Project Title : One Stop

- **Group Name: THUNDER**

- **Group Members :**
  - Varun Raj
  - Krishna
  - Monis Anis
  - Yusuf Akhtar
  - Dikshit Kumar

Project Overview

One Stop is an online restaurant food ordering system for a particular restaurant. This project works on a dual module where through the user module one can order food items online and through the admin module one can accept/reject the orders and also can update the food items.

Project Approach

In this project we will be moving forward with the principle of RESTful API where the project will completely operate on Swagger API Documentation where Customers and Admin can perform their work.

Business Scenario

'One Stop' is an online restaurant food ordering system for a particular restaurant. One Stop provides its customers with the details of available food items in their restaurant, and lets its users order it from their location .

The One Stop application will work on a RESTful architecture where we will perform all methods of the backend part of the application.Every Restaurant have some rules and Regulations that are followed:

One Stop

RESTful Services

- **For Admin :**
  1) **PinCodeController:**
     1. **createDetails() -** Admin using this method will create new areas where the restaurant will be providing services with the help of pincode.
     2. **getall()** - Admin using this method will be able to display all the areas where their restaurant is delivering.
     3. **getupdate()** - Admin using this method will be able to alter the existing records in the database.
     4. **delete()** - Admin using this method will be able to delete the existing records in the database.
     5. **getid()** - Admin will be able to see the record of a particular pincode by entering its id using this method

  2) **FoodMenuList:**

     1. **savemenu() -** Admin will be able to add new items into their restaurant menu using this method.
     2. **updatemenu()** - Admin will be able to alter the existing records in the database.
     3. **deletemenu()** - Admin will be able to delete the existing records in the database.
     4. **getfooddetails()** - Admin will be able to see the record of a particular dish by entering its id using this method.
     5. **fetchalldata()** - Admin will be able to display all the records in the database using this method.

- **For Customer :**
  1. **PostMapping :** Customers will Create orders for the restaurant.
  2. **PutMapping :** Customers will update their order.
  3. **GetMapping :** Customers Will Able to See Status (whether its delivered or in process)

One Stop

Software Requirements

#	Item	Specification/Version
1	Eclipse IDE	Oxygen/2022-06
2	Maven	3.x
3	JDK	11
4	MYSQL	MYSQL 8.0 commandLine
5	SpringTools 4 suite(SpringBoot)	4.15.3
6	Swagger Documentation (springdoc-openapi-ui)	1.6.9
7	Spring DEVTOOLS	Auto_Taken

Dependency :

List
1. Lombok
2. Swagger-open-ui
3. MySQL-Driver
4. Spring data jpa
5. Dev Tool

One Stop

DataBase Model :

1. **Restaurant Menu Card :**

   Command for Creating Food List Tables

```
create table food_menu(
 -> dishid int primary key,
 -> foodname varchar(50),
 -> foodtype varchar(10),
 -> description varchar(100),
 -> price float);
```

Food List	
dishId	int PK
foodname	varchar(50)
foodtype	varchar(10)
description	
price	float

2. **PINCODE Details for giving Service in Your Area :**

   Command for Creating Pin List Tables

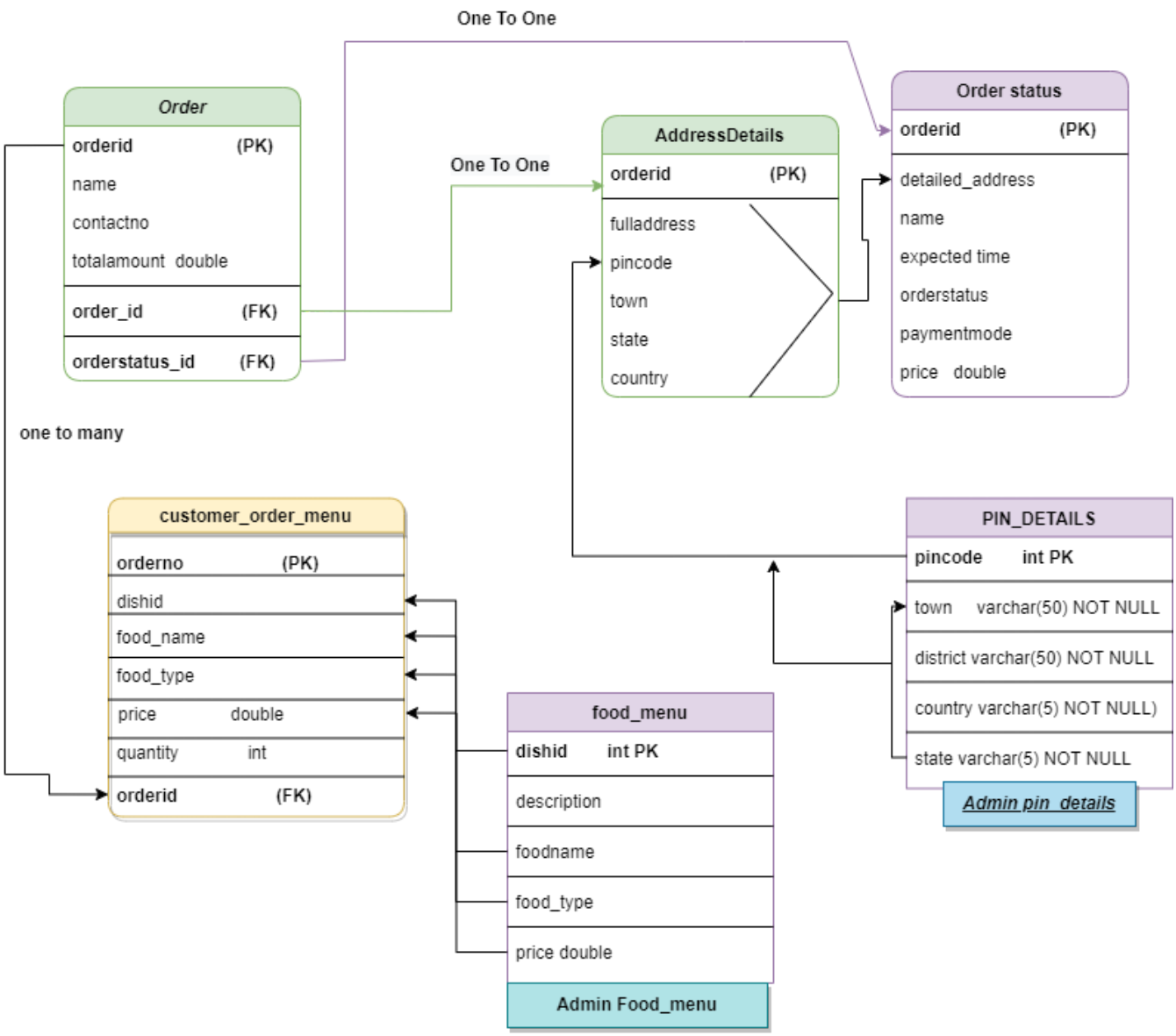
```
create table pin_details(
-> pincode double primary key,
-> town varchar(50) NOT NULL,
-> district varchar(50) NOT NULL,
-> country varchar(5) NOT NULL),
-> state varchar(5) NOT NULL;
```

PIN_DETAILS
pincode Double PK
town varchar(50) NOT NULL
district varchar(50) NOT NULL
country varchar(5) NOT NULL)
state varchar(5) NOT NULL

Method	Details
1. saveMenu() (PostMapping)	adding a dish into the list.
2. updateMenu() (PutMapping)	updating dish item with help of id.
3. fetchallData() (GetMapping)	showing all data to the only to admin
4. deletemenu() (DeleteMapping)	it will delete the record help of dish id.

One Stop

Data-Model :

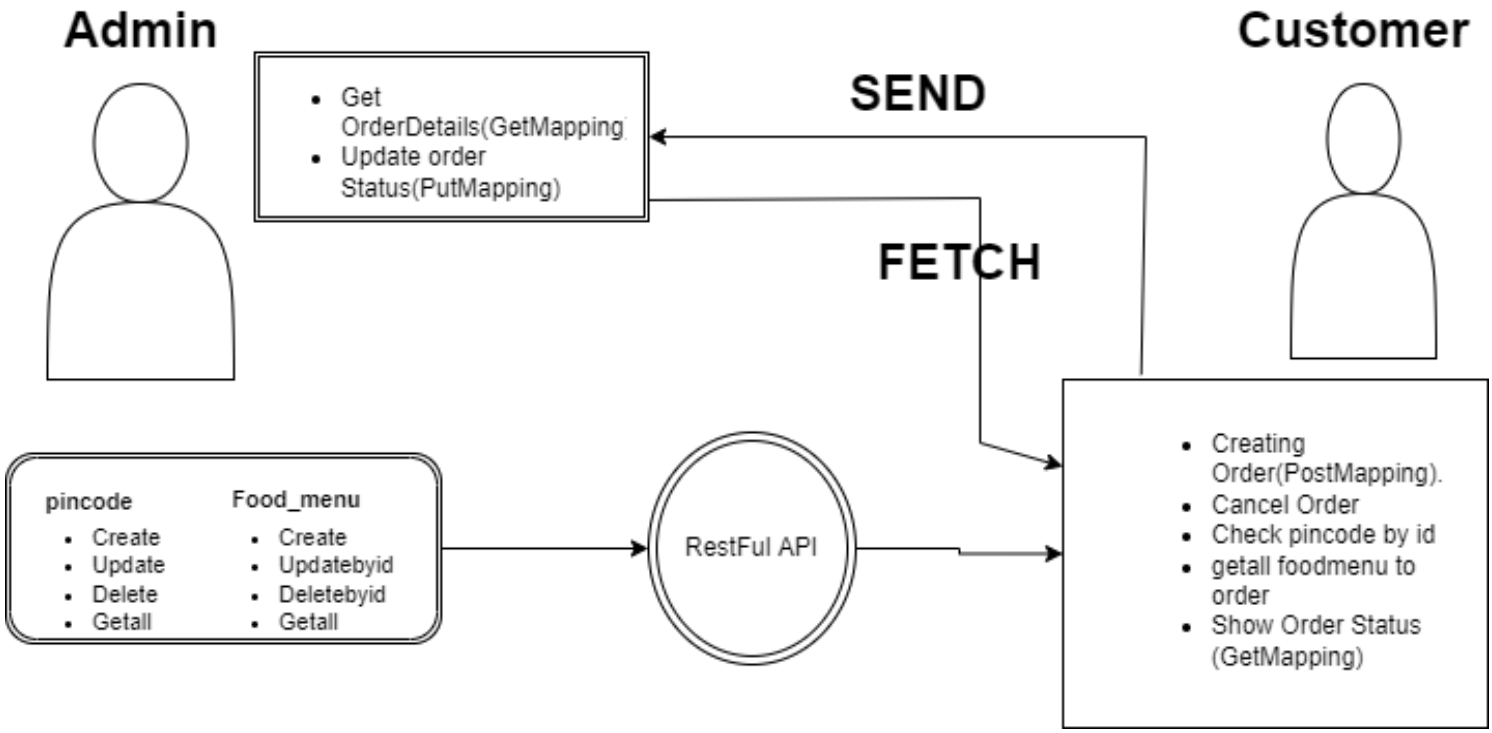


Exception-Handling :

1. **ProductNotFoundException :** This is for handling Food not found in the food records.
2. **PinCodeNotFoundException :** This is for handling PinCode not found in the pincode records.

3. **InvalidPincodeFoundException:** This is for handling if the PinCode is not valid before saving the data into records.
4. **OpeningClosingoneStopException:** This is for handling whether a shop is open or closed at that time for ordering Food.
5. **OrderNotFoundException:** This is for handling whether an order is present or not after ordering food.

Use Case Diagram



One Stop

Used Annotation:

@SpringBoot	This marks a configuration class that declares one or more @Bean methods and also triggers auto-configuration.
@Entity	This specifies that the class is an entity and is mapped to a database table.
@Table	This allows you to specify the details of the table that will be used to persist the entity in the database.
@Data	It is a convenient shortcut annotation that bundles the features of getter/setter, RequiredArgsConstructor ,toString and some more features.
@Service	This marks the class as a service provider.
@Id	This indicates the member field below in the primary key of the current entity.
@GeneratedValue	Provides the generation strategies for the values of the primary keys.
@NoArgsConstructor	This is used to generate the no-argument constructor for the class.
@Autowired	This provides more fine-grained control over where and how autowiring should be accomplished
@ResponseStatus	This marks a method or exception class with the status code and reason message that should be returned
@Requestmapping	This used to map HTTP requests to handler methods of MVC and REST controllers
@RestController	This annotation is applied to a class to mark it as a request handler
@RequestBody	This maps the HttpRequest body to a transfer or domain object, enabling automatic deserialization of the inbound HttpRequest body onto a Java object.

One Stop

@GetMapping	Annotation for mapping HTTP GET requests onto specific handler methods.
@PostMapping	Annotation for mapping HTTP POST requests onto specific handler methods.
@PutMapping	Annotation for mapping HTTP PUT requests onto specific handler methods.
@DeleteMapping	Annotation for mapping HTTP DELETE requests onto specific handler methods.
@PathVaribale	Annotation which indicates that a method parameter should be bound to a URI template variable
@Valid	It automatically bootstraps the default JSR 380 implementation — Hibernate Validator — and validates the argument.
@RestControllerAdvice	It is a kind of interceptor that surrounds the logic in our Controllers and allows us to apply some common logic to them
@ExceptionalHandler	Annotation for handling exceptions in specific handler classes or handler methods.
@Override	Annotation denotes that the child class method overrides the base class method.
@NotBlank	It is used for validation of the input.