

CSE2004 DBMS DPJ - Final Project

Project on

Developing an App for the delivery/product
Management for a Supermarket”

Title

“Delivery/Product Management for a Supermarket”

Submitted by

Team number – 06

Register number – 19BIT0196

Name – Omkar Kulkarni

**School of Computer Science and Engineering,
Vellore Institute of Technology,
Vellore, Tamilnadu, India - 632014**

Topic Chosen:

The topic chosen for the project is:



“Delivery/Product management for a supermarket”



Abstract

- In recent days, due to the COVID-19 pandemic situation in the world, the most important problem faced by common public is the grocery items delivery and its product management.



An App is being made through which, all the problems regarding the product/delivery management for the retailer and the customer will be solved.

Problem Statement and its solution



Problem Statement

The retailers and shopkeepers have many issues regarding:

- 1) managing order's of people
- 2) maintaining a database of all items/ products
- 3) managing the communication of the product details between the retailer and the customer.

Click to add text

Problem Statement and its solution



Solution:

- An app is created for the above problems, where the retailer can sell his products to a customer without any problems faced, also an interface is created through which the customers can buy their products during the COVID-19 situation with full safety.

Click to add text

Flow of the app

The app has two modes:

- 1) for buyer
- 2) for seller

for buyer:

- 1) category wised the shop is displayed and each category will consist of multiple/various products
 - 2) buyer can place multiple orders of various products from various categories.
 - 3) also the buyer can view the status of the order placed and the items to be delivered.
-

Flow of the app continued

for seller:

- 1) he can list all the products by its quantity and set amount for each product.
 - 2) also He/She can view all the order information including the address and the items purchased with the final cost.
 - 3) He can specify the payment mode (by default-COD) else paytm /google pay if chosen.
-

Entities and Attributes

1) Seller Login

<u>Username</u>	Password	<u>phoneNumber</u>	<u>Shop ID</u>
-----------------	----------	--------------------	----------------

2) Customer

<u>CID</u>	Name
------------	------

3) Shop

<u>Shop ID</u>	Location	<u>Phone no</u>	<u>Owner Name</u>	Category
----------------	----------	-----------------	-------------------	----------

Entities and Attributes

4) Products

Name	Price	Quantity	<u>prodID</u>	<u>Pno</u>	Company
------	-------	----------	---------------	------------	---------

5) Review

Cid	<u>shopID</u>	Comments	
-----	---------------	----------	--

Relationship

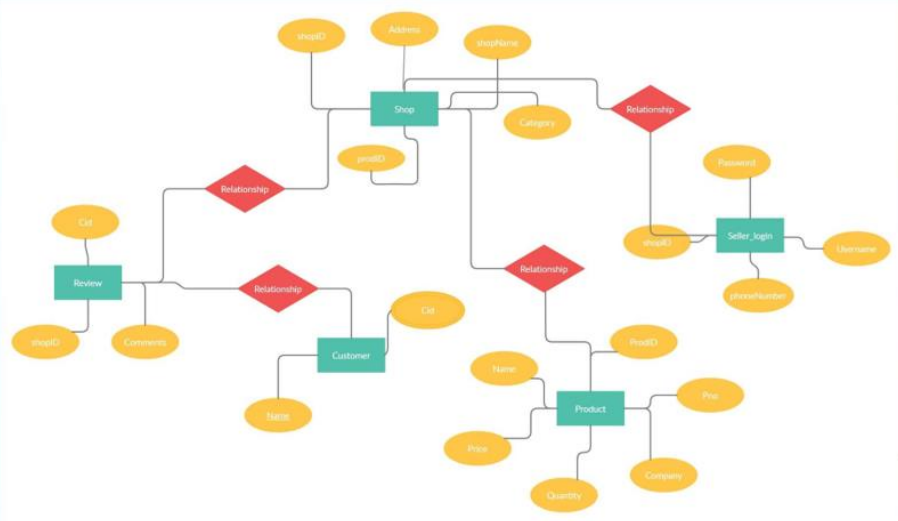
SELLER LOGIN AND SHOP HAVE ONE TO ONE
RELATIONSHIP

PRODUCT AND SHOP HAVE MANY TO ONE
RELATIONSHIP

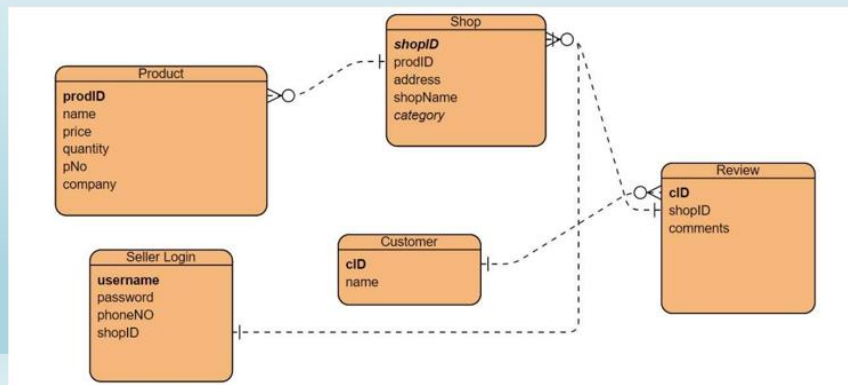
CUSTOMER AND REVIEW HAVE ONE TO MANY
RELATIONSHIP

REVIEW AND SHOP HAVE MANY TO ONE
RELATIONSHIP

•ER Diagram



Conceptual Schema



Implementation of the schema

1. SELLER LOGIN

```
CREATE TABLE SELLER_LOGIN(  
  USERNAME VARCHAR(20) PRIMARY KEY,  
  PASSWORD VARCHAR(20) NOT NULL,  
  PHONE_NO NUMBER(10) NOT NULL,  
  SHOP_ID NUMBER(10) REFERENCES SHOP(SHOP_ID)  
);
```

Implementation of the schema continued

2. CUSTOMER TABLE

```
CREATE TABLE CUSTOMER(  
  C_ID NUMBER(20) PRIMARY KEY,  
  ADDRESS VARCHAR(40),  
  NAME VARCHAR(30) NOT NULL,  
  EMAIL VARCHAR(40),  
  CONTACT_NO NUMBER(10) NOT NULL  
);
```

Implementation of the schema continued

3. BUYER LOGIN

```
CREATE TABLE BUYER_LOGIN(  
  USERNAME VARCHAR(20) NOT NULL,  
  PASSWORD VARCHAR(20) NOT NULL,  
  PHONE_NO NUMBER(10) NOT NULL,  
  C_ID NUMBER(20) REFERENCES CUSTOMER(C_ID) ON DELETE  
  CASCADE  
);
```

Implementation of the schema continued

4. SHOP TABLE

```
CREATE TABLE SHOP(  
  SHOP_ID NUMBER(20) PRIMARY KEY,  
  LOCATION VARCHAR(40) NOT NULL,  
  PHONE_NO NUMBER(10) NOT NULL,  
  OWNER_NAME VARCHAR(30) NOT NULL,  
  CATEGORY VARCHAR(20) NOT NULL  
);
```

Implementation of the schema continued

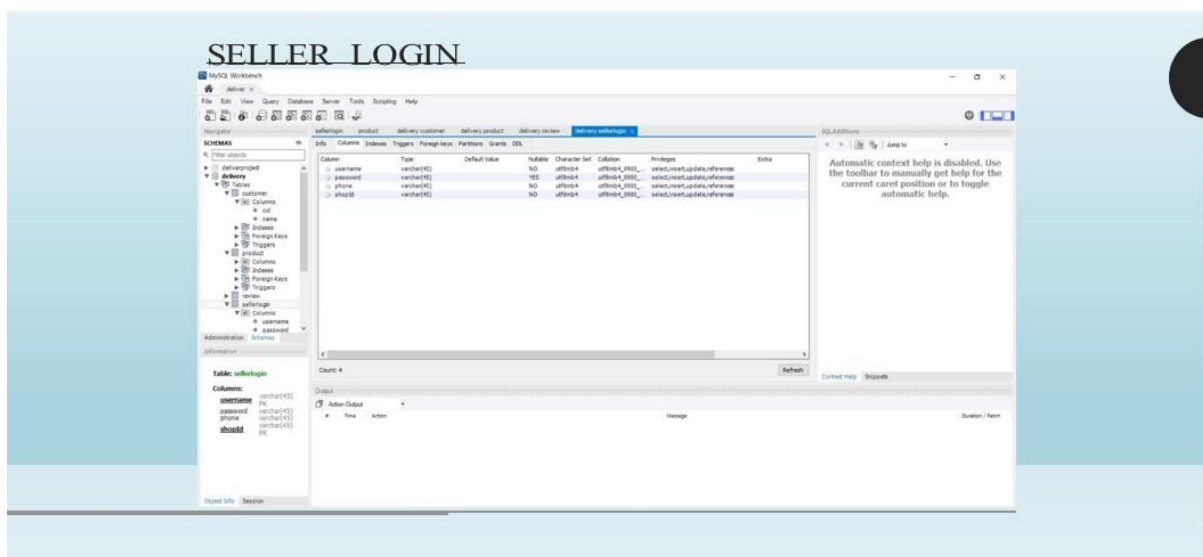
5. PRODUCT TABLE

```
CREATE TABLE PRODUCTS(  
  P_ID NUMBER(20) PRIMARY KEY,  
  SHOP_ID NUMBER(20) REFERENCES SHOP(SHOP_ID),  
  P_NAME VARCHAR(30) NOT NULL,  
  BRAND VARCHAR(30) NOT NULL,  
  QTY NUMBER(10) NOT NULL,  
  PRICE NUMBER(5) NOT NULL  
);
```

Implementation of the schema continued

1. SELLER LOGIN

```
CREATE TABLE SELLER_LOGIN(  
  USERNAME VARCHAR(20) PRIMARY KEY,  
  PASSWORD VARCHAR(20) NOT NULL,  
  PHONE_NO NUMBER(10) NOT NULL,  
  SHOP_ID NUMBER(10) REFERENCES SHOP(SHOP_ID)  
);
```



CUSTOMER

The screenshot shows the MySQL Workbench interface with the 'customer' table selected in the 'Schemas' pane. The 'Table: customer' information pane on the left lists the columns: 'id' (varchar(20)) and 'name' (varchar(45)). The main pane displays the table structure with the following columns:

Column	Type	Default value	Nullable	Character Set	Collation	Privileges	Extra
id	varchar(20)		NO	utf8mb4	utf8mb4_2500...	select,insert,update,references	
name	varchar(45)		NO	utf8mb4	utf8mb4_2500...	select,insert,update,references	

The right pane shows a message: "Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help."

Table Structure

CUSTOMER REVIEW

The screenshot shows the MySQL Workbench interface with the 'customer_review' table selected in the 'Schemas' pane. The 'Table: customer_review' information pane on the left lists the columns: 'id' (varchar(45)), 'comment' (varchar(45)), and 'shopid' (varchar(45)). The main pane displays the table structure with the following columns:

Column	Type	Default value	Nullable	Character Set	Collation	Privileges	Extra
id	varchar(45)		NO	utf8mb4	utf8mb4_2500...	select,insert,update,references	
comment	varchar(45)		NO	utf8mb4	utf8mb4_2500...	select,insert,update,references	
shopid	varchar(45)		NO	utf8mb4	utf8mb4_2500...	select,insert,update,references	

The right pane shows a message: "Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help."

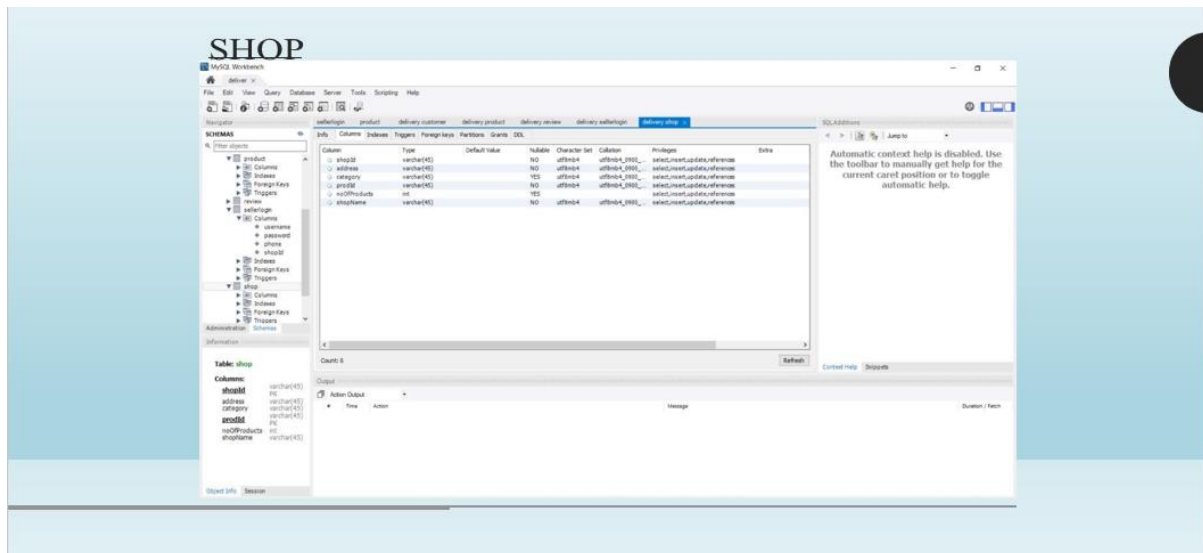
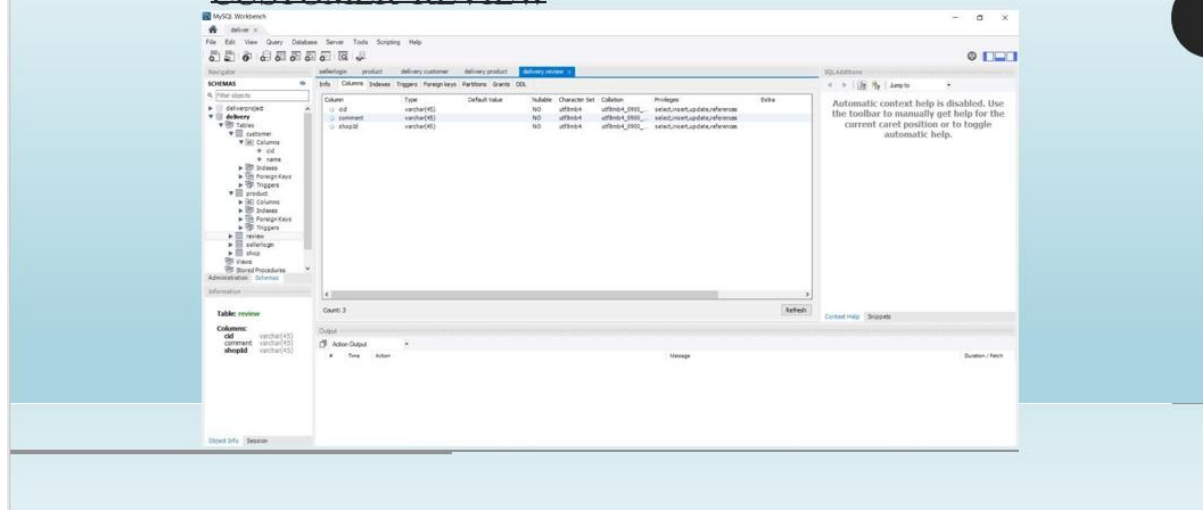


Table Structure

CUSTOMER REVIEW



Note:

The Normalisation for the tables done ahead is done on random values taken by the application. The example of normalisation is given in the next slides. The database is just having tables. The data which is entered in a database can be done through the app made.

Normalisation

SHOP TABLE

Shop id	location	Phone number	Owner name	Category
1	Bangalore	457	Ramesh	Electronics
2	Chennai	545	Suresh	Decoration
3	Bangalore	552	Ramesh	Electronics
4	Mumbai	987	Jay	Sports

Normalization of tables

SELLER_LOGIN

SELLER_LOGIN(USERNAME, PHONENUMBER, PASSWORD, SHOPID)

SELLER_LOGIN(USERNAME \square PHONENUMBER, PHONENUMBER \square SHOPID, SHOPID \square USERNAME)

CANDIDATE KEY = {USERNAME, PHONENUMBER, SHOPID}

PRIME ATTRIBUTES = {USERNAME, PHONENUMBER, SHOPID}

1ST NORMAL FORM

THERE IS NO MULTIVALUED ATTRIBUTE SO IT IS IN 1ST NORMAL FORM.

2ND NORMAL FORM

THERE SHOULD BE NO PARTIAL DEPENDENCY (LHS MUST BE PROPER SUBSET OF ANY CANDIDATE KEY AND RHS MUST BE A NON-PRIME ATTRIBUTE).

USERNAME \square PHONENUMBER

Normalization of tables contd

USERNAME IS NOT A PROPER SUBSET AND PHONENUMBER IS PRIME ATTRIBUTE ($T \&\& T = T$).

THEREFORE, FULL DEPENDENCY.

PHONENUMBER \square SHOPID

PHONENUMBER IS NOT A PROPER SUBSET AND SHOPID IS PRIME ATTRIBUTE ($T \&\& T = T$).

THEREFORE, FULL DEPENDENCY.

SHOPID \square USERNAME

SHOPID IS NOT A PROPER SUBSET AND USERNAME IS PRIME ATTRIBUTE ($T \&\& T = T$). THEREFORE,

FULL DEPENDENCY.

THEREFORE, THERE ARE NO PARTIAL DEPENDENCIES.

THEREFORE, IT IS IN 2ND NORMAL FORM.

3RD NORMAL FORM

LHS MUST BE A CANDIDATE KEY OR RHS MUST BE A PRIME ATTRIBUTE.

USERNAME \square PHONENUMBER

Normalization of tables contd

USERNAME IS A CANDIDATE KEY AND PHONENUMBER IS A PRIME ATTRIBUTE ($T \parallel T = T$).

THEREFORE, THE CONDITION IS TRUE.

PHONENUMBER \square SHOPID

PHONENUMBER IS A CANDIDATE KEY AND SHOPID IS A PRIME ATTRIBUTE ($T \parallel T = T$).

THEREFORE, THE CONDITION IS TRUE.

SHOPID \square USERNAME

SHOPID IS A CANDIDATE KEY AND USERNAME IS A PRIME ATTRIBUTE ($T \parallel T = T$). THEREFORE, THE

CONDITION IS TRUE.

THEREFORE, IT IS IN 3RD NORMAL FORM.

4TH NORMAL FORM (BCNF)

LHS MUST BE A CANDIDATE KEY.

Normalization of tables contd

USERNAME \square PHONENUMBER

USERNAME IS A CANDIDATE KEY. THEREFORE, THE CONDITION IS TRUE.

PHONENUMBER \square SHOPID

PHONENUMBER IS A CANDIDATE KEY. THEREFORE, THE CONDITION IS TRUE.

SHOPID \square USERNAME

SHOPID IS A CANDIDATE KEY. THEREFORE, THE CONDITION IS TRUE.

THEREFORE, IT IS IN BCNF.

Normalization of tables contd

SHOP

SHOP (SHOPID, ADDRESS, SHOPNAME, PRODID, CATEGORY)

CANDIDATE KEY = {SHOPID, ADDRESS, PRODID}

SHOP (SHOPID \square ADDRESS, ADDRESS \square PRODID, PRODID \square SHOPID)

PRIME ATTRIBUTE = {SHOPID, ADDRESS, PRODID}

1ST NORMAL FORM

THERE IS NO MULTIVALUED ATTRIBUTE SO IT IS IN 1ST NORMAL FORM.

2ND NORMAL FORM

THERE SHOULD BE NO PARTIAL DEPENDENCY (LHS MUST BE PROPER SUBSET OF ANY CANDIDATE KEY AND RHS MUST BE A NON-PRIME ATTRIBUTE).

Normalization of tables contd

SHOPID \square ADDRESS

SHOPID IS NOT A PROPER SUBSET AND ADDRESS IS PRIME ATTRIBUTE ($T \&\& T = T$). THEREFORE, FULL DEPENDENCY.

ADDRESS \square PRODID

PHONENUMBER IS NOT A PROPER SUBSET AND PRODID IS PRIME ATTRIBUTE ($T \&\& T = T$). THEREFORE, FULL DEPENDENCY.

PRODID \square SHOPID

SHOPID IS NOT A PROPER SUBSET AND PRODID IS PRIME ATTRIBUTE ($T \&\& T = T$). THEREFORE, FULL DEPENDENCY.

THEREFORE, THERE ARE NO PARTIAL DEPENDENCIES.

THEREFORE, IT IS IN 2ND NORMAL FORM.

3RD NORMAL FORM

LHS MUST BE A CANDIDATE KEY OR RHS MUST BE A PRIME ATTRIBUTE.

Normalization of tables contd

SHOPID \square ADDRESS

SHOPID IS A CANDIDATE KEY AND ADDRESS IS A PRIME ATTRIBUTE ($T \parallel T = T$). THEREFORE, THE CONDITION IS TRUE.

ADDRESS \square PRODID

ADDRESS IS A CANDIDATE KEY AND SHOPID IS A PRIME ATTRIBUTE ($T \parallel T = T$). THEREFORE, THE CONDITION IS TRUE.

PRODID \square SHOPID

SHOPID IS A CANDIDATE KEY AND PRODID IS A PRIME ATTRIBUTE ($T \parallel T = T$). THEREFORE, THE CONDITION IS TRUE.

THEREFORE, IT IS IN 3RD NORMAL FORM.

4TH NORMAL FORM (BCNF)

LHS MUST BE A CANDIDATE KEY.

Normalization of tables contd

PRODUCT

PRODUCT (NAME, PRICE, QUANTITY, PRODID, PNO, COMPANY)

CANDIDATE KEY={PNO}

PRIME ATTRIBUTES={PNO}

IT IS NORMALIZED.

Normalization of tables contd

CUSTOMER

CUSTOMER (CID, NAME)

CUSTOMER (CID \square NAME)

CANDIDATE KEY = {CID, NAME}

PRIME ATTRIBUTES= {CID, NAME}

1ST NORMAL FORM

THERE IS NO MULTIVALUED ATTRIBUTE SO IT IS IN 1ST NORMAL FORM.

2ND NORMAL FORM

THERE SHOULD BE NO PARTIAL DEPENDENCY (LHS MUST BE PROPER SUBSET OF ANY CANDIDATE KEY AND RHS MUST BE A NON-PRIME ATTRIBUTE).

CID \square NAME

Normalization of tables contd

CID IS NOT A PROPER SUBSET AND NAME IS PRIME ATTRIBUTE ($T \twoheadrightarrow T = T$). THEREFORE, FULL DEPENDENCY.

THEREFORE, IT IS IN 2ND NORMAL FORM.

3RD NORMAL FORM

CID \square NAME

CID IS A CANDIDATE KEY AND NAME IS A PRIME ATTRIBUTE ($T \parallel T = T$). THEREFORE, THE CONDITION IS TRUE.

THEREFORE, IT IS IN 3RD NORMAL FORM.

4TH NORMAL FORM (BCNF)

CID \square NAME

CID IS A CANDIDATE KEY. THEREFORE, THE CONDITION IS TRUE.

THEREFORE, IT IS IN BCNF

Frontend and backend connectivity code

```
IMPORT 'DART:CONVERT';
IMPORT 'PACKAGE:FLUTTER/CUPERTINO.DART';
IMPORT 'PACKAGE:HTTP/HTTP.DART' AS HTTP;

CLASS SHOP {
  STRING SHOPID;
  STRING SHOPNAME;
  STRING ADDRESS;
  STRING CATEGORY;
  INT NOOFFPRODUCTS;
  SHOP(
    {THIS.SHOPNAME,
     THIS.ADDRESS,
     THIS.CATEGORY,
     THIS.NOOFFPRODUCTS,
     THIS.SHOPID});
}
```

Frontend and backend connectivity code contd.

```
CLASS PRODUCT {
  STRING NAME;
  INT PRICE;
  STRING COMPANY;
  INT QUANTITY;
  STRING PNO;
  PRODUCT({THIS.COMPANY, THIS.NAME, THIS.PNO, THIS.PRICE, THIS.QUANTITY});
}

CLASS CUSTOMERVIEW WITH CHANGENOTIFIER {
  STRING PHONENUMBER;
  LIST<DYNAMIC> SHOPS = [];
  STRING PRODID;
  STRING SHOPNAME;
  STRING ADDRESS;
  INT NOOFFPRODUCTS;
  STRING CATEGORY;
  LIST<DYNAMIC> PRODUCT = [];
  STRING CID;
  LIST<DYNAMIC> COMMENTS = [];
}
```

Frontend and backend connectivity code contd.

```
FUTURE<VOID> GETALLSHOPS() ASYNC {
    FINAL RESPONSE = AWAIT HTTP.GET(
        'HTTP://10.0.2.2:3000/SHOPS/ALL',
    );
    FINAL RESPONSEBODY = JSONDECODE(RESPONSE.BODY);
    PRINT(RESPONSEBODY);
    LIST SHOP = RESPONSEBODY['DATA']
        .MAP(
            (I) => SHOP(
                SHOPNAME: I['SHOPNAME'],
                ADDRESS: I['ADDRESS'],
                CATEGORY: I['CATEGORY'],
                NOOFPRODUCTS: I['NOOFPRODUCTS'],
                SHOPID: I['SHOPID'],
            )
        )
        .TOLIST();
    SHOPS = SHOP;
}
```

Frontend and backend connectivity code contd.

```
FUTURE<VOID> GETSHOPBYID(String ID) ASYNC {
    FINAL RESPONSE = AWAIT HTTP.POST('HTTP://10.0.2.2:3000/SHOP/ID',
        HEADERS: <STRING, STRING>{
            'CONTENT-TYPE': 'APPLICATION/JSON; CHARSET=UTF-8',
        },
        BODY: JSONENCODE(<STRING, DYNAMIC>{'SHOPID': ID}));
    FINAL RESPONSEBODY = JSONDECODE(RESPONSE.BODY);
    PRODID = RESPONSEBODY['DATA'][0]['PRODID'];
    SHOPNAME = RESPONSEBODY['DATA'][0]['SHOPNAME'];
    ADDRESS = RESPONSEBODY['DATA'][0]['ADDRESS'];
    CATEGORY = RESPONSEBODY['DATA'][0]['CATEGORY'];
    NOOFPRODUCTS = RESPONSEBODY['DATA'][0]['NOOFPRODUCTS'];

    PRINT(RESPONSEBODY);
}
```

Frontend and backend connectivity code contd.

```
FUTURE<VOID> GETPRODUCTS(String PRODID) ASYNC {
    FINAL RESPONSE = AWAIT HTTP.POST('HTTP://10.0.2.2:3000/PRODUCT',
        HEADERS: <STRING, STRING>{
            'CONTENT-TYPE': 'APPLICATION/JSON; CHARSET=UTF-8',
        },
        BODY: JSONENCODE(<STRING, DYNAMIC>{'PRODID': PRODID}));
    FINAL RESPONSEBODY = JSONDECODE(RESPONSE.BODY);
    LIST PRO = RESPONSEBODY['DATA']
        .MAP((I) => PRODUCT(
            PNO: I['PNO'],
            COMPANY: I['COMPANY'],
            NAME: I['NAME'],
            QUANTITY: I['QUANTITY'],
            PRICE: I['PRICE'],
        ))
        .TOLIST();
    PRODUCT = PRO;
}
```

Frontend and backend connectivity code contd.

```
FUTURE<VOID> GETSHOPOWNER(String ID) ASYNC {  
    FINAL RESPONSE = AWAIT HTTP.POST('HTTP://10.0.2.2:3000/SHOPOWNER',  
        HEADERS: <String, String>{  
            'CONTENT-TYPE': 'APPLICATION/JSON'; CHARSET=UTF-8',  
        },  
        BODY: JSONENCODE(<String, Dynamic>{'SHOPID': ID}));  
    FINAL RESPONSEBODY = JSONDECODE(RESPONSE.BODY);  
    PHONENUMBER = RESPONSEBODY['DATA'][0]['PHONE'];  
  
    PRINT(RESPONSEBODY);  
}
```

Frontend and backend connectivity code contd.

```
FUTURE<VOID> ADDCUSTOMER(String NAME) ASYNC {  
    FINAL RESPONSE = AWAIT HTTP.POST('HTTP://10.0.2.2:3000/ADD/CUSTOMER',  
        HEADERS: <String, String>{  
            'CONTENT-TYPE': 'APPLICATION/JSON'; CHARSET=UTF-8',  
        },  
        BODY: JSONENCODE(<String, Dynamic>{'NAME': NAME}));  
    FINAL RESPONSEBODY = JSONDECODE(RESPONSE.BODY);  
    CID = RESPONSEBODY['CID'];  
  
    PRINT(RESPONSEBODY);  
}
```

Frontend and backend connectivity code contd.

```
FUTURE<VOID> GETREVS(String SHOPID) ASYNC {  
    FINAL RESPONSE = AWAIT HTTP.POST('HTTP://10.0.2.2:3000/COMMENTS',  
        HEADERS: <String, String>{  
            'CONTENT-TYPE': 'APPLICATION/JSON'; CHARSET=UTF-8',  
        },  
        BODY: JSONENCODE(<String, Dynamic>{'SHOPID': SHOPID}));  
    FINAL RESPONSEBODY = JSONDECODE(RESPONSE.BODY);  
    LIST Y = RESPONSEBODY['DATA'].MAP((I) => I['COMMENT']).TOLIST();  
    COMMENTS = Y;  
    NOTIFYLISTENERS();  
}
```

Frontend and backend connectivity code contd.

```
FUTURE<VOID> ADDCOMMENT({STRING COMMENT, STRING CID, STRING SHOPID}) ASYNC {  
  FINAL RESPONSE = AWAIT HTTP.POST('HTTP://10.0.2.2:3000/ADD/COMMENT',  
    HEADERS: <STRING, STRING>{  
      'CONTENT-TYPE': 'APPLICATION/JSON; CHARSET=UTF-8',  
    },  
    BODY: JSONENCODE(<STRING, DYNAMIC>{  
      'SHOPID': SHOPID,  
      'CID': CID,  
      'COMMENT': COMMENT  
    }));  
  FINAL RESPONSEBODY = JSONDECODE(RESPONSE.BODY);  
  PRINT(RESPONSEBODY);  
}
```

Hardware and software requirements to install your software:

Hardware Requirements:

1. Any laptop or pc with minimum of 8gb ram and 4Gb storage space available
2. Any Laptop with a 2gb graphic card.

Software Requirements:

1. Android Studio must be installed in your device.
2. Any code running application like Visual Studio Code which gives an environment to code must be installed.
3. Git and GitHub Desktop application can be used for saving your project in your repository.
4. MySQL Must be Installed.
5. Flutter must be installed.

Software used/required: 1) Front end: Flutter (dart)
2) Database: MySQL
3) Server: Android Studio Emulator.

Database details:

No. of tables as per the normalized schema: 8

No. of tables in the final project : 8

Front end details:

How many interface pages? : 15

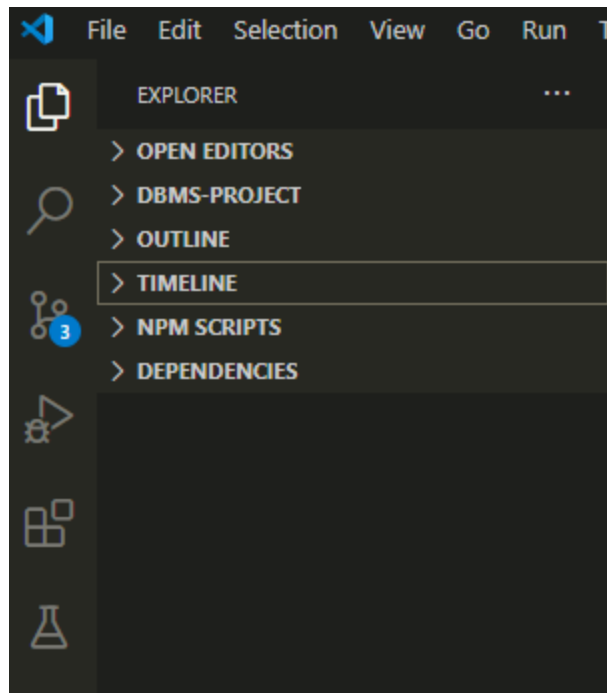
Type of interface (web page/application) : Android Studio Emulator

HELP:

How to open the files in VS Code and connect to Android Studio:

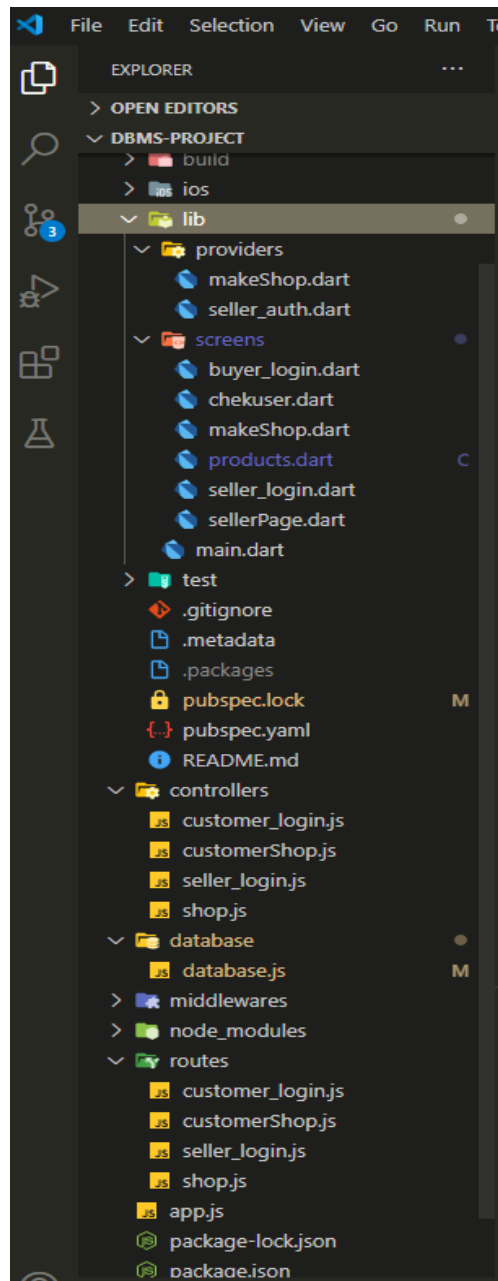
The Zip File Provided has the code and files required to run this project.

- 1) Open the entire “dbms project” folder in VS Code application.



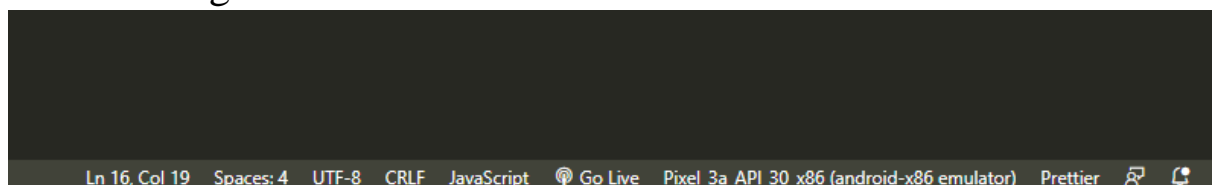
You can see the projects you are working on in the explorer section which is available on the left side of the VS Code.

- 2) Now Double tap the “DBMS-PROJECT” section to view the files inside.



Thus, you get the view of all the files inside the project.

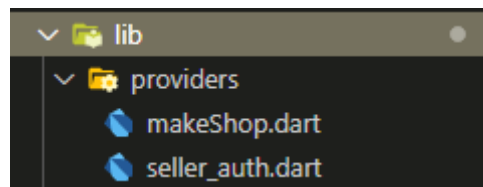
- 3) Now you connect VS Code to Android Studio Emulator (After installing Android Studio). You must see this after its connected in bottom right section of VS Code:



We can see that it's connected to Android Studio Emulator's Pixel 3a phone.

Understanding The files:

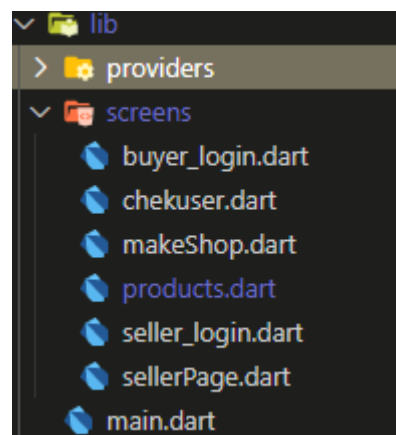
1) Providers:



The Project's Front-end is made in flutter (.dart files) and the backend is done in JavaScript (.js files) and the database is connected through MySQL application.

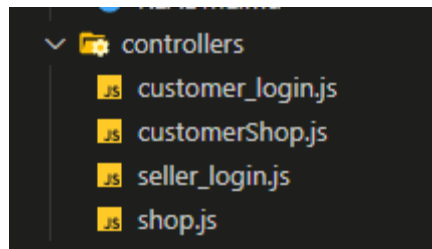
These two files have the Integration of Front-end with the Back-end. In These two Files the API has been connected via routes.

2) Screens:



All the files in this folder provides the front-end code for the designing of each page in the application. The Code of these files is written in Flutter.

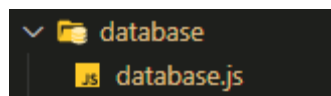
3) Controllers:



These are the back-end JavaScript files. So, I used json web token system (JWT) system here on the seller login which encrypts the data being entered. Through this system we get to know whether the user is locked or not.

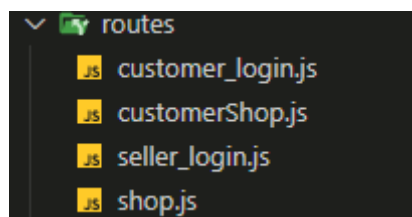
Token is used to insert data into shop table and products tables. The data is made into json object format this then converts and restructures it with API to reach the frontend code.

4) Database:



This file connects the MySQL database with the back-end code.

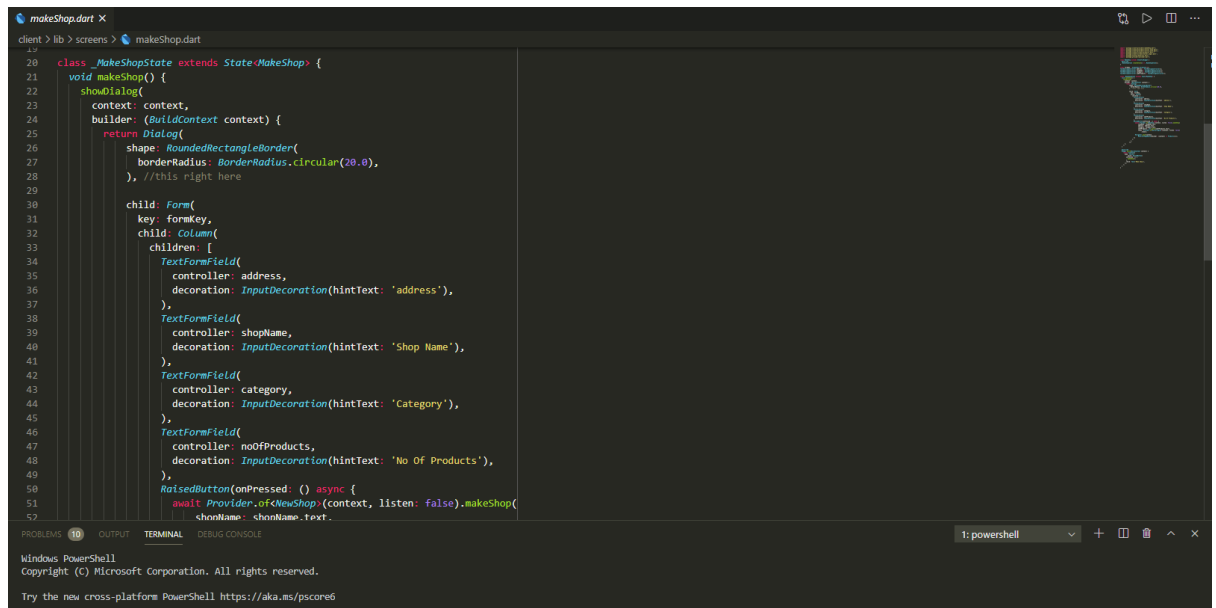
5) Routes:



These are routes (back-end files) which are connected to the front-end API files in the Providers section.

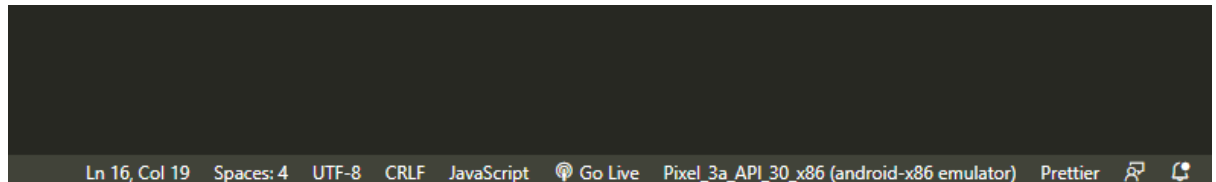
Running the Application:

- 1) Running the Application is very easy through Flutter.
- 2) We open any .dart file first.

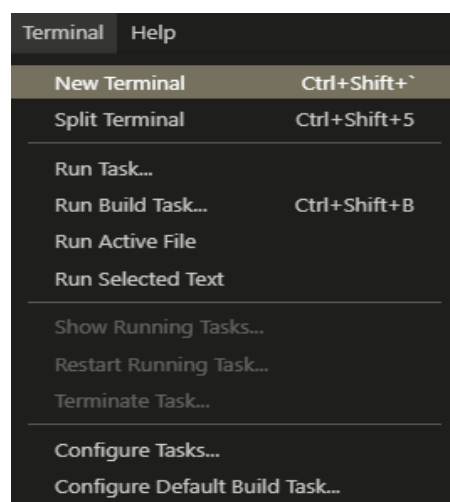


```
client > lib > screens > makeShop.dart
20 class _MakeShopState extends State<MakeShop> {
21   void makeShop() {
22     showDialog(
23       context: context,
24       builder: (BuildContext context) {
25         return Dialog(
26           shape: RoundedRectangleBorder(
27             borderRadius: BorderRadius.circular(20.0),
28           ), //this right here
29         );
30         child: Form(
31           key: formKey,
32           child: Column(
33             children: [
34               TextFormField(
35                 controller: address,
36                 decoration: InputDecoration(hintText: 'address'),
37               ),
38               TextFormField(
39                 controller: shopName,
40                 decoration: InputDecoration(hintText: 'Shop Name'),
41               ),
42               TextFormField(
43                 controller: category,
44                 decoration: InputDecoration(hintText: 'Category'),
45               ),
46               TextFormField(
47                 controller: noOfProducts,
48                 decoration: InputDecoration(hintText: 'No Of Products'),
49               ),
50               RaisedButton(onPressed: () async {
51                 await Provider.of<NewShop>(context, listen: false).makeShop(
52                   shopName: shopName.text,
53                 );
54               },
55             ),
56           ],
57         ),
58       ),
59     );
60   }
61 }
```

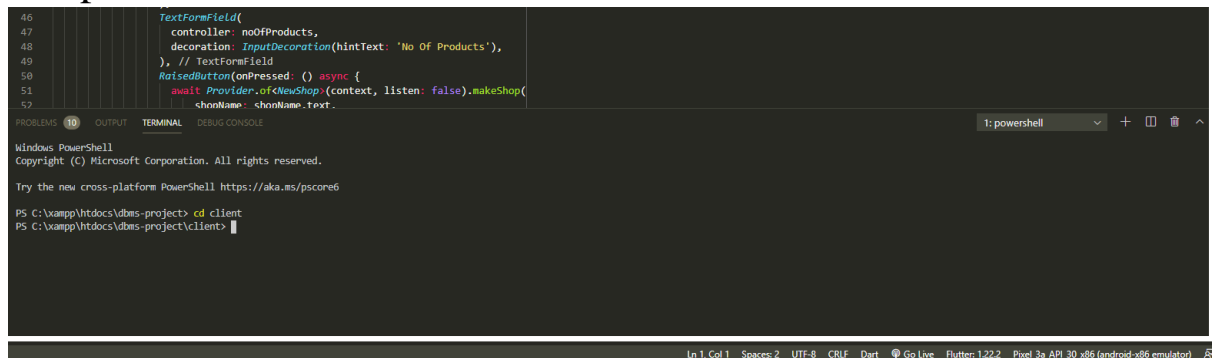
- 3) Make sure you have connected VS Code with Android Studio as mentioned above.



- 4) Open terminal in VS Code.



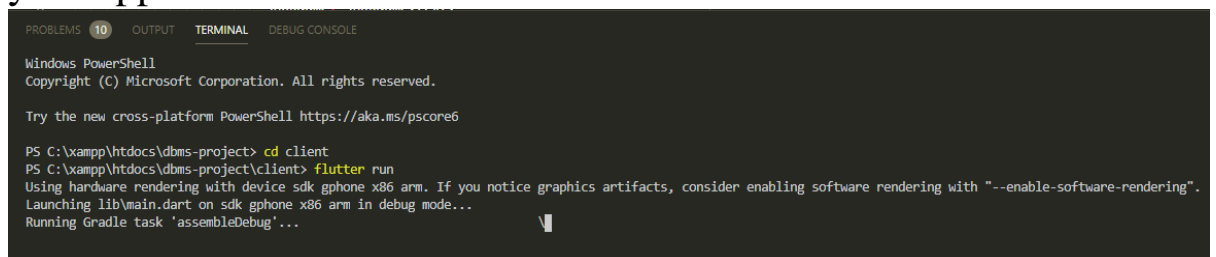
- 5) See if the file is in client or not. Or else simply type “cd client” and press enter in terminal.



```
46       TextFormField(  
47         controller: noOfProducts,  
48         decoration: InputDecoration(hintText: 'No Of Products'),  
49       ), // TextFormField  
50       RaisedButton(onPressed: () async {  
51         await Provider.of<NewShop>(context, listen: false).makeShop(  
52           shopName: shopName.text,  
        
```

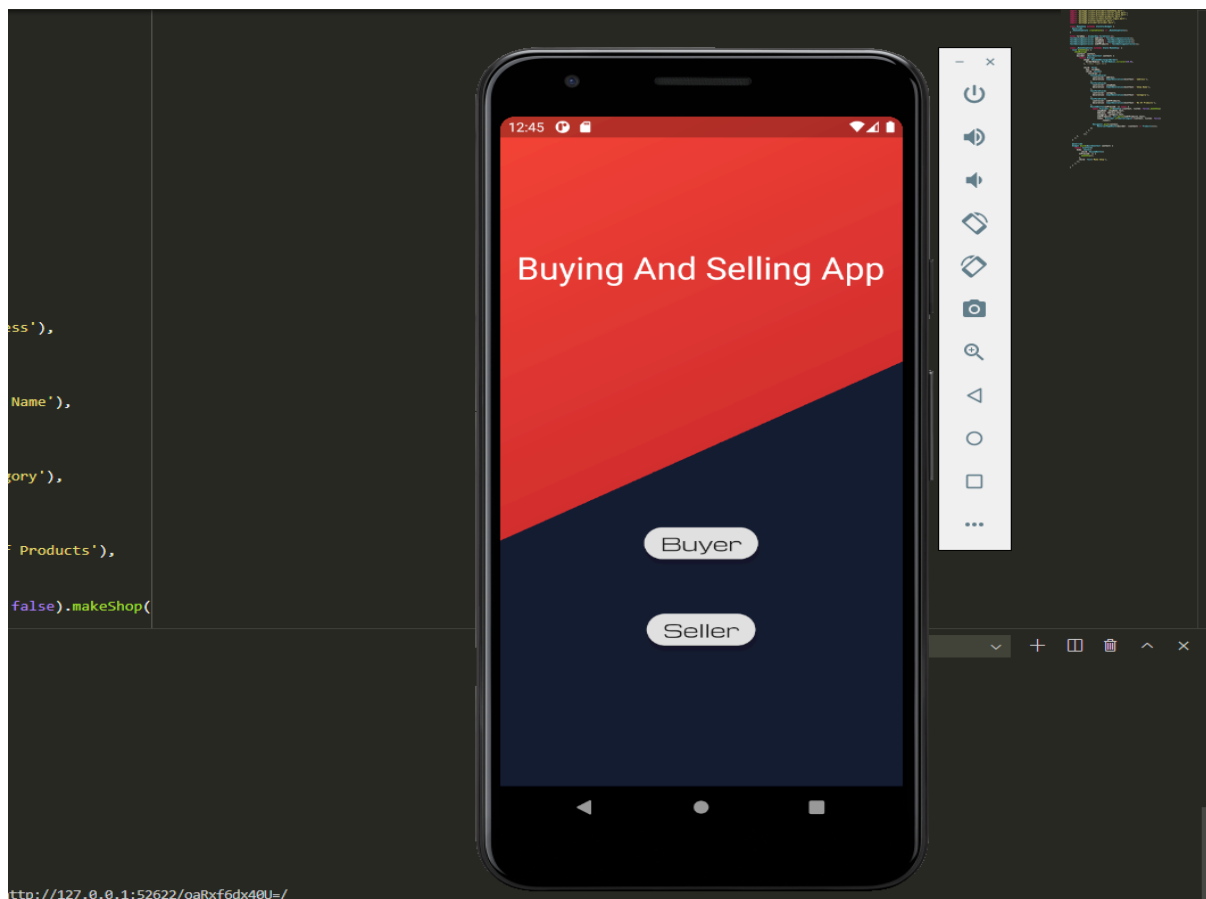
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.
Try the new cross-platform PowerShell <https://aka.ms/pscore6>
PS C:\xampp\htdocs\vbms-project> cd client
PS C:\xampp\htdocs\vbms-project\client>

- 6) Now you can type “flutter run” and press enter in terminal to run your app in android studio.



```
Windows PowerShell  
Copyright (C) Microsoft Corporation. All rights reserved.  
Try the new cross-platform PowerShell https://aka.ms/pscore6  
PS C:\xampp\htdocs\vbms-project> cd client  
PS C:\xampp\htdocs\vbms-project\client> flutter run  
Using hardware rendering with device sdk gphone x86 arm. If you notice graphics artifacts, consider enabling software rendering with "--enable-software-rendering".  
Launching lib/main.dart on sdk gphone x86 arm in debug mode...  
Running Gradle task 'assembleDebug'...
```

- 7) Now u can see your application running in Android Studio!



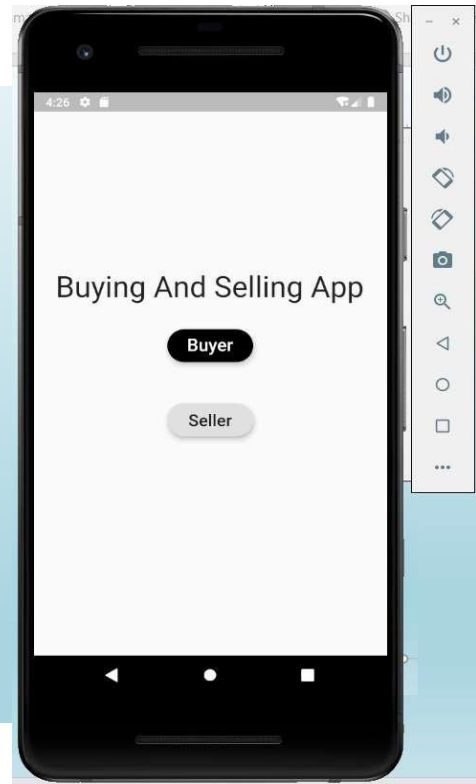
- 8) The app provides 2 sections, the buyer and seller. Buyer can view products and buy them. The Seller section can add products in the database.
- 9) The Database and its tables have been provided in the review 1 and 2 ppt file given in the zip file. The tables have its values inserted via the app.

You can exit the app by closing the emulator. Also you can restart it by pressing ctrl+R.

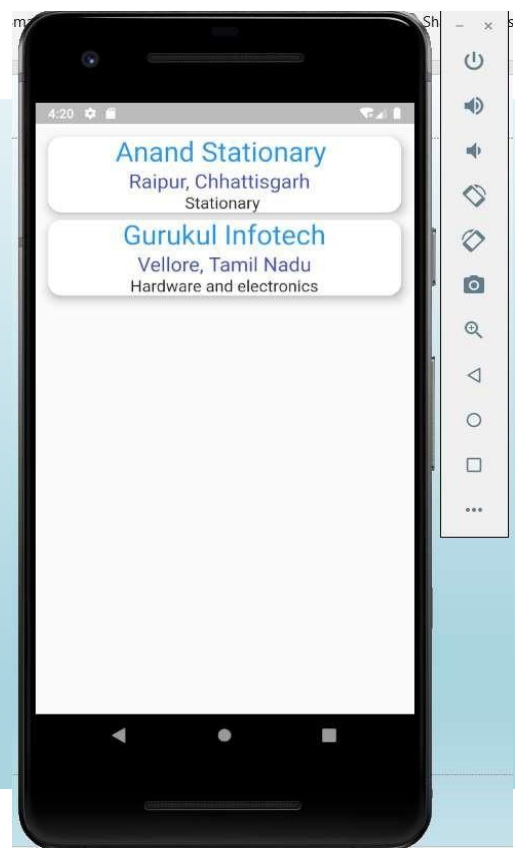
Flow of the application:

1) Once you open the App, we see two options. Buyer or Seller

Frontend
1. First you
will be
asked are
you a buyer
or a seller.

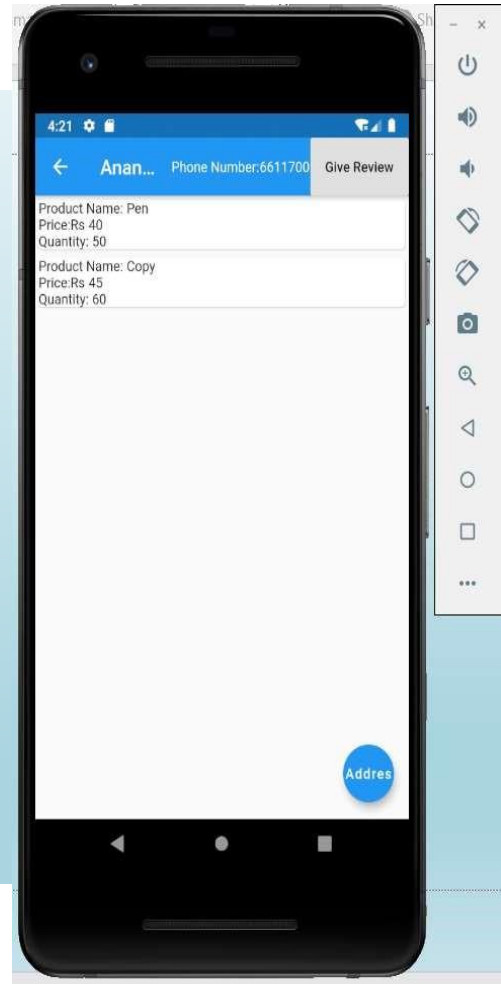


Frontend
2. If you select
the buyer
option then you
will be shown
the list of shops
from where you
can buy.



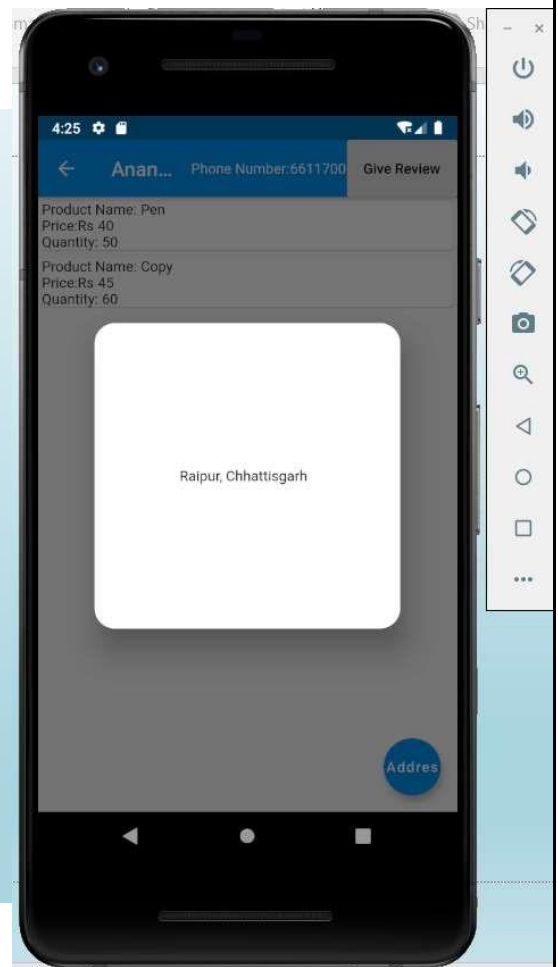
Frontend

3. Selecting a particular shop will list all the products that are available in the shop.

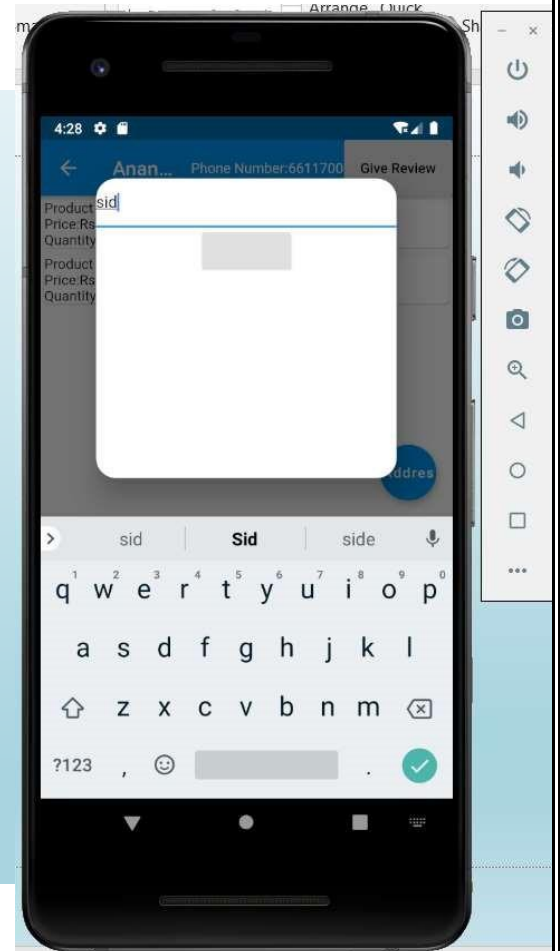


Frontend

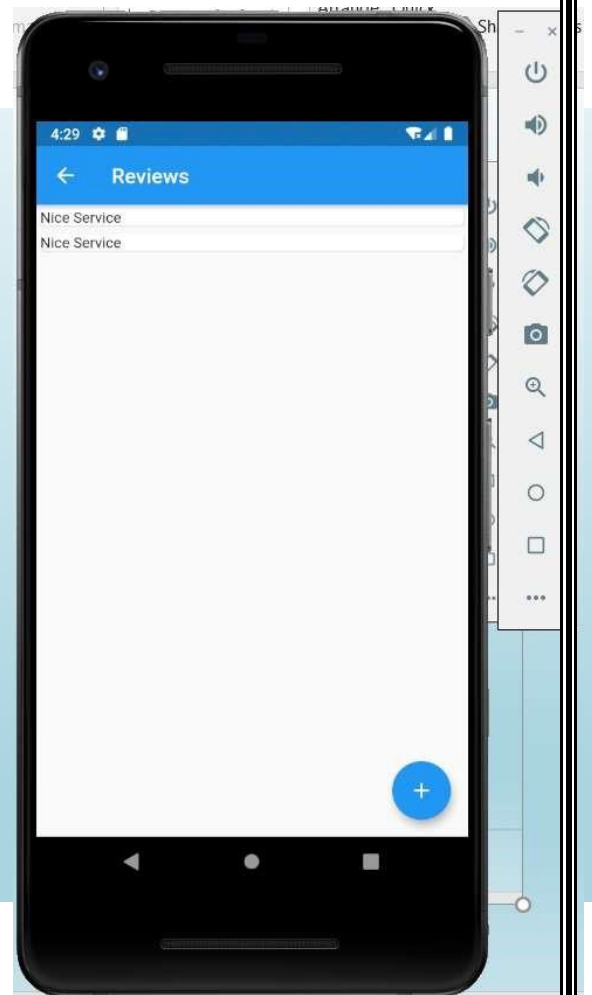
4. If you select the address button then the complete address will be shown to you.



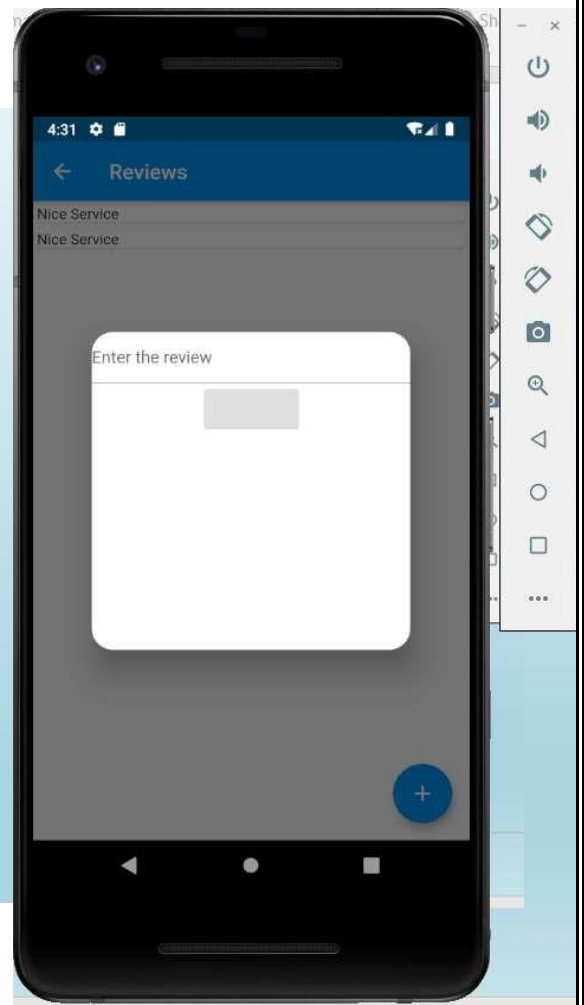
Frontend
5. Buyer can
giver review
about the shop
by give review
button. He will
be asked name.



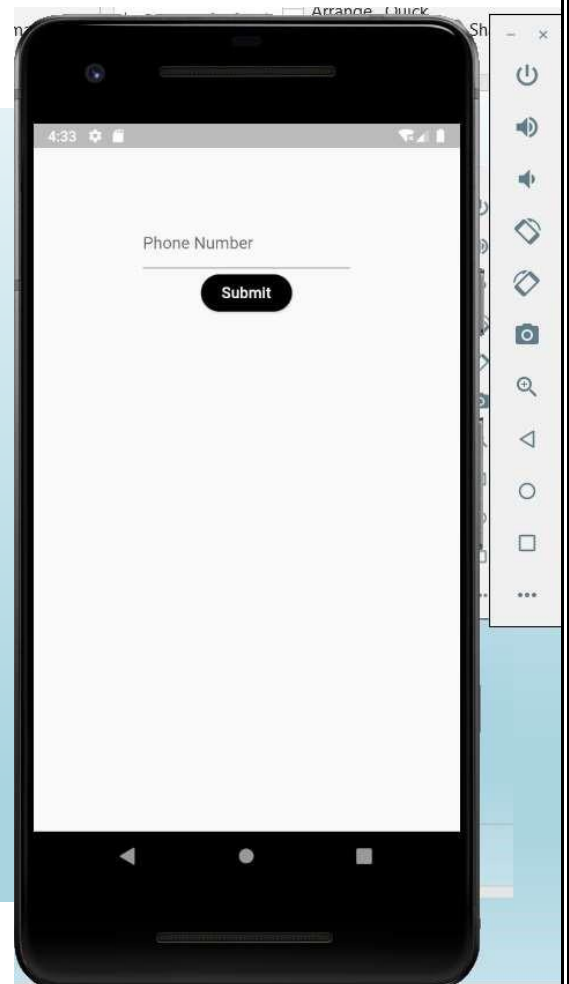
Frontend
6. After that he
can see others
reviews. By
clicking the
plus sign he
can give his
own review.



Frontend
7. After giving
his review he
can submit by
using the
submit button.

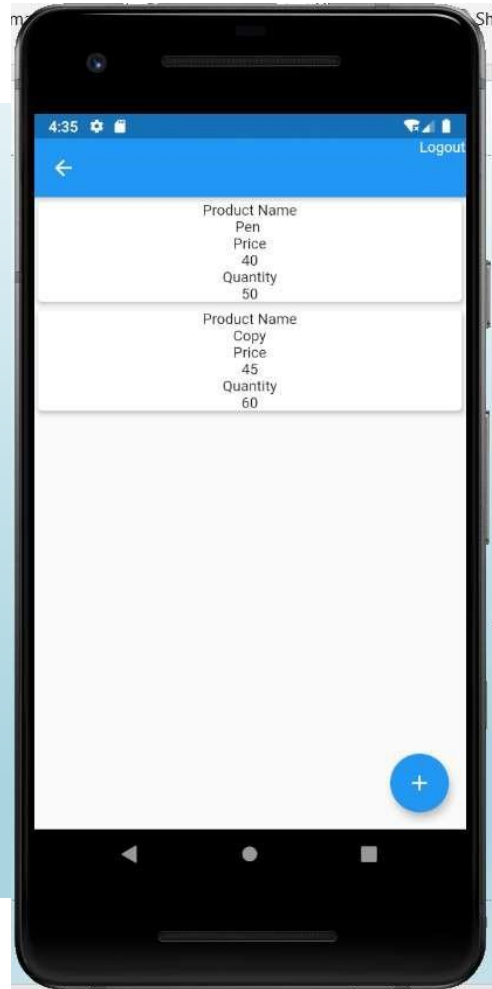


Frontend
8. If you select
the seller
option then you
will be asked
your mobile
number.



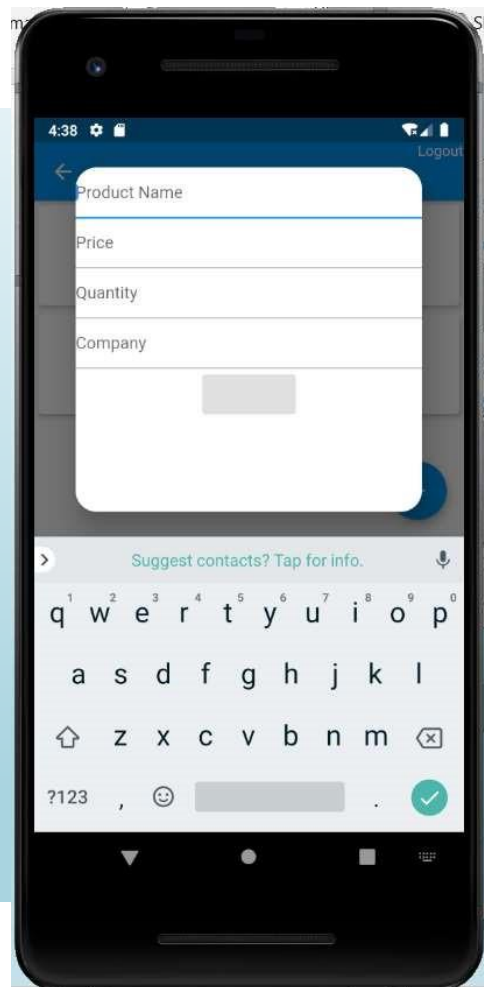
Frontend

9. If the mobile number is already registered then he will be asked password and he will be directly navigated to his shop.

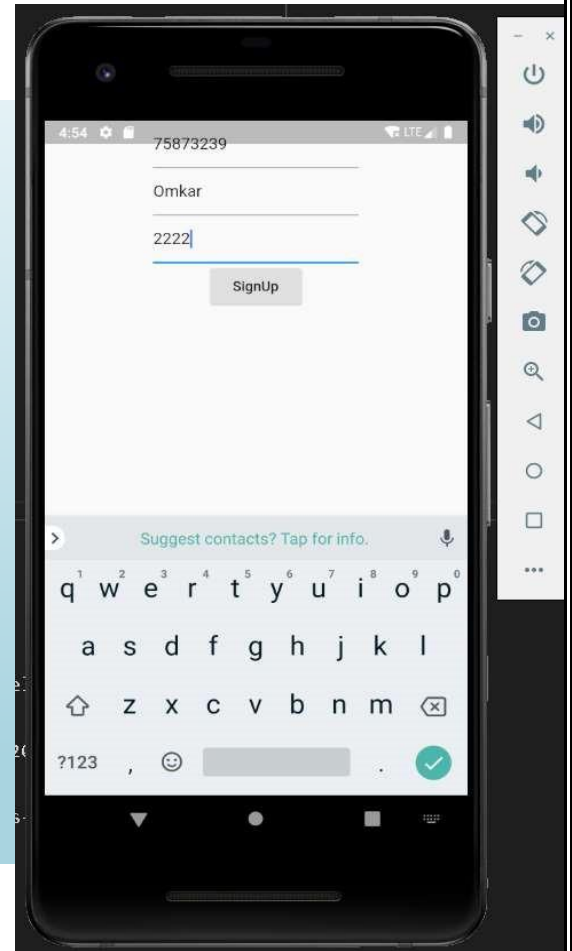


Frontend

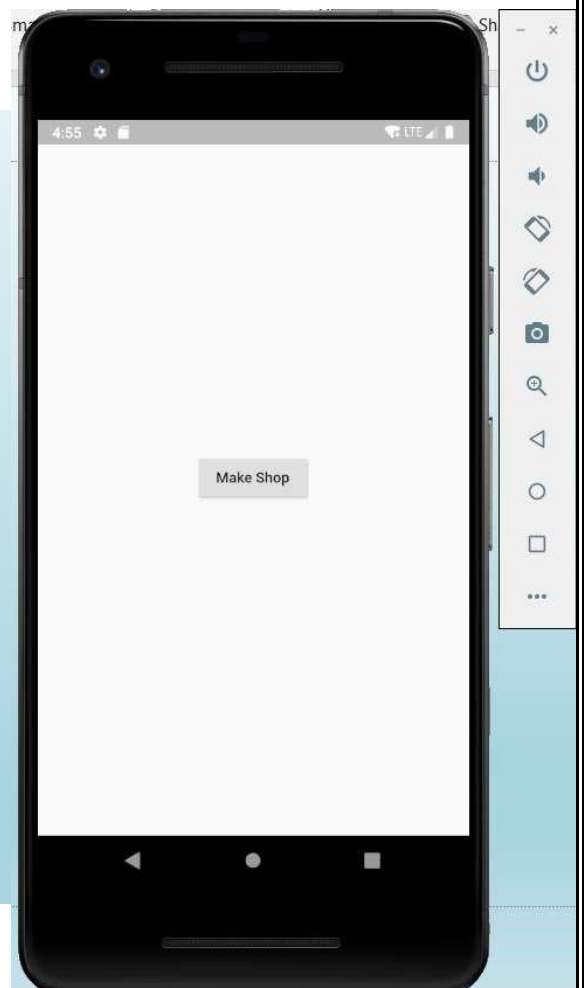
10. By plus button he can add more products to his shop. He will be asked product name, price, quantity and company.



Frontend
11. If a different user appears then he will be asked to enter a new username and password.

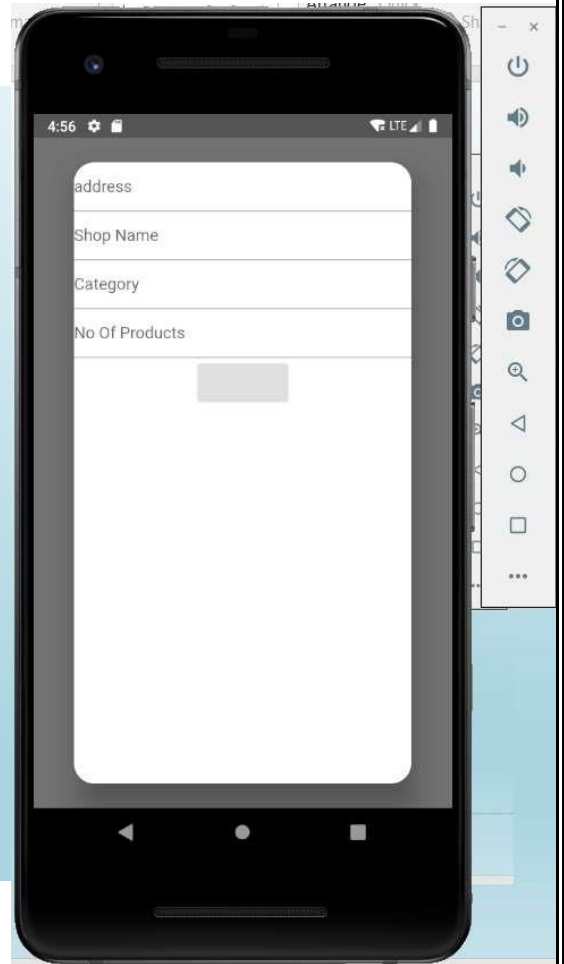


Frontend
12. After entering new username and password a button will be shown make shop.



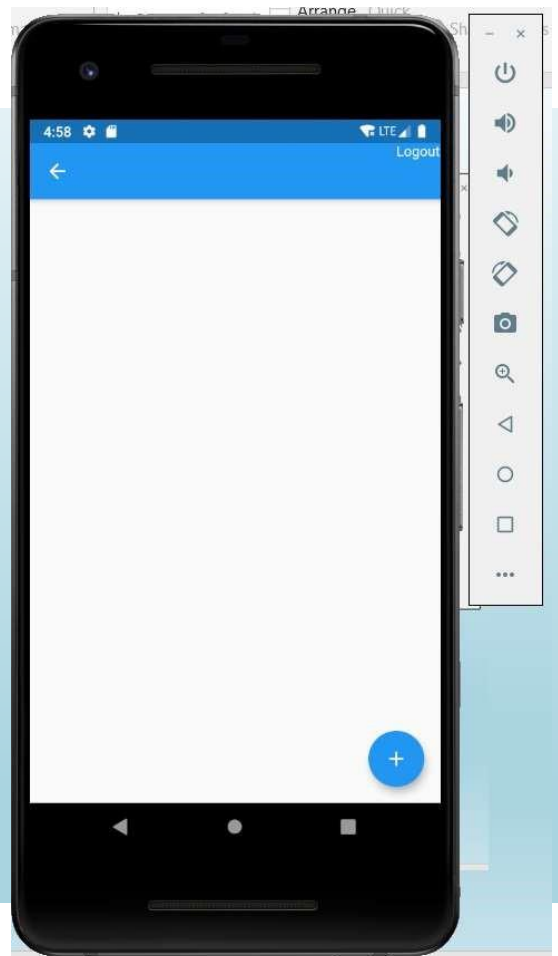
Frontend

13. After clicking make shop he will be asked to enter address, shop name, category and number of products.

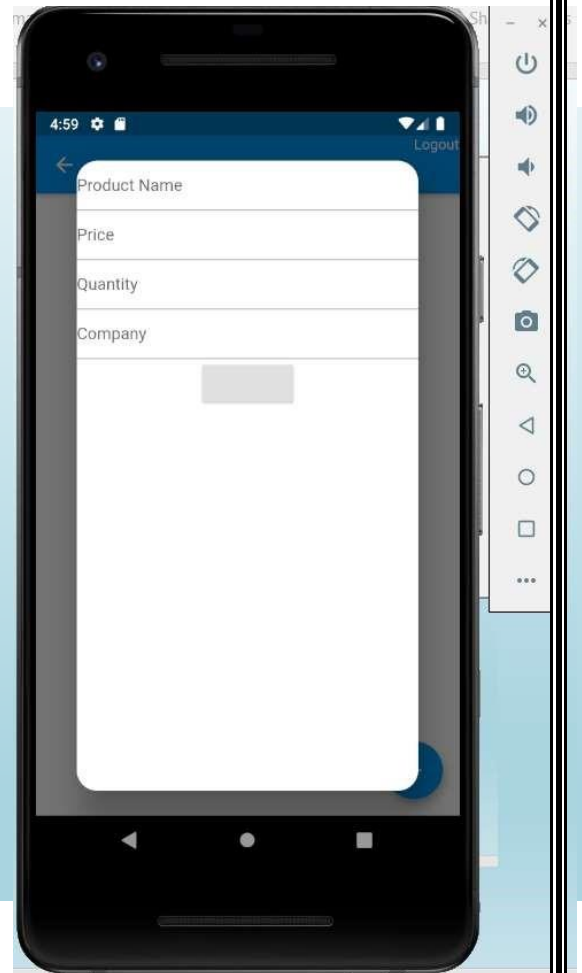


Frontend

14. He will be navigated to the screen where he can add items to his shop.



Frontend
15. After clicking plus he will be asked to enter product name, price, quantity and company.



References

<https://stackoverflow.com>

<https://www.youtube.com>

<https://www.udemy.com>

<https://www.geeksforgeeks.org>

Learning Through This Project:

This Project made me understand the concepts of DBMS and how to implement these concepts in the projects we make. I understood how important it is to Normalise the tables for the convenient flow of the project.

Also, the project has made me learn these concepts of building an application with so much of detail. It made me work hard and to keep myself strong while I faced how to fix the errors and issues that came in my way.

This project was possible only due to Dr. Saravanakumar sir. His classes and assignments made me understand the tough concepts in a very easy manner. These topics taught by sir were so greatly used in the project.

Also, I could learn languages like Flutter and Node js to make this project possible. I am happy to now create apps using the concepts of RDBMS and I hope to stay in touch with them throughout my life in the field of computer science by making projects and by solving various assignments.

*Thank
you*