SNACK SQUAD

S.no	Contents	Page No
	CHAPTER-1	1
	INTRODUCTION	1
	1.1 Overview	1
	1.2 Purpose	2
	CHAPTER-2	3
	PROBLEM DEFINITION AND DESIGN THINKING	3
	2.1 Empathy Mapping	3
	2.2 Ideation & Brainstorming Map	5
	CHAPTER- 3	6
	RESULT	6
	3.1 Screenshots	6
	CHAPTER- 4	11
	PROPOSED SYSTEM	11
	4.1 Advantages	11
	4.2 Disadvantages	11
	CHAPTER- 5	12
	APPLICATIONS	12
	CHAPTER- 6	13
	CONCLUSION	13
	CHAPTER- 7	14
	FUTURE ENHANCEMENT	14
	CHAPTER- 8	15
	APPENDIX	15
	8.1 Coding	15

CHAPTER-1

INTRODUCTION

1.1 Overview

In our current reality where time is a significant product, individuals are dependably watching out for helpful and effective arrangements. Snacking is one industry that has seen a rise in demand for such solutions. Whether you're in a bustling business day or partaking in a sluggish end of the week at home, eating is something we as a whole enjoy. But it can be hard to find the time and energy to go shopping for your favourite snacks.

Welcome to our app for snack ordering and delivery. Customers will be able to order and receive their favourite snacks without having to leave the comfort of their homes with the help of this app, which is designed to be both convenient and effective. With only a couple of taps on your Smartphone, you can peruse a broad menu of delightful tid bits and have them conveyed right to your doorstep quickly.

Customers can quickly register and begin placing orders for their favourite snacks right away thanks to our user-friendly and straightforward app. The registration procedure is straightforward and only requires basic information like your name, email address, and password. Customers can look through the main page, which include a wide range of snacks like popcorn, sandwich, burger, once they register.

Our app is made to offer a delivery service that is both dependable and effective. We have collaborated with neighbourhood conveyance organizations to guarantee that clients accept their snacks in an ideal and helpful way. Through the app, customers can monitor their order status in real time, ensuring that their snacks are on their way.

Modules:

1. Registration Page – Here the user create they accounts
2. Login Page – Here the user login using they account

3. Admin Page – Here the administrator can see amount of request and conveyance area
4. Ordering Page – Here the user can order snacks by quantity and specify the delivery location
5. Snacks Page – Here the user can scroll through wide variety of snacks and order they favourite snacks

1.2 Purpose

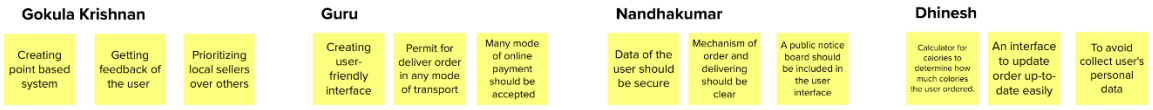
The snack ordering and delivery app was created to meet the growing demand in the snacking industry for convenient and effective solutions. Customers will be able to order and receive their preferred snacks from the convenience of their own homes with the help of the app, which is intended to facilitate a streamlined and stress-free snacking experience. Customers will be able to easily navigate the menu, customize their orders with the app's user-friendly interface. Additionally, the app aims to offer a dependable and effective delivery service, ensuring that customers receive their snacks promptly and in a way that is convenient for them. The app is anticipated to increase customer satisfaction and loyalty while also providing a valuable revenue stream for the company. By offering customers a one-of-a-kind and satisfying experience, the snacks ordering and delivery app aims to gain a competitive advantage in the snacks delivery market. In the end, the app's goal is to offer customers a profitable business opportunity while also making snacking more accessible and enjoyable for them.

CHAPTER-2

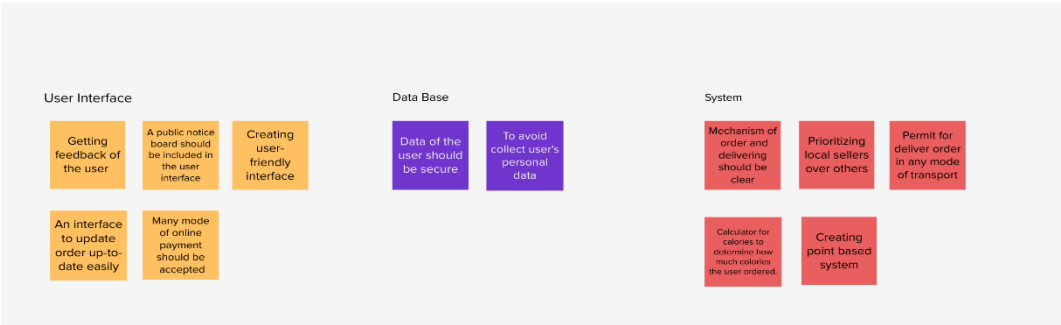
PROBLEM DEFINITION AND DESIGN THINKING

2.1 Empathy Mapping

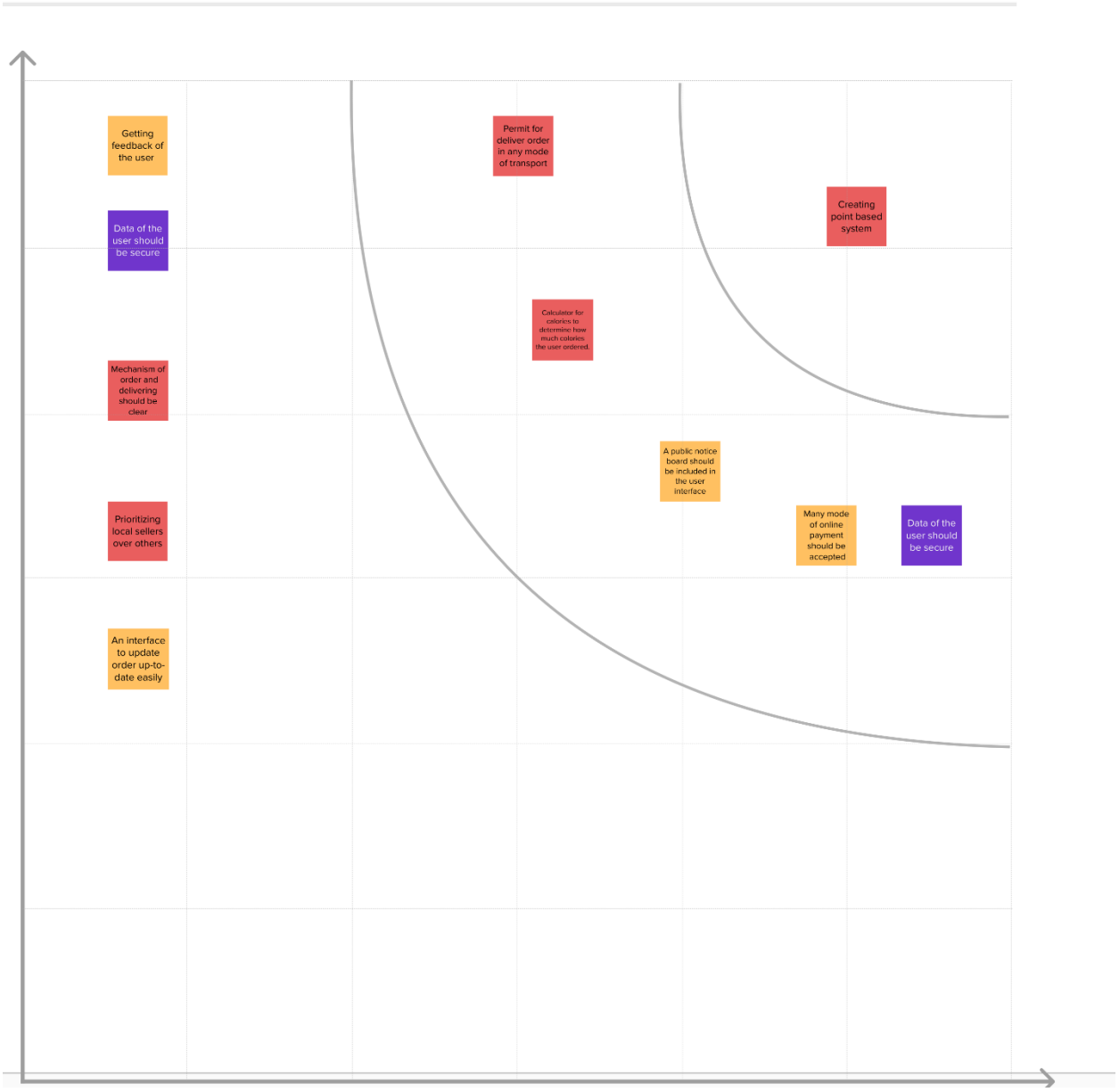
Brain Storming



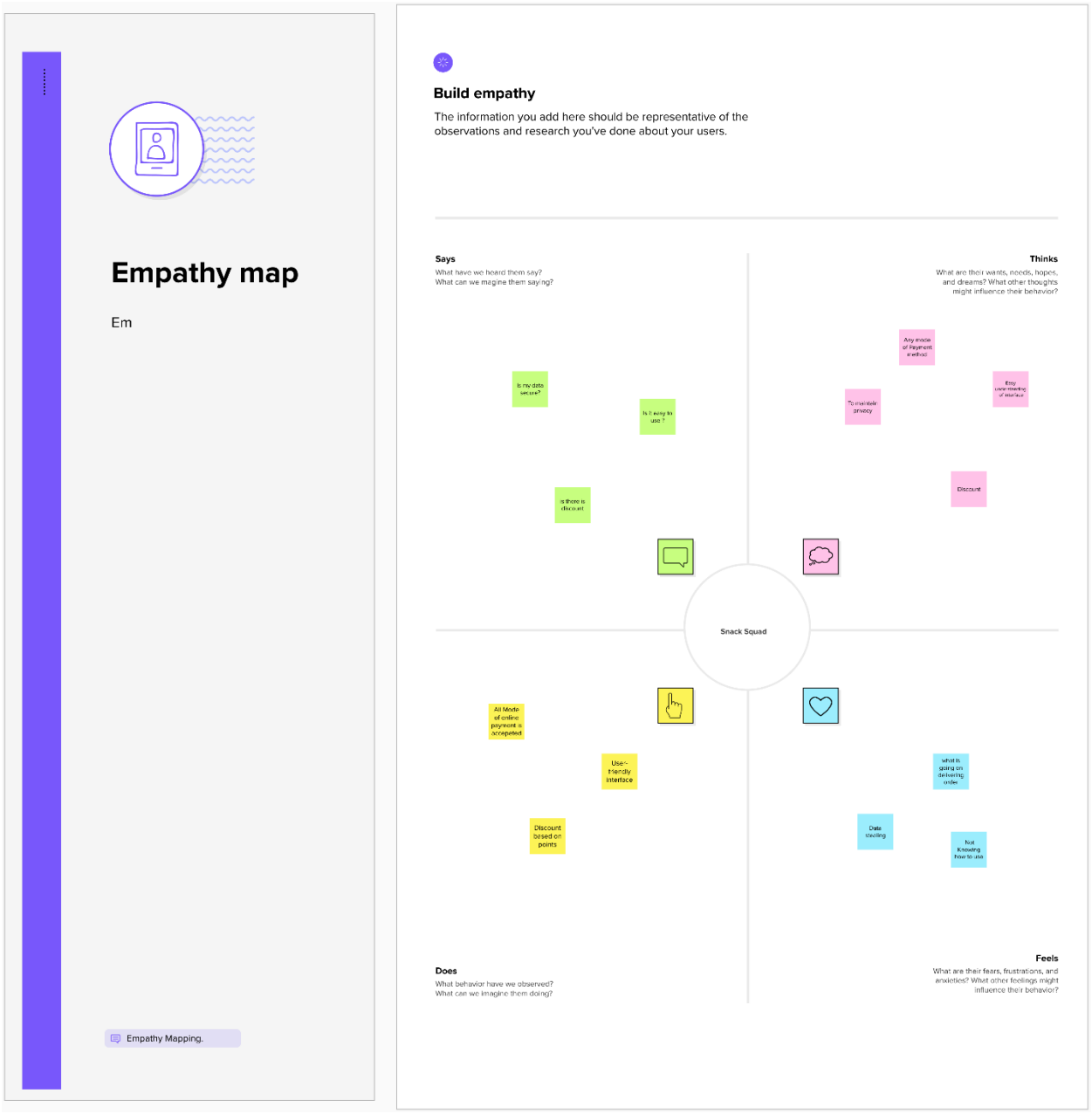
Group ideas



Prioritize


Getting feedback of the user
Data of the user should be secure
Mechanism of order and delivering should be clear
Prioritizing local sellers over others
An interface to update order up-to-date easily
Permit for deliver order in any mode of transport
Calculator for calories to determine how much calories the user ordered.
Creating point based system
A public notice board should be included in the user interface
Many mode of online payment should be accepted
Data of the user should be secure

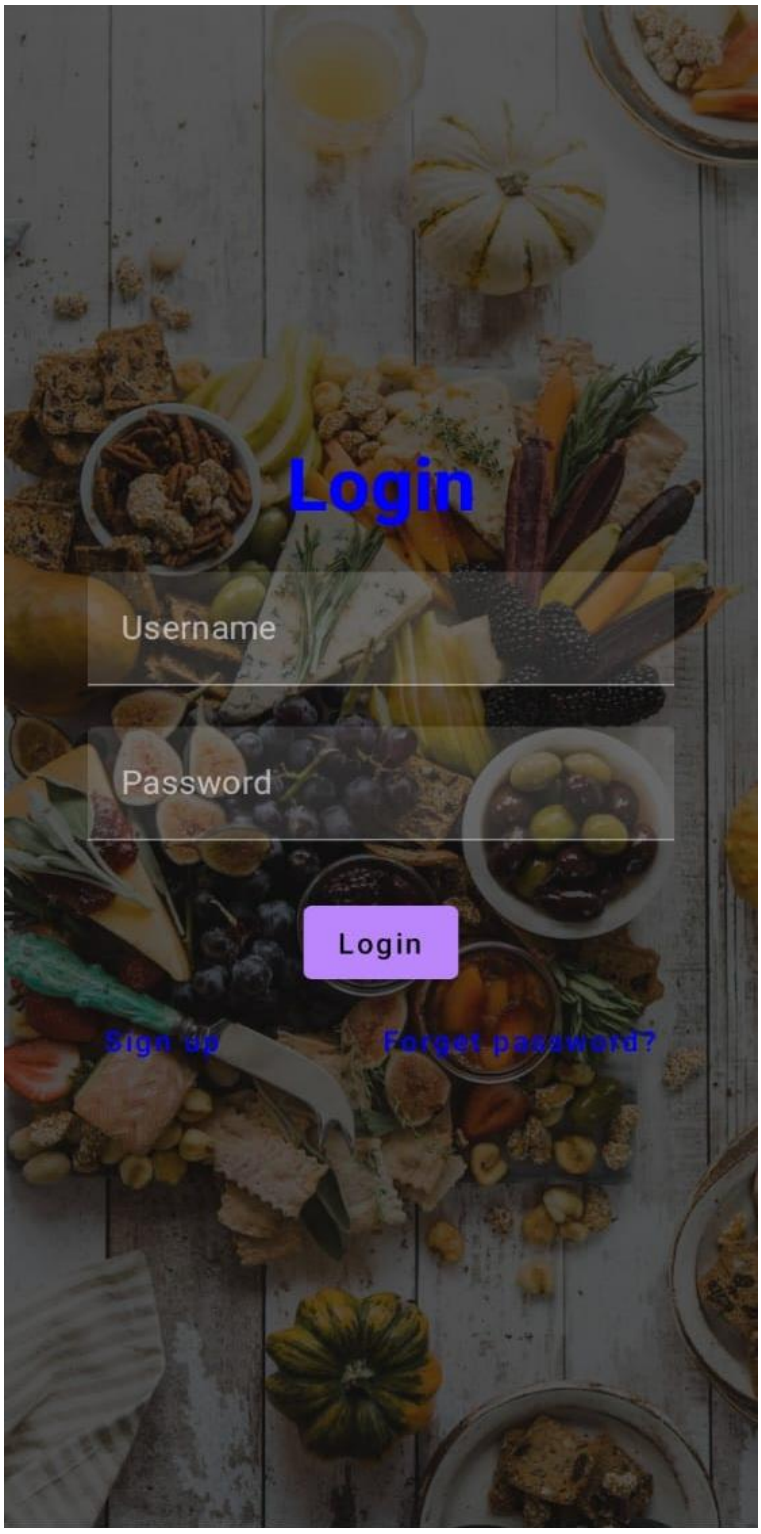
2.2 Ideation & Brainstorming Map



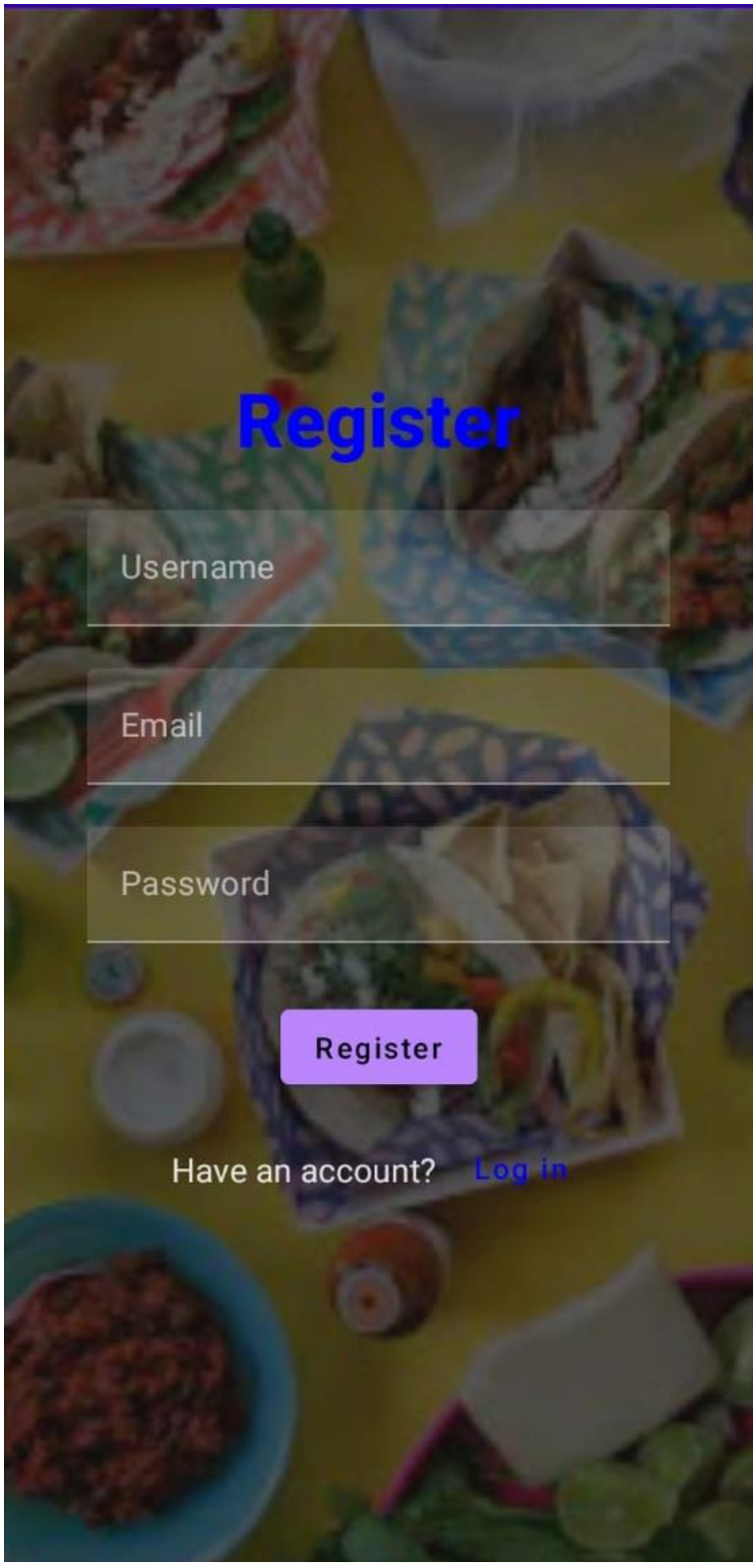
CHAPTER- 3

RESULT

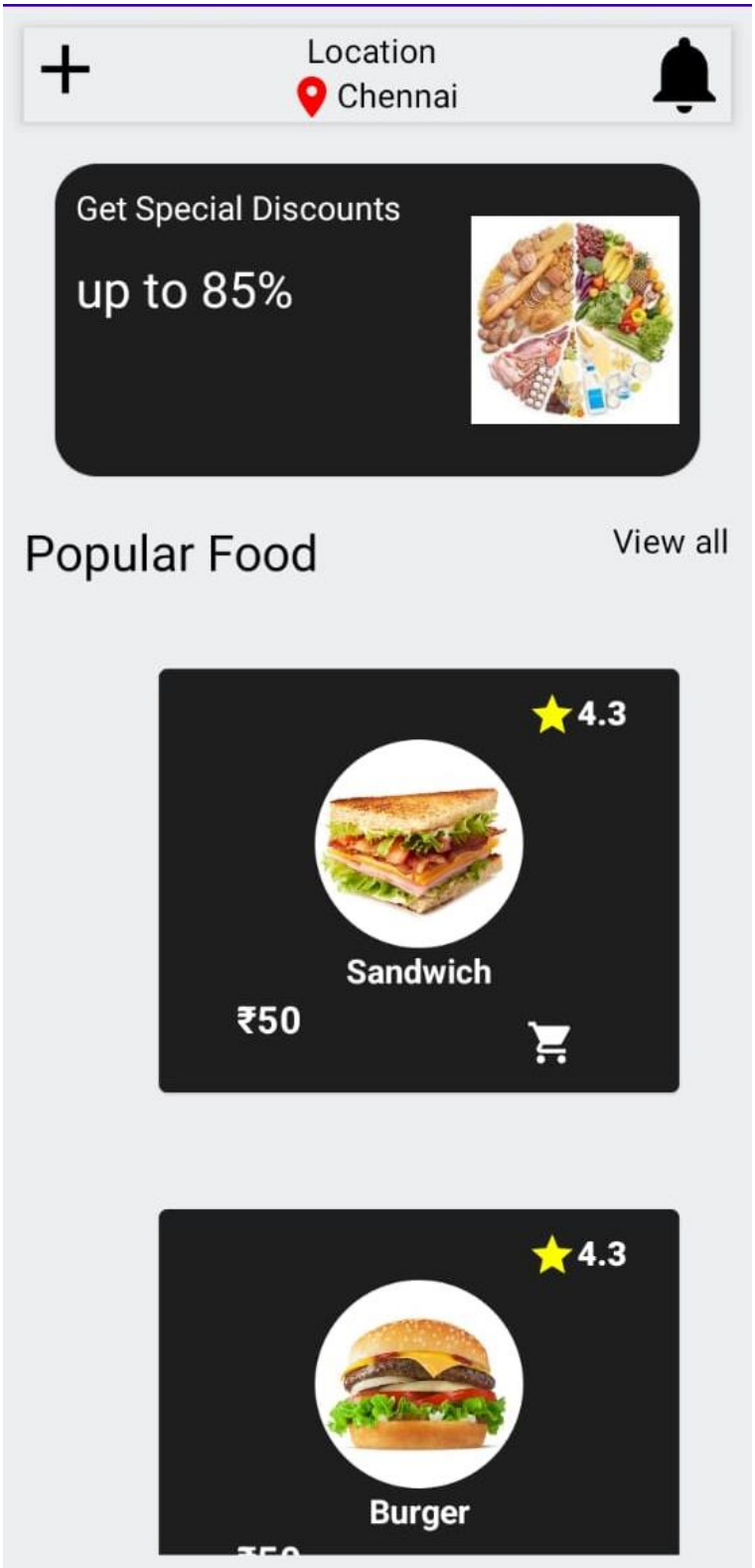
3.1 Screenshots



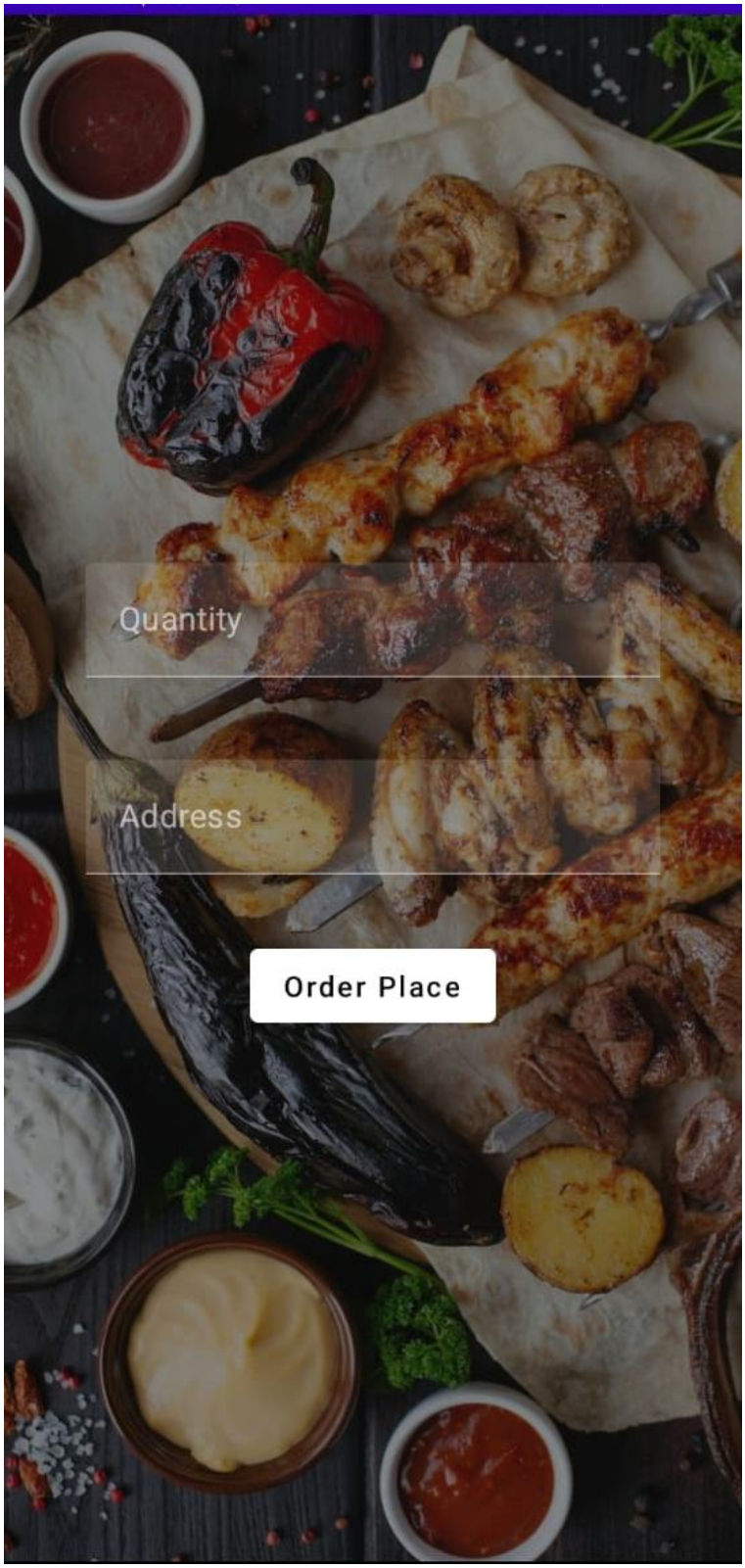
Login Page



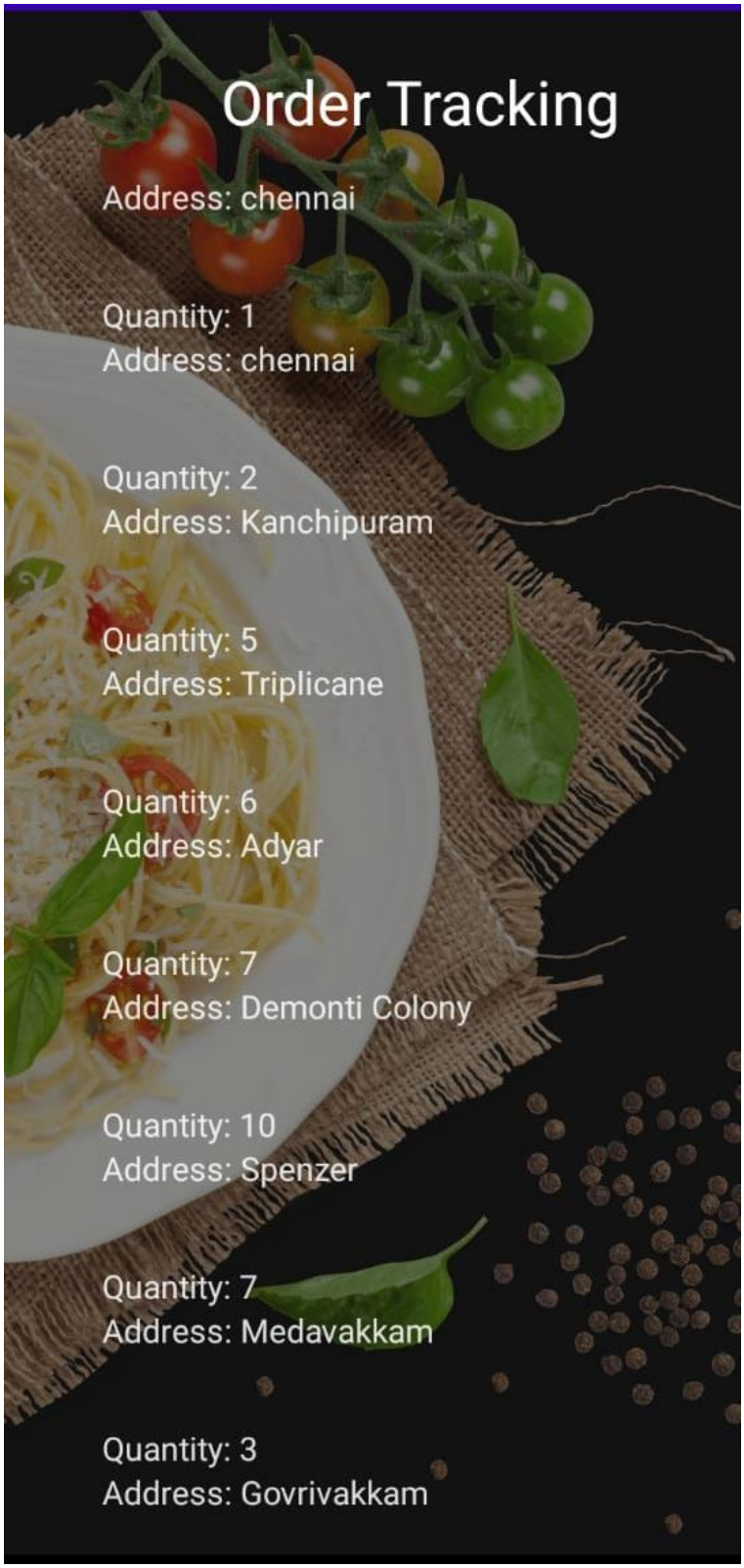
Registration Page


Location
Chennai
Get Special Discounts
up to 85%
Popular Food
View all
4.3
Sandwich
₹50
4.3
Burger

Snacks Page



Snacks Ordering Page


Order Tracking
Address: chennai
Quantity: 1
Address: chennai
Quantity: 2
Address: Kanchipuram
Quantity: 5
Address: Triplicane
Quantity: 6
Address: Adyar
Quantity: 7
Address: Demonti Colony
Quantity: 10
Address: Spenzer
Quantity: 7
Address: Medavakkam
Quantity: 3
Address: Govrivakkam

Admin Page

CHAPTER- 4

PROPOSED SYSTEM

4.1 Advantages

1. Convenience: Customers can order food from anywhere and at any time with snacks ordering, making it very convenient for them.
2. Time-saving: Clients can save a great deal of time by requesting food online as opposed to going to an actual store.
3. Customization: Customers' overall experience is enhanced when online ordering systems let them tailor their orders to their preferences.
4. Error reduction: Online ordering makes the ordering process more accurate and reduces the likelihood of miscommunication-related errors.
5. Increase in sales: Because customers are able to place orders online from any location, sales rise and revenue rise as a result.

4.2 Disadvantages

1. Problems with the technology: Orders can be delayed or canceled when online ordering systems encounter technical issues.
2. Reliance on innovation: The success of online ordering systems is dependent on the technology's dependability, which can be detrimental in the event of its failure.
3. Personal interaction is limited: There is less personal interaction with customers when they order online, which can hurt their experience.
4. Security concerns: If not properly secured, online ordering systems may result in customer data theft or misuse, posing security risks.
5. An additional cost: The initial investment in technology and staff training required to implement an online ordering system can be costly.

CHAPTER- 5

APPLICATIONS

The Snacks Ordering app can be used in a lot of different fields. The app can be used by businesses in the food and beverage industry to offer their products to customers in a way that is both convenient and effective. Customers are able to place orders and make payments directly through the app, which can be customized to feature the products of a specific business.

The Snacks Ordering app can be integrated into existing platforms in the e-commerce industry to offer snack products as an additional product category, particularly for platforms that focus on food and beverage products. The app can also be used by retail stores to sell snacks to customers, and convenience stores can offer more snack options.

Corporate workplaces can utilize the application to furnish their representatives with an assortment of nibble choices. Employees can place orders and have the snacks delivered directly to their office using the app, which can be customized to feature the products of a specific company or a variety of snack options from various brands.

Occasion coordinators can likewise use the Bites Requesting application to offer nibble items to participants. The app can be made to show products from a specific brand or a variety of snack options from different brands. This lets people order snacks and have them delivered to the event location.

Food truck owners can use the snacks ordering app to sell snacks and food to customers. The application can assist them with advancing their business, deal with their orders and installments, and speak with clients.

CHAPTER- 6

CONCLUSION

It is possible to draw the conclusion that the goal of the project was to create a user-friendly online ordering platform for popular snacks.

Android Studio and Kotlin were the technology and Language utilized during the development process. The agile development methodology was followed by the development team, allowing them to complete the project on time and within budget.

The Learning process is tough. The platform used to learn was naanmudhalvan. Working on this project improved team spirit between teammates. Improved the knowledge about kotlin language and the basic idea of how to use android studio. Slight knowing of github and how it works.

Improved the ideation, time management, typing skills and an outline of how a project works. Achieved many things and experienced pleasure of working in a project

The Snacks ordering app is a successful and useful tool for people who love snacks and want to order their favorite snacks online. The app's features and capabilities make it simple to use, and the development process adhered to industry best practices, resulting in a high-quality final product.

CHAPTER- 7

FUTURE ENHANCEMENT

1. Groups: An group chat room where the customer and they partner recommend they favourite snacks.
2. Rewards programs: A program that offers discounts or free products to repeat customers could be included in the app.
3. Recommendations: The application could utilize client request information to give customized nibble proposals in view of their past orders or interests.
4. Online entertainment incorporation: Customers could be able to share their orders or reviews with friends and followers by integrating the app with social media platforms like Facebook , Instagram etc.
5. Tracking in real time: Customers would be able to see when their snack order is being prepared, when it is on its way for delivery, and when it has been delivered if the app included real-time tracking.
6. Voice control: Customers might be able to place orders through the app with the help of voice commands if it includes voice recognition technology like Google assistant and Amazon's Alexa.
7. Chatbot powered by AI: An AI-powered chatbot could be included in the app to answer common snack-related questions and provide customer support.

CHAPTER- 8

APPENDIX

8.1 Coding

LoginActivity.kt

```
package com.example.snackordering


import android.content.Context

import android.content.Intent

import android.os.Bundle

import androidx.activity.ComponentActivity

import androidx.activity.compose.setContent

import androidx.compose.foundation.Image

import androidx.compose.foundation.layout.*

import androidx.compose.material.*

import androidx.compose.runtime.*

import androidx.compose.ui.Alignment

import androidx.compose.ui.Modifier

import androidx.compose.ui.graphics.Color

import androidx.compose.ui.layout.ContentScale

import androidx.compose.ui.res.painterResource

import androidx.compose.ui.text.font.FontFamily
```

```
import androidx.compose.ui.text.font.FontWeight
import androidx.compose.ui.unit.dp
import androidx.compose.ui.unit.sp
import androidx.core.content.ContextCompat
import com.example.snackordering.ui.theme.SnackOrderingTheme

class LoginActivity : ComponentActivity() {
    private lateinit var databaseHelper: UserDatabaseHelper
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        databaseHelper = UserDatabaseHelper(this)
        setContent {
            SnackOrderingTheme {
                // A surface container using the 'background' color from the theme
                Surface(
                    modifier = Modifier.fillMaxSize(),
                    color = MaterialTheme.colors.background
                ) {
                    LoginScreen(this, databaseHelper)
                }
            }
        }
```

```
    }
}
@Composable
fun LoginScreen(context: Context, databaseHelper: UserDatabaseHelper) {

    Image(painterResource(id = R.drawable.feed), contentDescription = "",
        alpha =0.3F,
        contentScale = ContentScale.FillHeight,

    )

    var username by remember { mutableStateOf("") }
    var password by remember { mutableStateOf("") }
    var error by remember { mutableStateOf("") }

    Column(
        modifier = Modifier.fillMaxSize(),
        horizontalAlignment = Alignment.CenterHorizontally,
        verticalArrangement = Arrangement.Center
    ) {

        Text(
```

```
    fontSize = 36.sp,
    fontWeight = FontWeight.ExtraBold,
    fontFamily = FontFamily.SansSerif,
    color = Color.Blue,
    text = "Login"
)
Spacer(modifier = Modifier.height(10.dp))

TextField(
    value = username,
    onValueChange = { username = it },
    label = { Text("Username") },
    modifier = Modifier.padding(10.dp)
        .width(280.dp)
)

TextField(
    value = password,
    onValueChange = { password = it },
    label = { Text("Password") },
    modifier = Modifier.padding(10.dp)
        .width(280.dp)
```

```
)

if (error.isNotEmpty()) {
    Text(
        text = error,
        color = MaterialTheme.colors.error,
        modifier = Modifier.padding(vertical = 16.dp)
    )
}

Button(
    onClick = {
        if (username.isNotEmpty() && password.isNotEmpty()) {
            val user = databaseHelper.getUserByUsername(username)
            if (user != null && user.password == password) {
                error = "Successfully log in"
                context.startActivity(
                    Intent(
                        context,
                        MainPage::class.java
                    )
                )
```

```
            //onLoginSuccess()
        }
            if (user != null && user.password == "admin") {
                error = "Successfully log in"
                context.startActivity(
                    Intent(
                        context,
                        AdminActivity::class.java
                    )
                )
            }
            else {
                error =  "Invalid username or password"
            }

        } else {
            error = "Please fill all fields"
        }
    },
    modifier = Modifier.padding(top = 16.dp)
) {
    Text(text = "Login")
```

```
        }
        Row {
            TextButton(onClick = {context.startActivity(
                Intent(
                    context,
                    MainActivity::class.java
                )
            )}
            )
            { Text(color = Color.Blue,text = "Sign up") }
            TextButton(onClick = {
            })

            {
                Spacer(modifier = Modifier.width(60.dp))
                Text(color = Color.Blue,text = "Forget password?")
            }
        }
    }
}
private fun startMainPage(context: Context) {
    val intent = Intent(context, MainPage::class.java)
```

```
    ContextCompat.startActivity(context, intent, null)
}
```

RegisterActivity.kt

```
package com.example.snackordering


import android.content.Context
import android.content.Intent
import android.os.Bundle
import androidx.activity.ComponentActivity
import androidx.activity.compose.setContent
import androidx.compose.foundation.Image
import androidx.compose.foundation.layout.*
import androidx.compose.material.*
import androidx.compose.runtime.*
import androidx.compose.ui.Alignment
import androidx.compose.ui.Modifier
import androidx.compose.ui.graphics.Color
import androidx.compose.ui.layout.ContentScale
import androidx.compose.ui.res.painterResource
import androidx.compose.ui.text.font.FontFamily
import androidx.compose.ui.text.font.FontWeight
import androidx.compose.ui.unit.dp
```

```
import androidx.compose.ui.unit.sp
import androidx.core.content.ContextCompat
import com.example.snackordering.ui.theme.SnackOrderingTheme


class MainActivity : ComponentActivity() {
    private lateinit var databaseHelper: UserDatabaseHelper
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        databaseHelper = UserDatabaseHelper(this)
        setContent {
            SnackOrderingTheme {
                // A surface container using the 'background' color from the theme
                Surface(
                    modifier = Modifier.fillMaxSize(),
                    color = MaterialTheme.colors.background
                ) {


                    RegistrationScreen(this,databaseHelper)
                }
            }
        }
    }
```

```
}



@Composable
fun RegistrationScreen(context: Context, databaseHelper: UserDatabaseHelper) {

    Image(
        painterResource(id = R.drawable.trail), contentDescription = "",
        alpha =0.3F,
        contentScale = ContentScale.FillHeight,


        )


    var username by remember { mutableStateOf("") }
    var password by remember { mutableStateOf("") }
    var email by remember { mutableStateOf("") }
    var error by remember { mutableStateOf("") }


    Column(
        modifier = Modifier.fillMaxSize(),
        horizontalAlignment = Alignment.CenterHorizontally,
        verticalArrangement = Arrangement.Center
```

```
) {

    Text(
        fontSize = 36.sp,
        fontWeight = FontWeight.ExtraBold,
        fontFamily = FontFamily.SansSerif,
        color = Color.Blue,
        text = "Register"
    )

    Spacer(modifier = Modifier.height(10.dp))
    TextField(
        value = username,
        onValueChange = { username = it },
        label = { Text("Username") },
        modifier = Modifier
            .padding(10.dp)
            .width(280.dp)

    )

    TextField(
```

```
    value = email,
    onValueChange = { email = it },
    label = { Text("Email") },
    modifier = Modifier
        .padding(10.dp)
        .width(280.dp)
)

TextField(
    value = password,
    onValueChange = { password = it },
    label = { Text("Password") },
    modifier = Modifier
        .padding(10.dp)
        .width(280.dp)
)



if (error.isNotEmpty()) {
    Text(
        text = error,
        color = MaterialTheme.colors.error,
```

```
        modifier = Modifier.padding(vertical = 16.dp)
    )
}

Button(
    onClick = {
        if (username.isNotEmpty() && password.isNotEmpty() && email.isNotEmpty()) {
            val user = User(
                id = null,
                firstName = username,
                lastName = null,
                email = email,
                password = password
            )
            databaseHelper.insertUser(user)
            error = "User registered successfully"
            // Start LoginActivity using the current context
            context.startActivity(
                Intent(
                    context,
                    LoginActivity::class.java
                )
```

```
            )


        } else {
            error = "Please fill all fields"
        }
    },
    modifier = Modifier.padding(top = 16.dp)
) {
    Text(text = "Register")
}
Spacer(modifier = Modifier.width(10.dp))
Spacer(modifier = Modifier.height(10.dp))

Row() {
    Text(
        modifier = Modifier.padding(top = 14.dp), text = "Have an account?"
    )
    TextButton(onClick = {
        context.startActivity(
            Intent(
                context,
                LoginActivity::class.java
```

```
                )
            )
        })

        {
            Spacer(modifier = Modifier.width(10.dp))
            Text(color = Color.Blue,text = "Log in")
        }
    }
  }
}
private fun startLoginActivity(context: Context) {
  val intent = Intent(context, LoginActivity::class.java)
  ContextCompat.startActivity(context, intent, null)
}
```

MainPage.kt

```
package com.example.snackordering

import android.annotation.SuppressLint
import android.content.Context
import android.os.Bundle
```

```
import android.widget.Toast
import androidx.activity.ComponentActivity
import androidx.activity.compose.setContent
import androidx.annotation.DrawableRes
import androidx.annotation.StringRes
import androidx.compose.foundation.Image
import androidx.compose.foundation.background
import androidx.compose.foundation.layout.*
import androidx.compose.foundation.shape.CircleShape
import androidx.compose.foundation.shape.RoundedCornerShape
import androidx.compose.material.*
import androidx.compose.material.icons.Icons
import androidx.compose.material.icons.filled.*
import androidx.compose.runtime.Composable
import androidx.compose.ui.Alignment
import androidx.compose.ui.Modifier
import androidx.compose.ui.draw.clip
import androidx.compose.ui.graphics.Color
import androidx.compose.foundation.lazy.LazyColumn
import androidx.compose.foundation.lazy.items
import androidx.compose.material.Text
import androidx.compose.ui.unit.dp
```

```
import androidx.compose.ui.graphics.RectangleShape
import androidx.compose.ui.layout.ContentScale
import androidx.compose.ui.platform.LocalContext
import androidx.compose.ui.res.painterResource
import androidx.compose.ui.res.stringResource
import androidx.compose.ui.text.font.FontWeight
import androidx.compose.ui.unit.sp
import androidx.core.content.ContextCompat.startActivity
import com.example.snackordering.ui.theme.SnackOrderingTheme

import android.content.Intent as Intent1



class MainPage : ComponentActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContent {
            SnackOrderingTheme {
                // A surface container using the 'background' color from the theme
                Surface(
                    modifier = Modifier.fillMaxSize(),
                    color = Color.Green
```

```
            ) {
                FinalView(this)
                val context = LocalContext.current
                //PopularFoodColumn(context)
            }
        }
    }
}


@Composable
fun TopPart() {
    val mContext = LocalContext.current
    Row(
        modifier = Modifier
            .fillMaxWidth()
            .background(Color(0xffeceef0)), Arrangement.SpaceBetween
    ) {
        Icon(
            imageVector = Icons.Default.Add, contentDescription = "Menu Icon",
            Modifier
```

```
        .clip(CircleShape)
        .size(40.dp),
    tint = Color.Black,
)
Column(horizontalAlignment = Alignment.CenterHorizontally) {
    Text(text = "Location", style = MaterialTheme.typography.subtitle1, color = Color.Black)
    Row {
        Icon(
            imageVector = Icons.Default.LocationOn,
            contentDescription = "Location",
            tint = Color.Red,
        )
        Text(text = "Chennai" , color = Color.Black)
    }

}
Icon(
    imageVector = Icons.Default.Notifications, contentDescription = "Notification Icon",

    Modifier
        .size(45.dp),
```

```
            tint = Color.Black,
        )
    }
}

@Composable
fun CardPart() {
    Card(modifier = Modifier.size(width = 310.dp, height = 150.dp), RoundedCornerShape(20.dp))
{
        Row(modifier = Modifier.padding(10.dp), Arrangement.SpaceBetween) {
            Column(verticalArrangement = Arrangement.spacedBy(12.dp)) {
                Text(text = "Get Special Discounts")
                Text(text = "up to 85%", style = MaterialTheme.typography.h5)
            }
            Image(
                painter = painterResource(id = R.drawable.food_tip_im),
                contentDescription = "Food Image", Modifier.size(width = 100.dp, height = 200.dp)
            )
        }
    }
}
```

```
@Composable
fun PopularFood(
    @DrawableRes drawable: Int,
    @StringRes text1: Int,
    context: Context
) {
    Card(
        modifier = Modifier
            .padding(top=20.dp, bottom = 20.dp, start = 65.dp)
            .width(250.dp)

    ) {
        Column(
            verticalArrangement = Arrangement.Top,
            horizontalAlignment = Alignment.CenterHorizontally
        ) {
            Spacer(modifier = Modifier.padding(vertical = 5.dp))
            Row(
                modifier = Modifier
                    .fillMaxWidth(0.8f), Arrangement.End
            ) {
```

```
    Icon(
        imageVector = Icons.Default.Star,
        contentDescription = "Star Icon",
        tint = Color.Yellow
    )
    Text(text = "4.3", fontWeight = FontWeight.Black)
}
Image(
    painter = painterResource(id = drawable),
    contentDescription = "Food Image",
    contentScale = ContentScale.Crop,
    modifier = Modifier
        .size(100.dp)
        .clip(CircleShape)
)
Text(text = stringResource(id = text1), fontWeight = FontWeight.Bold)
Row(modifier = Modifier.fillMaxWidth(0.7f), Arrangement.SpaceBetween) {
    /*TODO Implement Prices for each card*/
    Text(
        text = "₹50",
        style = MaterialTheme.typography.h6,
        fontWeight = FontWeight.Bold,
```

```
            fontSize = 18.sp
        )

        IconButton(onClick = {

            //var no=FoodList.lastIndex;
            //Toast.
            val intent = Intent1(context, TargetActivity::class.java)
            context.startActivity(intent)

        }) {
            Icon(
                imageVector = Icons.Default.ShoppingCart,
                contentDescription = "shopping cart",
            )
        }
      }
    }
  }
```

```
private val FoodList = listOf(
    R.drawable.sandwish to R.string.sandwich,
    R.drawable.burger to R.string.burgers,
    R.drawable.pack to R.string.pack,
    R.drawable.pasta to R.string.pasta,
    R.drawable.tequila to R.string.tequila,
    R.drawable.wine to R.string.wine,
    R.drawable.salad to R.string.salad,
    R.drawable.pop to R.string.popcorn
).map { DrawableStringPair(it.first, it.second) }

private data class DrawableStringPair(
    @DrawableRes val drawable: Int,
    @StringRes val text1: Int
)


@Composable
fun App(context: Context) {
```

```
Column(
    modifier = Modifier
        .fillMaxSize()
        .background(Color(0xffeceef0))
        .padding(10.dp),
    verticalArrangement = Arrangement.Top,
    horizontalAlignment = Alignment.CenterHorizontally
) {
    Surface(modifier = Modifier, elevation = 5.dp) {
        TopPart()
    }
    Spacer(modifier = Modifier.padding(10.dp))
    CardPart()


    Spacer(modifier = Modifier.padding(10.dp))
    Row(modifier = Modifier.fillMaxWidth(), Arrangement.SpaceBetween) {
        Text(text = "Popular Food", style = MaterialTheme.typography.h5, color = Color.Black)
        Text(text = "View all", style = MaterialTheme.typography.subtitle1,  color = Color.Black)
    }
    Spacer(modifier = Modifier.padding(10.dp))
    PopularFoodColumn(context) // <- call the function with parentheses
}
```

```
}




@Composable
fun PopularFoodColumn(context: Context) {

    LazyColumn(
        modifier = Modifier.fillMaxSize(),

        content = {
            items(FoodList) { item ->
                PopularFood(context = context,drawable = item.drawable, text1 = item.text1)
                abstract class Context
            }
        },
        verticalArrangement = Arrangement.spacedBy(16.dp))
}



@SuppressLint("UnusedMaterialScaffoldPaddingParameter")
```

```
@Composable
fun FinalView(mainPage: MainPage) {
    SnackOrderingTheme {
        Scaffold() {
            val context = LocalContext.current
            App(context)
        }
    }
}
```

TargetActivity.kt

```
package com.example.snackordering

import android.content.Context
import android.content.Intent
import android.os.Bundle
import android.util.Log
import android.widget.Toast
import androidx.activity.ComponentActivity
import androidx.activity.compose.setContent
import androidx.compose.foundation.Image
import androidx.compose.foundation.background
```

```
import androidx.compose.foundation.layout.*
import androidx.compose.foundation.text.KeyboardActions
import androidx.compose.foundation.text.KeyboardOptions
import androidx.compose.material.*
import androidx.compose.runtime.*
import androidx.compose.ui.Alignment
import androidx.compose.ui.Modifier
import androidx.compose.ui.graphics.Color
import androidx.compose.ui.layout.ContentScale
import androidx.compose.ui.platform.LocalContext
import androidx.compose.ui.platform.textInputServiceFactory
import androidx.compose.ui.res.painterResource
import androidx.compose.ui.text.input.KeyboardType
import androidx.compose.ui.tooling.preview.Preview
import androidx.compose.ui.unit.dp
import androidx.core.content.ContextCompat
import com.example.snackordering.ui.theme.SnackOrderingTheme

class TargetActivity : ComponentActivity() {
    private lateinit var orderDatabaseHelper: OrderDatabaseHelper
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
```

```
        orderDatabaseHelper = OrderDatabaseHelper(this)
        setContent {
            SnackOrderingTheme {
                // A surface container using the 'background' color from the theme
                Surface(
                    modifier = Modifier
                        .fillMaxSize()
                        .background(Color.White)

                ) {
                    Order(this, orderDatabaseHelper)
                    val orders = orderDatabaseHelper.getAllOrders()
                   Log.d("swathi", orders.toString())

                }
            }
        }
    }
}

@Composable
fun Order(context: Context, orderDatabaseHelper: OrderDatabaseHelper){
```

```
Image(painterResource(id = R.drawable.food), contentDescription = "",
    alpha =0.5F,
contentScale = ContentScale.FillHeight)
Column(
    horizontalAlignment = Alignment.CenterHorizontally,
    verticalArrangement = Arrangement.Center) {

    val mContext = LocalContext.current
    var quantity by remember { mutableStateOf("") }
    var address by remember { mutableStateOf("") }
    var error by remember { mutableStateOf("") }



    TextField(value = quantity, onValueChange = {quantity=it},
        label = { Text("Quantity") },
       keyboardOptions = KeyboardOptions(keyboardType = KeyboardType.Number),
        modifier = Modifier
            .padding(10.dp)
            .width(280.dp))

    Spacer(modifier = Modifier.padding(10.dp))
```

```
TextField(value = address, onValueChange = {address=it},
    label = { Text("Address") },
    modifier = Modifier
        .padding(10.dp)
        .width(280.dp))

Spacer(modifier = Modifier.padding(10.dp))



if (error.isNotEmpty()) {
    Text(
        text = error,
        color = MaterialTheme.colors.error,
        modifier = Modifier.padding(vertical = 16.dp)
    )
}



Button(onClick = {
    if( quantity.isNotEmpty() and address.isNotEmpty()){
        val order = Order(
            id = null,
```

```
                quantity = quantity,
                address = address
            )
            orderDatabaseHelper.insertOrder(order)
        Toast.makeText(mContext, "Order Placed Successfully", Toast.LENGTH_SHORT).show()}
    },
        colors = ButtonDefaults.buttonColors(backgroundColor = Color.White))
    {
        Text(text = "Order Place", color = Color.Black)
    }


  }
}
private fun startMainPage(context: Context) {
  val intent = Intent(context, LoginActivity::class.java)
  ContextCompat.startActivity(context, intent, null)
}
```

User class

```
package com.example.snackordering

import androidx.room.ColumnInfo
```

```
import androidx.room.Entity
import androidx.room.PrimaryKey

@Entity(tableName = "user_table")
data class User(
    @PrimaryKey(autoGenerate = true) val id: Int?,
    @ColumnInfo(name = "first_name") val firstName: String?,
    @ColumnInfo(name = "last_name") val lastName: String?,
    @ColumnInfo(name = "email") val email: String?,
    @ColumnInfo(name = "password") val password: String?,

    )
```

UserDao interface

```
package com.example.snackordering

import androidx.room.*

@Dao
interface UserDao {

    @Query("SELECT * FROM user_table WHERE email = :email")
    suspend fun getUserByEmail(email: String): User?
```

```
    @Insert(onConflict = OnConflictStrategy.REPLACE)
    suspend fun insertUser(user: User)

    @Update
    suspend fun updateUser(user: User)

    @Delete
    suspend fun deleteUser(user: User)
}
```

UserDatabase

```
package com.example.snackordering

import android.content.Context
import androidx.room.Database
import androidx.room.Room
import androidx.room.RoomDatabase

@Database(entities = [User::class], version = 1)
abstract class UserDatabase : RoomDatabase() {
```

```
    abstract fun userDao(): UserDao

    companion object {

        @Volatile
        private var instance: UserDatabase? = null

        fun getDatabase(context: Context): UserDatabase {
            return instance ?: synchronized(this) {
                val newInstance = Room.databaseBuilder(
                    context.applicationContext,
                    UserDatabase::class.java,
                    "user_database"
                ).build()
                instance = newInstance
                newInstance
            }
        }
    }
}
```

UserDatabaseHelper class

```
package com.example.snackordering
```

```
import android.annotation.SuppressLint
import android.content.ContentValues
import android.content.Context
import android.database.Cursor
import android.database.sqlite.SQLiteDatabase
import android.database.sqlite.SQLiteOpenHelper

class UserDatabaseHelper(context: Context) :
    SQLiteOpenHelper(context, DATABASE_NAME, null, DATABASE_VERSION) {

    companion object {
        private const val DATABASE_VERSION = 1
        private const val DATABASE_NAME = "UserDatabase.db"

        private const val TABLE_NAME = "user_table"
        private const val COLUMN_ID = "id"
        private const val COLUMN_FIRST_NAME = "first_name"
        private const val COLUMN_LAST_NAME = "last_name"
        private const val COLUMN_EMAIL = "email"
        private const val COLUMN_PASSWORD = "password"
    }
```

```
override fun onCreate(db: SQLiteDatabase?) {
    val createTable = "CREATE TABLE $TABLE_NAME (" +
            "$COLUMN_ID INTEGER PRIMARY KEY AUTOINCREMENT, " +
            "$COLUMN_FIRST_NAME TEXT, " +
            "$COLUMN_LAST_NAME TEXT, " +
            "$COLUMN_EMAIL TEXT, " +
            "$COLUMN_PASSWORD TEXT" +
            ")"

    db?.execSQL(createTable)
}

override fun onUpgrade(db: SQLiteDatabase?, oldVersion: Int, newVersion: Int) {
    db?.execSQL("DROP TABLE IF EXISTS $TABLE_NAME")
    onCreate(db)
}

fun insertUser(user: User) {
    val db = writableDatabase
    val values = ContentValues()
    values.put(COLUMN_FIRST_NAME, user.firstName)
```

```
        values.put(COLUMN_LAST_NAME, user.lastName)
        values.put(COLUMN_EMAIL, user.email)
        values.put(COLUMN_PASSWORD, user.password)
        db.insert(TABLE_NAME, null, values)
        db.close()
    }

    @SuppressLint("Range")
    fun getUserByUsername(username: String): User? {
        val db = readableDatabase
        val cursor: Cursor = db.rawQuery("SELECT * FROM $TABLE_NAME WHERE $COLUMN_FIRST_NAME = ?", arrayOf(username))
        var user: User? = null
        if (cursor.moveToFirst()) {
            user = User(
                id = cursor.getInt(cursor.getColumnIndex(COLUMN_ID)),
                firstName = cursor.getString(cursor.getColumnIndex(COLUMN_FIRST_NAME)),
                lastName = cursor.getString(cursor.getColumnIndex(COLUMN_LAST_NAME)),
                email = cursor.getString(cursor.getColumnIndex(COLUMN_EMAIL)),
                password = cursor.getString(cursor.getColumnIndex(COLUMN_PASSWORD)),
            )
        }
```

```
        cursor.close()
        db.close()
        return user
    }
    @SuppressLint("Range")
    fun getUserById(id: Int): User? {
        val db = readableDatabase
        val cursor: Cursor = db.rawQuery("SELECT * FROM $TABLE_NAME WHERE $COLUMN_ID =
?", arrayOf(id.toString()))
        var user: User? = null
        if (cursor.moveToFirst()) {
            user = User(
                id = cursor.getInt(cursor.getColumnIndex(COLUMN_ID)),
                firstName = cursor.getString(cursor.getColumnIndex(COLUMN_FIRST_NAME)),
                lastName = cursor.getString(cursor.getColumnIndex(COLUMN_LAST_NAME)),
                email = cursor.getString(cursor.getColumnIndex(COLUMN_EMAIL)),
                password = cursor.getString(cursor.getColumnIndex(COLUMN_PASSWORD)),
            )
        }
        cursor.close()
        db.close()
        return user
```

```
}

@SuppressLint("Range")
fun getAllUsers(): List<User> {
    val users = mutableListOf<User>()
    val db = readableDatabase
    val cursor: Cursor = db.rawQuery("SELECT * FROM $TABLE_NAME", null)
    if (cursor.moveToFirst()) {
        do {
            val user = User(
                id = cursor.getInt(cursor.getColumnIndex(COLUMN_ID)),
                firstName = cursor.getString(cursor.getColumnIndex(COLUMN_FIRST_NAME)),
                lastName = cursor.getString(cursor.getColumnIndex(COLUMN_LAST_NAME)),
                email = cursor.getString(cursor.getColumnIndex(COLUMN_EMAIL)),
                password = cursor.getString(cursor.getColumnIndex(COLUMN_PASSWORD)),
            )
            users.add(user)
        } while (cursor.moveToNext())
    }
    cursor.close()
    db.close()
    return users
```

```
    }
    }
```

Build gradle

```
buildscript {
    ext {
        compose_ui_version = '1.2.0'
    }
}// Top-level build file where you can add configuration options common to all sub-projects/modules.
plugins {
    id 'com.android.application' version '7.4.2' apply false
    id 'com.android.library' version '7.4.2' apply false
    id 'org.jetbrains.kotlin.android' version '1.7.0' apply false
}
plugins {
    id 'com.android.application'
    id 'org.jetbrains.kotlin.android'
}


android {
    namespace 'com.example.snackordering'
    compileSdk 33
```

```
    defaultConfig {
        applicationId "com.example.snackordering"
        minSdk 24
        targetSdk 33
        versionCode 1
        versionName "1.0"

        testInstrumentationRunner "androidx.test.runner.AndroidJUnitRunner"
        vectorDrawables {
            useSupportLibrary true
        }
    }

    buildTypes {
        release {
            minifyEnabled false
            proguardFiles getDefaultProguardFile('proguard-android-optimize.txt'), 'proguard-
rules.pro'
        }
    }
    compileOptions {
```

```
        sourceCompatibility JavaVersion.VERSION_1_8
        targetCompatibility JavaVersion.VERSION_1_8
    }
    kotlinOptions {
        jvmTarget = '1.8'
    }
    buildFeatures {
        compose true
    }
    composeOptions {
        kotlinCompilerExtensionVersion '1.2.0'
    }
    packagingOptions {
        resources {
            excludes += '/META-INF/{AL2.0,LGPL2.1}'
        }
    }
}

dependencies {

    implementation 'androidx.core:core-ktx:1.7.0'
```

```
    implementation 'androidx.lifecycle:lifecycle-runtime-ktx:2.3.1'
    implementation 'androidx.activity:activity-compose:1.3.1'
    implementation "androidx.compose.ui:ui:$compose_ui_version"
    implementation "androidx.compose.ui:ui-tooling-preview:$compose_ui_version"
    implementation 'androidx.compose.material:material:1.2.0'
    implementation 'androidx.room:room-common:2.5.0'
    implementation 'androidx.room:room-ktx:2.5.0'
    testImplementation 'junit:junit:4.13.2'
    androidTestImplementation 'androidx.test.ext:junit:1.1.5'
    androidTestImplementation 'androidx.test.espresso:espresso-core:3.5.1'
    androidTestImplementation "androidx.compose.ui:ui-test-junit4:$compose_ui_version"
    debugImplementation "androidx.compose.ui:ui-tooling:$compose_ui_version"
    debugImplementation "androidx.compose.ui:ui-test-manifest:$compose_ui_version"
}
```

Settings.gradle

```
pluginManagement {
    repositories {
        google()
        mavenCentral()
        gradlePluginPortal()
    }
}
```

```
dependencyResolutionManagement {
    repositoriesMode.set(RepositoriesMode.FAIL_ON_PROJECT_REPOS)
    repositories {
        google()
        mavenCentral()
    }
}
rootProject.name = "SnackOrdering"
include ':app'
```

AndroidManifest.xml

```
pluginManagement {
    repositories {
        google()
        mavenCentral()
        gradlePluginPortal()
    }
}
dependencyResolutionManagement {
    repositoriesMode.set(RepositoriesMode.FAIL_ON_PROJECT_REPOS)
    repositories {
        google()
```

```
        mavenCentral()
    }
}
rootProject.name = "SnackOrdering"
include ':app'
```

AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools">

    <application
        android:allowBackup="true"
        android:dataExtractionRules="@xml/data_extraction_rules"
        android:fullBackupContent="@xml/backup_rules"
        android:icon="@drawable/fast_food"
        android:label="@string/app_name"
        android:supportsRtl="true"
        android:theme="@style/Theme.SnackOrdering"
        tools:targetApi="31">
        <activity
            android:name=".AdminActivity"
```

```
    android:exported="false"
    android:label="@string/title_activity_admin"
    android:theme="@style/Theme.SnackOrdering" />
<activity
    android:name=".LoginActivity"
    android:exported="true"
    android:label="SnackSquad"
    android:theme="@style/Theme.SnackOrdering">
    <intent-filter>
        <action android:name="android.intent.action.MAIN" />

        <category android:name="android.intent.category.LAUNCHER" />
    </intent-filter>
</activity>
<activity
    android:name=".TargetActivity"
    android:exported="false"
    android:label="@string/title_activity_target"
    android:theme="@style/Theme.SnackOrdering" />
<activity
    android:name=".MainPage"
    android:exported="false"
```

```
        android:label="@string/title_activity_main_page"
        android:theme="@style/Theme.SnackOrdering" />
    <activity
        android:name=".MainActivity"
        android:exported="false"
        android:label="MainActivity"
        android:theme="@style/Theme.SnackOrdering" />
  </application>

</manifest>
```