Project Report Template

Snack Squad: A Customizable Snack Ordering and Delivery App

1 INTRODUCTION

1.1 Overview

A food delivery app that provides food delivery at your door in very less time and with the best packaging.Providing food from every famous food place near you.Order food with the best user experience.

Food delivery apps are a type of restaurant delivery/ takeout software that connects consumers with local restaurants, grocery stores, convenience stores, etc., by providing a convenient way to order food that's delivered to their doorstep.

An online food ordering system allows your business to accept and manage orders placed online for delivery or takeaway. Customers browse a digital menu, either on an app or website and place and pay for their order online.

Food delivery is a home delivery service in which a store, restaurant, or third-party app delivers food to consumers, whenever they ask for it. These days, the offers are generally placed through a mobile app, website, or phone.

Online Food ordering system is a process in which one can order various foods and beverages from some local restaurant and hotels through the use of internet, just by sitting at home or any place. And the order is delivered to the told location.

An ordering system is referred as a set of detail methods that is being used in handling the ordering process Food ordering can be computerized or done manually. A computerized ordering system or more often known as Ordering Management System (OMS) can be defined in several ways.

1.2 Purpose

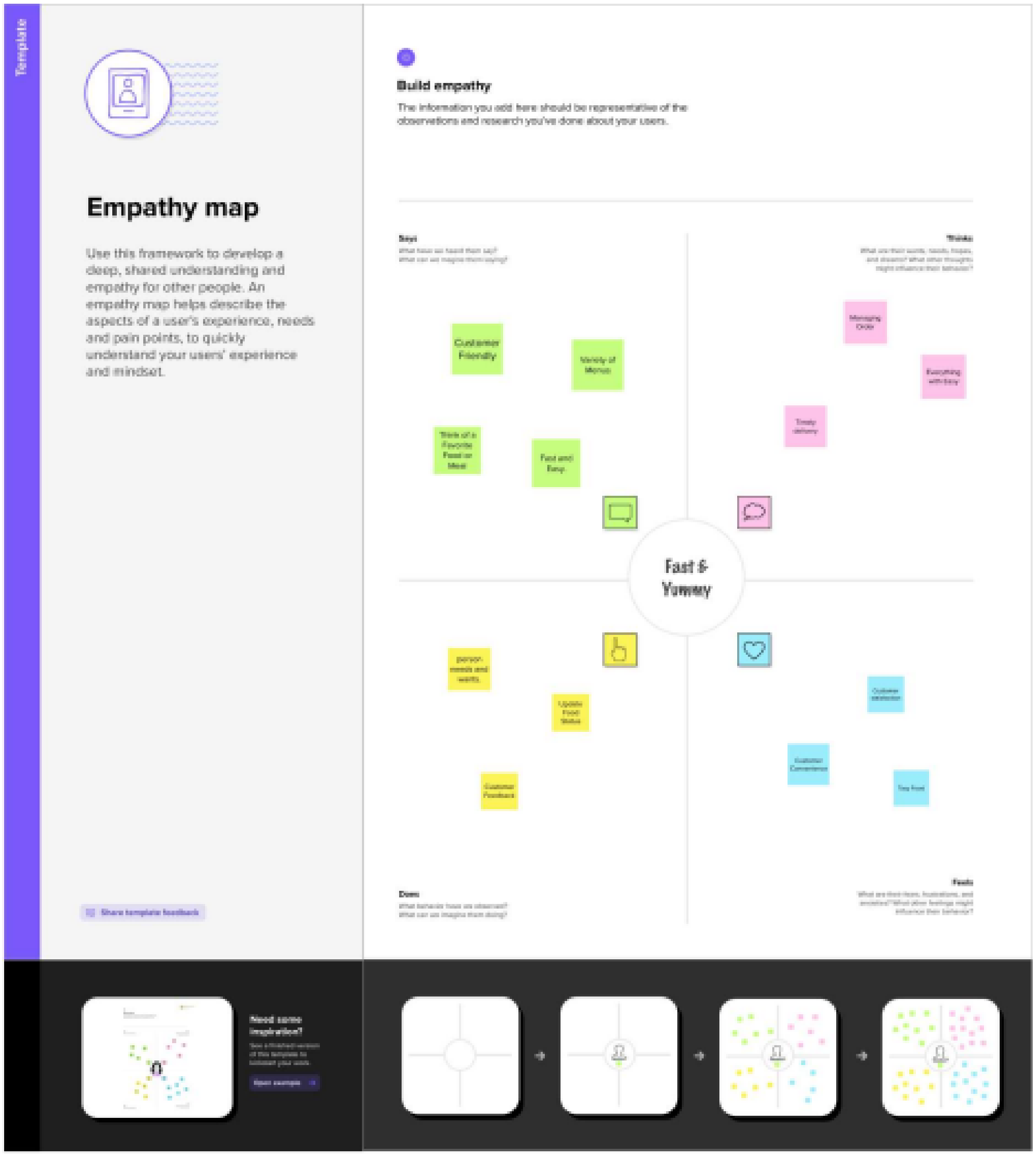
The Purpose of this app was to build a food ordering client server application. This application provides a view of current food information on the website and Android application. The customer can order food from these two platforms. For the administrator in restaurant, this application offers a series of operations to add, update, delete and query the information of food, food order and employees.

Food-delivery apps allow customers to order from a nearby restaurant at their convenience. The customers can get their order delivered, they can pick it up themselves or they can dine in. The restaurants receive the order on the restaurant app and prepare the meal.

An online food ordering system or an online ordering platform is a place where customers can directly order from the restaurant instead of going through a third-party food delivery business. It is a web-based ordering system where customers using a mobile app can use the online user interface to order online. What is the purpose of food delivery app?

2 PROBLEM DEFINITION & DESIGN THINKING

2.1 Empathy Map



Template

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.

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

- Customer Friendly
- Variety of Menus
- Think of a Favorite Food or Meal
- Fast and Easy

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

- Managing Order
- Everything with Easy
- Timely delivery

Fast & Yummy

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

- person needs and wants
- Update Food Status
- Customer Feedback

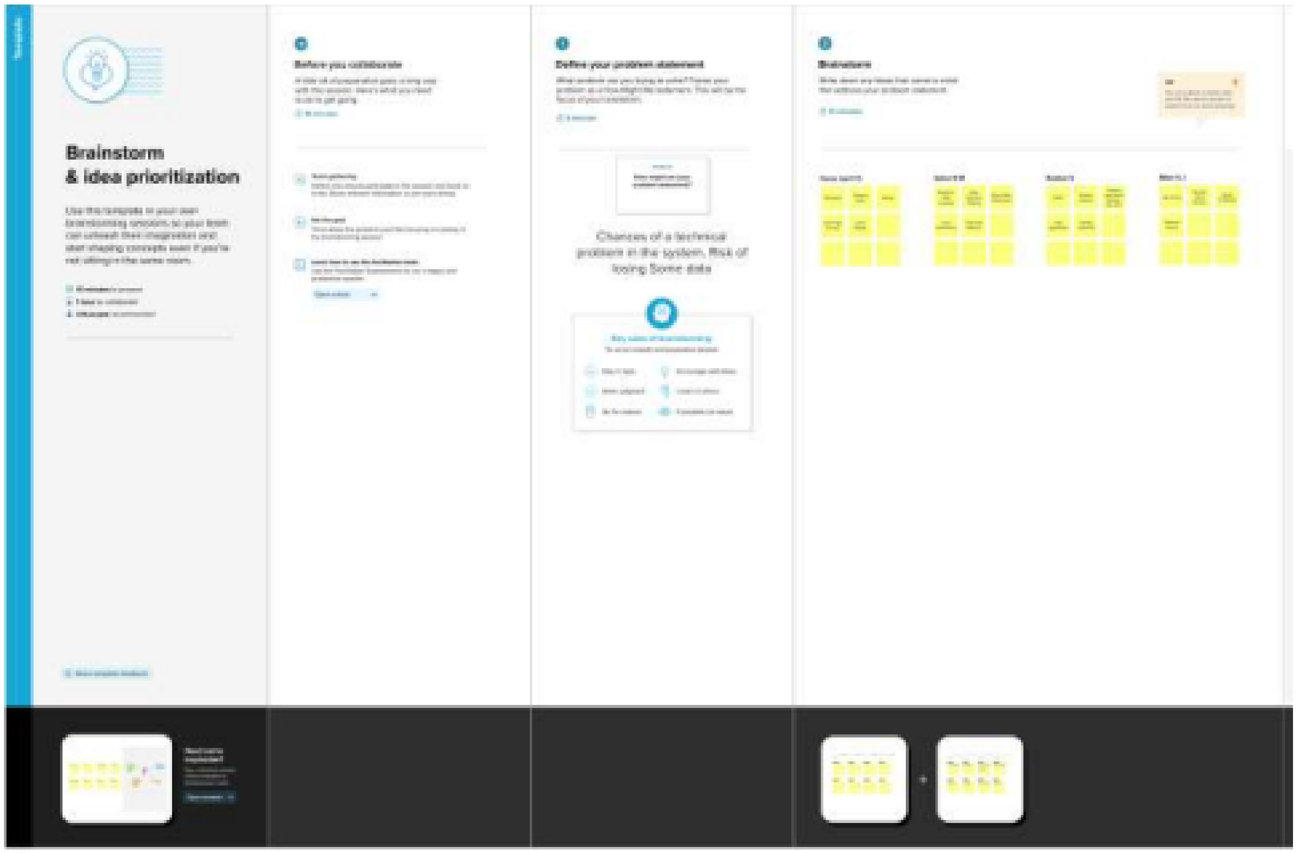
Feels
What are their fears, frustrations, and anxieties? What other feelings might influence their behavior?

- Customer satisfaction
- Customer Convenience
- Tiny Food

Need some inspiration?
See a finished version of this template to kickstart your work.

Open example

2.2 Ideation & Brainstorming Map

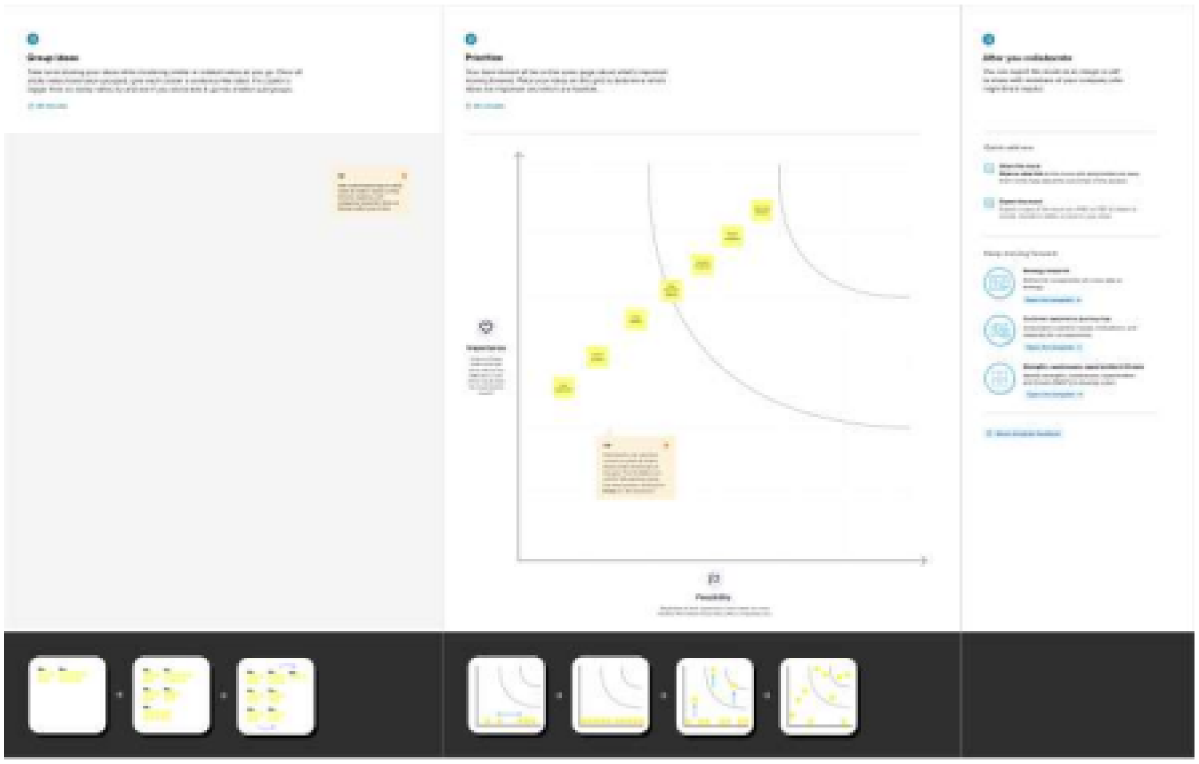

Brainstorm
& idea prioritization

Before you collaborate

Define your problem statement

Chances of a technical problem in the system, Risk of losing Some data

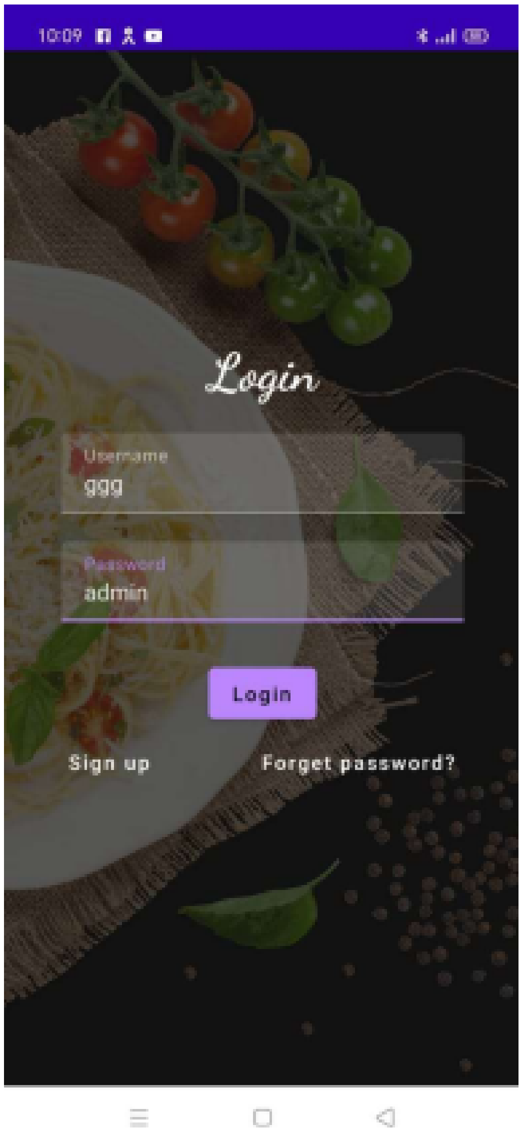
Brainstorm

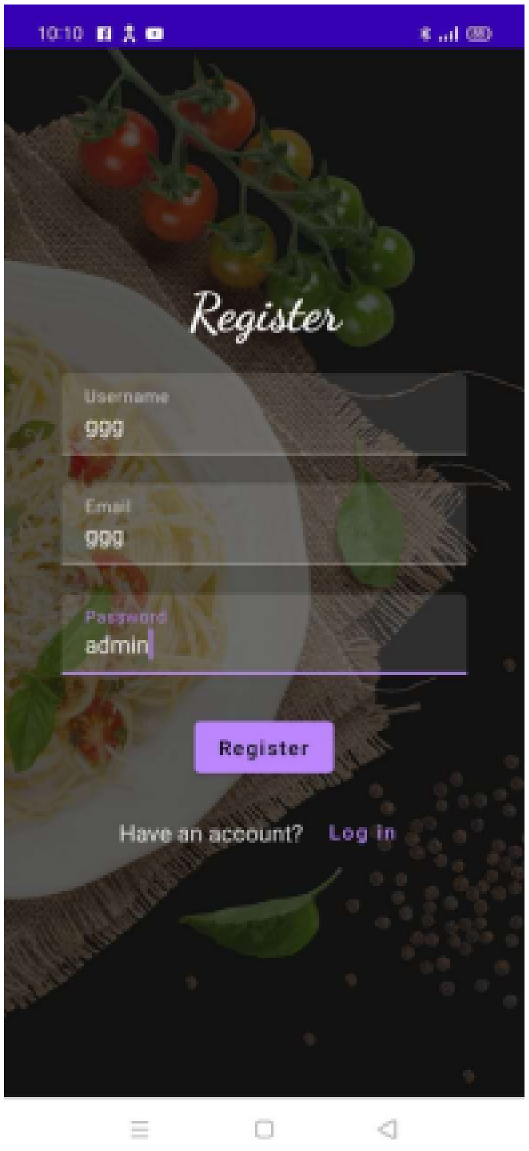

Group ideas

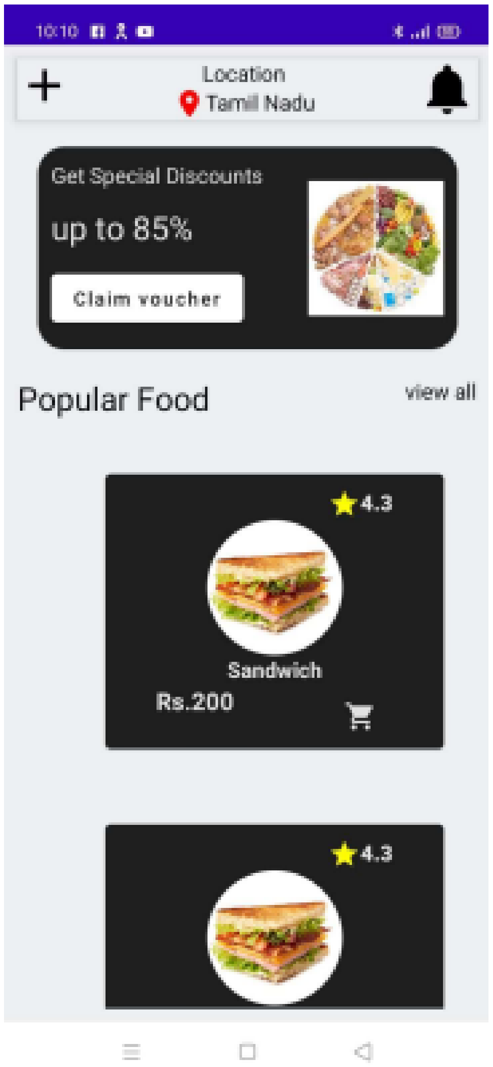
Prioritize

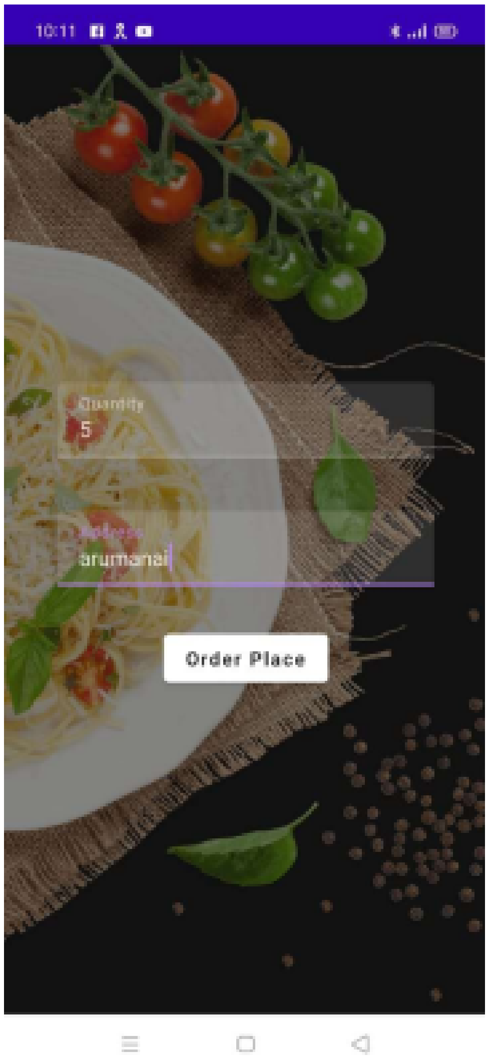
After you collaborate

3 RESULT

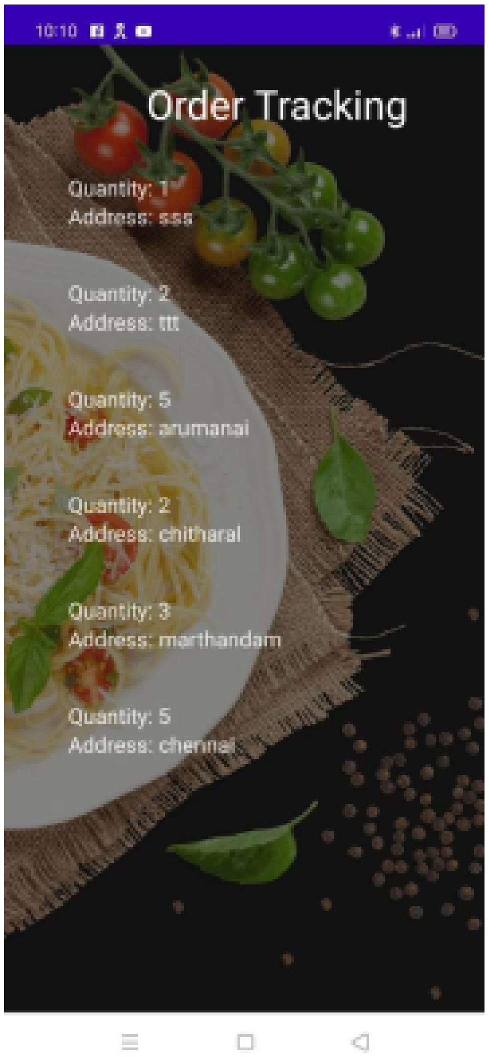

10:09
Login
Username
ggg
Password
admin
Login
Sign up
Forget password?


10:10
Register
Username
ggg
Email
ggg
Password
admin
Register
Have an account?
Log in





Quantity: 5
Address: arumanai



Order Tracking

Quantity: 1
Address: sss

Quantity: 2
Address: ttt

Quantity: 5
Address: arumanai

Quantity: 2
Address: chitharal

Quantity: 3
Address: marthandam

Quantity: 5
Address: chennai

4. ADVANTAGES & DISADVANTAGES

Advantages:

- It expands your customer base, increases your revenue, gives your customers a variety of options, and it also offers unparalleled convenience.
- An online food ordering system or an online ordering platform is a place where customers can directly order from the restaurant instead of going through a third-party food delivery business.
- Mobile apps provide the freedom to order from any place at any time without pausing everything and making a call to the restaurant. The food experience has come a long way it has become a much more hassle-free experience for the customers.

- Convenience
- Less cost of Ad
- Time –savvy
- Increase loyalty
- More customers
- Increase visibility
- Wider market
- Revamp revenue

Disadvantages:

Irregular sales. It is difficult to determine when demand may arise for a particular customized product. Lengthy delivery time. Since production starts after receiving an order, the product reaches the customer after some time. Availability of raw materials.

The two main disadvantages of make to order manufacturing are promptitude and cost of customization. Since products are not created until there is a customer order, it may take a while for the customer to actually get the product after the completed customized production and delivery.

- Food quality compromised
- High competition
- High delivery charges
- Limited & Irregular Menu
- Food can get cold
- Late and Incorrect orders

5 APPLICATIONS

Food delivery is a home delivery service in which a store, restaurant, or third-party app delivers food to consumers, whenever they ask for it. These days, the offers are generally placed through a mobile app, website, or phone.

Convenience is everything that food delivery apps look forward to putting its focus on food delivery services. This is a major transformation in the way consumers eat food. Realizing the fact about that the modern consumers are reluctant to cook and more likely to buy food online from nearby restaurants, it has added to the steep demand for on-demand food ordering app development both for web and mobile devices.

For consumers, it is easy to download on the device, select the food, place an order and make a payment using an in-app purchase feature upon delivery. So much convenience for consumers! Isn’t it? The sellers also can have a similar benefit if they invest in food delivery application development. If you have yet to have such a plan, finding answers to the following question can eliminate your confusion about food application development.

Convenience is everything that food delivery apps look forward to putting its focus on food delivery services. This is a major transformation in the way consumers eat food. Realizing the fact about that the modern consumers are reluctant to cook and more likely to buy food online from nearby restaurants, it has added to the steep demand for on-demand food ordering app development both for web and mobile devices.

For consumers, it is easy to download on the device, select the food, place an order and make a payment using an in-app purchase feature upon delivery. So much convenience for consumers! Isn't it? The sellers also can have a similar benefit if they invest in food delivery application development. If you have yet to have such a plan, finding answers to the following question can eliminate your confusion about food application development.

6 CONCLUTION

- Made statement of the aims and objectives of the project.
- The description of Purpose. Scope, and applicability
- We define the problem on which we are working in the project.
- We describe the requirement Specifications of the system and the actions thatcan be done on these things.
- We understand the problem domain and produce a model of the system, whichdescribes operations that can be performed on the system.

7 Future Scope

Food ordering and delivery processes in global restaurants and eateries have been revolutionized by online technology. With the advent of innovative delivery tracking apps, customers no longer have to talk and book their orders over the phone. Nor there is any need for explaining the finer details and preferences to the restaurant which may sometimes be neglected or misunderstood, causing unease or dissatisfaction. No more missed or misinterpreted orders, thanks to the novel delivery tracking app that renders food ordering and delivery a cinch.

An online food ordering system or an online ordering platform is a place where customers can directly order from the restaurant instead of going through a third-party food delivery business. It is a web-based ordering system where customers using a mobile app can use the online user interface to order online.

As health and wellness are surfacing as a high priority among food lovers, more and more Consumer Product Good (CPG) makers are trying to fuse traditional functional ingredients into their products. Probiotics, nootropics and adaptogens are being incorporated into beverages and food in a completely new way.

8 APPENDIX

A. Source Code

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}")

}

}

}

}


}

}
```

LoginActivity.kt

```
package com.example.snackordering


import android.content.Context

import android.content.Intent

import android.os.Bundle

import androidx.activity.ComponentActivity

import androidx.activity.compose.setContent

import androidx.compose.foundation.Image

import androidx.compose.foundation.layout.*

import androidx.compose.material.*
import androidx.compose.runtime.*

import androidx.compose.ui.Alignment

import androidx.compose.ui.Modifier

import androidx.compose.ui.graphics.Color

import androidx.compose.ui.layout.ContentScale

import androidx.compose.ui.res.painterResource

import androidx.compose.ui.text.font.FontFamily

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

import androidx.compose.ui.unit.dp

import androidx.compose.ui.unit.sp

import androidx.core.content.ContextCompat

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

```
class LoginActivity : ComponentActivity() {

 private lateinit var databaseHelper: UserDatabaseHelper

override fun onCreate(savedInstanceState: Bundle?) {

 super.onCreate(savedInstanceState)

 databaseHelper = UserDatabaseHelper(this)

 setContent {

 SnackOrderingTheme {

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

Surface(

 modifier = Modifier.fillMaxSize(),

 color = MaterialTheme.colors.background
 ) {

 LoginScreen(this, databaseHelper)

 }

 }

 }

 }

}

@Composable

fun LoginScreen(context: Context, databaseHelper: UserDatabaseHelper) {



 Image(painterResource(id = R.drawable.order), contentDescription = "",
alpha =0.3F,

 contentScale = ContentScale.FillHeight,


 )


 var username by remember { mutableStateOf("") }

 var password by remember { mutableStateOf("") }

 var error by remember { mutableStateOf("") }


 Column(

 modifier = Modifier.fillMaxSize(),

 horizontalAlignment = Alignment.CenterHorizontally,

verticalArrangement = Arrangement.Center

 ) {


 Text(

 fontSize = 36.sp,

 fontWeight = FontWeight.ExtraBold,

fontFamily = FontFamily.Cursive,   color =

Color.White,

 text = "Login"

 )

 Spacer(modifier = Modifier.height(10.dp))
 TextField(
 value = username,
 onValueChange = { username = it },
label = { Text("Username") },   modifier =
Modifier.padding(10.dp)   .width(280.dp)
 )

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

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

 onClick = {

 if (username.isNotEmpty() && password.isNotEmpty()) {   val user =

databaseHelper.getUserByUsername(username)   if (user != null &&

user.password == password) {   error = "Successfully log in"

 context.startActivity(

 Intent(

 context,

 MainPage::class.java

 )

 )

 //onLoginSuccess()
 }

 if (user != null && user.password == "admin") {   error =

"Successfully log in"

 context.startActivity(

 Intent(

 context,

 AdminActivity::class.java   )

 )

 }

 else {
```

```
error = "Invalid username or password"   }



} else {

error = "Please fill all fields"

}

},

modifier = Modifier.padding(top = 16.dp)   ) {

Text(text = "Login")

}

Row {

TextButton(onClick = {context.startActivity(
Intent(

context,

MainActivity::class.java

)

)}

)

{ Text(color = Color.White,text = "Sign up") }

TextButton(onClick = {

})



{

Spacer(modifier = Modifier.width(60.dp))   Text(color =
Color.White,text = "Forget password?")   }

 }

 }

}

private fun startMainPage(context: Context) {

 val intent = Intent(context, MainPage::class.java)

ContextCompat.startActivity(context, intent, null)  }
```

MainPage.kt

```
package com.example.snackordering


import android.annotation.SuppressLint

import android.content.Context

import android.os.Bundle

import android.widget.Toast

import androidx.activity.ComponentActivity

import androidx.activity.compose.setContent

import androidx.annotation.DrawableRes

import androidx.annotation.StringRes

import androidx.compose.foundation.Image

import

androidx.compose.foundation.background

import androidx.compose.foundation.layout.*

import androidx.compose.foundation.shape.CircleShape  import
```

```
androidx.compose.foundation.shape.RoundedCornerShape

import androidx.compose.material.*

import androidx.compose.material.icons.Icons

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

import androidx.compose.runtime.Composable

import androidx.compose.ui.Alignment

import androidx.compose.ui.Modifier

import androidx.compose.ui.draw.clip

import androidx.compose.ui.graphics.Color
import androidx.compose.foundation.lazy.LazyColumn

import androidx.compose.foundation.lazy.items

import androidx.compose.material.Text

import androidx.compose.ui.unit.dp

import androidx.compose.ui.graphics.RectangleShape

import androidx.compose.ui.layout.ContentScale

import androidx.compose.ui.platform.LocalContext

import androidx.compose.ui.res.painterResource

import androidx.compose.ui.res.stringResource

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

import androidx.compose.ui.unit.sp

import androidx.core.content.ContextCompat.startActivity  import

com.example.snackordering.ui.theme.SnackOrderingTheme


import android.content.Intent as Intent1
```

```
class MainPage : ComponentActivity() {

 override fun onCreate(savedInstanceState: Bundle?) {

 super.onCreate(savedInstanceState)

 setContent {

 SnackOrderingTheme {

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

Surface(

 modifier = Modifier.fillMaxSize(),

 color = MaterialTheme.colors.background

 ) {

 FinalView(this)

 val context = LocalContext.current

 //PopularFoodColumn(context)

 }

 }

 }

 }

}



@Composable
```

```
fun TopPart() {

 Row(
 modifier = Modifier
 .fillMaxWidth()
 .background(Color(0xffeceef0)), Arrangement.SpaceBetween   ) {
 Icon(
 imageVector = Icons.Default.Add, contentDescription = "Menu Icon",
Modifier
 .clip(CircleShape)
 .size(40.dp),
 tint = Color.Black,
 )
 Column(horizontalAlignment = Alignment.CenterHorizontally) {
 Text(text = "Location", style = MaterialTheme.typography.subtitle1, color = Color.Black)   Row
{
 Icon(
 imageVector = Icons.Default.LocationOn,
 contentDescription = "Location",
 tint = Color.Red,
 )
 Text(text = "Tamil Nadu" , 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 = "Rs.200",
 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)

 }

 }

}
```

Order.kt

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

```
import androidx.room.ColumnInfo

import androidx.room.Entity

import androidx.room.PrimaryKey


@Entity(tableName = "order_table")

data class Order(

 @PrimaryKey(autoGenerate = true) val id: Int?,

 @ColumnInfo(name = "quantity") val quantity: String?,

@ColumnInfo(name = "address") val address: String?,  )
```

OrderDao.kt

```
package com.example.snackordering


import androidx.room.*


@Dao

interface OrderDao {


 @Query("SELECT * FROM order_table WHERE address= :address")

suspend fun getOrderByAddress(address: String): Order?

 @Insert(onConflict = OnConflictStrategy.REPLACE)

suspend fun insertOrder(order: Order)
```

```
 @Update

 suspend fun updateOrder(order: Order)


 @Delete

 suspend fun deleteOrder(order: Order)

}
```

OrderDatabase.kt

```
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
 }
 }
 }
}
```

RegisterActivity.kt

```
package com.example.snackordering


import android.content.Context
import android.content.Intent
import android.os.Bundle
import androidx.activity.ComponentActivity
import
androidx.activity.compose.setContent
import androidx.compose.foundation.Image
import androidx.compose.foundation.layout.*

import androidx.compose.material.*

import androidx.compose.runtime.*

import androidx.compose.ui.Alignment

import androidx.compose.ui.Modifier

import androidx.compose.ui.graphics.Color

import androidx.compose.ui.layout.ContentScale

import androidx.compose.ui.res.painterResource

import androidx.compose.ui.text.font.FontFamily

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

import androidx.compose.ui.unit.dp

import androidx.compose.ui.unit.sp

import androidx.core.content.ContextCompat

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


class MainActivity : ComponentActivity() {

 private lateinit var databaseHelper: UserDatabaseHelper

override fun onCreate(savedInstanceState: Bundle?) {

 super.onCreate(savedInstanceState)

 databaseHelper = UserDatabaseHelper(this)

 setContent {

 SnackOrderingTheme {

 // A surface container using the 'background' color from the theme
 Surface(
 modifier = Modifier.fillMaxSize(),

 color = MaterialTheme.colors.background

 ) {


 RegistrationScreen(this,databaseHelper)

 }

 }

 }

 }

}



@Composable

fun RegistrationScreen(context: Context, databaseHelper: UserDatabaseHelper) {


 Image(

 painterResource(id = R.drawable.order), contentDescription = "",

alpha =0.3F,

 contentScale = ContentScale.FillHeight,


 )


 var username by remember { mutableStateOf("") }
 var password by remember { mutableStateOf("") }

var email by remember { mutableStateOf("") }   var

error by remember { mutableStateOf("") }


 Column(

 modifier = Modifier.fillMaxSize(),

 horizontalAlignment = Alignment.CenterHorizontally,

verticalArrangement = Arrangement.Center   ) {


 Text(

 fontSize = 36.sp,

 fontWeight = FontWeight.ExtraBold,

fontFamily = FontFamily.Cursive,

 color = Color.White,

 text = "Register"

 )


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

TextField(

 value = username,

 onValueChange = { username = it },

 label = { Text("Username") },

 modifier = Modifier
 .padding(10.dp)
 .width(280.dp)

 )

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

 TextField(
 value = password,
 onValueChange = { password = it },
label = { Text("Password") },   modifier =
Modifier
 .padding(10.dp)
 .width(280.dp)
 )
 if (error.isNotEmpty()) {
 Text(
 text = error,
 color = MaterialTheme.colors.error,

 modifier = Modifier.padding(vertical = 16.dp)

 )

 }


 Button(

 onClick = {

 if (username.isNotEmpty() && password.isNotEmpty() && email.isNotEmpty()) {   val

user = User(

 id = null,

 firstName = username,

 lastName = null,

 email = email,

 password = password

 )

 databaseHelper.insertUser(user)

 error = "User registered successfully"

 // Start LoginActivity using the current context

 context.startActivity(

 Intent(

 context,
 LoginActivity::class.java

 )

 )
 } else {

 error = "Please fill all fields"

 }

 },

 modifier = Modifier.padding(top = 16.dp)

 ) {

 Text(text = "Register")

 }

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

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


 Row() {

 Text(

 modifier = Modifier.padding(top = 14.dp), text = "Have an account?"   )

 TextButton(onClick = {

 context.startActivity(

 Intent(

 context,

 LoginActivity::class.java
 )

 )

 })
```

```
 {

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

Text(text = "Log in")

 }

 }

 }

}

private fun startLoginActivity(context: Context) {   val

intent = Intent(context, LoginActivity::class.java)

ContextCompat.startActivity(context, intent, null)  }
```

OrderDatabaseHelper.kt

```
package com.example.snackordering



import android.annotation.SuppressLint

import android.content.ContentValues

import android.content.Context

import android.database.Cursor

import android.database.sqlite.SQLiteDatabase
import android.database.sqlite.SQLiteOpenHelper



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


 companion object {

 private const val DATABASE_VERSION = 1

 private const val DATABASE_NAME = "OrderDatabase.db"


 private const val TABLE_NAME = "order_table"

 private const val COLUMN_ID = "id"

 private const val COLUMN_QUANTITY = "quantity"

 private const val COLUMN_ADDRESS = "address"

 }


 override fun onCreate(db: SQLiteDatabase?) {

 val createTable = "CREATE TABLE $TABLE_NAME (" +   "${COLUMN_ID}

INTEGER PRIMARY KEY AUTOINCREMENT, " +   "${COLUMN_QUANTITY}

Text, " +

 "${COLUMN_ADDRESS} TEXT " +

 ")"


 db?.execSQL(createTable)

 }
 override fun onUpgrade(db: SQLiteDatabase?, oldVersion: Int, newVersion: Int) {

db?.execSQL("DROP TABLE IF EXISTS $TABLE_NAME")

 onCreate(db)
 }


 fun insertOrder(order: Order) {

 val db = writableDatabase

 val values = ContentValues()

 values.put(COLUMN_QUANTITY, order.quantity)

 values.put(COLUMN_ADDRESS, order.address)

 db.insert(TABLE_NAME, null, values)

 db.close()

 }




 @SuppressLint("Range")

 fun getOrderByQuantity(quantity: String): Order? {

 val db = readableDatabase

 val cursor: Cursor = db.rawQuery("SELECT * FROM $TABLE_NAME WHERE
$COLUMN_QUANTITY = ?", arrayOf(quantity))

 var order: Order? = null

 if (cursor.moveToFirst()) {

 order = Order(
 id = cursor.getInt(cursor.getColumnIndex(COLUMN_ID)),

 quantity = cursor.getString(cursor.getColumnIndex(COLUMN_QUANTITY)),

address = cursor.getString(cursor.getColumnIndex(COLUMN_ADDRESS)),   )
```

```
 }

 cursor.close()

 db.close()

 return order

 }

 @SuppressLint("Range")

 fun getOrderById(id: Int): Order? {

 val db = readableDatabase

 val cursor: Cursor = db.rawQuery("SELECT * FROM $TABLE_NAME WHERE  $COLUMN_ID =
?", arrayOf(id.toString()))

 var order: Order? = null

 if (cursor.moveToFirst()) {

 order = Order(

 id = cursor.getInt(cursor.getColumnIndex(COLUMN_ID)),

 quantity = cursor.getString(cursor.getColumnIndex(COLUMN_QUANTITY)),

address = cursor.getString(cursor.getColumnIndex(COLUMN_ADDRESS)),   )

 }

 cursor.close()

 db.close()

 return order
 }



 @SuppressLint("Range")

 fun getAllOrders(): List<Order> {

 val orders = mutableListOf<Order>()
 val db = readableDatabase

 val cursor: Cursor = db.rawQuery("SELECT * FROM $TABLE_NAME", null)   if

(cursor.moveToFirst()) {

 do {

 val order = Order(

 id = cursor.getInt(cursor.getColumnIndex(COLUMN_ID)),   quantity =

cursor.getString(cursor.getColumnIndex(COLUMN_QUANTITY)),   address =

cursor.getString(cursor.getColumnIndex(COLUMN_ADDRESS)),   )

 orders.add(order)

 } while (cursor.moveToNext())

 }

 cursor.close()

 db.close()

 return orders

 }


}
```

Build.gradle

```
plugins {

 id 'com.android.application'

 id 'org.jetbrains.kotlin.android'

}
```

```
android {

 namespace 'com.example.snackordering'

 compileSdk 33


 defaultConfig {

 applicationId "com.example.snackordering"

 minSdk 24

 targetSdk 33

 versionCode 1

 versionName "1.0"


 testInstrumentationRunner "androidx.test.runner.AndroidJUnitRunner"

vectorDrawables {

 useSupportLibrary true

 }

 }


 buildTypes {
 release {

 minifyEnabled false

 proguardFiles getDefaultProguardFile('proguard-android-optimize.txt'), 'proguard rules.pro'

 }

 }
 compileOptions {

 sourceCompatibility JavaVersion.VERSION_1_8

 targetCompatibility JavaVersion.VERSION_1_8

 }

 kotlinOptions {

 jvmTarget = '1.8'

 }

 buildFeatures {

 compose true

 }

 composeOptions {

 kotlinCompilerExtensionVersion '1.2.0'

 }

 packagingOptions {

 resources {

 excludes += '/META-INF/{AL2.0,LGPL2.1}'

 }

 }

}
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"


 implementation 'androidx.room:room-common:2.5.0'

 implementation 'androidx.room:room-ktx:2.5.0'

}
```