

CAPSTONE PROJECT

FOODIE APP

Group members:

1. Drishya Satheesh
2. Roshni Anna Shaju
3. Srikrishna Batchu

CONTENTS

Sl. No	Title	Page No
1.	Introduction	3
2.	Description of Existing System	3
3.	Objectives of the Project	4
4.	OOAD of Project	4-9
5.	Database design of Project	10-11
6.	Technologies used in Project	12
7.	Limitation and Scope of future research	12
8.	Working Code Screenshots	12-19
9.	References	20

FOODIE APP

1. INTRODUCTION

Foodie App is online food delivery application where Restaurant owners can register their details and users can find different cuisines based on location and dishes in this application. This application is used by Foodie App admins, Restaurants and Customers. This is a website designed primarily for use in the food delivery industry. The system also allows to quickly and easily manage an online menu which customers can browse and use to place orders with just few clicks.

In this system Restaurants can register under correct category such as Bar, Cafe, Cloud Kitchen, Takeaway with location, restaurant images and contact details. They can add and remove dishes under various categories such as Vegetarian, Non-Vegetarian, Chinese etc. with available time/day. Also they can modify details of the dishes like price, description, image, available time/day. Restaurants can view online orders received and deliver the items accordingly.

This system will allow an administrator to approve Restaurants only after that it will be visible to customers and can disable restaurant from listing. Admin can view feedbacks received from customers and view status of placed orders.

Using this app user can find restaurants by name, location, dish name, rating, hygiene etc. and can place order at a particular restaurant. Users can give rating on restaurants where they placed order.

2. DESCRIPTION OF EXISTING SYSTEM

This existing system is Web application designed using JSP and HTML and it gives insight about how GUI interacts with server-side language, Java, and finally with the Oracle database. The orders are not functional on Sunday. So our project will solve this problem because this deals with 24 hours per 7 weeks. The existing system does not functional on holiday as our system is functional on every day.

3. OBJECTIVES OF THE PROJECT

The objective of this project is to change from manual system to automated system. There are existing system which uses older technologies such as HTML, JQuery, PHP, Servlet JSP etc. In our system we are using full stack application with angular, spring boot etc., so that we can change the front end very easily and support many devices.

The specific objectives of this project are:

- Developing user friendly computer based delivery of foods.
- Enable to manage the user's request in timely manner.
- Computerized ordering of food.
- No need to go to buy food items.
- Can find variety of foods in single application.

The main advantage of this system is that it greatly simplifies the ordering process for both the customer and the restaurant and also greatly lightens the load on the restaurant's end, as the entire process of taking orders is automated.

4. OOAD OF PROJECT

4.1. Use Case Diagram

4.1.1 Actor description

Restaurant: Performs adding and removing dishes, modify details of the dishes and can view online orders received.

Administrator: Approve restaurants and can disable restaurant from listing. Admin can view feedbacks received from customers and view status of placed orders.

Customer: User can find restaurants and can place order at a particular restaurant. Users can give rating on restaurants where they placed order.

4.1.2 Use Case Textual Description

Use case name: Login use case

Identification: UC01

Description: use case to ensure security in system usage.

Actor: user: (Customer, Restaurant and Administrator.)

Precondition: the user must have username and password.

Post condition: user get access to the system according to their predefined system privilege and finally he/she logout or turn off the page.

Basic course of action:

1. User activates the system.
2. System response by displaying the login interfaces and prompts the user for the user ID and password.
3. User fills his or her user ID and password and presses enter.
4. System verifies user ID and Password.
5. User authenticated and gets access to the system.
6. System displays its home page.
7. Use case ends.

Alternative course of action (if user enters wrong user ID and / or password)

- A1. User is not authenticated and is denied access to the system.
- A2. System displays an incorrect user ID and password message.
- A3. System enables user to try again.
- A4. Use case returns to step 2 of main use case

Use case name: apply to register

Identification: UCO2

Description: use case to register new user and restaurant.

Actor: restaurant, customer.

Precondition: the restaurant/customer must be able to fill all the criteria perfectly and the restaurant/customer. has the ability to access the internet.

Post condition: a restaurant or new customer is registered.

Basic course of action

1. Not include login use case.

2. Restaurant/customer selects restaurant/customer link
3. Restaurant/ customer select form.
4. System display form.
5. Fills necessary data.
6. Submit
7. Use case ends

Alternative course of action (if applicant enters wrong information)

- A1. Restaurant/customer is not authenticated and is denied access to the system.
- A2. System displays an incorrect message.
- A3. Use case returns to step 5 of main use case

Use case name: Search Restaurant

Identifier: UC03

Description: use case to retrieve all restaurant in a particular location.

Actor: customer

Precondition: the customer must be a member.

Post condition: System display information about the restaurants available.

Basic course of action:

1. Include login use case.
2. Customer activates the user interface.
3. Customer selects restaurant from the displayed link.
4. System responds by displaying the browse user interface.
5. Customer selects the search criteria and enters the search key word.
6. System consults the database and displays the collection matching the search key word.
7. Use case ends.

Alternative course of action (user entered a search key word that doesn't exist in the system)

A1. System responds stating there is no resource matching the search in its result display.

A2.use case returned to step 5.

Use case name: Order Food

Identifier: UC04

Description: use case to order a food to browse from food collection.

Actor: customer

Precondition: the customer must be a member.

Post condition: system displays information about the available dishes.

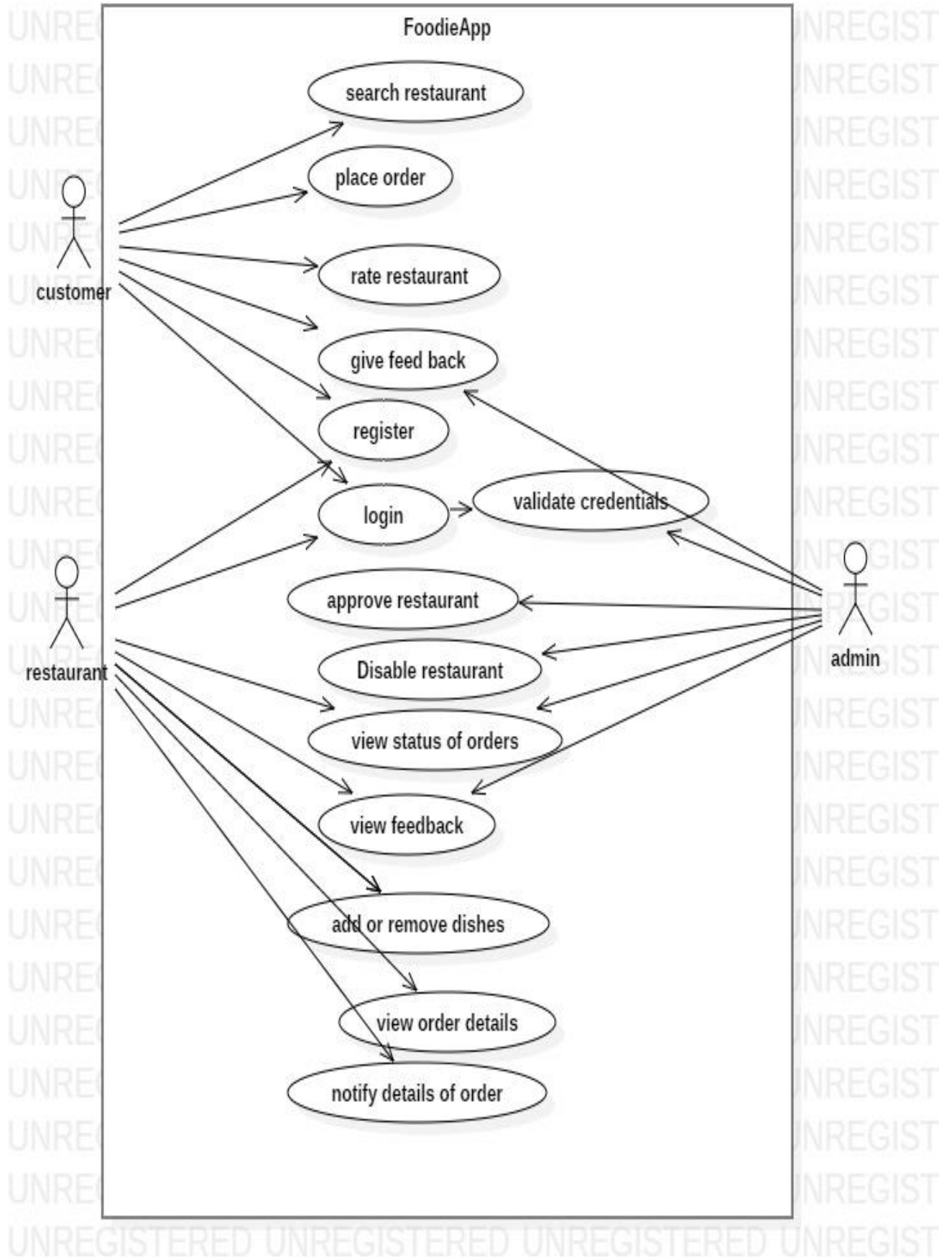
Basic course of action:

1. Include login use case.
2. Customer activates the order interface.
3. Customer selects brows order from the displayed link.
4. System responds by displaying the browse order interface.
5. Customer selects the order criteria and enters the order key word
6. System consults the database and displays message.
7. Use case ends.

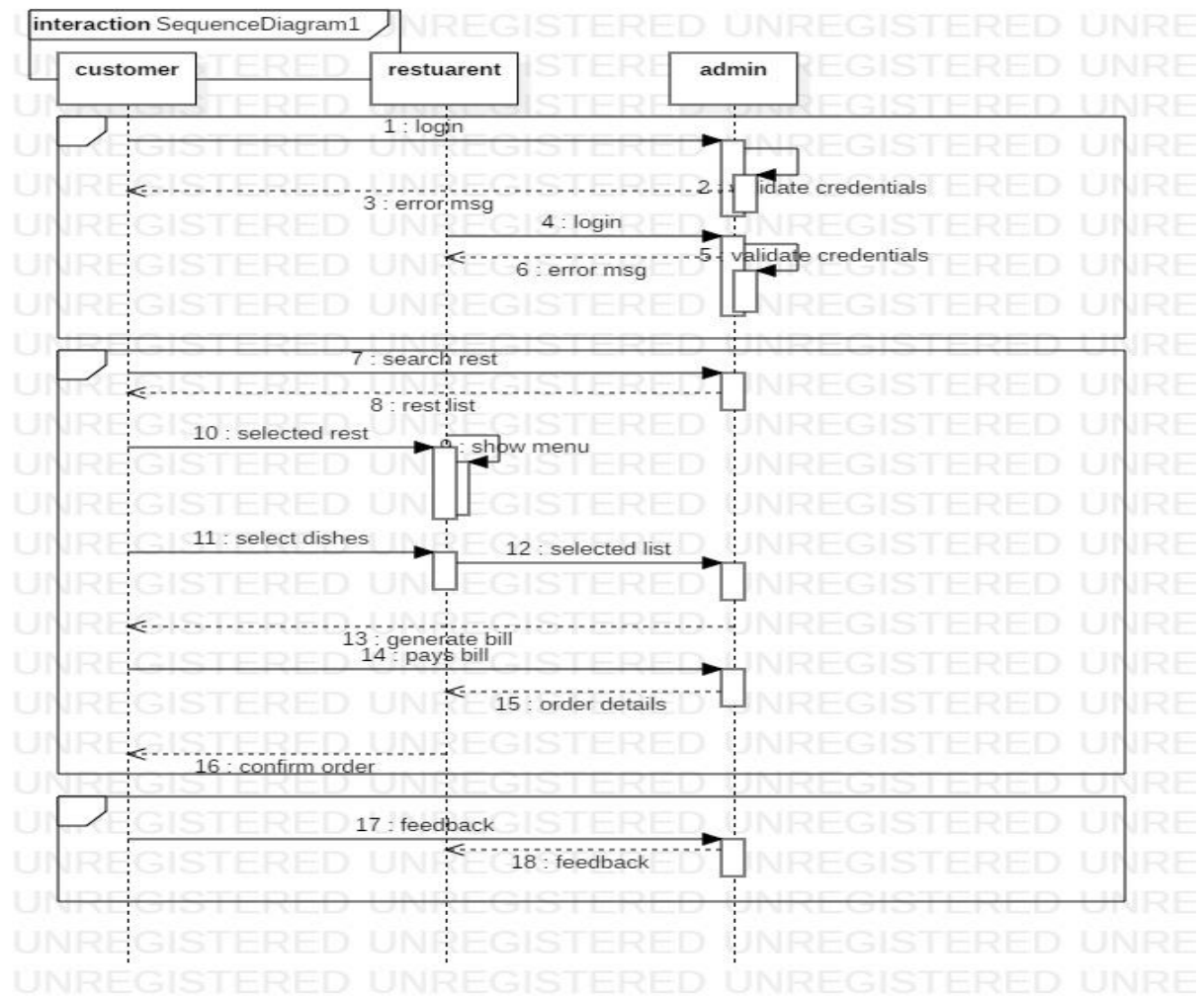
Alternative course of action (user entered an order key word that doesn't much in the system)

A1. System responds to fill the criteria again message.

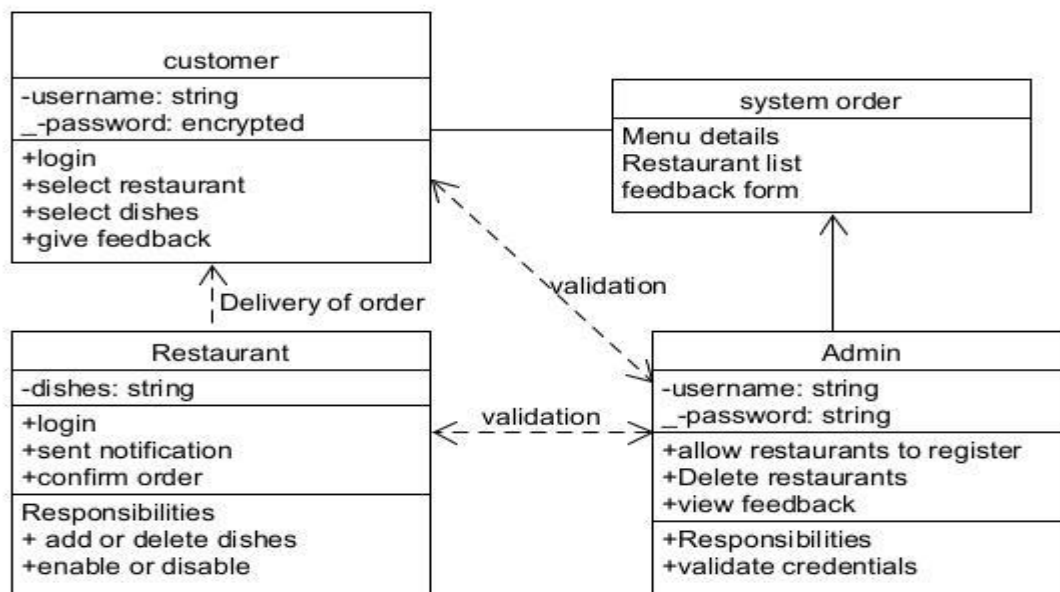
A2.use case returned to step 5 case.



4.2. Sequence Diagram



4.3. Class Diagram



5. DATABASE DESIGN OF PROJECT

Table Structure

1. **tbladmin** – information of administrator
 - admin_id – int, primary key
 - admin_username- varchar
 - admin_password- varchar

2. **tblrestaurant** – information of authorized restaurant
 - user_id – primary key, auto_increment, int
 - name – varchar
 - phone – int
 - email_address – varchar
 - username – varchar
 - password – varchar
 - confirm_password - varchar.
 - location – varchar
 - image – file

3. **tbluser** – information of authorized user of the system.
 - user_id – primary key, auto_increment ,int
 - name – varchar
 - phone – int
 - email_address – varchar
 - username – varchar
 - password – varchar
 - confirm_password - varchar.
 - Address – varchar

4. **tblmenu** – this table will store the list of menu/food.
 - menu_id – primary key, auto_increment , int
 - menu_name – varchar.

- price –float.
- menu_image – file
- available_quantity – int
- available_time – time

5. **tblorder** – customer orders are stored in the tblorder.

- order_id – primary key, auto_increment , int
- customer_id – int.
- order_date – date.
- total_amount –float
- order_status – boolean
- delivery_time – time.
- menu_id –int.
- amount –float
- quantity-int

6. **tblfeedback**- customer feedback

- menu_id-int
- menu_name-varchar
- date-date
- content-varchar

7. **tblrating** – rating and comments of the customers are stored in this table.

- rating_id – primary key, auto_increment
- menu_id –int.
- score – int
- remarks- varchar.
- date_recorde- date
- customer_id –int.

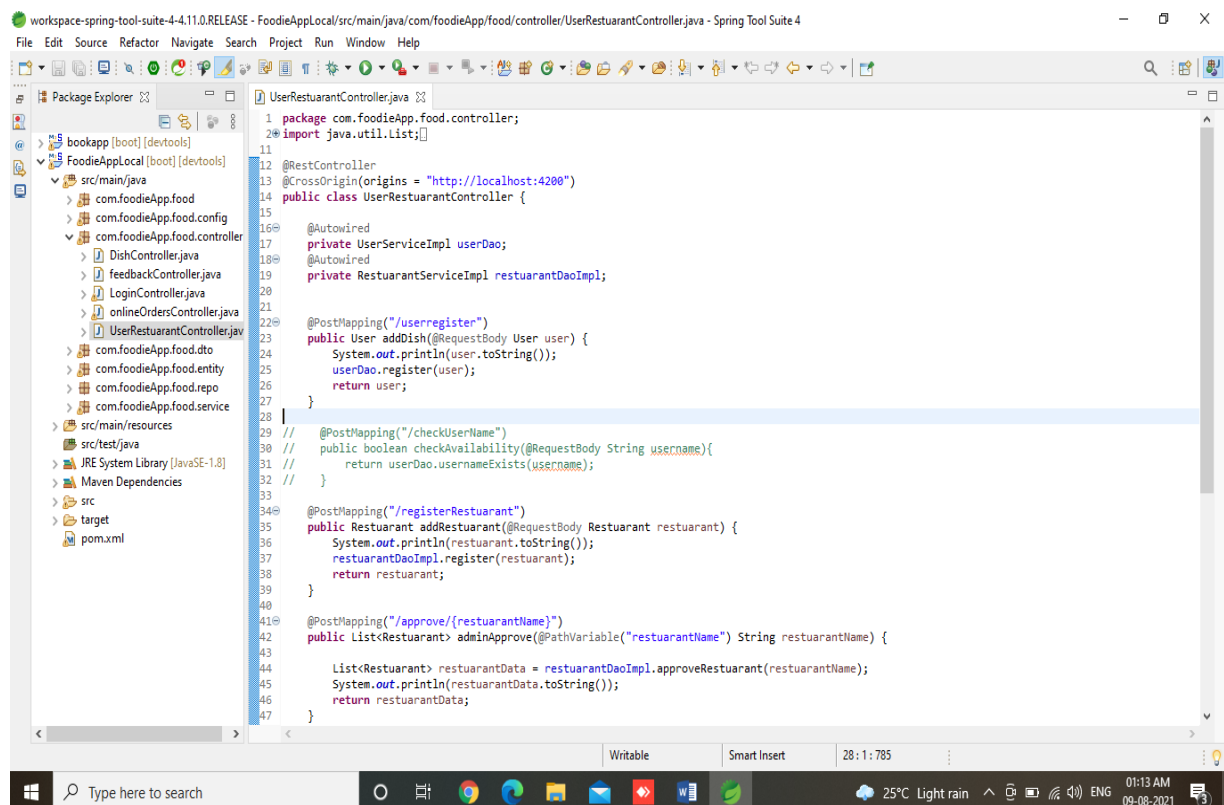
6. TECHNOLOGIES USED IN PROJECT

- Front End – Angular, CSS, Bootstrap etc.
- Back End – Java 11, Spring Boot, Micro service, JWT, AWS etc.

7. LIMITATIONS AND SCOPE OF FUTURE RESEARCH

Right now we have not integrated payment gateway as PayPal as it will take more time. If we get more time we will do the payment gateway as to make our project to complete application in running stage. As a part of future research we will try to develop our system using modern technologies.

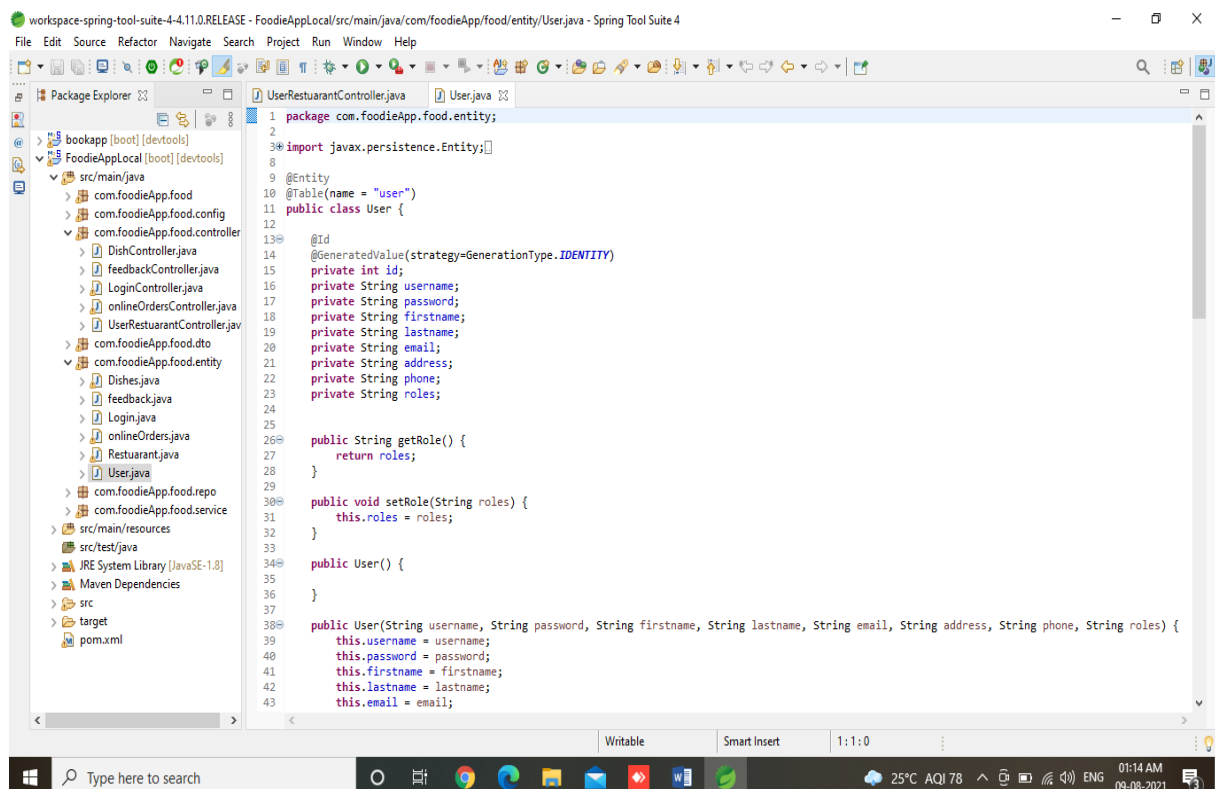
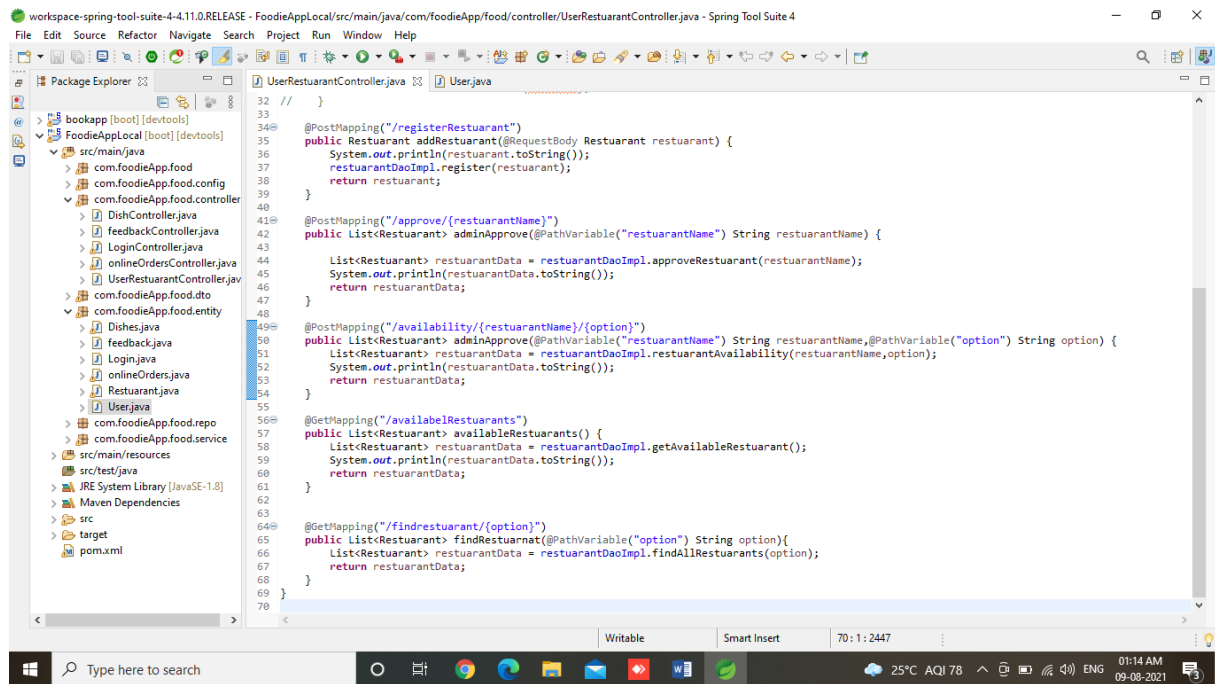
8. WORKING CODE SCREENSHOTS



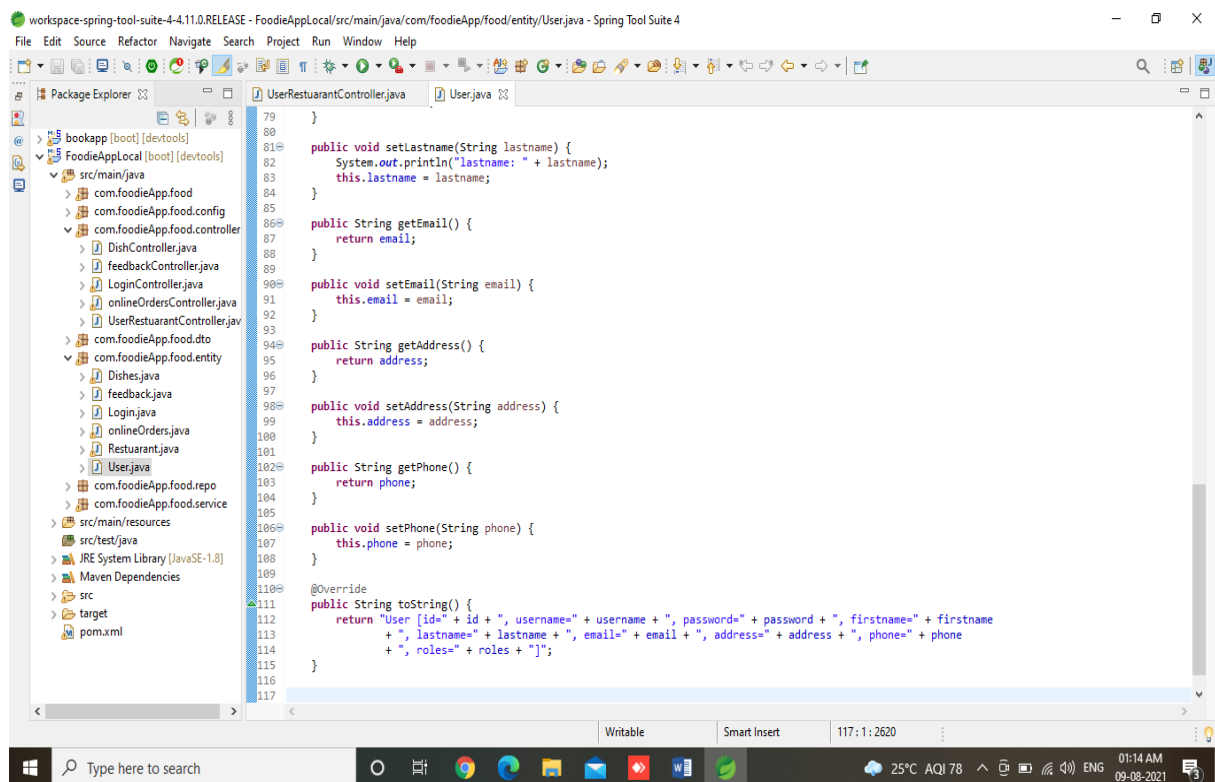
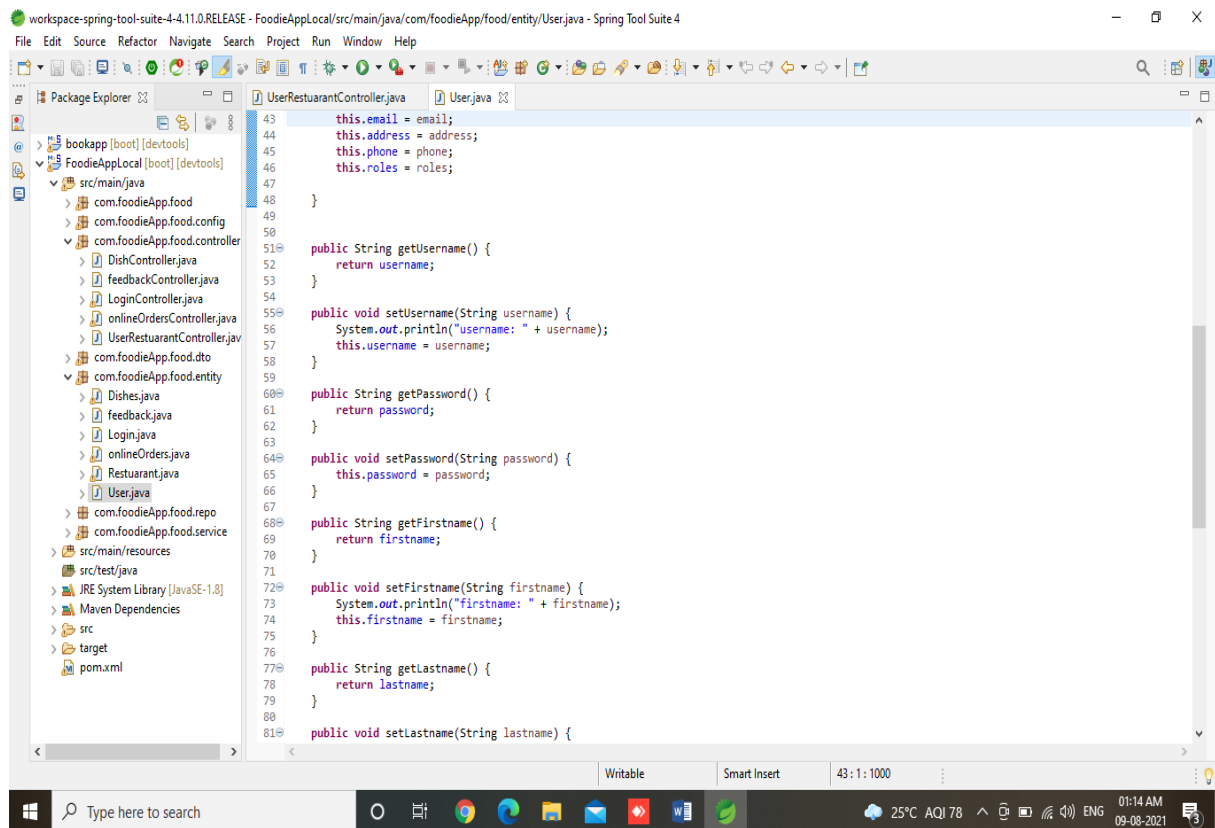
```
1 package com.foodieApp.food.controller;
2 import java.util.List;

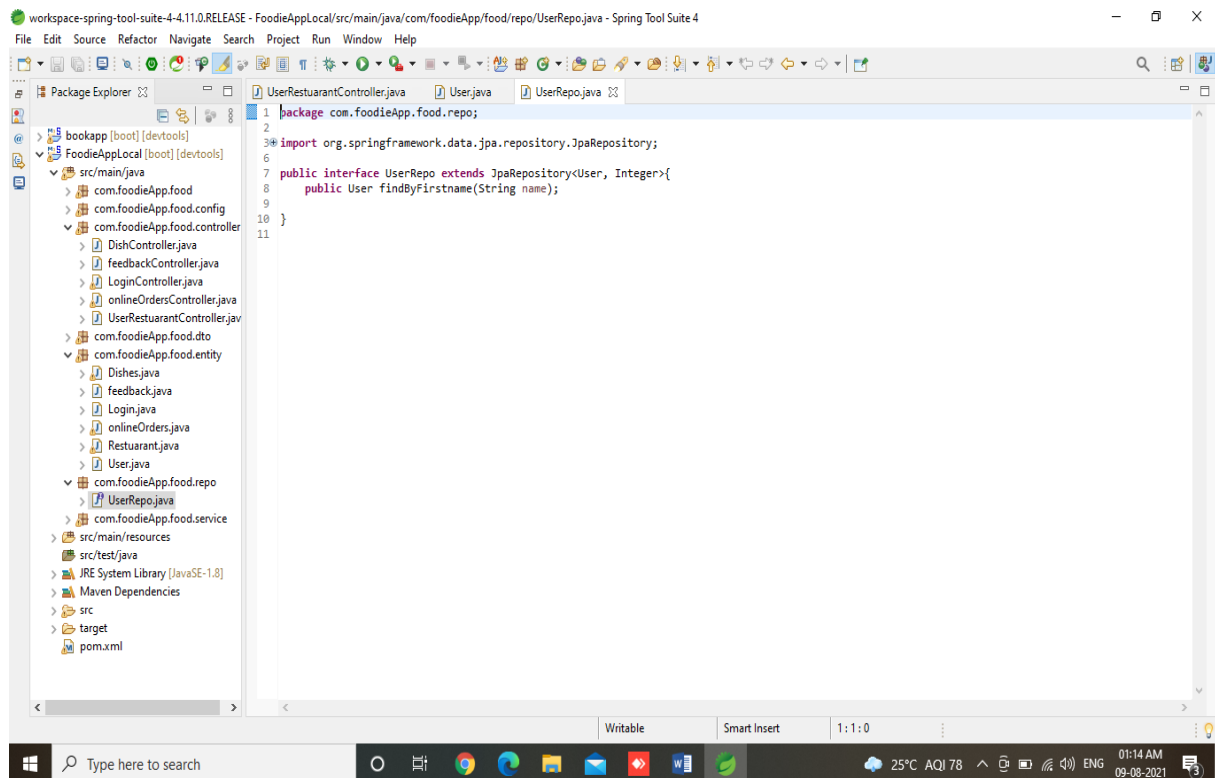
11
12 @RestController
13 @CrossOrigin(origins = "http://localhost:4200")
14 public class UserRestaurantController {
15
16     @Autowired
17     private UserServiceImpl userDao;
18     @Autowired
19     private RestaurantServiceImpl restaurantDaoImpl;
20
21
22     @PostMapping("/userregister")
23     public User addDish(@RequestBody User user) {
24         System.out.println(user.toString());
25         userDao.register(user);
26         return user;
27     }
28
29     // @PostMapping("/checkUserName")
30     // public boolean checkAvailability(@RequestBody String username){
31     //     return userDao.usernameExists(username);
32     // }
33
34     @PostMapping("/registerRestaurant")
35     public Restaurant addRestaurant(@RequestBody Restaurant restaurant) {
36         System.out.println(restaurant.toString());
37         restaurantDaoImpl.register(restaurant);
38         return restaurant;
39     }
40
41     @PostMapping("/approve/{restaurantName}")
42     public List<Restaurant> adminApprove(@PathVariable("restaurantName") String restaurantName) {
43
44         List<Restaurant> restaurantData = restaurantDaoImpl.approveRestaurant(restaurantName);
45         System.out.println(restaurantData.toString());
46         return restaurantData;
47     }
48 }
```

8.1 UserRestaurantController.java

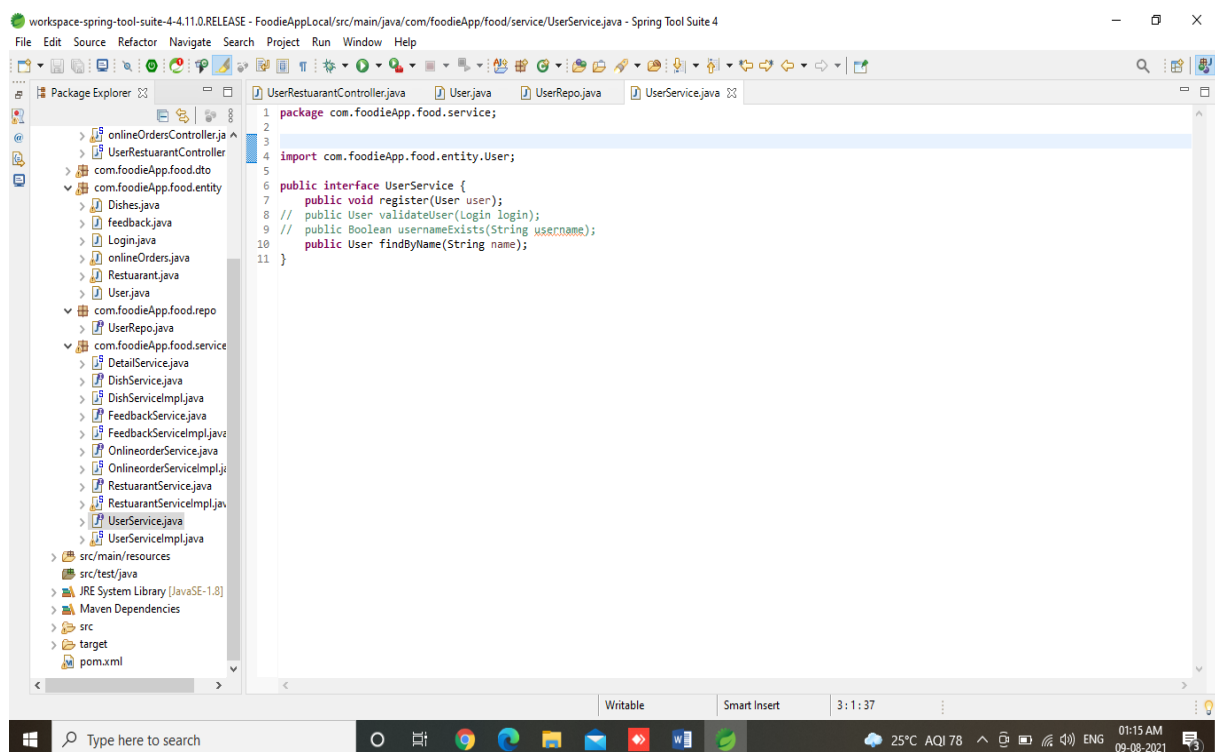


8.2 User.java

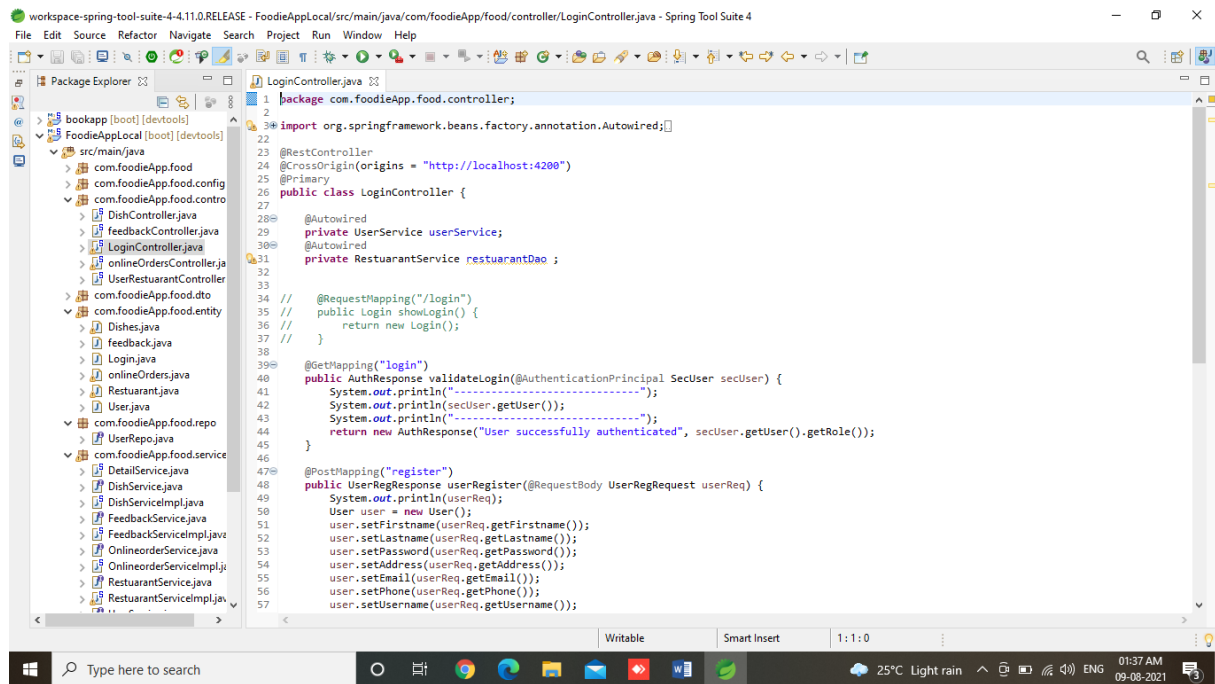




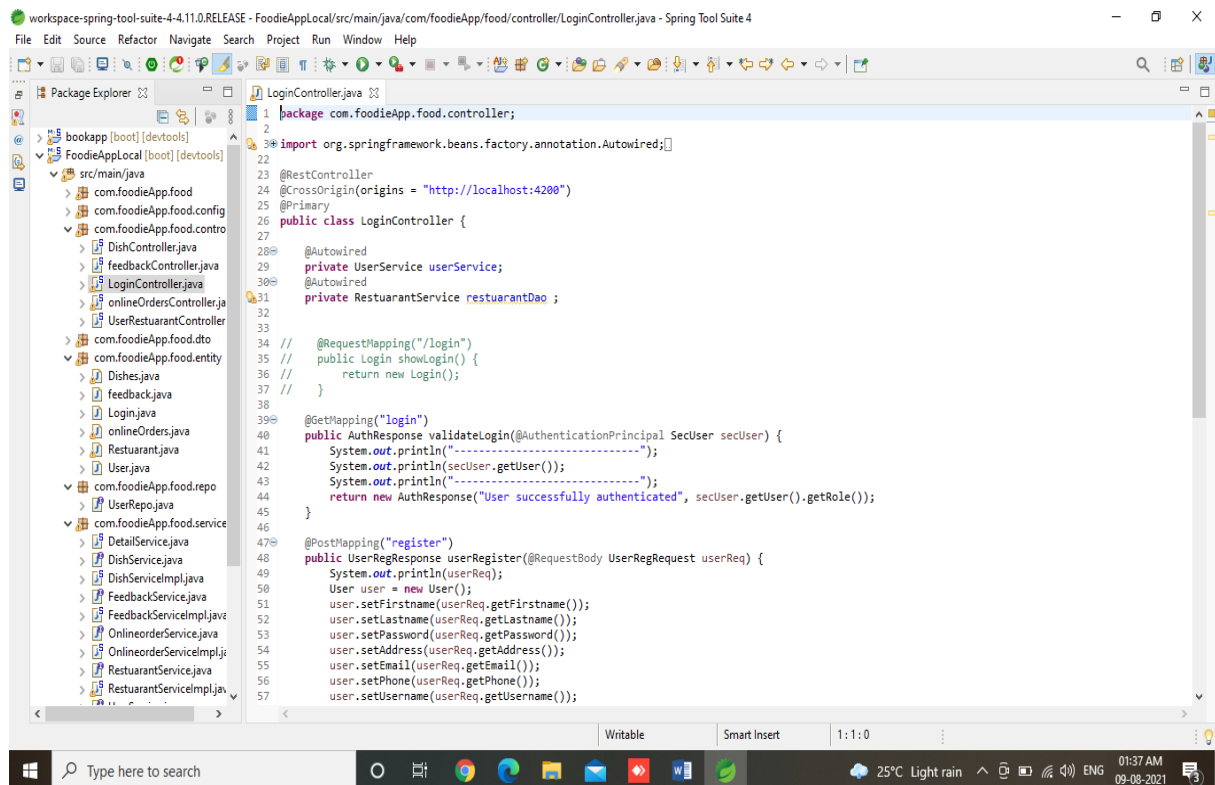
8.3 UserRepo.java



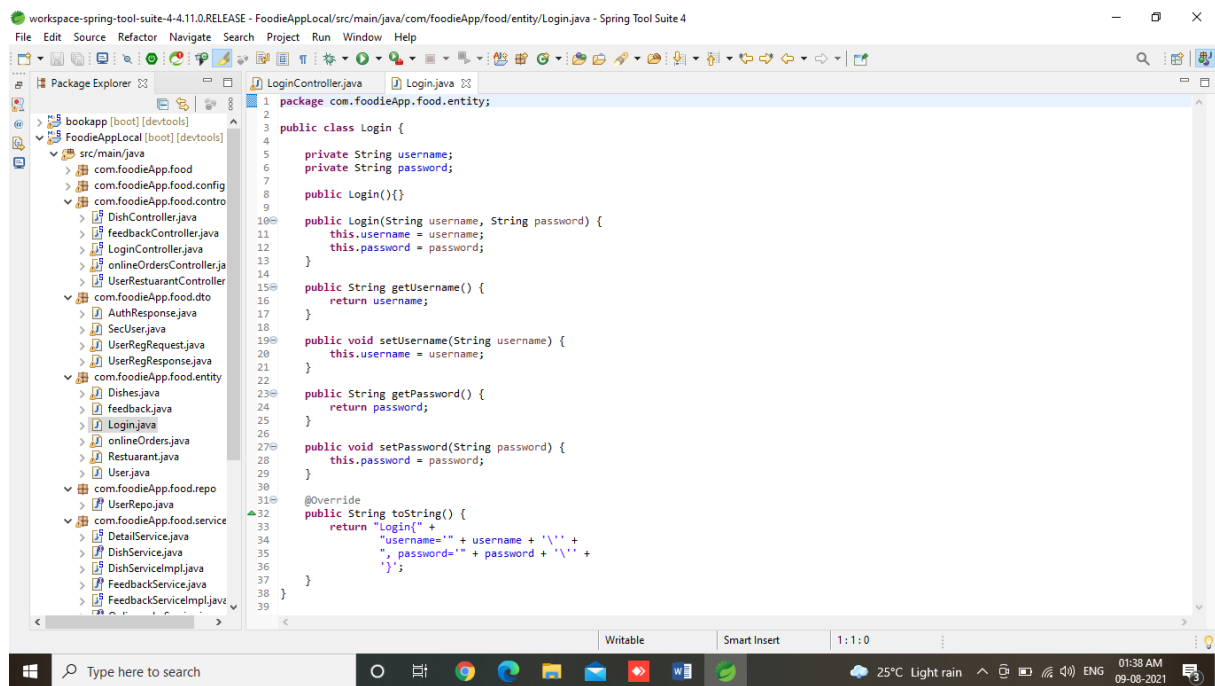
8.4 UserService.java



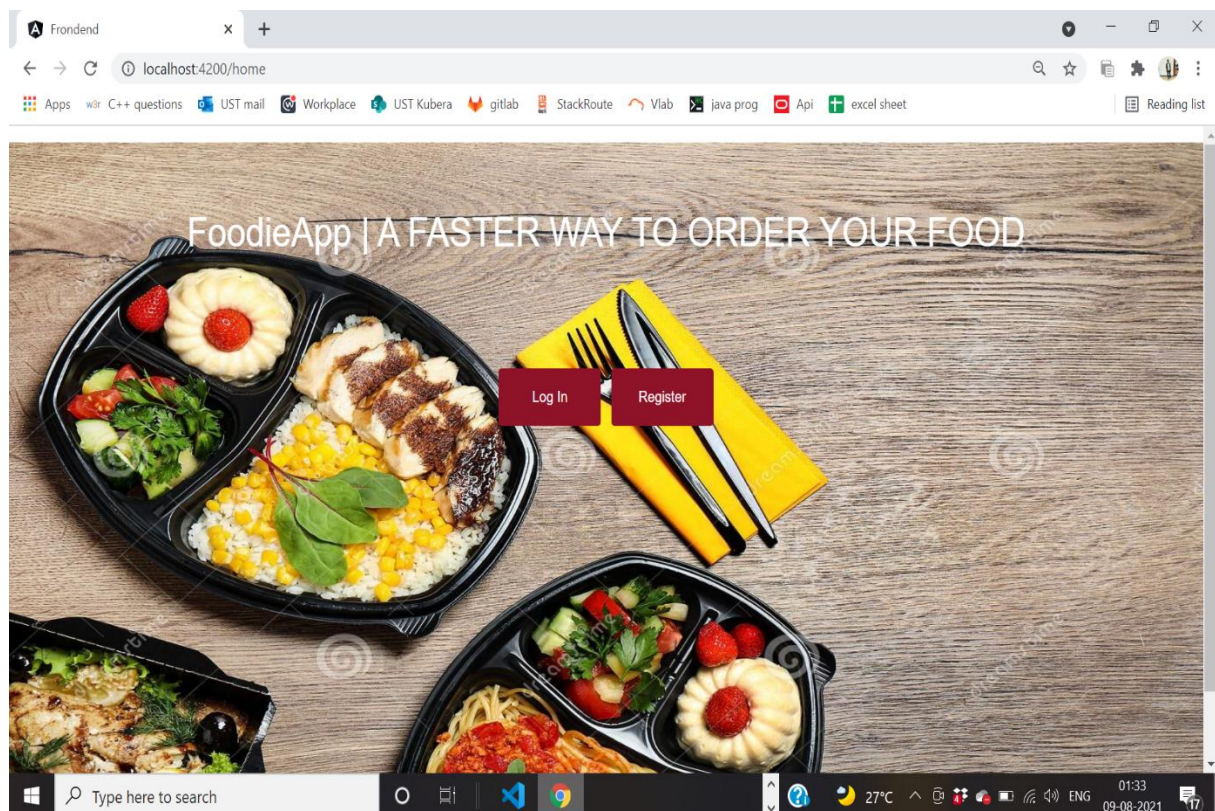
8.5 UserServiceImpl.java



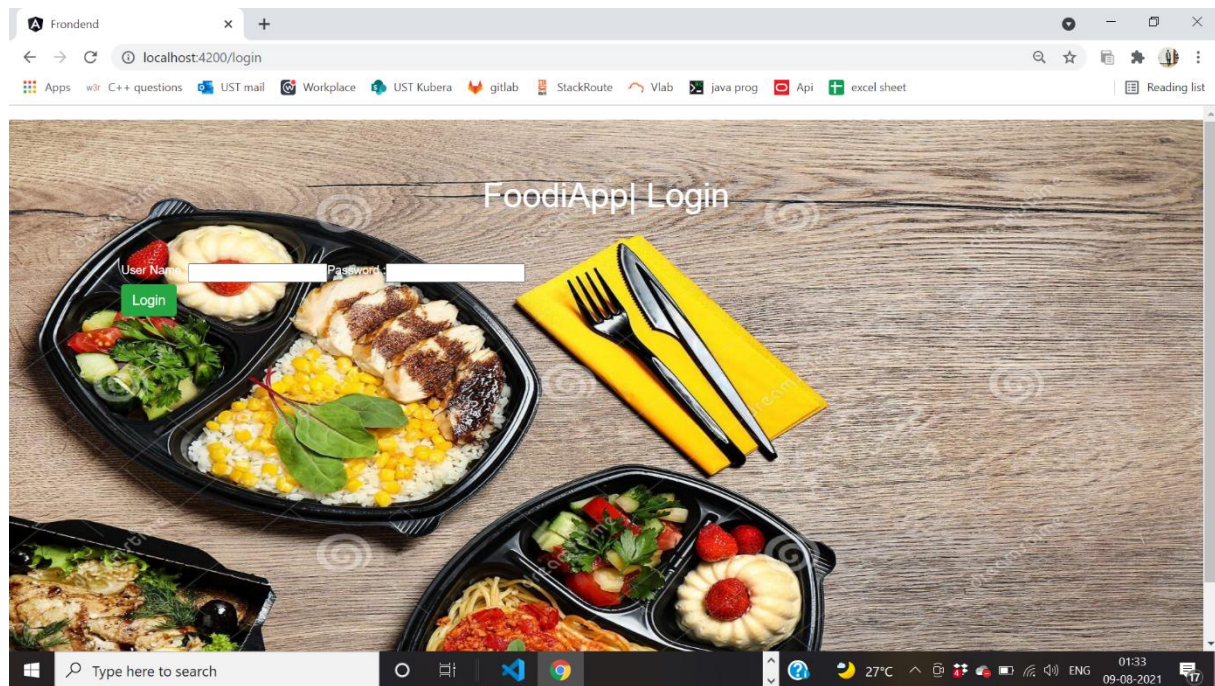
8.6 LoginController.java



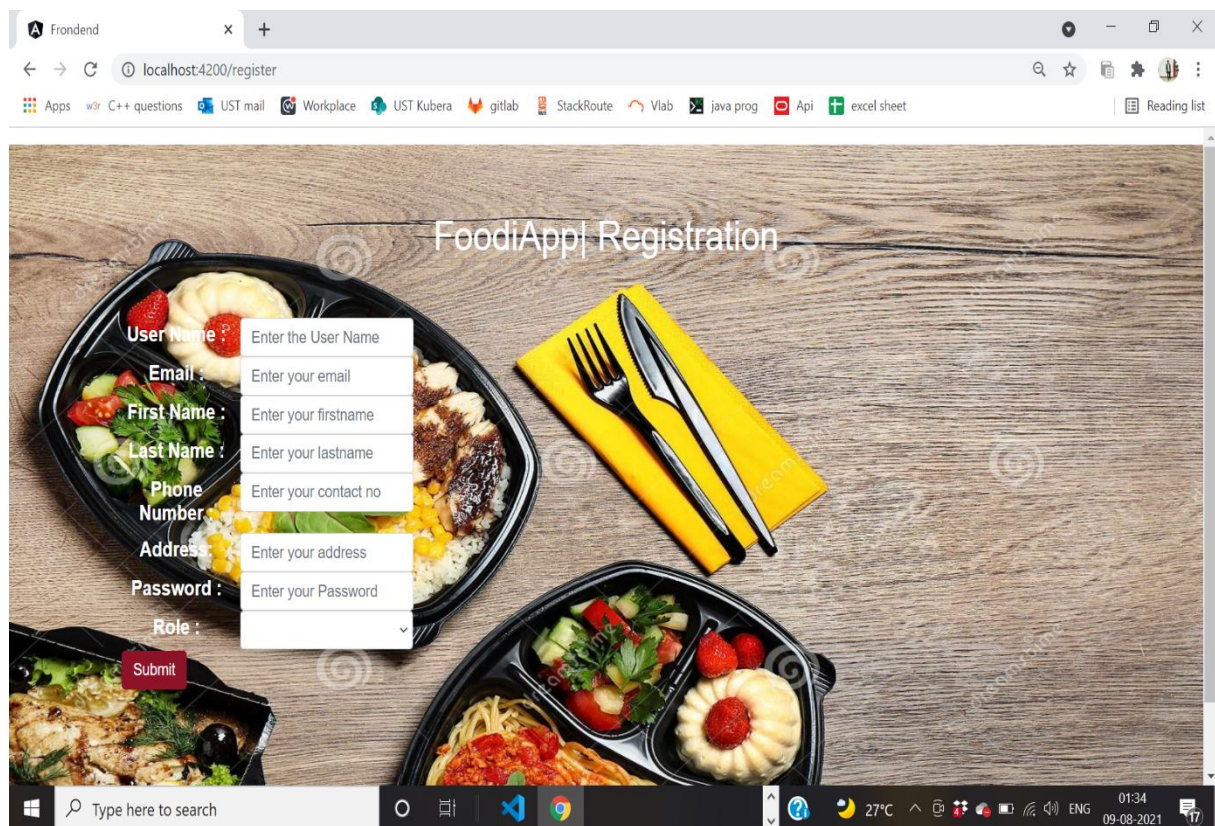
8.7 Login.java



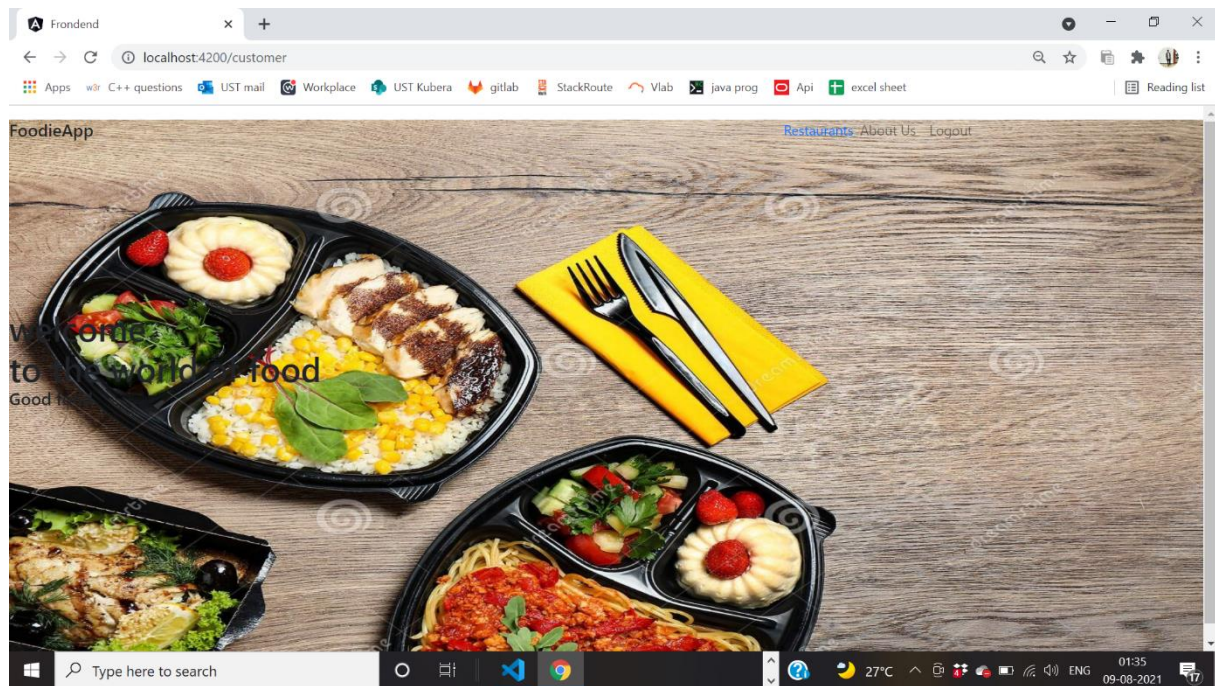
8.8 Homepage



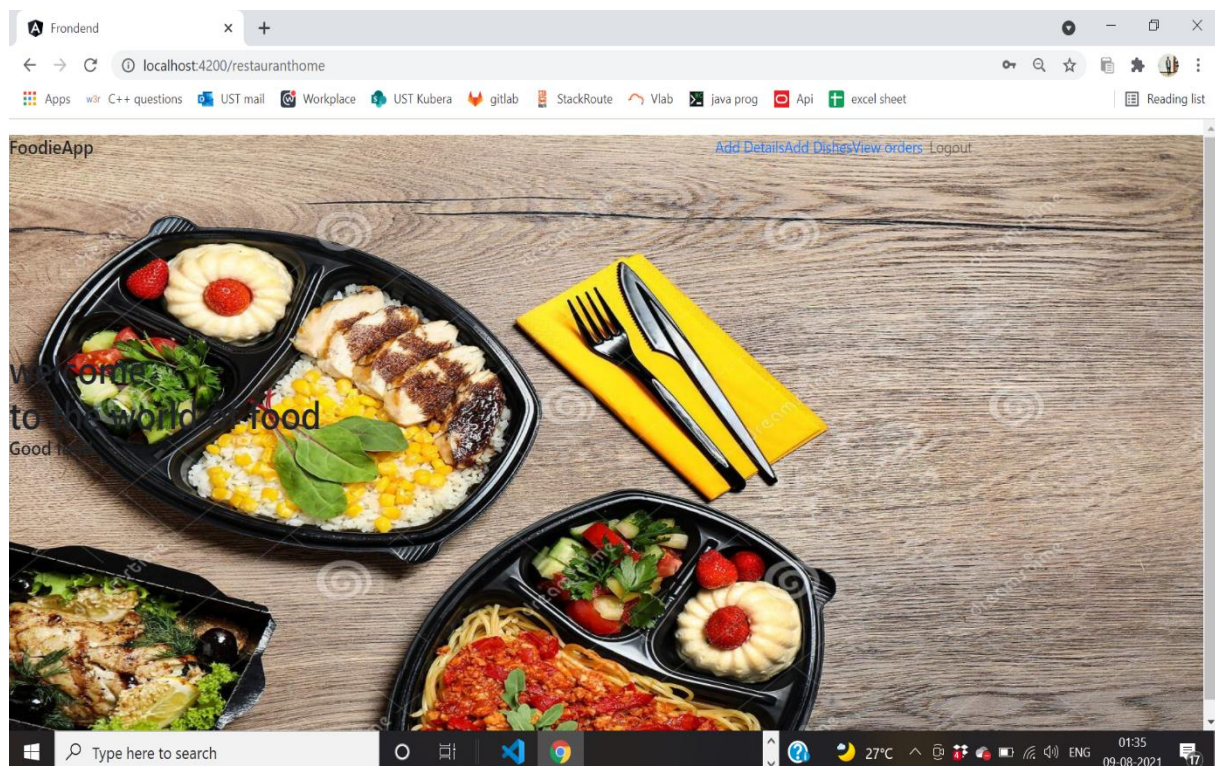
8.9 Login



8.10 Registration



8.11 Customer Homepage



8.12 Restaurant Homepage

9. REFERENCES

- <https://www.youtube.com/watch?v=USQvn3O2mBY&list=PLJ-lrQx0LAbxJB6B7go8Tj39xD33fYDP6>
- <https://www.bezkoder.com/angular-11-spring-boot-jwt-auth/>