Android Application Development

A Customizable Snack Ordering And Delivery App

Team ID -NM2024TMID05837

Submitted By

Akhil Karthikeyan R(Team Leader)- 640BAC43C62DF818D78484B3207A2D3B

Gowtham S(Team Member)- C75562B9B55F63C86214396ED6166148

Mohamed Aslam S(Team Member)- FABE545CCFA3A2595484A45CB9F5C231

Udhaya Prakash P(Team Member)- BB08FDA4DFA492FA12A4135CC7CB38B1

Department of Computer Science

Anna University Regional Campus Coimbatore(Code-7100)

Coimbatore-641046

DEPARTMENT OF COMPUTER SCIENCE

ANNA UNIVERSITY REGIONAL CAMPUS COIMBATORE

COIMBATORE-641046

TABLE OF CONTENTS

INTRODUCTION

1.1 OBJECTIVE

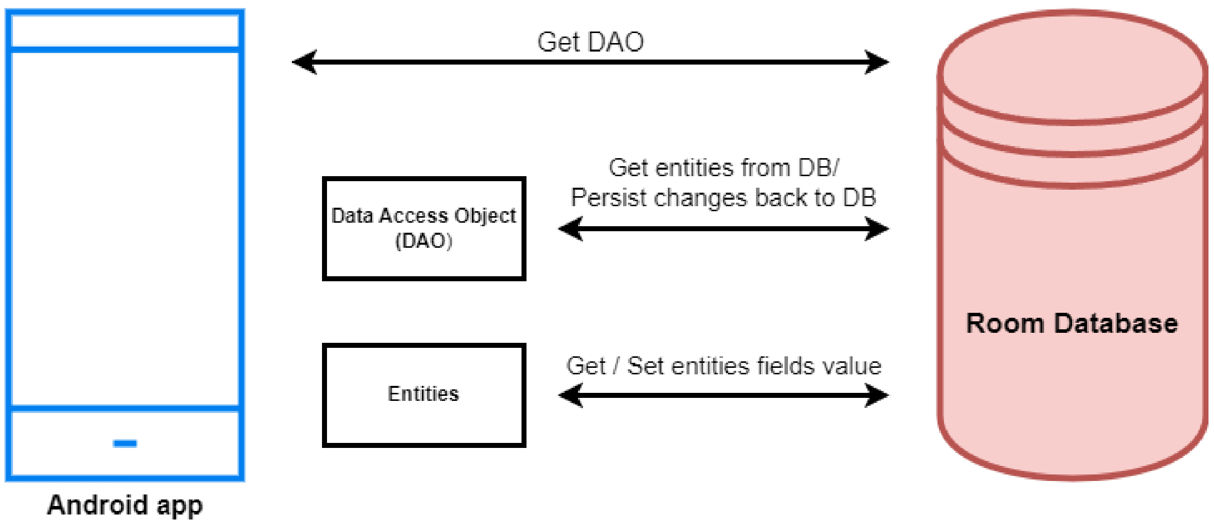
In order to give consumers a comfortable and easy way to purchase snacks via a mobile platform, this project demonstrates the development of a snack delivery application. With a simple functioning architecture that enables customers to peruse the snack options, add products to a cart, and place purchases, the project is now in its early phases. This basic model lays the framework for future development and feature growth while showcasing crucial capabilities.

This early-stage application's main objective is to provide a smooth user experience that satisfies the rising need for easy and quick access to snacks. Users will find it easy to navigate and place orders thanks to the app's simple interface design. A simple snack catalog, order placement functionality, and an order summary display are among the initial features. Enhancing user involvement through the integration of safe payment methods, real-time order tracking, and a customized recommendation system will be the main emphasis of future initiatives.

This early version offers a proof of concept and permits testing of the fundamental app functionality by focusing on a small and manageable collection of core features. The team will be able to tweak the app's interface, add more snacks to the inventory, and add more functionality in later iterations thanks to the feedback from these early tests. Even though it is still in its infancy, this snack delivery service shows promise in meeting the convenience-driven demands of contemporary customers looking for quick and simple snack options.

FUNCTIONALITIES

2.1 ARCHITECTURE


Get DAO
Data Access Object (DAO)
Get entities from DB/ Persist changes back to DB
Entities
Get / Set entities fields value
Room Database
Android app

2.2 PROJECT WORKFLOW

- **Research & Planning:** Validate the idea with market research, define features, and set requirements.

- **Design:** Create a user-friendly interface with simple navigation.

- **Development:** Build the frontend and backend, integrating core features like browsing, ordering, and payments.

- **Testing & Feedback:** Test functionality, gather user feedback, and make improvements.

- **Launch & Improve:** Launch the app, monitor feedback, and plan future updates.

3.1 PROBLEM STATEMENT

Traditional food delivery services, which frequently concentrate on whole meals rather than meeting the specialized demands of clients searching for quick, accessible snacks, are unable to sufficiently meet the growing need for quick, convenient, and dependable snack options. Customers usually encounter issues including lengthy wait times, a small selection of snacks, and a lack of personalization on current platforms. This necessitates the creation of a specialized snack delivery service that offers a large selection of snacks with quick delivery times and an intuitive mobile app. By providing users with customisable, on-demand snack alternatives, the app seeks to close this gap and meet their need for speed and convenience when eating.

3.2 SOLUTION

A mobile-based snack delivery app that is tailored to the unique requirements of users searching for quick and easy snack options is the suggested solution to the issue of limited access to handy and rapid snack delivery. With only a few taps, users will be able to place orders and peruse a wide variety of snacks, from healthy selections to decadent indulgences, thanks to the app's user-friendly interface. In order to guarantee that customers receive their snacks on time, the app's primary features will include an effective search function, configurable snack options, and real-time delivery tracking. Additionally, the app will offer a variety of safe payment methods, increasing consumer ease and trust.

SYSTEM REQUIREMENTS

4.1 HARDWARE REQUIREMENTS

1. **Development Device (Computer/Laptop):**
   - **Processor**: Intel i5 (8th generation or higher) or AMD Ryzen 5 equivalent
   - **RAM**: 8 GB minimum (16 GB recommended for smoother performance)
   - **Storage**: 100 GB of available storage space
   - **Graphics**: Integrated graphics (discrete GPU optional for emulator acceleration)

2. **Mobile Device for Testing (Optional)**:
   - **Operating System**: Android 8.0 (Oreo) or higher
   - **RAM**: 2 GB minimum
   - **Storage**: At least 50 MB of free storage space for the app

4.2 SOFTWARE REQUIREMENTS

1. **Operating System** (for development):
   - Windows 10 or 11 (64-bit), macOS (Big Sur or later), or Linux (Ubuntu 20.04 or later)
2. **Development Tools**:
   - **Android Studio**: Version 4.2 or higher
   - **JDK**: Java Development Kit 11 or higher
   - **Android SDK**: Android SDK API Level 26 or higher
3. **Libraries and Frameworks**:
   - **Android Jetpack Compose**: For building UI components
   - **Room Database**: For local data storage (optional but recommended for sleep data persistence)
   - **Kotlin Coroutines**: For managing asynchronous tasks smoothly
   - **Dagger Hilt**: For dependency injection (optional, for larger app projects)
4. **Database**:

    - **SQLite or Room Database**: For offline data storage and sleep record persistence
    - **Shared Preferences**: For lightweight data (e.g., user settings, last session data)
5. **Additional Tools (Optional)**:
    - **Firebase**: For remote data storage, analytics, and crash reporting (optional)
    - **Git**: For version control and collaboration, with a GitHub or GitLab repository for project storage
    - **Postman**: For testing any APIs (if integrated with other health services)

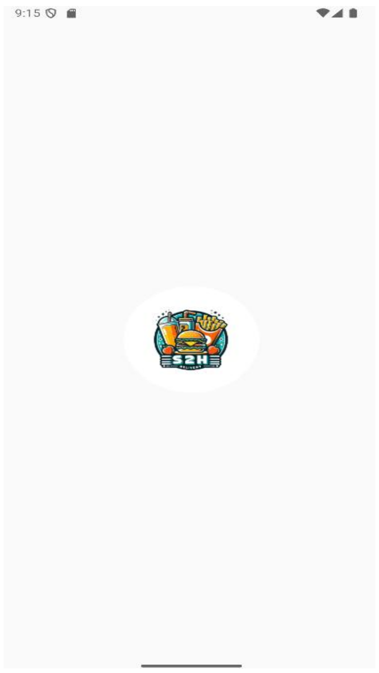
4.3 TESTING REQUIREMENTS

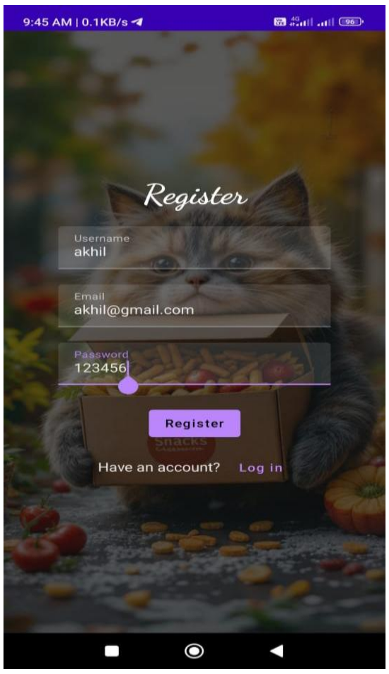
1. **Android Emulator**: Configured in Android Studio with API Level 26 or higher
2. **Physical Android Device** (optional): For real-world testing, ideally with Android 8.0 (Oreo) or higher

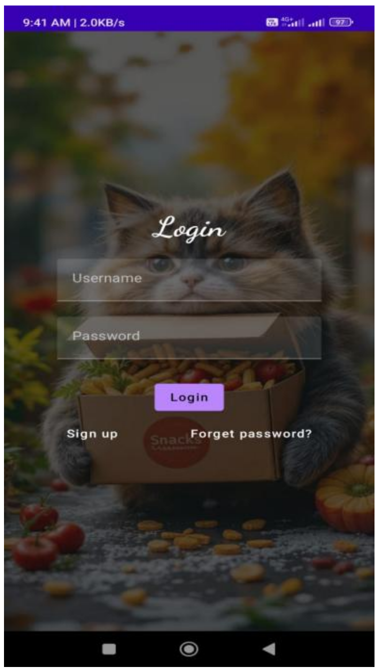
4.4 NETWORK REQUIREMENTS (Optional)

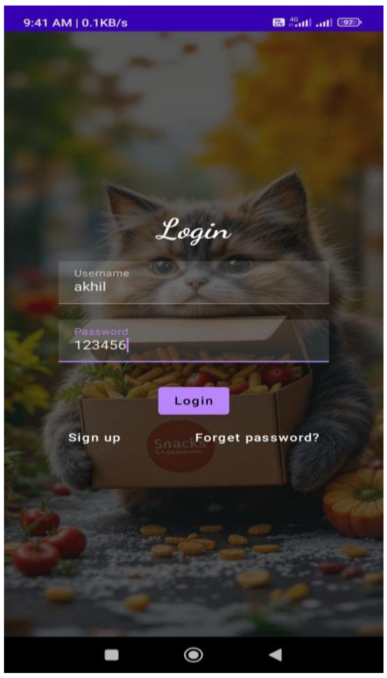
- **Internet Connection**: Required only for features such as updates, analytics, or remote data storage if using Firebase or other cloud services.

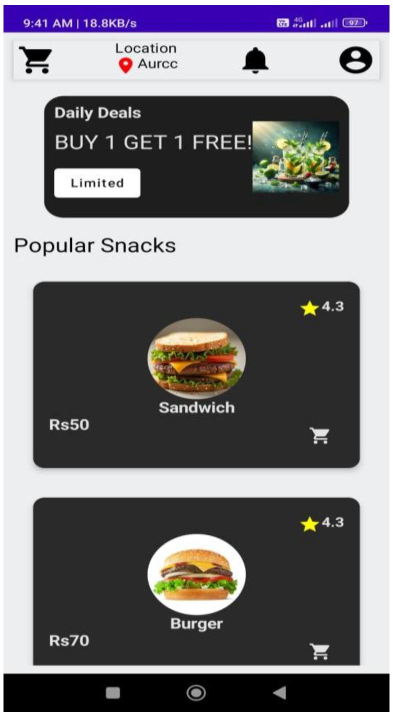
APPLICATION SCREENSHOTS

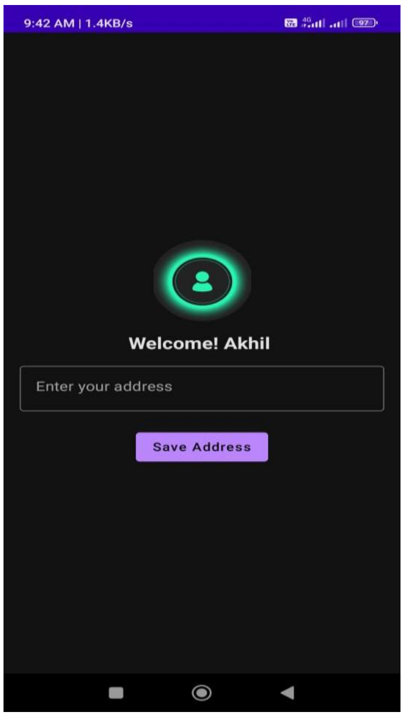


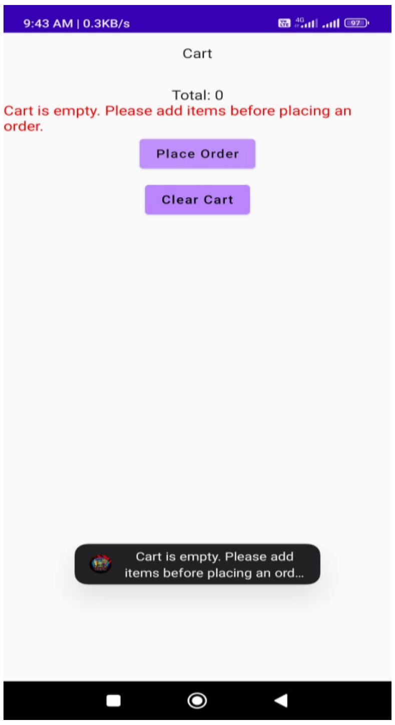






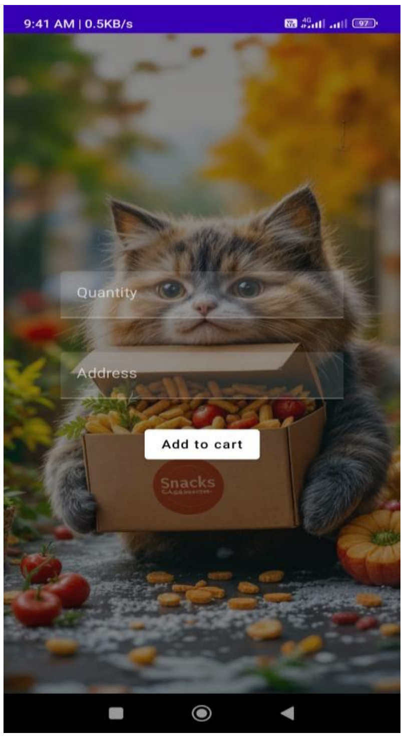


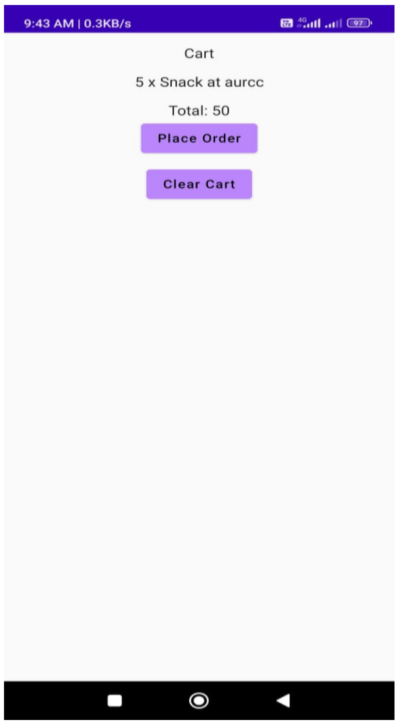




Cart is empty. Please add items before placing an order.

Cart is empty. Please add items before placing an ord...

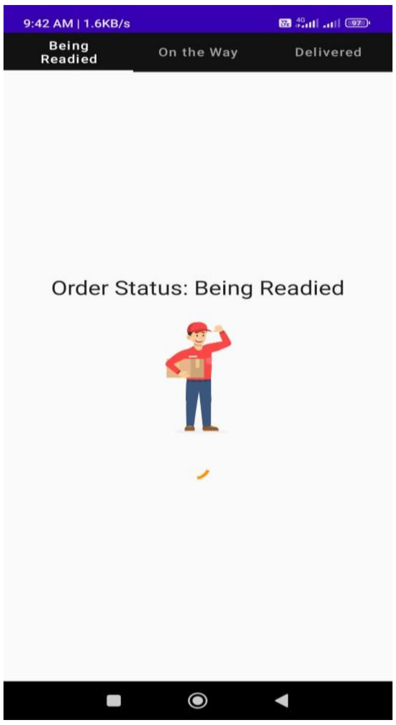



9:43 AM | 0.3KB/s

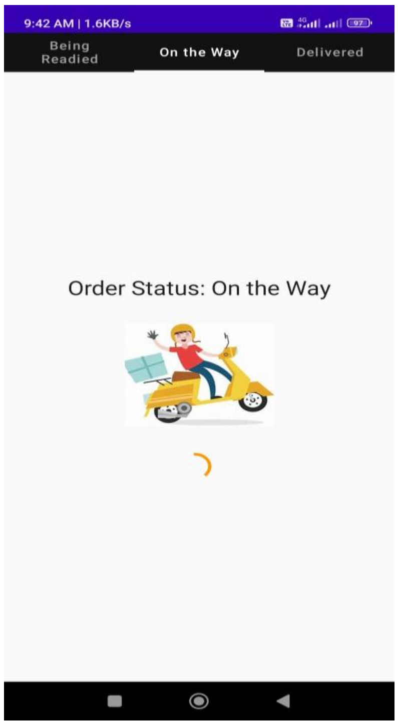
Cart

5 x Snack at aurcc

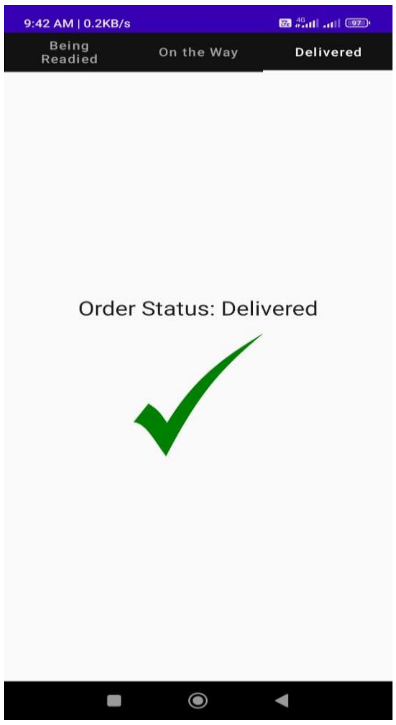
Total: 50


9:42 AM | 1.6KB/s

Order Status: Being Readied


9:42 AM | 1.6KB/s

Order Status: On the Way


9:42 AM | 0.2KB/s

Order Status: Delivered

TESTING

- **Username Validation**

  The user should register in the application with a username and email id before signing in to the application. The application will validate whether the provided username and password matches with the previously registered entries. The user can only sign in to the application only if they enter the valid username and password.

- **Cart Validation**

The user can add the snack into the cart that they want to order. After adding the items they can order the items by clicking the place order button in the cart section. It will only work if there is at least one item in the cart. If the cart is empty there pop a error message "The Cart Can't be Empty".

ACTIVITIES

LOGINACTIVITY.kt

The `LoginActivity.kt` handles user login authentication, verifying credentials, and initiating the login process. Once authenticated, it passes the username to `ProfileDashboardActivity` and then starts the `MainPage`.

MAINPAGE.kt

The `MainPage.kt` file defines the main layout and UI components for the snack ordering app, including a top navigation bar for Cart, Order Status, Profile, and a list of popular food items. It sets up the main screen layout with interactive elements that lead to different app activities, like viewing the cart and checking order statuses.

ADMINACTIVITY.kt

`AdminActivity.kt` manages administrative functions, allowing the admin to oversee user accounts, orders, and product listings. It provides controls for viewing, updating, or removing items within these categories.

CARTACTIVITY.kt

`CartActivity.kt` handles the user's shopping cart, displaying items selected for purchase along with their quantities and total price. It provides options to adjust item quantities, remove items, and proceed to place the order.

ORDERSTATUSACTIVITY.kt

The `OrderStatusActivity.kt` displays the current status of the user's order a dedicated screen. It updates and shows the order's progress through different stages such as "Being Readied," "On the Way," and "Delivered," providing real-time updates to the user.

ORDERDISPLAYSTATUSACTIVITY.kt

`OrderDisplayStatus.kt` shows the order status in a tabbed format, updating the progress at various stages. It tracks and displays stages like "Being Readied," "On the Way," and "Delivered."

PROFILEDASHBOARDACTIVITY.kt

`ProfileDashboardActivity.kt` displays the user's profile, including their profile image, username, and email. It receives the username from `LoginActivity` and presents the user information in a structured format.

REGISTERACTIVITY.kt

`RegisterActivity.kt` handles the user registration process, allowing users to create a new account by entering their details like username, email, and password. After successful registration, it typically navigates to the `LoginActivity` for the user to log in.

TARGETACTIVITY.kt

`TargetActivity.kt` likely refers to an activity in the app that is designed for a specific purpose, such as handling a particular task or displaying certain content. It could be a dedicated screen for showing details, managing tasks, or interacting with specific features within the app, depending on the app's flow and design

Applications

- **Convenience:** Customers can save time and effort by ordering snacks online and having them delivered right to their door.
- **Variety:** To accommodate a diversity of tastes and preferences, the app provides a large selection of snack options, from nutritious to decadent sweets.
- **User-Friendly:** The app's straightforward design guarantees that all users can easily navigate its intuitive layout.
- **Time-saving**: The app makes it possible to order snacks quickly, which is perfect for people with hectic schedules who require prompt fixes.
- **Personalized Recommendations:** To enhance the user experience, users are given snack recommendations based on their past orders and preferences.
- **Real-Time Order Tracking:** Customers may monitor their orders in real time, guaranteeing dependability and transparency.
- **Secure Payments:** A variety of safe and flexible payment options are available, such as credit cards and digital wallets.
- **Rewards and Promotions:** To promote recurring use and client retention, the app offers loyalty plans and discounts.
- **Scalability:** By adding more features, food alternatives, and cutting-edge technology, the app may develop.

- **Niche Focus:** The app caters to a particular market demand that isn't entirely met by larger food platforms by concentrating only on snack delivery.

CONCLUSION

An important initial step in meeting the rising need for quick and easy snack options is the creation of the snack delivery app's fundamental operational concept. The project is still in its early phases, even though we have successfully added essential functions like order placement, cart management, and snack browsing. The challenges encountered in the early stages of development, like creating a user-friendly interface, establishing a productive database, and incorporating simple payment features, have given important insights into the intricacies of app development. Notwithstanding these difficulties, the working model ensures that the app can eventually exceed user expectations by providing the groundwork for future development and improvement.

The success of this initial model shows that the app has the potential to provide a streamlined, on-demand snack delivery service that meets the needs of busy people. By utilizing user feedback and improving key features, we hope to improve the app's functionality and get it ready for wider use. The next stages will concentrate on expanding the app's capabilities, optimizing the user experience, and getting ready for real-world implementation.

Future Scope

The snack delivery app's future scope includes a number of exciting opportunities for growth and enhancement. Working with delivery partners to create a dependable and effective delivery network—which will entail integrating third-party logistics and streamlining delivery routes to guarantee timely service—will be a top priority. Additionally, broadening the snack catalog to include a greater range of products, such as healthier options, regional favourites, and popular snacks, will further appeal to a larger customer base.

Advanced personalization capabilities like snack recommendations based on user preferences, dietary restrictions, and order history will also be included in the app. Transactions will be more streamlined and dependable if the payment gateway is improved to incorporate several safe options, such UPI and digital wallets.

Additionally, by providing transparency and usability, incorporating real-time order tracking and order history tools would enhance the overall customer experience. The app's growth in a cutthroat industry will be fueled by the introduction of loyalty programs and special offers, which will aid in client retention and promote repeat business.

Finally, the app's backend will need to be expanded to accommodate higher traffic and order volumes as the user base expands. This will entail putting policies in place for effective performance at scale, protecting data, and optimizing server infrastructure.

The snack delivery app will develop into a complete solution that satisfies the demands of contemporary consumers by addressing these issues and keeping track of user input, providing them with a quick and customized snack ordering experience.

APPENDIX

AndroidManifest.xml:

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

    <application
        android:allowBackup="true"
        android:dataExtractionRules="@xml/data_extraction_rules"
        android:fullBackupContent="@xml/backup_rules"
        android:icon="@drawable/fast_food"
        android:label="@string/app_name"
        android:supportsRtl="true"
        android:theme="@style/Theme.SnackOrdering"
        tools:targetApi="31">
        <activity
            android:name=".AdminActivity"
            android:exported="false"
            android:label="@string/title_activity_admin"
            android:theme="@style/Theme.SnackOrdering" />
        <activity
            android:name=".LoginActivity"
            android:exported="true"
            android:label="SnackSquad"
            android:theme="@style/Theme.SnackOrdering">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER"
/>
            </intent-filter>
        </activity>
        <activity android:name=".CartActivity" />
        <activity android:name=".OrderStatusActivity" />
        <activity android:name=".ProfileDashboardActivity" />
        <activity android:name=".OrderStatusDisplayActivity"/>
        <activity
            android:name=".TargetActivity"
            android:exported="false"
            android:label="@string/title_activity_target"
            android:theme="@style/Theme.SnackOrdering" />
        <activity
            android:name=".MainPage"
            android:exported="false"
            android:label="@string/title_activity_main_page"
            android:theme="@style/Theme.SnackOrdering" />
        <activity
            android:name=".MainActivity"
            android:exported="false"
            android:label="MainActivity"
            android:theme="@style/Theme.SnackOrdering" />
    </application>

</manifest>
```

AdminActivity.kt:

```
package com.example.snackordering

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

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("akhil" ,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}")
                    }
                }
            }
        }

    }
}
```

CartActivity.kt:

```
package com.example.snackordering

import android.content.Context
import android.content.Intent
import android.os.Bundle
import android.widget.Toast
import androidx.activity.ComponentActivity
import androidx.activity.compose.setContent
import androidx.compose.foundation.layout.*
import androidx.compose.foundation.lazy.LazyColumn
import androidx.compose.foundation.lazy.items
import androidx.compose.material.Button
import androidx.compose.material.Text
import androidx.compose.runtime.*
import androidx.compose.ui.Alignment
import androidx.compose.ui.Modifier
import androidx.compose.ui.unit.dp
import com.example.snackordering.ui.theme.SnackOrderingTheme

class CartActivity : ComponentActivity() {
   private lateinit var orderDatabaseHelper: OrderDatabaseHelper

   override fun onCreate(savedInstanceState: Bundle?) {
      super.onCreate(savedInstanceState)
      orderDatabaseHelper = OrderDatabaseHelper(this)

      setContent {
         SnackOrderingTheme {
            CartScreen(this, orderDatabaseHelper)
         }
      }
   }
}

@Composable
fun CartScreen(context: Context, orderDatabaseHelper:
OrderDatabaseHelper) {
   var orders by remember {
mutableStateOf(orderDatabaseHelper.getAllOrders()) }
   var totalAmount by remember { mutableStateOf(orders.sumOf {
it.quantity!!.toInt() * 10 }) } // Assuming each snack costs 10 units
   var showError by remember { mutableStateOf(false) }

   Column(modifier = Modifier.fillMaxSize(), horizontalAlignment =
Alignment.CenterHorizontally) {
      Text(text = "Cart", modifier = Modifier.padding(16.dp))

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

```
            Text(text = "${order.quantity} x Snack at ${order.address}")
         }
      }

      Spacer(modifier = Modifier.height(16.dp))

      Text(text = "Total: $totalAmount")

      if (showError) {
         Text(text = "Cart is empty. Please add items before placing an
order.", color = androidx.compose.ui.graphics.Color.Red)
      }

      Button(onClick = {
         if (orders.isEmpty()) {
            showError = true
            Toast.makeText(context, "Cart is empty. Please add items before
placing an order.", Toast.LENGTH_SHORT).show()
         } else {
            showError = false
            val intent = Intent(context, OrderStatusActivity::class.java)
            context.startActivity(intent)
         }
      }) {
         Text("Place Order")
      }

      Spacer(modifier = Modifier.height(8.dp))

      // Button to clear the orders
      Button(onClick = {
         orderDatabaseHelper.clearAllOrders()  // Clears all orders in the
database
         orders = emptyList()                 // Update orders list to empty
         totalAmount = 0                      // Reset total amount
      }) {
         Text("Clear Cart")
```

```
        }
    }
}
```

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 logged in"

                // Send username to ProfileDashboardActivity
```

```
                    // Start MainPage
                    val mainPageIntent = Intent(context, MainPage::class.java)
                    context.startActivity(mainPageIntent)

                } else if (user != null && user.password == "admin") {
                    error = "Successfully logged in as admin"
                    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)
}
```

MainPage.kt:

```
package com.example.snackordering

import android.annotation.SuppressLint
import android.content.Context
import android.content.Intent
import android.os.Bundle
import androidx.activity.ComponentActivity
import androidx.activity.compose.setContent
import androidx.annotation.DrawableRes
import androidx.compose.foundation.Image
import androidx.compose.foundation.background
import androidx.compose.foundation.layout.*
import androidx.compose.foundation.lazy.LazyColumn
import androidx.compose.foundation.lazy.items
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.ui.layout.ContentScale
import androidx.compose.ui.platform.LocalContext
import androidx.compose.ui.res.painterResource
import androidx.compose.ui.text.font.FontWeight
import androidx.compose.ui.unit.dp
import androidx.compose.ui.unit.sp
import com.example.snackordering.ui.theme.SnackOrderingTheme

class MainPage : ComponentActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContent {
            SnackOrderingTheme {
                Surface(
                    modifier = Modifier.fillMaxSize(),
                    color = MaterialTheme.colors.background
                ) {
                    FinalView(this)
                }
            }
        }
    }
}

@Composable
fun TopPart(context: Context) {
    Row(
        modifier = Modifier
            .fillMaxWidth()
            .background(Color(0xffeceef0)),
        horizontalArrangement = Arrangement.SpaceBetween
    ) {
        IconButton(onClick = {
            context.startActivity(Intent(context, CartActivity::class.java))
        }) {
            Icon(
                imageVector = Icons.Default.ShoppingCart,
                contentDescription = "Cart",
                tint = Color.Black,
                modifier = Modifier.size(40.dp)
            )
        }

        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 = "Aurcc", color = Color.Black)
            }
        }

        IconButton(onClick = {
            context.startActivity(Intent(context,
OrderStatusDisplayActivity::class.java))
        }) {
            Icon(
                imageVector = Icons.Default.Notifications,
                contentDescription = "Order Status",
                tint = Color.Black,
                modifier = Modifier.size(40.dp)
            )
        }
        IconButton(onClick = {
            context.startActivity(Intent(context,
ProfileDashboardActivity::class.java))
        }) {
            Icon(
                imageVector = Icons.Default.AccountCircle,
                contentDescription = "Profile",
                tint = Color.Black,
                modifier = Modifier.size(40.dp)
            )
        }
```

```
    }
}

@Composable
fun CardPart() {
    Card(modifier = Modifier.size(width = 310.dp, height = 150.dp), shape
= RoundedCornerShape(20.dp)) {
        Row(modifier = Modifier.padding(10.dp), horizontalArrangement =
Arrangement.SpaceBetween) {
            Column(verticalArrangement = Arrangement.spacedBy(12.dp)) {
                Text(text = "Daily Deals", fontWeight = FontWeight.Bold,
fontSize = 18.sp)
                Text(text = "BUY 1 GET 1 FREE!", style =
MaterialTheme.typography.h5)
                Button(onClick = {}, colors =
ButtonDefaults.buttonColors(Color.White)) {
                    Text(text = "Limited", color = MaterialTheme.colors.surface)
                }
            }
            Image(
                painter = painterResource(id = R.drawable.food_tip_im),
                contentDescription = "Food Image",
                modifier = Modifier.size(width = 100.dp, height = 200.dp)
            )
        }
    }
}

@Composable
fun PopularFood(
    @DrawableRes drawable: Int,
    name: String,
    price: Int,
    context: Context
) {
    Card(
        modifier = Modifier
```

```
            .padding(vertical = 10.dp, horizontal = 20.dp)
            .fillMaxWidth(),
        shape = RoundedCornerShape(12.dp),
        elevation = 5.dp
    ) {
        Column(
            modifier = Modifier.padding(16.dp),
            verticalArrangement = Arrangement.Center,
            horizontalAlignment = Alignment.CenterHorizontally
        ) {
            Spacer(modifier = Modifier.height(5.dp))
            Row(
                modifier = Modifier
                    .fillMaxWidth(),
                horizontalArrangement = Arrangement.End
            ) {
                Icon(
                    imageVector = Icons.Default.Star,
                    contentDescription = "Rating",
                    tint = Color.Yellow
                )
                Text(text = "4.3", fontWeight = FontWeight.Bold)
            }
            Image(
                painter = painterResource(id = drawable),
                contentDescription = name,
                contentScale = ContentScale.Crop,
                modifier = Modifier
                    .size(100.dp)
                    .clip(CircleShape)
            )
            Text(text = name, fontWeight = FontWeight.Bold, fontSize =
18.sp)
            Row(modifier = Modifier.fillMaxWidth(), horizontalArrangement =
Arrangement.SpaceBetween) {
                Text(
                    text = "Rs$price",
                    style = MaterialTheme.typography.h6,
                    fontWeight = FontWeight.Bold,
                    fontSize = 18.sp
                )
                IconButton(onClick = {
                    val intent = Intent(context, TargetActivity::class.java)
                    context.startActivity(intent)
                }) {
                    Icon(
                        imageVector = Icons.Default.ShoppingCart,
                        contentDescription = "Add to Cart"
                    )
                }
            }
        }
    }
}

// Define each food item with image, name, and price
private val FoodList = listOf(
    SnackItem(R.drawable.sandwish, "Sandwich", 50),
    SnackItem(R.drawable.burger, "Burger", 70),
    SnackItem(R.drawable.pack, "Snack Pack", 80),
    SnackItem(R.drawable.pasta, "Pasta", 60),
    SnackItem(R.drawable.tequila, "Falooda", 40),
    SnackItem(R.drawable.wine, "Mojito", 55),
    SnackItem(R.drawable.salad, "Salad", 30),
    SnackItem(R.drawable.pop, "Popcorn", 25)
)

// Data class for each food item
private data class SnackItem(
    @DrawableRes val drawable: Int,
    val name: String,
    val price: 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(context)
      }
      Spacer(modifier = Modifier.padding(10.dp))
      CardPart()
      Spacer(modifier = Modifier.padding(10.dp))
      Row(modifier = Modifier.fillMaxWidth(), horizontalArrangement =
Arrangement.SpaceBetween) {
         Text(text = "Popular Snacks", style =
MaterialTheme.typography.h5, color = Color.Black)
         Text(text = "", style = MaterialTheme.typography.subtitle1, color =
Color.Black)
      }
      Spacer(modifier = Modifier.padding(10.dp))
      PopularFoodColumn(context)
   }
}

@Composable
fun PopularFoodColumn(context: Context) {
   LazyColumn(
      modifier = Modifier.fillMaxSize(),
      verticalArrangement = Arrangement.spacedBy(16.dp)
   ) {
      items(FoodList) { item ->
         PopularFood(
            drawable = item.drawable,
                name = item.name,
                price = item.price,
                context = context
            )
        }
    }
}

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

Order:

```
package com.example.snackordering

import androidx.room.ColumnInfo
import androidx.room.Entity
import androidx.room.PrimaryKey

@Entity(tableName = "order_table")
data class Order(
    @PrimaryKey(autoGenerate = true) val id: Int?,
    @ColumnInfo(name = "quantity") val quantity: String?,
    @ColumnInfo(name = "address") val address: String?,
)
```

OrderDao:

```
package com.example.snackordering

import androidx.room.*
```

```
@Dao
interface OrderDao {

    @Query("SELECT * FROM order_table WHERE  address= :address")
    suspend fun getOrderByAddress(address: String): Order?

    @Insert(onConflict = OnConflictStrategy.REPLACE)
    suspend fun insertOrder(order: Order)

    @Update
    suspend fun updateOrder(order: Order)

    @Delete
    suspend fun deleteOrder(order: Order)
}
```

OrderDatabase:

```
package com.example.snackordering

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

@Database(entities = [Order::class], version = 1)
abstract class OrderDatabase : RoomDatabase() {

    abstract fun orderDao(): OrderDao

    companion object {

        @Volatile
        private var instance: OrderDatabase? = null

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

OrderDatabaseHelper:

```
package com.example.snackordering

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

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

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

        private const val TABLE_NAME = "order_table"
        private const val COLUMN_ID = "id"
        private const val COLUMN_QUANTITY = "quantity"
        private const val COLUMN_ADDRESS = "address"
    }

    override fun onCreate(db: SQLiteDatabase?) {
        val createTable = "CREATE TABLE $TABLE_NAME (" +
                "$COLUMN_ID INTEGER PRIMARY KEY
AUTOINCREMENT, " +
                "$COLUMN_QUANTITY Text, " +
                "$COLUMN_ADDRESS TEXT " +
                ")"

        db?.execSQL(createTable)
    }

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

    fun insertOrder(order: Order) {
        val db = writableDatabase
        val values = ContentValues()
        values.put(COLUMN_QUANTITY, order.quantity)
        values.put(COLUMN_ADDRESS, order.address)
        db.insert(TABLE_NAME, null, values)
        db.close()
    }



    @SuppressLint("Range")
    fun getOrderByQuantity(quantity: String): Order? {
        val db = readableDatabase
        val cursor: Cursor = db.rawQuery("SELECT * FROM
$TABLE_NAME WHERE $COLUMN_QUANTITY = ?",
arrayOf(quantity))
        var order: Order? = null
        if (cursor.moveToFirst()) {
            order = Order(
                id = cursor.getInt(cursor.getColumnIndex(COLUMN_ID)),
            quantity =
cursor.getString(cursor.getColumnIndex(COLUMN_QUANTITY)),
            address =
cursor.getString(cursor.getColumnIndex(COLUMN_ADDRESS)),
        )
      }
      cursor.close()
      db.close()
      return order
   }
   @SuppressLint("Range")
   fun getOrderById(id: Int): Order? {
      val db = readableDatabase
      val cursor: Cursor = db.rawQuery("SELECT * FROM
$TABLE_NAME WHERE $COLUMN_ID = ?", arrayOf(id.toString()))
      var order: Order? = null
      if (cursor.moveToFirst()) {
         order = Order(
            id = cursor.getInt(cursor.getColumnIndex(COLUMN_ID)),
            quantity =
cursor.getString(cursor.getColumnIndex(COLUMN_QUANTITY)),
            address =
cursor.getString(cursor.getColumnIndex(COLUMN_ADDRESS)),
        )
      }
      cursor.close()
      db.close()
      return order
   }

   @SuppressLint("Range")
   fun getAllOrders(): List<Order> {
      val orders = mutableListOf<Order>()
      val db = readableDatabase
      val cursor: Cursor = db.rawQuery("SELECT * FROM
$TABLE_NAME", null)
      if (cursor.moveToFirst()) {
         do {
            val order = Order(
               id = cursor.getInt(cursor.getColumnIndex(COLUMN_ID)),
               quantity =
cursor.getString(cursor.getColumnIndex(COLUMN_QUANTITY)),
               address =
cursor.getString(cursor.getColumnIndex(COLUMN_ADDRESS)),
            )
            orders.add(order)
         } while (cursor.moveToNext())
      }
      cursor.close()
      db.close()
      return orders
   }
   @SuppressLint("Range")
   fun clearAllOrders() {
      val db = writableDatabase
      db.delete(TABLE_NAME, null, null)  // Deletes all rows from the
order_table
      db.close()
   }


}
```

OrderDisplayStatus.kt:

```
package com.example.snackordering

import android.os.Bundle
import androidx.activity.ComponentActivity
import androidx.activity.compose.setContent
import androidx.annotation.DrawableRes
import androidx.compose.animation.core.*
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.draw.scale
import androidx.compose.ui.graphics.Color
import androidx.compose.ui.res.painterResource
import androidx.compose.ui.unit.dp
import com.example.snackordering.ui.theme.SnackOrderingTheme
import kotlinx.coroutines.delay

class OrderStatusDisplayActivity : ComponentActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContent {
            SnackOrderingTheme {
                OrderStatusScreen()
            }.
        }
    }
}

@Composable
fun OrderStatusScreen() {
    var tabIndex by remember { mutableStateOf(0) }
    val tabTitles = listOf("Being Readied", "On the Way", "Delivered")

    Column(modifier = Modifier.fillMaxSize()) {
        TabRow(selectedTabIndex = tabIndex) {
            tabTitles.forEachIndexed { index, title ->
                Tab(
                    selected = tabIndex == index,
                    onClick = { tabIndex = index },
                    text = { Text(text = title) }
                )
            }
        }
        Spacer(modifier = Modifier.height(16.dp))

        when (tabIndex) {
            0 -> OrderStatusContent(
                status = "Being Readied",
                imageRes = R.drawable.preparing_image, // Replace with your
actual drawable resource
                animationDuration = 5000
            )
            1 -> OrderStatusContent(
                status = "On the Way",
                imageRes = R.drawable.delivery_bike, // Replace with your
actual drawable resource
                animationDuration = 7000
            )
            2 -> OrderStatusContent(
                status = "Delivered",
                imageRes = R.drawable.delivered_image, // Replace with your
actual drawable resource
                animationDuration = 0
            )
        }
    }

    LaunchedEffect(Unit) {
        simulateOrderProgress { newStatusIndex ->
            tabIndex = newStatusIndex
        }
    }
}

@Composable
fun OrderStatusContent(status: String, @DrawableRes imageRes: Int,
animationDuration: Int) {
    Column(
        modifier = Modifier.fillMaxSize(),
        verticalArrangement = Arrangement.Center,
        horizontalAlignment = Alignment.CenterHorizontally
    ) {
        Text(text = "Order Status: $status", style =
MaterialTheme.typography.h5)

        Spacer(modifier = Modifier.height(16.dp))

        // Display image related to the status
        Image(
            painter = painterResource(id = imageRes),
            contentDescription = "$status Image",
            modifier = Modifier.size(150.dp)
        )

        Spacer(modifier = Modifier.height(16.dp))

        // If there's an animation duration, show a pulsating progress indicator
        if (animationDuration > 0) {
            PulsatingProgressBar(animationDuration = animationDuration)
        }
    }
}

@Composable
fun PulsatingProgressBar(animationDuration: Int) {
    // Pulsating animation using scale
    val scale by rememberInfiniteTransition().animateFloat(
        initialValue = 0.8f,
        targetValue = 1.2f,
        animationSpec = infiniteRepeatable(
            animation = tween(animationDuration, easing =
FastOutSlowInEasing),
            repeatMode = RepeatMode.Reverse
        )
    )

    Box(
        modifier = Modifier
            .size(50.dp)
            .scale(scale)
    ) {
        CircularProgressIndicator(
            color = Color(0xFFFF9800), // Orange color for fun pulsating
effect
            strokeWidth = 4.dp
        )
    }
}

suspend fun simulateOrderProgress(onStatusChange: (Int) -> Unit) {
    val delays = listOf(5000L, 10000L, 15000L) // Times in milliseconds for
each status change

    for ((index, delayTime) in delays.withIndex()) {
        delay(delayTime)
        onStatusChange(index)
    }
}
```

OrderStatusActivity.kt:

```
package com.example.snackordering

import android.os.Bundle
import androidx.activity.ComponentActivity
import androidx.activity.compose.setContent
import androidx.compose.animation.core.*
import androidx.compose.foundation.Image
import androidx.compose.foundation.layout.*
import androidx.compose.material.Text
import androidx.compose.runtime.*
import androidx.compose.ui.Alignment
import androidx.compose.ui.Modifier
import androidx.compose.ui.draw.scale
import androidx.compose.ui.res.painterResource
import androidx.compose.ui.unit.dp
import com.example.snackordering.ui.theme.SnackOrderingTheme

class OrderStatusActivity : ComponentActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContent {
            SnackOrderingTheme {
                OrderStatusScreen1()
            }
        }
    }
}

@Composable
fun OrderStatusScreen1() {
    var isDelivered by remember { mutableStateOf(false) }

    LaunchedEffect(Unit) {
        // Mock delay to simulate delivery
        kotlinx.coroutines.delay(5000)
        isDelivered = true
    }

    Column(
        modifier = Modifier.fillMaxSize(),
        horizontalAlignment = Alignment.CenterHorizontally,
        verticalArrangement = Arrangement.Center
    ) {
        if (isDelivered) {
            Text(text = "Your order will be there soon....!!!")
        } else {
            Text(text = "Your order is on the way...")
            Spacer(modifier = Modifier.height(16.dp))
            DeliveryAnimation()
        }
    }
}

@Composable
fun DeliveryAnimation() {
    val infiniteTransition = rememberInfiniteTransition()
    val scale by infiniteTransition.animateFloat(
        initialValue = 1f,
        targetValue = 1.2f,
        animationSpec = infiniteRepeatable(
            animation = tween(800),
            repeatMode = RepeatMode.Reverse
        )
    )

    Image(
        painter = painterResource(id = R.drawable.delivery_bike),
        contentDescription = "Delivery Animation",
        modifier = Modifier
            .size(100.dp)
            .scale(scale)
    )
}
```

ProfileDashboardActivity.kt:

```
package com.example.snackordering

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.foundation.shape.CircleShape
import androidx.compose.material.*
import androidx.compose.runtime.*
import androidx.compose.ui.Alignment
import androidx.compose.ui.Modifier
import androidx.compose.ui.draw.clip
import androidx.compose.ui.res.painterResource
import androidx.compose.ui.text.font.FontWeight
import androidx.compose.ui.unit.dp
import androidx.compose.ui.unit.sp
import com.example.snackordering.ui.theme.SnackOrderingTheme

class ProfileDashboardActivity : ComponentActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)

        // Retrieve username from intent extras
        val username = intent.getStringExtra("username") ?: “Guest”

        setContent {
            SnackOrderingTheme {
                Surface(
                    modifier = Modifier.fillMaxSize(),
                    color = MaterialTheme.colors.background
                ) {
                    ProfileDashboard(username)
                }
            }
        }
    }
}

@Composable
fun ProfileDashboard(username: String) {
    var address by remember { mutableStateOf("") }
    var savedAddress by remember { mutableStateOf<String?>(null) }

    Column(
        modifier = Modifier
            .fillMaxSize()
            .padding(16.dp),
        verticalArrangement = Arrangement.Center,
        horizontalAlignment = Alignment.CenterHorizontally
    ) {
        // Profile image
        Image(
            painter = painterResource(id = R.drawable.profile),  // Replace
with actual profile image resource
            contentDescription = "Profile Image",
            modifier = Modifier
                .size(100.dp)
                .clip(CircleShape)
        )

        Spacer(modifier = Modifier.height(16.dp))

        // Username display
        Text(text = "Welcome! $username ", fontSize = 20.sp, fontWeight =
FontWeight.Bold)

        Spacer(modifier = Modifier.height(8.dp))

        // Display saved address if available
        savedAddress?.let {
            Text(text = "Address: $it", fontSize = 16.sp, fontWeight =
FontWeight.Medium)
            Spacer(modifier = Modifier.height(8.dp))
        }

        // Address input field
        OutlinedTextField(
            value = address,
            onValueChange = { address = it },
            label = { Text("Enter your address") },
            modifier = Modifier.fillMaxWidth()
        )

        Spacer(modifier = Modifier.height(20.dp))

        // Save address button
        Button(onClick = {
            savedAddress = address  // Save the entered address
        }) {
            Text("Save Address")
        }
    }
}
```

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)
}
```

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.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.res.painterResource
import androidx.compose.ui.text.input.KeyboardType
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("Akhil", 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, "Added to cart Successfully",
Toast.LENGTH_SHORT).show()}
        },
            colors = ButtonDefaults.buttonColors(backgroundColor =
Color.White))
        {
            Text(text = "Add to cart", color = Color.Black)
        }


    }
```

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

User:

```
package com.example.snackordering

import androidx.room.ColumnInfo
import androidx.room.Entity
import androidx.room.PrimaryKey

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

    )
```

UserDao:

```
package com.example.snackordering

import androidx.room.*

@Dao
interface UserDao {

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

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

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

UserDatabase:

```
package com.example.snackordering

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

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

    abstract fun userDao(): UserDao

    companion object {

        @Volatile
        private var instance: UserDatabase? = null

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

UserDatabaseHelper:

```
package com.example.snackordering

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

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

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

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

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

        db?.execSQL(createTable)
    }

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

    fun insertUser(user: User) {
        val db = writableDatabase
        val values = ContentValues()
        values.put(COLUMN_FIRST_NAME, user.firstName)
        values.put(COLUMN_LAST_NAME, user.lastName)
        values.put(COLUMN_EMAIL, user.email)
        values.put(COLUMN_PASSWORD, user.password)
        db.insert(TABLE_NAME, null, values)
        db.close()
    }

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

   @SuppressLint("Range")
   fun getAllUsers(): List<User> {
      val users = mutableListOf<User>()
      val db = readableDatabase
```

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

}
```