1.Introduction

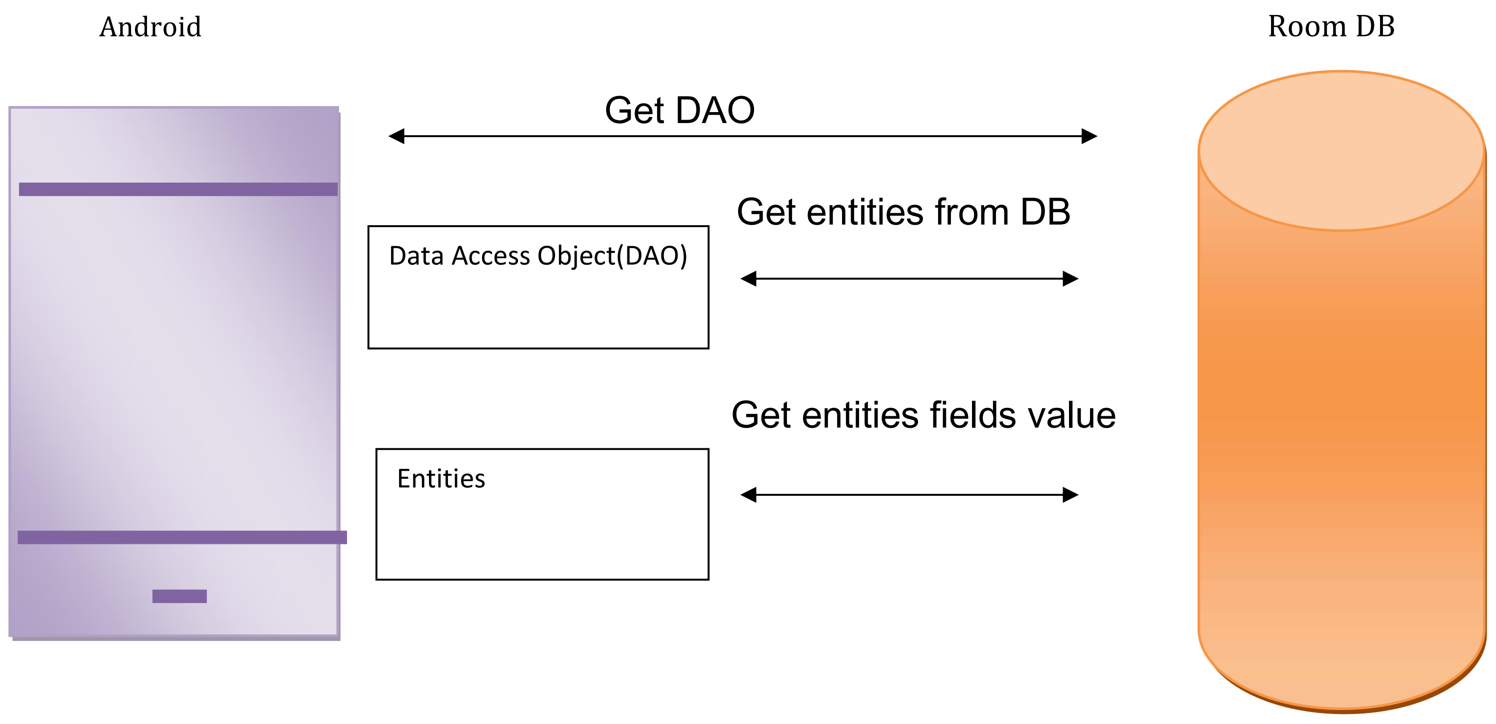
1.1 project overview

Snack Squad: A Customizable Snack Ordering And Delivery App

Project Description:

Snack Squad is a demonstration project showcasing the capabilities of Android Jetpack Compose for crafting intuitive and visually appealing UIs. Designed as a sample e-commerce application, it provides a streamlined shopping experience focused on snacks. Users can explore a curated list of snacks, select their favorites, and seamlessly add them to a virtual cart with a single tap. The app also includes a cart view where users can review their selected items and proceed to checkout to finalize their purchase. This project highlights the use of Compose libraries to build modern, responsive, and user-friendly Android applications.

Architecture:



LEARNING OUTCOMES:

- Gain proficiency in Android Studio for building mobile applications.

- Learn to implement and manage database integrations effectively.

Project Workflow:

- Users create an account to register within the app.

- After registration, users log in to access the application.

- Users land on the main screen upon login.

- Users can browse available items, select desired products, and place orders.

- Admin access allows viewing and managing placed orders.

Note:

This project guides you through setting up, developing, and testing a snack-based e-commerce app using Android Jetpack Compose and database integration.

Tasks:

1. Preliminary Setup

- Set up the project environment and prepare your development tools.

2. Project Initialization

- Start a new Android project in Android Studio.

3. Adding Necessary Libraries

- Include required dependencies and libraries in your project for functionality.

4. Designing Database Architecture

- Define and create the database models and classes for data storage.

5. Developing User Interface and Database Integration

- Design the app’s user interface using Jetpack Compose, and link it with the database for dynamic content.

6. Configuring AndroidManifest.xml

- Modify the AndroidManifest.xml file to declare app components and permissions.

7. Running and Testing the Application

- Execute the app on an emulator or physical device to test functionality and ensure it works as expected.

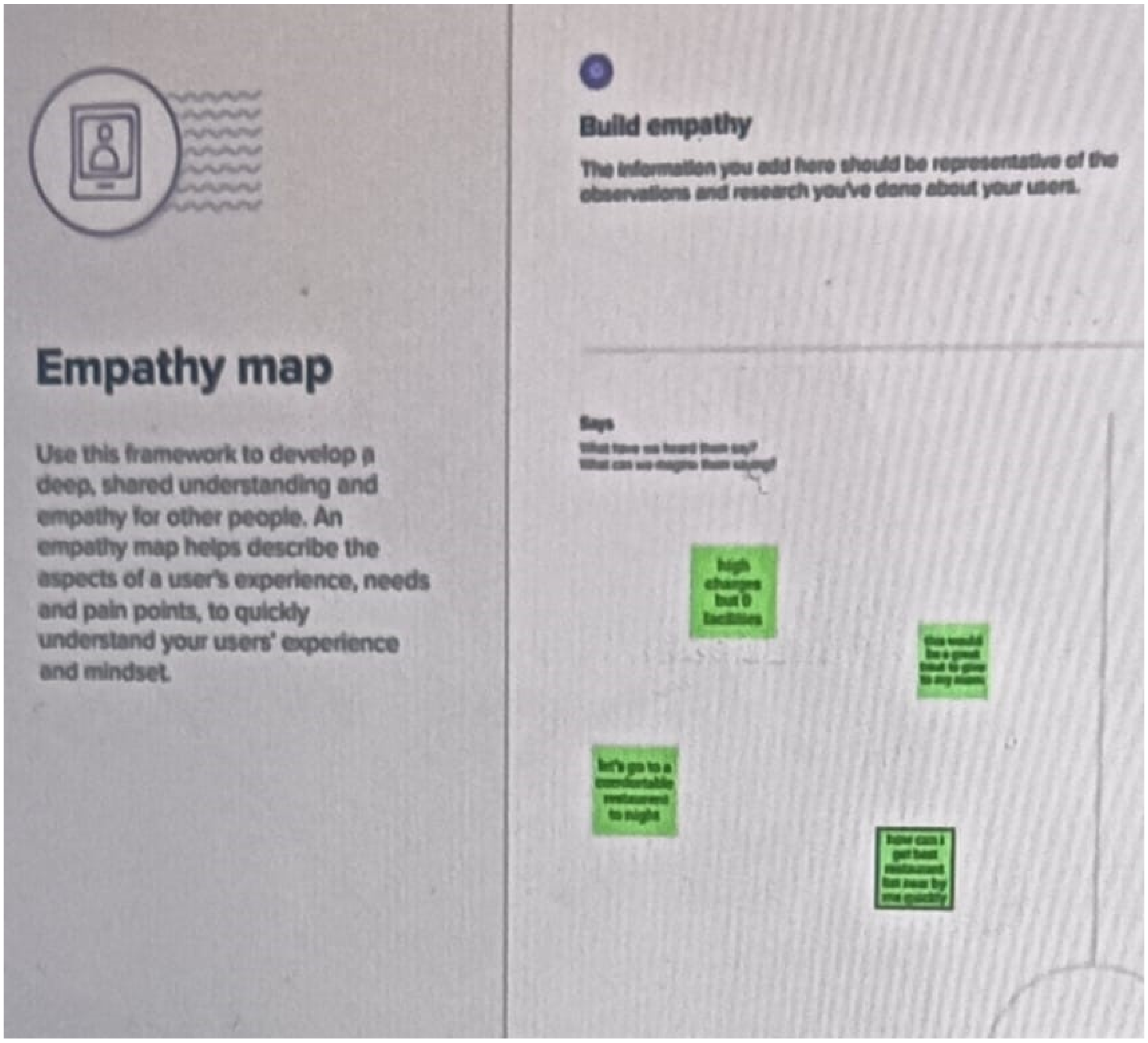
1.2 Purpose of the Project:

The Snack Squad app aims to simplify snack ordering and delivery by providing users with a customizable, user-friendly platform to order their favorite snacks and have them delivered to any location. This app focuses on making snack selection easier while allowing users to personalize their orders according to preferences, dietary needs, or group requirements.

Snack Squad is ideal for both individual users and organizations like offices or schools, offering a wide range of snacks—including healthy alternatives. It enables users to customize their orders and avoid the inconvenience of shopping or making multiple trips. With an integrated delivery feature, Snack Squad ensures that snacks are brought directly to the user's doorstep, offering a convenient and time-saving solution.

2.Problem Definition and Design Thinking:

2.1.Empathymap:



Empathy map

Use this framework to develop a deep, shared understanding and empathy for other people. An empathy map helps describe the aspects of a user's experience, needs and pain points, to quickly understand your users' experience and mindset.

Build empathy

The information you add here should be representative of the observations and research you've done about your users.

Says

What have we heard them say?
What can we imagine them saying?

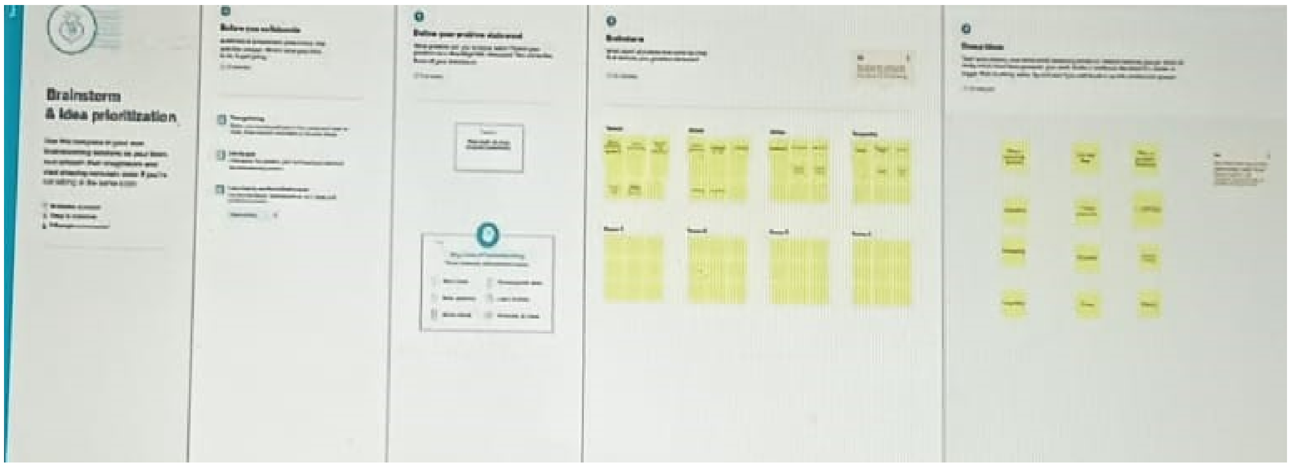
high charges but 0 facilities

this would be a great treat to give to my mom

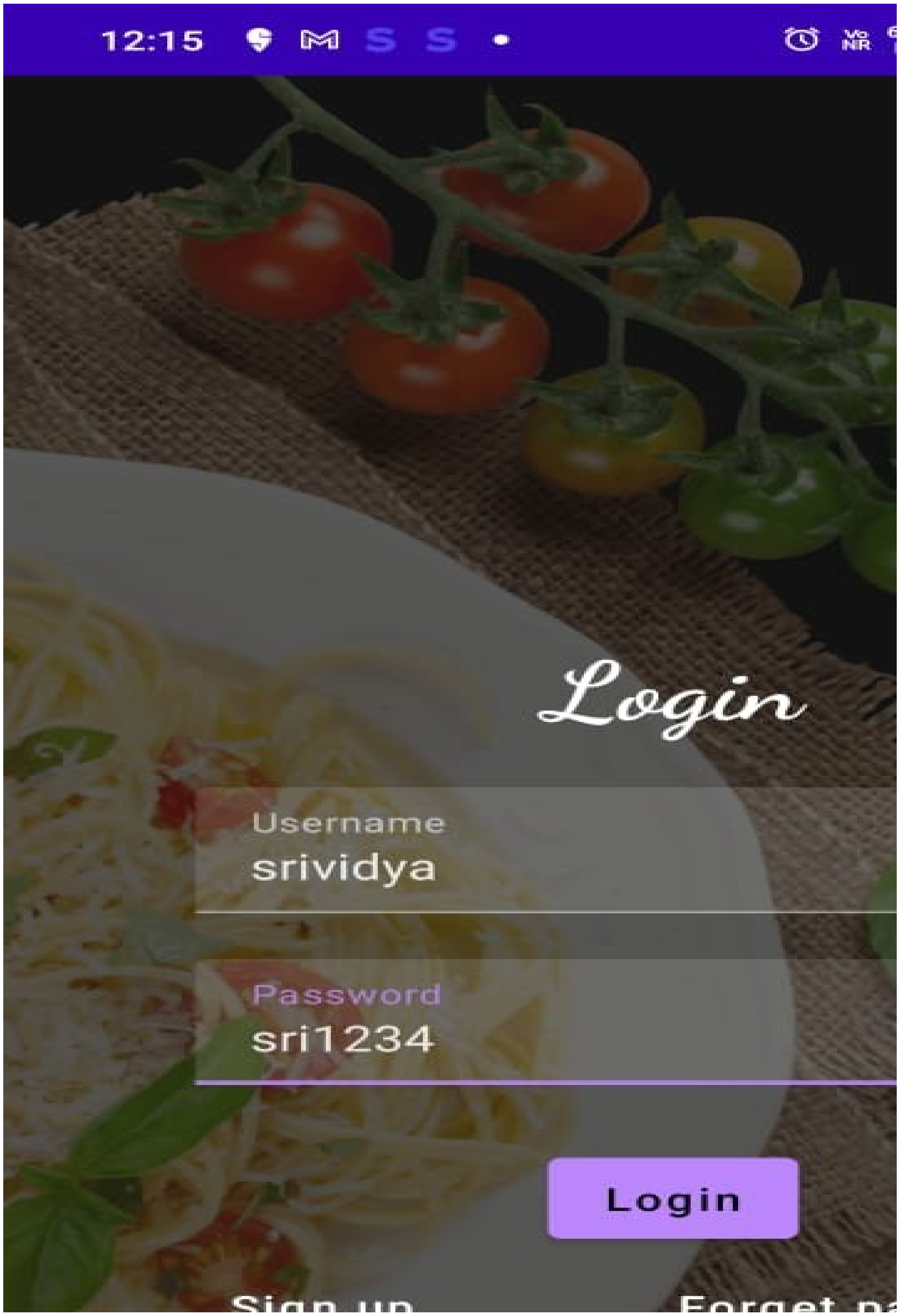
let's go to a comfortable restaurant to night

how can i get best restaurant list near by me quickly

2.2.Ideation&Brainstroming map



3.Result


12:15
Login
Username
srividya
Password
sri1234
Login
Sign up
Forget pa

Advantages:

1.Enhanced User Experience: A customizable snack ordering and delivery app can provide a seamless and enjoyable experience for customers by offering a user-friendly interface, easy navigation, and efficient order placement.

2.Wider Customer Reach: The app can cater to a larger and more diverse customer base, including individuals who might not typically visit the physical store. Customers from different locations can place orders, increasing market reach.

3.Loyalty and Retention: The app can offer features like loyalty programs, personalized discounts, or reward systems for frequent users, encouraging repeat purchases and increasing customer retention.

4.Real-Time Tracking: Customers can track their orders in real-time, giving them more control over the delivery process and ensuring they are updated on the status of their snack orders.

5.Operational Efficiency: For the business, the app can streamline the ordering process, reduce human error, and allow better inventory management by tracking popular items, order history, and customer preferences.

Disadvantages:

1.High Development and Maintenance Costs: Developing and maintaining a fully functional app can be expensive, especially if customization options and advanced features (like real-time tracking or a recommendation system) are included. Regular updates and bug fixes also add to the cost.

2.Dependence on Technology: A highly digital system means customers rely on their smartphones and internet connectivity to place orders, which may limit access for people who are less tech-savvy or do not have access to modern smartphones.

3. Delivery Challenges: The success of the app heavily depends on the reliability of delivery services. Late or incorrect deliveries can lead to dissatisfied customers and negative reviews, affecting the business's reputation.

4.Customer Privacy Concerns: The app collects sensitive data, such as customer preferences, payment details, and location information. This raises concerns about data privacy and security, requiring businesses to invest in robust security measures.

5.Limited Social Interaction: While the app offers convenience, it removes the personal interaction that some customers might enjoy when ordering in-store, potentially reducing customer engagement and brand loyalty.

Applications:

1.Grocery Stores: Grocery stores can use the app to offer snack items, along with other grocery products, allowing customers to place orders for snacks and have them delivered along with their regular grocery shopping.

2.Movie Theaters: Movie theaters could partner with the app to allow customers to pre-order snacks like popcorn, candy, and beverages, either for delivery to their seats or for easy pickup before the movie starts.

3.Corporate Offices: Offices and workplaces can adopt the app to provide employees with an easy way to order snacks or drinks for office meetings or personal breaks, offering both delivery and group ordering options.

4.Event Catering: Catering services for events, parties, or conferences can use the app to offer a wide range of snack options that can be ordered and delivered on-demand, making it easier for event organizers to manage snack logistics.

5.Convenience Stores: Convenience stores can use the app to provide an array of snack options for local customers, offering them the ability to place orders for immediate or scheduled delivery, catering to people looking for quick, accessible snacks without leaving home.

Conclusion:

In conclusion, a customizable snack ordering and delivery app presents an innovative solution to meet the increasing demand for convenience and personalization in the snack industry. The app should not only provide a variety of snack options but also offer seamless order customization, efficient delivery tracking, and secure payment methods. By simplifying the snack ordering process and enabling a tailored experience for customers, such an app can enhance both customer satisfaction and operational efficiency for snack vendors. Additionally, incorporating features like loyalty programs or personalized promotions can drive engagement and customer retention.

Overall, the customizable snack ordering and delivery app can reshape the snack service landscape, offering busy consumers an easy and quick way to satisfy their snack cravings while benefiting businesses with greater reach and sales potential.

Future Scope:

The future of customizable snack ordering and delivery apps looks promising, driven by technological advancements and shifting consumer preferences toward more personalized and efficient services. Here are some key areas for future growth and innovation:

1**.Enhanced Personalization with AI and Data Analytics:** Leveraging artificial intelligence (AI) and advanced data analytics can take user personalization to the next level. By analyzing customer preferences, purchase history, and behavior, the app can suggest snack combinations, promotions, or new products that align with individual tastes, improving user satisfaction and boosting sales.

2.Voice-Activated Ordering: As voice-activated technology becomes more widespread, the app could integrate with popular voice assistants (like Alexa, Siri, or Google Assistant). This feature would allow customers to place snack orders hands-free, making the process even more convenient, especially for those multitasking or when on the go.

3.Subscription-Based Models: The app could explore a subscription model where customers sign up for a regular delivery of their favorite snacks, offering both convenience and cost savings. Subscriptions can be customized for different tastes and dietary preferences, encouraging repeat business and fostering loyalty.

4.Sustainability Initiatives: As sustainability becomes increasingly important to consumers, future versions of the app could focus on eco-friendly packaging, offering options for organic or locally-sourced snacks, and providing carbon-offset delivery choices. This would appeal to environmentally-conscious customers and enhance the brand’s reputation.

5.Partnership with Health and Wellness Brands: With a growing trend towards healthy eating, the app could partner with health-conscious snack brands to offer a wider range of nutritious, organic, or diet-specific options, such as gluten-free, keto, or vegan snacks.

6.Global Expansion and Local Customization: In the future, the app could expand into international markets, adapting its offerings to cater to regional snack preferences, dietary habits, and local tastes. This could make the app more relevant in diverse markets and help increase its global footprint.

By integrating these innovative features and expanding into new territories, a customizable snack ordering and delivery app can continue to evolve, providing significant value to both customers and businesses while keeping up with the fast-paced changes in the tech and food delivery industries.

8.Appendix

A.source code 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)

}

}

}
```

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)}
```