited for web development and can be embedded into HTML. It has following features:

* It can run on both UNIX and windows servers.
* It works in combination with HTML to display dynamic elements on the page.
* It only parses PHP code and doesn't parse HTML tags so it is very efficient.
* It supports large number of relational database management system.

This system also uses a relational database to store all the data. The database that has been used here is MySQL. It has following features:

* It's easy to use since all we need to know is the basic knowledge of SQL.
* It's secure since rights can be set to allow some or all privileges to individuals and passwords are encrypted.
* It's inexpensive and is available by free download from MySQL Web site.
* It's fast and scalable since it can handle almost any amount of data, up to as much as 50 million rows or more.
* It manages memory very well and has been thoroughly tested to prevent memory leaks.
* It runs on many operating systems including Novell NetWare, Windows\* Linux\*, many varieties of UNIX\* (such as Sun\* Solaris\*, AIX, and DEC\* UNIX), OS/2, FreeBSD\*, and others.
* It supports several development interfaces including JDBC, ODBC, and scripting language such as PHP and Perl.

## **5.2 Tools & Technologies**

### **5.2.1 Tools**

Even though PHP backed this system, there is yet another type of tool/framework that has been used on top of PHP within this system. That framework is called ‘CodeIgniter’. CodeIgniter is a powerful PHP framework with a very small footprint, built for developers who need a simple and elegant toolkit to create full-featured web applications. It has following features:

* It is a framework with a small file size including complete user guide.
* It encourages MVC pattern but doesn't force it upon user.
* It is very fast as compared to other similar framework.
* It is consists of built-in protection against CSRF and XSS attacks.

**5.2.2 Technologies**

Other than CodeIgniter, this system utilizes 3 tier architecture. Each tier has following functionalities:

* **Presentation tier**

The top most level of the application is the user interface. The main function of the interface is to translate tasks and results to something the user can understand.

* **Logic tier**

This layer coordinates the application, processes commands, makes logical decisions and evaluations, and performs calculations. It also moves and processes data between the two surrounding layers.

* **Data tier**

Here information is stored and retrieved from a database of file system. The information is the passed back to the logic tier for processing, and then eventually back to the user.

## **5.3 Testing & Debugging**

Testing is one of the crucial stages in Software Development Life Cycle. Testing is a mechanism of checking the functioning and usability of a system. Test cases are made as soon as the coding work was complete for a task. The details of test performed and the criteria for testing (test cases) each and every functionalities are given below:

### **5.3.1 Test Cases**

**ID:** TC1

**Title:** Login Page – Authenticate successfully as member or administrative user

**Description:** A registered user should be able to successfully login at to their respective panels.

**Precondition:** The user must already be added to the system and must have username and password.

**Assumption:** User has required privilege.

**Test Steps:**

1. Navigate to the login URL by typing **http://localhost/bus\_reservation/member** or click on **login** link at the home page.
2. In the **Username** field, enter the username and in **password** field, enter the password.
3. Click the **Login** button.

**Expected Result:** A page displaying (admin or member) panel should appear on the browser.

**Debugging:** Check user's roles in the **user** and **role** table of the database. Check and enter the correct password.

**ID:** TC2

**Title:** Member Home Page

**Description:** A logged in member should be able to see 4 icons (schedules, buses, tickets and settings) in the dashboard.

**Precondition:** The user must be logged into the system.

**Assumption:** User has required privilege.

**Test Steps:**

1. Check 4 icons (schedules, buses, tickets and settings) in the dashboard.
2. Click the username on the upper right side of the webpage to confirm the user access.

**Expected Result:** A page displaying 4 icons (schedules, buses, tickets and settings) in the dashboard should appear on the browser. Also, a member with agency admin role should be able to see a link with name **Users** under the username on the upper right of the webpage.

**Debugging:** Check user privilege in the database.

**ID:** TC3

**Title:** Bus Schedules Page

**Description:** A logged in member should be able to see a list of scheduled bus for specific dates on the screen.

**Precondition:** The user must be logged in to the system.

**Assumption:** Bus schedule has been placed earlier by same or another user.

**Test Steps:**

1. Click on the **Schedules** icon on the dashboard icons or click on the **Schedules** in the navigation bar.
2. Check the list of scheduled bus and confirm departure date.

**Expected Result:** A list displaying all the buses with departure date greater than today should appear.

**Debugging:** If no records, add new record to the list. If issue persists, requires code level changes.

**ID:** TC4

**Title:** Add Schedules Page

**Description:** A logged in member should be able add new schedule for specific date.

**Precondition:** The user must be logged into the system.

**Assumption:** User has required privilege.

**Test Steps:**

1. Click on the **Add New** button just above the list of bus schedules.
2. Choose a destination from the list inside the drop down.
3. Choose the **date** and enter **time** in 24 hour format.
4. Click on **Add**.
5. A message showing successful addition should appear on the screen.

**Expected Result:** A new entry should be visible on the list of schedules.

**Debugging:** Requires code level changes.

**ID:** TC5

**Title:** Edit Schedules Page

**Description:** A logged in member should be able edit schedule for specific date.

**Precondition:** The user must be logged into the system.

**Assumption:** User has required privilege.

**Test Steps:**

1. Click on the **Edit** link under **Actions** column in the list of bus schedules.
2. Change the values of **date** and **time**.
3. Click on **Update**.
4. A message showing successful update should appear on the screen.

**Expected Result:** Date or Time should have updated values on the list of schedules.

**Debugging:** Requires code level changes.

**ID:** TC6

**Title:** Delete Schedules

**Description:** A logged in member should be delete existing schedule for specific date from the list of schedule.

**Precondition:** The user must be logged into the system and list of bus schedules should be visible.

**Assumption:** User has required privilege.

**Test Steps:**

1. Click on the **Delete** link under **Actions** column in the list of bus schedules.
2. A popup should appear showing the details of choice for delete.
3. Confirm to delete the record.
4. A message showing successful deletion should appear on the screen.
5. Selected entry from the list should disappear.

**Expected Result:** Selected entry for deletion should disappear from the list of schedules.

**Debugging:** Requires code level changes.

**ID:** TC7

**Title:** Bus Page

**Description:** A logged in member should be able to see a list of buses.

**Precondition:** The user must be logged into the system.

**Assumption:** User has required privilege. Bus list has been placed earlier by same or another user.

**Test Steps:**

1. Click on the **Buses** on the dashboard icons or click on the **Buses** in the navigation bar.
2. A list of available buses should be visible.

**Expected Result:** A list displaying all the buses with routes, bus number and ticket price should appear.

**Debugging:** If no records, add new record to the list. If issue persists, requires code level changes.

**ID:** TC8

**Title:** Add Bus Page

**Description:** A logged in member should be able add new bus.

**Precondition:** The user must be logged into the system.

**Assumption:** User has required privilege.

**Test Steps:**

1. Click on the **Add New** button just above the list of buses.
2. Choose a value for **From**, **To** and **Bus Type**.
3. Then set **total seat**, **bus number** and **price** of the seat.
4. Click on **Add**.
5. A message showing successful addition should appear on the screen.

**Expected Result:** A new entry should be visible on the list of buses.

**Debugging:** Requires code level changes.

**ID:** TC9

**Title:** Edit Bus Page

**Description:** A logged in member should be able edit bus from the list of buses.

**Precondition:** The user must be logged into the system and list of buses should be visible.

**Assumption:** User has required privilege.

**Test Steps:**

1. Click on the **Edit** link under **Actions** column in the list of buses.
2. Change the values of **From**, **To** and **Bus Type**.
3. Then set **total seat**, **bus number** and **price** of the seat as per requirement.
4. Click on **Update**.
5. A message showing successful update should appear on the screen.

**Expected Result:** Updated values should be visible on the list of schedules.

**Debugging:** Requires code level changes.

**ID:** TC10

**Title:** Delete Bus

**Description:** A logged in member should be delete an existing bus from the list of buses.

**Precondition:** The user must be logged into the system and list of buses should be visible.

**Assumption:** User has required privilege.

**Test Steps:**

1. Click on the **Delete** link under **Actions** column in the list of buses.
2. A popup should appear showing the details of choice for delete.
3. Confirm to delete the record.
4. A message showing successful deletion should appear on the screen.
5. Selected entry from the list should disappear.

**Expected Result:** Selected entry for deletion should disappear from the list of buses.

**Debugging:** Requires code level changes.

**ID:** TC11

**Title:** Ticket Page

**Description:** A logged in member should be able to see a list of tickets.

**Precondition:** The user must be logged into the system.

**Assumption:** Ticket list has been placed earlier by passengers.

**Test Steps:**

1. Click on the **Tickets** on the dashboard icons or click on the **Tickets** in the navigation bar.
2. A list of booked tickets should be visible.

**Expected Result:** A list displaying all the tickets with Customer Name, Address, Contact Details, Ticket Price etc should appear.

**Debugging:** Add new record to the list by booking a ticket. If issue persists, requires code level changes.

**ID:** TC12

**Title:** Delete Ticket

**Description:** A logged in member should be delete an existing ticket from the list of tickets.

**Precondition:** The user must be logged into the system and list of tickets should be visible.

**Assumption:** User has required privilege.

**Test Steps:**

1. Click on the **Delete** link under **Actions** column in the list of tickets.
2. A popup should appear showing the details of choice for delete.
3. Confirm to delete the record.
4. A message showing successful deletion should appear on the screen.
5. Selected entry from the list should disappear.

**Expected Result:** Selected entry for deletion should disappear from the list of tickets.

**Debugging:** Requires code level changes.

**ID:** TC13

**Title:** Settings Page

**Description:** A logged in member should be able to update their credentials.

**Precondition:** The user must be logged into the system.

**Assumption:** User has required privilege.

**Test Steps:**

1. Click on the **Settings** on the dashboard icons or click on the **Settings** in the navigation bar.
2. Change values of **username**, **old password**, **and new password** and **confirm password**.
3. The value of can be null on both confirm or new password if the user wishes to reset the username only.
4. Click on **Update** to update the credentials.
5. A message showing successful update should appear on the screen.

**Expected Result:** The user should use new credential for login.

**Debugging:** Requires code level changes.

**ID:** TC14

**Title:** Users Page

**Description:** A logged in member with agency admin role should be able to see a list of users.

**Precondition:** The user must be logged into the system and must have agency admin role.

**Assumption:** User has required privilege and User's list has been placed earlier by same user.

**Test Steps:**

1. Click on the **username** on the upper right corner.
2. Click on **Users** link.
3. A list of current agency member should be visible.

**Expected Result:** A list of current agency member should appear.

**Debugging:** If no users, add new record to the list. If issue persists, requires code level changes.

**ID:** TC15

**Title:** Add User Page

**Description:** A logged in member should be able add new user.

**Precondition:** The user must be logged into the system and must have agency admin role.

**Assumption:** User has required privilege.

**Test Steps:**

1. Click on the **Add New** button just above the list of users.
2. Set **name**, **email**, **phone**, **address**, **username** and **passwords** in the text boxes.
3. Click on **Add**.
4. A message showing successful addition should appear on the screen.

**Expected Result:** A new entry should be visible on the list of users.

**Debugging:** Requires code level changes.

**ID:** TC16

**Title:** Edit User Page

**Description:** A logged in member should be able edit user.

**Precondition:** The user must be logged into the system and must have agency admin role.

**Assumption:** User has required privilege.

**Test Steps:**

1. Click on the **Edit** link under **Actions** column in the list of users.
2. Change the values of **name**, **email**, **phone**, **address**, **username** and **password** in the text boxes.
3. Click on **Update**.
4. A message showing successful update should appear on the screen.

**Expected Result:** Updated values should be visible on the list of users.

**Debugging:** Requires code level changes.

**ID:** TC17

**Title:** Delete User

**Description:** A logged in member should be delete an existing user from the list of users.

**Precondition:** The user must be logged into the system and must have agency admin role.

**Assumption:** User has required privilege.

**Test Steps:**

1. Click on the **Delete** link under **Actions** column in the list of users.
2. A popup should appear showing the details of choice for delete.
3. Confirm to delete the record.
4. A message showing successful deletion should appear on the screen.
5. Selected entry from the list should disappear.

**Expected Result:** Selected entry for deletion should disappear from the list of users.

**Debugging:** Requires code level changes.

**ID:** TC18

**Title:** Admin Home Page

**Description:** A logged in member should be able to 4 icons (Manage Slider, Places, Travel Agencies and Settings) in the dashboard.

**Precondition:** The user must be logged into the system and must have admin access.

**Assumption:** User has required privilege.

**Test Steps:**

1. Check 4 icons (Manage Slider, Places, Travel Agencies and Settings) in the dashboard.

**Expected Result:** A page displaying 4 icons (Manage Slider, Places, Travel Agencies and Settings) in the dashboard should appear on the browser.

**Debugging:** Check user privilege in the database.

**ID:** TC19

**Title:** Slider Gallery Page

**Description:** A logged in admin should be able to view slider images.

**Precondition:** The user must be logged into the system and must have admin access.

**Assumption:** User has required privilege.

**Test Steps:**

1. Click on **Manage Slider** Icon in the dashboard or **Manage Slider** link in the navigation bar.

**Expected Result:** List of uploaded images should be visible in the following page.

**Debugging:** Check user privilege in the database and also list of image file in the folder.

**ID:** TC20

**Title:** Upload Image Page

**Description:** A logged in admin should be able to upload new slider images.

**Precondition:** The user must be logged into the system and must have admin access.

**Assumption:** User has required privilege.

**Test Steps:**

1. Click on **Upload Image** button in the slider gallery page.
2. Then click on **Browse** button on next page and choose picture from the local disk.
3. Preview of image should be visible.
4. Click on **Add** button to save upload and save the image.
5. A message with successful upload should be visible.

**Expected Result:** New image should be visible in the slider gallery.

**Debugging:** Check if the images folder is writable.

**ID:** TC21

**Title:** Showing Image In Public Slider

**Description:** A logged in admin should be able to show slider images to public user.

**Precondition:** The user must be logged in to the system and must have admin access.

**Assumption:** User has required privilege.

**Test Steps:**

1. Click on the preview of the image in the slider gallery page.
2. Then click on the image on next page.
3. Select the region of the image that needs be visible in slider.
4. Click on **Crop & Save** button.
5. A success message should be visible.

**Expected Result:** Cropped section of the image should be visible in the public slider and also in **Active Slider** section.

**Debugging:** Check if the images folder is writable and GD library is enabled in the application server.

**ID:** TC22

**Title:** Deleting Image from the Slider

**Description:** A logged in admin should be able to remove slider images.

**Precondition:** The user must be logged into the system and must have admin access.

**Assumption:** User has required privilege.

**Test Steps:**

1. Click on the preview of the image in the slider gallery page.
2. Then click on the image on next page.
3. Click on **Delete Image** button to delete the image.
4. Confirm on the delete action.
5. A success message should be visible.

**Expected Result:** Deleted image should not be visible in **Slider Gallery** or **Active Slider** section.

**Debugging:** Check if the image folder is writable.

**ID:** TC23

**Title:** Places Page

**Description:** A logged in admin should be able to see the list of places.

**Precondition:** The user must be logged into the system and must have admin access.

**Assumption:** User has required privileges and places list has been placed earlier by same or similar user.

**Test Steps:**

1. Click on the **Places** icon in the dashboard or **Places** link in the navigation bar.

**Expected Result:** A list of places should appear.

**Debugging:** If empty, add new record to the list. If issue persists, requires code level changes.

**ID:** TC24

**Title:** Add Places Page

**Description:** A logged in admin should be able to add new place to the list.

**Precondition:** The user must be logged into the system and must have admin access.

**Assumption:** User has required privilege.

**Test Steps:**

1. Click on the **Add New** button just above the list of buses.
2. Set **name** of the place in the text boxes.
3. Click on **Add**.
4. A message showing successful addition should appear on the screen.

**Expected Result:** A new entry should be visible on the list of places.

**Debugging:** Requires code level changes.

**ID:** TC25

**Title:** Edit Places Page

**Description:** A logged in admin should be able to edit a place from the list of places.

**Precondition:** The user must be logged into the system and must have admin access.

**Assumption:** User has required privilege.

**Test Steps:**

1. Click on the **Edit** link under **Actions** column in the list of places.
2. Change the values of name in the text boxes.
3. Click on **Update**.
4. A message showing successful update should appear on the screen.

**Expected Result:** Updated values should be visible on the list of places.

**Debugging:** Requires code level changes.

**ID:** TC26

**Title:** Delete Places

**Description:** A logged in admin should be able to delete a place from the list of places.

**Precondition:** The user must be logged into the system and must have admin access.

**Assumption:** User has required privilege.

**Test Steps:**

1. Click on the **Delete** link under **Actions** column in the list of places.
2. A popup should appear showing the details of choice for delete.
3. Confirm to delete the record.
4. A message showing successful deletion should appear on the screen.
5. Selected entry from the list should disappear.

**Expected Result:** Selected entry for deletion should disappear from the list of places.

**Debugging:** Requires code level changes.

**ID:** TC27

**Title:** Travel Agencies Page

**Description:** A logged in admin should be able to see the list of travel agencies.

**Precondition:** The user must be logged into the system and must have admin access.

**Assumption:** User has required privileges and list of travel agencies has been placed earlier by same or similar user.

**Test Steps:**

1. Click on the **Travel Agencies** icon in the dashboard or **Travel Agencies** link in the navigation bar.

**Expected Result:** A list of travel agencies should appear.

**Debugging:** If no records, add new record to the list. If issue persists, requires code level changes.

**ID:** TC28

**Title:** Add Travel Agency Page

**Description:** A logged in admin should be able to add new travel agency to the list.

**Precondition:** The user must be logged into the system and must have admin access.

**Assumption:** User has required privilege.

**Test Steps:**

1. Click on the **Add New** button just above the list of agencies.
2. Set **Agency Name**, **Address**, **Contact**, **and Email** in the text boxes.
3. Click on **Add**.
4. A message showing successful addition should appear on the screen.

**Expected Result:** A new entry should be visible on the list of travel agencies.

**Debugging:** Requires code level changes.

**ID:** TC29

**Title:** Edit Travel Agency Page

**Description:** A logged in admin should be able to edit a place from the list of travel agencies.

**Precondition:** The user must be logged into the system and must have admin access.

**Assumption:** User has required privilege.

**Test Steps:**

1. Click on the **Edit** link under **Actions** column in the list of travel agencies.
2. Change the values of **Agency Name**, **Address**, **Contact**, **and Email** in the text boxes.
3. Click on **Update**.
4. A message showing successful update should appear on the screen.

**Expected Result:** Updated values should be visible on the list of travel agencies.

**Debugging:** Requires code level changes.

**ID:** TC30

**Title:** Delete Travel Agency

**Description:** A logged in admin should be able to delete a travel agency from the list of travel agencies.

**Precondition:** The user must be logged into the system and must have admin access.

**Assumption:** User has required privilege and list of travel agencies is also available.

**Test Steps:**

1. Click on the **Delete** link under **Actions** column in the list of travel agencies.
2. A popup should appear showing the details of choice for delete.
3. Confirm to delete the record.
4. A message showing successful deletion should appear on the screen.
5. Selected entry from the list should disappear.

**Expected Result:** Selected entry for deletion should disappear from the list of travel agencies.

**Debugging:** Requires code level changes.

**ID:** TC31

**Title:** Agency Admin Page

**Description:** A logged in admin should be able to see a list of users.

**Precondition:** The user must be logged into the system and must have admin access.

**Assumption:** User has required privilege and User's list has been placed earlier by same or similar user.

**Test Steps:**

1. Click on the **Travel Agencies** on the dashboard or from the navigation bar.
2. Click on **Agency Admins** link.
3. A list of current agency admins should be visible.

**Expected Result:** A list of current agency admins should appear.

**Debugging:** If no users, add new record to the list. If issue persists, requires code level changes.

**ID:** TC32

**Title:** Add Agency Admin Page

**Description:** A logged in admin should be able add new user.

**Precondition:** The user must be logged into the system and must have admin access.

**Assumption:** User has required privilege.

**Test Steps:**

1. Click on the **Add New** button just above the list of users.
2. Set **name**, **email**, **phone**, **address**, **username**, **password** in the text boxes and select value from **travel agency** dropdown.
3. Click on **Add**.
4. A message showing successful addition should appear on the screen.

**Expected Result:** A new entry should be visible on the list of users.

**Debugging:** Requires code level changes.

**ID:** TC33

**Title:** Edit Agency Admin Page

**Description:** A logged in admin should be able edit user.

**Precondition:** The user must be logged into the system and must have admin access.

**Assumption:** User has required privilege.

**Test Steps:**

1. Click on the **Edit** link under **Actions** column in the list of users.
2. Change the values of **name**, **email**, **phone**, **address**, **username**, **password** in the text boxes and select value from **travel agency** dropdown.
3. Click on **Update**.
4. A message showing successful update should appear on the screen.

**Expected Result:** Updated values should be visible on the list of users.

**Debugging:** Requires code level changes.

**ID:** TC34

**Title:** Delete Agency Admin

**Description:** A logged in admin should be delete an existing user from the list of users.

**Precondition:** The user must be logged into the system and must have admin access.

**Assumption:** User has required privilege.

**Test Steps:**

1. Click on the **Delete** link under **Actions** column in the list of users.
2. A popup should appear showing the details of choice for delete.
3. Confirm to delete the record.
4. A message showing successful deletion should appear on the screen.
5. Selected entry from the list should disappear.

**Expected Result:** Selected entry for deletion should disappear from the list of users.

**Debugging:** Requires code level changes.

**ID:** TC35

**Title:** Settings Page

**Description:** A logged in admin should be able to update its credentials.

**Precondition:** The user must be logged into the system.

**Assumption:** User has required privilege.

**Test Steps:**

1. Click on the **Settings** on the dashboard icons or click on the **Settings** in the navigation bar.
2. Change values of **username**, **old password**, **and new password** and **confirm password**.
3. The value of can be null on both confirm or new password if the user wishes to reset the username only.
4. Click on **Update** to update the credentials.
5. A message showing successful update should appear on the screen.

**Expected Result:** The user should use new credential for login.

**Debugging:** Requires code level changes.

**ID:** TC36

**Title:** Bus Searching

**Description:** A normal user should be able to search a bus with a given date and destination.

**Precondition:** Bus should be available for that route.

**Assumption:** Places entry should be in the database.

**Test Steps:**

1. Open browser and hit **http://localhost/bus\_reservation/**
2. Choose and set values for **From** and **To** dropdown list
3. Give travel date as an input.
4. A list of available bus should be shown below right below the search panel.

**Expected Result:** List of available bus should be shown based on date and routes.

**Debugging:** Requires code level changes.

**ID:** TC37

**Title:** Seat Selection

**Description:** A normal user should be able to select seat of a bus.

**Precondition:** Bus should have available seats.

**Assumption:** Bus is available for that route..

**Test Steps:**

1. Open browser and hit **http://localhost/bus\_reservation/**
2. Choose and set values for **From** and **To** dropdown list
3. Give travel date as an input.
4. A list of available bus should be shown below right below the search panel.
5. Click on the bus entry.
6. On next page, select the seats of your choice.
7. Press **Next** button

**Expected Result:** Selected seat is correctly recognized. Also, **bus details** section should be updating with new values upon selecting or deselecting seat.

**Debugging:** Requires code level changes.

**ID:** TC38

**Title:** Passenger Details Page

**Description:** User should be able to provide information about the passenger.

**Precondition:** Seats must be selected in previous page.

**Assumption:** Information provided by the user is authentic.

**Test Steps:**

1. Set values of **first name**, **last name**, **address**, **contact** and **email**.
2. Click on **Submit** to book the ticket.
3. Clicking on **Cancel** button will take user back to previous page.

**Expected Result:** The user should be able to see the ticket booked.

**Debugging:** Requires code level changes.

**ID:** TC39

**Title:** Ticket Page and Ticket Download

**Description:** User should be able to see the ticket booked.

**Precondition:** Ticket needs to be booked by the user.

**Assumption:** Unique ticket code is genuinely unique.

**Test Steps:**

1. Paste unique ticket code in the search box at the navigation bar of the webpage.
2. Press enter. The user should be redirected to ticket details page.
3. Clicking on **PDF** button will download the ticket in PDF format.

**Expected Result:** The user should be able to see the ticket booked and should be able to download the ticket.

**Debugging:** Requires code level changes.

**ID:** TC40

**Title:** Travel Agencies

**Description:** User should be able to see the list of available travel agencies.

**Precondition:** List of travel agencies must be present in the database.

**Assumption:** N/A.

**Test Steps:**

1. Click on **Travel Agencies** link in the navigation bar.
2. A list of all the registered Travel Agencies will appear.

**Expected Result:** List of registered travel agencies along with their contact details.

**Debugging:** If no records, admin should register new travel agencies. If the issue persists may require code changes.

**5.3.2 Summary**

During this project, every component was tested by using the test cases mentioned above. There were a lot of issue while testing. But they were resolved after fixing the codes. The type of testing done in this project is mainly **Module-level Testing** (during development phase) and **Smoke Testing** (after development and module-level testing). All tests came positive for given set of inputs.

## **5.4 Implementation & Result**

### **5.4.1 Implementation**

After development and testing, this is the final stage of this project. For actual implementation of the project, at least, following tools and technology must be used.

* **Hardware requirements**
  + **Processor:** Pentium IV or greater.
  + **Operating System:** any OS (32 bit or 64 bit)
  + **RAM:** at least 512 MB or greater
  + **HDD:** at least 200 MB for project deployment only
* **Software requirements**
  + **Database Server:** MySQL Ver. 5.1.36 or greater (for data storage)
  + **Application Server:** Apache 2.4.9 and PHP 5.5.12 or greater (for application deployment)
  + **Browser:** any modern browser (user’s choice)

### **5.4.2 Result**

As mentioned in the **Requirement Specification** section, this project has tried to meet all of its functional as well as non functional requirements. This project has tried to bring different bus agency into a single platform. This has been achieved by providing separate panel for different agency admins and member of that agency within this system.

## **5.5 Maintenance**

Like any other software project, this project also may contain unseen bugs and issue that may not have been covered by our testing methods. There is always chance of some issue to come over time. Since, we are already aware of such unseen possible issues, an OOP language has been used to build this project to help prevent the hassle of doing maintenance task on the system. A simple but powerful framework will allow developers to the fix issues much faster compared to custom codes develop.

Not only that, the framework chosen is so easy to use that adding other enhancement to the system will be faster and easier. Thus, use of codeigniter framework has made it easy for the current developers as well as future developers to add new feature and fix issues within this system.

# **Chapter 6: Conclusion and Future Enhancement**

## **6.1 Conclusion**

Based on the **Literature Review,** this type of system has already been created by other individuals prior this project. But it tries to solve one of the major problems. That problem is easy availability of buses for journey. Compared to similar systems available on the market, this system tires its best to link various bus agencies into a single system. As mentioned earlier (See **System Analysis** for more details), this system provides facility to browse bus, select seat of your desire and finally pay for the booked seats. Further, it allows user to search for their ticket using the unique ticket id provided during ticket purchase. This allows the user and the bus agency to cross check the booking details.

## **6.2 Future Enhancement**

The following list of enhancements can be done to this project in the future:

* Adding support for mobile devices.
* Provide online payment system such as e-sewa and i-pay.
* Ability for bus agency to add their own type of seat layout.
* News and update section where people can get details about new buses and routes.

*The source codes of this project can be found at my github account: https://github.com/vinayadahal/bus\_reservation. Further development task will be carried out using this repository.*

# **References**

[1] Eckerson, Wayne W. "Three Tier Client/Server Architecture: Achieving Scalability, Performance, and Efficiency in Client Server Applications." Open Information Systems 10, 1 (January 1995): 3(20)

[2] Buschmann, Frank; Meunier, Regine; Rohnert, Hans; Sommerlad, Peter; Stal, Michael (1996-08). Pattern-Oriented Software Architecture, Volume 1, A System of Patterns. Wiley, August 1996. ISBN 978-0-471-95869-7. Retrieved from <http://www.wiley.com/WileyCDA/WileyTitle/productCd-0471958697.html>.

[3] Pigoski, Thomas M., 1997: Practical software maintenance: Best practices for managing your software investment. Wiley Computer Pub. (New York)

[4] Lientz B., Swanson E., 1980: Software Maintenance Management. Addison Wesley, Reading, MA

[5] Sarah, Geagea; Sheng, Zhang; Niclas,Sahlin; Faegheh, Hasibi; Farhan, Hameed; Elmira, Rafiyan; Magnus, Ekberg; (2010). Software Requirements Specification. Retrieved from <http://www.cse.chalmers.se/~feldt/courses/reqeng/examples/srs_example_2010_group2.pdf>

[6] Lerdorf, Rasmus (2007-04-26). "PHP on Hormones – history of PHP presentation by Rasmus Lerdorf given at the MySQL Conference in Santa Clara, California". The Conversations Network.

[7] "Rasmus Lerdorf, Senior Technical Yahoo: PHP, Behind the Mic". 2003-11-19. Archived from the original on 2013-07-28., cited at Felipe Ribeiro (Aug 7, 2012). "This Is Not Another "PHP Sucks" Article".

[8] Pawel Krawczyk (2013). "Most common attacks on web applications". IPSec.pl.

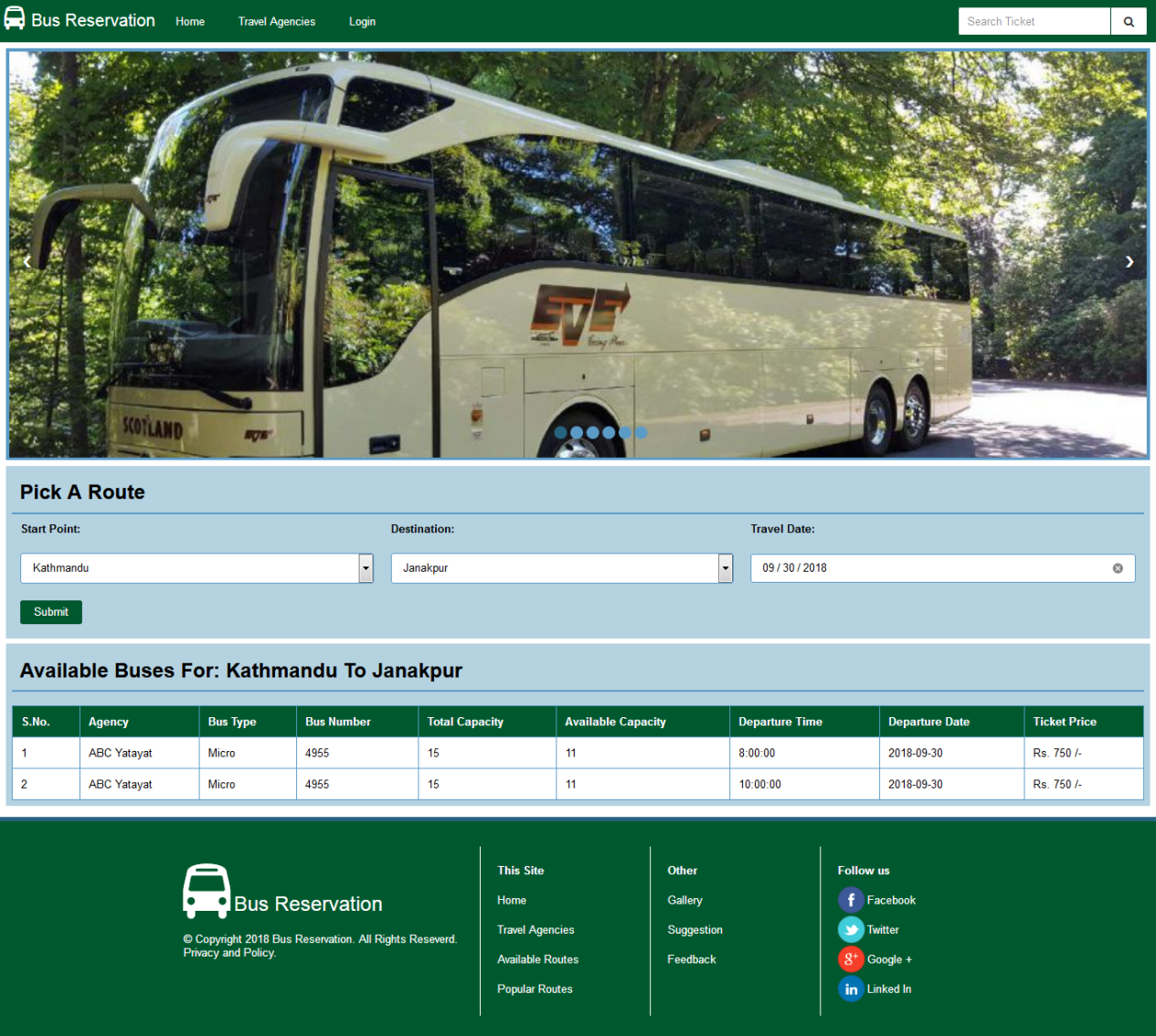
[9] Golemon, Sara (2006). Extending and Embedding PHP. ISBN 978-0-672-32704-9.

[10] How to show foreign key, https://www.quora.com/How-do-I-show-attributes-with-a-foreign-key-in-ER-diagrams, accessed on 2018/09/25

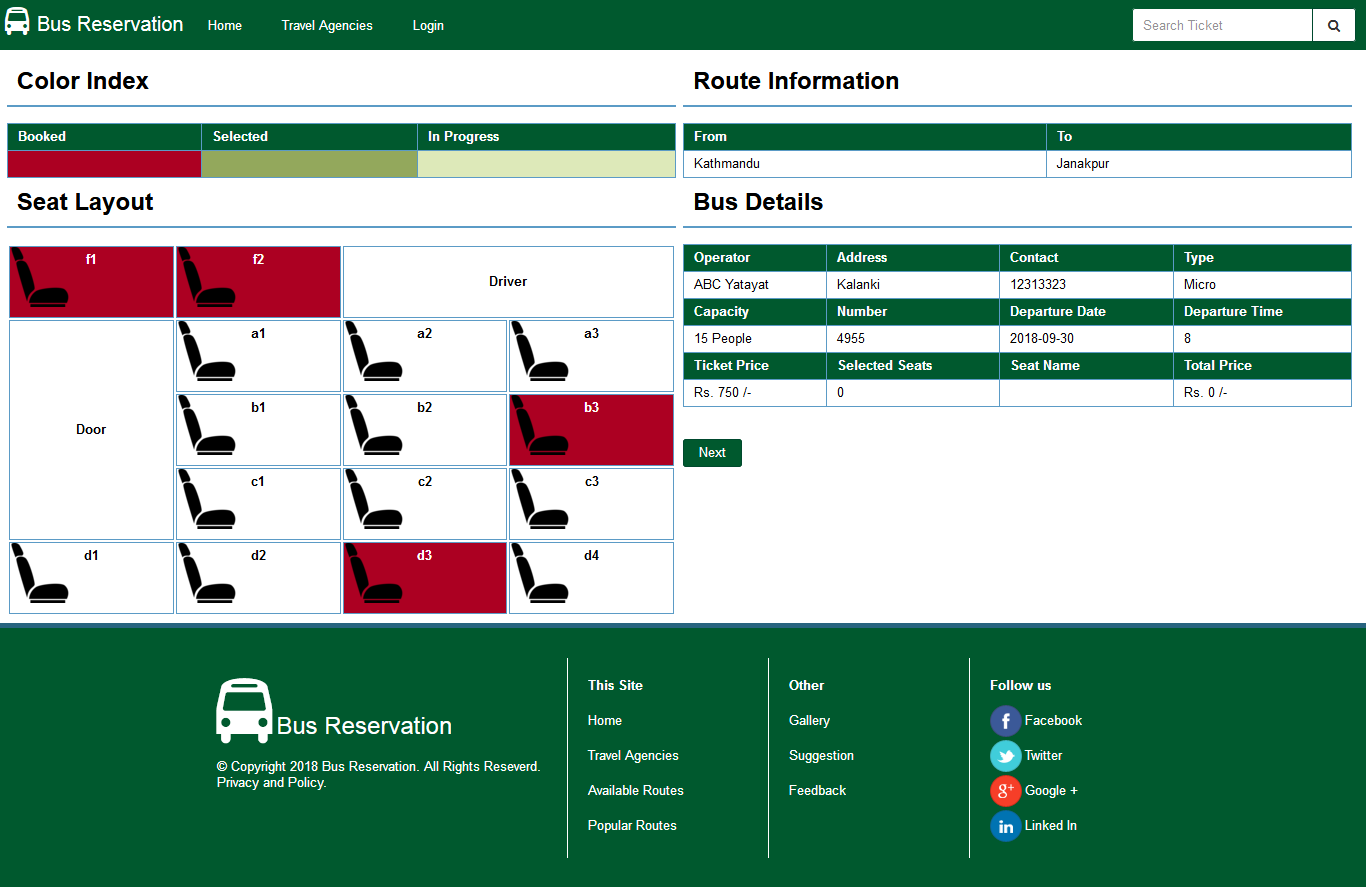
[11] How to write test cases, https://blog.testlodge.com/how-to-write-test-cases-for-software-with-sample/, accessed in 2018/09/26

# **Appendices**

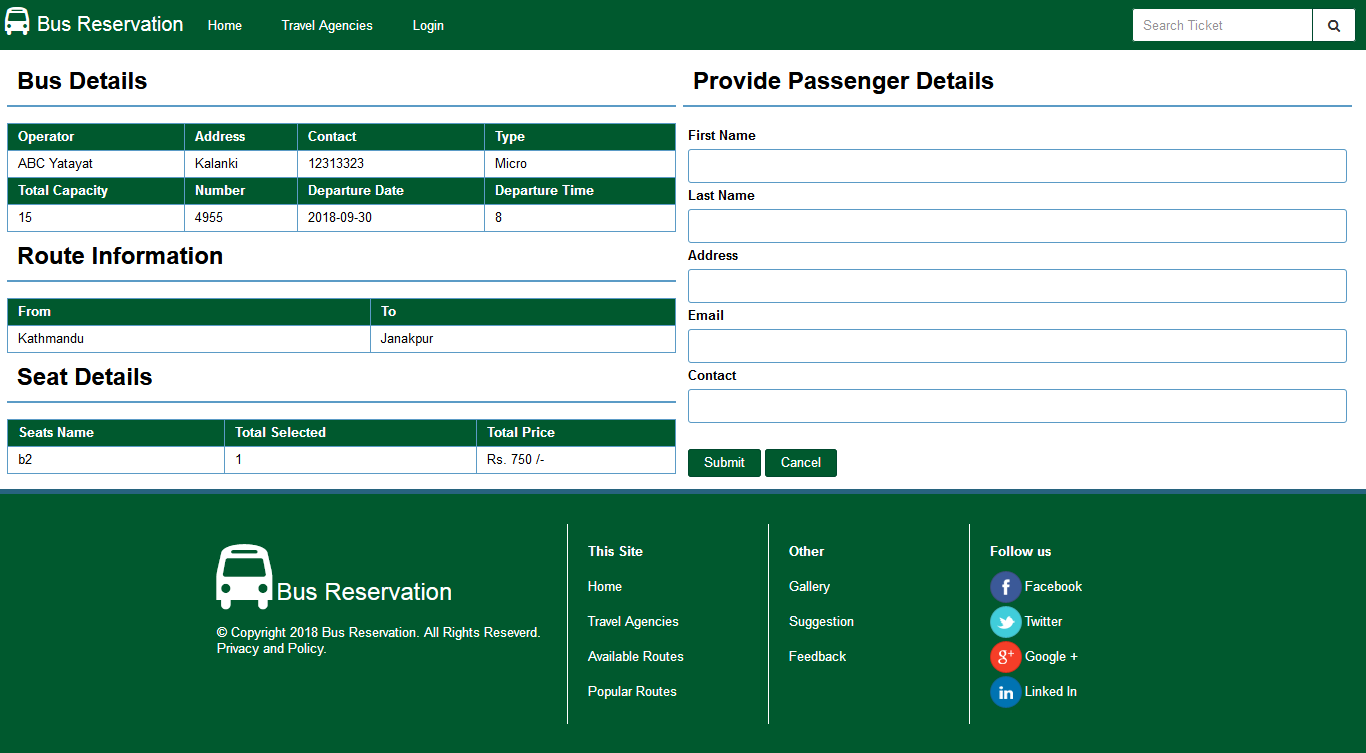
## **Screenshots**



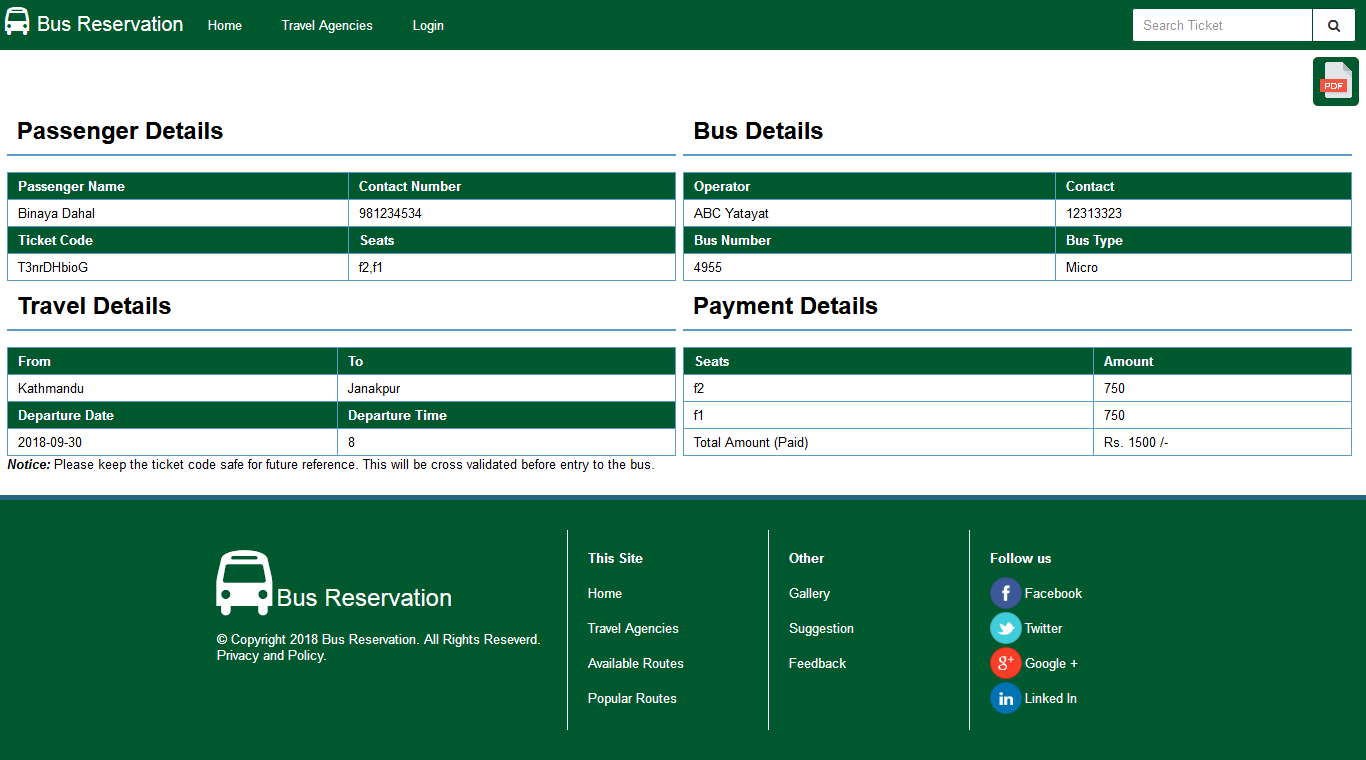
Screenshot 1: Home Page - Users



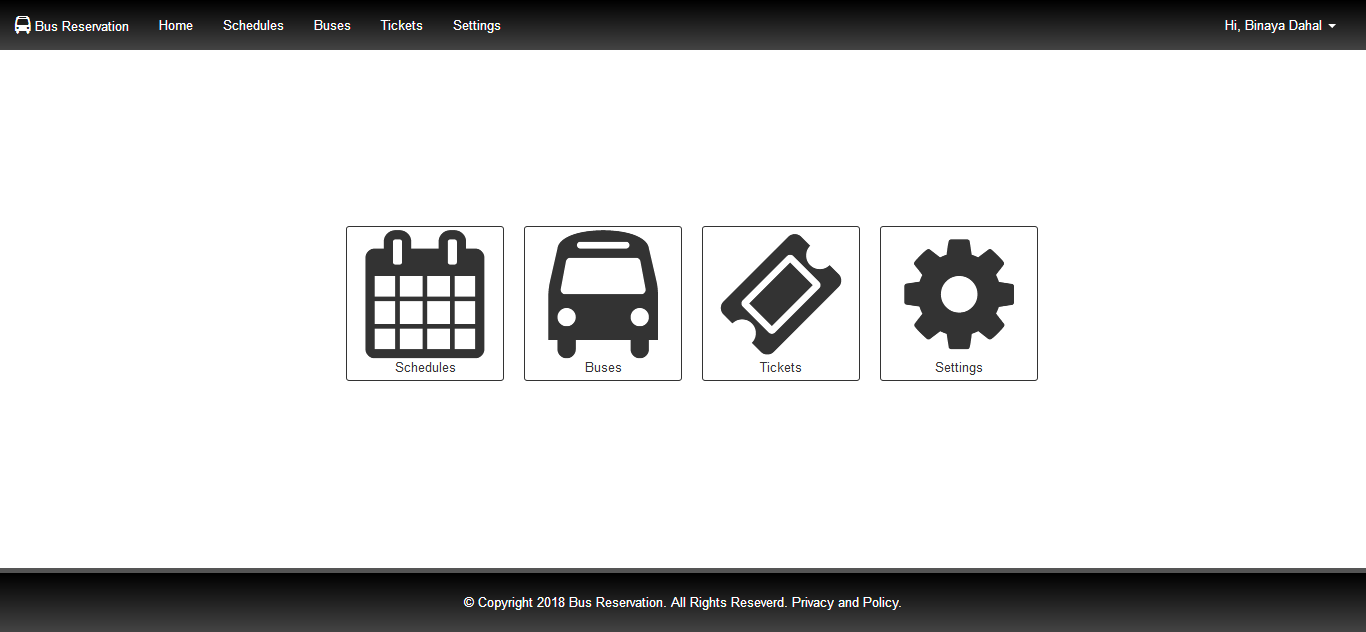
Screenshot 2: Choose Seats - User



Screenshot 3: Seat Booking - User



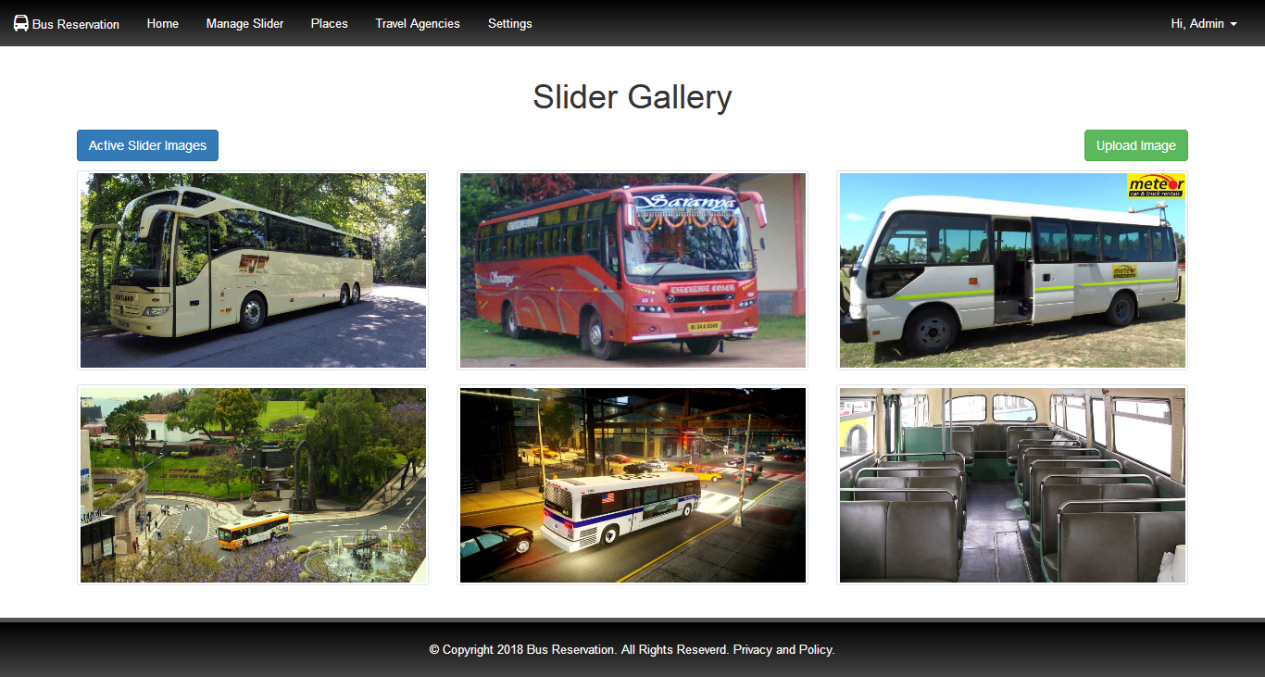
Screenshot 4: Tickets - User



Screenshot 5: Home Page - Member



Screenshot 6: Home Page - Admin



Screenshot 7: Silder Gallery - Admin



Screenshot 8: Slider Image Editor - Admin