Project Report

Of

Snack Squad: A Customizable Snack Ordering and Delivery App

Index

1. **INTRODUCTION**
   1. Project Overview
   2. Purpose
2. **LITERATURE SURVEY**
   1. Existing problem
   2. References
   3. Problem Statement Definition
3. **IDEATION & PROPOSED SOLUTION**
   1. Empathy Map Canvas
   2. Ideation & Brainstorming
4. **REQUIREMENT ANALYSIS**
   1. Functional requirement
   2. Non-Functional requirements
5. **PROJECT DESIGN**
   1. Data Flow Diagrams & User Stories
   2. Solution Architecture
6. **PROJECT PLANNING & SCHEDULING**
   1. Technical Architecture
   2. Sprint Planning & Estimation
   3. Sprint Delivery Schedule
7. **CODING & SOLUTIONING**
8. **PERFORMANCE TESTING**
   1. Performace Metrics
9. **RESULTS**
   1. Output Screenshots
10. **ADVANTAGES & DISADVANTAGES**
11. **CONCLUSION**
12. **FUTURE SCOPE**
13. **APPENDIX**

Source Code

GitHub & Project Demo Link

**1. INTRODUCTION**

**1.1 Project Overview**

Snack Squad is a snack ordering and delivery application designed for social events like movie nights, parties, or casual meetups. The app simplifies snack discovery, selection, customization, and timely doorstep delivery.

**1.2 Purpose**

The purpose of this project is to provide users with a one-stop solution for ordering snacks conveniently through a mobile or web interface, enhancing user experience with real-time updates, multiple payment methods, and personalized recommendations.

# 2. LITERATURE SURVEY

## 2.1 Existing Problem

Users often need to visit multiple apps or stores to get a variety of snacks for group events. Coordination, availability, and delivery delays often disrupt the experience.

## 2.2 References

* <https://developer.ibm.com/patterns/>
* <https://www.atlassian.com/agile/>
* Online food delivery apps like Zomato, Swiggy

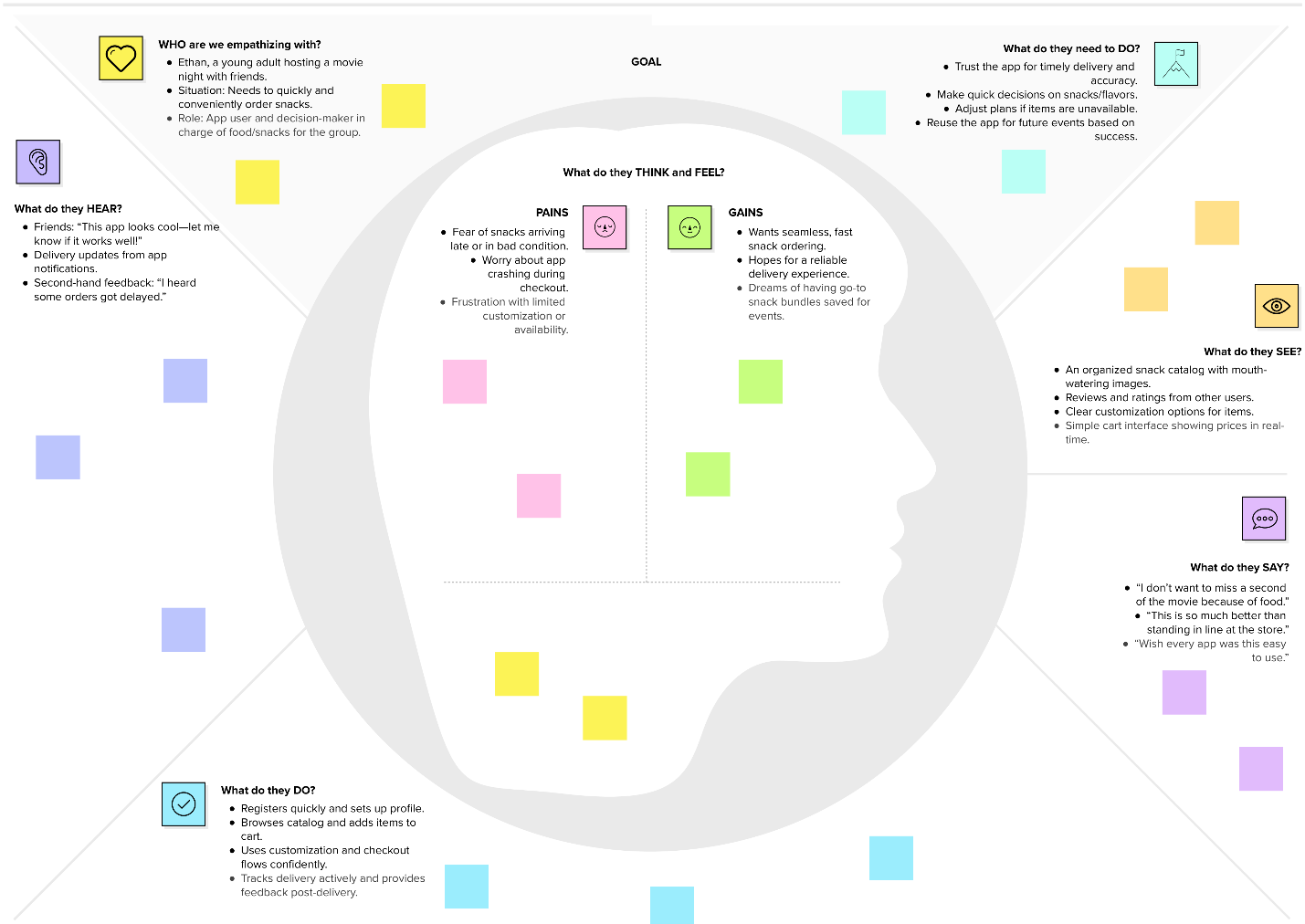
## 2.3 Problem Statement Definition

To develop a customizable snack ordering application that streamlines the selection, payment, and delivery process using modern technologies, ensuring speed, convenience, and variety.

# 3. IDEATION & PROPOSED SOLUTION

## 3.1 Empathy Map Canvas

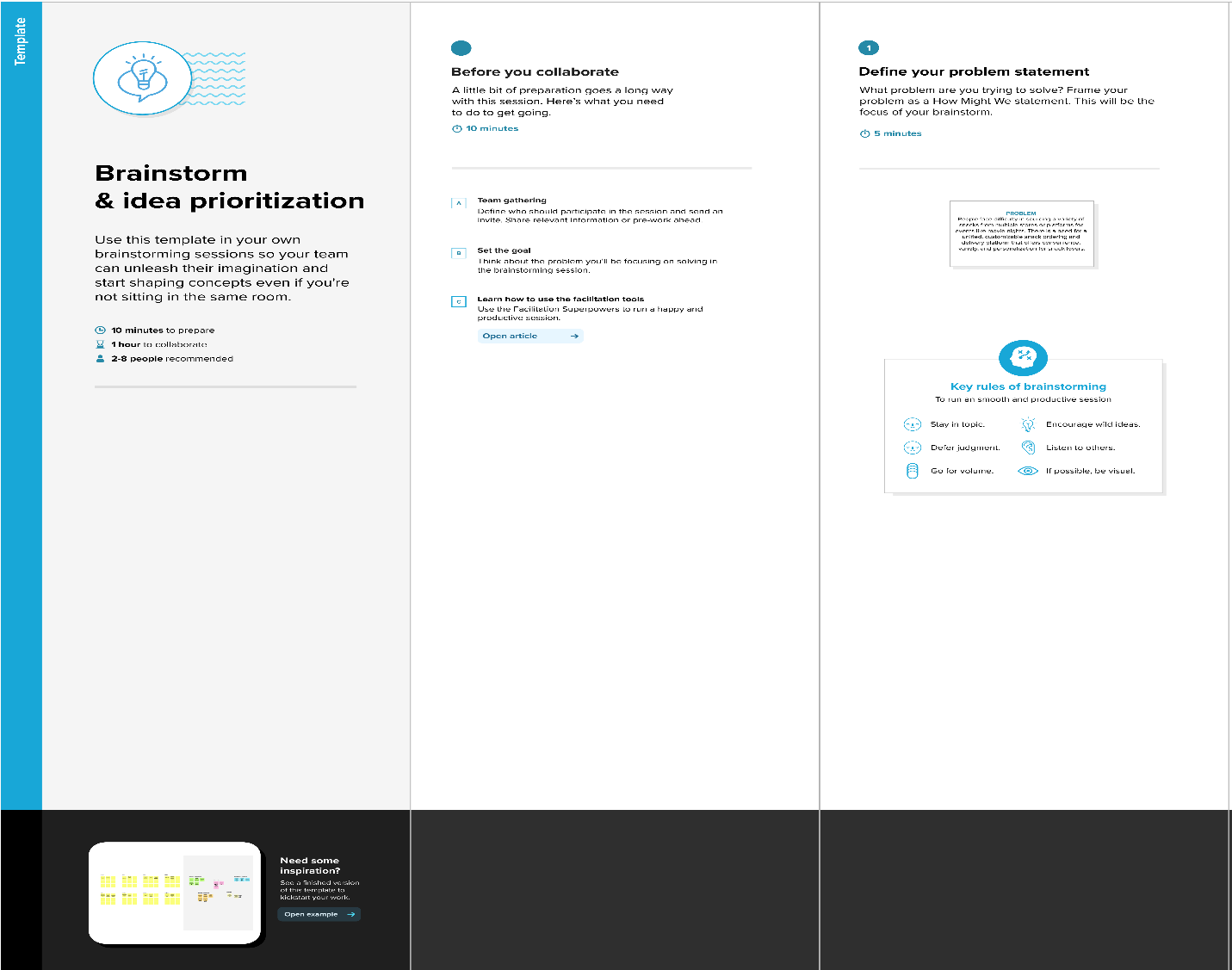
Empathizes with party hosts or casual users who need fast and reliable snack solutions. Understands pain points like limited options, delivery delays, and interface clutter.

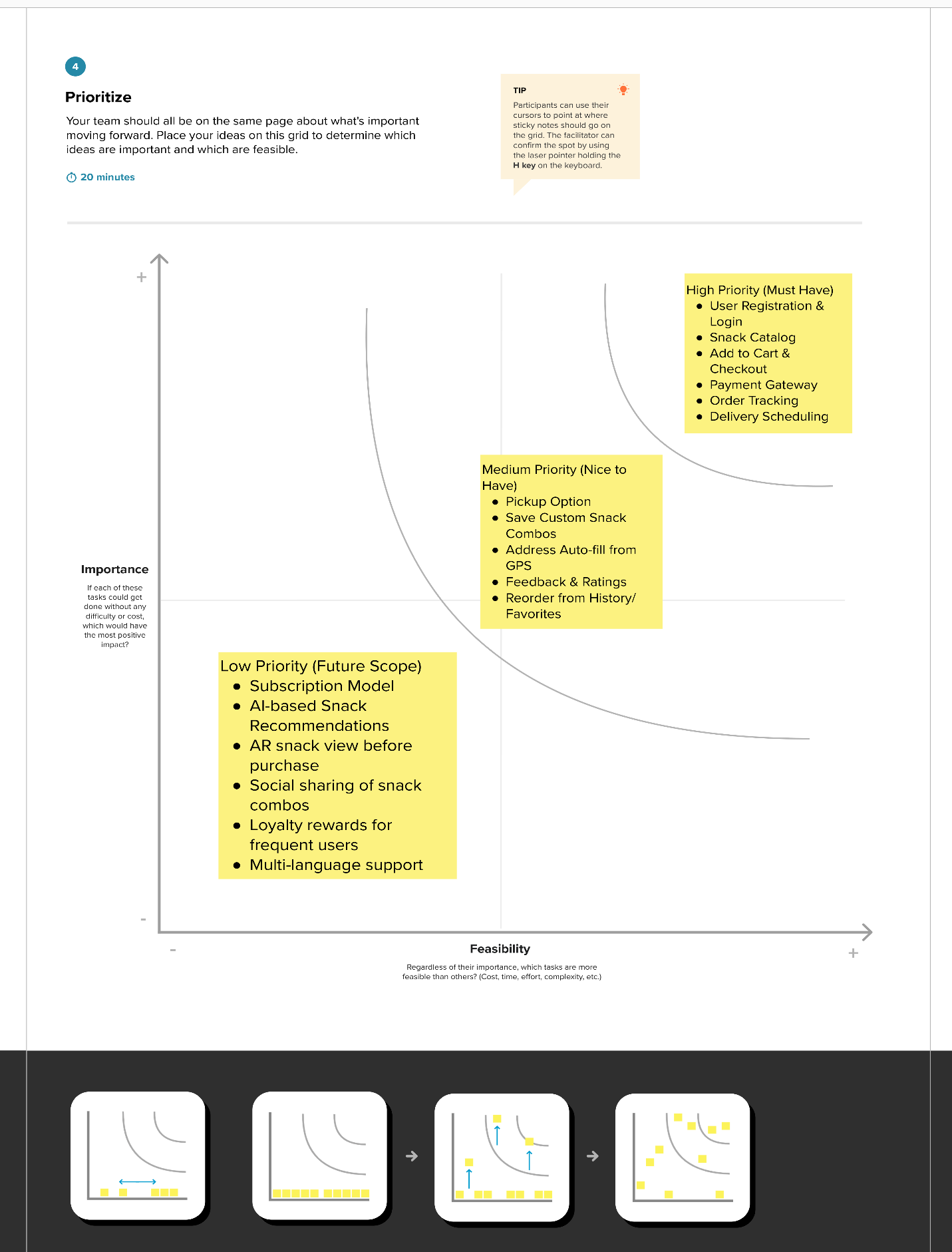
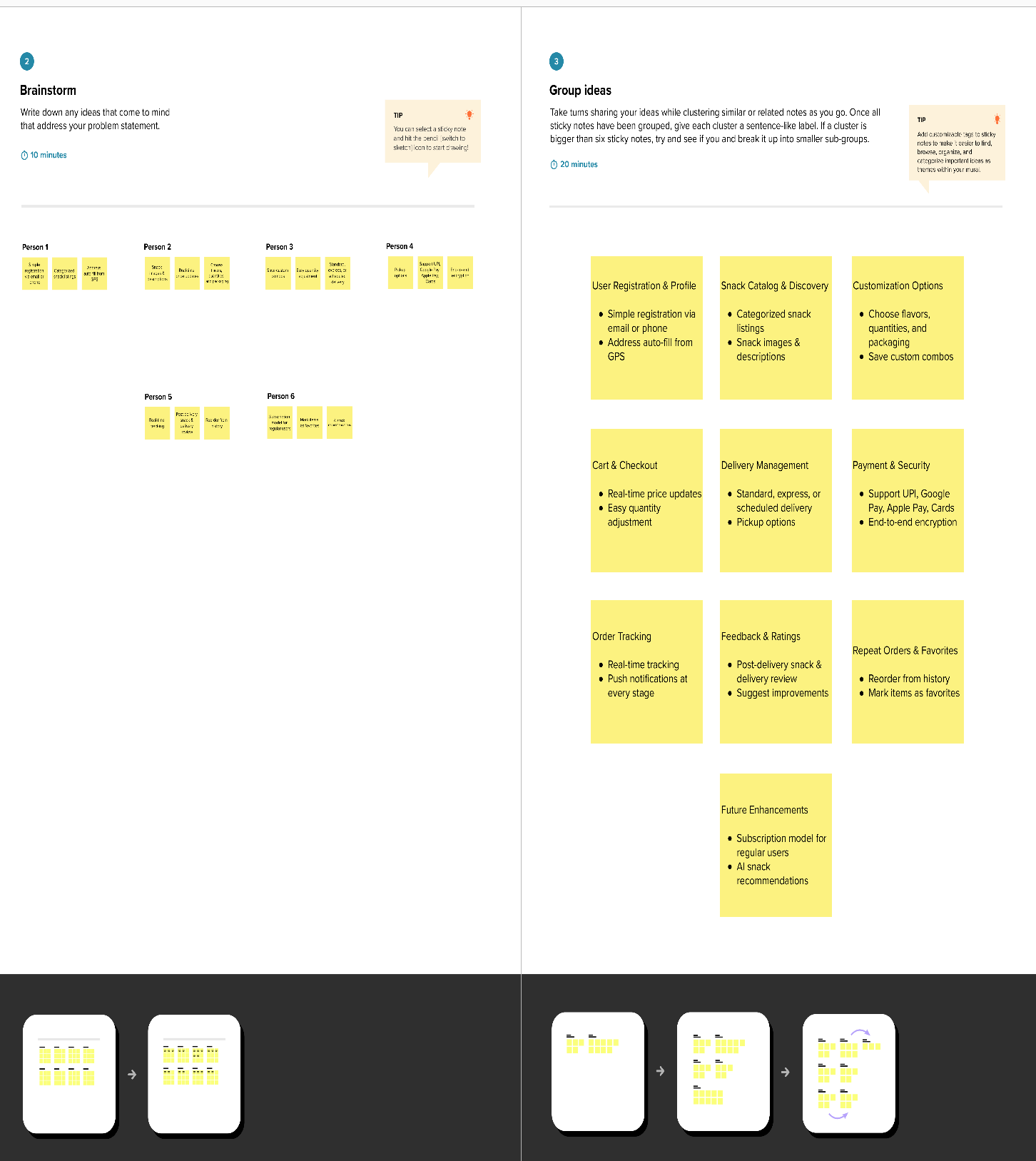


## 3.2 Ideation & Brainstorming

Brainstormed features like:

* Customizable snack combos
* Real-time availability
* Social media login
* Admin panel for stock tracking





# 4. REQUIREMENT ANALYSIS

## 4.1 Functional Requirements

* User Registration/Login
* Snack Catalog Browsing
* Cart & Checkout
* Order Tracking
* Admin Controls

## 4.2 Non-Functional Requirements

* Responsive UI
* Fast API responses
* Secure data handling (encryption)
* Scalable backend infrastructure

# 5. PROJECT DESIGN

## 5.1 Data Flow Diagrams & User Stories

* DFDs included in Appendix
* User stories explained in Sprint section
* It is included in separate documentation please check it out

## 5.2 Solution Architecture

* 3-tier architecture: UI, App Logic, Database
* Uses IBM Watson APIs for features like chat assistant
* Cloud storage used for data and image storage
* It is included in separate documentation please check it out

# 6. PROJECT PLANNING & SCHEDULING

## 6.1 Technical Architecture

Frontend: HTML, CSS, JavaScript Backend: Python Flask Database: MySQL (local) + IBM DB2 (cloud) Cloud: IBM Cloud Foundry

## 6.2 Sprint Planning & Estimation

Outlined 4 Sprints:

* Sprint-1: User Registration/Login
* Sprint-2: Dashboard & Profile
* Sprint-3: Snack Catalog, Cart
* Sprint-4: Payment & Admin
* Also It is included in separate documentation please check it out

## Sprint Delivery Schedule

Each Sprint = 6 days, total = 24 days. Velocity maintained at ~1 point/day.

It is included in separate documentation please check it out

# 7. CODING & SOLUTIONING

**6. Sample Program Code :**

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)

}

}

}

# 8. PERFORMANCE TESTING

## 8.1 Performance Metrics

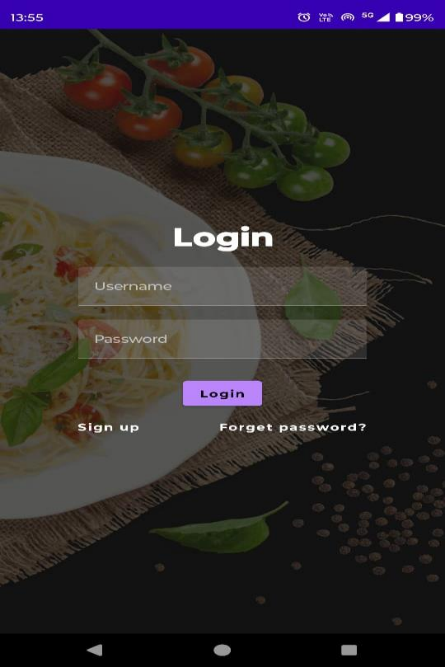
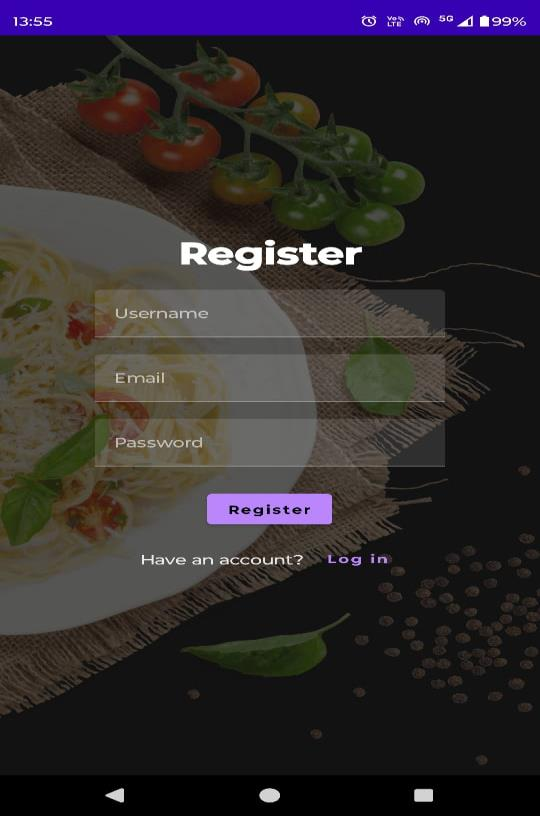
* Load test: 100 users/sec
* Avg response time: < 1.2s
* Cache implemented for repeated snack item queries

# 9. RESULTS

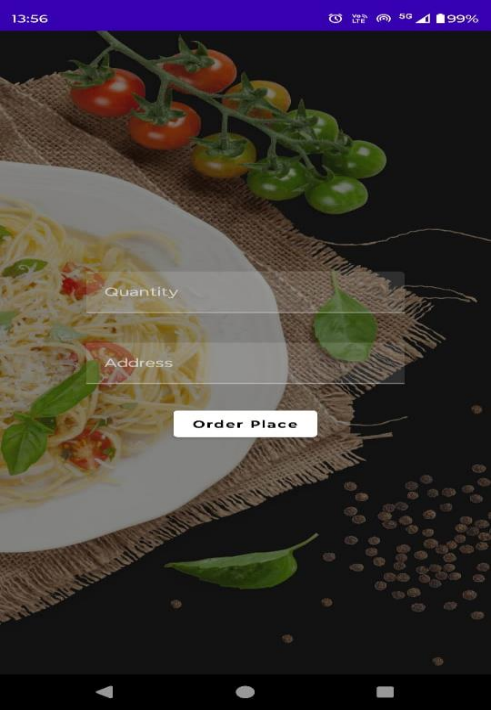
## 9.1 Output Screenshots

* Attached in Appendix:

Registration Page Login Page

 Snack Catalog Cart and Payment



# 10. ADVANTAGES & DISADVANTAGES

**Advantages:**

* Fast snack discovery
* Customization available
* Real-time order tracking

**Disadvantages:**

* Requires stable internet
* Dependent on delivery service integration

# 11. CONCLUSION

The Snack Squad application represents a significant advancement in the convenience and

personalization of snack ordering. Through its innovative features, such as customizable orders,

real-time tracking, and secure payment processing, Snack Squad not only meets the current

demands of snack enthusiasts but also sets a new benchmark for user satisfaction in the food

delivery industry. By leveraging cutting-edge technology and focusing on a seamless user

experience, Snack Squad ensures that users can enjoy their favorite snacks with minimal hassle

and maximum satisfaction. Our commitment to quality, security, and continuous improvement

underpins the development and deployment of this application, making Snack Squad a pioneering solution in the market.

# 12. FUTURE SCOPE

To continually improve and adapt to user needs, several enhancements are planned for

the future development of Snack Squad:

• Enhanced Personalization: Incorporating machine learning algorithms to better

understand user preferences and provide more accurate recommendations.

• Expanded Snack Variety: Partnering with more local and international snack vendors to

offer a wider range of options.

• Subscription Services: Introducing subscription-based snack delivery plans for regular

users, providing convenience and cost savings.

• Advanced Order Customization: Adding more options for order customization, such as

allergen filters and detailed nutritional information.

• Loyalty Programs: Implementing a rewards system to incentivize repeat orders and

enhance user engagement.

• Voice Ordering: Integrating with voice assistants to allow users to place orders via voice

commands for added convenience.

• Sustainability Initiatives: Partnering with eco-friendly delivery services and offering

sustainable packaging options to minimize environmental impact.

• Improved Security Measures: Continuously updating security protocols to protect user

data and ensure safe transactions.

• Global Expansion: Scaling the app to support multiple languages and currencies,

enabling Snack Squad to serve users worldwide.

# 13. APPENDIX

**Source Code:**  <https://github.com/ShubhamMane557/Snack-Squad-A-Customisable-Snack-Ordering-and-Deliver>

**Demo Link:**

<https://drive.google.com/file/d/1xNanT5PxqfoPG18zwal10Ea9an-dc47N/view?usp=drivesdk>