



VIT[®]

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

Course Code: CSE1007

Course Name: Java Programming

Digital Assignment 1

Online Food Order Application

Submitted by

Ateeth Arun (19BDS0113)

To

Prof. Manikandan N

INDEX

S.No	Topic	Page No.
1.	Explanation of the various concepts and frameworks used in the application	4
2.	Basic Features provided within the application	4
3.	CODE	6
	3.1 UNDER src/main/java folder	6
	3.1.1. food.app.connection package	6
	3.1.1.1. DbCon.java	6
	3.1.2. food.app.dao package	7
	3.1.2.1. FoodDao.java	7
	3.1.2.2. OrderDao.java	12
	3.1.2.3. UserDao.java	15
	3.1.3. food.app.model package	19
	3.1.3.1. Cart.java	19
	3.1.3.2. Dessert.java	20
	3.1.3.3. Food.java	23
	3.1.3.4. MainCourse.java	25
	3.1.3.5. Order.java	28
	3.1.3.6. Starter.java	31
	3.1.3.7. User.java	34
	3.1.4. food.app.servlet package	36
	3.1.4.1. AddToCartServlet.java	36
	3.1.4.2. CancelOrderServlet.java	39
	3.1.4.3. CheckOut.java	41
	3.1.4.4. LoginServlet.java	44
	3.1.4.5. LogoutServlet.java	46
	3.1.4.6. OrderNowServlet.java	47
	3.1.4.7. QuantityIncDecServlet.java	51
	3.1.4.8. RegisterServlet.java	53

	3.1.4.9. RemoveFromCartServlet.java	56
	3.2. UNDER src/main/webapp folder	58
	3.2.1. Inside includes folder	58
	3.2.1.1. footer.jsp	58
	3.2.1.2. header.jsp	58
	3.2.1.3. navbar.jsp	59
	3.2.2. UNDER src/main/webapp folder	61
	3.2.2.1. cart.jsp	61
	3.2.2.2. dessert.jsp	65
	3.2.2.3. index.jsp	68
	3.2.2.4. login.jsp	70
	3.2.2.5. maincourse.jsp	72
	3.2.2.6. orders.jsp	75
	3.2.2.7. register.jsp	78
	3.2.2.8. starter.jsp	81
	3.3. pom.xml	83
	3.4. MySQL database where the tables are stored	85
4	Screenshots of some of the views in this application	86
	4.1. User Registration page	86
	4.2. User Login Page	86
	4.3. Home Page	87
	4.4. Starters Page	87
	4.5. Main Course Dishes Page	88
	4.6. Desserts and Beverages Page	88
	4.7. Cart Page	89
	4.8. Orders Page on Clicking Checkout	89

FOOD ORDERING APPLICATION

1. Explanation of the various concepts and frameworks used in the application

A food ordering application has been created using java related concepts such as jdbc connectivity, jsp pages, servlets and also MVC framework was used. The Models used for this application include User, Cart, Dessert, Food, MainCourse, Order, Starter. The various properties associated with the objects are set and a number of functions to get and set attributes have been used. The controllers that have been used within this application include a number of servlets which serve as an interface between the view and model. It is used to intercept incoming requests. Some of the servlets used within this application include AddToCartServlet, CancelOrderServlet, CheckOutServlet, LoginServlet, LogoutServlet, OrderNowServlet, QuantityIncDecServlet, RegisterServlet, RemoveFromCartServlet. A number of View pages such as the User interface to select the food items, Login/Register page, Cart Page, Buy Now page are included. Also, the application is linked with MySQL where the details of all Users, Food Items and various Orders placed are stored. MySQL workbench was used to operate the database in the backend. Also, the logic of showing all the features of the Navigation bar to only users who have logged in is included as the people who have not logged in / registered with the application must not be able to see features of the Navigation bar and they can only view Login form / Registration forms. Also, Apache tomcat server is used to host the application on localhost. And the application was developed in Eclipse IDE and mysqlconnector was used to link the application with MySQL.

2. Basic Features provided within the application

Some of the features that are present in this application include:

- 1) Login: A form for the existing users to log into the application by entering their email id and password provided at the time of registration.
- 2) Registration: A form for the registration for new users where they can register their accounts by providing Name, Password and EmailId. When a new user creates their account, they are redirected to the login page from where they can enter their credentials and login to the application.
- 3) Home: Home page of the application where all the foods and their details which have been stored in the database have been displayed. Each item is displayed in the form of a card which includes its image, name, price in Rupees, Category the food item belongs to (Vegetarian and Non-Vegetarian). Along with each card an Add to Cart option is displayed from where the user can add

the food item to the card. Also, a Buy Now button is there which can allow the users to go straight to the orders page.

4) Starter: The starter page consists of all of the food items in the database which come under the starters category so that if users only want to order starters it will be convenient for them to look among the various food items that are available within starters. Just like the Home page all starters and their details are displayed in the form of cards which includes its image, name, price and also the category it belongs to. Also, along with each card Add To Cart and Buy Now buttons are also displayed.

5) Main Course: The main course page consists of all of the food items in the database which come under the main course category so that if users only want to order main course dishes it will be convenient for them to look among the various food items that are available within main course dishes. Just like the home page all main course dishes and their details are displayed in the form of cards which includes its image, name, price and also the category it belongs to. Also, along with each card Add to Cart and Buy Now buttons are also displayed.

6) Dessert: The dessert page consists of all of the food items in the database which come under the desserts category so that if users only want to order starters it will be convenient for them to look among the various food items that are available within desserts. Just like the home page all desserts and their details are displayed in the form of cards which includes its image, name, price and also the category it belongs to. Also, along with each card Add To Cart and Buy Now buttons are also displayed.

7) Cart: This page consists of the various items that have been added to the cart by the user. In the cart page the user can view that items that have been added to the cart. In this page the user can increase or decrease the number of items. With each increase or decrease of item in the card the total price will get updated. Also, the user can remove any items from the cart if he/she does not want to buy it. A buy now option is available which redirects the user to the orders page where the user can view the order and the total price.

8) Orders: This page consists of the various pending orders and the details of various orders are maintained in the orders database. The date the order was placed and the items that were ordered are displayed. When items are added to the orders list, they are removed from the cart list. Also, orders can be cancelled as well.

9) Navbar: This is used to access the various views which have been created using jsp files. Various services such as the different food categories, cart and list of orders can be accessed using the navigation bar.

3. CODE

3.1. UNDER src/main/java folder

3.1.1. food.app.connection package

3.1.1.1. DbCon.java

This file is used to establish a connection with the MySQL database using java.sql package.

CODE:

```
package food.app.connection;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;

public class DbCon {
    private static Connection connection = null ;

    public static Connection getConnection() {
        if( connection == null ) {
            try {
                Class.forName("com.mysql.cj.jdbc.Driver");
            } catch (ClassNotFoundException e1) {
                // TODO Auto-generated catch block
                e1.printStackTrace();
            }
            try {
```

```

        connection =
DriverManager.getConnection("jdbc:mysql://localhost:3306/foodapp", "root",
"Ateeth@2001");

        System.out.println("connected");

    } catch (SQLException e) {

        // TODO Auto-generated catch block

        e.printStackTrace();

    }

}

return connection;

}

}

```

3.1.2. food.app.dao package

3.1.2.1. FoodDao.java

Within this file the following functionalities are carried out:

- i) getAllFoods() – To get the details of all foods from MySQL database
- ii) getCartFoods() – To get the details of all foods to be stored in the cart
- iii) getTotalCartPrice() – To calculate the amount of price of all the items in the cart
- iv) getSingleFood() – To get the details of a single food item
- v) getAllStarters – To get the details of all the starters from MySQL database
- vi) getAllMainCourseDishes() - To get the details of all the main course dishes from MySQL database
- vii) getAllDesserts() - To get the details of all the desserts from MySQL database

CODE:

```

package food.app.dao;

import java.sql.Connection;
import java.sql.PreparedStatement;

```

```

import java.sql.ResultSet;
import java.util.ArrayList;
import java.util.List;

import food.app.model.*;

public class FoodDao {
    private Connection con ;
    private String query ;
    private PreparedStatement pst ;
    private ResultSet rs ;

    public FoodDao(Connection con) {
        this.con = con;
    }

    public List<Food> getAllFoods(){
        List<Food> foods = new ArrayList<Food>() ;

        try {
            query = "select * from foods" ;
            pst = this.con.prepareStatement(query) ;
            rs = pst.executeQuery();

            while(rs.next()) {
                Food row = new Food() ;
                row.setId(rs.getInt("id"));
                row.setName(rs.getString("name")) ;
                row.setCategory(rs.getString("category")) ;
                row.setPrice(rs.getDouble("price")) ;
                row.setImage(rs.getString("image"));

                foods.add(row) ;
            }
        } catch (Exception e) {
            e.printStackTrace() ;
        }
        return foods;
    }

    public List<Cart> getCartFoods(ArrayList<Cart> cartList){
        List<Cart> foods = new ArrayList<Cart>() ;
        try {
            if(cartList.size() > 0) {
                for(Cart item : cartList) {
                    query = "select * from foods where id = ?" ;

```



```

        pst = this.con.prepareStatement(query) ;
        pst.setInt(1, item.getId());
        rs = pst.executeQuery();
        while(rs.next()) {
            Cart row = new Cart() ;
            row.setId(rs.getInt("id"));
            row.setName(rs.getString("name"));
            row.setCategory(rs.getString("category"));
            row.setPrice(rs.getDouble("price") *
item.getQuantity());

            row.setQuantity(item.getQuantity()) ;
            foods.add(row);
        }
    }
} catch(Exception e) {
    e.printStackTrace();
}
return foods ;
}

public double getTotalCartPrice(ArrayList<Cart> cartList) {
    double sum = 0 ;

    try {
        if(cartList.size() > 0) {
            for(Cart item : cartList) {
                query = "select price from foods where id=?" ;
                pst = this.con.prepareStatement(query) ;
                pst.setInt(1, item.getId());
                rs = pst.executeQuery();

                while(rs.next()) {
                    sum +=
rs.getDouble("price")*item.getQuantity() ;
                }
            }
        }

    } catch(Exception e) {
        e.printStackTrace();
    }

    return sum ;
}

```

```

public Food getSingleFood(int id) {
    Food row = null ;

    try {
        query = "select * from foods where id = ?" ;
        pst = this.con.prepareStatement(query) ;
        pst.setInt(1, id);
        rs = pst.executeQuery() ;

        while(rs.next()) {
            row = new Food() ;
            row.setId(rs.getInt("id"));
            row.setName(rs.getString("name"));
            row.setCategory(rs.getString("category")) ;
            row.setPrice(rs.getDouble("price")) ;
            row.setImage(rs.getString("image")) ;

        }
    }catch(Exception e) {
        e.printStackTrace();
    }

    return row ;
}

```

```

public List<Starter> getAllStarters(){
    List<Starter> starters = new ArrayList<Starter>() ;

    try {
        query = "select * from starters" ;
        pst = this.con.prepareStatement(query) ;
        rs = pst.executeQuery();

        while(rs.next()) {
            Starter row = new Starter() ;
            row.setId(rs.getInt("id"));
            row.setName(rs.getString("name")) ;
            row.setCategory(rs.getString("category")) ;
            row.setPrice(rs.getDouble("price")) ;
            row.setImage(rs.getString("image"));

            starters.add(row) ;

        }
    }catch(Exception e) {
        e.printStackTrace() ;
    }
}

```

```

    }
    return starters;
}

public List<MainCourse> getAllMainCourseDishes(){
    List<MainCourse> maindishes = new ArrayList<MainCourse>();

    try {
        query = "select * from maincoursedishes" ;
        pst = this.con.prepareStatement(query) ;
        rs = pst.executeQuery();

        while(rs.next()) {
            MainCourse row = new MainCourse() ;
            row.setId(rs.getInt("id"));
            row.setName(rs.getString("name")) ;
            row.setCategory(rs.getString("category")) ;
            row.setPrice(rs.getDouble("price")) ;
            row.setImage(rs.getString("image"));

            maindishes.add(row) ;
        }
    }catch(Exception e) {
        e.printStackTrace() ;
    }
    return maindishes;
}

public List<Dessert> getAllDesserts(){
    List<Dessert> desserts = new ArrayList<Dessert>();

    try {
        query = "select * from desserts" ;
        pst = this.con.prepareStatement(query) ;
        rs = pst.executeQuery();

        while(rs.next()) {
            Dessert row = new Dessert() ;
            row.setId(rs.getInt("id"));
            row.setName(rs.getString("name")) ;
            row.setPrice(rs.getDouble("price")) ;
            row.setImage(rs.getString("image"));

            desserts.add(row) ;
        }
    }catch(Exception e) {

```

```

        e.printStackTrace() ;
    }
    return desserts;
}
}

```

3.1.2.2. OrderDao.java

Within this file the following functionalities are carried out:

- i) insertOrder() – To insert orders into the orders table in MySQL Database
- ii) userOrders() – To fetch all the orders from the orders table made by the particular user
- iii) cancelOrder() – To cancel a particular order from the orders table which will be reflected in the application

CODE:

```

package food.app.dao;

import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.util.*;

import food.app.model.*;

public class OrderDao {
    private Connection con ;
    private String query ;
    private PreparedStatement pst ;
    private ResultSet rs ;

```

```

public OrderDao(Connection con) {
    this.con = con ;
}

public boolean insertOrder(Order model) {
    boolean result = false ;

    try {
        query = "insert into orders (p_id , u_id , o_quantity , o_date)
values(?,?,?,?)" ;

        pst = this.con.prepareStatement(query) ;
        pst.setInt(1, model.getId());
        pst.setInt(2, model.getUid());
        pst.setInt(3, model.getQuantity());
        pst.setString(4 , model.getDate());
        pst.executeUpdate() ;
        result = true ;
    }catch(Exception e) {
        e.printStackTrace();
    }
    return result ;
}

public List<Order> userOrders(int id){

```

```

List<Order> list = new ArrayList<>() ;

try {

    query = "select * from orders where u_id = ? order by orders.o_id
desc" ;

    pst = this.con.prepareStatement(query) ;
    pst.setInt(1, id);
    rs = pst.executeQuery() ;

    while(rs.next()) {
        Order order = new Order() ;
        FoodDao foodDao = new FoodDao(this.con) ;
        int pld = rs.getInt("p_id") ;

        Food food = foodDao.getSingleFood(pld) ;
        order.setOrderId(rs.getInt("o_id"));
        order.setId((pld));
        order.setName(food.getName()) ;
        order.setCategory(food.getCategory());
        order.setPrice(food.getPrice()*rs.getInt("o_quantity"));
        order.setQuantity(rs.getInt("o_quantity"));
        order.setDate(rs.getString("o_date"));
        list.add(order) ;

    }
}

```

```

        }catch(Exception e) {
            e.printStackTrace();
        }
        return list ;
    }

    public void cancelOrder(int id) {
        try {
            query = "delete from orders where o_id = ?" ;
            pst = this.con.prepareStatement(query) ;
            pst.setInt(1, id) ;
            pst.execute() ;

        } catch(Exception e) {
            e.printStackTrace();
        }
    }
}

```

3.1.2.3. UserDao.java

Within this file the following functionalities are carried out:

- i) userLogin() – To select the user from the MySQL Database based on email and password entered by the user who is trying to login to the application
- ii) registerUser() – To register the details of a new user who has just created a new account and pass the details to the users table in MySQL Database

```
package food.app.dao;
```

```
import java.sql.Connection;
```

```
import java.sql.DriverManager;
```

```
import java.sql.PreparedStatement;
```

```
import java.sql.ResultSet;
```

```
import java.sql.SQLException;
```

```
import food.app.model.User;
```

```
public class UserDao {
```

```
    private Connection con ;
```

```
    private String query ;
```

```
    private PreparedStatement pst ;
```

```
    private ResultSet rs ;
```

```
    public UserDao(Connection con) {
```

```
        this.con = con;
```

```
    }
```

```
    public User userLogin(String email , String password) {
```

```
        User user = null ;
```

```
        try {
```

```
            //sql injection prevented using preparedStatement
```

```
            query = "select * from users where email = ? and password = ?" ;
```



```

        pst = this.con.prepareStatement(query) ;
        pst.setString(1,email) ;
        pst.setString(2, password);
        rs = pst.executeQuery() ;

        if(rs.next()) {
            user = new User();
            user.setId(rs.getInt("id"));
            user.setName(rs.getString("name")) ;
            user.setEmail(rs.getString("email")) ;
        }
    }catch (Exception e) {
        e.printStackTrace();
        System.out.print(e.getMessage());
    }
    return user ;
}

public int registerUser(User user) throws ClassNotFoundException {
    String INSERT_USERS_SQL = "INSERT INTO users" +
        " ( name, email , password) VALUES " +
        " ( ?, ?, ?);";

    int result = 0;

```

```

Class.forName("com.mysql.jdbc.Driver");

try (Connection connection = DriverManager
    .getConnection("jdbc:mysql://localhost:3306/foodapp", "root",
"Ateeth @2001");

    // Step 2: Create a statement using connection object
    PreparedStatement preparedStatement =
connection.prepareStatement(INSERT_USERS_SQL)) {
    //preparedStatement.setInt(1, user.getId());
    preparedStatement.setString(1, user.getName());
    preparedStatement.setString(2, user.getEmail());
    preparedStatement.setString(3, user.getPassword());
    System.out.println(preparedStatement);
    // Step 3: Execute the query or update query
    result = preparedStatement.executeUpdate();

} catch (SQLException e) {
    // process sql exception
    printSQLException(e);
}

return result;
}

private void printSQLException(SQLException ex) {

```

```

        for (Throwable e: ex) {
            if (e instanceof SQLException) {
                e.printStackTrace(System.err);
                System.err.println("SQLState: " + ((SQLException) e).getSQLState());
                System.err.println("Error Code: " + ((SQLException)
e).getErrorCode());
                System.err.println("Message: " + e.getMessage());
                Throwable t = ex.getCause();
                while (t != null) {
                    System.out.println("Cause: " + t);
                    t = t.getCause();
                }
            }
        }
    }
}

```

3.1.3. food.app.model package

3.1.3.1. Cart.java

In this file declare model of Cart Class. Cart extends Food class

CODE:

```

package food.app.model;

public class Cart extends Food{
    private int quantity ;

```

```

    public Cart() {}

    public int getQuantity() {
        return quantity;
    }

    public void setQuantity(int quantity) {
        this.quantity = quantity;
    }
}

```

3.1.3.2. Dessert.java

In this file model of Dessert class is declared

CODE:

```

package food.app.model;

public class Dessert {
    private int id ;
    private String name ;
    private String category ;
    private double price ;
    private String image ;

    public Dessert(int id , String name, String category, double price, String image) {
        this.id = id ;
        this.name = name;
    }
}

```

```
        this.category = category;  
        this.price = price;  
        this.image = image;  
    }
```

```
    public Dessert() {  
  
    }
```

```
    public int getId() {  
        return id;  
    }
```

```
    public void setId(int id) {  
        this.id = id;  
    }
```

```
    public String getName() {  
        return name;  
    }
```

```
    public void setName(String name) {  
        this.name = name;  
    }
```

```
    public String getCategory() {
```

```
        return category;
    }

    public void setCategory(String category) {
        this.category = category;
    }

    public double getPrice() {
        return price;
    }

    public void setPrice(double price) {
        this.price = price;
    }

    public String getImage() {
        return image;
    }

    public void setImage(String image) {
        this.image = image;
    }

    @Override
    public String toString() {
```

```
        return "Food [id = " + id + " , name=" + name + " , category=" + category +  
        ", price=" + price + " , image=" + image + " ]";  
    }  
}
```

3.1.3.3. Food.java

In this file model of Food class is declared

CODE:

```
package food.app.model;
```

```
public class Food {
```

```
    private int id ;
```

```
    private String name ;
```

```
    private String category ;
```

```
    private double price ;
```

```
    private String image ;
```

```
    public Food(int id , String name, String category, double price, String image) {
```

```
        this.id = id ;
```

```
        this.name = name;
```

```
        this.category = category;
```

```
        this.price = price;
```

```
        this.image = image;
```

```
    }
```

```
    public Food() {
```

```
}
```

```
public int getId() {
```

```
    return id;
```

```
}
```

```
public void setId(int id) {
```

```
    this.id = id;
```

```
}
```

```
public String getName() {
```

```
    return name;
```

```
}
```

```
public void setName(String name) {
```

```
    this.name = name;
```

```
}
```

```
public String getCategory() {
```

```
    return category;
```

```
}
```

```
public void setCategory(String category) {
```

```
    this.category = category;
```

```
}
```



```

    public double getPrice() {
        return price;
    }

    public void setPrice(double price) {
        this.price = price;
    }

    public String getImage() {
        return image;
    }

    public void setImage(String image) {
        this.image = image;
    }

    @Override
    public String toString() {
        return "Food [id = " + id + ", name=" + name + ", category=" + category + ",
price=" + price + ", image=" + image + "]";
    }
}

```

3.1.3.4. MainCourse.java

In this file model of MainCourse class is declared

CODE:

```
package food.app.model;
```

```
public class MainCourse {
```

```
    private int id ;
```

```
    private String name ;
```

```
    private String category ;
```

```
    private double price ;
```

```
    private String image ;
```

```
    public MainCourse(int id , String name, String category, double price, String  
image) {
```

```
        this.id = id ;
```

```
        this.name = name;
```

```
        this.category = category;
```

```
        this.price = price;
```

```
        this.image = image;
```

```
    }
```

```
    public MainCourse() {
```

```
    }
```

```
    public int getId() {
```

```
        return id;
```

```
    }
```

```
public void setId(int id) {  
    this.id = id;  
}
```

```
public String getName() {  
    return name;  
}
```

```
public void setName(String name) {  
    this.name = name;  
}
```

```
public String getCategory() {  
    return category;  
}
```

```
public void setCategory(String category) {  
    this.category = category;  
}
```

```
public double getPrice() {  
    return price;  
}
```

```
public void setPrice(double price) {  
    this.price = price;  
}
```

```

    }

    public String getImage() {
        return image;
    }

    public void setImage(String image) {
        this.image = image;
    }

    @Override
    public String toString() {
        return "Food [id = " + id + ", name=" + name + ", category=" + category +
            ", price=" + price + ", image=" + image + "]\n";
    }
}

```

3.1.3.5. Order.java

In this file declare model of Order Class. Order extends Food class

CODE:

```

package food.app.model;

public class Order extends Food {
    private int orderId ;
    private int uid ;
    private int quantity ;
    private String date ;
}

```

```
public Order() {
```

```
}
```

```
public Order(int orderId, int uid, int quantity, String date) {
```

```
    super();
```

```
    this.orderId = orderId;
```

```
    this.uid = uid;
```

```
    this.quantity = quantity;
```

```
    this.date = date;
```

```
}
```

```
public Order(int uid, int quantity, String date) {
```

```
    super();
```

```
    this.uid = uid;
```

```
    this.quantity = quantity;
```

```
    this.date = date;
```

```
}
```

```
public int getOrderId() {
```

```
    return orderId;
```

```
}
```

```
public void setOrderId(int orderId) {
```

```
    this.orderId = orderId;
```

```
}
```

```
public int getUid() {
```

```
    return uid;
```

```
}
```

```
public void setUid(int uid) {
```

```
    this.uid = uid;
```

```
}
```

```
public int getQuantity() {
```

```
    return quantity;
```

```
}
```

```
public void setQuantity(int quantity) {
```

```
    this.quantity = quantity;
```

```
}
```

```
public String getDate() {
```

```
    return date;
```

```
}
```

```
public void setDate(String date) {
```

```
    this.date = date;
```

```
}
```

```

        @Override

        public String toString() {

                return "Order [orderId=" + orderId + ", uid=" + uid + ", quantity=" + quantity
+ ", date=" + date + "]\n";

        }

}

```

3.1.3.6. Starter.java

In this file model of Starter class is declared

CODE:

```
package food.app.model;
```

```

public class Starter {

        private int id ;

        private String name ;

        private String category ;

        private double price ;

        private String image ;


        public Starter(int id , String name, String category, double price, String image) {

                this.id = id ;

                this.name = name;

                this.category = category;

                this.price = price;

                this.image = image;

        }

}

```

```
public Starter() {
```

```
}
```

```
public int getId() {
```

```
    return id;
```

```
}
```

```
public void setId(int id) {
```

```
    this.id = id;
```

```
}
```

```
public String getName() {
```

```
    return name;
```

```
}
```

```
public void setName(String name) {
```

```
    this.name = name;
```

```
}
```

```
public String getCategory() {
```

```
    return category;
```

```
}
```

```
public void setCategory(String category) {
```



```

        this.category = category;
    }

    public double getPrice() {
        return price;
    }

    public void setPrice(double price) {
        this.price = price;
    }

    public String getImage() {
        return image;
    }

    public void setImage(String image) {
        this.image = image;
    }

    @Override
    public String toString() {
        return "Food [id = " + id + ", name=" + name + ", category=" + category + ",
price=" + price + ", image=" + image + "]";
    }
}

```

3.1.3.7. User.java

In this file model of User class is declared

CODE:

```
package food.app.model;
```

```
public class User {
```

```
    private int id ;
```

```
    private String name ;
```

```
    private String email ;
```

```
    private String password ;
```

```
    public User() {
```

```
    }
```

```
    public User(int id , String name, String password, String email) {
```

```
        this.id = id ;
```

```
        this.name = name;
```

```
        this.password = password;
```

```
        this.email = email ;
```

```
    }
```

```
    public int getId() {
```

```
        return id ;
```

```
    }
```

```
public void setId(int id) {  
    this.id = id ;  
}
```

```
public String getName() {  
    return name;  
}
```

```
public void setName(String name) {  
    this.name = name;  
}
```

```
public String getEmail() {  
    return email;  
}
```

```
public void setEmail(String email) {  
    this.email = email;  
}
```

```
public String getPassword() {  
    return password;  
}
```

```
public void setPassword(String password) {  
    this.password = password;  
}
```

```

    }

    @Override
    public String toString() {
        return "User [id=" + id + ", name=" + name + ", email = " + email + ",
password=" + password + "]";
    }

}

```

3.1.4. food.app.servlet package

3.1.4.1. AddToCartServlet.java

In this file the functionality of adding a food item to cart is handled.

CODE:

```

package food.app.servlet;

import java.io.IOException;
import java.io.PrintWriter;
import java.util.ArrayList;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;

```

```

import food.app.model.Cart;

/**
 * Servlet implementation class AddToCartServlet
 */
@WebServlet("/add-to-cart")
public class AddToCartServlet extends HttpServlet {
    private static final long serialVersionUID = 1L;

    /**
     * @see HttpServlet#HttpServlet()
     */
    public AddToCartServlet() {
        super();
        // TODO Auto-generated constructor stub
    }

    /**
     * @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse
    response)
     */
    protected void doGet(HttpServletRequest request, HttpServletResponse
    response) throws ServletException, IOException {
        response.setContentType("text/html;charset=UTF-8") ;
    }
}

```

```

try(PrintWriter out = response.getWriter()){

    ArrayList<Cart> cartList = new ArrayList<>() ;

    int id = Integer.parseInt(request.getParameter("id")) ;

    Cart cm = new Cart() ;

    cm.setld(id);

    cm.setQuantity(1);

    HttpSession session = request.getSession();

    ArrayList<Cart> cart_list = (ArrayList<Cart>)
session.getAttribute("cart-list") ;

    if(cart_list == null) {

        cartList.add(cm) ;

        session.setAttribute("cart-list", cartList);

        response.sendRedirect("index.jsp");

    }else {

        cartList = cart_list ;

        boolean exist = false ;

        for(Cart c: cart_list) {

            if(c.getld() == id) {

                exist = true ;

                out.println("<h3 style = 'color: crimson; text-align:center'>Item already exist in Cart.<a href='cart.jsp'>Go to cart page</a></h3>") ;

            }

        }

    }
}

```

```

        if(!exist) {
            cartList.add(cm) ;
            response.sendRedirect("index.jsp");
        }
    }
}
}
}

```

3.1.4.2. CancelOrderServlet.java

In this file the functionality of cancelling an order from orders page is handled.

CODE:

```
package food.app.servlet;
```

```
import java.io.IOException;
```

```
import java.io.PrintWriter;
```

```
import javax.servlet.ServletException;
```

```
import javax.servlet.annotation.WebServlet;
```

```
import javax.servlet.http.HttpServlet;
```

```
import javax.servlet.http.HttpServletRequest;
```

```
import javax.servlet.http.HttpServletResponse;
```

```
import food.app.connection.DbCon;
```

```
import food.app.dao.OrderDao;
```

```

/**
 * Servlet implementation class CancelOrderServlet
 */
@WebServlet("/cancel-order")
public class CancelOrderServlet extends HttpServlet {
    private static final long serialVersionUID = 1L;

    protected void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
        try(PrintWriter out = response.getWriter()){
            String id = request.getParameter("id") ;
            if(id != null) {
                OrderDao orderDao = new
OrderDao(DbCon.getConnection()) ;
                orderDao.cancelOrder(Integer.parseInt(id));
            }
            response.sendRedirect("orders.jsp");
        } catch(Exception e) {
            e.printStackTrace() ;
        }
    }

    protected void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
        // TODO Auto-generated method stub

```



```
        doGet(request, response);  
    }  
}
```

3.1.4.3. CheckOut.java

In this file the functionality of checking out from cart page is handled.

CODE:

```
package food.app.servlet;  
  
import java.io.IOException;  
import java.io.PrintWriter;  
import java.text.SimpleDateFormat;  
import java.util.*;  
  
import javax.servlet.ServletException;  
import javax.servlet.annotation.WebServlet;  
import javax.servlet.http.HttpServlet;  
import javax.servlet.http.HttpServletRequest;  
import javax.servlet.http.HttpServletResponse;  
  
import food.app.connection.DbCon;  
import food.app.model.*;  
import food.app.dao.*;  
  
/**
```

```

* Servlet implementation class CheckOut

*/

@WebServlet("/cart-check-out")

public class CheckOut extends HttpServlet {

    private static final long serialVersionUID = 1L;

    protected void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {

        try(PrintWriter out = response.getWriter()){

            SimpleDateFormat formatter = new SimpleDateFormat("yyyy-MM-
dd") ;

            Date date = new Date() ;

            //retrieve all cart products

            ArrayList<Cart> cart_list = (ArrayList<Cart>)
request.getSession().getAttribute("cart-list");

            //user authentication

            User auth = (User) request.getSession().getAttribute("auth") ;

            //check auth and cart list

            if(cart_list != null && auth != null) {

                for(Cart c : cart_list) {

                    //prepare order object

                    Order order = new Order() ;

                    order.setld(c.getld());

```

```

        order.setUid(auth.getId());
        order.setQuantity(c.getQuantity());
        order.setDate(formatter.format(date));

        //instantitate dao class
        OrderDao oDao = new
OrderDao(DbCon.getConnection());

        //call the insert method
        boolean result = oDao.insertOrder(order) ;
        if(!result) {
            break ;
        }
    }

    cart_list.clear();
    response.sendRedirect("orders.jsp");
}else {
    if(auth == null) {
        response.sendRedirect("login.jsp");
    }
    response.sendRedirect("cart.jsp");
}

}catch(Exception e) {

```

```

        e.printStackTrace();
    }
}

/**
 * @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse
response)
 */
protected void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
    // TODO Auto-generated method stub
    doGet(request, response);
}
}

```

3.1.4.4. LoginServlet.java

In this file the functionality of Logging into the application is handled.

CODE:

```

package food.app.servlet;

import java.io.IOException;
import java.io.PrintWriter;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;

```

```

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import food.app.connection.DbCon;
import food.app.dao.UserDao;
import food.app.model.User;

/**
 * Servlet implementation class LoginServlet
 */
@WebServlet("/user-login")
public class LoginServlet extends HttpServlet {
    private static final long serialVersionUID = 1L;

    protected void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
        response.sendRedirect("login.jsp");
    }

    protected void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
        response.setContentType("text/html;charset=UTF-8");
        try(PrintWriter out = response.getWriter()){
            String email = request.getParameter("login-email") ;
            String password = request.getParameter("login-password") ;

            UserDao udao = new UserDao(DbCon.getConnection());

```

```

        User user = udao.userLogin(email, password) ;

        if(user != null) {
            request.getSession().setAttribute("auth", user) ;
            response.sendRedirect("index.jsp") ;
        }else {
            out.print("User Login Fail Incorrect UserName / Password") ;
        }
        //out.print(email + " " + password) ;
    }
}
}

```

3.1.4.5. LogoutServlet.java

In this file the functionality of Logging out of the application is handled.

CODE:

```

package food.app.servlet;

import java.io.IOException;
import java.io.PrintWriter;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;

```

```

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

/**
 * Servlet implementation class LogoutServlet
 */
@WebServlet("/log-out")
public class LogoutServlet extends HttpServlet {
    private static final long serialVersionUID = 1L;

    protected void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
        try(PrintWriter out = response.getWriter()){
            if(request.getSession().getAttribute("auth") != null) {
                request.getSession().removeAttribute("auth");
                response.sendRedirect("login.jsp") ;
            }else {
                response.sendRedirect("index.jsp");
            }
        }
    }
}

```

3.1.4.6. OrderNowServlet.java

In this file the functionality of order now button from the cart page of the application is handled.

CODE:

```
package food.app.servlet;
```

```
import java.io.IOException;
```

```
import java.io.PrintWriter;
```

```
import java.text.SimpleDateFormat;
```

```
import java.util.ArrayList;
```

```
import java.util.Date;
```

```
import javax.servlet.ServletException;
```

```
import javax.servlet.annotation.WebServlet;
```

```
import javax.servlet.http.HttpServlet;
```

```
import javax.servlet.http.HttpServletRequest;
```

```
import javax.servlet.http.HttpServletResponse;
```

```
import food.app.connection.DbCon;
```

```
import food.app.dao.OrderDao;
```

```
import food.app.model.Cart;
```

```
import food.app.model.Order;
```

```
import food.app.model.User;
```

```
@WebServlet("/order-now")
```

```
public class OrderNowServlet extends HttpServlet {
```

```
    private static final long serialVersionUID = 1L;
```

```
    protected void doGet(HttpServletRequest request, HttpServletResponse  
response) throws ServletException, IOException {
```



```

try(PrintWriter out = response.getWriter()){

    SimpleDateFormat formatter = new SimpleDateFormat("yyyy-MM-
dd");

    Date date = new Date();

    User auth = (User) request.getSession().getAttribute("auth");
    if(auth != null) {

        String foodId = request.getParameter("id");

        int foodQuantity =
Integer.parseInt(request.getParameter("quantity"));

        if(foodQuantity <= 0) {
            foodQuantity = 1;
        }

        Order orderModel = new Order();
        orderModel.setId(Integer.parseInt(foodId));
        orderModel.setUid(auth.getId());
        orderModel.setQuantity(foodQuantity);
        orderModel.setDate(formatter.format(date));

        OrderDao orderDao = new
OrderDao(DbCon.getConnection());

        boolean result = orderDao.insertOrder(orderModel);

```

```

        if(result) {

            ArrayList<Cart> cart_list = (ArrayList<Cart>)
request.getSession().getAttribute("cart-list");

            if(cart_list != null) {

                for(Cart c : cart_list) {

                    if(c.getId() == Integer.parseInt(foodId)) {

cart_list.remove(cart_list.indexOf(c)) ;

                                break ;

                            }

                        }

                    }

                response.sendRedirect("orders.jsp");

            }else {

                out.println("order failed") ;

            }

            }else {

                response.sendRedirect("login.jsp");

            }

        }catch(Exception e) {

            e.printStackTrace();

        }

    }

    protected void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {

```

```

        // TODO Auto-generated method stub
        doGet(request, response);
    }
}

```

3.1.4.7. QuantityIncDecServlet.java

In this file the functionality of increase and decrease button for product quantity from the cart page of the application is handled.

CODE:

```

package food.app.servlet;

import java.io.IOException;
import java.io.PrintWriter;
import java.util.ArrayList;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import food.app.model.Cart;

/**
 * Servlet implementation class QuantityIncDecServlet
 */

```

```

@WebServlet("/quantity-inc-dec")

public class QuantityIncDecServlet extends HttpServlet {

    private static final long serialVersionUID = 1L;

    protected void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {

        response.setContentType("text/html;charset=UTF-8");
        try(PrintWriter out = response.getWriter();) {

            String action = request.getParameter("action") ;
            int id = Integer.parseInt(request.getParameter("id")) ;

            ArrayList<Cart> cart_list = (ArrayList<Cart>)
request.getSession().getAttribute("cart-list");

            if(action != null && id >= 1) {
                if(action.equals("inc")) {
                    for(Cart c : cart_list) {
                        if(c.getId() == id) {
                            int quantity = c.getQuantity() ;
                            ++quantity ;
                            c.setQuantity(quantity) ;
                            response.sendRedirect("cart.jsp") ;
                        }
                    }
                }
            }
        }
    }
}

```

```

        if(action.equals("dec")) {
            for(Cart c : cart_list) {
                if(c.getId() == id && c.getQuantity() > 1) {
                    int quantity = c.getQuantity() ;
                    --quantity ;
                    c.setQuantity(quantity) ;
                    break ;
                }
            }
            response.sendRedirect("cart.jsp") ;
        }else {
            response.sendRedirect("cart.jsp") ;
        }
    }
} catch(Exception e) {
    e.printStackTrace() ;
}
}
}

```

3.1.4.8. RegisterServlet.java

In this file the functionality of registering a new user into the application is handled.

CODE:

```
package food.app.servlet;
```

```
import java.io.IOException;
```

```
import java.io.PrintWriter;
```

```
import javax.servlet.ServletException;
```

```
import javax.servlet.annotation.WebServlet;
```

```
import javax.servlet.http.HttpServlet;
```

```
import javax.servlet.http.HttpServletRequest;
```

```
import javax.servlet.http.HttpServletResponse;
```

```
import food.app.connection.DbCon;
```

```
import food.app.dao.UserDao;
```

```
import food.app.model.User;
```

```
@WebServlet("/user-register")
```

```
public class RegisterServlet extends HttpServlet {
```

```
    private static final long serialVersionUID = 1L;
```

```
    protected void doGet(HttpServletRequest request, HttpServletResponse  
response) throws ServletException, IOException {
```

```

        response.sendRedirect("login.jsp");
    }

    protected void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {

        response.setContentType("text/html;charset=UTF-8");
        try(PrintWriter out = response.getWriter()){

            //int id = Integer.parseInt(request.getParameter("register-id")) ;
            String name = request.getParameter("register-name") ;
            String email = request.getParameter("register-email") ;
            String password = request.getParameter("register-password") ;

            UserDao userDao = new UserDao(DbCon.getConnection()) ;
            User user = new User() ;
            // user.setId(id) ;
            user.setName(name) ;
            user.setEmail(email) ;
            user.setPassword(password) ;
            try {
                userDao.registerUser(user) ;
            } catch (ClassNotFoundException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }
            response.sendRedirect("login.jsp") ;
        }
    }
}

```

```
}
```

3.1.4.9. RemoveFromCartServlet.java

In this file the functionality of removing an item from the cart page of the application is handled.

CODE:

```
package food.app.servlet;

import java.io.IOException;
import java.io.PrintWriter;
import java.util.ArrayList;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import food.app.model.Cart;

/**
 * Servlet implementation class RemoveFromCartServlet
 */
@WebServlet("/remove-from-cart")
public class RemoveFromCartServlet extends HttpServlet {
    private static final long serialVersionUID = 1L;
```



```

protected void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {

    response.setContentType("text/html;charset=UTF-8");

    try(PrintWriter out = response.getWriter()){

        String id = request.getParameter("id") ;

        if( id != null) {

            ArrayList<Cart> cart_list = (ArrayList<Cart>)
request.getSession().getAttribute("cart-list");

            if(cart_list != null) {

                for(Cart c : cart_list) {

                    if(c.getId() == Integer.parseInt(id)) {

                        cart_list.remove(cart_list.indexOf(c)) ;

                        break ;

                    }

                }

                response.sendRedirect("cart.jsp") ;

            }

        }else {

            response.sendRedirect("cart.jsp") ;

        }

    }catch(Exception e) {

        e.printStackTrace();

    }

}

}

```

3.2. UNDER src/main/webapp folder

3.2.1. Inside includes folder

This folder contains the files that are common to all jsp files.

3.2.1.1. footer.jsp

This file includes the javascript files of bootstrap 4.0 version

CODE:

```
<script src="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.0.0-beta3/js/all.min.js"></script>

<script src="https://code.jquery.com/jquery-3.2.1.slim.min.js" ></script>

<script
src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.12.9/umd/popper.min.js"></script
>

<script
src="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/js/bootstrap.min.js"></script>
```

3.2.1.2. header.jsp

This file includes the css files of bootstrap 4.0 version and the css of the application is done

CODE:

```
<meta charset="ISO-8859-1">

<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.0.0-beta3/css/all.min.css"/>

<link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.min.css">

<style>

    body{

        background-color : #E6E6FA;

    }
```

```

        .card{
            background-color : #66CDAA ;
        }

        .navbar{
            background-color : #4B0082;
            background-color : #191970;
            background-color :#000080 ;
        }
    </style>

```

3.2.1.3. navbar.jsp

This file includes the configuration of the navigation bar and the links to various jsp files

CODE:

```

<nav class="navbar navbar-expand-lg ">
    <div class="container">
        <a class="navbar-brand" href="index.jsp">Food Ordering App</a>
        <button class="navbar-toggler" type="button" data-toggle="collapse"
            data-target="#navbarSupportedContent"
            aria-controls="navbarSupportedContent" aria-expanded="false"
            aria-label="Toggle navigation">
            <span class="navbar-toggler-icon"></span>
        </button>
    </div>

```

```

<div class="collapse navbar-collapse" id="navbarSupportedContent">
    <ul class="navbar-nav ml-auto">
        <%if( auth != null){%>
            <li class="nav-item active"><a class="nav-link"
href="index.jsp">Home</a></li>
            <li class="nav-item active"><a class="nav-link"
href="starter.jsp">Starters</a></li>
            <li class="nav-item active"><a class="nav-link"
href="maincourse.jsp">Main Course Dishes</a></li>
            <li class="nav-item active"><a class="nav-link"
href="dessert.jsp">Desserts and Beverages</a></li>
            <li class="nav-item"><a class="nav-link"
href="cart.jsp">Cart<span class="badge badge-
danger">${cart_list.size()}</span></a></li>
            <li class="nav-item"><a class="nav-link"
href="orders.jsp">Orders</a></li>
            <li class="nav-item"><a class="nav-link" href="log-
out">Logout</a></li>
        <%}else{%>
            <li class="nav-item"><a class="nav-link"
href="login.jsp">Login</a></li>
            <li class="nav-item"><a class="nav-link"
href="register.jsp">Register</a></li>
        <%}%>
    </ul>
</div>
</div>
</nav>

```

3.2.2. UNDER *src/main/webapp* folder

3.2.2.1. *cart.jsp*

This file contains the content and design of the cart page

CODE:

```
<% @page import="java.util.*"%>
<% @page import="food.app.connection.DbCon"%>
<% @page import="food.app.dao.*"%>
<% @page import="food.app.model.*"%>
<% @page import="java.text.DecimalFormat"%>
<% @ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<%
    DecimalFormat dcf = new DecimalFormat("#.##") ;
    request.setAttribute("dcf" , dcf) ;
    User auth = (User) request.getSession().getAttribute("auth");
    if (auth != null) {
        request.setAttribute("auth", auth);
    }else{
        response.sendRedirect("login.jsp") ;
    }

    ArrayList<Cart> cart_list = (ArrayList<Cart>) session.getAttribute("cart-list") ;
    List<Cart> cartFood = null ;
    if(cart_list != null){
        FoodDao fDao = new FoodDao(DbCon.getConnection()) ;
```

```

        cartFood = fDao.getCartFoods(cart_list) ;
        double total = fDao.getTotalCartPrice(cart_list) ;
        request.setAttribute("cart_list" , cart_list) ;
        request.setAttribute("total" , total) ;
    }
%>
<!DOCTYPE html>
<html>
<head>
<title>Cart Page</title>
<% @include file="includes/head.jsp"%>
<style type = "text/css">
    .table tbody td{
        vertical-align = middle ;
    }

    .btn-incre, .btn-decre{
        color : blue;
        box-shadow : none ;
        font-size : 25px ;
    }
</style>
</head>
<body>
    <% @include file="includes/navbar.jsp"%>

```

```

<div class="container">
    <div class="d-flex py-3">
        <h3>Total Price : ${{total > 0} ?dcf.format(total):0} Rs</h3>
        <a class="mx-3 btn btn-primary " href = "cart-check-out">Check
Out</a>
    </div>
    <table class="table table-light">
        <thead>
            <tr>
                <th scope="col">Name</th>
                <th scope="col">Category</th>
                <th scope="col">Price</th>
                <th scope="col">Buy Now</th>
                <th scope="col">Cancel</th>
            </tr>
        </thead>
        <tbody>
            <%if(cart_list != null){
                for(Cart c: cartFood){ %>
                    <tr>
                        <td><%=c.getName() %></td>
                        <td><%=c.getCategory() %></td>
                        <td><%=dcf.format(c.getPrice()) %>Rs</td>
                        <td>
                            <form action="order-now" method="post"
class="form-inline">

```

```

value="<%=c.getId() %>" class="form-input">
<input type="hidden" name="id"
<div class="form-group d-flex justify-
content-between w-50">
<a class="btn btn-sm btn-decre"
href="quantity-inc-dec?action=dec&id=<%=c.getId()%>">
<i class="fas fa-minus-
square"></i>
</a>
<input type="number"
name="quantity" class="form-control w-50" value="<%=c.getQuantity() %>" readonly>
<a class="btn btn-sm btn-incre"
href="quantity-inc-dec?action=inc&id=<%=c.getId()%>">
<i class="fas fa-plus-
square"></i>
</a>
</div>
<button type = "submit" class = "btn btn-
primary btn-sm">Buy</button>
</form>
</td>
<td>
<a class = "btn btn-sm btn-danger" href =
"remove-from-cart?id=<%=c.getId()%>">Remove</a>
</td>
</tr>
<%}
}%>

```



```

        </tbody>

    </table>

</div>

    <% @include file="includes/footer.jsp"%>

</body>

</html>

```

3.2.2.2. dessert.jsp

This file contains the content and design of the dessert page

CODE:

```

<% @page import="java.util.*"%>

<% @page import="food.app.connection.DbCon"%>

<% @page import="food.app.model.*"%>

<% @page import="food.app.dao.FoodDao"%>

<% @ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>

<%
    User auth = (User) request.getSession().getAttribute("auth");
    if (auth != null) {
        request.setAttribute("auth", auth);
    }else{
        response.sendRedirect("login.jsp") ;
    }
%>

```

```
}
```

```
FoodDao de = new FoodDao(DbCon.getConnection());
```

```
List<Dessert> desserts = de.getAllDesserts();
```

```
ArrayList<Cart> cart_list = (ArrayList<Cart>) session.getAttribute("cart-list");
```

```
if(cart_list != null){
```

```
    request.setAttribute("cart_list", cart_list);
```

```
}
```

```
%>
```

```
<!DOCTYPE html>
```

```
<html>
```

```
    <head>
```

```
        <title>Home Page</title>
```

```
        <% @include file="includes/head.jsp"%>
```

```
    </head>
```

```
    <body>
```

```
        <% @include file="includes/navbar.jsp"%>
```

```
        <div class="container">
```

```
            <div class="card-header my-3">All Desserts</div>
```

```
            <div class="row">
```

```
                <%if(!desserts.isEmpty()){
```

```
                    for(Dessert d : desserts){ %>
```

```
                        <div class="col md-3 my-3">
```

```
                            <div class="card" style="width: 18rem;">
```

```
        
```

```
        <div class="card-body">
```

```
            <h5 class="card-title"><%= d.getName() %></h5>
```

```
            <h6 class = "price">Price: <%= d.getPrice() %> Rs </h6>
```

```
            <div class = "mt-3 d-flex justify-content-between">
```

```
                <a href="add-to-cart?id=<%=d.getId() %>" class="btn btn-dark">Add to cart</a>
```

```
                <a href="order-now?quantity=1&id=<%=d.getId() %>" class="btn btn-primary">Buy now</a>
```

```
            </div>
```

```
        </div>
```

```
    </div>
```

```
    </div>
```

```
    <%= }
```

```
}%>
```

```
</div>
```

```
</div>
```

```
<%= @include file="includes/footer.jsp"%>
```

```
</body>
```

```
</html>
```

3.2.2.3. index.jsp

This file contains the content and design of the home page

CODE:

```
<% @page import="java.util.*"%>
<% @page import="food.app.connection.DbCon"%>
<% @page import="food.app.model.*"%>
<% @page import="food.app.dao.FoodDao"%>
<% @ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<%
    User auth = (User) request.getSession().getAttribute("auth");
    if (auth != null) {
        request.setAttribute("auth", auth);
    }else{
        response.sendRedirect("login.jsp") ;
    }

    FoodDao fd = new FoodDao(DbCon.getConnection()) ;
    List<Food> foods = fd.getAllFoods() ;

    ArrayList<Cart> cart_list = (ArrayList<Cart>) session.getAttribute("cart-list") ;
    if(cart_list != null){
        request.setAttribute("cart_list" , cart_list) ;
    }
}
```

```

%>

<!DOCTYPE html>

<html>

<head>

<title>Home Page</title>

<% @include file="includes/head.jsp"%>

</head>

<body>

    <% @include file="includes/navbar.jsp"%>

    <div class="container">

        <div class="card-header my-3">All food Items</div>

        <div class="row">

            <%if(!foods.isEmpty()){

                for(Food f : foods){ %>

                    <div class="col md-3 my-3">

                        <div class="card" style="width: 18rem;">

                                <div class="card-body">

                                    <h5 class="card-title"><%= f.getName()

%></h5>

                                    <h6 class = "price">Price: <%=

f.getPrice() %> Rs</h6>

                                    <h6 class = "category">Category: <%=

f.getCategory()%></h6>

```

```

                                <div class = "mt-3 d-flex justify-content-
between">

                                <a href="add-to-
cart?id=<%=f.getId() %>" class="btn btn-dark">Add to cart</a>

                                <a href="order-
now?quantity=1&id=<%=f.getId() %>" class="btn btn-primary">Buy now</a>

                                </div>

                                </div>

                                </div>

                                </div>

                                <%}

                                }

                                %>

                                </div>

                                </div>

                                <% @include file="includes/footer.jsp"%>

</body>

</html>

```

3.2.2.4. login.jsp

This file contains the content and design of the login page

CODE:

```

<% @page import="java.util.*"%>

<% @page import = "food.app.model.*" %>

```

```

<% @ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>

<%

    User auth = (User) request.getSession().getAttribute("auth") ;
    if( auth != null ){
        response.sendRedirect("index.jsp") ;
    }

    ArrayList<Cart> cart_list = (ArrayList<Cart>) session.getAttribute("cart-list") ;
    if(cart_list != null){
        request.setAttribute("cart_list" , cart_list) ;
    }

%>

<!DOCTYPE html>

<html>

    <head>

        <title>Login Page</title>

        <% @include file = "includes/head.jsp" %>

    </head>

    <body>

        <% @include file="includes/navbar.jsp"%>

        <div class = "container">

            <div class = "card w-50 mx-auto my-5">

                <div class = "card-header text-center">User Login</div>

                <div class = "card-body">

                    <form action = "user-login" method = "post">

```

```

        <div class = "form-group">
            <label>Email ID</label>
            <input type = "email" class = "form-
control" name = "login-email" placeholder = "Enter Your EmailID" required>
        </div>
        <div class = "form-group">
            <label>Password</label>
            <input type = "password" class = "form-
control" name = "login-password" placeholder = "*****" required>
        </div>
        <div class = "text-center">
            <button type = "submit" class = "btn btn-
primary">Login</button>
        </div>
    </form>
</div>
</div>
</div>

    <% @include file = "includes/footer.jsp" %>
</body>
</html>

```

3.2.2.5. maincourse.jsp

This file contains the content and design of the main course page

CODE:


```

<% @page import="java.util. *" %>
<% @page import="food.app.connection.DbCon" %>
<% @page import="food.app.model. *" %>
<% @page import="food.app.dao.FoodDao" %>
<% @ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1" %>
<%
User auth = (User) request.getSession().getAttribute("auth");
if (auth != null) {
    request.setAttribute("auth", auth);
}else{
    response.sendRedirect("login.jsp") ;
}

FoodDao md = new FoodDao(DbCon.getConnection()) ;
List<MainCourse> maindishes = md.getAllMainCourseDishes() ;

ArrayList<Cart> cart_list = (ArrayList<Cart>) session.getAttribute("cart-list") ;
if(cart_list != null){
    request.setAttribute("cart_list" , cart_list) ;
}
%>
<!DOCTYPE html>
<html>
    <head>
        <title>Home Page</title>

```

```

        <% @include file="includes/head.jsp"%>

</head>

<body>

        <% @include file="includes/navbar.jsp"%>


        <div class="container">

                <div class="card-header my-3">All Main Course Dishes</div>

                <div class="row">

                        <%if(!maindishes.isEmpty()){

                                for(MainCourse m : maindishes){ %>

                                        <div class="col md-3 my-3">

                                                <div class="card" style="width: 18rem;">

                                                        <div class="card-body">

                                                                <h5 class="card-title"><%=
m.getName() %></h5>

                                                                <h6 class = "price">Price: <%=
m.getPrice() %> Rs </h6>

                                                                <h6 class = "category">Category: <%=
m.getCategory() %></h6>

                                                                <div class = "mt-3 d-flex justify-content-
between">

                                                                        <a href="add-to-
cart?id=<%=m.getId() %>" class="btn btn-dark">Add to cart</a>

                                                                        <a href="order-
now?quantity=1&id=<%=m.getId() %>" class="btn btn-primary">Buy now</a>

```

```

                                </div>
                            </div>
                        </div>
                    </div>
                <%}
            }%>

        </div>
    </div>

    <% @include file="includes/footer.jsp"%>
</body>
</html>

```

3.2.2.6. orders.jsp

This file contains the content and design of the orders page

CODE:

```

<% @page import="java.util.*"%>
<% @page import = "food.app.connection.DbCon" %>
<% @page import = "food.app.model.*" %>
<% @page import = "food.app.dao.*" %>
<% @page import="java.text.DecimalFormat"%>
<% @ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<%

```

```

        DecimalFormat dcf = new DecimalFormat("#.##") ;
        request.setAttribute("dcf" , dcf) ;

        User auth = (User) request.getSession().getAttribute("auth") ;
        List<Order> orders = null ;

        if( auth != null ){

            request.setAttribute("auth" , auth) ;

            orders = new OrderDao(DbCon.getConnection()).userOrders(auth.getId())
;

        }else{

            response.sendRedirect("login.jsp") ;

        }

        ArrayList<Cart> cart_list = (ArrayList<Cart>) session.getAttribute("cart-list") ;
        List<Cart> cartFood = null ;

        if(cart_list != null){

            FoodDao fDao = new FoodDao(DbCon.getConnection()) ;

            cartFood = fDao.getCartFoods(cart_list) ;

            request.setAttribute("cart_list" , cart_list) ;

        }

%>
<!DOCTYPE html>
<html>

    <head>

        <title>Order Page</title>

        <% @include file = "includes/head.jsp" %>

    </head>

```

```

<body>

    <% @include file="includes/navbar.jsp"%>

    <div class = "container">

        <div class = "card-header my-3">All Orders</div>

        <table class = "table table-light">

            <thead>

                <tr>

                    <th scope = "col">Date</th>

                    <th scope = "col">Name</th>

                    <th scope = "col">Category</th>

                    <th scope = "col">Quantity</th>

                    <th scope = "col">Price</th>

                    <th scope = "col">Cancel</th>

                </tr>

            </thead>

            <tbody>

                <%

                    if(orders != null){

                        for(Order o : orders){ %>

                            <tr>

                                <td><%=o.getDate() %></td>

                                <td><%=o.getName() %></td>

                                <td><%=o.getCategory() %></td>

                                <td><%=o.getQuantity() %></td>

                                <td><%=o.getPrice() %></td>

```

```

                                <td><a class = "btn btn-sm btn-danger"
href = "cancel-order?id=<%=o.getOrderId()%>">Cancel</a></td>
                                </tr>
                                <%=>
                                }
                                %>
                                </tbody>
                                </table>
                                </div>

                                <%= @include file = "includes/footer.jsp" %>
                                </body>
                                </html>

```

3.2.2.7. register.jsp

This file contains the content and design of the registration page

CODE:

```

<%= @page import="java.util.*"%>
<%= @page import = "food.app.model.*" %>
<%= @ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<%
    User auth = (User) request.getSession().getAttribute("auth") ;
    if( auth != null ){
        request.setAttribute("auth" , auth) ;
    }

```

```

        ArrayList<Cart> cart_list = (ArrayList<Cart>) session.getAttribute("cart-list") ;
        if(cart_list != null){
            request.setAttribute("cart_list" , cart_list) ;
        }
    %>
<!DOCTYPE html>
<html>
    <head>
        <title>Registration Page</title>
        <% @include file = "includes/head.jsp" %>
    </head>
    <body>
        <% @include file="includes/navbar.jsp"%>
        <div class = "container">
            <div class = "card w-50 mx-auto my-5">
                <div class = "card-header text-center">User
Registration</div>
                <div class = "card-body">
                    <form action = "user-register" method = "post">
                        <!-- <div class = "form-group">
                            <label>User ID</label>
                            <input type = "number" class = "form-
control" name = "register-id" placeholder = "Enter Your ID" required>
                        </div> -->
                        <div class = "form-group">

```

```

        <label>User Name</label>

        <input type = "text" class = "form-
control" name = "register-name" placeholder = "Enter Your Username" required>

    </div>

    <div class = "form-group">

        <label>Email ID</label>

        <input type = "email" class = "form-
control" name = "register-email" placeholder = "Enter Your EmailID" required>

    </div>

    <div class = "form-group">

        <label>Password</label>

        <input type = "password" class = "form-
control" name = "register-password" placeholder = "*****" required>

    </div>

    <div class = "text-center">

        <button type = "submit" class = "btn btn-
primary">Register</button>

    </div>

</form>

</div>

</div>

</div>

    <% @include file = "includes/footer.jsp" %>

</body>

</html>

```


3.2.2.8. starter.jsp

This file contains the content and design of the starter page

CODE:

```
<% @page import="java.util.*"%>
<% @page import="food.app.connection.DbCon"%>
<% @page import="food.app.model.*"%>
<% @page import="food.app.dao.FoodDao"%>
<% @ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<%
    User auth = (User) request.getSession().getAttribute("auth");
    if (auth != null) {
        request.setAttribute("auth", auth);
    }else{
        response.sendRedirect("login.jsp") ;
    }

    FoodDao st = new FoodDao(DbCon.getConnection()) ;
    List<Starter> starters = st.getAllStarters() ;

    ArrayList<Cart> cart_list = (ArrayList<Cart>) session.getAttribute("cart-list") ;
    if(cart_list != null){
        request.setAttribute("cart_list" , cart_list) ;
    }
%>
```

```

<!DOCTYPE html>

<html>

    <head>

        <title>Home Page</title>

        <% @include file="includes/head.jsp"%>

    </head>

    <body>

        <% @include file="includes/navbar.jsp"%>

        <div class="container">

            <div class="card-header my-3">All Starters</div>

            <div class="row">

                <%if(!starters.isEmpty()){

                    for(Starter s : starters){ %>

                        <div class="col md-3 my-3">

                            <div class="card" style="width: 18rem; ">

                                    <div class="card-body">

                                        <h5 class="card-title"><%= s.getName()

%></h5>

                                        <h6 class = "price">Price: <%=

s.getPrice() %> Rs </h6>

                                        <h6 class = "category">Category: <%=

s.getCategory() %></h6>

                                        <div class = "mt-3 d-flex justify-content-
between">

```

```

                                <a href="add-to-
cart?id=<%=s.getId() %>" class="btn btn-dark">Add to cart</a>
                                <a href="order-
now?quantity=1&id=<%=s.getId() %>" class="btn btn-primary">Buy now</a>
                                </div>
                                </div>
                                </div>
                                </div>
                                <%}
                                }%>

                                </div>
                                </div>

                                <% @include file="includes/footer.jsp"%>
</body>
</html>

```

3.3. pom.xml

This file contains of all dependencies with mysql connector

CODE:

```

<project xmlns="http://maven.apache.org/POM/4.0.0"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
https://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>

```

```

<groupId>ateeth.food</groupId>
<artifactId>FoodApplication</artifactId>
<version>0.0.1-SNAPSHOT</version>
<packaging>war</packaging>
<dependencies>
    <!-- https://mvnrepository.com/artifact/mysql/mysql-connector-java -->
    <dependency>
        <groupId>mysql</groupId>
        <artifactId>mysql-connector-java</artifactId>
        <version>8.0.27</version>
    </dependency>
</dependencies>
<build>
    <plugins>
        <plugin>
            <artifactId>maven-compiler-plugin</artifactId>
            <version>3.8.1</version>
            <configuration>
                <release>16</release>
            </configuration>
        </plugin>
        <plugin>
            <artifactId>maven-war-plugin</artifactId>
            <version>3.2.3</version>
        </plugin>
    </plugins>
</build>

```

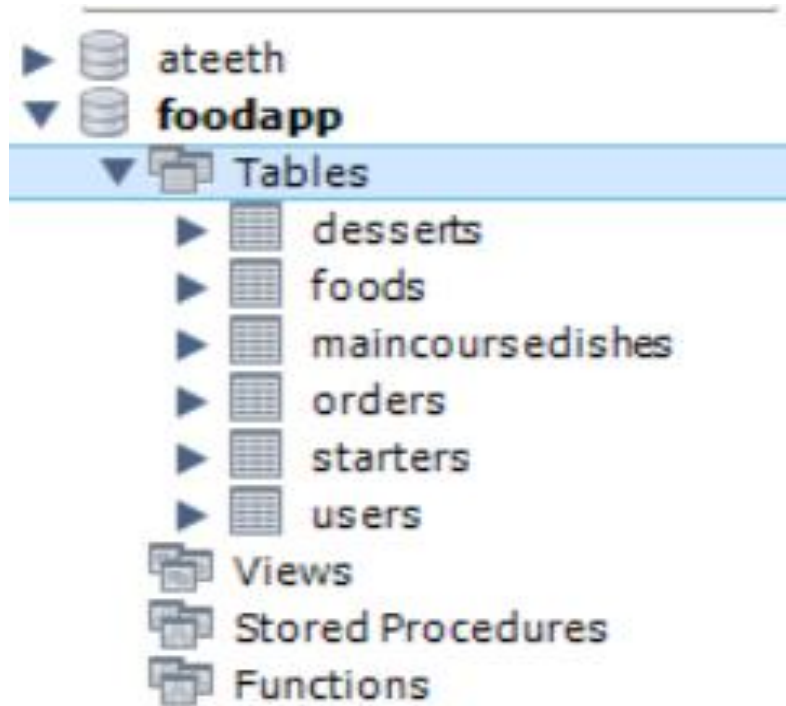
</plugins>

</build>

</project>

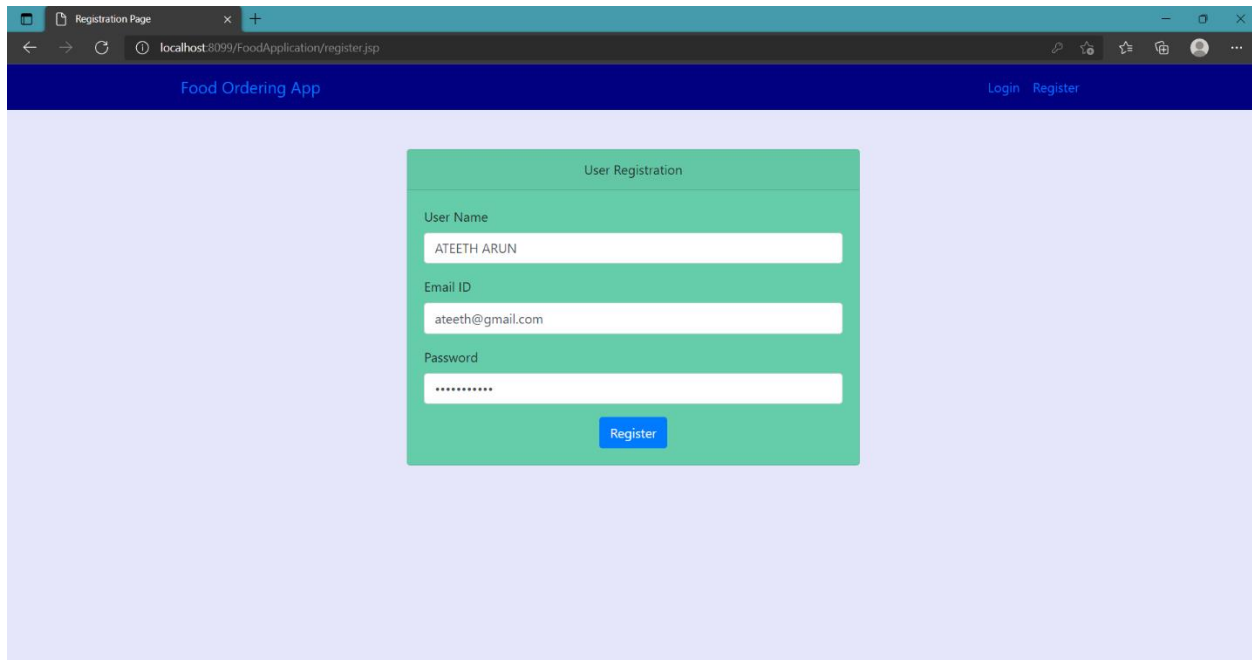
3.4. MySQL database where the tables are stored

A screenshot of the 6 tables which were used to store all data related to this application



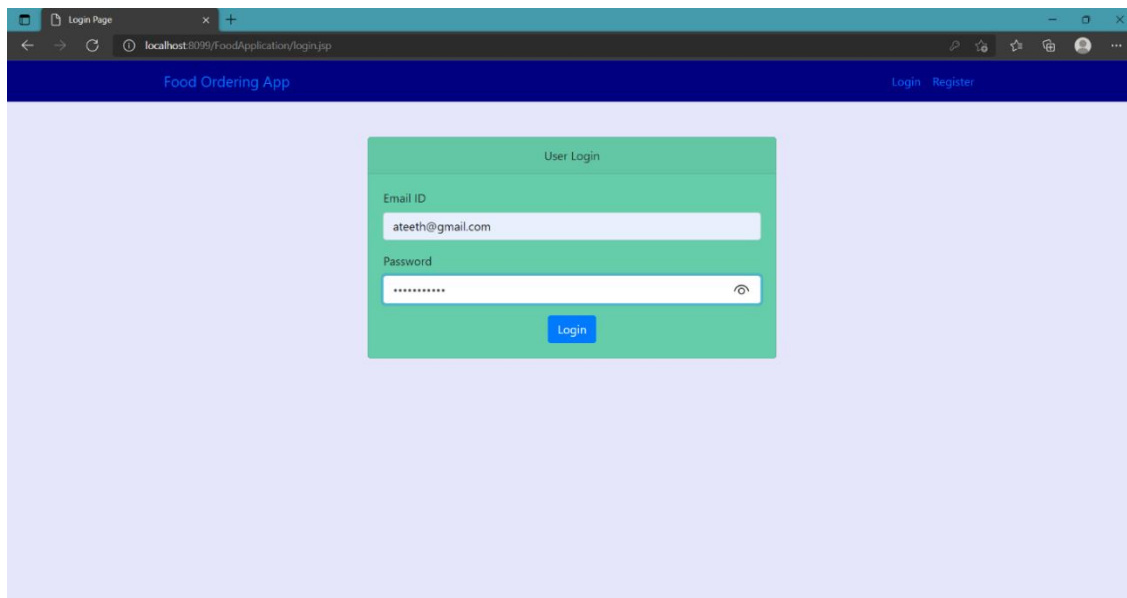
4. Screenshots of some of the views in this application

4.1. User Registration page



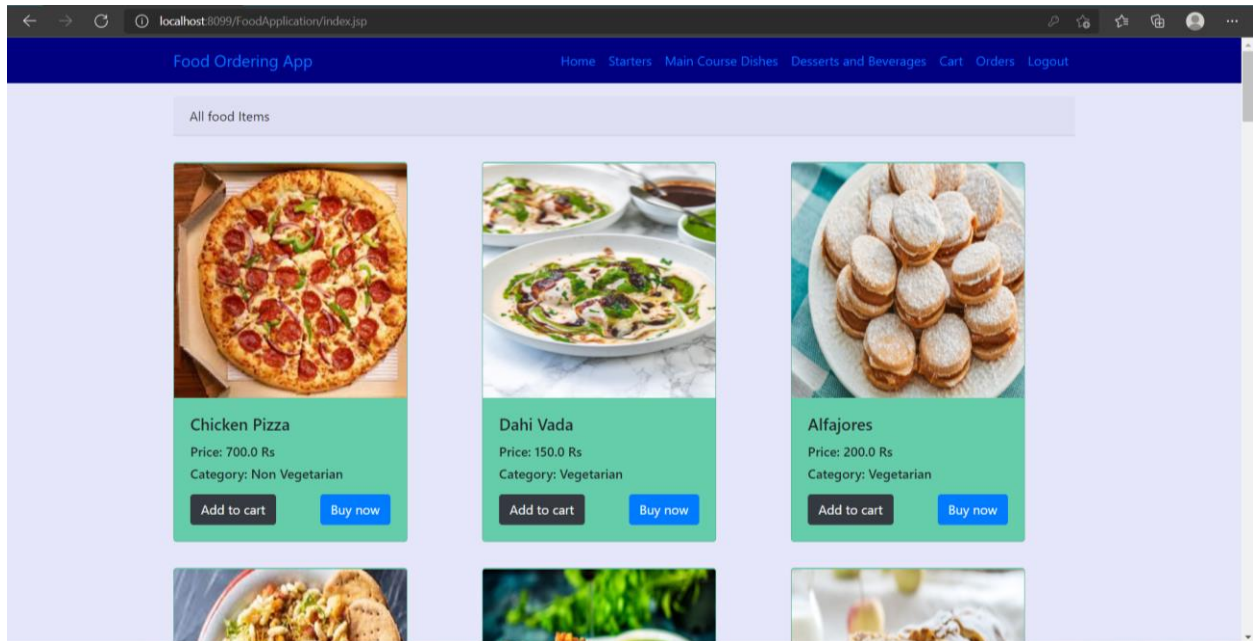
The screenshot shows a web browser window with the title "Registration Page" and the URL "localhost:8099/FoodApplication/register.jsp". The page features a dark blue header with the text "Food Ordering App" on the left and "Login Register" on the right. The main content area has a light purple background. In the center, there is a green rectangular form titled "User Registration". The form contains three input fields: "User Name" with the text "ATEETH ARUN", "Email ID" with the text "ateeth@gmail.com", and "Password" with masked characters "*****". A blue "Register" button is located at the bottom right of the form.

4.2. User Login Page

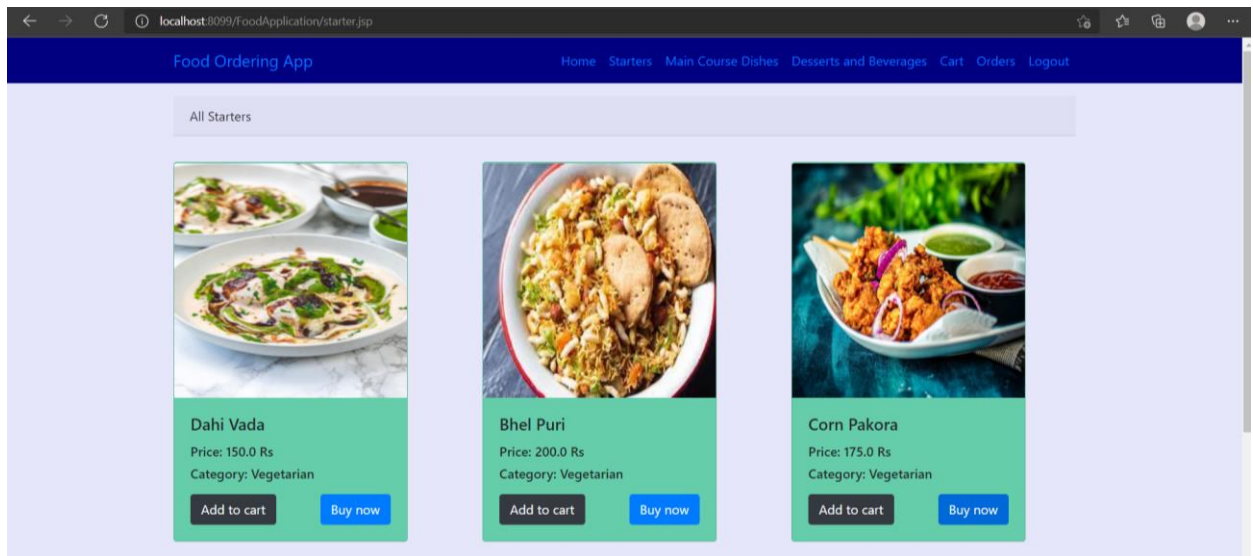


The screenshot shows a web browser window with the title "Login Page" and the URL "localhost:8099/FoodApplication/login.jsp". The page features a dark blue header with the text "Food Ordering App" on the left and "Login Register" on the right. The main content area has a light purple background. In the center, there is a green rectangular form titled "User Login". The form contains two input fields: "Email ID" with the text "ateeth@gmail.com" and "Password" with masked characters "*****" and an eye icon for toggling visibility. A blue "Login" button is located at the bottom right of the form.

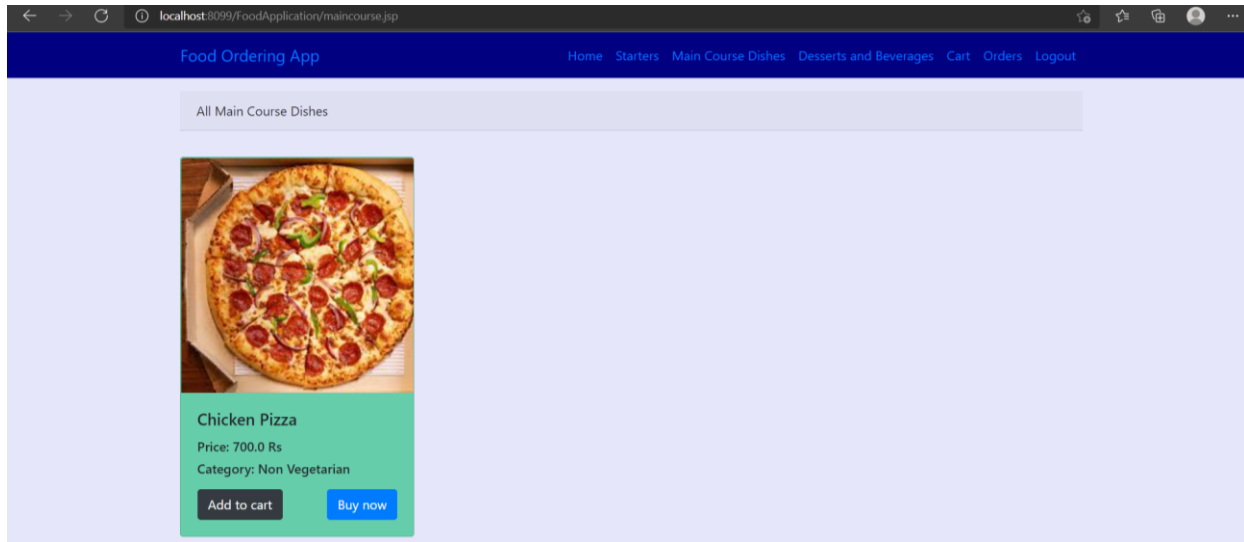
4.3. Home Page



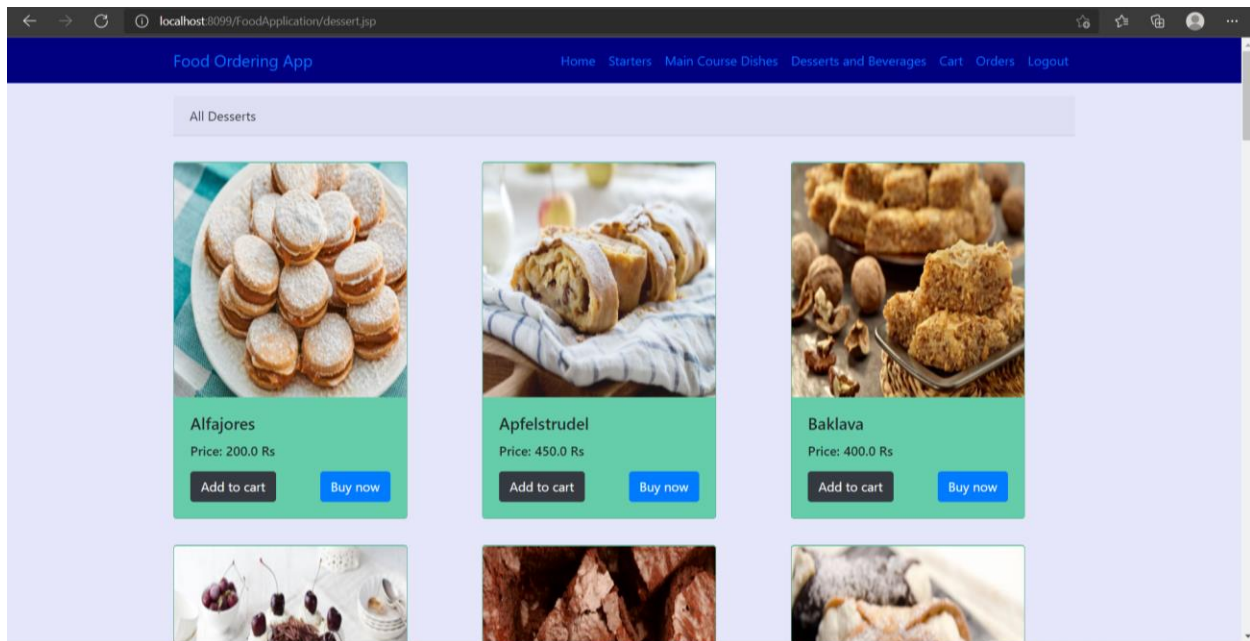
4.4. Starters Page



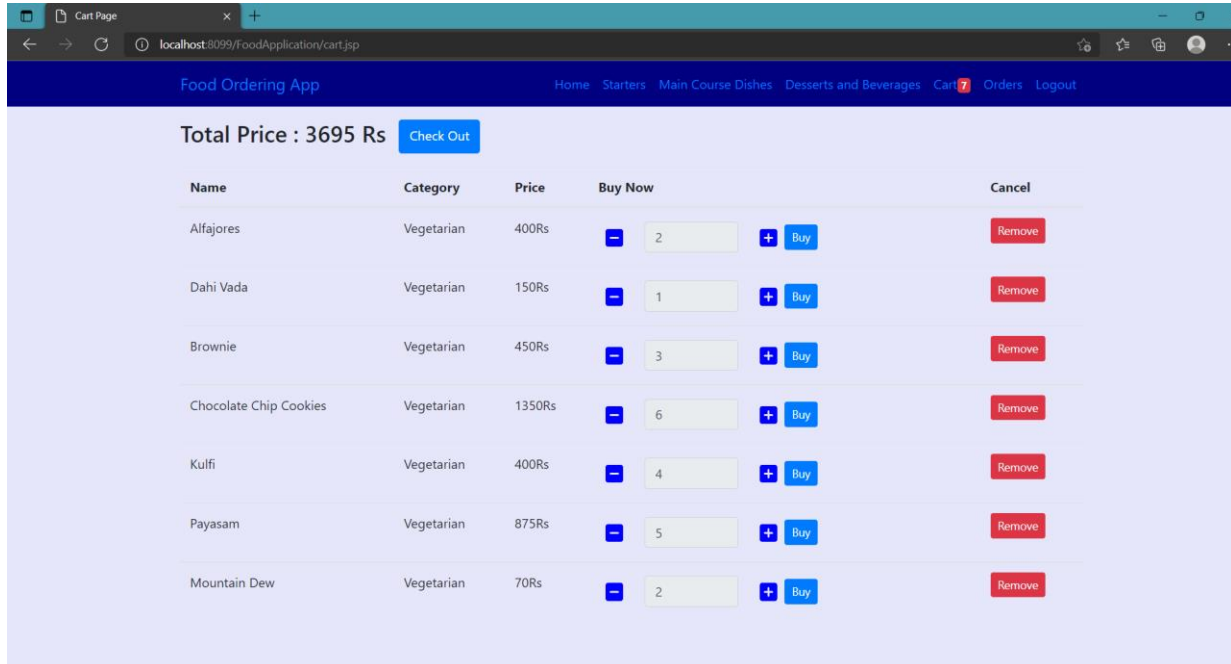
4.5. Main Course Dishes Page



4.6. Desserts and Beverages Page



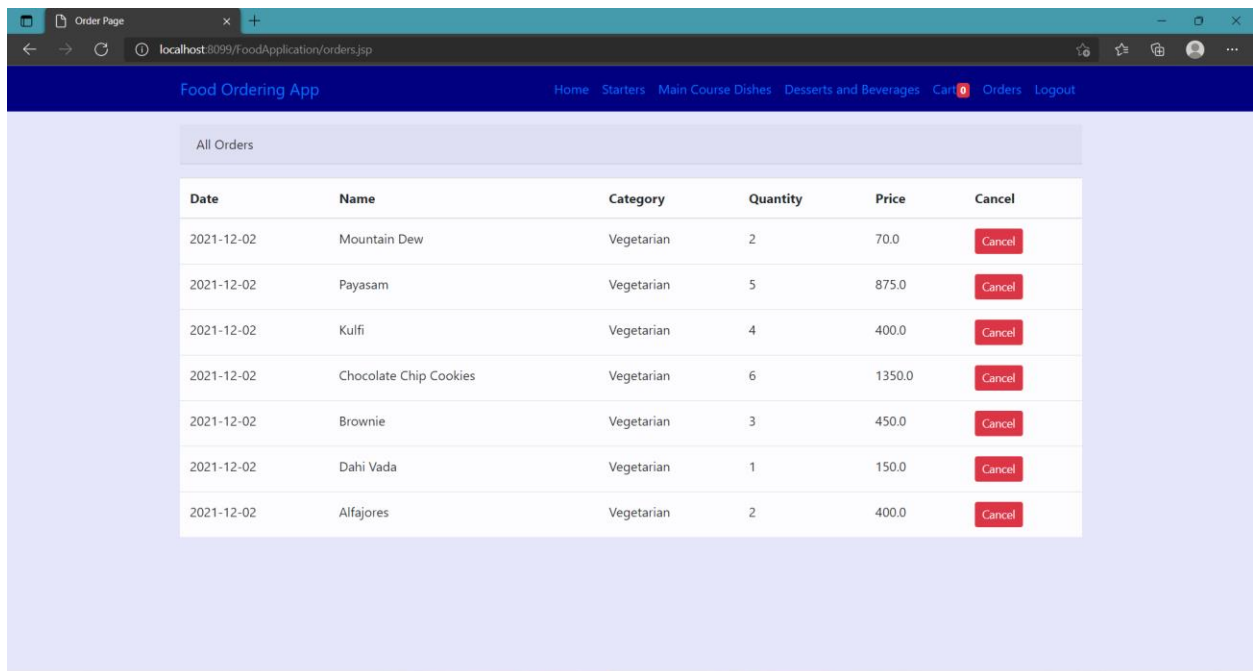
4.7. Cart Page



The screenshot shows the 'Cart Page' of a 'Food Ordering App'. The browser address bar indicates the URL is localhost:8099/foodApplication/cart.jsp. The app's navigation bar includes links for Home, Starters, Main Course Dishes, Desserts and Beverages, Cart (7 items), Orders, and Logout. The main content area displays the 'Total Price : 3695 Rs' and a 'Check Out' button. Below this is a table listing the items in the cart.

Name	Category	Price	Buy Now	Cancel
Alfajores	Vegetarian	400Rs	<input type="button" value="-"/> <input type="text" value="2"/> <input type="button" value="+"/> <input type="button" value="Buy"/>	<input type="button" value="Remove"/>
Dahi Vada	Vegetarian	150Rs	<input type="button" value="-"/> <input type="text" value="1"/> <input type="button" value="+"/> <input type="button" value="Buy"/>	<input type="button" value="Remove"/>
Brownie	Vegetarian	450Rs	<input type="button" value="-"/> <input type="text" value="3"/> <input type="button" value="+"/> <input type="button" value="Buy"/>	<input type="button" value="Remove"/>
Chocolate Chip Cookies	Vegetarian	1350Rs	<input type="button" value="-"/> <input type="text" value="6"/> <input type="button" value="+"/> <input type="button" value="Buy"/>	<input type="button" value="Remove"/>
Kulfi	Vegetarian	400Rs	<input type="button" value="-"/> <input type="text" value="4"/> <input type="button" value="+"/> <input type="button" value="Buy"/>	<input type="button" value="Remove"/>
Payasam	Vegetarian	875Rs	<input type="button" value="-"/> <input type="text" value="5"/> <input type="button" value="+"/> <input type="button" value="Buy"/>	<input type="button" value="Remove"/>
Mountain Dew	Vegetarian	70Rs	<input type="button" value="-"/> <input type="text" value="2"/> <input type="button" value="+"/> <input type="button" value="Buy"/>	<input type="button" value="Remove"/>

4.8. Orders Page on Clicking Checkout



The screenshot shows the 'Order Page' of the 'Food Ordering App'. The browser address bar indicates the URL is localhost:8099/foodApplication/orders.jsp. The app's navigation bar is the same as the cart page. The main content area displays 'All Orders' and a table listing the orders.

Date	Name	Category	Quantity	Price	Cancel
2021-12-02	Mountain Dew	Vegetarian	2	70.0	<input type="button" value="Cancel"/>
2021-12-02	Payasam	Vegetarian	5	875.0	<input type="button" value="Cancel"/>
2021-12-02	Kulfi	Vegetarian	4	400.0	<input type="button" value="Cancel"/>
2021-12-02	Chocolate Chip Cookies	Vegetarian	6	1350.0	<input type="button" value="Cancel"/>
2021-12-02	Brownie	Vegetarian	3	450.0	<input type="button" value="Cancel"/>
2021-12-02	Dahi Vada	Vegetarian	1	150.0	<input type="button" value="Cancel"/>
2021-12-02	Alfajores	Vegetarian	2	400.0	<input type="button" value="Cancel"/>