



# Build an E-marketplace mobile application.







Introduction to the course: What does this course aim to achieve?	Building customer-facing e-marketplace application from scratch using Flutter framework, Java, and the Spring Boot environment to create a robust backend supporting payments with Razor pay payment gateway. In this course, you'll build a full-stack e-marketplace application which require full-stack development, involving a backend to handle users, inventory, and payments, and a frontend for customers to view products, manage their cart, and checkout using razor pay. User profiles will also store order history.
What is being built in this course	E-marketplace mobile application integrated with Razor pay payment gateway.
How is it being tested	<ol> <li>Install the generated .APK file onto an Android device.</li> <li>Open the postman REST client and test the APIs.</li> <li>Make a test payment to verify the Razor pay payment gateway integration</li> </ol>
Course Prerequisites	Basic knowledge of Dart language Basic Java programming





# **Contents**

Prerequisites

Components

Software

Building the frontend of E-marketplace mobile application using Flutter framework.

- Installing Flutter and Android Studio
- Creating a responsive E-marketplace mobile application.
- Importing the Project

Building the backend of E-marketplace using Spring Boot framework.

- Installing Java SE 13 (JDK)
- Installing Apache NetBeans IDE
- Importing the Project
- Installing Resin
- Deploying war file in the resin.

Creating a database for E-marketplace in PostgreSQL.

Testing the backend with the mobile application.





# Prerequisites:

ТОРІС	LINK
Introduction to Flutter	https://docs.flutter.dev/
Create a Flutter Project	Flutter Tutorial Part 1: Build a Flutter App From Scratch - DZone
from Scratch	
Add to the app using the	https://docs.flutter.dev/development/tools/pubspec/
Pub spec file	
Flutter Logo:	https://www.geeksforgeeks.org/flutter-flutterlogo-widget/
Flutter Toaster	https://pub.dev/packages/fluttertoast
Flutter Drawer	https://docs.flutter.dev/cookbook/design/drawer
Shared preferences	https://blog.logrocket.com/using-sharedpreferences-in-
	flutter-to-store-data-locally/
Widgets	https://docs.flutter.dev/development/ui/widgets-intro/
Container	https://api.flutter.dev/flutter/widgets/Container-class.html
Row and Column	https://www.geeksforgeeks.org/row-and-column-widgets-in-
	flutter-with-example/
Expanded Widget	https://api.flutter.dev/flutter/widgets/Expanded-class.html
Floating Action Button	https://api.flutter.dev/flutter/material/FloatingActionButton-
	<u>class.html</u>
List Tile Widgets	https://api.flutter.dev/flutter/material/ListTile-class.html
Card Widget	https://api.flutter.dev/flutter/material/Card-class.html
List view	https://api.flutter.dev/flutter/widgets/ListView-class.html
Grid View	https://api.flutter.dev/flutter/widgets/GridView-class.html
Custom Fonts	https://docs.flutter.dev/cookbook/design/font
Material Icons	https://docs.flutter.dev/development/ui/widgets/material
Making Responsive App	https://docs.flutter.dev/release/breaking-changes/buttons
Stateful Widget	https://api.flutter.dev/flutter/widgets/StatefulWidget-
statera mager	<u>class.html</u>





Stateless Widget	https://api.flutter.dev/flutter/widgets/StatelessWidget- class.html
Text Field	https://docs.flutter.dev/cookbook/forms/text-field-changes/
API	https://www.tutorialspoint.com/flutter/flutter accessing rest_ api.htm/
JSON	https://docs.flutter.dev/development/data-and-backend/json/ https://blog.logrocket.com/how-parse-json-strings-flutter/ https://medium.com/flutter-community/flutter-part-4-fetch- data-from-the-network-1b5949d84d44/
Cart Feature	https://www.dbestech.com/tutorials/how-to-remove-an-item- from-dart-list-flutter/

# <u>Components</u>

Components	Quantity
Window 10 / Linux 64-bit Pc or Laptop	
• RAM: Min: 8GB	
Recommended: 16GB <ul> <li>Free Disk Space:</li> </ul>	1 No





Min: 10GB	
Recommended: 30GB	
Screen Resolution: Min: 1280x800px	
Recommended: 1920x1080px	

# <u>Software</u>

Software	Download Link
Android	https://developer.android.com/studio
Studio IDE	
and SDK	
Flutter SDK	https://docs.flutter.dev/get-started/install
Java SE 13	https://www.oracle.com/java/technologies/javase/jdk13-
	archive-downloads.html
Apache	https://netbeans.apache.org/download/index.html
NetBeans	
Resin	www.caucho.com/resin-4.0/admin/starting-resin.xtp
PostgreSQL	https://www.postgresql.org/download/





<u>Part- A Build the frontend of E-marketplace mobile application.</u> Building the frontend of E-marketplace mobile application using Flutter framework.

### **Installing Flutter and Android Studio:**

### **1.System requirements:**

To install and run Flutter, your development environment must meet these minimum requirements:

**Operating Systems**: Windows 7 SP1 or later (64-bit), x86-64 based.

Disk Space: 1.64 GB (does not include disk space for IDE/tools).

Tools: Flutter depends on these tools being available in your environment.

Windows PowerShell 5.0or newer (this is pre-installed with Windows 10)

Git for Windows 2.x, with the **Use Git from the Windows Command Prompt** option. If Git for Windows is already installed, make sure you can run git commands from the command prompt or PowerShell.

### 2.Get the Flutter SDK:

- 1. Download the following installation bundle to get the latest stable release of the
- 2. FlutterSDK: <u>https://storage.googleapis.com/flutter\_infra\_release/releases/stabl\_e/windows/\_flutter\_windows\_2.10.3-stable.zip</u>
- 3. Extract the zip file and place the contained flutterin the desired installation location for flutter SDK (for example, C:\Users\<your-username>\Documents).
- 4. If you don't want to install a fixed version of the installation bundle, you can skip steps 1 and 2. Instead, get the source code from the <u>https://github.com/flutter/flutter</u> on GitHub, and change branches or tags as needed. For example: git clone https://github.com/flutter/flutter.git -b stable

You are now ready to run Flutter commands in the Flutter Console.





### 3.Update your path.

If you wish to run Flutter commands in the regular Windows console, take these steps to addFlutter to the PATHenvironment variable:

- From the Start search bar, enter 'env' and select **Edit environment** variables for youraccount.
- Under **User variables** check if there is an entry called **Path**:
- If the entry exists, append the full path to flutter\bin using; as a separator from existing values.
- If the entry doesn't exist, create a new user variable named Pathwith the full path to flutter\binas its value.

You must close and reopen any existing console windows for these changes to take effect.

### 4.Run flutter doctor.

From a console window that has the Flutter directory in the path (see above), run the followingcommand to see if there are any platform dependencies you need to complete the setup:

### C:\src\flutter>flutter doctor

This command checks your environment and displays a report of the status of your Flutter installation. Check the output carefully for other software you might need to install or further tasks to perform.

For example:

[-] Android toolchain - develop for Android devices.

• Android SDK at D:\Android\sdk

X Android SDK is missing command line tools; download from

### https://goo.gl/XxQghQ

- Try re-installing or updating your Android SDK,
- visit https://docs.flutter.dev/setup/#android-setup for
- detailedinstructions.

The following sections describe how to perform these tasks and finish the setup process. Once youhave installed any missing dependencies, you can run the flutter doctor command again to verify that you've set everything up correctly.





### 5.Install Android Studio

- 1. Download and install Android studio.
- 2. Start Android Studio and go through the 'Android Studio Setup Wizard'. This installs thelatest Android SDK, Android SDK Command-line Tools, and Android SDK Build-Tools, which are required by Flutter when developing for Android.
- 3. Run flutter doctor to confirm that Flutter has located your installation of AndroidStudio. If Flutter cannot locate it, run flutter config android <u>https://developer.android.com/studio</u>-studio-dir.

<directory>to set the directory that Android Studio is installed to.

#### 6.Set up an Android device:

To prepare to run and test your Flutter app on an Android device, you need an Android devicerunning Android 4.1 (API level 16) or higher.

- 1. Enable **Developer options** and **USB debugging** on your device. Detailed instructions areavailable in the Android Documentation.
- 2. Windows-only: Install the Google USB driver.
- 3. Using a USB cable, plug your phone into your computer. If prompted on your device, authorize your computer to access your device.
- 4. In the terminal, run the flutter devices command to verify that Flutter recognizes your connected Android device. By default, Flutter uses the version of the Android SDK where your adb tool is based. If you want Flutter to use a different installation of the Android SDK, you must set the ANDROID\_SDK\_ROOT environment variable to that installation directory.

### 7.Set up the Android emulator.

To prepare to run and test your Flutter app on the Android emulator, follow these steps:

- Enable VM acceleration on your machine.
- Launch Android Studio, click the AVD Manager icon, and select Create Virtual Device...
- In older versions of Android Studio, you should instead launch Android Studio >Tools > Android > AVD Manager and select Create Virtual Device.... (The Android submenu is only present when inside an Android project.)





- If you do not have a project open, you can choose **Configure > AVD Manager** and select **Create Virtual Device**...
- Choose a device definition and select **Next**.
- Select one or more system images for the Android versions you want to emulate and select.

**Next**. An *x86* or *x86\_64* image is recommended.

- Under Emulated Performance, select **Hardware GLES 2.0** to enable hardware acceleration.
- Verify the AVD configuration is correct and

### select Finish.

For details on the above steps, see Managing

AVDs.

• In Android Virtual Device Manager, click **Run** in the toolbar. The emulator starts up and displays the default canvas for your selected OS version and device.

### **8.Agree to Android Licenses**

Before you can use Flutter, you must agree to the licenses of the Android SDK platform. This stepshould be done after you have installed the tools listed above.

1. Make sure that you have a version of Java 8 installed and that your JA-VA\_HOME environment variable is set to the JDK's folder.

Android Studio versions 2.2 and higher come with a JDK, so this should already be done.

- 2. Open an elevated console window and run the following command begin.
- 3. signing licenses.flutter doctor –android-licenses.
- 4. Review the terms of each license carefully before agreeing to them.
- 5. Once you are done agreeing with licenses, run flutter doctoragain to confirm thatyou are ready to use Flutter.

### 9.Install the Flutter and Dart plugins.

The installation instructions vary by platform.





#### Mac

Use the following instructions for macOS:

- 1. Start Android Studio.
- 2. Open plugin preferences (**Preferences > Plugins** as of v3.6.3.0 or later).
- 3. Select the Flutter plugin and click Install.
- 4. Click **Yes** when prompted to install the Dart plugin.
- 5. Click **Restart** when prompted.

### **Linux or Windows**

Use the following instructions for Linux or Windows:

- 1. Open plugin preferences (File > Settings > Plugins).
- 2. Select **Marketplace**, select the Flutter plugin and click **Install**.

### **10.Configure Android Studio for Flutter Development:**

After installing Dart and Flutter plugins create a flutter app to check if it is workingproperly or not, to do so follow the steps mentioned below:

1: Open the IDE and select Start a new Flutter project



2: Select the Flutter Application as the project type. Then click Next.







**3:** Verify the **Flutter SDK path** specifies the SDK's location **(select Install SDK... if the textfield is blank)**.

ġ,	New Flutter Application	
	Project name	
	flutter_app	
	Flutter SDK path	
	C:\flutter	▼ ¥ Install SDK
	Project location	
	C:\Users\msaur\AndroidStudioProjects	<u> </u>
	Description	
	A new Flutter application.	
		Create project offline

**4:** Enter a project name (for example, *myapp*). Then click **Next**.

Crea	ate New Flutter Project X	
<b>\$</b>	New Flutter Application	
	Set the package name	
	Applications and plugins need to generate platform-specific code	
	Package name	
	com.example.flutterapp	
	AndroidX	
	Use androidx.* artifacts	
	Platform channel language	
	Include Swift support for IOS code	
	Previous Next Cancel Finish	
lick <b>Fin</b>	lish.	





6: Wait for Android Studio to install the SDK and create the project.

**Note:** When creating a new Flutter app, some Flutter IDE plugins ask for a company domain namein reverse order, something like co. Example. The company domain name and project name are used together as the package name for Android (the Bundle ID for iOS) when the app is released. If you think that the app might be released, it's better to specify the package name now. The packagename can't be changed once the app is released, so make the name unique.

The above steps create a Flutter project directory called *flutter\_app* that contains a simple demo appthat uses Material Components.

### **11.Running the application:**

Follow the below steps to run the flutter application that was structured above:



**1:** Locate the main Android Studio toolbar:

**Step 2:** In the **target selector**, select an Android device for running the app. If none are listed as available, select **Tools> Android > AVD Manager** and create one there. For details, see <u>ManagingAVDs</u>.

**Step 3:** Click the run icon in the toolbar or invoke the menu item

**Run > Run**. After the app build completes, you'll see the starter

app on your device.





Creating a responsive E-marketplace mobile application with following features for merchant and customer

#### Merchant Customer Merchant login page Customer login page Merchant home page Customer registration page Merchant category list page Customer home page Merchant category details page My profile page. Product display page Merchant categories add page. Merchant product list page Cart page Payment page Merchant product details page My order page. Merchant products add page. Order summary page Merchant products edit page. Merchant order received page

## **Import the Project**

1) Open -Android Studio->select unzip emarketplace\_app file->open project.







### **Inside lib Folder**

~	
	> 🖿 api
	> Components
	> 🗖 Screens
	shared_preferences
	🗞 generated_plugin_registrant.dart
	🗞 main.dart
>	linux

**API:** Inside api/api.dart it contains api call details all page

**Components:** All components (rounded button, text field container) are available here. we can use all the above using the required class.

Shared preferences: In this dart file contain variable store the login details data.

### Create Customer login Page.

Here Text field widget used for user input as email id and password. Flat Button widget, to show action. Also, used Image to set logo for login page.

### Inside lib/Screen/customer\_login







Background. Dart: All page background class are available here

**CustomerLogin\_pojo.dart**: There have a static method called from Json which receives Map object. Then set mail id and password values from our Map object called data. Now can use our function to convert our json to class.

**customer\_detail\_pojo.dart:** There have a static method called from Json which receives Map object. Then we set name, emailid, password address and phone number values from our Map object called data. Now use our function to convert our json to class.

**customer\_login\_screen.dart:** It contain main function entry of my program execution, and MyApp class which takes an object of Login class as a parameter of the home property.

### body.dart:

- Set Scaffold's app Bar property as follows to make heading for our application.
- For this UI, all widgets are placed inside the Column widget, into the Scaffold body. The first child of Column is the Container widget which holds Image widget as it's child.
- flutter-logo.png file copied into asset/images folder in this flutter application and write into pubspec.yaml file to get it in our code.
- Then, for email id and password use the TextField widget from inside component.RoundedmailLoginfield function for emailid and Roundedpasswardfield password is an input widget that helps you to take input from the user
- For the login button, use from component Roundedbutton 'Login' as a child and onPressed() of this button we can write code for control navigation to another home screen. After pressed login button all customer detail sends





to backend

# // API Call from server //

String url = api.customerlogin;

## // Write the following code Inside lib/Screen/customer\_login/body.dart //

var res = await http.post(Uri.parse(url), headers: {'Content-Type': 'application/json'}, body: json.encode({'email': login.email, 'password': login.password})); // Connect both frond and end back server // var data = json.decode(res.body); // Status command from backend server // var Response = data["Customerdetails"] as List; // mapping with POJO code with customer detail // customer\_detail = Response.map<Customer\_detail>((json)=> Customer\_detail.fromJson(json)).toList();

**Response:** When customer logins, api call occurs and a list appears which contain customer details and in that all customer details are stored using shared preference.

### Stringvalue.email = sharedPreferences.getString("email");

**http.post**: Request the specified url through POST method by posting the supplied data and return the response as Future<Response>

http.get: is used to fetch the data from the Internet.

**json.encode:** The Encodable function is used to convert it to an object that must be directly encodable.

**json.decode:** Is used to decode the JSON data into the Dart Map object. Once JSON data is decoded, it will be converted into List<customerdetails> using fromMap of the CustomerLogin class.

Flutter toast: Once get the response from backend "Login Successfully" add





flutter toaster message.

### **Create Customer Registration Page**

Here Text field widget used for user input as name, emailid, password and phonenumber Flat Button widget, to show action. Also, used Image to set logo for Registration page.

### Inside lib/Screen/customer\_signup



background. Dart: All page background class are available here

**Customersignup\_Pojo.dart**: There have a static method called from Json which receives Map object. Then set name, emailid, password phone number and address values from our Map object called data. Now use our function to convert our json to class.

or\_divider.dart: It contains divider line function

**social\_icon.dart:** It contains the social icon function

**customer\_signup\_screen.dart:** It contains main function entry point of my program execution, and MyApp class which takes an object of Login class as a parameter of the home property.





### Body.dart:

- Set Scaffold's appBar property as follows to make heading for our application.
- For this UI, all widgets are placed inside the Column widget, into the Scaffold body. The first child of Column is the Container widget which holds Image widget as it's child.
- flutter-logo.png file copied into asset/images folder in this flutter application and write into pubspec.yaml file to get it in our code.
- Then, for emailed and password use the TextField widget from inside component.RoundedmailLoginfield for emailed, name, password and phone number is an input widget that helps you to take input from the user
- For the login button, use from component Roundedbutton 'Signup' as a child and onPressed() of this button we can write code for control navigation to another home screen. After pressed login button all customer detail sends to backend

### // API Call from server //

### String url = api.customerregister;

**http.post**: Request the specified url through POST method by posting the supplied data and return the response as Future<Response>

http.get: Is used to fetch the data from the Internet.

**json.encode**: The Encodable function is used to convert it to an object that must be directly encodable.

**Flutter toast**: Once get the response from backend "Register Successfully" add flutter toaster message.





# // Write the following code Inside lib/Screen/customer\_signup/body.dart //

```
var res = await http.post(Uri.parse(url),
    headers: {'Content-Type': 'application/json'},
    body: json.encode({
        'name': siginup.name,
        'email': siginup.email,
        'password': siginup.password,
        'phone': siginup.phone,
    }));
    // status command get from back end server //
    if (res.body == "Registered Successfully") {
        Navigator.push(
            context,
            MaterialPageRoute(
            builder: (context) => LoginScreen(),
            ));
    ));
```

```
// flutter toast command //
Fluttertoast.showToast(
    msg: "Registered Successfully",
    toastLength: Toast.LENGTH_SHORT,
    gravity: ToastGravity.CENTER,
    timeInSecForIosWeb: 2,
    backgroundColor: Colors.black,
    textColor: Colors.white);
} else {
```

```
Fluttertoast.showToast(
msg: "Invalid user",
toastLength: Toast.LENGTH_SHORT,
gravity: ToastGravity.CENTER,
timeInSecForlosWeb: 2,
backgroundColor: Colors.black,
textColor: Colors.white);
```

```
}
```

### **Customer Home Page:**

#### Inside lib/Screen/customer\_home





- customer\_home
  - 🗞 ApiServiceProjectDetail.dart
  - 🗞 CategoryDetail.dart
  - 🔷 home.dart
  - 🔷 profile.dart
  - 🗞 ProfileDetail\_pojo.dart
  - 🗞 ProfileDetailMatch\_pojo.dart

**CategoryDetail.dart:** There have a static method called from Json which receives Map object. Then set category name and category image values from our Map object called data. Now use our function to convert our json to class.

### home.dart:

Set Scaffold's app Bar property as follows to make heading for our application.

### // API Call from server //

### String url = api.category;

// Write the following code Inside lib/Screen/customer\_home/home.dart
//

```
Future GetCategoryList() async {
var res = await http
.get(Uri.parse(url), headers: {'Content-Type': 'application/json'});
print("category list success${res.body}");
if (res.body != null) {
var data = json.decode(res.body);
// List the category name image in this list //
var Response = data["Category"] as List;
//Map the category detail //
setState(() {
    categorydetial =
    Response.map <CategoryDetail>((json) => CategoryDetail.fromJson(json))
.toList();
}); }}
```





When category, api call occurs and a list appears which contain category details.

http.get: Is used to fetch the data from the Internet.

**json.encode:** The Encodable function is used to convert it to an object that must be directly encodable.

**json.decode:** Is used to decode the JSON data into the Dart Map object. Once JSON data is decoded, it will be converted into List<categorydetail> using from Map of the Categorydetail class pojo code.

• In Flutter, you can encode a local or network image (or another kind of file) to a base64 string like this Ref link:

# https://www.kindacode.com/snippet/flutter-turn-an-image-into-abase64-string-and-vice-versa/

- Drawer and list view My profile, My cart, My order, logout.
- Display the Category image Base64 is an encoding scheme that can carry data stored in binary formats. The application of base64 string is common in web and mobile app development.
- The Image. Memory constructor helps to display images from bytes. Hence, we must convert the base64 string to bytes using dart convert and display mage list view constructor. The standard List View constructor works well for small lists. To work with lists that contain many items, it's best to use the ListView.builder constructor.
- In contrast to the default List View constructor, which requires creating all items at once, the ListView.builder() constructor creates items as they're scrolled onto the screen.





# **Create My Profile Page:**

ApiServiceProjectDetail.dart: It contains api detail update profile.

### String get updateprofile

### =>"http://121.242.232.216:7070/emarket/updateprofile";

When customer logins, api call occurs and a list appears which contain customer details using shared preferences.

**profiledetailpojo.dart**: There have a static method called from Json which receives Map object. Then we set name, emailid, phonenumber and address values from our Map object called data. Now use our function to convert our json to class.

**Flutter toast**: Once get the response from backend server "Update Successfully" add flutter toaster message.

# /// Write the following code Inside lib/Screen/customer\_home/profile.dart //

```
Map data = {
  "email": "${Stringvalue.email}",
  "password": "${Stringvalue.password}"
 };
 final loginRequestJson = jsonEncode(data);
 var res = await http.post(Uri.parse(url),
    headers: {'Content-Type': 'application/json'}, body: loginRequestJson);
 if (res.body != null) {
  var data = json.decode(res.body);
  // list of customer detial from server //
  var Response = data["Customerdetails"] as List;
// map profile detail //
  profiledetailmatch =
     Response.map < ProfileDetailMatch > ((json) = > ProfileDetailMatch.fromJson(json))
        .toList();
}
```





# // API Call from server //

### String url = api.customerlogin;

**json.decode:** Is used to decode the JSON data into the Dart Map object. Once JSON data is decoded, it will be converted into List<profiledetailmatch> using from Map of the ProfileDetailMatch class pojo code.

**profile.dart**: Flutter User Profile Page UI where you can access and edit your user's information within your Flutter app Text field Controller

it's useful to run a call back function every time the text in a text field change have edit form where some data in text fields controller from database.

Here is my requirement, when I click the Update button, dynamically new cards with nine Text Fields should be generated,

eg: name, emailid password, doorno, area, city, state and pincode. Once update all value click update button all data send to backend using **customer login api** call

### **<u>Create Product List and Add to Cart Page:</u>**

Inside lib/Screen/product\_management





product\_management
 add\_product.dart
 api\_service.dart
 category\_pojo.dart
 edit\_product.dart
 edit\_product.dart
 main.dart
 product\_detail.dart
 product\_list.dart
 product\_pojo.dart

We will be using SQLite and Shared Preferences in our application to store the data locally on the device itself. SQLite and Shared Preferences store data, while Provider manages the application's state.

### **API CALL PRODUCT LIST**

When products, api call occurs and a list appears which contain product details.

## // API call product //

### String url = api.product;

### // Write the following code Inside

### lib/Screen/product\_management/product\_list.dart //

List < Product\_detail > product\_detail; bool isloading = false; Future < dynamic > productdetailsfuture; // function get all product list // Future getallproductlist() async { var res = await http.post(Uri.parse(url), headers: {'Content-Type': 'application/json'}, body: json.encode({'category': categoryname})); if (res.body != null) { var data = json.decode(res.body); // get the product detail from database // var Response = data["Productdetails"] as List;





```
product_detail =
Response.map <Product_detail>((json) => Product_detail.fromJson(json))
.toList();
print ("the product descr ${product_detail[0]
.productdetails
.toString()}");
setState(() {
    isloading = true;
});
}
```

**http.post**: Request the specified url through POST method by posting the supplied data and return the response as Future<Response>

**json.encode:** The Encodable function is used to convert it to an object that must be directly encodable.

**json.decode:** Is used to decode the JSON data into the Dart Map object. Once JSON data is decoded, it will be converted into List<product\_detail> using from Map of the product\_detail class.

## **<u>1. Build shopping cart:</u>**

The first is a product screen, which displays a list of products along with photos, the name of the product, and the price. Each list item includes a button. that allows you to add it to your shopping basket. The AppBar includes a shopping cart icon with a badge that updates the item count whenever a user presses the Add to Cart button. The second screen, the shopping cart screen, displays a list of the things that the user added to it. If the user decides to remove it from the cart, a delete button removes the item from the cart screen. The entire cost is shown at the bottom of the screen. A button that, for the time being, displays a Snack Bar confirming that the payment has been processed.

### <u>2. SETUP:</u>





Next, we are going to start off with creating our model classes named Cart and Item. So, create a new Dart file and name it cart\_model, or you can also name it per your requirements **Ref (Cartmodel.dart)** 

Create another Dart file and enter product\_pojo (Ref:product\_pojo.dart)

# 3. Add Sqflite:

As previously stated, we will be utilizing SQFlite, which is essentially SQLite for Flutter, and we will save the data locally within the phone memory. We are not uploading or retrieving data from the cloud because the objective of this post is to learn the fundamental operation of a cart screen. So, using the SQLite package, we're constructing a database class called DB Helper **(Ref:DBHelper.dart)** 

# 4. Add the Provider Class:

The next step will be to develop our Provider class, which will include all our methods and will separate our UI from the logic that will eventually manage our entire application. We use Shared Preferences in addition to SQLite. The reason for using Shared Preferences is because it wraps platform-specific persistence to store simple data such as the item count and total price, so that even if the user exits the application and returns to it, that information will still be available. **(Ref cart\_provider.dart)** 

# 5. Create a basic Shopping cart UI:

So, starting from the top that is the AppBar, we have added an Icon Button wrapped with our Badge package that we added to our application. The Icon is of a shopping cart and the badge over it shows how many items have been added to our cart. Please have a look at the image and code below. We have wrapped the Text widget with a Consumer widget because every time a user clicks on the Add





to Cart button, the whole UI does not need to get rebuilt when the Text widget must update the item count. And the Consumer widget does exactly that for us

The Scaffold 's body is a List View builder that returns a Card widget with the information from the lists we created, the name of the product, unit, and price per unit, and a button to add that item to the cart.

We have initialized our Cart Provider class and created a function that will save data to the database when the Add to Cart button is clicked. It also updates the Text widget badge in the AppBar and add total price to the Database that will eventually show up in the Cart screen.

### 6. Create Cart Screen:

Moving on to the cart screen, the layout is like the product list screen. When the user clicks the Add to Cart button, the entire information is carried onto the cart screen. The implementation is like what we've seen with other ecommerce applications. The primary distinction between the two layouts is that the cart screen includes an increment and decrement button for increasing and decreasing the quantity of the item. When users click the plus sign, the quantity increases, and when they click the minus sign, the quantity decreases. The total price of the cart is added or subtracted when the plus and minus buttons are pressed. The delete button deletes the item from the cart list and subtracts the price from the total price. Again, we have wrapped our ListView builder with the Consumer widget because only parts of the UI need to be rebuilt and updated, not the whole page. **(Ref:cartscreen.dart)** 

Look towards the end of the code, just before the bottom navigation bar, for a Consumer widget that returns Value Notifier Builder from within the Column widget. It is responsible for updating the quantity for the specific item when the





user clicks either the plus or minus button on the cart screen. There is a bottom navigation bar with a button at the bottom of the screen.

After Pressed the Continue button its navigator to Order summary page

### **Create Order Summary Page:**

Inside lib/Screen/order\_summary

- ✓ order\_summary
  - 🔦 order\_summary.dart
    - 🗞 order\_summary\_pojo.dart

**order\_summary\_pojo.dart**: There have a static method called from Json which receives Map object. Then set customer detail order date and time order id and product details values from our Map object called data. Now use our function to convert our json to class.

### order\_summary.dart:

- Previously, store the customer detail id each customer (name emailid phone number, address) data by using Shared Preferences is the way in which one can store and retrieve small amounts of primitive data as key/value pairs to a file on the device storage.
- Display the store all shared preferences top corner of the page.
- Now create a new class named as OrderSummaryScreen() this will be going to be a stateful class because our application does change its state at run time. And return MaterialApp().





- The Map object is a simple key/value pair. Keys and values in a map may be of any type. Map data list each element.
- **jsonencode**: The Encodable function is used to convert it to an object that must be directly encodable.

# // MAP DATA//

```
Map mapData;
list.forEach((element)
{
mapData = {
"productid": element.productId,
"productname": element.productName,
"productquantity": element.initialquantity,
"productprice": element.productPrice,
"productimage": element.productPrice,
"productdetail" : element.productDetails,
};
ls.add(json.encode(mapData));}
```

# // API call for order summary //

### String get order summary =>

### "http://121.242.232.216:7070/emarket/ordersummary"

• Send all detail to order summary api call to backend server using json format

### // Write the following code Inside

### lib/Screen/order\_summary/order\_summary.dart //

var request = json.encode([{"productlist":json.decode("\${ls}"),"totalprice":

"\${total}",

```
"customerid":"${Stringvalue.id}","customername": "${Stringvalue.name}","customeremail":
"${Stringvalue.email}",
"customerphoneno": "${Stringvalue.mobilenumber}",
```

```
"customeraddressno": "${Stringvalue.addressno}",
```





"customerarea": "\${Stringvalue.area}",
"customercity": "\${Stringvalue.city}",
"customerstate": "\${Stringvalue.state}",
"customerpincode": "\${Stringvalue.pincode}"
}]);// mapping
var res = await http.post(Uri.parse(ordersummary),
headers: {'Content-Type': 'application/json'},
body: request);

- Once Get the response "Inserted Successfully".
- Flutter toast: Once get the response from backend "Order Successfully "add flutter toaster message.
- Follow the same procedure Cart screen page.
- Total price value passed through payment function (Ref: payment.dart)
   We can use Navigator. push () to navigate to a new route and Navigator.
   pop () to navigate to the previous route.
- After Getting response from backend server customer get invoice mail to customer email id.

### **<u>Create Order Management Page:</u>**

Inside lib/Screen/order\_management

- Image of the second second
  - 🗞 order\_mangament.dart
  - 💿 orderlist\_pojo.dart

## //API CALL FOR ORDER PAGE //

String url = api.merchantorder;

orderlist\_pojo.dart: There have a static method called from Json which receives





Map object.

Then set orderdetail customer detail and product detail values from our Map object called data. Now we can use our function to convert our json to class.

### // Write the following code Inside

### lib/Screen/order\_management/order\_management.dart //

```
var res = await http.post(Uri.parse(url),
headers: {'Content-Type': 'application/json'});
if (res.body != null) {
var data = json.decode(res.body);
// get the orderdetail from backendd side //
var Response = data["orderdetails"] as List;
```

setState((){
merchantorder\_detail =
Response.map<Merchantorder\_detail>((json) => Merchantorder\_detail.fromJson(json)).toList();}
);}
json.decode: Is used to decode the JSON data into the Dart Map object. Once
JSON data is decoded, it will be converted into List<merchantorder\_detail> using
from Map of the Merchantorder\_detail class pojo code.

### **Create Merchant Login Page**:

### Inside lib/Screen/merchant\_login

- 🗸 🖿 merchant\_login
  - components
    - 🗞 background.dart
    - 🔷 body.dart
    - 🗞 MerchantLogin\_pojo.dart
    - nerchant\_login\_screen.dart

#### Follow up same procedure as Customer Login Page





body.dart:

## // API Call from server //

String url = api.merchantlogin;

### // Write the following code Inside lib/Screen/merchant\_login/body.dart //

Future GetMerchantLoginDetail() async { var res = await http.post(Uri.parse(url), // json mapping // headers: {'Content-Type': 'application/json'}, body: json.encode({'email': merchantlogindetail.email, 'password': merchantlogindetail.password})); // status command for backend server merchant login page // if (res.body == "Successfull") { Navigator.push( context, MaterialPageRoute( builder: (context) => MerchantScreen(), )); // flutter toast command // Fluttertoast.showToast( msg: "Login Successfully", toastLength: Toast.LENGTH SHORT, gravity: ToastGravity.CENTER, timeInSecForlosWeb: 2, backgroundColor: Colors.black, textColor: Colors.white); } else { Fluttertoast.showToast( msg: "Invalid user", toastLength: Toast.LENGTH\_SHORT, gravity: ToastGravity.CENTER, timeInSecForlosWeb: 2, backgroundColor: Colors.black, textColor: Colors.white); }}

### **Create Category Management:**

### Inside lib/Screen/category\_management





Screens

- category\_management
  - 🗞 add\_category.dart
  - 🗞 api\_service.dart
  - 🗞 category\_detail.dart
  - 🗞 category\_list.dart
  - 🗞 category\_pojo.dart
  - 🔷 main.dart

**add\_category**: Newly add category name and image send to through insertcategory api call to backend.

api\_service.dart: It contains api detail category.

String get insertcategory =>

"http://121.242.232.216:7070/emarket/insertcategory";

String get deletecategory =>

"http://121.242.232.216:7070/emarket/deletecategory";

String get category => "http://121.242.232.216:7070/emarket/category";

category\_detail.dart: Displays category details

category\_list.dart: Displays List of category name

**category\_pojo.dart:** There is a static method called from Json which receives Map object. Then set category name and image values from our Map object called data. Now use our function to convert our json to class.

**main.dart:** The main file of the generated project is the entry point of the Flutter application: void main () =>runApp(MyApp()); The main function by itself is the Dart entry point of an application.





As we mention in the first paragraph, we will use the HTTP library package to access the REST full API from the REST API server. For that, install this package by open and edit pubspec.yaml then add this dependency.

**category\_pojo.dart:** That represent the SQLite table. This class is about category detail.

**api\_service.dar**t Where we will put all CRUD (POST, GET, PUT, DELETE) methods to the REST API. Fill this class with this CRUD operation of HTTP requests to the REST API.

### // Write the following code Inside

### lib/Screen/category\_management/api\_service.dart //

```
List<dynamic> categorydetail = [];
var res = await http.get(Uri.parse(category));
if (res.body != null) {
var data = json.decode(res.body);
if (res.statusCode == 200) {
var Response = data["Category"] as List;
categorydetail=Response.map((item)=> CategoryDetail.fromJson(item)).toList();
} else {
throw "Failed to load cases list1";
}
return categorydetail;
```

http.get: Is used to fetch the data from the Internet.

**Response:** When product, api call occurs and a list appears which contain productdetail

**json.decode:** Is used to decode the JSON data into the Dart Map object. Once JSON data is decoded, it will be converted into List<categorydetail> using from Map of the Categorydetail class pojo code.





### // Write the following code Inside

lib/Screen/categorymanagement/api\_service.dart //

### **Update Category**

Map data = {

'name': updatecategory.name,
};
final Response response = await http.put(
Uri.parse('\$insertcategory/\$id'),
headers: <String, String>{
'Content-Type': 'application/json; charset=UTF-8',
},
body: jsonEncode(data),
);

### **1.CATEGORY LIST:**

- We will display the list of data in a separate Dart file that will call from the main. Dart home page body. For that, we need a dart file to view the list of data. (Ref:category\_list.dart)
- Class name that extends Stateless Widget object. Inside that class, declare these variables that hold Category list that loaded from the main. Dart and create Key for the list. Add an override method after the variables to build the ListView widget for the list of categories. That List View builder contains the Card that has the child of InkWell that use to navigate to the Detail Widget using MaterialPageRoute. The child of the Card is ListTile that contains an Icon (leading), Text (title), and Text(subtitle).
- The Inkwell widget has an on-Tap event with an action to Navigate to the details page. Container, Column, Image, and Text have their own properties to adjust the style or layout.




- Keep in mind, every widget that uses the child only has one widget as its child. If you need to put more than one widget to the parent widget, use children: <Widget> property.
- Next, open and main.dart then replace all Dart codes with these lines of codes to display the List View in the main home page. We use the existing floating button as the add-data button with an action to go to Add Category Widget.dart.

### 2.Category Detail:

We will display data details to another page that opened when tapping on a list item in the list page. For that, create a Dart file in the lib folder detail category.dart. We will use a scrollable Card widget to display a detail to prevent overflow if the Card content is longer. Next, open and edit lib/detailwidget.dart then add these imports of Flutter material, database helper, editdatawidget, and cases object model.

- Add a Detail Widget class that extends Stateful Widget. This class has a constructor with an object field, a field of Category object and \_DetailWidgetState that builds the view for data detail.
- Add a \_DetailWidgetState class that implementing all required widgets to display data details.
- To handle the delete button, we need to add a method or function after the above method that shows an alert dialog to confirm if data will be deleted.

### // Write the following code

Inside lib/Screen/categorymanagement/category\_detail.dart //





return showDialog<void>( context: context, barrierDismissible: false, // user must tap button! builder: (BuildContext context) { return AlertDialog( title: Text('Warning!'), content: SingleChildScrollView( child: ListBody( children: <Widget>[ Text('Are you sure want delete this item?'), ], ), ), actions: <Widget>[ ElevatedButton( child: Text('Yes'), onPressed: () { api.deleteCategory(categoryid); Navigator.push( context, MaterialPageRoute( builder: (context) { return CategoryMainPage(); }, ), ); }, ), ElevatedButton( child: const Text('No'), onPressed: () { Navigator.push( context, MaterialPageRoute( builder: (context) { return CategoryMainPage(); }, ), ); }, ),],);},);}

### **Create Product Management:**

### Inside lib/Screen/product\_management





product\_management
 add\_product.dart
 api\_service.dart
 category\_pojo.dart
 edit\_product.dart
 main.dart
 product\_detail.dart
 product\_list.dart
 product\_pojo.dart

**add\_Product**: Newly add product name, description, price, category, quantity, and image send to through insert product api call to back end.

api\_service.dart: It contains api detail insert delete product.

String get insertproduct =>
"http://121.242.232.216:7070/emarket/insertproduct";
String get updateproduct =>
"http://121.242.232.216:7070/emarket/updateproduct";
String get deleteproduct =>
"http://121.242.232.216:7070/emarket/deleteproduct";
String get listproduct =>
"http://121.242.232.216:7070/emarket/listproduct";
product\_detail.dart: It display product details

**product\_list.dart**: It display List of product details are product name, price, category, description, quantity and image

**category\_pojo.dart:** There is a static method called from Json which receives Map object. Then set category name and image values from our Map object called data. Now use our function to convert our json to class.





**main.dart:**The main file of the generated project is the entry point of the Flutter application: void main() => runApp(MyApp()); The main function by itself is the Dart entry point of an application.

As we mention in the first paragraph, we will use the HTTP library package to access the REST full API from the REST API server. For that, install this package by open and edit pubspec.yaml then add this dependency.

**category\_pojo.dart:** That represent the SQLite table. This class is about category detail.

**api\_service.dar**t where we will put all CRUD (POST, GET, PUT, DELETE) methods to the REST API. Fill this class with this CRUD operation of HTTP requests to the REST API.

## // Write the following code Inside

### lib/Screen/product\_management/api\_service.dart //

```
List<dynamic> product = [];
var res = await http.get(Uri.parse(listproduct));
if (res.body != null) {
var data = json.decode(res.body);
if (res.statusCode == 200) {
var Response = data["Product"] as List;
product = Response.map((item) => Productdetails.fromJson(item)).toList();
} else {
throw "Failed to load cases list1";
}
```

return product;

http.get: Is used to fetch the data from the Internet.

**Response:** Once fetch category api get data product list.it contain each product name, price, category, description, quantity, and image





**json.decode** is used to decode the JSON data into the Dart Map object. Once JSON data is decoded, it will be converted into List<product> using from Map of the Productdetail class pojo code. After getting the Product List it has product name, price, category, decription, quantity and image.

### // Write the following code Inside

### lib/Screen/product\_management/api\_service.dart //

```
Map data = {
'id': id,
'name': updateproducts.name,
'category': updateproducts.category,
'description': updateproducts.description,
'price': updateproducts.price,
'quantity': updateproducts.quantity,
};
final Response response = await http.post(
Uri.parse(updateproduct),
headers: <String, String>{
'Content-Type': 'application/json; charset=UTF-8',
},
body: jsonEncode(data),
);
```

if (response.statusCode == 200) { Fluttertoast.showToast( msg: "Product Update Succesfully", toastLength: Toast.LENGTH\_SHORT, gravity: ToastGravity.CENTER, timeInSecForlosWeb: 2, backgroundColor: Colors.black, textColor: Colors.white); } else { Fluttertoast.showToast( msg: "Product Update Faliure", toastLength: Toast.LENGTH SHORT, gravity: ToastGravity.CENTER, timeInSecForlosWeb: 2, backgroundColor: Colors.black, textColor: Colors.white); }





### **<u>1. PRODUCT LIST:</u>**

- We will display the list of data in a separate Dart file that will call from the main.dart home page body. For that, we need a dart file to view the list of data. (Ref:product\_list.dart)
- Class name that extends Stateless Widget object. Side that class, declare these variables that hold Product list that loaded from the main. Dart and create Key for the list. Add an override method after the variables to build the List View widget for the list of categories. That ListView builder contains the Card that has the child of Inkwell that use to navigate to the Detail Widget using MaterialPageRoute. The child of the Card is List Tile that contains an Icon (leading), Text (title), and Text(subtitle).
- The Inkwell widget has an on-Tap event with an action to Navigate to the details page. Container, Column, Image, and Text have their own properties to adjust the style or layout.
- Keep in mind, every widget that uses the child only has one widget as its child. If you need to put more than one widget to the parent widget, use children: <Widget> property.
- Next, open and edit lib/main.dart then replace all Dart codes with these lines of codes to display the ListView in the main home page. We use the existing floating button as the add-data button with an action to go to AddCategoryWidget.dart.

## 2. Product Detail:

We will display data details to another page that opened when tapping on a list





item in the list page. For that, create a Dart file in the lib folder detail product Dart. We will use a scrollable Card widget to display a detail to prevent overflow if the Card content is longer. Next, open and edit **detailwidget.dart** then add these imports of Flutter material, database helper, editdatawidget, and cases object model.

Add a Detail Widget class that extends Stateful Widget. This class has a constructor with an object field, a field of Product object and \_DetailWidgetState that builds the view for data detail.

Add a \_DetailWidgetState class that implementing all required widgets to display data details.

• To handle the delete button, we need to add a method or function after the above method that shows an alert dialog to confirm if data will be deleted.

## // Write the following code Inside

### lib/Screen/product\_management/product\_detail.dart //

```
return showDialog<void>(
context: context,
barrierDismissible: false, // user must tap button!
builder: (BuildContext context) {
return AlertDialog(
title: Text('Warning!'),
content: SingleChildScrollView(
child: ListBody(
children: <Widget>[
Text('Are you sure want delete this item?'),
1,
),
),
actions: <Widget>[
ElevatedButton(
child: Text('Yes'),
onPressed: () {
api.deleteProducts(id);
Navigator.push(
```





context, MaterialPageRoute( builder: (context) { return MyApp\_edit\_product(); }, ), ); }, ), ElevatedButton( child: const Text('No'), onPressed: () { Navigator.push( context, MaterialPageRoute( builder: (context) { 

### 3. Edit Product Detail

That codes build widgets combination of Container, Card, Column, Image, Text, and Raised Button. The Raised Buttons has on Pressed event that action to navigate to the EditDataWidget and trigger delete confirm dialog. Next, before the closing of \_DetailWidgetState class body add this method or function to navigate to the EditDataWidget with cases object params. The layout for edit data is the same as the add data view with additional object params that get from the details page. This object will fill the default value of the TextFormField and Submit Button. On the submit it will update the data based on the ID then redirect to the list view. First, create a new dart file in the lib folder lib/**editdatawidget.dart**. Open and edit that file then add these lines of the dart codes to build the edit form and function to submit this form to the REST API.





### **CREATE MY ORDER PAGE:**

#### Inside lib/Screen/myorder

- 🗸 🖿 myorder
  - 🗞 my\_order.dart
  - 🗞 my\_order\_pojo.dart

### //API CALL FOR ORDER PAGE //

#### String url = api.customerorder;

**my\_order\_pojo.dart**: There have a static method called from Json which receives Map object. Then set order detail customer detail and product detail values from our Map object called data. Now use our function to convert our json to class.

### // Write the following code Inside lib/Screen/myorder/my\_order.dart //

```
Map data = {
    "customerid": "${Stringvalue.id}",
    ;;
    // json script //
final loginRequestJson = jsonEncode(data);
var res = await http.post(Uri.parse(url),
    headers: {'Content-Type': 'application/json'}, body: loginRequestJson);
    if (res.body != null) {
        var data = json.decode(res.body);
        // list of customer order detail in server //
        var Response = data["orderdetails"] as List;
    setState((){
        customerorder.detail =
```

```
customerorder_detail =
Response.map<Customerorder_detail>((json) => Customerorder_detail.fromJson(json))
.toList();
});
```

```
}
```





Map data with each customer id

**json.decode:** Is used to decode the JSON data into the Dart Map object. Once JSON data is decoded, it will be converted into List<customerorder\_detail> using from Map of the CustomerOrder class pojo code.

my\_order.dart: Just display Customer detail and Product detail
(Ref:productlist.dart)

#### **PAYMENT GATEWAY INTEGRATION:**

Inside lib/Screen/payment

payment
payment.dart

#### **1. Razorpay Payment:**

Razor pay Payments provide a range of products to accept payments and make payouts.

#### 2.Create a Razor pay account and log in to the dashboard:

You must sign up for a Razor pay account to use the Razorpay Payments products and access the Razor pay Dashboard.

#### <u>Sign Up</u>

To create a Razorpay Account, go to the Razorpay website and click Get Started.

Follow these steps for a smooth sign-up process:





- 1. Contact Details
- 2. Platform Details
- 3. Business Type and PAN Details
- 4. Communication Details

### **Contact Details**

Provide your contact details to get started.

1. Enter your 10-digit Mobile Number.

2. Select the check box below to receive updates on WhatsApp. Click Send OTP.

<b>⊿</b> Razorpay		
	Contact Details	
	What's your mobile number?	
	+91	
	Get updates on WhatsApp 🛽	
	Send OTP	
		© 2017-2022 · Merchant agree Terms of use · Privacy policy ·

3. Enter the OTP sent to your mobile number. If you did not receive the OTP, click Resend OTP.

4. Click Submit OTP.

	< Contact Details	
	Enter the OTP sent to +91	
	Resend OT	P D D
	By clicking "Submit OTP", I agree to Terms and Conditions, Privacy Policy, and Service Agreement	
	Submit OTP	
		© 2017-2022 · Merchant agreement · Terms of use · Privacy policy · Support
5.Enter your Nan	ne and click Continue.	
5.Enter your Nan <i>ARazorpay</i>	ne and click Continue.	
5.Enter your Nan	ne and click Continue.	
5.Enter your Nan	ne and click Continue.  Contact Details What's your name? Gaurav Kumar	
5.Enter your Nan	Ane and click Continue.	
5.Enter your Nan	ne and click Continue.  Contact Details What's your name? Gaurav Kumar	
5.Enter your Nan	ne and click Continue. Contact Details What's your name? Gaurav Kumar	
5.Enter your Nan	Are and click Continue.	
5.Enter your Nan	to contact Details Contact Details Gaurav Kumar	
5.Enter your Nan	ne and click Continue. Contact Details What's your name? Gaurav Kumar	Putrazu Merchant agreement Terms of use - Privacy Policy - Support





### **Platform Details**

Tell us more about where you want to accept payments.

1.Select where you want to accept payments from the given options. You can also select multiple options if relevant.

2.In case you select Others as an option, add a description.Click Continue.

< Platform Details	
Where do you want to accept payments?	
Vebsite	
Android app	
iOS app	
Social Media (WhatsApp, Facebook, Instagram)	
Offline store	
C Others	
Continue	

3.If you select Website/App, add your website/app link and click Continue or click Add Later to add the details afterwards.

	research park
A Razorpay	
	Add your website link     This is the website where you would like to accept payments
	https://acmecorp.com
	Add Later Continue
	Terms of use - Privacy policy - Support

# **Business Type and PAN Details**

Provide the following business and PAN details:

1.Select your business type from the list. Know more about different business types.

2.Click Continue.

	resea	tm, arch Dark
Razorpay	<ul> <li>Business Type</li> <li>What's your business type?</li> <li>Pick only one that applies to your business</li> <li>NOT REGISTERED ①</li> </ul>	
	Individual Small business REGISTERED © Proprietorship Partnership LLP Private limited Public limited Trust Society NGO HUF	
3.Enter your PA	Continue N/Business PAN and click Contin	e 2017-2022 · Merchant agreement · Terms of use · Privacy policy · Support
verify the details	s with the Central PAN database.	
	PAN Details           What's your PAN number?           We require this to verify your identity	

4.Confirm the name associated with the PAN by clicking Yes, Confirm. To edit the PAN details, click Edit PAN.

© 2017-2022 · Merchan Terms of use · Privacy p



6.Select the relevant Business Category from the list. For example, retail store, online store/marketplace, government and so on.



 Business Category	
What specific category under "Online store / Marketplace" ?	
Ecommerce Fashion and Lifestyle Grocery	
General merchandise stores Others	
	© 2017-2022 - Merchant agreement Terms of use - Privacy policy - Suppo

8.Specify the category under the subcategory selected in the previous step. In case you selected Ecommerce, then specify the category under it from the list.









### 3.Test Mode

Once your account is created, you have access to the Test mode on the Dashboard. Test mode is used for testing purposes and does not involve actual money transactions. However, you would need to activate your account in order to accept live payments.

Payments Refunds Batcl	n Refunds Orders Disputes		<ul><li>Test Mode</li><li>Live Mode</li></ul>	
Payments Refunds Batcl	n Refunds Orders Disputes		Live Mode	
You are in Test Mode, so only test data is	shown. Activate your account to start making live transaction			
		15.		
Payment Id	Duration	Status		
	Past 7 Days - 03 Mar 2022 to 10 Mar	ar 2022 All	~	
Email	Notes Count			
	25	Search Clear		

### **4.API Keys**

API key is a combination of the key\_id and key\_secret and is required to make any API request to Razorpay. You also have to implement the API keyin your code as part of your integration process.

#### **5.Generate API Keys**

- Log into your Dashboard with appropriate credentials.
- Select the Test mode for which you want to generate the APIkey.
- **Test Mode**: The test mode is a simulation mode that you can use to test your integration flow. Your customers will not be able to make payments inthis mode.





 Navigate to Settings → API Keys → Generate Key to generate key for theselected mode.

Once generated, you will be able to see the Key Id, the date the key wascreated and the expiry date for the API Key on screen.

Configuration V	Vebhooks AF	PI Keys	Reminders			
You are in <b>Test Mode</b> , so	only test data is sho	wn. <u>Activate</u>	e your account to start making live transactions.			
Your Feedback M	latters H	lello, Deve Click here	eloper! Would you like to take a few second	ds to help us improve yo	our experience with Razorpay? ●	×
Key Id			Created At	Expiry	Action	
	Delete		E-6 1846 2022 12:27.42 DM	Navan		

### // Write the following code Inside lib/Screen/payment/payment.dart //

```
var options = {
    "key": "rzp_test_4B5CoaTyxFQh3I", // generate key from razorpay website //
    "amount": payment_price * 100, // payment price value get from order summary page //
    "name": "INdigrain",
    "description": "payment for the product",
    "prefill": {"contact": "${Stringvalue.mobilenumber}", "email": "${Stringvalue.email}"},
    };
    try {
        razorpay.open(options);
        } catch (e) {
            print(e.toString());
        }
    }
```

#### Replace generated key from Razorpay.

#### key: < your key >

```
example: rzp_test_vLqPyNVpDeLzJg
```

Pass the Checkout options. Ensure that you pass the order\_id that you received

in the response to the previous step.





### **Running the application**

After Completed all the code and Run the Main.dart File



### Install App in Mobile







### **FINAL OUTPUT:**

### **CUSTOMER LOGIN**

### **CUSTOMER REGISTRATION**







### PROFILE

	My Pro		
	in yr ro	inc.	
Name			
Moni	ca Dhamodh	aran	
Email i	d		
moni	dhamoshant	hi@gma	il.con
Passw	ord		
1234	56		
Phone	No		
9003	135888		
			10/10
Doorno	/Flatno		
129/9	99		
Street	name		
Perar	nbur High Ro	ad	
City			
Peran	nbur Chenna		
State -			
Tamil	nadu		
		-	4

### **CART SCREEN**



#### **PRODUCT LIST**



#### ORDERSUMMARY

۲

-



Total Price 100.0 Pay Now

•





#### ORDERMANAGEMENT







### **CATEGORY LIST**

4:14 PI	м   0.1KB/s づ 〜 Ө	* .atl *
÷		
-	millet	

### **CATEGORY DETAIL**



### PRODUCTLIST



(+

•

•

### PRODUCTDETAILS

۲

-



Delete

۲

•





IITM Incubation Cell

### **EDIT PRODUCT**

0:43 AM   0.0KB/s ඊ ⊻	Œ \$ lin. \$
Edit Product	
Product Name nillet biscuits	
Product Category	
Product Description	
Product Price	
Product Quantity	
Save	

### **MYORDER**

10:43 AM   0.5KB/s 🖲	ð ¥	\$ .atl ⊗ (		
Order Details				
Orderid:11				
Order Date & Time	e:25 Januar	y 2023 10:42:37 IST		
Name:Monica Dha	amodharan			
Email id:monidhar	moshanthi@	gmail.com		
Phone No:900313	15888			
Address:129/99,P Chennai, Tamilnad	'erambur Hi du	gh Road,Perambur		
PinCode:600011				
Total Price:100.0				
Product Details				
	ł			
Product Name:mi	illet biscuits			
Price ₹:100 Product Quantity:	:1			

۲ •

## SAMPLE OUTPUT PAYMENT

۲

.

2:29 PM   55.0KB/s 足 🗇 🛛 🔧 🔐	13 38	12:29 PM   21.8KB/s	5 A O		12:29 PM   21.4KB/s	₹
I INdigrain	à∽ ×	← I INdig	Jrain	€.~ ×		
referred Payment Methods		Select Bank			1	
🔰 UPI - PhonePe			Axis	📀 Kotak	Welcome to Razorpay So	ftware Private Ltd Bank
with UPI - PayTM			-	-	This is just a der	no bank page.
ards, UPI & More		Yes	()) IDBI	BOB	You can choose whether to ma or ne	ike this payment successful ot:
Card Visa, MasterCard, RuPay, and Maestro		Select a differe	nt bank	-	Success	Failure
UPI Pay with installed and, or use others		AND DATIK				
Coogle PhonePe PayTim PayTim	Others					
Netbanking All Indian banks						
Wallet Mobikwik & More						
EMI EMI via ZestMoney		Account A	Sec	ured by <b>ARazorpay</b>		
Pay Later Simpl, LazyPay, ICICI & More		₹ 100 View Details	Pay N	low		
100 Pay Now			۲	•	- 6	
rew Details						





# <u>Part- B Build the Backend and Database for E-marketplace mobile</u> <u>application.</u>

### Building the backend of E-marketplace using Spring Boot framework.

#### Install Java SE 13 (JDK)

**Note**: Although newer versions of the JDK are available, NetBeans requires a file included in versions 13 and earlier for the installation.

1. Follow this link to download Java SE 13:

https://www.oracle.com/java/technologies/javase/jdk13-archive-downloads.html

2. Select the Windows x64 Installer option for JDK 13.0.2 (scroll down the page to reach this spot). Click the link on the right side of this option to download it.

**Note**: You may need to create an Oracle User account to download this software. If so, you can use your college email account and address when setting up your account:

3. After downloading, double-click the downloaded file (likely in your Downloads folder) and follow the installation instructions. Leave default settings from the installer as they are.

4. Now, let us set the JDK path.

Now, we will see how to set Java JDK Path (Environment Variable).

At first, copy the path wherein you installed the Java JDK. Let us copy it first. Remember, we need to copy the bin path i.e. the following on our system:

C:\**Program** Files\Java\jdk-13\bin

Here's the screenshot of the "bin" path, wherein we installed Java 13:





📕 🛛 🔄 🖛 🛛 bin					
File Home Share	View				
← → ~ ↑ 📙 C:\P	rogram Files\Java\jdk-13\bin				
	Name	Date modified	Type	Size	
🖈 Quick access	sen/er	10/6/2019 11-16 PM	File folder		
lene One Drive	AmitThinks.java	10/6/2019 11:25 PM	JAVA File	1 KB	
This DC	api-ms-win-core-console-I1-1-0.dll	10/6/2019 11:16 PM	Application extens	21 KB	
	api-ms-win-core-console-I1-2-0.dll	10/6/2019 11:16 PM	Application extens	21 KB	
3D Objects	🗟 api-ms-win-core-datetime-I1-1-0.dll	10/6/2019 11:16 PM	Application extens	21 KB	
E Desktop	🗟 api-ms-win-core-debug-l1-1-0.dll	10/6/2019 11:16 PM	Application extens	21 KB	
🔮 Documents	🗟 api-ms-win-core-errorhandling-11-1-0.dll	10/6/2019 11:16 PM	Application extens	21 KB	
🕂 Downloads	🗟 api-ms-win-core-file-I1-1-0.dll	10/6/2019 11:16 PM	Application extens	24 KB	
h Music	api-ms-win-core-file-I1-2-0.dll	10/6/2019 11:16 PM	Application extens	21 KB	
Pictures	🗟 api-ms-win-core-file-I2-1-0.dll	10/6/2019 11:16 PM	Application extens	21 KB	
Videos	🗟 api-ms-win-core-handle-I1-1-0.dll	10/6/2019 11:16 PM	Application extens	21 KB	
1 0 (C)	🗟 api-ms-win-core-heap-I1-1-0.dll	10/6/2019 11:16 PM	Application extens	21 KB	
	🗟 api-ms-win-core-interlocked-I1-1-0.dll	10/6/2019 11:16 PM	Application extens	21 KB	
🚔 New Volume (E:)	api-ms-win-core-libraryloader-I1-1-0.dll	10/6/2019 11:16 PM	Application extens	22 KB	

5. To set JDK Path, the easiest way is to type "Environment Variables" on Start. On typing, the following would be visible:



6.

Now, click on "Edit Environment Variables" and a new dialog box would be visible:





Computer Name       Hardware       Advanced       System Protection       Remote         You must be logged on as an Administrator to make most of these changes.       Performance         Visual effects, processor scheduling, memory usage, and vitual memory       Settings         User Profiles       Desktop settings related to your sign-in         Startup and Recovery       System startup, system failure, and debugging information         Statup       Settings         Device       Settings	System Properti	es				$\times$
You must be logged on as an Administrator to make most of these changes.         Performance         Visual effects, processor scheduling, memory usage, and vitual memory         Settings         User Profiles         Desktop settings related to your sign-in         Startup and Recovery         System startup, system failure, and debugging information         Environment Variables         OK       Cancel	Computer Name	Hardware	Advanced	System Protection	Remote	
Performance         Visual effects, processor scheduling, memory usage, and virtual memory         Settings         User Profiles         Desktop settings related to your sign-in         Startup and Recovery         System startup, system failure, and debugging information         Settings         Environment Variables         OK       Cancel	You must be le	ogged on as a	an Administra	tor to make most of t	hese changes.	
Settings         User Profiles         Desktop settings related to your sign-in         Startup and Recovery         System startup, system failure, and debugging information         Settings         Environment Variables         OK       Cancel	Performance     Visual effects	. processor s	chedulina. m	emory usage, and vir	tual memory	
User Profiles       Desktop settings related to your sign-in       Startup and Recovery       System startup, system failure, and debugging information       Settings       Environment Variables       OK     Cancel						
User Profiles Desktop settings related to your sign-in Settings Startup and Recovery System startup, system failure, and debugging information Settings Environment Variables OK Cancel Apply				L	Settings	
Desktop settings related to your sign-in         Settings         Startup and Recovery         System startup, system failure, and debugging information         Settings         Environment Variables         OK       Cancel	User Profiles					
Settings Startup and Recovery System startup, system failure, and debugging information Settings Environment Variables OK Cancel Apply	Desktop sett	ngs related to	o your sign-in			
Startup and Recovery System startup, system failure, and debugging information Settings Environment Variables OK Cancel Apply					Settings	
System startup, system failure, and debugging information Settings Environment Variables OK Cancel Apply	Startup and F	Recovery				
Settings Environment Variables OK Cancel Apply	System startu	ıp, system fail	ure, and deb	ugging information		
Environment Variables OK Cancel Apply					Settings	
OK Cancel Apply				<b>F</b> errieren	at Veriables	
OK Cancel Apply				Environme	ent vanables	
OK Cancel Apply						
			OK	Cancel	Apply	

7. Now, click "Environment Variable" and a new dialog box will open. Go to "User Variables" section.

ironment variables		
ser variables for amit_		
Variable	Value	Γ
OneDrive	C:\Users\amit_\OneDrive	
OneDriveConsumer	C:\Users\amit_\OneDrive	
Path	C:\Users\amit_\AppData\Local\Microsoft\WindowsApps;"C:\Users\	
TEMP	C:\Users\amit_\AppData\Local\Temp	
тмр	C:\Users\amit_\AppData\Local\Temp	
	New 5da Delate	
	ivew Edit Derete	
ystem variables	New Cuit Denete	
ystem variables Variable	Value	^
ystem variables Variable NUMBER_OF_PROCESSORS	Value	^
vstem variables Variable NUMBER_OF_PROCESSORS OS	Value 4 Windows_NT	^
vstem variables Variable NUMBER_OF_PROCESSORS OS Path	Value 4 Windows_NT C\Program Files (x86)\ImageMagick-7.0.8-Q16;C\Program Files\I	^
vstem variables Variable NUMBER_OF_PROCESSORS OS Path PATHEXT	Value 4 Windows_NT CC\Program Files (x86)\ImageMagick-7.0.8-Q16;C:\Program Files\I .COM;.EXE;.BAT;.CMD;.VBS;.VBE;JS;JSE;.WSF;.WSF;.MSC	^
vstem variables Variable NUMBER_OF_PROCESSORS OS Path PATHEXT PROCESSOR_ARCHITECTURE	Value 4 Windows_NT C:\Program Files (x86)\ImageMagick-7.0.8-Q16;C:\Program Files\I C:QOM;EXE;BAT;.CMD;.VBS;.VBE;JS;JSE;WSF;WSH;.MSC AMD64	^
vatem variables Variable NUMBER_OF_PROCESSORS OS Path PATHEXT PROCESSOR_ARCHITECTURE PROCESSOR_IDENTIFIER	Value 4 Windows_NT C:\Program Files (x86)\ImageMagick-7.0.8-Q16;C:\Program Files\I .COM;.EXE;.BAT;.CMD;.VBS;.VBE;JS;JSE;.WSF;.WSH;.MSC AMD64 Intel64 Family 6 Model 78 Stepping 3, GenuineIntel	^
vstem variables Variable NUMBER_OF_PROCESSORS OS Path PATHEXT PROCESSOR_ARCHITECTURE PROCESSOR_IDENTIFIER PROCESSOR_LEVEL	Value 4 Windows_NT C:\Program Files (x86)\ImageMagick-7.0.8-Q16;C:\Program Files\I COM; EXE; BAT;.CMD;.VBS;.VBE; JS;JSE;.WSF;.WSH;.MSC AMD64 Intel64 Family 6 Model 78 Stepping 3, GenuineIntel 6	^ ~

8. Click "New". Type PATH in the Variable name and add the Java JDK path "C:\Program Files\Java\jdk-13\bin" as displayed in the below screenshot:





ent variable	×
n Files (x80) (Imageiviagick-7.0.8-Q10	
ent <sup>e</sup> () austern <sup>22</sup>	sers\
oot%/Surtem22\W/bem	
Browse	
n Files (v86)\GthSharn\2.12\bin	
ROOT%\System32\OpenSSH\	Dalata.
anache\bin	Jelete
n Files (x86)\Intel\Intel(R) Management Engine Component Move Up	
n Files\Intel\Intel(R) Management Engine Components\DAI	
n Files (x86)\Brackets\command Move Down	1 /
mit \Downloads\poi-bin-4.1.0-20190412\poi-4.1.0\poi-4.1.0	_
mit \Downloads\poi-bin-4.1.0-20190412\poi-4.1.0\poi-exa	
mit \Downloads\poi-bin-4.1.0-20190412\poi-4.1.0\poi-exce Edit text	c\1
mit \Downloads\poi-bin-4.1.0-20190412\poi-4.1.0\poi-oox	
mit \Downloads\poi-bin-4.1.0-20190412\poi-4.1.0\poi-oox	
mit_\Downloads\poi-bin-4.1.0-20190412\poi-4.1.0\poi-scrat	
m Files\Java\jdk-13\bin	
	Delete
OK Cancel	
OK	

Above, press Ok.

9. Follow a similar process to set System Variables.

10. Now, we will verify the JDK installation.

Now, we can easily verify java installation was successfully or not using the following command on command prompt:

java -version

#### **Install Apache NetBeans IDE**

Note: Don't run the Apache NetBeans installer before Java is installed on your system.

1. Open the web page <u>https://netbeans.apache.org/download/</u>.

Go to the NetBeans 17 download page by clicking one of the Download buttons.

2. In the next page, make sure to download the Windows 64-bit version of NetBeans.

3. Now go to your Downloads folder (or wherever you had NetBeans downloaded to) and double-click the NetBeans installer file to run it (Apache-NetBeans-17-bin-windows-x64.exe). Click the Next button on the NetBeans 17 installer window.

4. In the License Agreement window, click the checkbox to accept the terms. Then click Next.

5. In the next window, under JDK for the Apache NetBeans IDE, make sure that the





location of the correct JDK has been chosen. You may have multiple versions of JDK on your computer. The version you installed in Step 1 should be specified here (change to the right one if it says different).

6. Click Install in the next window.

Installation may take a few minutes. After it's done, click the Finish button.

A Java Spring project requires a set of libraries and packages that enable the requested features. For our project, we select Maven as the project management tool. Maven helps to build and manage your Java project. It creates a so-called POM (Project-Object-Model) with all the information and configuration details of the project, which is saved in a pom.xml file.

### **Importing the Project**

1.

File Edit View Navigate Source Profile Team Tools Window Help Project...
New File... Ctrl+Shift+N Ctrl+N 🕒 · T 🍞 🕨 · 65 🌇 · 🛞 · 🗄 🚥 🚱 🖓 Start Page 🛛 🗟 SimpleController.java **NetBeans IDE** Learn & Discover My NetBeans Show On Startup 🗸 Close All Projects Open File Open Recent File **My NetBeans** Project Groups.. Recent Projects Install Pluging Import Project Export Project ort for other languages and gies by installing plugins from the Save Save As.. Save All Page Setup. 1 Exit Output × 🗇 4 Notifications Inspector 🔍 Search Results

Open Apache NetBeans, select File > Open Project

2. Unzip the emarketplace-Copy.zip folder and select the unzip folder containing the Maven project you want to import.

E-Marketplace





Projects × Services	Start Page	× 🗟 SimpleController.java ×			$\leftrightarrow \lor \iota$
<ul> <li>Source Packages</li> <li>Source Packages</li> <li>Test Packages</li> <li>Test</li></ul>	C Open Proje	Apache t Look Ix Build an E-Marketplace mobile application enantet enantetplace_indigonin_app	Learn & Disconer Mv Mel Beans.	What's Ne X	Show On Startup 2
	Network	File Name C:US:ers18P:Downloads1Build an E-Marke Files of Type Project Folder	place mobile application\emarket	Open Project	e project located in the selected folder.

Click Open Project to complete the process.

3. The directory structure of the spring boot project will look like this.







#### To configure your project to use JDK 13:

1. Right-click the project in the Projects window and select Properties from the context menu.

2. In the Project Properties dialog box, choose Build -> Compile and set JDK 13 as the Java Platform



3. Click OK to save the changes.

Create POJOs (plain old Java object) for Merchant, Customer, Category, Product, Order, and Orderproducts.

1. Customer.java

In the Projects window, Inside project file > source packages > com.spring.app.model. Open Customer.java file and write the following code.

Inside Customer class, Create private fields with their data types for id, name, email, phone, password, addressno, area, city, state, and pincode.

private String id; private String email; private String password; private String phone;





private String name;

private String addressno;

private String area;

private String city;

private String state;

private String pincode;

2. Create an empty constructor (Hibernate, which handles the JPA requires an empty constructor). Write the following code.

public Customer() {}

3. Create a constructor with the arguments id, name, email, phone, password, addressno, area, city, state, and pincode. Write the following code.

public Customer(String email, String password, String phone, String name, String id, String addressno, String area, String city, String state, String pincode) {

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

this.email = email;

this.password = password;

this.name = name;

this.phone = phone;

this.addressno = addressno;

this.area = area;

this.city = city;

this.state = state;

this.pincode = pincode;



}



4. Create accessor methods (i.e., getter and setter methods) for this field.

The IDE can create accessor methods for you. In the editor, right-click on `value` and choose Insert Code (or press Alt-Insert). In the popup menu, choose Getter and Setter.

File Edit View Navigate Source Refactor	er Run Debug Profile Te	am Tools Window Help	emarket - Apache NetBeans IDE 13	Search (Ctrl+I)	- 🗆 ×
: 🔭 🚰 📑 🔩 : 🆻 🍼 : < default	config> 🗸 🍑 🖓	👸 🕨 · 🐔 🌇 · 🕦 · 🛛 🏜	📾 омв 🕼 🎧		
Projects × Services Favorites Files	Start Page × 🗟 Custom	• • • • • • • • • • • • • • • • • • •			$\langle \rangle \vee 0$
> com.spring.app.service	Source History	Generate Getters and Setters	×		
> in Test Packages	Jource Tristory				
> Cher Sources	38 this.er	Select fields to generate getters and setters for:			
> E Dependencies	39 this.pa	Customer			
> 📭 Runtime Dependencies	40 this.nd	🗹 🖤 addressno : String			
> 🐚 Java Dependencies	41 this.pi	🗹 🖳 area : String			
> Roject Files	42 this at	🗹 🖤 city : String			
> 🕘 app	44 this.cl	🗹 🗐 email : String			
> 🕘 emarket	45 this.st	🗹 💁 id : String			
🗸 🟐 emarket	46 this.p:	🗹 🗐 name : String			
V 🗿 Source Packages	47 }	🗹 🖤 password : String			
w B com.spring.app	48	🗹 🖳 phone : String			
SampleApplication.java	49	🗹 👰 pincode : String			
SimpleController.java	50	🗹 🖤 state : String			
UserMgtController.java	51 }				
Com.spring.app.model	52				
Category.java					
Customer.java					
Merchant.java					
Order.java					-
Orderproducts.java					
Product.java					
> 💼 com.spring.app.service					
Test Packages					
V In Other Sources	Output × Search Results				
✓ asrc/main/resources		Select All Select None			
default package>					
> 📓 application.properties			Encapsulate Fields		
> lig Dependencies					
Runtime Dependencies		l	Generate Cancel		
> 🐚 Java Dependencies					
✓ Ing Project Files					
pom.xml					
nb-configuration.xml					
I (1) Notifications					49:1 INS
🗯 💽 💼 🦉 🗮	强 🤹 🔍 🖷				∧ ENG 05:16 PM 28-02-2023 □

In the dialog that displays, select all the fields, then click Generate. The getValue() and setValue() methods are added to the Customer class.

#### 5. Merchant.java

In the Projects window, Inside project file > source packages > com.spring.app.model. Open

Merchant.java file and write the following code.

Inside Merchant class, Create private fields with their data types for id, name, email, phone, gstno, and password.

private String id;

private String email;

private String password;

private String phone;

private String name;

private String gstno;

6. Create an empty constructor (Hibernate, which handles the JPA requires an empty





constructor). Write the following code.

```
public Merchant() {}
```

7. Create a constructor with the arguments id, name, email, phone, gstno, and password. Write the following code.

public Merchant(String email, String password, String phone, String name, String
gstno, String id) {

this.email = email;

this.password = password;

this.phone = phone;

this.name = name;

this.gstno = gstno;

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

}

8. Create accessor methods (i.e., getter and setter methods) for this field.

The IDE can create accessor methods for you. In the editor, right-click on `value` and choose Insert Code (or press Alt-Insert). In the popup menu, choose Getter and Setter.

<ul> <li>Q File Edit View Navigate Source Refacto</li> <li>I Image: Im</li></ul>	r Run Debug Profile T config> 🗸 🌑 - 🚏	eam Tools Window Help emarket - Ap	ache NetBeans IDE 13 Q• Search (Ctrl+I)	- 0 ×
Projects × Services Favorites Files	Start Page × 🗟 Custom Source History 🚺	Generate Getters and Setters     Select fields to generate getters and setters for:	×	< > < 0 8
Supersonation Dependencies     Supersonation Dependencies     Supersonation Dependencies     Supersonation     Sup	0 0 [ ] 3 1 3 2 □ public Merc 3 3 □ public Merc 3 4 chis.s. 3 5 chis.s. 3 5 chis.s. 3 7 chis.s. 3 9 chis.s. 4 1 4 2 } 4 3	✓     ○     Metchant       ✓     ○     Barten: String       ✓     ○     Barten: String       ✓     ○     Barten: String       ✓     ●     Barten: String	<pre>&gt;, String gstno, String id) {</pre>	E -
Categoryjska Categoryjska Categoryjska Categoryk Merchantjava Crderjava Crderjava Productjava Productjava Productjava Text Packages Text Packages Categoryk	Output × Search Results	Select All Select None		-
Summarket@PC     S		Cenente	opulate Fields Cancel	41:4 INS

In the dialog that displays, select all the fields, then click Generate. The getValue() and setValue() methods are added to the Merchant class.




## 9. Category.java

In the Projects window, Inside project file > source packages > com.spring.app.model. Open Category.java file and write the following code.

Inside Category class, Create private fields with their data types for id, name, and image.

private String id;

private String image;

private String name;

10. Create an empty constructor (Hibernate, which handles the JPA requires an empty constructor). Write the following code.

```
public Category() {}
```

11. Create a constructor with the arguments name, image, and id. Write the following code.

public Category(String name, String image, String id) {

```
this.name = name;
this.image = image;
this.id = id;
```

12. Create accessor methods (i.e., getter and setter methods) for this field.

The IDE can create accessor methods for you. In the editor, right-click on `value` and choose Insert Code (or press Alt-Insert). In the popup menu, choose Getter and Setter.

}







In the dialog that displays, select all the fields, then click Generate. The getValue() and setValue() methods are added to the Category class.

#### 13. Product.java

In the Projects window, Inside project file > source packages > com.spring.app.model. Open Product.java file and write the following code.

Inside Product class, Create private fields with their data types for id, name, description, price, category, quantity, initial quantity and image.

private String name; private String id; private String description; private String image; private String price; private String category; private String quantity; private String initialquantity ="1"; Create an empty constructor (Hibernate, which handles the JPA requires an empty

14.





constructor). Write the following code.

```
public Product() {}
```

15. Create a constructor with the arguments id, name, description, price, category, quantity, initial quantity and image. Write the following code.

public Product(String name, String description, String image, String price, String category, String quantity, String initialquantity, String id) {

```
this.id = id;
this.name = name;
this.description = description;
this.image = image;
this.price = price;
this.category = category;
this.quantity = quantity;
this.initialquantity = initialquantity;
```

}

16. Create accessor methods (i.e., getter and setter methods) for this field.

The IDE can create accessor methods for you. In the editor, right-click on `value` and choose Insert Code (or press Alt-Insert). In the popup menu, choose Getter and Setter.







In the dialog that displays, select all the fields, then click Generate. The getValue() and setValue() methods are added to the Product class.

#### 17. Order.java

In the Projects window, Inside project file > source packages > com.spring.app.model. Open Order.java file and write the following code.

Inside Order class, Create private fields with their data types for customerid, customername, customeremail, customerphoneno, customeraddressno, customerarea, customercity, customerstate, customerpincode, totalprice, ordereddatetime, orderid, orderrefid, and productlist.

private String totalprice; private String customerid; private String customername; private String customeremail; private String customerphoneno; private String customeraddressno; private String customerarea; private String customercity;





private String customerpincode;
private String ordereddatetime;
private int orderid;

private int orderrefid;

private List<Orderproducts> productlist;

18. Create an empty constructor (Hibernate, which handles the JPA requires an empty constructor). Write the following code.

public Order() {}

19. Create a constructor with the arguments customerid, customername, customeremail, customerphoneno, customeraddressno, customerarea, customercity, customerstate, customerpincode, totalprice, ordereddatetime, orderid, orderrefid. Write the following code.

public Order(String totalprice, String customerid, String customername, String customeremail, String customerphoneno, String customeraddressno, String customerarea, String customercity, String customerstate, String customerpincode, String ordereddatetime, int orderid, int orderrefid) {

this.totalprice = totalprice;

this.customerid = customerid;

this.customername = customername;

this.customeremail = customeremail;

this.customerphoneno = customerphoneno;

this.customeraddressno = customeraddressno;

this.customerarea = customerarea;

this.customercity = customercity;





<pre>this.customerstate = customerstate;</pre>
<pre>this.customerpincode = customerpincode;</pre>
this.ordereddatetime = ordereddatetime;
this.orderid = orderid;
this.orderrefid = orderrefid;

# }

20. Create accessor methods (i.e., getter and setter methods) for this field.

The IDE can create accessor methods for you. In the editor, right-click on `value` and choose Insert Code (or press Alt-Insert). In the popup menu, choose Getter and Setter.

Project: X Service Faces Face          Y       Control       Start Page       Cutom       Cutom       Cutom       Control       Control </th <th>×</th>	×
Projects × Services       Fields       Star Page × @ Cutom         V : cidedup paper       Secret Fields       Secret Fields         > Manufactures       Secret Fields       Secret Fields	
Sected Hatop     Sected Hatop     Sected Hatop     Sected Hatop     Sected Hatop       Sected Hatop     Sected Hatop     Sected Hatop     Sected Hatop     Sected Hatop       Sected Hatop     Sected Hatop     Sected Hatop     Sected Hatop     Sected Hatop       Sected Hatop     Sected Hatop     Sected Hatop     Sected Hatop     Sected Hatop       Sected Hatop     Sected Hatop     Sected Hatop     Sected Hatop     Sected Hatop       Sected Hatop     Sected Hatop     Sected Hatop     Sected Hatop     Sected Hatop       Sected Hatop     Sected Hatop     Sected Hatop     Sected Hatop     Sected Hatop       Sected Hatop     Sected Hatop     Sected Hatop     Sected Hatop     Sected Hatop       Sected Hatop     Sected Hatop     Sected Hatop     Sected Hatop     Sected Hatop       Sected Hatop     Sected Hatop     Sected Hatop     Sected Hatop     Sected Hatop       Sected Hatop     Sected Hatop     Sected Hatop     Sected Hatop     Sected Hatop       Sected Hatop     Sected Hatop     Sected Hatop     Sected Hatop     Sected Hatop       Sected Hatop     Sected Hatop     Sected Hatop     Sected Hatop     Sected Hatop       Sected Hatop     Sected Hatop     Sected Hatop     Sected Hatop     Sected Hatop <td< td=""><td>~ 0</td></td<>	~ 0
<pre>&gt; Conspin app Provide &gt; Cuttomergian &gt; Cuttome</pre>	88
> Dependencies       String customerphonence, String customerphonerphonence, String customerphonence, String customerphonence, Stri	
> Dependencies       61       ctilisto         > Multimo Dependencies       61       ctilisto         > Multimo Dependencies       63       ctilisto         > Multimo Dependencies       63       ctilisto         > Multimo Dependencies       63       ctilisto         > Multimo Dependencies       64       ctilisto         > Multimo Dependencies       65       ctilisto         > Multimo Dependencies       63       ctilisto         > Multimo Dependencies       64 <td>ra</td>	ra
> is Automic Lappendencies     is an Unit and States       > is Automic Lappendencies     is an Unit and States       > is Automic Lappendencies     is an Unit and States       > is Automic Lappendencies     is an Unit and States       > is Automic Lappendencies     is an Unit and States       > is an Unit and States     is an Unit and States       > is an Unit and States     is an Unit and States       > is an Unit and States     is an Unit and States       > is an Unit and States     is an Unit and States       > is an Unit and States     is an Unit and States       > is any States     is an Unit and States       > is any States     is any States	
* Ard Deprivations     ***     Constrained     Constrained       * Broket     ***     Cons	
Image: Second	
imb     configuration.xml     eff     this.c       v     imaxist     imaxist     imaxist       v     imaxist     eff     this.c       v     imaxist     imaxist     imaxist       v     imaxist     imaxist       v     imaxist	
• • • • • • • • • • • • • • • • • • •	
v ⊕ Source Packages     40     this:       v ⊕ Source Packages     40     this:       v ⊕ Source Packages     40     this:       v ⊕ Source Packages     60     this:       ⊕ UseMapController java     50     this:       ⊕ UseMapController java     50     this:       ⊕ Cutioner jin: Controller java     50     this:       ⊕ Order poducting in an operative is String     Image: String operative is	
Implementation     69     Ctills     Ctill	
SampleApplication.java       00       Lift.s.o         SampleApplication.java       51       tit.s.o         WendbyEcontoller.java       52       tit.s.o         Category.java       53       tit.s.o         Category.java       56       jordererddwith:         Category.java       57       jordererddwith:         Category.java       53       jordererdwith:         Category.java       53       jordererdwith:         Category.java       53       jordererdwith:         Category.java       53       jordererdwith:         Category.java       53       jordererdwith: <tr< td=""><td></td></tr<>	
ImpleController/ava     53     ctilsc       ImpleController/ava     53     ctilsc       ImpleController/ava     54     ctilsc       ImpleController/ava     53     ctilsc       ImpleController/ava     54     ctilsc       ImpleController/ava     55     controller/ava       ImpleController/ava     56     controller/ava       ImpleController/ava     56     controller/ava       ImpleController/ava     56     controller/ava       ImpleController/ava     57     controller/ava       ImpleController/ava     50     controller/ava       ImpleControler/ava     50<	
Identifying approximation of the second o	-
✓     Conspring app provide     64	
Category ava     55	
B     Cutatomergiva     66       B     Mechantyava     67       B     Ordregrava     59       B     Ordregrava     59       B     Productyava     59       Chelperoductyava     59       B     Productyava       B     Productyava       Chelperoductyava     59	
Image: Select All     Select None	
Image: Solution of the solut	
● Producijna         ●           > Bernsping apparvice         ●	-
> ■ comspring appsærvice        > ■ fet Packages     Output ×       > ● Other Sources     Output ×       > ● default package>     Select All	
> Im Tet Package         Image: Search Results            Image: Search Results           > Im Chroma/resources         Output × Search Results           > Bet: Main Search Results         Select All	
∨         ©         Output ×         Search Results           √         Bit or/main/resources         Select All         Select None	
< default package>	-
Mi application.properties     Encapsulate Helds	
> To Dependencies	
> Wuntime Upgendencies	
A leg Java Dependencies	
way Project mes	
G (Q) Notifications 56.2 INS	
🗄 💽 🖻 📲 💿 📕 💁 📫 🗿 🔨 https://www.second.com/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/second/secon	$\Box$

In the dialog that displays, select all the fields, then click Generate. The getValue() and setValue() methods are added to the Order class.

# 21. Orderproducts.java

In the Projects window, Inside project file > source packages > com.spring.app.model. Open Orderproducts.java file and write the following code.

Inside Orderproducts class, Create private fields with their data types for productname, productprice, productquantity, productquantity, productid, productimage, and productdescription.

private String productname;

private String productprice;





private String productquantity;

private String productid;

private String productimage;

private String productdescription;

22. Create an empty constructor (Hibernate, which handles the JPA requires an empty constructor). Write the following code.

```
public Orderproducts() {}
```

23. Create a constructor with the arguments productname, productprice, productquantity, productquantity, productid, productimage, and productdescription. Write the following code.

public Orderproducts(String productname, String productprice, String

productquantity, String productid, String productimage, String productdescription)

{

this.productname = productname;

```
this.productprice = productprice;
```

this.productquantity = productquantity;

this.productid = productid;

this.productimage = productimage;

this.productdescription = productdescription;

```
}
```

24. Create accessor methods (i.e., getter and setter methods) for this field.

The IDE can create accessor methods for you. In the editor, right-click on `value` and choose Insert Code (or press Alt-Insert). In the popup menu, choose Getter and Setter.







In the dialog that displays, select all the fields, then click Generate. The getValue() and setValue() methods are added to the Orderproducts class.

# Create Spring Boot API Controller for merchant and customer.

controller package is used to implement a Spring Boot RestAPI controller to handle all incoming requests (post/get/put/delete) and response to rest-client.

Create REST end points that performs the basic database operations such as Create, Read, Update, Delete

# Merchant

- Handling merchant login
- Manage Categories
  - o List category
  - Insert category
  - Delete category
- Manage Products
  - List product
  - Insert product
  - o Update product
  - o Delete product
- List Received Order

## Customer

- Handling customer login
- Handling customer register
- Manage Profile





- List Order
- List Categories
- List Products
- Order checkout
- Generate Invoice & send via email

# 1. Handling merchant login

This method is used to login as merchant.

In the Projects window, Inside project file > source packages > com.spring.app. Open SimpleController.java file and write the following code.

```
@CrossOrigin(origins = "*")
```

```
@RequestMapping(value = "/merchantlogin", method = RequestMethod.POST)
```

```
public String loginMerchant(@RequestBody Merchant merchant) {
```

```
String s = "select count(*) from merchant where memail=? AND mpassword=?";
```

```
System.out.println("s = " + merchant.getEmail());
```

```
System.out.println("s = " + merchant.getPassword());
```

```
System.out.println("s = " + s);
```

```
int count = jdbc.queryForObject(s, new Object[]{merchant.getEmail(),
merchant.getPassword()}, Integer.class);
```

```
System.out.println("count = " + count);
```

```
if (count > 0) {
```

return "Successfull";

} else {

E-Marketplace



}

}



return "Failure";

@RestController: This annotation marks the SimpleController as an HTTP request handler and allows Spring to recognize it as a RESTful service.

@RequestMapping("/merchantlogin") annotation sets the base path to the resource endpoints in the controller as /merchantlogin.

@RequestMapping(method = RequestMethod.POST) is used to map HTTP POST request to the mapped controller methods. We used it to send email and password of a merchant.

@RequestBody: This annotation takes care of binding the web request body to the method parameter with the help of the registered HttpMessageConverters. So, when you make a POST request to the "/merchantlogin" URL with a Post JSON body, the HttpMessageConverters converts the JSON request body into a Post object and passes it to the loginMerchant method.

Inside loginMerchant method is where you create the query to count data values from the merchant table.

The SQL SELECT statement can be used along with COUNT (\*) function to count of all rows present in the merchant table and SQL query that returns a value object like String then you can use the queryForObject() method of JdbcTempalte class. This method takes an argument about what type of class query will return and then convert the result into that object and returns it to the caller.

#### 2. List category

This method is used to display categories.

In the Projects window, Inside project file > source packages > com.spring.app. Open SimpleController.java file and write the following code.

@CrossOrigin(origins = "\*")

```
@RequestMapping(value = "/category", method = RequestMethod.GET)
```

public JSONObject category() {

```
String s = "select catcategoryname AS name, catcategoryimage AS image,
```

```
catid AS id from category";
```





return json;

```
}
```

**@RestController:** This annotation marks the SimpleController as an HTTP request handler and allows Spring to recognize it as a RESTful service.

@RequestMapping("/category") annotation sets the base path to the resource endpoints in the controller as /category.

@RequestMapping(method = RequestMethod.GET), and is used to map HTTP GET requests to the mapped controller methods. We used it to return all the categories.

Inside category method is where you create the query to return a list of categories from the category table.

The SQL string contains a query to select all the category details from the category table and if your SQL query is going to return a List of objects instead of just one object then you need to use the query () method of JdbcTempalte. These methods provide to convert the result to a custom object. For instance, the simplest way to query and handle results is via the query (String, RowMapper) method. This method uses RowMapper to map the returned row to an object.

IITM Incubati Cell

TION





# 3. Insert category

This method is used to insert categories.

In the Projects window, Inside project file > source packages > com.spring.app. Open SimpleController.java file and write the following code.

```
@CrossOrigin(origins = "*")
```

```
@RequestMapping(value = "/insertcategory", method = RequestMethod.POST)
```

public String insertCategory(@RequestBody Category category) {

```
String s = "insert into
```

category(catcategoryname,catcategoryimage)values(?,?)";

```
System.out.println("s = " + category.getName());
```

System.out.println("s = " + category.getImage());

```
System.out.println("s = " + s);
```

int a = jdbc.update(s, category.getName(), category.getImage());

```
System.out.println("a = " + a);
```

if (a == 1) {

return "Inserted Successfully";

} else {

return "Inserted failure";

}

}

<code>@RestController: This annotation marks the SimpleController as an HTTP request handler and allows Spring to recognize it as a RESTful service.</code>





@RequestMapping("/insertcategory") annotation sets the base path to the resource endpoints in the controller as /insertcategory.

@RequestMapping(method = RequestMethod.POST) is used to map HTTP POST request to the mapped controller methods. We used it to send name and image of a category.

@RequestBody: This annotation takes care of binding the web request body to the method parameter with the help of the registered HttpMessageConverters. So, when you make a POST request to the "/insertcategory" URL with a Post JSON body, the HttpMessageConverters converts the JSON request body into a Post object and passes it to the insertCategory method.

Inside insertCategory method is where you create the query to insert a category in the category table.

The update method provided by JdbcTemplate can be used for insert, update, and delete operations.

The SQL string is used to perform a single insert operation. Here '?' means it acts as the parameter which we need to pass while executing the query. Now to execute the query, we have used the JdbcTemplate update() method, which takes the query as an argument, and other than the query there are 2 values that correspond to 2 '?' respectively.

4. Delete category

This method is used to delete categories.

In the Projects window, Inside project file > source packages > com.spring.app. Open SimpleController.java file and write the following code.

```
@CrossOrigin(origins = "*")
```

```
@RequestMapping(value = "/deletecategory", method = RequestMethod.POST)
```

public String deleteCategory(@RequestBody Category category) {

```
String s = "delete from category where catid= ('" + category.getId() +
```

"')";

```
System.out.println("s = " + s);
```

```
int a = jdbc.update(s);
```

```
System.out.println("a = " + a);
```

if (a == 1) {

E-Marketplace



}



return "Deleted Successfully";
} else {
 return "Deleted Failure";
}

@RestController: This annotation marks the SimpleController as an HTTP request handler and allows Spring to recognize it as a RESTful service.

@RequestMapping("/deletecategory") annotation sets the base path to the resource endpoints in the controller as / deletecategory.

@RequestMapping(method = RequestMethod.POST is used to map HTTP POST request to the mapped controller methods. We used it to send id of a category.

@RequestBody: This annotation takes care of binding the web request body to the method parameter with the help of the registered HttpMessageConverters. So, when you make a POST request to the "/deletecategory" URL with a Post JSON body, the HttpMessageConverters converts the JSON request body into a Post object and passes it to the deleteCategory method.

Inside deleteCategory method is where you create the query to delete categories from the category table.

Create a SQL string to delete category by ID from category table. Call the update method of JdbcTemplate and pass the string to be bound to the query.

5. List product

This method is used to list products.

In the Projects window, Inside project file > source packages > com.spring.app. Open SimpleController.java file and write the following code.

@CrossOrigin(origins = "\*")

@RequestMapping(value = "/listproduct", method = RequestMethod.GET)

public JSONObject product() {





String s = "select proid AS id, proname AS name, proimage AS image, prodescription AS description, proprice AS price, procategory AS category, proquantity AS quantity, proinitialquantity AS intialquantity from product";

```
List<Product> mrlist = jdbc.query(s, new
BeanPropertyRowMapper(Product.class));
```

System.out.println("mrlist = " + mrlist);

JSONObject json = new JSONObject();

json.put("Product", mrlist);

```
if (!mrlist.isEmpty()) {
```

json.put("Product", mrlist);

System.out.println("json = " + json);

return json;

```
}
```

return json;

}

**@RestController:** This annotation marks the SimpleController as an HTTP request handler and allows Spring to recognize it as a RESTful service.

@RequestMapping("/listproduct") annotation sets the base path to the resource endpoints in the controller as /listproduct.

@RequestMapping (method = RequestMethod.GET) is used to map HTTP GET request to the mapped controller methods. We used it to return all the products.

Inside product method is where you create the query to return a list of products from the product table.

The SQL string contains a query to select all the product details from the product table and if your SQL query is going to return a List of objects instead of just one object then you need to use the





query () method of JdbcTempalte. These methods provide to convert the result to a custom object. For instance, the simplest way to query and handle results is via the query (String, RowMapper) method. This method uses RowMapper to map the returned row to an object.

6. Insert product

This method is used to insert products.

In the Projects window, Inside project file > source packages > com.spring.app. Open SimpleController.java file and write the following code.

@CrossOrigin(origins = "\*")

```
@RequestMapping(value = "/insertproduct", method = RequestMethod.POST)
```

public String insertProduct(@RequestBody Product product) {

String s = "insert into

product(proname, proimage, proprice, prodescription, procategory, proquantity, proinitia

lquantity)values(?,?,?,?,?,?,?)";

```
System.out.println("s = " + product.getName());
System.out.println("s = " + product.getImage());
System.out.println("s = " + product.getPrice());
System.out.println("s = " + product.getDescription());
System.out.println("s = " + product.getCategory());
System.out.println("s = " + product.getQuantity());
System.out.println("s = " + product.getInitialquantity());
```

System.out.println("s = " + s);





int a = jdbc.update(s, product.getName(), product.getImage(), product.getPrice(), product.getDescription(), product.getCategory(), product.getQuantity(), product.getInitialquantity());

```
System.out.println("a = " + a);
if (a == 1) {
    return "Inserted Successfully";
```

} else {

}

}

return "Inserted failure";

@RestController: This annotation marks the SimpleController as an HTTP request handler and allows Spring to recognize it as a RESTful service.

@RequestMapping("/insertproduct") annotation sets the base path to the resource endpoints in the controller as /insertproduct.

@RequestMapping(method = RequestMethod.POST) is used to map HTTP POST request to the mapped controller methods. We used it to send name, image, price, description, quantity, and initial quantity of a product.

@RequestBody: This annotation takes care of binding the web request body to the method parameter with the help of the registered HttpMessageConverters. So, when you make a POST request to the "/insertproduct" URL with a Post JSON body, the HttpMessageConverters converts the JSON request body into a Post object and passes it to the insertProduct method.

Inside insertProduct method is where you create the query to insert a product in the product table.

The update method provided by JdbcTemplate can be used for insert, update, and delete operations.

The SQL string is used to perform a single insert operation. Here '?' means it acts as the parameter which we need to pass while executing the query. Now to execute the query, we have used the JdbcTemplate update() method, which takes the query as an argument, and other than the query there are 7 values that correspond to 7 '?' respectively.





# 7. Update product

This method is used to update products.

In the Projects window, Inside project file > source packages > com.spring.app. Open SimpleController.java file and write the following code.

@CrossOrigin(origins = "\*")

@RequestMapping(value = "/updateproduct", method = RequestMethod.POST)

public String updateProduct(@RequestBody Product product) {

```
String s = "update product set proname= ?, prodescription= ?, procategory=
```

?, proprice= ?, proquantity= ? where proid=('" + product.getId() + "')";

System.out.println("s = " + product.getName());

System.out.println("s = " + product.getDescription());

System.out.println("s = " + product.getCategory());

System.out.println("s = " + product.getPrice());

System.out.println("s = " + product.getQuantity());

System.out.println("s = " + s);

```
int a = jdbc.update(s, product.getName(), product.getDescription(),
product.getCategory(), product.getPrice(), product.getQuantity());
```

System.out.println("a = " + a);

```
if (a == 1) {
```

return "Updated Successfully";

} else {

return "Updated Failure";

```
}
```

E-Marketplace



}





@RequestMapping("/uodateproduct") annotation sets the base path to the resource endpoints in the controller as /updateproduct.

@RequestMapping (method = RequestMethod.POST) is used to map HTTP POST request to the mapped controller methods. We used it to send name, image, price, description, quantity and initial quantity of the product.

@RequestBody: This annotation takes care of binding the web request body to the method parameter with the help of the registered HttpMessageConverters. So, when you make a POST request to the "/updateproduct" URL with a Post JSON body, the HttpMessageConverters converts the JSON request body into a Post object and passes it to the updateProduct method.

Inside updateProduct method is where you create the query to update a product in the product table.

The update method provided by JdbcTemplate can be used for insert, update, and delete operations.

The SQL string is used to update the product details by ID and pass the string to the update method of JdbcTemplate followed by object arguments of type string which are the name, description, quantity, price, and category. Note that the ID is only used to find the product to be updated but the ID itself is not updated.

## 8. Delete product

This method is used to delete products.

In the Projects window, Inside project file > source packages > com.spring.app. Open SimpleController.java file and write the following code.

```
@CrossOrigin(origins = "*")
```

@RequestMapping(value = "/deleteproduct", method = RequestMethod.POST)

public String deleteProduct(@RequestBody Product product) {

String s = "delete from product where proid= ('" + product.getId() + "')";

System.out.println("s = " + s);





int a = jdbc.update(s);

System.out.println("a = " + a);

if (a == 1) {

return "Deleted Successfully";

} else {

}

}

return "Deleted Failure";

@RestController: This annotation marks the SimpleController as an HTTP request handler and allows Spring to recognize it as a RESTful service.

@RequestMapping("/deleteproduct") annotation sets the base path to the resource endpoints in the controller as /deleteproduct.

@RequestMapping(method = RequestMethod.POST) is used to map HTTP POST request to the mapped controller methods. We used it to send id of a product.

@RequestBody: This annotation takes care of binding the web request body to the method parameter with the help of the registered HttpMessageConverters. So, when you make a POST request to the "/deleteproduct" URL with a Post JSON body, the HttpMessageConverters converts the JSON request body into a Post object and passes it to the deleteProduct method.

Inside deleteProduct method is where you create the query to delete product from the product table.

Create a SQL string to delete the products by ID from product table. Call the update method of JdbcTemplate and pass the string to be bound to the query.

## 9. List Received Order

This method is used to display received orders.

In the Projects window, Inside project file > source packages > com.spring.app. Open SimpleController.java file and write the following code.





@CrossOrigin(origins = "\*")

@RequestMapping(value = "/merchantorder", method = RequestMethod.POST)
public JSONObject order() {

String s = "select distinct osrefid AS orderrefid, max(oscustomername) AS customername, max(oscustomeremail) AS customeremail, max(oscustomerphone) AS customerphoneno, max(oscustomeraddressno) AS customeraddressno, max(oscustomerarea) AS customerarea, max(oscustomercity) AS customercity, max(oscustomerstate) AS customerstate, max(oscustomerpincode) AS customerpincode, max(ostotalprice) AS totalprice, max(osordereddatetime) AS ordereddatetime from ordersummary group by osrefid order by osrefid";

System.out.println("s = " + s);

List<Order> orderidList = jdbc.query(s, new BeanPropertyRowMapper(Order.class));

System.out.println("orderidList = " + orderidList.isEmpty());

JSONObject orderDetObj = new JSONObject();

JSONArray orderArr = new JSONArray();

JSONObject orderObj = new JSONObject();

if (!orderidList.isEmpty()) {

for (Order orObj : orderidList) {

orderDetObj.put("orderrefid", orObj.getOrderrefid()); orderDetObj.put("totalprice", orObj.getTotalprice()); orderDetObj.put("customername", orObj.getCustomername()); orderDetObj.put("customeremail", orObj.getCustomeremail());





orObj.getCustomeraddressno());

orderDetObj.put("customerarea", orObj.getCustomerarea()); orderDetObj.put("customercity", orObj.getCustomercity()); orderDetObj.put("customerstate", orObj.getCustomerstate()); orderDetObj.put("customerpincode", orObj.getCustomerpincode()); orderDetObj.put("ordereddatetime", orObj.getOrdereddatetime());

String t = "select osproductname AS productname, osproductprice AS productprice, osproductimage AS productimage, osproductquantity AS productquantity, osproductprice AS productprice from ordersummary where osrefid=" + orObj.getOrderrefid();

System.out.println("t = " + t);

List<Orderproducts> productlist = jdbc.query(t, new

BeanPropertyRowMapper(Orderproducts.class));

System.out.println("productlist = " + productlist.isEmpty());

System.out.println("productlist = " + productlist);

JSONArray pdlistArr = new JSONArray();

if (!productlist.isEmpty()) {

for (Orderproducts pdlist : productlist) {

JSONObject pdlisObj = new JSONObject();

E-Marketplace





```
pdlisObj.put("productname", pdlist.getProductname());
pdlisObj.put("productquantity",
```

```
pdlist.getProductquantity());
```

```
pdlisObj.put("productprice", pdlist.getProductprice());
pdlisObj.put("productimage", pdlist.getProductimage());
```

```
pdlistArr.add(pdlisObj);
```

```
}
```

orderDetObj.put("pdlist", pdlistArr);

orderArr.add(orderDetObj);

```
}
```

orderObj.put("orderdetails", orderArr);

```
}
```

}

return orderObj;

}

**@RestController:** This annotation marks the SimpleController as an HTTP request handler and allows Spring to recognize it as a RESTful service.

@RequestMapping("/merchantorder") annotation sets the base path to the resource endpoints in the controller as /merchantorder.

@RequestMapping(method = RequestMethod.POST) is used to map HTTP POST request to the mapped controller methods. We used it to send customer details, order details and product details.

Inside order method is where you create the query to return customer details, product details and order details as list from the ordersummary table.





The SQL s string contains a query to select the customer details, product details and order details from the ordersummary table and if your SQL query is going to return a List of objects instead of just one object then you need to use the query () method of JdbcTempalte. These methods provide to convert the result to a custom object. For instance, the simplest way to query and handle results is via the query (String, RowMapper) method. This method uses RowMapper to map the returned row to an object.

10. Handling customer login

This method is used to login as customer.

In the Projects window, Inside project file > source packages > com.spring.app. Open SimpleController.java file and write the following code.

```
@CrossOrigin(origins = "*")
```

@RequestMapping(value = "/customerlogin", method = RequestMethod.POST)

public JSONObject loginCustomer(@RequestBody Customer customer) {

String s = "select cemail AS email, cpassword AS password, cname AS name, cid AS id, cphone AS phone,caddressno AS addressno, carea AS area, ccity AS city, cstate AS state, cpincode AS pincode from customer where cemail=CAST('" + customer.getEmail() + "' AS VARCHAR) AND cpassword=CAST('" + customer.getPassword() + "' AS VARCHAR)";

System.out.println("s = " + customer.getEmail());

System.out.println("s = " + customer.getPassword());

System.out.println("s = " + s);

```
List<Customer> mrlist = jdbc.query(s, new
```

BeanPropertyRowMapper(Customer.class));

System.out.println("mrlist = " + mrlist);

JSONObject json = new JSONObject();

```
json.put("Customerdetails", mrlist);
```





json.put("Customerdetails", mrlist);

System.out.println("json = " + json);

return json;

}

return json;

}

**@RestController:** This annotation marks the SimpleController as an HTTP request handler and allows Spring to recognize it as a RESTful service.

@RequestMapping("/customerlogin") annotation sets the base path to the resource endpoints in the controller as /customerlogin.

@RequestMapping(method = RequestMethod.POST) is used to map HTTP POST request to the mapped controller methods. We used it to send email and password of a customer.

@RequestBody: This annotation takes care of binding the web request body to the method parameter with the help of the registered HttpMessageConverters. So, when you make a POST request to the "/customerlogin" URL with a Post JSON body, the HttpMessageConverters converts the JSON request body into a Post object and passes it to the loginCustomer method.

Inside loginCustomer method is where you create the query to return customer details as list from the customer table.

The SQL s string contains a query to select the customer ID by email and password from the customer table and if your SQL query is going to return a List of objects instead of just one object then you need to use the query () method of JdbcTempalte. These methods provide to convert the result to a custom object. For instance, the simplest way to query and handle results is via the query (String, RowMapper) method. This method uses RowMapper to map the returned row to an object.

## 11. Handling customer registration

This method is used to register as customer.

In the Projects window, Inside project file > source packages > com.spring.app. Open SimpleController.java file and write the following code.

IITM Incubat

UBATION





# @CrossOrigin(origins = "\*")

@RequestMapping(value = "/customerregister", method = RequestMethod.POST)

public String customerRegister(@RequestBody Customer customer) {

String s = "insert into

customer(cname,cemail,cpassword,cphone)values(?,?,?,?)";

System.out.println("s = " + customer.getEmail());

System.out.println("s = " + customer.getPassword());

System.out.println("s = " + customer.getName());

System.out.println("s = " + customer.getPhone());

System.out.println("s = " + s);

int a = jdbc.update(s, customer.getName(), customer.getEmail(), customer.getPassword(), customer.getPhone());

```
System.out.println("a = " + a);
```

if (a == 1) {

return "Registered Successfully";

} else {

}

}

return "Registeration failure";

**@RestController:** This annotation marks the SimpleController as an HTTP request handler and allows Spring to recognize it as a RESTful service.

@RequestMapping("/customerregister") annotation sets the base path to the resource





endpoints in the controller as /customerregister.

@RequestMapping(method = RequestMethod.POST) is used to map HTTP POST request to the mapped controller methods. We used it to send name, email ID, password, and phone No of a customer.

@RequestBody: This annotation takes care of binding the web request body to the method parameter with the help of the registered HttpMessageConverters. So, when you make a POST request to the "/customerregister" URL with a Post JSON body, the HttpMessageConverters converts the JSON request body into a Post object and passes it to the customerRegister method.

Inside customerRegister method is where you create the query to insert customer details in the customer table.

The update method provided by JdbcTemplate can be used for insert, update, and delete operations.

The SQL string is used to perform a single insert operation. Here '?' means it acts as the parameter which we need to pass while executing the query. Now to execute the query, we have used the JdbcTemplate update() method, which takes the query as an argument, and other than the query there are 4 values that correspond to 4 '?' respectively.

12. Manage profile

This method is used to update customer profile.

In the Projects window, Inside project file > source packages > com.spring.app. Open SimpleController.java file and write the following code.

```
@CrossOrigin(origins = "*")
```

```
@RequestMapping(value = "/updateprofile", method = RequestMethod.POST)
```

public String updateProfile(@RequestBody Customer customer) {

```
String s = "update customer set cemail= ?, cpassword= ?, cname= ?, cphone=
?, caddressno= ?, carea= ?, ccity= ?, cstate= ?, cpincode= ? where cid=('" +
```

```
customer.getId() + "')";
```

```
System.out.println("s = " + customer.getEmail());
```

```
System.out.println("s = " + customer.getPassword());
```





```
System.out.println("s = " + customer.getName());
System.out.println("s = " + customer.getPhone());
System.out.println("s = " + customer.getAddressno());
System.out.println("s = " + customer.getArea());
System.out.println("s = " + customer.getCity());
System.out.println("s = " + customer.getState());
System.out.println("s = " + customer.getPincode());
System.out.println("s = " + s);
```

```
int a = jdbc.update(s, customer.getEmail(), customer.getPassword(),
customer.getName(), customer.getPhone(), customer.getAddressno(),
customer.getArea(), customer.getCity(), customer.getState(),
customer.getPincode());
```

System.out.println("a = " + a);

if (a == 1) {

return "Updated Successfully";

} else {

return "Updated Failure";

}

}

**@RestController:** This annotation marks the SimpleController as an HTTP request handler and allows Spring to recognize it as a RESTful service.

@RequestMapping("/updateprofile") annotation sets the base path to the resource endpoints in the controller as /updateprofile.





@RequestMapping(method = RequestMethod.POST) is used to map HTTP POST request to the mapped controller methods. We used it to send details of a customer.

@RequestBody: This annotation takes care of binding the web request body to the method parameter with the help of the registered HttpMessageConverters. So, when you make a POST request to the "/updateprofile" URL with a Post JSON body, the HttpMessageConverters converts the JSON request body into a Post object and passes it to the updateProfile method.

Inside updateProfile method is where you create the query to update customer details in the customer table.

The update method provided by JdbcTemplate can be used for insert, update, and delete operations.

The SQL string is used to update the customer details by ID and pass the string to the update method of JdbcTemplate followed by object arguments of type string which are the email, password, name, phone, address no, state, city, area, and pin code. Note that the ID is only used to find the customer to be updated but the ID itself is not updated.

## 13. List Order

This method is used to display history of orders.

In the Projects window, Inside project file > source packages > com.spring.app. Open SimpleController.java file and write the following code.

@CrossOrigin(origins = "\*")

@RequestMapping(value = "/customerorder", method = RequestMethod.POST)

public JSONObject customerOrder(@RequestBody Order order) {

String s = "select distinct osrefid AS orderrefid, oscid AS customerid, oscustomername AS customername, oscustomeremail AS customeremail, oscustomerphone AS customerphoneno, oscustomeraddressno AS customeraddressno, oscustomerarea AS customerarea, oscustomercity AS customercity, oscustomerstate AS customerstate, oscustomerpincode AS customerpincode, ostotalprice AS totalprice, osordereddatetime AS ordereddatetime from ordersummary where oscid=('" + order.getCustomerid() + "') order by osrefid";

System.out.println("s = " + s);





List<Order> orderidList = jdbc.query(s, new BeanPropertyRowMapper(Order.class

));

System.out.println("orderidList = " + orderidList.isEmpty());

JSONArray orderArr = new JSONArray();

JSONObject orderObj = new JSONObject();

if (!orderidList.isEmpty()) {

for (Order orObj : orderidList) {

JSONObject orderDetObj = new JSONObject();

orderDetObj.put("orderrefid", orObj.getOrderrefid());

orderDetObj.put("customerid", orObj.getCustomerid());

orderDetObj.put("customername", orObj.getCustomername());

orderDetObj.put("customeremail", orObj.getCustomeremail());

orderDetObj.put("customerphoneno", orObj.getCustomerphoneno());

orderDetObj.put("customeraddressno",

orObj.getCustomeraddressno());

orderDetObj.put("customerarea", orObj.getCustomerarea()); orderDetObj.put("customercity", orObj.getCustomercity()); orderDetObj.put("customerstate", orObj.getCustomerstate()); orderDetObj.put("customerpincode", orObj.getCustomerpincode()); orderDetObj.put("totalprice", orObj.getTotalprice());





orderDetObj.put("ordereddatetime", orObj.getOrdereddatetime());

String t = "select osproductname AS productname, osproductprice AS productprice, osproductimage AS productimage, osproductquantity AS productquantity, osproductprice AS productprice, osproductdescription AS productdescription from ordersummary where oscid= '" + orObj.getCustomerid() + "' AND osrefid= '" + orObj.getOrderrefid() + "'";

System.out.println("t = " + t);

List<Orderproducts> productlist = jdbc.query(t, new BeanPropertyRowMapper(Orderproducts.class

));

System.out.println("productlist = " + productlist.isEmpty());

System.out.println("productlist = " + productlist);

JSONArray pdlistArr = new JSONArray();

if (!productlist.isEmpty()) {

for (Orderproducts pdlist : productlist) {

JSONObject pdlisObj = new JSONObject();

pdlisObj.put("productname", pdlist.getProductname());

pdlisObj.put("productquantity",

pdlist.getProductquantity());

pdlisObj.put("productprice", pdlist.getProductprice());

pdlisObj.put("productimage", pdlist.getProductimage());





pdlisObj.put("productdescription",

```
pdlist.getProductdescription());
```

pdlistArr.add(pdlis0bj);

}
orderDetObj.put("pdlist", pdlistArr);

orderArr.add(orderDetObj);

}
}
orderObj.put("orderdetails", orderArr);

}

return orderObj;

}

**@RestController:** This annotation marks the SimpleController as an HTTP request handler and allows Spring to recognize it as a RESTful service.

@RequestMapping("/customerorder") annotation sets the base path to the resource endpoints in the controller as /customerorder.

@RequestMapping (method = RequestMethod.POST is used to map HTTP POST request to the mapped controller methods. We used it to send customer details, order details, and product details.

@RequestBody: This annotation takes care of binding the web request body to the method parameter with the help of the registered HttpMessageConverters. So, when you make a POST request to the "/customerorder" URL with a Post JSON body, the HttpMessageConverters converts the JSON request body into a Post object and passes it to the customerOrder method.

Inside order method is where you create the query to return customer details, product details and order details as list based on customer ID from the ordersummary table.

The SQL s string contains a query to select the customer details, product details and order details by customer ID from the ordersummary table and if your SQL query is going to return a List of objects instead of just one object then you need to use the query () method of JdbcTempalte. These





methods provide to convert the result to a custom object. For instance, the simplest way to query and handle results is via the query (String, RowMapper) method. This method uses RowMapper to map the returned row to an object.

#### 14. List Products

This method is used to display products.

In the Projects window, Inside project file > source packages > com.spring.app. Open SimpleController.java file and write the following code.

@CrossOrigin(origins = "\*")

@RequestMapping(value = "/product", method = RequestMethod.POST)

public JSONObject product(@RequestBody Product product) {

String s = "select procategory AS category, proname AS name, prodescription AS description, proprice AS price, proimage AS image, proinitialquantity AS initialquantity, proquantity AS quantity, proid AS id from product where procategory=CAST('" + product.getCategory() + "' AS VARCHAR)";

System.out.println("select procategory AS category, proname AS name, prodescription AS description, proprice AS price, proimage AS image, proinitialquantity AS initialquantity, proquantity AS quantity, proid AS id from product where procategory=CAST('" + product.getCategory() + "' AS VARCHAR)");

```
List<Product> mrlist = jdbc.query(s, new
```

BeanPropertyRowMapper(Product.class

));

System.out.println("mrlist = " + mrlist);

JSONObject json = new JSONObject();

json.put("Productdetails", mrlist);

if (!mrlist.isEmpty()) {

E-Marketplace





json.put("Productdetails", mrlist);

System.out.println("json = " + json);

return json;

}

return json;

}

@RestController: This annotation marks the SimpleController as an HTTP request handler and allows Spring to recognize it as a RESTful service.

@RequestMapping("/product") annotation sets the base path to the resource endpoints in the controller as /product.

@RequestMapping(method = RequestMethod.POST) is used to map HTTP POST request to the mapped controller methods. We used it to return all the products.

@RequestBody: This annotation takes care of binding the web request body to the method parameter with the help of the registered HttpMessageConverters. So, when you make a POST request to the "/product" URL with a Post JSON body, the HttpMessageConverters converts the JSON request body into a Post object and passes it to the product method.

Inside product method is where you create the query to return products based on category from the product table.

The SQL string contains a query to select the product details by category from the product table and if your SQL query is going to return a List of objects instead of just one object then you need to use the query () method of JdbcTempalte. These methods provide to convert the result to a custom object. For instance, the simplest way to query and handle results is via the query (String, RowMapper) method. This method uses RowMapper to map the returned row to an object.

## 15. Order checkout

This method is used to display order summary.

In the Projects window, Inside project file > source packages > com.spring.app. Open SimpleController.java file and write the following code.





@CrossOrigin(origins = "\*")

@RequestMapping(value = "/ordersummary", method = RequestMethod.POST)

public String insertOrder(@RequestBody List<Order> odlist) {

String maxid = "select coalesce(max(osrefid),0) AS refid from
ordersummary";

int maxrefid = jdbc.queryForObject(maxid, Integer.class);

if (!odlist.isEmpty()) {

for (Orderproducts odlist1 : odlist.get(0).getProductlist()) {
 int quantity = Integer.parseInt(odlist1.getProductquantity());

String s = "select proid AS id, proname AS name, proimage AS image, prodescription AS description, proprice AS price, procategory AS category, proquantity AS quantity, proinitial quantity AS intial quantity from product where proid = " + Integer.parseInt(odlist1.getProductid()) + "";

List<Product> mrlist = jdbc.query(s, new

BeanPropertyRowMapper(Product.class));

System.out.println("totalquantity = " + mrlist);

for (int i = 0; i < mrlist.size(); i++) {</pre>

int totalquantity =

Integer.parseInt(mrlist.get(i).getQuantity());

if (quantity > totalquantity) {

return "Some Products are Out of Stock";

E-Marketplace





```
String s = "insert into
```

}

}

}

}

ordersummary(osproductname,osproductprice,osproductquantity,osproductimage,ostotal price,osordereddatetime,ospid,oscid,oscustomername,oscustomeremail,oscustomerphone ,oscustomeraddressno,oscustomerarea,oscustomercity,oscustomerstate,oscustomerpinco de,osrefid)"

+ "values(?,?,?,?,?,?,?,?,?,?,?,?,?,?,?)";

List<Object[]> dataObjList = new ArrayList<>();

```
if (!odlist.isEmpty()) {
```

```
Date d = new Date();
```

SimpleDateFormat formatter = new SimpleDateFormat("dd MMMM yyyy
HH:mm:ss z");

String strDate = formatter.format(d); String id = odlist.get(0).getCustomerid(); String totalprice = odlist.get(0).getTotalprice(); String name = odlist.get(0).getCustomername(); String email = odlist.get(0).getCustomeremail(); String phone = odlist.get(0).getCustomerphoneno();




String address = odlist.get(0).getCustomeraddressno(); String area = odlist.get(0).getCustomerarea(); String city = odlist.get(0).getCustomercity(); String state = odlist.get(0).getCustomerstate(); String pincode = odlist.get(0).getCustomerpincode();

```
for (Orderproducts odlist1 : odlist.get(0).getProductlist()) {
    Object[] dataObjArr = new Object[17];
    dataObjArr[0] = odlist1.getProductname();
    dataObjArr[1] = odlist1.getProductprice();
    dataObjArr[2] = odlist1.getProductquantity();
    dataObjArr[3] = odlist1.getProductimage();
    dataObjArr[4] = totalprice;
    dataObjArr[5] = strDate;
    dataObjArr[6] = Integer.parseInt(odlist1.getProductid());
    dataObjArr[7] = Integer.parseInt(id);
    dataObjArr[8] = name;
    dataObjArr[9] = email;
    dataObjArr[10] = phone;
    dataObjArr[11] = address;
    dataObjArr[12] = area;
```



}

}

```
dataObjArr[13] = city;
dataObjArr[14] = state;
dataObjArr[15] = pincode;
dataObjArr[16] = maxrefid + 1;
System.out.println("dataObjArr = " + Arrays.toString(dataObjArr));
```

res

dataObjList.add(dataObjArr);

System.out.println("dataObjList = " + dataObjList);

```
int[] a = jdbc.batchUpdate(s, dataObjList);
System.out.println("a = " + Arrays.toString(a));
System.out.println("a.length = " + a.length);
if (a.length >= 1) {
```

```
String t = "";
```

```
for (Orderproducts odlist1 : odlist.get(0).getProductlist()) {
```

t += "update product set proquantity = (CAST(proquantity AS INTEGER)-" + odlist1.getProductquantity() + ") where proid = (" + odlist1.getProductid() + ");";

}

IITM INCUBATION CELL



}



@RestController: This annotation marks the SimpleController as an HTTP request handler and allows Spring to recognize it as a RESTful service.

@RequestMapping("/ordersummary") annotation sets the base path to the resource endpoints
in the controller as /ordersummary.

@RequestMapping(method = RequestMethod.POST) is used to map HTTP POST request to the mapped controller methods. We used it to send order details, product details and customer details.

@RequestBody: This annotation takes care of binding the web request body to the method parameter with the help of the registered HttpMessageConverters. So, when you make a POST request to the "/ordersummary" URL with a Post JSON body, the HttpMessageConverters converts the JSON request body into a Post object and passes it to the insertOrder method.

Inside order method is where you create the query to insert customer details, product details in the ordersummary table and update product quantity in the product table.

The update method provided by JdbcTemplate can be used for insert, update, and delete operations.

The SQL string "s" is used to perform a single insert operation. Here '?' means it acts as the parameter which we need to pass while executing the query. Now to execute the query, we have used the JdbcTemplate update() method, which takes the query as an argument, and other than the query there are 4 values that correspond to 4 '?' respectively.

The SQL string "t" is used to update the product quantity by ID and pass the string to the update method of JdbcTemplate. Note that the ID is only used to find the customer to be updated but the ID

IITM NCUBAT

CELL





itself is not updated.

16. Generate Invoice & send via email

This method is used to send invoice via Gmail.

In the Projects window, Inside project file > source packages > com.spring.app. Open SimpleController.java file and write the following code.

```
@CrossOrigin(origins = "*")
```

```
@RequestMapping(value = "/sendmail", method = RequestMethod.POST)
```

```
public String sendMail(List<Object[]> dataObjList) {
```

// Recipient's email ID needs to be mentioned.

```
String to = "";
```

```
String OrderId = "";
```

```
String OrderedDateTime = "";
```

```
String Customername = "";
```

String Customernumber = "";

String Customeraddress = "";

String Totalprice = "";

// Sender's email ID needs to be mentioned
String from = "indigrainmarketplace@gmail.com";

final String username = "indigrainmarketplace@gmail.com";



```
final String password = "qgcilfhiyxhviqvt";
String HtmLFinal = "";
String HtmlCntTableRow = "";
```

for (int i = 0; i < dataObjList.size(); i++) {</pre>

```
int sNo = i + 1;
```

```
Object[] s = dataObjList.get(i);
System.out.println("Object = " + s[0]);
System.out.println("Object = " + s[1]);
System.out.println("Object = " + s[2]);
Totalprice = s[4].toString();
OrderedDateTime = s[5].toString();
to = s[9].toString();
Customername = s[8].toString();
Customernumber = s[10].toString();
Customernumber = s[11] + "," + s[12] + "," + s[13] + "," + s[14] +
```

"," + s[15].toString();

```
OrderId = s[16].toString();
```

IITM INCUBATION CELL

res





}

String htmlCnt1 = "<html>"

- + "<head>"
- + "</head>"
- + "<body>"

+ "<h1 style='color: #25a7e7; text-align:</pre>

center;'>Invoice</h1><div>

src=\"https://lh3.googleusercontent.com/PmbcUFjN8f9K8\_5fR-6M-m8K-

uHjNqeEI69X\_arC2kVkqQcJeDzd5bhhVHJ3gv5mX3Qyc00hZYECu\_Vpx-CuHYr3Hdebey5t01QJstD-04q8ge7uynwkFcDQWhbhhOKxZ4dnluJwsGaGUsVXfJFzBXGjvBgV\_6yscWIdphFjdCCqFCxa3QWtP3wZll 6pk548FN55wPFjEirszVgsfMDxlI4Qin7VeYotLaikRfDMjiApqm3ifCFGymeWMAKvZaKrC2Km8SMGMhpS xRS9yi\_zvguEfxTavCKL10EbFi2HbbGdBmkaoc9wjuAU7Zam0Uu-FsN6prMCNVVdORz0PaXpH1YJEHco3ssQ\_LsHRG5HC5K09OayzWD5KAiINY-dlt1JB6ny50VQha778ZxT7UzsLZcXG4W11u8iquzLb2JSmvp44RyiPGfuT2fDLGybYG2AP67cH9Azhq3P\_6biCckbzJLt9Mo-





61yygpi1i4eg0NiIqoTlEo4N8ytEI2-

tFbEaRBfjQKYnW0eAmGh2oY7j3wLYwYENIYkUiUKmnp5AJzkd5nuEIn1ddRyowULr3ducnonywGSDkuHqQ ofgUetupW1La7B9sWxUSgdEbSCP8Vxf9jKo\_Nx13p4aiydZawcqW7tidFkpopf3AoFDUkUfsWKaX4Wmh7g HUfrd6V8uDKw58u1DlaDsbCfvmvFkq3qkJndy\_OSW\_dHiSF5fP3IPB\_RBY9bJQWwsuJwvoaf-

oCUIf2h5URUSSLkh7e=w285-h358-no?authuser=0\" /><br

/><b>Indigrain</b></div><br />1px solid #25a7e7; border-collapse: collapse; '>#25a7e7; border-collapse: collapse; background-color: #25a7e7; color: white; textalign: center;'>Item #align: center;'>Item #style='border: 1px solid #25a7e7; border-collapse: collapse; background-color: #25a7e7; color: white; text-align: center;'>Product NameNamestyle='border: 1px solid #25a7e7; border-collapse: collapse; background-color: #25a7e7; color: white; text-align: center;'>Quantitystyle='border: 1px solid #25a7e7; border-collapse: collapse; background-color: #25a7e7; border-collapse: collapse; background-color: #25a7e7; color: white; text-align: center;'>Quantity

String htmlCnt2 = "border-collapse: collapse; text-align: right;'><b>Bill Amount</b><td style='text-align: right;'><b>" + Totalprice + "&#8377;</b>"

+ "</body>"

```
+ "</html>";
```

HtmLFinal = htmlCnt1 + HtmlCntTableRow + htmlCnt2;

Properties prop = new Properties();

prop.put("mail.smtp.host", "smtp.gmail.com");

prop.put("mail.smtp.port", "465");

```
prop.put("mail.smtp.auth", "true");
```

E-Marketplace





prop.put("mail.smtp.socketFactory.port", "465");
prop.put("mail.smtp.socketFactory.class",

"javax.net.ssl.SSLSocketFactory");

Session session = Session.getInstance(prop,

new javax.mail.Authenticator() {

@Override

protected PasswordAuthentication getPasswordAuthentication() {

return new PasswordAuthentication(username, password);

}

});

try {

// Create a default MimeMessage object.
Message message = new MimeMessage(session);

// Set From: header field of the header.
message.setFrom(new InternetAddress(from));

// Set To: header field of the header.

message.setRecipients(Message.RecipientType.TO,





InternetAddress.parse(to));

// Set Subject: header field

message.setSubject("Invoice");

// Send the actual HTML message, as big as you like
message.setContent(HtmLFinal, "text/html");

// Send message

Transport.send(message);

System.out.println("Sent message successfully....");

} catch (MessagingException e) {

throw new RuntimeException(e);

return "Invoice Generated";

#### }

}

@RestController: This annotation marks the SimpleController as an HTTP request handler and allows Spring to recognize it as a RESTful service.

@RequestMapping("/sendmail") annotation sets the base path to the resource endpoints in the controller as /sendmail.

@RequestMapping(method = RequestMethod.POST is used to map HTTP POST request to the





mapped controller methods. We used it to send invoice via mail to customer.

#### Configure pom.xml.

Open pom.xml file

For handling the web-request and doing CRUD operations with PostgreSQL database, we need the supporting of 3 Spring Boot dependencies: <a href="mailto:spring-boot-starter-web">spring-boot-starter-web</a>, <a href="mailto:sprin

#### **Configure Spring Data source.**

application. Properties is used to add the Spring Boot application's configurations such as: database configuration (PostgreSQL), server configuration.

In the Projects window, Inside project file > other sources > src/main/resources > default package. Open application. properties file.

Since we're using PostgreSQL as our database, we need to configure the database URL, username, and password so that Spring can establish a connection with the database on startup.

spring.jpa.hibernate.ddl-auto=none

spring.datasource.url=jdbc:postgresql://localhost:5432/postgres

spring.datasource.username=postgres

spring.datasource.password=chonar@13

spring.mvc.hiddenmethod.filter.enabled=true

spring.datasource.hikari.maximum-pool-size=2

#### Run the Spring Boot Project file.

Right-click on the project file and click on "Clean and Build".

#### **Installing Resin**

- 1. Go to <u>link</u>. Click on Download for Resin 4.0
- 2. Unzip resin-4.0.x.zip
- 3. Define the environment variable RESIN\_HOME to the location of Resin, for





example C:\Users\RP\Downloads\resin

4. Follow the similar process like setting Environment Variables in Java to set RESIN HOME

5. Execute resin.exe or run-in command prompt

```
resin/bin ./start.bat;
tail -f ../log/jvm-app-0.log;
```

Note: The resin server listens at port 8080 in the default configuration.

To fix 8080 ports already in use

Step 1: Open command prompt as administrator and find the process id that is using the port 8080.

netstat -ano | findstr 8080

Step 2: Kill the process using process id in above result.

taskkill /F /pid 1088

#### Deploying war file in the resin.

- 1. Go to spring boot project folder -> inside target folder you will find emarker.war file
- 2. Copy the .war file (E.g.: emarket.war) -> inside resin folder -> webapps folder
- 3. Start the resin server

Execute resin.exe

or run-in command prompt

```
resin/bin ./start.bat;
tail -f ../log/jvm-app-0.log;
```

4. Your .war file will be extracted automatically to a folder that has the same name (without extension) (E.g.: webapp)





## Creating a database for E-marketplace in PostgreSQL.

1. Create a e market database and Create merchant, customer, category, product, and order summary table, populate the table with data, retrieve and store data for future use, or delete if needed

### 2. Database Design



## 3. Downloading PostgreSQL Installer for Windows

Go to link. Download PostgreSQL

#### 4. Installing the PostgreSQL installer

After downloading the installer double click on it and follow the below steps:

research park
- ×     Setup - PostgreSQL        Welcome to the PostgreSQL Setup Wizard.     -     -     -     -     -     -     -     -     -     -     -     -     -     -     -     -     -     -     -     -     -     -     -     -     -     -     -     -     -     -     -     -     -     -     -     -     -     -     -     -     -     -     -     -     -     -     -     -     -     -     -     -     -     -     -      -
< Back Next > Cancel

Step 1: Click the Next button.

Step 2: Choose the installation folder, where you want PostgreSQL to be installed, and click on Next.

	🗃 Setup — 🗆 🗙	
	Installation Directory	
	Please specify the directory where PostgreSQL will be installed. Installation Directory C:\Program Files\PostgreSQL\11	
	InstallBuilder 	121
F	E-Marketplace	141





Step 3: Select the components as per your requirement to install and click the Next button.

M Setup	- 🗆 ×
Select Components	
Select the components you want to install; clear the cor you are ready to continue.	nponents you do not want to install. Click Next when
<ul> <li>PostgreSQL Server</li> <li>pgAdmin 4</li> <li>Stack Builder</li> <li>Command Line Tools</li> </ul>	Stack Builder may be used to download and install additional tools, drivers and applications to complement your PostgreSQL installation
nstallBuilder	< Back Next > Cancel

Step 4: Select the database directory where you want to store the data and click on Next.

🖥 Setup		_		
Data Directory				
Please select a directory under which to store your data.				
Data Directory ::\Program Files\PostgreSQL\11\data				
InstallBuilder	< Back Nex	t>	Cancel	
E-Marketplace				122





# Step 5: Set the password for the database superuser (Postgres)

Setup	×
Password	
Please provide a password for the data	abase superuser (postgres).
Password *******	
Retype password *******	
stallBuilder	
	< Back Next > Cancel
tep 6: Set the port for Posto is port. If unsure leave it to	greSQL. Make sure that no other applications are using its default (5432) and click on Next.
tep 6: Set the port for Posto is port. If unsure leave it to	greSQL. Make sure that no other applications are using its default (5432) and click on Next. $ \Box$ $\times$
tep 6: Set the port for Postg is port. If unsure leave it to	greSQL. Make sure that no other applications are using its default (5432) and click on Next. $ \Box$ $\times$
tep 6: Set the port for Postg is port. If unsure leave it to Setup	greSQL. Make sure that no other applications are using its default (5432) and click on Next.
tep 6: Set the port for Postg is port. If unsure leave it to Setup Port	greSQL. Make sure that no other applications are using its default (5432) and click on Next. — □ ×
tep 6: Set the port for Postg is port. If unsure leave it to Setup Port	greSQL. Make sure that no other applications are using its default (5432) and click on Next.
tep 6: Set the port for Postg is port. If unsure leave it to Setup Port Please select the port number the serv	greSQL. Make sure that no other applications are using its default (5432) and click on Next.
tep 6: Set the port for Postg is port. If unsure leave it to Setup Port Please select the port number the serv Port 5432	preSQL. Make sure that no other applications are using its default (5432) and click on Next.
tep 6: Set the port for Postg is port. If unsure leave it to Setup Port Please select the port number the serv Port 5432	preSQL. Make sure that no other applications are using its default (5432) and click on Next.
tep 6: Set the port for Postg is port. If unsure leave it to Setup Port Please select the port number the serv Port 5432	preSQL. Make sure that no other applications are using its default (5432) and click on Next.
tep 6: Set the port for Postg is port. If unsure leave it to Setup Port Please select the port number the serv Port 5432	preSQL. Make sure that no other applications are using its default (5432) and click on Next.
tep 6: Set the port for Postg is port. If unsure leave it to Setup Port Please select the port number the serv Port 5432	preSQL. Make sure that no other applications are using its default (5432) and click on Next.
tep 6: Set the port for Postg is port. If unsure leave it to Setup Port Please select the port number the serv Port 5432	preSQL. Make sure that no other applications are using its default (5432) and click on Next.
tep 6: Set the port for Post is port. If unsure leave it to Setup Port Please select the port number the serv Port 5432	preSQL. Make sure that no other applications are using its default (5432) and click on Next.
tep 6: Set the port for Postg is port. If unsure leave it to Setup Port Please select the port number the serv Port 5432	preSQL. Make sure that no other applications are using its default (5432) and click on Next.
tep 6: Set the port for Postg is port. If unsure leave it to Setup Port Please select the port number the serv Port 5432	ereSQL. Make sure that no other applications are using its default (5432) and click on Next.
tep 6: Set the port for Postg is port. If unsure leave it to Setup Port Please select the port number the serv Port 5432	er should listen on.
tep 6: Set the port for Postg is port. If unsure leave it to Setup Port Please select the port number the serv Port 5432	preSQL. Make sure that no other applications are using its default (5432) and click on Next.
tep 6: Set the port for Postg is port. If unsure leave it to Setup Port Please select the port number the serv Port 5432	preSQL. Make sure that no other applications are using its default (5432) and click on Next.
tep 6: Set the port for Postor is port. If unsure leave it to Setup Port Please select the port number the serv Port 5432	preSQL. Make sure that no other applications are using its default (5432) and click on Next.
tep 6: Set the port for Postg is port. If unsure leave it to Setup Port Please select the port number the serv Port 5432	ere Sould listen on.





Step 7: Choose the default locale used by the database and click the Next button.

	etun	_		
Adv	vanced Options			
Sel	ect the locale to be used by the new database cluster.			
200				
Install	Builder	Neut	Canaal	
		Next > j	Cancel	
Sten	8: Click the Next button to start the installation			
otop				
	etup	_		
S Rea	etup ady to Install	_	×	,
etep S Rea	etup ady to Install	_	×	
Rea Setup	etup ady to Install o is now ready to begin installing PostgreSQL on your computer.	-		
Rea Setup	etup etup o is now ready to begin installing PostgreSQL on your computer.			
Rea Setup	etup <b>ady to Install</b> o is now ready to begin installing PostgreSQL on your computer.		×	
Rea Setup	etup <b>ady to Install</b> is now ready to begin installing PostgreSQL on your computer.			
Rea Setup	etup ady to Install o is now ready to begin installing PostgreSQL on your computer.	_		
Rea Setup	etup ady to Install o is now ready to begin installing PostgreSQL on your computer.		×	
Rea Setup	etup <b>ady to Install</b> o is now ready to begin installing PostgreSQL on your computer.			
Rea Setup	etup ady to Install is now ready to begin installing PostgreSQL on your computer.	_		
Rea Setup	etup edy to Install o is now ready to begin installing PostgreSQL on your computer.			
Setup	etup ady to Install o is now ready to begin installing PostgreSQL on your computer.			
Setup Setup	etup edy to Install o is now ready to begin installing PostgreSQL on your computer. Builder			
Setup Setup InstallE	etup  ady to Install  b is now ready to begin installing PostgreSQL on your computer.  Builder	Next >	Cancel	
Setup Setup	etup  dy to Install  is now ready to begin installing PostgreSQL on your computer.  Builder	Next >	Cancel	
Setup Setup	etup  ady to Install  p is now ready to begin installing PostgreSQL on your computer.  Builder	Next >	Cancel	124





Wait for the installation to complete, it might take a few minutes.

Setup		_		×
installing				
ase wait while Setup installs Po	stgreSQL on your computer.			
ipacking C: \Program [] \doc\p	ostgresql\html\catalog-pg-seclabel.html			
tallBuilder		Nette		
n 9: Click the Finish	<pre>Section to complete the Posto</pre>	Next >	ation	
Setup				×
Packaged by:	Completing the PostgreSQL Set	up Wizard		
POSTGRES	Setup has finished installing PostgreS	QL on your comput	er.	
PostgreSQL				
UG JE				
U				
	< Back	Finish	Can	cel
	< Back	Finish	Can	cel





4. When you install PostgreSQL, pgAdmin is installed. Start pgAdmin from your start menu.

## 5. Create Server

Go to the "Dashboard" tab. In the "Quick Link" section, click "Add New Server" to add a new connection.



- 6. In the General tab, enter the name for this server.
- 7. Select the "Connection" tab in the "Create-Server" window.





🗐 Creat	e - Server					×
General	Connectio	n SSL	SSH Tunnel	Advanced	ł	
Host name/ad	ddress	localhos	t			
Port		5432				
Mainten databas	ance e	postgres	5			
Usernan	ne	postgres	5			5
Passwor	ď	•••••				
Save passwor	d?					
Role						
Service						
i	?			🖺 Save	🗙 Cancel	🛟 Reset

In the default PostgreSQL setup, the administrator user is postgres with an empty password. In the connection tab be sure to have the host set to localhost. Click Save afterwards.

#### 8. Create New User

First, connect to the database by double-clicking on the instance name you created above.

Right click on Login/Group Roles, select Create and click on Login/Group Roles... for creating new user.



research park	IITM INCUBATION CELL

The following Create dialog box will appear. Type 'emarket in Name field as user's name and click on Definition tab.

	A Create - Login/Group Rc	ble		2 X	
Browser S III To Q >	General Definition Priv	vileges Membership Parameter	s Security SQL	ostgres/postg	S postgres/p < >
> Types	Name	emarket			
	Comments			h.	
<ul> <li>♥ contract</li> <li>♥ contract</li> <li>♥ postgres</li> <li>&gt; @ Casts</li> <li>&gt; ♥ Catalogs</li> <li>♥ Event Triggers</li> <li>&gt; ♥ Schemas (1)</li> <li>&gt; ♥ write</li> </ul>	i ?		X Cancel 🖧 Reset 🖬 S	ave	

Type login password for the user admin and click on Privileges tab. If you want to create user for limited time, then set the Account expires data and time value.



IITM Incubation Cell litm, research park

> (i)			oet	tares /posta	nostares /r	1 .
	Trigger Functions General Definit	ion Privileges Membership Parameters Security SQL	1	igres/postg	s postgres/p	*
> 🛅	Types Password					
> 🏷 Subscr	ptions Account expires	No Expiry	<b>#</b>			
> sticdevmgm	w1	Please note that if you leave this field blank, then password will never	r expire.			
> sicprod	Connection limit	-1				
> 📑 postgres						
> sprodmgmt	/1					
> ranchodb	1					
> A Login/Group F	oles					
> 🔁 Tablespaces						
✓						
🗸 🍔 Databases (1)						
✓ ≥ postgres						
> 🚱 Casts						
> 💖 Catalo	js					
> C Event 1	riggers					
	ons					
> 🗊 Extens	Data Wrannera		Down			
> 🎲 Extens > 🛒 Foreigr	Data Wrappers	Monard Street				
> 🔀 Extens > 📑 Foreign > 🤤 Langua > 🔊 Publica	ges tions	X Cancel 22 Reset	• Save			
>	a Data Wrappers ges tions is (1)	× Cancel	■ Save			

To set all permissions to emarket user make all options to 'Yes'. Click 'Save' button to create the user.

File	AutoSave Off) 🗐 Build Lab E-Market Pace	e • Saved to this PC • P s	Search		Rishi Gane	esh RG // - 0 X - 0 X
۲. الع الع	PgAdmin     File ×     Object ×     Tools       Browser     Image:	A Create - Login/Group Role			<b>∠* ×</b> ostgres/post	g 💲 postgres/; < 🗲 🕵
Und	<ul> <li>A Login/Group Roles</li> <li>Tablespaces</li> </ul>	General Definition Privileges Can login?	Membership Parameters	Security SQL	₩ 🗉 Υ	\$0 \$0 <b>₫</b> ~ <b>±</b>
L	<ul> <li>Image: Image: Ima</li></ul>	Superuser?	Yes			
	Br Casts     Section 2      Sec	Create roles?	Yes			
-	> 😱 Event Triggers > 😨 Extensions	Create databases?	Yes			
÷	> d Foreign Data Wrappers > 🤤 Languages	Inherit rights from the parent	Yes			
	> 🕸 Publications > 💖 Schemas (1)	roles?	No			
	✓ ♦ public > ♣↓ Collations	replication and backups?				
 m .	A Domains     B FTS Configurations     B FTS Distinguishing					
	Aa FTS Parsers     More TS Templates					
	> @ Foreign Tables > (@ Functions	i ?		🗙 Cancel 🛛 🖧 Reset	B Save	
. 9	<ul> <li>&gt; (c) Materialized Views</li> <li>&gt; (c) Procedures</li> </ul>				_	
. 7 .	1 2 Contronoor	9. Create a table.				
Page 114	i of 121 15336 words 🗰 English (United State	s) 📷 🔀 Accessibility: Investigate	) 💋 🧟 🖬		(D) Focus	ENG 03:53 AM 03-03-2023

'emarket user entry will be shown in Login/Group Roles section.





Proweer 💭 📼		heard Branartian SOI Statistics D	pondoncias Donondonta 🕿 postaros		
> Cicdevpbi > Cicdevpbi > Cicprod > postgres	<u>™</u> Q >_	nboard Properties SQL Statistics D	Bependencies Dependents B postgress	/postg s postgres/postg	spostgres/r 🕻 > 🛚
> 📑 prodmgmtv1		a Output Explain Messages Notificatio	ns		
<ul> <li>&gt; ■ anchodb</li> <li>&gt; ■ twinfoodb</li> <li>&gt; ▲ Login/Group Roles</li> <li>&gt; ▲ Tablespaces</li> <li>♥ ■ market</li> <li>■ Databases (1)</li> <li>&gt; ■ postgres</li> </ul>			No data output. Execute a query to ge	et output.	
✓ ♣ Login/Group Roles (10 O annucleat	)				
▲ pg_execute_server, ▲ pg_monitor ▲ pg_read_all_setting ▲ pg_read_all_stats	_program Is				
Apg_read_server_file	s	ry Editor Query History			
<ul> <li>Apg_signat_backend</li> <li>Apg_stat_scan_table</li> <li>Apg_write_server_fili</li> <li>Apostgres</li> <li>&gt; ■ Tablespaces</li> <li>&gt; ■ Tablespaces</li> </ul>	15 25				
> 📑 twinfood					

#### 9. Create a table.

Left click on the Database section and select the required database, in this case the name of database is postgres

	Dashboard	Prope	rties SQL	Statistics [	Dependencies	Depend	ents				
✓ ≣ rishi (6) > ≝ Build Club	Database s	sessions			Total 📕 Activ	re 📕 Idle	Transactions per secon	nd 📘 T	ransactions 📕 (	Commits 📕 Rollba	acks
<ul> <li>         Emarketplace     </li> <li>         emarket                   </li> </ul>	1						1				_
<ul> <li>✓ ■ postgres</li> <li>→          <sup>™</sup> Casts</li> <li>→          <sup>™</sup> Catalogs</li> </ul>	0						0				_
C Event Triggers	Tuples in		nearte 📕 I In	datas 📕 Dalata	Tuples out		Eatched Returned	Block I/O		Paade	Lite
<ul> <li>&gt;</li></ul>	0				0			0			
> Efranchoprogram > Eftwinfood	Server activity										
-	Sessions	Locks	Prepared	Transactions					Q Search		С
		PID	User	Application		Client	Backend start	State	Wait event	Blocking PIDs	

## 10. PostgreSQL- Database Selection

Now left click on the database and then select the Schemas section using the left mouse button. In this case we left click on postgres





Addition Pile + Object + Hous +	Help *										
Browser 🕃 🎟 To Q >_	Dashboard	Dashboard Properties SQL Statistics Dependencies Dependents									
<ul> <li>         Image: Image</li></ul>	Database se	essions		Total 📕 Active	e 📕 Idle	Transactions per secon	id 📘 T	Transactions 📕 Commits 📕 Rolibacks			
✓ ● postgres	1					3					
> 🐻 Casts								Λ			
> 💖 Catalogs						2		- 1			
<ul> <li>C Event Triggers</li> </ul>								1			
>						1				-	
> 🥃 Foreign Data Wrappers											
> 🤤 Languages	0				0		-				
> © Publications											
<ul> <li>Schemas (1)</li> </ul>	Tuples in	📕 Inserts 📕 Up	dates 📕 Delete	Tuples out		Fetched Returned	Block I/O		Reads K	Hits	
✓ ♦ public	1			1600			90		1		
> BUCollations				1400			70		-		
> 🏠 Domains				1000			60		1		
> (a) FTS Configurations				800			40				
> IN FIS Dictionaries				400			30		+		
> Aa FIS Parsers				200			10				
> (a) FTS Templates	0										
> (a Eurotiona											
Materialized Views	Server activity										
> (r) Procedures	Sessions	Locks Prepared	Transactions			Q Search	Q Search 2				
> 13Sequences											
> E Tables		PID User	Application		Client	Backend start	State	Wait event	Blocking PIDs		
> (a) Trigger Functions											
	C 🔳 🕨	2396 postgres	pgAdmin 4 - Di	B:postgres	::1	2023-03-02 22:57:31 IST	active				
			" Donna	113							

## 11. PostgreSQL- Selecting Schemas

Now right click on the public section to select the Create option from the drop-down menu.

Browser	8	III Ta Q >_	Dashboard	Propertie	s SQL	Statistics	Dependencies	Dependents	🛭 🗊 postgres/postg.	. 🗊 postgres,	/postg	🕏 postgres/j	< > <b>8</b>
	> 🥑 Foreign Da	ta Wrappers	s 🖕	8 × 8	Q ~			T v	No limit 🗸 🔳	▶ <u> </u>		≅ <b>₫</b> ~	*
	> Canguages	3	* <b>*</b> ~										
	> C Publication	15	Data Outour	Evolain	Massage	o Notifio	ations						
	<ul> <li>Schemas (</li> <li>Schemas (</li> </ul>	.0	Data Outpu	Expidin	wessaye	S NUTITO	auons						
	> 640	Create	>				•						
	> 🏠 E						No data	a output. Execu	te a query to get outpu	t.			
	> 🕼 F	Refresh	C alter	0									
	> 🕅 F	Delete/Drop	Edito	Query H	istory								
	> AaF	Drop Cascade											
	> 🖂 F	CREATE Script											
	> (@)F	Backup											
	> 🧱 N	Restore											
	> (() F	Grant Wizard											
	> 1.35	Search Objects											
	> 曲 I	PSQL Tool (Beta)	_										
	> 01	Query Tool											
	> 🥅 🔪	Properties											
	> "> Subscription	ons											
> 4	Login/Group Role	s											
	iaplespaces												
	vinfood												

## 12. PostgreSQL- Create Table

Create a table named customer:

Type the following query in the Query editor panel.

## CREATE TABLE IF NOT EXISTS public.customer





#### cid serial,

cname text COLLATE pg\_catalog."default" NOT NULL,

cemail text COLLATE pg\_catalog."default" NOT NULL,

cphone text COLLATE pg\_catalog."default",

cpassword text COLLATE pg\_catalog."default" NOT NULL,

caddressno text COLLATE pg\_catalog."default",

carea text COLLATE pg\_catalog."default",

ccity text COLLATE pg\_catalog."default",

cstate text COLLATE pg\_catalog."default",

cpincode text COLLATE pg\_catalog."default",

CONSTRAINT cid PRIMARY KEY (cid),

CONSTRAINT uniqemail UNIQUE (cemail),

CONSTRAINT uniqmobile UNIQUE (cphone)

In the admin table "cid", "cname", "cemail", "cphone", "cpassword", "caddressno", "carea", "ccity", "cstate", and "cpincode" represents the name of the columns. INT and TEXT are data types and NOT NULL defines the column constraint, NOT NULL means no acceptance of NULL values in that column. Here, "cid" is defined as the Primary Key Column. The primary key column is used for distinguishing a unique row in a table. AUTO\_INCREMENT to create a column whose value can be set automatically from a simple counter. You can only use AUTO\_INCREMENT on a column with an integer type. The column must be a key, and there can only be one AUTO\_INCREMENT column in a table. UNIQUE to specify that all values in the cstemail column must be distinct from each other. For UNIQUE indexes, you can specify a name for the constraint, using the CONSTRAINT keyword. That name will be used in error messages.

After entering query, select the Execute/Refresh icon from the toolbar.

13. Create a table named merchant:

Type the following query in the Query editor panel.





#### CREATE TABLE IF NOT EXISTS public.merchant

mid serial,

mname text COLLATE pg\_catalog."default",

mphone text COLLATE pg\_catalog."default",

memail text COLLATE pg\_catalog."default" NOT NULL,

maddress text COLLATE pg\_catalog."default",

mgstno text COLLATE pg\_catalog."default",

mpassword text COLLATE pg\_catalog."default" NOT NULL,

CONSTRAINT mid PRIMARY KEY (mid)

In the merchant table "mid", "mname", "mpassword", "memail", "maddress", "mgstno", and "mpassword" represents the name of the columns. INT and TEXT are data types and NOT NULL defines the column constraint, NOT NULL means no acceptance of NULL values in that column. Here, "mid" is defined as the Primary Key Column. The primary key column is used for distinguishing a unique row in a table. AUTO\_INCREMENT to create a column whose value can be set automatically from a simple counter. You can only use AUTO\_INCREMENT on a column with an integer type. The column must be a key, and there can only be one AUTO\_INCREMENT column in a table.

After entering query, select the Execute/Refresh icon from the toolbar.

14. Create a table named category:

Type the following query in the Query editor panel.

CREATE TABLE IF NOT EXISTS public.category

catid serial,

catcategoryname text COLLATE pg\_catalog."default" NOT NULL,





catcategoryimage text COLLATE pg\_catalog."default" NOT NULL,

CONSTRAINT catid PRIMARY KEY (catid)

In the category table "catid", "catcategoryname" and "catcategoryimage" represents the name of the columns. INT and TEXT are data types and NOT NULL defines the column constraint, NOT NULL means no acceptance of NULL values in that column. Here, "mid" is defined as the Primary Key Column. The primary key column is used for distinguishing a unique row in a table. AUTO\_INCREMENT to create a column whose value can be set automatically from a simple counter. You can only use AUTO\_INCREMENT on a column with an integer type. The column must be a key, and there can only be one AUTO\_INCREMENT column in a table.

After entering query, select the Execute/Refresh icon from the toolbar.

15. Create a table named product:

Type the following query in the Query editor panel.

#### CREATE TABLE IF NOT EXISTS public.product

proid serial,

proname text COLLATE pg\_catalog."default" NOT NULL,

prodescription text COLLATE pg\_catalog."default" NOT NULL,

procategory text COLLATE pg\_catalog."default" NOT NULL,

proimage text COLLATE pg\_catalog."default" NOT NULL,

proprice text COLLATE pg\_catalog."default" NOT NULL,

proquantity text COLLATE pg\_catalog."default" NOT NULL,

proinitialquantity text COLLATE pg\_catalog."default",

CONSTRAINT proid PRIMARY KEY (proid)





In the product table "proid", "proname", "prodescription", "procategory", "proimage", "proprice", "proquantity", and "proinitialquantity" represents the name of the columns. INT and TEXT are data types and NOT NULL defines the column constraint, NOT NULL means no acceptance of NULL values in that column. Here, "proid" is defined as the Primary Key Column. The primary key column is used for distinguishing a unique row in a table. AUTO\_INCREMENT to create a column whose value can be set automatically from a simple counter. You can only use AUTO\_INCREMENT on a column with an integer type. The column must be a key, and there can only be one AUTO\_INCREMENT column in a table.

After entering query, select the Execute/Refresh icon from the toolbar.

16. Create a table named ordersummary:

Type the following query in the Query editor panel.

#### CREATE TABLE IF NOT EXISTS public.ordersummary

oscid bigint,

osproductquantity text COLLATE pg\_catalog."default",

osproductname text COLLATE pg\_catalog."default",

osproductprice text COLLATE pg\_catalog."default",

ostotalprice text COLLATE pg\_catalog."default",

osordereddatetime text COLLATE pg\_catalog."default",

ospid bigint,

osid serial,

oscustomername text COLLATE pg\_catalog."default",

oscustomeremail text COLLATE pg\_catalog."default",

oscustomerphone text COLLATE pg\_catalog."default",

oscustomeraddressno text COLLATE pg\_catalog."default",

oscustomerarea text COLLATE pg\_catalog."default",





#### oscustomercity text COLLATE pg\_catalog."default",

oscustomerstate text COLLATE pg\_catalog."default",

oscustomerpincode text COLLATE pg\_catalog."default",

osproductimage text COLLATE pg\_catalog."default",

osproductdescription text COLLATE pg\_catalog."default",

osrefid bigint,

CONSTRAINT osid PRIMARY KEY (osid)

In the admin table "osid", "ospid", "osproductname", "osproductdescription", "osproductprice", "osproductimage", "osproductquantity", "ostotalprice", "osordereddatetime", "oscid", "oscustomername", "oscustomeremail", "oscustomerphone", "oscustomeraddressno", "oscustomerarea", "oscustomercity, "oscustomerstate", "oscustomerpincode", and "osrefid" represents the name of the columns. INT, BIGINT, and TEXT are data types and NOT NULL defines the column constraint, NOT NULL means no acceptance of NULL values in that column. Here, "osid", "oscid" are defined as the Primary Key Column. The primary key column is used for distinguishing a unique row in a table. AUTO\_INCREMENT to create a column whose value can be set automatically from a simple counter. You can only use AUTO\_INCREMENT on a column with an integer type. The column must be a key, and there can only be one AUTO\_INCREMENT column in a table.

After entering query, select the Execute/Refresh icon from the toolbar.

17. Insert a record into the merchant table.

Type the following query in the Query editor panel.

INSERT INTO merchant (mname, mphone, memail, maddress, mgstno, mpassword) VALUES ('indigrain', '1234567890', 'indigrain@gmail.com', 'chennai', '9876543210', 'indi@123');

The 'merchant is an already created table. Now we are adding a new row of records under the respective columns with the corresponding values: 'indigrain', '1234567890', 'indigrain@gmail.com', 'chennai', '9876543210', 'indi@123'.

After entering query, select the Execute/Refresh icon from the toolbar.





## Testing the backend with the mobile application.

Note: Make sure your computer and phone are on the same Wi-Fi network.

1. Change the api call URL from <u>http://121.242.232.216:7070/emarket/</u> to <u>http://<Wifi ipaddress>:8080/emarket/</u>

In flutter project file ->

- Inside lib/Screen/api/api.dart
- Inside lib/Screen/category\_management/api\_service.dart
- Inside lib/Screen/customerhome/ApiServiceProjectDetail.dart
- Inside lib/Screen/product\_management/api\_service.dart
- Inside lib/Screen/ordersummary/ordersummary.dart

## Example: (http://192.168.68.27:8080/emarket/)

2. Start the resin server.

Execute resin.exe

or run-in command prompt

resin/bin ./start.bat; tail -f ../log/jvm-app-0.log;

- 3. Perform functional tests and validate if all the functionalities work according to requirements.
  - 1. Merchant Login
    - 1. Add, View, Remove Categories
    - 2. Add, View, Edit, Remove Products
    - 3. Check Incoming/New Orders
  - 2. Customer Registration





- 3. Customer Login
  - 1. View Categories
  - 2. View Products
  - 3. Add to Cart
    - a. Add/ Remove Items
    - b. Increase/ Decrease Quantity
    - c. Quantity Check
  - 4. Update Profile, Address
  - 5. Make payment using different payment modes.
  - 6. Check history of orders