# NAANMUDHALVAN SCHEME – 2023

## Snack Squad A Customizable Snack Ordering and Delivery App

A major project report submitted to Naan Mudhalvan **Scheme – 2023** in partial fulfilment of the requirement for the award of Degree Bachelor of Science in Computer Science.

**SUBMITTED BY**


**TEAM LEADER:**

**C.ESAKKIRAJA – ASMSU2022010804(NANMUDHALVAN ID)**

**DEPARTMENT OF COMPUTER SCIENCE**

**ADITANAR COLLEGE OF ARTS AND SCIENCE**

**VIRAPANDIANPATNAM - 628216**

# PROJECT REPORT TEMPLATE

# 1.INTRODUCTION

## 1.1) Overview

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.

As industries are fast expanding, people are seeking for more ways to purchase products with much ease and still maintain cost effectiveness.

The vendors need to purchase the products in order to sell to end users.

The manual method of going to their local food sales outlets to purchase food is becoming obsolete and more tasking.

Food can be ordered through the internet and payment made without going to the restaurant or the food vendor.

Until recently, most of this delivery orders were placed over the phone, but there are many disadvantages to this system.

## 1.2) Purpose

Due to time and financial constraints, the software that is developed covers only the aspect of food ordering and payments.

**FOOD:** Any nutritious substance that people or animals eat or drink, or that plant absorbs, in order to maintain life and growth.

**MENU:** A list of dishes available in a restaurant or the food available or to be served in a restaurant or at a meal for example "a dinner-party menu", "politics and sport are on the menu tonight".

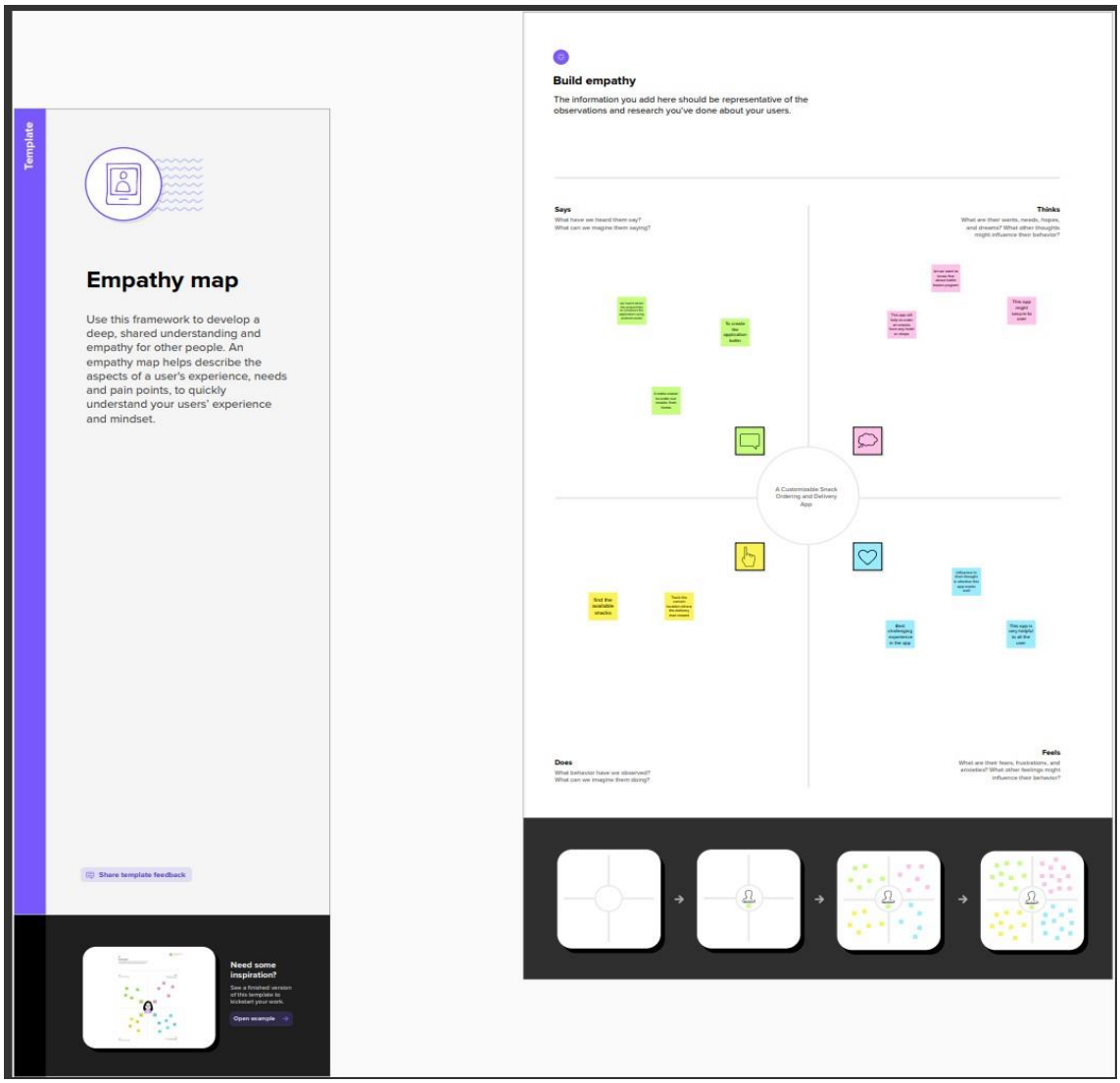
**ONLINE FOOD ORDERING:** Online food ordering services are websites that 5 feature interactive menus allowing customers to place orders with local restaurants and food cooperatives

**RESTAURANT:** (eating place) is a place where meals and drinks are sold and served to customers.

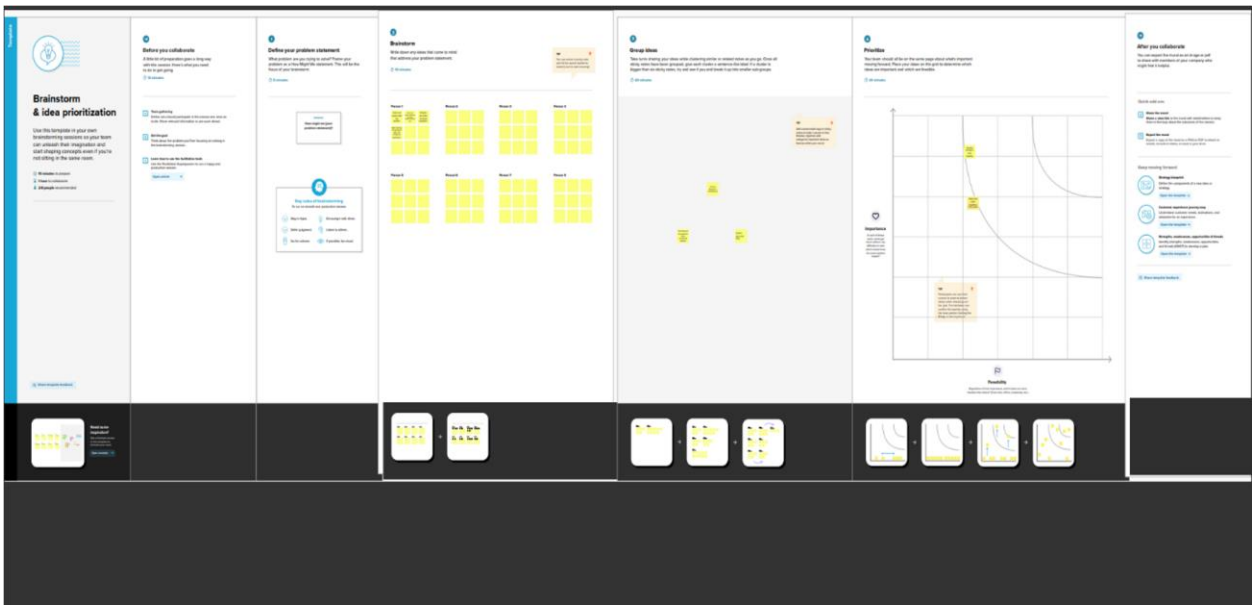
**CUSTOMER**: Sometimes known as a client, buyer, or purchaser) is the recipient of goods, services, products or idea obtained from a seller, vendor, or supplier for a monetary or other valuable consideration.

# 2.Problem Definition & Design Thinking

# 2.1) Empathy Map

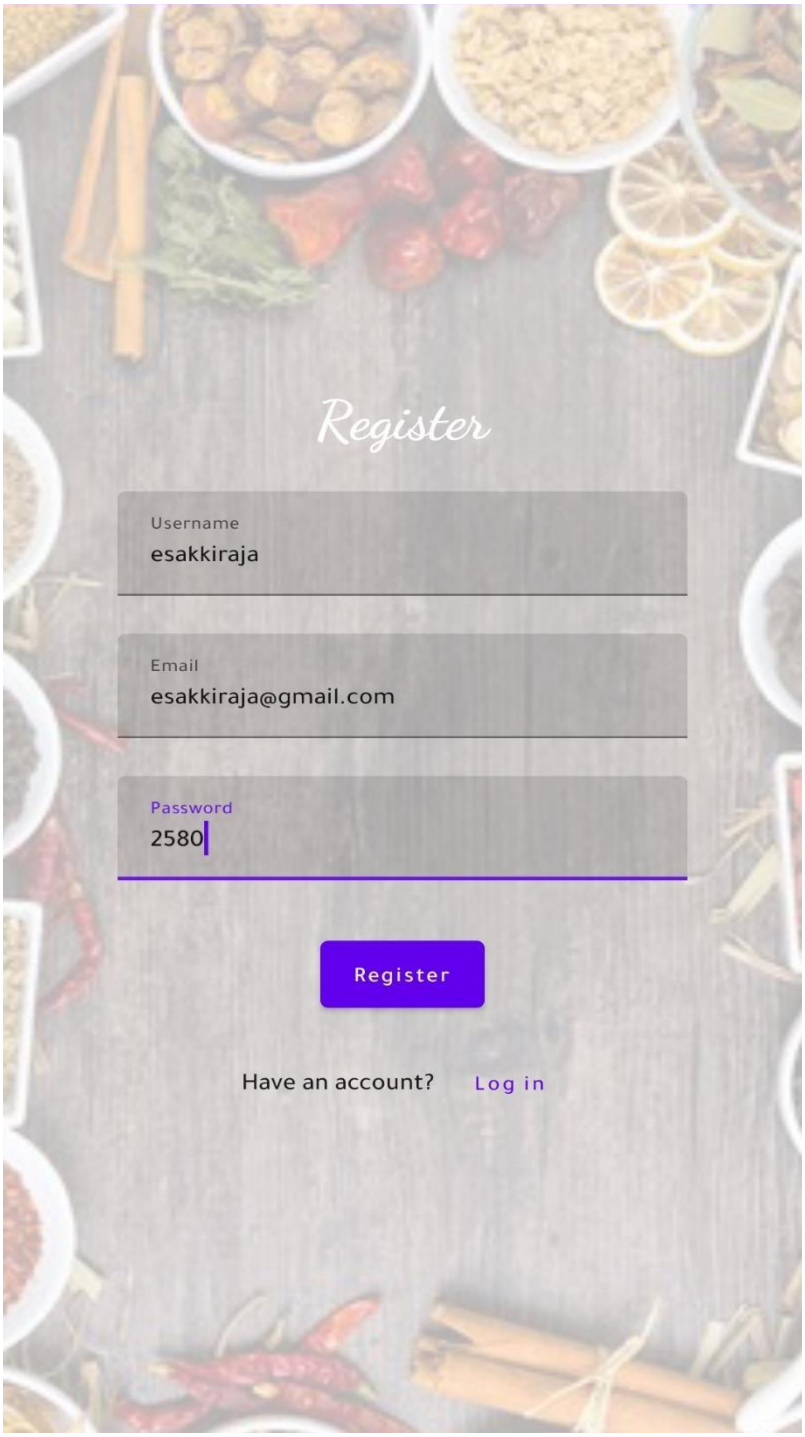


# 2.2) Ideation & Brainstorming Map

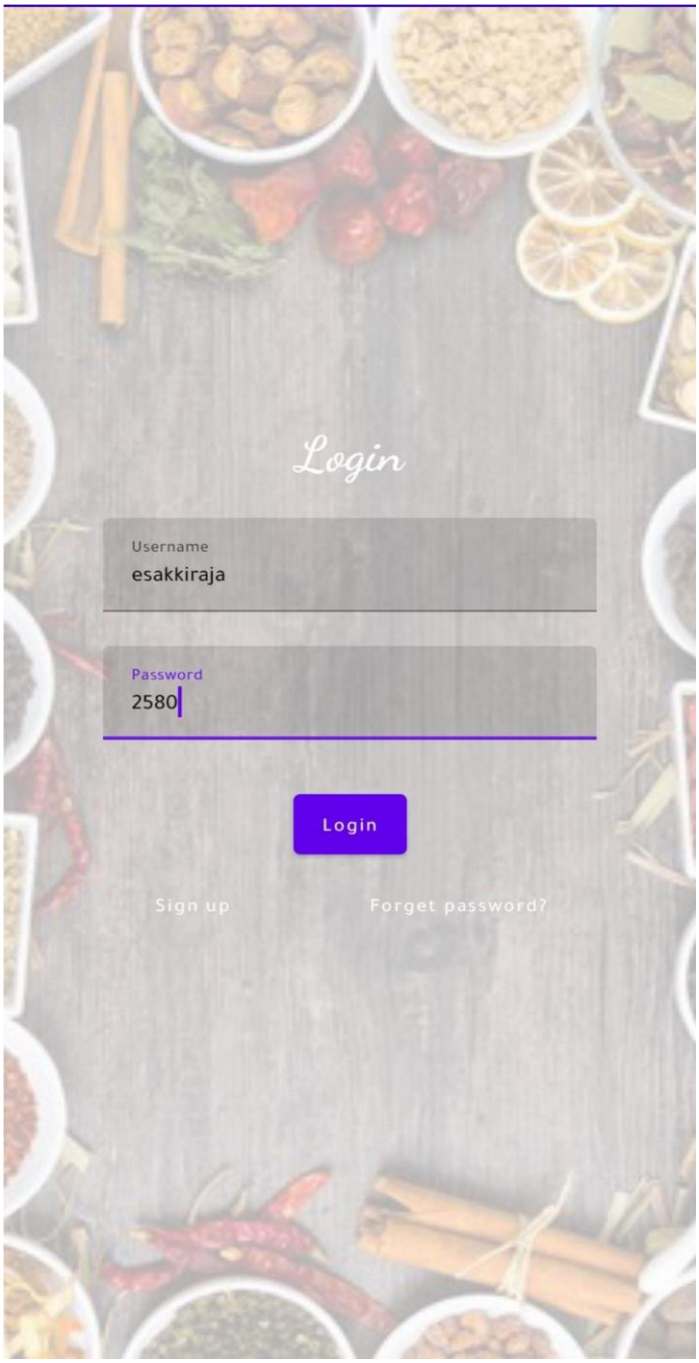


## 3.RESULT

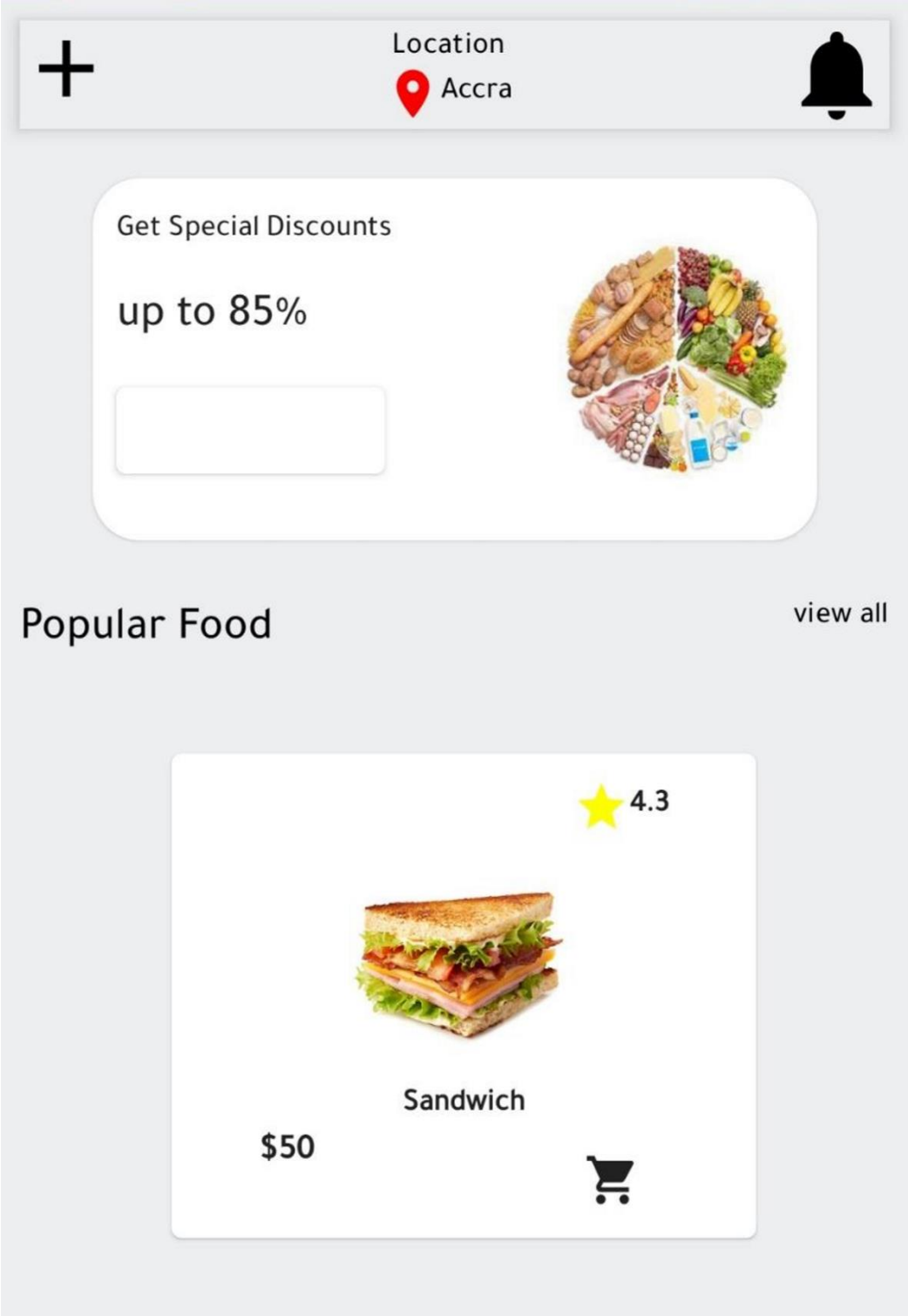
### OUTPUT 1



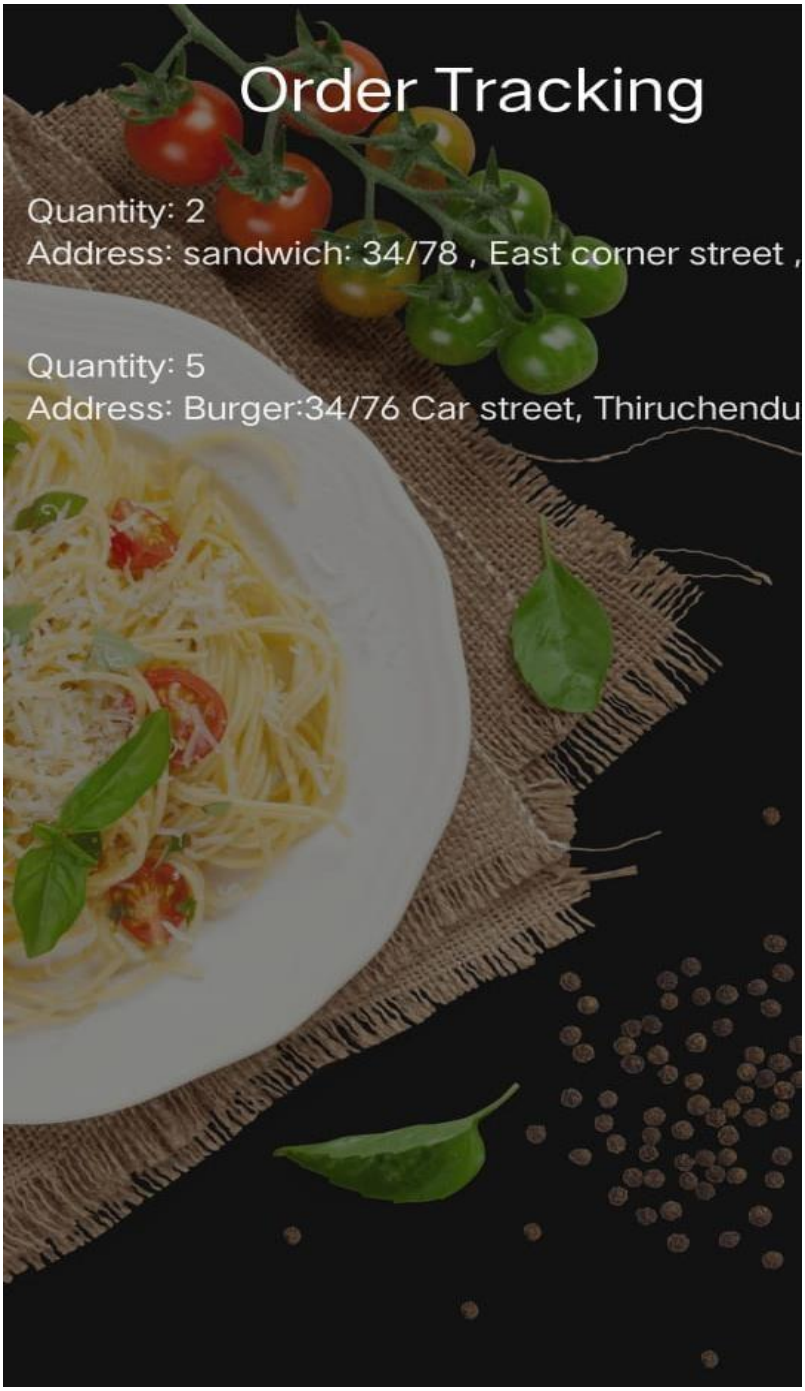
# OUTPUT 2


Login
Username
esakkiraja
Password
2580
Login
Sign up
Forget password?

# OUTPUT 3


Location
Accra
Get Special Discounts
up to 85%
Popular Food
view all
4.3
Sandwich
$50

# OUTPUT 4


Order Tracking
Quantity: 2
Address: sandwich: 34/78 , East corner street ,
Quantity: 5
Address: Burger:34/76 Car street, Thiruchendu

# 4.ADVANTAGES AND DISADVANTAGES

## ADVANTAGES

**Staffing cost is reduced:**

Customers can place an order from their mobile so that instead of an employee answering phone calls all the time. Once the restaurant receives the order, they prepare the food, and employees can spend their time in profitable ways.

**More sales and revenue:**

You are making your restaurant more competitive by making ordering more convenient. So that you can increase your customer base and boost your sales.

**Risk of disease transmission is low:**

Online food ordering systems make it easy to stay safe for both customers and staff. Customers can simply pick up their order from the pickup area within the restaurant and make a payment without any contact.

**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.

## DISADVANTAGES

**Delivery men in danger:**
Delivery men deliver the food, if it is sunny or rainy, he is waiting outside the restaurant to take the order and deliver your order on time.

**Health issues:**
The attractive dishes sometimes make health issues due to their ingredients, and the hot food packed in plastic bags or boxes leads to health issues. If you get this type of food on regular basis, it may cause food poisoning and makes you obese too.

# 5.Application

Food delivery apps have become a staple in the tech industry, with their popularity and market size continuing to grow.

Our blog delves into the industry's current state, including market size, growth, consumer trends, and future projections.

It offers a comprehensive look at the key players and their strategies for success, providing valuable insights for entrepreneurs looking to enter or expand the market. These food ordering and delivery apps are some of the market's most successful food delivery apps.

# 6.Conclusion

An online food ordering system is developed where the customers can make an order for the food and avoid the hassles of waiting for the order to be taken by the waiter.

Using the application, the end users register online, read the E-menu card and select the food from the e-menu card to order food online.

Once the customer selects the required food item the chef will be able to see the results on the screen and start processing the food.

This application nullifies the need of a waiter or reduces the workload of the waiter.

The advantage is that in a crowded restaurant there will be chances that the waiters are overloaded with orders and they are unable to meet the requirements of the customer in a satisfactory manner.

Therefore, by using this application, the users can directly place the order for food to the chef online.

# 7.Future Scope

- order product online
- upload product design online
- add, edit, delete product
- send order confirmation via email
- manage online order
- Ajax hierarchical combo box for payment method.
- add delivery charge outside the coverage area
- secure reservation
- forum for customer comments about the site
- generates various report
- and many more

# 8.APPENDIX

**SOURCE CODE:**

**https://github.com/esakkirajac/naanmudhalavan**

**CODING:**

**User.kt**

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

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

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

## UserDao.kt

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

```
import androidx.room.*

@Dao
interface UserDao {

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

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

    @Update
    suspend fun updateUser(user: User)

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

# UserDatabase.kt

```
package com.example.snackordering
 import
android.content.Context import
androidx.room.Database import
androidx.room.Room
import androidx.room.RoomDatabase

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

    abstract fun userDao(): UserDao

    companion object {

        @Volatile
        private var instance: UserDatabase? = null

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

```
                instance = newInstance
newInstance
            }
        }
    }
}
```

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

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

        private const val TABLE_NAME = "user_table"
private const val COLUMN_ID = "id"
        private const val COLUMN_FIRST_NAME = "first_name"
private const val COLUMN_LAST_NAME = "last_name"
private const val COLUMN_EMAIL = "email"         private
const val COLUMN_PASSWORD = "password"     }
    override fun onCreate(db: SQLiteDatabase?) {
        val createTable = "CREATE TABLE $TABLE_NAME (" +
                "$COLUMN_ID INTEGER PRIMARY KEY AUTOINCREMENT, " +
                "$COLUMN_FIRST_NAME TEXT, " +
                "$COLUMN_LAST_NAME TEXT, " +
                "$COLUMN_EMAIL TEXT, " +
                "$COLUMN_PASSWORD TEXT" +
                ")"

        db?.execSQL(createTable)
    }         override  fun  onUpgrade(db:  SQLiteDatabase?,  oldVersion:  Int,
newVersion: Int) {
```

```
        db?.execSQL("DROP TABLE IF EXISTS $TABLE_NAME")
onCreate(db)
    }
    fun insertUser(user: User) {
val db = writableDatabase
val values = ContentValues()
        values.put(COLUMN_FIRST_NAME, user.firstName)
values.put(COLUMN_LAST_NAME, user.lastName)
values.put(COLUMN_EMAIL, user.email)
values.put(COLUMN_PASSWORD, user.password)
        db.insert(TABLE_NAME, null, values)
db.close()
    }

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

```
cursor.getString(cursor.getColumnIndex(COLUMN_PASSWORD)),
            )
}
        cursor.close()
db.close()
return user
    }

    @SuppressLint("Range")     fun
getAllUsers(): List<User> {         val
users = mutableListOf<User>()         val
db = readableDatabase
        val cursor: Cursor = db.rawQuery("SELECT * FROM $TABLE_NAME", null)
if (cursor.moveToFirst()) {
```

```
            do {
                val user = User(
                    id = cursor.getInt(cursor.getColumnIndex(COLUMN_ID)),
firstName =
cursor.getString(cursor.getColumnIndex(COLUMN_FIRST_NAME)),
lastName =
cursor.getString(cursor.getColumnIndex(COLUMN_LAST_NAME)),
email =
cursor.getString(cursor.getColumnIndex(COLUMN_EMAIL)),
password =
cursor.getString(cursor.getColumnIndex(COLUMN_PASSWORD)),
                )
                users.add(user)
            } while (cursor.moveToNext())
        }
        cursor.close()
db.close()
return users
    }

}
```

# 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)
}
```

# SurveyDatabase.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
            }
        }
    }
}
```

## SurveyDatabaseHelper.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
    }

}
```

# 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) }
```

# MainActivity.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)
}
```

# 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 = "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)
        }
    }
}
```

# 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)
}
```

# 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}")
                    }
                }
            }
        }

    }
}
```

# 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