SNACK SQUAD: A CUSTOMIZEABLE SNACK ORDERING AND DELIVERY APP

1.INTRODUTION

1.1. Overview

A snack is a small portion of food generally eaten between meals.[1] Snacks come in a variety of forms including packaged snack foods and other processed foods, as well as items made from fresh ingredients at home.

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 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.

1.2. purpose

Traditionally, snacks are prepared from ingredients commonly available at home without a great deal of preparation. Often cold cuts, fruits, leftovers, nuts, sandwiches, and sweets are used as

Learning Outcomes :

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.

Project Workflow:

- Users register into the application

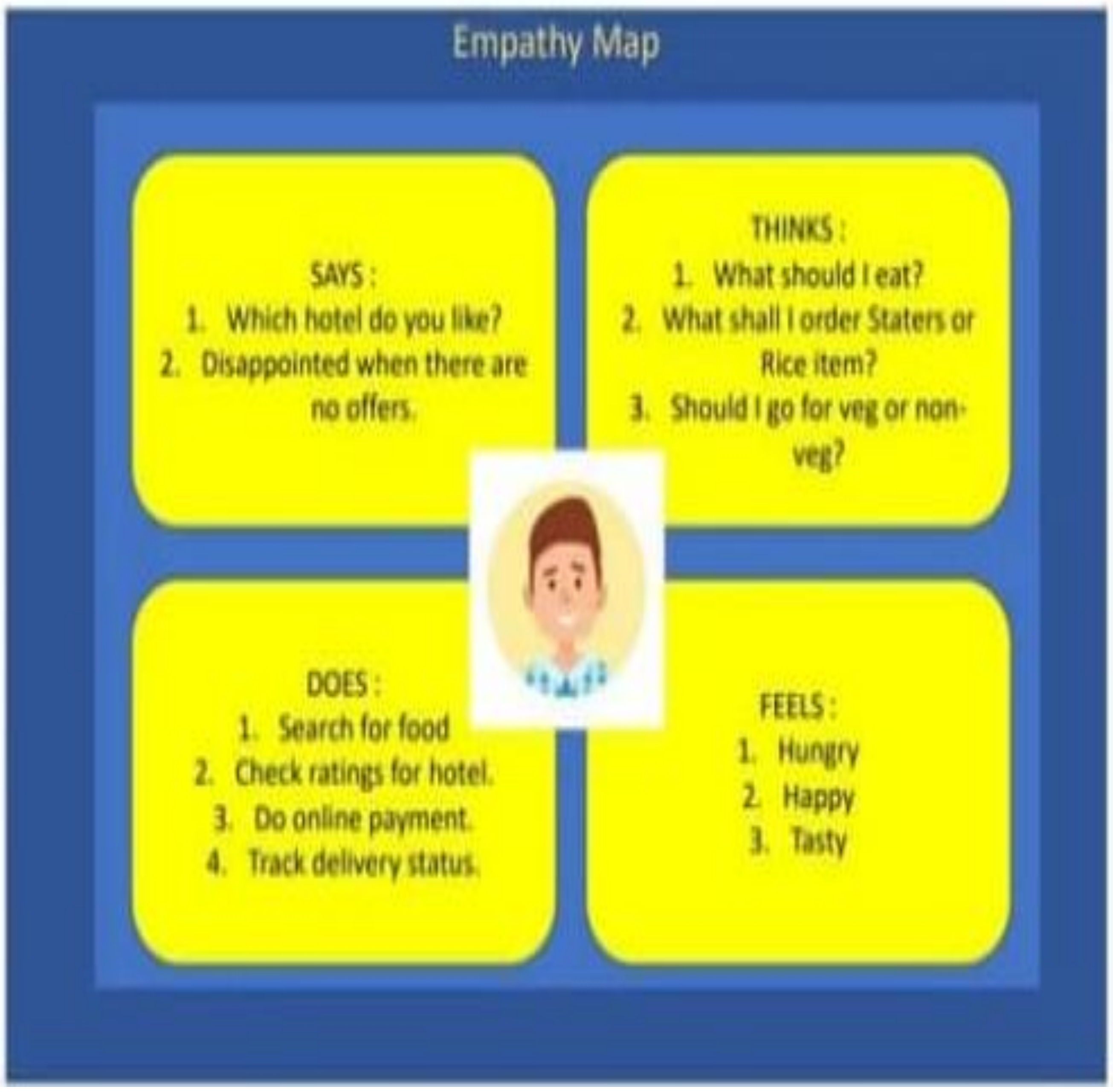
- After registration , user logins into the application

- User enters into the main page

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

02.PROBLEM DEFINITION & DESIGN THINKING

2.1.Empathy Map


Empathy Map
SAYS :
1. Which hotel do you like?
2. Disappointed when there are no offers.
THINKS :
1. What should I eat?
2. What shall I order Staters or Rice item?
3. Should I go for veg or non-veg?
DOES :
1. Search for food
2. Check ratings for hotel.
3. Do online payment.
4. Track delivery status.
FEELS :
1. Hungry
2. Happy
3. Tasty

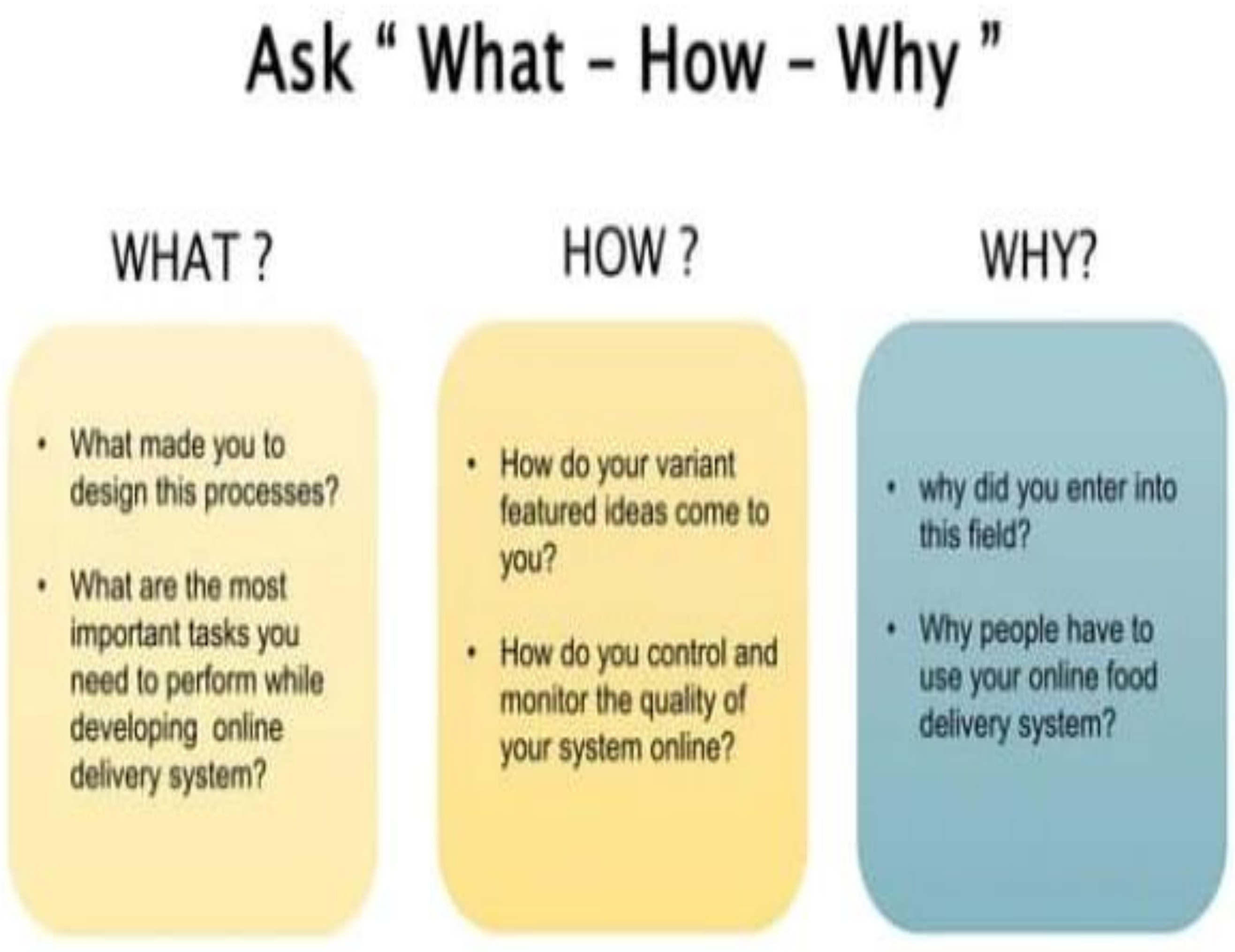
Problem Statement:

The online food ordering system setsup a food menu online and customer can easy place the order as per they like. Also, the online customer can easy track their orders. The management maintains customer database, and improve food delivery service. This system also provides a feedback system which user can rate the food items.

5-WHY'S

- Why does ordering offline may not be as visually pleasing compared to doing the same in online ordering system?
- Why online ordering system can never completely replace a traditional offline ordering system?
- Why does the technical problems arise in the system?
- Who do the data security has minimal errors that occer in the system?

Design:



Ask " What - How - Why "

WHAT ?

- What made you to design this processes?
- What are the most important tasks you need to perform while developing online delivery system?

HOW ?

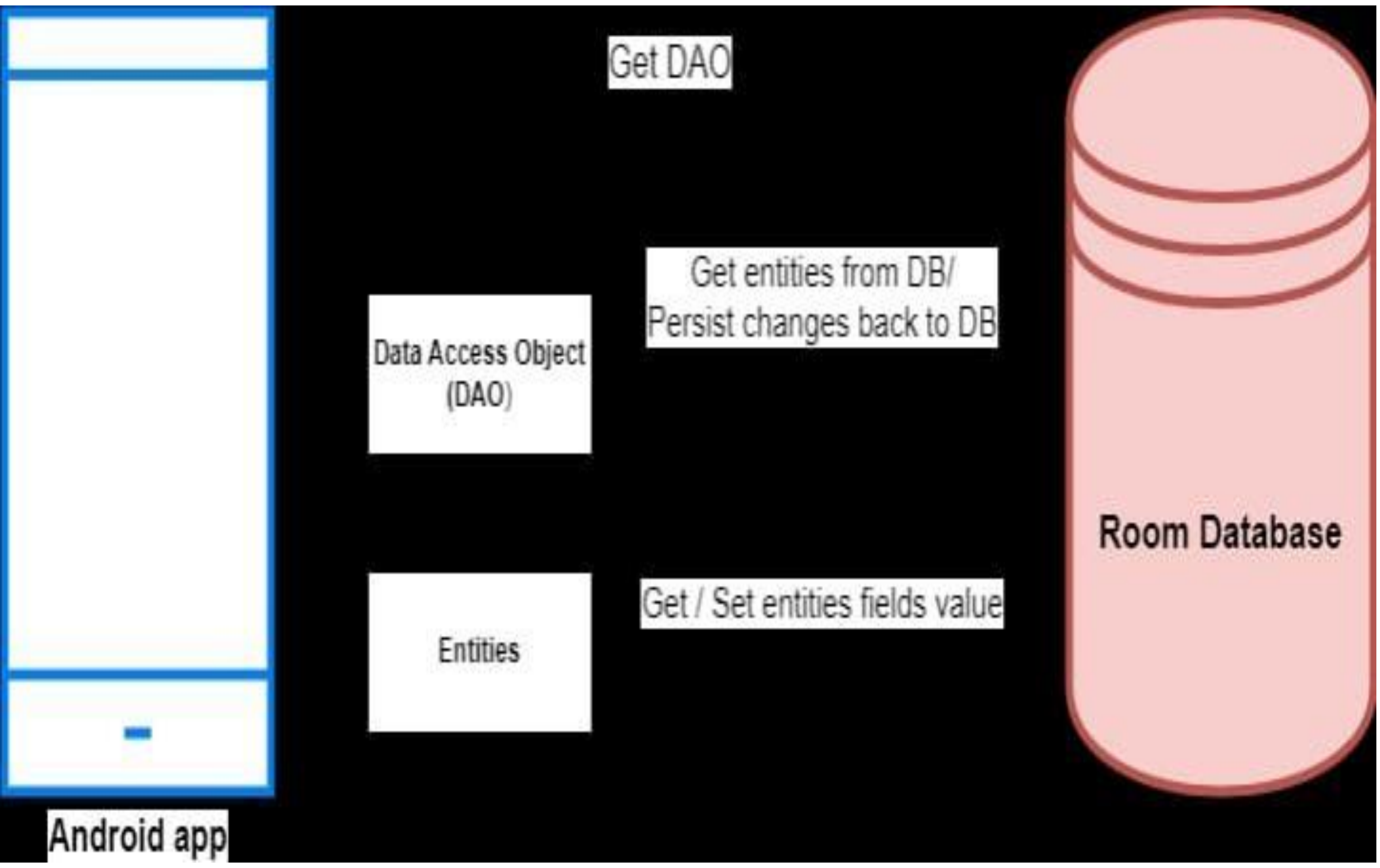
- How do your variant featured ideas come to you?
- How do you control and monitor the quality of your system online?

WHY?

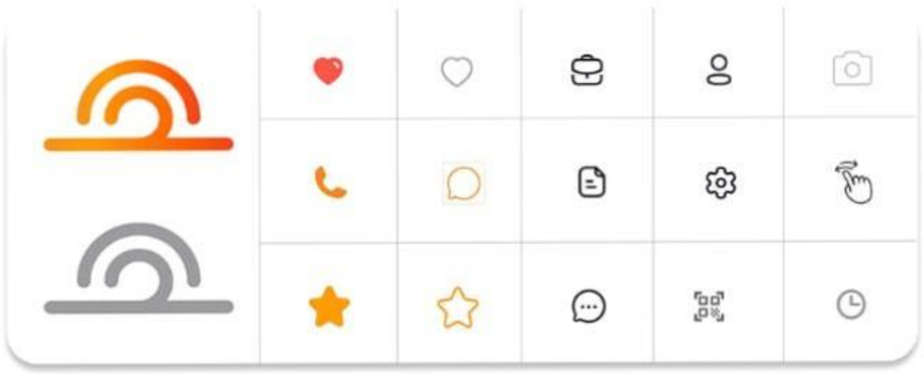
- why did you enter into this field?
- Why people have to use your online food delivery system?

- Why does the risk of losing customer's data is more in the online food delivery system?

2.2. Ideatication & Brainstorming map Screenshot


Get DAO
Get entities from DB/ Persist changes back to DB
Data Access Object (DAO)
Room Database
Get / Set entities fields value
Entities
Android app

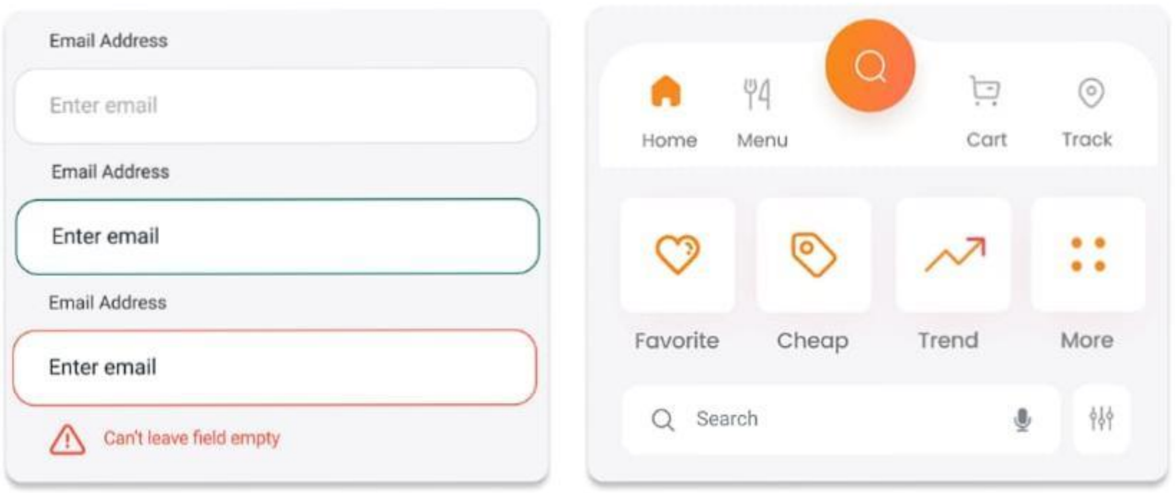
LOGO & iCONS



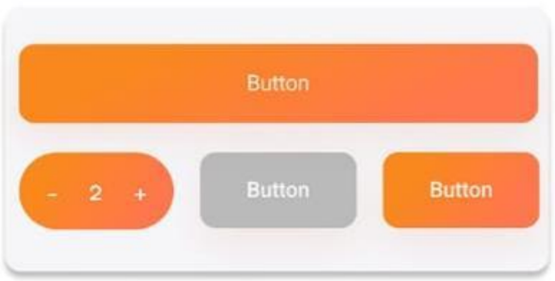
UI Components

Inputs

Tab bar



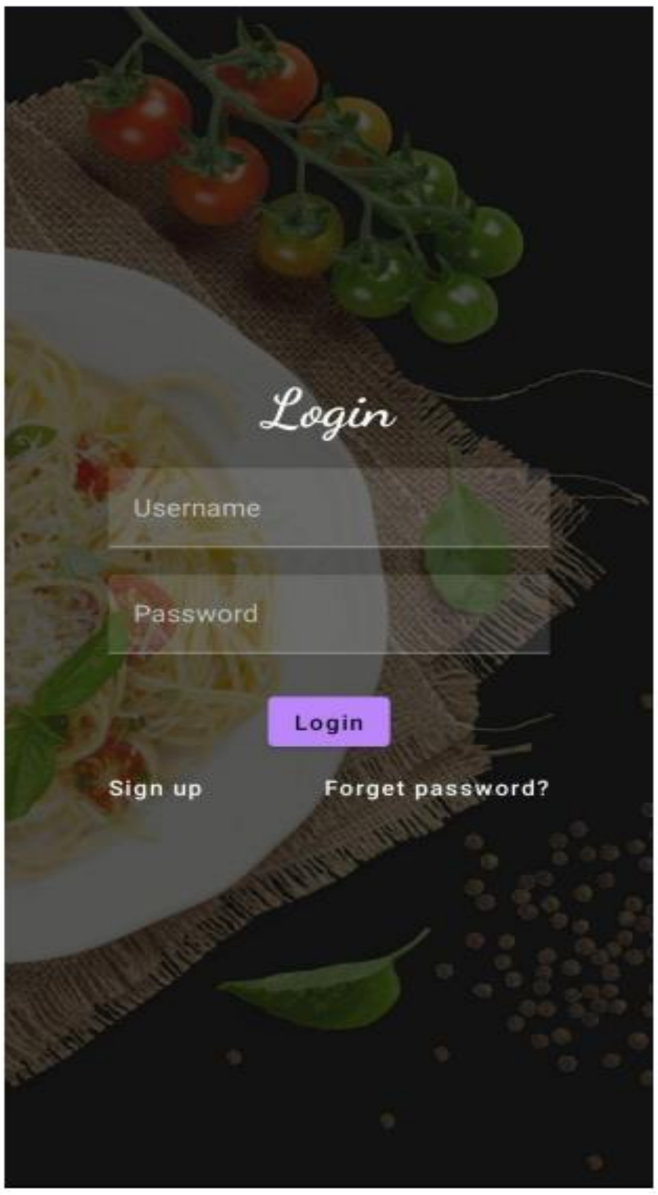
Buttons



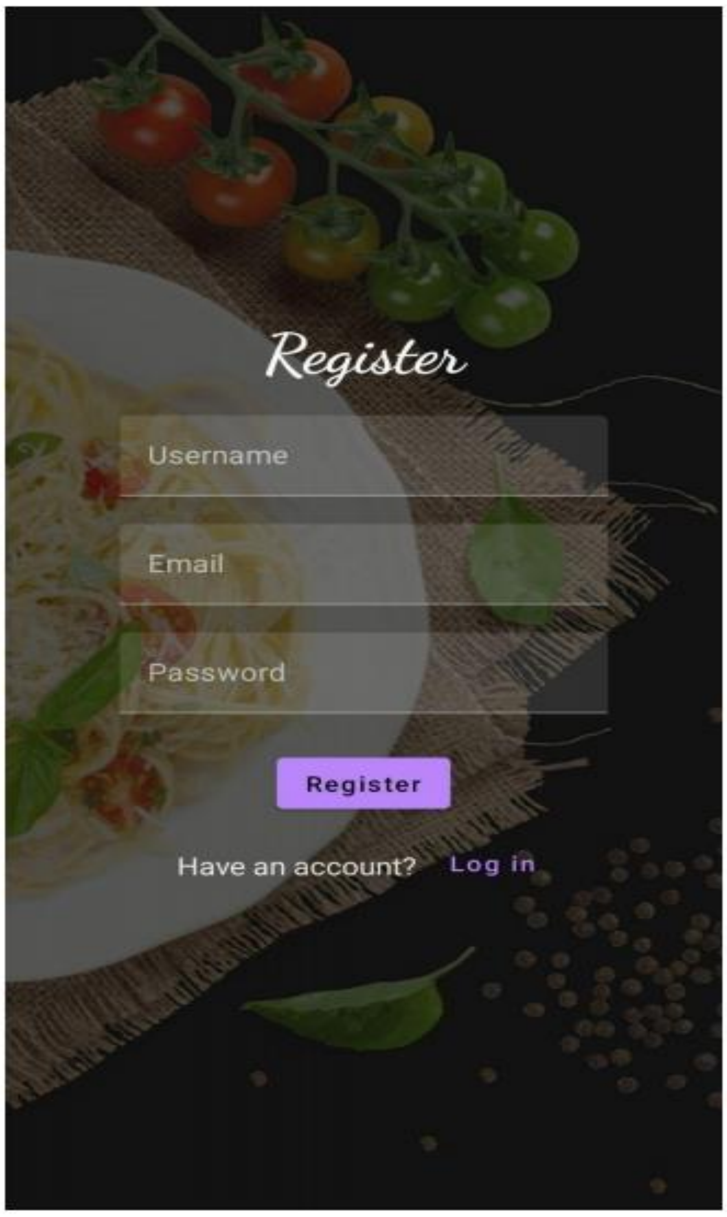
3. RESULT

Final Output of the Application :

Admin Module: After logging in with Admin Credentials which are hard coded.
Login Page :

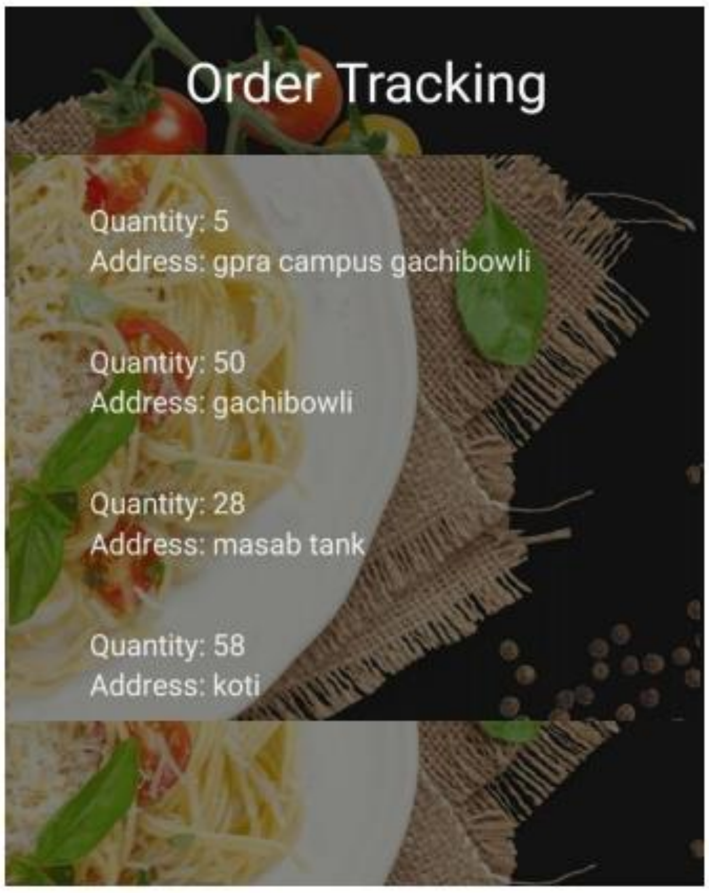


Register Page :



After logging in with Admin Credentials which are hard coded. Password must be "admin" .

Admin page:



Order Tracking

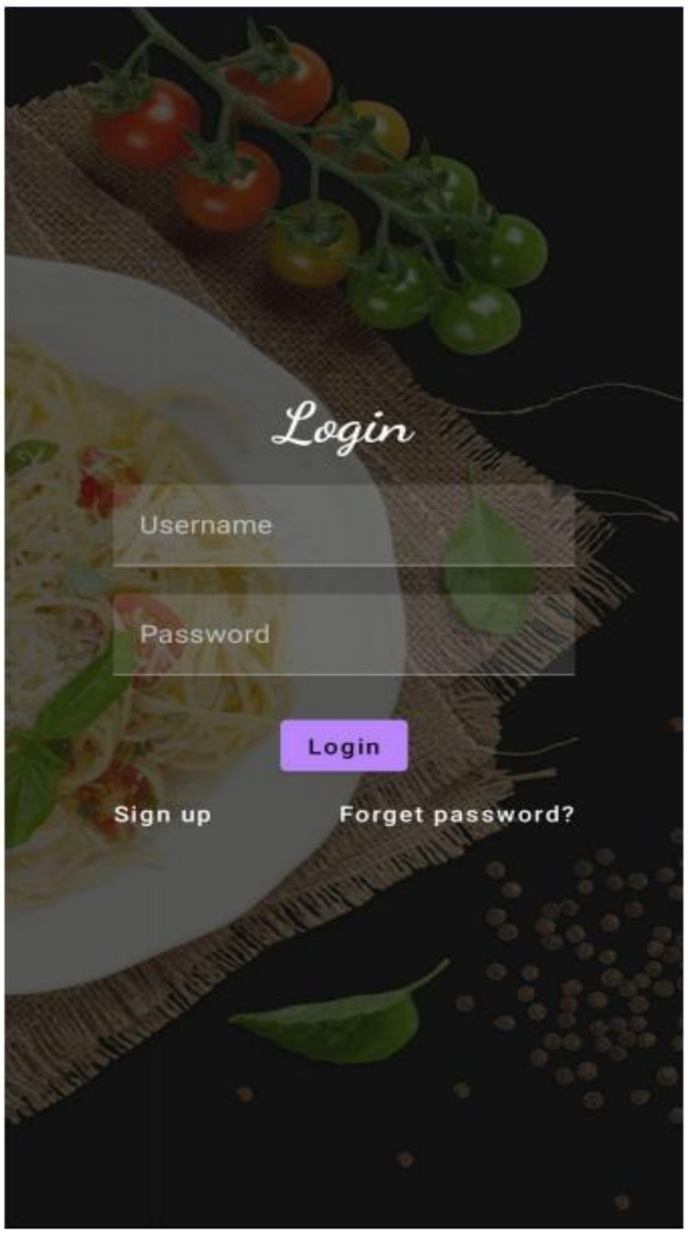
Quantity: 5
Address: gpra campus gachibowli

Quantity: 50
Address: gachibowli

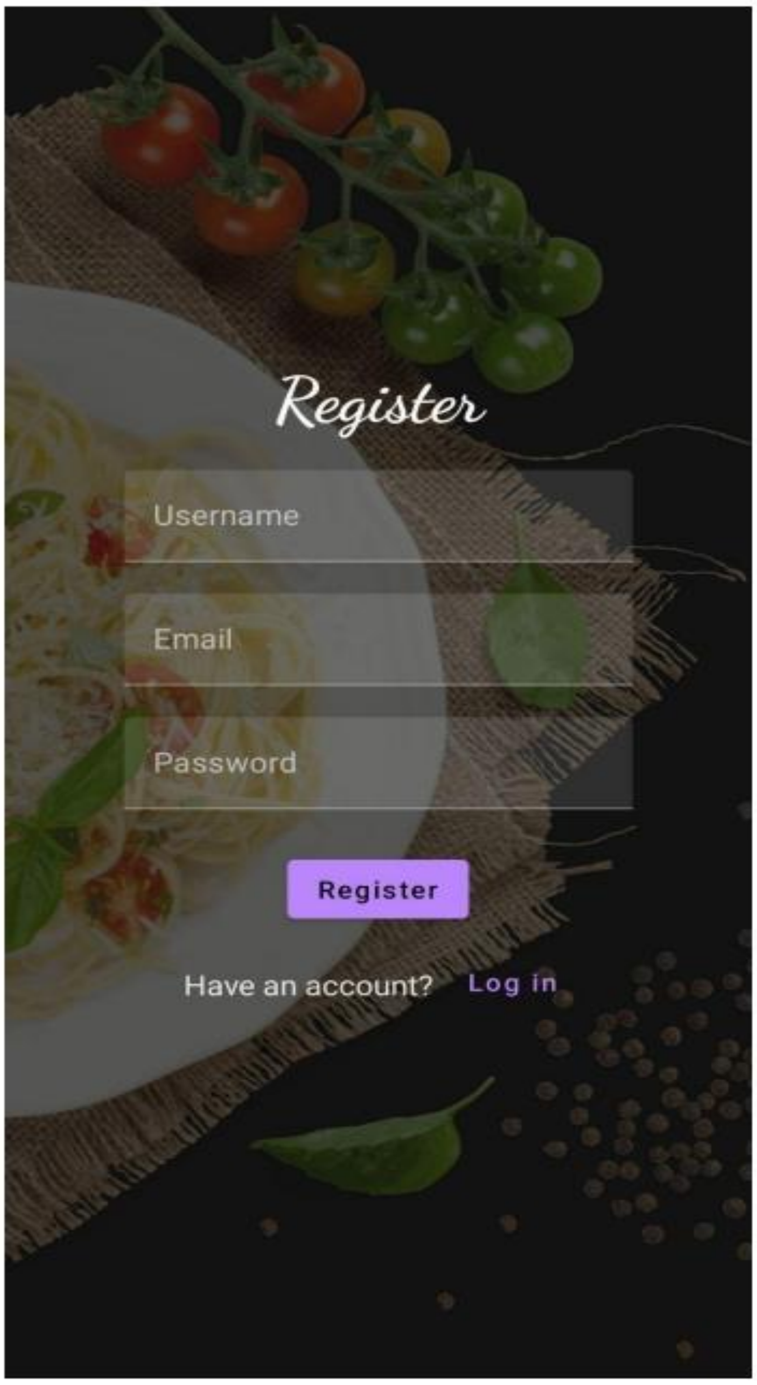
Quantity: 28
Address: masab tank

Quantity: 58
Address: koti

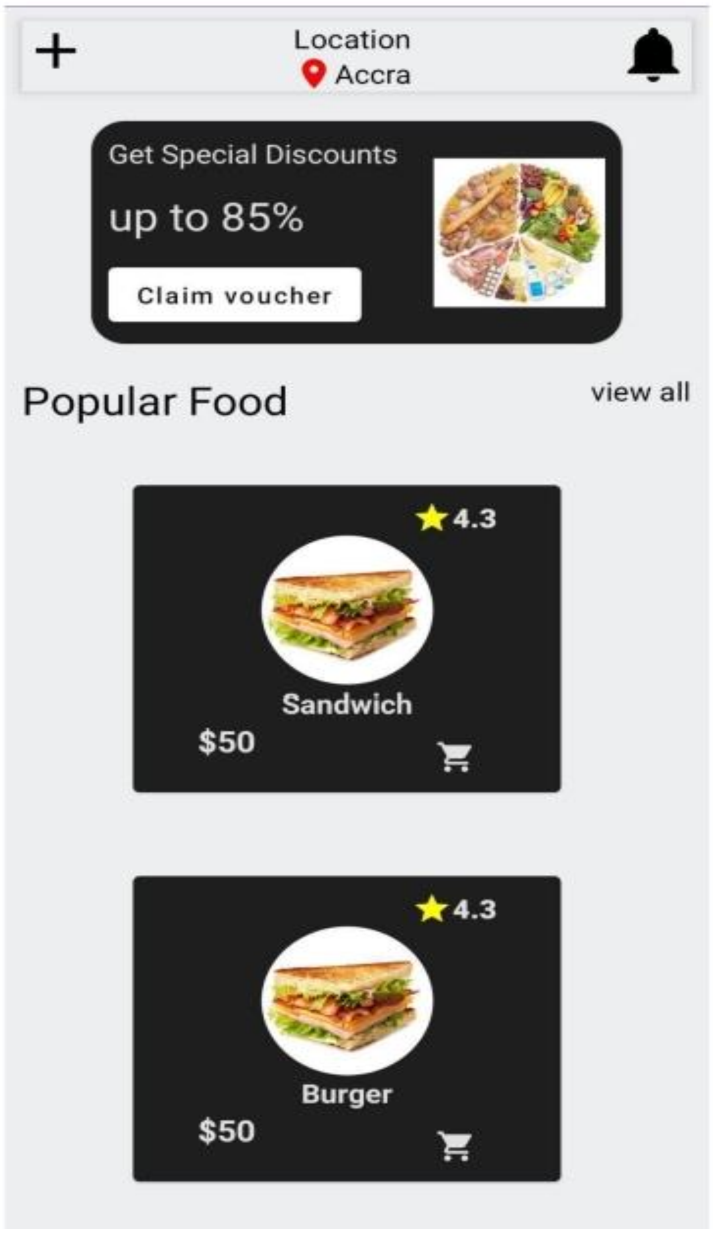
User Module:
Login Page :



Register Page :



Main Page :


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

4.ADVANTAGES & DISADVANTAGES

Advantage

1.Workaholics, Don’t Starve Anymore

Being a profession-driven person,we afford to miss on our diet but never to miss on our deadlines. But not to worry anymore! You can quickly order your favorite quick munchies from one of the nearest restaurants and grab a bite in no while refilling up your stomach and you are back on the trail.

2.Ladies, You Can Enjoy The Parties Too!

When you finally have your long-awaited guests coming home; Ladies, you can quickly order delicious food items, single or many, with just a few taps on your mobile screen.

3.Urban Restaurants, Reach Out to Remote Foodies

You captured the foodies of the complete city! Are you sure? Why not extend out the reach to the remote foodies.With the online food delivery in place, you can be available to the remote food-lovers by being visible with your restaurant menu and giving the option to deliver in their area.

4.Pinchpenny? Get Cashback

You love the restaurant food but you want to save money as well, don’t worry. We have got discounts for you. With this best online food ordering system, you get excellent deals for cashback and discount offers while ordering for delivery or eating at a restaurant. This is one of the significant advantages of online food ordering system.

5.Reserve that Quiet Side Table for Your Next Gathering

Tired of facing the problem of last-minute bookings and cancellations of the tables at the time of your gatherings.the meeting and get notified in advance if there is any cancellation and yet get a chance to book the table beforehand in another restaurant.

Dissadvatage

1.Deliverymen Put Themselves in Danger

Whether it is a heatwave boiling down the city or it is snowing or raining heavily, a Delivery Boy is waiting outside the restaurant to pick and deliver your order. This is one of the disadvantages of ordering food online

2.Disguised Increased Expense

We surely get attracted by yummy-looking food pictures on the app and a small but highlighting banner of cashback offer.However, we forget that despite cashback, it is costing us higher than the food which we can cook with the groceries available using all our magical cooking skills and spend blindly ordering the food online. This can be considered one of the disadvantages of online food ordering for customers.

3.Revenue Conflicts Between The Restaurant and delivery Providers

Not every restaurant owner can afford to employ ten delivery boys and bear all the transport and remuneration expenditure; so, they choose to contract with the delivery service providers through these apps. This brings the disadvantages of food delivery service.

5.APPLICATION

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 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.

6.CONCULSION

We conclude that snacks are an important part of the diet and involve the consumption of both favorable and less favorable foods. Snacks eaten at home or at work/school were generally healthier than snacks consumed during visits to other private households or at restaurants/café/fast-food outlets.

Snack consumption has been reported increase over recent decades. Little is known about possible associations between snack composition and snack eating location. In the present study, we aimed to describe the contribution of snacks to dietary intake in Norwegian adults and to investigate whether the

7.FUTURE SCOPE

It is easy to forget how far snack food delivery services have come while waiting for the ordered meal and watching Sunday Night NFL. Be it chicken fingers and fries or chicken sandwich or mac ‘n’ cheese or sushi or pad thai or iced coffee or milk tea or veggie burger- the most ordered online snack foods – are so quickly delivered that it feels like the food delivery apps have always existed.

In such a technological era, people find it difficult to visit restaurants. Most often, they are unable to manage time for picking up their order. Therefore, most of them like to use the food delivery app. It provides them with the option to choose the menu as per their choice and place the order instantly with a few clicks.

Future:

- Discount/Rewards, Cashback, and Loyalty Programs. …
- Real-Time GPS Tracking. …
- Easy Payment Options. …
- Social Media Integration. …

- Reviews & Ratings. …
- Easy Order Placement. …
- Order scheduling and pickup.

8. APPENDIX

Creating the database classes

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 USER DATABASE CLASSES

```
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

    }
```

```
        }

    }

}

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()
    }
```

CREATE AN ORDER DATABASE HELPER CLASS

```
  @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_QUANTI
TY)),

            Address =
cursor.getString(cursor.getColumnIndex(COLUMN_ADDRES
S)),

         )

      }

      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_QUANTI
TY)),

        Address =
cursor.getString(cursor.getColumnIndex(COLUMN_ADDRES
S)),

      )
```

```
        }

        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_QUANTI
TY)),

                Address =
cursor.getString(cursor.getColumnIndex(COLUMN_ADDRES
S)),

            )

            Orders.add(order)

        } while (cursor.moveToNext())

    }

    Cursor.close()

    Db.close()
```

```
        Return orders

    }

}
```