

# **CONTENTS**

## **1. INTRODUCTION**

1.1 Overview

1.2 Purpose

## **2. PROBLEM DEFINITION & DESIGN THINKING**

2.1 Empathy Map

2.2 Ideation & Brainstorming Map

## **3. RESULT**

## **4. ADVANTAGES & DISADVANTAGES**

## **5. APPLICATIONS**

## **6. CONCLUSION**

## **7. FUTURE SCOPE**

## **8. APPENDIX**

Source Code

## INTRODUCTION

The main objective for the Snack Squad is a customizable snack ordering and delivering app designed to make it easy and convenient for people to order their favorite snacks and have them delivered right to their doorstep. They can also choose to have their snacks delivered to their home or office, or even to a friend or family member as a gift. It's a great way to satisfy your snack cravings and save money at the same time.

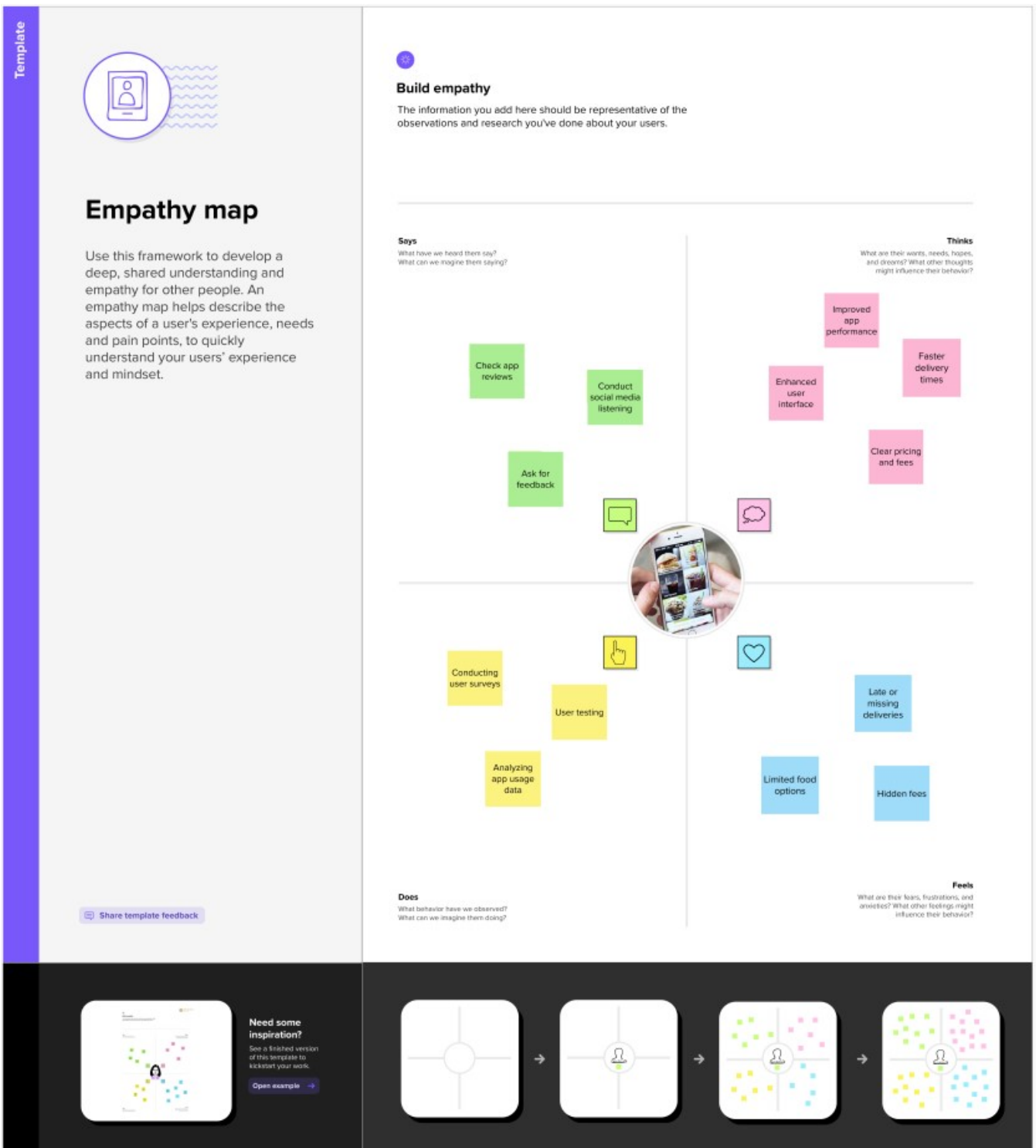
### 1.1 Overview:

- The snack ordering app for Android is a simple and user-friendly application that allows users to browse and order snacks with ease. The app features a sign-in and sign-up page for user privacy and security.
- During the ordering process, users can select the quantity of each snack item they want to order.
- Users can proceed to place their order. The app provides a summary of the order.
- The snack ordering app also offers flexible delivery options, allowing users to choose between immediate delivery or schedule a delivery.

### 1.2 Purpose:

The purpose of your snack ordering app in Android is to provide a convenient and user-friendly platform for customers to order snacks online from your store. By offering a digital ordering system, you can expand your reach and cater to a wider audience, including those who prefer to order online or on-the-go. The app can also provide a personalized and customizable experience for users, allowing them to select their desired snack items, customize their orders, and place their orders with ease. Additionally, the app can provide a more streamlined and efficient ordering process, reducing the potential for errors and minimizing wait times for customers.

### Empathy Map:



### Ideation & Brainstorming Map:

### Brainstorm & idea prioritization

Use this template in your own brainstorming sessions so your team can unleash their imagination and start shaping concepts even if you're not sitting in the same room.

- 6 slides in preview
- 2 hours in duration
- 3 examples included

#### Before you collaborate

A few key prerequisites go into making any of this online space really work, so we'll go through them in the beginning.

5 minutes

#### Define your problem statement

What problem are you trying to solve? Frame your problem as a how/what/for statement. The aim is to focus all your brainstorm

5 minutes

#### Brainstorm

With clear key ideas that come to mind that address your problem statement

15 minutes

#### Group ideas

Now we're taking your ideas into clustering groups or related ideas as you go. Group all ideas into four main groups, put each cluster in a separate box. It's easier to begin than to skip ideas, by and by you will know if you're under- or over-ideating.

15 minutes

#### Prioritize

Now we'll take all the ideas from the same page and start prioritizing them. Place your ideas on the grid to determine which ones are important and which are feasible.

30 minutes

#### After you collaborate

You can export the final work image to pdf to share with members of your company or right into figma.

Quick add-on

- Brainstorm: Brainstorm ideas for your problem statement
- Export: Export the final work image to pdf to share with members of your company or right into figma

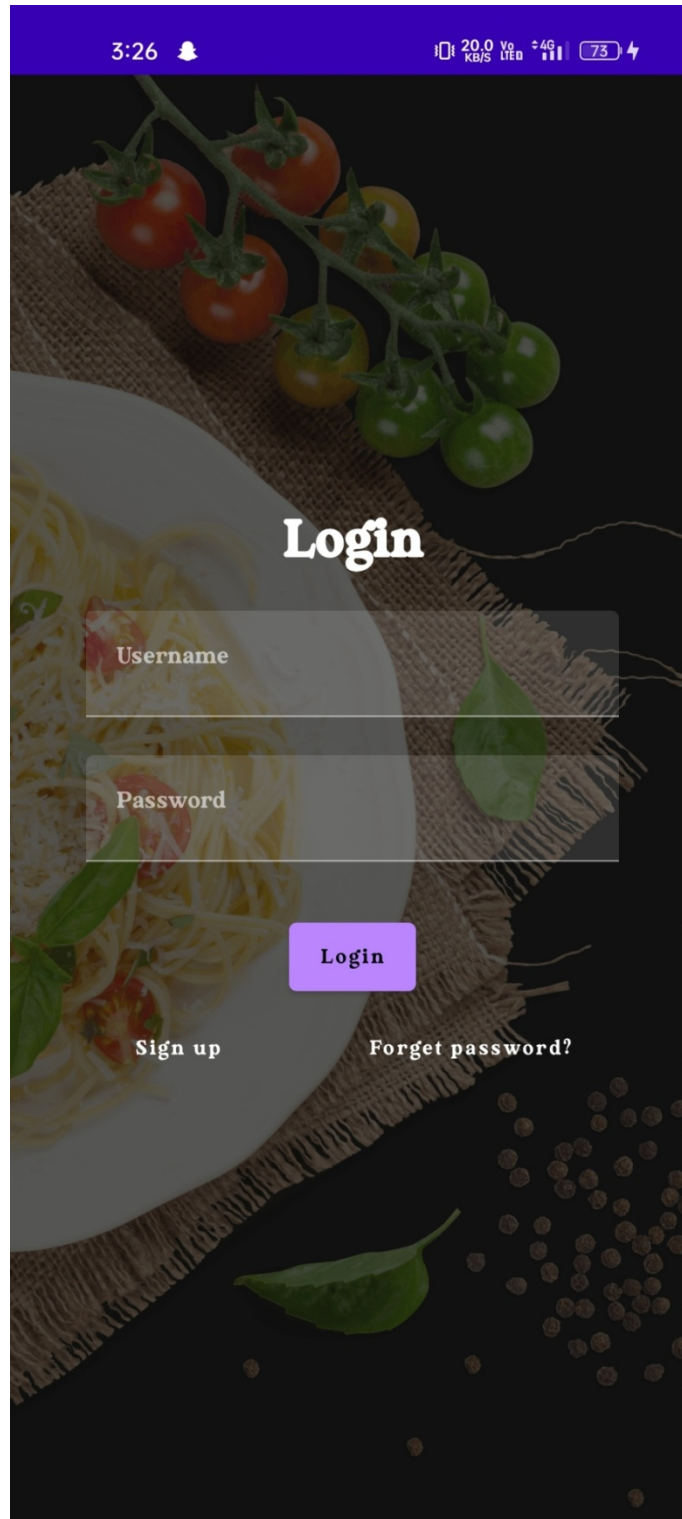
Keep meeting focused

- Brainstorming: Brainstorm ideas for your problem statement
- Export: Export the final work image to pdf to share with members of your company or right into figma

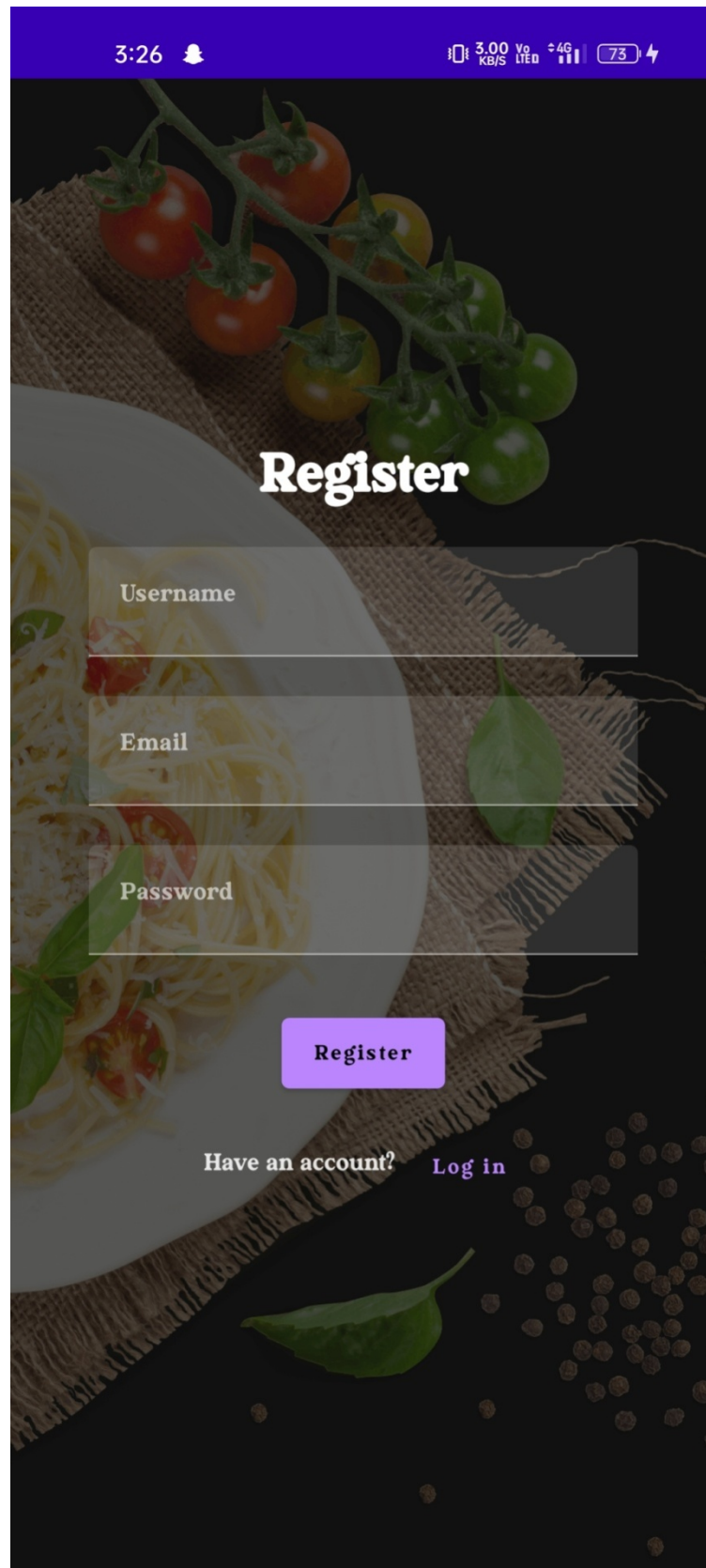
Brainstorming and prioritization: Brainstorm ideas for your problem statement and prioritize them

## RESULT

### Sign in page



## Sign up page:

A mobile application registration screen. The background is a dark, high-quality photograph of a plate of spaghetti with tomato sauce, fresh basil leaves, and a cluster of cherry tomatoes on a vine. The registration form is centered and consists of three input fields for 'Username', 'Email', and 'Password', followed by a blue 'Register' button. At the bottom, there is a link to 'Log in' for users who already have an account. The top of the screen shows a standard Android status bar with the time 3:26, a notification icon, network speed (3.00 KB/s), 4G LTE signal, and a 73% battery level.

3:26

3.00 KB/s 4G LTE 73%

# Register

Username

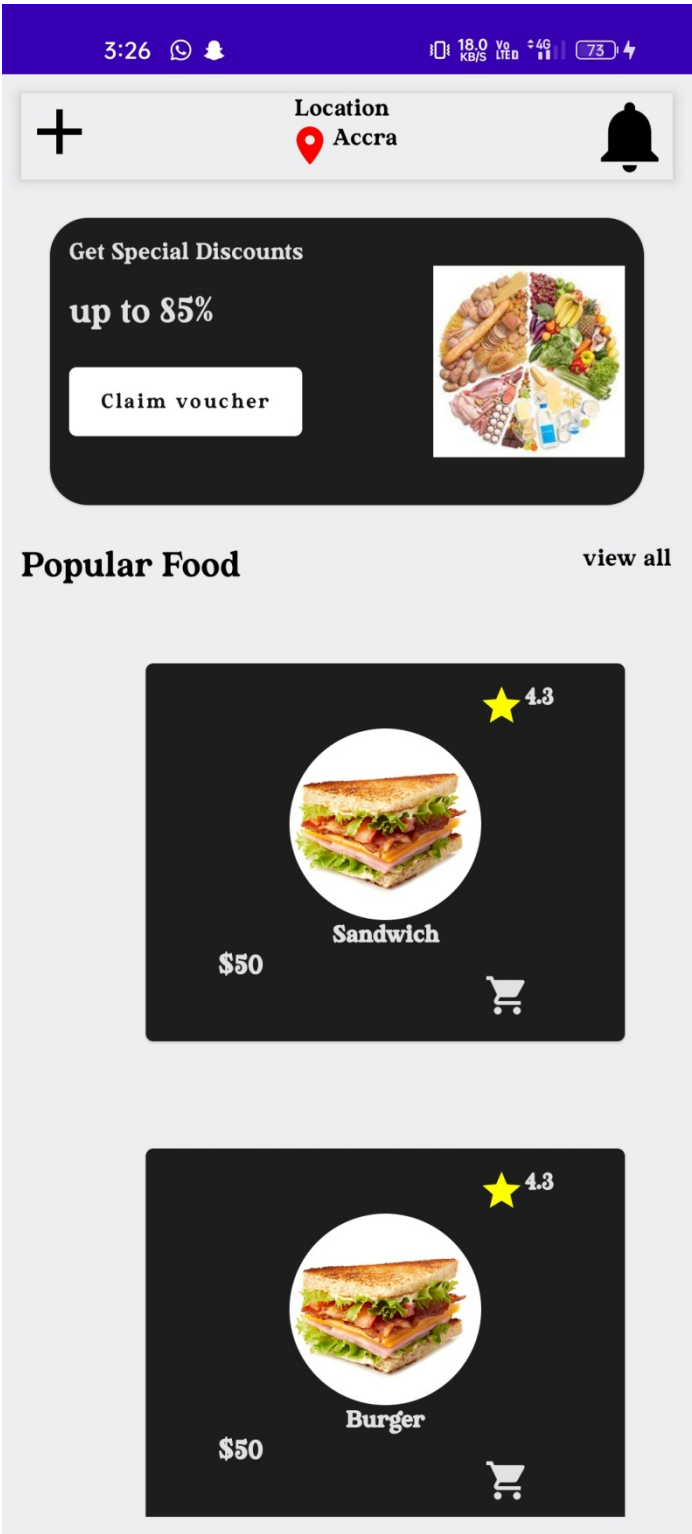
Email

Password

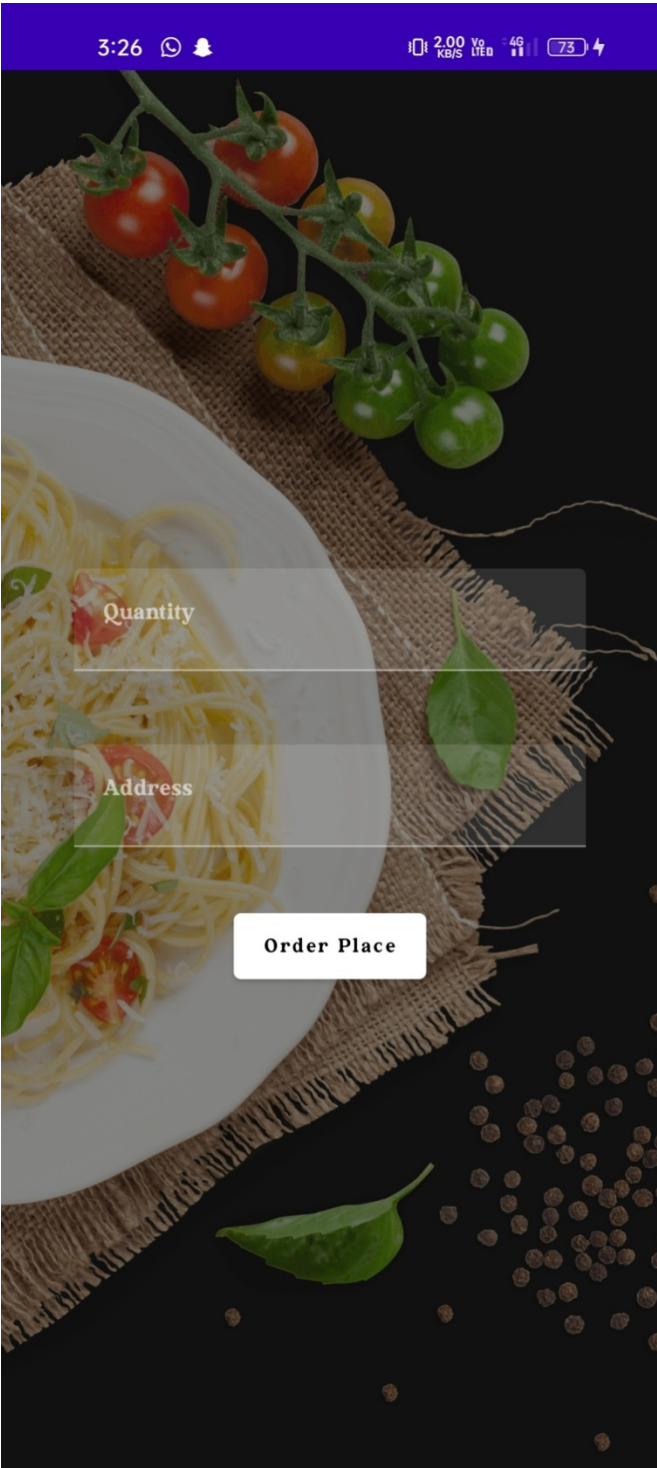
Register

Have an account? [Log in](#)

Home Page:



Order Screen page:





## **ADVANTAGES & DISADVANTAGES**

### **Advantages:**

- Convenience: Users can easily order their favorite snacks from the comfort of their own homes, without having to leave to go to the store.
- Selection: Users can access a wide selection of snack items through this app, including hard-to-find or specialty items.
- Delivery options: Users can choose from a variety of delivery options, such as immediate delivery or scheduled delivery times, to fit their needs and preferences.
- User-friendly: This app user interface is designed to be simple and easy to use, even for users who are not tech-savvy.
- Efficiency: This app checkout process is efficient and streamlined, allowing users to complete their orders quickly and easily.
- Privacy: This app sign-in and sign-up pages provide user privacy and ensure that their personal information is kept secure.

### **Disadvantages:**

- Technical difficulties: This app may encounter technical difficulties, such as server issues or bugs, which could impact its functionality.
- Delivery delays: This app delivery system may experience delays, which could lead to frustration and inconvenience for users.
- Quality control: This app quality control processes may not be as rigorous as in-person shopping, leading to potential issues with product quality or accuracy.
- Limited product availability: This app may not offer the same range of products or brands as in-person stores, limiting user choice.
- Security risks: This app payment system may be vulnerable to security risks, such as hacking or fraud.

## APPLICATIONS

- Restaurants and cafes: Many restaurants and cafes have their own ordering apps that allow customers to place orders for pickup or delivery.
- Grocery stores: Some grocery stores have their own ordering apps that allow customers to order groceries online for pickup or delivery.
- Food delivery services: Food delivery services such as Grubhub, Uber Eats, and DoorDash allow users to order food from a variety of restaurants and have it delivered to their door.
- Retail stores: Some retail stores have their own ordering apps that allow customers to order products online for pickup or delivery.
- Pharmacies: Some pharmacies have their own ordering apps that allow customers to order prescription medications and other health-related products for pickup or delivery.
- Convenience stores: Some convenience stores have their own ordering apps that allow customers to order snacks, beverages, and other items for pickup or delivery.

## **Conclusion**

We conclude that the snack ordering app in Android has the potential to offer a convenient and easy way for users to order snacks online. With features such as a sign in and signup page for user privacy, a list of snack items, and the ability to select quantity and place an order, the app can provide a seamless ordering experience for users. With careful planning, development, and testing, your snack ordering app in Android has the potential to be a successful and useful tool for users who want to order snacks online.

## **Future Scopes**

In future adding more sources to provide users with Integration with loyalty, Real-time order tracking, Social media integration, Integration with chatbots AI for assist and consider implementing a user review and rating system to provide social proof and help users make informed decisions about their snack orders. Integration with delivery services to provide users with a wider range of delivery options and faster delivery times.

## APPENDIX

Source Code :

<https://github.com/Kannan065/SnackSquad>

### 1. AdminActivity.kt

```
package com.example.snackordering

import android.icu.text.SimpleDateFormat
import android.os.Bundle
import android.util.Log
import androidx.activity.ComponentActivity
import androidx.activity.compose.setContent
import androidx.compose.foundation.Image
import androidx.compose.foundation.layout.*
import androidx.compose.foundation.lazy.LazyColumn
import androidx.compose.foundation.lazy.LazyRow
import androidx.compose.foundation.lazy.items
import androidx.compose.material.MaterialTheme
import androidx.compose.material.Surface
import androidx.compose.material.Text
import androidx.compose.runtime.Composable
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.unit.dp

import androidx.compose.ui.unit.sp

import com.example.snackordering.ui.theme.SnackOrderingTheme

import java.util.*
```

```
class AdminActivity : 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(),

                    color = MaterialTheme.colors.background

                ) {

                    val data=orderDatabaseHelper.getAllOrders();

                    Log.d("swathi" ,data.toString())
```

```

        val order = orderDatabaseHelper.getAllOrders()

        ListListScopeSample(order)

    }

}

}

}

}

```

@Composable

```

fun ListListScopeSample(order: List<Order>) {

    Image(

        painterResource(id = R.drawable.order), contentDescription = "",

        alpha = 0.5F,

        contentScale = ContentScale.FillHeight)

    Text(text = "Order Tracking", modifier = Modifier.padding(top = 24.dp, start = 106.dp,
bottom = 24.dp ), color = Color.White, fontSize = 30.sp)

    Spacer(modifier = Modifier.height(30.dp))

    LazyRow(

        modifier = Modifier

            .fillMaxSize()

            .padding(top = 80.dp),

```

```
horizontalArrangement = Arrangement.SpaceBetween
```

```
) {
```

```
  item {
```

```
    LazyColumn {
```

```
      items(order) { order ->
```

```
        Column(modifier = Modifier.padding(top = 16.dp, start = 48.dp, bottom = 20.dp)) {
```

```
          Text("Quantity: ${order.quantity}")
```

```
          Text("Address: ${order.address}")
```

```
        }
```

```
      }
```

```
    }
```

```
  }
```

```
}
```

```
}
```



## 2. 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.order), 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.Cursive,
    color = Color.White,
```

```
        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.White,text = "Sign up") }

    TextButton(onClick = {

    })

    {

        Spacer(modifier = Modifier.width(60.dp))

        Text(color = Color.White,text = "Forget password?")

    }

}

}

}

private fun startMainPage(context: Context) {

    val intent = Intent(context, MainPage::class.java)

    ContextCompat.startActivity(context, intent, null)

}
```

### 3. 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 = MaterialTheme.colors.background
```

```
        ) {
```

```
            FinalView(this)
```

```
            val context = LocalContext.current
```

```
            //PopularFoodColumn(context)
```

```
        }
```

```
    }
```

```
}
```

```
}
```

```
}
```

```
@Composable
```

```
fun TopPart() {
```

```
    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 = "Accra" , 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)

                Button(onClick = {}, colors = ButtonDefaults.buttonColors(Color.White)) {

                    Text(text = "Claim voucher", color = MaterialTheme.colors.surface)
                }
            }
        }
    }
}

```

```

        }

    }

    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(0.7f), 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.sandwish 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)
```

```
        }
```

```
    }
```

```
}
```

#### 4. 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.order), 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.Cursive,  
    color = Color.White,  
    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(text = "Log in")

    }

}

}

}

private fun startLoginActivity(context: Context) {

    val intent = Intent(context, LoginActivity::class.java)

    ContextCompat.startActivity(context, intent, null)

}

```

## 5. 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.order), 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)

}

```