Cheap Eats

Software Requirements Specification

Version 1.0

6/11/2019

Benjamin Miller
Software Engineer
Christian Ford
Software Engineer
Scott Sheffer
Software Engineer
Nathan R. Hall
Software Engineer

Prepared for
Software Engineering

Revision History

Date	Description	Author	Comments
6/6/2019	Version 1	Benjamin Miller	First Revision
6/6/2019	Version 1	Christian Ford	Introduction, Purpose, etc.
6/11/2019	Version 2	BM. CF. SS. NH	Section 3 to Appendix A

Document Approval

The following Software Requirements Specification has been accepted and approved by the following:

Signature	Printed Name	Title	Date
Benjamin Miller	Benjamin Miller	Software Eng.	6/11/2019
Christian Ford	Christian Ford	Software Eng.	6/11/2019
Scott Sheffer	Scott Sheffer	Software Eng.	6/11/2019
Nathan R. Hall	Nathan R. Hall	Software Eng.	6/11/2019

Table of Contents

Revision History **2**

Document Approval **2**

1. Introduction **6**

1.1 Purpose 6

1.2 Scope 6

1.3 Definitions, Acronyms, and Abbreviations 6

2. General Description **7**

2.1 Product Perspective 7

2.2 Product Functions 7

2.3 Constraints, Assumptions and Dependencies 7

3. Specific Requirements **8**

3.1 External Interface Requirements 8

3.1.1 Software Interfaces *8*

3.2 Functional Requirements 8

3.2.1 User Account Functionality *8*

3.2.2 Search Functionality *8*

3.2.3 Google Login *8*

3.2.4 Set Search Preferences *8*

3.2.5 Set Notification Preferences *8*

3.2.6 Notification about Event in Your Area *8*

3.2.7 Notification from Food Preferences *8*

3.2.8 Notification from Favorites *9*

3.2.9 Blocking Hosts *9*

3.2.10 Blocking Food Types/Items *9*

3.2.11 Like a Post *9*

3.2.12 Favorite A Host *9*

3.2.13 Application Navigation *9*

3.2.14 Filter By Cuisine *9*

3.2.15 Filter By Distance *9*

3.2.16 Filter By Deal Quality *9*

3.2.17 Sort By Number of "Likes" *10*

3.2.18 Post Rating *10*

3.2.19 Post Reporting *10*

3.2.20 Create Post *10*
3.3 Use Cases 11
3.3.1 Register a New User 11
3.3.2 Login User *11*
3.3.3 Set Food Preferences *11*
3.3.4 Set Notification Preferences *11*
3.3.5 Set Favorites *11*
3.3.6 Search for Events *11*
3.3.7 Filter Results *11*
3.3.8 Browse Events *11*
3.3.9 Create a Post *11*
3.3.10 Edit a Post *11*
3.3.11 Delete a Post *11*
3.3.12 Rate a Post *11*
3.3.13 Like a Post *11*
3.3.14 Dislike a Post *11*
3.3.15 Flag a Post *11*
3.3.16 Get Directions *11*
3.3.17 Block a Host or Event *11*
3.4 Classes / Objects 11
3.4.1 Restaurant *11*
3.4.2 User *11*
3.4.3 Post *12*
3.4.4 PostDisplay *12*
3.4.5 PostSearcher *12*
3.4.6 DisplayManger *13*
3.4.7 PostManager *13*
3.4.8 NotificationManager *13*
3.4.9 MapManager *13*
3.4.10 UserManager *13*
3.5 Non-Functional Requirements 13
3.5.1 Performance *13*
3.5.2 Reliability *13*
3.5.3 Availability *14*
3.5.4 Security *14*
3.5.5 Maintainability *14*
3.5.6 Portability *14*

3.5.7 Usability *14*
3.5.8 Scalability *14*
3.5.9 Dynamic Refreshing *14*
3.5.10 Aesthetic Appeal *14*
3.6 Logical Database Requirements 14
3.7 Other Requirements 14

4. Analysis Models **15**
4.1 Use Case Diagrams 15
4.1.1 Browsing and Searching Use Case Diagram 15
4.1.2 Networking Use Case Diagram 15
4.2 Class Diagrams 16
4.2.1 Class Diagram 17
4.3 Sequence Diagrams 17
4.3.1 Authentication Sequence Diagram 18
4.3.2 Browse Events Sequence Diagram 19
4.4 Activity Diagrams 19
4.4.1 Authentication Activity Diagram *20*
4.4.2 Core Functionality Activity Diagram *20*
4.5 State-Transition Diagram 20
4.5.1 User State Diagram *22*

5. Change Management Process **22**

Appendix A: Table of Contents **22**

Approvals **28**

Use Case List **28**

User Account Functionality **29**
Feature Process Flow / Use Case Model 29
Use Case(s) 29

Browsing Functionality **31**
Feature Process Flow / Use Case Model 31
Use Case(s) 31

User Preferences **33**
Feature Process Flow / Use Case Model 33
Use Case(s) 33

Notifications **35**
Feature Process Flow / Use Case Model 35
Use Case(s) 35

Posting Functionality **36**
Feature Process Flow / Use Case Model 36
Use Case(s) 36

Directions **41**
Feature Process Flow / Use Case Model 41
Use Case(s) 41

1. Introduction

Cheap Eats gives an easy-to-read, easy-to-follow way for students to find the cheap/free food they never knew about.

1.1 Purpose

The purpose of the Cheap Eats Application is to offer an easy solution to both hosts and students; hosts are given an easy way to communicate to a larger base of people what events they are holding, and students are given an easy source of information about cheap or free food during their time on campus.

1.2 Scope

The product's name is Cheap Eats, it is a community driven Android application that allows hosts to advertise events that offer free or cheap food to students in their area. Giving students the knowledge and whereabouts of such events and hosts greater foot-traffic at said events. Students, who are often low on funds, will have greater ease in finding ways of spreading their dollars further by being able to attend events that are offering cheap or free food. Additionally, hosts will have a larger audience for whatever event they are putting on.

1.3 Definitions, Acronyms, and Abbreviations

Post: A post is created by a user, specifying a specific event and its location, time, host, and principal deal(s). Posts are displayed when a user is browsing, or searching for events.
Event: A special service provided by a local restaurant and/or by the university that offers food at a discounted price or for free.
Clout: The quantification of a Post's engagement, rating, number of Likes, and Flags. Clout determines how quickly a post will be seen by any given user when browsing or searching.
Sink: When the Post's Weight exceeds its Clout, causing the Post to require more browsing to find.
Float: When the Post's Clout exceeds its Weight, causing the Post to be higher in browsing

Weight: A calculated value that causes a post to sink in priority when being searched for.
Notification Preference: Options selected by the user that prioritize certain events more than others based on the events attributes, such as food type, proximity, etc., which causes a notification to appear on the user's phone home screen.
Flag: Value that is applied to Posts by Users when the Post is inaccurate, redundant, fraudulent, etc. Is a negative factor in the Post's Weight, causing it to Sink faster.
Browse, Browsing: Browsing is the functionality under which a list of posts is displayed to a user.
Favorites: Specific Hosts that a User wants to see most/every one of their Posts

2. General Description

2.1 Product Perspective

Cheap Eats somewhat resembles the functionality of GrubHub, a food delivery application that registers restaurants, allows users to search for food near them, and apply search functionality to the places listed.

Cheap Eats takes the premise of the base functionality of searching for local food and orients it towards the affordability problem space. The target user base of Cheap Eats is college students who seek affordable dining options, student organizations wishing to draw traffic by hosting food events, and campus adjacent restaurants that wish to advertise specials or deals.

Cheap Eats provides a unique platform for finding these food-hosting events, ameliorating a college students' need for affordable dining options.

2.2 Product Functions

The functions of the CheapEats application allow the users of the application to explore a vast amount of options within the application including getting notifications about events, users posting about events, and other users rating / liking / disliking those posts. Any user can filter these posts and events to their choosing.

2.3 Constraints, Assumptions and Dependencies

Google Firebase
Android API

3. Specific Requirements

3.1 External Interface Requirements

3.1.1 Software Interfaces

3.2 Functional Requirements

3.2.1 User Account Functionality

A User must be able to register as a new user using their Google account as their login. The user account will house all preferences that the user specifies; food preferences, favorite foods, notification preferences.

3.2.2 Search Functionality

The User will have the ability to search the application for any events they would like to attend. This search function will have different filters that the user can use to ensure they find what they are looking for. The user will be able to search by the type of food offered, the distance to the event, the price of the food being offered, the amount of food being offered, etc.

3.2.3 Google Login

A User may log into the system from the application's login screen by logging in to their google account.

3.2.4 Set Search Preferences

A User shall be able to specify their dietary preferences that shall restrict what event postings appear when searching or browsing

3.2.5 Set Notification Preferences

A User shall be able to set their notification preferences from their account settings page. These settings allow a user to receive a) notifications about all new posts, b) notifications of posts that only fit their food preferences, c) no notifications

3.2.6 Notification about Event in Your Area

A User will receive a notification, depending on their notification preferences, that informs them about new events that have been added within their area.

3.2.7 Notification from Food Preferences

A User will receive a notification, depending on their notification preferences, about deals that are offering items on their Food Preferences.

3.2.8 Notification from Favorites

A User will receive notifications for any event or offer that contains information on that user's Favorites.

3.2.9 Blocking Hosts

A User will be given the option to block specific hosts on their application. Any new events or deals offered by that host will not notify the user, regardless of proximity or food offered.

3.2.10 Blocking Food Types/Items

A User will be given the option to block specific foods on their application. Any event or deals offering that food will not notify the user, regardless of proximity or host.

3.2.11 Like a Post

While searching for posts, a user shall be able to "like" a post by pressing a like-button on a post. Doing so shall increase a posts' "clout"

3.2.12 Favorite A Host

A User shall be able to add the host destination of a post to their favorites list from the search page.

3.2.13 Application Navigation

User will be able to scroll through upcoming events, clicking on Events and Users they would like to engage with. Additionally, there will be menus that the User can click menus to change their preferences and settings.

3.2.14 Filter By Cuisine

A User is able to focus the scope of their searches based on the type of cuisine for which they are trying to search.

3.2.15 Filter By Distance

A User shall be able to filter the list of posts on a given search to only include events within a specified distance.

3.2.16 Filter By Deal Quality

A User can filter their food searches by the quality of the deal they are trying to find. Free Food, Cheap Food, and Cheapish Food are varying levels of deal quality by which a User would be able to search.

3.2.17 Sort By Number of "Likes"

A User will be able to sort upcoming events by the number of "Likes" that a Post has received from the community.

3.2.18 Post Rating

A Post will receive a rating based on its age, engagement, and any flags that have been levied against it.

3.2.19 Post Reporting

A User shall be able to report a post for inaccuracy of location, time, day, deal-details or for duplication by pressing a report button found on any given post.

3.2.20 Create Post

A User shall be able to create a new post by navigating to a create post tab, filling out fields for Title, Host, Location, deal/type of deal, start date/time, end date/time, and pressing a button that reads "publish".

3.3 Use Cases

3.3.1 Register a New User

3.3.2 Login User

3.3.3 Set Food Preferences

3.3.4 Set Notification Preferences

3.3.5 Set Favorites

3.3.6 Search for Events

3.3.7 Filter Results

3.3.8 Browse Events

3.3.9 Create a Post

3.3.10 Edit a Post

3.3.11 Delete a Post

3.3.12 Rate a Post

3.3.13 Like a Post

3.3.14 Dislike a Post

3.3.15 Flag a Post

3.3.16 Get Directions

3.3.17 Block a Host or Event

3.4 Classes / Objects

3.4.1 Restaurant

3.4.1.1 Attributes

- name
- address
- hours
- phone number
- rating

3.4.1.2 Methods

- editName()
- editAddress()
- editHours()
- editPhoneNum()

3.4.2 User

3.4.2.1 Attributes

- username
- email
- avatar
- foodPreferences
- notificationPreferences
- favoritesList

3.4.2.2 Methods

- makePost()
- setFavoriteFood()
- addFoodPreference()
- removeFoodPreference()

3.4.3 Post

3.4.3.1 Attributes

- title
- poster
- clout
- flagCount
- location
- foodType
- foodQuantity
- foodCost

3.4.3.2 Methods

- increaseClout()
- increaseFlags()

3.4.4 PostDisplay

3.4.4.1 Attributes

- commentCount
- distanceFromUser
- postTitle

3.4.4.2 Method

- printUser()
- printRestaurant()
- printCommentCount()

3.4.5 PostSearcher

3.4.5.1 Attributes

- filterList
- price

3.4.5.2 Method

- getPosts()
- applyFilter()
- calculateDistance()
- changePrice()
- listByClout()

3.4.6 DisplayManger

3.4.6.1 Attributes

3.4.6.2 Method

- renderProfile()
- renderBrowse()
- renderSettings()

3.4.7 PostManager

3.4.7.1 Attributes

- flagCount
- cloutCount
- weight

3.4.7.2 Method

- deletePost()
- calculateWeight()
- sink()
- float()

3.4.8 NotificationManager

3.4.8.1 Attributes

3.4.8.2 Method

- handleNewPost()

3.4.9 MapManager

3.4.9.1 Attributes

3.4.9.2 Method

- sendNavigation()

3.4.10 UserManager

3.4.10.1 Attributes

- username
- password

3.4.10.2 Method

- addNewUser()
- deleteUser()
- recoverPassword()
- changePassword()
- authenticateUser()
- firebaseOAuth()

3.5 Non-Functional Requirements

3.5.1 Performance

Immediate-response searching for Users. The initial launch will allow for up to 1000 concurrent Users without affecting the application's performance.

3.5.2 Reliability

Users who have a working Internet connection will be able to connect to the Cheap Eats Database and browse the application. The initial launch will allow for up to 1000 concurrent Users without affecting the application's reliability.

3.5.3 Availability

The Cheap Eats Application will be available to any User who has and Android device and the ability to connect to the Internet.

3.5.4 Security

Leverage Google Firestore's security features to protect all User personal information.

3.5.5 Maintainability

Google Firebase, and Android Studio are two tools that will allow for continued work and improvement of the Cheap Eats Application throughout its lifetime.

3.5.6 Portability

Any Android device with an Internet connection will be able to run the Cheap Eats Application. Future prospects may allow for the porting of the application to Apple IOS as well.

3.5.7 Usability

An intuitive scrolling screen will be implemented, allowing User a natural understanding of how to browse the application for upcoming deals in their area.

3.5.8 Scalability

The Cheap Eats application will initially launch with the University of Pittsburgh Campus, surrounding area, and students as the target location and audience, respectively. The application will have the possibility of being expanded (scaled) to encompass larger areas and more university campuses.

3.5.9 Dynamic Refreshing

A "weighting" system will be put in place to allow popular upcoming event to remain high on User's browsing, while flagged and unpopular events fall lower in browsing results. Additionally, expired events will be removed from the application, removing clutter from the User Experience.

3.5.10 Aesthetic Appeal

The Cheap Eats Application will display an appealing-to-look-at browsing display that will be easily readable on various standard screen sizes without error.

3.6 Logical Database Requirements

Google FireStore will be used as part of the Firebase backend to provide responsive, low-latency for application data. FireStore has the capability required to reach the 1000 concurrent user mark set for the application's initial launch, and is scalable enough to support a large user base should the application find popularity.

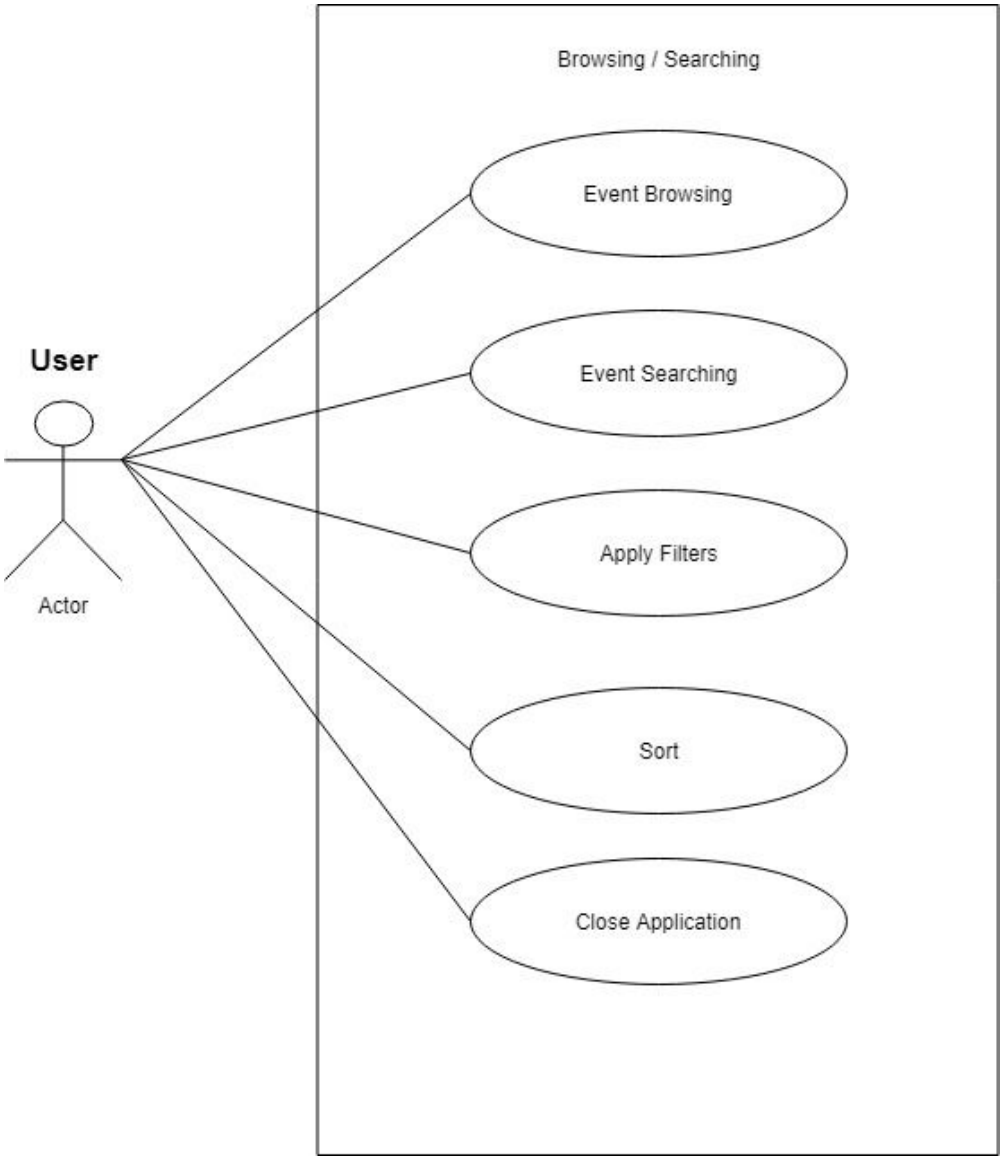
3.7 Other Requirements

4. Analysis Models

4.1 Use Case Diagrams

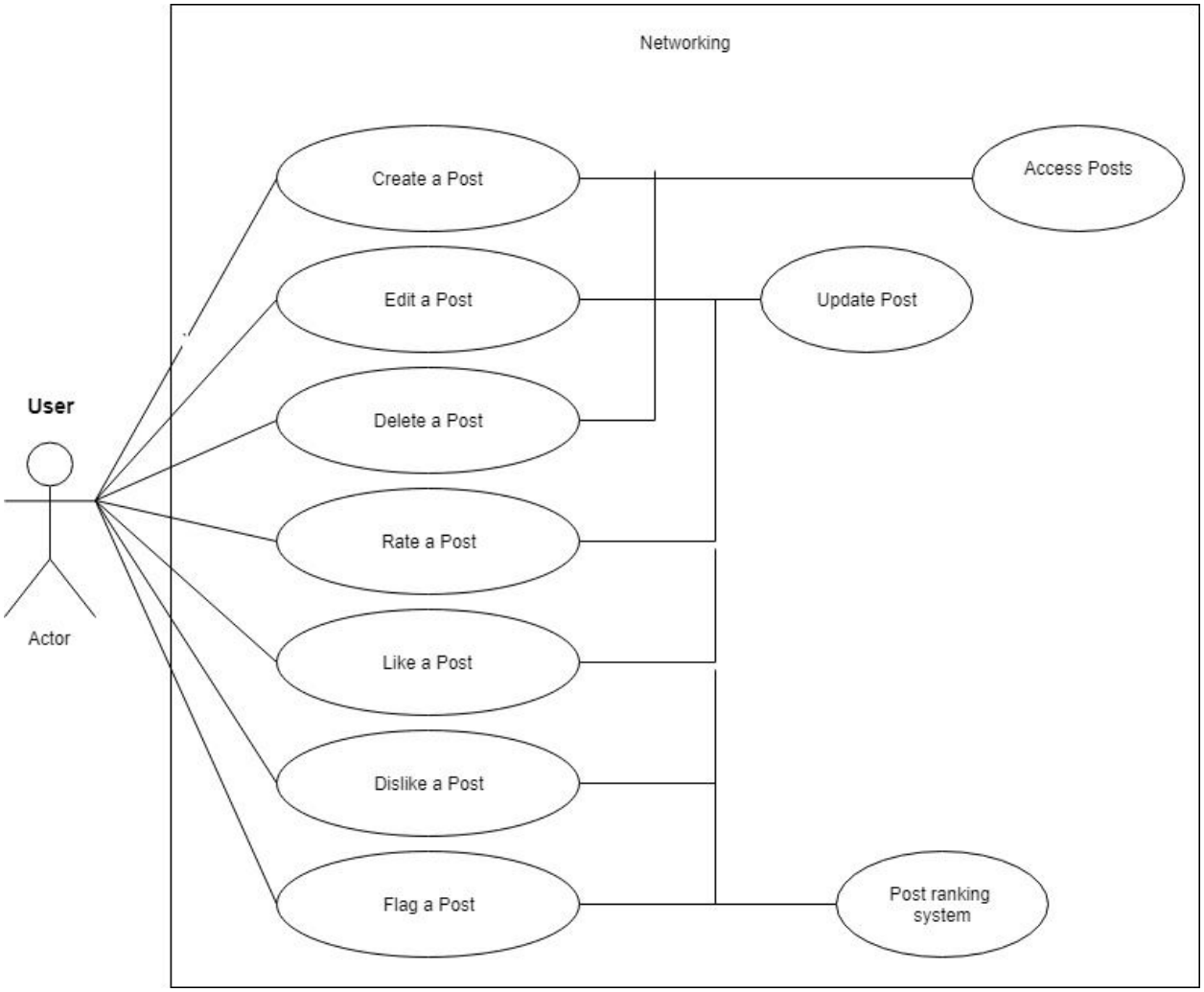
4.1.1 Browsing and Searching Use Case Diagram

This use case diagram depicts a single user as an actor, working in the browsing / searching boxes. The main use cases that the user acts on is Event Browsing, Event Searching, Apply Filters, Sort, Close Application



4.1.2 Networking Use Case Diagram

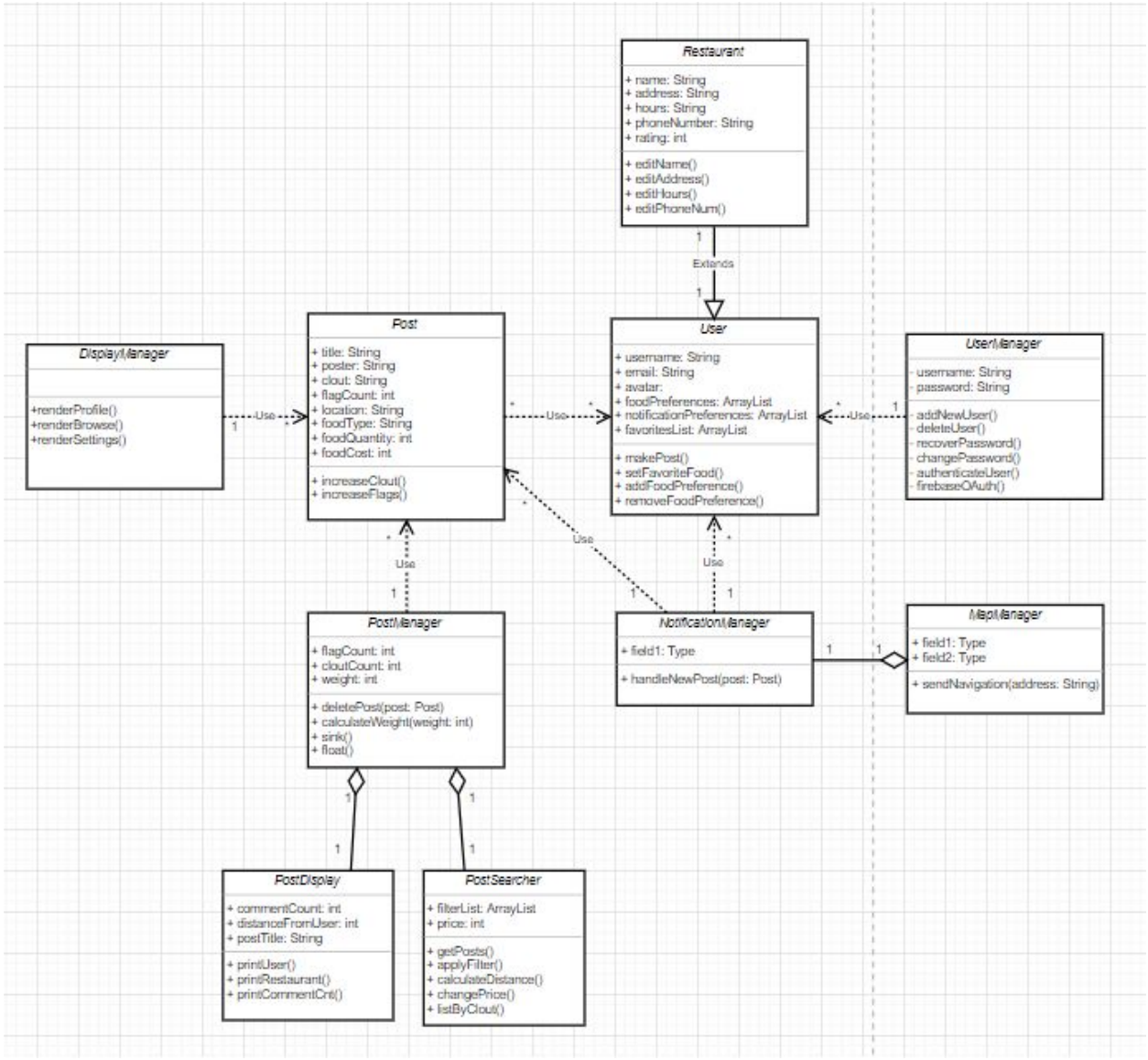
This use case diagram depicts a single user as an actor, working in the main networking box. The main use cases that the actor uses are Create / Edit / Delete / Rate / Like / Dislike / Flag a post. Create, Edit, and Delete a post all work with the Update post and Access post subsystems.



4.2 Class Diagrams

4.2.1 Class Diagram

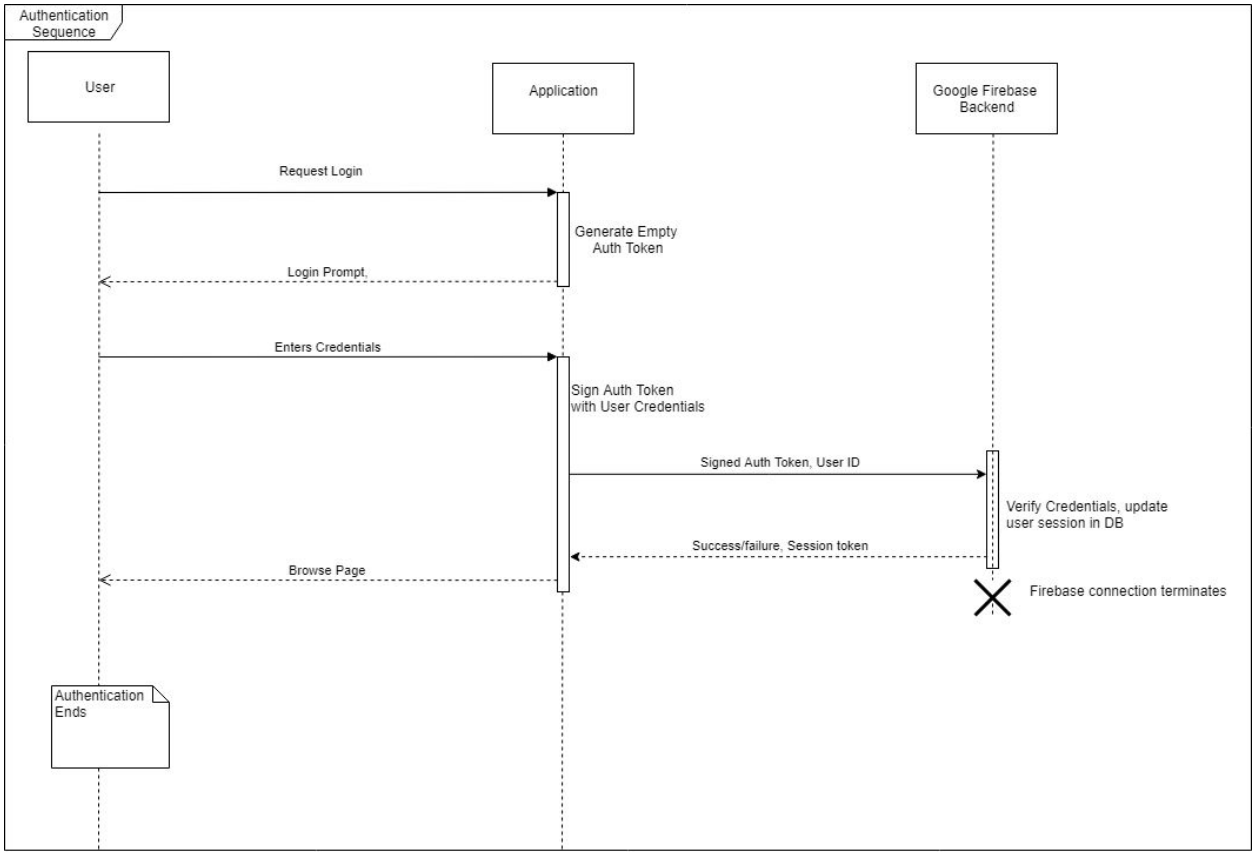
The following diagram is a representation of the structure of the Cheap Eats system including each class' corresponding attributes and methods.



4.3 Sequence Diagrams

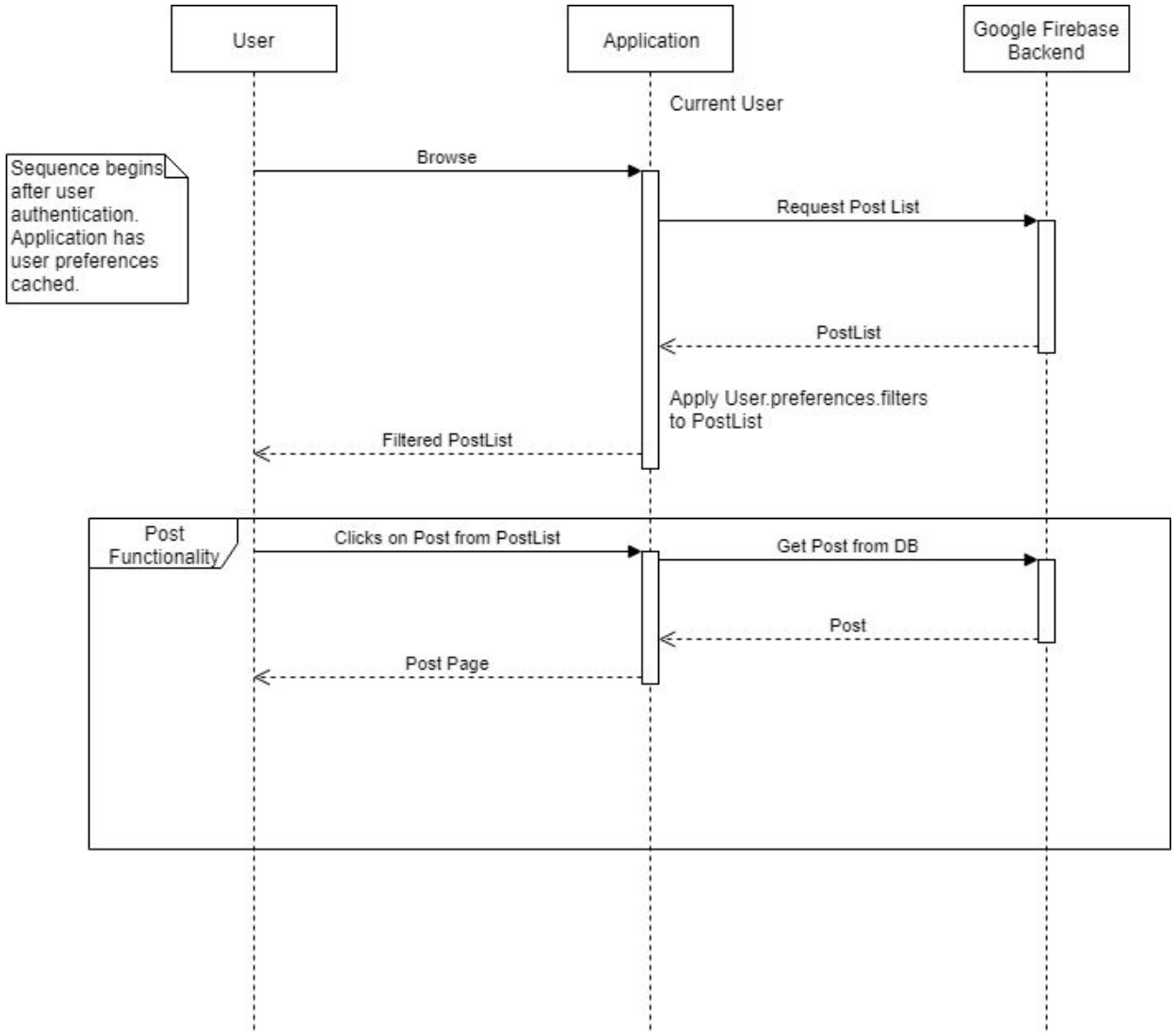
4.3.1 Authentication Sequence Diagram

The authentication sequence requires three major communication switches across our stack. The communication occurs between the user, our application, and Google Firebase which has built in functionality for authentication between the user and the database system.



4.3.2 Browse Events Sequence Diagram

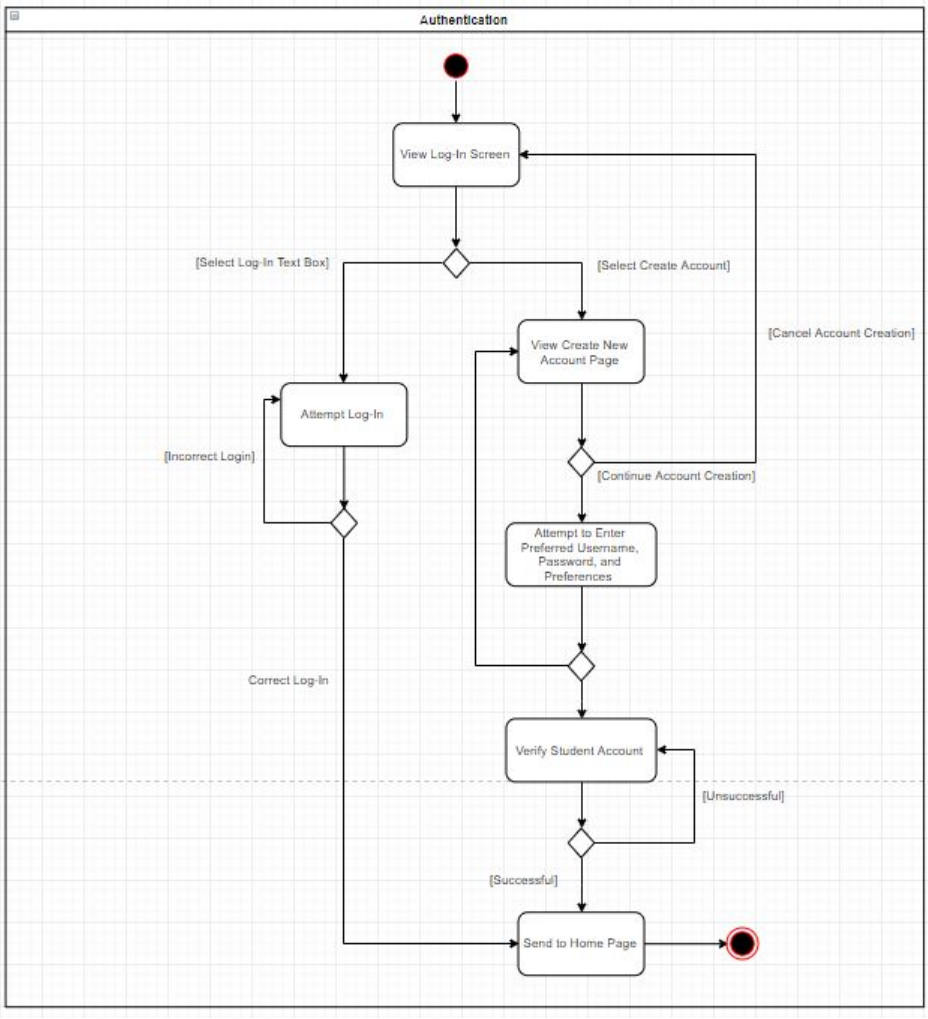
The sequences that may occur when browsing an event is separated between two paths, the natural ordering of posts and the filter system which includes food preferences, and other user specific information. The rest of the sequence involves relaying post data across the application and database back to the user.



4.4 Activity Diagrams

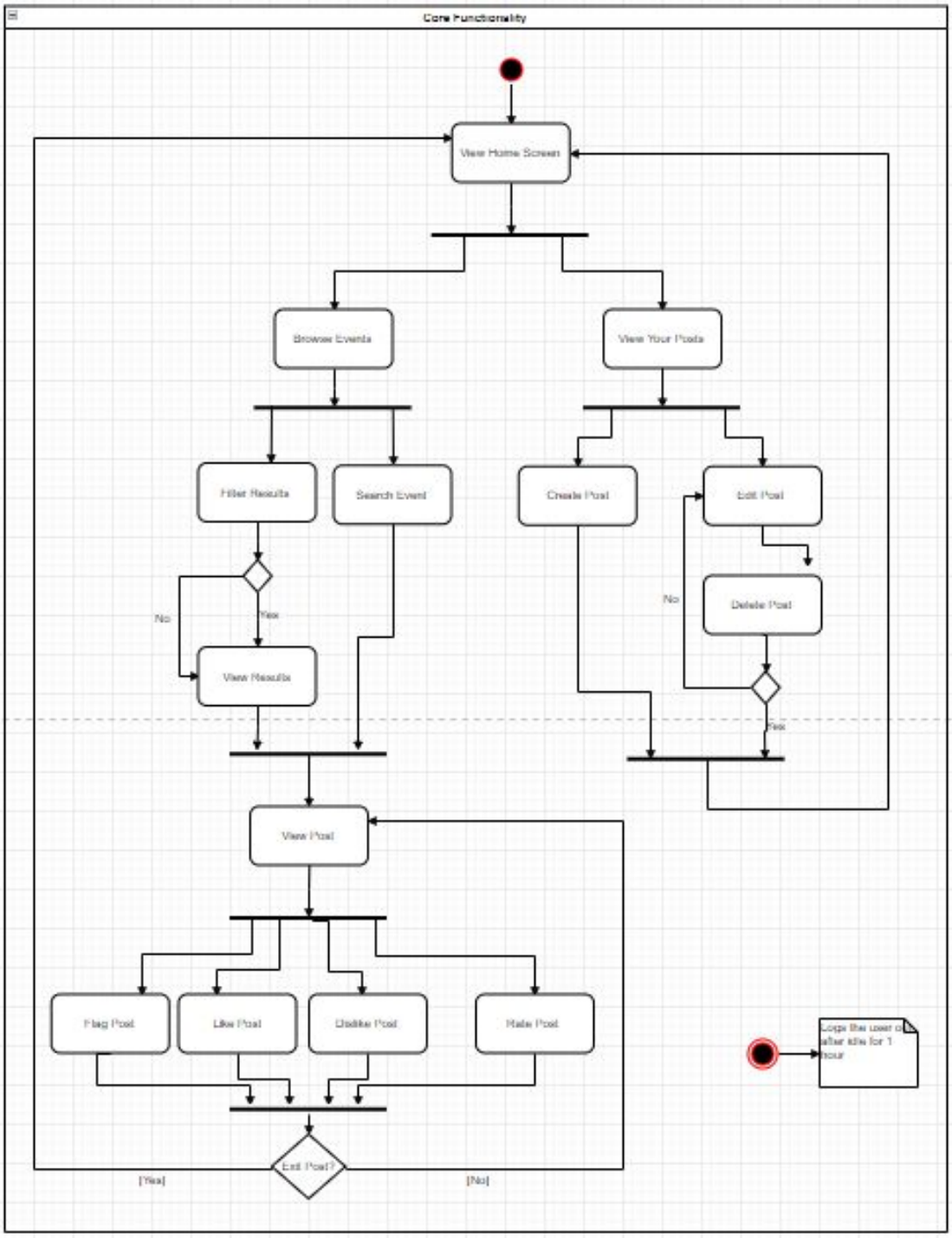
4.4.1 Authentication Activity Diagram

Authentication occurs whenever there is no active session between the user and the application. This is typically when the application first opens, or when a session expires. The authentication system is a prerequisite for all functionality of the application.



4.4.2 Core Functionality Activity Diagram

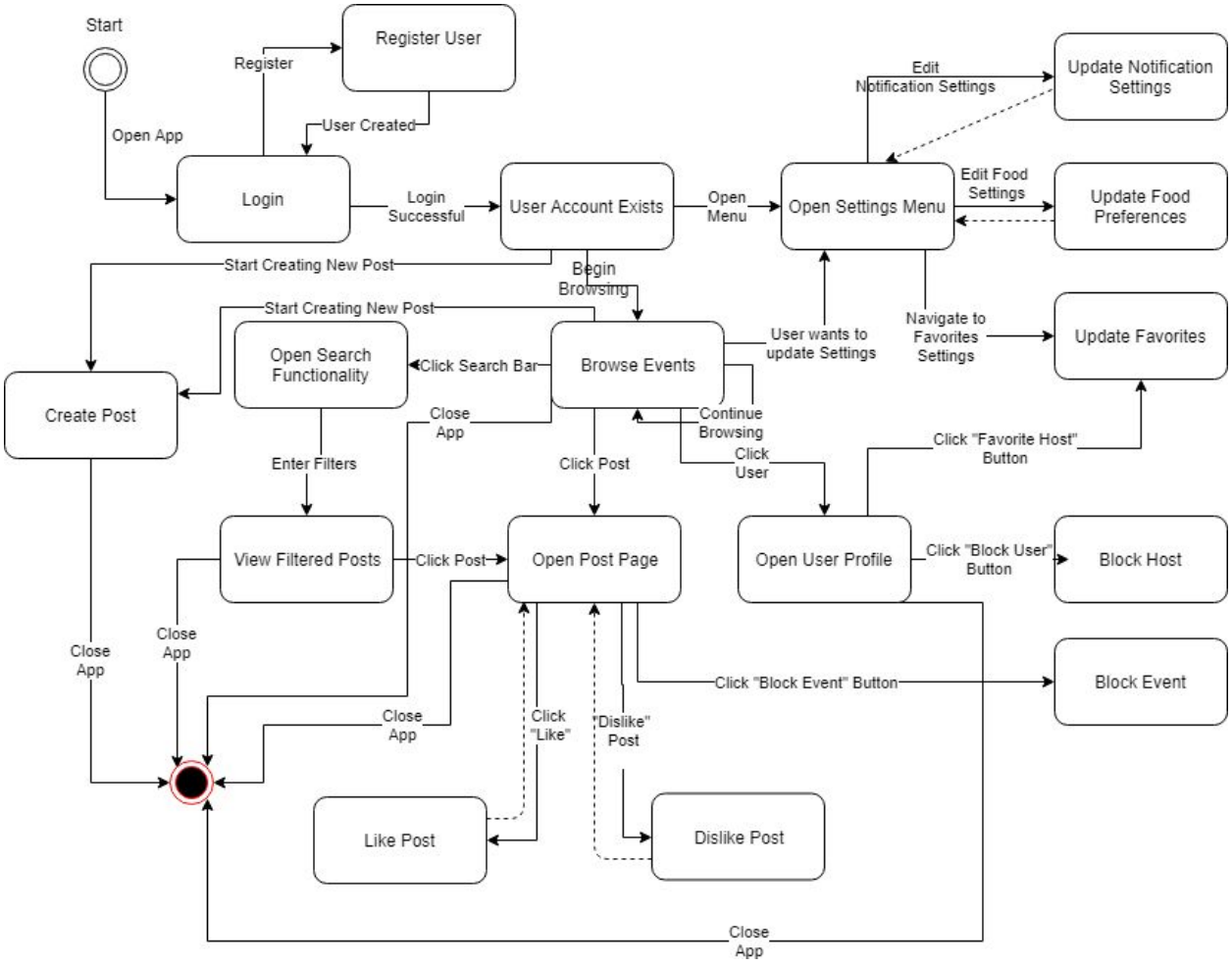
The following diagram represents the core functionality and the possible methodology when navigating the Cheap Eats software.



4.5 State-Transition Diagram

4.5.1 User State Diagram

After opening the application for the first time, Users will be prompted to register a new account, and all Users will be asked to login. After logging in, the User will be able to Browse Posts, Update Settings, Create Posts. Additionally, Users will be able to Like and Dislike Posts, Block Users and Events, and Search Posts by various filters. The app can be closed at any time.



5. Change Management Process

This section has intentionally been left blank, as there have been no core changes to the System Requirements Specification document as of version 2. Please see **Appendix B** for information about possible future changes to the System Requirement Specification Document.

Appendix A: Table of Contents

Revision History **2**

Document Approval **2**

1. Introduction **6**

1.1 Purpose 6

1.2 Scope 6

1.3 Definitions, Acronyms, and Abbreviations 6

2. General Description **7**

2.1 Product Perspective 7

2.2 Product Functions 7

2.3 Constraints, Assumptions and Dependencies 7

3. Specific Requirements **8**

3.1 External Interface Requirements 8

3.1.1 Software Interfaces 8

3.2 Functional Requirements 8

3.2.1 User Account Functionality 8

3.2.2 Search Functionality 8

3.2.3 Google Login 8

3.2.4 Set Search Preferences 8

3.2.5 Set Notification Preferences 8

3.2.6 Notification about Event in Your Area 8

3.2.7 Notification from Food Preferences 8

3.2.8 Notification from Favorites 9

3.2.9 Blocking Hosts 9

3.2.10 Blocking Food Types/Items 9

3.2.11 Like a Post 9

3.2.12 Favorite A Host 9

3.2.13 Application Navigation 9

3.2.14 Filter By Cuisine 9

3.2.15 Filter By Distance 9

3.2.16 Filter By Deal Quality 9

3.2.17 Sort By Number of “Likes” 10

3.2.18 Post Rating 10

3.2.19 Post Reporting 10

3.2.20 Create Post 10

3.3 Use Cases 11
3.3.1 Register a New User 11
3.3.2 Login User 11
3.3.3 Set Food Preferences 11
3.3.4 Set Notification Preferences 11
3.3.5 Set Favorites 11
3.3.6 Search for Events 11
3.3.7 Filter Results 11
3.3.8 Browse Events 11
3.3.9 Create a Post 11
3.3.10 Edit a Post 11
3.3.11 Delete a Post 11
3.3.12 Rate a Post 11
3.3.13 Like a Post 11
3.3.14 Dislike a Post 11
3.3.15 Flag a Post 11
3.3.16 Get Directions 11
3.3.17 Block a Host or Event 11
3.4 Classes / Objects 11
3.4.1 Restaurant 11
3.4.2 User 11
3.4.3 Post 12
3.4.4 PostDisplay 12
3.4.5 PostSearcher 12
3.4.6 DisplayManger 13
3.4.7 PostManager 13
3.4.8 NotificationManager 13
3.4.9 MapManager 13
3.4.10 UserManager 13
3.5 Non-Functional Requirements 13
3.5.1 Performance 13
3.5.2 Reliability 14
3.5.3 Availability 14
3.5.4 Security 14
3.5.5 Maintainability 14
3.5.6 Portability 14
3.5.7 Usability 14
3.5.8 Scalability 14
3.5.9 Dynamic Refreshing 14

3.5.10 Aesthetic Appeal 14
3.6 Logical Database Requirements 14
3.7 Other Requirements 15

4. Analysis Models 15
4.1 Use Case Diagrams 15
4.1.1 Browsing and Searching Use Case Diagram 15
4.1.2 Networking Use Case Diagram 16
4.2 Class Diagrams 17
4.2.1 Class Diagram 17
4.3 Sequence Diagrams 18
4.3.1 Authentication Sequence Diagram 18
4.3.2 Browse Events Sequence Diagram 19
4.4 Activity Diagrams 20
4.4.1 Authentication Activity Diagram 20
4.4.2 Core Functionality Activity Diagram 20
4.5 State-Transition Diagram 22
4.5.1 User State Diagram 22

5. Change Management Process 22

Appendix A: Table of Contents 23

Approvals 27

Use Case List 28

User Account Functionality 29
Feature Process Flow / Use Case Model 29
Use Case(s) 29

Browsing Functionality 31
Feature Process Flow / Use Case Model 31
Use Case(s) 31

User Preferences 33
Feature Process Flow / Use Case Model 33
Use Case(s) 33

Notifications 35
Feature Process Flow / Use Case Model 35
Use Case(s) 35

Posting Functionality 36
Feature Process Flow / Use Case Model 36

Use Case(s) 36

Directions **41**

Feature Process Flow / Use Case Model 41

Use Case(s) 41

Revision History

Version	Date	Revision Description
.01	6/6/2019	Initial Creation
1.0	6/11/2019	Completed Use Cases for SRS v1

Approvals

We have carefully assessed the Use Cases for this project. This document has been completed in accordance with the requirements of the System Development Methodology.

MANAGEMENT CERTIFICATION - Please check the appropriate statement.

_____ the document is accepted.

_____ the document is accepted pending the changes noted.

_____ the document is not accepted.

We fully accept the changes as needed improvements and authorize initiation of work to proceed. Based on our authority and judgment, the continued operation of this system is authorized.
(*=Required **= Submit for Review Approval Not Required)

Executive Sponsor**	DATE
Project Sponsor*	DATE
Quality Assurance Manager / Team Lead*	DATE
Business Analyst Manager / Team Lead*	DATE
Project Manager	DATE

Use Case List

Use Case ID	Primary Actor	Use Cases
1.1	User	Register a New User
1.2	User	Log-in
2.1	User	Event Browsing
2.2	User	Event Searching
2.3	User	Apply Filters
2.4	User	Sort
3.1	User	Set Food Preferences
3.2	User	Set Favorites
4.1	User	Set Notification Setting
4.2	User	Block a Host or Event
5.1	User	Create a Post
5.2	User	Edit a Post
5.3	User	Delete a Post
5.4	User	Rate a Post
5.5	User	Like a Post
5.6	User	Dislike a Post
5.7	User	Flag a Post
6.1	User	Get Directions

1 User Account Functionality

1.1 Feature Process Flow / Use Case Model

1.2 Use Case(s)

Use Case ID:	UC-1.1		
Use Case Name:	Register a New User		
Created By:	Scott Sheffer	**Last Updated By:**	Scott Sheffer
Date Created:	6/8/2019	**Last Revision Date:**	6/8/2019
Actors:	Restaurant, Student		
Description:	Any user can create a new account for personal preferences and information being saved across multiple uses of the application. It also gives access to commenting on posts and rating events/restaurants.		
Trigger:	Button is selected titled “Sign-up for new account”		
Preconditions:	1. The user has the application open.		
Postconditions:	1. A new account is added to the database along with the information provided at sign up.		
Normal Flow:	1. The user is on the initial screen when opening the app and selects the “Sign-up for new account” button 2. The user inputs their new username, email, selects their food preferences, notification preferences, and selects an avatar. 3. The use selects the “Create Account” button. 4. The application opens up to the home screen.		
Alternative Flows: **[Alternative Flow 1 – Not in Network]**	4a. In step 4 of the normal flow, if the customer selects the cancel button 1. The system will take the user back to the log-in screen.		
Exceptions:	2a. In step 2 of the normal flow, if the customer enters an already used username or email 1. A notification appears that explains its the inputted information is already in use.		
Includes:	N/A		
Frequency of Use:	On demand		
Special Requirements:	Leverage Google Firestore’s security features		
Assumptions:	Prior to use of this application the user has entered data for account sign up on another platform/website.		
Notes and Issues:	1. What are the maximum amount of accounts allowed in the database		

Use Case ID:	UC-1.2		
Use Case Name:	Log in		
Created By:	Benjamin Miller	**Last Updated By:**	Benjamin Miller
Date Created:	6/9/2019	**Last Revision Date:**	6/9/2019
Actors:	User		
Description:	A user that does not have an active session, in other words is not logged in, may log in by pressing a login button and authenticate themselves by logging in to their google account.		
Trigger:	Login button is pressed		
Preconditions:	User is not logged in, has no active session, and is on the homepage of the application.		
Postconditions:	The user has been logged in, has an active session, and may use any functions only available to an authenticated user.		
Normal Flow:	1. The user is redirected to the google authentication login screen. 2. The user enters their credentials 3. The user completes the google login and is redirected to the application's landing page.		
Alternative Flows:	1. The user is redirected to the google authentication login screen 2. The user fails to enter their credentials correctly 3. The user completes google auth's account recovery		
Exceptions:	1. The user is redirected to the google authentication login page. 2. The fails to enter their credentials 3. The user fails to complete google authentication account recovery 4. the user is redirected to applications login page they began with.		
Includes:			
Frequency of Use:	1 per hour.		
Special Requirements:	Google Authentication, Security, Availability, Reliability		
Assumptions:	1. The user has already completed account registration.		
Notes and Issues:			

2 Browsing Functionality

2.1 Feature Process Flow / Use Case Model

2.2 Use Case(s)

Use Case ID:	UC-2.1		
Use Case Name:	Event Browsing		
Created By:	Benjamin Miller	**Last Updated By:**	Benjamin Miller
Date Created:	6/9/19	**Last Revision Date:**	6/10/19
Actors:	User		
Description:	A User navigates browse tab and is returned a scrolling list of event postings		
Trigger:	User navigates to the browse tab		
Preconditions:	- User is logged in - User has navigated to the browse tab - User has an open connection to the application server		
Postconditions:	- A scrollable list of event postings is displayed - The list of posts has any requested filters applied		
Normal Flow:	1. User presses the “browse events” tab		
Alternate Flow 1- Return from post:	1. User clicks on a post from a browse session 2. User returns from that event’s page to the previous Browse session 3. User’s view is the same as before they clicked on the post		
Exceptions:			
Includes:			
Frequency of Use:	On demand		
Special Requirements:			
Assumptions:			
Notes and Issues:			

Use Case ID:	UC-2.2		
Use Case Name:	Event Searching		
Created By:	Benjamin Miller	**Last Updated By:**	Benjamin Miller
Date Created:	6/9/19	**Last Revision Date:**	6/10/19
Actors:	User		
Description:	A User has navigated to a tab labeled “Browse Events” and executes a search, specifying of keywords that may match details of a kind of post they are looking for, and is returned with a list of event postings that match the search.		
Trigger:	User enters keywords into the search bar and presses “search”		
Preconditions:	- User is logged in - User is currently viewing the Browse Events Tab - User has an open connection to the application’s server		
Postconditions:	- User is returned a list of posts match the given keywords - Default Filters have been reset to false - Newly Applied filters persist		
Normal Flow:	1. User enters keywords into the search bar 2. User presses the search icon		

Alternative Flow 1- Apply Filter:	User has already executed a search, and filter the results 0. Repeat steps 1 and 2 of Normal Flow 1. User clicks a “filters” button, which reveals a drawer 2. User selects which filters to apply 3. User hits ‘apply’ 4. Drawer hides and post list is updated
Exception 1:	
Includes:	UC 2.1, UC 2.3,
Frequency of Use:	On demand
Special Requirements:	
Assumptions:	
Notes and Issues:	

Use Case ID:	UC-2.3		
Use Case Name:	Apply Filters		
Created By:	Benjamin Miller	**Last Updated By:**	Benjamin Miller
Date Created:	6/9/19	**Last Revision Date:**	6/10/19
Actors:	User		
Description:	User applies filters to a current browsing session, returning a new search with the selected filters applied to the displayed list of posts		
Trigger:	User clicks the “filter” button on the “Browse” tab, revealing the filter tab in a drawer.		
Preconditions:	- User is logged in - User is currently on the “Browse” tab - User has an active connection to the application server		
Postconditions:	- A new list is returned with the requested filters applied		
Normal Flow:	1. User selects any filters they wish to apply 2. User selects “apply”		
Alternative Flow1- Pre Existing Filters:	0. User is already on the “Browse” tab, and already has filters applied 1. User opens the filter menu, displaying some filters that have already been selected 2. User adds selects new filters 3. User clicks “apply”		
Exceptions:			
Includes:	UC-2.1, UC-2.2		
Frequency of Use:	On demand		
Special Requirements:			
Assumptions:			
Notes and Issues:			

Use Case ID:	UC-2.4		
Use Case Name:	Sort		
Created By:	Benjamin Miller	Last Updated By:	Benjamin Miller
Date Created:	6/10/19	Last Revision Date:	6/10/19
Actors:	User		
Description:	User sorts the list of posts displayed on the “Browse” tab		
Trigger:	User clicks the “filter” button on the “Browse” tab, revealing a drop down menu with sorting options		
Preconditions:	- User is logged in - User is currently on the “Browse” tab - User has an active connection to the application server		
Postconditions:	- The list is sorted by the chosen sort option		
Normal Flow:	1. User selects any filters they wish to apply 2. User selects “apply”		
Alternative Flow1- Pre Existing Filters:	0. User is already on the “Browse” tab, and already has filters applied 1. User opens the filter menu, displaying some filters that have already been selected 2. User adds selects new filters 3. User clicks “apply”		
Exceptions:			
Includes:	UC-2.1, UC-2.2		
Frequency of Use:	On demand		
Special Requirements:			
Assumptions:			
Notes and Issues:			

3 User Preferences

3.1 Feature Process Flow / Use Case Model

3.2 Use Case(s)

Use Case ID:	UC-3.1		
Use Case Name:	Set Food Preferences		
Created By:	Christian Ford	Last Updated By:	Christian Ford
Date Created:	6/10/19	Last Revision Date:	6/10/19
Actors:	User		
Description:	A User is able to update their Food Preferences.		
Trigger:	User volition.		
Preconditions:	- User account exists - User account is logged in		
Postconditions:	- User Food Preferences changed		
Normal Flow:	1. User opens application 2. User navigates to settings menu		

	3. User opens Food Preferences 4. User updates Food Preferences a. Adding or Deleting Foods they would like to be notified about, depending on their Notification Settings 5. User navigates back to browsing
Alternative Flows:	1. User receives notification about a food they no longer want to receive notifications about 2. User opens notification, navigating them to the event that caused the notification 3. User navigates from event post to their settings menu 4. User updates Food Preferences a. Deleting the food they no longer want to receive notifications about b. User can also add new foods they would like to be notified about 5. User navigates back to browsing or closes the application
Exceptions:	- User enters blank as a food they would like to be notified about
Includes:	N/A
Frequency of Use:	As often as the User wants to change their settings
Special Requirements:	N/A
Assumptions:	User can read, write, and understand English
Notes and Issues:	

Use Case ID:	UC-3.2		
Use Case Name:	Set Favorites		
Created By:	Christian Ford	**Last Updated By:**	Christian Ford
Date Created:	6/10/19	**Last Revision Date:**	6/10/19
Actors:	User		
Description:	A User is able to add and delete hosts from their Favorites		
Trigger:	User volition.		
Preconditions:	- User account exists - User account is logged in		
Postconditions:	- User Favorites Changed		
Normal Flow:	1. User opens application 2. User navigates to settings menu 3. User opens their Favorites 4. User updates Favorites a. Adding of deleting Hosts they want to receive notifications about, depending on their notification settings 5. User navigates back to browsing		
Alternative Flows:	1. User is browsing the application 2. User finds a Host that they like 3. User clicks on the Host profile 4. User clicks the “Favorite Host” Button a. This adds the Host to the User’s list of Favorites 5. User navigates back to browsing or closes the application		
Exceptions:	- User tries to add a blank as a Favorite		
Includes:	N/A		
Frequency of Use:	As often as the User wants to update their Favorites		

Special Requirements:	N/A
Assumptions:	User can read, write, and understand English
Notes and Issues:	

4 Notifications

4.1 Feature Process Flow / Use Case Model

4.2 Use Case(s)

Use Case ID:	UC-4.1		
Use Case Name:	Set Notification Settings		
Created By:	Christian Ford	**Last Updated By:**	Christian Ford
Date Created:	6/10/19	**Last Revision Date:**	6/10/19
Actors:	User		
Description:	A User is able to set what types of events/updates are allowed to cause notifications to occur on their Android device.		
Trigger:	User volition.		
Preconditions:	- User account exists - User account is logged in		
Postconditions:	- User Notification Setting changed		
Normal Flow:	1. User opens application 2. User navigates to settings menu 3. User opens Notification Settings 4. User updates Notification Settings a. Specifying what types of food, what distances to events, etc. cause a notification to occur 5. User navigates back to browsing		
Alternative Flows:	1. User receives unwanted notification 2. User opens notification, navigating them to the event that caused the notification 3. User navigates from event post to their settings menu 4. User updates Notifications Settings b. Specifying what types of food, what distances to events etc. cause a notification to occur 5. User navigates back to browsing or closes the application		
Exceptions:	- User enters negative distance as a preference		
Includes:	N/A		
Frequency of Use:	As often as the User wants to change their settings		
Special Requirements:	N/A		
Assumptions:	User can read, write, and understand English		
Notes and Issues:			

Use Case ID:	UC-4.2		
Use Case Name:	Block a Host or Event		
Created By:	Christian Ford	**Last Updated By:**	Christian Ford

Date Created:	6/10/19	Last Revision Date:	6/9/19
Actors:	User		
Description:	A User is able to set what types of events/updates are allowed to cause notifications to occur on their Android device.		
Trigger:	User volition, User finds a Host, or an Event that they never want to appear on their application again.		
Preconditions:	- Existing User account - User account is logged in - User able to connect to the Internet/Database		
Postconditions:	- User blocks desired Hosts and Events		
Normal Flow:	1. User opens application 2. User is browsing the application 3. User finds an upcoming Event that they do not want to see again 4. User navigates to that Event 5. User finds a button that says “Block Event” 6. User clicks the button 7. This Event will not appear on the User’s browsing again		
Alternative Flows:	1. User opens application 2. User is browsing the application 3. User finds a Host that they do not want to see any Posts from again 4. User clicks on the Host’s icon, navigating to their profile 5. User finds a button that says “Block User” 6. User clicks the button 7. This Host will not appear on the User’s browsing again		
Exceptions:	- User does not have access to the Internet		
Includes:	N/A		
Frequency of Use:	As often as the User wants to change their settings		
Special Requirements:	N/A		
Assumptions:	User has access to the Internet, and is able to read, write, and understand English		
Notes and Issues:			

5 Posting Functionality

5.1 Feature Process Flow / Use Case Model

5.2 Use Case(s)

Use Case ID:	UC-5.1		
Use Case Name:	Create a Post		
Created By:	Nathan R. Hall	**Last Updated By:**	Nathan R. Hall
Date Created:	6/10/2019	**Last Revision Date:**	6/10/19
Actors:	User		
Description:	A user is able to create a 280 character post about certain food deals for other users to view / rate / like / dislike / flag		
Trigger:	User taps on the create a post button		
Preconditions:	- User is logged in		

	- User has an open connection to the database - User has a valid way to tap on the create a post button
Postconditions:	- Shows a preview of what the post said - Shows the user that the post successfully went up
Normal Flow:	1. User enters a post with size 280 characters or less 2. Post is able to reach the databases before going up 3. User hits the send button to post the content
Alternative Flows:	1. User is not connected to network or database 2. User enters a post with size greater than 280 characters 3. User does not hit the send button 4. User tries to send a post with 0 characters
Exceptions:	- User can't post more than twice a minute - User tries to send a post with more than 280 chars, the post will be denied
Includes:	None
Frequency of Use:	The user will use this case at an on demand frequency
Special Requirements:	- Valid connection to database
Assumptions:	- User is logged in and has valid credentials to create a post
Notes and Issues:	None

Use Case ID:	UC-5.2		
Use Case Name:	Edit a Post		
Created By:	Nathan R. Hall	**Last Updated By:**	Nathan R. Hall
Date Created:	6/10/2019	**Last Revision Date:**	6/10/19
Actors:	User		
Description:	A user is able to open up a post that they created and edit it		
Trigger:	User taps on the edit button on a post that they created		
Preconditions:	- User has an open connection to the application's server - User is logged in - User has a valid post that they can edit		
Postconditions:	- Shows the old post side by side with the new post and then sends the new post - Shows that the post was edited successfully - Puts an edited symbol on the post to show that it was edited		
Normal Flow:	1. User is openly connecting the applications server 2. User keeps the edited post below 280 characters 3. User own the original post that is being edited		
Alternative Flows:	1. User is logged out 2. User loses connection to the applications server 3. User edits the post to above 280 characters 4. User does not own the post that is being edited		
Exceptions:	- User can't edit a post over a week old		
Includes:	None		
Frequency of Use:	On demand		
Special Requirements:	- Valid connection to the applications server		
Assumptions:	- User is logged in and has valid credentials to edit a post		
Notes and Issues:	None		

Use Case ID:	UC-5.3		
Use Case Name:	Delete a Post		
Created By:	Nathan R. Hall	Last Updated By:	Nathan R. Hall
Date Created:	6/10/2019	Last Revision Date:	6/10/19
Actors:	User		
Description:	A user is able to delete a post they created		
Trigger:	User taps on the delete button on a post that they created		
Preconditions:	- User has an open connection to the application's server - User is logged in - User has a valid post that they can delete		
Postconditions:	- Shows that the post has been successfully deleted		
Normal Flow:	1. User is openly connecting the applications server 2. User clicks on the delete post button 3. User own the original post that is being deleted		
Alternative Flows:	1. User is logged out 2. User loses connection to the applications server 3. User does not own the post that is being deleted		
Exceptions:	In normal flow 2. there is a checker to make sure the user wants to actually delete the post.		
Includes:	None		
Frequency of Use:	on demand		
Special Requirements:	- Valid connection to the application's server		
Assumptions:	- User is logged in and has valid credentials to delete a post		
Notes and Issues:	None		

Use Case ID:	UC-5.4		
Use Case Name:	Rate a Post		
Created By:	Nathan R. Hall	Last Updated By:	Nathan R. Hall
Date Created:	6/10/2019	Last Revision Date:	6/10/19
Actors:	User		
Description:	A user is able to rate another user's post using a 1-5 stars rating system that will go on to the post		
Trigger:	A user taps on the "rate" option for the post that they see		
Preconditions:	- User has an open connection to the application's server - User is logged in - User is able to tap on a valid post to rate		
Postconditions:	- The post that was rated show the average of all ratings - A post that has never been rated will show the new rating on it - The user's rating score will be incremented for every post they rate		
Normal Flow:	1. User is actively browsing the posts with a valid login 2. User sees a post they want to rate 3. User clicks on the "rate" button on the post and brings up the rating box 4. User chooses an option to rate the post from 1, 2, 3, 4 or 5 5. User then confirms the rating and goes back to browsing		
Alternative Flows:	1. User does not have a valid connection to the application's server		

	2. User cannot rate a post because it was deleted while they clicked on it 3. User never confirmed the rating 4. User cant choose 1-5 on the rating 5. User cant see the “rate” button on the post
Exceptions:	- In normal flow 1 the user could not have a valid login
Includes:	None
Frequency of Use:	on demand
Special Requirements:	- Valid connection to the application’s server
Assumptions:	- User is logged in and has valid credentials to rate a post
Notes and Issues:	None

Use Case ID:	UC-5.5		
Use Case Name:	Like a Post		
Created By:	Scott Sheffer	**Last Updated By:**	Scott Sheffer
Date Created:	6/10/2019	**Last Revision Date:**	6/10/2019
Actors:	User		
Description:	When viewing an event post the user has the ability to like an event post.		
Trigger:	User selects the like button.		
Preconditions:	- The user logs in - The user views a post		
Postconditions:	- The like count increases by one		
Normal Flow:	- User is actively browsing the posts with a valid login - User sees a post they want to rate - User clicks on the “like” button on the post		
Alternative Flows: **[Alternative Flow 1 – Not in Network]**	- User does not have a valid connection to the application’s server - User cannot like a post because it was deleted while they clicked on it - The “like” button does not display properly		
Exceptions:	- The user does not have a valid log-in		
Includes:	N/A		
Frequency of Use:	On Demand		
Special Requirements:	- Valid connection to the server		
Assumptions:	- User is logged in and has valid credentials to like a post		
Notes and Issues:	N/A		

Use Case ID:	UC-5.6		
Use Case Name:	Dislike a Post		
Created By:	Scott Sheffer	**Last Updated By:**	Scott Sheffer
Date Created:	6/10/2019	**Last Revision Date:**	6/10/2019
Actors:	User		
Description:	When viewing an event post the user has the ability to like an event post.		
Trigger:	User selects the like button.		

Preconditions:	- The user logs in - The user views a post
Postconditions:	- The dislike count increases by one
Normal Flow:	- User is actively browsing the posts with a valid login - User sees a post they want to rate - User clicks on the “dislike” button on the post
Alternative Flows: **[Alternative Flow 1 – Not in Network]**	- User does not have a valid connection to the application’s server - User cannot like a post because it was deleted while they clicked on it - The “dislike” button does not display properly
Exceptions:	- The user does not have a valid log-in
Includes:	N/A
Frequency of Use:	On Demand
Special Requirements:	- Valid connection to the server
Assumptions:	- User is logged in and has valid credentials to like a post
Notes and Issues:	N/A

Use Case ID:	UC-5.7		
Use Case Name:	Flag a Post		
Created By:	Scott Sheffer	**Last Updated By:**	Scott Sheffer
Date Created:	6/10/2019	**Last Revision Date:**	6/10/2019
Actors:	User		
Description:	When viewing an event post the user has the ability to like an event post.		
Trigger:	User selects the like button.		
Preconditions:	- The user logs in - The user views a post		
Postconditions:	- The flag count increases by one		
Normal Flow:	- User is actively browsing the posts with a valid login - User sees a post they want to rate - User clicks on the “flag” button on the post		
Alternative Flows: **[Alternative Flow 1 – Not in Network]**	- User does not have a valid connection to the application’s server - User cannot like a post because it was deleted while they clicked on it - The “flag” button does not display properly		
Exceptions:	- The user does not have a valid log-in		
Includes:	N/A		
Frequency of Use:	On Demand		
Special Requirements:	- Valid connection to the server		
Assumptions:	- User is logged in and has valid credentials to like a post		
Notes and Issues:	N/A		

6 Directions

6.1 Feature Process Flow / Use Case Model

6.2 Use Case(s)

Use Case ID:	UC-6.1		
Use Case Name:	Get Directions		
Created By:	Christian Ford	**Last Updated By:**	Christian Ford
Date Created:	6/10/19	**Last Revision Date:**	6/11/19
Actors:	User		
Description:	User will be able to get directions to an event from the application		
Trigger:	User clicks “Get Directions” on the Post of an Event they would like to go to		
Preconditions:	- User has an active account - User logged in - User connected to the Internet		
Postconditions:	- User has directions to event		
Normal Flow:	1. User opens application 2. User browses upcoming Events 3. User finds an Event they would like to attend 4. User clicks on the Event’s Post 5. User clicks the “Get Directions” Button 6. User receives directions to upcoming event		
Alternative Flows:	1. User receives notification about upcoming Event 2. User clicks on notification 3. Application opens to Event’s Post 4. User clicks the “Get Directions” Button 5. User receives directions to upcoming event		
Exceptions:	- User does not have Internet access - Application does not have permission to use the device’s GPS		
Includes:	N/A		
Frequency of Use:	As often as User’s find Events they would like to attend		
Special Requirements:	N/A		
Assumptions:	- User has Internet Access - Application has access to GPS - User can read and understand English		
Notes and Issues:	N/A		

Appendix B: Acknowledgements of Desired Functionality

Preface

The purpose of Appendix B is to specify the future goals and functionality desired to be implemented of the Cheap Eats application that are not currently specified in the main body of this System Requirement Specification document.

Section 1 Expansion of Google Firebase Responsibilities as a Backend System

It is acknowledged that as of version 2.0 of this System Requirement Specification document that the role of Google Firebase as a backend for the application needs to be expanded upon. The functionality of Post Reporting. Post Trending, Notification Generation, and Post Filtering (Referenced in section 2.3 Functional Requirements), as well as the functionality for the validation of a user's university email on registration, will require additional back-end support. As the development process continues, this back-end support will result in revisions to the various diagrams in Section 4 that are not currently reflected in version 2.0 of this System Requirement Specification document.