MODERN APPLICATION DEVELOPMENT
JAVA SPRING BOOT

PROJECT TITLE: FOOD ORDERING SYSTEM

TEAM MEMBERS:

20BCE2339 - YOGAVARSHNI D
20BCE0897 - KISLAY KRISHNAN
20BBS0181 - J TARUN RAMA CHANDRA RAJU
20BIT0374 - TRISHIT GUPTA

INTRODUCTION:

The Food Ordering System is a web-based application designed specifically for university students. With Spring Boot, HTML, CSS, JS, and MySQL, we bring restaurants inside your campus to deliver meals to your hostels. Our intuitive interface lets you browse menus and place orders effortlessly. We offer a diverse range of culinary options to satisfy all tastes and dietary preferences. Our reliable delivery system ensures prompt service and hot, fresh meals. Backed by a secure MySQL database, your data is safely stored. Experience convenience, choice, and quality with our comprehensive solution and enjoy delicious food without leaving your hostel!

SYSTEM ARCHITECTURE:

The Food Ordering System follows a client-server architecture. The frontend is built using HTML, CSS, and JavaScript, providing an interactive user interface for browsing menus and placing orders. The backend is powered by Spring Boot, a Java-based framework known for its efficiency and scalability. It handles the business logic, database interactions, and communication with external services.

TECHNOLOGIES USED:

1. Spring Boot: Used as the backend framework for implementing the server-side logic and handling HTTP requests and responses.
2. HTML: Used to structure the webpages and create the user interface elements.
3. CSS: Used to style the HTML elements and enhance the visual appearance of the application.
4. JavaScript: Used for client-side scripting, enabling dynamic and interactive features on the webpages.
5. MySQL: Used as the relational database management system to store and manage data related to restaurants, menus, orders, and customer information.

KEY FEATURES:

1. Restaurant Selection: Students can browse through a list of available restaurants within the university campus.
2. Menu Viewing: Each restaurant provides a detailed menu with food items, descriptions, and prices.
3. Order Placement: Students can add desired items to their cart, specify customization options, and place orders.
4. Payment Integration: The system supports secure online payment options for a seamless transaction process.
5. User Authentication: Students can create accounts, log in securely, and manage their personal information.
6. Admin Dashboard: An administrative interface allows restaurants to manage menus, inventory, and order processing.

Literature Survey

Title	Author	Abstract	Tools/Methodology	Link
Netfood: A Software System for Food Ordering and Delivery	Cristina-Edina Domokos, Károly Simon	The Netfood project aims to develop a delivery-oriented order management system that allows users to order from multiple restaurants simultaneously and helps the work of the delivery personnel in tracking the orders. There are many applications for food ordering from local restaurants in particular cities. Some systems support group orders too, but these are usually restaurant-oriented: an order can only contain items requested from a single restaurant. This model works well where the delivery can be solved separately by the restaurants, but there are settlements (e.g. small towns) where multiple restaurants are working together with the same delivery company.	The Android Software Development Kit, Gradle build system,Retrofit	https://edu.codespring.ro/wp-content/uploads/2019/06/NetFood_SISY_2018.pdf
Live Event Food Service	Shraddha Barve , Bhavik Sakhiya	software where any person sitting at the Live Event can order food online any hustle. The food ordered will directly be brought to the person's seat. The process consists of a customer choosing the food of their choice, scanning the menu items, choosing an item, and finally choosing for pick-up or delivery. Payment is then administered by paying	Android Studio, Php MyAdmin	http://www.ijsred.com/volume3/issue2/IJSRED-V3I2P32.pdf

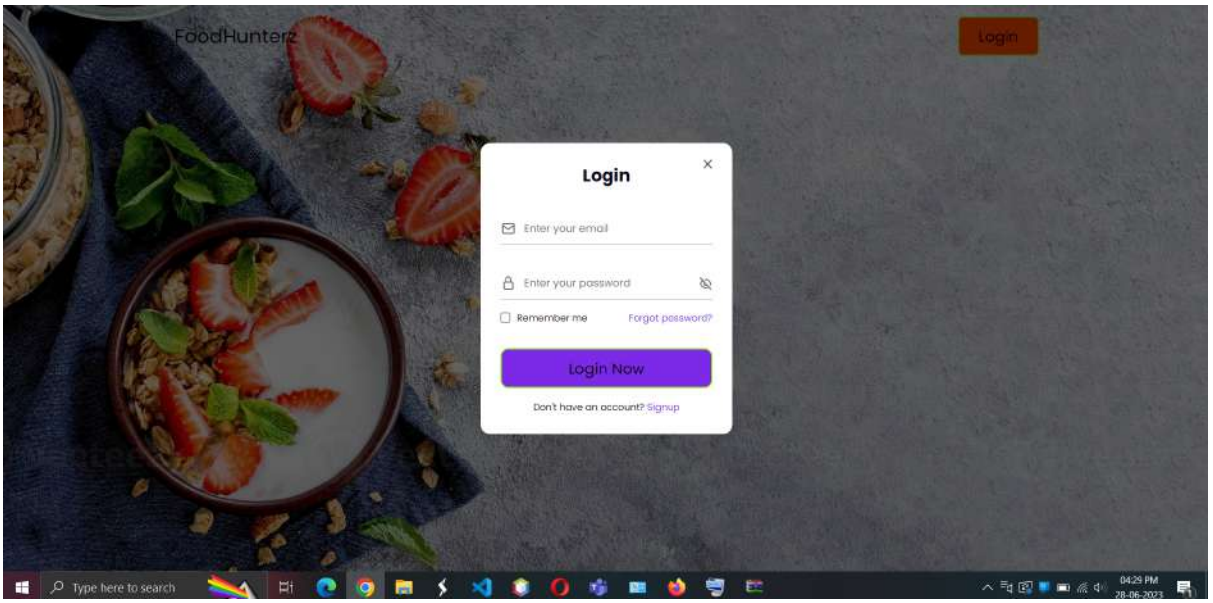
		with a credit card or debit card through the app or website or in cash at the restaurant when going to pick up. The website and app inform the customer of the food quality, duration of food preparation, and when the food is ready for pick-up or the amount of time it will take for delivery		
Online Food Ordering System	Abhishek Singh, Salma Pathan	It gives information needed in making order to customer. The Food website application made for restaurant and mess can help restaurant and mess in receiving orders and modifying its data and it is also made for admin so that it helps admin in controlling all the Food system.	Android SDK,GPS,WLAN devices	https://www.researchgate.net/profile/Roshan-Adithya/publication/321844341_Online_Food_Ordering_System/links/611df9431ca20f6f8630b883/Online-Food-Ordering-System.pdf
Automated Food Ordering System with Real-Time Customer Feedback	Shweta Shashikant Tanpure, Madhura M. Joshi	implementation of automated food ordering system with real time customer feedback (AOS-RTF) for restaurants. This system, implements wireless data access to servers. The android application on user's mobile will have all the menu details. The order details from customer's mobile are wirelessly updated in central database and subsequently sent to kitchen and cashier respectively. The restaurant	(AOS-RTF). It is a wireless food ordering system using android devices,	https://d1wqtxts1xzle7.cloudfront.net/40272220/food_order-libre.pdf?1448226915=&response-content-disposition=inline%3B+filename%3DFood_order.pdf&Expires=1688159242&Signature=gY~ipweU6BU-i

		owner can manage the menu modifications easily. The wireless application on mobile devices provide a means of convenience, improving efficiency and accuracy for restaurants by saving time, reducing human errors and real-time customer feedback.		XyVSAyLL8nd4256DrCMYLhwQtAW3bRFuP4awEDgJUgcBlqssg0iYv4MGkM7KtyiALx8m7APQRsWkB0GivhALhqkeJTBuFyVJFLGCo5PXxrjOK55ChlkRG0OBR24mR3jEcW~Cn8Livy8QHVs4657IvVxITFnF5DKPWSmnzlaoeeli8lSp4ZF3OXYf6k1TIO3NgrTxuy4XNEhL8ICwnyRcAnXYKK6EWLiwxVfj4KjAx1Nmej4f3OvLBKd4inBlu5G6JJUEER2EXfXC-iKtHqvgtuG20TfWJSYO2alxjZf--xjghujzF660~ZOeyfnHVxlRt5scp~dOw__&Key-Pair-Id=APKAJLOHF5GGSLRBV4ZA
Implementing Customizable Online Food	Varsha Chavan, Priya Jadhav, Snehal	“Food Pre-Order System using Web Based Application” in which customer can be able to create the order before they approach the restaurant.	Global System of Mobile communication,	https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=b11

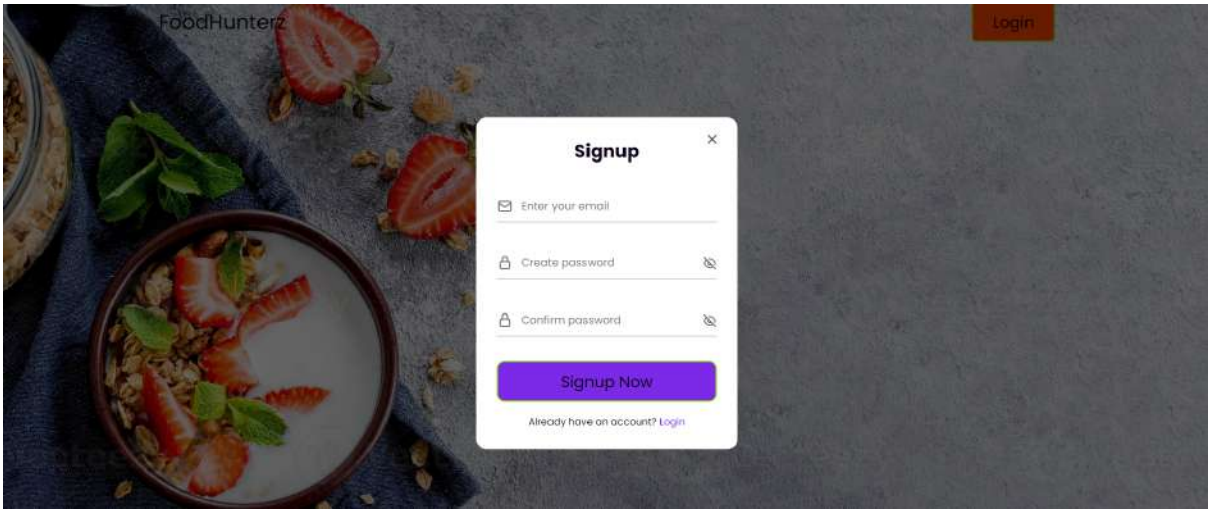
Ordering System Using Web Based Application	Korade	Customer using Smartphone. When the customer approach to the restaurant, the saved order can be confirmed by touching the Smartphone	User tablet,SMS integration,	5d85b90d36c5e7bac6fe5c6607635a705c19a
Online Food Ordering Application	Prof. Rohini Gurav, Sonali Jathar,Ashravanti Lavhare,Omkar Khadul	Online Food Ordering App is an application designed primarily for use in the food delivery industry. This system will allow hotels and restaurants to increase scope of business by reducing the labor cost involved. 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. Admin employees then use these orders through an easy to navigate graphical interface for efficient processing.	GPS,Andoroid,WLAN devices	https://ijarcce.com/wp-content/uploads/2021/05/IJARCCE.2021.10433.pdf

FRONTEND:

Customer Login Page:

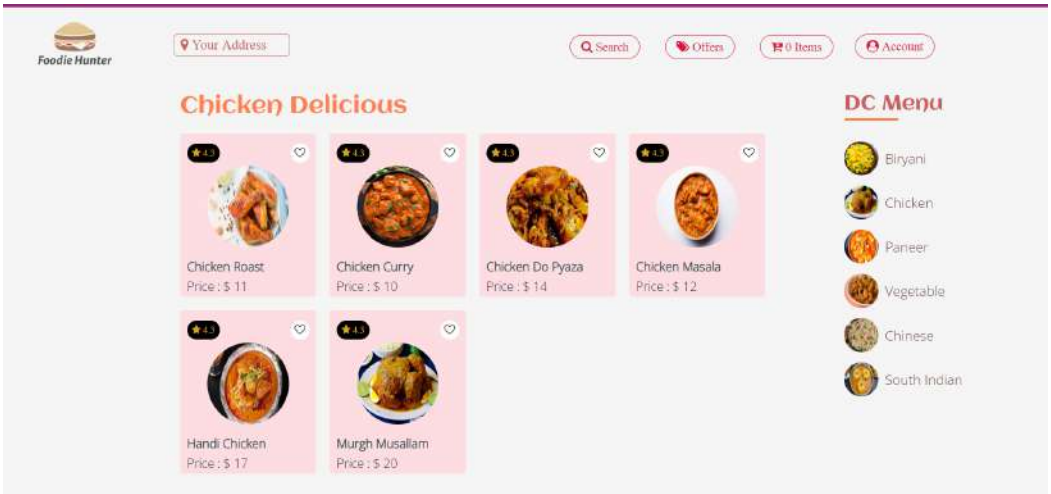


Signup:

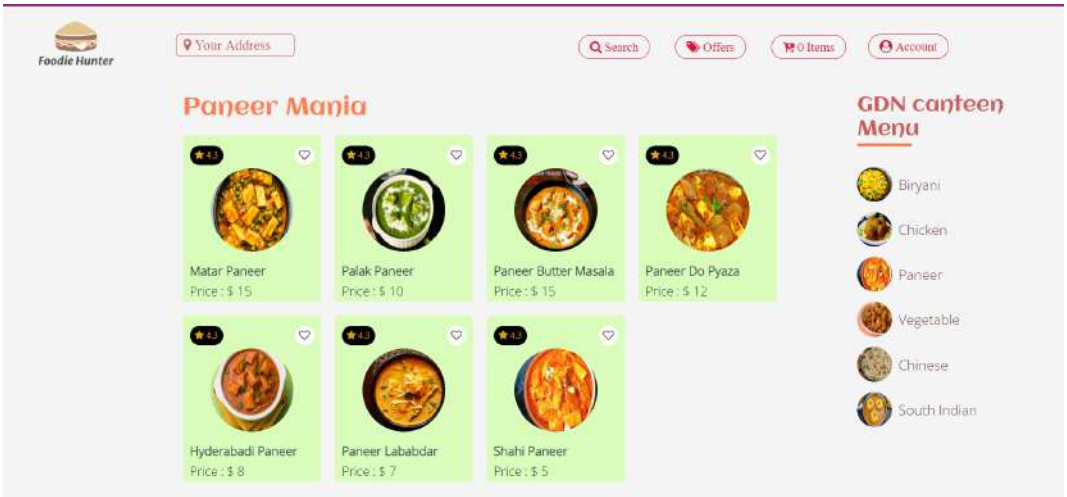


Select Restaurant:

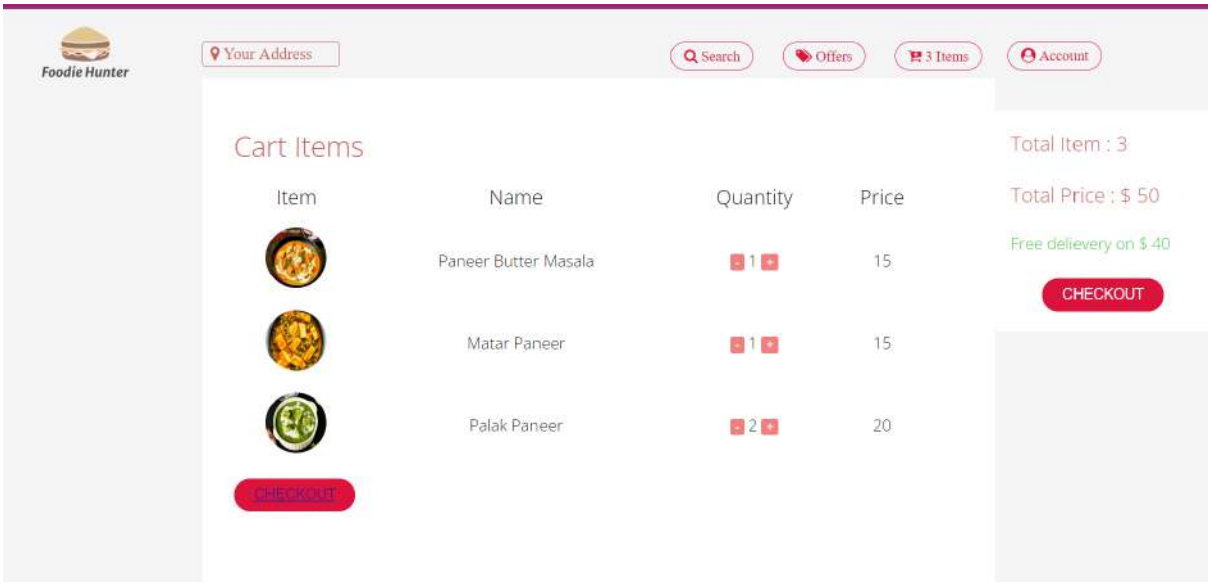
Restaurant Menu: (FC Restaurant menu):



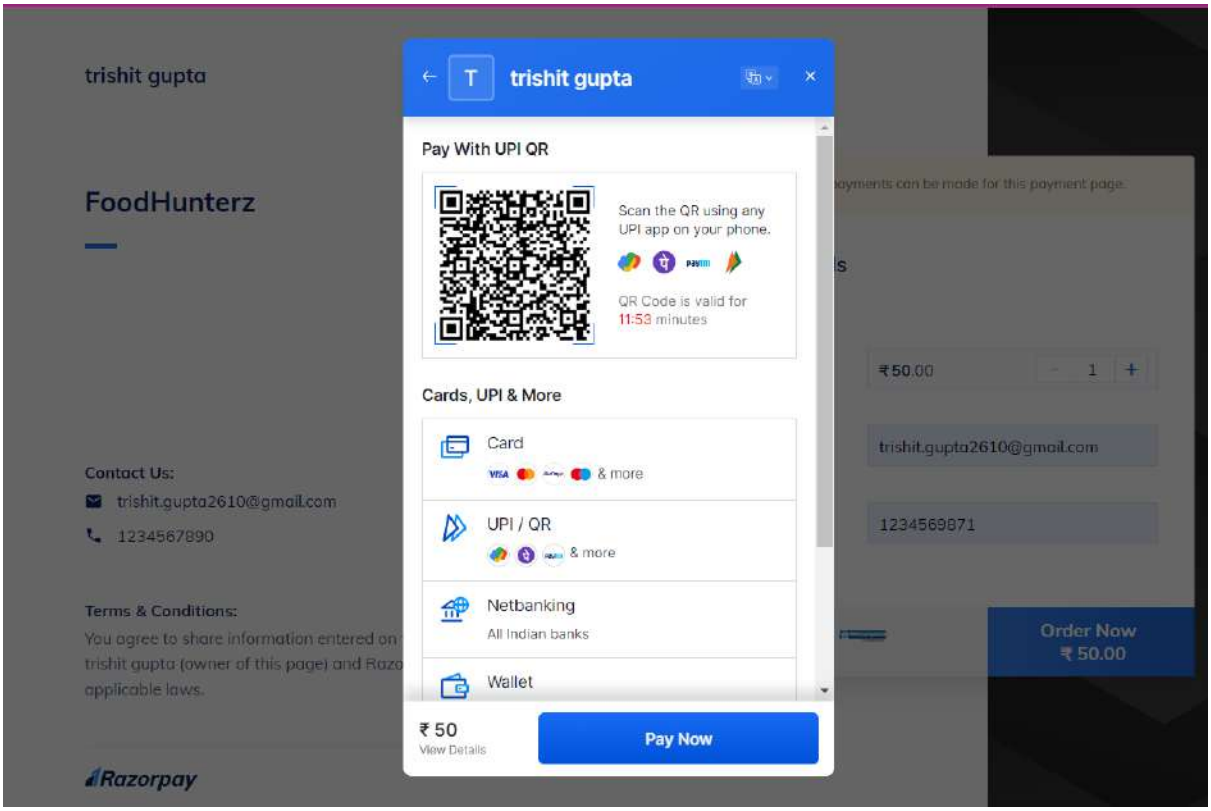
GDN Canteen Menu:



Cart:

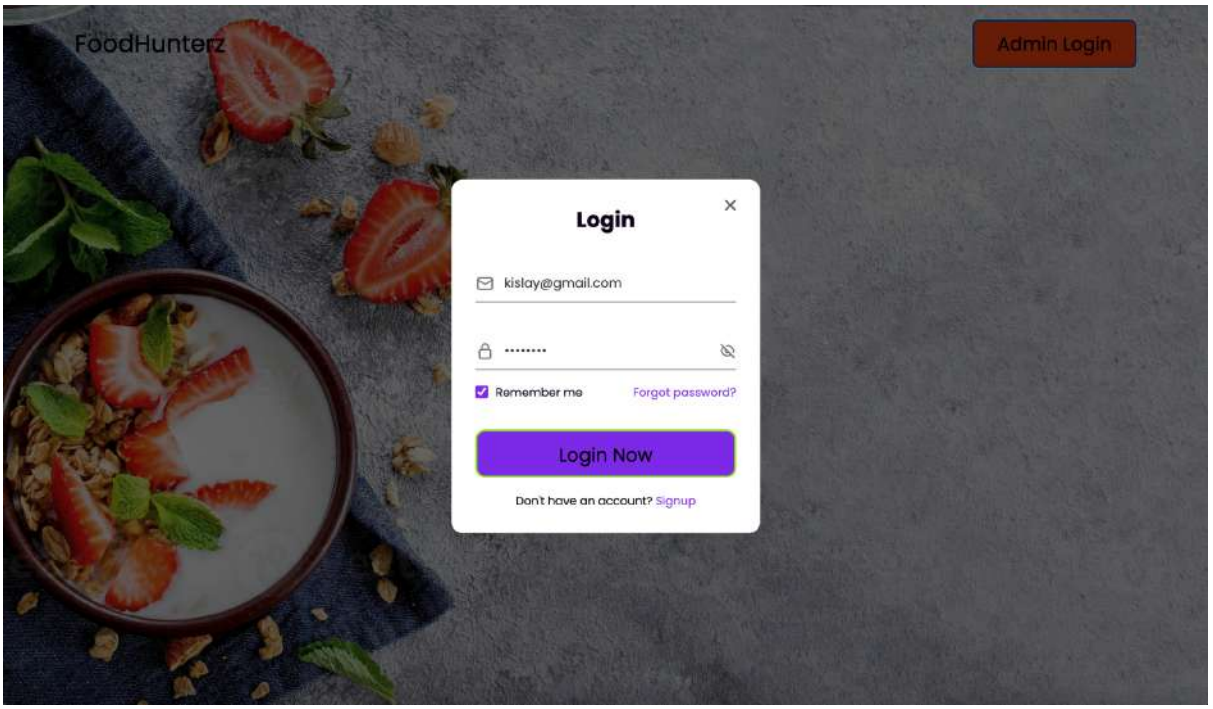


Checkout (Integrated with Razorpay Gateway):

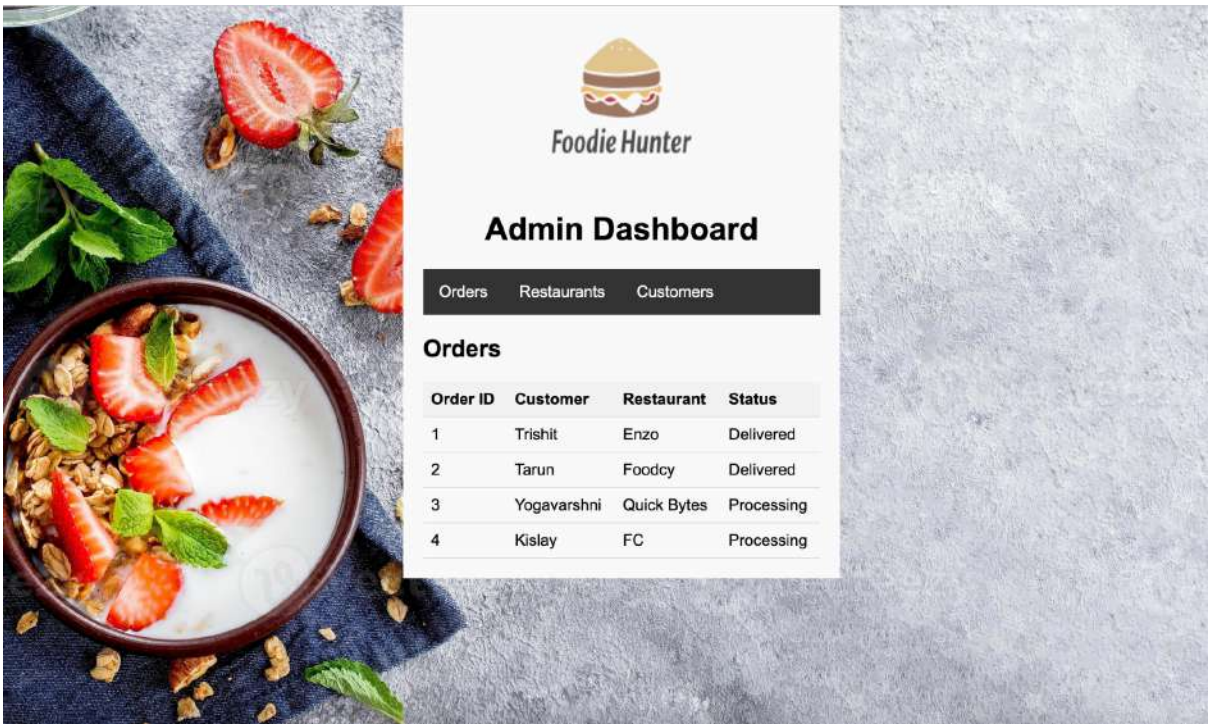


ADMIN’S PAGE:

ADMIN LOGIN PAGE:



ADMIN DASHBOARD:



ADMIN SELECT RESTAURANT:

ADD/DELETE/UPDATE MENU:

BACKEND:

PART 1: User Login, Admin Login, User Signup, Admin Signup, UserDetails Collection:

```
package com.example.demo;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class FoodApplication {

	public static void main(String[] args) {
		SpringApplication.run(FoodApplication.class, args);
	}

}
```

Controller:

```
package com.example.demo.LoginController;
import java.util.Objects;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.ModelAttribute;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.servlet.ModelAndView;
import com.example.demo.LoginDomain.Login;
import com.example.demo.LoginDomain.Admin;
import com.example.demo.LoginDomain.CustomerDetails;
import com.example.demo.LoginRepository.AdminRepository;
import com.example.demo.LoginRepository.CustomerDetailsRepository;
import com.example.demo.LoginRepository.LoginRepository;
import com.example.demo.LoginService.LoginService;
import jakarta.servlet.http.HttpServletRequest;
import jakarta.servlet.http.HttpServletResponse;
@Controller
public class LoginController {

	@Autowired
	private LoginRepository rep;

	@Autowired
	private AdminRepository repa;

	@Autowired
```

```
	private CustomerDetailsRepository cdres;

@Autowired
private LoginService userService;
@PostMapping("/restaurants")
public String userinsert(@ModelAttribute CustomerDetails cds) {
	cdres.save(cds);
	return "restaurants";
}
@GetMapping("CustomerDetails")
public String custdet() {
	return "CustomerDetails";
}
@GetMapping("/adminlogin")
public ModelAndView Admin() {
ModelAndView mav = new ModelAndView("adminlogin");
mav.addObject("user", new Admin());
return mav;
}
@GetMapping("/login")
public ModelAndView login() {
ModelAndView mav = new ModelAndView("login");
mav.addObject("user", new Login());
return mav;
}
@PostMapping("/index")
public String login(@ModelAttribute("user") Login user ) {
Login oauthUser = userService.login(user.getUsername(), user.getPassword());
System.out.print(oauthUser);
if(Objects.nonNull(oauthUser))
{
return "redirect:/";
} else {
return "redirect:/login";
}
}
@PostMapping("/adminpage")
public String admin(@ModelAttribute("user") Admin user ) {
Login oauthUser = userService.login(user.getUsername(), user.getPassword());
System.out.print(oauthUser);
if(Objects.nonNull(oauthUser))
{
return "redirect:/adminpage";
} else {
return "redirect:/adminlogin";
}
}
@GetMapping("adminpage")
public String adminhome() {
	return "adminpage";
}
@PostMapping("/adminlogin")
```

```
public String checkina(@ModelAttribute Login ins) {
	String pwd = ins.getPassword();
	String cpwd = ins.getCpassword();
	if(pwd.equals(cpwd)) {
		rep.save(ins);
		return "adminlogin";
	}
	return "signup";
}
@PostMapping("/login")
public String checkin(@ModelAttribute Admin ins) {
	String pwd = ins.getPassword();
	String cpwd = ins.getCpassword();
	if(pwd.equals(cpwd)) {
		repa.save(ins);
		return "login";
	}
	return "signup";
}
@GetMapping("/adminsignup")
public String signupi(){
return "adminsignup";
}
@GetMapping("/Signup")
public String signup(){
return "Signup";
}
@RequestMapping(value = {"/logout"}, method = RequestMethod.POST)
public String logoutDo(HttpServletRequest request,HttpServletResponse response)
{
return "redirect:/login";
}
}
```

Entity:

Admin Table:

```
package com.example.demo.LoginDomain;
import jakarta.persistence.Entity;
import jakarta.persistence.GeneratedValue;
import jakarta.persistence.GenerationType;
import jakarta.persistence.Id;
import jakarta.persistence.Table;
@Entity
@Table(name="admin")
public class Admin {
	@Id
	@GeneratedValue(strategy= GenerationType.IDENTITY)
private Long id;
private String username;
```

```
private String password;
private String cpassword;
	public Admin() {

	}

	public Admin(Long id, String username, String password, String cpassword) {

		this.id = id;
		this.username = username;
		this.password = password;
		this.cpassword = cpassword;
	}
	public Long getId() {
		return id;
	}
	public void setId(Long id) {
		this.id = id;
	}
	public String getUsername() {
		return username;
	}
	public void setUsername(String username) {
		this.username = username;
	}
	public String getPassword() {
		return password;
	}
	public void setPassword(String password) {
		this.password = password;
	}
	public String getCpassword() {
		return cpassword;
	}
	public void setCpassword(String cpassword) {
		this.cpassword = cpassword;
	}
}
```

Customer Details Table:

```
package com.example.demo.LoginDomain;
import jakarta.persistence.Entity;
import jakarta.persistence.GeneratedValue;
import jakarta.persistence.GenerationType;
import jakarta.persistence.Id;
import jakarta.persistence.Table;
@Entity
@Table(name="customerdetails")
public class CustomerDetails {

	@Id
	@GeneratedValue(strategy= GenerationType.IDENTITY)
```

```
	private Long id;
	private String name;
	private String number;
	private String email;
	private String hostelcategory;
	private String hostelblock;
	public CustomerDetails() {

	}
	public CustomerDetails(Long id, String name, String number, String email, String
hostelcategory,
				String hostelblock) {
		this.id = id;
		this.name = name;
		this.number = number;
		this.email = email;
		this.hostelcategory = hostelcategory;
		this.hostelblock = hostelblock;
	}
	public Long getId() {
		return id;
	}
	public void setId(Long id) {
		this.id = id;
	}
	public String getName() {
		return name;
	}
	public void setName(String name) {
		this.name = name;
	}
	public String getNumber() {
		return number;
	}
	public void setNumber(String number) {
		this.number = number;
	}
	public String getEmail() {
		return email;
	}
	public void setEmail(String email) {
		this.email = email;
	}
	public String getHostelcategory() {
		return hostelcategory;
	}
	public void setHostelcategory(String hostelcategory) {
		this.hostelcategory = hostelcategory;
	}
	public String getHostelblock() {
		return hostelblock;
	}
```

```
	public void setHostelblock(String hostelblock) {
		this.hostelblock = hostelblock;
	}
}
```

User Table:

```
package com.example.demo.LoginDomain;
import jakarta.persistence.Entity;
import jakarta.persistence.GeneratedValue;
import jakarta.persistence.GenerationType;
import jakarta.persistence.Id;
import jakarta.persistence.Table;
@Entity
@Table(name="login")
public class Login {
@Id
@GeneratedValue(strategy= GenerationType.IDENTITY)
private Long id;
private String username;
private String password;
private String cpassword;
public Login()
{
}
public Login(Long id, String username, String password, String cpassword) {
this.id = id;
this.username = username;
this.password = password;
this.cpassword = cpassword;
}
	public Long getId() {
return id;
}
public void setId(Long id) {
this.id = id;
}
public String getUsername() {
return username;
}
public void setUsername(String username) {
this.username = username;
}
public String getPassword() {
return password;
}
public void setPassword(String password) {
this.password = password;
}
public String getCpassword() {
	return cpassword;
}
```

```
public void setCpassword(String cpassword) {
	this.cpassword = cpassword;
}
}
```

Repository:

AdminRepository

```
package com.example.demo.LoginRepository;
import org.springframework.data.repository.CrudRepository;
import com.example.demo.LoginDomain.Admin;

public interface AdminRepository extends CrudRepository<Admin,Long>{
	Admin findByUsernameAndPassword(String username, String password);
}
```

Customer Repository:

```
package com.example.demo.LoginRepository;
import org.springframework.data.repository.CrudRepository;
import com.example.demo.LoginDomain.CustomerDetails;
public interface CustomerDetailsRepository extends CrudRepository<CustomerDetails,Long>{
}
```

User Repository:

```
package com.example.demo.LoginRepository;
import org.springframework.data.repository.CrudRepository;
import com.example.demo.LoginDomain.CustomerDetails;
public interface CustomerDetailsRepository extends CrudRepository<CustomerDetails,Long>{
}
```

Service:

AdminService:

```
package com.example.demo.LoginService;
import org.springframework.beans.factory.annotation.Autowired;
import com.example.demo.LoginDomain.Admin;
import com.example.demo.LoginRepository.AdminRepository;
public class AdminService {
	@Autowired
	private AdminRepository repa;

	public Admin login(String username, String password) {
		Admin user = repa.findByUsernameAndPassword(username, password);
		return user;
	}
}
```

Login Service(User Service):

```
package com.example.demo.LoginService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import com.example.demo.LoginDomain.Login;
import com.example.demo.LoginRepository.LoginRepository;
@Service
public class LoginService {
@Autowired
private LoginRepository repo;
public Login login(String username, String password) {
Login user = repo.findByUsernameAndPassword(username, password);
return user;
}
}
```

PART 2. Admin Rights of adding data,updating,deleting data of menu items in the restaurant.

1. Restaurant Tables:

I. @Entity

```
@Data
@Table(name = "tblmenu")
public class TblMenu {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "item_id")
    private int itemId;

    @Column(name = "item_name")
    private String itemName;

    @Column(name = "price")
    private float price;

    @Column(name = "food_item_type_id")
    private int foodItemTypeId;

    @Column(name = "item_image")
    private byte[] itemImage;

    @Column(name = "ingredients")
    private String ingredients;

  }
```

ii.@Entity
@Data
@Table(name = "foodmenutype")
public class FoodMenuType {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "food_item_type_id")
    private int foodItemTypeId;

    @Column(name = "type_name")
    private String typeName;

    @Column(name = "description")
    private String description;

    // Getters and setters
}

2. Repository
i.public interface TblMenuRepository extends JpaRepository<TblMenu, Integer> {
}

ii.public interface FoodMenuTypeRepository extends JpaRepository<FoodMenuType,
Integer> {
}

3.Services
1. TblMenuService.java
@Service
public class TblMenuService {
    private TblMenuRepository tblMenuRepository;

    @Autowired
    public TblMenuService(TblMenuRepository tblMenuRepository) {
        this.tblMenuRepository = tblMenuRepository;
    }

    public TblMenu addItem(TblMenu tblMenu) {
        return tblMenuRepository.save(tblMenu);
    }

    public void deleteItem(int itemId) {
        tblMenuRepository.deleteById(itemId);
    }

    public TblMenu updateItem(TblMenu tblMenu) {
        return tblMenuRepository.save(tblMenu);
    }

```
    public List<TblMenu> getAllItems() {
        return tblMenuRepository.findAll();
    }
}
```

2.FoodMenuTypeService.java

```
@Service
public class FoodMenuTypeService {
    private FoodMenuTypeRepository foodMenuTypeRepository;

    @Autowired
    public FoodMenuTypeService(FoodMenuTypeRepository foodMenuTypeRepository) {
        this.foodMenuTypeRepository = foodMenuTypeRepository;
    }

    public FoodMenuType addType(FoodMenuType foodMenuType) {
        return foodMenuTypeRepository.save(foodMenuType);
    }

    public void deleteType(int typeId) {
        foodMenuTypeRepository.deleteById(typeId);
    }

    public FoodMenuType updateType(FoodMenuType foodMenuType) {
        return foodMenuTypeRepository.save(foodMenuType);
    }

    public List<FoodMenuType> getAllTypes() {
        return foodMenuTypeRepository.findAll();
    }
}
```

4.Controllers

1.menuController

```
@RestController
@RequestMapping("/menu")
public class MenuController {
    private TblMenuService tblMenuService;
    private FoodMenuTypeService foodMenuTypeService;

    @Autowired
    public MenuController(TblMenuService tblMenuService, FoodMenuTypeService
foodMenuTypeService) {
        this.tblMenuService = tblMenuService;
        this.foodMenuTypeService = foodMenuTypeService;
    }
```

```
    @PostMapping("/add-item")
    public TblMenu addItem(@RequestBody TblMenu tblMenu) {
        return tblMenuService.addItem(tblMenu);
    }

    @DeleteMapping("/delete-item/{itemId}")
    public void deleteItem(@PathVariable int itemId) {
        tblMenuService.deleteItem(itemId);
    }

    @PutMapping("/update-item")
    public TblMenu updateItem(@RequestBody TblMenu tblMenu) {
        return tblMenuService.updateItem(tblMenu);
    }

    @GetMapping("/get-all-items")
    public List<TblMenu> getAllItems() {
        return tblMenuService.getAllItems();
    }

    @PostMapping("/add-type")
    public FoodMenuType addType(@RequestBody FoodMenuType foodMenuType) {
        return foodMenuTypeService.addType(foodMenuType);
    }

    @DeleteMapping("/delete-type/{typeId}")
    public void deleteType(@PathVariable int typeId) {
        foodMenuTypeService.deleteType(typeId);
    }

    @PutMapping("/update-type")
    public FoodMenuType updateType(@RequestBody FoodMenuType foodMenuType) {
        return foodMenuTypeService.updateType(foodMenuType);
    }

    @GetMapping("/get-all-types")
    public List<FoodMenuType> getAllTypes() {
        return foodMenuTypeService.getAllTypes();
    }
}
```

CONCLUSION:

The Food Ordering System provides a user-friendly and efficient platform for university students to order food from local restaurants within the campus. Built using Spring Boot, HTML, CSS, JavaScript, and MySQL, it offers a range of features such as menu browsing, order placement, payment integration, and order tracking. The system enhances the dining experience of students by ensuring timely delivery of fresh meals to their hostels.