SNACK SQUAD: A CUSTOMIZABLE SNACK ORDERING AND DELIVERY APP

A PROJECT

Submitted by

M. PERUMAL (20201231401227)

N. MARIAPPAN (20201231401221)

K. MUTHU KUMAR (20201231401222)

R. SATHEESH KUMAR (20201231401235)

MENTOR

E.JACQULINE, M.C.A., M.Phil.,

in partial fulfillment of the requirements for the award of degree of

BACHELOR OF COMPUTER APPLICATION

SRI PARAMAKALYANI COLLEGE

ALWARKURICHI – 627 412

APRIL - 2023

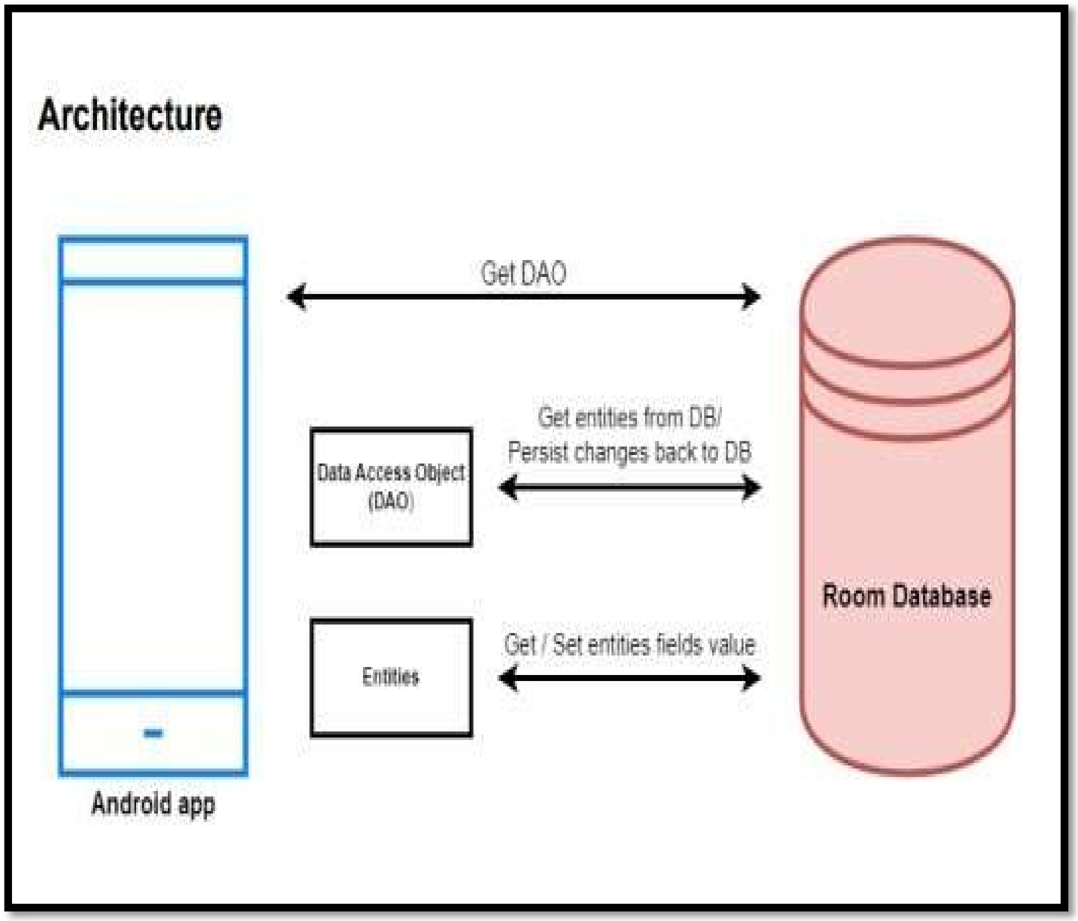
TABLE OF CONTENTS

CHAPTER - I

INTRODUCTION

1.1. OVERVIEW

A project that demonstrates the use of Android Jetpack Compose to build a UI for a snack squad app. Snack Squad is a sample project built using the Android Compose UI toolkit. It demonstrates how to create a simple e-commerce app for snacks using the Compose libraries.



- The goal of this project is to create an application to present available menu items for all campus locations in a unified manner. This will be done by allowing vendors to establish menus and daily specials for users to sort through.

- Online food/snack ordering is the process of ordering food and snacks, for delivery or pickup, from a website or other application.

- The product can be either ready to-eat food (e.g., direct from a home-kitchen, restaurant, or a virtual restaurant) or food that has not been specially prepared for direct consumption (e.g., vegetables direct from a farm/garden, fruits, frozen meats. etc). Online food ordering/delivery through third-party companies have emerged as a global industry, leading to a "delivery revolution". From 2018 to 2021, global revenues for the online food delivery sector rose from $90 billion to $294 billion.

- A snack ordering app is a mobile application that allows users to browse and order a variety of snacks, drinks, and other food items from a menu. Users can create an account, log in, and select their desired items from the menu. They can then add the items to their cart and proceed to checkout. Payment can be made through the app using a secure payment gateway, and users can track the status of their order and estimated delivery time.

1.2. PURPOSE

The user can see a list of snacks, and by tapping on a snack, and by tapping on the "Add to Cart" button, the snack will be added to the cart. The user can also see the list of items in the cart and can proceed to checkout to make the purchase.

By end of this project:

- You'll be able to work on Android studio and build an app.
- You'll be able to integrate the database accordingly.

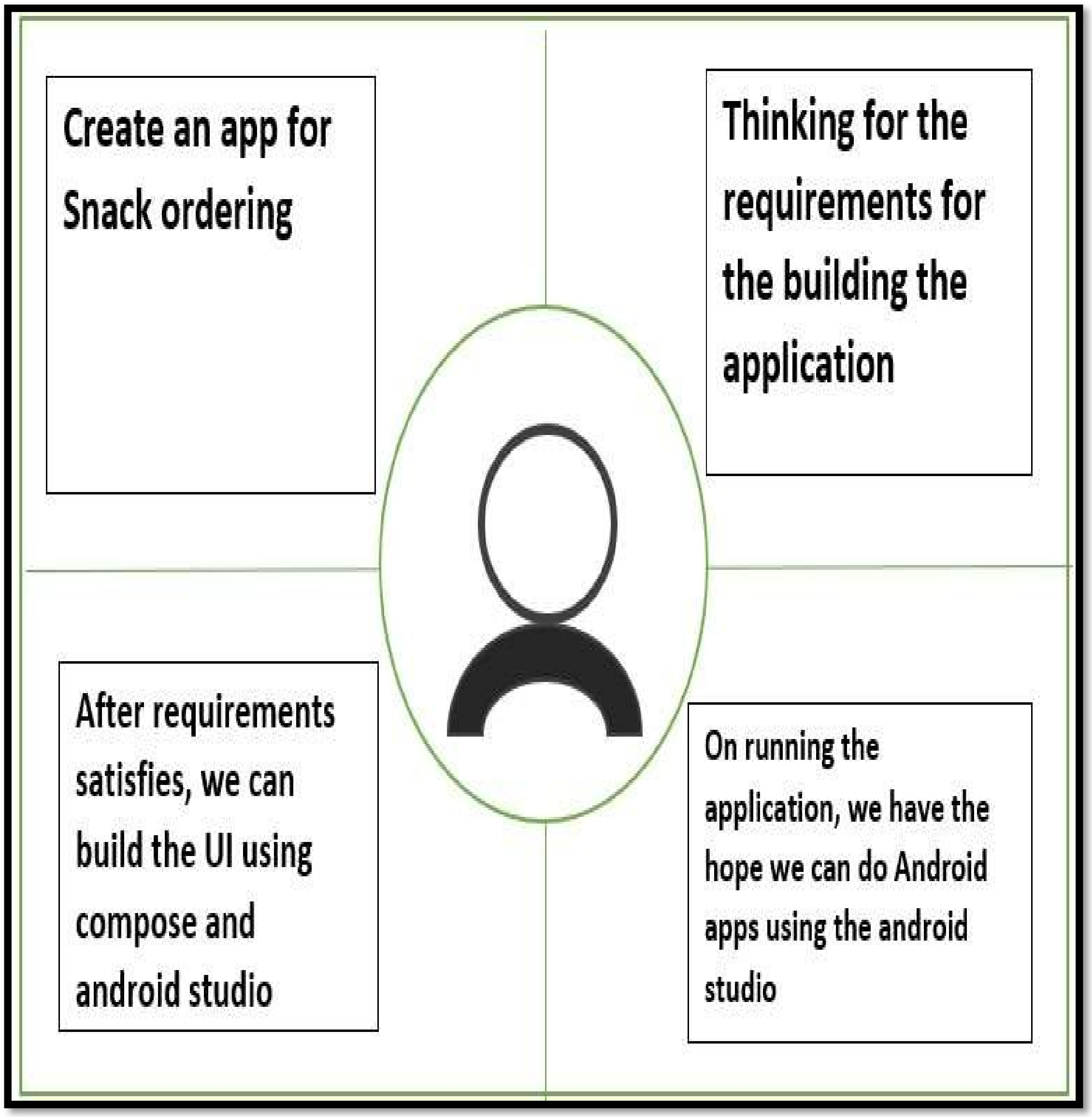
This app provides users with a simple ordering functionality. Thus, while you plan your on-demand delivery app, make sure you cross-check your app's check-out features. As you are planning for a food apps development, you can implement a feature "favourite list" where users can personalize their search and can find or reorder the items just in a tap, rather than roaming through the menu. The SNACK SQUAD App are offering you a snack that are perfect to eat with your favourite squad.

The snacks ordering app also benefits businesses by allowing them to reach a wider audience and increase their revenue by offering online ordering and delivery services. The app can help businesses streamline their operations by reducing the time and resources required to process orders and manage inventory.

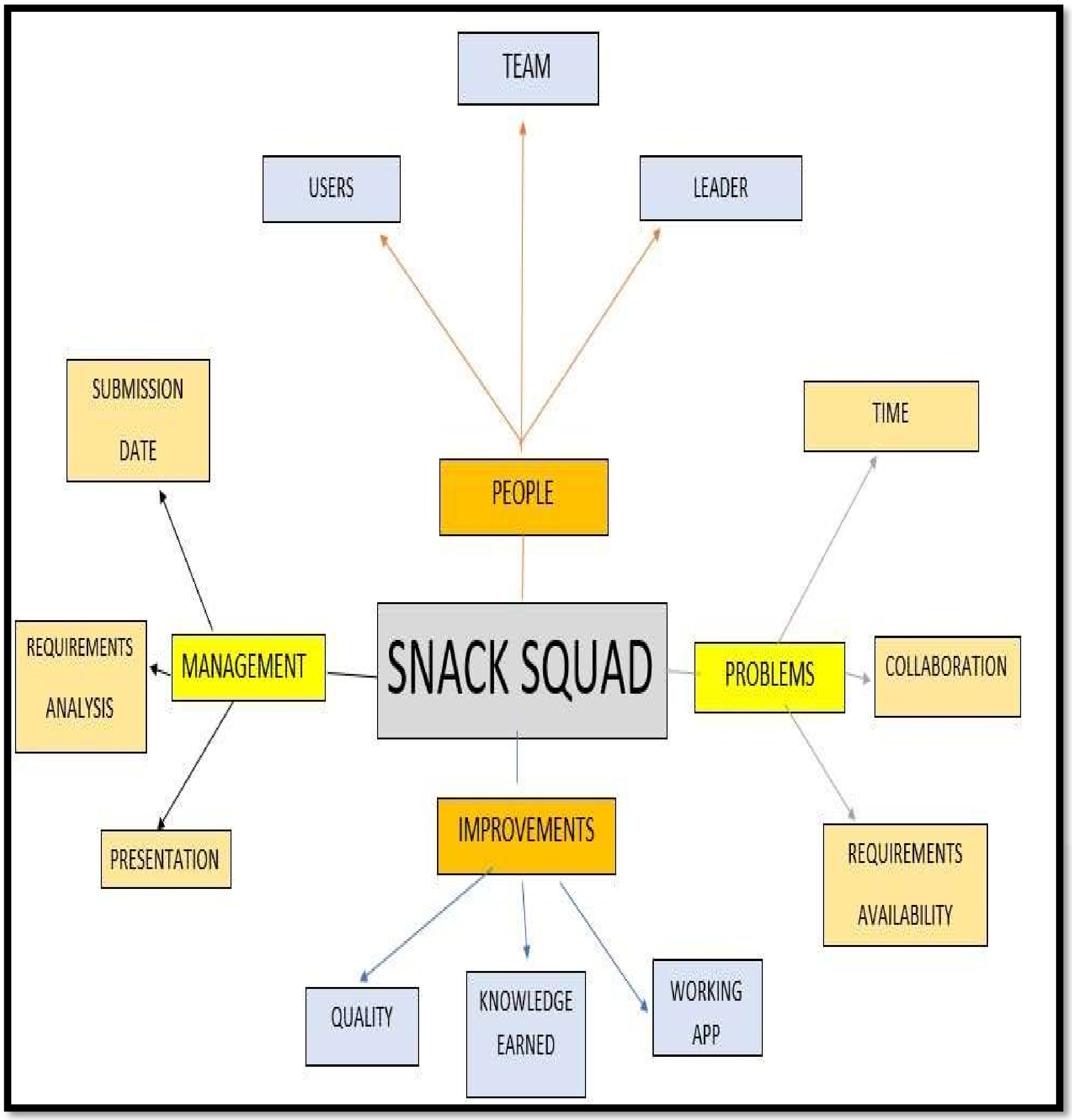
CHAPTER - II

PROBLEM DEFINITION & DESIGN THINKING

2.1. EMPATHY MAP


Create an app for Snack ordering
Thinking for the requirements for the building the application
After requirements satisfies, we can build the UI using compose and android studio
On running the application, we have the hope we can do Android apps using the android studio

2.2. IDEATION & BRAINSTORMIN MAP

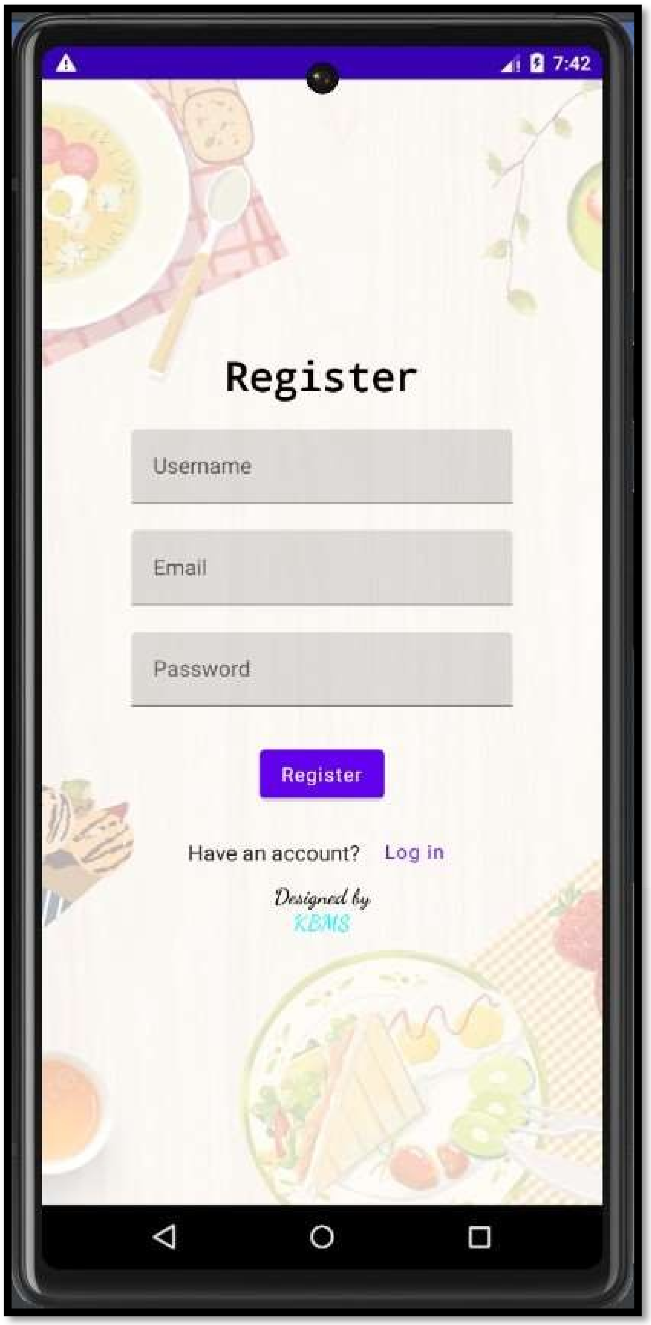


CHAPTER – III

RESULT

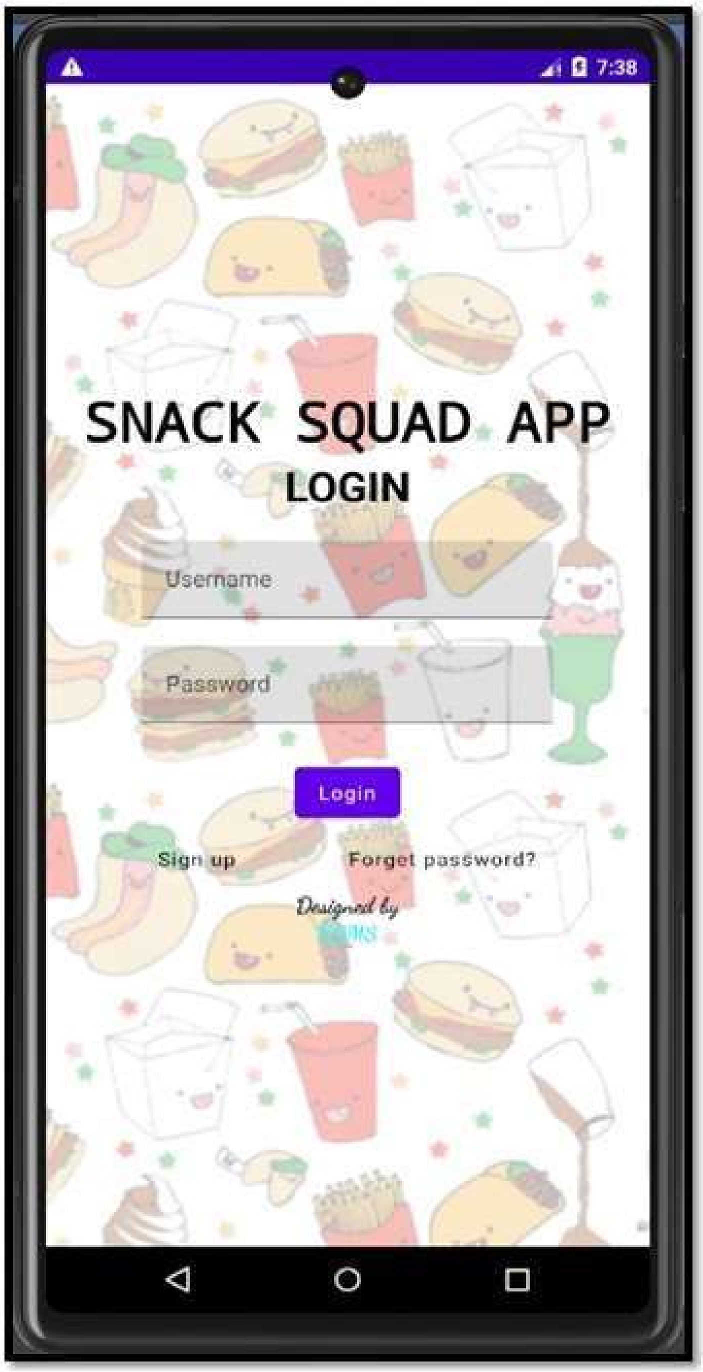
REGISTER PAGE

- Users register into the application.



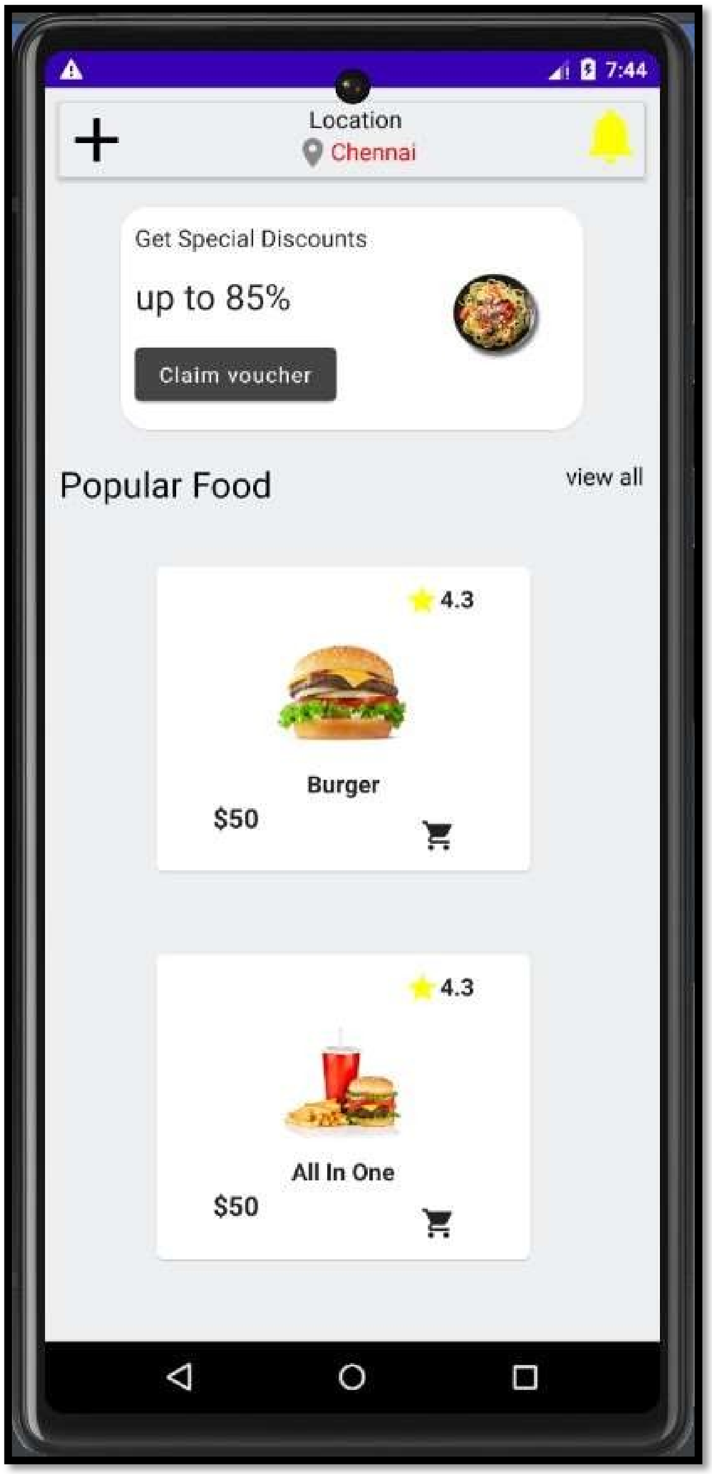
LOGIN PAGE

- After registration , user logins into the application.



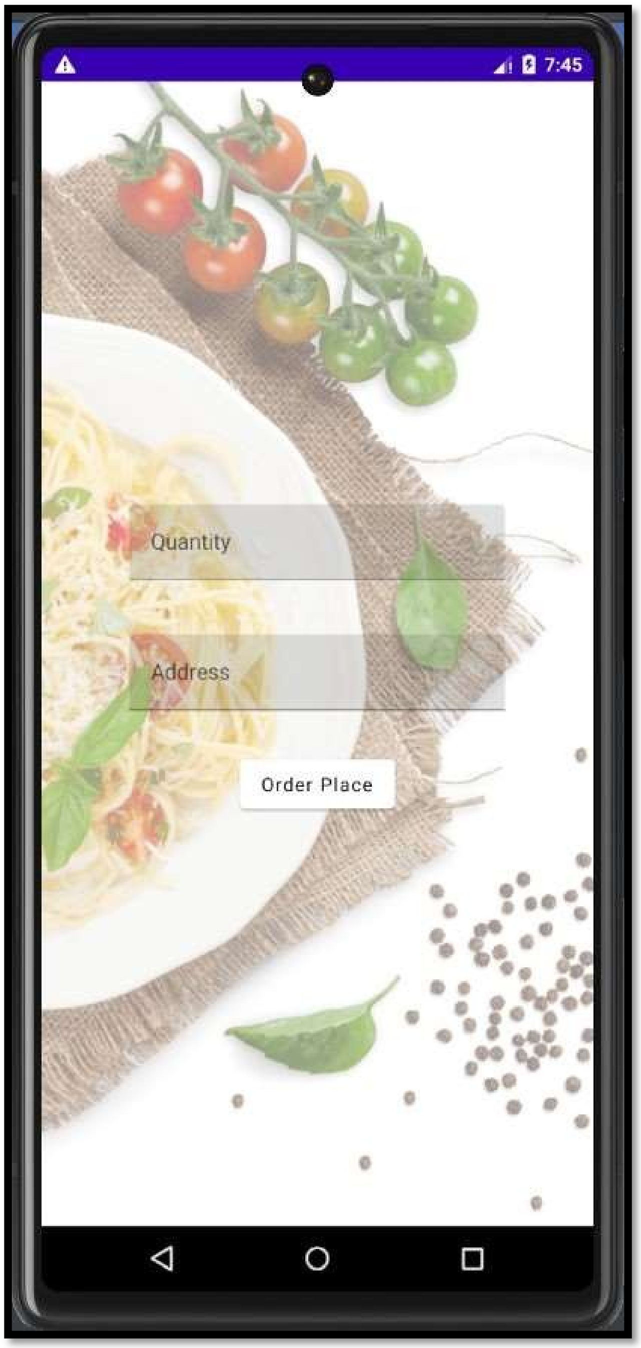
MAIN PAGE

- User enters into the main page



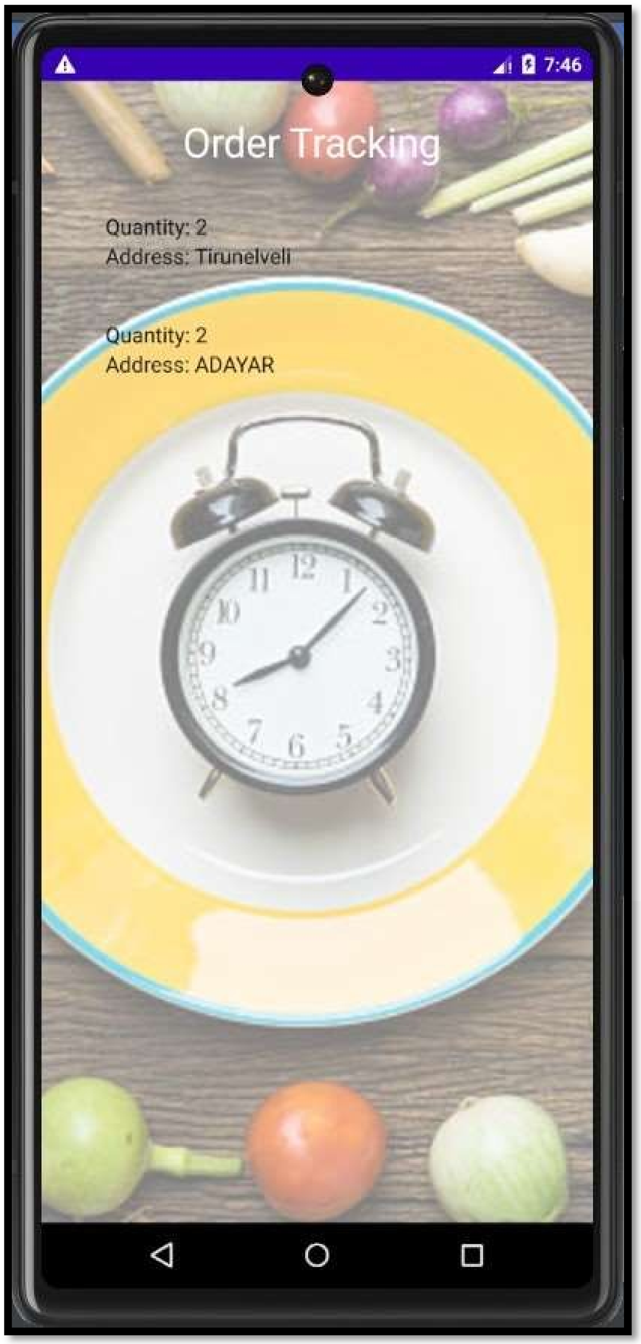
ORDER PAGE

- User can view the items, select and order the items.



ADMIN PAGE

- From admin login we can view the orders placed.



Order Tracking

Quantity: 2
Address: Tirunelveli

Quantity: 2
Address: ADAYAR

CHAPTER - IV

ADVANTAGES & DISADVANTAGES

ADVANTAGES

- Running an online food and 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 food online because it is really at their fingertips.
  - So, using the online food ordering system is the easiest way to attract millennials.
- Health benefits:
  - One of the important benefits of food ordering systems is health benefits. Because the meal is planned, it is easy to determine the exact number of calories consumed in each meal.
- 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.
- Highly customizable:
  - Food and Snack ordering apps are highly customizable so you can easily advertise your logo, brand colours, or other features that make your business unique.
  - Snack applications can have several advantages such as providing a boost of energy if several hours pass between meals and blood glucose levels drop, helping curb your appetite to prevent overeating at the next meal, providing extra nutrients when

choosing certain snacks like fresh fruit or nuts, and can help maintain adequate nutrition if one has a busy schedule .

- Snacks can also be beneficial in a diet by increasing nutrient intake, sustaining energy levels, helping the body recover from exercise and giving individuals plenty of healthy options.

DISADVANTAGES

- While there are many advantages to the online food ordering system, there are also some disadvantages to online Snack ordering systems. They are
- Price:
    - One of the major drawbacks of online food ordering systems is price.
    - When food is ordered for more than one person, the cost is usually equal to eating at a good restaurant every night.
- Limited menu:
    - Another disadvantage for food and snack ordering systems is menu choices.
    - Most food ordering systems have a limited number of meals.
- There are several disadvantages of snacking that you should be aware of. One of the most common disadvantages is that snacking can lead to unwanted weight gain if portions or frequency of snacking is too much, adding excess calories. Too much snacking can reduce hunger at meal times or cause one to skip a meal entirely, which increases the risk of nutrient deficiencies .

CHAPTER - V

APPLICATIONS

- Anyone with a smartphone can order food online from their favourite restaurant.
- More than 97% of millennials use their phones for anything. Ordering food online comes into the same broad category.
- 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.
- According to a recent survey out of the 1000 customers, 50% said they change brands that offer a worse online experience, and 73% expect online customization.
- Furthermore, the rising export of various Indian snacks to western countries, along with the increasing popularity of various fusion snacks that have a blend of Indian and western spices, is bolstering the market growth in India.
- It is used in some institutions and family functions to deliver the bulk orders on time.
- Also gets the food in all cities and villages without any partiality.
- Online food ordering gives the customers freedom and choice to place an order to virtually anytime, everywhere, saving the time and resources typically spend on travelling to pick up a meal.
- It also gives the customers the advantage of reordering the favourite order in the easiest and hassle-free manner.

CHAPTER - VI

CONCLUSION

An Online Snack ordering system gives start-up snack restaurants complete control over their services. You won’t have to pay any charges or commission on any orders, increasing your profit margins. In addition, the analytics dashboards equip you with the valuable insights you need to enhance your services. Also, with a snack menu online, tracking the order is done easily, it maintains customer's database and improve food delivery service.

CHAPTER - VII

FUTURE SCOPE

At present, the increasing demand for snacks, as they are convenient and flavourful, represents one of the primary factors influencing the market positively in India.

- Besides this, the rising popularity of various convenient food products among working individuals, as they save time and do not require the hassle of cooking, is propelling the growth of the market.
- In addition, the growing consumption of various snacks with ethnic tastes is offering a favourable market outlook in the country.
- Apart from this, the increasing number of e-commerce brands and distribution channels selling low-calorie, sugar-free, preservative-free, and gluten-free snacks with interesting flavour’s is contributing to the growth of the market.

CHAPTER - VIII

APPENDIX

(i) SOURCE CODE

The source code for the above project was given in the GitHub link.

https://github.com/Perumalmahe2023/Snack-Squad-A-Customizable-Snack-Ordering-and-Delivery-App.git

ADDING REQUIRED DEPENDENCIES

Gradle scripts > build.gradle(Module :app)

```
dependencies {
implementation 'androidx.core:core-ktx:1.7.0'
implementation 'androidx.lifecycle:lifecycle-runtime-ktx:2.3.1'
implementation 'androidx.activity:activity-compose:1.3.1'
implementation  "androidx.compose.ui:ui:$compose_ui_version"
implementation "androidx.compose.ui:ui-tooling-preview:$compose_ui_version"
implementation 'androidx.compose.material:material:1.2.0'
implementation 'androidx.room:room-common:2.5.0'
implementation 'androidx.room:room-ktx:2.5.0'
testImplementation 'junit:junit:4.13.2'
androidTestImplementation  'androidx.test.ext:junit:1.1.5'
androidTestImplementation  'androidx.test.espresso:espresso-core:3.5.1'
androidTestImplementation "androidx.compose.ui:ui-test-
junit4:$compose_ui_version"
debugImplementation "androidx.compose.ui:ui-tooling:$compose_ui_version"
debugImplementation "androidx.compose.ui:ui-test-manifest:$compose_ui_version"
}
```

CREATE USER DATA CLASS

```
package com.example.snackordering
import androidx.room.ColumnInfo
import androidx.room.Entity
import androidx.room.PrimaryKey
@Entity(tableName = "user_table")
data class User(
@PrimaryKey(autoGenerate = true) val id: Int?,
@ColumnInfo(name = "first_name") val firstName: String?,
@ColumnInfo(name = "last_name") val lastName: String?,
@ColumnInfo(name = "email") val email: String?,
@ColumnInfo(name = "password") val password: String?,
)
```

CREATE AN USERDAO INTERFACE

```
package com.example.snackordering

import androidx.room.*

@Dao
interface UserDao {

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

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

@Update
suspend fun updateUser(user: User)

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

CREATE AN USERDATABASE CLASS

```
package  com.example.snackordering

import   android.content.Context
import  androidx.room.Database
import androidx.room.Room
import androidx.room.RoomDatabase

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

abstract fun userDao(): UserDao
companion object {
@Volatile
private var instance: UserDatabase? = null

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

CREATE AN USERDATABASEHELPER CLASS

```
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
}
}

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

CREATE ORDER DATA CLASS

```
package com.example.snackordering

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

@Entity(tableName = "order_table")
data class Order(
@PrimaryKey(autoGenerate = true) val id: Int?,
@ColumnInfo(name = "quantity") val quantity: String?,
@ColumnInfo(name = "address") val address: String?,
)
```

CREATE ORDERDATABASE CLASS

```
package com.example.snackordering

import android.content.Context
import androidx.room.Database
import androidx.room.Room
import androidx.room.RoomDatabase

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

abstract fun orderDao(): OrderDao

companion object {
```

```
@Volatile
private var instance: OrderDatabase? = null

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

CREATE ORDERDATABASEHELPER CLASS

```
package com.example.snackordering

import android.annotation.SuppressLint
import android.content.ContentValues
import android.content.Context
import  android.database.Cursor
import android.database.sqlite.SQLiteDatabase
import android.database.sqlite.SQLiteOpenHelper

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

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

private const val TABLE_NAME = "order_table"
private const val COLUMN_ID = "id"
private const val COLUMN_QUANTITY = "quantity"
private const val COLUMN_ADDRESS = "address"
}

override fun onCreate(db: SQLiteDatabase?) {
val createTable = "CREATE TABLE $TABLE_NAME (" +
"${COLUMN_ID} INTEGER PRIMARY KEY AUTOINCREMENT, " +
"${COLUMN_QUANTITY} Text, " +
"${COLUMN_ADDRESS} TEXT " +
")"

db?.execSQL(createTable)
}

override fun onUpgrade(db: SQLiteDatabase?, oldVersion: Int, newVersion: Int) {
db?.execSQL("DROP TABLE IF EXISTS $TABLE_NAME")
onCreate(db)
}

fun insertOrder(order: Order) {
val db = writableDatabase
val values = ContentValues()
values.put(COLUMN_QUANTITY, order.quantity)
values.put(COLUMN_ADDRESS, order.address)
db.insert(TABLE_NAME, null, values)
db.close()
}



@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 } }
```

CREATING LOGINACTIVITY.KT WITH DATABASE

```
package com.example.snackordering


import android.content.Context

import android.content.Intent

import android.os.Bundle

import androidx.activity.ComponentActivity

import androidx.activity.compose.setContent

import androidx.compose.foundation.Image

import androidx.compose.foundation.layout.*
import androidx.compose.material.*
import androidx.compose.runtime.*
import androidx.compose.ui.Alignment
import androidx.compose.ui.Modifier
import  androidx.compose.ui.graphics.Color
import androidx.compose.ui.layout.ContentScale
import androidx.compose.ui.res.painterResource
import androidx.compose.ui.text.font.FontFamily
import androidx.compose.ui.text.font.FontWeight
import androidx.compose.ui.text.input.PasswordVisualTransformation
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.login_screen), 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("SNACK SQUAD APP",
fontSize = 40.sp,
fontWeight = FontWeight.Bold,
fontFamily = FontFamily.Monospace,
color = Color.Black
)

Text(
fontSize = 30.sp,
fontWeight = FontWeight.SemiBold,
fontFamily = FontFamily.SansSerif,
color = Color.Black,
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") },
visualTransformation= PasswordVisualTransformation(),
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 == "admin") {
error = "Admin Successfully log in"
context.startActivity(
Intent(
context,
AdminActivity::class.java
)
)
}
else if (user != null && user.password == password) {
error = "User Successfully log in"
context.startActivity(
Intent(
context,
MainPage::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.Black, text = "Sign up") }
TextButton(onClick = {
})

{
Spacer(modifier = Modifier.width(60.dp))
Text(color = Color.Black, text = "Forget password?")
}
}
Spacer(modifier = Modifier.width(160.dp))
Text("Designed by",
fontSize = 16.sp,
fontWeight = FontWeight.ExtraBold,
fontFamily = FontFamily.Cursive,
color = Color.Black,
)

Text("KBMS",
fontSize = 16.sp,
fontWeight = FontWeight.ExtraBold,
fontFamily = FontFamily.Cursive,
color = Color.Cyan
)
}
```

```
}

private fun startMainPage(context: Context) {
val intent = Intent(context, MainPage::class.java)
ContextCompat.startActivity(context, intent, null)
}
```

CREATING MAINACTIVITY.KT WITH DATABASE

MainActivity is converted into RegisterActivity.kt as follows below:

```
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.text.input.PasswordVisualTransformation
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.register), 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 = 30.sp,
fontWeight  =  FontWeight.ExtraBold,
fontFamily  =  FontFamily.Monospace,
color = Color.Black,
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") },
visualTransformation= PasswordVisualTransformation(),
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")
}
}
Spacer(modifier = Modifier.width(160.dp))
Text("Designed by",
fontSize = 16.sp,
fontWeight = FontWeight.ExtraBold,
fontFamily = FontFamily.Cursive,
color = Color.Black,
)
Text("KBMS",
fontSize = 16.sp,
fontWeight = FontWeight.ExtraBold,
fontFamily = FontFamily.Cursive,
color = Color.Cyan
)

}
}
private fun startLoginActivity(context: Context) {
val intent = Intent(context, LoginActivity::class.java)
ContextCompat.startActivity(context, intent, null)
}
```

CREATING MAINPAGE.KT FILE

```
package com.example.snackordering


import android.annotation.SuppressLint
import android.content.Context
import android.os.Bundle
import 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.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  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(RectangleShape)
.size(50.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.Gray,
)
Text(text = "Chennai" , color = Color.Red)
}

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

Modifier
.size(45.dp),
tint = Color.Yellow
)
}
```

```
}

@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.DarkGray)) {
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(RectangleShape)
)
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.burger to R.string.burgers,
R.drawable.pack to R.string.pack,
R.drawable.salad to R.string.salad,
R.drawable.popcorn 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)
}
}
}
```

CREATING TARGETACTIVITY.KT

```
package com.example.snackordering

import android.content.Context
import android.content.Intent
import android.os.Bundle
import android.util.Log
import android.widget.Toast
import androidx.activity.ComponentActivity
import androidx.activity.compose.setContent
import androidx.compose.foundation.Image
import androidx.compose.foundation.background
import androidx.compose.foundation.layout.*
import androidx.compose.foundation.text.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.res.painterResource
import androidx.compose.ui.text.input.KeyboardType
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("kathir", 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("") }
val error by remember { mutableStateOf("") }


TextField(value = quantity, onValueChange = {quantity=it},
label = { Text("Quantity") },
keyboardOptions = KeyboardOptions(keyboardType = KeyboardType.Number),
modifier = Modifier
.padding(10.dp)
.width(280.dp))

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

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

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


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


Button(onClick = {
if( quantity.isNotEmpty() and address.isNotEmpty()){
val order = Order(
id = null,
quantity = quantity,
address = address
)
orderDatabaseHelper.insertOrder(order)
Toast.makeText(mContext, "Order Placed Successfully",
Toast.LENGTH_SHORT).show()}
},
colors = ButtonDefaults.buttonColors(backgroundColor = Color.White))
{
Text(text = "Order Place", color = Color.Black)
}


}
}

private fun startMainPage(context: Context) {
val intent = Intent(context, LoginActivity::class.java)
ContextCompat.startActivity(context, intent, null)
}
```

CREATING ADMINACTIVITY.KT

```
package com.example.snackordering


import android.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

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("kathir" ,data.toString())
val order = orderDatabaseHelper.getAllOrders()
ListListScopeSample(order)
}
}
}
}
}

@Composable
fun ListListScopeSample(order: List<Order>) {
Image(
painterResource(id = R.drawable.order1), contentDescription = "",
alpha =0.5F,
contentScale = ContentScale.FillHeight)
Text(text = "Order Tracking", modifier = Modifier.padding(top = 24.dp, start =
106.dp, bottom = 24.dp ), color = Color.White, fontSize = 30.sp)
Spacer(modifier = Modifier.height(30.dp))
LazyRow(
modifier = Modifier
.fillMaxSize()
.padding(top = 80.dp),

horizontalArrangement = Arrangement.SpaceBetween
){
item {

LazyColumn {
items(order) { order ->
Column(modifier = Modifier.padding(top = 16.dp, start = 48.dp, bottom = 20.dp)) {
Text("Quantity: ${order.quantity}")
Text("Address: ${order.address}")
}
}
}
}

}
}
```

MODIFYING ANDROIDMANIFEST.XML

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
```

```
xmlns:tools="http://schemas.android.com/tools">

<application
android:allowBackup="true"
android:dataExtractionRules="@xml/data_extraction_rules"
android:fullBackupContent="@xml/backup_rules"
android:icon="@drawable/icon"
android:label="@string/app_name"
android:supportsRtl="true"
android:theme="@style/Theme.SnackOrdering"
tools:targetApi="31">
<activity
android:name=".AdminActivity"
android:exported="true"
android:label="@string/title_activity_admin"
android:theme="@style/Theme.SnackOrdering" />
<activity
android:name=".LoginActivity"
android:exported="true"
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>
```

APP WORKING DEMO

DRIVE LINK - https://drive.google.com/file/d/1RvZa0X8scAo9ICTlv_BnOQEk_mvpvL1n/view?usp=drivesdk

&

YOUTUBE LINK - https://youtu.be/T1QuGs9qWdk

THANK YOU