Project Report Template

Introduction:

Online food ordering is the process of ordering food, for delivery or pickup, from a website or other application. The product can be either ready-to-eat food (e.g., direct from a home-kitchen, restaurant, or a virtual restaurant) or food that has not been specially prepared for direct consumption (e.g., vegetables direct from a farm/garden, fruits, frozen meats. etc).

Online food ordering/delivery through third-party companies have emerged as a global industry, leading to a "delivery revolution". From 2018 to 2021, global revenues for the online food delivery sector rose from $90 billion to $294 billion.

Purpose:

1.DoorDash

This is an on-demand restaurant app that delivers breakfast, lunch, and dinner from online user's preferred restaurants. They also have an alcohol-delivery service facility from restaurants, stores, breweries

2.GrubHub

This Uber for delivery is considered to be the most seamless restaurant delivery service. It allows users to search for their desired cuisines or browse through the list of local restaurants which are nearby through its "food near me" functionality.

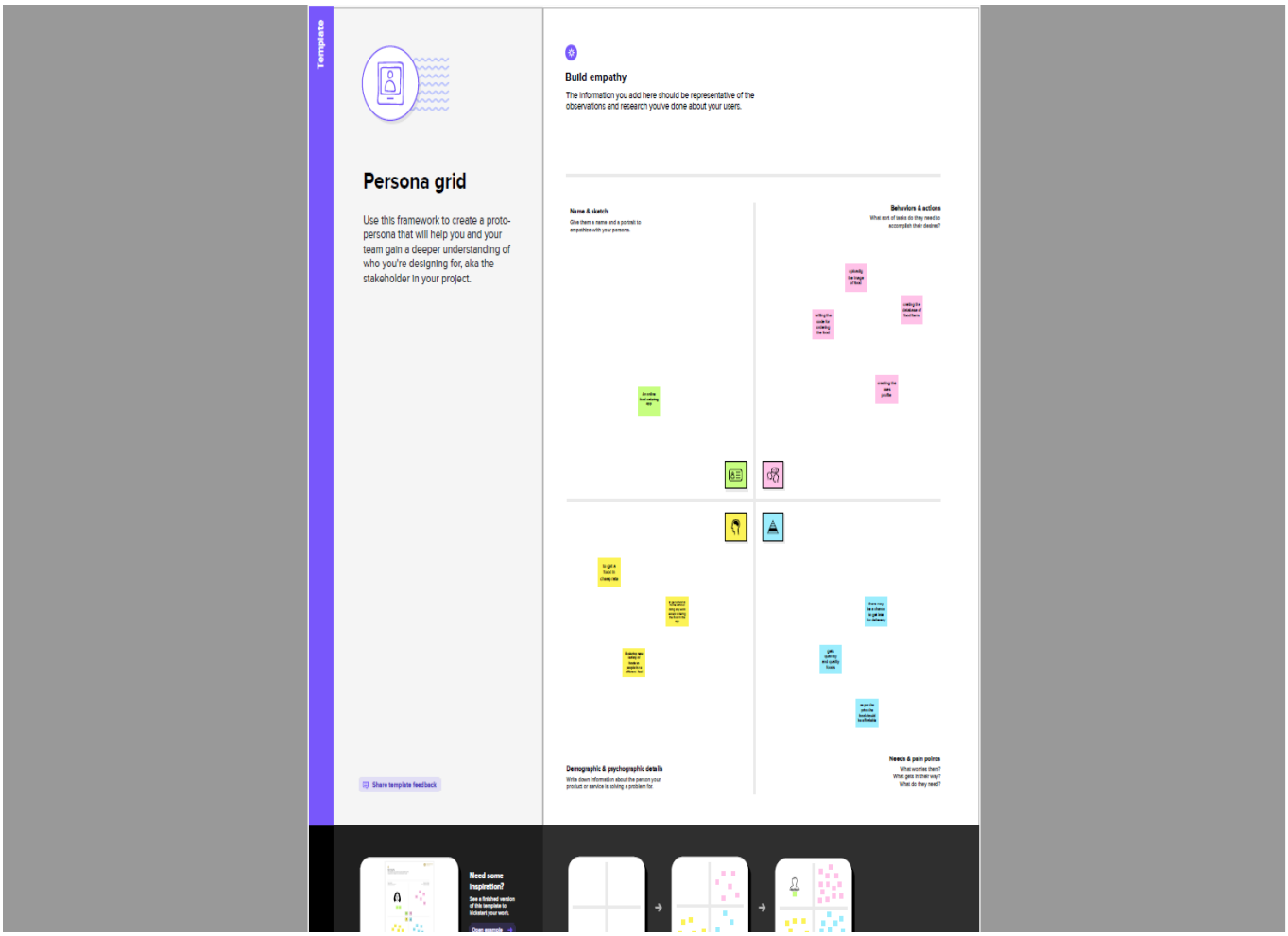
As you are planning for a food apps development, you can implement a feature "favourite list" where users can personalize their search and can find or reorder the items just in a tap, rather than roaming through the menu.

3.Uber Eats

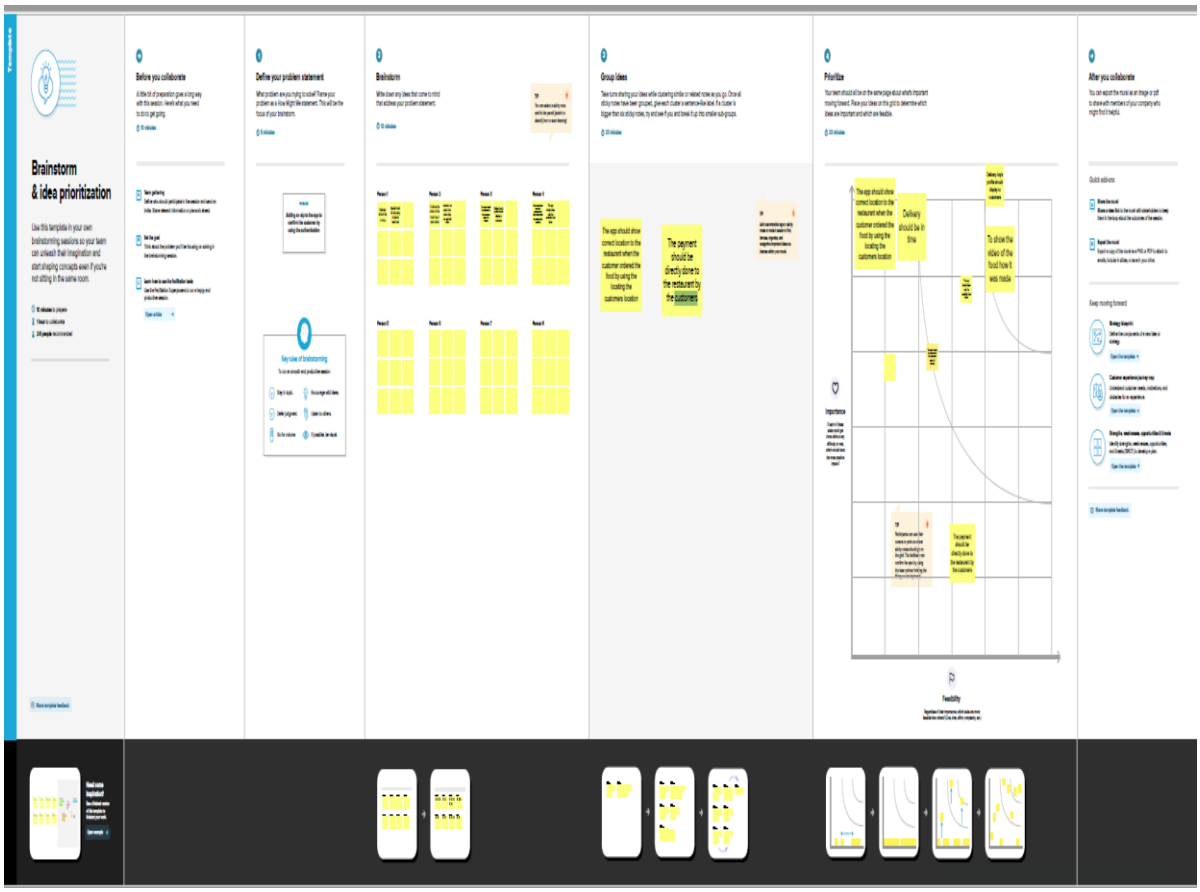
This app provides users with an easy payment gateway and simple ordering functionality. Thus while you plan your on-demand delivery app, make sure you cross-check your app's check-out features.

Problem definition and Design thinking:

Empathy map:

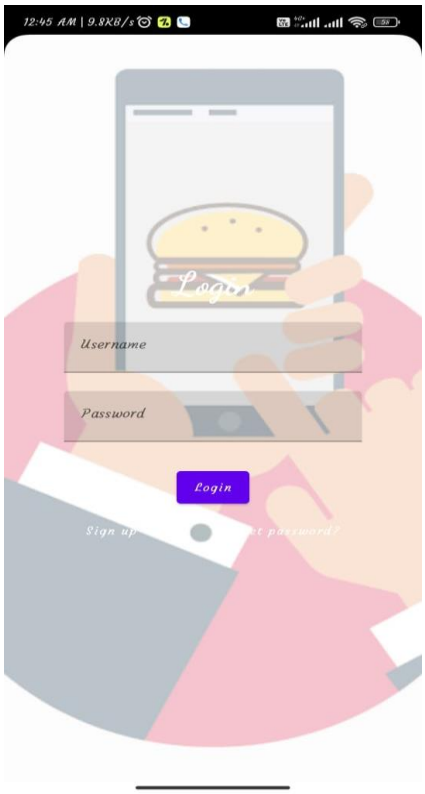


Ideation & Brainstorming map:

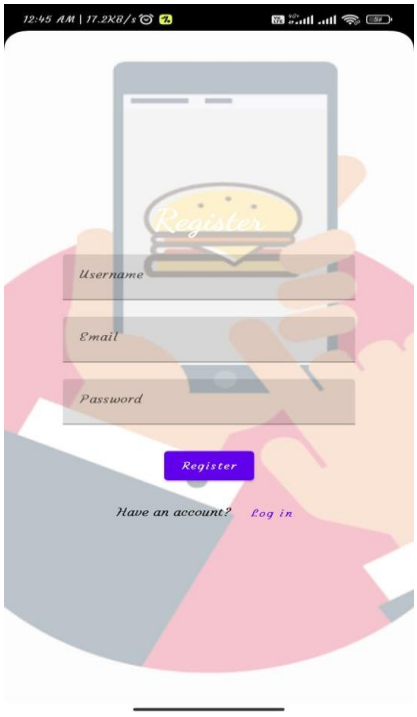


Result:

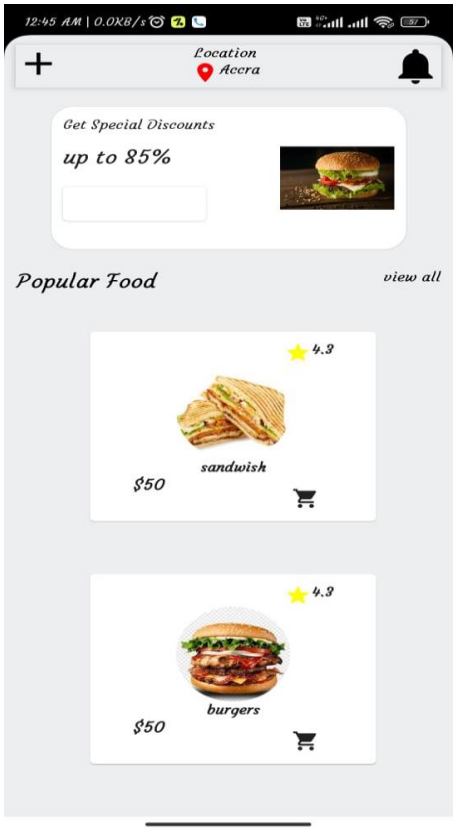
Login page:



Register page:

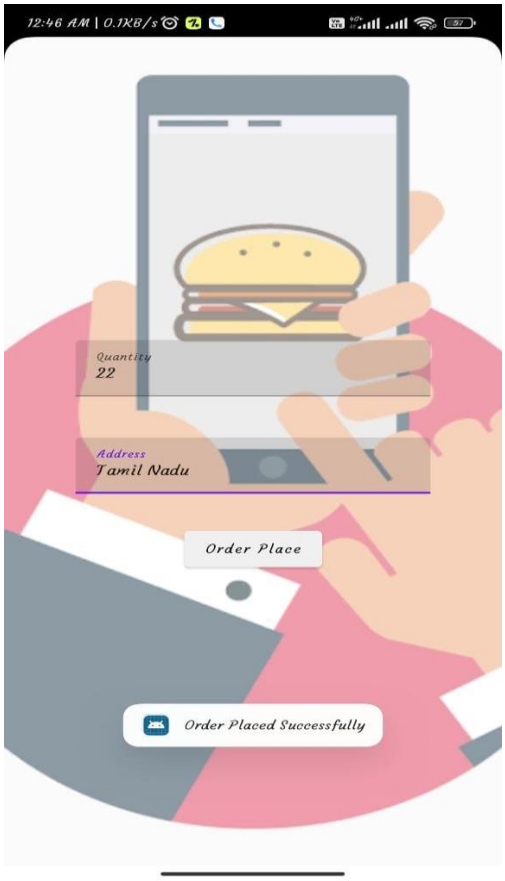


Main page:

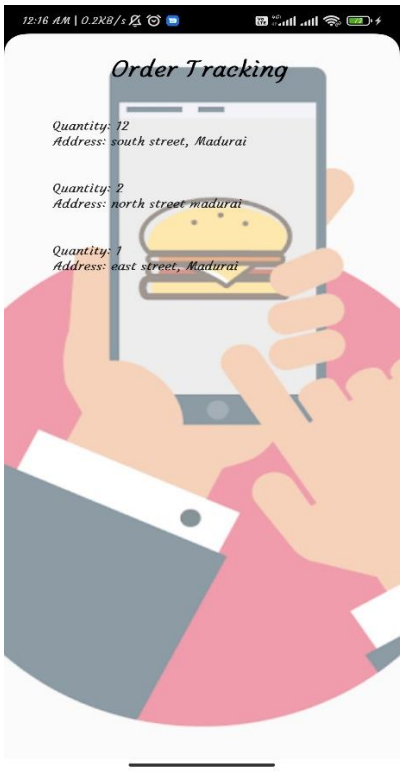


Ordering Page:

Order Confirmed page:



Admin Page:



Advantage:

- **Variety of Food:** Often, you get bored of eating the same food over time and want to eat something delicious and different. Online food delivery apps offer a variety of foods at your doorstep. From local to international, you can enjoy having the taste of any country or locality whenever you want.

- **Convenience:** When you are at home, you may feel reluctant to make food after the office or household work. And in this situation, neither you want to go to Kitchen to make food nor a restaurant to eat. So, an **online food ordering system** allows you to have food at your comfort.

- **Less Cost of Ad:** When you open a new restaurant, you have to spend a lot to advertise your business. But if you go for on-demand app development or for a website to digitize your business, you have to spend less on ads because they run on online platforms. Moreover, easy to reach potential customers.

- **Low Maintenance:** Off-line food business requires many resources and effort. While you can run the business flawlessly if you run it online because the maintenance cost will get reduced. You can collaborate with the restaurant owners and run an **online food ordering system**.

- **Time-Savvy:** The most important advantage of an online food delivery app is, it saves time. You will not have to go to the restaurant and wait for the food to get ready. You can plan the food as per your time and order 30 minutes before your eating time. Meanwhile, you can do your work while waiting for the food. Many want to know **how to hire mobile app developers**, because of the advantages of online food ordering system offers.

- **Increase Loyalty:** Customers will know your business and like to order using your app if you offer variety and quality. So, the online food business will help you increase the loyalty of your customers and ultimately the revenue.

Disadvantage:

- **Food Quality Compromised:** Due to high orders, some restaurants boil or heat the food on high flame to prepare it fast. It results, in destroying all the nutrients of the food and when customers eat it, they don't get any benefits. It is one of the important **cons of website ordering systems**.

- **Effects on Health:** Food in plastic boxes affects health adversely. People like fast and junk food, they order them online without worrying about bad effects on health. Often, over cheesy and artificial ingredients included in the food cause food poison.

- **High Competition:** Today, almost all people like to order food online, and due to the high demand, there are many apps and websites to fill various demands of people

related to food. It increases competition between different food ordering systems. So, high competition is a limitation in the **online food ordering system**.

- **High Delivery Charges:** You must have noticed, when you order food online, you have to pay some delivery charges. But heavy offers have psychological impacts on your mind and you ignore the heavy delivery or shipping charges.

- **Limited & Irregular Menu:** Often, you don't get the food types you last time ordered. It is due to irregularities in the menu. The customers can move to another app or restaurant due to it. In addition, the limited items to order become a hurdle to order food because people can't find their favourite ones.

Application:

It is used in some institutions and family functions to deliver the bulk orders on time .Also gets the food in all cities and villages without any partiality.-

Online food ordering gives the customers freedom and choice to place an order to virtually anytime, everywhere, saving the time and resources typically spend on travelling to pick up a meal. It also gives the customers the advantage of reordering the favourite order in the easiest and hassle-free manner

Conclusion:

This project used to learn that how the food ordering app works and what are the difficulties on doing this app. This project helps in gaining the knowledge about the coding. It also helps in keeping the touch with friends for long time.

Future Scope:

There is a feature called 'Advance Order' or 'Food Pre-Ordering' which allows users to schedule their order's delivery time. With the help of the food pre-ordering feature, customers get the freedom of choosing delivery or pickup time, at the time of placing their orders. Customers can select their usual order to be delivered immediately or set a particular time for future delivery. The restaurant is immediately notified about your customers' preferred schedule.

In this app, we have to add an tracking system and notify to the restaurant what the customer have ordered

Appendix:

Adding dependencies:

```
dependencies
{
```

```
    implementation 'androidx.core:core-ktx:1.7.0'
    implementation 'androidx.lifecycle:lifecycle-runtime-ktx:2.3.1'
    implementation 'androidx.activity:activity-compose:1.3.1'
    implementation "androidx.compose.ui:ui:$compose_ui_version"
    implementation "androidx.compose.ui:ui-tooling-
preview:$compose_ui_version"
    implementation 'androidx.compose.material:material:1.2.0'
    implementation 'androidx.room:room-common:2.5.0'
    implementation 'androidx.room:room-ktx:2.5.0'
    testImplementation 'junit:junit:4.13.2'
    androidTestImplementation 'androidx.test.ext:junit:1.1.5'
    androidTestImplementation 'androidx.test.espresso:espresso-core:3.5.1'
    androidTestImplementation "androidx.compose.ui:ui-test-
junit4:$compose_ui_version"
    debugImplementation "androidx.compose.ui:ui-
tooling:$compose_ui_version"
    debugImplementation "androidx.compose.ui:ui-test-
manifest:$compose_ui_version"
}
```

Creating User Data Class:

```
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
            }
        }
    }
}
```

Creating UserDao Interface:

```
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)
}
```

Creating UserDatabase Class:

```
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
            }
        }
    }
}
```

Creating UserDatabaseHelper class:

```
package
com.example.snackorde
ring

                    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_N
AME)),
                email =
cursor.getString(cursor.getColumnIndex(COLUMN_EMAIL)
),
```

```
                password =
cursor.getString(cursor.getColumnIndex(COLUMN_PASSW
ORD)),
            )
        }
        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_N
AME)),
                email =
cursor.getString(cursor.getColumnIndex(COLUMN_EMAIL)
),
                password =
cursor.getString(cursor.getColumnIndex(COLUMN_PASSW
ORD)),
            )
        }
        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_N
AME)),
                    email =
cursor.getString(cursor.getColumnIndex(COLUMN_EMAIL)
),
                    password =
cursor.getString(cursor.getColumnIndex(COLUMN_PASSW
ORD)),
                )
                users.add(user)
            } while (cursor.moveToNext())
        }
        cursor.close()
        db.close()
        return users
    }


}
```

Creating Order Data Class:

```
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?,
)
```

Creating OrderDao Interface:

```
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)
}
```

Creating OrderDatabase Class:

```
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
            }
        }
    }
}
```

Creating OrderDatabaseHelper:

```
package
com.example.snackorder
ing

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_QUANT
ITY)),
                address =
cursor.getString(cursor.getColumnIndex(COLUMN_ADDRE
SS)),
            )
        }
        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_QUANT
ITY)),
                address =
cursor.getString(cursor.getColumnIndex(COLUMN_ADDRE
SS)),
            )
        }
        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_QUANT
ITY)),
                    address =
cursor.getString(cursor.getColumnIndex(COLUMN_ADDRE
SS)),
                )
                orders.add(order)
            } while (cursor.moveToNext())
        }
        cursor.close()
        db.close()
        return orders
    }


}
```

Creating Login Page:

```
package
com.example.snackorderin
g

                    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.SnackOrderingThem
                    e


                    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)
    }
```

Creating Main Page:

```
package
com.example.snackorderin
g
```

```
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)
        }
    }
}
```

Creating Target Activity:

```
package
com.example.snackorderin
g

                                    import android.content.Context
                                    import android.content.Intent
                                    import android.os.Bundle
                                    import android.util.Log
                                    import android.widget.Toast
                                    import androidx.activity.ComponentActivity
                                    import androidx.activity.compose.setContent
                                    import androidx.compose.foundation.Image
                                    import androidx.compose.foundation.background
                                    import androidx.compose.foundation.layout.*
                                    import
                                    androidx.compose.foundation.text.KeyboardActions
                                    import
                                    androidx.compose.foundation.text.KeyboardOptions
                                    import androidx.compose.material.*
                                    import androidx.compose.runtime.*
                                    import androidx.compose.ui.Alignment
                                    import androidx.compose.ui.Modifier
                                    import androidx.compose.ui.graphics.Color
                                    import androidx.compose.ui.layout.ContentScale
                                    import androidx.compose.ui.platform.LocalContext
                                    import
                                    androidx.compose.ui.platform.textInputServiceFactory
                                    import androidx.compose.ui.res.painterResource
                                    import androidx.compose.ui.text.input.KeyboardType
                                    import androidx.compose.ui.tooling.preview.Preview
                                    import androidx.compose.ui.unit.dp
                                    import androidx.core.content.ContextCompat
                                    import
                                    com.example.snackordering.ui.theme.SnackOrderingThem
                                    e


                                    class TargetActivity : ComponentActivity() {
                                       private lateinit var orderDatabaseHelper:
                                    OrderDatabaseHelper
                                       override fun onCreate(savedInstanceState: Bundle?) {
                                          super.onCreate(savedInstanceState)
```

```
        orderDatabaseHelper = OrderDatabaseHelper(this)
        setContent {
            SnackOrderingTheme {
                // A surface container using the 'background'
color from the theme
                Surface(
                    modifier = Modifier
                        .fillMaxSize()
                        .background(Color.White)


                ) {
                    Order(this, orderDatabaseHelper)
                    val orders =
orderDatabaseHelper.getAllOrders()
                   Log.d("swathi", orders.toString())


                }
            }
        }
    }
}


@Composable
fun Order(context: Context, orderDatabaseHelper:
OrderDatabaseHelper){
    Image(painterResource(id = R.drawable.order),
contentDescription = "",
        alpha =0.5F,
    contentScale = ContentScale.FillHeight)
    Column(
        horizontalAlignment = Alignment.CenterHorizontally,
        verticalArrangement = Arrangement.Center) {


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

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


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


        TextField(value = address, onValueChange =
{address=it},
            label = { Text("Address") },
            modifier = Modifier
                .padding(10.dp)
                .width(280.dp))


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




        if (error.isNotEmpty()) {
            Text(
                text = error,
                color = MaterialTheme.colors.error,
                modifier = Modifier.padding(vertical = 16.dp)
            )
        }




        Button(onClick = {
            if( quantity.isNotEmpty() and
address.isNotEmpty()){
                val order = Order(
                    id = null,
                    quantity = quantity,
                    address = address
                )
```

```
                orderDatabaseHelper.insertOrder(order)
            Toast.makeText(mContext, "Order Placed
Successfully", Toast.LENGTH_SHORT).show()}
        },
            colors =
ButtonDefaults.buttonColors(backgroundColor =
Color.White))
        {
            Text(text = "Order Place", color = Color.Black)
        }




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

Creating Admin Activity:

```
package
com.example.snackorderin
g


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

```
import androidx.compose.ui.layout.ContentScale
import androidx.compose.ui.res.painterResource
import androidx.compose.ui.unit.dp
import androidx.compose.ui.unit.sp
import
com.example.snackordering.ui.theme.SnackOrderingThem
e
import java.util.*


class AdminActivity : ComponentActivity() {
   private lateinit var orderDatabaseHelper:
OrderDatabaseHelper
   override fun onCreate(savedInstanceState: Bundle?) {
      super.onCreate(savedInstanceState)
      orderDatabaseHelper = OrderDatabaseHelper(this)
      setContent {
         SnackOrderingTheme {
            // A surface container using the 'background'
color from the theme
            Surface(
               modifier = Modifier.fillMaxSize(),
               color = MaterialTheme.colors.background
            ) {
               val data=orderDatabaseHelper.getAllOrders();
               Log.d("swathi" ,data.toString())
               val order =
orderDatabaseHelper.getAllOrders()
               ListListScopeSample(order)
            }
         }
      }
   }
}


@Composable
fun ListListScopeSample(order: List<Order>) {
   Image(
      painterResource(id = R.drawable.order),
contentDescription = "",
      alpha =0.5F,
      contentScale = ContentScale.FillHeight)
   Text(text = "Order Tracking", modifier =
Modifier.padding(top = 24.dp, start = 106.dp, bottom =
24.dp ), color = Color.White, fontSize = 30.sp)
```

```
                Spacer(modifier = Modifier.height(30.dp))
                LazyRow(
                    modifier = Modifier
                        .fillMaxSize()
                        .padding(top = 80.dp),


                    horizontalArrangement = Arrangement.SpaceBetween
                ){
                    item {


                        LazyColumn {
                            items(order) { order ->
                                Column(modifier = Modifier.padding(top =
            16.dp, start = 48.dp, bottom = 20.dp)) {
                                    Text("Quantity: ${order.quantity}")
                                    Text("Address: ${order.address}")
                                }
                            }
                        }
                    }


                }
            }
```

Modifying the 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=".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>
```

Thank You