**Food Delivery Website Documentation**

Contents

[**Food Delivery Website Documentation** 1](#_Toc145414404)

[*1. Overview* 2](#_Toc145414405)

[*1.1 Purpose of Documentation* 2](#_Toc145414406)

[*1.2 Audience* 2](#_Toc145414407)

[*2. Getting Started* 2](#_Toc145414408)

[*2.1 Prerequisites* 2](#_Toc145414409)

[*2.2* Installation 3](#_Toc145414410)

[*2.3* Configuration 3](#_Toc145414411)

[*2.4* Dependencies 3](#_Toc145414412)

[3.Project Structure 4](#_Toc145414413)

[*4.User Guide* 5](#_Toc145414414)

[4.1 Homepage 5](#_Toc145414415)

[5.Controller Structure 6](#_Toc145414416)

[5.1 UserController 6](#_Toc145414417)

[5.2 Seller Controller 7](#_Toc145414418)

[5.3 Dependencies: 7](#_Toc145414419)

[5.4 Main Functionalities: 7](#_Toc145414420)

[6.1 User Management 7](#_Toc145414421)

[7.Seller Management 12](#_Toc145414422)

[7.1 Show Coupon Form 12](#_Toc145414423)

[7.2 Process Order 12](#_Toc145414424)

[7.3 Get Dashboard 12](#_Toc145414425)

[7.4 Approve Order 12](#_Toc145414426)

[8. Security Configuration 13](#_Toc145414427)

[8.1 BCrypt Password Encoder 13](#_Toc145414428)

[8.2 User Details Service 13](#_Toc145414429)

[8.3 DAO Authentication Provider 13](#_Toc145414430)

[8.4 Security Filter Chain 14](#_Toc145414431)

[8.5 Custom Authentication Success Handler 14](#_Toc145414432)

[8.6 Http Security Configuration 15](#_Toc145414433)

[9 Connecting to Local Database view MongoDB Compass 15](#_Toc145414434)

[9.1 Connecting it to Spring Boot 16](#_Toc145414435)

[9.2. Configure MongoDB Connection: 17](#_Toc145414436)

[10 Implementing Email Service 17](#_Toc145414437)

[10.1 Configurating Email Service for spring boot 17](#_Toc145414438)

## 1. Overview

Welcome to the documentation for the Food Delivery Website. This document provides a detailed guide for both users and administrators of the website. Whether you are a customer looking to order delicious meals or an administrator managing the platform, this documentation will help you navigate through all the features and functionalities.

### 1.1 Purpose of Documentation

The purpose of this documentation is to:

* **Guide Users:** Help users understand how to use the website to order food, explore menu items, and complete their purchases.
* **Guide Administrators:** Assist administrators in managing restaurant details, menu items, orders, and payments.
* **Provide Clarity:** Explain the functionality of each page and feature within the website, making it easier for users and administrators to navigate and utilize the platform effectively.

### 1.2 Audience

This documentation is intended for two primary audiences:

1. **Users:** Individuals who visit the Food Delivery Website to browse and order food. Users will find instructions on how to navigate the site, browse menus, add items to their cart, and complete orders.
2. **Administrators:** Individuals responsible for managing the website, including adding and updating restaurant details, menu items, and processing orders. Administrators will find instructions on how to use the admin features of the site.

Let's get started with the documentation. Please navigate to the relevant sections based on your role as a user or administrator.

## 2. Getting Started

### 2.1 Prerequisites

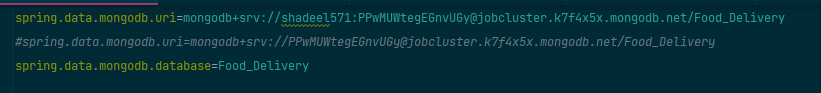
* **Java JDK 8 or later**
* **Spring Boot**
* **Integrated Development Environment (IDE)**
* **Database (e.g., MySQL or PostgreSQL)**
* **PayPal Developer Account**

### 2.2 Installation

1. **Clone the Repository:** Download or clone the project from the source.

Link:- https://github.com/moxhadeel571/Food\_ordering.git

1. **Configure the Database:** Set up the database connection in application.properties.



1. **Build and Run:** Execute ./mvnw spring-boot:run in the project directory.

### 2.3 Configuration

* Customize email service and file storage.
* Enhance security with authentication and authorization.

### 2.4 Dependencies

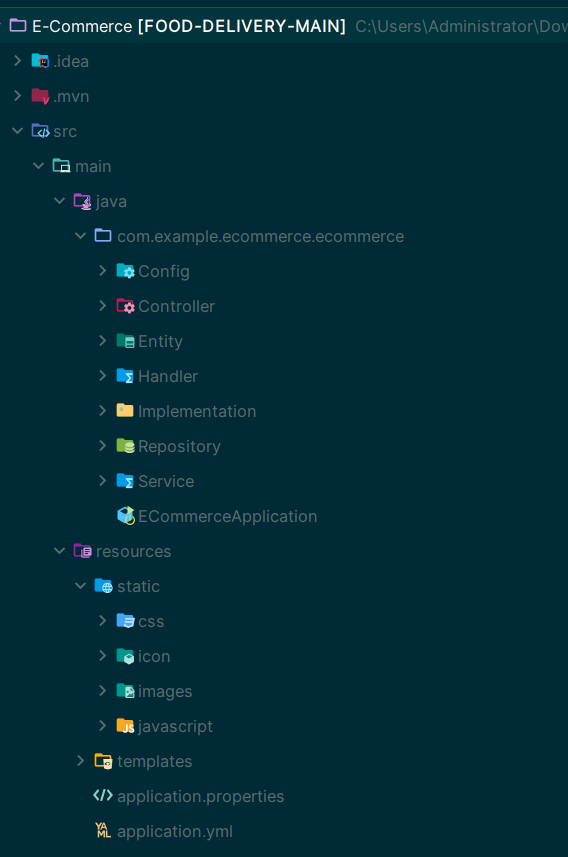
Dependencies are managed using Maven in the pom.xml file.

* Spring Boot Starter Data MongoDB:
* Gson (Google's JSON library): Version: 2.6.1
* Stripe Java SDK: Version: 22.30.0
* Spring Boot Starter Mail:
* Spring Boot Starter Thymeleaf:
* Spring Boot Starter Web:
* Spring Security Crypto:
* Spring Security Config:
* Spring Boot DevTools (runtime scope):
* Project Lombok (optional):
* Annotations by JetBrains:Version: 13.0
* JSON Path:
* Spring Test:Version: 6.0.10
* Hibernate Validator:Version: 5.2.4.Final
* Apache Derby Database:Version: 10.12.1.1
* Hibernate Core:Version: 6.2.2.Final
* Spring Boot Starter Security:

## 3.Project Structure

* **src/main/java**: Contains your Java code, including controllers, services, and repositories.
* **src/main/resources**: Holds configuration files, static resources (HTML, CSS, JS), and templates.
* **src/test**: Contains unit and integration tests.
* **pom.xml**: Manages project dependencies and build settings using Maven.
* **application.properties or application.yml**: Stores application configuration.
* **com.yourpackage**: Base package for Java classes.
* **com.yourpackage.controller**: Contains MVC controllers.
* **com.yourpackage.service**: Holds business logic and services.
* **com.yourpackage.repository**: Defines data repositories.
* **com.yourpackage.model**: Stores data models.
* **resources/templates**: Templates for template engines (Thymeleaf/Freemarker).
* **resources/static**: Static resources (CSS, JS, images).
* **test**: Contains test classes.
* **target**: Generated directory with compiled code.

This is the file structure for the project and maintain the same structure for the future projects too, if the files are not categorized as what they work is for then it will be in circular dependency error.



## 4.User Guide

### 4.1 Homepage

* **Description**: The Homepage serves as the main entry point to the Food Delivery Website. It provides an inviting and informative introduction to the website.
* **Usage**: Users should visit this page as the first step when accessing the Food Delivery Website. Here's what you can do on the Homepage:
  + **Explore Restaurants**: Browse through a list of available restaurants in your area. Each restaurant will typically include its name, cuisine type, location, and a brief description.
  + **Discover Menu Items**: Get a sneak peek of the delicious menu items offered by these restaurants. You might find images of mouthwatering dishes to whet your appetite.
  + **User-Friendly Interface**: The Homepage is designed to be user-friendly, making it easy to navigate and find the information you need.
  + **Quick Access**: It offers quick access to other sections of the website, such as menu listings, discounts, and your shopping cart.
  + **Search Functionality**: Look for specific restaurants or cuisine types using the search bar if you have something particular in mind.
  + **Enjoy the Visuals**: Visuals play a significant role on this page. High-quality images and enticing descriptions aim to engage users and encourage exploration.
  + **Stay Updated**: Some websites may also feature special promotions or announcements on the Homepage, so it's a good place to stay updated on the latest news and offers.
  + **Start Ordering**: Once you've explored the options and made your selection, you can start ordering by clicking on the desired restaurant or menu items.

## 5.Controller Structure

In the Food Ordering System project, the controllers play a crucial role in handling incoming requests, processing data, and coordinating the flow of the application. This section provides an explanation of the organization and dependencies of the main controller classes within the system.

### 5.1 UserController

The **UserController** is responsible for managing user-related operations, including menu browsing, shopping cart management, order processing, and user communication. It interacts with various services and repositories to fulfill these functionalities.

Dependencies:

* **MenuPhotoRepository**: Manages menu photo data storage.
* **RestaurentRepository**: Handles restaurant data retrieval and storage.
* **RestaurantService**: Provides restaurant-related operations.
* **MenuPhotoService**: Manages menu photo operations.
* **checkOutService**: Manages shopping cart and order checkout.
* **EmailService**: Handles email notifications for orders and communication.
* **Email**: Entity class representing email details.
* **Restaurant**: Entity class representing restaurant details.
* **checkOut**: Entity class representing shopping cart and order details.

Main Functionalities:

* Displaying menu categories and listings.
* Managing shopping cart contents and order processing.
* Handling discounts and pricing calculations.
* Uploading and displaying restaurant images.
* Managing order checkout and payment.
* Sending email notifications.
* Displaying order history and details.

### 5.2 Seller Controller

The **SellerController** is responsible for seller-specific functionalities, such as coupon management, order returns, product management, and shipment handling. It interacts with various services and repositories dedicated to seller-related operations.

### 5.3 Dependencies:

* **CouponService**: Manages coupon creation and discount generation.
* **OrderReturnService**: Handles customer order returns and approvals.
* **OrderReturnRepository**: Manages order return data storage.
* **checkOutService**: Manages shopping cart and order checkout.
* **ProductsService**: Provides product-related operations.
* **ShipmentService**: Manages shipment creation and status updates.
* **ProductsRepository**: Handles product data retrieval and storage.
* **Products**: Entity class representing product details.
* **Coupon**: Entity class representing coupon details.
* **Shipment**: Entity class representing shipment details.
* **OrderReturn**: Entity class representing order return details.

### 5.4 Main Functionalities:

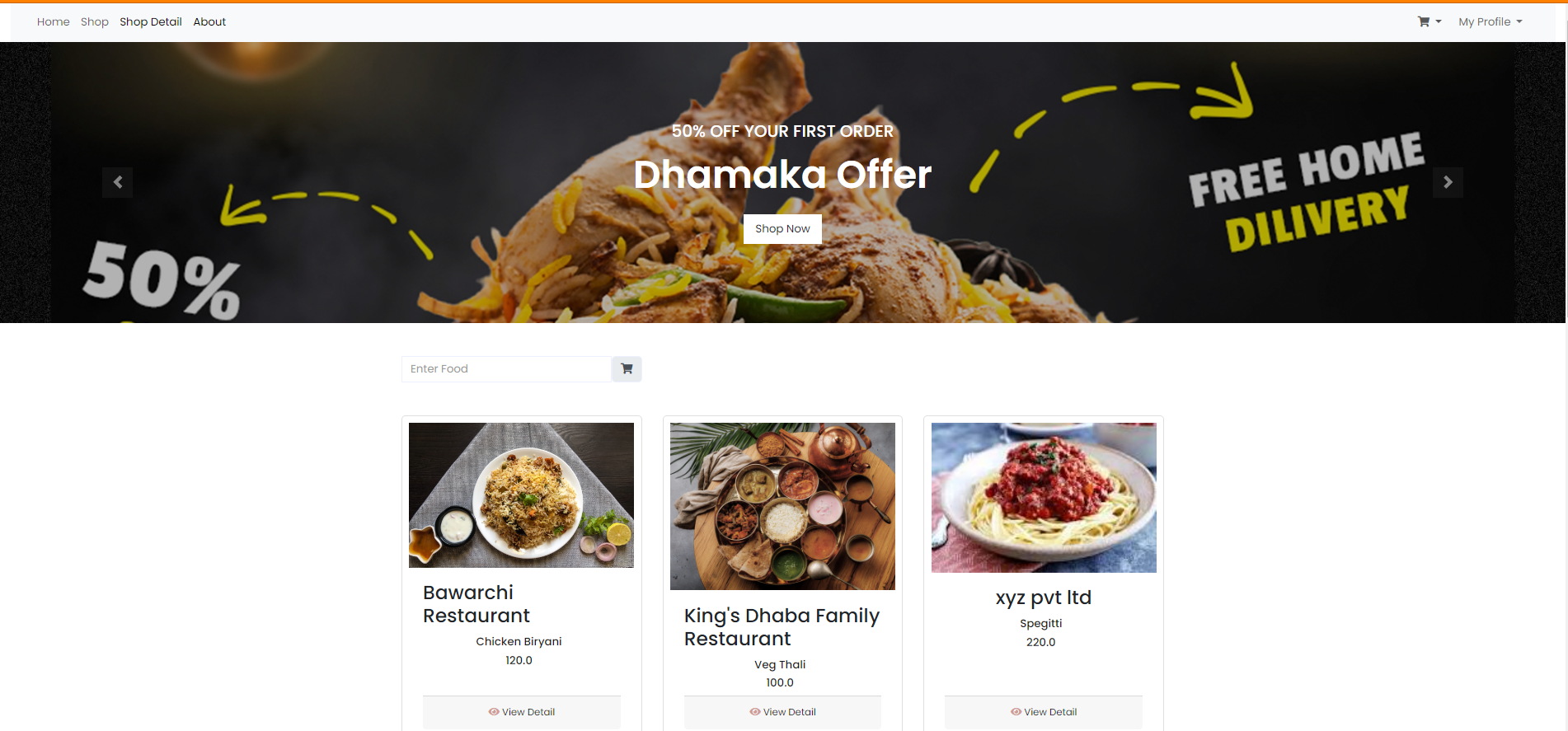
* Creating and managing discount coupons.
* Processing customer order returns.
* Managing product details and inventory.
* Creating and updating shipment records.
* Approving and rejecting order returns.
* Handling product image storage and retrieval.
* Updating product quantities.
* Managing the seller's dashboard with order and product information.

## 6.1 User Management

User management is a critical aspect of the Food Ordering System, encompassing a range of functionalities that enhance the user experience and facilitate the ordering process. This section delves into the details of the main user-related operations and their associated controller methods.

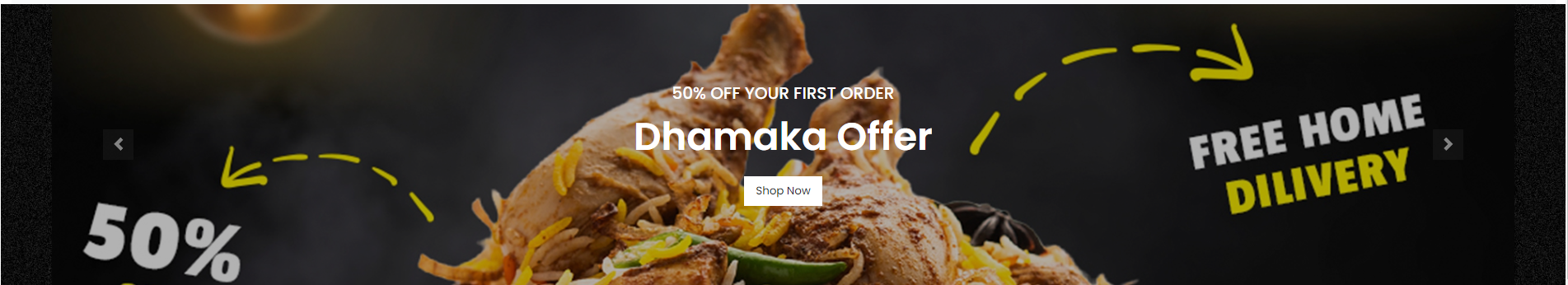
**Get Menu Listing**

* **Description**: Users can browse and view the entire menu listing. This functionality provides a comprehensive list of available food items.



**Get Discount Listing**

* **Description**: Displays discounted items along with their pricing information. Users can see special offers and promotions. (when viewing food details and adding it to cart it will automatically gives discount of 50% off)

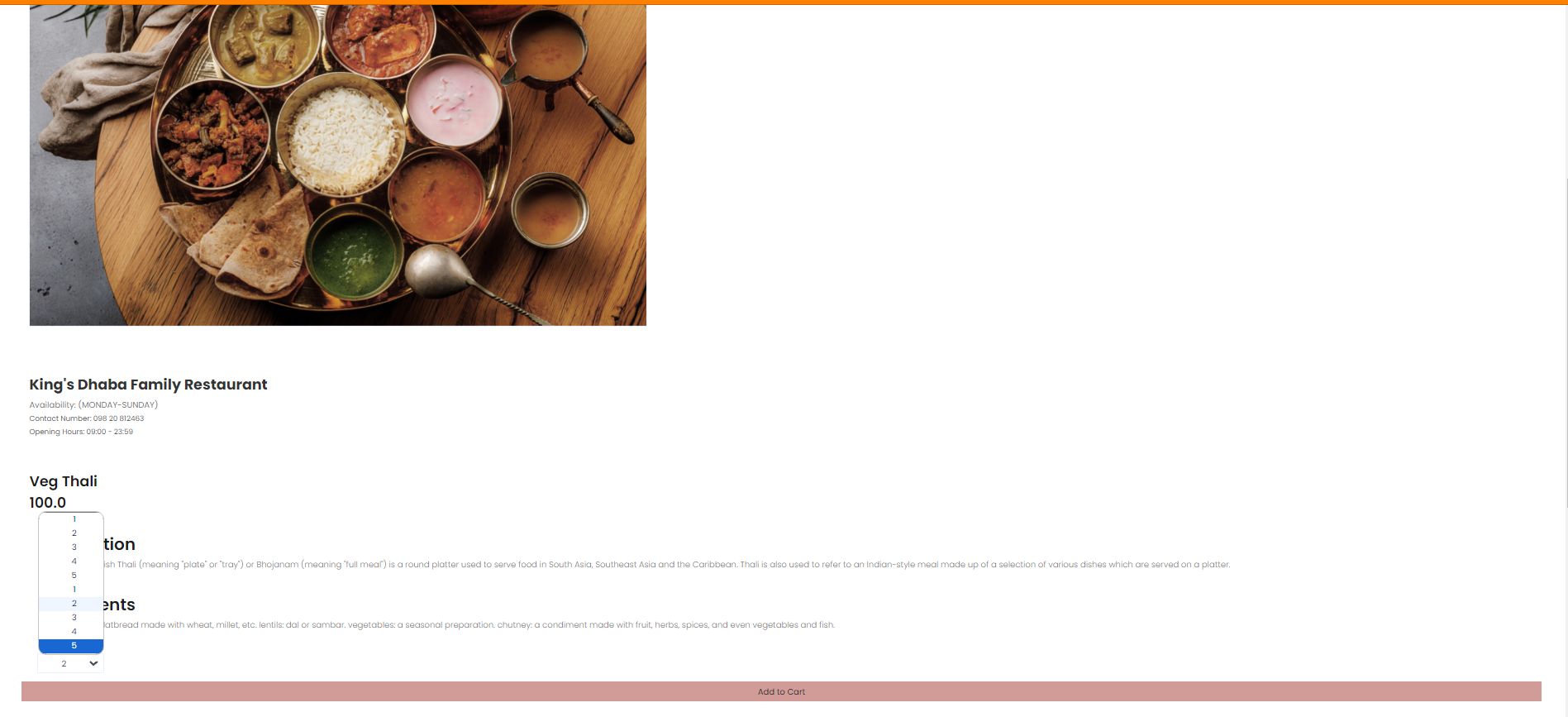


**Save Form**

* **Description**: Handles the submission and storage of user form data, typically used for order placement. This functionality captures user preferences and requirements for order processing.

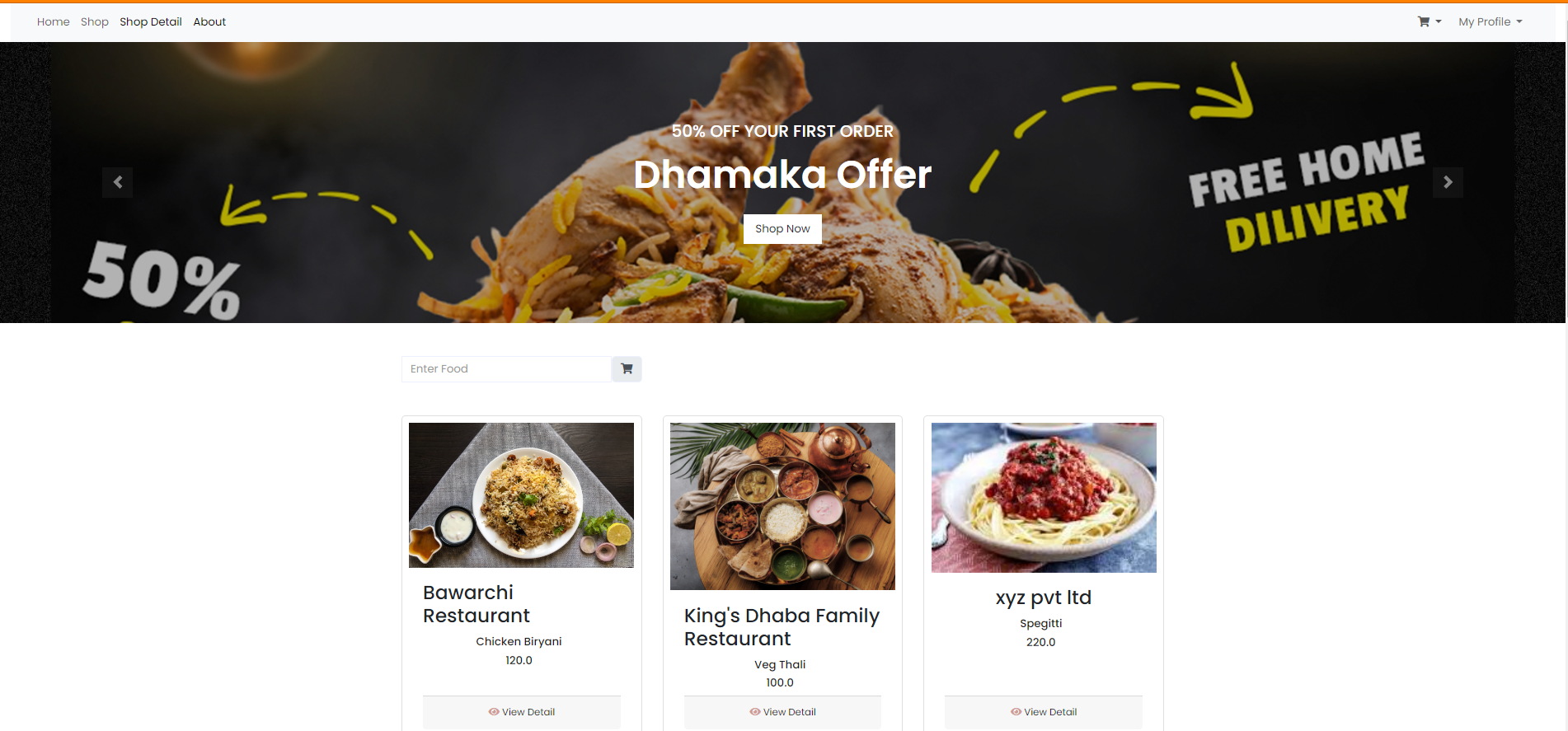
**Update Quantity**

* **Description**: Manages changes in product quantities within the user's shopping cart. Users can adjust the quantity of items they wish to order.



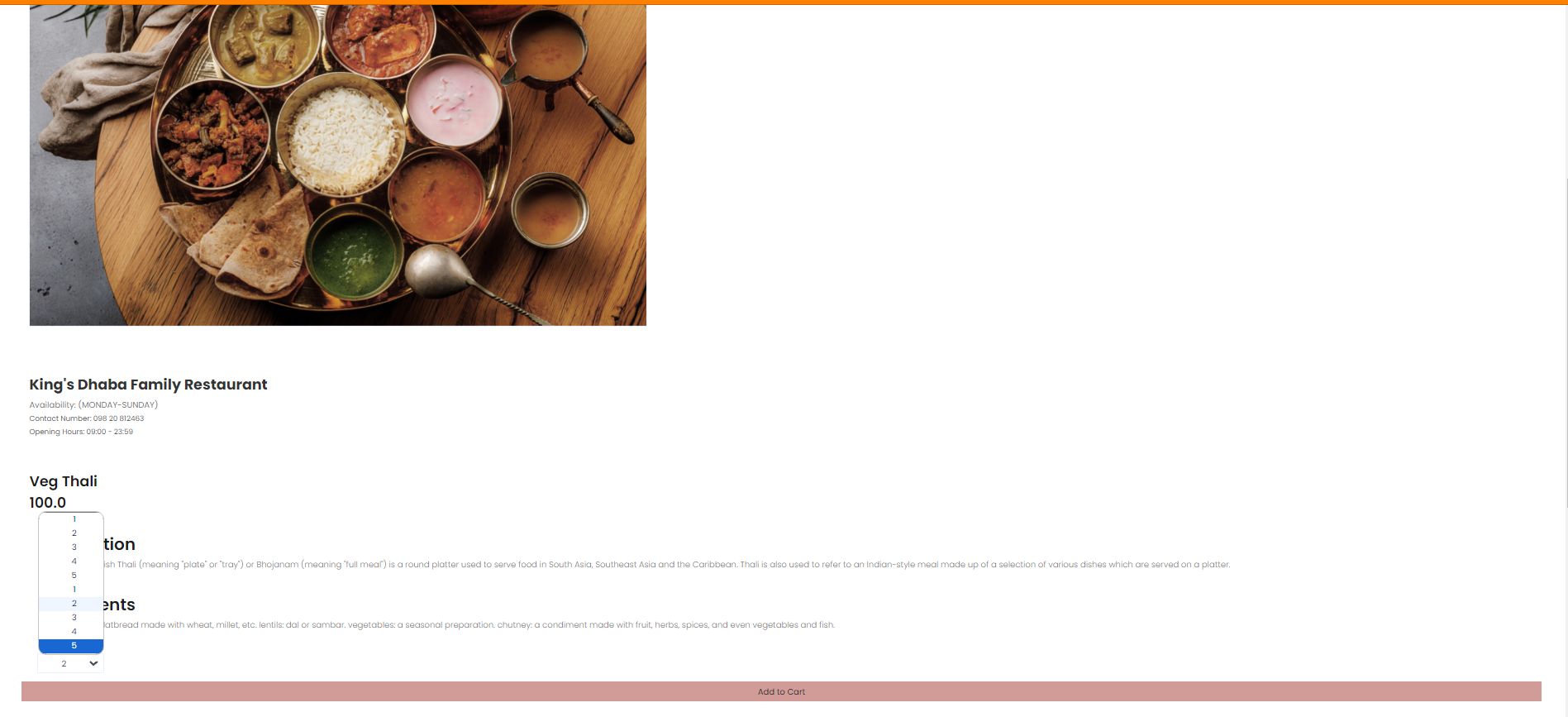
**Display Image**

* **Description**: Retrieves and displays restaurant images associated with menu items. Users can visually explore restaurant options.



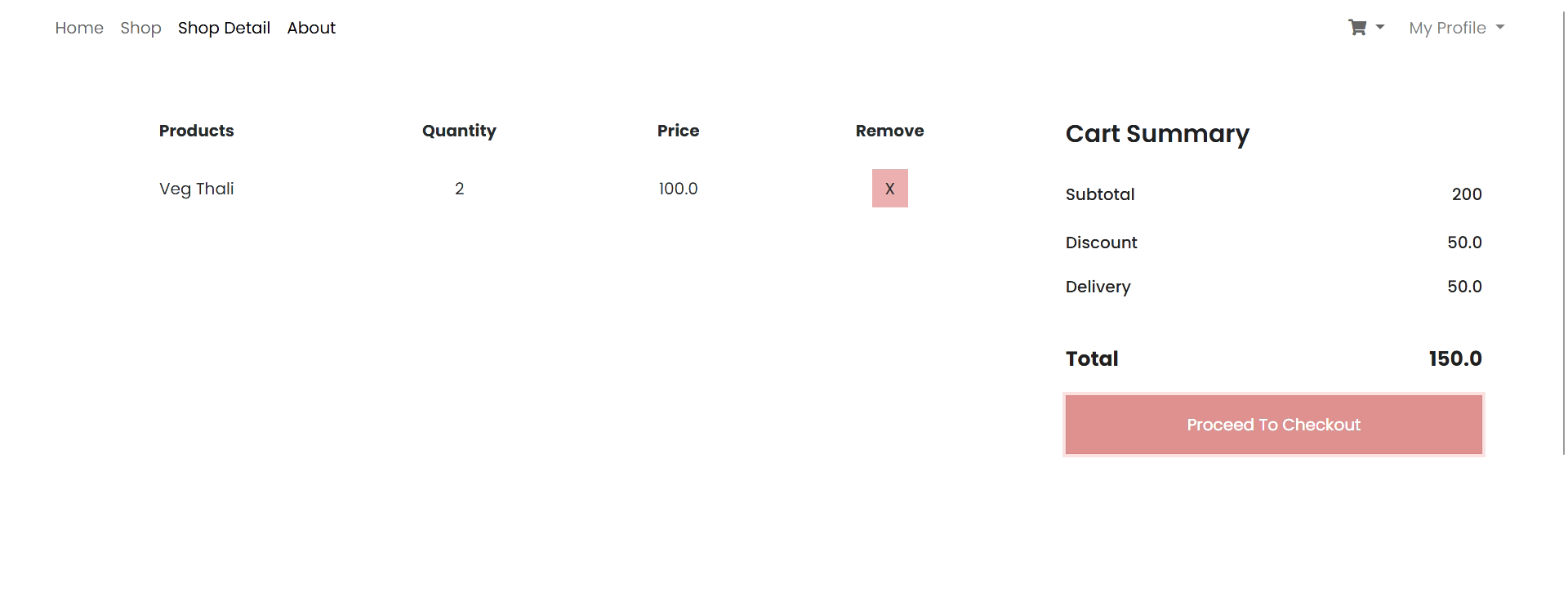
**Get Restaurant Listing**

* **Description**: Presents restaurant listings to users, showcasing the available dining options. Users can select their preferred restaurants.



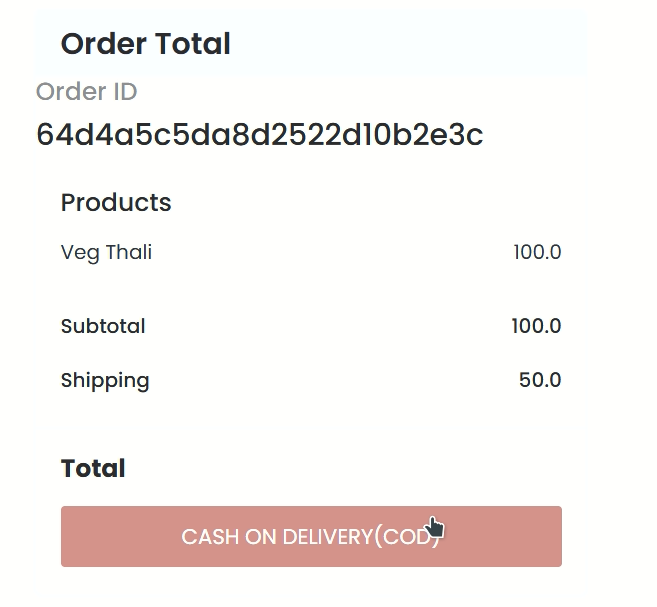
**Get Cart**

* **Description**: Displays the contents of the user's shopping cart. This feature provides users with an overview of their selected items, quantities, and prices.



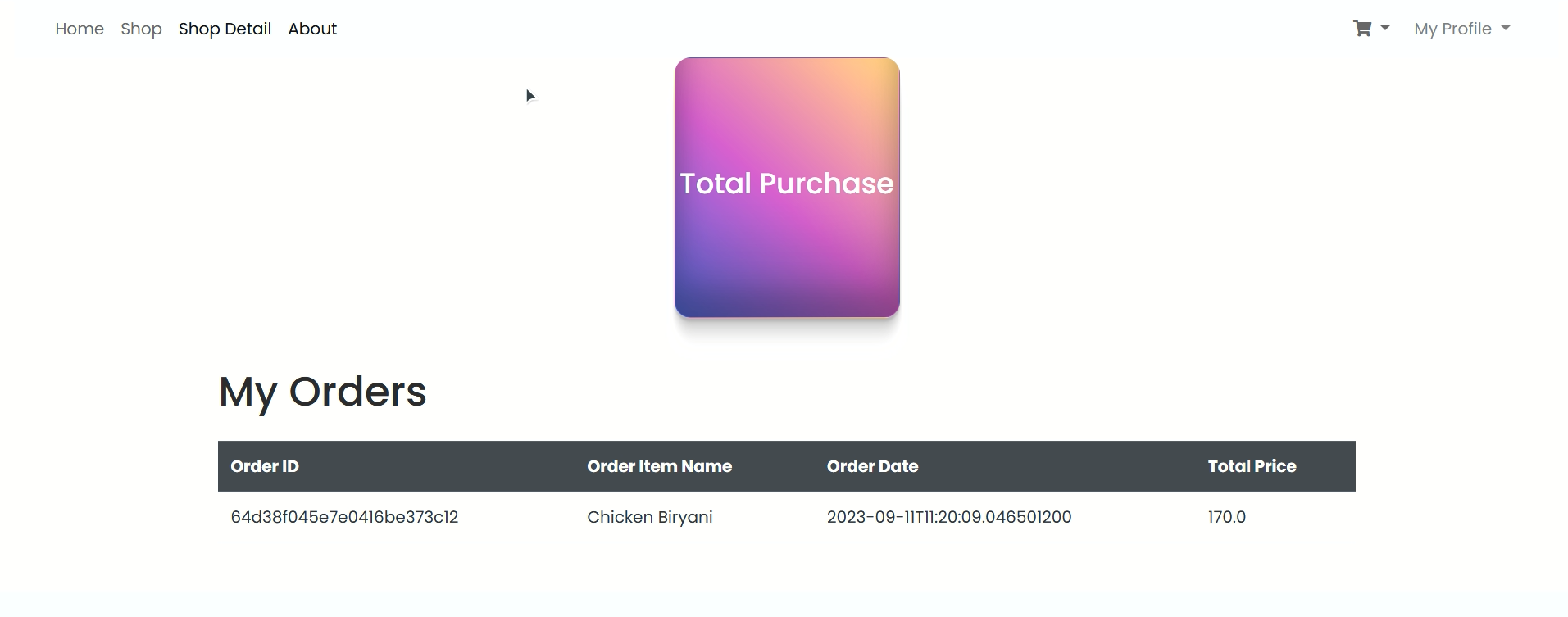
**Cash on Delivery**

* **Description**: Enables users to process orders with cash-on-delivery payment. This payment method allows users to pay for their orders upon delivery.



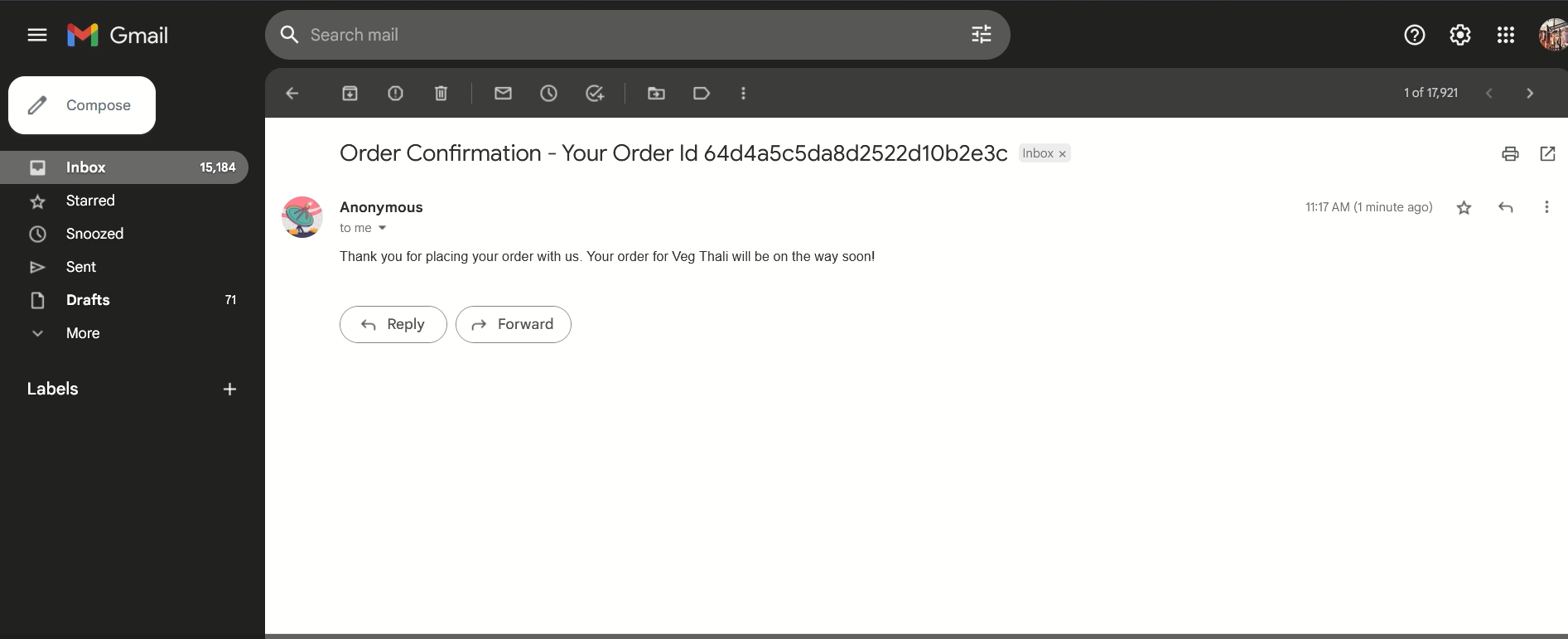
**Save Checkout**

* **Description**: Stores order checkout details, ensuring a smooth and convenient ordering process for users. It captures essential order information.

****

**Send Email**

* **Description**: Sends order-related emails to users, providing confirmation, updates, and important order information via email.

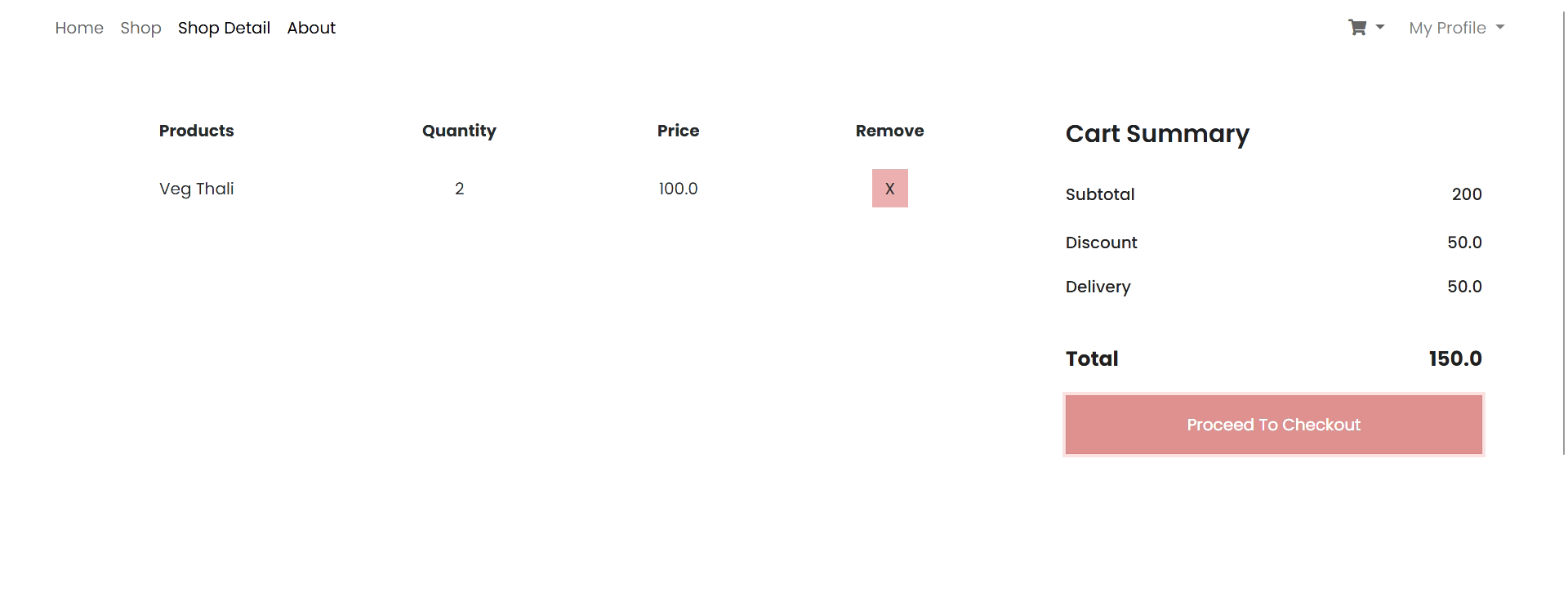


## 7.Seller Management

Seller management involves a wide range of functionalities to assist sellers in efficiently managing their products, processing orders, and handling various administrative tasks. This section provides an in-depth explanation of key seller-related operations and their associated controller methods.

### 7.1 Show Coupon Form

* **Description**: This functionality allows sellers to get Automated discount coupon which is automatically redeem when adding the food into cart.

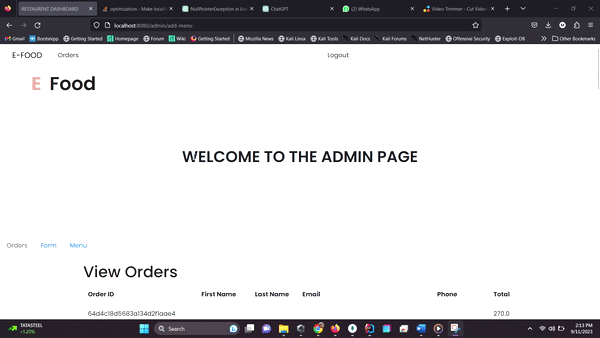


### 7.2 Process Order

* **Description**: Handles order processing task, when food is added into cart it will redirect to checkout form after filling the form it will place the order with a confirmation email.

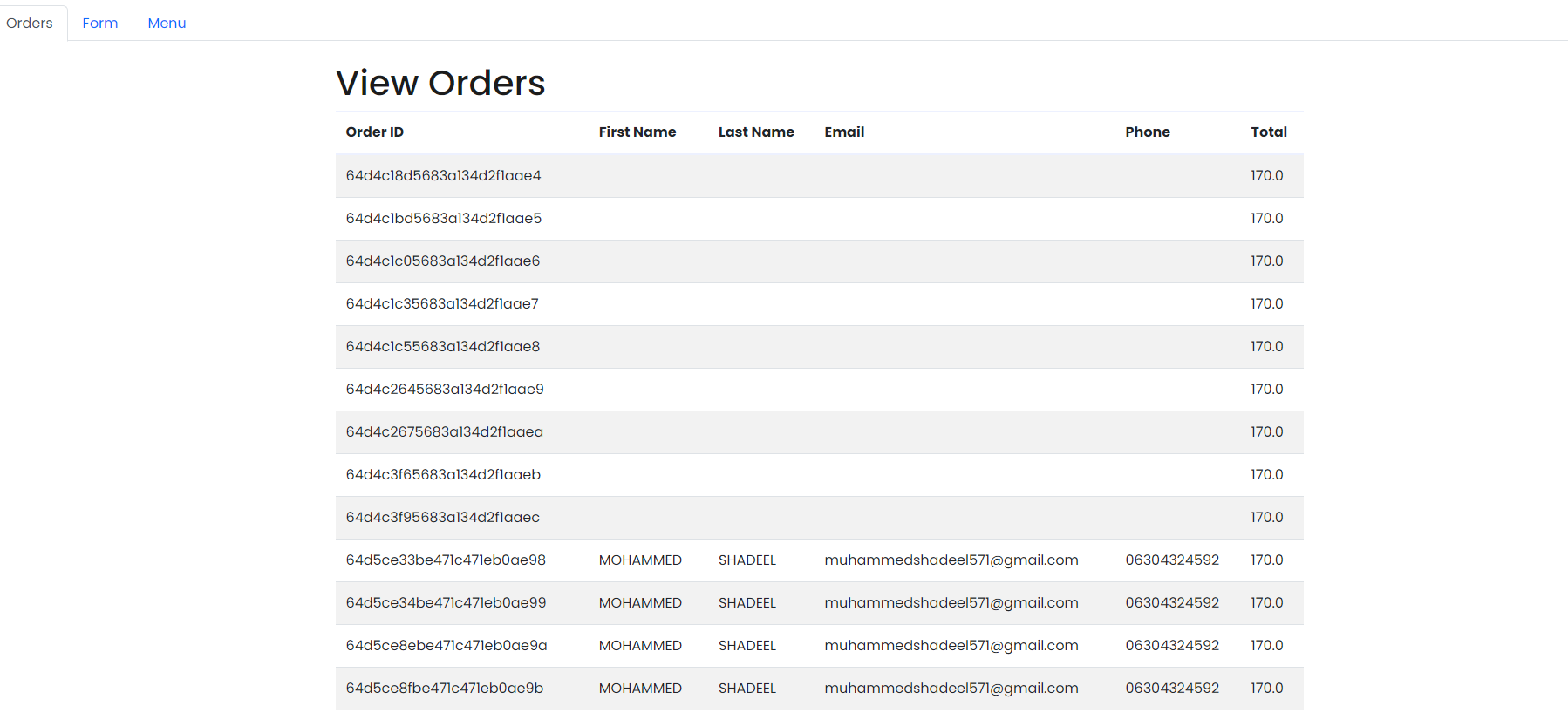
### 7.3 Get Dashboard

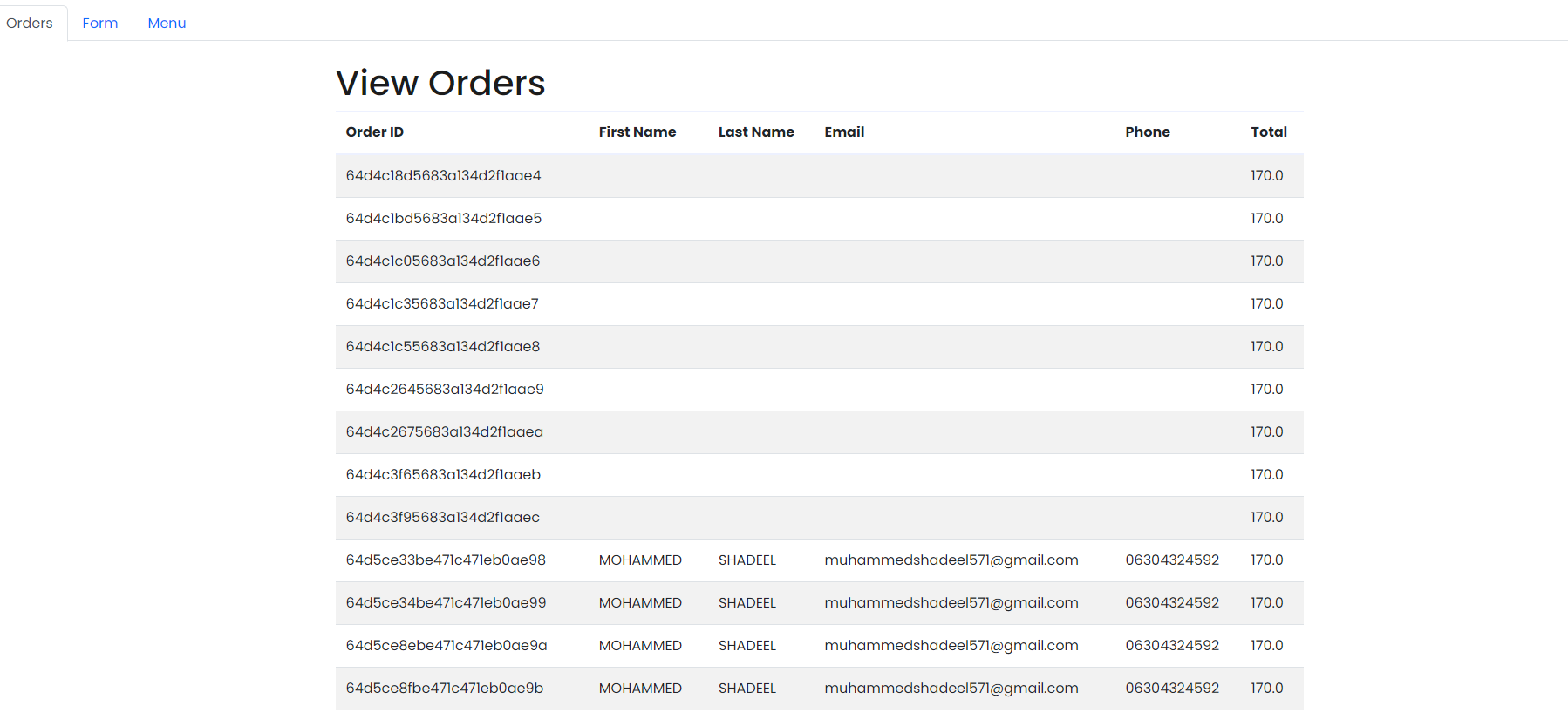
* Description: Displays the seller's dashboard, providing insights into order and product information. Sellers can track their business operations and performance.



### 7.4 Approve Order

* **Description**: Allows sellers to approve customer order, streamlining the process and ensuring customer satisfaction.





## 8. Security Configuration

The security configuration of the Food Ordering System is responsible for managing user authentication and authorization. It defines the access rules, authentication providers, and login processing settings. Below, you'll find an overview of the key elements and functionalities within the security configuration.

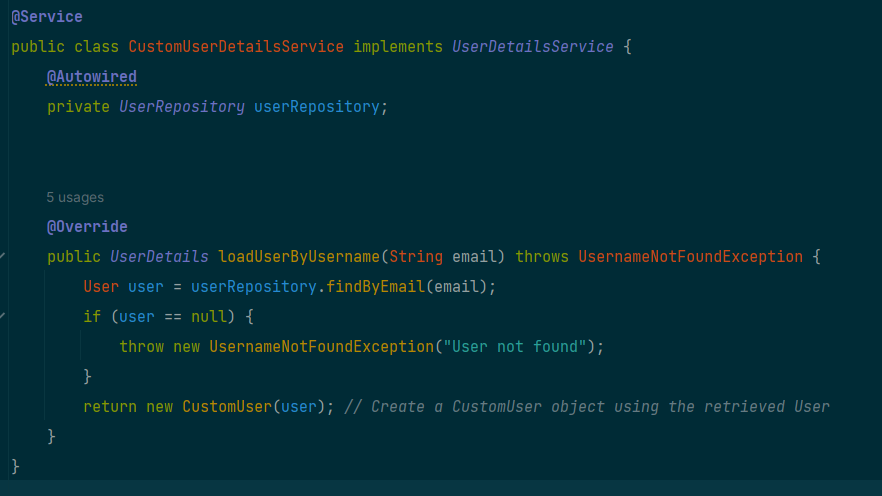
### 8.1 BCrypt Password Encoder

* **Description**: The **BCryptPasswordEncoder** bean is responsible for securely hashing and encoding user passwords.



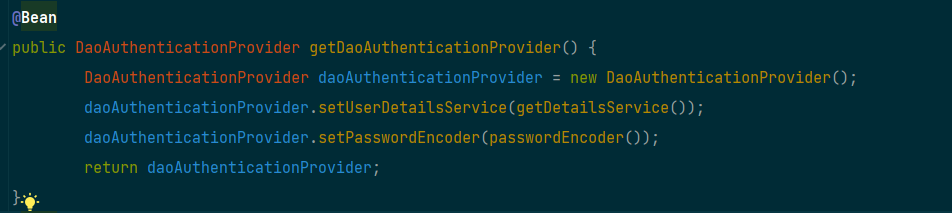
### 8.2 User Details Service

* **Description**: The **UserDetailsService** bean is used to retrieve user details from the database during authentication.



### 8.3 DAO Authentication Provider

* **Description**: The **DaoAuthenticationProvider** bean configures the authentication provider, specifying the user details service and password encoder.



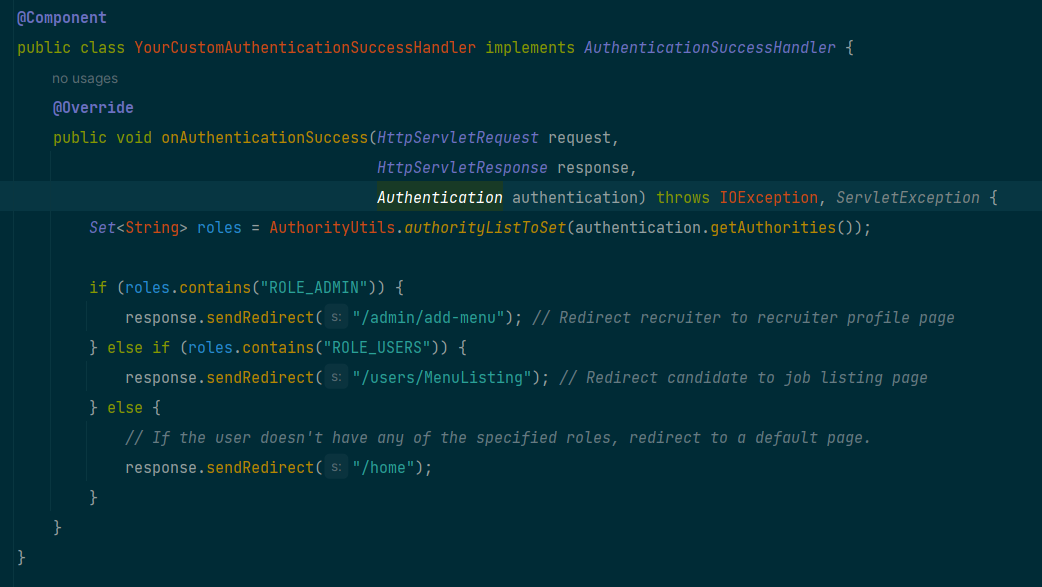
### 8.4 Security Filter Chain

* **Description**: The **SecurityFilterChain** bean defines the security filter chain, specifying various security configurations and access rules.

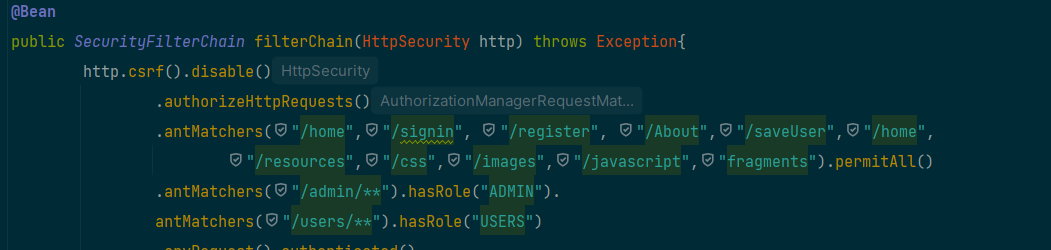


### 8.5 Custom Authentication Success Handler

* **Description**: A custom authentication success handler (**YourCustomAuthenticationSuccessHandler**) is configured to handle successful login attempts.



### 8.6 Http Security Configuration



* **Description**: The **HttpSecurity** configuration defines the security rules and access restrictions for different URL patterns. It specifies which URLs are accessible without authentication (**permitAll**), which require specific roles (**hasRole**), and configures the login process.
  + **/home**, **/signin**, **/register**, **/About**, **/saveUser**, **/home**, **/resources**, **/css**, **/images**, **/javascript**, **fragments**: These URLs are permitted for access without authentication.
  + **/admin/\*\***: Requires users with the "ADMIN" role to access URLs under this pattern.
  + **/users/\*\***: Requires users with the "USERS" role to access URLs under this pattern.
  + Any other URL requires authentication.
  + **Form Login Configuration**: Configures form-based login with a custom login page (**/signin**), login processing URL (**/userLogin**), and a custom authentication success handler.

The security configuration ensures that users have appropriate access rights based on their roles and authentication status. It also incorporates secure password hashing and customizable login handling.

## 9 Connecting to Local Database view MongoDB Compass

To connect to your local MongoDB database using MongoDB Compass, you can follow these steps:

1. **Install MongoDB Compass**:
   * If you haven't already installed MongoDB Compass, you can download it from the official MongoDB website: [MongoDB Compass Download](https://www.mongodb.com/try/download/compass)
2. **Launch MongoDB Compass**:
   * After installing, launch MongoDB Compass on your computer.
3. **Connect to Your Local MongoDB**:
   * When you first open MongoDB Compass, you'll be presented with a "Connect to Host" screen.
   * In the "Hostname" field, enter the address of your local MongoDB server. By default, MongoDB runs on **localhost** and port **27017**, so you can leave these values as is.
   * You can also specify a connection name for reference.
4. **Advanced Options**:
   * If you have additional configurations or need to connect to a specific database, you can click on "More Options" to expand the settings.
   * Here, you can specify authentication credentials, replica set, SSL options, and other advanced settings. For a local development environment, you may not need these.
5. **Test Connection**:
   * Before connecting, you can click the "Test Connection" button to ensure that MongoDB Compass can connect to your local MongoDB server without issues.
6. **Connect**:
   * Once you are satisfied with the settings, click the "Connect" button to establish a connection.
7. **Explore Your Local Database**:
   * After successfully connecting, you will see a list of available databases on your local MongoDB server.
   * Click on a database to explore its collections and documents.
8. **Perform Database Operations**:
   * You can perform various database operations using MongoDB Compass, such as querying data, inserting documents, updating records, and more. Simply navigate to the relevant collection and use the provided GUI tools.

### 9.1 Connecting it to Spring Boot

To connect your Spring Boot application to a local MongoDB database using MongoDB Compass, you need to configure your Spring Boot application's **application.properties** or **application.yml** file with the MongoDB connection details. Here's how you can do it:

**1. Add MongoDB Dependency:**

First, make sure you have the MongoDB dependency added to your Spring Boot project. You can add it to your **pom.xml** file if you're using Maven.

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-data-mongodb</artifactId>

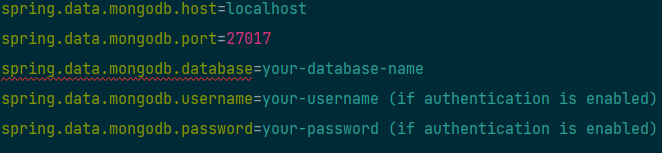
</dependency>

I have provided MongoConfig too so use it to properly connect database to your compass

### 9.2. Configure MongoDB Connection:

In your **application.properties** or **application.yml** file, specify the MongoDB connection details. Make sure to adjust the values to match your local MongoDB setup.

**Using application.properties (Recommended):**



## 10 Implementing Email Service

**Overview**

The **Email** class in your Spring Boot application is responsible for sending various types of emails, including promotional emails and order confirmation emails. This documentation provides an in-depth explanation of the class, its methods, and how to configure and use it effectively.

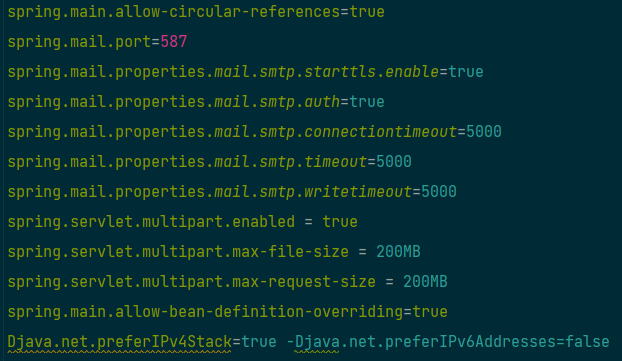
The **EmailServiceImpl** class encapsulates email-sending functionality in your application. It offers two key methods: **sendEmail** for sending general promotional or informational emails and **order\_email** for sending order confirmation emails. This documentation provides comprehensive guidance on setting up and using the **EmailServiceImpl** class effectively.

**Dependencies**

As for dependency I have already provided you the one in pom.xml file

### 10.1 Configurating Email Service for spring boot

Adding this to pom.xml file will configure the service with smtp service choose which smtp service you want below I have added the guidance for SMTP service from google



Click on this to know how SMTP works and how to generate Auth for SMTP

[Link :-](https://springjava.com/spring-boot/sending-mail-using-gmail-smtp-server-in-spring-boot) https://springjava.com/spring-boot/sending-mail-using-gmail-smtp-server-in-spring-boot