GOVERNMENT COLLEGE OF TECHNOLOGY
COIMBATORE-641013

SNACK SQUAD

A Customizable Snack Ordering and Delivery App

Submitted by

Abinaya D (71772217103)

Sundari K (71772217148)

Thavasi Prabha S (71772217150)

Rajalalshmi S (71772217L03)

Keerthana R (71772217305)

Introduction

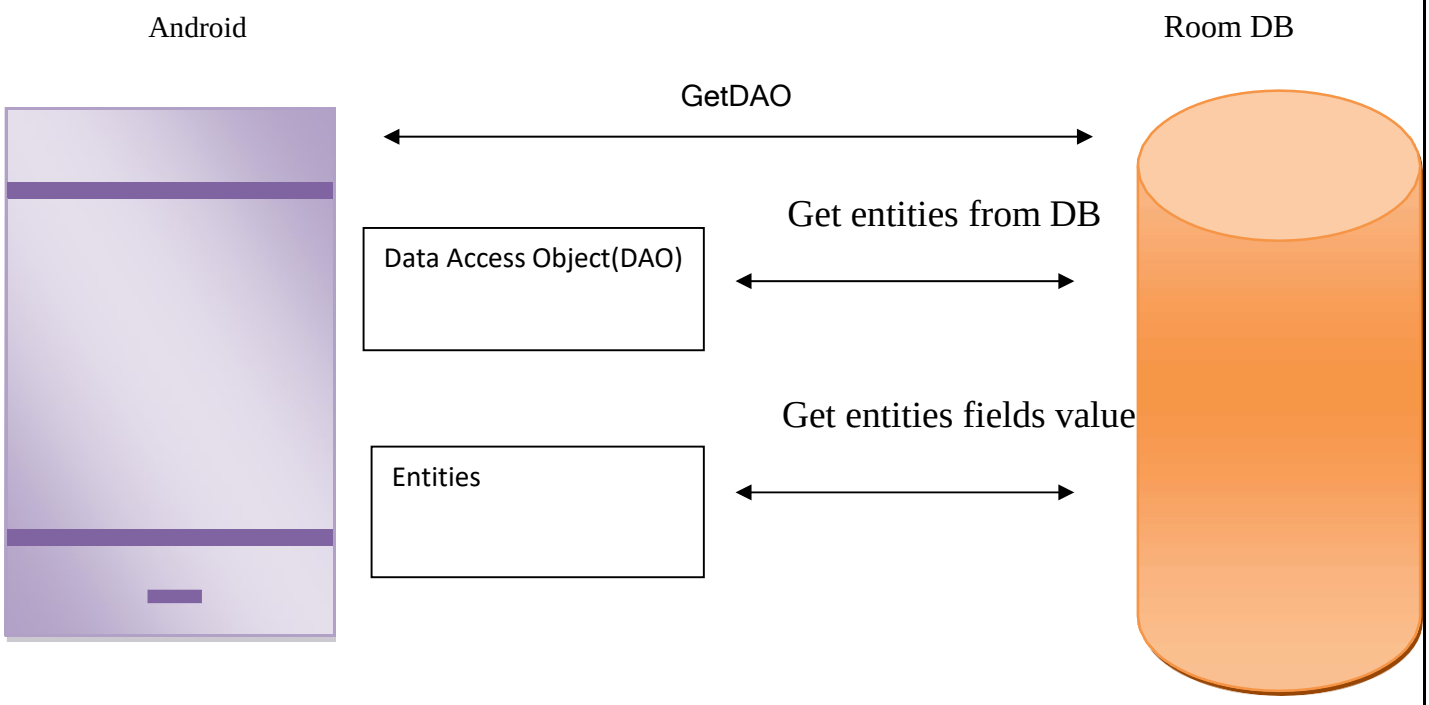
1.1 project overview

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:

- **User Registration**: Users create an account to sign up on the app.
- **Login Process**: After registration, users log in to access the app's features.
- **Main Dashboard**: Upon successful login, users are directed to the main dashboard.
- **Browsing and Ordering**: Users can explore available snack items, add their desired snacks to the cart, and place an order.
- **Admin Panel**: Admins can view and manage all the placed orders and oversee the app's operations.

Project Tasks:

1. **Environment Setup**:
   - Set up the development environment and tools required to build the app.
2. **Creating the Project**:
   - Start a new Android project in Android Studio.
3. **Adding Libraries and Dependencies**:
   - Include necessary libraries and dependencies for app functionality (such as Jetpack Compose and database libraries).
4. **Database Design**:
   - Define the database schema, models, and classes needed to store and manage data effectively.
5. **Building User Interface and Database Integration**:
   - Design the user interface using Jetpack Compose and integrate it with the database to ensure dynamic content loading.
6. **Configuring AndroidManifest.xml**:
   - Modify the AndroidManifest.xml file to configure the app components, activities, and necessary permissions.
7. **Testing the Application**:
   - Run the app on an emulator or real device, thoroughly testing all features to ensure everything functions properly.

Project Purpose:

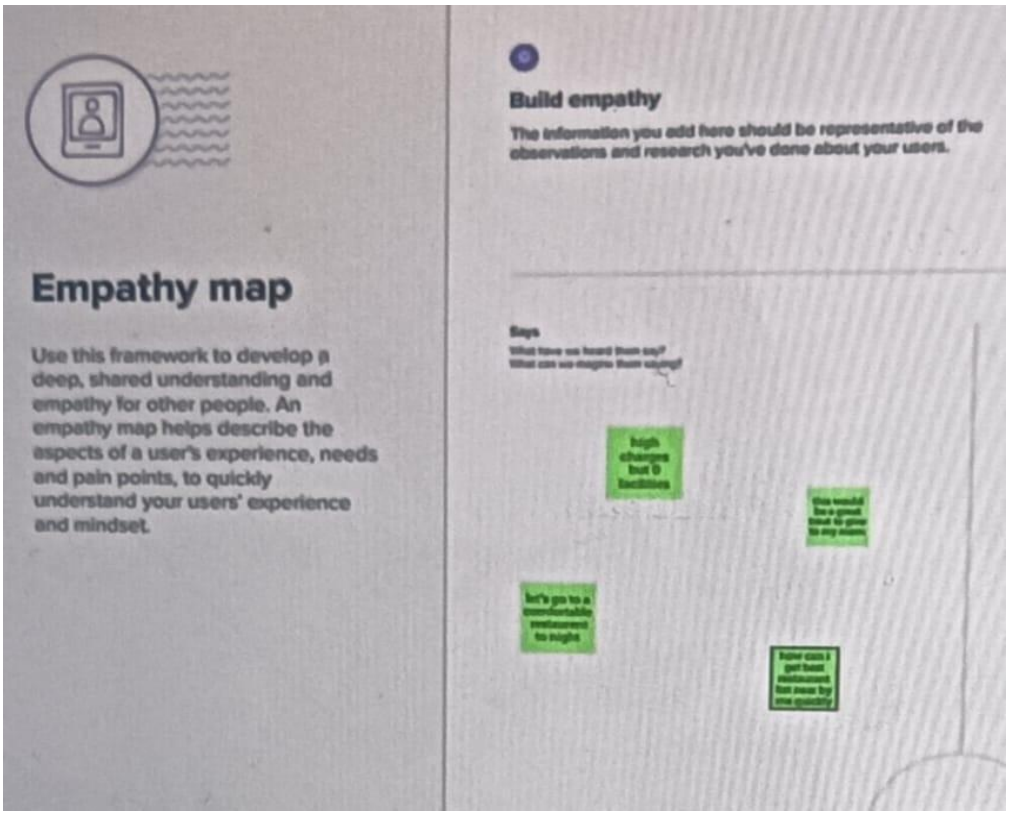
The **Snack Squad** app aims to streamline snack ordering and delivery, offering a seamless, user-friendly platform for users to purchase their favorite snacks and have them delivered to any location. The app is designed to provide easy access to a variety of snacks and allows users to customize their orders based on preferences, dietary restrictions, or specific needs.

Snack Squad is perfect for individual users or groups (like offices or schools), featuring a wide selection of snacks, including healthy options. The app’s customization feature ensures users can tailor their snack choices, making the entire process of snack selection more convenient. The integrated delivery service brings snacks directly to the user’s door, eliminating the hassle of shopping or making

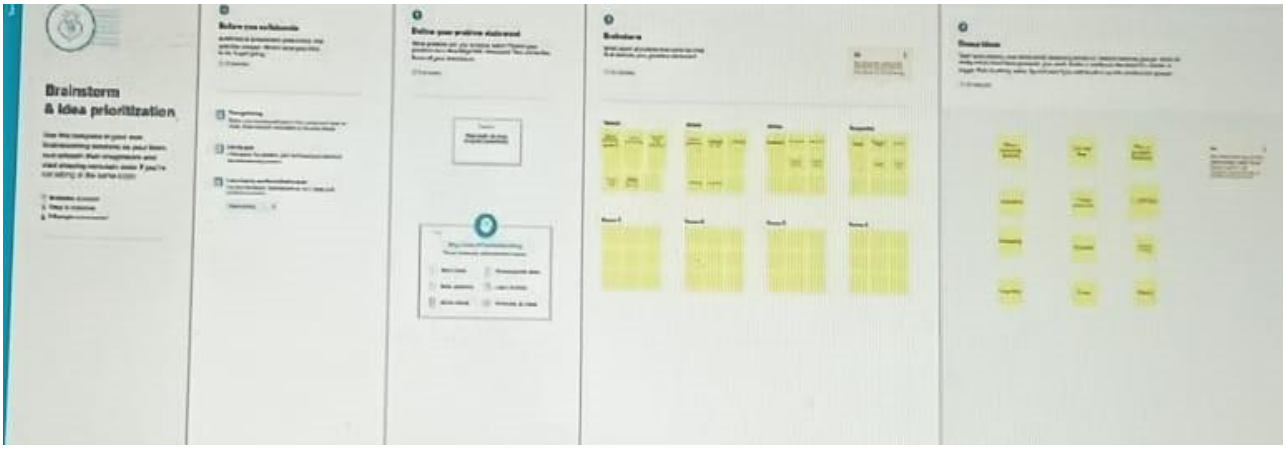
multiple trips. Snack Squad ensures a convenient, time-saving solution for anyone looking to enjoy snacks without the effort.

2.Problem Definition and Design Thinking:

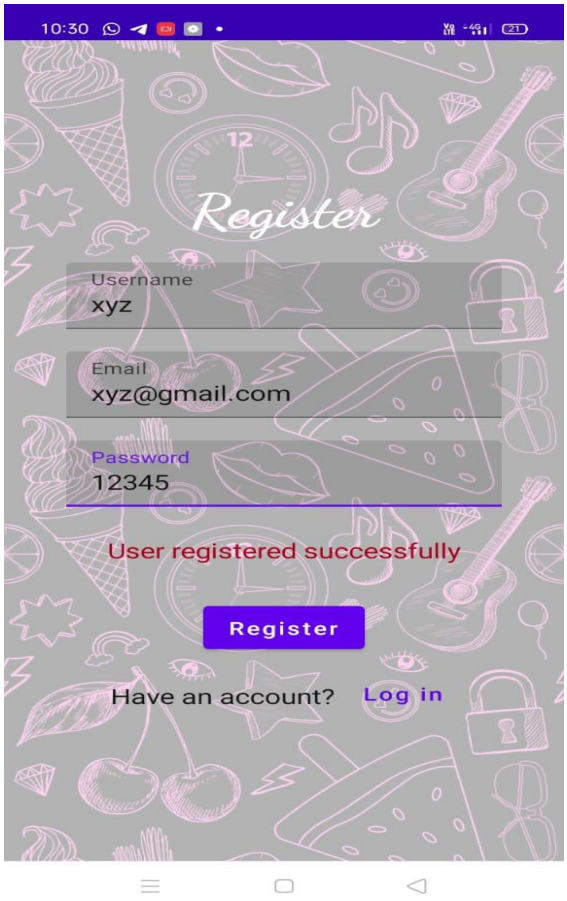
2.1. Empathymap:

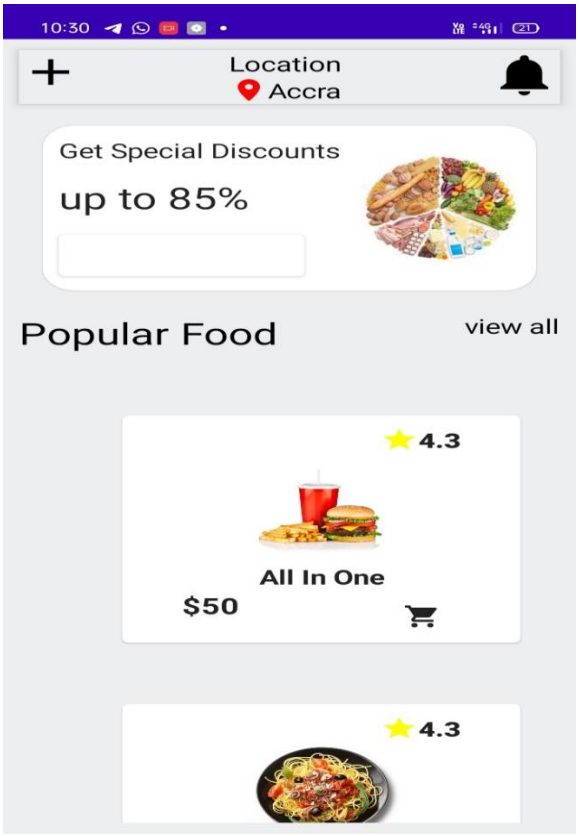


2.2 Ideation&Brainstroming map

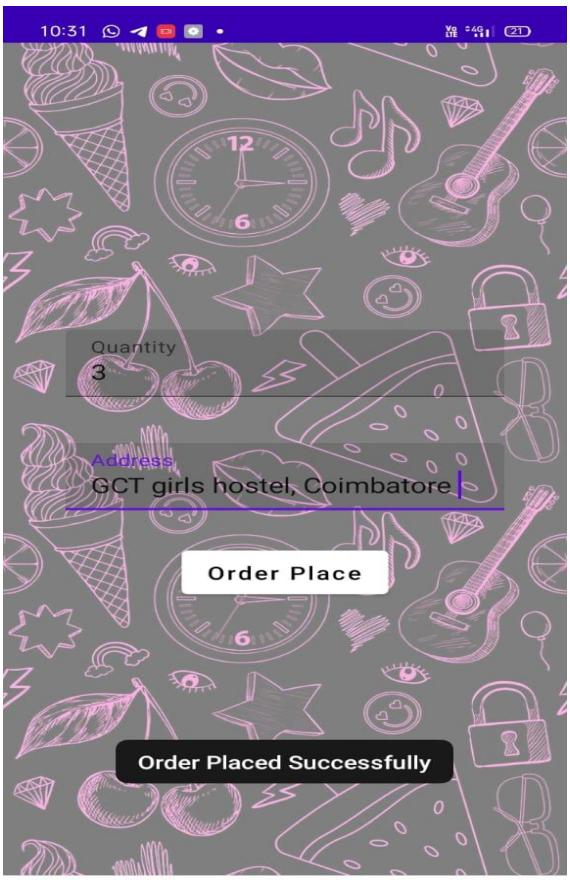


2.3 Result


10:30
Register
Username
xyz
Email
xyz@gmail.com
Password
12345
User registered successfully
Register
Have an account?
Log in


10:30
Location
Accra
Get Special Discounts
up to 85%
Popular Food
view all
4.3
All In One
$50
4.3


10:30
Login
Username
xyz
Password
12345
Login
Sign up
Forget password?


10:31
Quantity
3
Address
GCT girls hostel, Coimbatore
Order Place
Order Placed Successfully

Advantages:

1. **Improved User Experience**: A customizable snack ordering app enhances the customer experience by providing an intuitive, easy-to-navigate interface, smooth order placement, and seamless delivery. This helps create a pleasant, hassle-free experience for users.
2. **Broader Market Reach**: The app allows businesses to reach a wider customer base, including those who may not visit physical stores. With the ability to order from various locations, businesses can tap into new markets and attract customers who prefer online shopping.
3. **Customer Loyalty & Retention**: By integrating features such as personalized discounts, loyalty rewards, and frequent user incentives, the app encourages repeat usage. These features foster customer loyalty, keeping users coming back and increasing retention rates.
4. **Live Order Tracking**: Customers can track their orders in real-time, which gives them greater control and transparency over the delivery process, while ensuring they are informed at every stage of the journey.
5. **Operational Efficiency**: The app streamlines the ordering process, minimizes human error, and helps manage inventory more effectively. By tracking customer preferences and popular items, the app can assist businesses in predicting demand and optimizing their operations.

Disadvantages:

1. **High Development and Maintenance Costs**: Building and maintaining a feature-rich app with customization options and advanced capabilities (like real-time tracking or personalized recommendations) can be expensive. Regular updates and bug fixes further increase the ongoing cost.
2. **Reliance on Technology**: The app's reliance on smartphones and internet connectivity may exclude users who lack access to these technologies. It may also be less accessible for individuals who are not tech-savvy or prefer in-person interactions.
3. **Challenges in Delivery**: Delivery reliability is critical for the app's success. Late, incorrect, or damaged deliveries can lead to customer dissatisfaction, negative reviews, and damage to the business's reputation.
4. **Data Privacy Concerns**: The app collects sensitive user information, such as payment details and delivery addresses. This raises concerns over data security, requiring businesses to invest in robust security systems and comply with data protection regulations.

5. **Reduced Personal Interaction**: While the app offers convenience, it removes the personal interaction that some customers enjoy when shopping in-store. This lack of human engagement might lead to a decrease in brand loyalty and customer connections.

Applications:

1. **Grocery Stores**: Grocery stores can use the app to offer a wide range of snacks alongside their regular inventory, allowing customers to place orders for snacks and have them delivered with their other grocery items.
2. **Cinemas**: Movie theaters can integrate the app to enable customers to pre-order snacks such as popcorn, candy, and drinks for delivery to their seats or for easy pickup before the movie starts.
3. **Corporate Offices**: Businesses can utilize the app to offer snacks or drinks for office meetings, breaks, or events. The app can also support group orders, making it easier for employees to order together.
4. **Event Catering**: Catering services for events, parties, or conferences can use the app to offer a range of snacks, simplifying the ordering and delivery process for organizers while ensuring timely delivery.
5. **Convenience Stores**: Local convenience stores can use the app to provide quick snack deliveries, allowing customers to order snacks for immediate or scheduled delivery, catering to those who prefer not to leave home.

Future Prospects:

The future of customizable snack ordering and delivery apps is promising, driven by emerging technologies and the growing demand for personalized services. Here are some potential areas for growth:

1. **Advanced Personalization with AI**: Using AI and data analytics, the app can better understand customer preferences, recommend snack combinations, and offer personalized discounts based on past purchases, which can increase user engagement and sales.
2. **Voice-Activated Ordering**: Integrating with voice assistants like Alexa, Siri, or Google Assistant can allow customers to place orders hands-free, making the process even more convenient, especially when multitasking.

3. **Subscription Models**: A subscription-based approach could provide customers with regular deliveries of their favorite snacks, offering convenience and cost savings. Customizable subscription plans could cater to different tastes, dietary preferences, and snack quantities.
4. **Eco-Friendly Practices**: As sustainability becomes increasingly important to consumers, future app versions could focus on reducing environmental impact through eco-friendly packaging, offering organic or locally-sourced snack options, and providing carbon-neutral delivery choices.
5. **Health-Conscious Partnerships**: Partnering with health and wellness brands could expand the app's snack offerings to include nutritious, organic, or diet-specific snacks, such as gluten-free, keto, or vegan options, catering to health-conscious consumers.
6. **Global Expansion and Local Adaptation**: Expanding the app's reach to international markets and customizing offerings based on regional preferences, dietary habits, and local flavors can significantly increase its global presence.

Conclusion:

In conclusion, a customizable snack ordering and delivery app offers a unique solution to the growing demand for convenience and personalization in snack consumption. It not only provides users with an easy way to access a wide variety of snacks but also enables them to customize orders according to their preferences. The app can help businesses streamline operations, improve customer satisfaction, and increase sales through advanced features like real-time tracking, personalized recommendations, and loyalty programs.

By simplifying the ordering process and offering tailored experiences, the app can become a valuable tool for both customers and businesses. For consumers, it offers a quick and easy way to satisfy snack cravings, while for businesses, it provides greater market reach and operational efficiency.

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
importandroidx.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' colorfrom the theme
Surface( modifier = Modifier.fillMaxSize(), color = MaterialTheme.colors.background)
 {

FinalView(this)

val context = LocalContext.current

//PopularFoodColumn(context)

}

}
```

```
}

}

}
@Composablefun
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 = "shoppingcart", )
}
}
}
```

```
}
}
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 withParentheses
}
}
@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.currentApp(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)
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)
}
```