PROJECT REPORT

Snack Squad : A Customizable Snack Ordering and Delivery App

1. INTRODUCTION

1.1 Overview

A snack ordering app is a mobile application that allows users to order snacks and have them delivered to their location. The app typically features a menu of snacks from various vendors or restaurants, along with prices and other relevant details. Users can browse the menu, select the snacks they want to order, and pay for their order through the app.

The app may also include features such as real-time order tracking, personalized recommendations based on previous orders, loyalty programs, and customer support. The app may also allow users to rate and review snacks and vendors, providing feedback to other users and helping to improve the overall quality of the app.

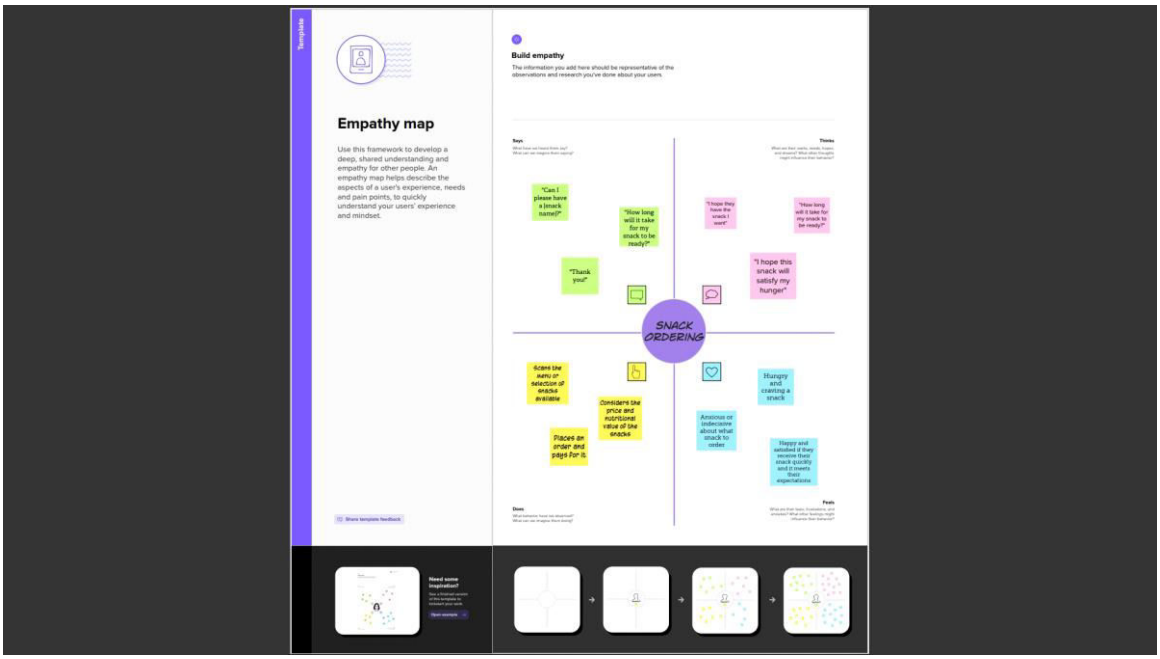
The app may be designed for use by individual consumers or may be targeted towards businesses or organizations, allowing them to order snacks in bulk for their employees or events. In either case, the app streamlines the ordering and delivery process, making it easier and more convenient for users to enjoy their favorite snacks.

1.2 Purpose

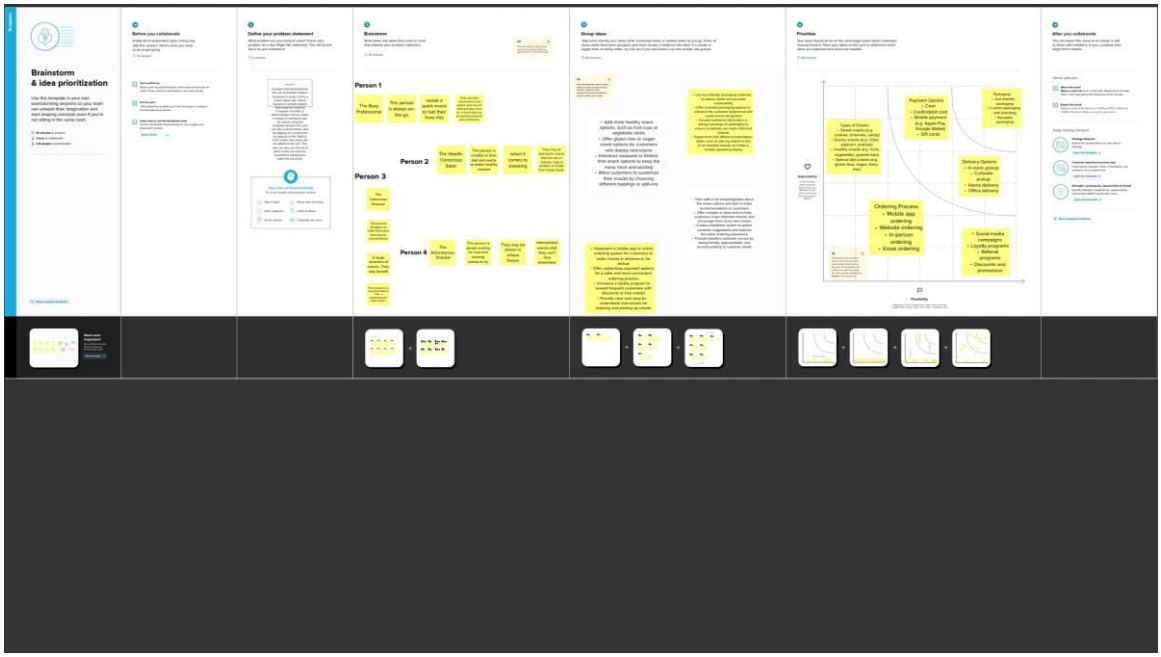
With a snack ordering app, users can browse through available snacks, select the items they want to order, and pay for their order using a secure payment system. The app may also provide features such as the ability to customize orders, view nutritional information, track orders in real-time, and receive notifications when orders are ready for pickup or delivery.

2. PROBLEM DEFINITION & DESIGN THINKING

2.1Empathy Map

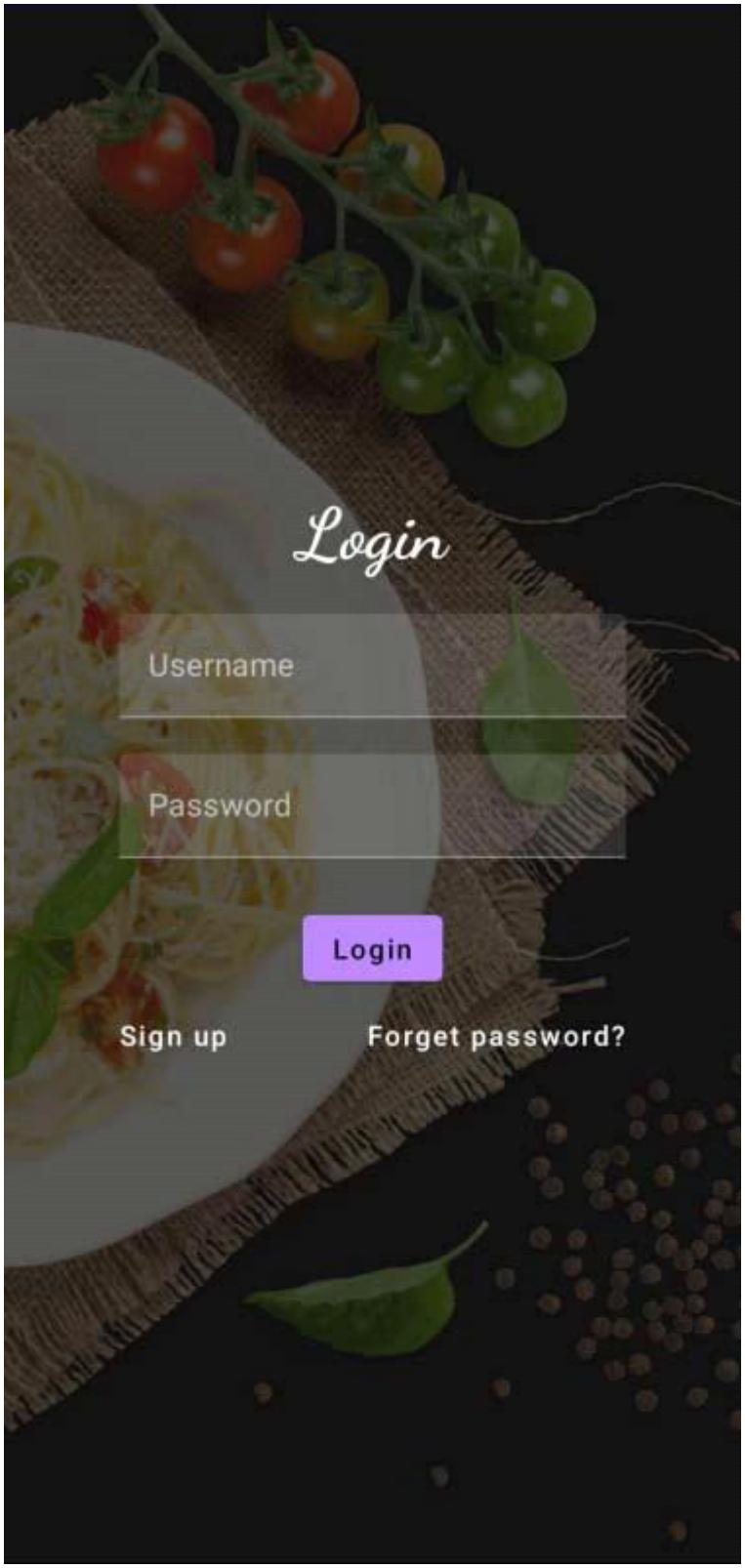


2.2 Ideation & Brainstorming Map

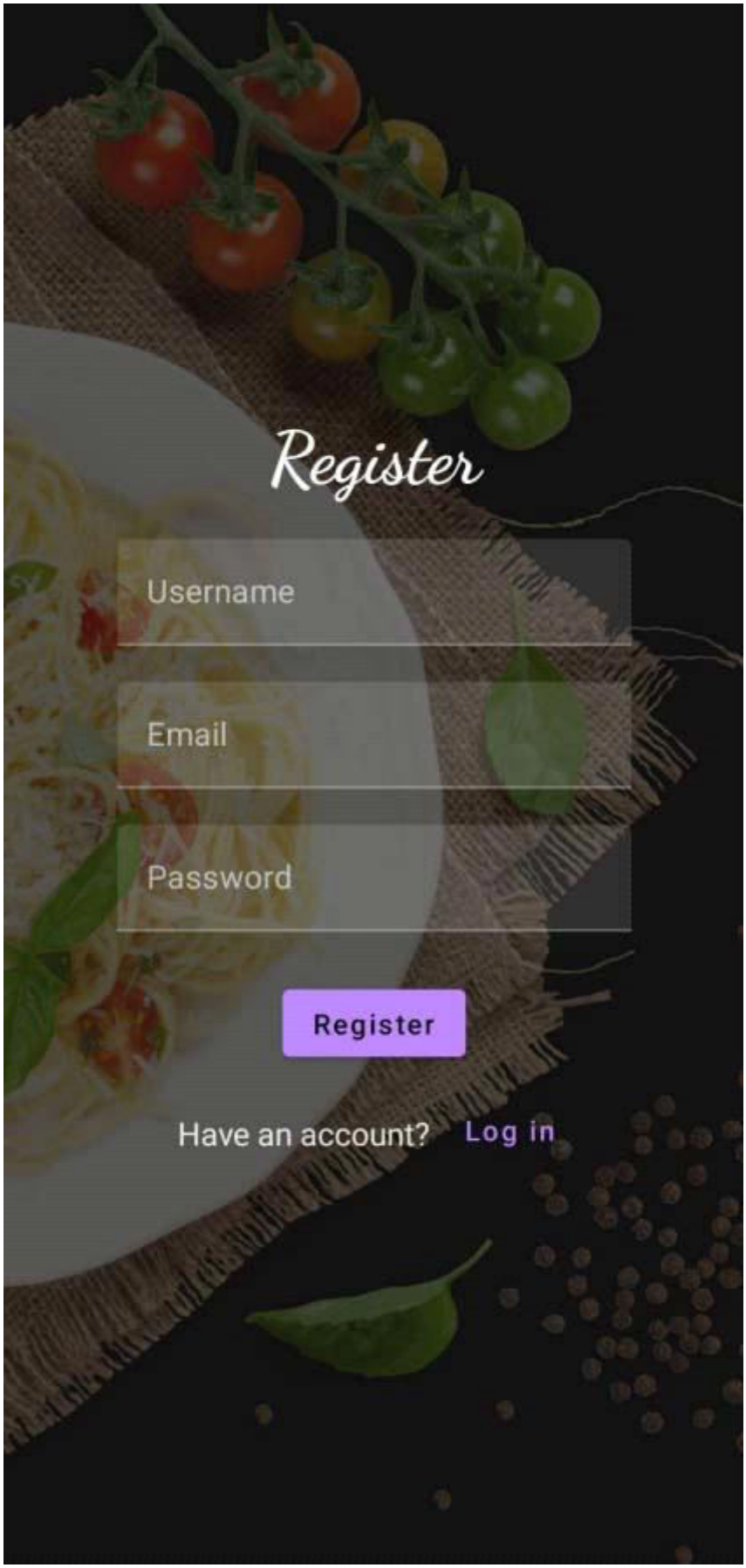


3. RESULT

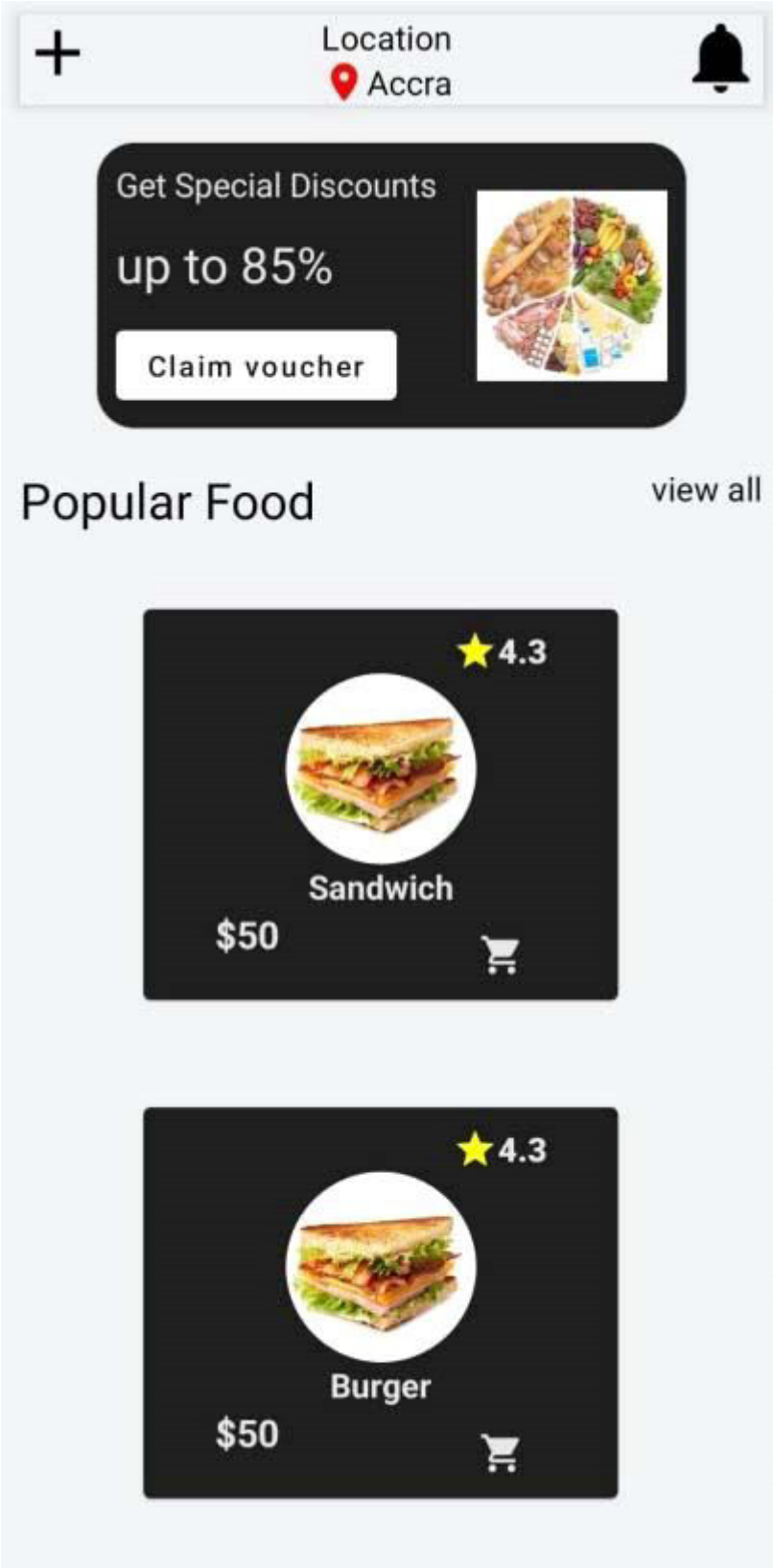
Login Page :



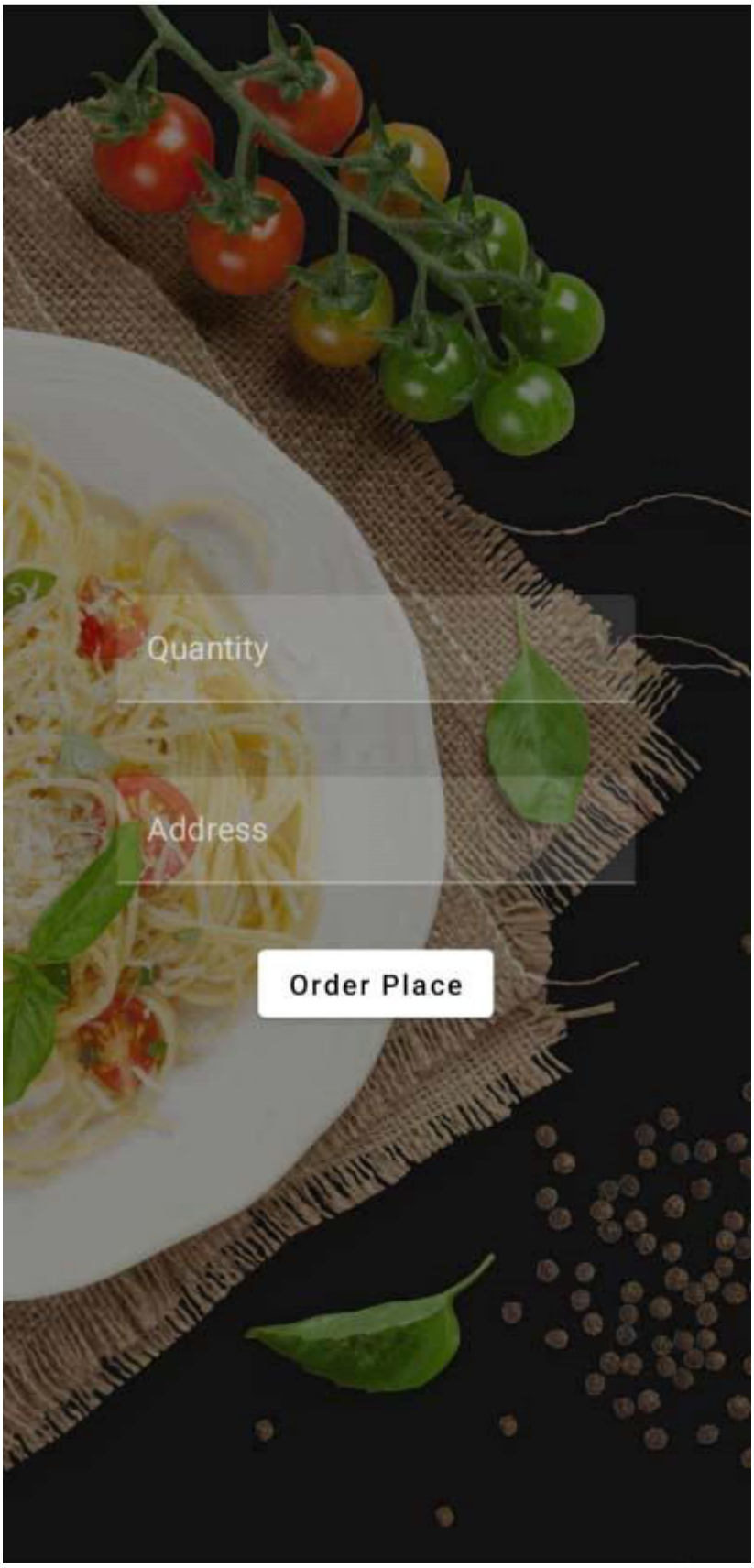
Register Page :


Register
Username
Email
Password
Register
Have an account?
Log in

Main Page :


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

Order Page:



4. ADVANTAGES & DISADVANTAGES

4.1 Advantages

- Easy to use
- Increases efficiency and accuracy of snack orders
- Reduces the need for manual processing, saving time and labor costs

4.2 Disadvantages

- Requires a reliable internet connection for customers to access the app
- May be less personal than ordering directly from a person

5. APPLICATIONS

The App can be used as personal app for individuals. The App is mainly made for order snack or food from our home, nowadays everything is online . The project will be useful to all the sections of the society.

6. CONCLUSION

In conclusion, a snack ordering app is a useful tool for businesses in the food industry to provide their customers with a fast and convenient way to order and pay for snacks. The app can be used in a variety of settings, such as cafes, food trucks,

vending machines, and office break rooms, and can help businesses to streamline their ordering process and increase sales.

7. FUTURE SCOPE

The future scope of snack ordering apps looks promising, as there is a growing trend towards digital transformation and the adoption of mobile technologies.

As voice assistants such as Amazon's Alexa and Google Assistant become more ubiquitous, snack ordering apps may integrate with these platforms to allow customers to place orders using voice commands.

8.APPENDIX

// 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)
}
```

// 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
            }
        }
    }
}
// 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
    }

}
```

// 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)
}

// 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)
```

```
}
```

```
// 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.xlm

```
<?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>
```