ONLINE FURNITURE SELLING APPLICATION

Contents

A. Project Proposal

Introduction

Project Objectives

Scope

Methodology

TOOLS AND TECHNOLOGIES

Timeline

Resource

Expected Outcome

References

B. Event list

The event list contains the “list” of all events. Events are the tasks performed by the end users. Event list helps in clarifying the events which could be performed. For this project, there is only one event list as user and admin panel does not differ.

Here are the events the users can perform

Users can login and register to our system using their Gmail
Registered / logged-in users can shop the furniture from our application
Registered /logged in user can shop different types of home decorative furniture’s
Registered /logged in the system can view, and purchases the furniture

Registered /logged in can save the items and add to cart when ever he wants

The registered / logged in user can change the themes according to there privilege

Registered/logged in user can buy the furniture by using the cod or via credit card

Admin can perform add product which will reflect to the client application

Admin can perform update product which will reflect to client application

Admin can delete the product

Admin can view the number of orders are done

Admin can view the total number of payment is been done

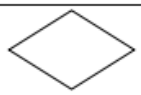
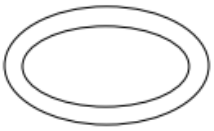
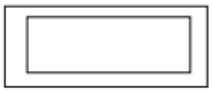
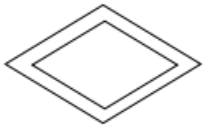
Admin can view payment details performed while purchasing the furniture

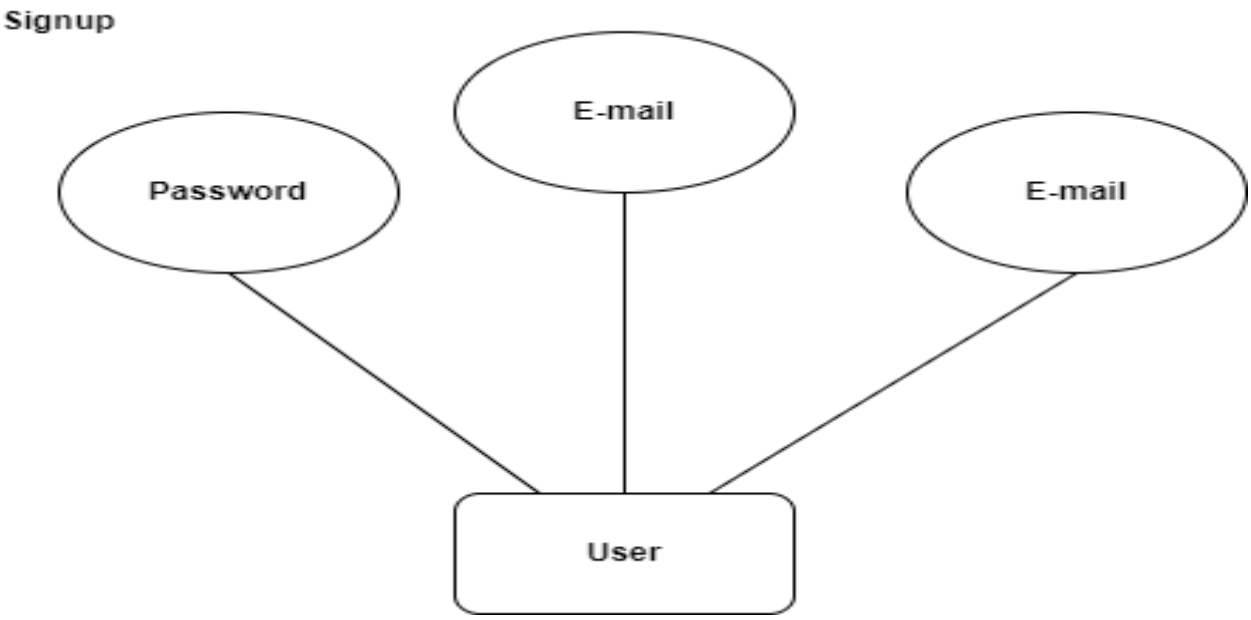
C. Diagrams

1.1. Entity Relationship Diagram (ER-Diagram)

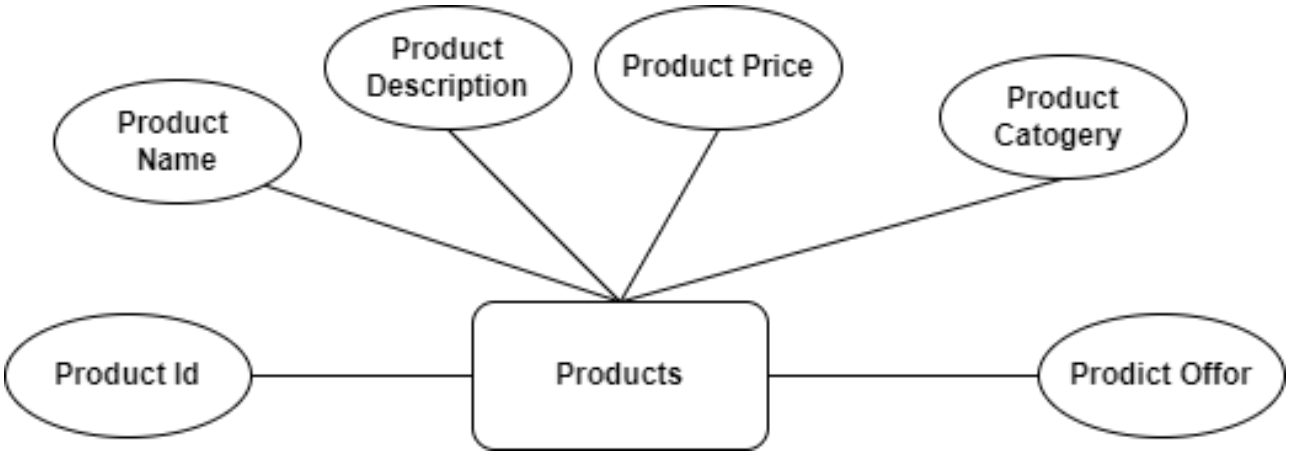
- Entity relationship diagram can express overall logical structure of database logically.
- 2. ER Diagrams are simple and clear.
- 3. ER Diagrams represents entities and tables and their relationship with one another.

Components of ER diagram:

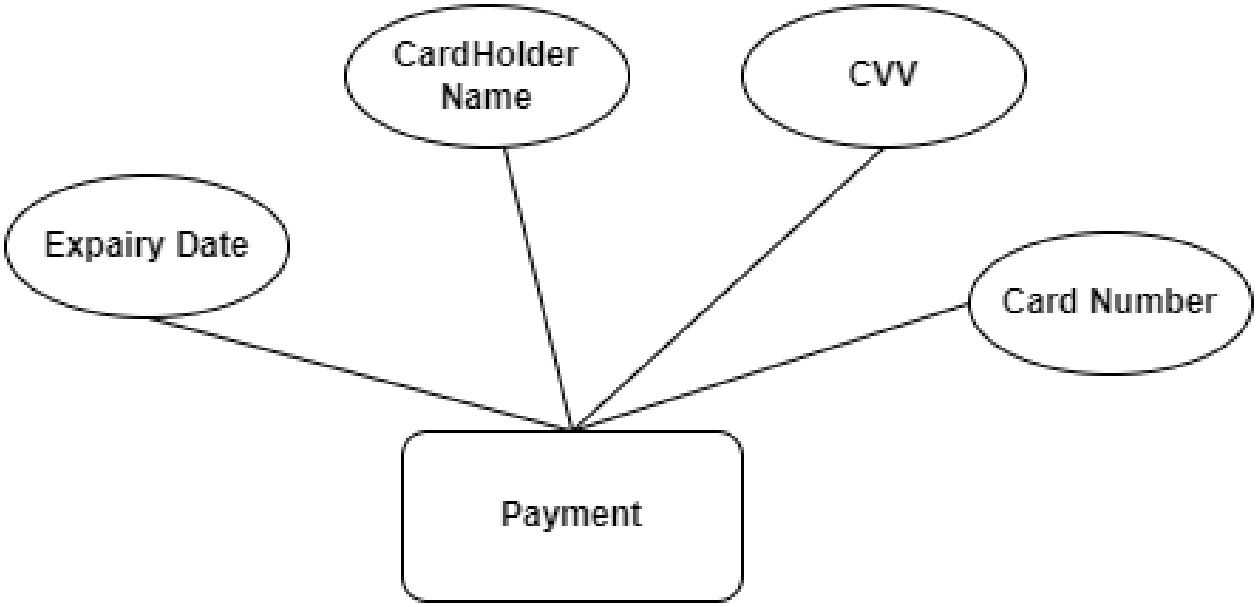
Sr. No.		Shape	Description
1	Rectangle		Represents entity set.
2	Ellipse		Represents attributes.
3	Diamond		Represents relationship.
4	Flow lines		Represents link between 2 entities set.
5	Double ellipse		Represents multivalve attributes.
6	Dashed ellipse		Denotes derived attributes.
7	Double Rectangle		Represents weak entity set.
8	Double Diamond		Represents relationship set for weak entity set.


Signup
E-mail
Password
E-mail
User

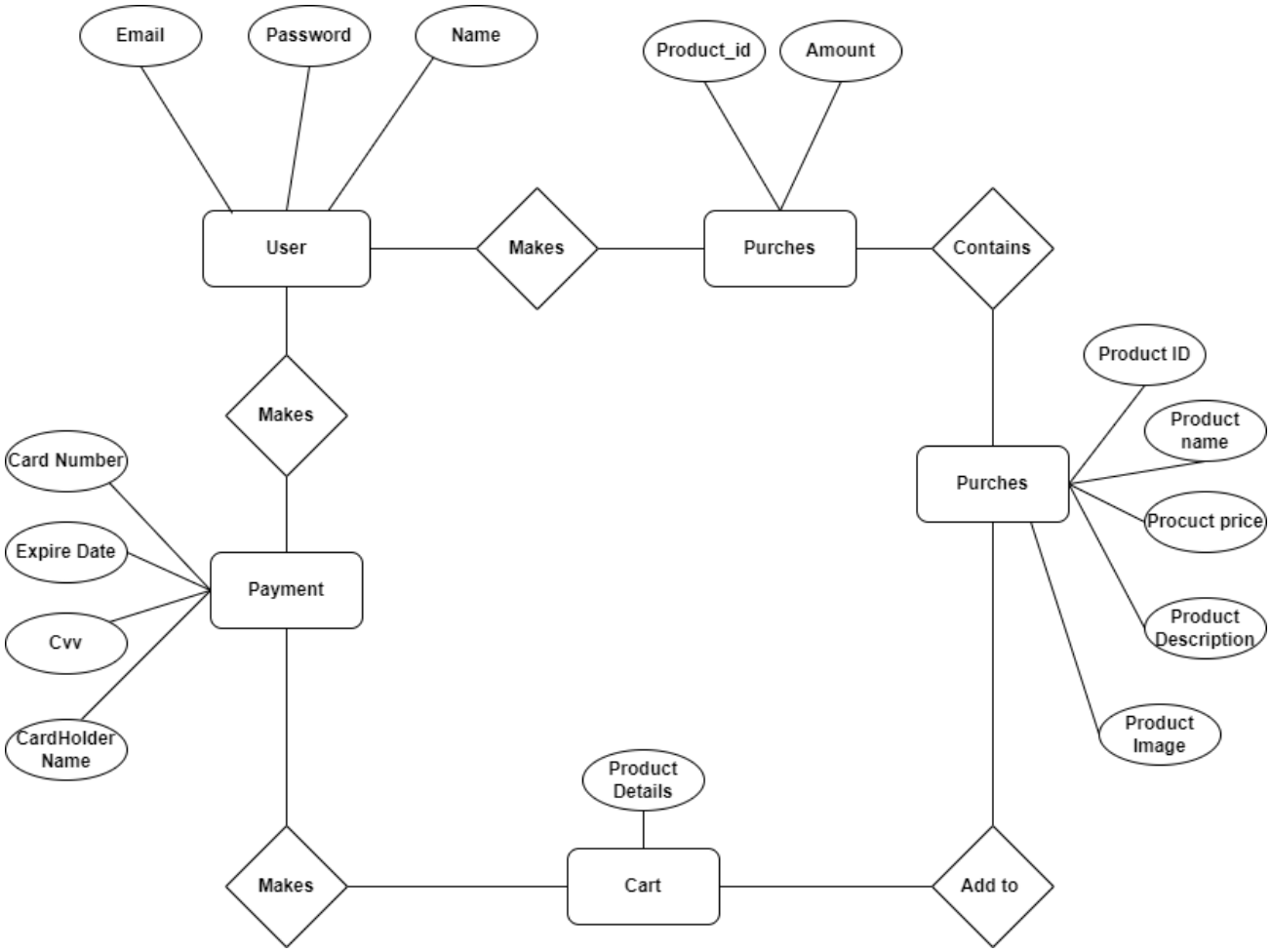
Products


Product Description
Product Price
Product Name
Product Catogery
Product Id
Products
Prodict Offor

Payment



ER-Diagram


Email
Password
Name
Product_id
Amount
User
Makes
Purches
Contains
Makes
Product ID
Product name
Card Number
Purches
Procuct price
Expire Date
Payment
Cvv
Product Description
CardHolder Name
Product Image
Product Details
Makes
Cart
Add to

1.2. Use Case Diagram

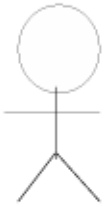
Admin Subsystem:

- Admin add/update/delete details of product.
- Admin view the order details.
- Admin view the user details and update/delete user

Customer Subsystem:

- New customers register into the system.
- Register customer log into the system.
- Customers view their own details.
- Customer order product.
- Customers pay the payment.
- Customers can generate receipt

A use case diagram is a set of scenarios that describing an interaction between user and system. A use case diagram displays the relationship among actors & use cases. The 2 main components of use case diagram are use case and actor.

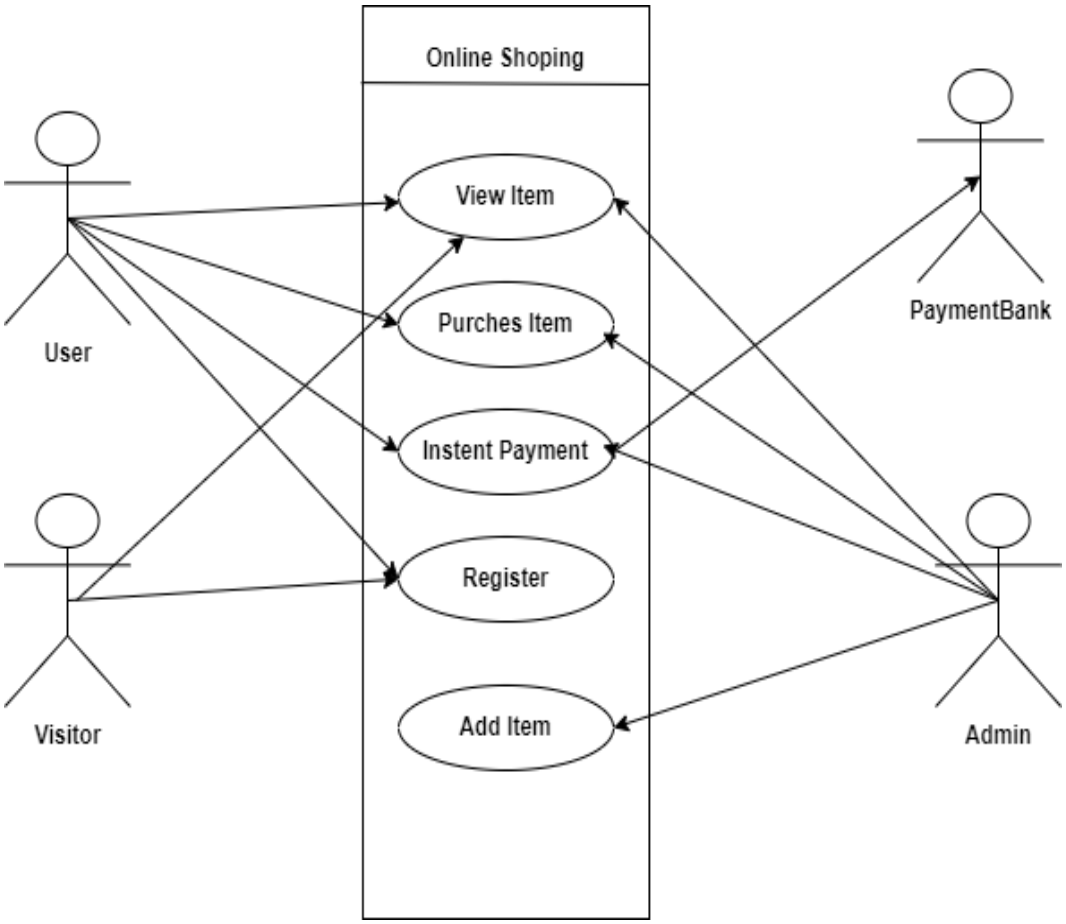


Actor

Use Case

An actor represents a user or another system will interact with the system that you are modelling. A use case is an external view of the system that represents some action that might perform in order to complete a task.

Use Case Diagram



1.3. Class Diagram

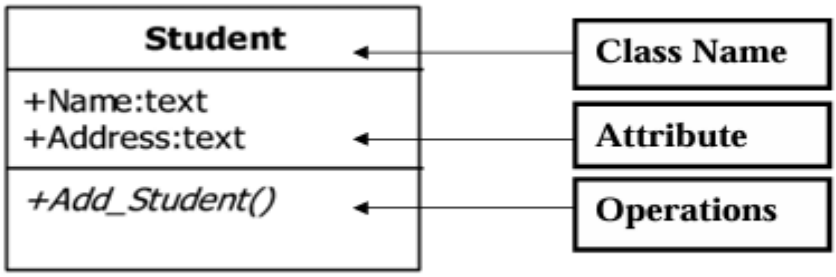
Class diagrams are widely used to describe the types of objects used in system and their relationship. Class diagram models class structure and contents using design elements such as classes and packages and objects.

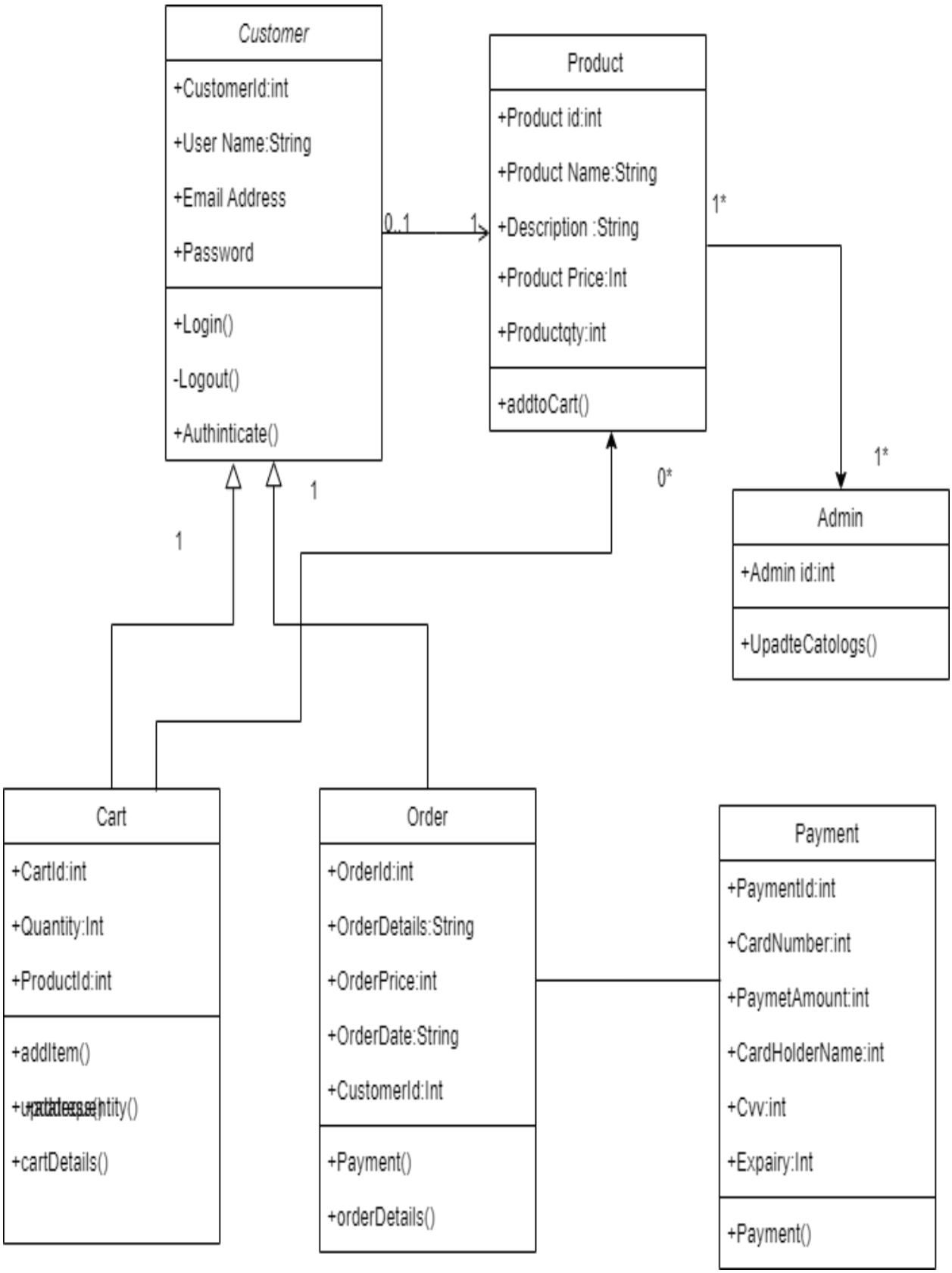
Class diagram describes 3 different perspectives when designing a system. These perspectives become evident as the diagram is created and help solidify the design.

Classes are composed of 3 things:

1. Class name.
2. Attributes and
3. Operations.

For Example Diagram:




Customer
+CustomerId:int
+User Name:String
+Email Address
+Password
+Login()
-Logout()
+Authinticate()
0..1
1
Product
+Product id:int
+Product Name:String
+Description :String
+Product Price:Int
+Productqty:int
+addtoCart()
1*
1*
0*
1
1
Admin
+Admin id:int
+UpadteCatologs()
Cart
+CartId:int
+Quantity:Int
+ProductId:int
+addItem()
+cartDetails()
Order
+OrderId:int
+OrderDetails:String
+OrderPrice:int
+OrderDate:String
+CustomerId:Int
+Payment()
+orderDetails()
Payment
+PaymentId:int
+CardNumber:int
+PaymetAmount:int
+CardHolderName:int
+Cvv:int
+Expairy:Int
+Payment()

1.4. Object Diagram

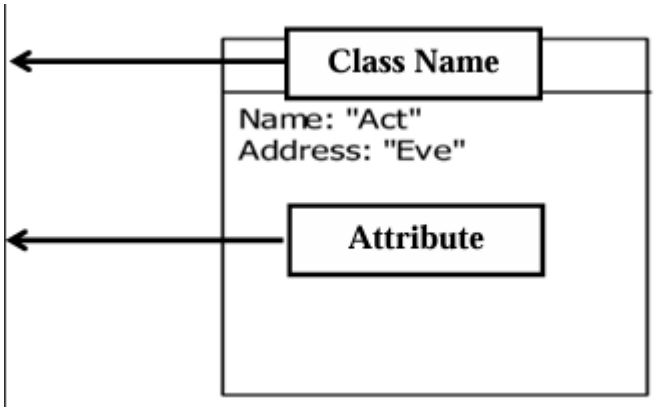
Object diagram are same as that of class diagram. Instead they contain the values in place of data types.

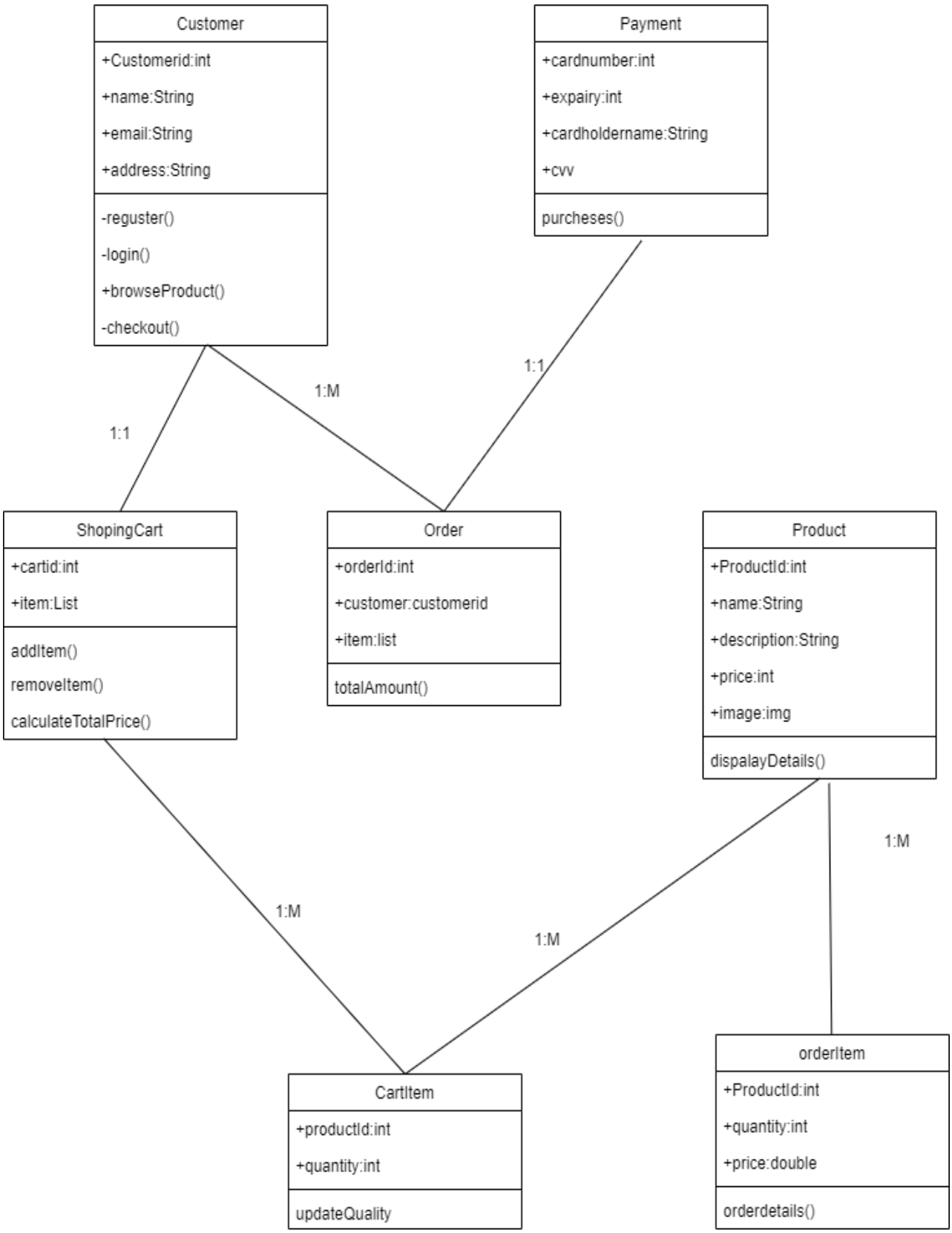
The object diagram describes 3 different perspectives when designing a system. This perspective becomes evident as the system is created & helps solidify the design.

They are composed of 2 things:

1. Class name.
2. Attributes.

For Example Diagram:




Customer
+Customerid:int
+name:String
+email:String
+address:String
-reguster()
-login()
+browseProduct()
-checkout()
Payment
+cardnumber:int
+expairy:int
+cardholdername:String
+cvv
purcheses()
1:1
1:M
1:1
ShopingCart
+cartid:int
+item:List
addItem()
removeItem()
calculateTotalPrice()
Order
+orderId:int
+customer:customerid
+item:list
totalAmount()
Product
+ProductId:int
+name:String
+description:String
+price:int
+image:img
dispalayDetails()
1:M
1:M
1:M
CartItem
+productId:int
+quantity:int
updateQuality
orderItem
+ProductId:int
+quantity:int
+price:double
orderdetails()

1.5. Activity Diagram

An activity diagram visually present flow of control in a system. Activity Diagrams are a type of behavioural diagram meaning they illustrate the dynamic aspects of a software system, showcasing the behaviour, response to any stimuli and undergo state changes during runtime. Activity Diagrams help in modelling sequential and concurrent activities together. In simple terms, we depict the workflow visually using Activity Diagrams. Activity Diagrams put more weight on the condition of the workflow and the sequence in which it happens. Depiction of what causes a particular event is done using Activity Diagram.

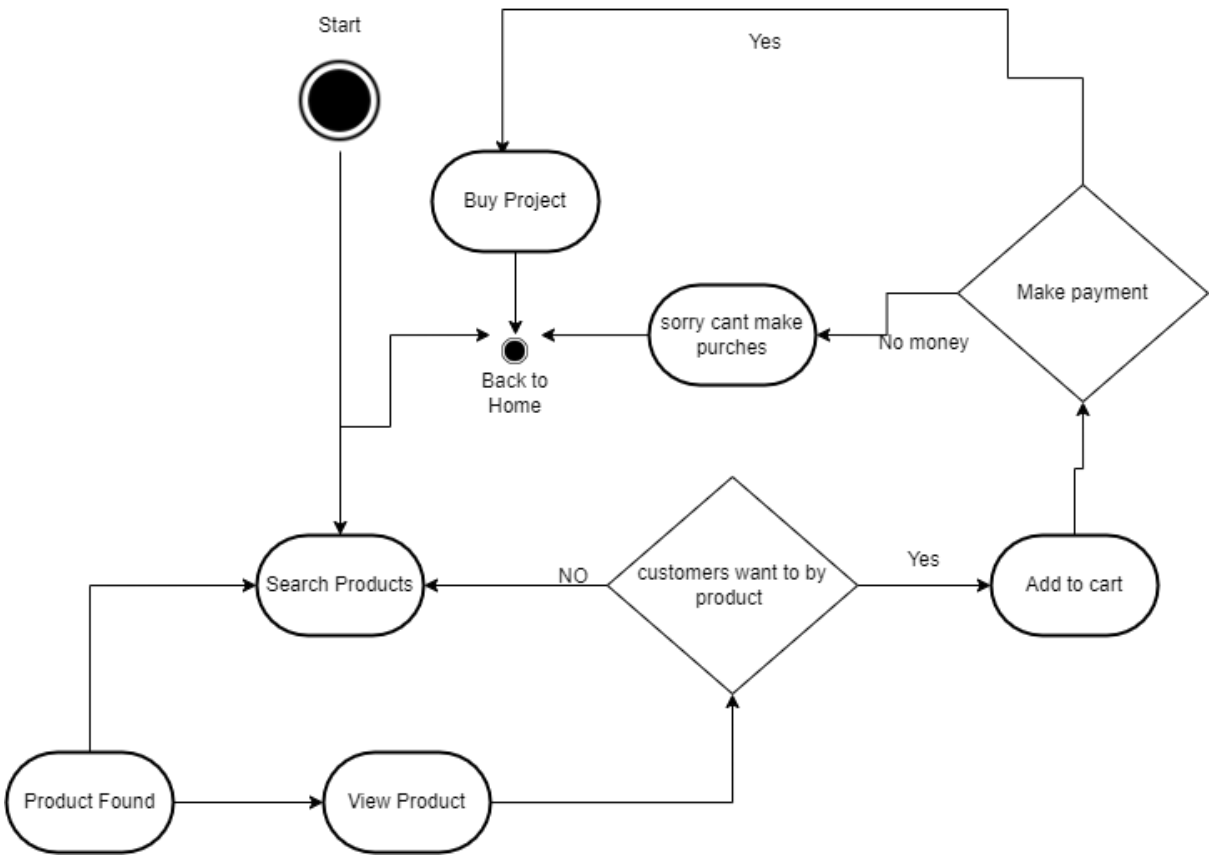
Use of Activity Diagram

Activity Diagrams are used to model and visualize dynamic aspects of an application/software. They are extremely useful to understand the control and workflows of an application/software. Some common features of Activity Diagram are:

- They help in dynamic Modelling of a system.
- They illustrate the various steps involved in a UML use case.
- They help to model software components like functions, methods, etc.
- They help to document concurrent activities easily.
- They also highlight constraints, conditions and logic behind algorithms.
- They help in depicting the dynamic aspects of user interactions in the software requirement analysis phase.
- steps for an easy construction of Activity Diagrams
  1. Identifying the initial and final states.
  2. Identifying the intermediate activities.
  3. Identifying the conditions or constraints.
  4. Applying appropriate notations and symbols.

  What makes Activity Diagrams different from Flowcharts?
  Many times, Activity Diagrams and flowcharts often get mixed up. Activity Diagrams are very similar to a flowchart but still there is a line between the two which differentiate them. The main difference between an activity diagram and flowchart is relation with UML, activity diagrams are only associated with UML, meanwhile flowchart is associated with the entire programming. Another key difference is what they represent. Activity Diagrams are used to model dynamic aspects and illustrate the workflow or control of a system. On the other hand, Flowcharts often illustrate a solution to any given problem and are widely used for showcasing algorithms, decisions or structures.

Symbol	Name	Description
	Initial Node	Starting point of any activity.
	Action State	Represents any action/task that will take place.
	Control Flow	Depict the workflow or control of an activity.
	Decision Node	There are multiple options available here. Two or more conditions can be considered here.
	Fork	Depicts that two process execute or run either concurrently or in parallel at this location.
	Join	Combination of results from two concurrent activities.
	Final Node/End State	Last state of an activity diagram. Activity ends here.


Start
Yes
Buy Project
Make payment
sorry cant make purches
No money
Back to Home
Search Products
NO
customers want to by product
Yes
Add to cart
Product Found
View Product

1.6. State Diagram

As the name suggests, a State Diagram is used to represent states. States in a software/application can be condition of the system at finite instances of time. State Diagrams are also behavioural diagrams like Activity Diagrams. They represent the behaviour using finite state transitions. State Diagrams are also known as "State Charts" or "State Machine Diagrams". State Diagrams are used to model the dynamic behaviour or a class or function in response to time and dynamic external stimuli

Use of State Diagram

- To get a clear understanding of software/application's behaviour.
- Each state depicted by State Diagrams shows information about the object.
- Depicts the execution flow from one state to another.
- Gives a thorough understanding of an object state by visualizing the object state from its creation to the end/termination.
- It documents an overall representation of an interactive system and the entities inside the system.

Types of States in State Diagrams

1. Simple State: Does not constitute any substructure.

2. Composite State: Consists the nested states/substates under the condition that it has only one initial state and final state. It can be nested to any level.

3. Submachine State: Semantically identical to the composite state, but gives the feature of reusability.

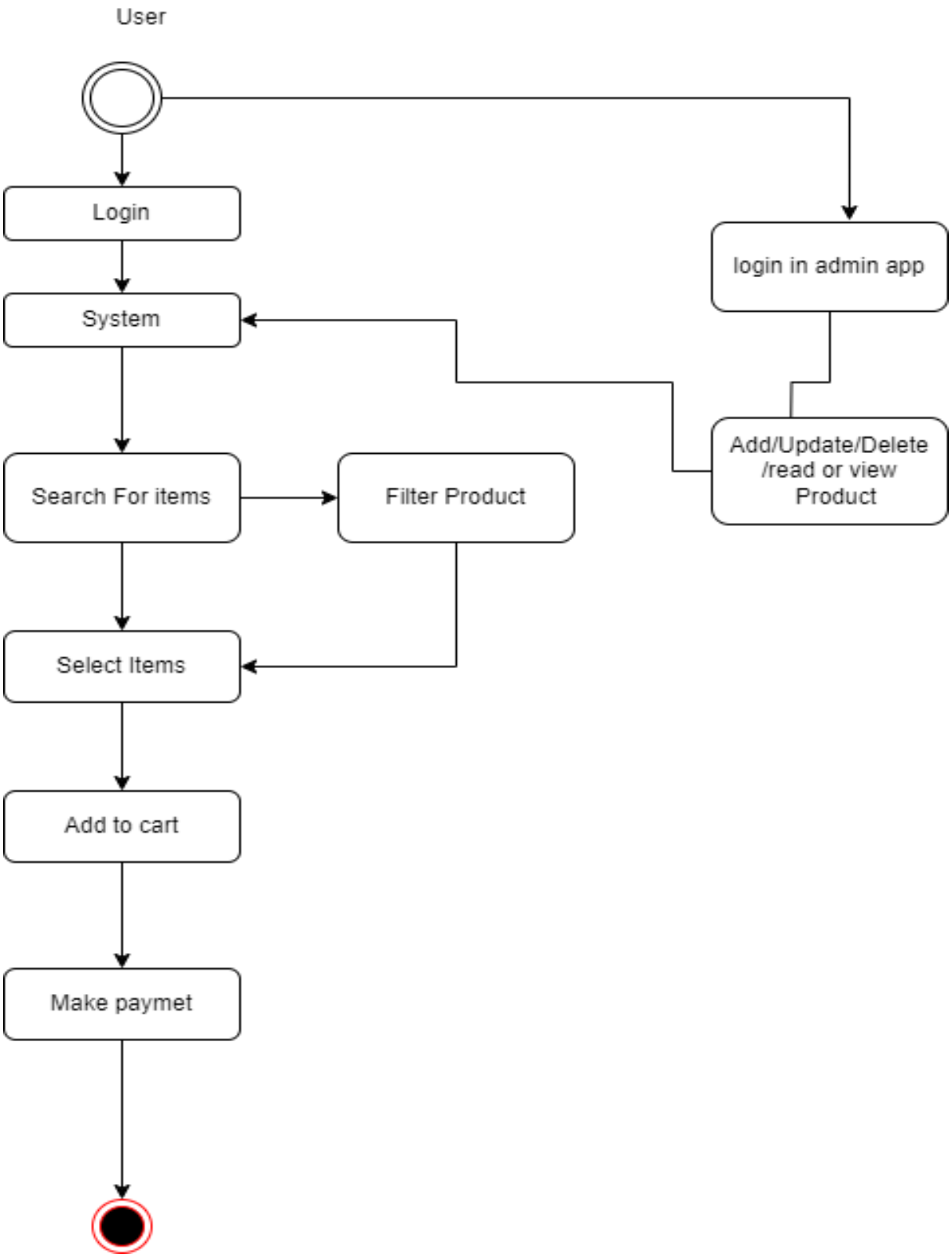
How to Prepare a State Diagram?

State Diagrams or State Charts can be easily prepared following the below steps:

1. Identifying the System
   Initial phase is also understanding our system, whether it's a software/application/machine or process etc. By identifying the system, we can then figure out what different situations or conditions might be possible.

2. Identifying Initial and Final States
   After system identification, we can then find out the starting and final states. We plot out the beginning and the ending of our system.
3. Identifying Possible States
   We then find out the possible states, these are all the different situations/conditions that the system can be in.
4. Labelling Triggering States
   We label the triggers, i.e. find out what cause our system to move from one state to another.
5. Drawing the diagram making use of appropriate notations
   Finally, after gathering all the information using the above steps, we can prepare State Diagrams using appropriate notations.

Symbol	Name	Description
	Initial Node	Starting point of any activity.
	Action State	Represents any action/task that will take place.
	Control Flow	Depict the workflow or control of an activity.
	Decision Node	There are multiple options available here. Two or more conditions can be considered here.
	Fork	Depicts that two process execute or run either concurrently or in parallel at this location.
	Join	Combination of results from two concurrent activities.
	Final Node/End State	Last state of an activity diagram. Activity ends here.


User
Login
System
Search For items
Filter Product
Select Items
Add to cart
Make paymet
login in admin app
Add/Update/Delete /read or view Product

1.7. Sequence Diagram

Sequence diagram demonstrates the behavior of the objects in a use- case by describing the objects and the messages they pass. The diagrams are read left to right & descending

How to create Sequence Diagrams?

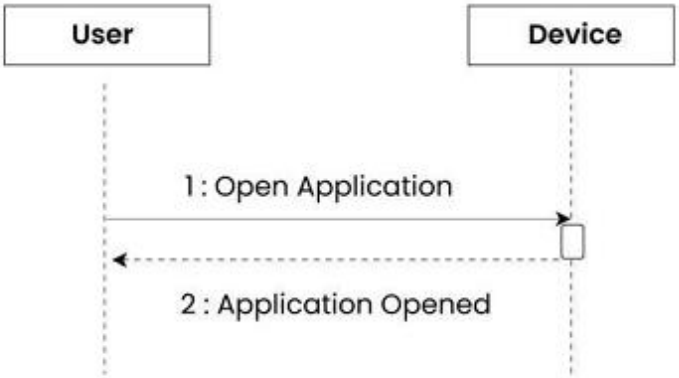
Various steps are to be followed when creating a Sequence Diagram. These diagrams are put forth typically during the design phase of the SDLC (Software Development Life Cycle) to demonstrate how different components or objects interact over time. The detailed steps to create a Sequence Diagram are mentioned below.

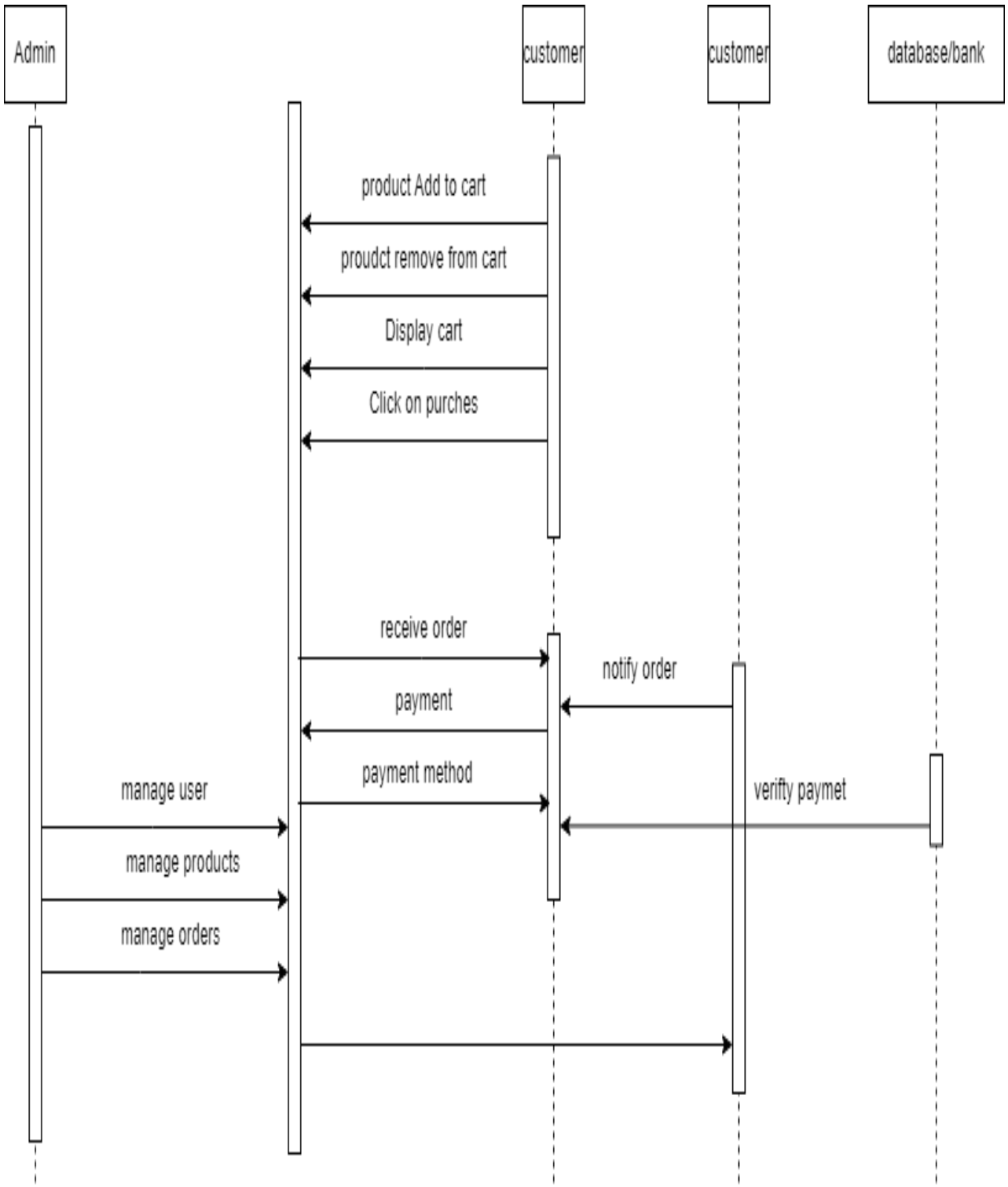
1. **Identifying the Scenario**: Initial step in creating a Sequence Diagram is to understand the use-case or specific scenario which has to be demonstrated in the Sequence Diagram. This could be any specific interaction between objects or the flow of messages in a particular process.
2. **Listing the Participants**: Participants, i.e. objects or actors involved in the scenario should be identified in the second step. Participants can be users, systems or external entities.
3. **Defining Lifelines:** o A vertical dashed line should be drawn for each participant, demonstrating the lifeline of each object over time. The timeline represents the existence of an object during the interaction.
4. **Arranging Lifelines**: Lifelines should be positioned horizontally in order of their involvements in the interaction. This helps in visualizing the message flow between participants.
5. **Adding Activation Bars**: o An Activation Bar should be drawn on the lifeline of the sending participant. This Activation Bar represents the duration of time during which the participant is actively processing the message.
6. **Drawing Messages**: o Arrows should be used to demonstrate the messages between participants. Messages can flow horizontally between lifelines, indicating the communication between objects.
7. **Including Return Messages**: o If response messages are sent, then they should be illustrated by drawing a dashed arrow returning to the original sender.
8. **Indicating Timing and Order**: o Numbers should be used to indicate the order of messages in the sequence. Vertical dashed lines can be used to represent occurrences of events or the passage of time.
9. **Indicating Conditions and Loops**: o Conditions (like if-else statements) and loops in the interaction, should be represented using combined fragments. This ensures complexity along with the detailed control flow in Sequence Diagram.
10. **Considering Parallel Execution**: o For any parallel activities taking place, they should be demonstrated by drawing parallel vertical dashed lines and placing the messages accordingly.
11. **Reviewing and Refining**: o Sequence Diagrams must be reviewed in order to ensure their correctness. Refining can be then done as per the need.

12. **Adding Annotations and Comments**: o Any additional information, annotations or comments that provide clarity should be included.
13. **Documenting Assumptions and Constraints**: o If there occur any assumptions or constraints related to the interaction, they should be documented alongside the diagram.
14. **Tools**: o Using UML Modelling tools or diagramming software to create a neat and professional Sequence Diagram. These tools often provide with various features which make creating, editing simple and easy to use and understand.

Components of Sequence Diagram:-

1		Represent object activation.
2		Represents the objects of case study.
3		Represents life of objects.
4		Represents end of objects.




Admin
customer
customer
database/bank
product Add to cart
proudct remove from cart
Display cart
Click on purches
receive order
notify order
payment
payment method
verifty paymet
manage user
manage products
manage orders

1.8. Package Diagram

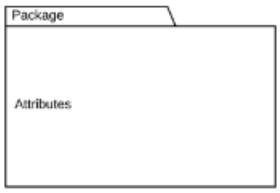
- In addition to standard UML dependency relationship there are two special types decencies defined between packages:
  Package Import
  Package Merge
- A package imports a relationship between an import namespace and a package indicating that importing namespace adds the names of members an unlabeled dependency between two packages an interpreted as a package import relationship. In this relationship elements within the target package will be imported in source package.
- A package merge is a directed relationship between two packages that indicates that the contents of two packages that are to be combined. It is very similar to generalization.

Elements of package Diagram:-

1. Package: It is a general purpose mechanism for organizing model contains elements is designed diagrams into groups. It provides an encapsulated namespace within which all names must be unique.
2. Class: It is representation of objects that reflects their structure and behavior of system.
3. Interface:It is specification of behavior. By implementing interface classes are required to support the behavior
4. Object: It is instance of class. It often used in analysis to represent numerous artifacts and item that exist.
5. Table: It is stereotyped class.

Basic components of a package diagram

The makeup of a package diagram is relatively simple. Each diagram includes only two symbols:

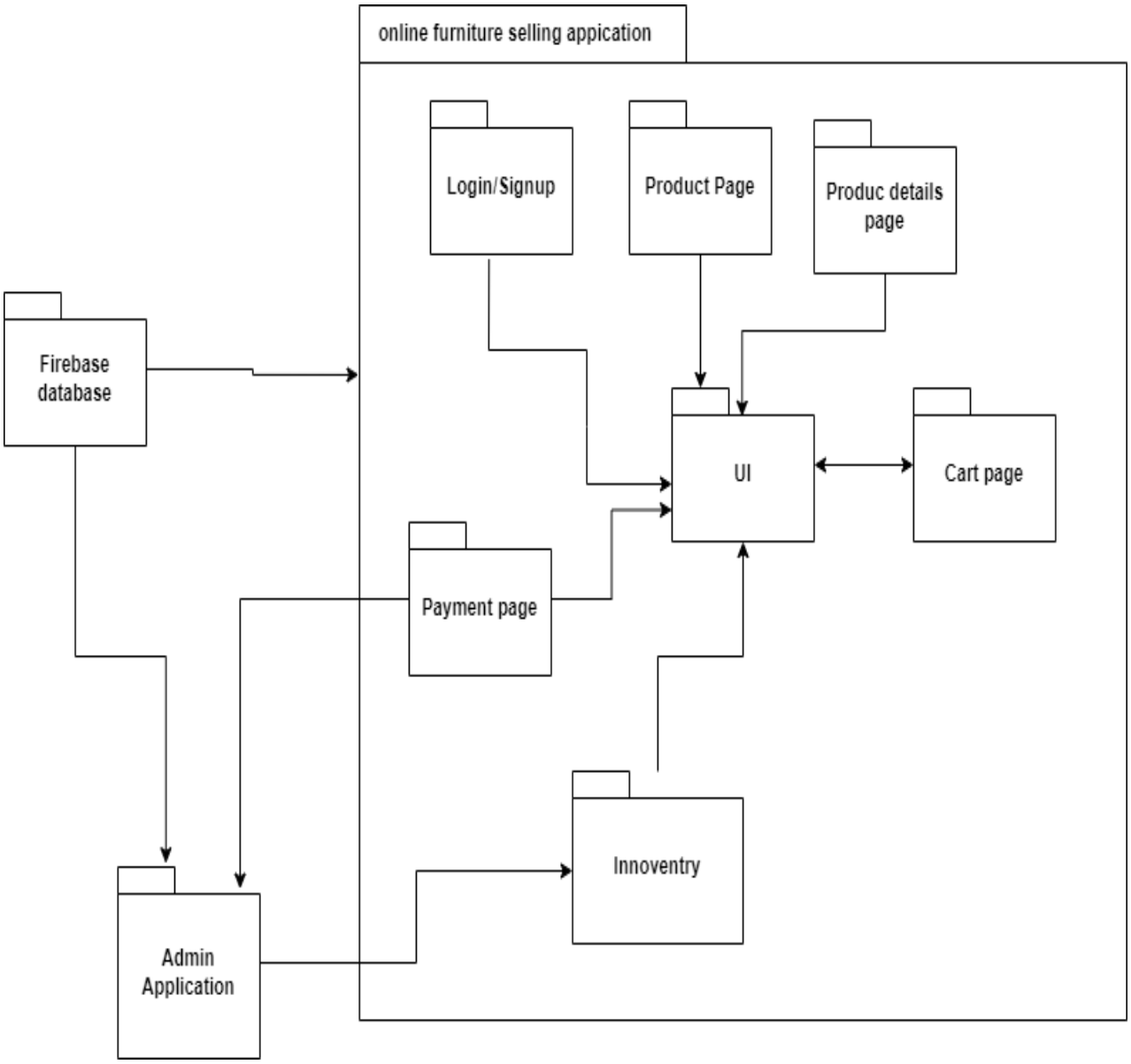
Symbol Image	Symbol Name	Description
	Package	Groups common elements based on data, behavior, or user interaction
	Dependency	Depicts the relationship between one element (package, named element, etc) and another

These symbols can be used in a variety of ways to represent different iterations of packages, dependencies, and other elements within a system. Here are the basic components you'll find within a package diagram:

- **Package**: A namespace used to group together logically related elements within a system. Each element contained within the package should be a packageable element and have a unique name.
- **Packageable element**: A named element, possibly owned directly by a package. These can include events, components, use cases, and packages themselves. Packageable elements can also be rendered as a rectangle within a package, labeled with the appropriate name.
- **Dependencies**: A visual representation of how one element (or set of elements) depends on or influences another. Dependencies are divided into two groups: access and import dependencies. (See next section for more info.)

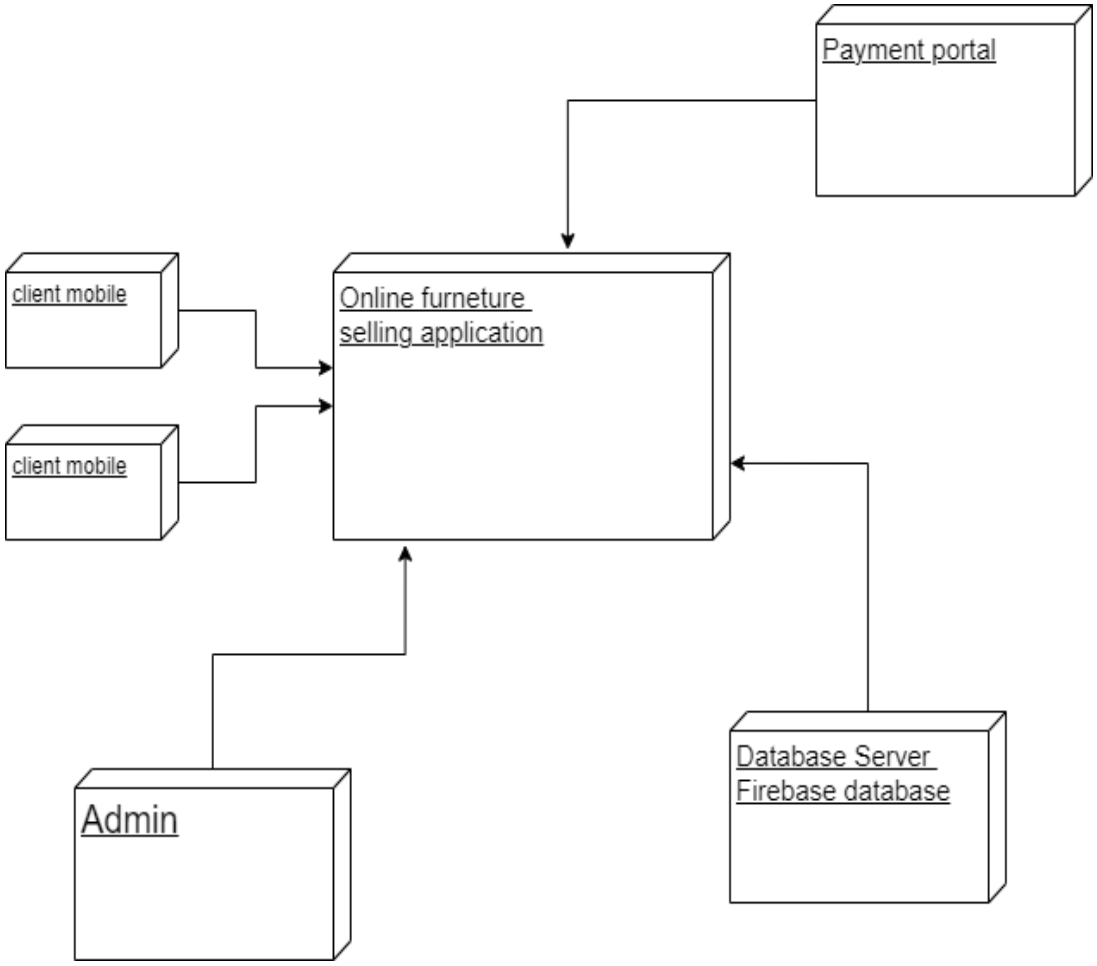
- **Package import:** A directed relationship between and importing namespace and an imported package. This type of directed relationship adds the names of the members of the imported package to its own namespace

- **Package merge**: A directed relationship in which the contents of one package are extended by the contents of another. Essentially, the content of two packages are combined to produce a new package.



1.9. Deployment Diagram

- The deployment diagram contains nodes & connections.
- A node usually represents a piece of hardware in the system.
- A connection depicts the communication path used by the hardware to communicate & usually indicates a method such as TCP/IP.



3. Database table

A. Table name: users table
Description: Stores the information of the account created by the user.

Field Name	Input Type
Name	String
Email	String
Password	String

B.
2) Table name: products
Description: Stores the products details.

Field Name	Input Type
Product id	String
Name	String
Description	String
Price	Number
Offer	Boolean
Category	String
Image	String

C. Table name: user orders
Description: Stores the information of the user's order details.

Field Name	Input Type
Address	String
Card Info	
Card Holder Name	String
Card Number	Number
Cvv	Number
ExpairyDate	Number
Order Date	DateTimeStamp
Product	
Name	String
Product Id	String
Product Quantity	Number
Total Amount	Number

D. Table name: Cart details
Description: Stores the information Cart

Field Name	Data Type
Category	String
Quantity	Number

E. Table name: user payments
Description: Stores the information of the user's payments details.

Field Name	Data Type
Payment id	String
Payment details	String
Amount	Number
Date	TimeStamp

4. Project documentation

4.1 Source Code

Client_Side

File Name :Main.dart

```
import 'package:firebase_core/firebase_core.dart';
import 'package:flutter/material.dart';
import 'package:get/get.dart';
import 'package:ofsa_client_main/controller/home_controller.dart';
import 'package:ofsa_client_main/pages/cart_page.dart';
import 'package:ofsa_client_main/pages/home_page.dart';
import 'package:ofsa_client_main/pages/login.dart';
import 'package:ofsa_client_main/pages/payment_page.dart';
// import '../dump/razorpay_page.dart';
import 'package:provider/provider.dart';
import 'controller/purchase_controller.dart';
import 'firebase_options.dart';
import 'themes/theme_provider.dart'; // Import the ThemeProvider

Future<void> main() async {
  WidgetsFlutterBinding.ensureInitialized();
  try {
    // Initialize Firebase
    await Firebase.initializeApp(options: DefaultFirebaseOptions.android);
  } catch (e) {
    print("Error initializing Firebase: $e");
  }
  Get.put(HomeController());
  Get.put(PurchaseController());
  runApp(const MyApp());
}

class MyApp extends StatelessWidget {
  const MyApp({super.key});
  @override
  Widget build(BuildContext context) {
    return ChangeNotifierProvider(
      create: (_) => ThemeProvider(),
      child: Consumer<ThemeProvider>(
        builder: (context, themeProvider, child) {
          return GetMaterialApp(
            debugShowCheckedModeBanner: false,
            title: 'Client App',
            theme: themeProvider.themeData, // Apply the theme dynamically
```

```
          home:  Login(),
        );
      },
    ),
  );
 }
}
```

File Name :Login.dart

```
import 'package:ofsa_client_main/pages/forgot_password.dart';
import 'package:ofsa_client_main/pages/home_page.dart';
import 'package:ofsa_client_main/service/auth.dart';
import 'package:ofsa_client_main/pages/signup.dart';
import 'package:firebase_auth/firebase_auth.dart';
import 'package:flutter/material.dart';

class LogIn extends StatefulWidget {
  const LogIn({super.key});

  @override
  State<LogIn> createState() => _LogInState();
}

class _LogInState extends State<LogIn> {
  String email = "", password = "";

  TextEditingController mailcontroller = new TextEditingController();
  TextEditingController passwordcontroller = new TextEditingController();

  final _formkey = GlobalKey<FormState>();

  userLogin() async {
    try {
      await FirebaseAuth.instance
          .signInWithEmailAndPassword(email: email, password: password);
      Navigator.push(context, MaterialPageRoute(builder: (context) => HomePage()));
    } on FirebaseAuthException catch (e) {
      if (e.code == 'user-not-found') {
        ScaffoldMessenger.of(context).showSnackBar(SnackBar(
            backgroundColor: Colors.orangeAccent,
            content: Text(
              "No User Found for that Email",
              style: TextStyle(fontSize: 18.0),
            )));
      } else if (e.code == 'wrong-password') {
```

```
      ScaffoldMessenger.of(context).showSnackBar(SnackBar(
        backgroundColor: Colors.orangeAccent,
        content: Text(
          "Wrong Password Provided by User",
          style: TextStyle(fontSize: 18.0),
        )));
    }
  }
}

@override
Widget build(BuildContext context) {
  return Scaffold(

    backgroundColor: Colors.white,
    body: Container(

      child: Column(

        children: [
          Container(

            width: MediaQuery.of(context).size.width,
            child: Image.asset(
              "lib/image/car.png",
              fit: BoxFit.cover,
            )
          ),
          SizedBox(
            height: 50.0,
          ),
          Padding(
            padding: const EdgeInsets.only(left: 20.0, right: 20.0),
            child: Form(
              key: _formkey,
              child: Column(
                children: [
                  Container(
                    padding:
                    EdgeInsets.symmetric(vertical: 2.0, horizontal: 30.0),
                    decoration: BoxDecoration(
                      color: Color(0xFFedf0f8),
                      borderRadius: BorderRadius.circular(30)),
                    child: TextFormField(
                      validator: (value) {
                        if (value == null || value.isEmpty) {
```

```
        return 'Please Enter E-mail';
      }
      return null;
    },
    controller: mailcontroller,
    decoration: InputDecoration(
        border: InputBorder.none,
        hintText: "Email",
        hintStyle: TextStyle(
            color: Color(0xFFb2b7bf), fontSize: 18.0)),
  ),
),
SizedBox(
  height: 30.0,
),
Container(
  padding:
  EdgeInsets.symmetric(vertical: 2.0, horizontal: 30.0),
  decoration: BoxDecoration(
      color: Color(0xFFedf0f8),
      borderRadius: BorderRadius.circular(30)),
  child: TextFormField(
    controller: passwordcontroller,
    validator: (value) {
      if (value == null || value.isEmpty) {
        return 'Please Enter Password';
      }
      return null;
    },
    decoration: InputDecoration(
        border: InputBorder.none,
        hintText: "Password",
        hintStyle: TextStyle(
            color: Color(0xFFb2b7bf), fontSize: 18.0)),
    obscureText: true,   ),
),
SizedBox(
  height: 30.0,
),
GestureDetector(
  onTap: (){
    if(_formkey.currentState!.validate()){
      setState(() {
        email= mailcontroller.text;
        password=passwordcontroller.text;
      });
```

```
                }
                userLogin();
              },
              child: Container(
                width: MediaQuery.of(context).size.width,
                padding: EdgeInsets.symmetric(
                  vertical: 13.0, horizontal: 30.0),
                decoration: BoxDecoration(
                  color: Color(0xFF273671),
                  borderRadius: BorderRadius.circular(30)),
                child: Center(
                  child: Text(
                    "Sign In",
                    style: TextStyle(
                      color: Colors.white,
                      fontSize: 22.0,
                      fontWeight: FontWeight.w500),
                  ))),
            ),
          ],
        ),
      ),
    ),
    SizedBox(
      height: 20.0,
    ),
    GestureDetector(
      onTap: (){
        Navigator.push(context, MaterialPageRoute(builder: (context)=>
ForgotPassword()));
      },
      child: Text("Forgot Password?",
        style: TextStyle(
          color: Color(0xFF8c8e98),
          fontSize: 18.0,
          fontWeight: FontWeight.w500)),
    ),
    SizedBox(
      height: 40.0,
    ),
    Text(
      "or LogIn with",
      style: TextStyle(
        color: Color(0xFF273671),
        fontSize: 22.0,
        fontWeight: FontWeight.w500),
```

```
),
SizedBox(
  height: 30.0,
),
Row(
  mainAxisAlignment: MainAxisAlignment.center,
  children: [
    GestureDetector(
      onTap: (){
        AuthMethods().signInWithGoogle(context);
      },
      child: Image.asset(
        "lib/image/google.png",
        height: 45,
        width: 45,
        fit: BoxFit.cover,
      ),
    ),

  ],
),
SizedBox(
  height: 40.0,
),
Row(
  mainAxisAlignment: MainAxisAlignment.center,
  children: [
    Text("Don't have an account?",
        style: TextStyle(
            color: Color(0xFF8c8e98),
            fontSize: 18.0,
            fontWeight: FontWeight.w500)),
    SizedBox(
      width: 5.0,
    ),
    GestureDetector(
      onTap: () {
        Navigator.push(context,
            MaterialPageRoute(builder: (context) => SignUp()));
      },
      child: Text(
        "SignUp",
        style: TextStyle(
            color: Color(0xFF273671),
            fontSize: 20.0,
            fontWeight: FontWeight.w500),
```

```
                ),
              ),
            ],
          )
        ],
      ),
    ),
  );
 }
}
```

File Name : Signup.dart

```
import 'package:ofsa_client_main/pages/home_page.dart';
import 'package:ofsa_client_main/pages/login.dart';
import 'package:firebase_auth/firebase_auth.dart';
import 'package:flutter/material.dart';

class SignUp extends StatefulWidget {
  const SignUp({super.key});

  @override
  State<SignUp> createState() => _SignUpState();
}

class _SignUpState extends State<SignUp> {
  String email = "", password = "", name = "";
  TextEditingController namecontroller = new TextEditingController();
  TextEditingController passwordcontroller = new TextEditingController();
  TextEditingController mailcontroller = new TextEditingController();

  final _formkey = GlobalKey<FormState>();

  registration() async {
    if (password != null && namecontroller.text != "" && mailcontroller.text != "") {
      try {
        UserCredential userCredential = await FirebaseAuth.instance
            .createUserWithEmailAndPassword(email: email, password: password);
        ScaffoldMessenger.of(context).showSnackBar(SnackBar(
            content: Text(
              "Registered Successfully",
              style: TextStyle(fontSize: 20.0),
            )));
        // ignore: use_build_context_synchronously
        Navigator.push(
            context, MaterialPageRoute(builder: (context) => HomePage()));
```

```
    } on FirebaseAuthException catch (e) {
      if (e.code == 'weak-password') {
        ScaffoldMessenger.of(context).showSnackBar(SnackBar(
            backgroundColor: Colors.orangeAccent,
            content: Text(
              "Password Provided is too Weak",
              style: TextStyle(fontSize: 18.0),
            )));
      } else if (e.code == "email-already-in-use") {
        ScaffoldMessenger.of(context).showSnackBar(SnackBar(
            backgroundColor: Colors.orangeAccent,
            content: Text(
              "Account Already exists",
              style: TextStyle(fontSize: 18.0),
            )));
      }
    }
  }
}

@override
Widget build(BuildContext context) {
  return Scaffold(
    backgroundColor: Colors.white,
    body: Stack(
      children: [
        Column(
          children: [
            Container(
                width: MediaQuery.of(context).size.width,
                child: Image.asset(
                  "lib/image/car.png",
                  fit: BoxFit.cover,
                )),
            SizedBox(
              height: 30.0,
            ),
            Padding(
              padding: const EdgeInsets.only(left: 20.0, right: 20.0),
              child: Form(
                key: _formkey,
                child: Column(
                  children: [
                    Container(
                      padding: EdgeInsets.symmetric(vertical: 2.0, horizontal: 30.0),
                      decoration: BoxDecoration(
```

```
        color: Color(0xFFedf0f8),
        borderRadius: BorderRadius.circular(30)),
    child: TextFormField(
      validator: (value) {
        if (value == null || value.isEmpty) {
          return 'Please Enter Name';
        }
        return null;
      },
      controller: namecontroller,
      decoration: InputDecoration(
          border: InputBorder.none,
          hintText: "Name",
          hintStyle: TextStyle(
              color: Color(0xFFb2b7bf), fontSize: 18.0)),
    ),
  ),
  SizedBox(
    height: 30.0,
  ),
  Container(
    padding: EdgeInsets.symmetric(vertical: 2.0, horizontal: 30.0),
    decoration: BoxDecoration(
        color: Color(0xFFedf0f8),
        borderRadius: BorderRadius.circular(30)),
    child: TextFormField(
      validator: (value) {
        if (value == null || value.isEmpty) {
          return 'Please Enter Email';
        }
        return null;
      },
      controller: mailcontroller,
      decoration: InputDecoration(
          border: InputBorder.none,
          hintText: "Email",
          hintStyle: TextStyle(
              color: Color(0xFFb2b7bf), fontSize: 18.0)),
    ),
  ),
  SizedBox(
    height: 30.0,
  ),
  Container(
    padding: EdgeInsets.symmetric(vertical: 2.0, horizontal: 30.0),
    decoration: BoxDecoration(
```

```
        color: Color(0xFFedf0f8),
        borderRadius: BorderRadius.circular(30)),
      child: TextFormField(
        validator: (value) {
          if (value == null || value.isEmpty) {
            return 'Please Enter Password';
          }
          return null;
        },
        controller: passwordcontroller,
        decoration: InputDecoration(
            border: InputBorder.none,
            hintText: "Password",
            hintStyle: TextStyle(
                color: Color(0xFFb2b7bf), fontSize: 18.0)),
        obscureText: true,
      ),
    ),
    SizedBox(
      height: 30.0,
    ),
    GestureDetector(
      onTap: () async {
        if (_formkey.currentState!.validate()) {
          setState(() {
            email = mailcontroller.text;
            name = namecontroller.text;
            password = passwordcontroller.text;
          });
        }
        await registration();
      },
      child: Container(
        width: MediaQuery.of(context).size.width,
        padding: EdgeInsets.symmetric(
            vertical: 13.0, horizontal: 30.0),
        decoration: BoxDecoration(
            color: Color(0xFF273671),
            borderRadius: BorderRadius.circular(30)),
        child: Center(
            child: Text(
              "Sign Up",
              style: TextStyle(
                  color: Colors.white,
                  fontSize: 22.0,
                  fontWeight: FontWeight.w500),
```

```
                )),
            ),
          ),
        ],
      ),
    ),
  ),
  SizedBox(
    height: 40.0,
  ),
  Text(
    "or LogIn with",
    style: TextStyle(
        color: Color(0xFF273671),
        fontSize: 22.0,
        fontWeight: FontWeight.w500),
  ),
  SizedBox(
    height: 30.0,
  ),
  Row(
    mainAxisAlignment: MainAxisAlignment.center,
    children: [
      Image.asset(
        "lib/image/google.png",
        height: 45,
        width: 45,
        fit: BoxFit.cover,
      ),
    ],
  ),
  SizedBox(
    height: 40.0,
  ),
  Row(
    mainAxisAlignment: MainAxisAlignment.center,
    children: [
      Text("Already have an account?",
          style: TextStyle(
              color: Color(0xFF8c8e98),
              fontSize: 18.0,
              fontWeight: FontWeight.w500)),
      SizedBox(
        width: 5.0,
      ),
      GestureDetector(
```

```
                onTap: () {
                  Navigator.push(context,
                      MaterialPageRoute(builder: (context) => LogIn()));
                },
                child: Text(
                  "LogIn",
                  style: TextStyle(
                      color: Color(0xFF273671),
                      fontSize: 20.0,
                      fontWeight: FontWeight.w500),
                ),
              ),
            ],
          ),
        ],
      ),
      Positioned(
        top: 20,
        left: 20,
        child: SafeArea(
          child:Opacity(
            opacity: 0.4,
            child: Container(
            decoration: BoxDecoration(
              color: Theme.of(context).colorScheme.secondary,
              shape: BoxShape.circle,
            ),
            child: IconButton(
              icon: const Icon(Icons.arrow_back_ios_rounded),
              onPressed: () => Navigator.pop(context),
            ),
          ),
        ),
      ),
      ),
    ],
  ),
);
}
}
```

File Name :Home_page.dart

```
import 'package:flutter/material.dart';
import 'package:get/get.dart';
import 'package:ofsa_client_main/controller/home_controller.dart';
```

```
import 'package:ofsa_client_main/pages/product_details_page.dart';
import 'package:ofsa_client_main/pages/setting_page.dart';
import 'package:ofsa_client_main/widgets/drop_down_btn.dart';
import 'package:ofsa_client_main/widgets/multi_select_drop_down.dart';
import 'package:ofsa_client_main/widgets/product_card.dart';
import 'package:provider/provider.dart';
import '../themes/theme_provider.dart'; // Import the ThemeProvider
import '../widgets/multi_select_popup.dart';

class HomePage extends StatelessWidget {
  const HomePage({super.key});

  @override
  Widget build(BuildContext context) {
    return GetBuilder<HomeController>(builder: (ctrl) {
      return RefreshIndicator(
        onRefresh: () async {
          await ctrl.fetchProducts();
        },
        child: Scaffold(
          appBar: AppBar(
            title: const Text('Online Furniture Selling App'),
            centerTitle: true,
          ),
          drawer: Drawer(
            child: ListView(
              padding: EdgeInsets.zero,
              children: <Widget>[
                DrawerHeader(
                  decoration: BoxDecoration(
                    color: Colors.blue,
                  ),
                  child: Text(
                    'Menu',
                    style: TextStyle(
                      color: Colors.white,
                      fontSize: 24,
                    ),
                  ),
                ),
                ListTile(
                  leading: Icon(Icons.home),
                  title: Text('Home'),
                  onTap: () {
                    Navigator.pop(context); // Close the drawer
                  },
```

```
        ),
        ListTile(
          leading: Icon(Icons.settings),
          title: Text('Settings'),
          onTap: () {
            Navigator.push(
              context,
              MaterialPageRoute(
                builder: (context) => SettingsPage(),
              ),
            );
          },
        ),
        ListTile(
          leading: Icon(Icons.contact_page),
          title: Text('Contact Us'),
          onTap: () {
            // Handle contact us navigation
          },
        ),
      ],
    ),
  ),
  body: Column(
    children: [
      SizedBox(
        height: 50,
        child: ListView.builder(
          scrollDirection: Axis.horizontal,
          itemCount: ctrl.productCategory.length,
          itemBuilder: (context, index) {
            final category = ctrl.productCategory[index];
            return InkWell(
              onTap: () {
                ctrl.filterByCategory(category.name ?? '');
              },
              child: Padding(
                padding: const EdgeInsets.all(6),
                child: Chip(label: Text(category.name ?? 'Error')),
              ),
            );
          },
        ),
      ),
      Row(
        children: [
```

```
            Flexible(
              child: DropDownBtn(
                items: ['Rs: Low to High', 'Rs: High to Low'],
                selectedItemsText: 'Sort',
                onSelected: (selected) {
                  ctrl.sortByPrice(
                      ascending: selected == 'Rs: Low to High');
                },
              ),
            ),
            Flexible(
              child: MultiSelectDropDown(
                items: ['Table', 'Chair', 'Bed', 'Desk', 'Cupboard', 'Sofa'], // Replace with your
brand names
                onSelectionChanged: (selectedItems) {
                  ctrl.filterByBrand(selectedItems); // Adjust the logic here as per your
requirement
                  // print(selectedItems);
                },
              ),
            ),

          ],
        ),
        Expanded(
          child: GridView.builder(
            gridDelegate: SliverGridDelegateWithFixedCrossAxisCount(
              crossAxisCount: 2,
              childAspectRatio: 0.9,
              crossAxisSpacing: 8,
              mainAxisSpacing: 8,
            ),
            itemCount: ctrl.productShowInUi.length,
            itemBuilder: (context, index) {
              final product = ctrl.productShowInUi[index];
              return ProductCard(
                name: product.name ?? 'No name',
                imageUrl: product.image ?? 'url',
                price: product.price ?? 00,
                offerTag: '30 % off',
                onTap: () {
                  Get.to(
                      () => const ProductDetailsPage(),
                    arguments: {'data': product},
                  );
                },
```

```
                  );
                },
              ),
            ),
          ],
        ),
      ),
    );
  });
 }
}
```

File Name :Product_details_page.dart

```
import 'package:flutter/material.dart';
import 'package:get/get.dart';
import 'package:ofsa_client_main/controller/purchase_controller.dart';
import 'package:ofsa_client_main/models/product/product.dart';
import 'package:ofsa_client_main/pages/cart_page.dart'; // Import the CartPage

class ProductDetailsPage extends StatelessWidget {
  const ProductDetailsPage({super.key});

  @override
  Widget build(BuildContext context) {
    final Product? product = Get.arguments['data'] as Product?;

    if (product == null) {
      return Scaffold(
        body: const Center(child: Text('Product not found')),
      );
    }

    return GetBuilder<PurchaseController>(builder: (ctrl) {
      return Scaffold(
        appBar: AppBar(
```

```
    title: const Text('Product Details'),
    centerTitle: true,
    actions: [
      IconButton(
        onPressed: () {
          // Add logout functionality here
        },
        icon: const Icon(Icons.logout),
      ),
    ],
  ),
  body: SingleChildScrollView(
    padding: const EdgeInsets.all(20.0),
    child: Column(
      crossAxisAlignment: CrossAxisAlignment.start,
      children: [
        ClipRRect(
          borderRadius: BorderRadius.circular(10),
          child: Image.network(
            product.image ?? 'https://example.com/fallback_image.png',
            fit: BoxFit.contain,
            width: double.infinity,
            height: 200,
          ),
        ),
        const SizedBox(height: 20),
        Text(
          product.name ?? 'Unknown Product',
          style: const TextStyle(
            fontSize: 24,
```

```
    fontWeight: FontWeight.bold,
  ),
),
const SizedBox(height: 20),
Text(
  product.description ?? 'No description available',
  style: const TextStyle(
    fontSize: 16,
    height: 1.5,
  ),
),
const SizedBox(height: 20),
Text(
  'Rs: ${product.price?.toStringAsFixed(2) ?? 'Unknown Price'}',
  style: const TextStyle(
    color: Colors.green,
    fontSize: 20,
    height: 1.5,
    fontWeight: FontWeight.bold,
  ),
),
const SizedBox(height: 20),
TextField(
  controller: ctrl.addressController,
  maxLines: 3,
  decoration: InputDecoration(
    border: OutlineInputBorder(
      borderRadius: BorderRadius.circular(12),
    ),
    labelText: 'Enter your billing address',
```

```
            ),
          ),
          const SizedBox(height: 20),
          SizedBox(
            width: double.infinity,
            child: ElevatedButton(
              style: ElevatedButton.styleFrom(
                padding: const EdgeInsets.symmetric(vertical: 15),
                backgroundColor: Colors.indigo,
              ),
              child: const Text(
                'Add to Cart',
                style: TextStyle(fontSize: 18, color: Colors.white),
              ),
              onPressed: () {
                ctrl.addToCart(product);
                Get.to(() => CartPage());
              },
            ),
          ),
        ],
      ),
    ),
  );
});
}
}
```

File Name:Cart_Page.dart

```
import 'package:flutter/material.dart';
import 'package:get/get.dart';
import 'package:ofsa_client_main/controller/purchase_controller.dart';
import 'package:ofsa_client_main/pages/payment_page.dart';

import 'home_page.dart';
// import '../../dump/razorpay_page.dart';

class CartPage extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return GetBuilder<PurchaseController>(
      builder: (ctrl) {
        return Scaffold(
          appBar: AppBar(
            title: const Text('Cart'),
            centerTitle: true,
            actions: [
              IconButton(
                onPressed: () {
                  // Optionally handle logout
                },
                icon: const Icon(Icons.logout),
              ),
            ],
          ),
          body: Column(
            crossAxisAlignment: CrossAxisAlignment.start,
            children: [
              Expanded(
                child: Obx(() {
                  if (ctrl.cartItems.isEmpty) {
                    return const Center(child: Text('Your cart is empty'));
                  }
                  return ListView.builder(
                    itemCount: ctrl.cartItems.length,
                    itemBuilder: (context, index) {
                      final item = ctrl.cartItems[index];
                      return ListTile(
                        leading: Image.network(
                          item.image ?? 'https://example.com/fallback_image.png',
                          width: 50,
                          height: 50,
                          fit: BoxFit.cover,
                        ),
```

```
                title: Text(item.name ?? 'Unknown Product'),
                subtitle: Text('Rs: ${item.price?.toStringAsFixed(2) ?? 'Unknown Price'}'),
                trailing: IconButton(
                  icon: const Icon(Icons.remove_circle),
                  onPressed: () {
                    ctrl.removeFromCart(item);
                  },
                ),
              );
            },
          );
        }),
      ),
      Padding(
        padding: const EdgeInsets.all(16.0),
        child: Text(
          'Total: Rs ${ctrl.totalAmount.toStringAsFixed(2)}',
          style: const TextStyle(
            fontSize: 18,
            fontWeight: FontWeight.bold,
          ),
        ),
      ),
      Padding(
        padding: const EdgeInsets.all(16.0),
        child: SizedBox(
          width: double.infinity,
          child: ElevatedButton(
            style: ElevatedButton.styleFrom(
              padding: const EdgeInsets.symmetric(vertical: 15),
              backgroundColor: Colors.indigo,
            ),
            child: const Text(
              'Proceed to Payment',
              style: TextStyle(fontSize: 18, color: Colors.white),
            ),
            onPressed: () {
              Navigator.push(context,MaterialPageRoute(builder: (context)=>HomePage(),));
              // ctrl.submitOrder();
              // Get.to(
              //       () => PaymentPage(),
              //   arguments: {'amount': ctrl.totalAmount}, // Passing total amount to payment
page
              // );
            },
          ),
```

```
            ),
          ),
        ],
      ),
    );
  },
);
}
}
```

File Name: payment_page.dart

```
import 'package:flutter/material.dart';
import 'package:flutter_credit_card/flutter_credit_card.dart';
import 'package:get/get.dart';
import 'package:ofsa_client_main/pages/dilivery_progress_page.dart';
import 'package:ofsa_client_main/pages/home_page.dart';
import '../controller/cart_controller.dart';
import '../widgets/my_button.dart'; // Import your custom button

class PaymentPage extends StatefulWidget {
  const PaymentPage({super.key});

  @override
  State<PaymentPage> createState() => _PaymentPageState();
}

class _PaymentPageState extends State<PaymentPage> {
  GlobalKey<FormState> formKey = GlobalKey<FormState>();
  String cardNumber = '';
  String expiryDate = '';
  String cardHolderName = '';
  String cvvCode = '';
  bool isCvvFocused = false;

  // final PurchaseController purchaseController = Get.find();

  void userTappedPay (){
    if(formKey.currentState!.validate()){
      showDialog(context: context,
        builder: (context)=>AlertDialog(
        title: const Text('Confirm payment '),
          content:  SingleChildScrollView(
            child: ListBody(
              children: [
                Text('Card Number : $cardNumber'),
                Text('Expiry date : $expiryDate'),
```

```
            Text('Card Holder Name : $cardHolderName'),
            Text('CVV Code  : $cvvCode')

          ],
        ),
      ),
      actions: [
        // TextButton(
        //   onPressed: () async {
        //     Navigator.pop(context);
        //     await purchaseController.saveOrderToFirebase(
        //       cardNumber: cardNumber,
        //       cardHolderName: cardHolderName,
        //       expiryDate: expiryDate,
        //       cvvCode: cvvCode, // Not secure to store CVV
        //     );
        //     // Navigate to HomePage after payment success
        //     Navigator.pushReplacement(
        //       context,
        //       MaterialPageRoute(builder: (context) => HomePage()),
        //     );
        //   },
        //   child: const Text('Confirm Payment'),
        // ),
        TextButton(

          onPressed:()=> {
            Navigator.pop(context),
            Navigator.push(context,MaterialPageRoute(builder: (context)=>HomePage(),))},
              child: Text('Back to HomePage'),
        ),


      ],
    ),
    );
  }
}

@override
Widget build(BuildContext context) {
  return Scaffold(
    backgroundColor: Theme.of(context).colorScheme.background,
    appBar: AppBar(
      backgroundColor: Colors.transparent,
      foregroundColor: Theme.of(context).colorScheme.inversePrimary,
```

```
      title: const Text('Checkout'),
    ),
    body: Column(
      children: [
        //credit card
        CreditCardWidget(
            cardNumber: cardNumber,
            expiryDate: expiryDate,
            cardHolderName: cardHolderName,
            cvvCode: cvvCode,
            showBackView: isCvvFocused,
            onCreditCardWidgetChange: (p0) {}),
        //Credit card Form
        CreditCardForm(
            cardNumber: cardNumber,
            expiryDate: expiryDate,
            cardHolderName: cardHolderName,
            cvvCode: cvvCode,
            onCreditCardModelChange: (data){
              setState(() {
                cardNumber=data.cardNumber;
                expiryDate=data.expiryDate;
                cardHolderName=data.cardHolderName;
                cvvCode=data.cvvCode;

              });
            },
            formKey: formKey),
           const Spacer(),
        MyButton(onTap: userTappedPay, text: 'Pay Now'),
        const SizedBox(height: 75,)// we can make height to 75 and change the height to box
and add padding

      ],
    ),
  );
 }
}
```

File Name : Home_controller.dart

```
import 'package:cloud_firestore/cloud_firestore.dart';
import 'package:flutter/material.dart';
import 'package:get/get.dart';
import 'package:ofsa_client_main/models/product_category/product_category.dart';
import '../models/product/product.dart';
```

```
class HomeController extends GetxController {
  final FirebaseFirestore firestore = FirebaseFirestore.instance;
  late final CollectionReference productCollection;
  late final CollectionReference categoryCollection;

  List<Product> products = [];
  List<Product> productShowInUi = [];
  List<ProductCategory> productCategory = [];
  List<String> selectedBrands = [];

  @override
  Future<void> onInit() async {
    super.onInit();
    productCollection = firestore.collection('product');
    categoryCollection = firestore.collection('category');
    await fetchCategory();
    await fetchProducts();
  }

  Future<void> fetchProducts() async {
    try {
      QuerySnapshot productSnapshot = await productCollection.get();
      final List<Product> retrievedProducts = productSnapshot.docs.map((doc) =>
          Product.fromJson(doc.data() as Map<String, dynamic>)).toList();

      products.clear();
      products.assignAll(retrievedProducts);
      productShowInUi.assignAll(products);
      Get.snackbar('Success', 'Products fetched successfully', colorText: Colors.green);
    } catch (e) {
      Get.snackbar('Error', 'Failed to fetch products: $e', colorText: Colors.red);
      print(e);
    } finally {
      update();
    }
  }

  Future<void> fetchCategory() async {
    try {
      QuerySnapshot categorySnapshot = await categoryCollection.get();
      final List<ProductCategory> retrievedCategories = categorySnapshot.docs.map((doc) =>
          ProductCategory.fromJson(doc.data() as Map<String, dynamic>)).toList();

      productCategory.clear();
      productCategory.assignAll(retrievedCategories);
      // Get.snackbar('Success', 'Category fetched successfully', colorText: Colors.green);
```

```
    } catch (e) {
      Get.snackbar('Error', 'Failed to fetch categories: $e', colorText: Colors.red);
      print(e);
    } finally {
      update();
    }
  }

  void filterByCategory(String category) {
    productShowInUi.clear();
    productShowInUi = products.where((product) => product.category == category).toList();
    update();
  }

  void filterByBrand(List<String> selectedBrands) {
    if (selectedBrands.isEmpty) {
      productShowInUi = products;
    } else {
      productShowInUi = products.where((product) =>
          selectedBrands.any((brand) => product.brand?.toLowerCase() ==
brand.toLowerCase())).toList();
    }
    update();
  }

  void sortByPrice({required bool ascending}) async {
    List<Product> sortedProducts = List<Product>.from(productShowInUi);
    sortedProducts.sort((a, b) => ascending ? a.price!.compareTo(b.price!) :
b.price!.compareTo(a.price!));
    productShowInUi = sortedProducts;
    update();
  }
}
```

File Name:PurchesController.dart

```
import 'package:flutter/material.dart';
import 'package:get/get.dart';
import 'package:http/http.dart' as http;
import 'dart:convert';
import 'package:ofsa_client_main/models/product/product.dart';

class PurchaseController extends GetxController {
  TextEditingController addressController = TextEditingController();

  RxList<Product> cartItems = <Product>[].obs;
  double get totalAmount => cartItems.fold(0, (sum, item) => sum + (item.price ?? 0));
```

```
@override
void onInit() {
  super.onInit();
}

@override
void onClose() {
  addressController.dispose();
  super.onClose();
  // addressController.clear();
}

void addToCart(Product product) {
  cartItems.add(product);
}

void removeFromCart(Product product) {
  cartItems.remove(product);
  // addressController.clear();
}

Future<void> submitOrder() async {
  double amount = totalAmount;
  String address = addressController.text;

  // Validate address
  if (!isValidAddress(address)) {
    Get.snackbar('Error', 'Invalid address');
    return;
  }

  try {
    final response = await http.post(
      Uri.parse('http://localhost:3000/create-order'),
      headers: {'Content-Type': 'application/json'},
      body: json.encode({'amount': amount, 'address': address}),
    );

    if (response.statusCode == 200) {
      final data = json.decode(response.body);
      final String orderId = data['orderId'];

      // Handle successful payment
      _handlePaymentSuccess(orderId);
    } else {
```

```
      _handlePaymentError('Failed to create order');
    }
  } catch (e) {
    _handlePaymentError('Failed to connect to server: $e');
  }
}

bool isValidAddress(String address) {
  // Implement your address validation logic here
  // For example, you could use regular expressions or a validation library
  return address.isNotEmpty; // Basic validation for now

}

void _handlePaymentSuccess(String orderId) {
  Get.snackbar('Payment Success', 'Order ID: $orderId');
  cartItems.clear();
}

void _handlePaymentError(String message) {
  Get.snackbar('Payment Error', 'Error: $message');
}

}
```

File Name:PurchesController.dart

```
import 'package:get/get.dart';
import 'package:ofsa_client_main/models/product/product.dart';

class PurchaseController extends GetxController {
  List<Product> cartItems = [];

  void addToCart(Product product) {
    cartItems.add(product);
    update(); // To update the UI
  }

  void removeFromCart(Product product) {
    cartItems.remove(product);
    update();
  }

  double get totalAmount {
    return cartItems.fold(0, (sum, item) => sum + (item.price ?? 0));
  }
```

```
  saveOrderToFirebase(
      {required String cardNumber,
      required String cardHolderName,
      required String expiryDate,
      required String cvvCode}) {}
}
```

File Name: DropDown_Btn.dart

```
import 'package:dropdown_button2/dropdown_button2.dart';
import 'package:flutter/material.dart';

class DropDownBtn extends StatefulWidget {
  final List<String> items;
  final String selectedItemsText;
  final Function(String?) onSelected;

  const DropDownBtn({
    super.key,
    required this.items,
    required this.selectedItemsText,
    required this.onSelected,
  });

  @override
  _DropDownBtnState createState() => _DropDownBtnState();
}

class _DropDownBtnState extends State<DropDownBtn> {
  String? selectedValue;

  @override
  Widget build(BuildContext context) {
    return Card(
      child: Center(
        child: DropdownButtonHideUnderline(
          child: DropdownButton2<String>(
            isExpanded: true,
            hint: Text(
              widget.selectedItemsText,
              style: TextStyle(
                fontSize: 14,
                color: Theme.of(context).hintColor,

              ),
            ),
```

```
          items: widget.items
              .map((String item) => DropdownMenuItem<String>(
            value: item,
            child: Text(
              item,
              style: const TextStyle(
                fontSize: 14,
              ),
            ),
          ))
              .toList(),
          value: selectedValue,
          onChanged: (String? value) {
            setState(() {
              selectedValue = value;
            });
            widget.onSelected(value);
          },
          buttonStyleData: const ButtonStyleData(
            padding: EdgeInsets.symmetric(horizontal: 16),
            height: 40,
            width: 140,
          ),
          menuItemStyleData: const MenuItemStyleData(
            height: 40,
          ),
        ),
      ),
    ),
  );
 }
}
```

File Name:Firebase.options.dart

```
import 'package:firebase_core/firebase_core.dart';

class DefaultFirebaseOptions {
  // Return FirebaseOptions for Android only
  static FirebaseOptions get android => _android;

  static const FirebaseOptions _android = FirebaseOptions(
    apiKey: 'AIzaSyCuePtnWKmrJwRbb3OhNmOkBeJugXbCAyk', // Replace with your
Firebase API key
    appId: '1:738504089346:android:1c5da95f1fca0bd7faec1b', // Replace with your Firebase
App ID
    messagingSenderId: '738504089346', // Replace with your Firebase Messaging Sender ID
```

```
    projectId: 'ofsa-3c2fc', // Replace with your Firebase Project ID
    storageBucket: 'gs://ofsa-3c2fc.appspot.com', // Replace with your Firebase Storage Bucket
URL
  );
}
```

Admin_side

File Name: main.dart

```
import 'package:firebase_core/firebase_core.dart';
import 'package:flutter/material.dart';
import 'package:get/get.dart';
// import 'package:get/get_navigation/src/root/get_material_app.dart';
import 'package:osfa_admin/controllor/home_controller.dart';
import 'package:osfa_admin/firebase_options.dart';
import 'package:osfa_admin/pages/home_page.dart';

Future<void> main() async {
  //binding the databse by using the wigit flutterbind
  WidgetsFlutterBinding.ensureInitialized();
  //for using firebase
  await Firebase.initializeApp(options: firebaseOptions);
  // regester the home controller because of error
  Get.put(HomeController());
  runApp(const MyApp() );
}
class MyApp extends StatelessWidget {
  const MyApp({super.key});
  @override
  Widget build(BuildContext context) {
    return GetMaterialApp(
      debugShowCheckedModeBanner: false,
      title: 'Flutter proj',
      theme: ThemeData(
        colorScheme: ColorScheme.fromSeed(seedColor: Colors.deepPurple)
      ),
      home: const HomePage(),
      // initialBinding: StoreBindings(),
    );
  }
}
```

File HomePage.dart

```
import 'package:flutter/material.dart';
import 'package:get/get.dart';
import 'package:osfa_admin/controllor/home_controller.dart';
import 'package:osfa_admin/pages/add_product_page.dart';  // For adding product
```

```
import 'package:osfa_admin/pages/edit_product_page.dart';  // For editing product

class HomePage extends StatelessWidget {
  const HomePage({super.key});

  @override
  Widget build(BuildContext context) {
    return GetBuilder<HomeController>(builder: (ctrl) {
      return Scaffold(
        appBar: AppBar(
          title: const Text('Online Furniture Selling App'),
          centerTitle: true,
        ),
        body: ListView.builder(
          itemCount: ctrl.products.length,
          itemBuilder: (context, index) {
            return ListTile(
              title: Text(ctrl.products[index].name ?? ''),
              subtitle: Text((ctrl.products[index].price ?? 0).toString()),
              trailing: Row(
                mainAxisSize: MainAxisSize.min,
                children: [
                  // Edit button
                  IconButton(
                    icon: const Icon(Icons.edit),
                    onPressed: () async {
                      await Get.to(EditProductPage(product: ctrl.products[index]));
                      // Refresh product list after returning from EditProductPage
                      ctrl.fetchProducts();
                      ctrl.update();
                    },
                  ),
                  // Delete button
                  IconButton(
                    icon: const Icon(Icons.delete),
                    onPressed: () async {
                      ctrl.deleteProduct(ctrl.products[index].id ?? '');
                      try {
                        await ctrl.productCollection.doc(ctrl.products[index].id).delete();
                        ctrl.fetchProducts();
                        Get.snackbar('Success', 'Product deleted successfully', colorText:
Colors.green);
                      } catch (e) {
                        Get.snackbar('Error', 'Failed to delete product', colorText: Colors.red);
                      }
                    },
```

```
                ),
              ],
            ),
          );
        },
      ),
      floatingActionButton: FloatingActionButton(
        onPressed: () async {
          await Get.to(const AddProductPage());
          ctrl.fetchProducts();  // Refresh product list after returning from AddProductPage
          ctrl.update();
        },
        child: const Icon(Icons.add),
      ),
    );
  });
 }
}
```

File Name:Edit_product_page.dart

```
import 'package:dropdown_button2/dropdown_button2.dart';
import 'package:flutter/material.dart';
import 'package:get/get.dart';
import 'package:image_picker/image_picker.dart';
import 'package:osfa_admin/controllor/home_controller.dart';
import 'package:osfa_admin/model/product/product.dart';

class EditProductPage extends StatelessWidget {
  final Product product;

  const EditProductPage({super.key, required this.product});

  @override
  Widget build(BuildContext context) {
    return GetBuilder<HomeController>(builder: (ctrl) {
      // Pre-fill the controllers with the existing product data
      ctrl.productNameCtrl.text = product.name ?? '';
      ctrl.productDescriptionCtrl.text = product.description ?? '';
      ctrl.productImgCtrl.text = product.image ?? '';
      ctrl.productPriceCtrl.text = product.price?.toString() ?? '';

      // Ensuring default values are set properly
      ctrl.category = product.category ?? 'general';
      ctrl.brand = product.brand ?? 'Unbranded'; // Default to 'Unbranded'
      ctrl.offer = product.offer ?? false;
```

```
return Scaffold(
  appBar: AppBar(
    title: const Text('Edit Product'),
    centerTitle: true,
  ),
  body: SingleChildScrollView(
    child: Container(
      margin: const EdgeInsets.all(10),
      width: double.maxFinite,
      child: Column(
        crossAxisAlignment: CrossAxisAlignment.center,
        children: [
          const Text(
            'Edit Product',
            style: TextStyle(
              fontSize: 30,
              color: Colors.indigoAccent,
              fontWeight: FontWeight.bold,
            ),
          ),
          const SizedBox(height: 15),
          TextField(
            controller: ctrl.productNameCtrl,
            decoration: InputDecoration(
              border: OutlineInputBorder(
                borderRadius: BorderRadius.circular(10),
              ),
              label: const Text('Product Name'),
              hintText: 'Enter Your Product Name',
            ),
          ),
          const SizedBox(height: 15),
          TextField(
            controller: ctrl.productDescriptionCtrl,
            decoration: InputDecoration(
              border: OutlineInputBorder(
                borderRadius: BorderRadius.circular(10),
              ),
              label: const Text('Product Description'),
              hintText: 'Enter the Description of the Product',
            ),
            maxLines: 4,
          ),
          const SizedBox(height: 15),
          TextField(
            controller: ctrl.productImgCtrl,
```

```
  decoration: InputDecoration(
    border: OutlineInputBorder(
      borderRadius: BorderRadius.circular(10),
    ),
    label: const Text('Image URL'),
    hintText: 'Enter Your Image URL',
  ),
),
  const SizedBox(height: 15),
ElevatedButton(
  style: ElevatedButton.styleFrom(
    backgroundColor: Colors.indigo,
    foregroundColor: Colors.white,
  ),
  onPressed: () async {
    final picker = ImagePicker();
    try {
      final pickedFile = await picker.pickImage(source: ImageSource.gallery);
      if (pickedFile != null) {
        // For local file path
        ctrl.productImgCtrl.text = pickedFile.path;
      }
    } catch (e) {
      Get.snackbar("Error", "Failed to pick image: $e", colorText: Colors.red);
      print("Error picking image: $e");
    }
  },
  child: const Text('Pick Image from Gallery'),
),
const SizedBox(height: 15),
TextField(
  controller: ctrl.productPriceCtrl,
  decoration: InputDecoration(
    border: OutlineInputBorder(
      borderRadius: BorderRadius.circular(10),
    ),
    label: const Text('Product Price'),
    hintText: 'Enter Your Product Price',
  ),
),
const SizedBox(height: 15),

// Column for dropdowns
// Column(
//   crossAxisAlignment: CrossAxisAlignment.stretch,
//   children: [
```

```
//     _buildDropdown(
//       context: context,
//       items: ['Table', 'Chair', 'Bed', 'Desk', 'Cupboard', 'Sofa'],
//       selectedValue: ctrl.category,
//       hint: 'Select Category',
//       onChanged: (String? value) {
//         ctrl.category = value ?? 'general';
//         ctrl.update();
//       },
//     ),
//     const SizedBox(height: 15),
//     _buildDropdown(
//       context: context,
//       items: ['IKEA', 'Unbranded', 'Nilkamal', 'WoodenStreet'],
//       selectedValue: ctrl.brand,
//       hint: 'Select Brand',
//       onChanged: (String? value) {
//         ctrl.brand = value ?? 'Unbranded';
//         ctrl.update();
//       },
//     ),
//   ],
// ),

const SizedBox(height: 5),
const Text('Is this product on offer?'),
const SizedBox(height: 5),
_buildDropdown(
  context: context,
  items: ['true', 'false'],
  selectedValue: ctrl.offer.toString(),
  hint: 'Is this product on offer?',
  onChanged: (String? value) {
    ctrl.offer = value == 'true';
    ctrl.update();
  },
),
const SizedBox(height: 15),
ElevatedButton(
  style: ElevatedButton.styleFrom(
    backgroundColor: Colors.indigo,
    foregroundColor: Colors.white,
  ),
  onPressed: () async {
    try {
      // Update the product in Firestore
```

```
                    await ctrl.updateProduct(product.id ?? '');
                    // Fetch the updated product list
                    await ctrl.fetchProducts();
                    // Update the UI and go back
                    Get.back();
                    // Show a success message
                    Get.snackbar("Success", "Product updated successfully", colorText:
Colors.green);
                  } catch (e) {
                    // Show an error message in case of failure
                    Get.snackbar("Error", "Failed to update product: $e", colorText: Colors.red);
                    print("Error updating product: $e");
                  }
                },
                child: const Text('Update Product'),
              ),
            ],
          ),
        ),
      ),
    );
  });
}

// Widget for building the dropdown menus
Widget _buildDropdown({
  required BuildContext context,
  required List<String> items,
  required String selectedValue,
  required String hint,
  required void Function(String?) onChanged,
}) {
  return DropdownButtonFormField2<String>(
    value: items.contains(selectedValue) ? selectedValue : null,
    items: items
        .map((String item) => DropdownMenuItem<String>(
              value: item,
              child: Text(item),
            ))
        .toList(),
    onChanged: onChanged,
    decoration: InputDecoration(
      border: OutlineInputBorder(
        borderRadius: BorderRadius.circular(10),
      ),
      labelText: hint,
```

```
      ),
      hint: Text(hint),
    );
  }
}
```

File Name:Home_Controllor.dart

```
  import 'package:cloud_firestore/cloud_firestore.dart';
import 'package:flutter/material.dart';
import 'package:get/get.dart';
import 'package:osfa_admin/model/product/product.dart';

class HomeController extends GetxController {
  final FirebaseFirestore firestore = FirebaseFirestore.instance;
  late final CollectionReference productCollection;

  final TextEditingController productNameCtrl = TextEditingController();
  final TextEditingController productDescriptionCtrl = TextEditingController();
  final TextEditingController productImgCtrl = TextEditingController();
  final TextEditingController productPriceCtrl = TextEditingController();

  String category = 'category';
  String brand = 'Unbranded';
  bool offer = false;

  final List<Product> products = [];

  @override
  Future<void> onInit() async {
    super.onInit();
    productCollection = firestore.collection('product');
    await fetchProducts();
  }

  Future<void> addProduct() async {
    try {
      DocumentReference doc = productCollection.doc();
      Product newProduct = Product(
        id: doc.id,
        name: productNameCtrl.text.trim(),
        category: category,
        description: productDescriptionCtrl.text.trim(),
        price: double.tryParse(productPriceCtrl.text) ?? 0.0,
        brand: brand,
        image: productImgCtrl.text.trim(),
```

```
      offer: offer,
    );
    await doc.set(newProduct.toJson());
    resetFormFields();
    await fetchProducts();
  } catch (e) {
    Get.snackbar("Error", "Failed to add product: $e", colorText: Colors.red);
    print("Error adding product: $e");
  }
}

Future<void> fetchProducts() async {
  try {
    QuerySnapshot snapshot = await productCollection.get();
    products.clear();
    for (var doc in snapshot.docs) {
      var data = doc.data() as Map<String, dynamic>;
      double price = data['price'] is int
          ? (data['price'] as int).toDouble()
          : data['price'] as double;
      products.add(Product.fromJson({...data, 'price': price}));
    }
    update();
  } catch (e) {
    Get.snackbar("Error", "Failed to fetch products: $e", colorText: Colors.red);
    print("Error fetching products: $e");
  }
}

Future<void> deleteProduct(String id) async {
  try {
    await productCollection.doc(id).delete();
    await fetchProducts();
    Get.snackbar("Success", "Product deleted successfully", colorText: Colors.green);
  } catch (e) {
    Get.snackbar("Error", "Failed to delete product: $e", colorText: Colors.red);
    print("Error deleting product: $e");
  }
}

Future<void> updateProduct(String id) async {
  try {
    if (id.isNotEmpty) {
      final updatedData = {
        'name': productNameCtrl.text.trim(),
        'category': category,
```

```
        'description': productDescriptionCtrl.text.trim(),
        'price': double.tryParse(productPriceCtrl.text) ?? 0.0,
        'brand': brand,
        'image': productImgCtrl.text.trim(),
        'offer': offer,
      };
      await productCollection.doc(id).update(updatedData);
      Get.snackbar("Success", "Product updated successfully", colorText: Colors.green);
    } else {
      throw Exception("Product ID is empty.");
    }
  } catch (e) {
    Get.snackbar("Error", "Failed to update product: $e", colorText: Colors.red);
    print("Error updating product: $e");
  }
}

void resetFormFields() {
  productNameCtrl.clear();
  productDescriptionCtrl.clear();
  productImgCtrl.clear();
  productPriceCtrl.clear();
  category = 'category';
  brand = 'Unbranded';
  offer = false;
  update();
}

@override
void onClose() {
  productNameCtrl.dispose();
  productDescriptionCtrl.dispose();
  productImgCtrl.dispose();
  productPriceCtrl.dispose();
  super.onClose();
}
}
```

File Name:Add_product_page.dart

```
import 'package:dropdown_button2/dropdown_button2.dart';
import 'package:flutter/material.dart';
import 'package:get/get.dart';
import 'package:image_picker/image_picker.dart';
```

```
import 'package:osfa_admin/controllor/home_controller.dart';

class AddProductPage extends StatelessWidget {
  const AddProductPage({super.key});

  @override
  Widget build(BuildContext context) {
    return GetBuilder<HomeController>(builder: (ctrl) {
      return Scaffold(
        appBar: AppBar(
          title: const Text('Add Your Products'),
          centerTitle: true,
        ),
        body: SingleChildScrollView(
          child: Container(
            margin: const EdgeInsets.all(10),
            width: double.maxFinite,
            child: Column(
              crossAxisAlignment: CrossAxisAlignment.center,
              children: [
                const Text(
                  'Add New Product',
                  style: TextStyle(
                    fontSize: 30,
                    color: Colors.indigoAccent,
                    fontWeight: FontWeight.bold,
                  ),
                ),
                const SizedBox(height: 15),
                TextField(
                  controller: ctrl.productNameCtrl,
                  decoration: InputDecoration(
                    border: OutlineInputBorder(
                      borderRadius: BorderRadius.circular(10),
                    ),
                    label: const Text('Product Name'),
                    hintText: 'Enter Your Product Name',
                  ),
                ),
                const SizedBox(height: 15),
                TextField(
                  controller: ctrl.productDescriptionCtrl,
                  decoration: InputDecoration(
                    border: OutlineInputBorder(
                      borderRadius: BorderRadius.circular(10),
                    ),
```

```
    label: const Text('Product Description'),
    hintText: 'Enter the Description of the Product',
  ),
  maxLines: 4,
),
const SizedBox(height: 15),
TextField(
  controller: ctrl.productImgCtrl,
  decoration: InputDecoration(
    border: OutlineInputBorder(
      borderRadius: BorderRadius.circular(10),
    ),
    label: const Text('Image URL'),
    hintText: 'Enter Your Image URL',
  ),
),
const SizedBox(height: 15),
ElevatedButton(
  onPressed: () async {
    final picker = ImagePicker();
    try {
      final pickedFile = await picker.pickImage(source: ImageSource.gallery);
      if (pickedFile != null) {
        // For local file path
        ctrl.productImgCtrl.text = pickedFile.path;
      }
    } catch (e) {
      Get.snackbar("Error", "Failed to pick image: $e", colorText: Colors.red);
      print("Error picking image: $e");
    }
  },
  child: const Text('Pick Image from Gallery'),
),
const SizedBox(height: 15),
TextField(
  controller: ctrl.productPriceCtrl,
  decoration: InputDecoration(
    border: OutlineInputBorder(
      borderRadius: BorderRadius.circular(10),
    ),
    label: const Text('Product Price'),
    hintText: 'Enter Your Product Price',
  ),
),
const SizedBox(height: 15),
```

```
Column(
  crossAxisAlignment: CrossAxisAlignment.stretch,
  children: [
    _buildDropdown(
      context: context,
      items: ['Table', 'Chair', 'Bed', 'Desk', 'Cupboard', 'Sofa'],
      selectedValue: ctrl.category,
      hint: 'Select Category',
      onChanged: (String? value) {
        ctrl.category = value ?? 'general';
        ctrl.update();
      },
    ),
    const SizedBox(height: 15),
    _buildDropdown(
      context: context,
      items: ['IKEA', 'Unbranded', 'Nilkamal', 'WoodenStreet'],
      selectedValue: ctrl.brand,
      hint: 'Select Brand',
      onChanged: (String? value) {
        ctrl.brand = value ?? 'Unbranded';
        ctrl.update();
      },
    ),
  ],
),

const SizedBox(height: 15),
const Text('Is this product on offer?'),
const SizedBox(height: 15),
_buildDropdown(
  context: context,
  items: ['true', 'false'],
  selectedValue: ctrl.offer.toString(),
  hint: 'Is this product on offer?',
  onChanged: (String? value) {
    ctrl.offer = value == 'true';
    ctrl.update();
  },
),
const SizedBox(height: 15),
ElevatedButton(
  style: ElevatedButton.styleFrom(
    backgroundColor: Colors.indigo,
    foregroundColor: Colors.white,
  ),
```

```
                onPressed: () async {
                  try {
                    await ctrl.addProduct();
                    Get.snackbar("Success", "Product added successfully", colorText:
Colors.green);
                    Get.back(); // Go back after adding the product
                  } catch (e) {
                    Get.snackbar("Error", "Failed to add product: $e", colorText: Colors.red);
                    print("Error adding product: $e");
                  }
                },
                child: const Text('Add Product'),
              ),
            ],
          ),
        ),
      ),
    );
  });
}

Widget _buildDropdown({
  required BuildContext context,
  required List<String> items,
  required String selectedValue,
  required String hint,
  required void Function(String?) onChanged,
}) {
  return DropdownButtonFormField2<String>(
    value: items.contains(selectedValue) ? selectedValue : null,
    items: items
        .map((String item) => DropdownMenuItem<String>(
              value: item,
              child: Text(item),
            ))
        .toList(),
    onChanged: onChanged,
    decoration: InputDecoration(
      border: OutlineInputBorder(
        borderRadius: BorderRadius.circular(10),
      ),
      labelText: hint,
    ),
    hint: Text(hint),
  );
```

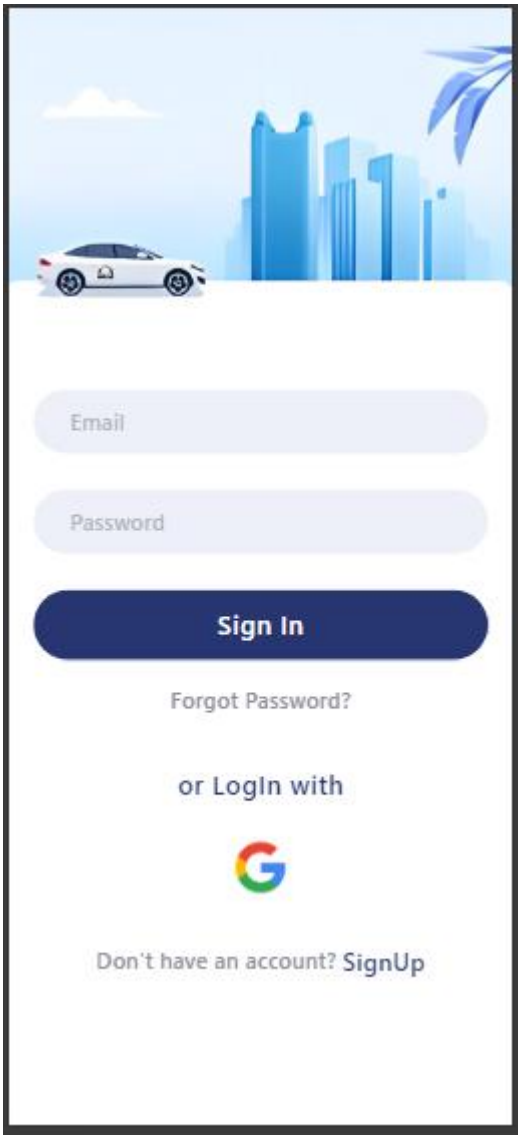
```
  }
}
```

File Name:DropDownBtn.dart

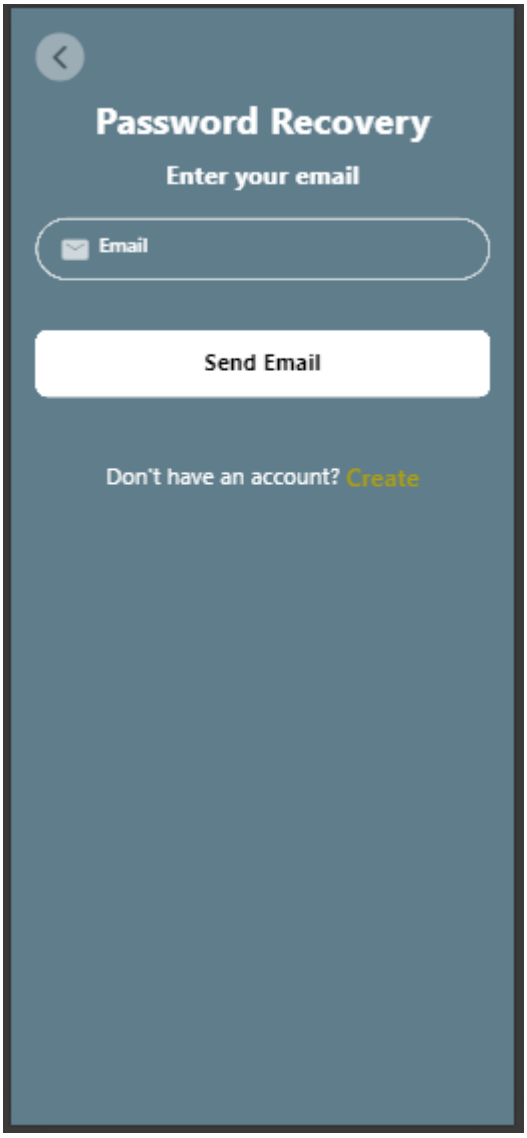
```
import 'package:dropdown_button2/dropdown_button2.dart';
import 'package:flutter/foundation.dart';
import 'package:flutter/material.dart';

String? selectedValue;

class DropDownBtn extends StatelessWidget {
  final List<String> items;
  final String selectedItemsText;
  final Function(String?) onSelected;

  const DropDownBtn({
    super.key,
    required this.items,
    this.selectedItemsText = 'Select Item',
    required this.onSelected,
  });

  @override
  Widget build(BuildContext context) {
    return Card(
      child: Center(
        child: DropdownButtonHideUnderline(
          child: DropdownButton2<String>(
            isExpanded: true,
            hint: Text(
              selectedItemsText,
              style: TextStyle(
                fontSize: 14,
                color: Theme.of(context).hintColor,
              ),
            ),
            items: items
                .map((String item) => DropdownMenuItem<String>(
              value: item,
              child: Text(
                item,
                style: const TextStyle(
                  fontSize: 14,
                ),
              ),
            ))
```

```
            .toList(),
          value: selectedValue,
          onChanged: (String? value) {
            selectedValue = value;
            onSelected(value);
          },
          buttonStyleData: const ButtonStyleData(
            padding: EdgeInsets.symmetric(horizontal: 16),
            height: 40,
            width: 140,
          ),
          menuItemStyleData: const MenuItemStyleData(
            height: 40,
          ),
        ),
      ),
    ),
  );
}

@override
void debugFillProperties(DiagnosticPropertiesBuilder properties) {
  super.debugFillProperties(properties);
  properties.add(DiagnosticsProperty('selectedItemsText', selectedItemsText));
}
}
```

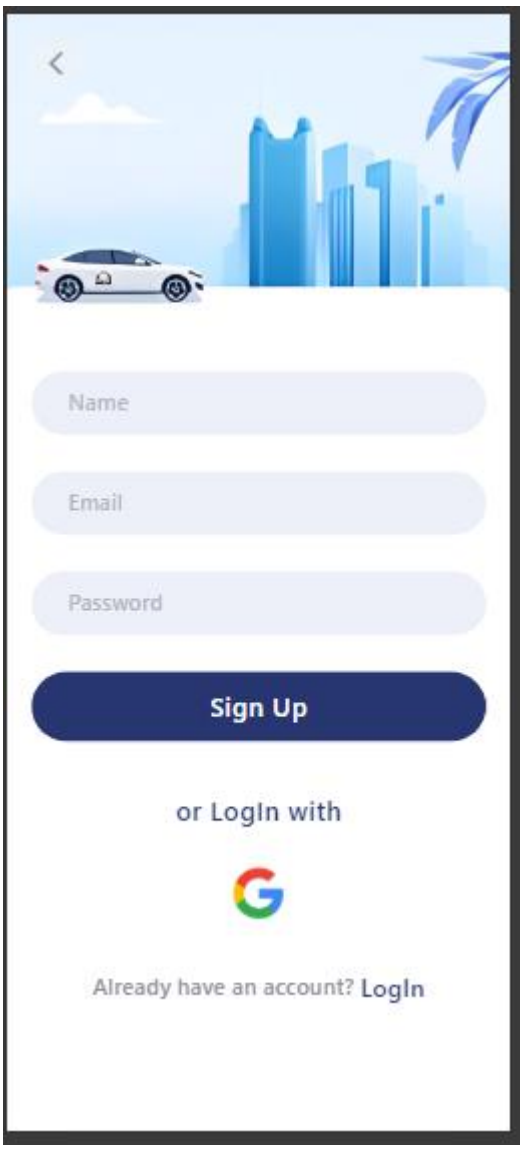
4.2 Screen Shorts



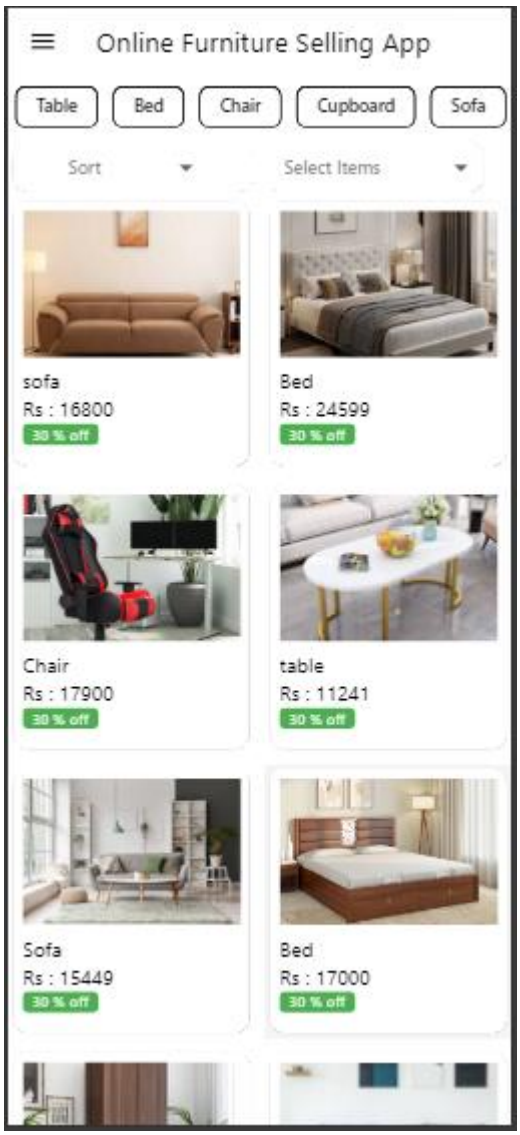
Login Screen



Forget password page

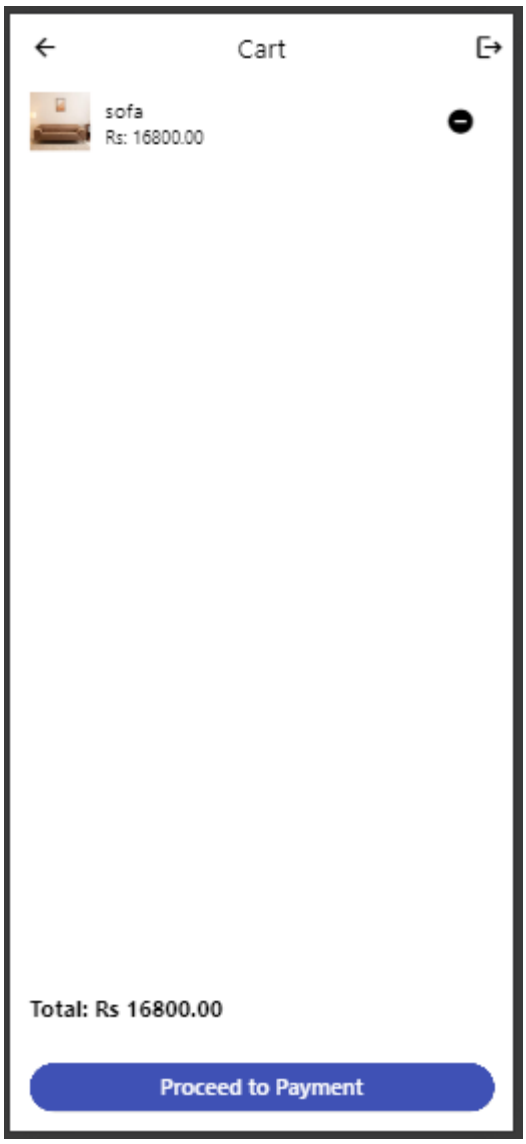

Name
Email
Password
Sign Up
or LogIn with
Already have an account? LogIn

Signup Page

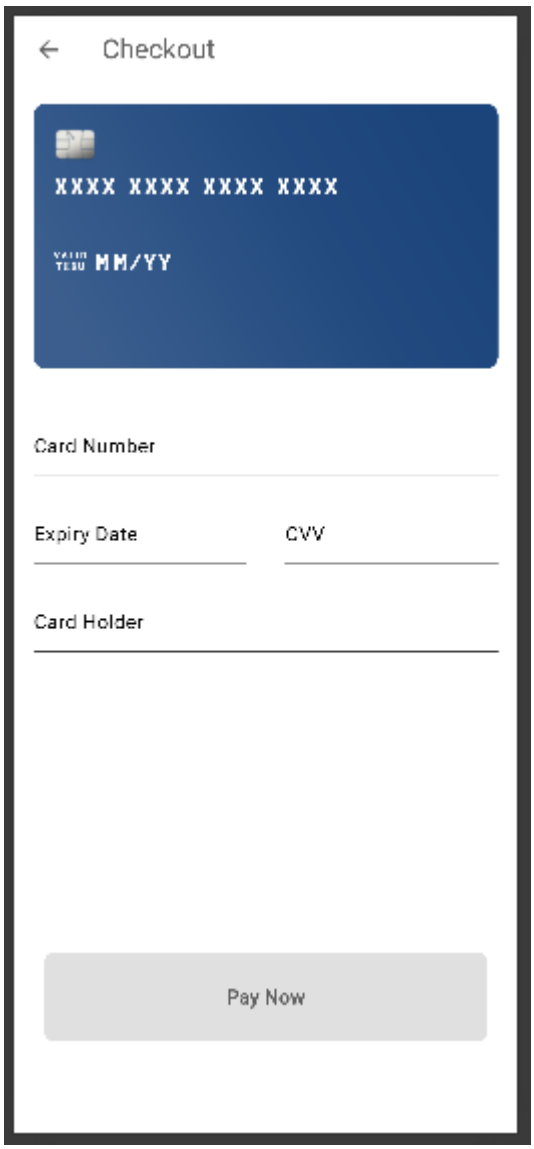

Online Furniture Selling App
Table
Bed
Chair
Cupboard
Sofa
Sort
Select Items
sofa
Rs : 16800
30 % off
Bed
Rs : 24599
30 % off
Chair
Rs : 17900
30 % off
table
Rs : 11241
30 % off
Sofa
Rs : 15449
30 % off
Bed
Rs : 17000
30 % off

HomePage

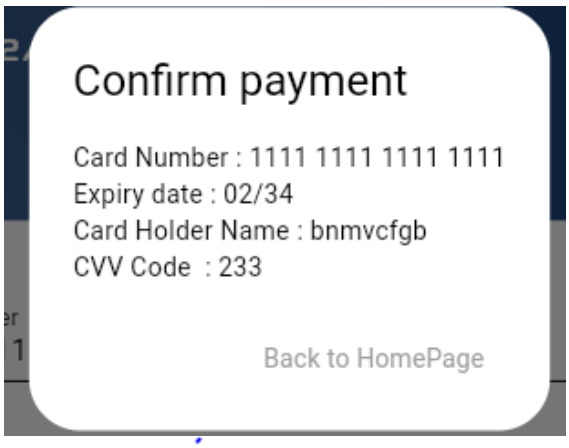
Product Details Page

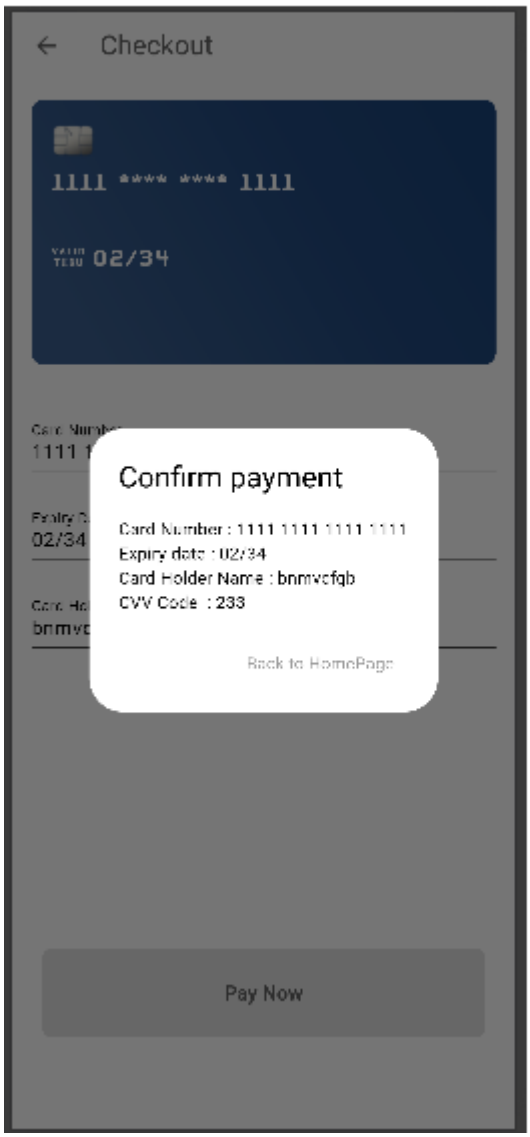


Cart page



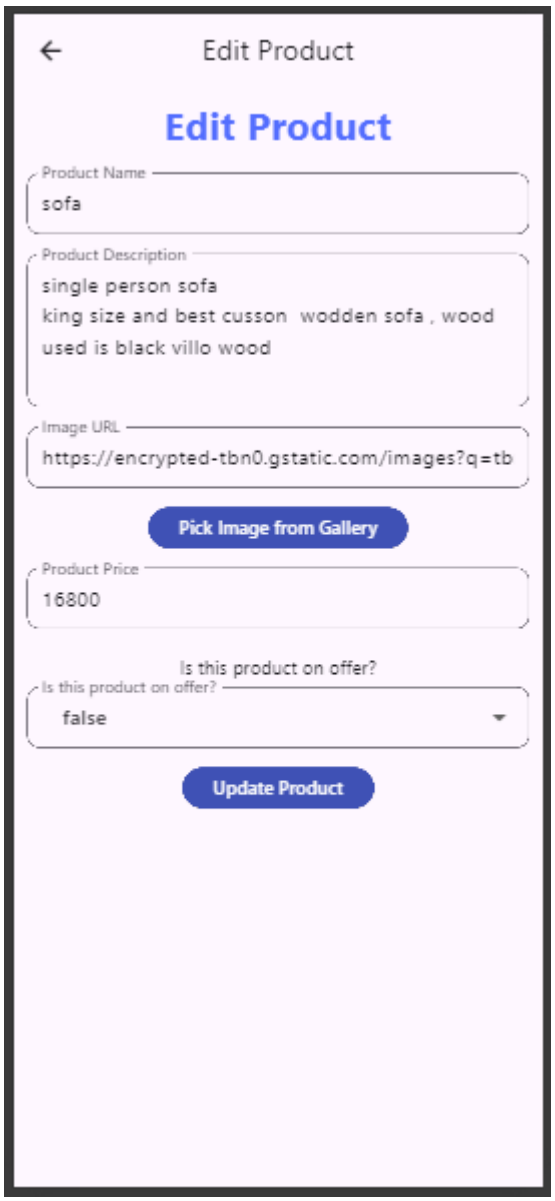
Payment Page


Confirm payment
Card Number : 1111 1111 1111 1111
Expiry date : 02/34
Card Holder Name : bnmvcfgb
CVV Code : 233
Back to HomePage

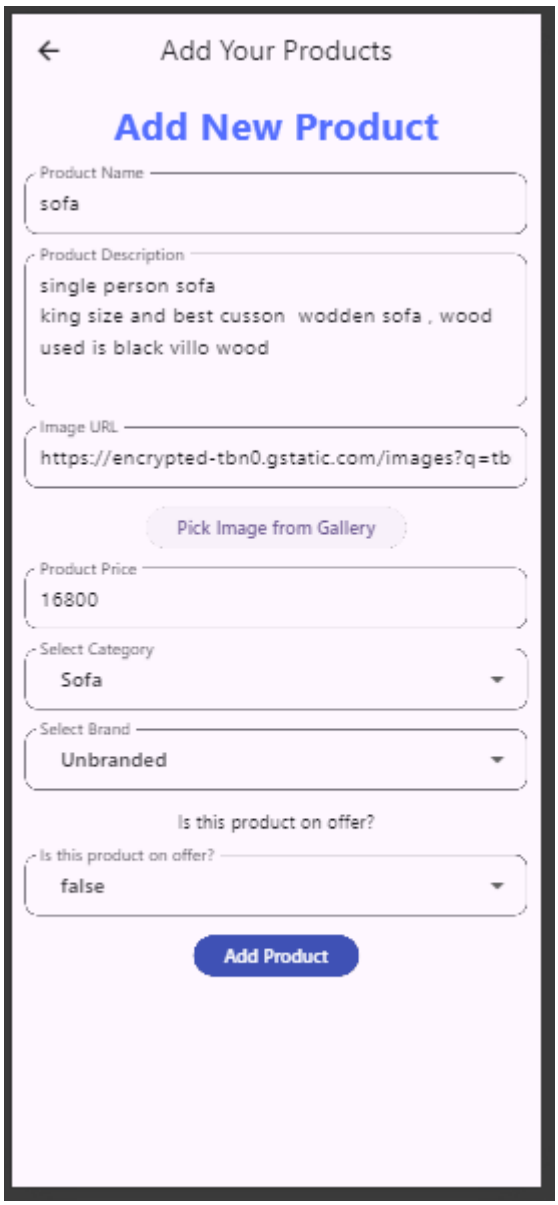

Checkout
1111 **** **** 1111
02/34
Confirm payment
Card Number : 1111 1111 1111 1111
Expiry date : 02/34
Card Holder Name : bnmvcfgb
CVV Code : 233
Back to HomePage
Pay Now


Online Furniture Selling App
sofa
16800
Bed
24599
Chair
17900
table
11241
Sofa
15449
Bed
17000
Cupboard
12998
Sofa
7000
table
12534
+

Admin Page



Product_Edit_page_admin



Product Add page Admin

5. Validation

1. Test Cases

Test cases are specifications of the inputs to the test and the expected output from the system (the test results), plus a statement of what is being tested. Test data are the inputs that have been devised to test a system. Test data can sometimes be generated automatically, but automatic test case generation is impossible, as people who understand what the system is supposed to do, must be involved to specify the expected test results.

In simple terms, test case is a defined format for software testing which checks whether the particular software/applications functions as intended. Test cases, generally including testing each and every part of the software/application. It can be any component, any function and so on. Testing is an important aspect of software development life cycle and gives developers the idea whether their software is working as intended or not.
Why Prepare Test Cases? As we just stated, testing is an important aspect of software development life cycle, as testing gives a clear understanding whether the function of the software is an intended or not and checks for any loop holes, any accidental mistakes left, before deploying the project. Here are some reasons as to why test cases are important:

- Checking whether software/application meets the stakeholder requirement
- Checking the consistency of software/application with conditions.
- Deduce the future updates related to bugs/errors. Instead, they can be focused on implementing more functionality or to enhance user experience.
- Making sure all the aspects of the software/application work as intended.
- Taking notes of certain tests, which will help during the maintenance phase.

When to Prepare Test Cases? Test cases are for the whole and sole purpose of testing. Testing can be done at various phases according to the development team or stakeholder requirements. Generally, test cases are prepared before development of product, during the development of product or after development of the product.

- Before Development Test Cases: These are prepared before the actual development of product, meaning before the coding part. This gives an idea to the developers to identify the requirements of end-product and can be carried out after the end product gets developed. This makes sure any bugs or errors will be identified beforehand and speed up the coding process.
- During Development Test Cases: These are prepared parallel to the development. Usually used when a huge module (a function/module implementing maximum functionality) is developed, in order to prevent problems in future and help in partial debugging.

- After Development Test Cases: These are prepared after the coding part is done but deployment is remaining. Most of development teams use these test cases, as these are most efficient and ensure professional deployment of the end-product without any problems or issues. These test cases ensure all the functionality are functioning as intended and the deployment phase can be conducted without any problems.

For this web application too, there exist certain test cases, most of test cases are for sign up/login components, while others are for communities. In this documentation, we will check all of these test cases. We have followed "During Development Test Cases" and "After Development Test Cases". As these were perfect for us, since using React Components, it was the best to test it after implementing each component, to ensure accurate functionality without any issues.

Testing was an important factor for us, as it was just what we needed to give that spicy "professional" look to our web application. Using "During Development Test Cases" ensured our final test case was performed without a lot of issues and took comparatively less time than expected. This ensured that we could provide more time to the "debugging" phase, encountering any future error-possibilities, which may arise after development, and making changes to solve those problems.

Here are some functionalities/features where we performed "During Development Test Cases":

1. After developing the Sign-Up and Sign-In Page.
2. After developing the Login Page.

TEST DATA

Test Data is nothing but set of input values used to run tests on software/application, used to confirm its dependability, functionality and performance. In simple words, test data is what developers input/enter during testing to find out whether everything works as intended. Test data is an integral part of the testing process. It provides with important information like finding defects, whether the software/application can handle large amounts of data. Various possibilities arise when entering test data, the data entered for testing may be normal, invalid, error-prone, stressed and so on.

Let's see explanation of these types of test data.

- Normal Test Data: It represents typical user inputs. What possibilities arise when end user data enter the data.
- Invalid Test Data: It enables the software/application's ability to handle unusual input data or wrong input data.
- Error-Prone Test Data: It enables the software's resistance to errors.
- Stressed Test Data: It represents the software/applications ability to handle high loads of data.

During testing phase of this web application, we've made various test cases and ran them through thoroughly, to make sure there are no loopholes, errors in the application. Following are the fields/components we tested.

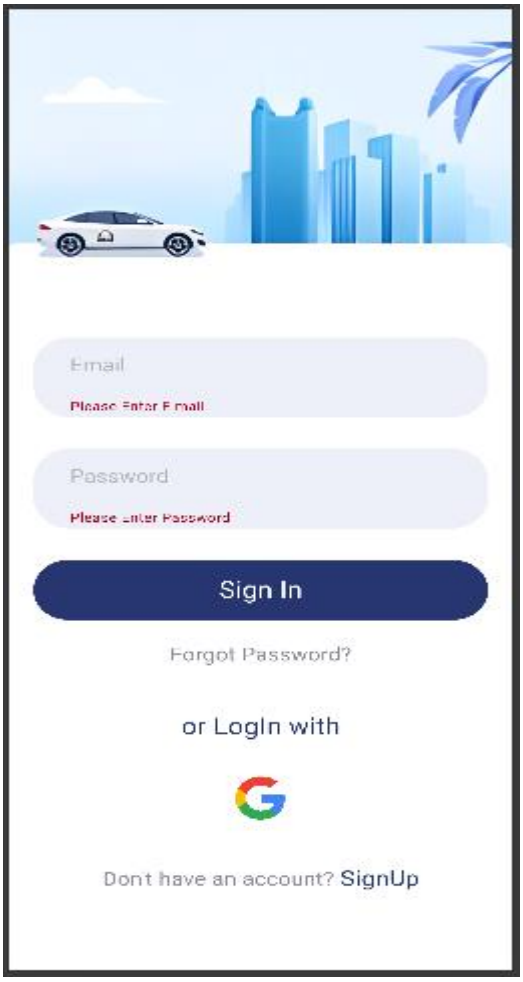
Username:

1. When user enter any symbol other than underscore (_) or hyphen (-), it gives them an alert saying username cannot contain those characters.
2. During new user creation, an already existing username cannot be used. An alert is giving saying "User already exists."
3. Whenever user tries to proceed without entering any username, an alert box saying "This field cannot be empty" is shown.

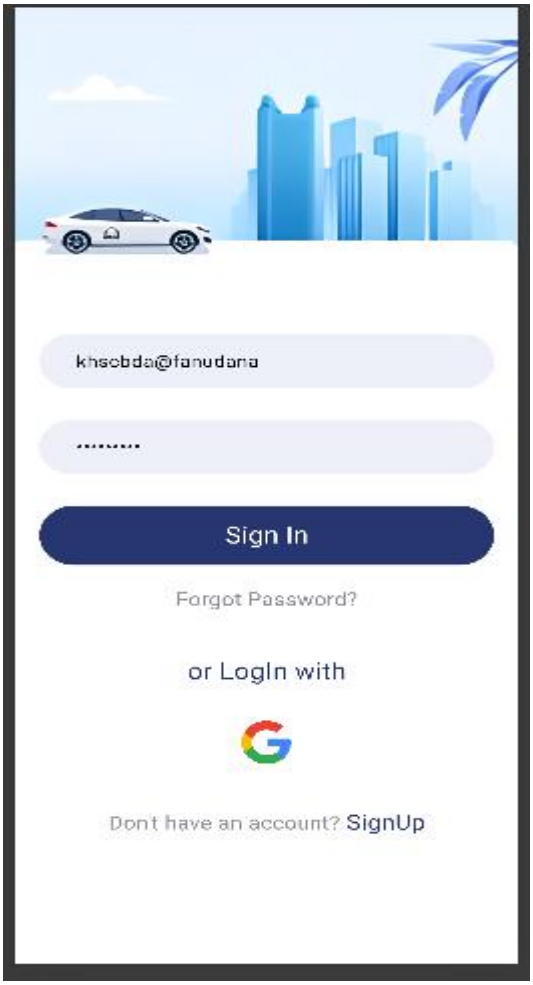
Email:

1. If user enters an invalid/incomplete email address, an alert box saying "Identifier Invalid" is prompted.
2. If user enters an incorrect email address, an alert box saying "Couldn't find account" is prompted.
3. If user try to create 2 accounts using the same email address, an alert box saying "This Email address is already taken" prompts.

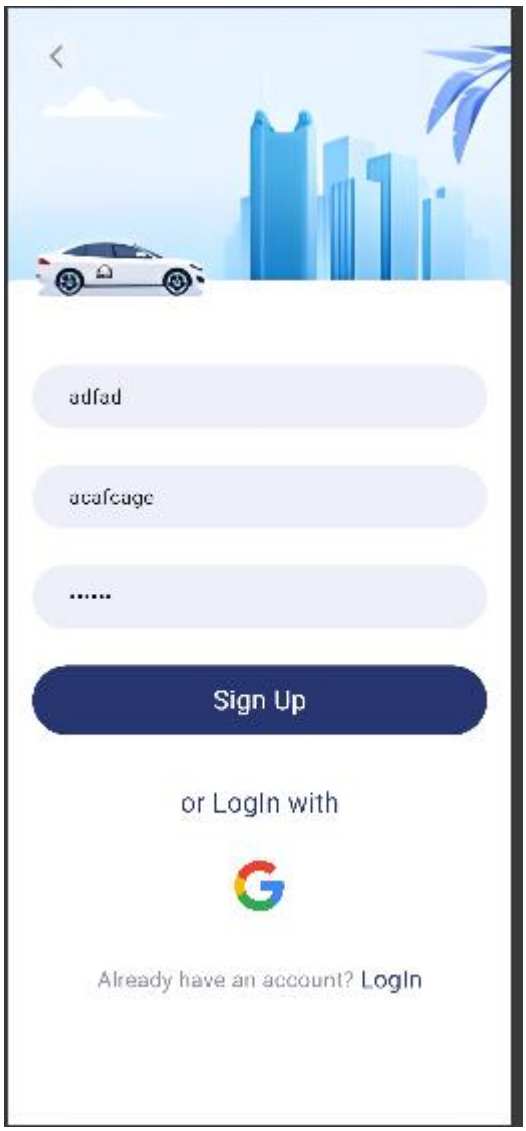
TEST Result



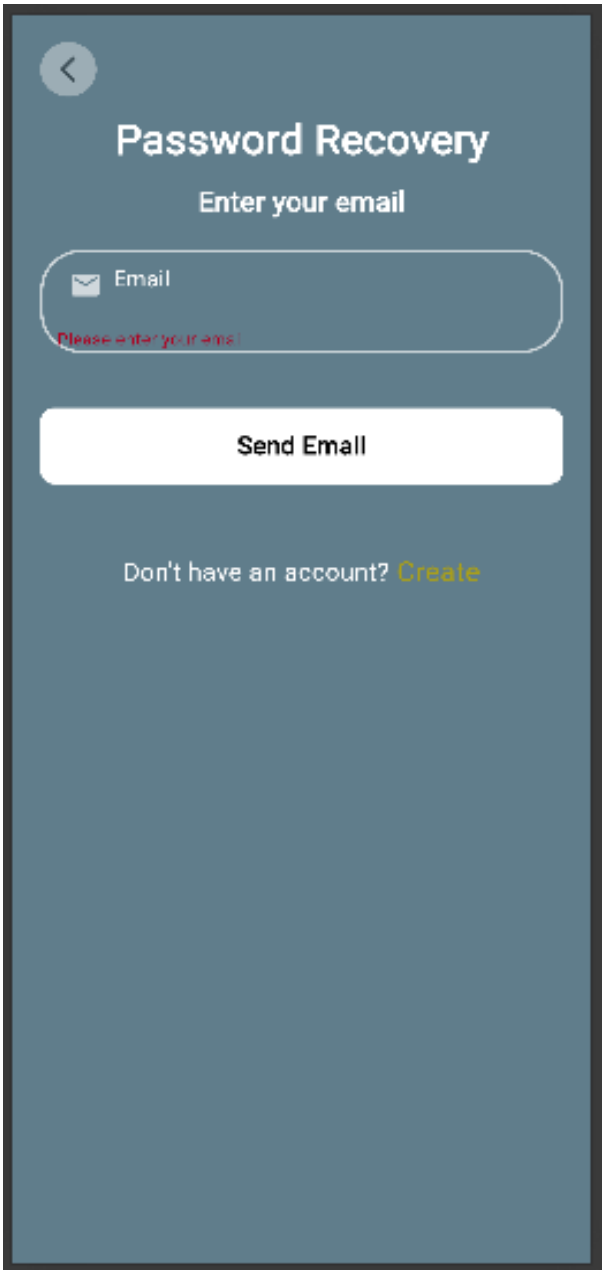
1.If user enters incorrect/incomplete email.



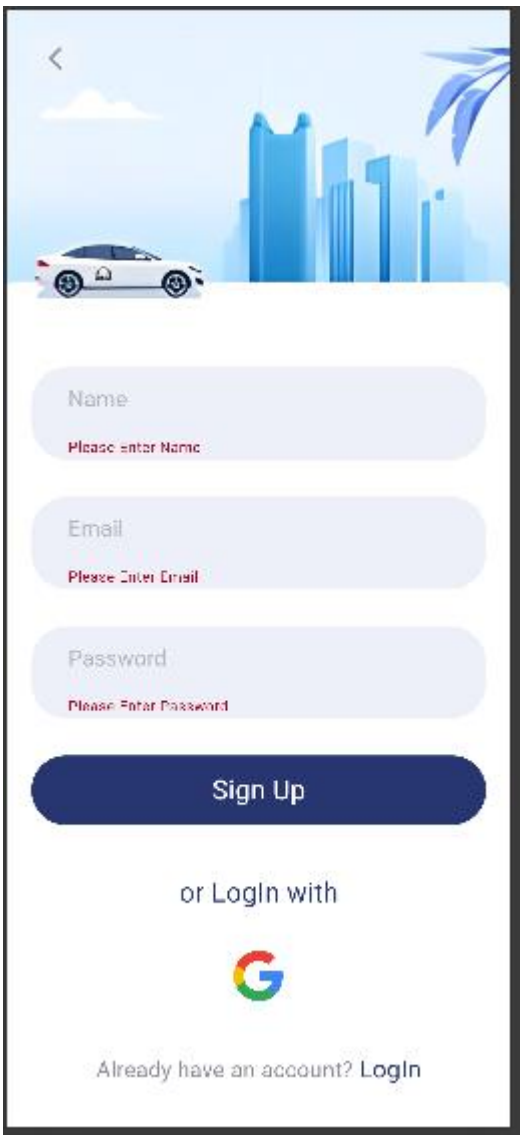
2.If user enters wrong email address.



4.Try's to enter wrong email or password



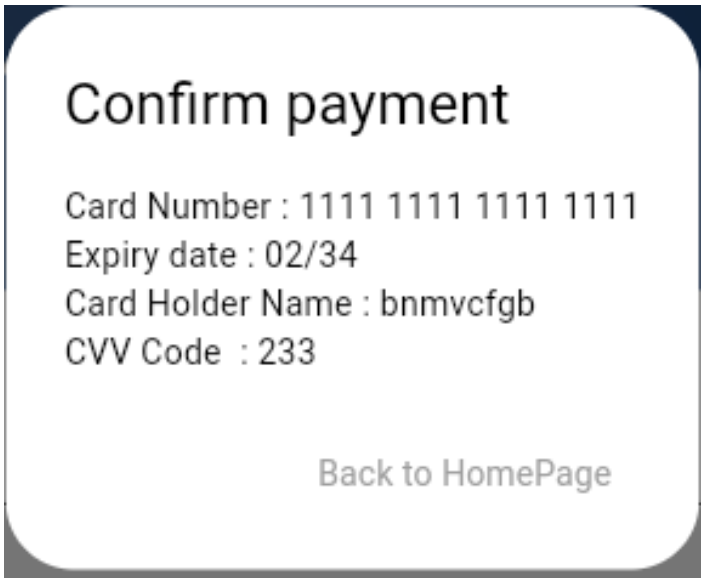
Try’s to send email with empty field or any random email , it will not send the email



Try's to enter the wrong input fields or try's it without keywords

6. Report Layout

Payment report



7. Bibliography

Mitch Koko - YouTube

Flutter – Material Design 3

Firebase Documentation (google.com)

Geeksforgeeks

https://www.youtube.com/watch?v=sv3sS9igiEw&t=2424s

draw.io

Sequence Diagrams (https://sequencediagram.org)

UML Activity Diagram (https://www.educba.com/uml-activity-diagram)

A complete Shopping App using Flutter-Part 1 | by Sanjib Maharjan | Medium