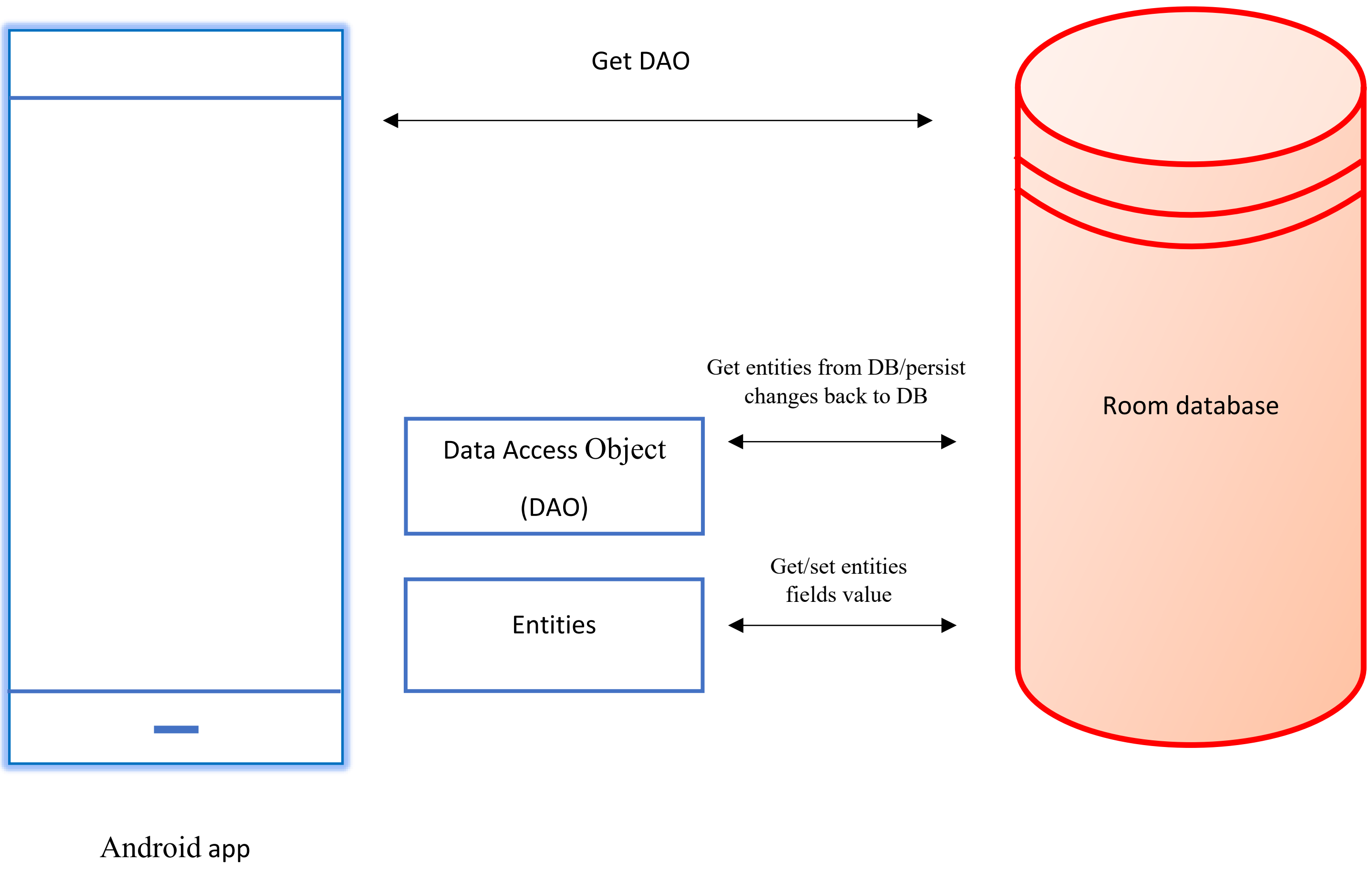
1.INTRODUCTION

1.1 overview

This project demonstrates the use of android jet pack compose to build a UI for a snack squad app. Snack squad is a sample project built using the android compose UI tool kit. It demonstrates how to create a simple E-commerce app for snacks using the compose libraries.

The user can see a list of snacks and by tapping on a snack and by tapping on the "ADD TO CART" button the snacks will be added to the cart. The user can also see the list of items in the cart and can proceed to check out to make the purchase.


Get DAO
Get entities from DB/persist changes back to DB
Data Access Object (DAO)
Get/set entities fields value
Entities
Room database
Android app

1.2 purpose

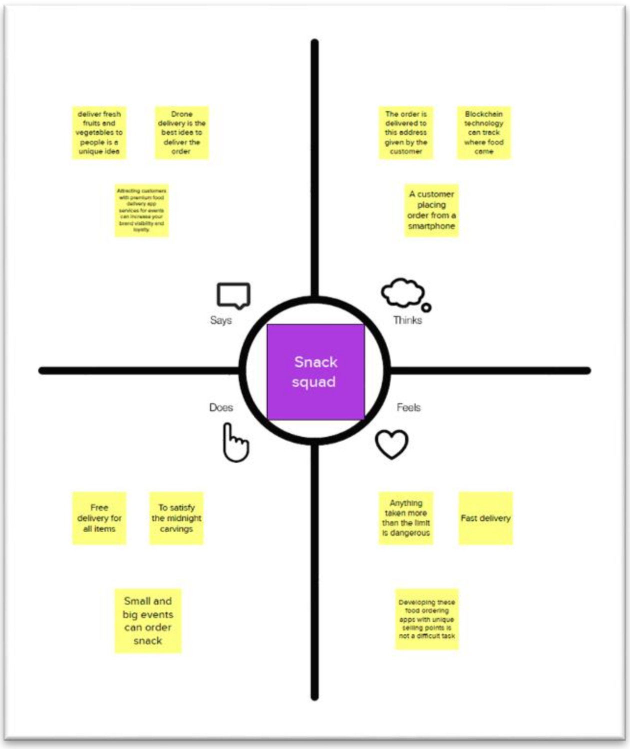
We have a handpicked selection of the best snack, sweets and drinks from around the world. Ready to be delivered straight to your door.

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

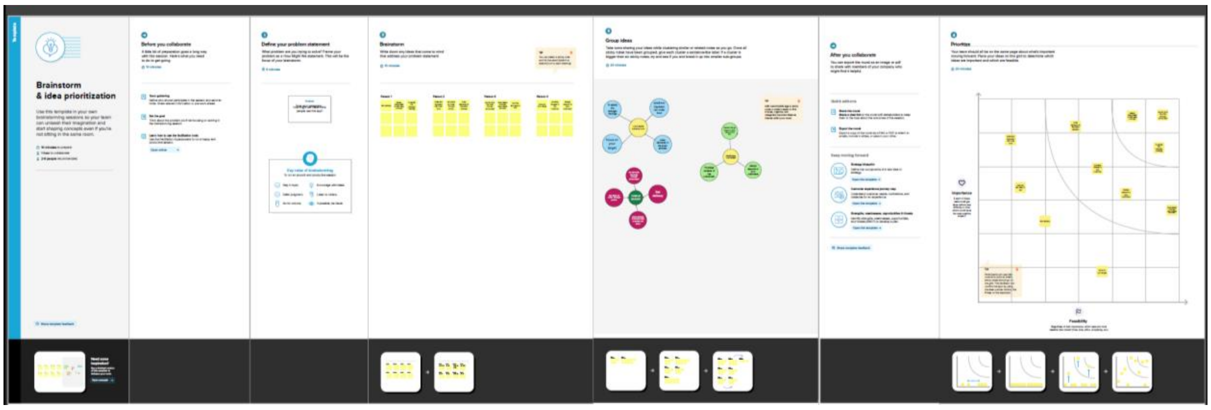
The purpose of online snack delivery system is to automate the existing manual system by the help of computerized equipment's and full-fledged computer software, fulfilling their requirements, so that their valuable data/information can be stored for a longer period with easy accessing and manipulation of the same. The required software and hardware are easily available and easy to work with. The online snacks ordering systems main purpose is to maintain track of information. Such item category, snack, delivery address, order and shopping cart. It keeps track of information about the item category, the customer, the shopping cart and the item category. Only the administrator gets access to the projects, because it is totally built at the administrative level. The project purpose is to develop software that will cut down on the time spent manually managing item category, customer, and delivery address. It saves the delivery address order and shopping cart information.

2.PROBLEM DEFINITION AND DESIGH THINKING

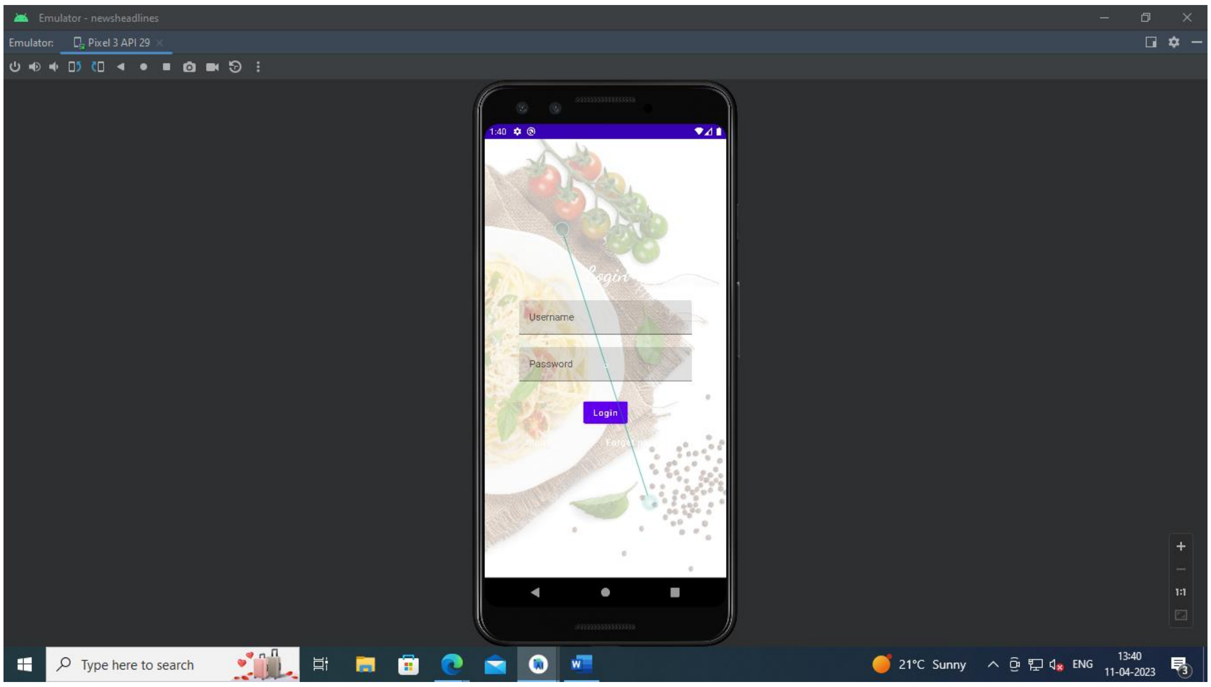
2.1 empathy map

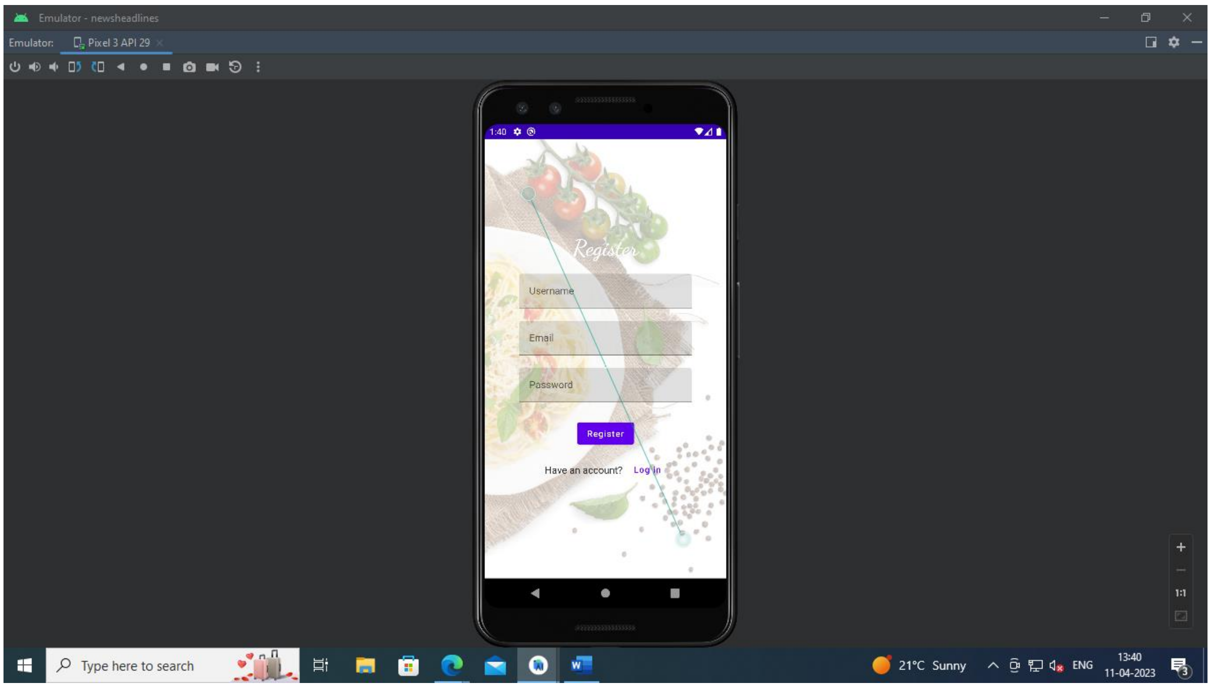

deliver fresh fruits and vegetables to people is a unique idea
Drone delivery is the best idea to deliver the order
Attracting customers with premium food delivery app services for events can increase your brand visibility and loyalty.
The order is delivered to this address given by the customer
Blockchain technology can track where food came
A customer placing order from a smartphone
Says
Thinks
Snack squad
Does
Feels
Free delivery for all items
To satisfy the midnight carvings
Small and big events can order snack
Anything taken more than the limit is dangerous
Fast delivery
Developing these food ordering apps with unique selling points is not a difficult task

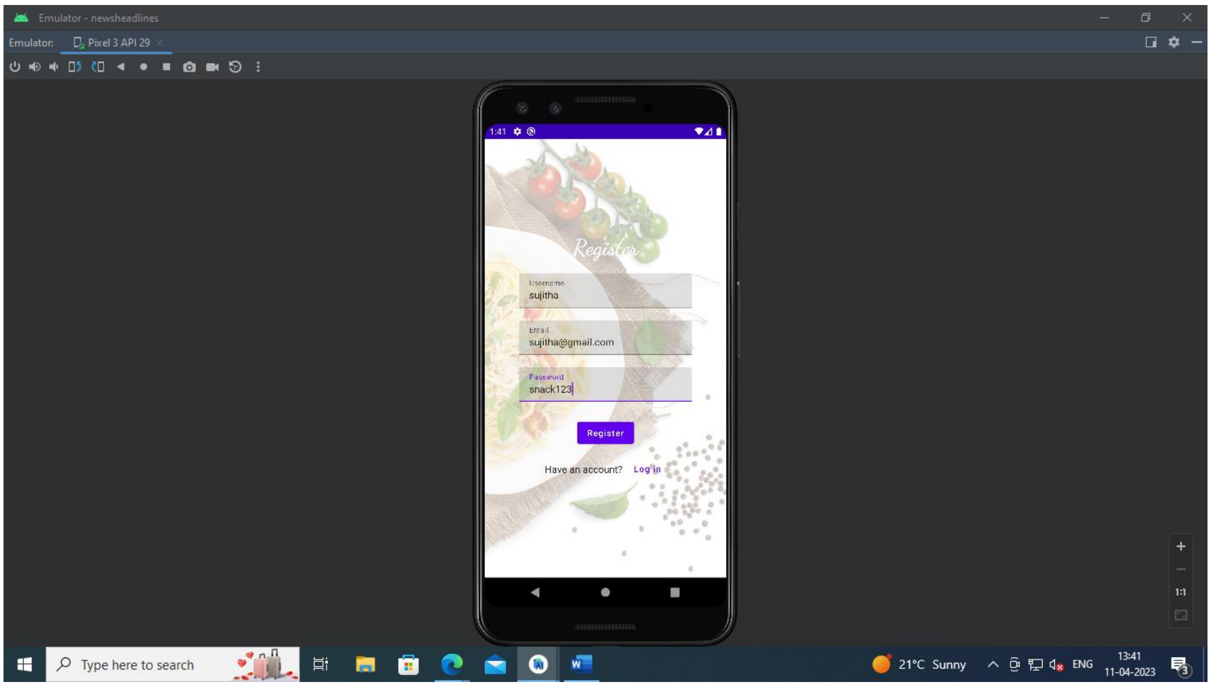
2.2 ideation and brainstorming map-



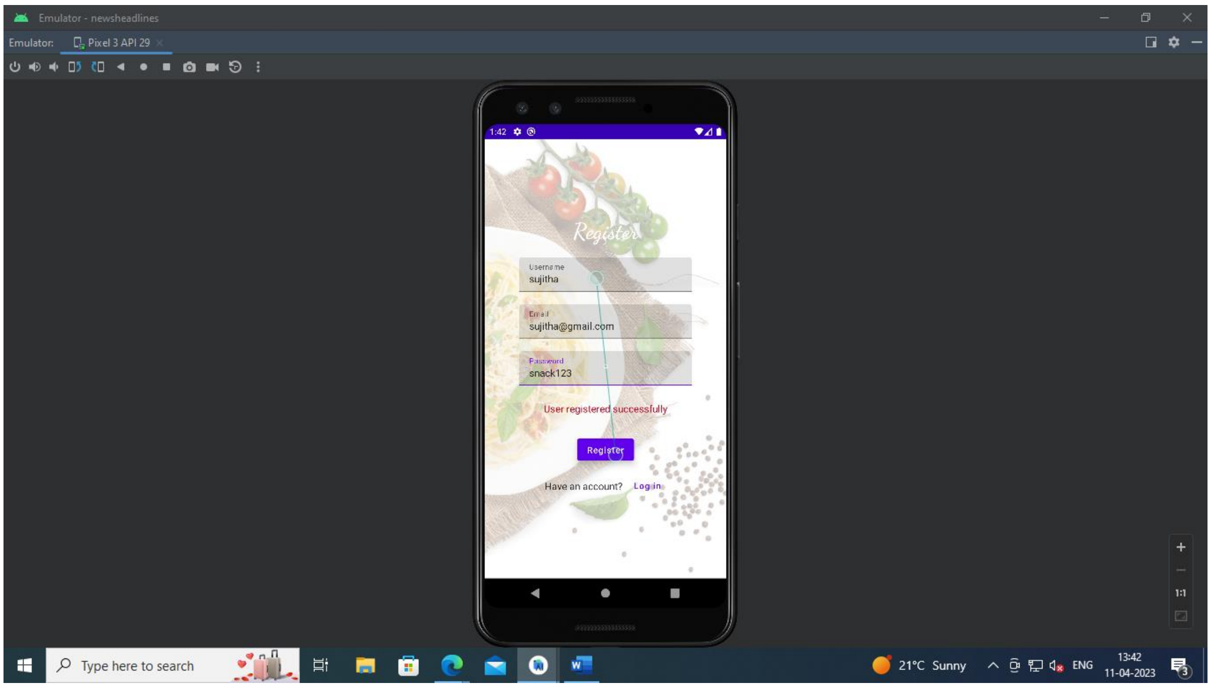
3.RESULT


Emulator - newsheadlines
Emulator: Pixel 3 API 29
1:40
Login
Username
Password
Login
Forgot
Type here to search
21°C Sunny
ENG
13:40
11-04-2023


Emulator - newsheadlines
Emulator: Pixel 3 API 29
1:40
Register
Username
Email
Password
Register
Have an account? Log in
Type here to search
21°C Sunny
ENG
13:40
11-04-2023

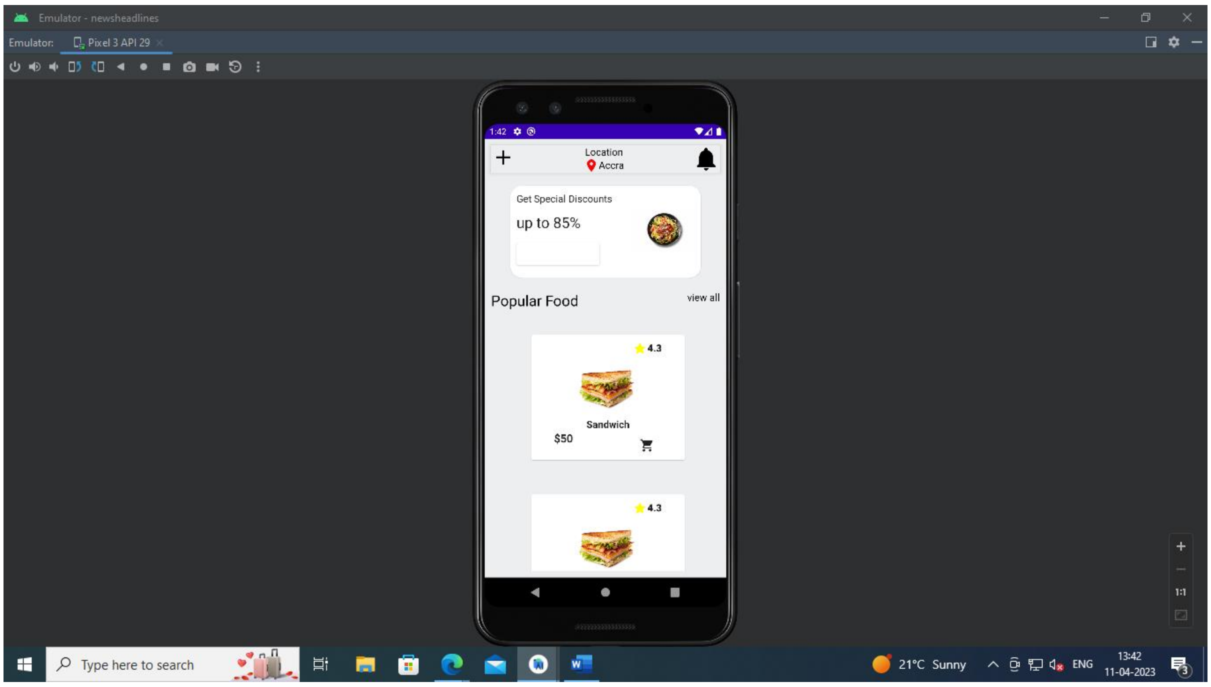


Username: sujitha
Email: sujitha@gmail.com
Password: snack123



Username: sujitha
Email: sujitha@gmail.com
Password: snack123

User registered successfully



4.ADVANTAGES AND DISADVANTAGES

ADVANTAGES:

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

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

In short, your customers choose to order snack online because it is really at their fingertips. Anyone with a smartphone can order snack online from their favourite restaurant.

- **Health benefits:**

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

- **Safer and healthier:**

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

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

- **Less chance for errors:**

One of the best advantages of an online snack ordering system for customers is that it ensures prices are accurate and there is less room for error when it comes time to settle the bill. This is because customers have to select an item in the menu at the appropriate price and make sure that the right amount is always paid.

- **More customers:**

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

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

- **Increased customer loyalty:**

If you give customers a reason to come back, they will choose your store over your competitor. You can promote their loyalty through the loyalty program.

According to a recent study, a personalized digital experience is also a great way to encourage customers to come back

- **Higher customer spends:**

We all know that more and more customers are now engaging in digital products and services than ever before.

They also spend more when ordering online. Because reading the online menu is different from standing in line.

- **Highly customizable:**

Snack ordering apps are highly customizable so you can easily advertise your logo, brand colours, or other features that make your business unique. Additionally, if you want to delete or add an item to the menu, you must sign in, make your changes, and it’s done.

DISADVANTAGE:

While there are many advantages to the online snack ordering system, there are also some disadvantages to online snack ordering systems. They are

- **Price:**

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

- **Limited menu:**

Another disadvantage for snack ordering systems is menu choices. Most snack ordering systems have a limited number of meals. The menu changes every few weeks or months, but if you stick to the system for more than a few months the menu items will come back again and again. You should also eat the snack provided for that week. If you do not want to eat that particular snack, you may have to order another snack from another place or eat snack you do not like.

- **Preparation:**

The preparation factor may be a disadvantage to snack ordering systems. Most snack ordering systems give frozen snack. They are usually easy to prepare, but they usually take more than an hour to cook because the snack is frozen. To avoid long cooking times, you can remove the snack from the freezer the day before. However, remember to eliminate snack from the freezer to reduce cooking time.

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

One problem with the snack ordering system is that the quality of the snack served is often worse than eating at a restaurant. Often, snack has to be fed over long distances, and over time, precious vitamins can be lost. Also, snack from the ordering system is often served in plastic packaging, which may not be very appealing to your eyes compared to the snack neatly placed on your plate in a restaurant.

- **Snack may get cold:**

Due to the long ordering distances, your snack may also be cold once it is finally delivered to your home. You need to reheat it or eat it cold.

This is especially true, if you order in an emergency the streets are often crowded and the ordering person will be stuck in traffic.

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

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

4.APPLICATIONS

- Restaurants.
- Bakeries.
- Independent snacks apps.
- Food cooperatives.
- Delivery platforms.
- Virtual restaurants.
- Online ordering apps.
- Supermarkets.

5.CONCLUSION

online snack ordering system is a mobile-based technology that aids the bakery and mini restaurant and snack industry in carrying out tasks effectively and efficiently. It aids in managing cash flow for managers. Managers can view analytics data to access company growth. The manager can control orders and employee schedules by using this system. The full complement is a snack ordering management system, it provides access to the online order platform, third-party connectors software, and comprehensive CRM solution, which together cover a sizable portion of your snack's requirements.

In the "online snack ordering app project", we made every effort to meet all the demands of the bakeries and snack industries. Because it is straightforward and adaptable, the project is successful. The biggest benefit of this project is that it draws plenty of users because of its simplicity. This project will undoubtedly succeed in replacing the antiquated manual way of storing secure information, the work plan also specifies the specific front end and back end characteristics of the technology being used in the project. Future project goals and its scope have been elaborated.

7.FUTURE SCOPE

Each project should pay close attention to future development because it contains the system's most recent features. It lessens software issues and defects. It developes a close relationship with customers based on their comments or preferences. Developer will incorporate certain dynamic elements that are briefly described.

Reporting module with real time mechanism.

- Modern architecture with smooth transitions.
- System foe email and mobile confirmation.
- Selling points.

8.SOURCE CODE

Gradel

```
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'
    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'


}
```

creating database

user

```
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?,


    )
```

user Dao

```
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)
}
```

user database

```
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
            }
        }
    }
}
```

user database helper

```
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) {
```

```kotlin
    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
    }

}
```

database 2:

order dataclass:

```
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?,
)
```

order Dao:

```
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)
}
```

order database

```
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
            }
        }
    }
}
```

database helper

```
package com.example.snackordering


import android.annotation.SuppressLint
import android.content.ContentValues
import android.content.Context
import android.database.Cursor
```

```kotlin
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
    }

}
```

login activity

```
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
```

```kotlin
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,
                RegisterActivity::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)
}
```

main activity

```
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.*
```

```kotlin
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.pasta),

                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) }

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)
        }
    }
}
```

target activity

```
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())

                }
            }
        }
    }
}
```

```kotlin
@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)
}
```

admin Activity

```
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}")

                    }

                }

            }

        }



    }

}
```

register activity

```
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 RegisterActivity : 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)
}
```

manifest:

```
<?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=".MainPage"
            android:exported="false"
            android:label="@string/title_activity_main_page"
            android:theme="@style/Theme.Snackordering" />
        <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=".RegisterActivity"

            android:exported="false"

            android:label="MainActivity"

            android:theme="@style/Theme.Snackordering"                    />

    </application>


</manifest>
```