

GOVERNMENT COLLEGE OF TECHNOLOGY, COIMBATORE

SNACK SQUAD

**A CUSTOMIZABLE SNACK ORDERING AND
DELIVERY APP**

Submitted by:

P.Manoranjini (71772218126)

S.Pavithra(71772218136)

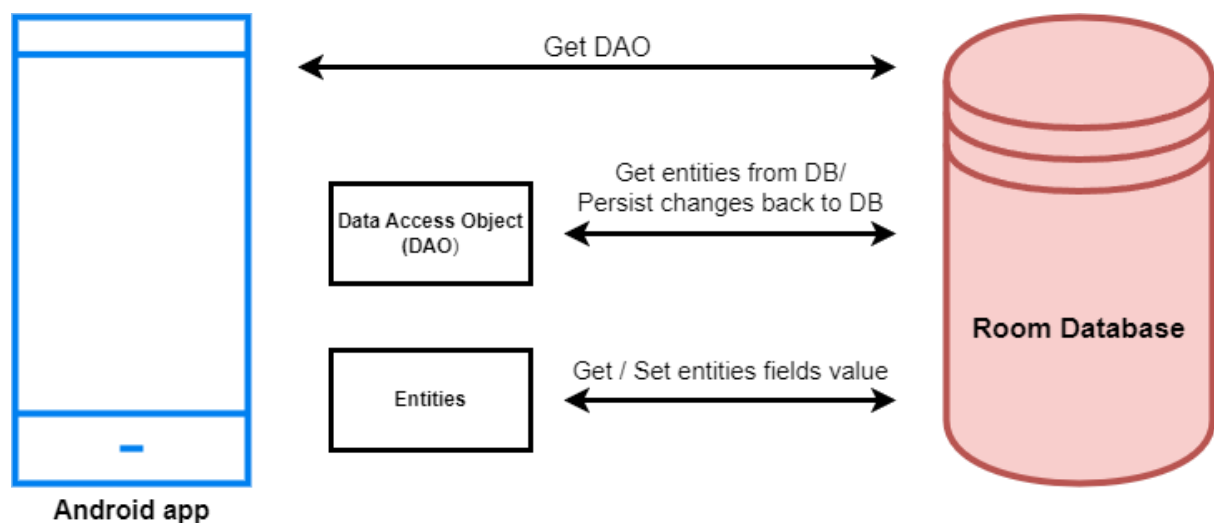
V.Srividhya(71772218150)

R.Yukesh Raj (71772218160)

Project Description:

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.

Architecture:



Purpose of project:

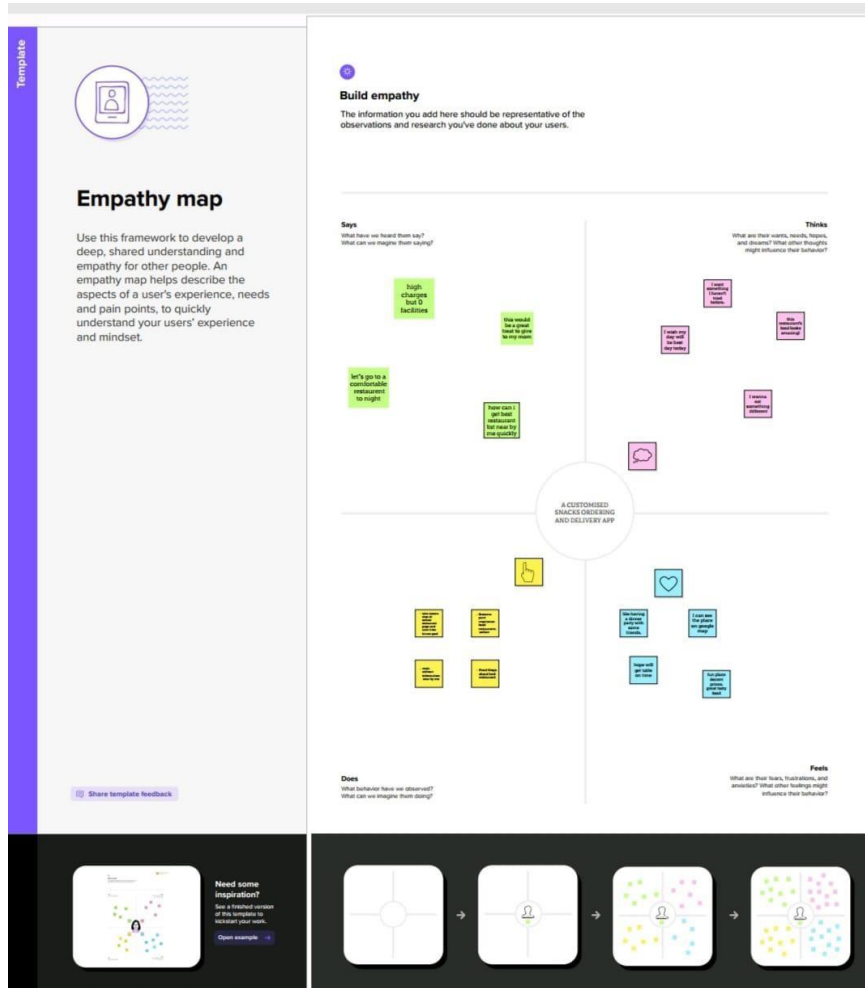
The purpose of Snack Squad, a customizable snack ordering and delivery app, is to provide a convenient and personalized way for people to order snacks and have them delivered to their location. The app is designed to streamline the process of selecting and ordering snacks, while also giving users the ability to customize their orders based on their individual preferences and dietary restrictions.

The app can be used by individuals or businesses, such as offices or schools, to easily order snacks for themselves or for a group of people. Snack Squad offers a wide variety of snacks to choose from, including healthy options, and allows users to customize their orders to include only the snacks they want.

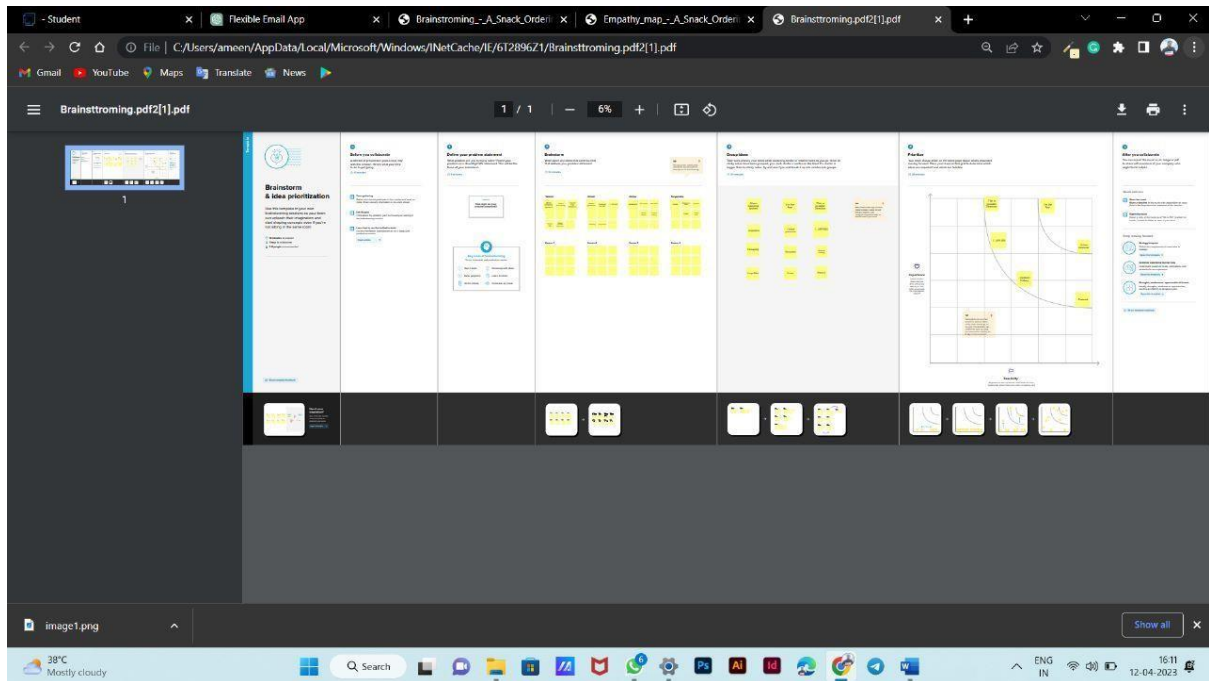
By using Snack Squad, users can save time and effort by avoiding trips to the store and the hassle of choosing and purchasing snacks themselves. The app also offers a delivery service, so snacks can be conveniently delivered to the user's location.

Problem Definition and Design Thinking:

Empathy map:



Ideation & Brainstroming map



12:00

1.1 MB/s 93%

Login

Username
manoranjini

Password
mano6268

Login

[Sign up](#)

[Forget password?](#)



Location

Accra



Get Special Discounts

up to 85%

Claim voucher



Popular Food

[view all](#)

★ 4.3



Sandwich

\$50



★ 4.3



Burger

\$50



4:20

Location

Accra

+

🔔

Get Special Discounts

up to 85%

Claim voucher

Popular Food

view all

★ 4.3

Sandwich

\$50

🛒

★ 4.3

Burger

\$50

🛒

★ 4.3

All In One

\$50

🛒

★ 4.3

Pasta

\$50

🛒

4:20

Location

Accra

+

🔔

Get Special Discounts

up to 85%

Claim voucher

Popular Food

view all

★ 4.3

Sandwich

\$50

🛒

★ 4.3

Burger

\$50

🛒

★ 4.3

All In One

\$50

🛒

★ 4.3

Pasta

\$50

🛒

4:18



Vol
LTE



5%

Register

Username

Email

Password

Register

Have an account? [Log in](#)

Advantages of a Customizable Snack Ordering and Delivery App:

Convenience: A customizable snack ordering and delivery app can make it easier for customers to order their favorite snacks at any time of the day or night, without having to leave their home or office.

Personalization: Customizable apps allow customers to personalize their snack orders according to their taste and preferences.

Increased Sales: A snack ordering and delivery app can increase sales for a snack shop, especially if the app allows customers to order from multiple locations.

Disadvantages of a Customizable Snack Ordering and Delivery App:

Technical Issues: Technical issues can arise with the app, causing it to malfunction or crash, which can result in lost sales and customer frustration.

Security Concerns: Personal data and payment information stored on the app may be at risk of being hacked, leading to identity theft and fraud.

Dependence on Technology: A snack shop that relies solely on a customizable app for ordering and delivery may experience difficulty if the app experiences technical issues or goes offline.

Application:

Customizable Snack Ordering and Delivery App can have a variety of applications, including:

Snack Shops: Snack shops can use the app to allow customers to order their favorite snacks online and have them delivered to their doorstep.

Cafes: Cafes can use the app to offer their customers a convenient way to order coffee, pastries, and other menu items for pickup or delivery.

Food Trucks: Food trucks can use the app to notify customers of their location and allow them to order food for pickup or delivery.

Conclusion:

A customizable snack ordering and delivery app is a great solution to address the growing demand for convenient and personalized snack services. The app should offer a wide range of snack options, allow for customization of orders, provide real-time tracking of deliveries, and integrate a secure payment system.

Such an app can benefit both customers and snack vendors by providing a more streamlined and efficient way to connect and conduct transactions. Additionally, the app could leverage user data to offer personalized recommendations and promotions, further enhancing the user experience.

Overall, a customizable snack ordering and delivery app has the potential to revolutionize the snack industry, providing a modern and convenient solution for busy consumers looking for a quick snack fix.

Future scope:

The future scope of a customizable snack ordering and delivery app is very promising. As technology continues to advance and consumer behavior evolves, the demand for convenient and personalized snack services is only going to increase. Here are some potential future developments for such an app:

Integration of AI and Machine Learning - AI and machine learning algorithms can be integrated into the app to analyze user data and offer personalized recommendations for snacks and promotions.

Overall, the future scope of a customizable snack ordering and delivery app is very exciting and full of potential. By embracing new technologies and expanding their services, such an app can continue to grow and provide even more value to its customers and partners

Integration with Smart Home Devices - The app could be integrated with smart home devices like Amazon Alexa or Google Home to allow users to order snacks using voice commands.

Expansion to New Markets - The app could expand to new markets, both nationally and globally, to reach a wider customer base and offer more snack options.

Appendix :

A.source code :[MainPage.kt](#)

```
package com.example.snackordering

import android.annotation.SuppressLint import
import android.content.Context import android.os.Bundle import
import android.widget.Toast import
import androidx.activity.ComponentActivity import
import androidx.activity.compose.setContent import
import androidx.annotation.DrawableRes import
import androidx.annotation.StringRes import
import androidx.compose.foundation.Image import
import androidx.compose.foundation.background import
import androidx.compose.foundation.layout.* import
import androidx.compose.foundation.shape.CircleShape import
import androidx.compose.foundation.shape.RoundedCornerShape import
import androidx.compose.material.* import
import androidx.compose.material.icons.Icons import
import androidx.compose.material.icons.filled.* import
import androidx.compose.runtime.Composable import
import androidx.compose.ui.Alignment import
import androidx.compose.ui.Modifier import
```

```

androidx.compose.ui.draw.clip                                import
androidx.compose.ui.graphics.Color
import      androidx.compose.foundation.lazy.LazyColumn
import      androidx.compose.foundation.lazy.items    import
androidx.compose.material.Text                               import
androidx.compose.ui.unit.dp
import      androidx.compose.ui.graphics.RectangleShape
import      androidx.compose.ui.layout.ContentScale   import
androidx.compose.ui.platform.LocalContext                 import
androidx.compose.ui.res.painterResource                   import
androidx.compose.ui.res.stringResource                    import
androidx.compose.ui.text.font.FontWeight                 import
androidx.compose.ui.unit.sp
import androidx.core.content.ContextCompat.startActivity import
com.example.snackordering.ui.theme.SnackOrderingTheme

import android.content.Intent as Intent1

```

```

class MainPage : ComponentActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
super.onCreate(savedInstanceState)      setContent {

```

```

        SnackOrderingTheme {

            // A surface container using the 'background' color from the
theme

            Surface(

                modifier = Modifier.fillMaxSize(),
color = MaterialTheme.colors.background

            ) {

                FinalView(this)

                val context = LocalContext.current

                //PopularFoodColumn(context)

            }

        }

    }

}

```

@Composable

```
fun TopPart() {
```

```
    Row(
```

```
        modifier = Modifier
```

```

        .fillMaxWidth()

        .background(Color(0xffeceef0)),
Arrangement.SpaceBetween

    ) {

        Icon(

            imageVector = Icons.Default.Add, contentDescription = "Menu
Icon",

            Modifier

                .clip(CircleShape)

                .size(40.dp), tint =

                Color.Black,

            )

        Column(horizontalAlignment = Alignment.CenterHorizontally) {

            Text(text = "Location", style =

                MaterialTheme.typography.subtitle1, color = Color.Black)

            Row {

                Icon(

                    imageVector    =    Icons.Default.LocationOn,

                    contentDescription = "Location", tint = Color.Red,

                )

                Text(text = "Accra" , color = Color.Black)

            }

```



```

    }

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

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

```

```

@Composable
fun CardPart() {
    Card(modifier = Modifier.size(width = 310.dp, height =
    150.dp), RoundedCornerShape(20.dp)) {
        Row(modifier = Modifier.padding(10.dp),
            Arrangement.SpaceBetween) {
            Column(verticalArrangement =
            Arrangement.spacedBy(12.dp)) {
                Text(text = "Get Special Discounts")
                Text(text = "up to 85%", style =

```

```
MaterialTheme.typography.h5)
```

```
        Button(onClick = {}, colors =  
ButtonDefaults.buttonColors(Color.White)) {
```

```
            Text(text = "Claim voucher", color =  
MaterialTheme.colors.surface)
```

```
        }
```

```
    }
```

```
    Image(  
        painter = painterResource(id = R.drawable.food_tip_im),  
        contentDescription = "Food Image", Modifier.size(width =  
100.dp, height = 200.dp)
```

```
    )
```

```
    }
```

```
}
```

```
}
```

```
@Composable
```

```
fun PopularFood(  
    @DrawableRes drawable: Int,  
    @StringRes text1: Int,    context:  
Context
```

```
) {
```

```

Card(
    modifier = Modifier
        .padding(top=20.dp, bottom = 20.dp, start = 65.dp)
        .width(250.dp)

) {
    Column(
        verticalArrangement = Arrangement.Top,
        horizontalAlignment = Alignment.CenterHorizontally
    ) {
        Spacer(modifier = Modifier.padding(vertical = 5.dp))
        Row(
            modifier = Modifier
                .fillMaxWidth(0.7f), Arrangement.End
        ) {
            Icon(
                imageVector = Icons.Default.Star,
                contentDescription = "Star Icon", tint =
                Color.Yellow
            )
            Text(text = "4.3", fontWeight = FontWeight.Black)
        }
    }
}

```

```

    }

    Image(
        painter = painterResource(id = drawable),
        contentDescription = "Food Image",        contentScale
        = ContentScale.Crop,        modifier = Modifier
        .size(100.dp)

        .clip(CircleShape)
    )

    Text(text = stringResource(id = text1), fontWeight =
    FontWeight.Bold)

    Row(modifier = Modifier.fillMaxWidth(0.7f),
    Arrangement.SpaceBetween) {

        /*TODO Implement Prices for each card*/

        Text(
            text = "$50",

            style = MaterialTheme.typography.h6,
            fontWeight = FontWeight.Bold,        fontSize =
            18.sp
        )

        IconButton(onClick = {

```

```

        //var                no=FoodList.lastIndex;

//Toast.

        val intent = Intent1(context, TargetActivity::class.java)
        context.startActivity(intent)

    )) {

        Icon(

            imageVector        =        Icons.Default.ShoppingCart,
contentDescription = "shopping cart",

        )

    }

}

}

}

}

```

```

private val FoodList = listOf(

    R.drawable.sandwish to R.string.sandwich,

    R.drawable.sandwish to R.string.burgers,

```

```
R.drawable.pack to R.string.pack,  
R.drawable.pasta to R.string.pasta,  
R.drawable.tequila to R.string.tequila,  
R.drawable.wine to R.string.wine,  
R.drawable.salad to R.string.salad,  
R.drawable.pop to R.string.popcorn  
).map { DrawableStringPair(it.first, it.second) }
```

```
private data class DrawableStringPair(  
    @DrawableRes val drawable: Int,  
    @StringRes val text1: Int  
)
```

```
@Composable
```

```
fun App(context: Context) {
```

```
    Column(  
        modifier = Modifier  
            .fillMaxSize()  
            .background(Color(0xffeceef0))  
            .padding(10.dp),
```

```

        verticalArrangement = Arrangement.Top,
        horizontalAlignment = Alignment.CenterHorizontally
    ) {
        Surface(modifier = Modifier, elevation = 5.dp) {
            TopPart()
        }
        Spacer(modifier = Modifier.padding(10.dp))
        CardPart()

        Spacer(modifier = Modifier.padding(10.dp))
        Row(modifier = Modifier.fillMaxWidth(),
            Arrangement.SpaceBetween) {
            Text(text = "Popular Food", style =
                MaterialTheme.typography.h5, color = Color.Black)

            Text(text = "view all", style =
                MaterialTheme.typography.subtitle1, color = Color.Black)
        }

        Spacer(modifier = Modifier.padding(10.dp))
        PopularFoodColumn(context) // <- call the function with parentheses
    }
}

```

@Composable

fun PopularFoodColumn(context: Context) {

LazyColumn(

modifier = Modifier.fillMaxSize(),

content = {

items(FoodList) { item ->

PopularFood(context = context, drawable =
item.drawable, text1 = item.text1) abstract class

Context

}

},

verticalArrangement = Arrangement.spacedBy(16.dp)) }

@SuppressLint("UnusedMaterialScaffoldPaddingParameter")

@Composable

fun FinalView(mainPage: MainPage) {

SnackOrderingTheme {


```

        Scaffold() {
            val context = LocalContext.current
            App(context)
        }
    }
}

```

LoginActivity.kt

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

```

import android.content.Context import
android.content.Intent          import
android.os.Bundle

import androidx.activity.ComponentActivity import
androidx.activity.compose.setContent      import
androidx.compose.foundation.Image         import
androidx.compose.foundation.layout.*      import
androidx.compose.material.*              import
androidx.compose.runtime.*               import
androidx.compose.ui.Alignment             import
androidx.compose.ui.Modifier             import
androidx.compose.ui.graphics.Color       import
androidx.compose.ui.layout.ContentScale  import

```

```

androidx.compose.ui.res.painterResource      import
androidx.compose.ui.text.font.FontFamily     import
androidx.compose.ui.text.font.FontWeight     import
androidx.compose.ui.unit.dp                  import
androidx.compose.ui.unit.sp                  import
androidx.core.content.ContextCompat import
com.example.snackordering.ui.theme.SnackOrderingTheme

```

```

class LoginActivity : ComponentActivity() {
    private lateinit var databaseHelper: UserDatabaseHelper
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)        databaseHelper =
        UserDatabaseHelper(this)        setContent {

            SnackOrderingTheme {

                // A surface container using the 'background' color from the
                theme

                Surface(

                    modifier        =        Modifier.fillMaxSize(),

                    color = MaterialTheme.colors.background

                ) {

                    LoginScreen(this, databaseHelper)

                }
            }
        }
    }
}

```

```

        }
    }
}
}

```

@Composable

```

fun LoginScreen(context: Context, databaseHelper:
    UserDatabaseHelper) {

```

```

    Image(painterResource(id = R.drawable.order),
        contentDescription = "", alpha = 0.3F,
        contentScale = ContentScale.FillHeight,

    )

```

```

    var username by remember { mutableStateOf("") } var
    password by remember { mutableStateOf("") } var error by
    remember { mutableStateOf("") }

```

```

    Column(
        modifier = Modifier.fillMaxSize(),
        horizontalAlignment = Alignment.CenterHorizontally,
        verticalArrangement = Arrangement.Center
    )

```

```

) {

    Text(

        fontSize = 36.sp,

        fontWeight = FontWeight.ExtraBold,
        fontFamily = FontFamily.Cursive,          color
        = Color.White,          text = "Login"

    )

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

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

    )

    TextField(          value
= password,

```

```

        onChange = { password = it },
label = { Text("Password") },          modifier =
Modifier.padding(10.dp)
        .width(280.dp)
    )

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

    Button(
onClick = {
        if (username.isNotEmpty() && password.isNotEmpty()) {
            val user =
databaseHelper.getUserByUsername(username)
            if (user != null && user.password == password) {
error = "Successfully log in"
                context.startActivity(
Intent(
                    context,
                        MainPage::class.java

```

```

        )
    )
    //onLoginSuccess()
}

if (user != null && user.password == "admin") {
error = "Successfully log in"
context.startActivity(
Intent(
context,
AdminActivity::class.java
)
)
} else {
error = "Invalid username or password"
}

} else {
error = "Please fill all fields"
}

},
modifier = Modifier.padding(top = 16.dp)
) {
Text(text = "Login")
}

```

```

Row {
    TextButton(onClick = {context.startActivity(
        Intent(
context,
            MainActivity::class.java
        )
    ))
    )
    { Text(color = Color.White,text = "Sign up") }
    TextButton(onClick = {
    })

    {
        Spacer(modifier = Modifier.width(60.dp))
        Text(color = Color.White,text = "Forget password?")
    }
}

}

private fun startMainPage(context: Context) {    val
intent = Intent(context, MainPage::class.java)

    ContextCompat.startActivity(context, intent, null)

```

}