

Project Report

Snack Squad: A Customizable Snack Ordering and Delivery App

1. Introduction

1.1 Overview

We have developed a Snack Squad App using Kotlin and jetpack Compose. It is a compact app that you order any food or snack items.

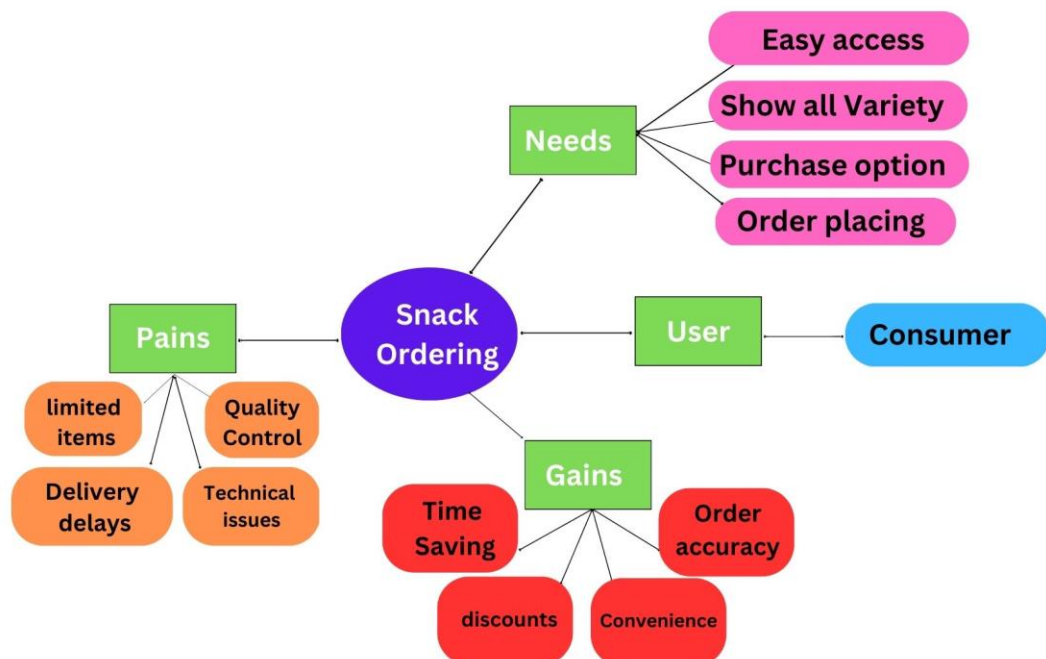
1.2 Purpose

The main purpose of the Snack Squad app is that it is comfortable to order your delicious food at anywhere you locate and you can get your item at your location.

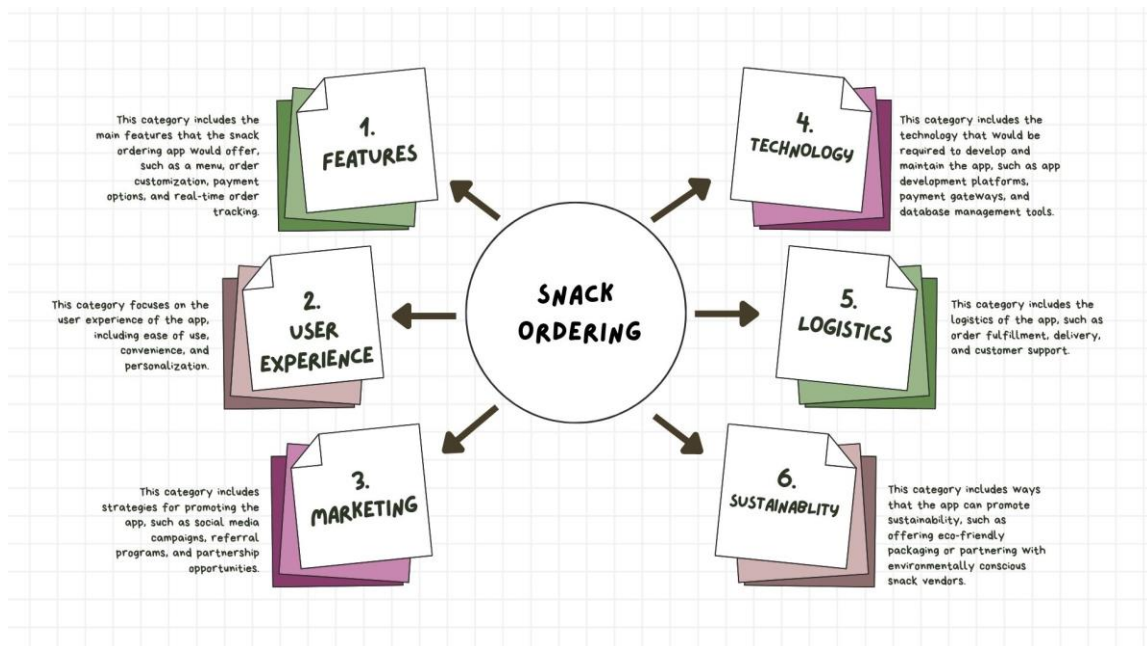
2. Problem Definition & Design Thinking

2.1 Empathy Map

Design Thinking Workshop

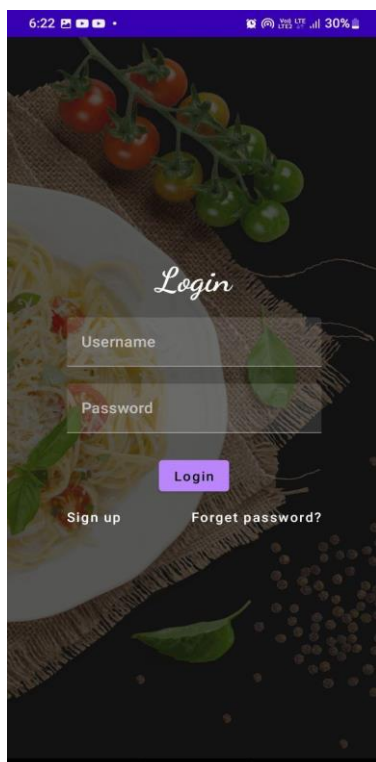


2.2 Ideation and Brainstorm Map

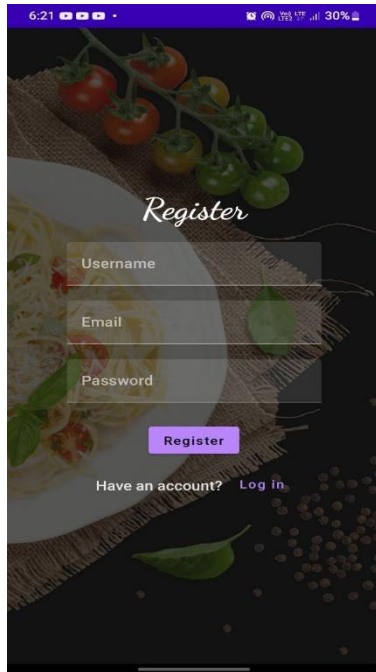


3. Result:

Login Page:



Register page



A screenshot of a mobile application's registration page. The background features a dark, artistic image of a pizza and cherry tomatoes. The word "Register" is written in a white, cursive font. Below it are three white input fields for "Username", "Email", and "Password". A blue "Register" button is positioned below the password field. At the bottom, the text "Have an account? Log in" is displayed, with "Log in" as a link.

6:21 30%

Register

Username

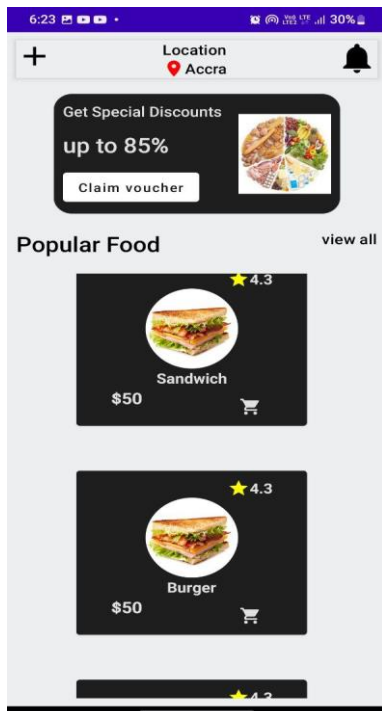
Email

Password

Register

Have an account? [Log in](#)

Main page



A screenshot of a mobile application's main page. The top status bar shows the time as 6:23 and 30% battery. The header includes a plus icon, the location "Accra" with a red pin, and a bell icon. A promotional banner for "Special Discounts up to 85%" with a "Claim voucher" button and a food collage image is present. The "Popular Food" section lists items with their prices and ratings. A "view all" link is next to the section title. A bottom bar shows a 4.2 rating.

6:23 30%

+ Location Accra

Get Special Discounts up to 85%

Claim voucher

Popular Food view all

★ 4.3

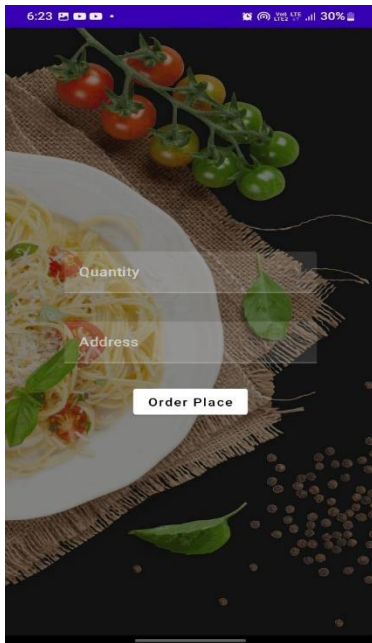
Sandwich \$50

★ 4.3

Burger \$50

★ 4.2

Target page



4. Advantages

- Convenience: With a snack ordering app, you can order your favorite snacks from the comfort of your home or office. You don't have to go to the store or stand in line to place an order. You can order snacks anytime, anywhere, and have them delivered to your doorstep.
- Time saving: Ordering snacks from an app saves you time, as you don't have to spend time going to the store, standing in line, or waiting for your order to be prepared. You can order snacks with just a few taps on your smartphone or tablet and have them delivered to you quickly.
- Wide variety: Snack ordering apps usually offer a wide variety of snacks, from healthy options to indulgent treats. You can easily browse through different snacks and choose the ones that best suit your taste and preferences.
- Personalization: Some snack ordering apps allow you to customize your order, so you can add or remove ingredients, choose the level of spiciness, or specify any dietary restrictions.
- Loyalty rewards: Many snack ordering apps offer loyalty rewards programs, where you can earn points or discounts for every order you place. This can save you money in the long run and encourage you to use the app more frequently.
- User reviews: Most snack ordering apps allow users to leave reviews and ratings, so you can see what other customers think about a particular snack or vendor. This can help you make more informed

Disadvantages

- **Technical issues:** Like all software, snack ordering apps can sometimes experience technical issues or glitches that can result in delayed orders or incorrect charges. This can be frustrating for users and may require customer support to resolve.
- **Delivery issues:** While most snack ordering apps have reliable delivery services, there can sometimes be issues with late or missing deliveries. This can be especially frustrating if you're hungry and waiting for your order to arrive.
- **Limited selection:** While many snack ordering apps offer a wide variety of snacks, some may have a limited selection or only work with a few vendors in your area. This can be disappointing if you're looking for a specific snack or vendor that isn't available on the app.
- **Hidden fees:** Some snack ordering apps may have hidden fees or surcharges that aren't clearly disclosed upfront. This can lead to unexpected charges and a higher overall cost for your order.
- **Dependence on technology:** While snack ordering apps are convenient, they do require a certain level of technological savvy to use. This can be a barrier for some users, especially those who aren't comfortable with smartphones or tablets.
- **Privacy concerns:** Snack ordering apps often require users to create an account and provide personal information, such as their name, email address, and credit card information. This can raise privacy concerns for some users, especially if the app's privacy policy isn't clear or transparent.

5. Application:

- **User -friendly interface:** The application should have a simple and intuitive interface that allows users to easily see food items and they can order it.
- **Login and Sign-up pages:** Users should be able to create an account or log in to access personalized features such as having many snack items and order it.
- **Main page:** The application should have a page dedicated to displaying a variety of food and snack items.
- **Target page:** The application should have a target page for snack items to buy it.

6. Conclusion:

Overall Snack ordering app is a simple to use, that provides a very easy to use interface for everybody. Further improvements can be added to this app which is being discussed in the next session.

7. Future Scope

Personalized Suggestions:

The predominant thing that is lacking in this app is a personalized suggestions due to lack of resources.

snack ordering app could be to incorporate a "Favorites" section where users can save their most frequently ordered snacks. This feature could allow users to quickly access their preferred snacks and place an order with just a few clicks.

Personalized Pages:

Allowing the user to customize the user interface like adding night mode and etc.

8.Appendix:

Login Activity:

```
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
```

```

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

```

```

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

```

```

    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( text: "Username" ) },
        modifier = Modifier.padding(10.dp)
    )

```

```

        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"
}
},

```

```

        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"
}
},

```

```

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

        { this: RowScope
            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, options: null)
}

```


Register Activity:

```
package com.example.snackordering

import ...

class MainActivity : ComponentActivity() {
    private lateinit var databaseHelper: UserDatabaseHelper
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        databaseHelper = UserDatabaseHelper(context: this)
        setContent {
            SnackOrderingTheme {
                // A surface container using the 'background' color from the theme
                Surface(
                    modifier = Modifier.fillMaxSize(),
                    color = MaterialTheme.colors.background
                ) {
                    RegistrationScreen(context: 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(value: "") }
    var password by remember { mutableStateOf(value: "") }
    var email by remember { mutableStateOf(value: "") }
    var error by remember { mutableStateOf(value: "") }

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

        Text(
            fontSize = 36.sp,
            fontWeight = FontWeight.ExtraBold
        )
    }
}
```

```

        .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)
            }
        }
    )
}

```

```

        .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)
            }
        }
    )
}

```

```

        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)
) { this: RowScope
    Text(text = "Register")
}
Spacer(modifier = Modifier.width(10.dp))
Spacer(modifier = Modifier.height(10.dp))

```

```

@Composable
fun TopPart() {

    Row(
        modifier = Modifier
            .fillMaxWidth()
            .background(Color( color: 0xffeceef0)), Arrangement.SpaceBetween
    ) { this: RowScope
        Icon(
            imageVector = Icons.Default.Add, contentDescription = "Menu Icon",
            Modifier
                .clip(CircleShape)
                .size(40.dp),
            tint = Color.Black,
        )
        Column(horizontalAlignment = Alignment.CenterHorizontally) { this: ColumnScope
            Text(text = "Location", style = MaterialTheme.typography.subtitle1, color = Color.Black)
            Row { this: RowScope
                Icon(
                    imageVector = Icons.Default.LocationOn,
                    contentDescription = "Location",
                    tint = Color.Red,
                )
            }
        }
    }
}

```

Mainpage

```

package com.example.snackordering
//yes
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

```

```

@Composable
fun TopPart() {

    Row(
        modifier = Modifier
            .fillMaxWidth()
            .background(Color( color: 0xffeceef0)), Arrangement.SpaceBetween
    ) { this: RowScope
        Icon(
            imageVector = Icons.Default.Add, contentDescription = "Menu Icon",
            Modifier
                .clip(CircleShape)
                .size(40.dp),
            tint = Color.Black,
        )
        Column(horizontalAlignment = Alignment.CenterHorizontally) { this: ColumnScope
            Text(text = "Location", style = MaterialTheme.typography.subtitle1, color = Color.Black)
            Row { this: RowScope
                Icon(
                    imageVector = Icons.Default.LocationOn,
                    contentDescription = "Location",
                    tint = Color.Red,
                )
            }
        }
    }
}

```

```

@Composable
fun TopPart() {

    Row(
        modifier = Modifier
            .fillMaxWidth()
            .background(Color( color: 0xffeceef0)), Arrangement.SpaceBetween
    ) { this: RowScope
        Icon(
            imageVector = Icons.Default.Add, contentDescription = "Menu Icon",
            Modifier
                .clip(CircleShape)
                .size(40.dp),
            tint = Color.Black,
        )
        Column(horizontalAlignment = Alignment.CenterHorizontally) { this: ColumnScope
            Text(text = "Location", style = MaterialTheme.typography.subtitle1, color = Color.Black)
            Row { this: RowScope
                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) { this: RowScope
            Column(verticalArrangement = Arrangement.spacedBy(12.dp)) { this: ColumnScope
                Text(text = "Get Special Discounts")
                Text(text = "up to 85%", style = MaterialTheme.typography.h5)
                Button(onClick = {}, colors = ButtonDefaults.buttonColors(Color.White)) { this: RowScope
                    Text(text = "Claim voucher", color = MaterialTheme.colors.surface)
                }
            }
        }
    }
}

```

```

        Column(verticalArrangement = Arrangement.spacedBy(12.dp)) { this: ColumnScope
            Text(text = "Get Special Discounts")
            Text(text = "up to 85%", style = MaterialTheme.typography.h5)
            Button(onClick = {}, colors = ButtonDefaults.buttonColors(Color.White)) { this: RowScope
                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

```

```

        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
            ) { this: ColumnScope
                Spacer(modifier = Modifier.padding(vertical = 5.dp))
                Row(
                    modifier = Modifier
                        .fillMaxWidth(fraction: 0.7f), Arrangement.End
                ) { this: RowScope
                    Icon(
                        imageVector = Icons.Default.Star,
                        contentDescription = "Star Icon",
                        tint = Color.Yellow
                    )
                    Text(text = "4.3", fontWeight = FontWeight.Black)
                }
                Image(

```



```
private val FoodList = listOf(
    R.drawable.sandwich to R.string.sandwich,
    R.drawable.sandwich 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
```

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

    Column(
        modifier = Modifier
            .fillMaxSize()
            .background(Color( color: 0xffeceef0))
            .padding(10.dp),
        verticalArrangement = Arrangement.Top,
        horizontalAlignment = Alignment.CenterHorizontally
    ) { this: ColumnScope
        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) { this: RowScope
            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
```

```

fun PopularFoodColumn(context: Context) {

    LazyColumn(
        modifier = Modifier.fillMaxSize(),

        content = { this: LazyListScope
            items(FoodList) { this: LazyItemScope 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() { it: PaddingValues
            val context = LocalContext.current
            App(context)
        }
    }
}

```

T

TargetActivity:

```
package com.example.snackordering

import ...

class TargetActivity : ComponentActivity() {
    private lateinit var orderDatabaseHelper: OrderDatabaseHelper
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        orderDatabaseHelper = OrderDatabaseHelper(context: this)
        setContent {
            SnackOrderingTheme {
                // A surface container using the 'background' color from the theme
                Surface(
                    modifier = Modifier
                        .fillMaxSize()
                        .background(Color.White)
                ) {
                    Order(context: this, orderDatabaseHelper)
                    val orders = orderDatabaseHelper.getAllOrders()
                    Log.d(tag: "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) { this: ColumnScope

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

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

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

```

        label = { Text( 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

```

```

        Button(onClick = {
            if( quantity.isNotEmpty() and address.isNotEmpty()){
                val order = Order(
                    id = null,
                    quantity = quantity,
                    address = address
                )
                orderDatabaseHelper.insertOrder(order)
                Toast.makeText(mContext, text = "Order Placed Successfully", Toast.LENGTH_SHORT).show()
            },
            colors = ButtonDefaults.buttonColors(backgroundColor = Color.White))
        { this: RowScope
            Text(text = "Order Place", color = Color.Black)
        }
    }
}

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

```