Contents

Chapter 1: Product Description

Introduction

Overview

Our project team developed and conducted a survey among our classmates and their family members of each participant to reach the conclusion of which app has the best potential on the Canadian market and what services our local community needs that could be satisfied by features of our application. We concluded that a Recipe Application would be our perfect choice to practice React native coding, meet the expectations of our code expert and teacher Mr. Xing Liu and potential customers. For our project we chose to code a mobile recipes app for users to be able to view, create and share their favourites with other users. Our application will allow users to create a library of all their recipes in the convenience of their phones.

Goals and Objectives

- Authentication via email and password
- Password reset
- Create, view, edit and delete own recipes
- View all recipes
- Search among recipes
- Like recipes of other users
- Sort recipes of other users
- Sort recipes by likes
- View favourite recipes
- View followers
- Help share different ethnic backgrounds with the art of food

Scope

The scope of our project is to give users the ability to redefine how they create store and share recipes with friends and family.

General Design Constraints

Application Environment

- The front-end of our application is centred around React native as an imminent software environment constraint of 2204
- Back-end of our application, we decided to go with Rails API
- We will be using Visual Studio Code as it is lightweight and it’s our preferred coding app.
- Main constraints for the user will be Memory, battery life and network bandwidth while using the application.

User Characteristics

- Responsive UI
- Beautiful UI
- Familiarity between screens to avoid confusion for the user
- Ease of use
- Security
- Offline viewing access of recipes

Non-Functional Requirements

Operation Requirements

Supported operating systems for end users are Android and Apple's IOS.

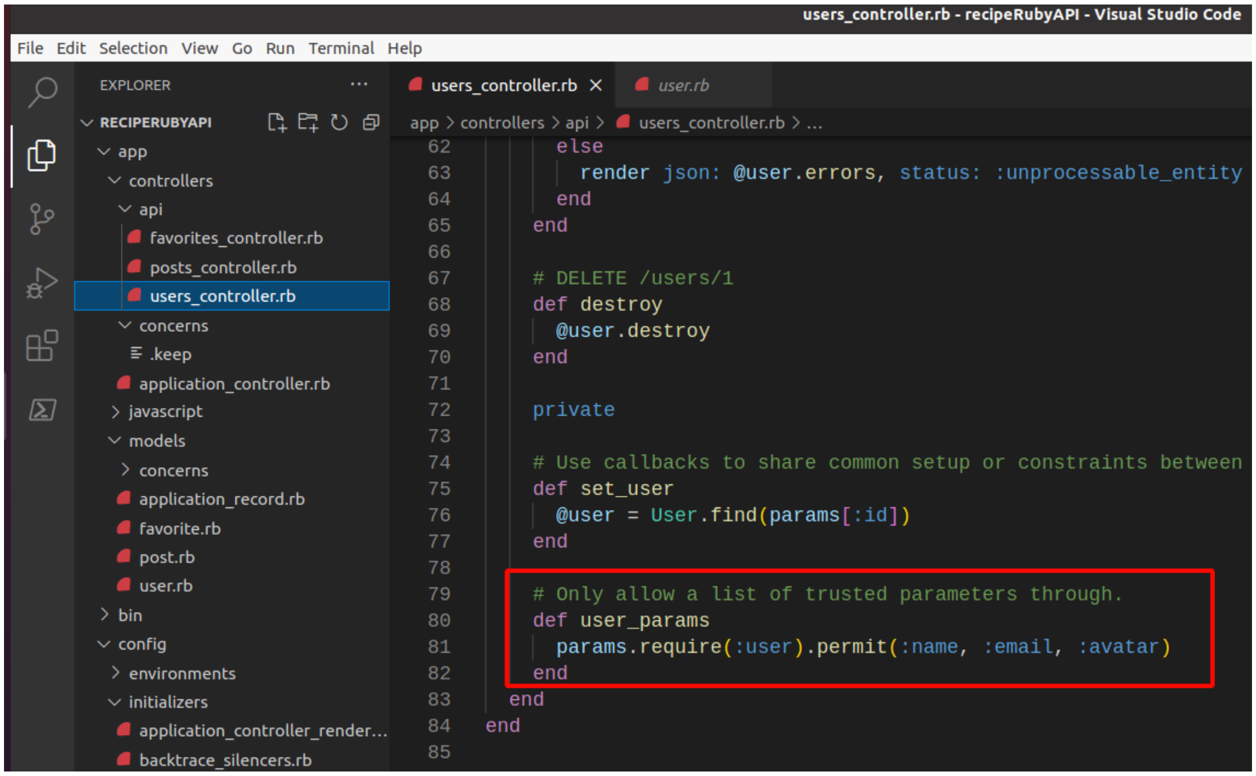
Hosting for back-end is Heroku free tier 3600 SQL CRUD operations per hour.

Supported user's devices are limited to phones that are no older than 5 years.

Validation Requirements

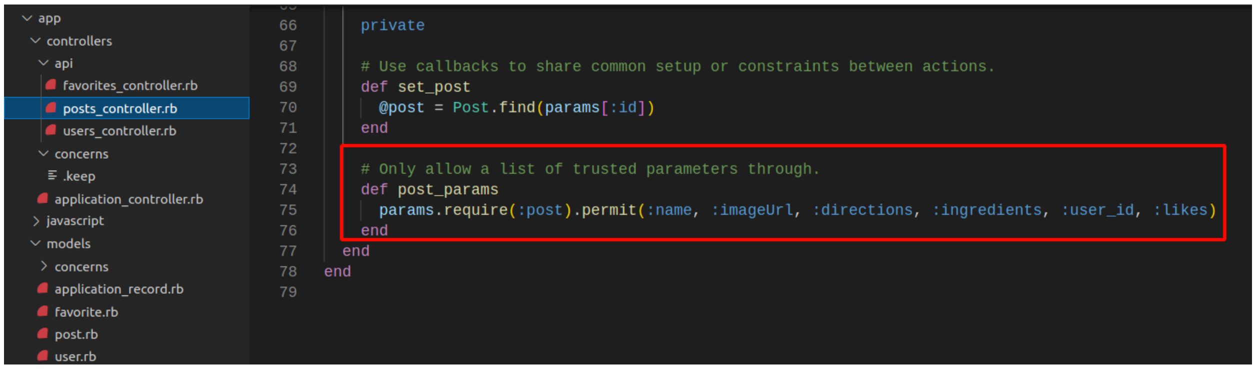
- Photo of the recipe should not be empty
- Description of the recipe should not be empty
- Ingredients of the recipe should not be empty
- Only fields described by table schemas are allowed to be posted in order to prevent SQL injection

User Entity:



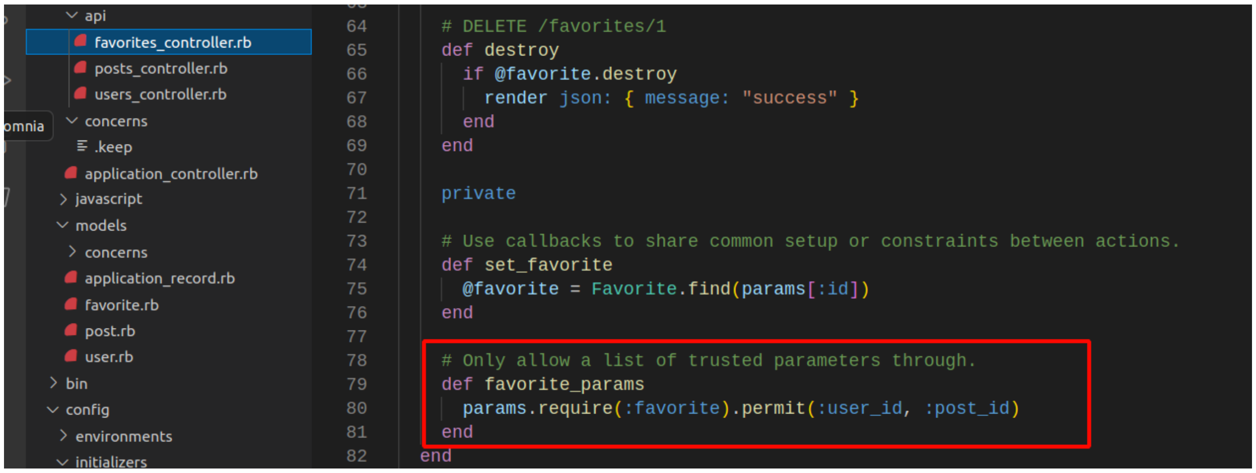
```
      else
        render json: @user.errors, status: :unprocessable_entity
      end
    end

    # DELETE /users/1
    def destroy
      @user.destroy
    end

    private

    # Use callbacks to share common setup or constraints between
    def set_user
      @user = User.find(params[:id])
    end

    # Only allow a list of trusted parameters through.
    def user_params
      params.require(:user).permit(:name, :email, :avatar)
    end
  end
end
```

Post Entity:



```
    private

    # Use callbacks to share common setup or constraints between actions.
    def set_post
      @post = Post.find(params[:id])
    end

    # Only allow a list of trusted parameters through.
    def post_params
      params.require(:post).permit(:name, :imageUrl, :directions, :ingredients, :user_id, :likes)
    end
  end
end
```

Favourite Entity:



```
    # DELETE /favorites/1
    def destroy
      if @favorite.destroy
        render json: { message: "success" }
      end
    end

    private

    # Use callbacks to share common setup or constraints between actions.
    def set_favorite
      @favorite = Favorite.find(params[:id])
    end

    # Only allow a list of trusted parameters through.
    def favorite_params
      params.require(:favorite).permit(:user_id, :post_id)
    end
  end
```

Security Requirements

Our application will need a google account to authenticate for the application. Our users will use this login to store their recipes within our database. The emails used to sign up for this application will not be shared with any other users. It's up to the full discretion of the users to choose which information they make public to other users of this application (I.E. Name, Age, email, etc.).

Documentation and Training Requirements

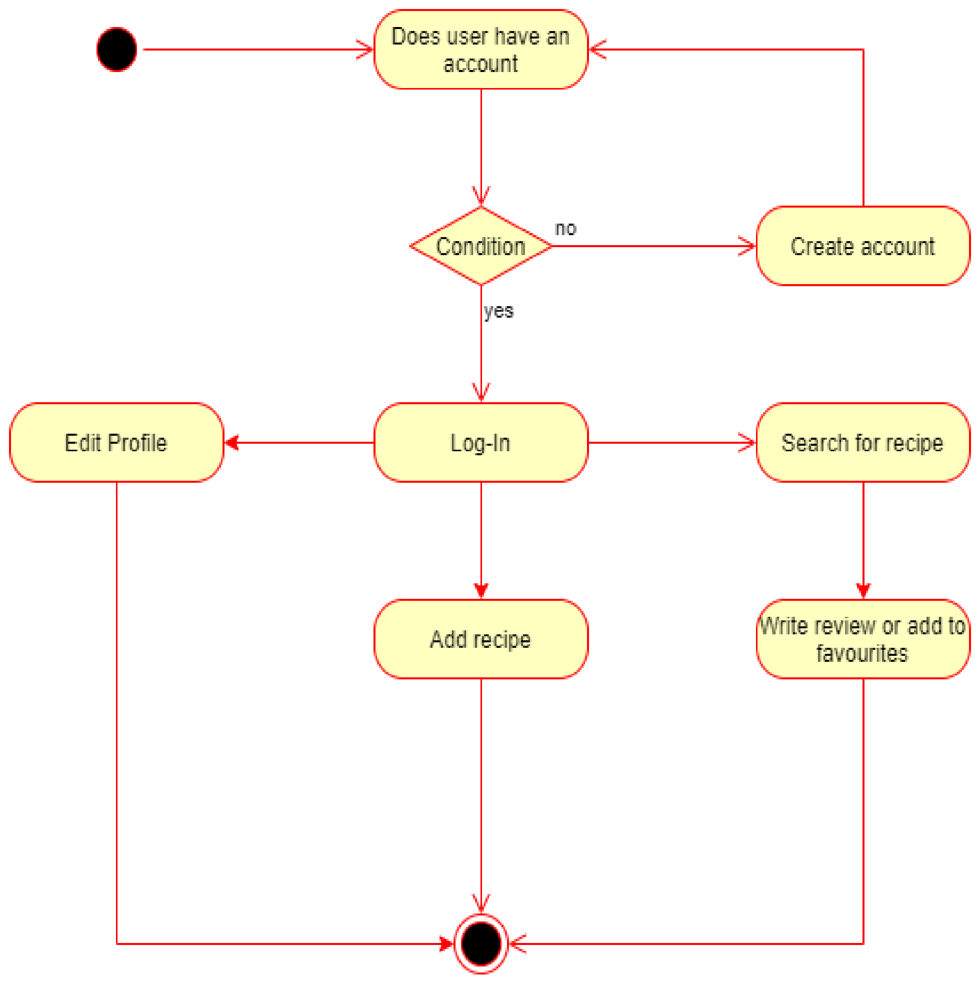
This application will be available for users to download without documentation. The application will have large clear text prompts below the buttons to allow the user to know what they are clicking.

Functional Requirements

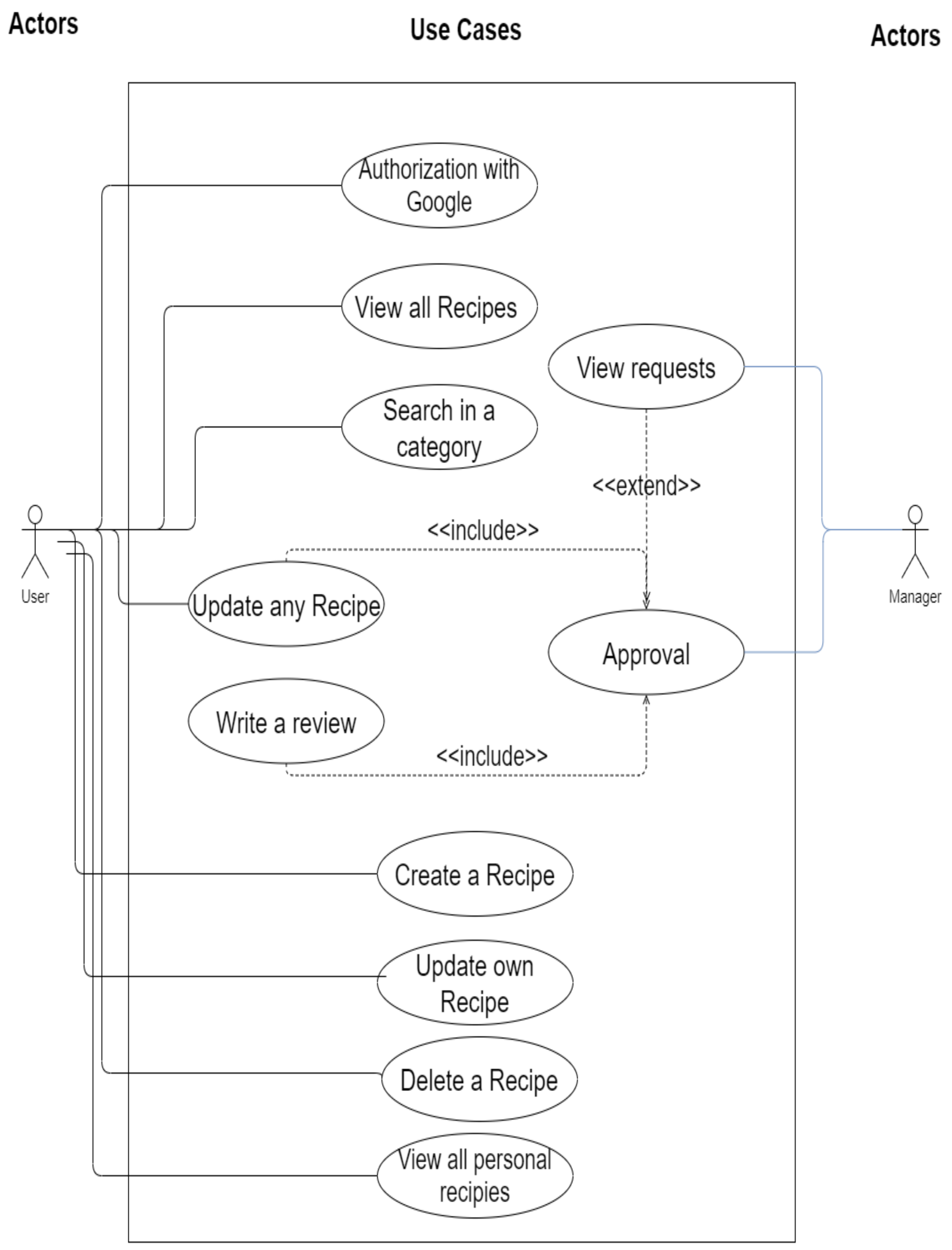
Required Features

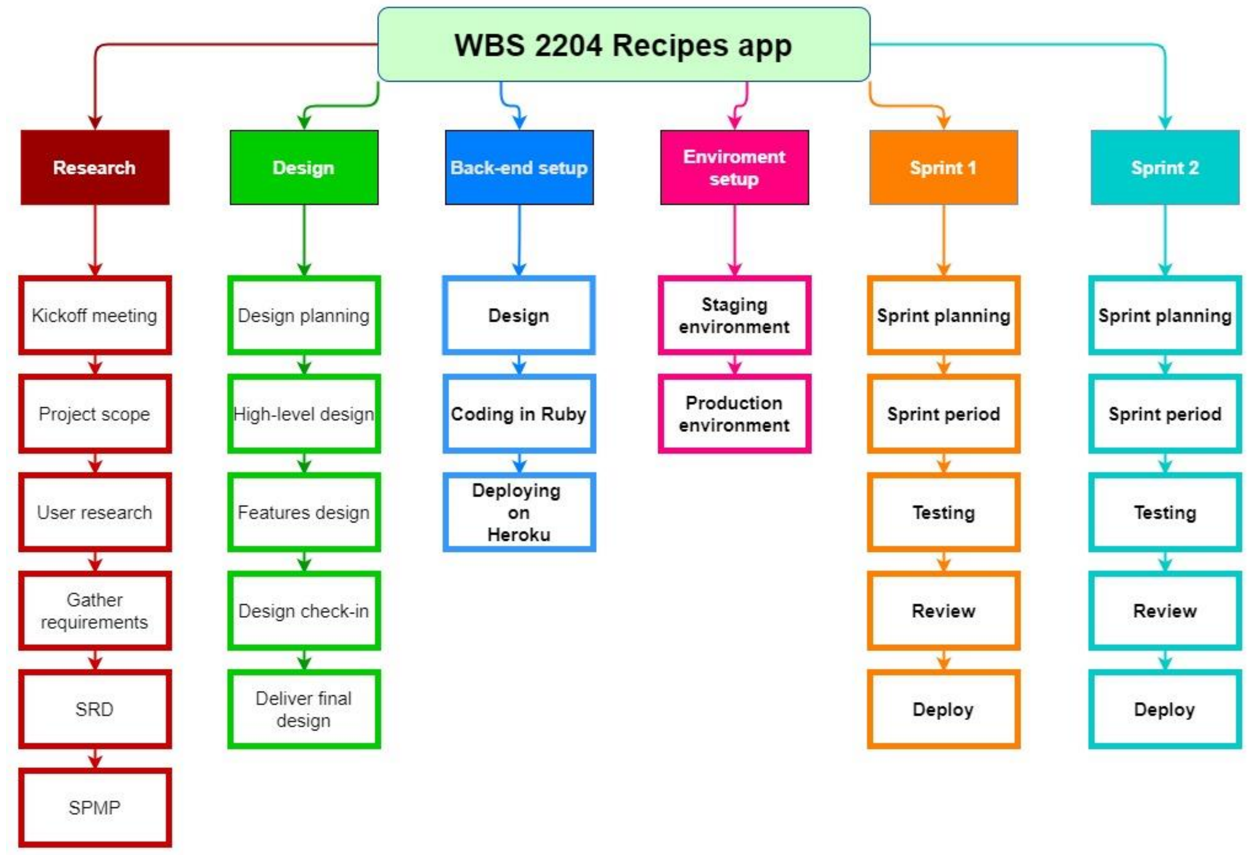
Use Case: Logging in & using the application

Recipe Gods - Activity Diagram



UML Use Case Diagram





Chapter 2: Project Plan

Overview

Purpose and Scope

Our project team developed and conducted a survey among our classmates and their family members of each participant to reach the conclusion of which app has the best potential on the Canadian market and what services our local community needs that could be satisfied by features of our application. We concluded that a Recipe Application would be our perfect choice to practice React native coding, meet the expectations of our code expert and teacher Mr. Xing Liu and potential customers. For our project we chose to code a mobile recipes app for users to be able to view, create and share their favourites with other users. Our application will allow users to create a library of all their recipes in the convenience of their phones.

Goals and Objectives

- Authentication via email and password
- Password reset
- Create, view, edit and delete own recipes
- View all recipes
- Search among recipes
- Like recipes of other users
- Sort recipes of other users
- Sort recipes by likes
- View favourite recipes
- View followers
- Help share different ethnic backgrounds with the art of food

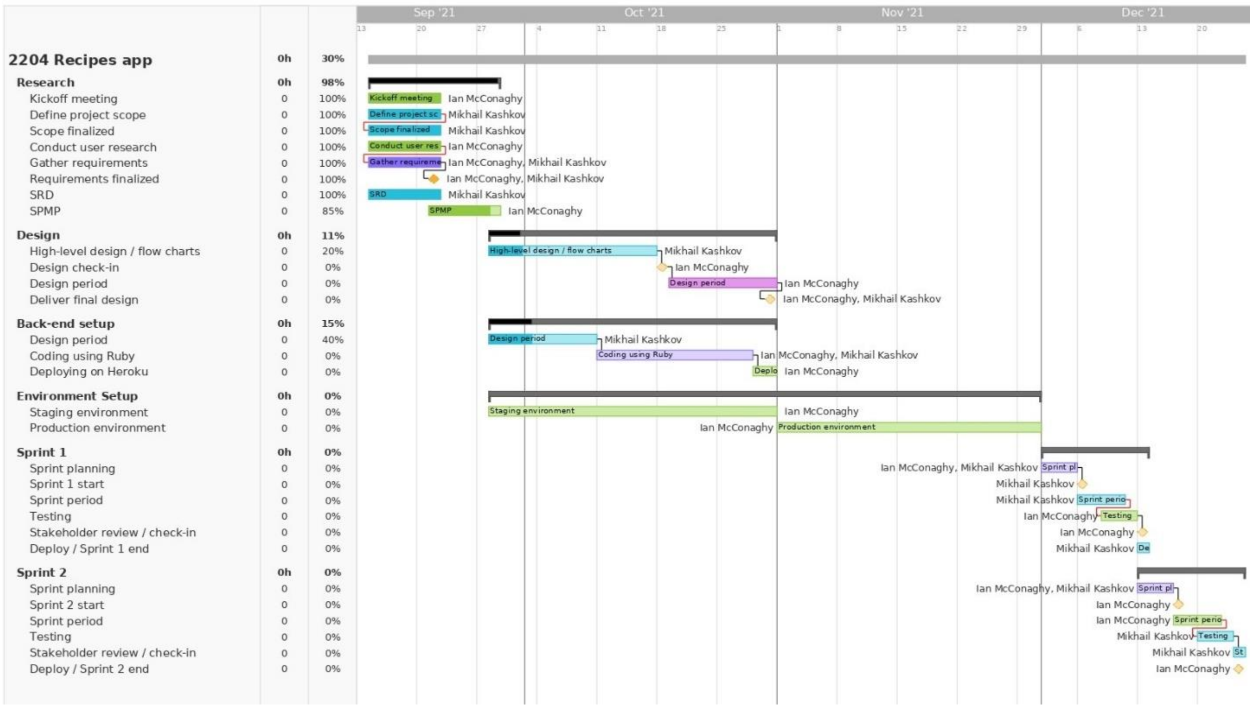
Project Deliverables

These are the main milestones for the project; however, we will break each milestone into smaller achievable tasks for each 2-week Milestone.

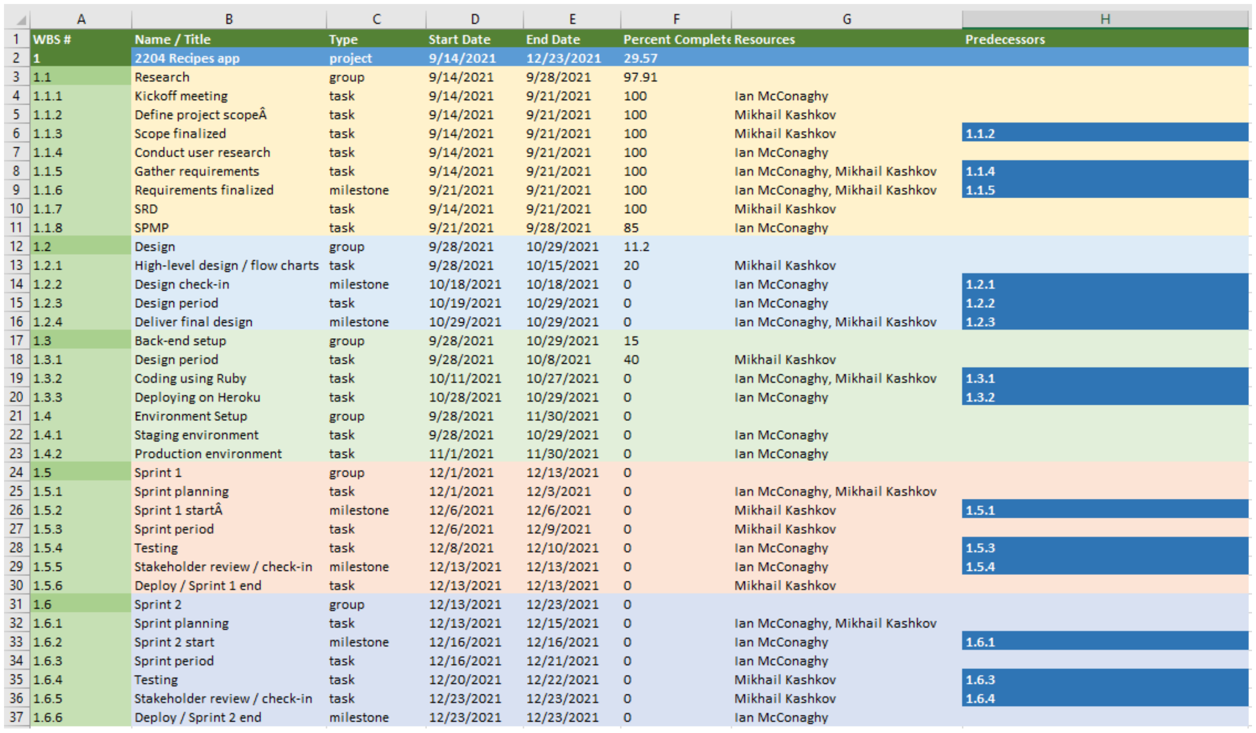
- Milestone 1: Research (SRD, SPMP) – 2 Weeks

- Milestone 2: Design – 1 – 2 Weeks
- Milestone 3: Back-End Setup – 2 Weeks
- Milestone 4: Environment Setup – 2 Weeks
- Milestone 5: Sprint 1 – 3 Weeks
- Milestone 6: Sprint 2 – 2 – 3 Weeks

6 Milestones = 14 Weeks total for project



2204 Recipes app	**0h**	**30%**
Research	**0h**	**98%**
Kickoff meeting	0	100%
Define project scope	0	100%
Scope finalized	0	100%
Conduct user research	0	100%
Gather requirements	0	100%
Requirements finalized	0	100%
SRD	0	100%
SPMP	0	85%
Design	**0h**	**11%**
High-level design / flow charts	0	20%
Design check-in	0	0%
Design period	0	0%
Deliver final design	0	0%
Back-end setup	**0h**	**15%**
Design period	0	40%
Coding using Ruby	0	0%
Deploying on Heroku	0	0%
Environment Setup	**0h**	**0%**
Staging environment	0	0%
Production environment	0	0%
Sprint 1	**0h**	**0%**
Sprint planning	0	0%
Sprint 1 start	0	0%
Sprint period	0	0%
Testing	0	0%
Stakeholder review / check-in	0	0%
Deploy / Sprint 1 end	0	0%
Sprint 2	**0h**	**0%**
Sprint planning	0	0%
Sprint 2 start	0	0%
Sprint period	0	0%
Testing	0	0%
Stakeholder review / check-in	0	0%
Deploy / Sprint 2 end	0	0%



	A	B	C	D	E	F	G	H
1	WBS #	Name / Title	Type	Start Date	End Date	Percent Complete	Resources	Predecessors
2	1	2204 Recipes app	project	9/14/2021	12/23/2021	29.57		
3	1.1	Research	group	9/14/2021	9/28/2021	97.91		
4	1.1.1	Kickoff meeting	task	9/14/2021	9/21/2021	100	Ian McConaghy	
5	1.1.2	Define project scopeÂ	task	9/14/2021	9/21/2021	100	Mikhail Kashkov	
6	1.1.3	Scope finalized	task	9/14/2021	9/21/2021	100	Mikhail Kashkov	1.1.2
7	1.1.4	Conduct user research	task	9/14/2021	9/21/2021	100	Ian McConaghy	
8	1.1.5	Gather requirements	task	9/14/2021	9/21/2021	100	Ian McConaghy, Mikhail Kashkov	1.1.4
9	1.1.6	Requirements finalized	milestone	9/21/2021	9/21/2021	100	Ian McConaghy, Mikhail Kashkov	1.1.5
10	1.1.7	SRD	task	9/14/2021	9/21/2021	100	Mikhail Kashkov	
11	1.1.8	SPMP	task	9/21/2021	9/28/2021	85	Ian McConaghy	
12	1.2	Design	group	9/28/2021	10/29/2021	11.2		
13	1.2.1	High-level design / flow charts	task	9/28/2021	10/15/2021	20	Mikhail Kashkov	
14	1.2.2	Design check-in	milestone	10/18/2021	10/18/2021	0	Ian McConaghy	1.2.1
15	1.2.3	Design period	task	10/19/2021	10/29/2021	0	Ian McConaghy	1.2.2
16	1.2.4	Deliver final design	milestone	10/29/2021	10/29/2021	0	Ian McConaghy, Mikhail Kashkov	1.2.3
17	1.3	Back-end setup	group	9/28/2021	10/29/2021	15		
18	1.3.1	Design period	task	9/28/2021	10/8/2021	40	Mikhail Kashkov	
19	1.3.2	Coding using Ruby	task	10/11/2021	10/27/2021	0	Ian McConaghy, Mikhail Kashkov	1.3.1
20	1.3.3	Deploying on Heroku	task	10/28/2021	10/29/2021	0	Ian McConaghy	1.3.2
21	1.4	Environment Setup	group	9/28/2021	11/30/2021	0		
22	1.4.1	Staging environment	task	9/28/2021	10/29/2021	0	Ian McConaghy	
23	1.4.2	Production environment	task	11/1/2021	11/30/2021	0	Ian McConaghy	
24	1.5	Sprint 1	group	12/1/2021	12/13/2021	0		
25	1.5.1	Sprint planning	task	12/1/2021	12/3/2021	0	Ian McConaghy, Mikhail Kashkov	
26	1.5.2	Sprint 1 startÂ	milestone	12/6/2021	12/6/2021	0	Mikhail Kashkov	1.5.1
27	1.5.3	Sprint period	task	12/6/2021	12/9/2021	0	Mikhail Kashkov	
28	1.5.4	Testing	task	12/8/2021	12/10/2021	0	Ian McConaghy	1.5.3
29	1.5.5	Stakeholder review / check-in	milestone	12/13/2021	12/13/2021	0	Ian McConaghy	1.5.4
30	1.5.6	Deploy / Sprint 1 end	task	12/13/2021	12/13/2021	0	Mikhail Kashkov	
31	1.6	Sprint 2	group	12/13/2021	12/23/2021	0		
32	1.6.1	Sprint planning	task	12/13/2021	12/15/2021	0	Ian McConaghy, Mikhail Kashkov	
33	1.6.2	Sprint 2 start	milestone	12/16/2021	12/16/2021	0	Ian McConaghy	1.6.1
34	1.6.3	Sprint period	task	12/16/2021	12/21/2021	0	Ian McConaghy	
35	1.6.4	Testing	task	12/20/2021	12/22/2021	0	Mikhail Kashkov	1.6.3
36	1.6.5	Stakeholder review / check-in	task	12/23/2021	12/23/2021	0	Mikhail Kashkov	1.6.4
37	1.6.6	Deploy / Sprint 2 end	milestone	12/23/2021	12/23/2021	0	Ian McConaghy	

Assumptions and Constraints

Assumptions

- Extra time during the coding portion of the application to do additional research on the technologies we are using

Constraints

- Software must be ready for presentation on December 16th 2021

Schedule and Budget Summary

- Milestone 1: Research (SRD, SPMP) – 2 Weeks
- Milestone 2: Design – 1 – 2 Weeks
- Milestone 3: Back-End Setup – 2 Weeks

- Milestone 4: Environment Setup – 2 Weeks
- Milestone 5: Sprint 1 – 3 Weeks
- Milestone 6: Sprint 2 – 2 – 3 Weeks

Success Criteria

- Making sure all Milestones are followed to date
- Making sure application is written so it’s accessible to the majority user
- Making sure our database is set up correctly and saving all the necessary data
- Making sure our application has a fool proof and secure user sign up and sign out feature

Definitions and Acronyms

Application – Software described in this document

Client – Any person that isn’t invested in the project that is not a developer

Database – Collection of the user information that is stored withing the application to allow users ease of access throughout their user experience

Developer – Persons who develop the software for the user

Use Case – A model that shows all the user functionality throughout the application

Evolution of the Project Plan

Based on the speed of how we complete our milestones we will adjust our timeline throughout the term. Risk will be added to the project as they turn up and we will mitigate them as we see fit.

Start-Up Plan

Team Organization

Ian – Designer and Writing specialist

Michael – Developer and back-end specialist

Project Communication

Our team will be communicating through discord and zoom. We will also use Google drive to share any written portions of the course and use GitHub to store all code that we collaborate on.

Technical Process

We will be following the Work Breakdown Structure for our project. This structure will ensure we complete our project at the set upon date.

Tools

- Visual Studio

    - React Native
    - RAILS API
- Windows Laptop
- Apple Laptop
- Heroku
- GitHub
- Google Drive
- Discord
- Expo Go

Work Plan

Activities and Tasks

Ian:

- Kick-off Meeting
- Conduct User Research
- Gather requirements
- SRD and SPMP
- Design and Workflow
- Design Period
- Coding with Ruby
- Staging Environment
- Production Environment
- Sprint Planning

- Deploy to Heroku / sprint 2 end
- Final Report

Michael:

- Define Project Scope
- Scope Finalized
- Gather requirements
- SRD and SPMP
- Design and Workflow
- Design period
- Coding with Ruby
- Sprint planning
- Sprint 1 Start
- Sprint 1 period
- Testing
- Stakeholder review / check in

Release Plan

- Milestone 1: Research (SRD, SPMP) – 2 Weeks
- Milestone 2: Design – 1 – 2 Weeks
- Milestone 3: Back-End Setup – 2 Weeks
- Milestone 4: Environment Setup – 2 Weeks
- Milestone 5: Sprint 1 – 3 Weeks
- Milestone 6: Sprint 2 – 2 – 3 Weeks

After we complete these 6 milestones, we will start the Final report as well as prepare for our presentation.

Iteration Plan

- Do our research
- Create wireframes of the application
- Complete components of the application and test
- Fix any bugs
- Save to GitHub

Budget

This project has no financial costs.

Control Plan

Monitoring and Control

- 1 to 2 meetings weekly to confirm what parts of the project we are working on, as well as update team members on any complete tasks

Support Process Plans

Risk Identification

- Health issues: If one of us get sick during the term it will slow down some of our milestones
- Project is lost/corrupted: Even with a backup locally and on GitHub. If this were to happen, we would likely have to figure out what corners are feasible to cut, in order to complete the project at the set end date

Policies and Contingency Plans

- If one of us were to get sick, we would update our team member daily to inform them on what they missed in class, as well as what was worked on.
- If our project was lost/corrupted, we would first check if our project is stored locally. We will also use cloud backups for every milestone to ensure this does not happen

Risk Management Plan

Any new risks that come up during the project will be added to the support process and we will figure out a policy and procedure that follows suit.

Verification and Validation Plan

- Issues will be tracked on our Trello board
- Questionnaire midway through on progress
- Code will be checked at each milestone to ensure quality and consistency

Product Acceptance Plan

- Application opens smoothly
- User Authentication Is quick
- The application is delivered in time for presentation
- No issues on GitHub
- All Milestones completed

Chapter 3: Project Design

Introduction

Purpose

Our project team developed and conducted a survey among our classmates and their family members of each participant to reach the conclusion of which app has the best potential on the Canadian market and what services our local community needs that could be satisfied by features of our application. We concluded that a Recipe Application would be our perfect choice to practice React native coding, meet the expectations of our code expert and teacher Mr. Xing Liu and potential customers. For our project we chose to code a mobile recipes app for users to be able to view, create and share their favourites with other users. Our application will allow users to create a library of all their recipes in the convenience of their phones.

Scope

The scope of our project is to give users the ability to redefine how they create, store and share recipes with friend and family.

Definitions and Acronyms

Application – Software described in this document

Client – Any person that isn't invested in the project that is not a developer

Database – Collection of the user information that is stored withing the application to allow users ease of access throughout their user experience

Developer – Persons who develop the software for the user

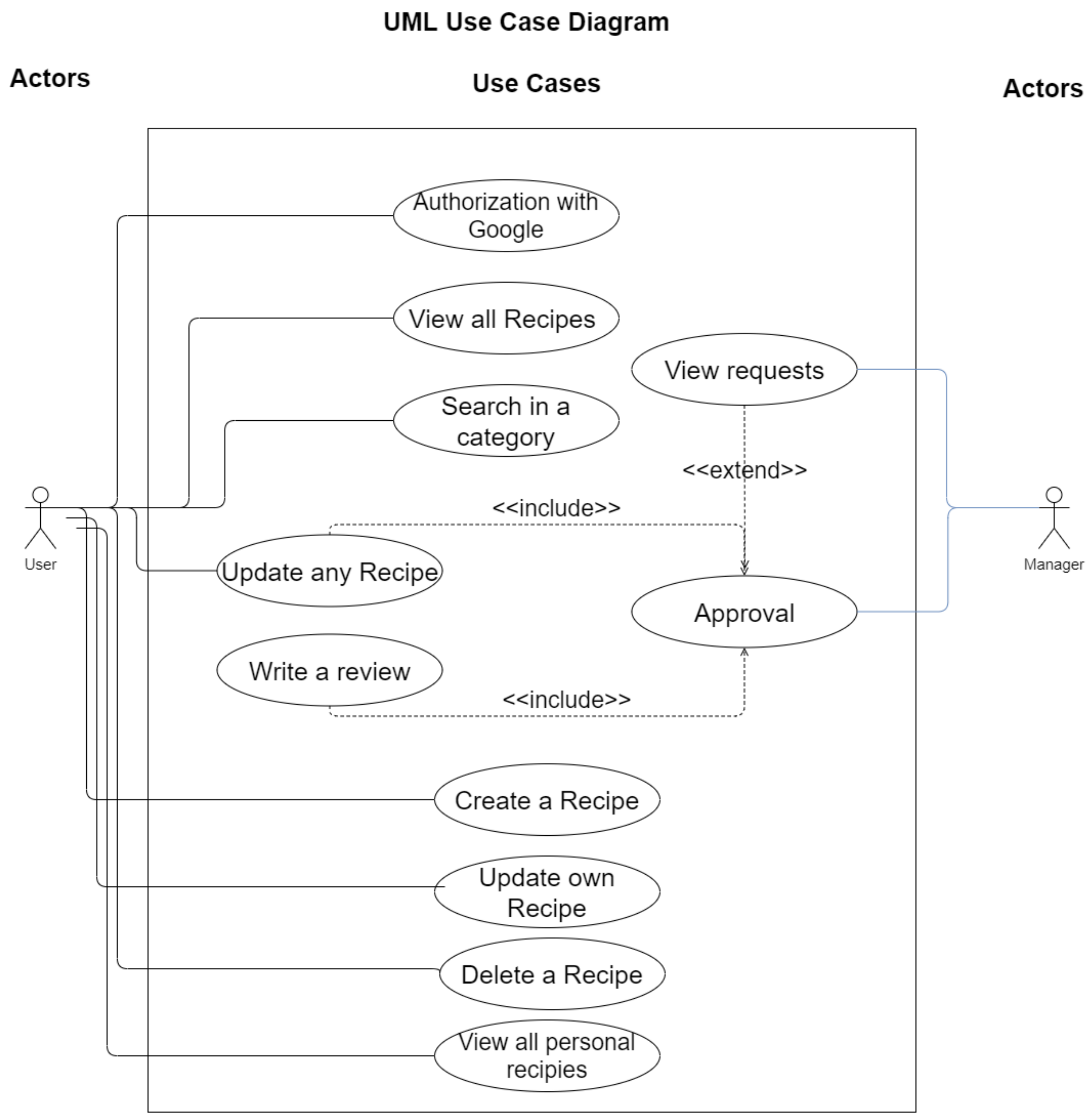
Use Case – A model that shows all the user functionality throughout the application

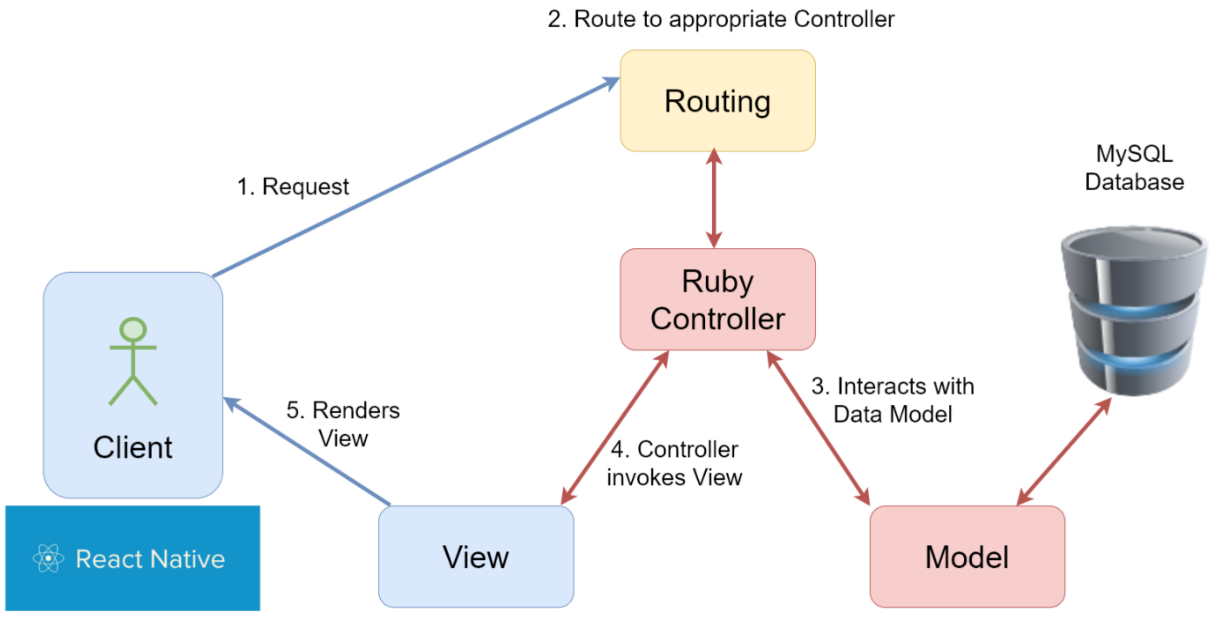
References

- RUBY API Documentation
- GitHub Documentation

System Overview

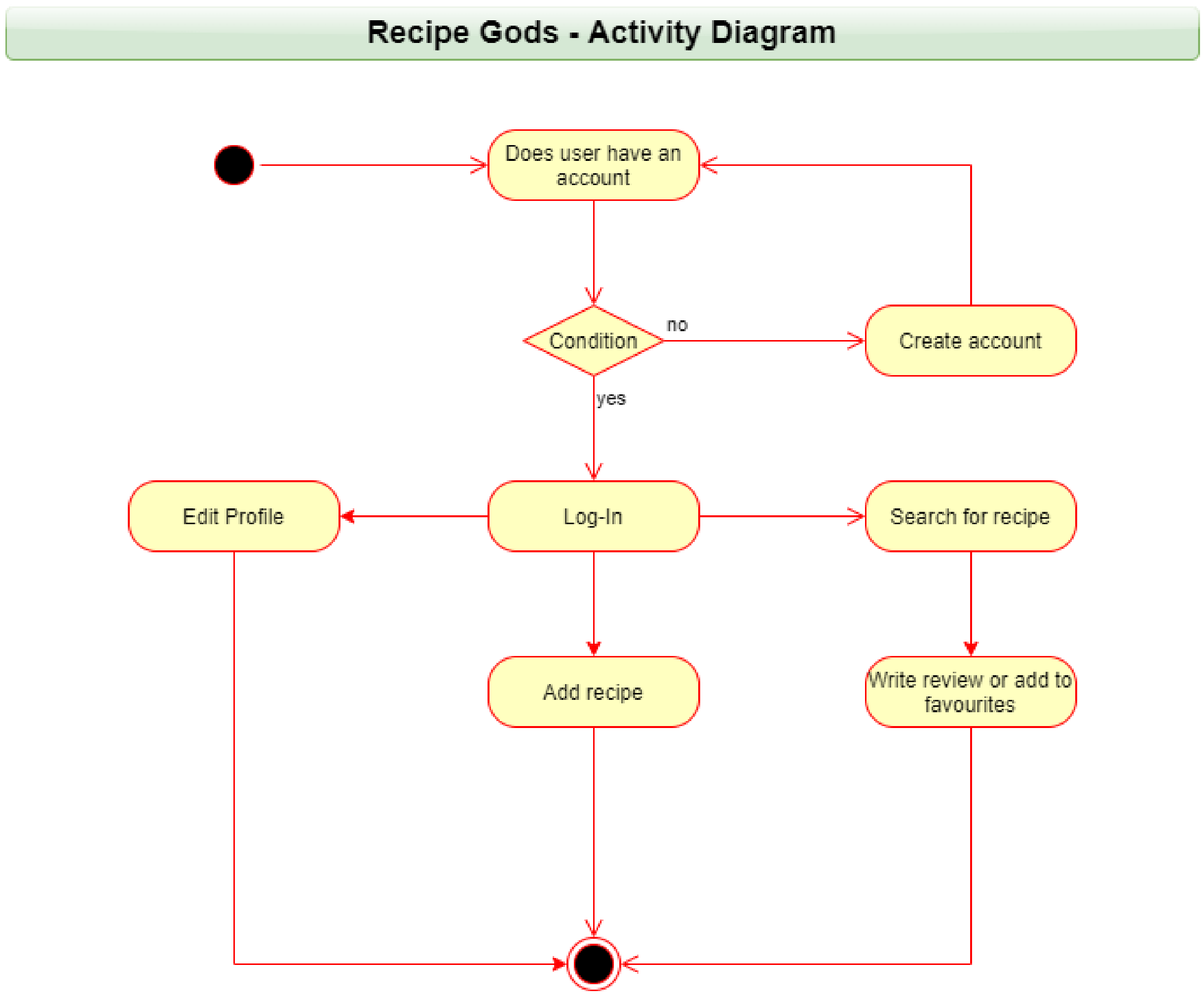
Use Case Diagram

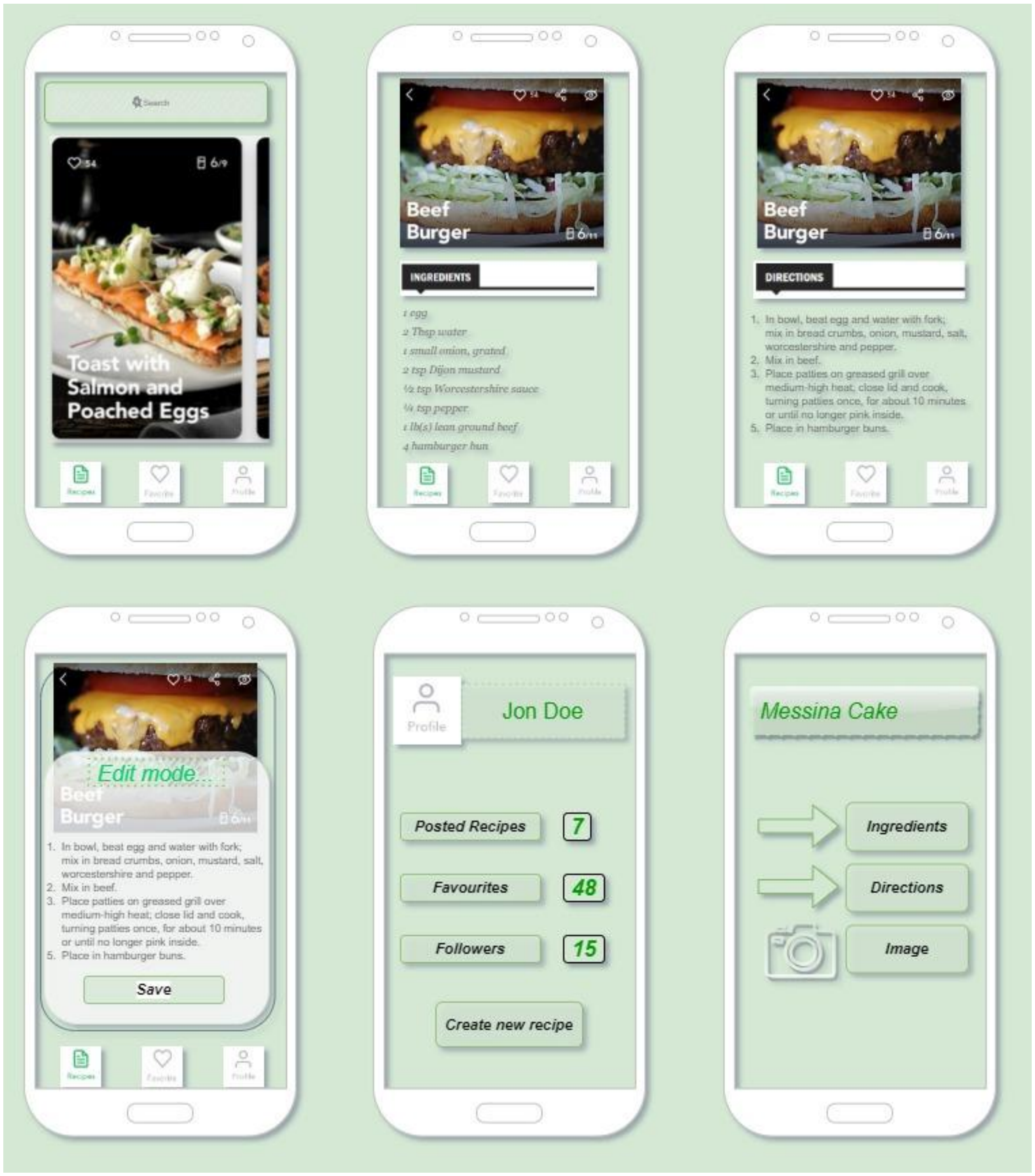



2. Route to appropriate Controller
Routing
1. Request
MySQL Database
Ruby Controller
Client
5. Renders View
3. Interacts with Data Model
4. Controller invokes View
React Native
View
Model

System Architecture

Architectural Design


Recipe Gods - Activity Diagram
Does user have an account
Condition
no
Create account
yes
Edit Profile
Log-In
Search for recipe
Add recipe
Write review or add to favourites



1 egg
2 Tbsp water
1 small onion, grated
2 tsp Dijon mustard
½ tsp Worcestershire sauce
¼ tsp pepper
1 lb(s) lean ground beef
4 hamburger bun

1. In bowl, beat egg and water with fork; mix in bread crumbs, onion, mustard, salt, worcestershire and pepper.
2. Mix in beef.
3. Place patties on greased grill over medium-high heat; close lid and cook, turning patties once, for about 10 minutes or until no longer pink inside.
5. Place in hamburger buns.

1. In bowl, beat egg and water with fork; mix in bread crumbs, onion, mustard, salt, worcestershire and pepper.
2. Mix in beef.
3. Place patties on greased grill over medium-high heat; close lid and cook, turning patties once, for about 10 minutes or until no longer pink inside.
5. Place in hamburger buns.

Data Design

Data Description

The data that will be stored within our system consists of users, recipes and images. Each user has Posted Recipes, Favourites and Followers saved.

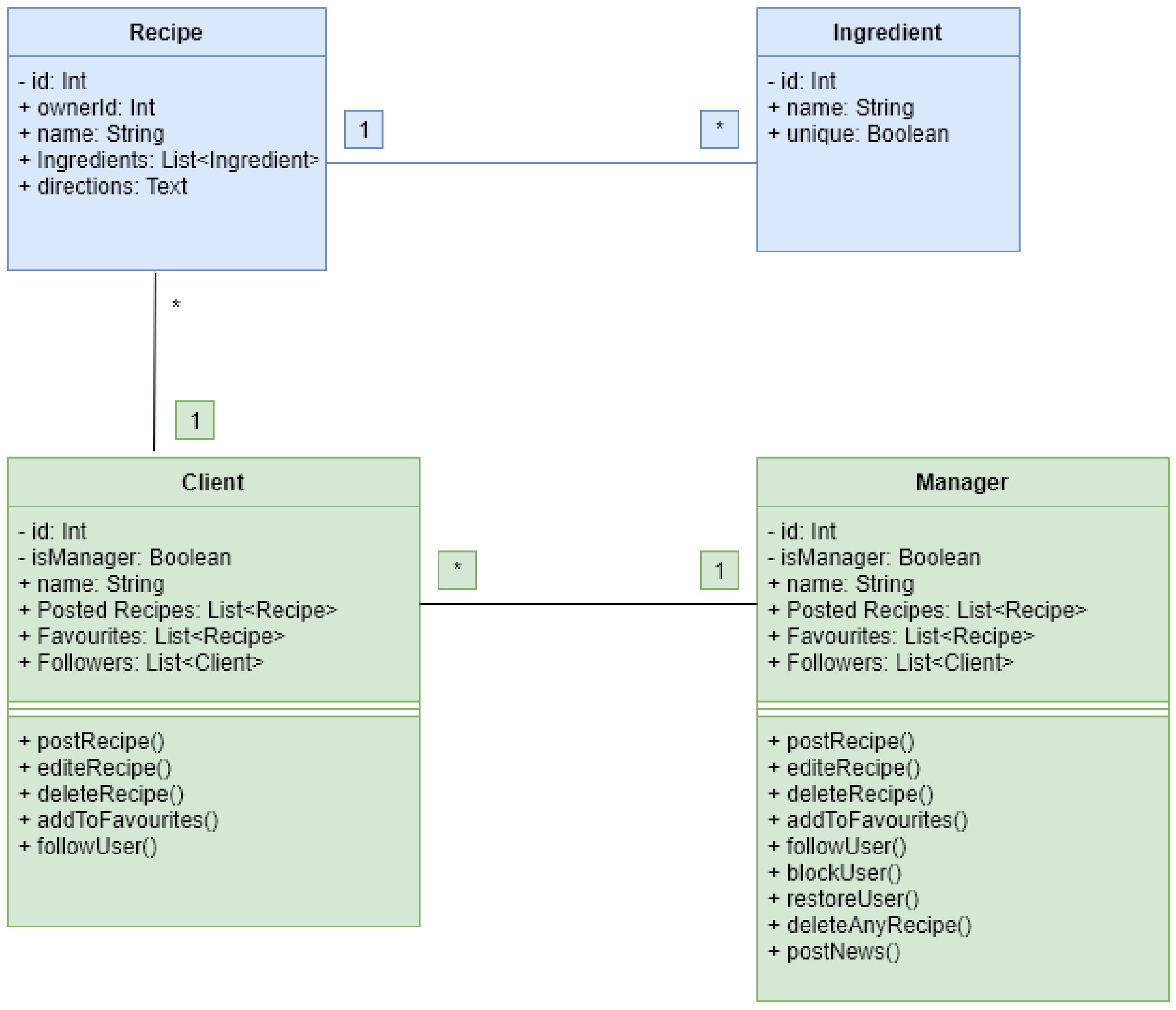
Data Storage

The data for our application will be stored using Firebase using RUBY API for full CRUD functionality.

Component Design

Classes in Components

The classes in this application are Recipe, Ingredients, Client and Manager.



User Interface Design

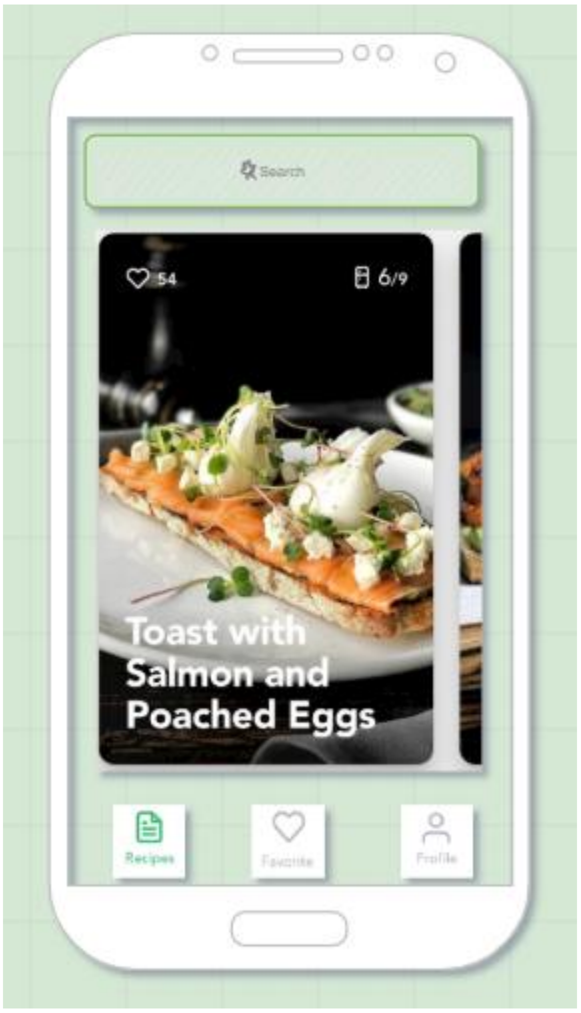
Overview of User Interface

The initial user interface will be a login screen that allows users to login to their account.

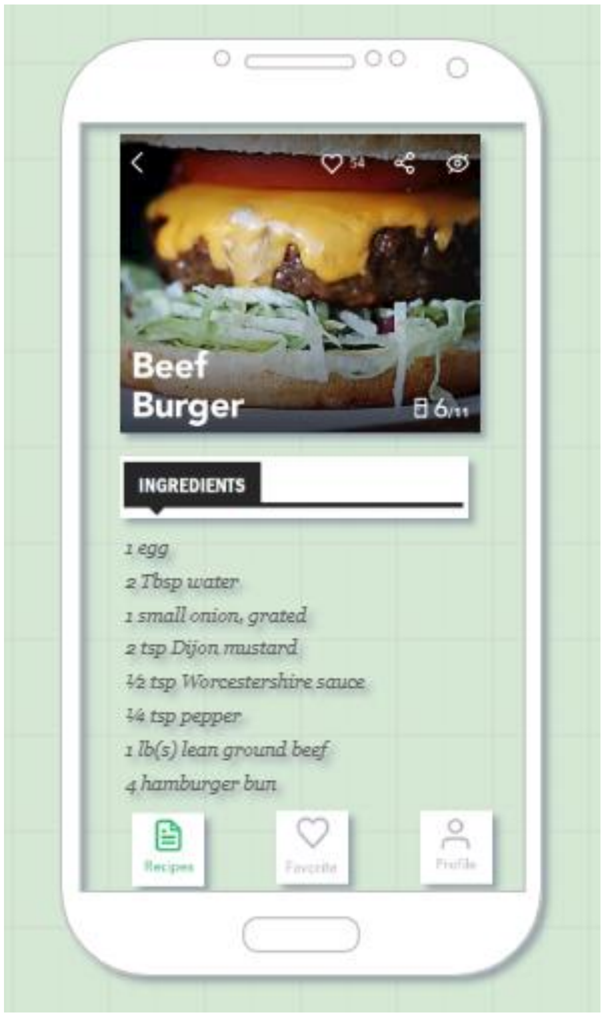
Once logged in, users will be shown the homepage that will show all available recipes. They can use our live search functionality. If they want to favourite a recipe they can, and then it will be moved to their favourites, which is in the navigation at the bottom. Users can post a new recipe, as well as do full CRUD operation. Any recipe on the home screen can be clicked on, so they can get more details about it.

Screen Design

Home Screen

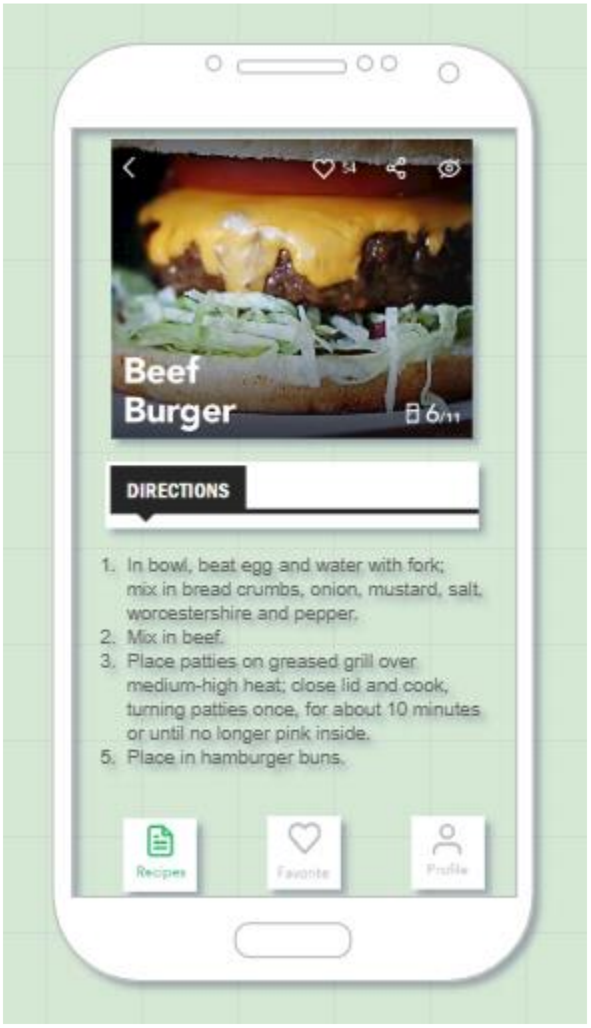


Recipe Card – Ingredients



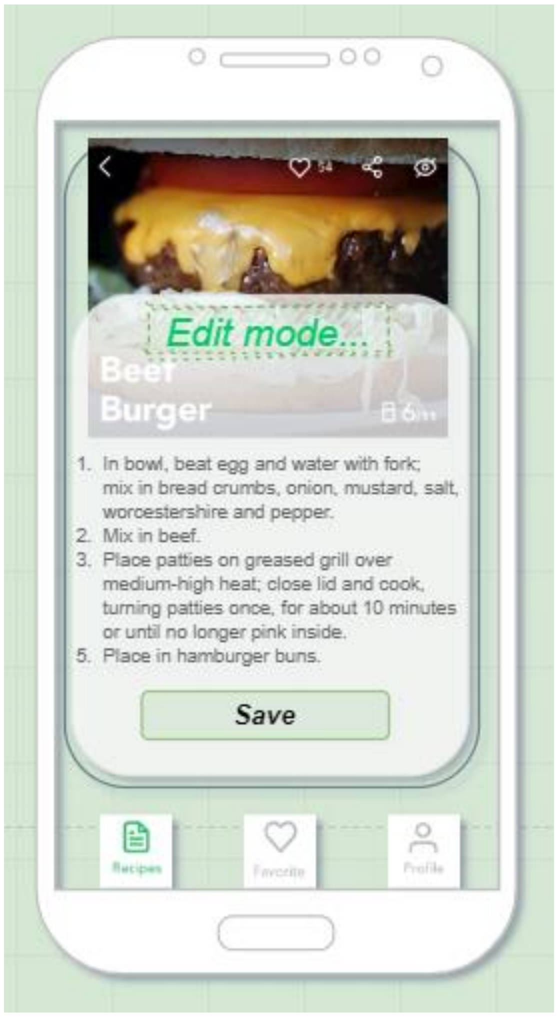
1 egg
2 Tbsp water
1 small onion, grated
2 tsp Dijon mustard
½ tsp Worcestershire sauce
¼ tsp pepper
1 lb(s) lean ground beef
4 hamburger bun

Recipe Card – Directions



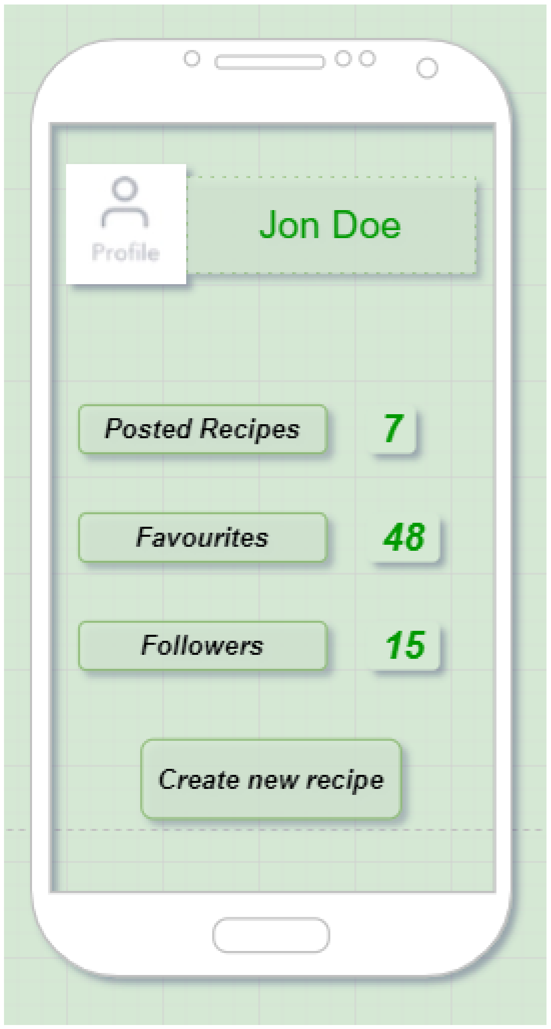
1. In bowl, beat egg and water with fork; mix in bread crumbs, onion, mustard, salt, worcestershire and pepper.
2. Mix in beef.
3. Place patties on greased grill over medium-high heat; close lid and cook, turning patties once, for about 10 minutes or until no longer pink inside.
5. Place in hamburger buns.

Recipe Card – Edit Mode (if owner of recipe)

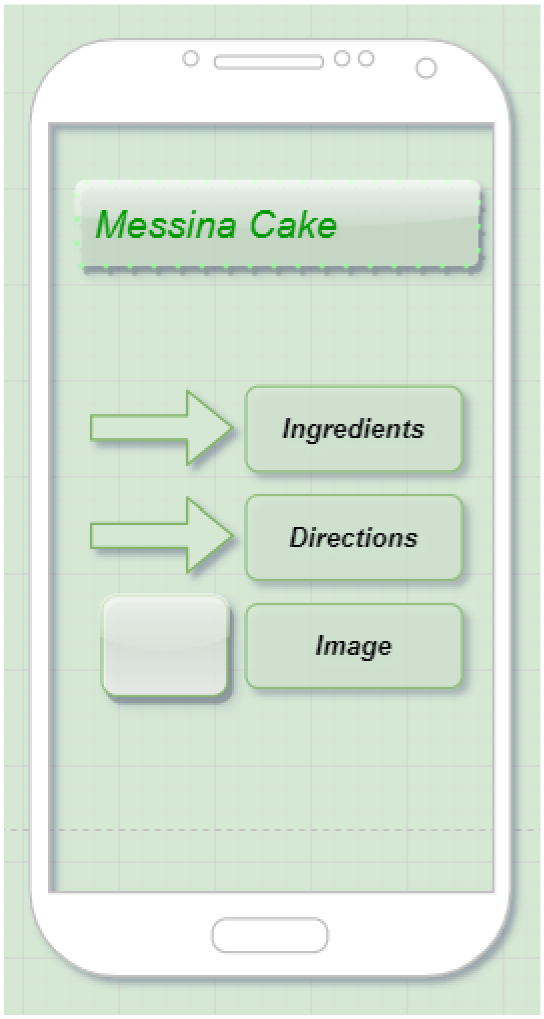


1. In bowl, beat egg and water with fork; mix in bread crumbs, onion, mustard, salt, worcestershire and pepper.
2. Mix in beef.
3. Place patties on greased grill over medium-high heat; close lid and cook, turning patties once, for about 10 minutes or until no longer pink inside.
5. Place in hamburger buns.

User Profile Screen



Add new recipe

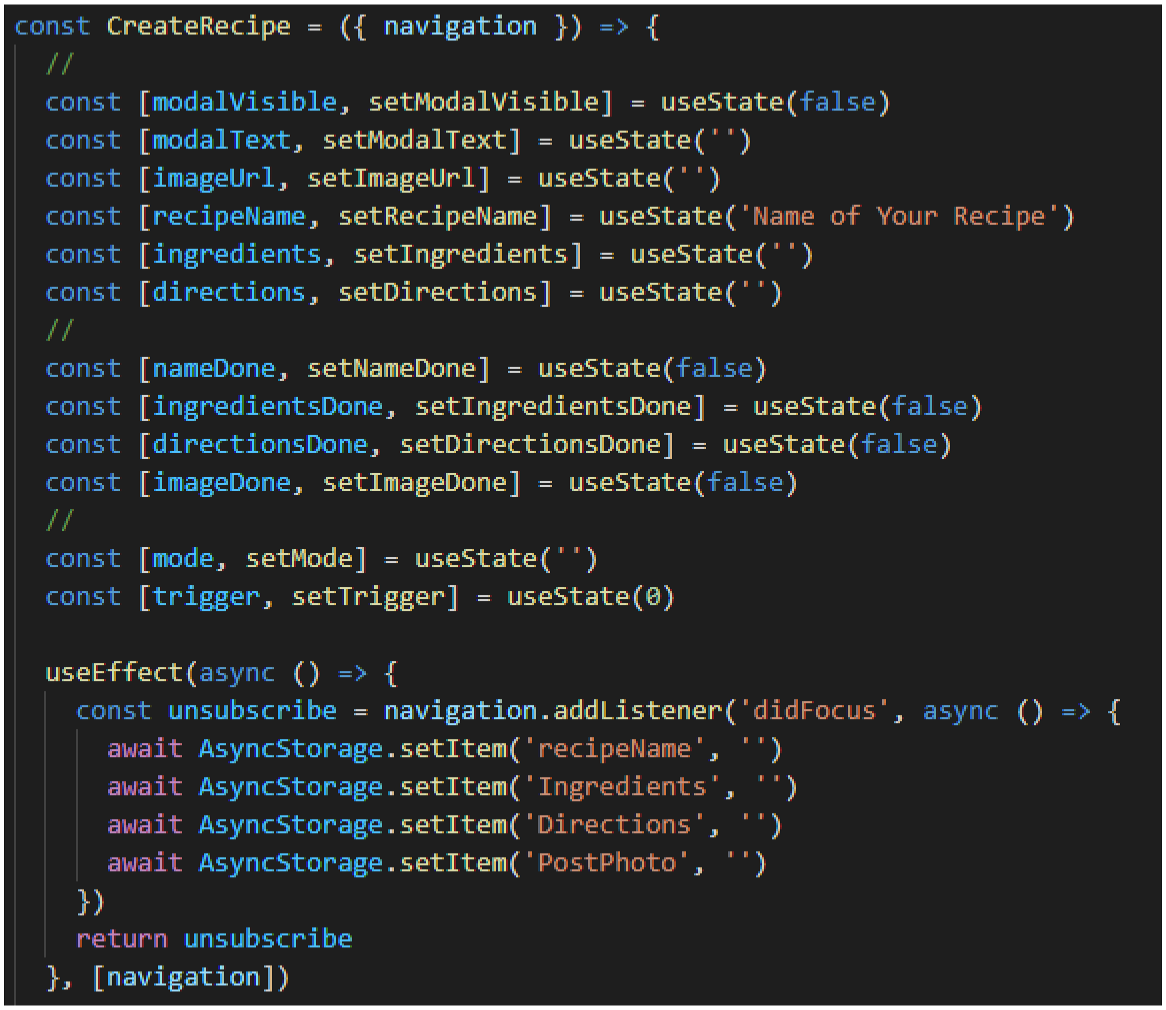


Chapter 4: Project Implementation

Classes and Class Diagrams

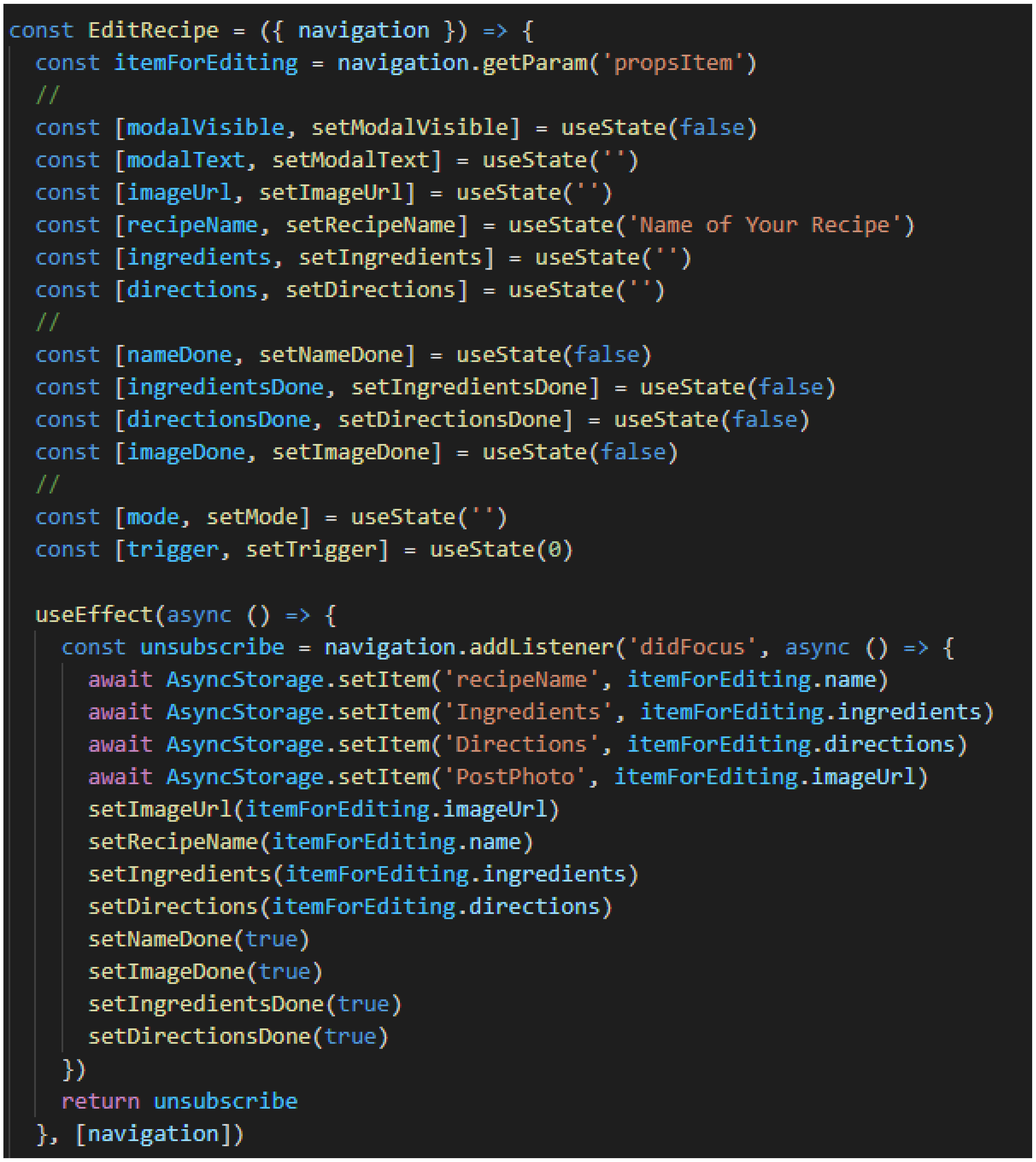
Individual And Detailed Class Diagrams

Create Recipe Class



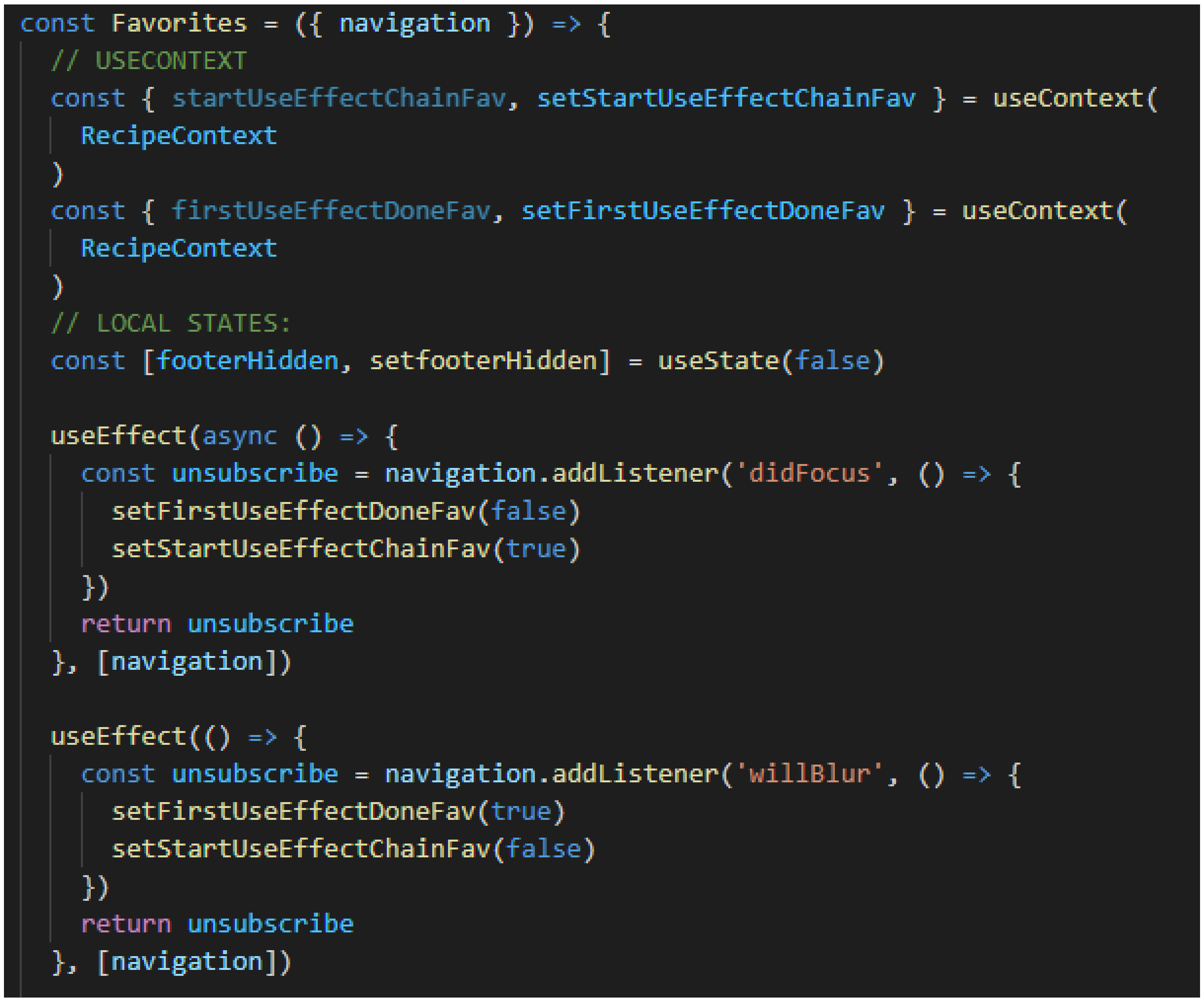
```
const CreateRecipe = ({ navigation }) => {
  //
  const [modalVisible, setModalVisible] = useState(false)
  const [modalText, setModalText] = useState('')
  const [imageUrl, setImageUrl] = useState('')
  const [recipeName, setRecipeName] = useState('Name of Your Recipe')
  const [ingredients, setIngredients] = useState('')
  const [directions, setDirections] = useState('')
  //
  const [nameDone, setNameDone] = useState(false)
  const [ingredientsDone, setIngredientsDone] = useState(false)
  const [directionsDone, setDirectionsDone] = useState(false)
  const [imageDone, setImageDone] = useState(false)
  //
  const [mode, setMode] = useState('')
  const [trigger, setTrigger] = useState(0)

  useEffect(async () => {
    const unsubscribe = navigation.addListener('didFocus', async () => {
      await AsyncStorage.setItem('recipeName', '')
      await AsyncStorage.setItem('Ingredients', '')
      await AsyncStorage.setItem('Directions', '')
      await AsyncStorage.setItem('PostPhoto', '')
    })
    return unsubscribe
  }, [navigation])
```

Edit Recipe Class



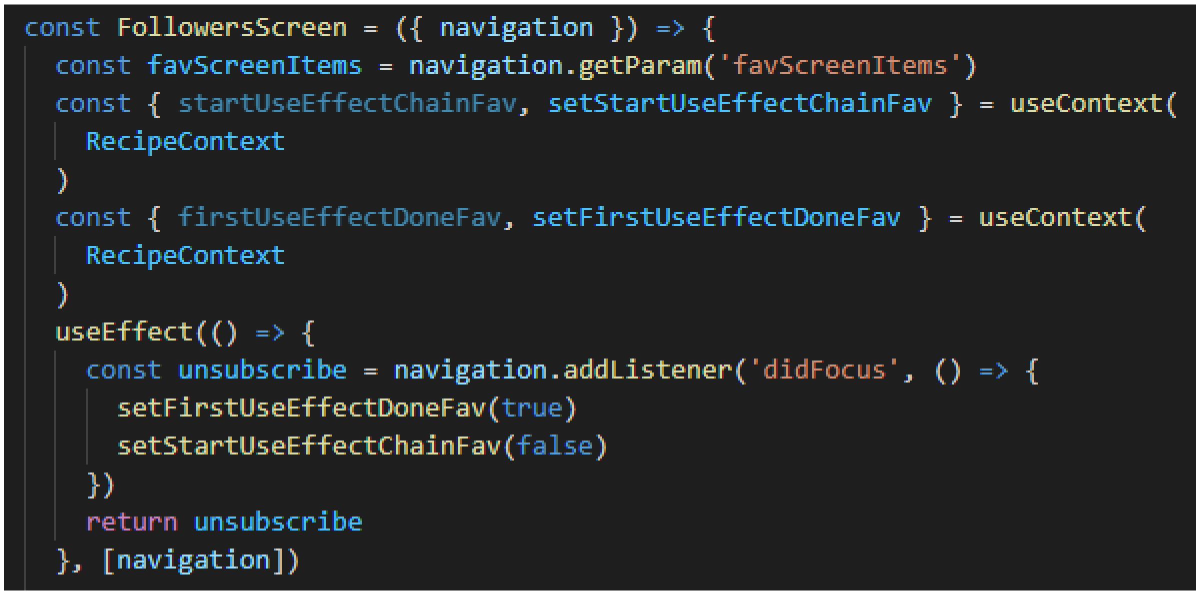
```
const EditRecipe = ({ navigation }) => {
  const itemForEditing = navigation.getParam('propsItem')
  //
  const [modalVisible, setModalVisible] = useState(false)
  const [modalText, setModalText] = useState('')
  const [imageUrl, setImageUrl] = useState('')
  const [recipeName, setRecipeName] = useState('Name of Your Recipe')
  const [ingredients, setIngredients] = useState('')
  const [directions, setDirections] = useState('')
  //
  const [nameDone, setNameDone] = useState(false)
  const [ingredientsDone, setIngredientsDone] = useState(false)
  const [directionsDone, setDirectionsDone] = useState(false)
  const [imageDone, setImageDone] = useState(false)
  //
  const [mode, setMode] = useState('')
  const [trigger, setTrigger] = useState(0)

  useEffect(async () => {
    const unsubscribe = navigation.addListener('didFocus', async () => {
      await AsyncStorage.setItem('recipeName', itemForEditing.name)
      await AsyncStorage.setItem('Ingredients', itemForEditing.ingredients)
      await AsyncStorage.setItem('Directions', itemForEditing.directions)
      await AsyncStorage.setItem('PostPhoto', itemForEditing.imageUrl)
      setImageUrl(itemForEditing.imageUrl)
      setRecipeName(itemForEditing.name)
      setIngredients(itemForEditing.ingredients)
      setDirections(itemForEditing.directions)
      setNameDone(true)
      setImageDone(true)
      setIngredientsDone(true)
      setDirectionsDone(true)
    })
    return unsubscribe
  }, [navigation])
```

Favourites Class



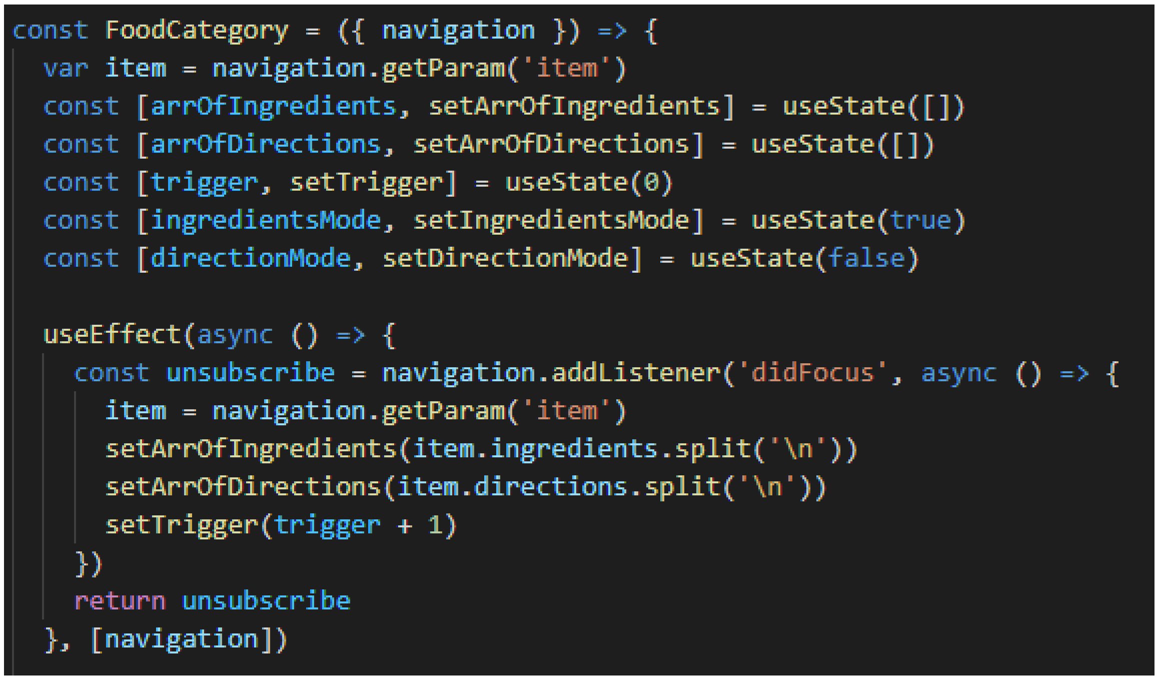
```
const Favorites = ({ navigation }) => {
  // USECONTEXT
  const { startUseEffectChainFav, setStartUseEffectChainFav } = useContext(
    RecipeContext
  )
  const { firstUseEffectDoneFav, setFirstUseEffectDoneFav } = useContext(
    RecipeContext
  )
  // LOCAL STATES:
  const [footerHidden, setfooterHidden] = useState(false)

  useEffect(async () => {
    const unsubscribe = navigation.addListener('didFocus', () => {
      setFirstUseEffectDoneFav(false)
      setStartUseEffectChainFav(true)
    })
    return unsubscribe
  }, [navigation])

  useEffect(() => {
    const unsubscribe = navigation.addListener('willBlur', () => {
      setFirstUseEffectDoneFav(true)
      setStartUseEffectChainFav(false)
    })
    return unsubscribe
  }, [navigation])
```

Followers Class



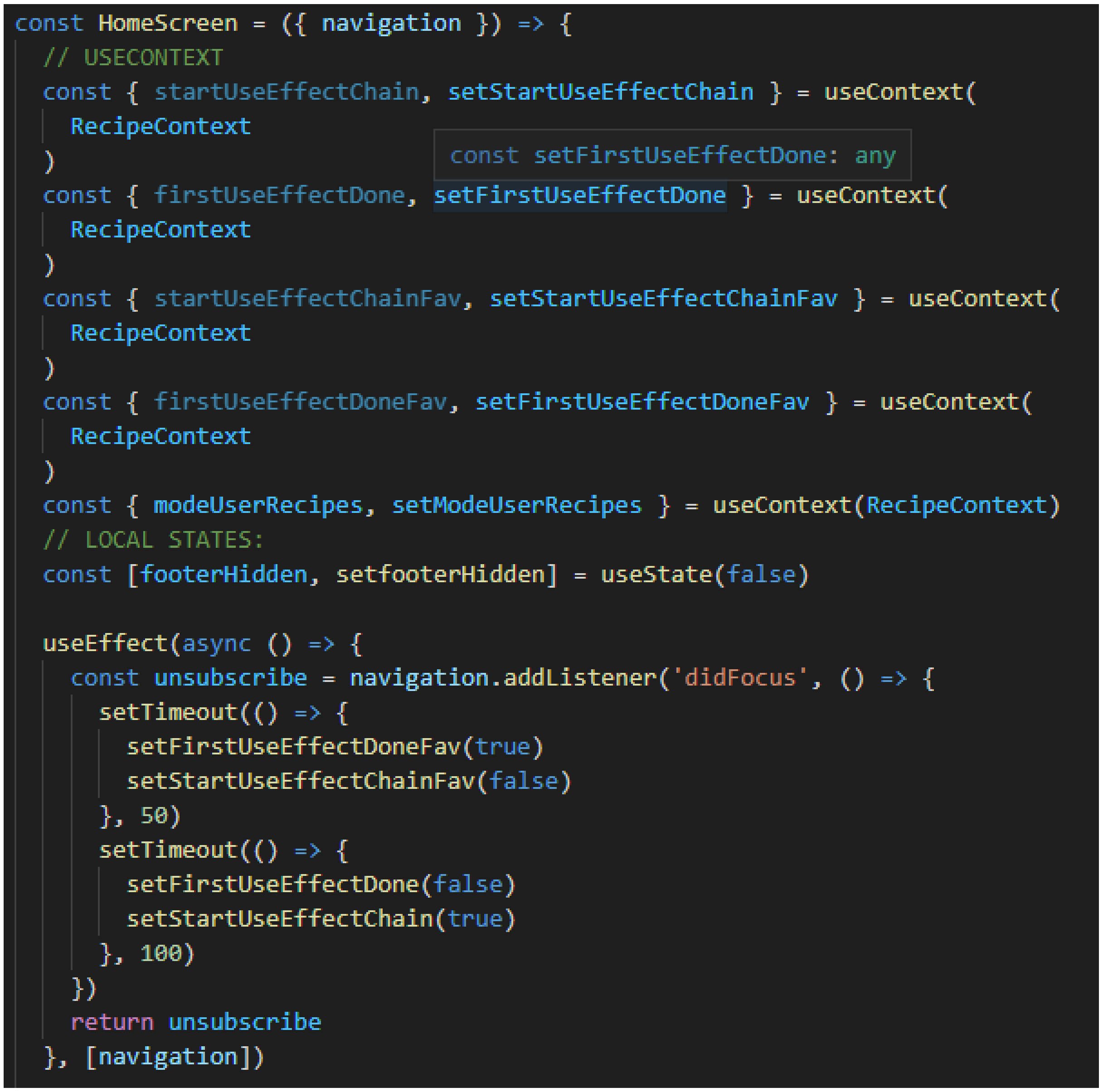
```
const FollowersScreen = ({ navigation }) => {
  const favScreenItems = navigation.getParam('favScreenItems')
  const { startUseEffectChainFav, setStartUseEffectChainFav } = useContext(
    RecipeContext
  )
  const { firstUseEffectDoneFav, setFirstUseEffectDoneFav } = useContext(
    RecipeContext
  )
  useEffect(() => {
    const unsubscribe = navigation.addListener('didFocus', () => {
      setFirstUseEffectDoneFav(true)
      setStartUseEffectChainFav(false)
    })
    return unsubscribe
  }, [navigation])
```

Food Category Class



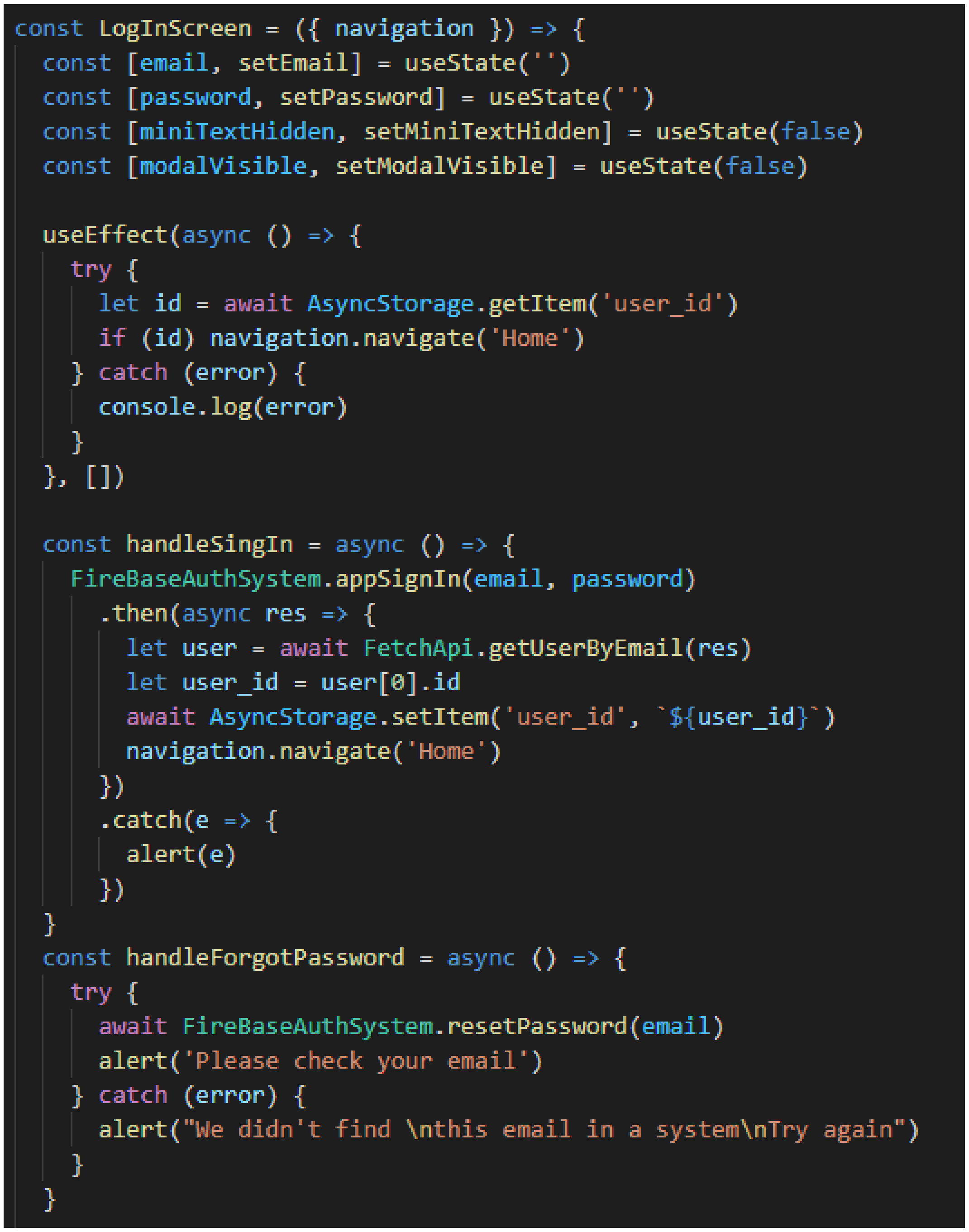
```
const FoodCategory = ({ navigation }) => {
  var item = navigation.getParam('item')
  const [arrOfIngredients, setArrOfIngredients] = useState([])
  const [arrOfDirections, setArrOfDirections] = useState([])
  const [trigger, setTrigger] = useState(0)
  const [ingredientsMode, setIngredientsMode] = useState(true)
  const [directionMode, setDirectionMode] = useState(false)

  useEffect(async () => {
    const unsubscribe = navigation.addListener('didFocus', async () => {
      item = navigation.getParam('item')
      setArrOfIngredients(item.ingredients.split('\n'))
      setArrOfDirections(item.directions.split('\n'))
      setTrigger(trigger + 1)
    })
    return unsubscribe
  }, [navigation])
```

Home Screen Class



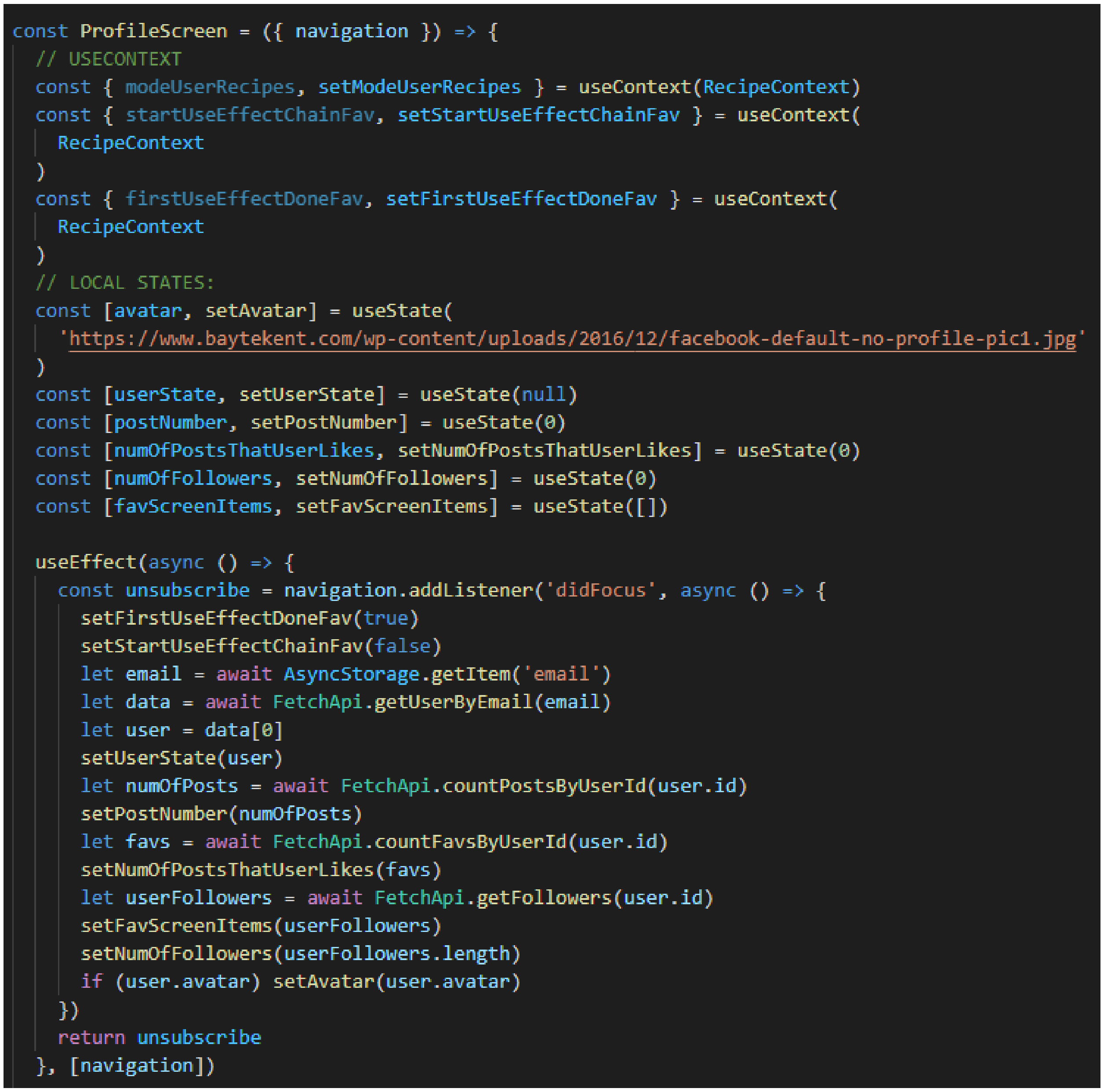
```
const HomeScreen = ({ navigation }) => {
  // USECONTEXT
  const { startUseEffectChain, setStartUseEffectChain } = useContext(
    RecipeContext
  )                                   const setFirstUseEffectDone: any
  const { firstUseEffectDone, setFirstUseEffectDone } = useContext(
    RecipeContext
  )
  const { startUseEffectChainFav, setStartUseEffectChainFav } = useContext(
    RecipeContext
  )
  const { firstUseEffectDoneFav, setFirstUseEffectDoneFav } = useContext(
    RecipeContext
  )
  const { modeUserRecipes, setModeUserRecipes } = useContext(RecipeContext)
  // LOCAL STATES:
  const [footerHidden, setfooterHidden] = useState(false)

  useEffect(async () => {
    const unsubscribe = navigation.addListener('didFocus', () => {
      setTimeout(() => {
        setFirstUseEffectDoneFav(true)
        setStartUseEffectChainFav(false)
      }, 50)
      setTimeout(() => {
        setFirstUseEffectDone(false)
        setStartUseEffectChain(true)
      }, 100)
    })
    return unsubscribe
  }, [navigation])
```

Log In Screen Class



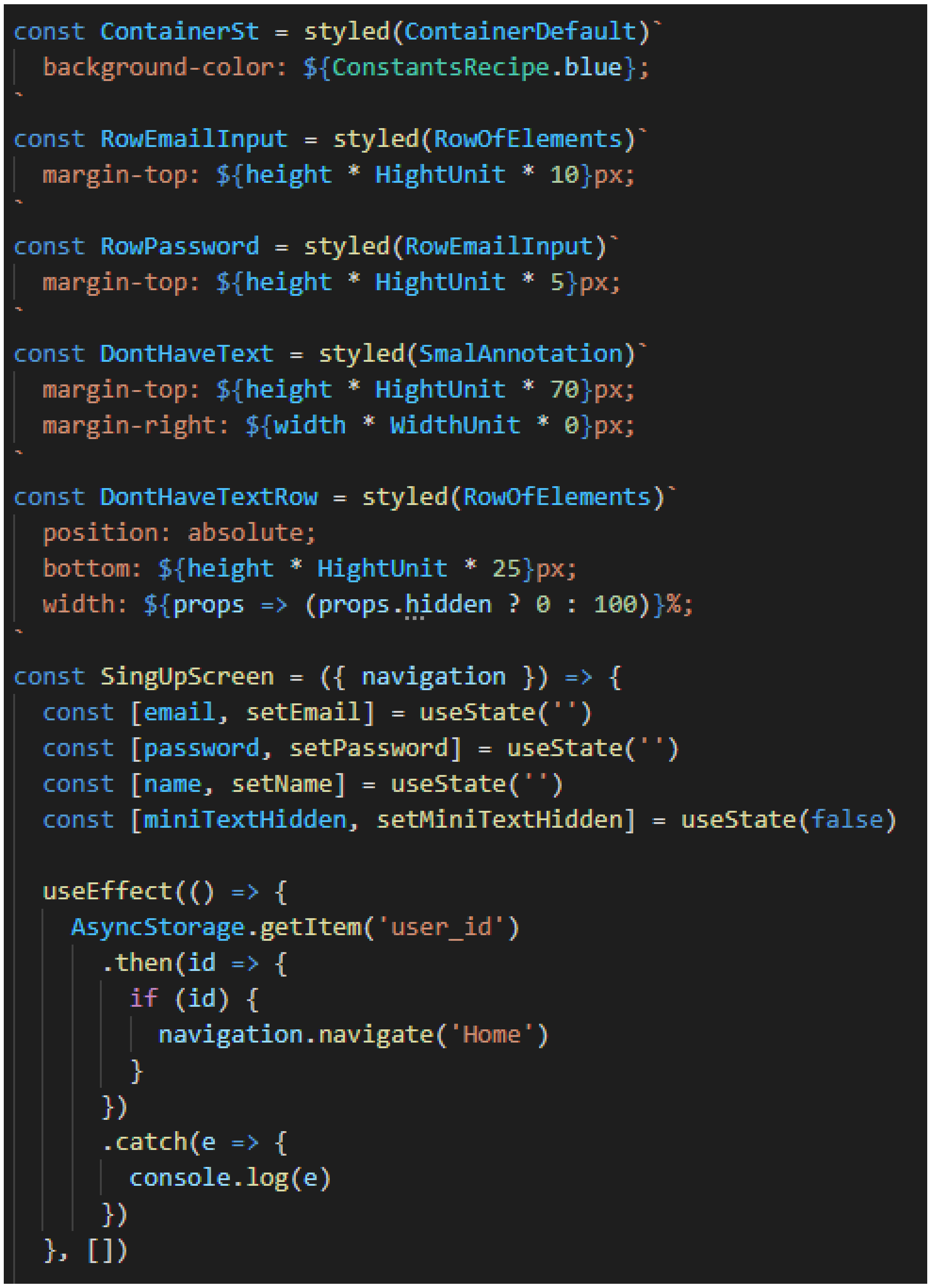
```
const LogInScreen = ({ navigation }) => {
  const [email, setEmail] = useState('')
  const [password, setPassword] = useState('')
  const [miniTextHidden, setMiniTextHidden] = useState(false)
  const [modalVisible, setModalVisible] = useState(false)

  useEffect(async () => {
    try {
      let id = await AsyncStorage.getItem('user_id')
      if (id) navigation.navigate('Home')
    } catch (error) {
      console.log(error)
    }
  }, [])

  const handleSingIn = async () => {
    FireBaseAuthSystem.appSignIn(email, password)
      .then(async res => {
        let user = await FetchApi.getUserByEmail(res)
        let user_id = user[0].id
        await AsyncStorage.setItem('user_id', `${user_id}`)
        navigation.navigate('Home')
      })
      .catch(e => {
        alert(e)
      })
  }
  const handleForgotPassword = async () => {
    try {
      await FireBaseAuthSystem.resetPassword(email)
      alert('Please check your email')
    } catch (error) {
      alert("We didn't find \nthis email in a system\nTry again")
    }
  }
```

Profile Screen Class



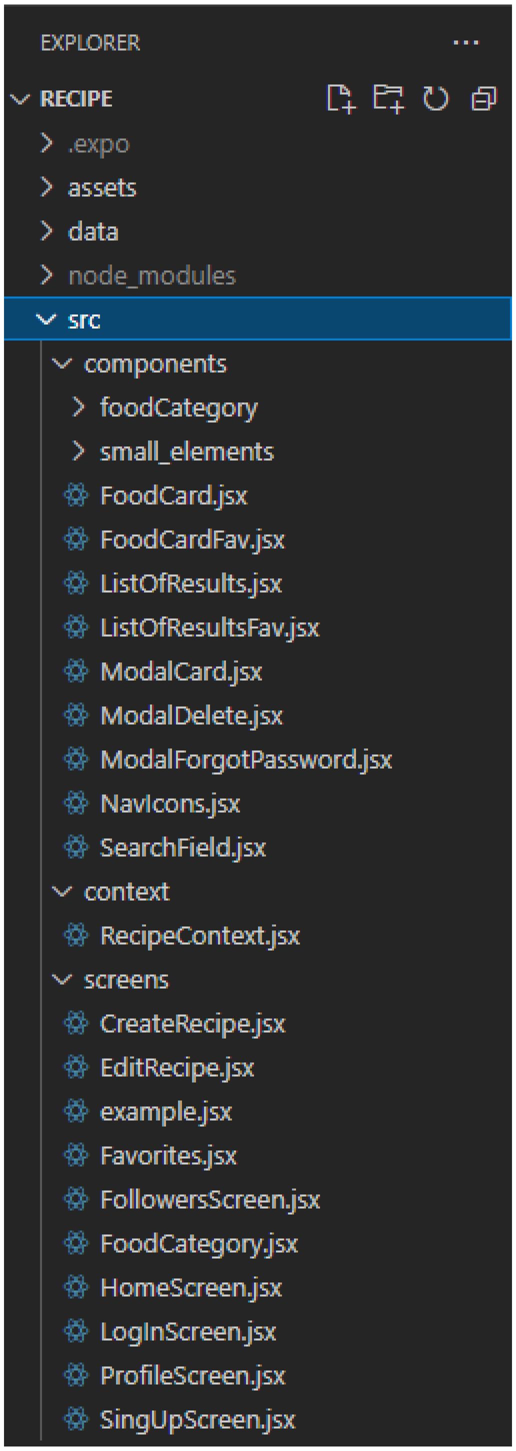
```
const ProfileScreen = ({ navigation }) => {
  // USECONTEXT
  const { modeUserRecipes, setModeUserRecipes } = useContext(RecipeContext)
  const { startUseEffectChainFav, setStartUseEffectChainFav } = useContext(
    RecipeContext
  )
  const { firstUseEffectDoneFav, setFirstUseEffectDoneFav } = useContext(
    RecipeContext
  )
  // LOCAL STATES:
  const [avatar, setAvatar] = useState(
    'https://www.baytekent.com/wp-content/uploads/2016/12/facebook-default-no-profile-pic1.jpg'
  )
  const [userState, setUserState] = useState(null)
  const [postNumber, setPostNumber] = useState(0)
  const [numOfPostsThatUserLikes, setNumOfPostsThatUserLikes] = useState(0)
  const [numOfFollowers, setNumOfFollowers] = useState(0)
  const [favScreenItems, setFavScreenItems] = useState([])

  useEffect(async () => {
    const unsubscribe = navigation.addListener('didFocus', async () => {
      setFirstUseEffectDoneFav(true)
      setStartUseEffectChainFav(false)
      let email = await AsyncStorage.getItem('email')
      let data = await FetchApi.getUserByEmail(email)
      let user = data[0]
      setUserState(user)
      let numOfPosts = await FetchApi.countPostsByUserId(user.id)
      setPostNumber(numOfPosts)
      let favs = await FetchApi.countFavsByUserId(user.id)
      setNumOfPostsThatUserLikes(favs)
      let userFollowers = await FetchApi.getFollowers(user.id)
      setFavScreenItems(userFollowers)
      setNumOfFollowers(userFollowers.length)
      if (user.avatar) setAvatar(user.avatar)
    })
    return unsubscribe
  }, [navigation])
```

Sign Up Screen Class



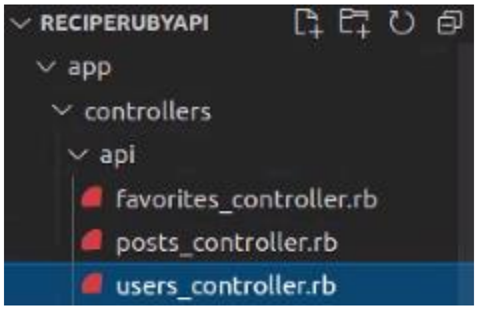
```
const ContainerSt = styled(ContainerDefault)`
  background-color: ${ConstantsRecipe.blue};
`

const RowEmailInput = styled(RowOfElements)`
  margin-top: ${height * HightUnit * 10}px;
`

const RowPassword = styled(RowEmailInput)`
  margin-top: ${height * HightUnit * 5}px;
`

const DontHaveText = styled(SmalAnnotation)`
  margin-top: ${height * HightUnit * 70}px;
  margin-right: ${width * WidthUnit * 0}px;
`

const DontHaveTextRow = styled(RowOfElements)`
  position: absolute;
  bottom: ${height * HightUnit * 25}px;
  width: ${props => (props.hidden ? 0 : 100)}%;
`

const SingUpScreen = ({ navigation }) => {
  const [email, setEmail] = useState('')
  const [password, setPassword] = useState('')
  const [name, setName] = useState('')
  const [miniTextHidden, setMiniTextHidden] = useState(false)

  useEffect(() => {
    AsyncStorage.getItem('user_id')
      .then(id => {
        if (id) {
          navigation.navigate('Home')
        }
      })
      .catch(e => {
        console.log(e)
      })
  }, [])
```

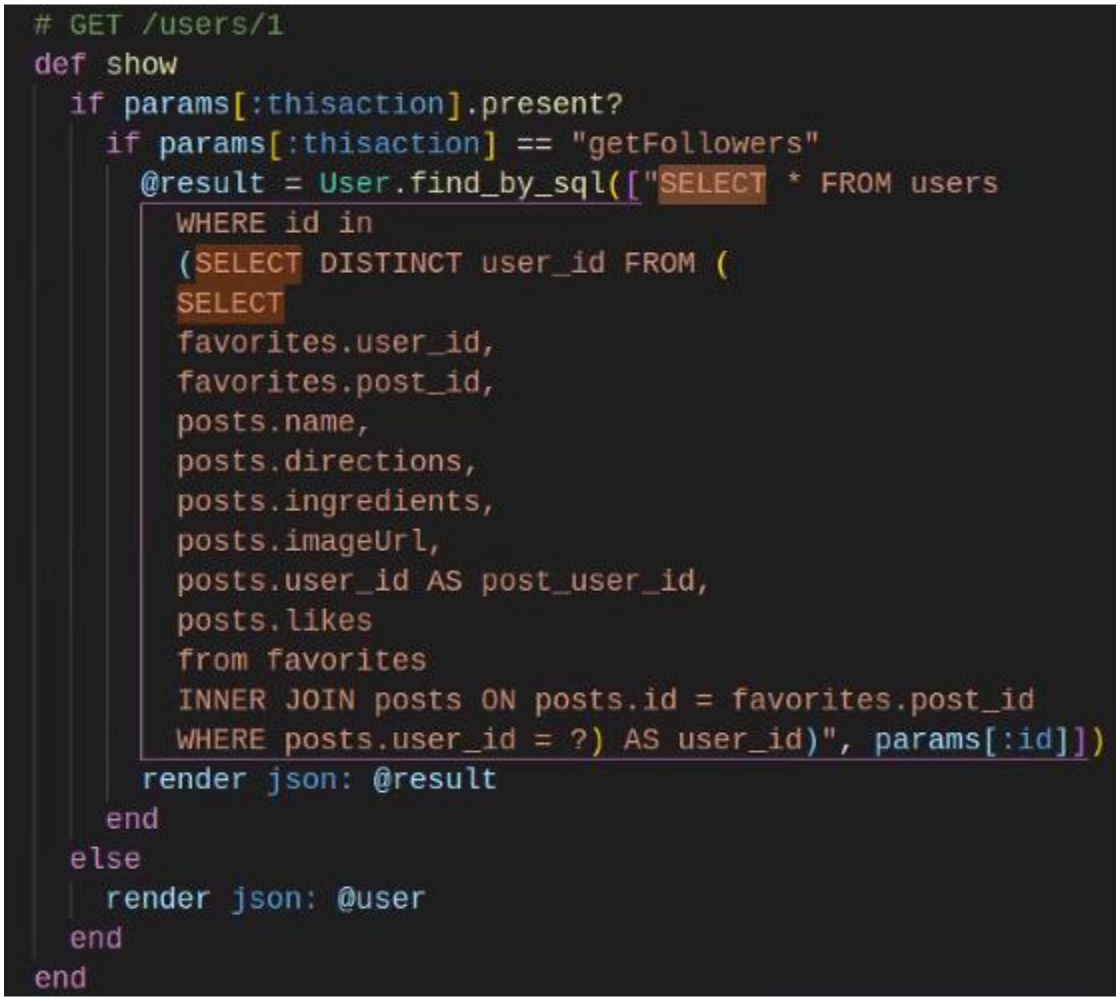
Overall Class Diagram



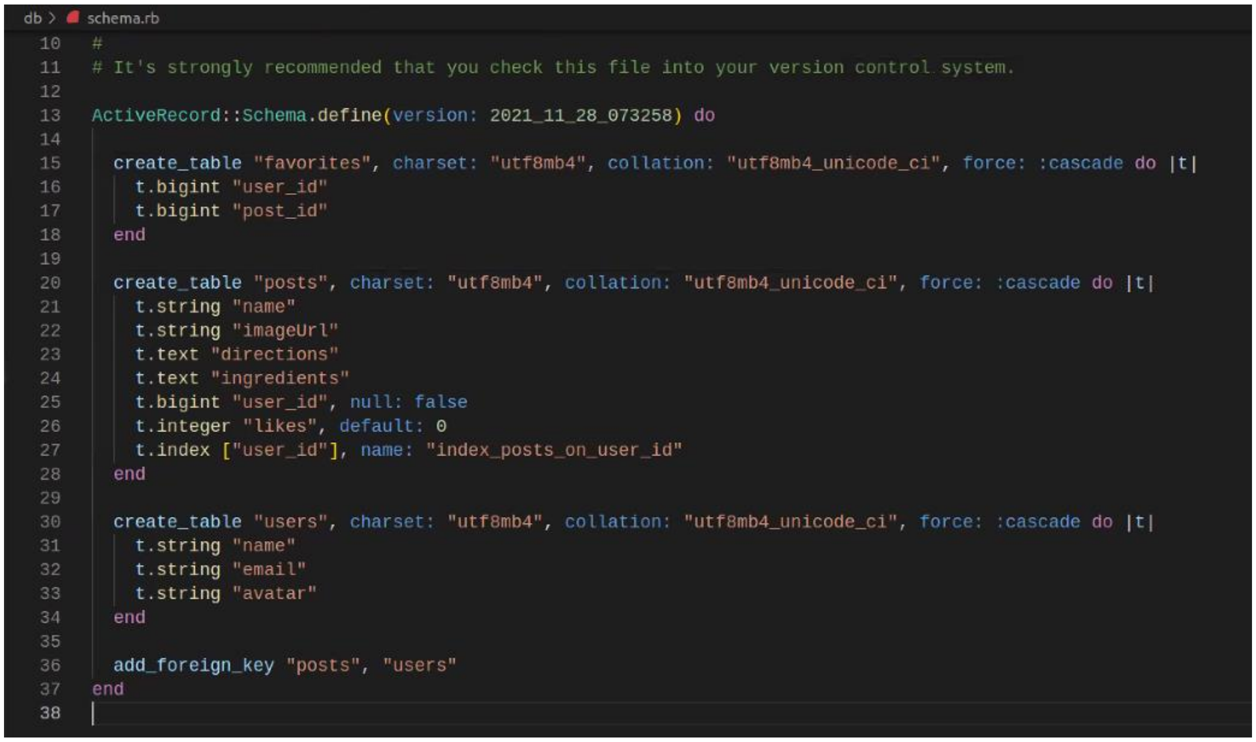
```
EXPLORER
RECIPE
  > .expo
  > assets
  > data
  > node_modules
  v src
    v components
      > foodCategory
      > small_elements
      FoodCard.jsx
      FoodCardFav.jsx
      ListOfResults.jsx
      ListOfResultsFav.jsx
      ModalCard.jsx
      ModalDelete.jsx
      ModalForgotPassword.jsx
      NavIcons.jsx
      SearchField.jsx
    v context
      RecipeContext.jsx
    v screens
      CreateRecipe.jsx
      EditRecipe.jsx
      example.jsx
      Favorites.jsx
      FollowersScreen.jsx
      FoodCategory.jsx
      HomeScreen.jsx
      LogInScreen.jsx
      ProfileScreen.jsx
      SingUpScreen.jsx
```

Database


RECIPERUBYAPI
app
controllers
api
favorites_controller.rb
posts_controller.rb
users_controller.rb



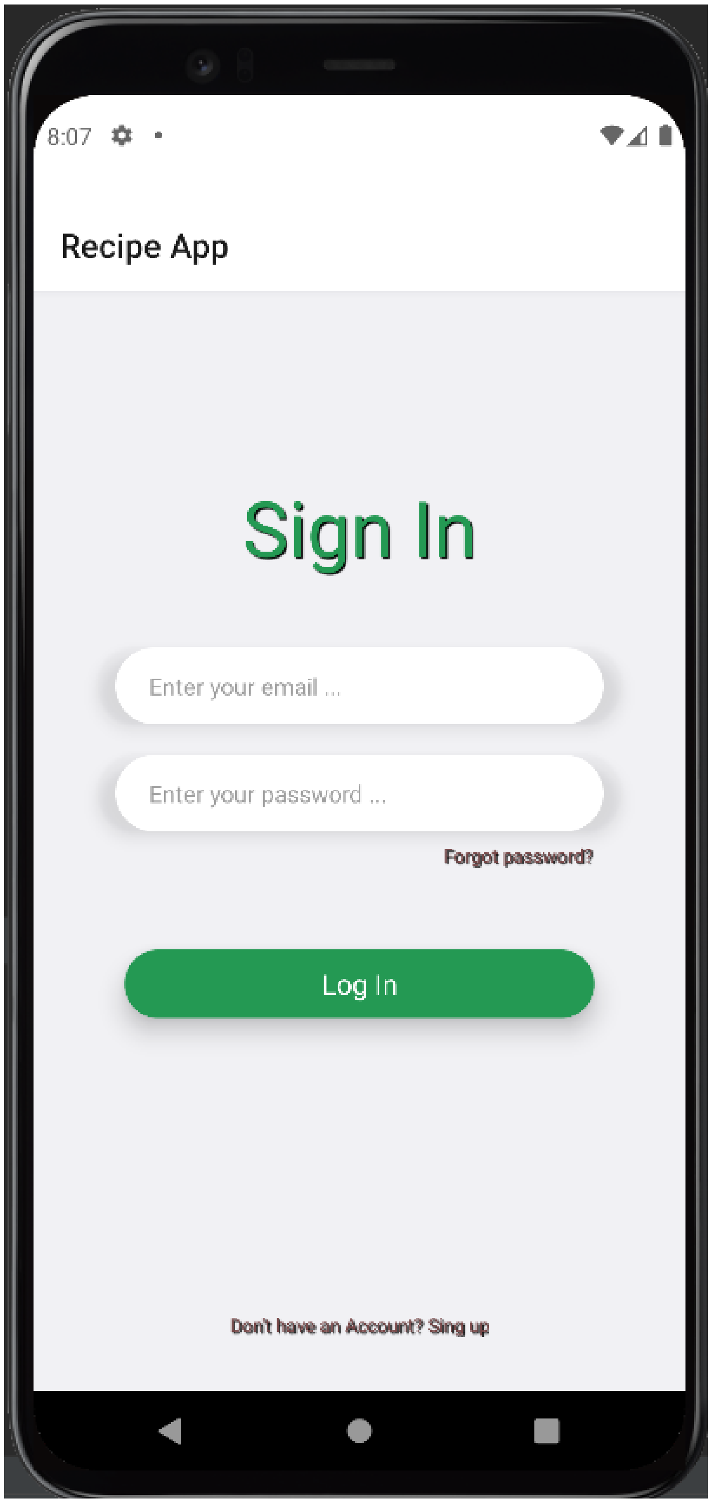
```ruby
# GET /users/1
def show
  if params[:thisaction].present?
    if params[:thisaction] == "getFollowers"
      @result = User.find_by_sql(["SELECT * FROM users
        WHERE id in
        (SELECT DISTINCT user_id FROM (
        SELECT
        favorites.user_id,
        favorites.post_id,
        posts.name,
        posts.directions,
        posts.ingredients,
        posts.imageUrl,
        posts.user_id AS post_user_id,
        posts.likes
        from favorites
        INNER JOIN posts ON posts.id = favorites.post_id
        WHERE posts.user_id = ?) AS user_id)", params[:id]])
      render json: @result
    end
  else
    render json: @user
  end
end
```

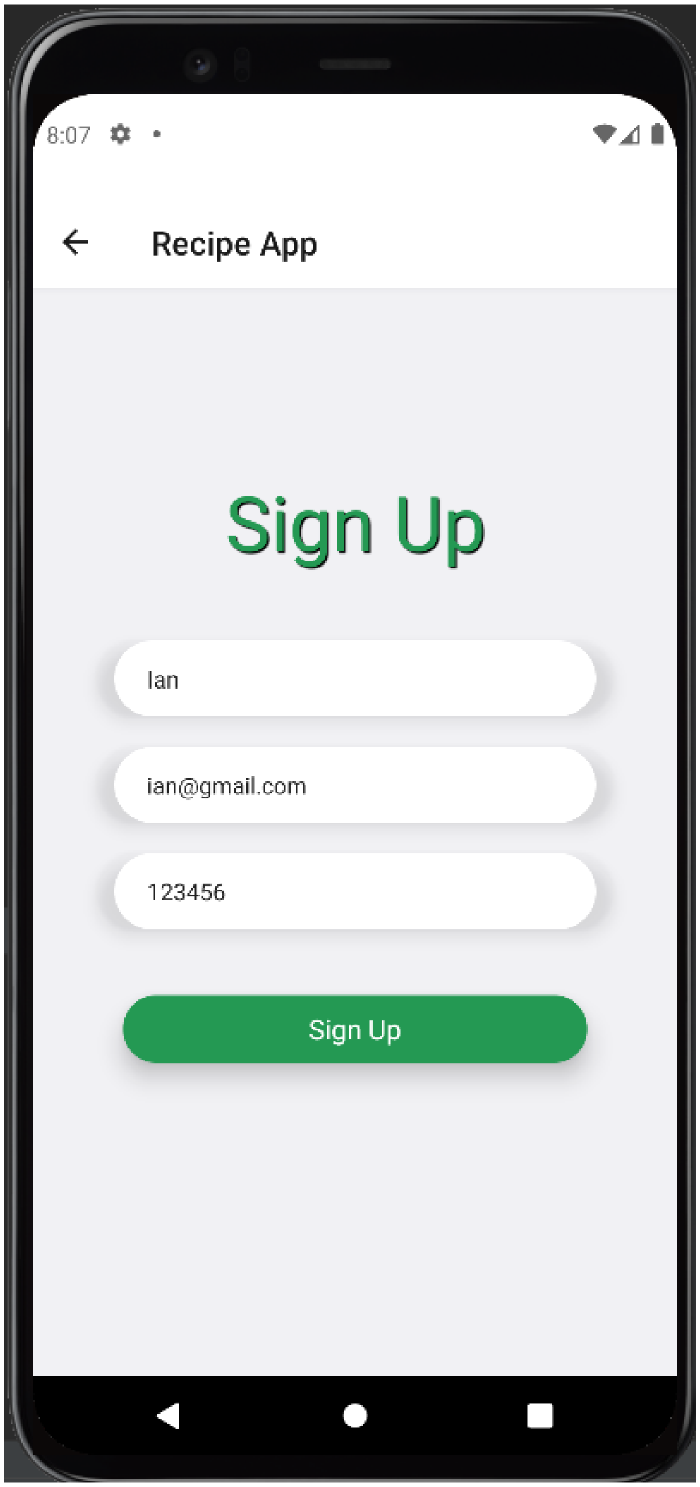


```ruby
#
# It's strongly recommended that you check this file into your version control system.

ActiveRecord::Schema.define(version: 2021_11_28_073258) do

  create_table "favorites", charset: "utf8mb4", collation: "utf8mb4_unicode_ci", force: :cascade do |t|
    t.bigint "user_id"
    t.bigint "post_id"
  end

  create_table "posts", charset: "utf8mb4", collation: "utf8mb4_unicode_ci", force: :cascade do |t|
    t.string "name"
    t.string "imageUrl"
    t.text "directions"
    t.text "ingredients"
    t.bigint "user_id", null: false
    t.integer "likes", default: 0
    t.index ["user_id"], name: "index_posts_on_user_id"
  end

  create_table "users", charset: "utf8mb4", collation: "utf8mb4_unicode_ci", force: :cascade do |t|
    t.string "name"
    t.string "email"
    t.string "avatar"
  end

  add_foreign_key "posts", "users"
end
```

Chapter 5: Test Results

Logging In



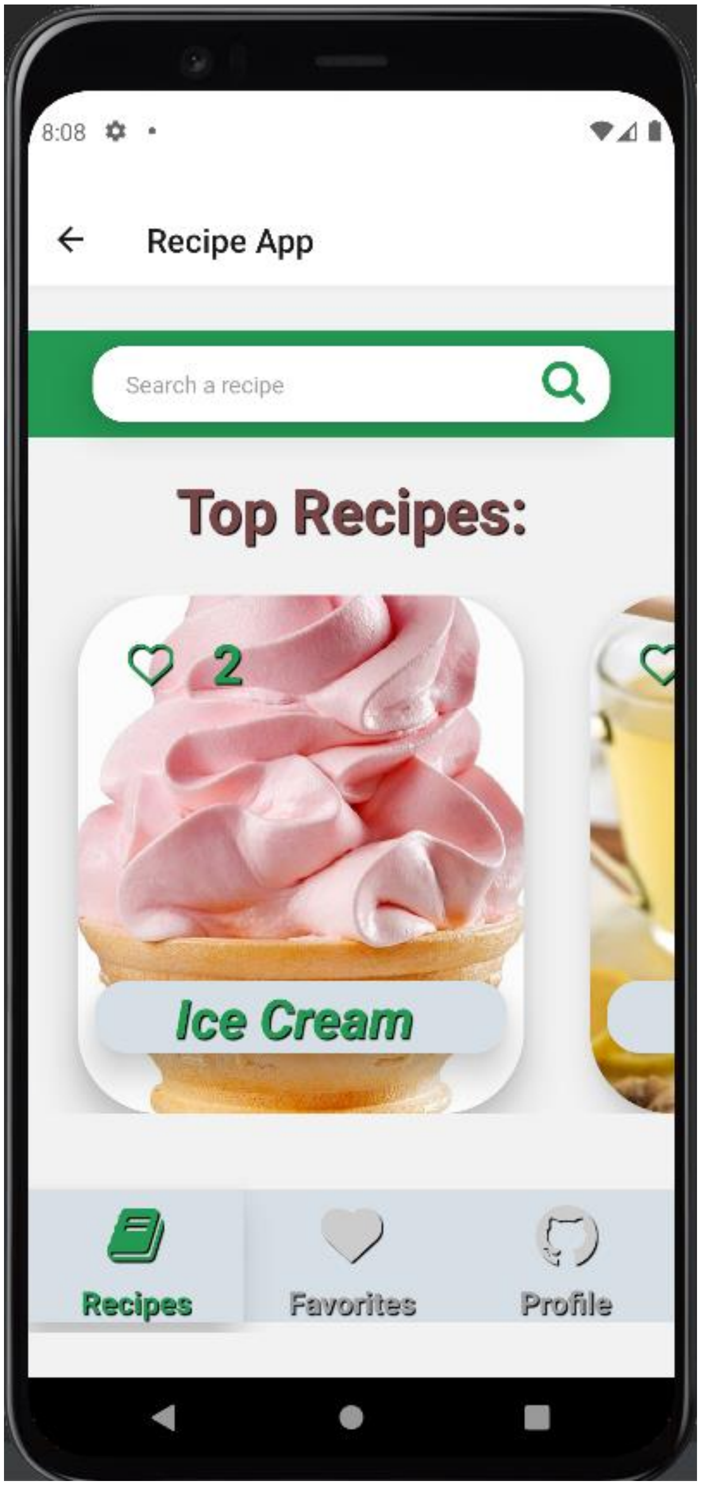


Ian

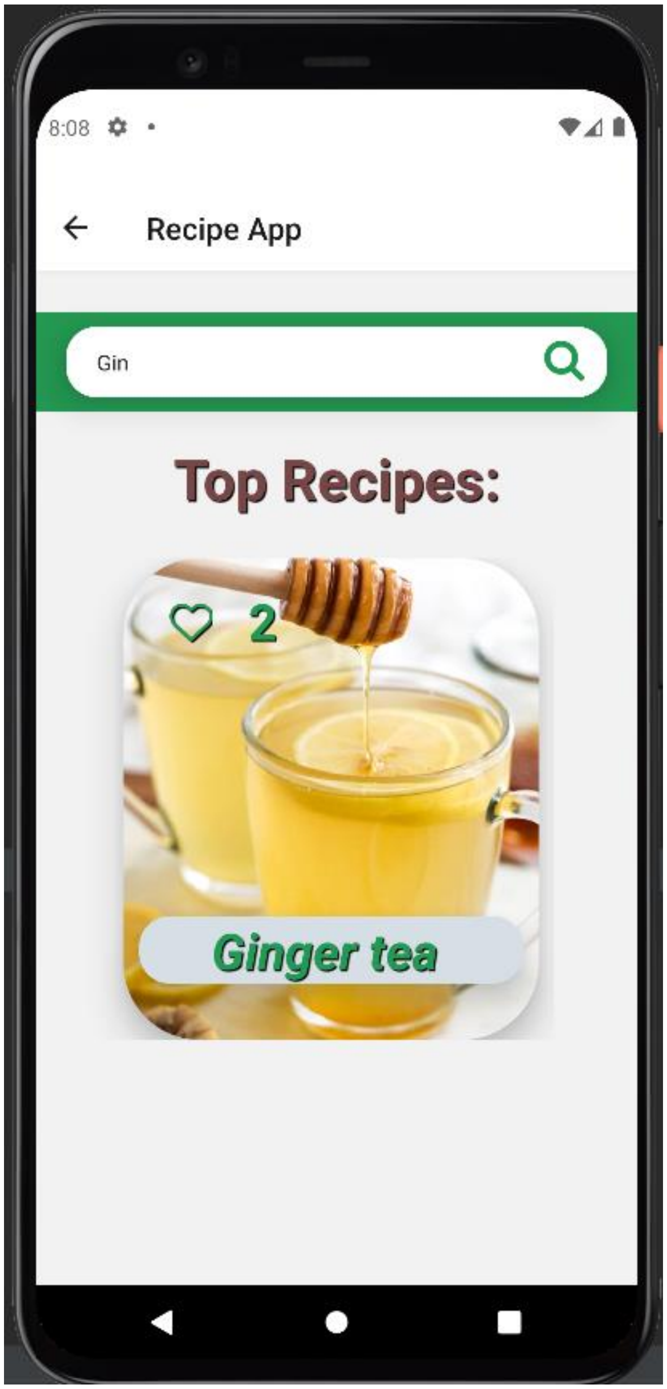
ian@gmail.com

123456

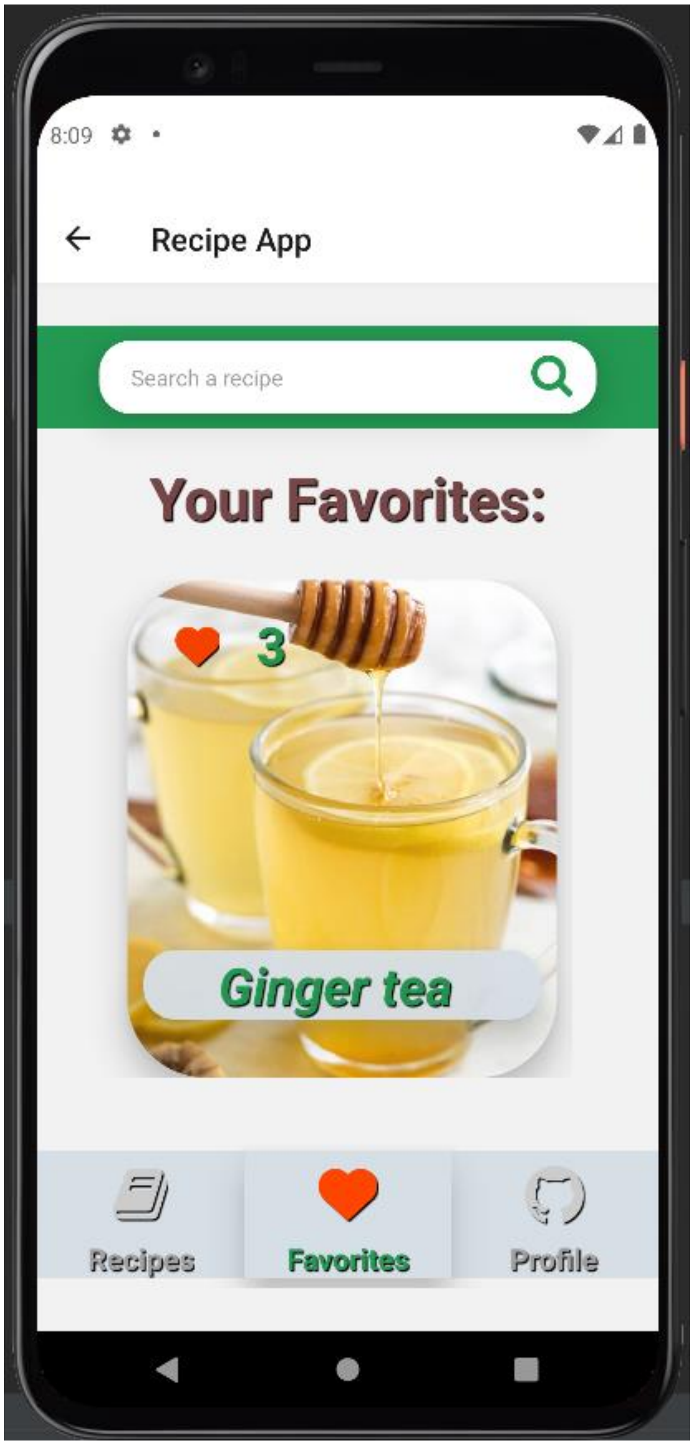
Home Screen


8:08
Recipe App
Search a recipe
Top Recipes:
2
Ice Cream
Recipes
Favorites
Profile

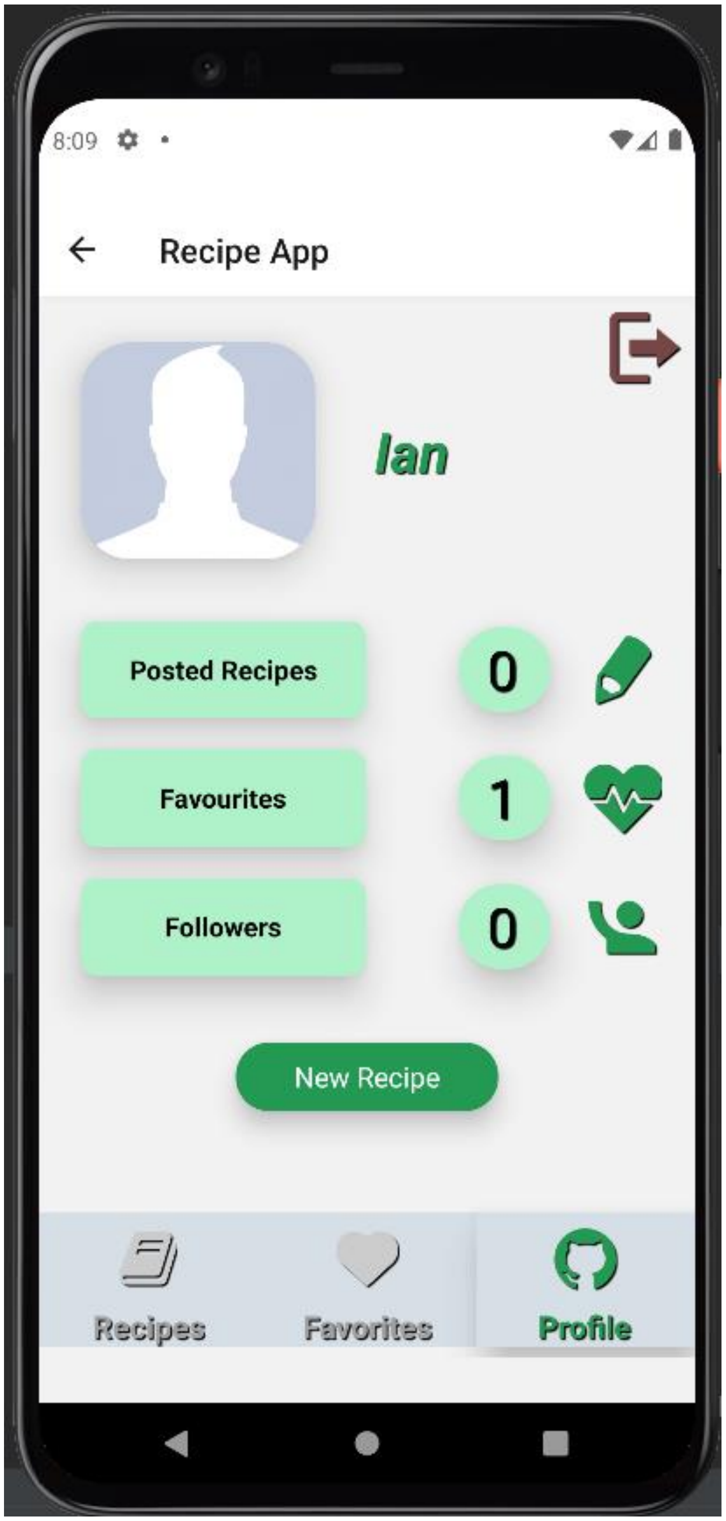
Search



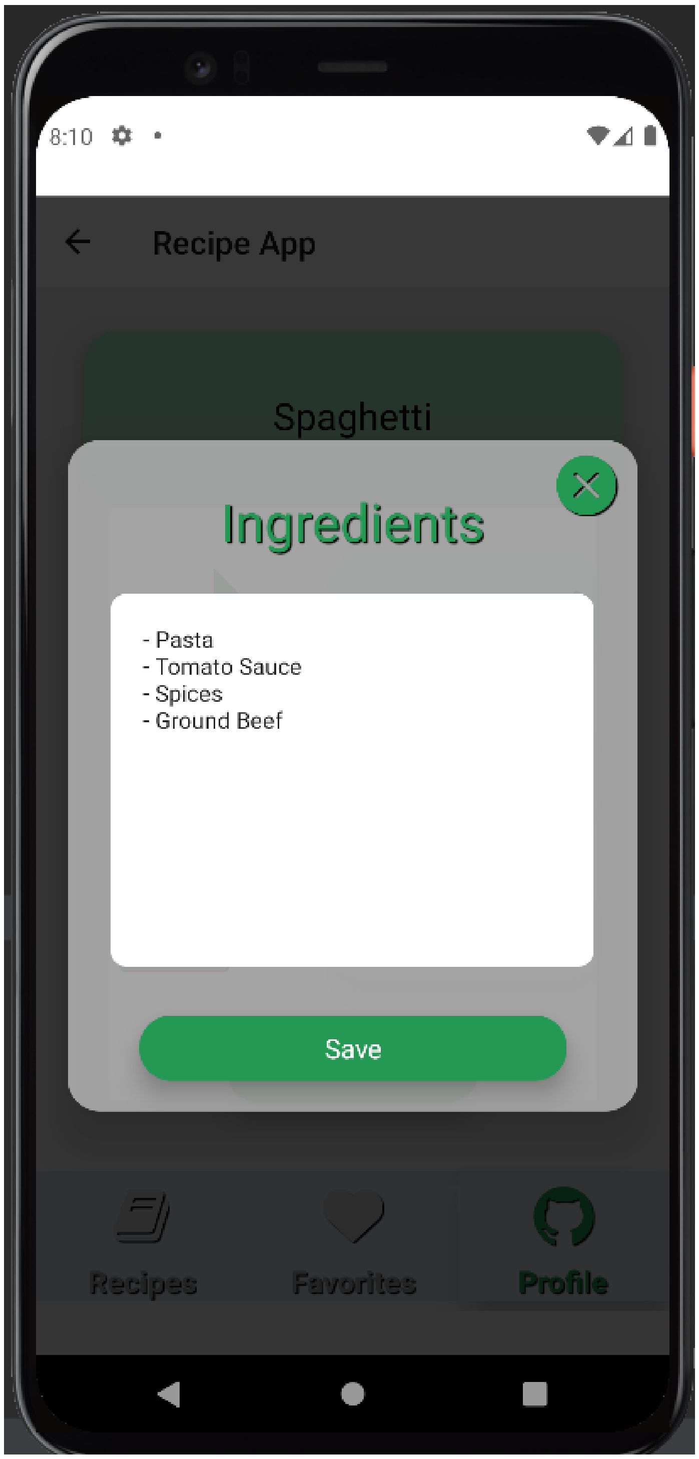
Favourites


8:09
Recipe App
Search a recipe
Your Favorites:
3
Ginger tea
Recipes
Favorites
Profile

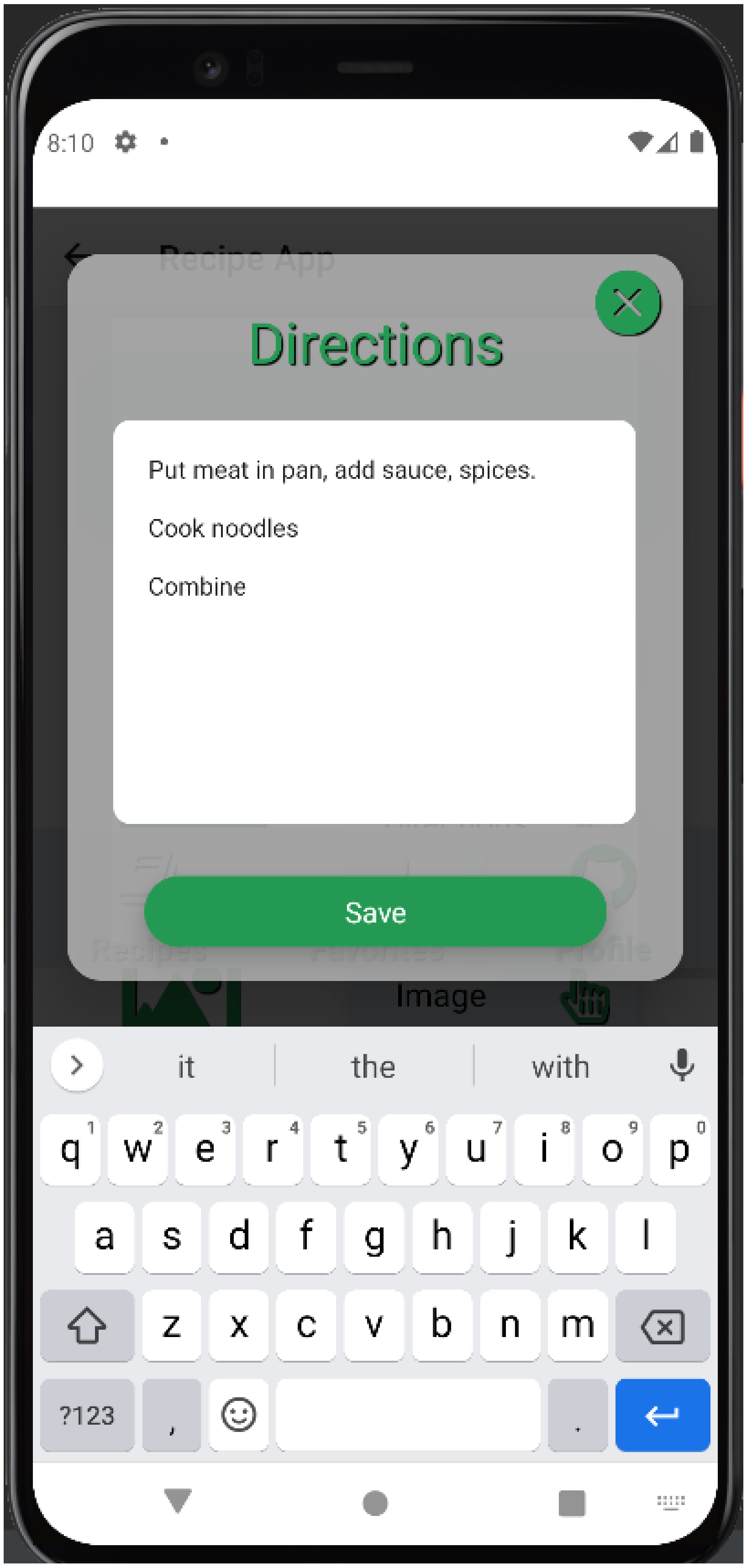
Profile


8:09
Recipe App
Ian
Posted Recipes
0
Favourites
1
Followers
0
New Recipe
Recipes
Favorites
Profile

Adding Recipe



- Pasta
- Tomato Sauce
- Spices
- Ground Beef

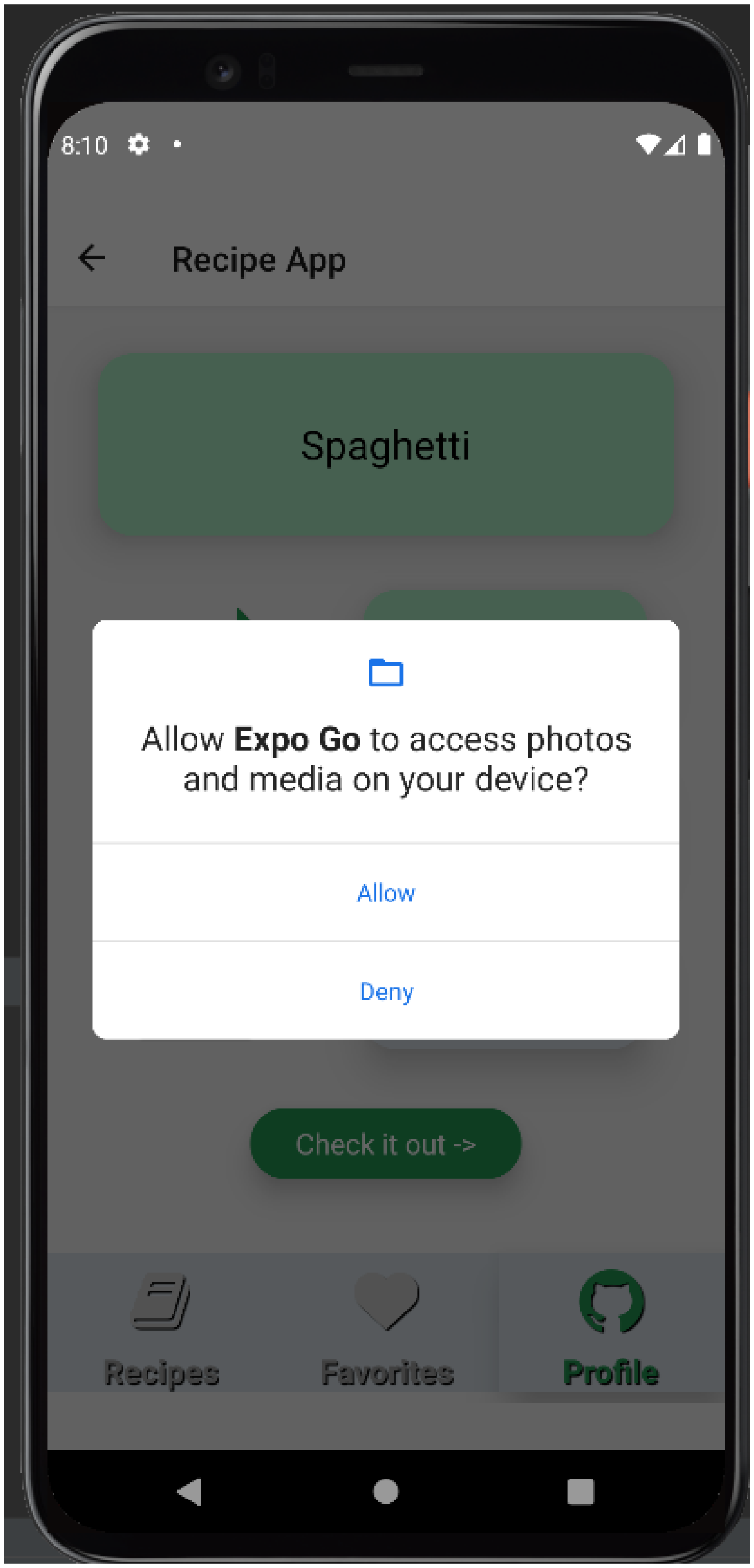


Put meat in pan, add sauce, spices.

Cook noodles

Combine

Permissions

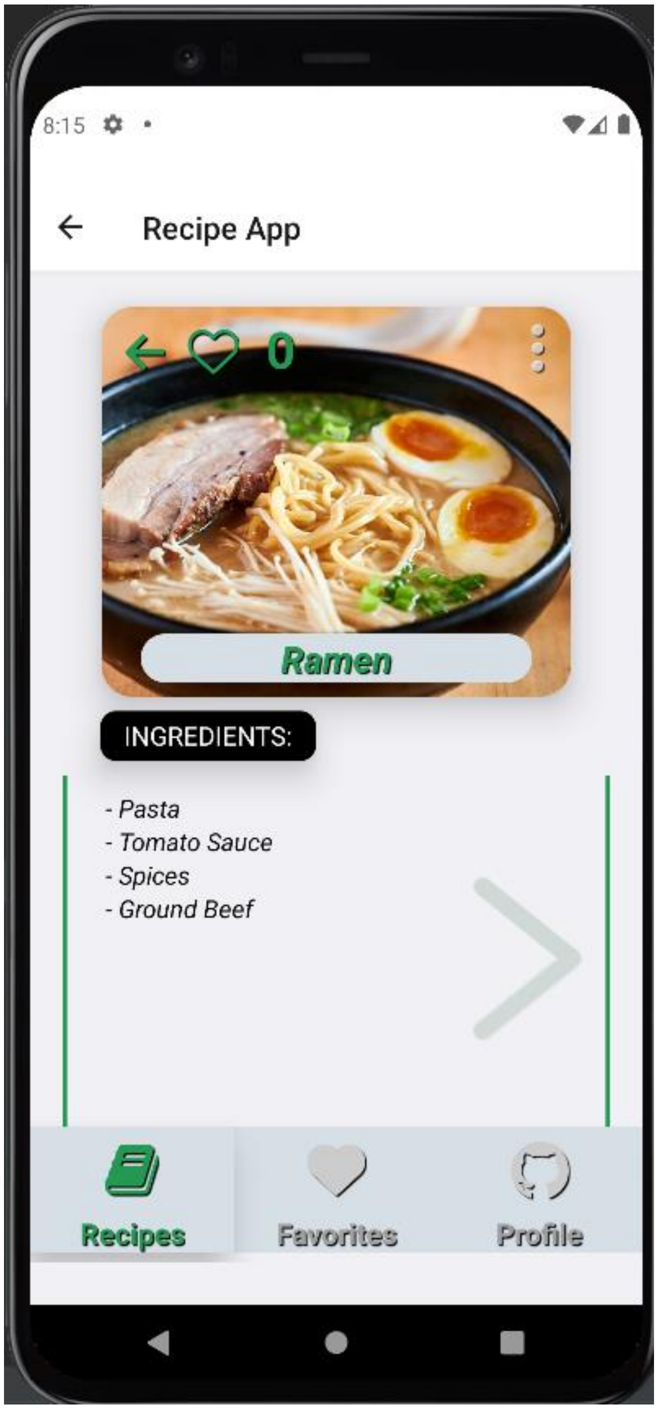


Allow **Expo Go** to access photos and media on your device?

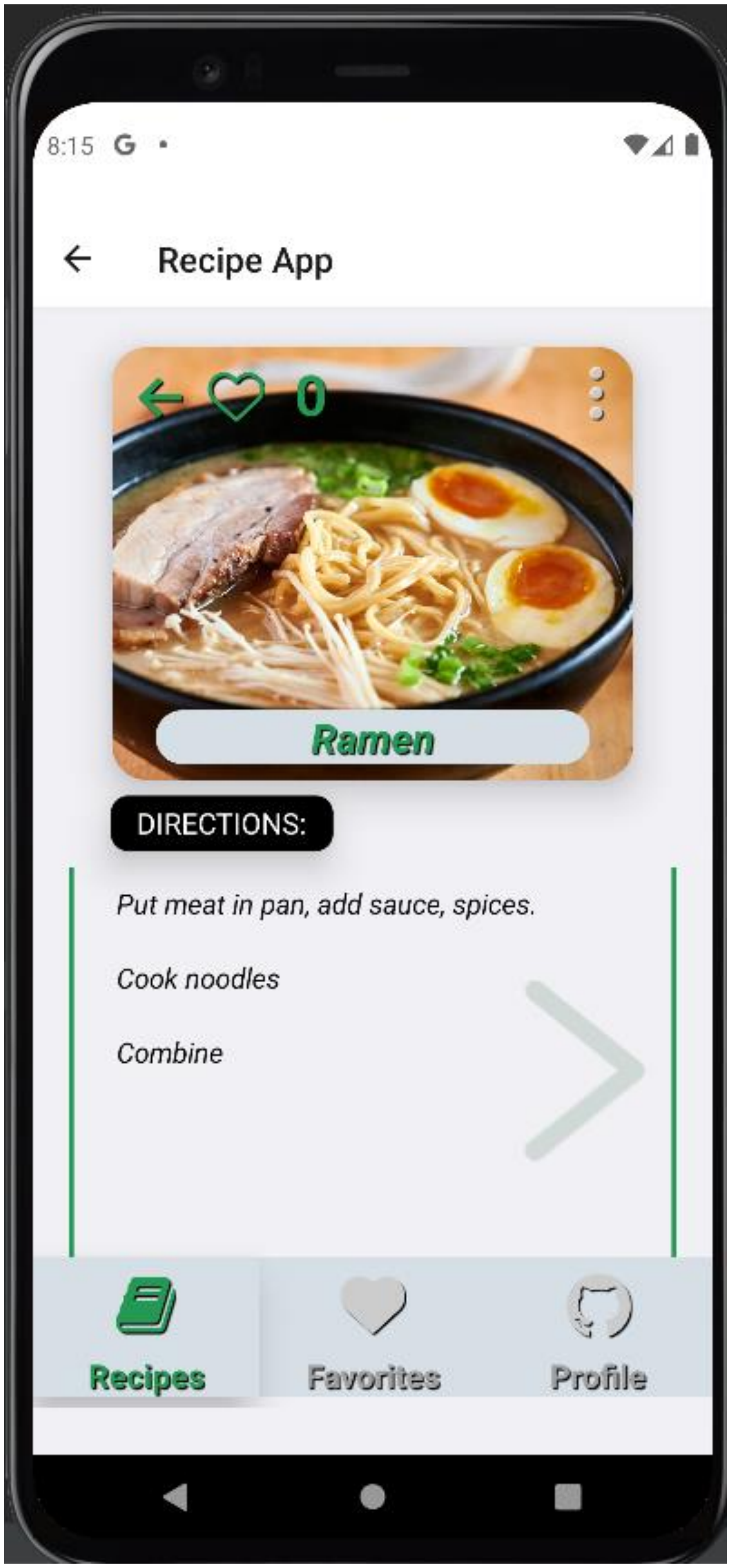
Allow

Deny

Displaying Recipe



- Pasta
- Tomato Sauce
- Spices
- Ground Beef

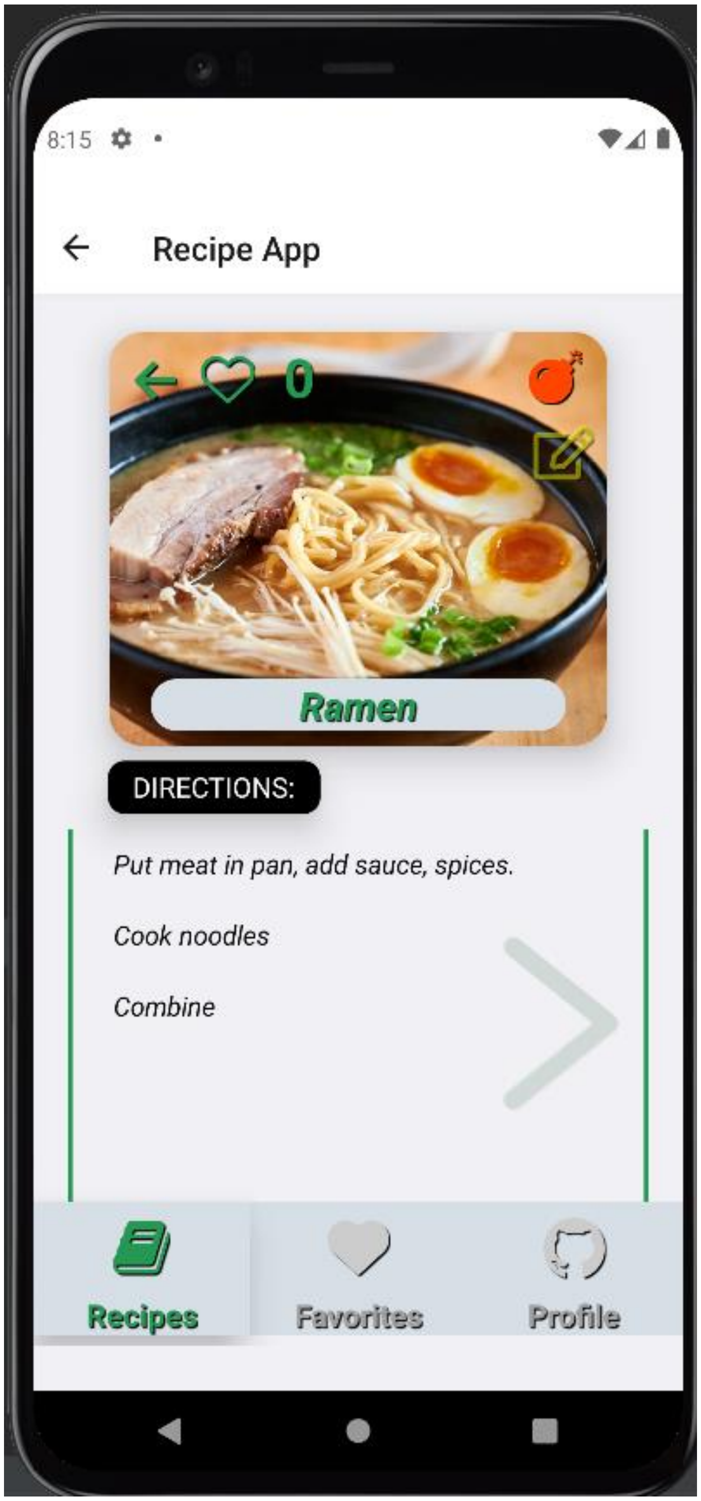


Put meat in pan, add sauce, spices.

Cook noodles

Combine

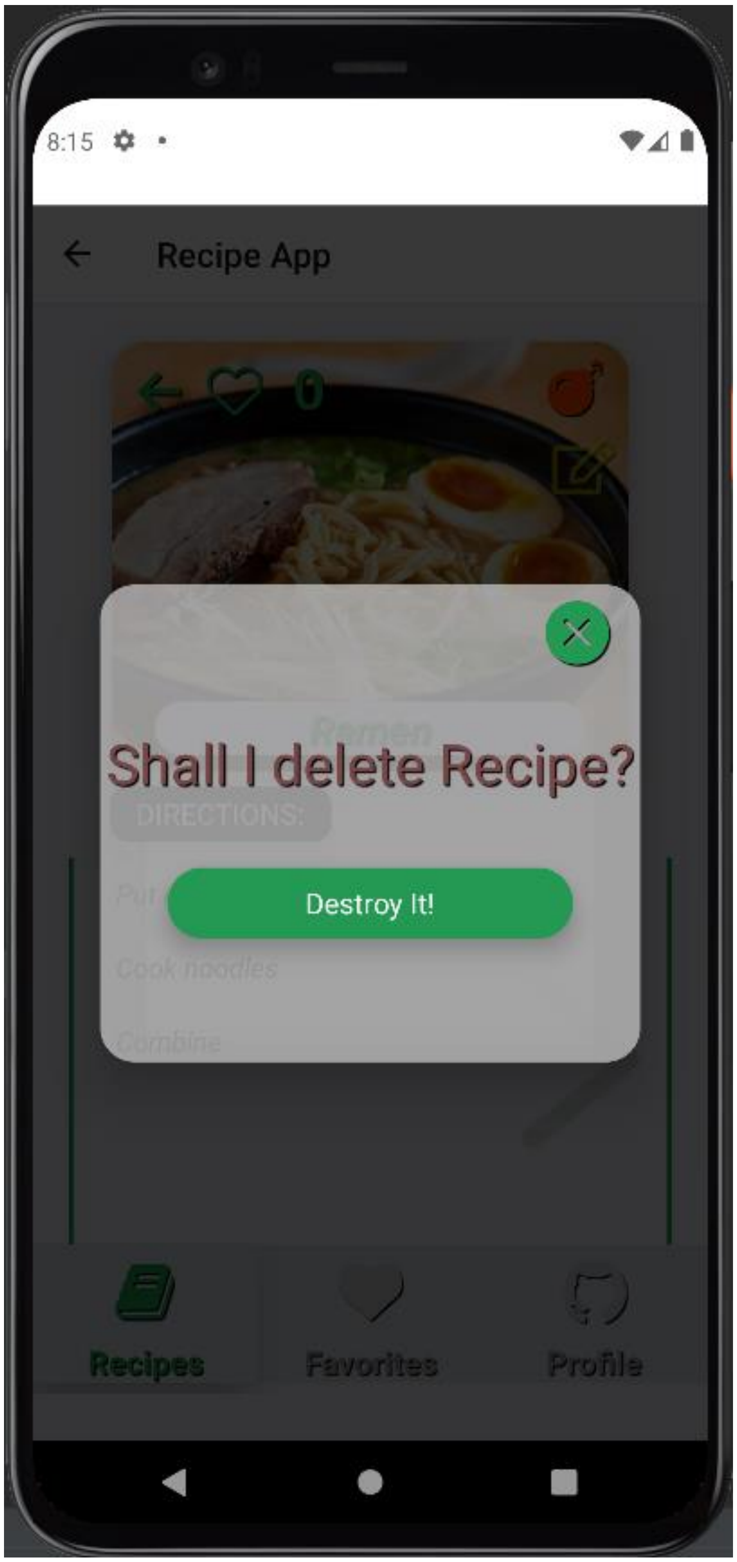
Deleting



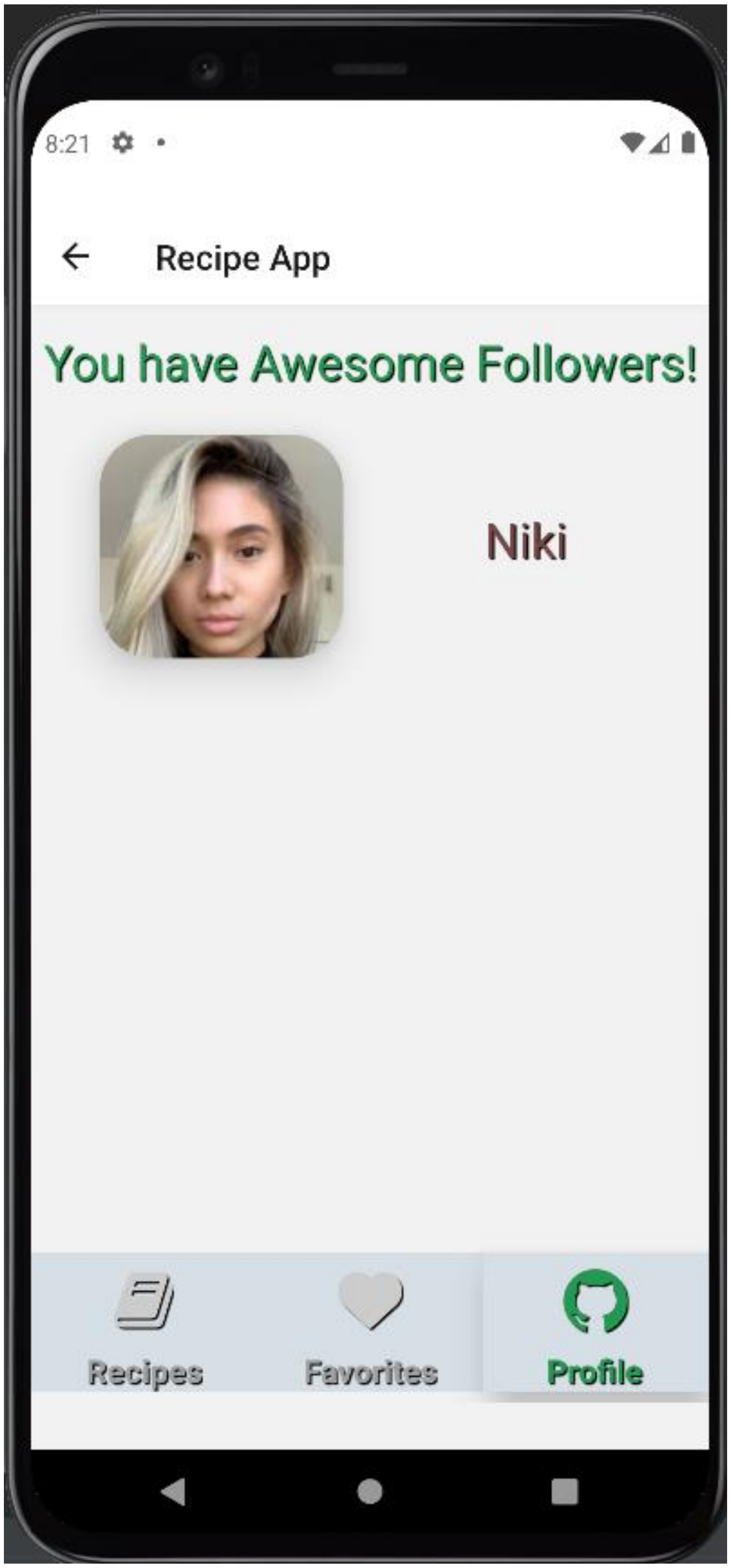
Put meat in pan, add sauce, spices.

Cook noodles

Combine


8:15
Recipe App
Ramen
Shall I delete Recipe?
DIRECTIONS:
Destroy It!
Put
Cook noodles
Combine
Recipes
Favorites
Profile

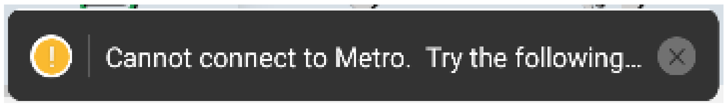
Followers



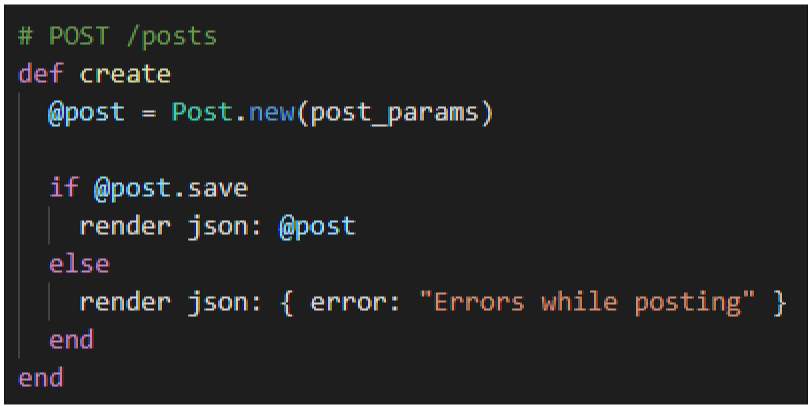
You have Awesome Followers!

Niki

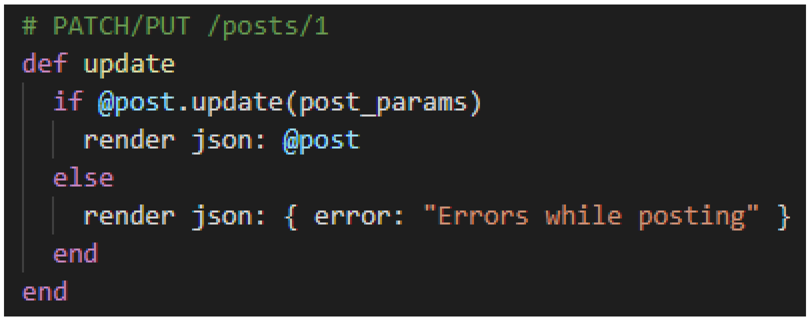
Error Handling and Error Messages



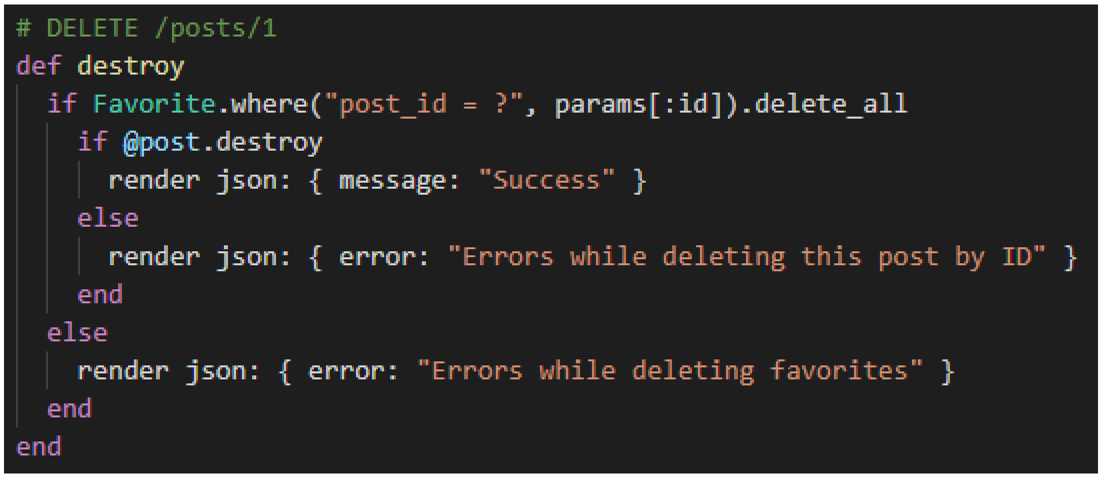
Cannot connect to Metro. Try the following...



```
# POST /posts
def create
  @post = Post.new(post_params)

  if @post.save
    render json: @post
  else
    render json: { error: "Errors while posting" }
  end
end
```



```
# PATCH/PUT /posts/1
def update
  if @post.update(post_params)
    render json: @post
  else
    render json: { error: "Errors while posting" }
  end
end
```



```
# DELETE /posts/1
def destroy
  if Favorite.where("post_id = ?", params[:id]).delete_all
    if @post.destroy
      render json: { message: "Success" }
    else
      render json: { error: "Errors while deleting this post by ID" }
    end
  else
    render json: { error: "Errors while deleting favorites" }
  end
end
```

Chapter 6: User Guide

Prerequisite Software

- Ensure phone software is up to date
- Network (Wi-Fi or cellular turned on)
- User accepts permissions

Chapter 7: Conclusion

Lessons Learned

Something we learned during this project was to communicate more often, even when we thought we had all tasks completed.

Another thing we learned during this project was to ensure all documentation were completed thoroughly as it would have helped during the writing of this final report as a lot of the information could be pulled from the original 3 documents. Creating this final report during preparation for finals week was quite overwhelming and next time we would start a lot earlier on this document as it takes a lot of time.

Existing Problems

At this time everything that we projected in our wireframes works to an ability that we are satisfied with. Nothing surprising came up during this project and we are very thankful for that.

Future Improvements

There are many improvements we could make to the current application; however, for a version 1 of the application we are satisfied with what we accomplished during this term. Things that could be modified in version 2 is a cleaner UI that has a modern twist to it. The version 1 we have now is more block oriented; however, in the future we would like clean up the navigation, as well as rounding out the edges and sharpening the text. Version 1 does work quickly and efficiently so the users that get in on version 1 will not be disappointed

References

- https://github.com/MK-314/2204_IT_Project
- https://github.com/MK-314/recipeRubyAPI

Appendix

- https://github.com/MK-314/2204_IT_Project
- https://github.com/MK-314/recipeRubyAPI