SNACK SQUAD

A Customizable Snack Ordering and Delivery App

Submitted by:

SREEJITH V S

SANJAY V K

HARISH K

GOHUL C V

1. INTRODUCTION :

1.1 OVERVIEW:

As an AI language model, I don't have access to specific information about every entity or organization that exists, but based on my general knowledge and internet research, Snack Squad may refer to a group of people who enjoy snacks and try out different snacks or snack products, sharing their reviews or experiences with others. It could also refer to a brand or company that produces or sells snacks, or a team or group of employees within a larger organization who are responsible for providing snacks or refreshments for their colleagues.

1.2 PURPOSE:

The use of Snack Squad can be beneficial in various ways, depending on the specific goals or objectives of the organization or group. Some potential uses and benefits of Snack Squad are:

1. Boosting team morale and engagement: Snack Squad can create a fun and engaging environment that can help increase employee satisfaction, engagement, and motivation.

2. Promoting workplace wellness: Snack Squad can help promote healthy snacking habits, offer healthier options, and encourage team members to take breaks and recharge.

3. Fostering team bonding: Snack Squad can create a sense of community and provide opportunities for team members to bond and socialize.

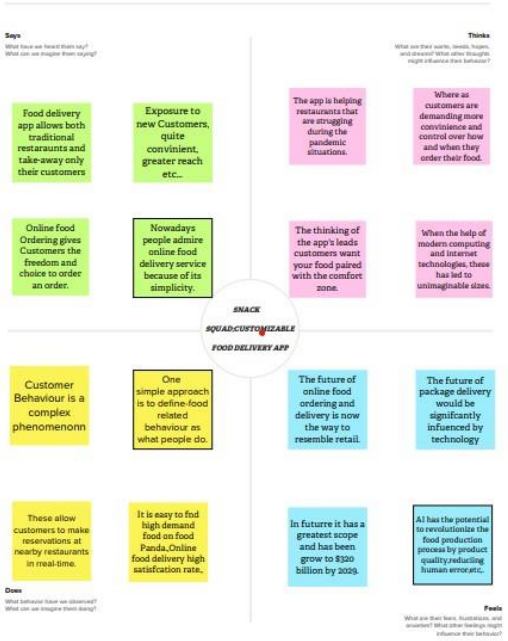
4. Enhancing productivity: Providing snacks can help team members avoid hunger-induced distractions, stay focused, and maintain energy levels throughout the day.

5. Building brand awareness: If Snack Squad is used by a company or brand, it can help increase brand awareness and customer loyalty by providing a positive and memorable experience for customers.

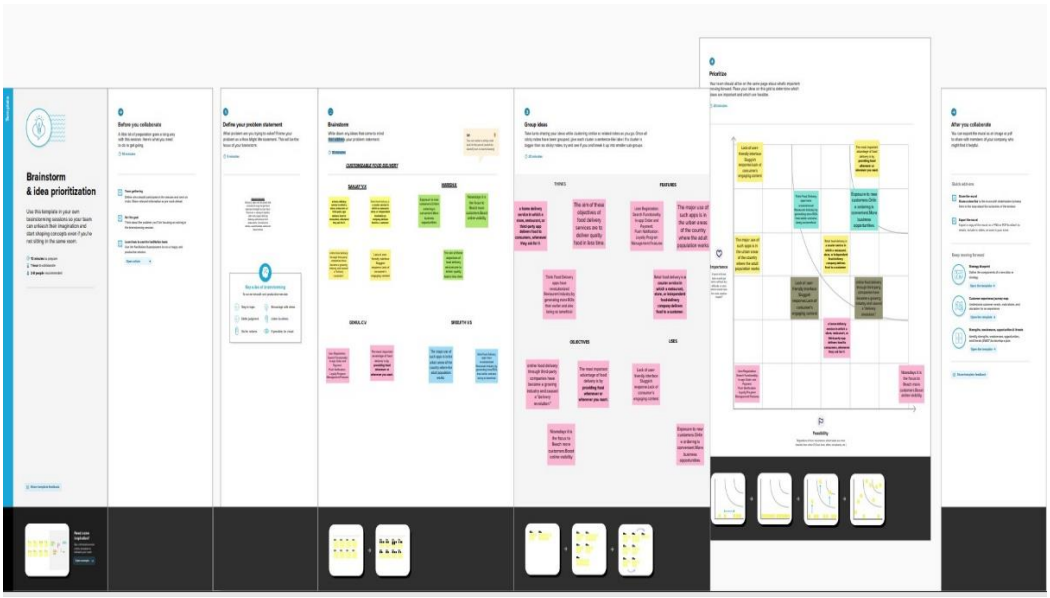
Overall, Snack Squad can be a simple yet effective tool to improve workplace culture, promote well-being, and enhance team performance

2. PROBLEM DEFINITION & DESIGN THINKING:

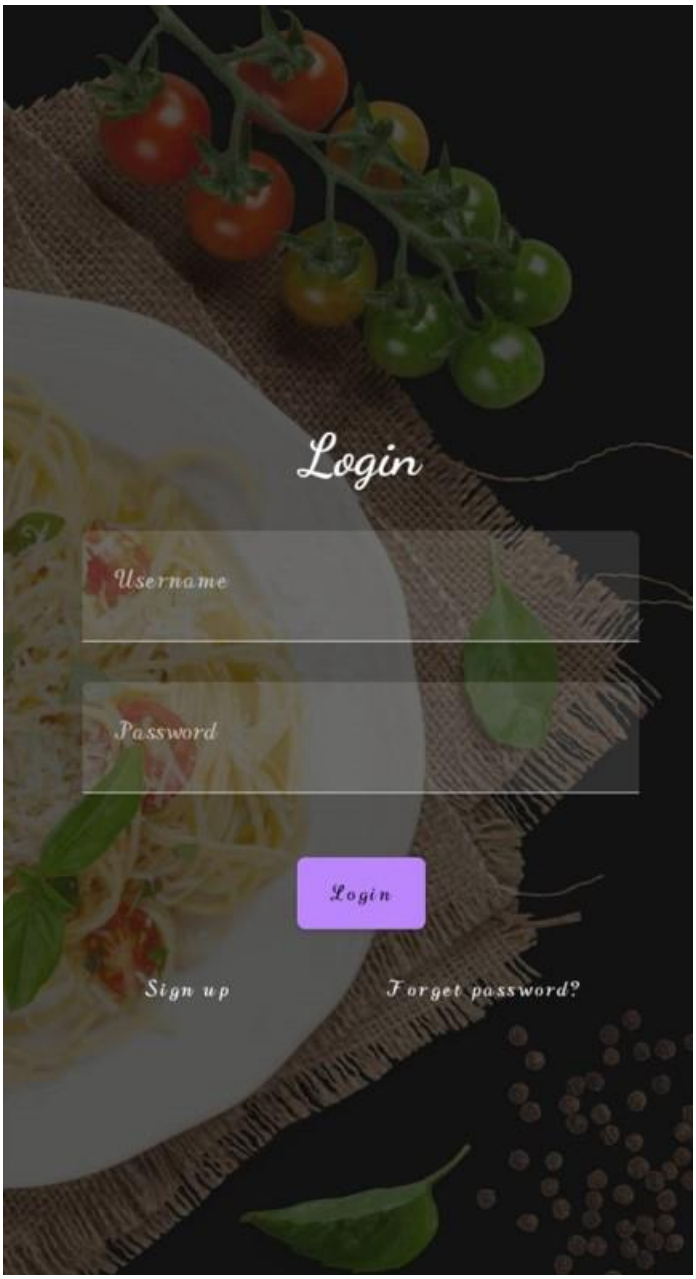
2.1 EMPATHY MAP:



2.2 IDEATION & BRAINSTORMING MAP



3. RESULT:



Location
Accra

Get Special Discounts

up to 85%

Claim voucher

Popular Food

view all

4.3

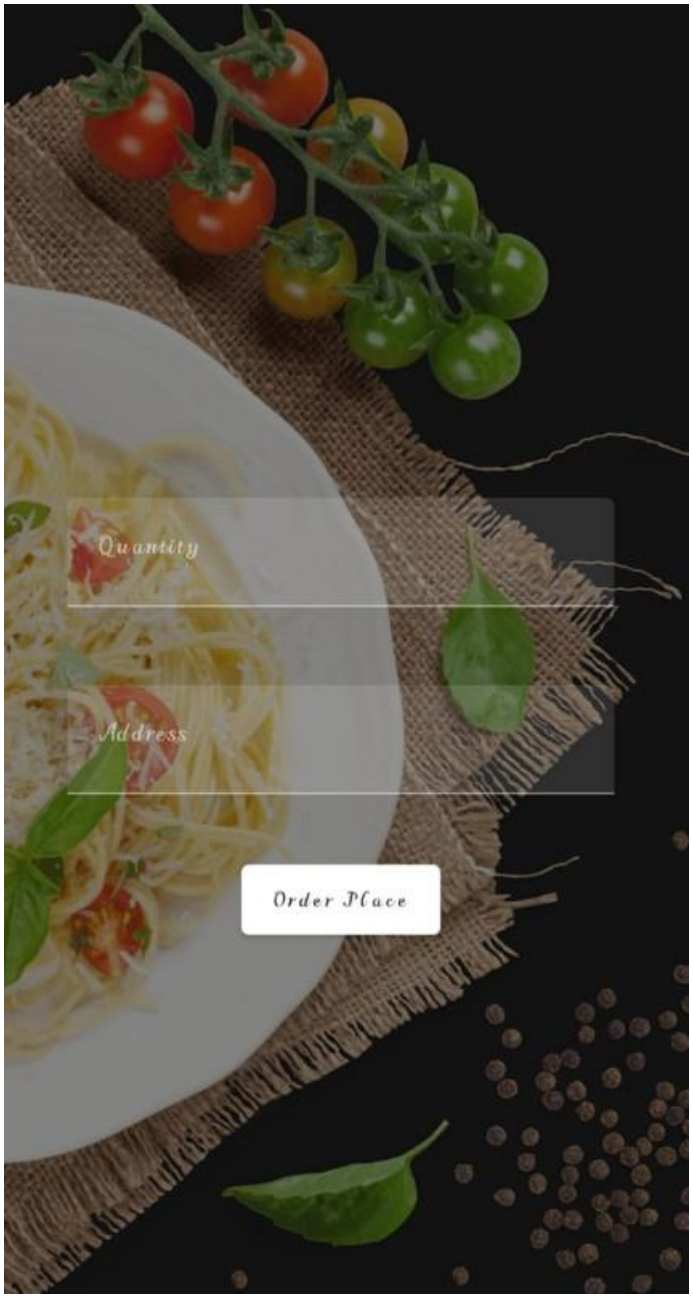
Wine

$50

4.3

Salad

$50


Quantity
Address
Order Place

4. ADVANTAGES & DISADVANTAGES:

Sure, here are some potential advantages and disadvantages of having a Snack Squad:

Advantages:

- Boosts team morale and engagement.
- Provides an opportunity for team members to bond and socialize.
- Promotes healthy snacking habits and well-being.
- Enhances productivity and focus by providing fuel throughout the day.
- Can be a unique and memorable way to promote brand awareness.

Disadvantages:

- Can be a distraction if team members spend too much time snacking instead of working.
- Can be expensive to maintain, depending on the snacks provided and the frequency of restocking.
- May not appeal to everyone's taste preferences or dietary restrictions.

- Can create competition or conflict if some team members are more interested in certain snacks than others.
- May require extra effort and resources to manage and maintain, such as organizing snack schedules and ensuring availability and variety of snacks

5. Application:

Snack Squad is a mobile application that allows users to order snacks and drinks from their favorite local convenience stores and have them delivered to their doorstep. The application can be used in several ways, including:

1. Convenience: Snack Squad offers users the convenience of ordering snacks and drinks from their favorite local convenience stores and having them delivered directly to their doorstep. This can be especially useful for people who are busy or unable to leave their homes.

2. Variety: Snack Squad offers a wide variety of snacks and drinks from different convenience stores, allowing users to choose from a range of options that may not be available at their local store.

3. Customization: Snack Squad allows users to customize their orders by selecting specific snacks and drinks, adding notes for the delivery driver, and even choosing a specific delivery time.

4. Time-saving: Snack Squad can save users time by eliminating the need to travel to a convenience store to purchase snacks and drinks. Instead, users can order from their phone and have their items delivered in as little as 30 minutes.

5. Support local businesses: Snack Squad partners with local convenience stores, allowing users to support small businesses in their community while enjoying their favorite snacks and drinks.

Overall, Snack Squad is a convenient and time-saving application that offers a wide variety of snacks and drinks from local convenience stores. It allows users to customize their orders and support local businesses, making it a valuable resource for anyone looking to satisfy their snack cravings.

6. CONCLUSION:

In conclusion, Snack Squad is a mobile application that offers a convenient way for users to order snacks and drinks from their favorite local convenience stores and have them delivered to their doorstep. The application provides a wide variety of snacks and drinks, allowing users to choose from different options that may not be available at their local store.

Snack Squad also allows users to customize their orders, add notes for the delivery driver, and even choose a specific delivery time. This level of customization can help ensure that users get exactly what they want and need.

Furthermore, Snack Squad supports local businesses by partnering with local convenience stores. This is an excellent way for users to support small businesses in their community while enjoying their favorite snacks and drinks.

Overall, Snack Squad offers a convenient and time-saving solution for people who want to satisfy their snack cravings without leaving their homes. With its

customization options and support for local businesses, Snack Squad is an excellent resource for anyone looking for a quick and easy snack delivery service.

7. FUTURE SCOPE:

As an AI language model, I don't have information on a specific "Snack Squad" company or organization. However, in general, the future scope of any snack-related business would depend on various factors, such as market trends, consumer preferences, and competition.

Assuming that "Snack Squad" is a food business that specializes in snacks, its future scope could depend on its ability to innovate and offer unique and healthy snack options that appeal to the changing tastes and preferences of consumers. It could also expand its market reach through strategic partnerships, collaborations, and by leveraging social media and online platforms to increase brand awareness and reach more customers.

Additionally, the business could consider exploring different channels for distribution, such as vending machines, subscription-based snack boxes, or partnerships with retail stores and supermarkets.

Overall, the key to success for any snack-related business would be to remain agile, adapt to changing market

conditions and consumer preferences, and consistently offer high-quality products that meet the needs of their customers.

8. APPENDIX

Source Code:

LoginActivity:

```
package com.example.snackordering

import android.content.Context
import android.content.Intent
import android.os.Bundle
import androidx.activity.ComponentActivity
import androidx.activity.compose.setContent
import androidx.compose.foundation.Image
import androidx.compose.foundation.layout.*
import androidx.compose.material.*
import androidx.compose.runtime.*
import androidx.compose.ui.Alignment
import androidx.compose.ui.Modifier
import androidx.compose.ui.graphics.Color
import androidx.compose.ui.layout.ContentScale
import androidx.compose.ui.res.painterResource
import androidx.compose.ui.text.font.FontFamily
import androidx.compose.ui.text.font.FontWeight
import androidx.compose.ui.unit.dp
import androidx.compose.ui.unit.sp
import androidx.core.content.ContextCompat
import com.example.snackordering.ui.theme.SnackOrderingTheme

class LoginActivity : ComponentActivity() {
    private lateinit var databaseHelper: UserDatabaseHelper
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        databaseHelper = UserDatabaseHelper(this)
        setContent {
            SnackOrderingTheme {
                // A surface container using the 'background' color from
the theme
                Surface(
                    modifier = Modifier.fillMaxSize(),
                    color = MaterialTheme.colors.background
                ) {
                    LoginScreen(this, databaseHelper)
                }
            }
        }
    }
}
@Composable
fun LoginScreen(context: Context, databaseHelper: UserDatabaseHelper) {
```

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

        )

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

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

        Text(
            fontSize = 36.sp,
            fontWeight = FontWeight.ExtraBold,
            fontFamily = FontFamily.Cursive,
            color = Color.White,
            text = "Login"
        )
        Spacer(modifier = Modifier.height(10.dp))

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

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

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

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

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

                } else {
                    error = "Please fill all fields"
                }
            },
            modifier = Modifier.padding(top = 16.dp)
        ) {
            Text(text = "Login")
        }
        Row {
            TextButton(onClick = {context.startActivity(
                Intent(
                    context,
                    MainActivity::class.java
                )
            )}
            )
            { Text(color = Color.White,text = "Sign up") }
            TextButton(onClick = {
            })

            {
                Spacer(modifier = Modifier.width(60.dp))
                Text(color = Color.White,text = "Forget password?")
            }
        }
    }
}
private fun startMainPage(context: Context) {
    val intent = Intent(context, MainPage::class.java)
    ContextCompat.startActivity(context, intent, null)
}
```

MainPage:

```
package com.example.snackordering

import android.annotation.SuppressLint
import android.content.Context
import android.os.Bundle
import android.widget.Toast
import androidx.activity.ComponentActivity
import androidx.activity.compose.setContent
import androidx.annotation.DrawableRes
import androidx.annotation.StringRes
import androidx.compose.foundation.Image
import androidx.compose.foundation.background
import androidx.compose.foundation.layout.*
import androidx.compose.foundation.shape.CircleShape
import androidx.compose.foundation.shape.RoundedCornerShape
import androidx.compose.material.*
import androidx.compose.material.icons.Icons
import androidx.compose.material.icons.filled.*
import androidx.compose.runtime.Composable
import androidx.compose.ui.Alignment
import androidx.compose.ui.Modifier
import androidx.compose.ui.draw.clip
import androidx.compose.ui.graphics.Color
import androidx.compose.foundation.lazy.LazyColumn
import androidx.compose.foundation.lazy.items
import androidx.compose.material.Text
import androidx.compose.ui.unit.dp
import androidx.compose.ui.graphics.RectangleShape
import androidx.compose.ui.layout.ContentScale
import androidx.compose.ui.platform.LocalContext
import androidx.compose.ui.res.painterResource
import androidx.compose.ui.res.stringResource
import androidx.compose.ui.text.font.FontWeight
import androidx.compose.ui.unit.sp
import androidx.core.content.ContextCompat.startActivity
import com.example.snackordering.ui.theme.SnackOrderingTheme

import android.content.Intent as Intent1


class MainPage : ComponentActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContent {
            SnackOrderingTheme {
                // A surface container using the 'background' color from
the theme
                Surface(
                    modifier = Modifier.fillMaxSize(),
                    color = MaterialTheme.colors.background
                ) {
                    FinalView(this)
                    val context = LocalContext.current
                    //PopularFoodColumn(context)
                }
```

```
            }
        }
    }
}


@Composable
fun TopPart() {

    Row(
        modifier = Modifier
            .fillMaxWidth()
            .background(Color(0xffeceef0)), Arrangement.SpaceBetween
    ) {
        Icon(
            imageVector = Icons.Default.Add, contentDescription = "Menu
Icon",
            Modifier

                .clip(CircleShape)
                .size(40.dp),
            tint = Color.Black,
        )
        Column(horizontalAlignment = Alignment.CenterHorizontally) {
            Text(text = "Location", style =
MaterialTheme.typography.subtitle1, color = Color.Black)
            Row {
                Icon(
                    imageVector = Icons.Default.LocationOn,
                    contentDescription = "Location",
                    tint = Color.Red,
                )
                Text(text = "Accra" , color = Color.Black)
            }

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

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

@Composable
fun CardPart() {
    Card(modifier = Modifier.size(width = 310.dp, height = 150.dp),
RoundedCornerShape(20.dp)) {
        Row(modifier = Modifier.padding(10.dp), Arrangement.SpaceBetween) {
            Column(verticalArrangement = Arrangement.spacedBy(12.dp)) {
                Text(text = "Get Special Discounts")
                Text(text = "up to 85%", style =
MaterialTheme.typography.h5)
                Button(onClick = {}, colors =
ButtonDefaults.buttonColors(Color.White)) {
                    Text(text = "Claim voucher", color =
MaterialTheme.colors.surface)
                }
```

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


@Composable
fun PopularFood(
    @DrawableRes drawable: Int,
    @StringRes text1: Int,
    context: Context
) {
    Card(
        modifier = Modifier
            .padding(top=20.dp, bottom = 20.dp, start = 65.dp)
            .width(250.dp)

    ) {
        Column(
            verticalArrangement = Arrangement.Top,
            horizontalAlignment = Alignment.CenterHorizontally
        ) {
            Spacer(modifier = Modifier.padding(vertical = 5.dp))
            Row(
                modifier = Modifier
                    .fillMaxWidth(0.7f), Arrangement.End
            ) {
                Icon(
                    imageVector = Icons.Default.Star,
                    contentDescription = "Star Icon",
                    tint = Color.Yellow
                )
                Text(text = "4.3", fontWeight = FontWeight.Black)
            }
            Image(
                painter = painterResource(id = drawable),
                contentDescription = "Food Image",
                contentScale = ContentScale.Crop,
                modifier = Modifier
                    .size(100.dp)
                    .clip(CircleShape)
            )
            Text(text = stringResource(id = text1), fontWeight =
FontWeight.Bold)
            Row(modifier = Modifier.fillMaxWidth(0.7f),
Arrangement.SpaceBetween) {
                /*TODO Implement Prices for each card*/
                Text(
                    text = "$50",
                    style = MaterialTheme.typography.h6,
                    fontWeight = FontWeight.Bold,
                    fontSize = 18.sp
                )

                IconButton(onClick = {
```

```
                    //var no=FoodList.lastIndex;
                    //Toast.
                    val intent = Intent1(context,
TargetActivity::class.java)
                    context.startActivity(intent)

                }) {
                    Icon(
                        imageVector = Icons.Default.ShoppingCart,
                        contentDescription = "shopping cart",
                    )
                }
            }
        }
    }
}




private val FoodList = listOf(
    R.drawable.sandwish to R.string.sandwich,
    R.drawable.sandwish to R.string.burgers,
    R.drawable.pack to R.string.pack,
    R.drawable.pasta to R.string.pasta,
    R.drawable.tequila to R.string.tequila,
    R.drawable.wine to R.string.wine,
    R.drawable.salad to R.string.salad,
    R.drawable.pop to R.string.popcorn
).map { DrawableStringPair(it.first, it.second) }

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


@Composable
fun App(context: Context) {

    Column(
        modifier = Modifier
            .fillMaxSize()
            .background(Color(0xffeceef0))
            .padding(10.dp),
        verticalArrangement = Arrangement.Top,
        horizontalAlignment = Alignment.CenterHorizontally
    ) {
        Surface(modifier = Modifier, elevation = 5.dp) {
            TopPart()
        }
        Spacer(modifier = Modifier.padding(10.dp))
        CardPart()

        Spacer(modifier = Modifier.padding(10.dp))
        Row(modifier = Modifier.fillMaxWidth(), Arrangement.SpaceBetween) {
            Text(text = "Popular Food", style =
MaterialTheme.typography.h5, color = Color.Black)
            Text(text = "view all", style =
MaterialTheme.typography.subtitle1,  color = Color.Black)
        }
```

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



@Composable
fun PopularFoodColumn(context: Context) {

    LazyColumn(
        modifier = Modifier.fillMaxSize(),

        content = {
            items(FoodList) { item ->
                PopularFood(context = context,drawable = item.drawable,
text1 = item.text1)
                abstract class Context
            }
        },
        verticalArrangement = Arrangement.spacedBy(16.dp))
}


@SuppressLint("UnusedMaterialScaffoldPaddingParameter")
@Composable
fun FinalView(mainPage: MainPage) {
    SnackOrderingTheme {
        Scaffold() {
            val context = LocalContext.current
            App(context)
        }
    }
}
```

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 MainActivity : ComponentActivity() {
    private lateinit var databaseHelper: UserDatabaseHelper
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        databaseHelper = UserDatabaseHelper(this)
        setContent {
            SnackOrderingTheme {
                // A surface container using the 'background' color from
the theme
                Surface(
                    modifier = Modifier.fillMaxSize(),
                    color = MaterialTheme.colors.background
                ) {

                    RegistrationScreen(this,databaseHelper)
                }
            }
        }
    }
}


@Composable
fun RegistrationScreen(context: Context, databaseHelper:
UserDatabaseHelper) {

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

        )

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

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

        Text(
            fontSize = 36.sp,
            fontWeight = FontWeight.ExtraBold,
            fontFamily = FontFamily.Cursive,
            color = Color.White,
            text = "Register"
        )

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

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

        )

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

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


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

        Button(
            onClick = {
                if (username.isNotEmpty() && password.isNotEmpty() &&
email.isNotEmpty()) {
                    val user = User(
                        id = null,
                        firstName = username,
                        lastName = null,
                        email = email,
                        password = password
                    )
                    databaseHelper.insertUser(user)
                    error = "User registered successfully"
                    // Start LoginActivity using the current context
                    context.startActivity(
                        Intent(
                            context,
                            LoginActivity::class.java
                        )
                    )

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

```
            },
            modifier = Modifier.padding(top = 16.dp)
        ) {
            Text(text = "Register")
        }
        Spacer(modifier = Modifier.width(10.dp))
        Spacer(modifier = Modifier.height(10.dp))

        Row() {
            Text(
                modifier = Modifier.padding(top = 14.dp), text = "Have an
account?"
            )
            TextButton(onClick = {
                context.startActivity(
                    Intent(
                        context,
                        LoginActivity::class.java
                    )
                )
            })

            {
                Spacer(modifier = Modifier.width(10.dp))
                Text(text = "Log in")
            }
        }
    }
}
private fun startLoginActivity(context: Context) {
    val intent = Intent(context, LoginActivity::class.java)
    ContextCompat.startActivity(context, intent, null)
}
```