

## Contents

DECLARATION

CERTIFICATE

ACKNOWLEDGEMENT

ABSTRACT

LIST OF FIGURES

INTRODUCTION .....	4
1.1    PROJECT OVERVIEW .....	5
1.1.1 Functional description.....	5
1.2    What is Digital Table Service .....	5
1.3    Product Function .....	<b>Error! Bookmark not defined.</b>
1.4    Models in Project .....	<b>Error! Bookmark not defined.</b>
1.5    Database Connector .....	6
CHAPTER 2 .....	7
REQUIREMENT ANALYSIS .....	7
2.1    PROJECT ORGANISATION.....	8
2.1.1    Roles and Responsibility.....	8
2.1.2    Tools and Techniques .....	8
2.2    Brief description of various tools used .....	9
2.2.1 Spring Framework .....	10
2.2.2 MYSQL Workbench.....	10
2.3    Project Management Plan .....	11
2.4    SOFTWARE REQUIRMENT SPECIFICATION (SRS).....	12
2.4.1    Hardware Requirement .....	12
2.4.2    Software Requirement.....	12
DESIGN PHASE .....	13
3.2    SOFTWARE DESIGN DESCRIPTION .....	14
3.1.1    Block Diagram for Digital Table Service .....	14
3.1.2    Detail description of components.....	155
3.2    OVERALL DESCRIPTION .....	16

3.2.1	Basic Steps of the Project.....	16
3.2.1	Product Functions .....	17
3.3	UML diagrams .....	18
3.3.1	Use case diagram .....	18
3.3.2	Class Diagram.....	19
3.3.3	Activity Diagram.....	20
3.3.4	Sequence Diagram .....	<b>Error! Bookmark not defined.</b>
CODING.....		25
TESTING.....		40
IMPLEMENTATION.....		45
CONCLUSION.....		61
REFERENCES.....		63
FUTURE ENHANCEMENTS.....		59

## LIST OF FIGURES

Figure 1: Block Diagram .....	14
Figure 2: Flow of Project.....	16
Figure 3: Various Models.....	17
Figure 4: Use Case Diagram.....	18
Figure 5: Class Diagram.....	19
Figure 6: Sequence Diagram.....	20
Figure 7: Activity Diagram.....	23

# **CHAPTER 1**

## **INTRODUCTION**

## **1.1 PROJECT OVERVIEW**

### **1.1.1 Functional description:**

Currently in Restaurants waiter comes and collects the order on either notepad or tab then they inform chef about those order, in that sometimes miscommunication happens and also Restaurants owner are not able to maintain their customer's records and feedback about their food. Even the customer cannot track his/her order.

So **Digital Table Service** will provide all these facility. Customers can track his/her order and after ordering food in the mean time they can watch video.. Also, a Restaurants owner can maintain a database of his customers and collect feedback from them. Then for payment, this application has two options – either customer can pay by cash or if the customer wants to pay through online.

### **1.2 What is Digital Table Service?**

Table service is food ordered by the customer at the table by using digital tab and served to the customer's table by waiters and waitresses, which can reduce man power as well as can avoid miscommunication.

### **1.3 Product Functions**

- The admin will be able to view customers records, Add menu, Add Special Offer, Add Discount.
- The customer will be able to track his/her order.
- After ordering food in mean time they can watch video.

### **1.4 Models in Project**

#### **Admin:**

This person will be able to access all the functions of the system. This person can view Customers record, Add Chef, Add Tables, Add/Update Menu, and Add Tab.

**Customer:**

Customer can view menu, ordered menu list and time which is required for order delivery. Customers will be able to give feedback for food.

**Chef:**

This person will receive the order and as per the availability of menu he can confirm the order.

**1.5 Database connector**

The database connector used in project is used for populating a set of records. The connector is used to insert specific records by the user. Once the user has entered all the records he has to recommend some of things like name, email-id and mobile number. This will help to maintain user history. This connector can also store records of orders, customer feedbacks and Chef details.

# **CHAPTER 2**

## **REQUIREMENT ANALYSIS**

## 2.1 **PROJECT ORGANISATION**

### 2.1.1 **Roles and Responsibility**

Understanding the requirements, purpose, goals and the scale of the project

- Finalizing the project problem definition
- Market study of various available open source search engines
- Studying & understanding of various criteria used in digital table service
- To implement Admin model
- To implement User model
- To implement Chef model
- To implement the database connector
- Designing the User Interface part of the project
- Integrate and test all modules
- Demonstrate and do suggested modifications
- Prepare a final report and a presentation

### 2.1.2 **Tools and Techniques**

The following tools have been used in the project:

At the initial stage of our project i.e.

1. At Stage 1 we gathered requirements from the client and formulated the requirement Analysis in **Microsoft Word 2013**.
2. At Stage 2 of our project we prepared a detailed project plan.
3. At Stage 3 the entire designing of the project was done.
4. Database Design



➤ Tool used – Apache Tomcat server 9.0

5. Programming Language – Java and spring

6. And finally at Stage 5 the Development of code was done.

## **2.2 Brief description of various tools used**

### **2.2.1 Spring**

Spring is the most popular application development framework for enterprise Java. Millions of developers around the world use Spring Framework to create high performing, easily testable, and reusable code. Spring framework is an open source Java platform. Spring is lightweight when it comes to size and transparency. The core features of the Spring Framework can be used in developing any Java application, but there are extensions for building web applications on top of the Java EE platform. Spring framework targets to make J2EE development easier to use and promotes good programming practices by enabling a POJO-based programming model.

#### **Benefits of Using the Spring Framework**

Following is the list of few of the great benefits of using Spring Framework –

- Spring enables developers to develop enterprise-class applications using POJOs. The benefit of using only POJOs is that you do not need an EJB container product such as an application server but you have the option of using only a robust servlet container such as Tomcat or some commercial product.
- Spring is organized in a modular fashion. Even though the number of packages and classes are substantial, you have to worry only about the ones you need and ignore the rest.
- Testing an application written with Spring is simple because environment-dependent code is moved into this framework. Furthermore, by using JavaBeanstyle POJOs, it becomes easier to use dependency injection for injecting test data.
- Spring's web framework is a well-designed web MVC framework, which provides a great alternative to web frameworks such as Struts or other over-engineered or less popular web frameworks.
- Spring provides a convenient API to translate technology-specific exceptions (thrown by JDBC, Hibernate, or JDO, for example) into consistent, unchecked exceptions.

- Lightweight IoC containers tend to be lightweight, especially when compared to EJB containers, for example. This is beneficial for developing and deploying applications on computers with limited memory and CPU resources.

### Dependency Injection (DI)

The technology that Spring is most identified with is the Dependency Injection (DI) flavor of Inversion of Control. The Inversion of Control (IoC) is a general concept, and it can be expressed in many different ways. Dependency Injection is merely one concrete example of Inversion of Control.

### Aspect Oriented Programming (AOP)

One of the key components of Spring is the Aspect Oriented Programming (AOP) framework. The functions that span multiple points of an application are called cross-cutting concerns and these cross-cutting concerns are conceptually separate from the application's business logic. There are various common good examples of aspects including logging, declarative transactions, security, caching, etc.

## 2.2.2 MySQL workbench

- **SQL Development:** This functionality provides the capability to execute SQL queries, create and manage connections to database servers using the built-in SQL Editor.
- **Data Modeling (Design):** This functionality enables you to create models of your database schema graphically, perform reverse and forward engineer between a schema and a live database, and edit all aspects of your database using the comprehensive Table Editor.
- **Server Administration:** This functionality enables you to administer MySQL server instances by administering users, performing backup and recovery, inspecting audit data, viewing database health, and monitoring the MySQL server performance.
- **Data Migration:** This functionality allows you to migrate from Microsoft SQL Server, Microsoft Access, and other RDBMS tables, objects, and data to MySQL.
- **MySQL Enterprise Support:** This functionality provides support for Enterprise products such as MySQL Enterprise Backup, MySQL Firewall, and MySQL Audit.

## 2.3 PROJECT MANAGEMENT PLAN

### **Task:**

A task set is a collection of software engineering work tasks, deliverables and milestones, resources, dependencies, constraints, risks and contingencies that must be accomplished to complete a particular project. Our project can be carried out with a structured degree of rigor.

Our project has the following main tasks to be carried out.

Task Name : Digital Table Service

- Description : Provides Digital service which reduce manual work . It is a process of maintaining day to day records of Restaurants.
- Resources needed:
  - MySQL Workbench, Eclipse
  - Risk & Contingencies- Failure of LAN connectivity
  - Apache Tomcat Server 9.0

### **Project plan:**

- 1) To implement Digital Table Service.
- 2) To implement connectors e.g. database connector.
- 3) To implement Various Models like Admin, customer and Chef
- 4) Create a user interface for ordering food and serving orders.

## **2.4 SOFTWARE REQUIRMENT SPECIFICATION (SRS)**

### **2.4.1 Hardware Requirement**

- Processor – Above Intel I3 with clock speed of 2.9 GHz
- RAM – 8 GB and Above
- Hard Disk – above 1TB

### **2.4.2 Software Requirement**

- Front-end: HTML, CSS, JavaScript and Bootstrap.
- Back-end: Java and spring.
- Database : MySQL

## **User Documentation**

User documentation mainly comprises of Help menu of application. It will give all the minute details about the project, if any user has any query about any module or functionality, one can refer it and see how to operate the application. This report is the complete documentation of our project. It gives complete details about the project, its functionality, users, software used, hardware requirement, and environment and so on.

## **System features**

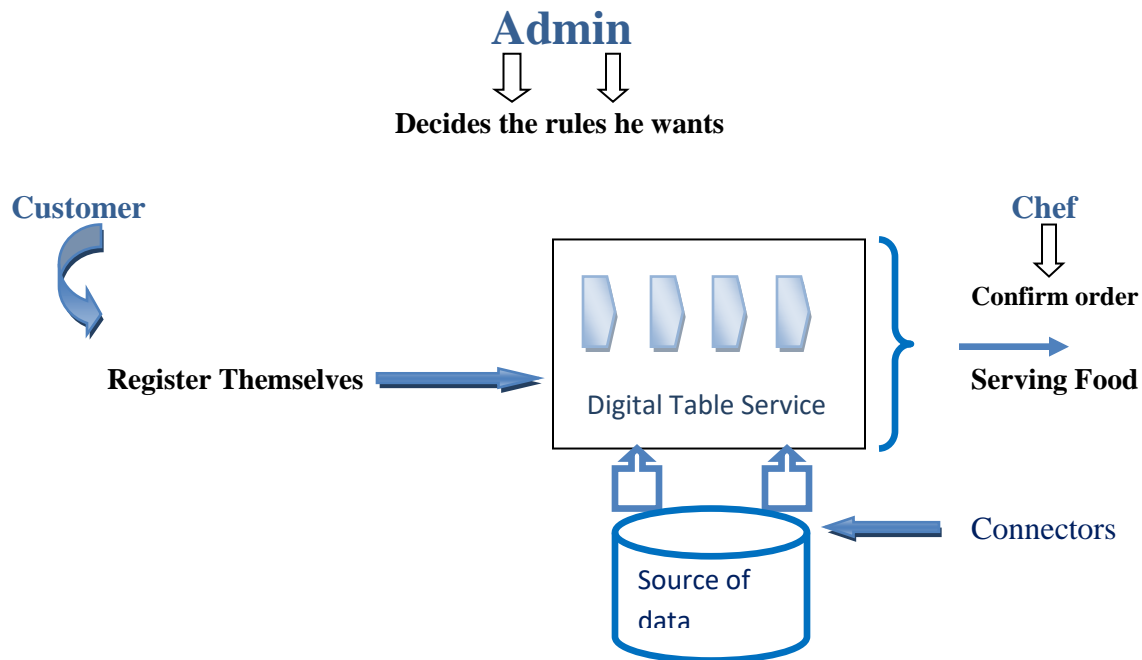
This system will help Restaurants Billing System and also Restaurants order management. This system will help to Restaurants owner for Account maintenance. This billing software also aims at carrying out maintain history of customers. And also improve the performance of billing process. And also maintain standard.

# **CHAPTER 3**

## **DESIGN PHASE**

### 3.1 SOFTWARE DESIGN DESCRIPTION

#### 3.1.1 Block Diagram for Digital Table Service



**Figure 1: Block Diagram for Digital Table Service**

This System will provide digital table service without human intervention, will reduce manual work. Customer have to register themselves before ordering food and if they are already registered then they can directly order food. Chef will confirm order and then food will get served to customer.

### 3.1.2 Detail description of components

#### Description

This system will help Restaurants Billing System and also Restaurants order management. This system will help to Restaurants owner for Account maintenance. This billing software also aims at carrying out maintain history of customers. And also improve the performance of billing process. And also maintain standard.

#### Customers History

- The system should maintain a log of the history of Customers.
- The system will help us to automate the process of manual Billing. Thus, lusing this system, we will be able to save a lot of time and energy.

#### Hotel menu

- Customer will get menu in hotel

#### Customer Feedback

- This system will maintain the Customers feedback.

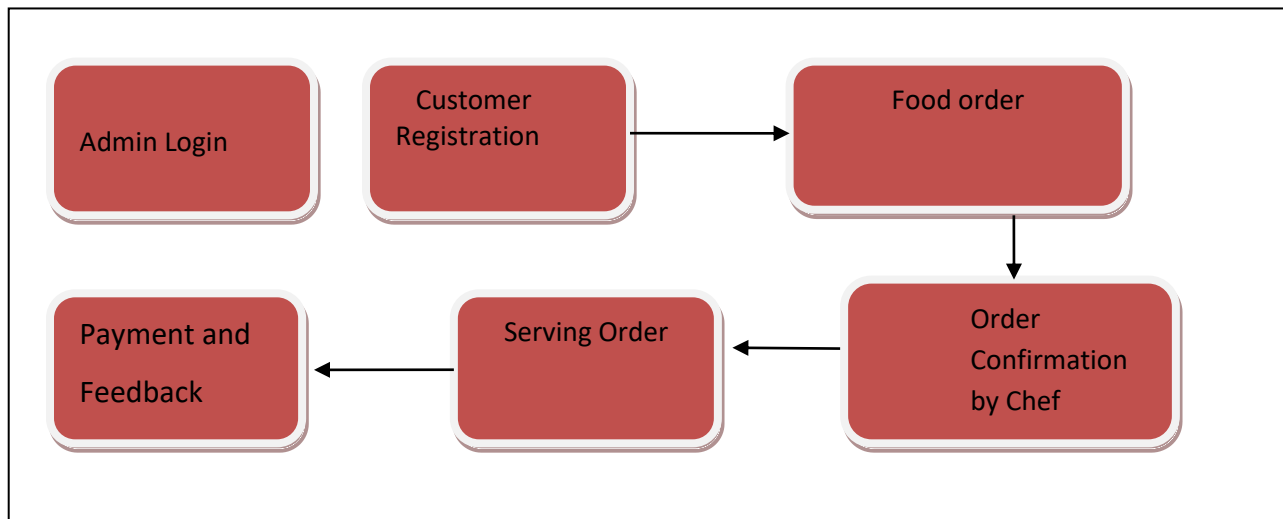
#### Payment Details

- This system will maintain the Customers Payment Details by using cash or online payment system.

## 3.2 OVERALL DESCRIPTION

### 3.2.1 Basic Steps of the Project

The figure represents the overall description of the project. It shows the basic structure of the project.



**Figure 2: FLOW OF PROJECT**

The various steps of the project are :

**Step 1.** Admin can add chef, add table, view customer record, add/update menu.

**Step 2.** Customer registration after this customer can order food.

**Step 3.** Then chef will receive order as per availability of menu and will confirm order.

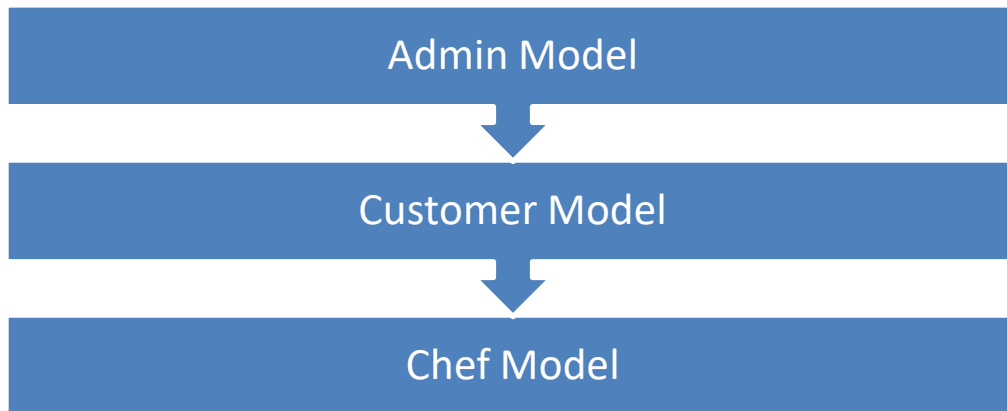


**Step 4.** Food will get served to customer after chef food order confirmation.

**Step 5.** system will display the result to the user.

### 3.2.2 Product Models

The following figure depicts the various modules:-



**Figure 3: Various Models**

### 3.3 UML diagrams

#### 3.3.1 Use case diagram

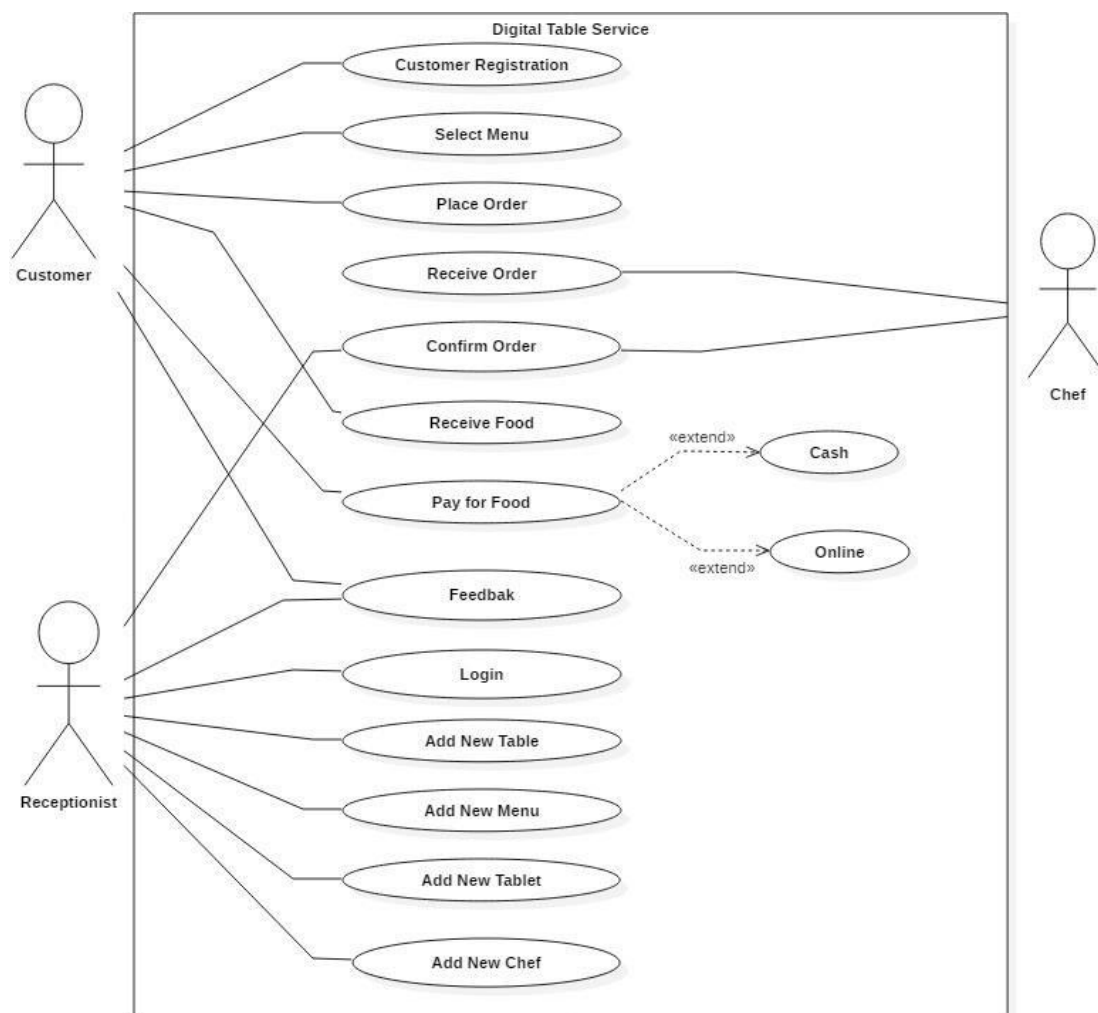


Figure 4. UML CASE DIAGRAM

### 3.3.2 Class Diagram

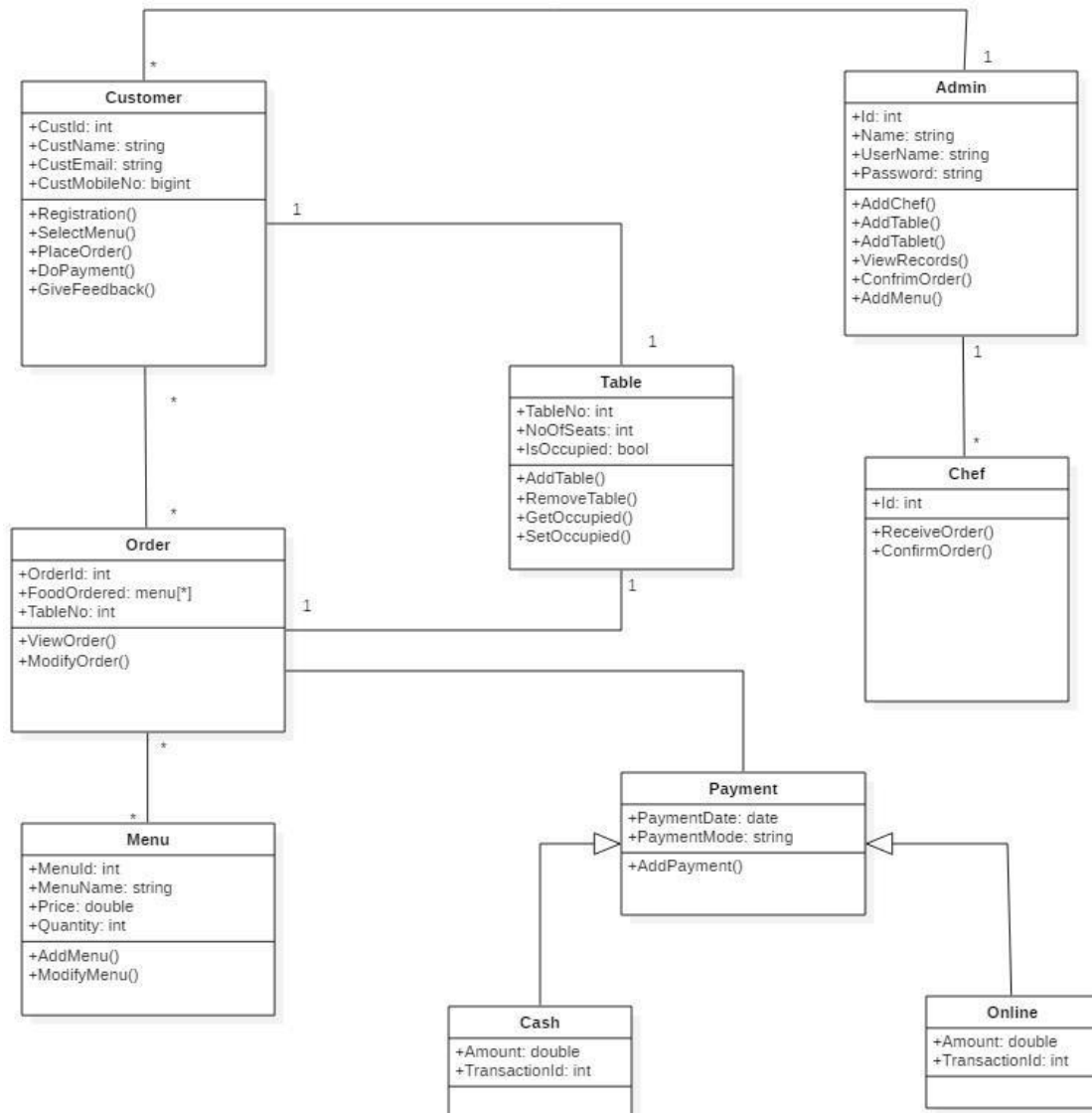


Figure 5. Class Diagram

### 3.3.3 Activity Diagram

Activity Diagram 6.1

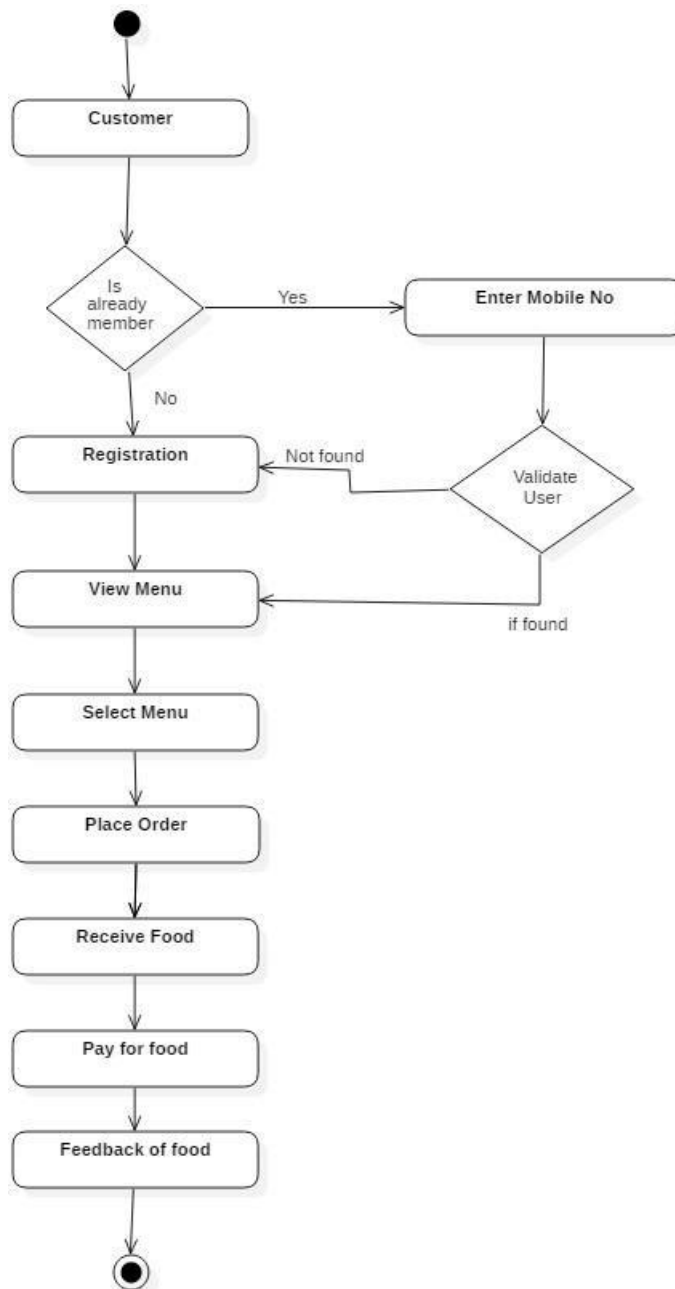
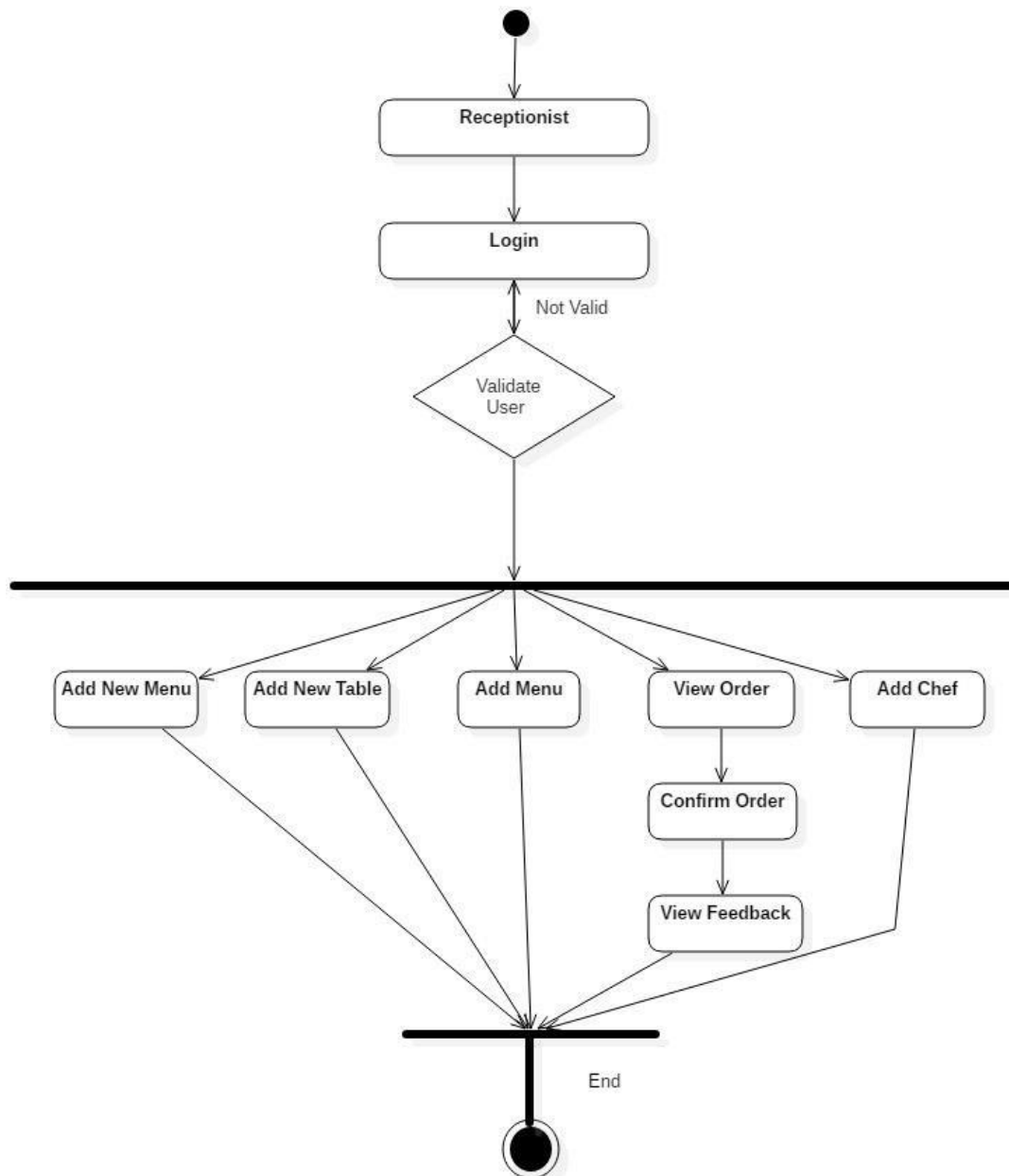
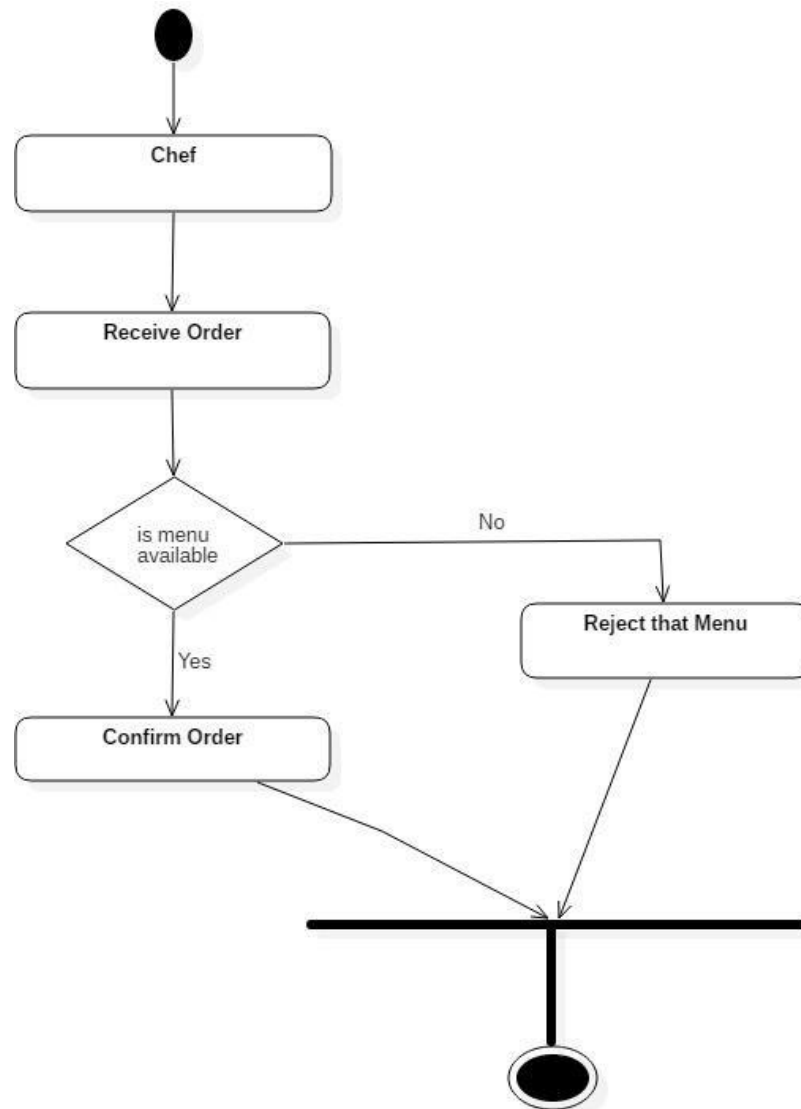


Figure 6.1 Activity Diagram

**Activity Diagram 6.2****Figure 6.2 Activity Diagram**

**Activity Diagram 6.3****Figure 6.2 Activity Diagram**

### 3.3.4 Sequence Diagram

#### Sequence Diagram 7.1

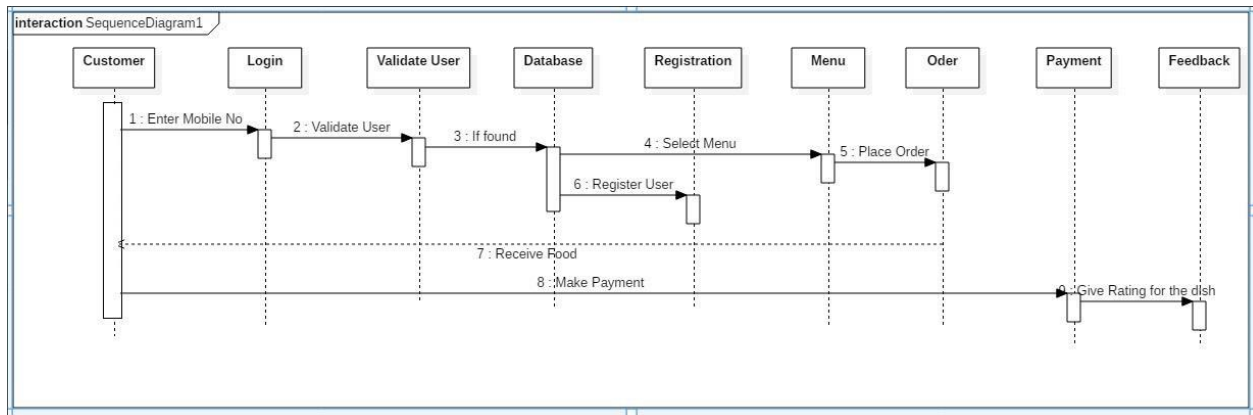


Figure 7.1 Activity Diagram

#### Sequence Diagram 7.2

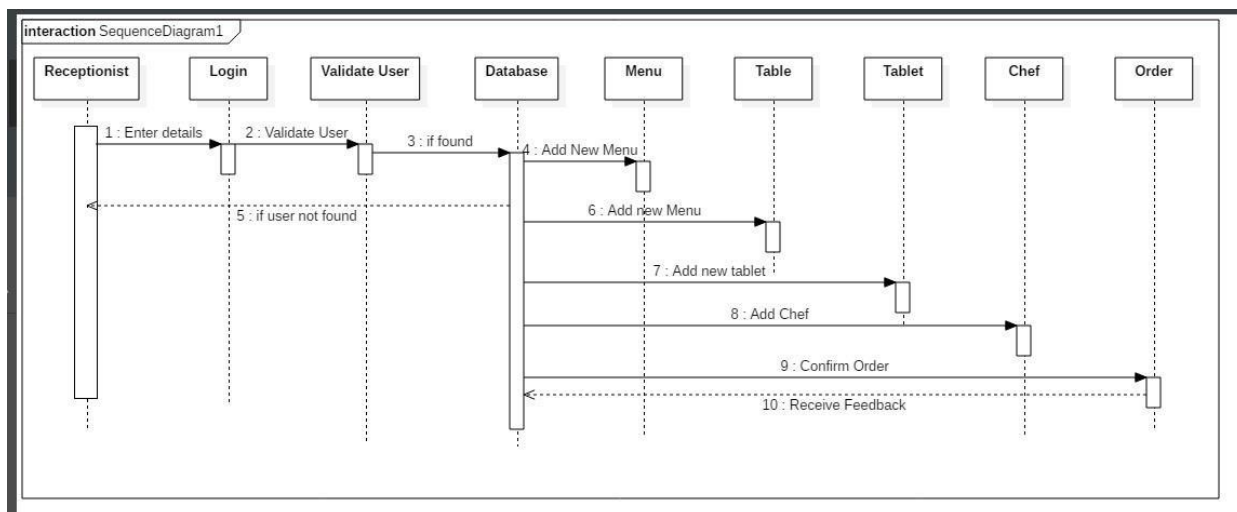


Figure 7.2 Activity Diagram

### Sequence Diagram 7.3

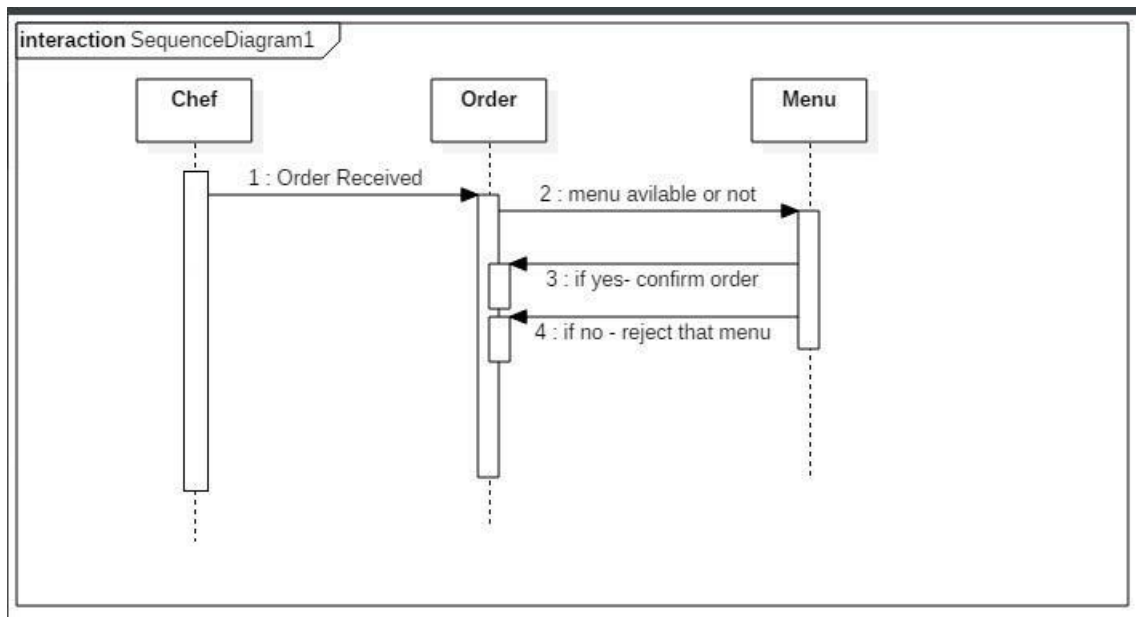


Figure 7.3 Activity Diagram



# **CHAPTER 4**

## **CODING**

## 4.1 IMPLEMENTATION DETAIL

### 4.1.1 Admin Controller

```
package com.cdac.dts.controller;

import javax.servlet.http.HttpSession;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.ui.ModelMap;
import org.springframework.web.bind.annotation.RequestMapping;

import com.cdac.dts.model.Admin;
import com.cdac.dts.model.Employee;
import com.cdac.dts.service.AdminService;

@Controller
public class AdminController {

    @Autowired
    private AdminService adminService;

    @RequestMapping(value="/loginView.htm")
    public String homeLogin(ModelMap model) {
        model.put("admin", new Admin());
    }
}
```

```
        return "adminlogin";
    }

    @RequestMapping(value = "/login.htm")
    public String login(Admin admin ,ModelMap model, HttpSession
session) {

        Boolean b = adminService.checkAdmin(admin);
        if(b) {
            session.setAttribute("admin", admin);
            return "adminhome";
        }
        model.put("admin", new Admin());
        return "adminlogin";
    }

    @RequestMapping(value = "/logout.htm")
    public String logout(Admin admin ,ModelMap model, HttpSession
session) {

        session.removeAttribute("admin");
        session.invalidate();
        return "adminlogin";
    }
}
```

#### 4.1.2 Category Controller

```
package com.cdac.dts.controller;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.ui.ModelMap;
import org.springframework.web.bind.annotation.RequestMapping;

import com.cdac.dts.model.Category;
import com.cdac.dts.service.CategoryService;

@Controller
public class CategoryController {

    @Autowired

    private CategoryService category Service;

    @RequestMapping(value = "/savecat.htm")
    public String prepLoginForm(ModelMap model) {
        model.put("category", new Category());
        return "saveCategory";
    }

    @RequestMapping(value = "/addcat.htm")
    public String register(Category cat,ModelMap model) {
```

```
        categoryService.saveCategory(cat);

        model.put("category", new Category());

        return "adminhome";

    }

}
```

### 4.1.3 Chef Controller

```
package com.cdac.dts.controller;

import javax.servlet.http.HttpSession;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.ui.ModelMap;
import org.springframework.web.bind.annotation.RequestMapping;

import com.cdac.dts.model.Chef;
import com.cdac.dts.service.ChefService;

@Controller
public class ChefController {

    @Autowired
    private ChefService chefService;

    @RequestMapping(value="/signinlogin.htm")
    public String homeLogin(ModelMap model) {
        model.put("chef", new Chef());
        return "cheflogin";
    }

    @RequestMapping(value = "/signin.htm")
```

```
        public String login(Chef chef ,ModelMap model, HttpSession session)
        {

            Boolean b = chefService.checkChef(chef);
            if(b) {
                session.setAttribute("chef", chef);
                return "ConfirmOrder";
            }
            model.put("chef", new Chef());
            return "cheflogin";
        }

        @RequestMapping(value = "/SignOut.htm")
        public String logout(Chef chef ,ModelMap model, HttpSession
session) {

            session.removeAttribute("chef");
            session.invalidate();
            return "cheflogin";

        }

    }
```

#### 4.1.4 Employee Controller

```
package com.cdac.dts.controller;

import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.ui.ModelMap;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.RequestParam;
```

```
import com.cdac.dts.model.Employee;
import com.cdac.dts.service.EmployeeService;

@Controller
public class Employee Controller {

    @Autowired
    private Employee Service employee Service;

    @RequestMapping(value = "/employeeHome.htm")
    public String prepLoginForm(ModelMap model) {
        model.put("employ", new Employee());

        return "add Chef";
    }

    //Inserting data
    @RequestMapping(value = "/addEmployee.htm")
    public String register(Employee employee, ModelMap model) {
        System.out.println("hey");
        employeeService.saveEmployee(employee);

        model.put("employ", new Employee());
        return "adminhome";
    }

    @RequestMapping(value = "/viewChef.htm")
    public String viewChef(ModelMap model) {
        List<Employee> emp = employeeService.EditEmployee();
        model.put("emp",emp);
        return "viewChef";

    }

    //Fetch Employee
    @RequestMapping(value = "/edittable.htm")

    public String requestEmployee(@RequestParam("empId") int
empId,ModelMap model) {
```

```
        System.out.println( "hello"+empId);
        //Employee emp = employeeService.EditEmployee(empId);
        List<Employee> emp = employeeService.EditEmployee(empId);
        System.out.print("aaaaaaaaa"+emp);
        model.put("emp",emp.get(0));

        return "editChef";
    }

    @RequestMapping(value = "/successs.htm")
    public String requestEmployeeUpdate(Employee emp,ModelMap
model) {

        employeeService.UpdateDetails(emp);
        List<Employee> ulist = employeeService.ShowAllUsers();
        model.put("empp", ulist);
        return "successs";
    }

//        @RequestMapping(value = "/employee_list.htm",method =
RequestMethod.GET)
//        public String userList(ModelMap model) {
//        List<Employee> elist = employeeService.ShowAllUsers();
//        model.put("empList", elist);
//        return "employeeList";
//        }

    }
```

#### 4.1.5 FoodTable Controller

```
package com.cdac.dts.controller;

import java.util.List;
```



```
import javax.servlet.http.HttpSession;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.ui.ModelMap;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestParam;

import com.cdac.dts.model.FoodTable;
import com.cdac.dts.service.FoodTableService;

@Controller
public class FoodTableController {

    @Autowired
    private FoodTableService foodTableService;

    @RequestMapping(value = "/savetable.htm")
    public String returnTableView(ModelMap model) {
        model.put("table", new FoodTable());
        return "addtable";
    }

    @RequestMapping(value = "/inserttable.htm")
    public String register(FoodTable table, ModelMap model) {
        foodTableService.saveTable(table);
        model.put("table", new FoodTable());
        return "adminhome";
    }

    @RequestMapping(value = "/viewdropdown.htm")
    public String viewdropdown(FoodTable table, ModelMap model) {
        model.put("table", new FoodTable());

        List<FoodTable> tableList = foodTableService.getAllTableList();
        model.addAttribute("tableList", tableList);
        return "SetTableNumber";
    }

    @RequestMapping(value = "/did.htm")
    public String viewdropdowndone(FoodTable table, ModelMap model,
        HttpSession session) {
```

```
        session.setAttribute("tableNo",table.getTableNo());
        model.put("tab", table);
        return "testcomplete";
    }

    @RequestMapping(value = "/viewFoodTable.htm")
    public String foodTableView(ModelMap model) {
        List<FoodTable> ulist = foodTableService.ShowAllFoodTables();
        model.put("table",ulist);
        return "viewFoodTable";
    }
    @RequestMapping(value = "/editTable.htm")
    public String requestEmployee(@RequestParam("tableNo") int
tableNo,ModelMap model) {
        //Employee emp = employeeService.EditEmployee(empId);
        List<FoodTable> table = foodTableService.EditFoodTable(tableNo);

        model.put("table",table.get(0));

        return "editFoodTable";
    }

    @RequestMapping(value = "/updatedFoodTable.htm")
    public String requestFoodTableUpdate(FoodTable
tableService,ModelMap model) {
        foodTableService.UpdateDetails(tableService);
        List<FoodTable> ulist = foodTableService.ShowAllFoodTables();
        model.put("FoodTable", ulist);
        return "updatedFoodTable";
    }
}
```

#### 4.1.6 Menu Controller

```
package com.cdac.dts.controller;
```

```
import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.ui.ModelMap;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestParam;

import com.cdac.dts.model.Category;
import com.cdac.dts.model.Menu;
import com.cdac.dts.service.MenuService;

@Controller
public class MenuController {

    @Autowired
    private MenuService menuService;

    @RequestMapping(value = "/savemenu.htm")
    public String returnMenuView(ModelMap model) {
        model.put("menu", new Menu());
        List<Category> lstCat = menuService.getCategoryList();
        model.addAttribute("lstCat",lstCat);
        return "addmenu";
    }

    @RequestMapping(value = "/insertmenu.htm")
    public String register(Menu menu, ModelMap model) {
        menuService.saveMenu(menu);
        model.put("menu", new Menu());
        return "adminhome";
    }

    @RequestMapping(value = "/viewMenu.htm")
    public String menuView(ModelMap model) {
        List<Menu> ulist = menuService.getAllMenu();
        model.put("menu",ulist);
        return "viewMenu";
    }

    @RequestMapping(value = "/edittablee.htm")
```

```
        public String requestMenu(@RequestParam("menuId") int
menuId,ModelMap model) {
            //Employee emp = employeeService.EditEmployee(empId);
            List<Menu> menu = menuService.EditMenu(menuId);
            System.out.print("aaaaaaa"+menu);
            model.put("menu",menu.get(0));

            return "editmenu";
        }

        @RequestMapping(value = "/done.htm")
        public String requestMenuUpdate(Menu menu,ModelMap model) {
            menuService.UpdateDetails(menu);
            List<Menu> ulist = menuService.getAllMenu();
            model.put("menu", ulist);
            return "viewMenu";
        }
    }
```

#### 4.1.7 Order Controller

```
package com.cdac.dts.controller;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Controller;
import org.springframework.ui.ModelMap;
import org.springframework.web.bind.annotation.RequestMapping;
```

```
import com.cdac.dts.model.OrderDetails;

import com.cdac.dts.model.UserRegistration;

import com.cdac.dts.service.OrderService;

@Controller

public class OrderController {

    @Autowired

    private OrderService orderService;


    @RequestMapping(value = "/addorder.htm")
    public String register(OrderDetails order,ModelMap model) {

        System.out.println("Calling controller");

        System.out.println(order);

        orderService.addOrder(order);

        model.put("order", new UserRegistration());

        return "UserRegistration";

    }

}
```

#### **4.1.7 User Controller**

```
package com.cdac.dts.controller;
```

```
import java.util.List;

import javax.servlet.http.HttpSession;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.ui.ModelMap;
import org.springframework.web.bind.annotation.RequestMapping;

import com.cdac.dts.model.Menu;
import com.cdac.dts.model.OrderDetails;
import com.cdac.dts.model.UserRegistration;
import com.cdac.dts.service.MenuService;
import com.cdac.dts.service.UserRegistrationService;

@Controller
public class UserController {

    @Autowired
    private UserRegistrationService userRegistrationService;

    @Autowired
    private MenuService menuService;

    @RequestMapping(value="/userRegistrationPage.htm")
    public String homeLogin(ModelMap model) {
        model.put("userReg", new UserRegistration());
        return "UserRegistration";
    }

    //User Registration "
    @RequestMapping(value="/addUserRegistration.htm")
    public String Registration(UserRegistration userRegistration,
ModelMap model, HttpSession session) {
        userRegistrationService.saveUserRegistration(userRegistration);
        model.put("userReg", new UserRegistration());
        return "waitingPage";
    }

    @RequestMapping(value="/getMenuList.htm")
    public String getMenuList(ModelMap model, HttpSession session) {
```

```
List<Menu> lstMenu = menuService.getAllMenu();  
session.setAttribute("list", lstMenu);  
model.put("orderdetails", new OrderDetails());  
return "menu";  
}
```

```
}
```

# **CHAPTER 5**

## **TESTING**

### **5.1 SYSTEM TEST CASES AND TEST RESULTS**

#### **5.1.1 Introduction**



- The aim of testing process is to identify all defects in a software product. Testing is any activity aimed at evaluating the software for quality results it produces and the quality of results it can handle. Testing is an operation to detect the differences between the expected (required) result and the actual result.
- Testing a program consists of subjecting the program to a test inputs or test cases and observing if the program behaves as expected. If the program fails to behave as expected, then the condition under which failures occurs are noted for later debugging and correction.
- Our goal is to design a series of test cases that would have a high likelihood of finding errors. The software testing techniques provide a systematic guidance for designing tests that exercise the internal logic of software components and exercise the input & output domains of the program to uncover errors in program function, behavior and performance.
- Software is tested from two different perspectives:
  - (1) Internal program logic is exercised using “White Box” test case design techniques.
  - (2) Software requirements are exercised using “Black Box” test case design techniques. In both cases, the intent is to find maximum number of errors with minimum effort and time.

### **5.1.2 System test objective and scope**

The main aim to test this is to insure that:

- The Proposed system permits only secure and authenticate access.
- Thus requires the user to enter the mobile no in 10 digit.
- Does all validation time to time as per the need.

- Takes a single input as user id for the detection of anomalies that is used to generate the recommendations.
- Does all the ranking calculations internally.
- Appropriate alerts are generated as per the condition for user convenience.
- Database is updated time to time as the user transaction process proceeds.

## **5.2 A full system test will be conducted including following type of tests**

### **➤ Functional testing:**

To be truly robust, application require more than simple functional testing before release into production.

- Permits only secure and authenticate access.
- Thus requires the user to be registered with the system before use.
- New User must have to register himself/herself
- Does all validation time to time as per the need.
- After registration/authentication user can order food
- Does all the conversion of data internally while requires.
- Appropriate alerts are generated as per the condition for user convenience.
- Database is updated time to time as the new entries.

### **➤ Items to be tested:**

The following items are the ones that constitute the proposed system.  
Here we ensure that all the modules, classes and libraries are included when integrated properly.

No	Name
1.	Admin Authentication

2.	Customer Authentication
3.	Registration validation
4.	Database Connectivity

### ➤ Feature to be tested:

Here we will be testing all features provided by the proposed system to ensure all the features that distinguish the system are implemented properly. The following are the features that will be testing here.

#### a) Admin Authentication:

Here we check whether the Admin login credentials are correct or not. If credentials get matched then Admin will get access to system else login process will get failed

#### b) Customer Authentication:

Here we check that all users log is maintained properly and if entered mobile number of user is already present then that particular user will get authenticated and will get access for ordering food.

#### c) User Registration Validation:

Here while entering registration details certain validations are provided like email-id format, mobile number should be in 10 digit.

#### d) User & system interaction be proper:

Here we check whether the entire process is working properly.

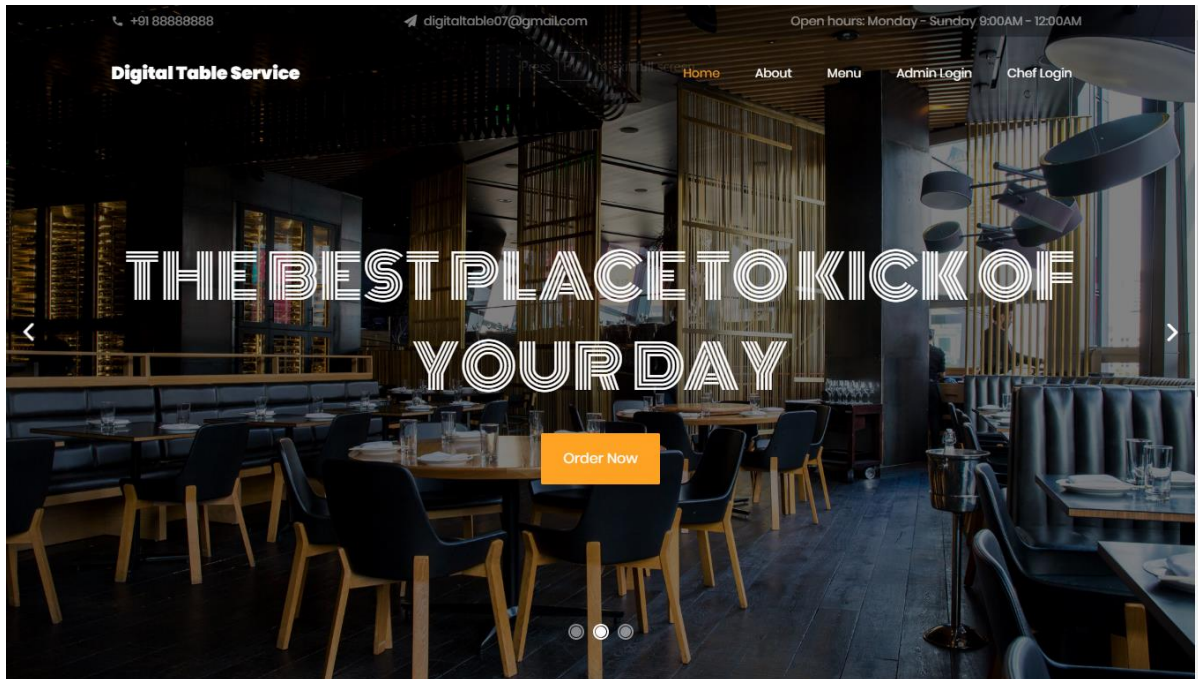
## 5.3 TEST CASES

Test	Items to be	Steps	Input	Actual	Expected	Pass/fail

id	tested			output	output	
1.	Admin id	Admin enters the Admin id	Admin id	Display success	Display message successful	Pass
2.	System check for proper mobile no i.e. id entered by the user	System compares the data entered by the user and the data present in the database.	-	-	-	-
		If Id is valid	-	Make connection	Make connection	Pass
		If Id is invalid	-	Report invalid user id and tell to register themselves	After successful registration make connection	Pass
3.	System Process		-	-	-	-

# **CHAPTER 6**

## **IMPLEMENTATION & RESULT**



Screenshot 1 : Homepage

## Admin Login

Username

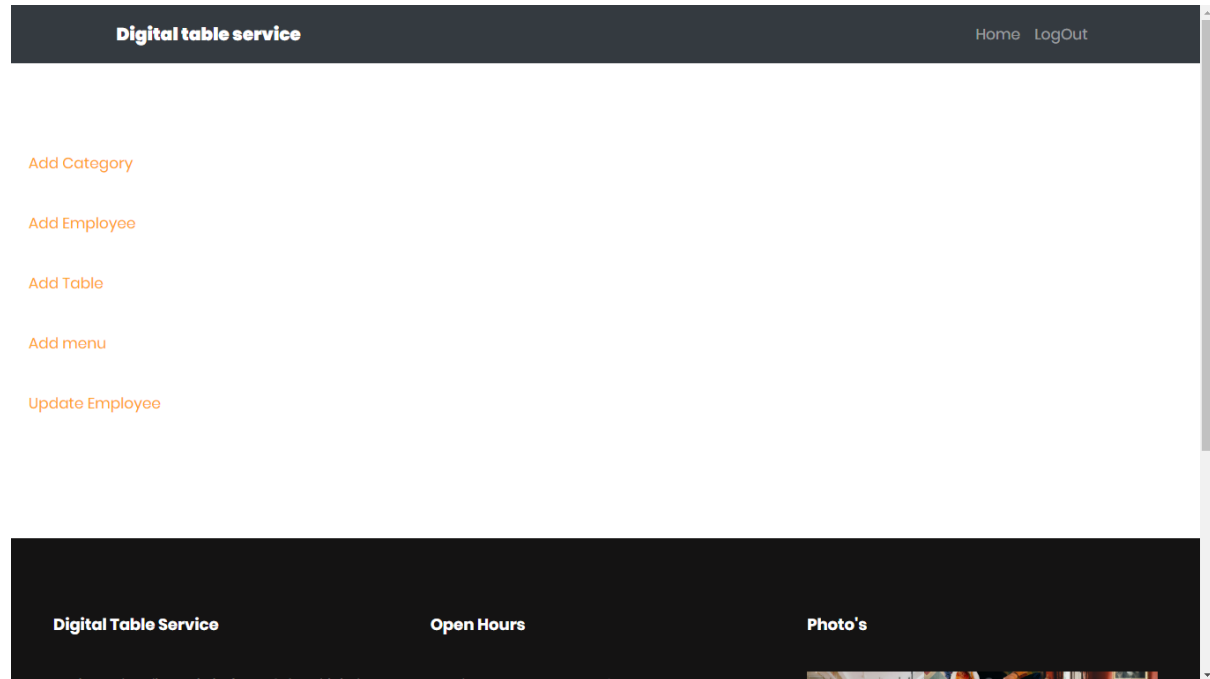
admin

Password

.....|

Login

Screenshot 2 : AdminLogin



Screenshot 3 : AdminFunctions



**Digital table service**[LogOut](#) [Back](#)




## Add Catagory

Category Name

Add Catagory


**Digital Table Service**

Is price and quality a priority for you? Then this is the ideal offer for you.



**Open Hours**

Monday	9:00 AM - 12:00 AM
Tuesday	9:00 AM - 12:00 AM
Wednesday	9:00 AM - 12:00 AM
Thursday	9:00 AM - 12:00 AM
Friday	9:00 AM - 12:00 AM

**Photo's**

Screenshot 4 : AddCategory

Digital table service

[Back](#) [LogOut](#)

## Add Employee

Employee-Id

0

FirstName

LastName

FirstName

LastName

City

State

City

State

Country

Pincode

Country

0

UserName

Password

UserName for chef

password for chef

Add

Screenshot 5 : AddEmployee

**Digital table service**[Back](#) [LogOut](#)

## Add Table

Enter Table Number

Enter no of seats

Enter table Disc

Add Table

[Digital Table Service](#) [Open Hours](#) [Photo's](#)

Screenshot 6 : AddTable

## Add Menu

menu id

Menu Name

select Category

ColdDrink



price half

price Full

Time for preparation

add On Time

Add

Screenshot 7 : AddMenu

**Digital Table Service**[Home](#)

*Life Specialties*  
**Our Menu**

<input checked="" type="checkbox"/>	panir masala	Quantity	20
-------------------------------------	--------------	----------	----

<input type="checkbox"/>	Aloo Matar	Quantity	100
--------------------------	------------	----------	-----

<input type="checkbox"/>	butterscotch	Quantity	0.0
--------------------------	--------------	----------	-----



Screenshot 8 : OurMenu

## Chef Login

Username

Password

Login

Screenshot 9 : ChefLogin

Digital table service

LogOut

## Welcome Chef Login

---

Screenshot 10: ChefWelcome

## Payment

- ☐ Cash  
☒ Card

pay

Screenshot 11: Payment



Digital table service

Home LogOut

FEEDBACK FORM

... Enter your Name ...

Enter your name

... Enter customer-id ...

Enter customer-id

... How you will Rate out of 5 ...

★★★★★

Enter your comment

Submit

Digital Table Service

Open Hours

Photo's

Is price and quality a priority for you? Then this is the ideal offer for you.

Monday

Tuesday

Wednesday

9:00 AM - 12:00 AM

9:00 AM - 12:00 AM

9:00 AM - 12:00 AM

Screenshot 11: Feedback

# Thank You

---

Screenshot 12: ThankYou

# **CHAPTER 8**

## **FUTURE WORKS**

**Future work:**

- We can develop payment gateway
- We can provide Authentication for each Chef.
- We can apply GST on order amount.

# **CHAPTER 9**

## **CONCLUSION**

## CONCLUSION:

**Digital Table Service** will provide all these facility. Customers can track his/her order and after ordering food in the mean time they can watch video. Also, a Restaurants owner can maintain a database of his customers and collect feedback from them. Then for payment, this application has two options – either customer can pay by cash or if the customer wants to pay through online then they can pay through the default payment gateway.

We have make three modules as discussed. These modules are as follows:

**Admin:** This person will be able to access all the functions of the system. This person can view Customers record, Add Chef, Add Tables, Add/Update Menu, and Add Tab.

**Customer:** Customer can view menu, ordered menu list and time which is required for order delivery. Customers will be able to give feedback for food.

**Chef:** This person will receive the order and as per the availability of menu he can confirm the order.

# **CHAPTER 7**

## **REFERENCES**

**BOOKS:**

1. Roger pressman Software Engineering

**WEBSITES:**

- <https://u.ubereats.com/en-IN/>
- <https://www.zomato.com>
- <https://www.swiggy.com/mumbai>