



Smart
Internz

android 

ANDROID STUDIO – EXPERIENCE BASED PROJECT LEARNING

Snack Squad: A Customizable Snack Ordering and Delivery App

Submitted by

NANTHAKUMAR.V	-	711022104033
PANDURANGA VITTALA.N	-	711022104038
PALRAJ.T	-	711022104037
ROHITH.S	-	711022104041

BACHELOR OF COMPUTER SCIENCE AND ENGINEERING

IN

FIFTH SEMESTER

COMPUTER SCIENCE AND ENGINEERING

INFO INSTITUTE OF ENGINEERING, COIMBATORE – 641107

NOVEMBER/DECEMBER - 2024

BONAFIDE CERTIFICATE

Certified that this project “Snack Squad: A Customizable Snack Ordering and Delivery App” is the Bonafide work of NANTHAKUMAR.V(711022104033), PANDURANGAVITTALA.N(711022104038), PALRAJ.T(711022104037), ROHITH.S(711022104041) who carried out the project work under any supervision.

SIGNATURE

STAFF COORDINATOR

Mrs. A. SARANYA M.E.,

ASSISTANT PROFESSOR

DEPT. COMPUTER SCIENCE AND ENGINEERING, INFO INSTITUTE OF ENGINEERING,
KOVILPALAYAM COIMBATORE - 641107

SIGNATURE

HEAD OF THE DEPARTMENT

Dr. G. SELVAVINAYAGAM Ph.D.,

HEAD OF THE DEPARTMENT

DEPT. COMPUTER SCIENCE AND ENGINEERING, INFO INSTITUTE OF ENGINEERING,
KOVILPALAYAM COIMBATORE - 641107

INTERNAL EXAMINER

EXTERNAL EXAMINER

ACKNOWLEDGEMENT

We sincerely thank to Tamil Nadu Skill Development Corporation (TNSDC), Naan Mudhalvan" Platform and ANDROID STUDIO – EXPERIENCE BASED PROJECT LEARNING(EBPL) for encouragement towards our project work for providing necessary skill training.

We sincerely thank our Principal Dr.N.KOTTISWARAN,M.E., Ph.D., and Head of the Department Dr. G. SELVAVINAYAGAM,M.E., Ph.D., and also Staff Coordinator Mrs. A. SARANYA M.Efor herencouragement towards our project works.

We also thank our project guide and our parents for the complete and whole hearted support, motivation guidance and help in making our project activities.

A Customizable Snack Ordering and Delivery App

ABSTRACT:

This project describes the creation of a simple yet effective Android-based meal delivery application aimed at streamlining the food ordering and administration process. The application has two primary interfaces:

The admin interface, accessible solely via a unique username and password, allows administrators to manage the food menu, monitor orders, and track delivery statuses. This interface ensures strong security and effective control over the application's operations.

User Page: This page, designed for an easy user experience, allows customers to browse the available menu, choose items, define quantities, and provide delivery information such as address. The clear style allows users to place orders with little effort, increasing convenience.

The application's goal is to provide a seamless food ordering experience by bridging communication between customers and administrators. Using the Android platform, this project aims to demonstrate the integration of secure authentication techniques, responsive user interfaces, and critical backend capabilities to suit the needs of a simple food delivery business.

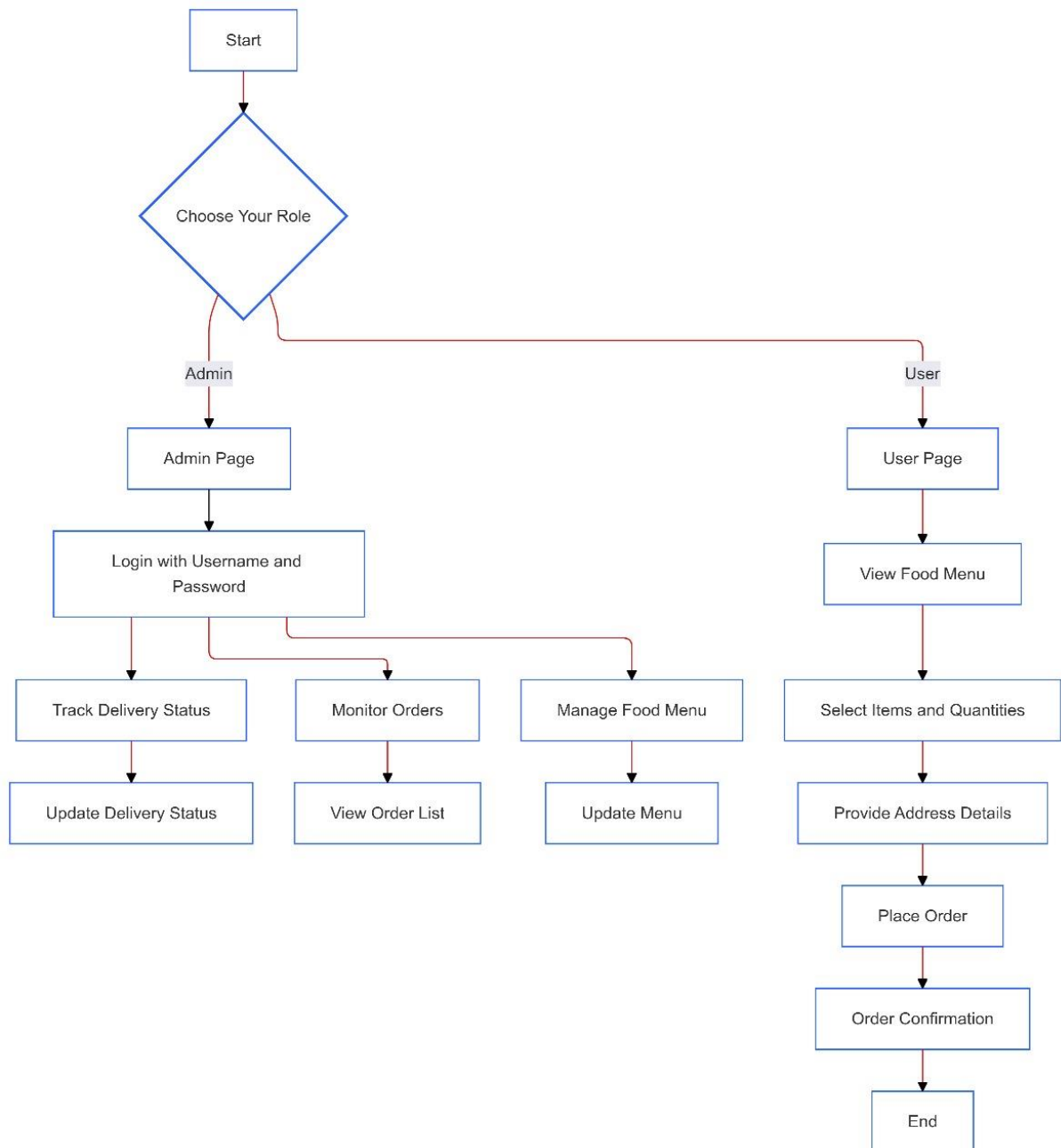
INTRODUCTION

In today's fast-paced world, convenience and customization are key to meeting consumer demands. Snack preferences vary widely, with individuals seeking not just quality and variety but also tailored options that suit their tastes, dietary restrictions, and lifestyle. Despite the abundance of snack delivery platforms, few cater to the specific needs of users looking for a seamless, personalized experience.

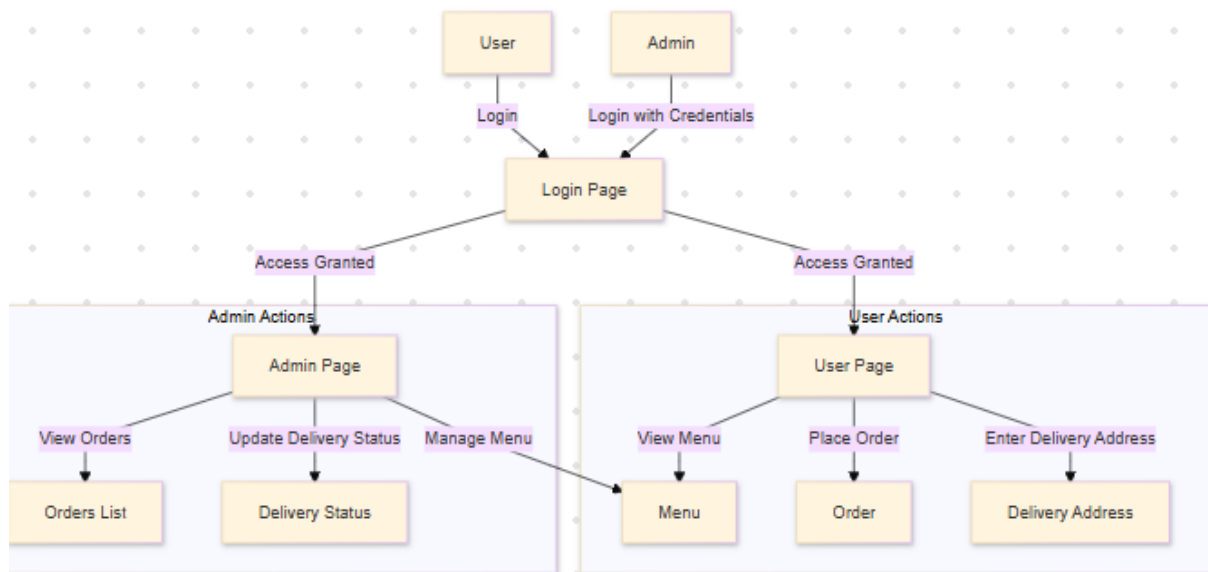
Enter *CrunchTime*, a next-generation snack ordering and delivery platform designed to bridge this gap. By blending cutting-edge technology with a user-centric approach, CrunchTime offers a customizable interface where users can mix and match their favorite snacks, set dietary preferences, and even explore curated recommendations. With real-time inventory updates and a robust delivery network, CrunchTime ensures quick, reliable service while maintaining the highest standards of freshness and quality.

Whether it's a spontaneous craving, a planned treat, or a gift for a loved one, CrunchTime is here to redefine how snacks are ordered, personalized, and delivered. This introduction explores the motivation behind the platform, its key features, and the innovative technology powering it.

DATA FLOW DIAGRAM



USER CASE DIAGRAM



SOFTWARE REQUIREMENTS

To develop CrunchTime, a customizable snack ordering and delivery app, the following software requirements are necessary to ensure a smooth development and deployment process:

Development Environment

1. Android Studio

- Latest stable version of Android Studio as the primary Integrated Development Environment (IDE) for app development.
- Features such as code editing, debugging, and performance tools.

2. Kotlin Programming Language

- Kotlin will be used as the primary language for Android app development due to its modern syntax, safety features, and full compatibility with Java.

3. Gradle Build System

- Gradle Script for build automation and dependency management.
- Integration with Android Studio for efficient handling of builds, plugins, and project configurations.

4. Virtual Machine (Emulators)

- Android Virtual Device (AVD) Manager within Android Studio for testing the app on various device configurations and screen sizes.

- Requirements:

- RAM: Minimum 4 GB for smooth emulator performance.
- CPU: x86/x64 architecture recommended.
- GPU support for enhanced graphical testing.

Software Dependencies and Libraries

1. Android SDK and Tools

- Android SDK Platform and build tools for compiling and debugging the app.
- Support for minimum and target API levels (e.g., API 23+ for broader compatibility).

Operating System Requirements

1. Development Machine

- Windows 10/11, macOS 11+, or Linux (Ubuntu recommended).
- Minimum Specs:
 - CPU: Multi-core processor (Intel i5/i7 or AMD Ryzen).
 - RAM: 8 GB (16 GB recommended).
 - Storage: SSD with at least 20 GB free space.

Testing and Debugging Tools

1. JUnit

- For unit testing Kotlin classes and components.

Version Control and Collaboration Tools

1. Git

- Version control system for tracking code changes and managing branches.

2. GitHub/GitLab/Bitbucket

- Repository hosting and team collaboration platform.

PROGRAM CODE

Main activity

```
package com.example.crunchtime

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.unit.sp
import androidx.core.content.ContextCompat.startActivity
import com.example.CrunchTime.ui.theme.SnackOrderingTheme

import android.content.Intent as Intent1

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

@Composable
fun TopPart() {

    Row(
        modifier = Modifier

```

```

fun TopPart() {
    modifier = Modifier
        .fillMaxWidth()
        .background(Color( 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(

```

```

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( fraction: 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)
    }
}

```



```

private val FoodList = listOf(
    R.drawable.sandwich 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(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()
    }
}

```



```

@Composable
fun PopularFoodColumn(context: Context) {

    LazyColumn(
        modifier = Modifier.fillMaxSize(),

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

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

```

Admin Activity

```
package com.example.crunchtime

import android.icu.text.SimpleDateFormat
import android.os.Bundle
import android.util.Log
import androidx.activity.ComponentActivity
import androidx.activity.compose.setContent
import androidx.compose.foundation.Image
import androidx.compose.foundation.layout.*
import androidx.compose.foundation.lazy.LazyColumn
import androidx.compose.foundation.lazy.LazyRow
import androidx.compose.foundation.lazy.items
import androidx.compose.material.MaterialTheme
import androidx.compose.material.Surface
import androidx.compose.material.Text
import androidx.compose.runtime.Composable
import androidx.compose.ui.Modifier
import androidx.compose.ui.graphics.Color
import androidx.compose.ui.layout.ContentScale
import androidx.compose.ui.res.painterResource
import androidx.compose.ui.unit.dp
import androidx.compose.ui.unit.sp
import com.example.CrunchTime.ui.theme.SnackOrderingTheme
import java.util.*

class AdminActivity : ComponentActivity() {
    private lateinit var orderDatabaseHelper: OrderDatabaseHelper
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        orderDatabaseHelper = OrderDatabaseHelper(context: this)
        setContent {
            CrunchTimeTheme {
```

```

class AdminActivity : ComponentActivity() {
    private lateinit var orderDatabaseHelper: OrderDatabaseHelper
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        orderDatabaseHelper = OrderDatabaseHelper(context, this)
        setContent {
            CrunchTimeTheme {
                // A surface container using the 'background' color from the theme
                Surface(
                    modifier = Modifier.fillMaxSize(),
                    color = MaterialTheme.colors.background
                ) {
                    val data = orderDatabaseHelper.getAllOrders();
                    Log.d(tag = "swathi", data.toString())
                    val order = orderDatabaseHelper.getAllOrders()
                    ListListScopeSample(order)
                }
            }
        }
    }
}

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

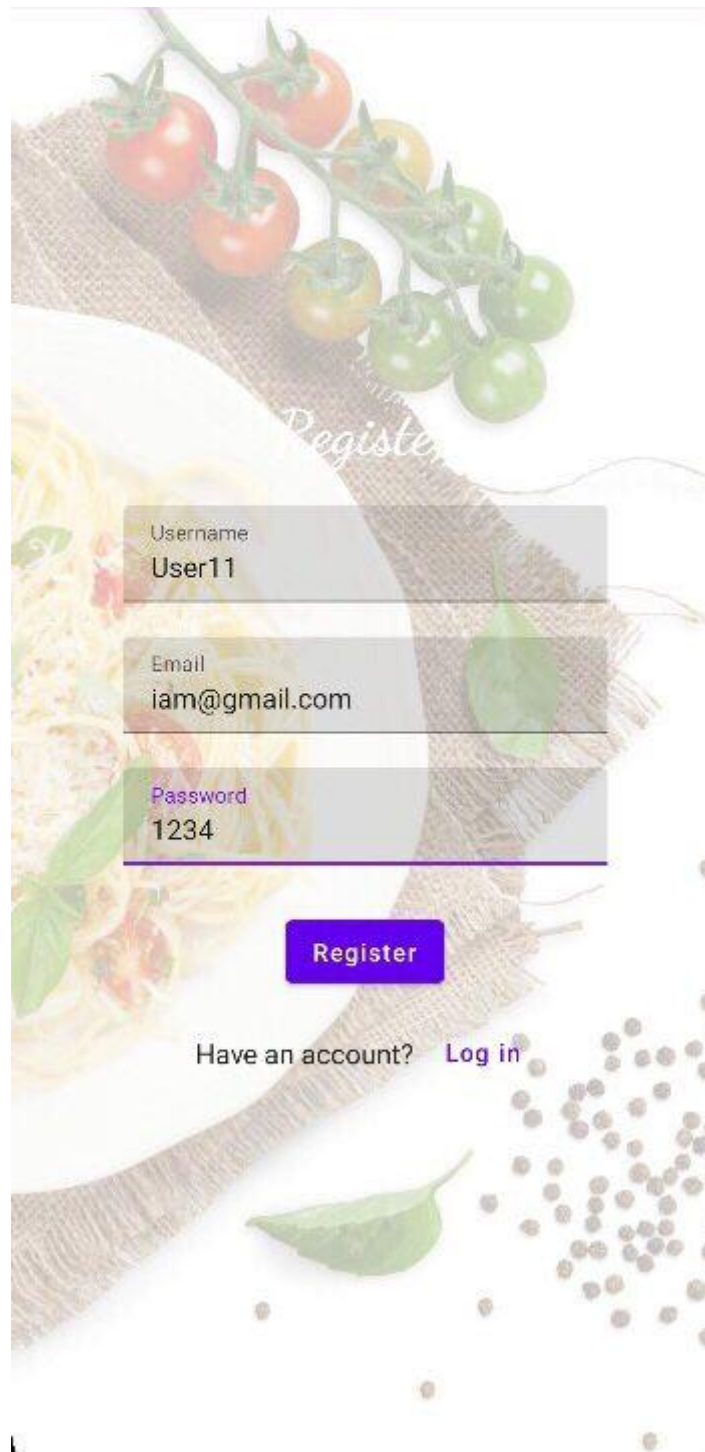
```

```

fun ListListScopeSample(order: List<Order>) {
    Image(
        painterResource(id = R.drawable.order), contentDescription = "",
        alpha = 0.5f,
        contentScale = ContentScale.FillHeight
    )
    Text(text = "Order Tracking", modifier = Modifier.padding(top = 24.dp, start = 106.dp, bottom = 24.dp), color = Color.White, fontSize = 30.sp)
    Spacer(modifier = Modifier.height(30.dp))
    LazyRow(
        modifier = Modifier
            .fillMaxSize()
            .padding(top = 80.dp),
        horizontalArrangement = Arrangement.SpaceBetween
    ) {
        item {
            LazyColumn {
                items(order) { order ->
                    Column(modifier = Modifier.padding(top = 16.dp, start = 48.dp, bottom = 20.dp)) {
                        Text("Quantity: ${order.quantity}")
                        Text("Address: ${order.address}")
                    }
                }
            }
        }
    }
}

```

OUTPUT

The background of the registration page features a high-quality photograph of a dish of spaghetti topped with a tomato and basil sauce. A piece of light brown burlap fabric is draped over the top left of the plate. A small cluster of cherry tomatoes, some red and some green, is attached to a vine and rests on the burlap. Several fresh green basil leaves are scattered around the plate. In the bottom right corner, there is a small pile of dark, round peppercorns. The overall aesthetic is clean, fresh, and appetizing.

Register

Username
User11

Email
iam@gmail.com

Password
1234

Register

Have an account? [Log in](#)

Fig: Registration Page

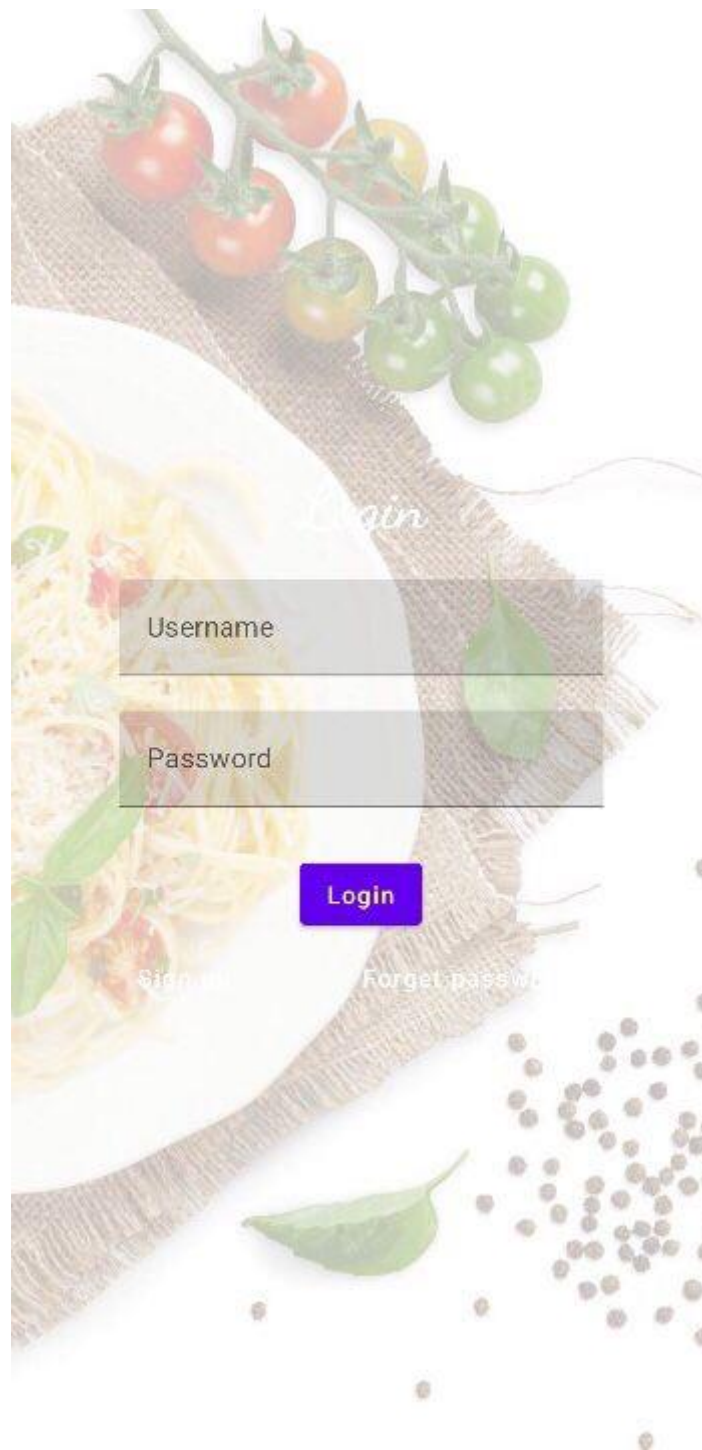


Fig: Login Page

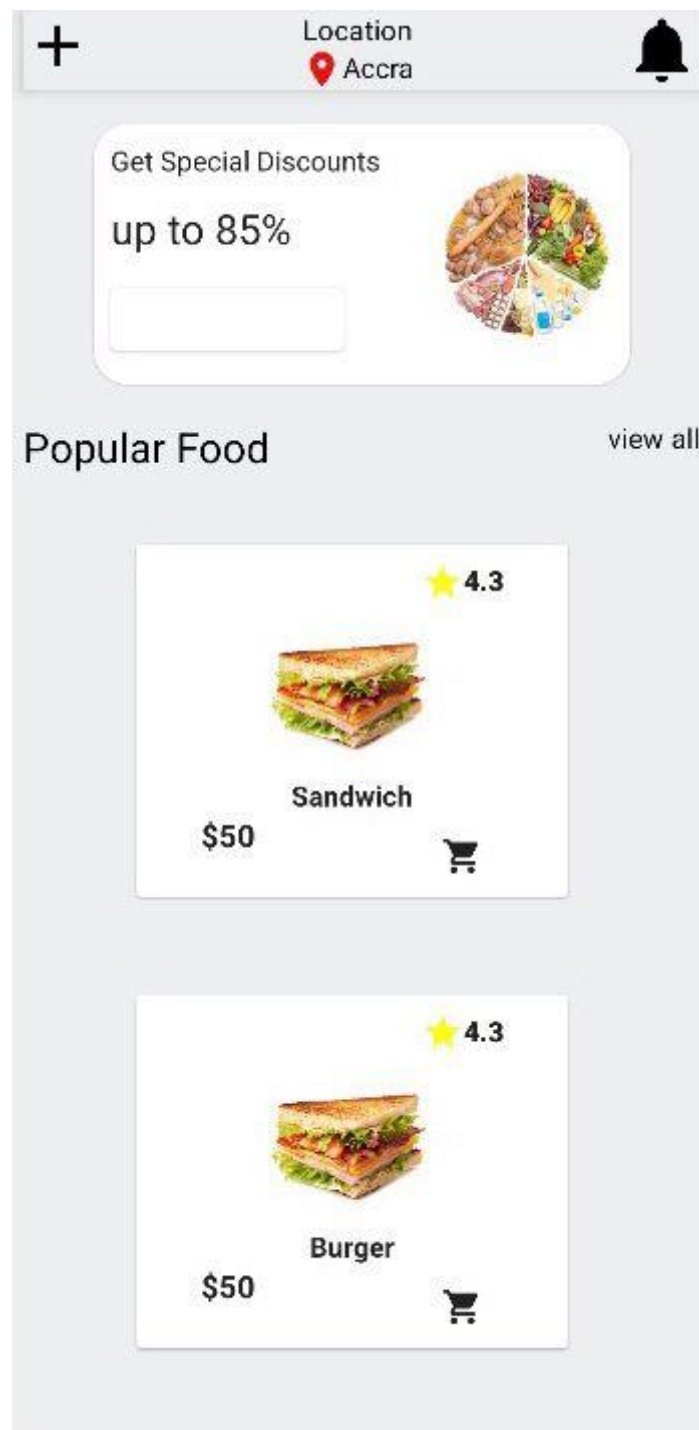


Fig: Home Page



Fig: Cart Page

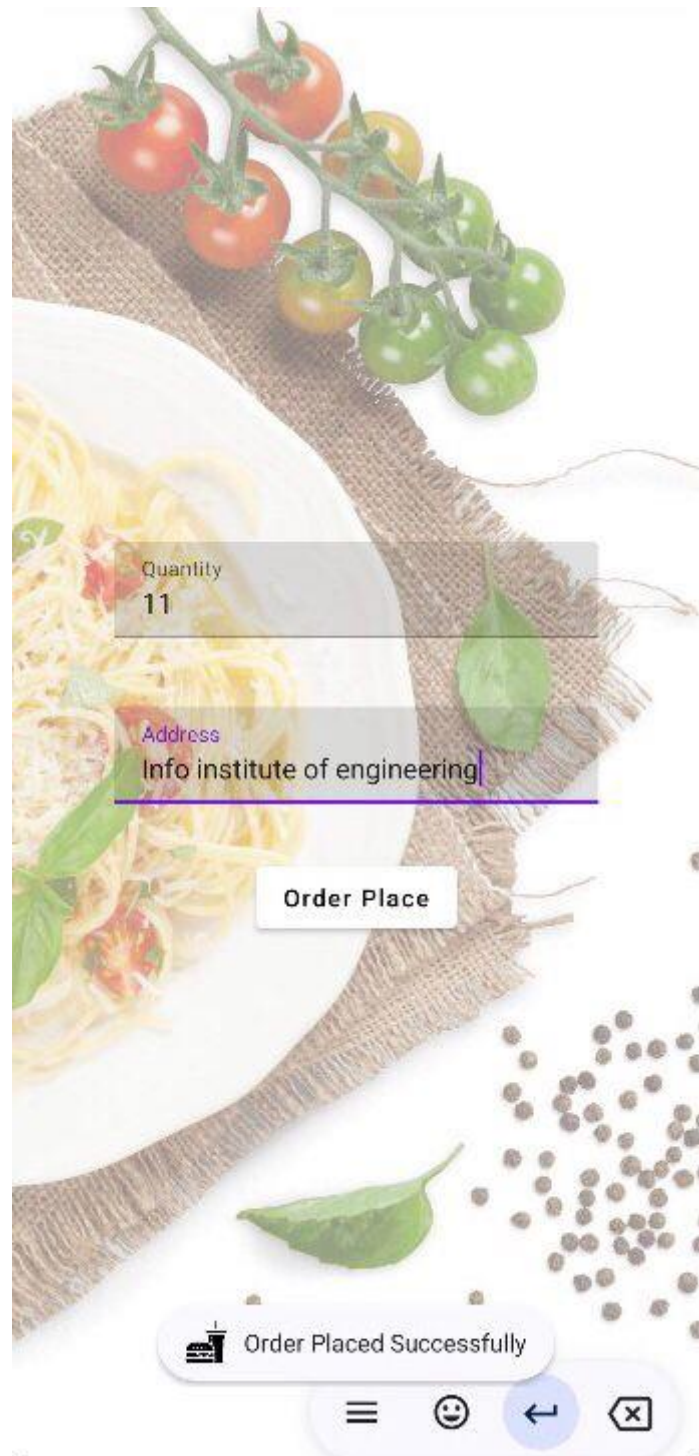


Fig: Order Placed Successful Message

FUTURE ENHANCEMENT

1. AI-Powered Recommendations:

Integrate AI to suggest personalized snack combinations based on user preferences and purchase history.

2. Subscription Plans:

Offer subscription services for regular snack deliveries at discounted rates.

3. AR Visualization:

Use augmented reality to let users visualize snacks in 3D before ordering.

4. Real-Time Order Tracking:

Enhance delivery experience with live tracking and estimated time of arrival updates.

5. Gamification Features:

Introduce loyalty points, badges, and challenges to boost user engagement.

6. Multi-Language Support:

Expand accessibility by supporting multiple regional and global languages.

7. Eco-Friendly Packaging Options:

Provide users the option to choose sustainable and reusable packaging.

8. Integration with Wearables:

Enable snack ordering and delivery notifications via smartwatches and fitness trackers.

CONCLUSION

CrunchTime revolutionizes snack ordering with its user-friendly interface, robust customization options, and seamless delivery experience. By leveraging modern technologies like Kotlin, Android Studio, and Firebase, it offers a reliable, efficient, and scalable solution for meeting diverse snacking needs. The app's ability to adapt to user preferences, provide real-time updates, and streamline the ordering process makes it a standout in the competitive food delivery market. Through continuous innovation and integration of user feedback, CrunchTime has the potential to become a go-to platform for snack lovers. With a strong foundation in place, the project sets the stage for future enhancements and market expansion.