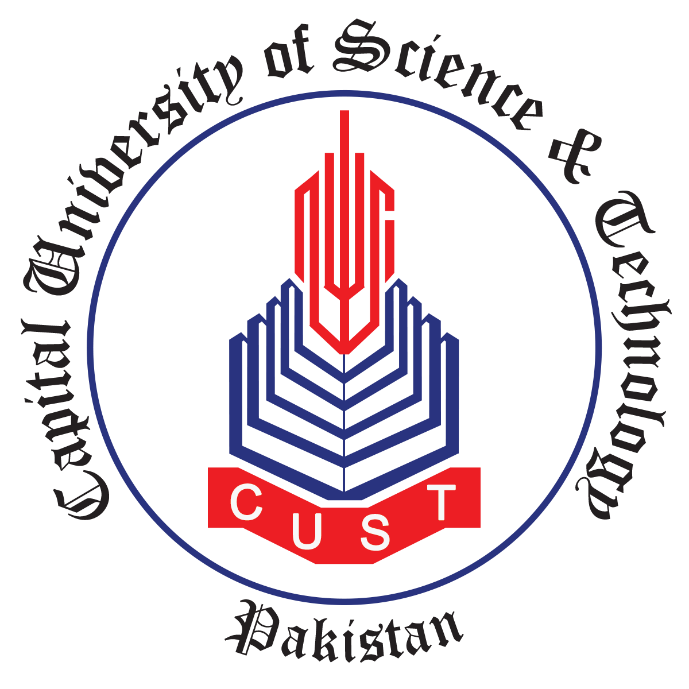
Wedding Vibes v.1.0.

Muhammad Ali BCS151043

Raja Shujaat BCS151049

Bilal Nisar BCS151062



Spring-2018

Department of Computer Science

Capital University of Science & Technology, Islamabad

# 

Submission Form for Final-Year

# PROJECT REPORT

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | **Version** | V 1.0 | | |  |  | | --- | --- | | **NUMBER OF MEMBERS** | 3 | |

|  |  |
| --- | --- |
| **TITLE** | Wedding Vibes Android Based Solution |

|  |  |
| --- | --- |
| **SUPERVISOR NAME** | Dr. Aamer Nadeem **Sir Amir Nadeem** |

|  |  |  |
| --- | --- | --- |
| **MEMBER NAME** | **REG. NO.** | **EMAIL ADDRESS** |
| Muhammad Ali | BCS 151043 | [alich4188@gmail.com](mailto:alich4188@gmail.com) |
| Raja Shujaat | BCS 151049 | [sheeje0786@gmail.com](mailto:sheeje0786@gmail.com) |
| Bilal Nisar | BCS 151062 | [bilalnisar32@yahoo.com](mailto:bilalnisar32@yahoo.com) |

**MEMBERS’ SIGNATURES**

**Supervisor’s Signature**

*Note 1: This paper must be signed by your supervisor*

**Table of Contents**

[i](#_Toc512872919)

[PROJECT REPORT i](#_Toc512872920)

[Chapter 1 1](#_Toc512872921)

[Introduction 1](#_Toc512872922)

[1.1. Project Introduction 1](#_Toc512872923)

[1.2. Existing Examples / Solutions 1](#_Toc512872924)

[1.3. Business Scope 2](#_Toc512872925)

[1.4. Useful Tools and Technologies 3](#_Toc512872926)

[1.5. Project Work Break Down 4](#_Toc512872927)

[1.6. Project TimeLine 5](#_Toc512872928)

[Chapter 2 6](#_Toc512872929)

[Requirement Specification and Analysis 6](#_Toc512872930)

[2.1. Functional Requirements 6](#_Toc512872931)

[2.2. Non-Functional Requirements 8](#_Toc512872932)

[2.3. Selected Functional Requirements 9](#_Toc512872933)

[2.4. System Use Case Modeling 9](#_Toc512872934)

[2.4.1. Use Case Diagram 9](#_Toc512872935)

[2.4.2. Use Case Description 11](#_Toc512872936)

[2.5. System Sequence Diagram 16](#_Toc512872937)

[2.6. Domain Model 20](#_Toc512872938)

[Chapter 3 22](#_Toc512872939)

[System Design 22](#_Toc512872940)

[3.1. Layer Definition 22](#_Toc512872941)

[3.1.1. Presentation Layer 22](#_Toc512872942)

[3.1.2. Business Layer 22](#_Toc512872943)

[3.1.3. Database Layer 22](#_Toc512872944)

[3.2. Class Diagram 22](#_Toc512872945)

[3.3. Sequence Diagram 23](#_Toc512872946)

[3.4. Entity Relationship Diagram 26](#_Toc512872947)

**List of Tables**

[Table 1.1 Comparison between Existing and Proposed System 2](#_Toc513030510)

[Table 2.1: Functional Requirements 7](#_Toc513030511)

**List of Figures**

[Figure 1.1 Project Breakdown 4](#_Toc512872035)

[Figure 1.2 Project Timeline 5](#_Toc512872036)

[Figure 3.3 signup 24](#_Toc512872037)

\

# Chapter 1

# Introduction

Wedding Vibes System will provide a platform for customers to make online reservation in Fortress on desire dates. It will be time saving for users to book marquee online.

## Project Introduction

Wedding Vibes System is an online platform which facilitates the process of booking marquee and it’s all services. Wedding Vibes system is providing fully automated wedding ceremony from reservation to billing. Wedding Vibes system provides online scheduling of dates to book online.

## Existing Examples / Solutions

There are applications that provide such functionalities:

* **Haveli** through its customized services ensures that we are able to reflect your lifestyles and personalities in to your wedding and other social events. URL: <http://www.haveli.pk/>
* The Paradise Complex provides an online application for reservation of Marquee and Halls. URL to the site is: <http://theparadise.pk/>

Table 1.1 Comparison between Existing and Proposed System

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Sr. No** | **Characteristics** | **Haveli** | **The Paradise Complex** | **Wedding Vibes System (Proposed System)** |
| **1** | Sign Up/Registration |  |  |  |
| **2** | Sign In |  |  |  |
| **3** | Menu Selection |  |  |  |
| **4** | Customize Menu Selection |  |  |  |
| **5** | Review/Users Feedback |  |  |  |
| **6** | Scheduling Dates |  |  |  |
| **7** | Reservation |  |  |  |
| **8** | Packages Booking |  |  |  |
| **9** | Client Authentication |  |  |  |
| **10** | Bill Payment |  |  |  |

## Business Scope

Customers face many difficulties during reserving and booking events. They have to come and visit the manager office to gather information regarding the services provided by Fortress while our system will provide all this information online. They have to visit manager office to check whether their desired dates for their events are available or not while our system will provide this service online. Wedding Vibes System is a platform which facilitates the process of booking marquee and different packages of marquee through online platform.

## Useful Tools and Technologies

Following are the list of all the possible technologies that will be required during the designing, developmental and testing phases of the project:

* **Programming Language:**
  + **C#** will be used at backend for web application.
  + **HTML5, CSS3** will be used at frontend for web application.

* **Development Environment:**
  + **Visual Studio:** It will be used for the development, debugging and simulation of the webserver.
* **Database:**
  + **SQL Server:** It will be used to store data, as it is compatible with Microsoft products.

## Project Work Break Down

Figure 1.1 Project Breakdown

## 1.6. Project TimeLine

4/25/18

5/30/18

Testing

Implementation

Design

4/2/18

Requirement Analysis

3/4/18

Requirement Gathering

Figure 1.2 Project Timeline

# Chapter 2

# Requirement Specification and Analysis

Requirements analysis is a process of determining user expectations for a new or modified product. These features, called requirements, must be quantifiable, relevant and detailed. In software engineering, such requirements are often called functional specifications. In Chapter 2 we will enlist the functional and non-functional requirements and model functional requirements in the form of use case model.

## Functional Requirements

A functional requirement defines a function of a system or its component. We have gathered these functional requirements from the Ed for which we are developing this product. Here we mentioned all the requirements of system.

Given below is a list of functional requirements of the proposed system.

Table 1.1: Functional Requirements

|  |  |  |  |
| --- | --- | --- | --- |
| **S. No.** | **Functional Requirement** | **Type** | **Status** |
| 1 | The system must be able to manage customers wedding reservation. | Core | Pending |
| 2 | User can Sign up. |  |  |
| 3 | User can Sign in. | Core | Pending |
| 4 | System will find the availability of dates. | Core | Pending |
| 5 | User can select Desired menu. | Core | Pending |
| 6 | Customer can cancel their booking from their account until specific date. | Core | Pending |
| 7 | User can book online and pay compensation money. | intermediate | Pending |
| 8 | System must send registration confirmation email. | Core | Pending |
| 9 | Admin able to edit customers booking information. | Core | Pending |
| 10 | Customer can send feedback. | Core | Pending |
| 11 | User can select additional services (Music, Appetizer, Decor). | Core | Pending |
| 12 | User can select additional services (Music, Appetizer, Decor). | Core | Pending |
| 13 | System should send notification to admin about booked date. | Core | Pending |
| 14 | Admin can view statistics of customers booking. | intermediate | Pending |
| 15 | Admin able to give discount to their loyal customers. | Core | Pending |

## Non-Functional Requirements

A non-functional requirement is a requirement that specifies criteria that can be used to judge the operation of a system, rather than specific behaviors. There are some non-functional requirements for our system

Given below is a list of non-functional requirements of the proposed system.

Table 2.2: Non-Functional Requirements

|  |  |  |
| --- | --- | --- |
| **S. No.** | **Non-Functional Requirements** | **Category** |
| 1 | The system should authenticate all users | Security |
| 2 | System should give help for signing up and entering password/username | Usability |
| 3 | Authorization and authentication should be done in secure connection. | Security |
| 4 | The system should be able to handle multiple transactions at a time. | Reliability |
| 5 | Customer need to cancel the booking before 24hours of event otherwise their compensation money will be charged. | Reliability |

## Selected Functional Requirements

Given below is a list of selected functional requirements of Process Flow Manager v.1.0.

Table 2.3: Selected Functional Requirements

|  |  |  |
| --- | --- | --- |
| **S. No.** | **Functional Requirement** | **Type** |
| 1 | User can Sign up. | New |
| 2 | User can sign in. | New |
| 3 | The system must be able to manage customers wedding reservation. | New |
| 4 | System will find the availability of dates. | New |
| 5 | User can select Desired menu. | New |

## System Use Case Modeling

## Use Case Diagram

Given below is the use case diagram of Process Flow Manager v.1.0.

A use case is a list of actions or event steps, typically defining the interactions between a role (known in the Unified Modeling Language as an actor) and a system, to achieve a goal. The use cases are given below

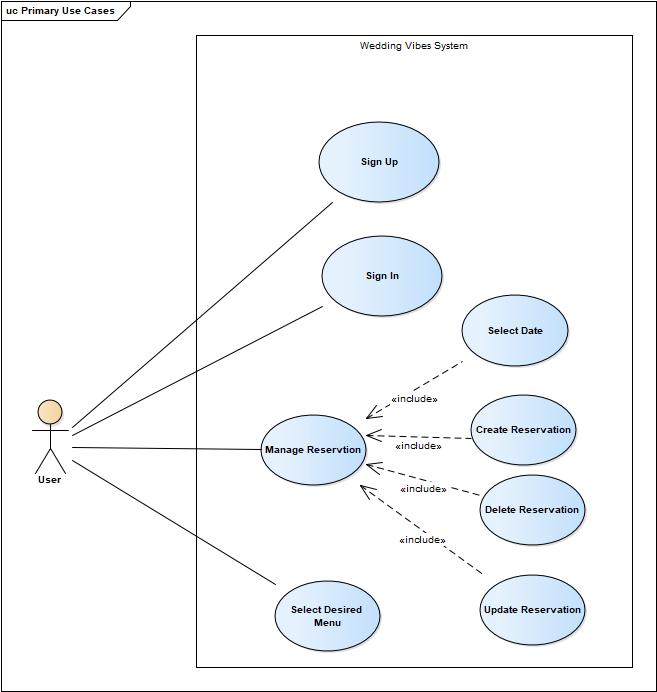


Figure 2.1: Use Case Diagram v.1.0.

## Use Case Description

Given below are the use case descriptions of Process Flow Manager v.1.0.

Table 2.4: Signup

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Use Case ID:** | Uc1 | | | |
| **Use Case Name:** | Signup | | | |
| **Created By:** | Muhammad Ali | Last Updated By: | | Muhammad Ali |
| **Date Created:** | 15/3/2018 | Last Revision Date: | | 29/3/2018 |
| **Actors:** | User | | | |
| **Description:** | The user can sign up by the first time he/she uses the system by providing a name, password, address, email and mobile number. | | | |
| **Trigger:** | Signup button | | | |
| **Preconditions:** | N/A | | | |
| **Post conditions:** | User will be signed up and able to use the system. | | | |
| **Normal Flow:** | User | | System | |
| 1: User clicks signup button to request for sign up | | 2: The system provides  User sign-up form. | |
|  | 3: User fills in form by providing name, password, address, email and mobile number. | | 4: System signs up the User. | |
| **Alternative Flows:** | \*a User cancels the signup form. | | | |
| **Exceptions:** | 4a. The database is not responding.   1. Display Error Exception message, user info is remaining same in fields.  4b. The user has not filled the form correctly.  1. Display message for enter data in correct format. Checks handled  \*a. The system is not responding.  1. Show Exception message and redirect to previous page. | | | |

Table 2.5: sign in

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Use Case ID:** | **Uc2** | | | |
| **Use Case Name:** | Sign in | | | |
| **Created By:** | Bilal Nisar | Last Updated By: | | Muhammad Ali |
| **Date Created:** | 17/3/2018 | Last Revision Date: | | 29/3/2018 |
| **Actors:** | User | | | |
| **Description:** | Actor will sign into the system by providing email and password and click on sign in button | | | |
| **Trigger:** | Sign in button | | | |
| **Preconditions:** | Actor must be Registered. | | | |
| **Post conditions:** | Actor will be signed in and able to use the system. | | | |
| **Normal Flow:** | **User** | | **System** | |
| 1: Actor enter the email and password in required field and actor will click sign in button to request for sign in | | 2: The system will provide Customer sign in form. | |
|  | 3: Actor will fill form by providing his username, password. | | 4: System authenticates the actor and system sign in the actor. | |
| **Alternative Flows:** | 3a. user can enter wrong credential.  1. Go to step 3.  \*a. user will cancel the sign in form. | | | |
| **Exceptions:** | 4a. The database is not responding.   1. Display Error Exception message, user info is remaining same in fields.  4b. The user has not filled the form correctly.  1. Display message for enter data in correct format. Checks handled.  4c. Actor entered incorrect credentials  1. Display message of wrong credentials and stay on same page.  \*a. The system is not responding.  1. Show Exception message and redirect to previous page. | | | |

Table 2.6: Select Menu

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Use Case ID:** | **Uc3** | | | |
| **Use Case Name:** | Select date | | | |
| **Created By:** | Raja Shujaat | Last Updated By: | | Bilal Nisar |
| **Date Created:** | 17/10/2017 | Last Revision Date: | | 11/08/2017 |
| **Actors:** | User | | | |
| **Description:** | user click on select date button and search their desired date. System will display the available dates of Marriage Hall. user click on save form button. System saves the form into database. | | | |
| **Trigger:** | Select date | | | |
| **Preconditions:** | user must be signed in. | | | |
| **Post conditions:** | Form is saved into database. | | | |
| **Normal Flow:** | **User** | | **System** | |
| 1: user can click on select form button. | | 2: System display the search bar and available dates of Hall. | |
|  | 3: user click on save form button. | | 4: System saves the form into database. | |
| **Alternative Flows:** | user will cancel the current form. | | | |
| **Exceptions:** | 4a. The database is not responding.   1. Display Error Exception message, admin info is remaining same in fields.  \*a. The system is not responding.  1. Show Exception message and redirect to previous page. | | | |

Table 2.7: Select Desire Menu

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Use Case ID: | Uc4 | | | |
| **Use Case Name:** | Select desire Menu | | | |
| **Created By:** | Muhammed Ali | Last Updated By: | | Raja Shujaat |
| **Date Created:** | 17/10/2017 | Last Revision Date: | | 11/08/2017 |
| **Actors:** | User | | | |
| **Description:** | User click on Menu button. System display the menu form where user. System save the user information into database. | | | |
| **Trigger:** | Select menu | | | |
| **Preconditions:** | user must be signed in | | | |
| **Post conditions:** | User select the menu from options. | | | |
| **Normal Flow:** | **User** | | **System** | |
| 1: User will click add menu button to request for select their desired menu. | | 2: System will provide the form of add menu. | |
|  | 3: user enter the information of their desired menu. | | 4: System save the information of user in database | |
| **Alternative Flows:** | \*a. user will cancel the current form. | | | |
| **Exceptions:** | 4a. The database is not responding.   1. Display Error Exception message, admin info is remaining same in fields.  4b. System cannot save the information in database.  1. Display the message of Internet connection is not available.  \*a. System is not responding. | | | |

Table 2.8: Manage Reservation

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Use Case ID: | Uc5 | | | |
| **Use Case Name:** | Manage Reservation | | | |
| **Created By:** | Raja Shujaat | Last Updated By: | | Muhammed Ali |
| **Date Created:** | 17/10/2017 | Last Revision Date: | | 11/08/2017 |
| **Actors:** | User | | | |
| **Description:** | User click on reservation button. System display the Reservation form. where User can add, cancel or update their reservation and the System save the user information into database. | | | |
| **Trigger:** | Manage reservation | | | |
| **Preconditions:** | user must be signed in | | | |
| **Post conditions:** | User select the menu from options. | | | |
| **Normal Flow:** | **User** | | **System** | |
| 1: User will click reservation button to request for add, cancel or update their reservation. | | 2: System will provide the form of reservation. | |
|  | 3: user enter the information about their reservation. | | 4: System save the information of user in database | |
| **Alternative Flows:** | \*a. user will cancel the current form. | | | |
| **Exceptions:** | 4a. The database is not responding.   1. Display Error Exception message, admin info is remaining same in fields.  4b. System cannot save the information in database.  1. Display the message of Internet connection is not available.  \*a. System is not responding. | | | |

## System Sequence Diagram

Given below are the System Sequence Diagrams of Process Flow Manager v.1.0.

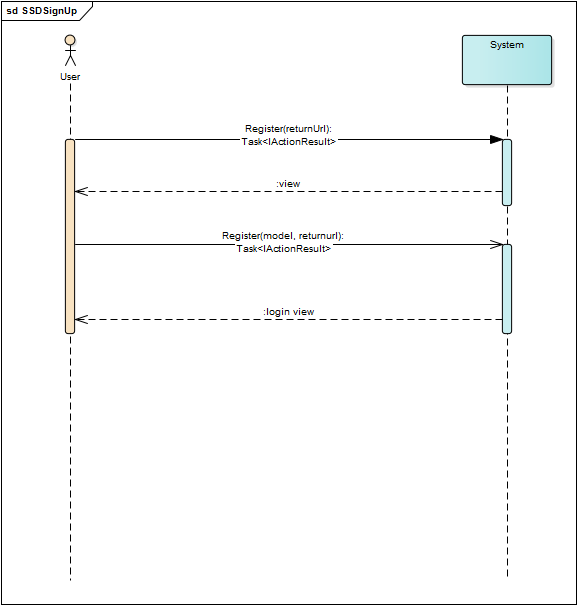


Figure 2.2: SSD Register

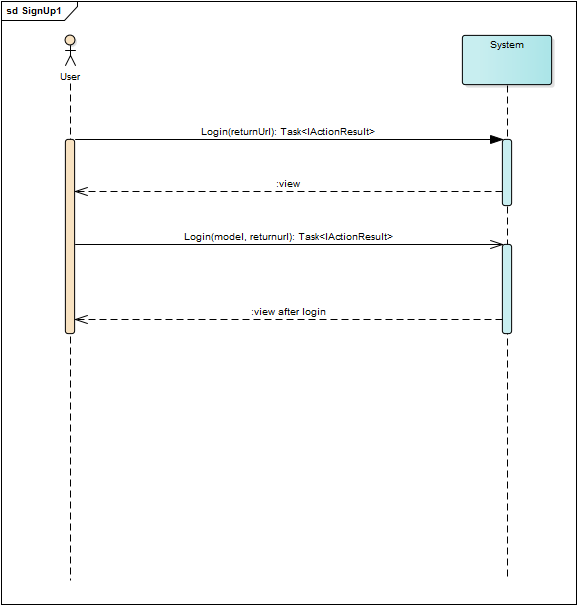


Figure 2.3: SSD Login

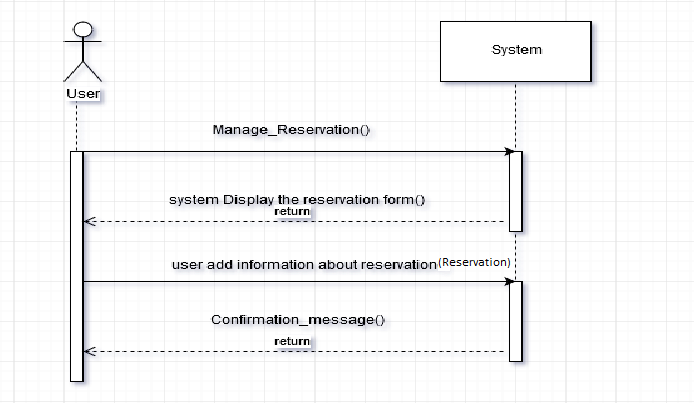


Figure 2.4: Manage Reservation

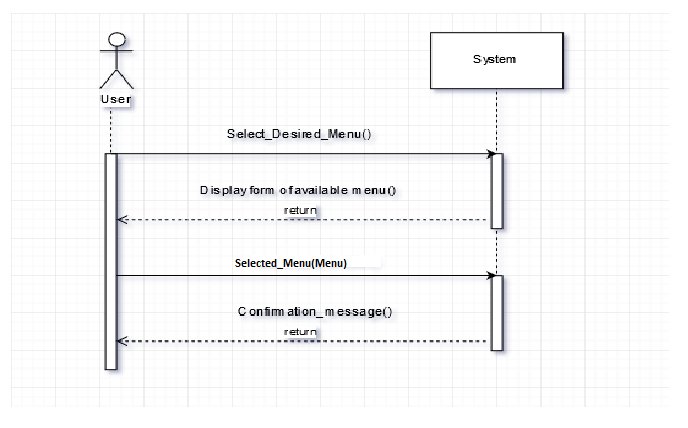


Figure 2.5: SSD select desire menu

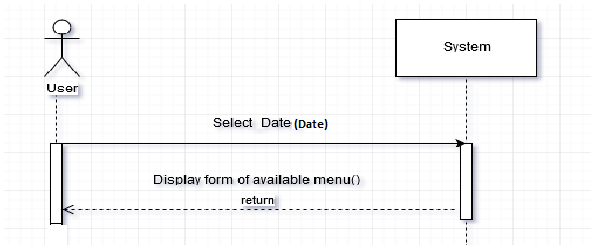


Figure 2.6: SSD Select Date

## Domain Model

Given below is the Domain Model of Process Flow Manager v.1.0.

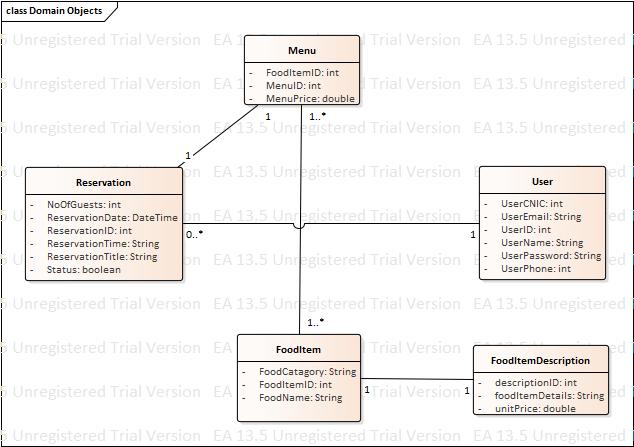


Figure 2.12: Domain Model v.1.0.

# Chapter 3

# System Design

## Layer Definition

## Presentation Layer

This layer defines how the graphical user interface interact with the business layer and the database layer. The main function of this layer is to translate tasks and results something the user can understand.

## Business Layer

This layer controls the system functionality by performing different processing or business rules related to the system. It also moves the data between the surrounding two layers. The main components of this layer are business rules and workflow of the system.

## Database Layer

This layer is used to store the any information in the database which is used in current system. This layer has own work process which handles the tasks related to the database. This layer for the permanent data storage of data.

## Class Diagram

Given below is the Class Diagram of Process Flow Manager v.1.0.

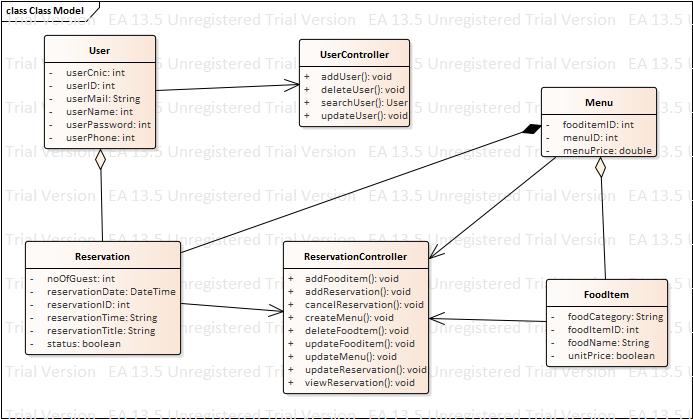


Table 3.1: Class Diagram

## Sequence Diagram

Given below are the Sequence Diagrams of Process Flow Manager v.1.0.

**Sign UP**

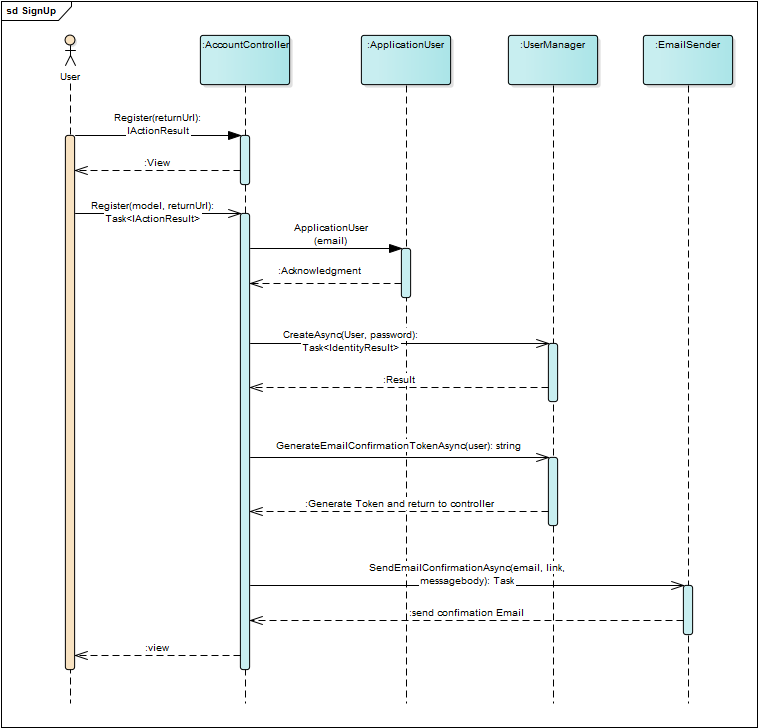
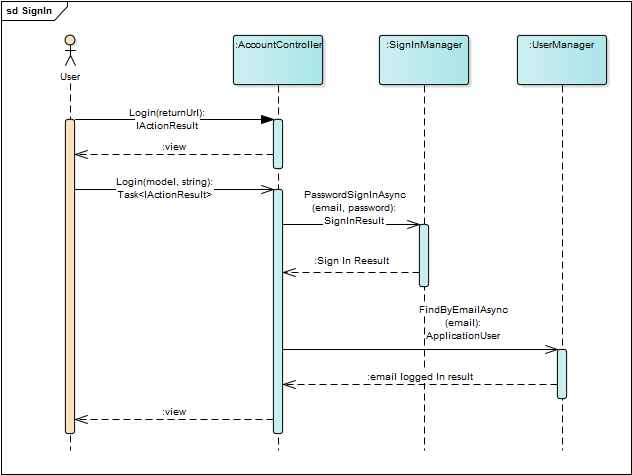
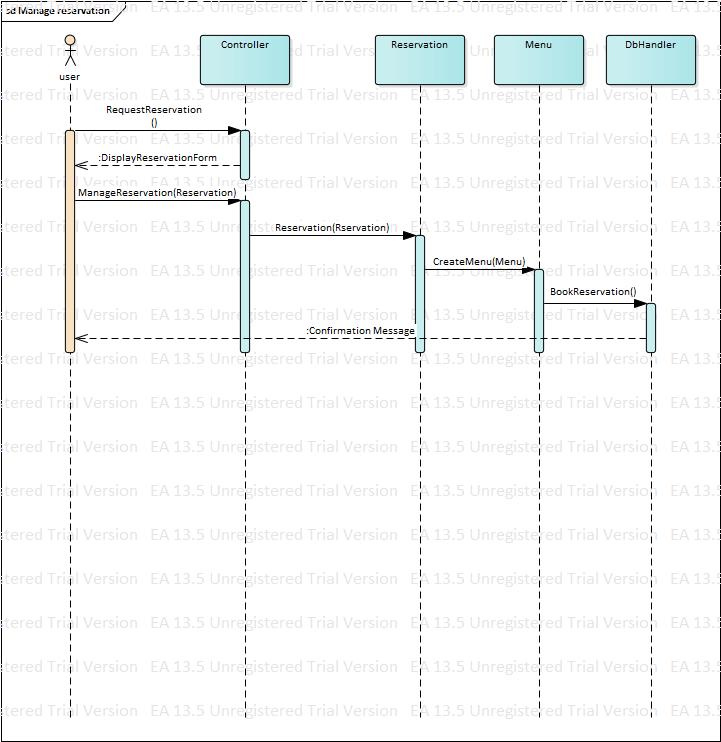


Figure 3.2 signup

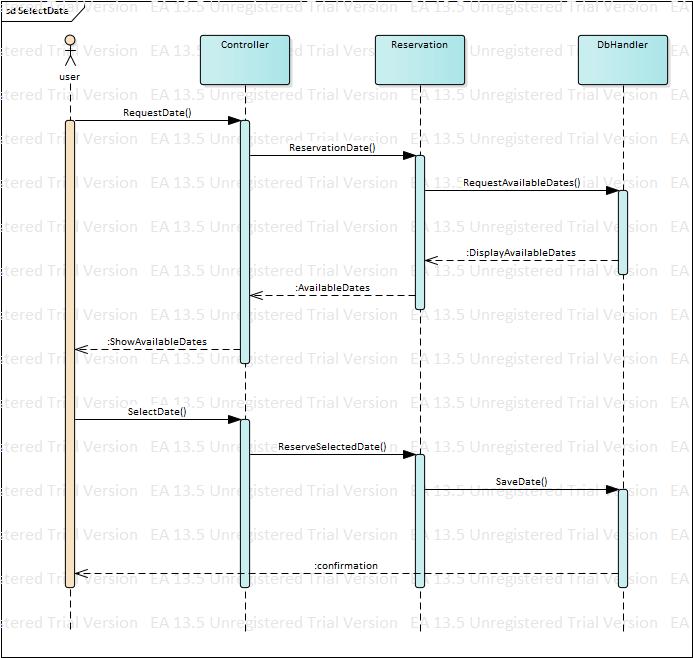
**Sign IN**

****

**Reservation**



**Select Date**



## Entity Relationship Diagram

Given below is the Entity Relationship Diagram of Process Flow Manager v.1.0.

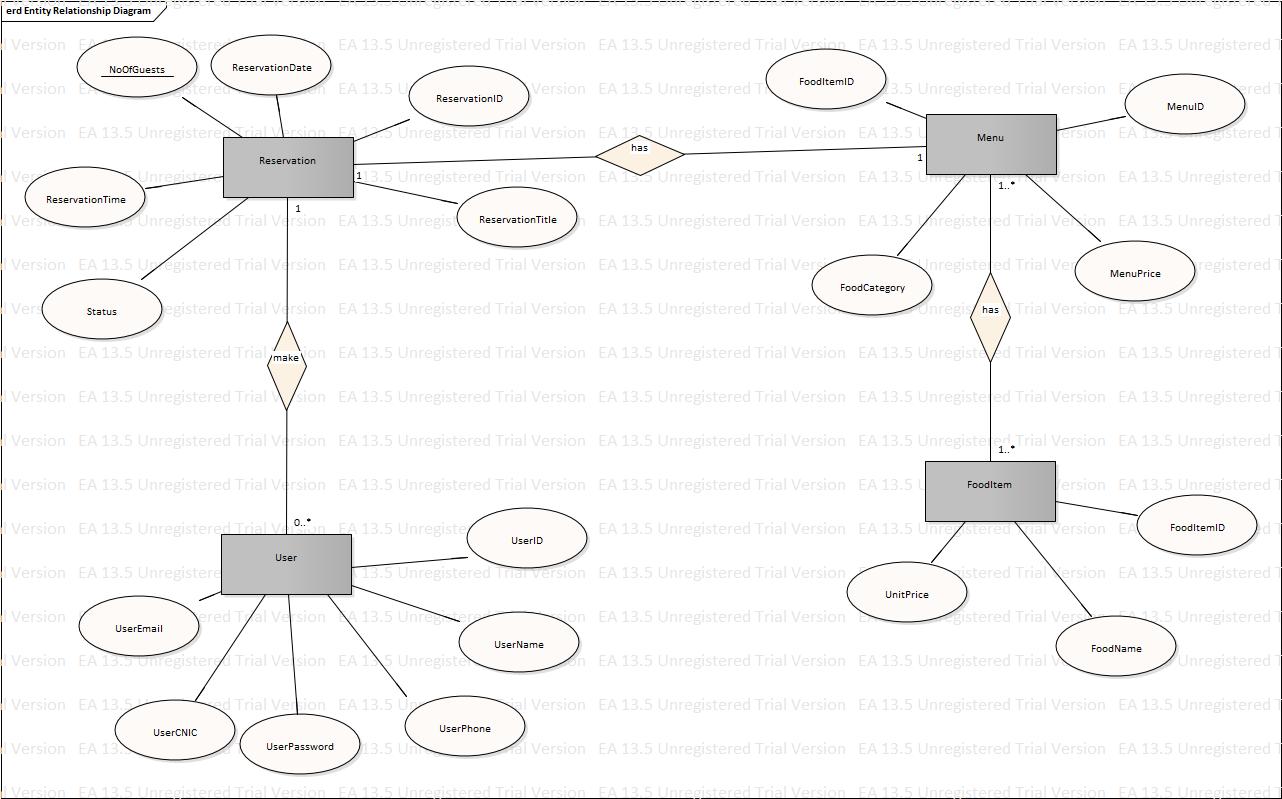
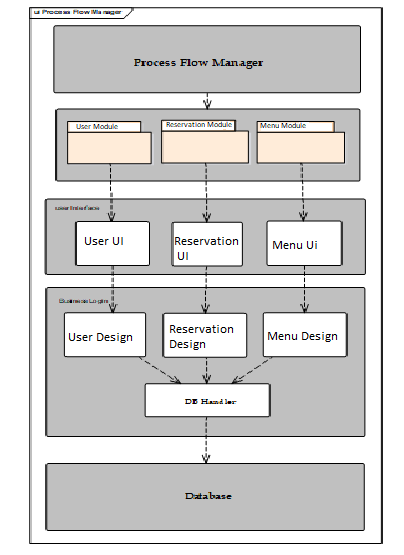
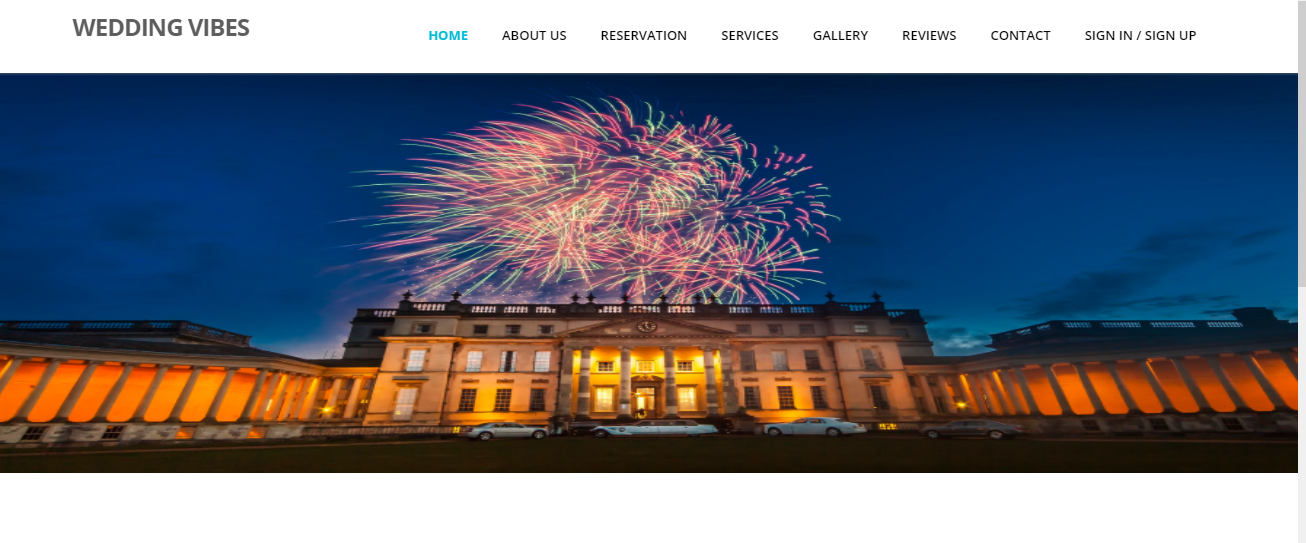


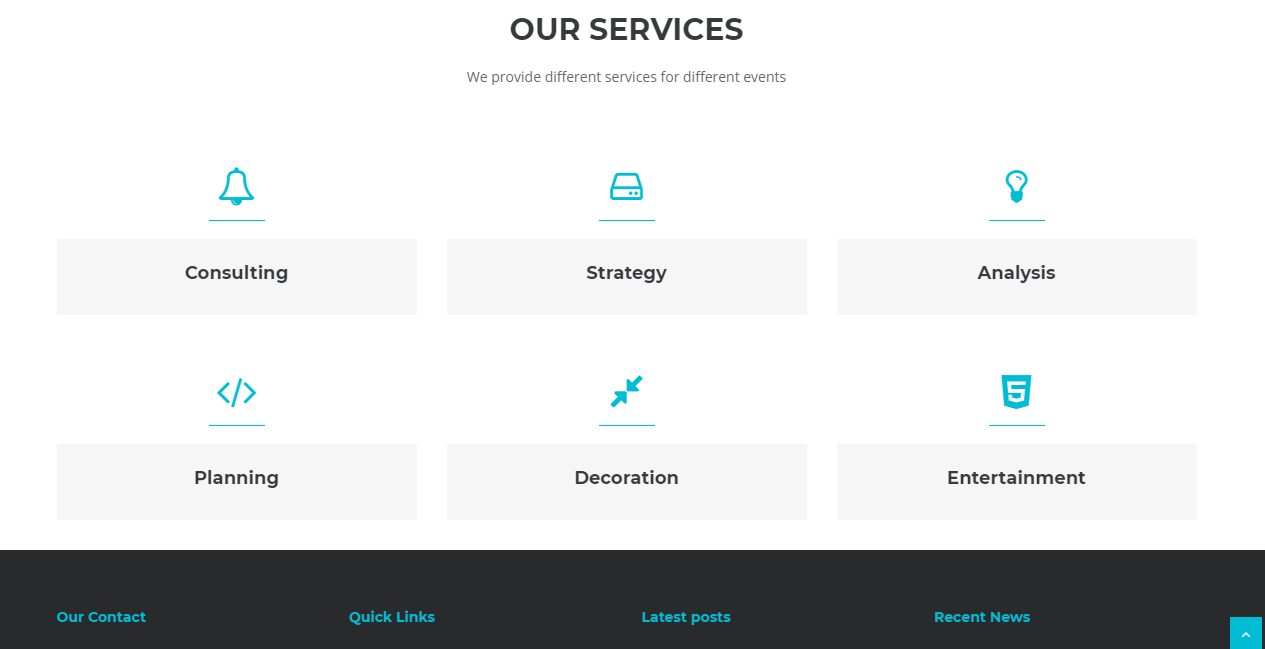
Figure 3.8: Entity Relationship Diagram v.1.0.

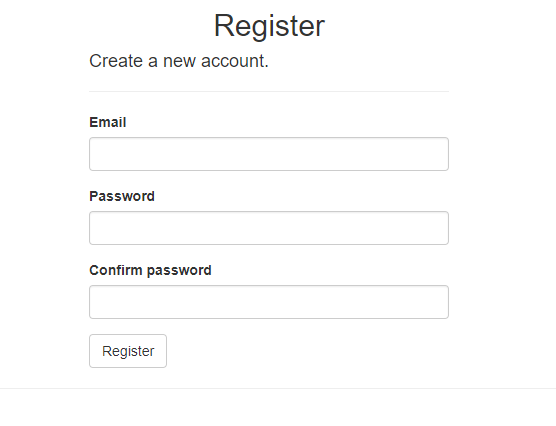
**3.5 : Software Architecture**

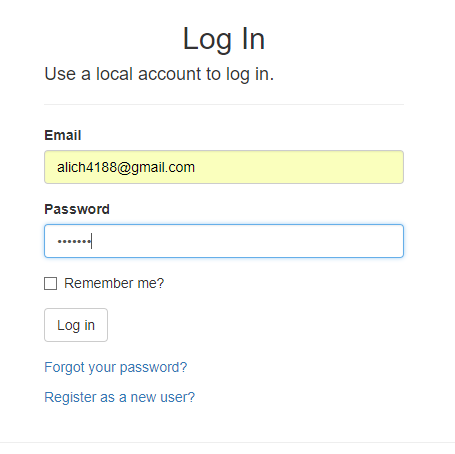
****

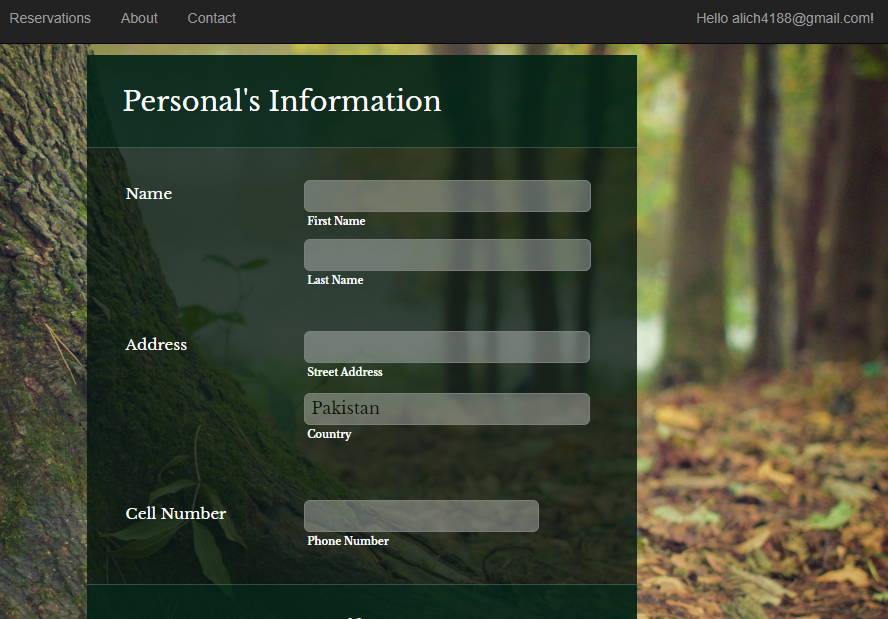
**3.6 : Screen Shots**

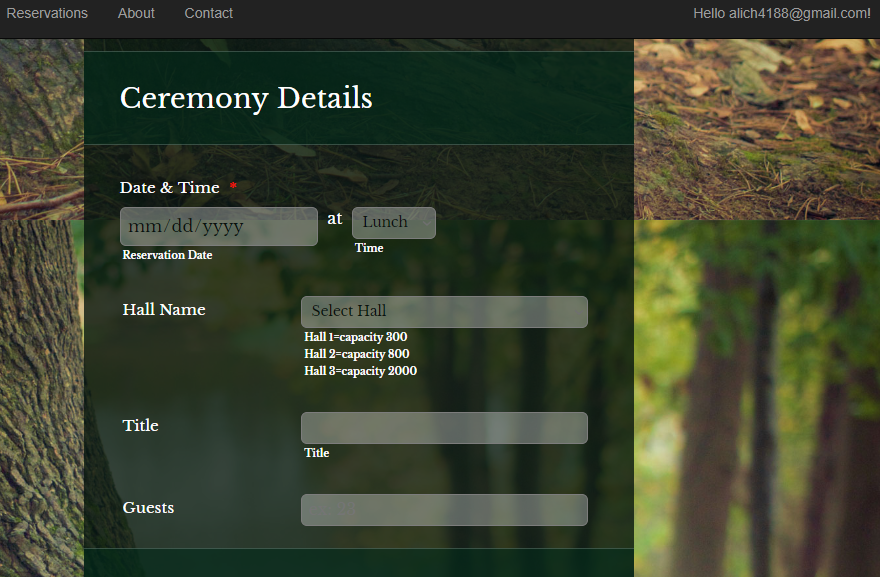




****

****

****

****

# Chapter 4

# Software Development

During implementation we face some difficulties but were later resolved.

* Data base Migration
* Connection of controller with view

## Coding Standards

* **Camel Case**: For arguments and local variable.
* **Pascal Case**: For class names and methods.

# 4.1.1 Indentation

Tab spaces (8 spaces) are used as unit of indentation. The indentation is followed throughout the project consistently

**4.1.2 Declaration**

* Instance variables are placed in the sequence: First public instance variables, protected,
* Package level with no access modifier and then private.
* Next the class constructors are declared.
* Html and stylesheet are used.

## Development Environment

There are different types of languages used in the project and the type of Environment used is given below.

* **Programming Language:**
  + **C#** will be used at backend for web application.
  + **HTML5, CSS3** will be used at frontend for web application.
* **Development Environment:**
* **Visual Studio:** It will be used for the development, debugging and simulation of the webserver.

## Software Description

Main modules of our project are

* User Registration.
* User Login.
* Reservation
* Menu.

**Snippet 1**

[HttpPost]

[AllowAnonymous]

[ValidateAntiForgeryToken]

public async Task<IActionResult> Login(LoginViewModel model, string returnUrl = null)

{

ViewData["ReturnUrl"] = returnUrl;

if (ModelState.IsValid)

{

// This doesn't count login failures towards account lockout

// To enable password failures to trigger account lockout, set lockoutOnFailure: true

var result = await \_signInManager.PasswordSignInAsync(model.Email, model.Password, model.RememberMe, lockoutOnFailure: false);

if (result.Succeeded)

{

\_logger.LogInformation("User logged in.");

return RedirectToLocal(returnUrl);

}

if (result.RequiresTwoFactor)

{

return RedirectToAction(nameof(LoginWith2fa), new { returnUrl, model.RememberMe });

}

if (result.IsLockedOut)

{

\_logger.LogWarning("User account locked out.");

return RedirectToAction(nameof(Lockout));

}

else

{

ModelState.AddModelError(string.Empty, "Invalid login attempt.");

return View(model);

}

}

// If we got this far, something failed, redisplay form

return View(model);

}

**Description**:

The above code describe how the login process will work.If its login successfully then what it shows and if it not works then what happens. Users are login into the system in this module. User enters the username and password if the username and password matches with the saved data in database it will navigate to the next page. If the required filed is incorrect it will display Email is required, and password is required

**Snippet :2**

public class Reservation

{

[Key]

public int id { get; set; }

public string FirstName { get; set; }

public string LastName { get; set; }

public String Address { get; set; }

public String Country { get; set; }

public String City { get; set; }

public int PhoneCode { get; set; }

public int PhoneNo { get; set; }

public String State { get; set; }

public int ZipCode { get; set; }

public int NumberofGuests { get; set; }

public DateTime ReservationDate { get; set; }

public string Title { get; set; }

public Boolean Status { get; set; }

public ApplicationUser user { get; set; }

}

}

**Description**:

Reservation class what attributes are used to make reservation. And all method of reservation is called in this class.

**Snippet:3**

[HttpPost]

[ValidateAntiForgeryToken]

public async Task<IActionResult> Create([Bind("id,FirstName,LastName,Address,Country,City,PhoneCode,PhoneNo,State,ZipCode,NumberofGuests,ReservationDate,Title,Status")] Reservation reservation)

{

if (ModelState.IsValid)

{

\_context.Add(reservation);

await \_context.SaveChangesAsync();

return RedirectToAction(nameof(Index));

}

return View(reservation);

}

**Description:**

When we make any reservation

# Chapter 5

# Software Testing

Software Testing is the most crucial part of Software Development Process. It is the investigation or evaluation of a software component, improving them, and finding bugs and defects. Testing is usually done by executing a system in such a way that it identifies any gaps, errors, or missing requirements in contrary to the actual requirements.

## 5.1. Testing Methodology.

It is essential to have a testing plan in place to ensure that the product delivered is stable, and is delivered on a predictable timeline.

* + Testing is done
  + Project is stable.
  + If a person gives a wrong value’s system should display error message.
  + If a person gives a Right value’s system should display right result.

## 5.2. Testing Environment

Black box testing is used in testing. A software testing technique whereby the internal workings of the item being tested are not known by the tester. For example, in a black box test on a software design the tester only knows the inputs and what the expected outcomes should be and not how the program arrives at those outputs. The tester does not ever examine the programming code and does not need any further knowledge of the program other than its specifications.

The advantages of this type of testing include:

* The test is unbiased because the designer and the tester are independent of each other.
* The tester does not need knowledge of any specific programming languages.
* The test is done from the point of view of the user, not the designer.
* Test cases can be designed as soon as the specifications are complete.

## 5.3. Test Cases

**Test Case 1:**

This test case is generated to test the password validation of system during the registration process when user is trying to gets register to website.

Table 5. 1 Test case Password validation

|  |  |
| --- | --- |
| Date: 26 June 2018 |  |
| *System: Wedding Vibes* |  |
| *Objective:* Registration/Password Validation | *Test ID:*1 |
| *Version:*1 | *Test Type:* Black Box testing |
| *Input:*  *Email: anymail@outlook.com*  Password: 1234 | |
| *Expected Result:* The Password must be at least 6 and at max 100 characters long. | |
| *Actual Result:* Passed | |

**Test Case 2:**

The test case is designed to test the email validation during registration and login process of system when user is entering invalid email address.

Table 5. 2 Test case email validation

|  |  |
| --- | --- |
| Date: 28 June 2018 |  |
| *System:* Wedding Vibes |  |
| *Objective:* Registration/ Email Validation | *Test ID:*2 |
| *Version:*1 | *Test Type:* Black Box testing |
| *Input:*  Email Id: ali | |
| *Expected Result:* The Email field is not a valid e-mail address. | |
| *Actual Result:* Passed | |

**Test Case 3:**

The test case is designed to test the required field validation while trying to login or registering in to the system.

Table 5. 3 Test case Requires field validation

|  |  |
| --- | --- |
| Date: 28 June 2018 |  |
| *System:* Wedding Vibes |  |
| *Objective:* Registration/ Required field Validation | *Test ID:*3 |
| *Version:*1 | *Test Type:* Black Box testing |
| *Input: null* | |
| *Expected Result:*  The Email field is required.  The Password field is required | |
| *Actual Result:* Passed | |

**Test Case 4:**

The test case is designed to test the validation and authorization while trying to login or registering in to the system.

Table 5. 4 Test case Login Authorization

|  |  |
| --- | --- |
| Date: 28 June 2018 |  |
| *System:* Wedding Vibes |  |
| *Objective:* Login/Fields authorization | *Test ID:*4 |
| *Version:*1 | *Test Type:* Black Box testing |
| *Input:*  *Email:* [*tayyabmushtaq2@gmail.com*](mailto:tayyabmushtaq2@gmail.com)  *Password: 11103740%Tax* | |
| *Expected Result:* User log in successfully | |
| *Actual Result:* Passed | |

**Test Case 5:**

**Test Case: Reservation**

Table 5.4 Test case Requires field validation

|  |  |
| --- | --- |
| Date: 28 May 2018 |  |
| *System:* wedding vibes |  |
| *Objective:* Reservation Required field Validation | *Test ID:*4 |
| *Version:*1 | *Test Type:* black Box testing |
| *Input: date of reservation* | |
| *Expected Result:* Date of reservation stored successfully along with other attributes. | |
| *Actual Result:* failed | |