

SNACK SQUAD : CUSTOMISABLE SNACK ORDERING AND DELIVERY APP PROJECT REPORT

SUBMITTED BY

Team ID:NM2023TMID10849

HEENU.K

HANISHA.S

FELIGA XAVIER,S

JENISHA.J S

CONTENTS

PREFACE

1. INTRODUCTION

1.1 Overview

1.2 Purpose

2. PROBLEM DEFINITION & DESIGN THINKING

2.1 Empathy Map

2.2 Ideation & Brainstorming Map

3. RESULT

3.1 Data Model

3.2 Activity & Screenshot

4. ADVANTAGES & DISADVANTAGE

5.1 Advantages

5.2 Disadvantage

6. APPLICATIONS

7. CONCLUSION

8. FUTURE SCOPE

1.INTRODUCTION

1.1 OVERVIEW:

The snack ordering app project is a mobile application that allows users to order their preferred snacks from their smartphones. This application provides a convenient and hassle-free way of ordering snacks, and it is targeted at people who are always on the go and do not have the time to visit a physical store to buy snacks. The application offers a range of snacks, and users can choose from different options, including fast food, healthy snacks, and other options. In this report, we will present the key findings of our analysis of the snack ordering app project and provide a conclusion on the feasibility and viability of the project.

Market Analysis:

The snack ordering app market is a fast-growing and competitive market. Many companies have entered this market, including established food delivery platforms such as GrubHub and Uber Eats. These companies have a large customer base and a strong brand reputation, which makes it difficult for new players to enter the market.

The global online food delivery market is segmented based on the delivery model and region. Based on delivery model, the market is segmented into the traditional delivery model, aggregators, and new delivery mode

Functionalities provided by Online Food Ordering System are as follows:

- Provides the searching facilities based on various factors. Such as Food Item , Customer, Order, Confirm Order

- Online Food Ordering System also manage the Payment details online for Order details, Confirm Order details, Food Item.
- It tracks all the information of Category, Payment, Order etc
- Manage the information of Category
- Shows the information and description of the Food Item, Customer
- To increase efficiency of managing the Food Item, Category
- It deals with monitoring the information and transactions of Order.
- Manage the information of Food Item
- Editing, adding and updating of Records is improved which results in proper resource management of Food Item data.
- Manage the information of Order
- Integration of all records of Confirm Order

1.2 PUPOSE

Online food ordering system, as described above, can lead to error free, secure, reliable and fast management system. It can assist the user to concentrate on the record keeping. Thus it will help organization in better utilization of resources. The organization can maintain computerized records without redundant entries. That means that one need not be distracted by information that is not relevant, while being able to reach the information.

1.order food and drink from an extensive

2.users got to choose between pick up or in-seat delivery

3. Real-time order tracking

4.Multiple payment options

5. save frequently used credit/debit for a fast checkout 8. Add films Titles to order and delivery

6.To receive newly added food coupons and discount 8. Push notification message enabled

9. Rating & feedback system

2. PROBLEM DEFINITION AND DESIGN THINKING

2.1 EMPATHY MAP

Empathy map

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

 [Share template feedback](#)

Build empathy

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

Team ID :
NM2023TMID10849

Project title:
Customizable snack
ordering and delivery
app

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

the ordering
experience
needs to be
easy

I am in the mood for something spicy

you should look through the application to track down your favorite dish and tap to place order

Thinks
What are their wants, needs, hopes, and dreams? What other thoughts might influence their behavior?

the biggest growth channel right now is online ordering

I want
snack that's
sweet and
crunchy

fast delivery
of snacks

multiple payment option such as cash ,card,mobile wallets.

they want multiple delivery option

browse through different snack options and categories

pay for the order

I order food either by phone or online is to get it quickly

online menus
always ensure
customer that
what they want is
readily available

waits for
the snack
delivery to
arrive

add
snackslist to
their cart they
select for
eating

ordering food online can be a convenient and efficient way to get a meal

you can choose from a variety of cuisines and preferences

Does
What behavior have we observed?
What can we imagine them doing?

Feels

What are their fears, frustrations, and anxieties? What other feelings might influence their behavior?

Need some
inspiration?

See a refreshed version of this template to kickstart your work.

Open example



2.2 IDEATION AND BRAINSTORMING MAP



Brainstorm

Write down any ideas that come to mind that address your problem statement.

🕒 10 minutes

TIP

You can select a sticky note and hit the pencil [switch to sketch] icon to start drawing!



Team ID :
NM2023TMID10849

Project title:
Snack squad:
customizable snack
ordering and delivery
app

Heenu K

24 hours customer service	Multiple menu	multiple payment options
Sell coupons	Easy to use interface	Food delivery
Asking feedback	frozen food items delivery	Healthy snack delivery

folija xavier S

reorder feature for frequently ordered items	Home made snacks	Review system
Good service	offers for orders	personalized recommendation
Add to cart option	receipe suggestions	super fast food delivery service

Hanisha s

secure payment option	Advanced search functionality	delivery option for special occasions and events
Easy ordering system	customer support	Real time order tracking
Avoid food waste	Discount	Full time service

Jenisha J S

Push notification	fast delivery of food	Day LifeEssential Delivery
delivery and pickup	rewards for order	fast delivery service
multiple payment option	meal scheduling	Good snacks

3. RESULT

Register

Username

Email

Password

Register

Have an account?

Log in

Login

Username

Password

Login

Sign up

Forget password?

★ 4.3




butterscotch cake

600



★ 4.3



All In One

150



★ 4.3



bread burger

120



★ 4.3



Pasta

90



2:56


Location
Accra

Get Special Discounts
up to 85%

Claim voucher


Popular Food

view



★ 4.3

mojito
150






★ 4.3


shawarmas
80






★ 4.3


Tequila
120





★ 4.3

Wine
120





★ 4.3


peach juice
120





★ 4.3

cantaloupe juice
200



Quantity
5

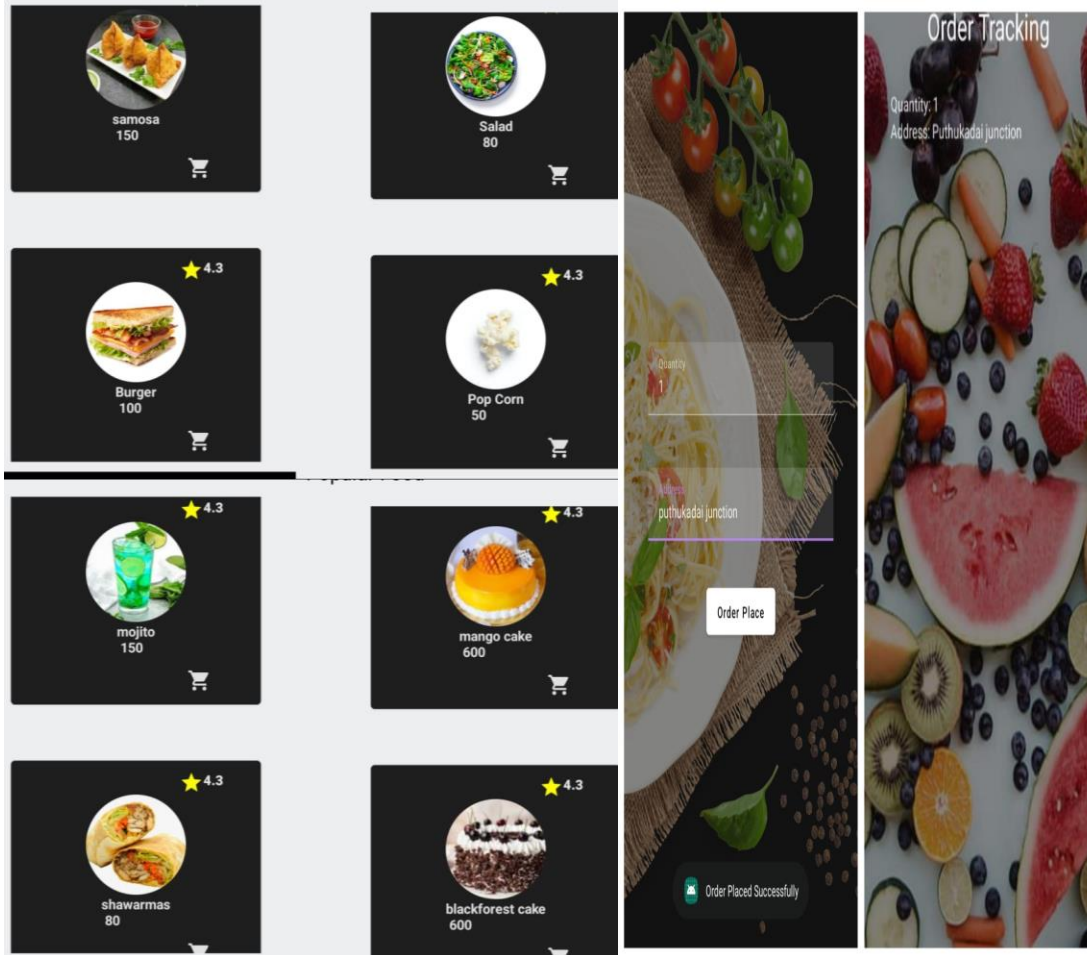
Address
villavilal

Order Place

Quantity

Address

Order Place



4. ADVANTAGES AND DISADVANTAGES

ADVANTAGES

Advantages:

Running an online food ordering system adds flexibility to the business, which will ultimately increase sales and profits.

- **Easy, fast, and comfortable:**

In short, your customers choose to order food online because it is really at their fingertips.

Anyone with a smartphone can order food online from their favorite restaurant.

More than 97% of millennials use their phones for anything. Ordering food online comes into the same broad category. So using the online food ordering system is the easiest way to attract millennials

- **Health benefits:**

One of the important benefits of food ordering systems is health benefits. Because the meal is planned, it is easy to determine the exact number of calories consumed in each meal. Many food ordering systems retain their menu for health benefits and weight loss, which can be very helpful for individuals who are trying to lose weight and start a healthy diet.

- **Safer and healthier:**

To reopen, food businesses will have to set up shop to meet the health and safety regulations of the Indian government. Owners must maintain social distances, use non-contact ordering and payment methods, and ensure surfaces are regularly cleaned.

Even if you are a small shop owner, Switching to the online ordering system for businesses means that your customers can order food without coming to the store and pay online without contact. This method not only brings profit to your business but also protects from the spread of covid-19.

- **Less chance for errors:**

One of the best advantages of an online food ordering system for customers is that it ensures prices are accurate and there is less room for error when it comes time to settle the bill.

This is because customers have to select an item in the menu at the appropriate price and make sure that the right amount is always paid.

This has some good benefits for your business; The chances of mischarging are low, less time in ordering errors, and helping to provide satisfactory service to customers.

- **More customers:**

As the new life progresses with technologies, online orders and payments are expected to be accepted. If your payment and menu method is hassle free, your regular customers will recommend you to their friends and will share on social media about your restaurant.

You can maximize your customers and your profits by providing a seamless customer experience.

DISASVANTAGES

- **Price:**

One of the major drawbacks of online food ordering systems is price. When food is ordered for more than one person, the cost is usually equal to eating at a good restaurant every night. Many food ordering systems cost more than \$ 20 per person per day. Even more expensive for some other food ordering systems. For individuals with a limited food budget, online food ordering systems are often too expensive

- **Quality of food may be suffer:**

One problem with the food ordering system is that the quality of the food served is often worse than eating at a restaurant. Often, food has to be fed over long distances, and over time, precious vitamins can be lost. Also, food from the ordering system is often served in plastic packaging, which

may not be very appealing to your eyes compared to the food neatly placed on your plate in a restaurant

- **The vibe of the restaurant is missing:**

In some restaurants, there is also a good circumstance which you will miss if you order your food at home.

For example, if you spend your evening in a good Chinese restaurant, you will often feel like you are actually in China because the decoration and the whole atmosphere are in line with the Chinese way of life. If you order food at your home, you will lose all of these.

Also, from time to time, it would be great if you could take your partner or family to a nice restaurant for dinner to spend a good evening.

Isolated Disconnect:

Dining at a restaurant is an exciting and delicious experience, meant to be shared with family and friends. Consumers lose the restaurant ambiance and dining interaction when opting to utilize an online food delivery service. Eating in solitude with only your favourite streaming shows to accompany you (another topic for a different blog post) fosters an attitude of isolationism, slowly disconnecting you from the outside world.

5. APPLICATION

The use of "A Customisable Snack Ordering and Delivery App" can be a convenient one that can be used in many application areas. That can be done by using mobile phones. We can implement it in on theaters as much as possible to reduce the time and work done. We can also apply this in small retailer shops and also in big retailers to makes them so fast and convenient and easy to maintainable one. Snack Ordering App can be

applicable in many real time usage for the snack delivery at any place that the range of the application that has been designed.

6. CONCLUSION:

The Food Ordering Android App is an online food ordering and delivery application that simplifies the food ordering process and provides a convenient way for customers to order food from the comfort of their own homes. The application will be developed using Java and Android Studio, and will include features such as user registration, menu management, search functionality, menu selection, order placement, order tracking, user rating, and an admin panel. The application will be tested to ensure that it meets the requirements and is free of bugs, and will be deployed to the Google Play Store for users to download and use.

Our project is only a humble venture to satisfy the needs to manage their project work. Several user friendly coding have also adopted. This package shall prove to be a powerful package in satisfying all the requirements of the school. The objective of software planning is to provide a frame work that enables the manger to make reasonable estimates made within a limited time frame at the beginning of the software project and should be updated regularly as the project progresses.

7. FUTURE SCOPE

In a Android Studio using Kotlin, it can be summarized that the future scope of the project circles around maintaining information regarding: ☐We can add printer in future. ☐We can

give more advance software for Online Snack Ordering System including more facilities.

- *We will host the platform on online servers to make it accessible worldwide.

- *Integrate multiple load balancers to distribute the loads of the system

- *Create the master and slave database structure to reduce the overload of the database queries.

- *Implement the backup mechanism for taking backup of code base and database on regular basis on different servers. The above-mentioned points are the enhancements which can be done to increase the applicability and usage of this project.

Here we can maintain the records of Snack Item and Category.

Also, as it can be seen that now-a-days the players are versatile, i.e. so there is a scope for introducing a method to maintain the Online Food Ordering System. Enhancements can be done to maintain all the Snack Item, Category, Customer, Order, Confirm Order. We have left all the options open so that if there is any other future requirement in the system by the user for the enhancement of the system then it is possible to implement them. In the last we would like to thanks all the persons involved in the development of the system directly or indirectly. We hope that the project will serve its purpose for which it is develop there by underlining success of process.

8.APPENDIX

SOURCE CODE

CREATING DATABASE CLASSES

Database 1 (Create User Data Class)

```
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?,

)
```

Create An 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)

}

```

Create An 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
    }
}
}
}
}

```

Create An UserDatabaseHelper Class

```

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()  
}
```

```
}
```

```
@SuppressWarnings("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
    }

    @SuppressWarnings("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  
}
```

```
@SuppressWarnings("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  
  
    }  
  
}
```

DATABASE

2 Create 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?,  
  
)
```

Create 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)
```

```
}
```

Create 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

    }

}

}

}

```

Create OrderDatabaseHelper Class

```

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()

}
```

```
@SuppressWarnings("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

}

}

```

Building Application UI And Connecting To Database.

Creating LoginActivity.Kt With Database

```

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,
```

```
        onChange = { username = it },
```

```
        label = { Text("Username") },
```

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

        .width(280.dp)

    )
```

```
    TextField(

        value = password,

        onChange = { 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 MainActivity.Kt With Database

```
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,  
  
        onChange = { username = it },  
  
        label = { Text("Username") },  
  
        modifier = Modifier  
            .padding(10.dp)  
            .width(280.dp)  
  
    )
```

```
    TextField(  
  
        value = email,  
  
        onChange = { 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)
}

```

Creating MainPage.Kt File

```

package com.example.snackordering

import android.annotation.SuppressLint
import android.content.Context
import android.os.Bundle
import android.widget.Toast

```

import androidx.activity.ComponentActivity

import androidx.activity.compose.setContent

import androidx.annotation.DrawableRes

import androidx.annotation.StringRes

import androidx.compose.foundation.Image

import androidx.compose.foundation.background

*import androidx.compose.foundation.layout.**

import androidx.compose.foundation.shape.CircleShape

import androidx.compose.foundation.shape.RoundedCornerShape

*import androidx.compose.material.**

import androidx.compose.material.icons.Icons

*import androidx.compose.material.icons.filled.**

import androidx.compose.runtime.Composable

import androidx.compose.ui.Alignment

import androidx.compose.ui.Modifier

import androidx.compose.ui.draw.clip

import androidx.compose.ui.graphics.Color

import androidx.compose.foundation.lazy.LazyColumn

import androidx.compose.foundation.lazy.items

import androidx.compose.material.Text

import androidx.compose.ui.unit.dp


```

import androidx.compose.ui.graphics.RectangleShape

import androidx.compose.ui.layout.ContentScale

import androidx.compose.ui.platform.LocalContext

import androidx.compose.ui.res.painterResource

import androidx.compose.ui.res.stringResource

import androidx.compose.ui.text.font.FontWeight

import androidx.compose.ui.unit.sp

import androidx.core.content.ContextCompat.startActivity

import com.example.snackordering.ui.theme.SnackOrderingTheme

import android.content.Intent as Intent1

class MainPage : ComponentActivity() {

    override fun onCreate(savedInstanceState: Bundle?) {

        super.onCreate(savedInstanceState)

        setContent {

            SnackOrderingTheme {

                // A surface container using the 'background' color from the theme

                Surface(

                    modifier = Modifier.fillMaxSize(),

                    color = MaterialTheme.colors.background

                ){

                    FinalView(this)

```

```

        val context = LocalContext.current

        //PopularFoodColumn(context)

    }

}

}

}

}

@Composable

fun TopPart() {

Row(

    modifier = Modifier

        .fillMaxWidth()

        .background(Color(0xffeceef0)), Arrangement.SpaceBetween

    ){

    Icon(

        imageVector = Icons.Default.Add, contentDescription = "Menu Icon",

        Modifier

            .clip(CircleShape)

            .size(40.dp),

        tint = Color.Black,

```

)

Column(horizontalAlignment = Alignment.CenterHorizontally) {

Text(text = "Location", style = MaterialTheme.typography.subtitle1, color = Color.Black)

Row {

Icon(

imageVector = Icons.Default.LocationOn,

contentDescription = "Location",

tint = Color.Red,

)

Text(text = "Accra" , color = Color.Black)

}

}

Icon(

imageVector = Icons.Default.Notifications, contentDescription = "Notification Icon",

Modifier

.size(45.dp),

tint = Color.Black,

)

}

}

@Composable

```

fun CardPart() {

    Card(modifier = Modifier.size(width = 310.dp, height = 150.dp), RoundedCornerShape(20.dp)) {

        Row(modifier = Modifier.padding(10.dp), Arrangement.SpaceBetween) {

            Column(verticalArrangement = Arrangement.spacedBy(12.dp)) {

                Text(text = "Get Special Discounts")

                Text(text = "up to 85%", style = MaterialTheme.typography.h5)

                Button(onClick = {}, colors = ButtonDefaults.buttonColors(Color.White)) {

                    Text(text = "Claim voucher", color = MaterialTheme.colors.surface)

                }

            }

        }

        Image(

            painter = painterResource(id = R.drawable.food_tip_im),

            contentDescription = "Food Image", Modifier.size(width = 100.dp, height = 200.dp)

        )

    }

}

@Composable

fun PopularFood(

    @DrawableRes drawable: Int,

    @StringRes text1: Int,

```

```

context: Context

){

Card(

    modifier = Modifier

        .padding(top=20.dp, bottom = 20.dp, start = 65.dp)

        .width(250.dp)

    ){

Column(

    verticalArrangement = Arrangement.Top,

    horizontalAlignment = Alignment.CenterHorizontally

    ){

Spacer(modifier = Modifier.padding(vertical = 5.dp))

Row(

    modifier = Modifier

        .fillMaxWidth(0.7f), Arrangement.End

    ){

Icon(

    imageVector = Icons.Default.Star,

    contentDescription = "Star Icon",

    tint = Color.Yellow

```

)

Text(text = "4.3", fontWeight = FontWeight.Black)

}

Image(

painter = painterResource(id = drawable),

contentDescription = "Food Image",

contentScale = ContentScale.Crop,

modifier = Modifier

.size(100.dp)

.clip(CircleShape)

)

Text(text = stringResource(id = text1), fontWeight = FontWeight.Bold)

Row(modifier = Modifier.fillMaxWidth(0.7f), Arrangement.SpaceBetween) {

/*TODO Implement Prices for each card*/

Text(

text = "\$50",

style = MaterialTheme.typography.h6,

fontWeight = FontWeight.Bold,

fontSize = 18.sp

)

```

IconButton(onClick = {

    //var no=FoodList.lastIndex;

    //Toast.

    val intent = Intent1(context, TargetActivity::class.java)

    context.startActivity(intent)

}) {

    Icon(

        imageVector = Icons.Default.ShoppingCart,

        contentDescription = "shopping cart",

    )

}

}

}

}

}

```

```

private val FoodList = listOf(

```

```

    R.drawable.sandwish to R.string.sandwich,

```

```
R.drawable.sandwich to R.string.burgers,  
  
R.drawable.pack to R.string.pack,  
  
R.drawable.pasta to R.string.pasta,  
  
R.drawable.tequila to R.string.tequila,  
  
R.drawable.wine to R.string.wine,  
  
R.drawable.salad to R.string.salad,  
  
R.drawable.pop to R.string.popcorn  
).map { DrawableStringPair(it.first, it.second) }
```

```
private data class DrawableStringPair(  
  
    @DrawableRes val drawable: Int,  
  
    @StringRes val text1: Int  
  
)
```

```
@Composable
```

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

```
    Column(  
  
        modifier = Modifier  
  
            .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))

        CardPa 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 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.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.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("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 AdminActivity.Kt

```
package com.example.snackordering
```

```
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.SnackOrderingTheme

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 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>
```