Software Requirements Specification Document

FoodSaver

1.0.5
3rd December 2022

FoodSavers
Aishwarya Kasthala

Submitted in partial fulfillment of the requirements of
IT 426 – Advanced Software Engineering

Table of Contents

1. Introduction

This SRS document is intended to provide the software development team, client, and other project stakeholders with an initial introduction to the product scope, product perspective, user characteristics, system environment, functional and non-functional requirements, and use cases.

1.1 Purpose

The purpose of this document is to present an in-depth understanding of the FoodSaver mobile application by defining the problem statement of the target users and the solution or a skeleton framework for it. This document will describe the application's features, the conditions under which it must operate, and how the system will respond to external inputs. This document is intended for both stakeholders and the developers of the application and will be submitted to the client for approval.

1.2 Scope

Left-over food wastage is one of the major problems in the world. ***FoodSaver*** *application will try to minimize this problem by establishing a platform between the donors and the receivers.*

This application has three types of users: The administrator, the donors, and the receivers. Both the donors and receiver parties may have two kinds of accounts: A single donor account or a donor group account such as a restaurant and an Individual receiver account or receiver group account such as non-governmental organizations (NGOs).

The donor who wants to donate food to receivers can do so by simply posting their information on the application, such as food details, contact information, location details, and so on. The receivers can view the information posted by the donors and can reserve food with a single click.

This application facilitates email communication between the donor and the recipient/ receiver. This application does not provide any payment gateways, as the users of this application provide food to others free of cost. Also, the application doesn't ensure the food quality.

1.3 Definitions, Acronyms, and Abbreviations

Product FoodSaver - the application deliverable
User - a person who uses the application
Donor - an individual who wants to donate food
Donor group - a group of individuals who want to donate food. For example, a restaurant.
Receiver - a needy person who wants to preserve food
Receiver group - a group of individuals who want to reserve food, like an NGO.
Admin – The administrator account that has the highest privileges within the system.
iOS - an operating system for mobile applications

1.4 References

1. *https://foodrescue.us*
2. *https://olioex.com*
3. *https://www.flashfood.com*

1.5 Overview

The remaining sections of this document provide a product perspective, the user characteristics, the environment under which it operates, and constraints and assumptions made while designing the product which is included in section 2. Section 3 gives a detailed description of the functional requirements and non-functional requirements which is primarily for the app developers to explain the functional details.

2. General Description

This section of the document is to make the user understand the functionalities of the software product.

2.1 Product Perspective

The purpose of the FoodSaver application is to resolve the hunger and food wastage problems simultaneously. As large amounts of food get wasted every day, FoodSaver acts as a platform to connect food donors with the needy. The primary distinction between FoodSaver and similar apps is that they are donor oriented. FoodSaver is an application developed taking both the donors and receivers into consideration. In this app, the receivers can look for available food donors, communicate and reserve according to their needs. NGOs and common people can use this app to solve the food crisis in society.

2.2 User Characteristics

This mobile application is aimed at a broad range of users from youngsters to elderly clients who are willing to help the needy. It is more useful for people who have food businesses, like restaurants where food is prepared in large quantities and is thrown away daily. Also, the organizations like NGO groups, as they do volunteer work.

2.3 System Environment

This application is an iOS mobile application that is supported only on iPhones and iPads. It's a mobile responsive application that is compatible with iPhones of different dimensions. This app is developed using Swift programming language.

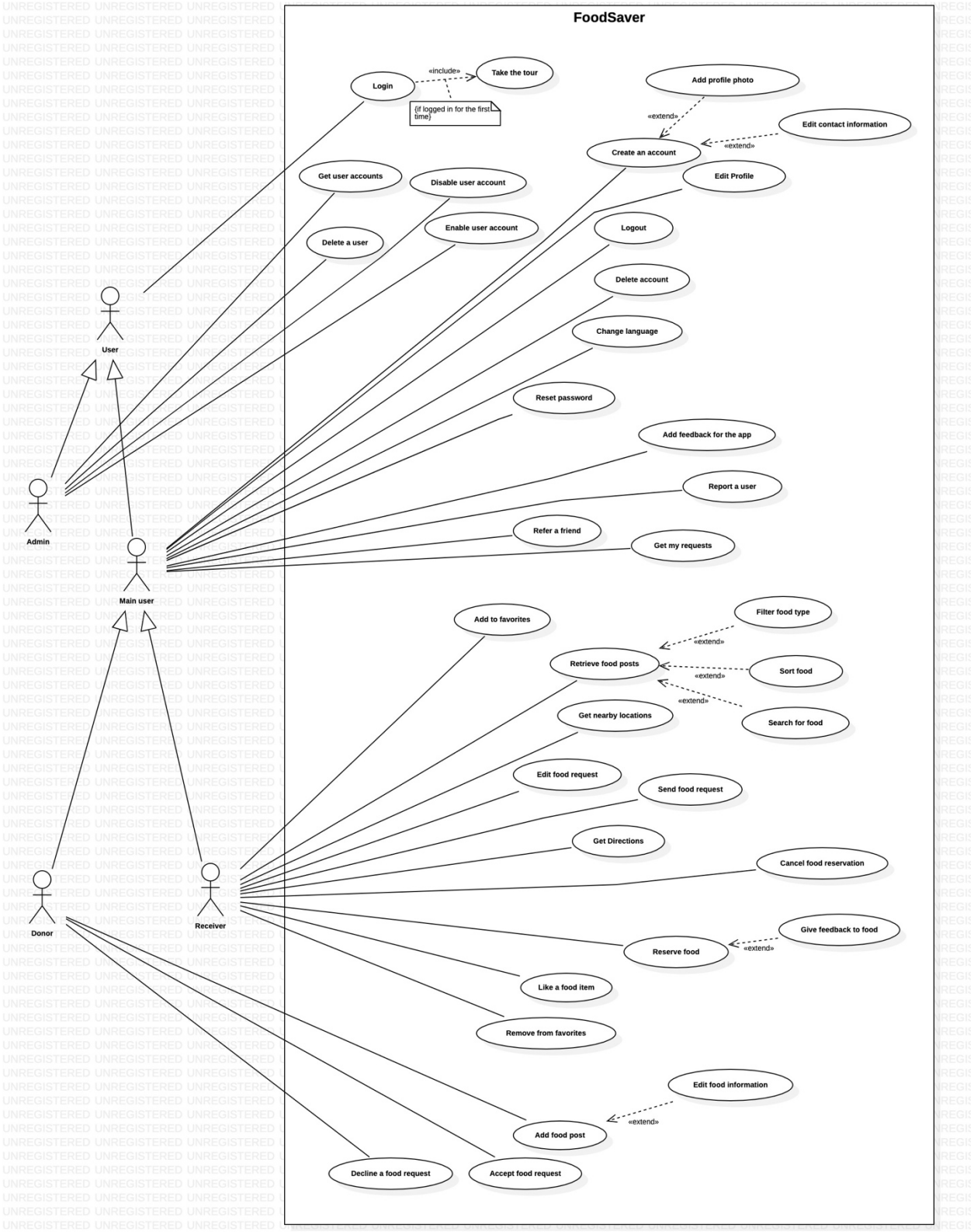
As this is an iOS application, it doesn't support Android phones or desktops of any operating system.

2.4 Assumptions and Dependencies

The user of this application must have an iOS-supporting device - iPhone or iPad.

3. Specific Requirements

This section describes specific features of the FoodSaver application.



FoodSaver: Use Case Diagram

3.1.1 Login

a. Description – *The user can log in to the application*.

b. Actor(s) – *User*

c. Trigger – *When the user clicks on the 'Login' button.*

d. Conditions

i. Pre - *The user must have an account within the system.*

ii. Post – *The user will be redirected to the Dashboard screen.*

e. Exceptions –

i. *If the input text fields are left blank, then an error message is thrown as "Please enter the details".*

ii. *If the login fails, then the user must close the application and try logging in again.*

3.1.2 Create an account

a. Description – *The user can create a new account*.

b. Actor(s) – *Main User*

c. Trigger – *When the user clicks on the 'Create account' button.*

d. Conditions

i. Pre - *The user must have the app installed on their phone.*

ii. Post – *The user will be able to log in to the application and will be redirected to the Take the tour screen.*

e. Exceptions –

i. *If the details entered by the user such as name, contact info, and so on are invalid, then an exception is shown as "Invalid details. Please try again."*

ii. *If the input text fields are left blank, then an error message is thrown as "Please enter the details".*

iii. *If the system is unable to create an account, the system will display an error message and will ask the user to try again later*.

3.1.3 Logout

a. Description – *The user can log out of the application.*

b. Actor(s) – *User*

c. Trigger – *When the user clicks on the logout button.*

d. Conditions

i. Pre – *The user must be logged into the application.*

ii. Post – *The user is logged out of the application and will be redirected to the app's main screen.*

e. Exceptions –
   i. *If there is any network error, the logout operation will fail, and the user has to re-launch the application.*

3.1.4 Take the tour

a. Description – *The user can take a tour of the application, which will show all the important features of the application*.

b. Actor(s) – *Main user*

c. Trigger – *When the user logs into the application for the first time.*

d. Conditions
   i. Pre – *The user must be logged into the application.*
   ii. Post – *The user can either click the 'Skip' or 'Next' buttons. After receiving the overview, the user is redirected to the home screen of the app.*

e. Exceptions: *NA*

3.1.5 Reset Password

a. Description: *The user can reset their account's password.*

b. Actor: *User*

c. Trigger: *When the user clicks on the "Forgot password?" button.*

d. Conditions:
   i. Pre*: The user must have an existing account.*
   ii. Post*: A message will be displayed as "Password changed successfully" and the new password will be saved to the database.*

e. Exceptions*:*
   i. *If there is any network error, the logout operation will fail, and the user has to re-launch the application.*

3.1.6 Delete account

a. Description – *The user account gets deleted permanently from the system.*

b. Actor(s) – *Main user*

c. Trigger – *When the user clicks on the "Delete account" button.*

d. Conditions -

i. Pre - *User must have an existing account within the system.*

ii. Post – *The user is signed out of the application and is directed to create an account page and the user details will be removed from the database.*

e. Exceptions*:*

i. If there is any network error, the delete operation will fail and the user must re-launch the application.

3.1.7 Delete a user

a. Description – *The admin can delete a user permanently from the system*.

b. Actor(s) – *Admin*

c. Trigger – *When the admin clicks on the 'Delete this user' button.*

d. Conditions

i. Pre – *The account admin wants to delete should be an existing user.*

ii. Post – *A pop-up message will be shown as "Deleted user successfully" and will be removed from the database.*

e. Exceptions:

i. If there is any network error while deleting the user, then an exception will be shown as "Delete failed. Please try again".

3.1.8 Enable user account

a. Description – *The admin can enable a user account*.

b. Actor(s) – *Admin*

c. Trigger – *When the admin clicks on the 'Enable this user' button.*

d. Conditions

i. Pre – *The account admin wants to enable should be an existing user.*

ii. Post – *A pop-up message will be shown as "Enabled user successfully" and the selected user will be unblocked from logging into the application.*

e. Exceptions:

i. If there is any network error during this operation, then an exception will be shown as "Failed. Please try again". The admin must close the application and re-try enabling it.

3.1.9 Disable user account

a. Description – *The admin can disable a user account temporarily from the system*.

b. Actor(s) – *Admin*

c. Trigger – *When the admin clicks on the 'Disable this user' button.*

d. Conditions

   i. Pre – *The account admin wants to disable should be an existing user.*

   ii. Post – *A pop-up message will be shown as "Disabled the user successfully" and the selected user will be blocked from logging into the application.*

e. Exceptions:

   i. *If there is any network error during this operation, then an exception will be shown as "Failed. Please try again". The admin must close the application and re-try disabling it.*

3.1.10 Report a user

a. Description – *The user can report a suspicious account.*

b. Actor(s) – *Receiver*

c. Trigger – *When the user clicks on the 'Report' icon button.*

d. Conditions

   i. Pre – *The user must be logged into the application.*

   ii. Post – *The reported account will be under review.*

e. Exceptions:

   i. *If the report fails due to a network issue, then an exception will be shown as "Failed. Please try again".*

3.1.10 Edit profile

a. Description – *The user can edit their profile, like updating their details.*

b. Actor(s) – *Main user*

c. Trigger – *When the user clicks on the 'Edit profile' icon button.*

d. Conditions

   i. Pre – *The user must be logged into the application.*

   ii. Post – *A pop-up message will be displayed as "Profile edited successfully" and new changes can be seen. The new details will be updated in the database.*

e. Exceptions:

   i. *If editing profile fails due to a network issue, then an exception will be shown as "Failed. Please try again".*

3.1.11 Add profile photo

a. Description – *The user can add a profile picture*.

b. Actor(s) – *Main user*

c. Trigger – *When the user clicks on the 'Photo' icon on the 'Edit' profile screen.*

d. Conditions

i. Pre – *The user must be logged into the application.*

ii. Post – *Selected pictures by the user will be displayed on the profile.*

e. Exceptions:

i. *If the attached file is bigger than the desired size, an exception will be thrown as "File attached exceeds size. Please try again"*.

3.1.12 Add food post

a. Description – *The user can add food posts along with contact and location information*.

b. Actor(s) – *Donor*

c. Trigger – *When the user clicks on the '+" icon on the dashboard screen.*

d. Conditions

i. Pre - *The user must be logged into the application.*

ii. Post – *Food post is added to the dashboard screen and the data will be saved to the database.*

e. Exceptions –

i. *If adding food post profile fails due to a network issue, then an exception will be shown as "Failed. Please try again"*.

3.1.13 Edit food information

a. Description – *The user can edit or update the food information*

b. Actor(s) – *Donor*

c. Trigger – *When the user clicks on the 'Pencil' icon button.*

d. Conditions

i. Pre – *The user must have an existing food post on the application.*

ii. Post – *A message will be displayed as "Updated food details" and the data will be saved to the database*

e. Exceptions:

i. *If editing food information fails due to a network issue, then an exception will be shown as "Failed. Please try again".*

3.1.14 Edit contact information

a. Description – *The user can edit or update the contact information on the food post.*
b. Actor(s) – *Donor*
c. Trigger – *When the user clicks on the 'Edit' icon button.*
d. Conditions
   i. Pre – *The user must have a food post already on the application.*
   ii. Post – *A message will be displayed as "Updated contact details" and the data will be saved to the database.*
e. Exceptions:
   i. *If editing contact information fails due to a network issue, then an exception will be shown as "Failed. Please try again".*

3.1.15 Delete a food post

a. Description – *The user can delete an already added food post.*
b. Actor(s) – *Donor*
c. Trigger – *When the user clicks on the 'Delete' icon on the food post.*
d. Conditions
   i. Pre – *There should be an existing food post added by the donor.*
   ii. Post – *A pop-up message will be shown as "Post deleted successfully" and will be deleted from the database.*
e. Exceptions:
   i. *If the post fails to delete, then an exception will be shown as "Delete failed. Please try again".*

3.1.16 Retrieve food posts

a. Description – *The user can view available food posts.*
b. Actor*(s)* – *Receiver*
c. Trigger – *When the user logins to the application.*
d. Conditions
   i. *Pre - The user must be logged into the application.*

ii. *Post – Users will be able to send food requests to the donor.*

e. Exceptions:

i. *If the application doesn't fetch results, then an exception will be thrown as "Unable to fetch. Please try again".*

3.1.17 Send food request

a. Description – *The user can send a food request to the donor.*

b. Actor(s) – *Receiver*

c. Trigger – *When the user clicks on the 'Send request' button.*

d. Conditions

i. Pre – *There must be food posts already added by the donor on the dashboard screen.*

ii. Post – *The food request will be received by the donor.*

e. Exceptions –

i. *If the food request fails to send due to network issues, then an exception will be thrown as "Failed. Please try again". The user must close the application and re-try sending the request again.*

3.1.18 Edit food request

a. Description – *The user can either edit the food request such as editing the food quantity or delete the food request.*

b. Actor(s) – *Receiver*

c. Trigger – *When the user clicks on the 'Edit request' button.*

d. Conditions

i. Pre - *The user must have already sent the food requests.*

ii. Post – *The edits made by the receiver can be seen and the updated food requests can be seen in the donor's food requests list.*

e. Exceptions –

i. *If the user makes an edit and is not reflected in the application, then an exception is thrown as "Edit failed. Please try again".*

ii. *If the user adds an unavailable quantity, then an exception will be thrown as "Unavailable. Please try again".*

3.1.19 Accept food request

a. Description – *The user can accept/ decline a food request sent by the receiver.*

b. Actor(s) – *Donor*

c. Trigger – *When the user clicks on the 'Accept' button.*

d. Conditions

i. Pre – *There must be food requests sent by receivers.*

ii. Post – *The food request will be processed according to the donor's wish.*

e. Exceptions – *NA*

3.1.20 Decline a food request

a. Description – *The user can decline a food request sent by the receiver*.

b. Actor(s) – *Donor*

c. Trigger – *When the user clicks on the 'Decline' button.*

d. Conditions

iii. Pre – *There must be food requests sent by receivers.*

iv. Post – *The food request will be processed according to the donor's wish.*

e. Exceptions – *NA*

3.1.21 My requests

a. Description – *The user can see their food request history*.

b. Actor(s) – *Main user*

c. Trigger – *When the user clicks on the 'My requests' button in the hamburger menu.*

d. Conditions

i. Pre – *The user must be logged into the application.*

ii. Post – *The user will be able to see their food request history, along with accepted and declined.*

e. Exceptions – *NA*

3.1.22 Reserve Food

a. Description – *The user can reserve food as a final step after the donor accepts their food request.*

b. Actor(s) – *Receivers*

c. Trigger – *When the user clicks on the 'Reserve food' button.*

d. Conditions

i. Pre – *There must be accepted food requests*

ii. Post – *A pop-up message will be shown as "Reserved food" and the details will be updated in the database.*

e. Exceptions*:*

i. *If the selected food quantity is more than the available food, then an error message is shown as "Unavailable. Please try again".*

ii. *If the entered food quantity is left blank, then an exception will be thrown as "Please try again".*

3.1.23 Cancel food reservation

a. Description – *The user can cancel the food reservation.*

b. Actor*(s) – Receiver*

c. Trigger – *When the user clicks on the cancel food button.*

d. Conditions*:*

i. *Pre - The order should be placed already.*

ii. *Post – An alert message will be shown as "Are you sure?", If clicked yes, then the order will be canceled, and the order details are removed from the database.*

e. Exceptions: *NA*

3.1.24 Filter food type

a. Description – *The user can filter the food posts list tiles based on the options shown.*

b. Actor(s) – *Receiver*

c. Trigger – *When the user clicks on the 'filter' icon button.*

d. Conditions

i. Pre – *The user must be on the dashboard screen.*

ii. Post – *The food details are shown based on the filter option selected by the user.*

e. Exceptions – *NA*

3.1.25 Sort food

a. Description – *The user can sort the food posts list tiles based on the quantity. This can be either increasing or decreasing order.*

b. Actor(s) – *Receiver*

c. Trigger – *When the user clicks on the 'sort' icon button.*

d. Conditions

i. Pre – *The user must be logged into the application.*

ii. Post – *The food details are shown based on the sort order selected by the user.*

e. Exceptions – *NA*

3.1.26 Search for a food item

a. Description: *The user can search for a particular food item.*

b. Actor: *Receiver*

c. Trigger: *When the user enters an input in the search field.*

d. Conditions:

i. Pre*: The user must be on the dashboard screen.*

ii. Post*: The food details are shown based on the input entered by the user.*

e. Exceptions*: NA*

3.1.27 Add to favorites

a. Description – *The user can add a particular food item to his/ her favorites.*

b. Actor(s) – *Receiver*

c. Trigger – *When the user clicks the 'love' icon on the food post.*

d. Conditions

i. Pre – *The food item should be available on the dashboard screen.*

ii. Post – *The food item selected will be added to the user's 'favorites' list.*

e. Exceptions – *NA*

3.1.28 Remove from favorites

a. Description – *The user can remove a food item from their favorites list.*

b. Actor(s) – *Receiver*

c. Trigger – *When the user deselects the 'love' icon button.*

d. Conditions

i. Pre – *The food item should already be present in their favorites list.*

ii. Post – *The food item will be removed from favorites and removed from the database.*

e. Exceptions:

i. If the favorite food item already added by the user doesn't exist, then the item won't be visible in their favorites.

3.1.29 Like a food item

a. Description – *The user can like the food item of their choice.*

b. Actor(s) – *Receiver*

c. Trigger – *When the user clicks on the 'Like' icon button.*

d. Conditions

   i. Pre – *The food item should be available on the dashboard screen*

   ii. Post – *An indication will be shown to the user and will be added to liked orders/ liked stores list.*

e. Exceptions: *NA*

3.1.30 Get Directions

a. Description – *The user can view directions to a particular location.*

b. Actor(s) – *Receiver*

c. Trigger – *When the user clicks on the Directions icon button.*

d. Conditions

   i. Pre – *There must be a food post.*

   ii. Post – *The user will be redirected to Apple maps or Google maps.*

e. Exceptions –

   i. *If the user doesn't give location permissions, the permissions dialog box will be shown asking the user to allow or decline location permissions. The user will be redirected to the Maps application only when the user agrees to the permissions.*

3.1.31 Get nearby Locations

a. Description – *The user can view nearby locations of the donors from the user's current location.*

b. Actor(s) – *Receiver*

c. Trigger – *When the user clicks on 'Show nearby locations.*

d. Conditions

   i. Pre – *The user must be logged into the application.*

   ii. Post – *Users will be shown nearby locations from their current location*

e. Exceptions –

   i. *If the user denies the location permissions, the user will be prompted with the permission dialog box.*

3.1.32 Change language

a. Description – *The user can change the language of the application.*

b. Actor(s) – *Receiver and Donor*

c. Trigger – *When the user changes the language on their phone settings.*

d. Conditions

i. Pre - *The user must be logged into the application.*

ii. Post – *The application will be shown in a different language that was selected by the user.*

e. Exceptions –

i. *If the user selects a language that is not supported by the application, then the language remains the same as the previous one.*

3.1.33 Refer a friend

a. Description – *The user can send a friend a referral link to the FoodSaver app.*

b. Actor(s) – *Main user*

c. Trigger –*When the user clicks on the 'Refer a friend button in the hamburger menu.*

d. Conditions -

i. Pre - *User must be logged into the application.*

ii. Post – *The user will be able to send an email with app information to the recipient's email.*

e. Exceptions:

i. *If the user doesn't select any other app to share information, then the user will be on the current screen they were on.*

3.1.34 Add feedback for the application

a. Description – *The user can give a rating and comment on the app experience*

b. Actor(s) – *Donor and Receiver*

c. Trigger – *When the user clicks on the 'Feedback' tile in the hamburger menu*

d. Conditions

i. Pre – *The user must be logged into the application.*

ii. Post – *A pop-up message will be displayed as "Thanks for your feedback".*

e. Exceptions:

i. *If the user doesn't give a rating, then an exception will be thrown as "Please select the stars".*

ii. *If the feedback comment section is left blank, then a validation error message will be shown as "Please enter comments".*

iii. *If the user enters more characters than the character limit, then an exception will be thrown as "You have exceeded the limit. Please try again".*

3.1.35 Add feedback for a food item

a. Description – *The user can give feedback on a particular food item*

b. Actor(s) – *Receiver*

c. Trigger – *When the user clicks on the 'Feedback 'button.*

d. Conditions

i. Pre – *The food item should be available on the dashboard screen.*

ii. Post – *A pop-up message will be displayed as "Thanks for your feedback".*

e. Exceptions:

i. *If the feedback comment section is left blank or the input entered is invalid, then an exception will be thrown "Please try again".*

3.1.36 Get user accounts

a. Description – *The user can retrieve all the user accounts.*

b. Actor(s) – *Admin*

c. Trigger – *When the user logs into the application.*

d. Conditions

i. Pre – *The user must be logged into the application.*

ii. Post – *The user will be able to see all user accounts of the application.*

e. Exceptions:

i. *If the user fails to retrieve the user accounts due to network issues, then an exception will be thrown as "Failed. Please close the application and re-try".*

3.2 Non-Functional Requirements

3.2.1 Performance

The FoodSaver app operates based on the internet speed of the user's device. Even with a heavy workload, the application's features load in less than 5 seconds. The application will be tested in

cycles to determine its performance. Each database update cycle in the application takes less than 60 seconds to complete. Following the development, the application will be in the production phase for one month, during which time the performance of the app will be evaluated by testers by trying the features that typically slow down the speed.

3.2.2 Reliability

The FoodSaver application will be tested in iterations to determine the pass and failure rate. The application's developers will have unit test cases to check the functionalities' pass and failure rates. There will also be a bi-weekly release of the features developed in that sprint, where manual testing is performed by the testers. If the failure rate of the test cases is relatively high, necessary changes will be implemented. In addition, the application code will be written in accordance with coding standards while keeping all object-oriented principles and design patterns in mind to increase reusability.

3.2.3 Availability

Except for network outages, the FoodSaver application will be available 24 hours a day, seven days a week. The application will have less downtime. If a database update fails to commit, the database update process will undo all relevant changes.

3.2.4 Security

All account-related information will be stored in a secure local database that is stored in the application's local memory. Private memory is the only way to access database information. According to the owner's privacy policy, this application will protect the privacy of their personal information. Passwords can only be viewed if the user so desires.

3.2.5 Maintainability

The FoodSaver application will be continuously monitored by the technical team. Bugs reported by testers will be fixed parallelly by developers. Various tools will be used to track all the changes made. The application will be regularly maintained to reflect the developers' updates. The maintenance will be carried out after regular business hours. Documentation will be updated in case of any modifications.

3.2.6 Portability

This application is only compatible with iOS devices. If the user purchases a new phone, the application can be installed from the App Store and will continue to function as before.

4. Design & Development

4.1Software Process Model

The software process model we have chosen for our application is 'Scrum model'. This would be a suitable model for our application since it allows us to adjust to new revisions if any new requirements arise. Since we'll accomplish this in iterations, we will have much more time to fix any bugs. We figured this would be ideal for our project because we could segregate the work in each sprint and integrate them at the end. This methodology also assures that our team communicates effectively. Each sprint lasts ten days (except for Saturday and Sunday). We intend to conduct standup meetings on weekdays. We will hold a sprint retrospection at the end of each sprint.

4.2 Deliverable 1

4.2.1 Description

Our team has been divided into two sub-teams: design and development. The team began revising the basic SRS document, requirement exceptions, and use case diagram for this milestone. For our application, our design team created UX wireframes. Our development team created database model classes as well as several UI View models. We worked together to create the application's initial class diagram. The team will modify this class diagram in subsequent iterations according to the implementation.

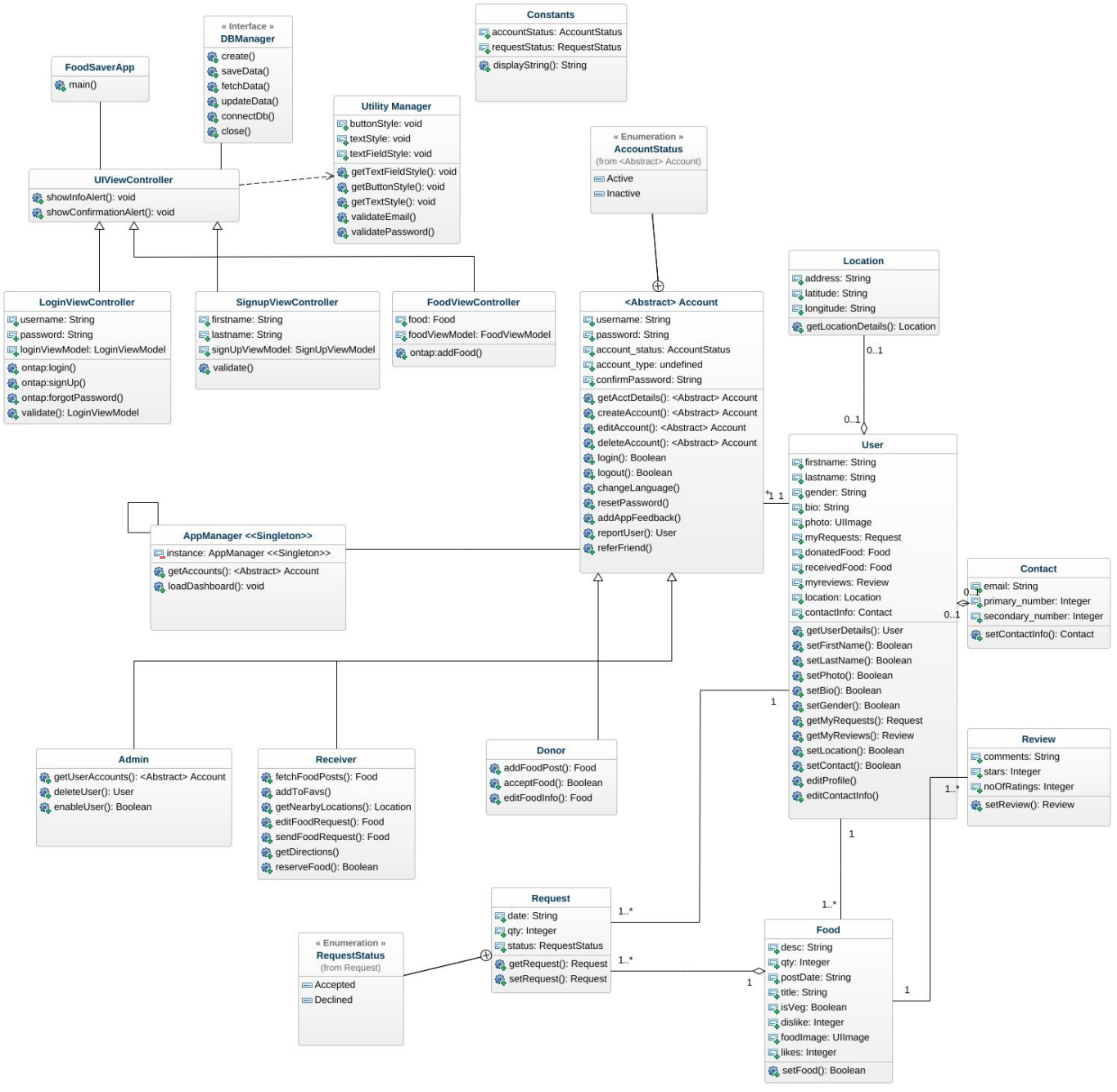
4.2.2 Design

4.2.2.1. Class Diagram

The class diagram attached shows how our system is organized and how they interact with each other.

FoodSaver App – Contains the main method of the application.
DBManager – Database manager class that controls all the database related operations.
UIViewController – Super class that controls the UI of the application - which has all the UI View models. Every screen in the application is added as child class to this.
LoginViewController – Controls UI of the login page. It has a default object called as 'loginViewModel'. LoginViewModel validates the account information and 'loginView' displays UI.
SignUpViewController – Controls UI of signup page
FoodViewController – Controls UI of Food posts page
AppManager – Singleton class which takes 'Account' as the only object. Whenever a user logs in, the AppManager stores the logged in account information.
Utility Manager – Contains styling of all the UI elements in the application, For e.g.: Button style, text field style, text style, etc.
Constants – Contains all the constant variables, which are used throughout the application.
Account – An abstract class which has all the account related information and behaviors related to it.
Admin – Child class of Account, which has behaviors related to Admin user.

Donor - Child class of Account, which has behaviors related to Donor user.
Receiver - Child class of Account, which has behaviors related to Receiver user.
User – Class that has all the user information and behaviors related to it.
Location – Class that contains all the location information.
Contact – Class that contains all the contact information.
Review – Class that contains all the review information.
Request – Class that contains all the food request information.
Food – Class that contains all the food post information.
AccountStatus – Enum that takes two values: Active/ Inactive - tells us about the account status
RequestStatus – Enum that takes two values: Accepted/ Declined – tells us about the food request status



FoodSaver: Class Diagram 1

4.3 Deliverable 2

4.3.1 Description

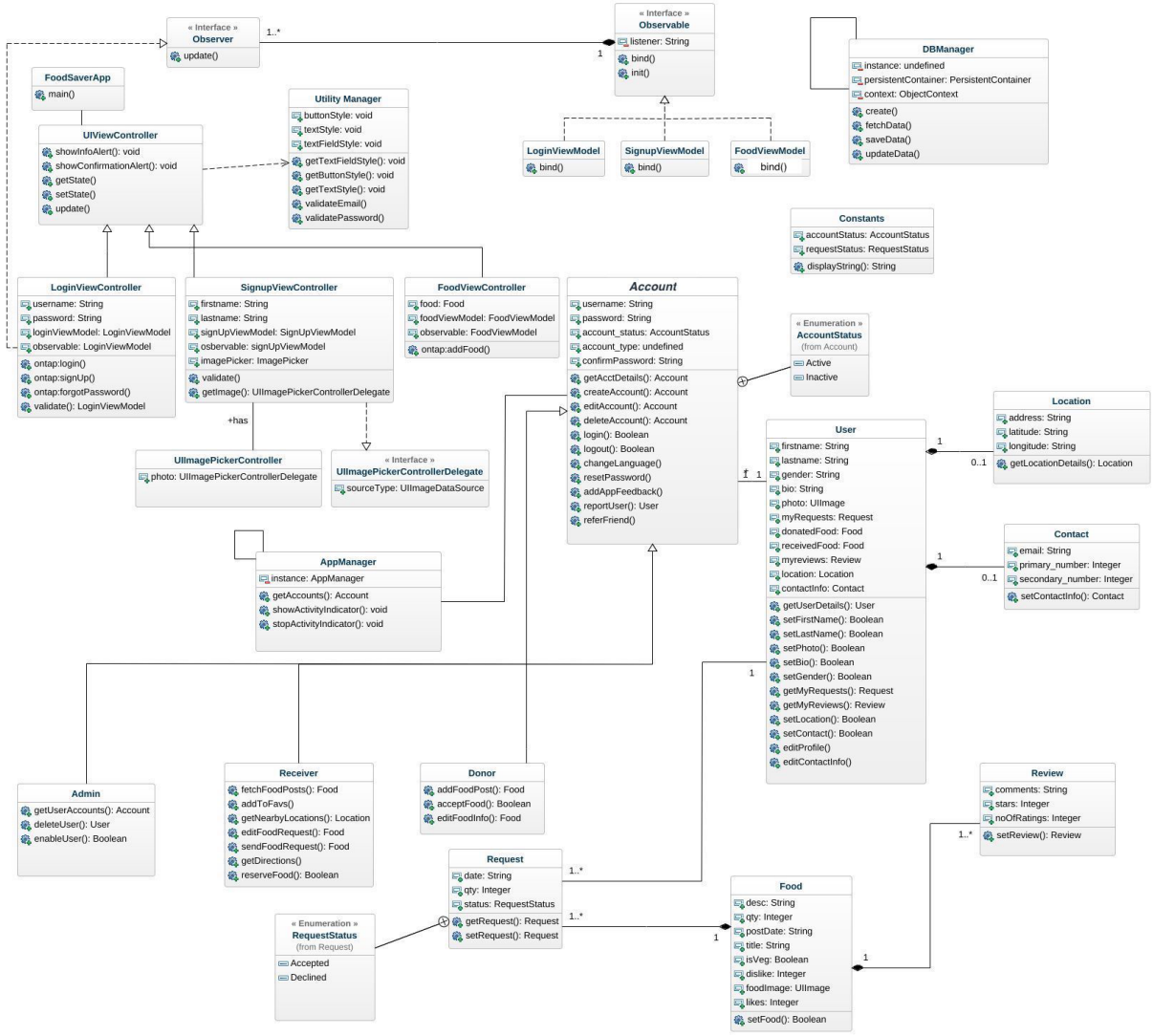
During this sprint, we have made few changes to our UX design. The development team first started with setting up the Database. They have written classes for database models. Also, they developed Login Screen and Signup Screen.

4.3.2 Design

4.3.2.1 Class Diagram

The class diagram attached shows few modifications to our existing system. As seen in the diagram, there are additional things as follows:

Observer – LoginViewController, SignupViewController and FoodViewController are the observers who observe the changes – if there is any change made in LoginViewModel, SignupViewModel and FoodViewModel respectively.
Observables - LoginViewModel, SignupViewModel and FoodViewModel – these are being observed by the observers – if there is any change in this viewModels, the obersvers get notified.
DBManager – Singleton class – which takes Persistent container and context as attributes
PersistentContainer – Default persistent store (storage) used in CoreData (Swift package for local storage) – in which managed object will be stored – stores the file in xml, binary, and SQLite.
Context – manages the data object context.
UIImagePickerController – manages the system's interface fir taking pictures and choosing image from the gallery – which takes and runs the delegate
UIImagePickerDelegate – An interface with sourceType as attribute – sourceType can be camera or gallery
SignupViewController – has a UIImagePickerController and UIImagePickerDelegate



FoodSaver: Class Diagram 2

4.4 Deliverable 3

4.4.1 Description

During this sprint, our development team completed Food details and food review screens. We added filter and sort functionality in our food view screen. We handled food requests from both donor and receiver perspective. We added a few custom delegates in our FoodViewController and MoreOptionsViewController classes. We did initial manual testing on our product and fixed few bugs that we encountered.

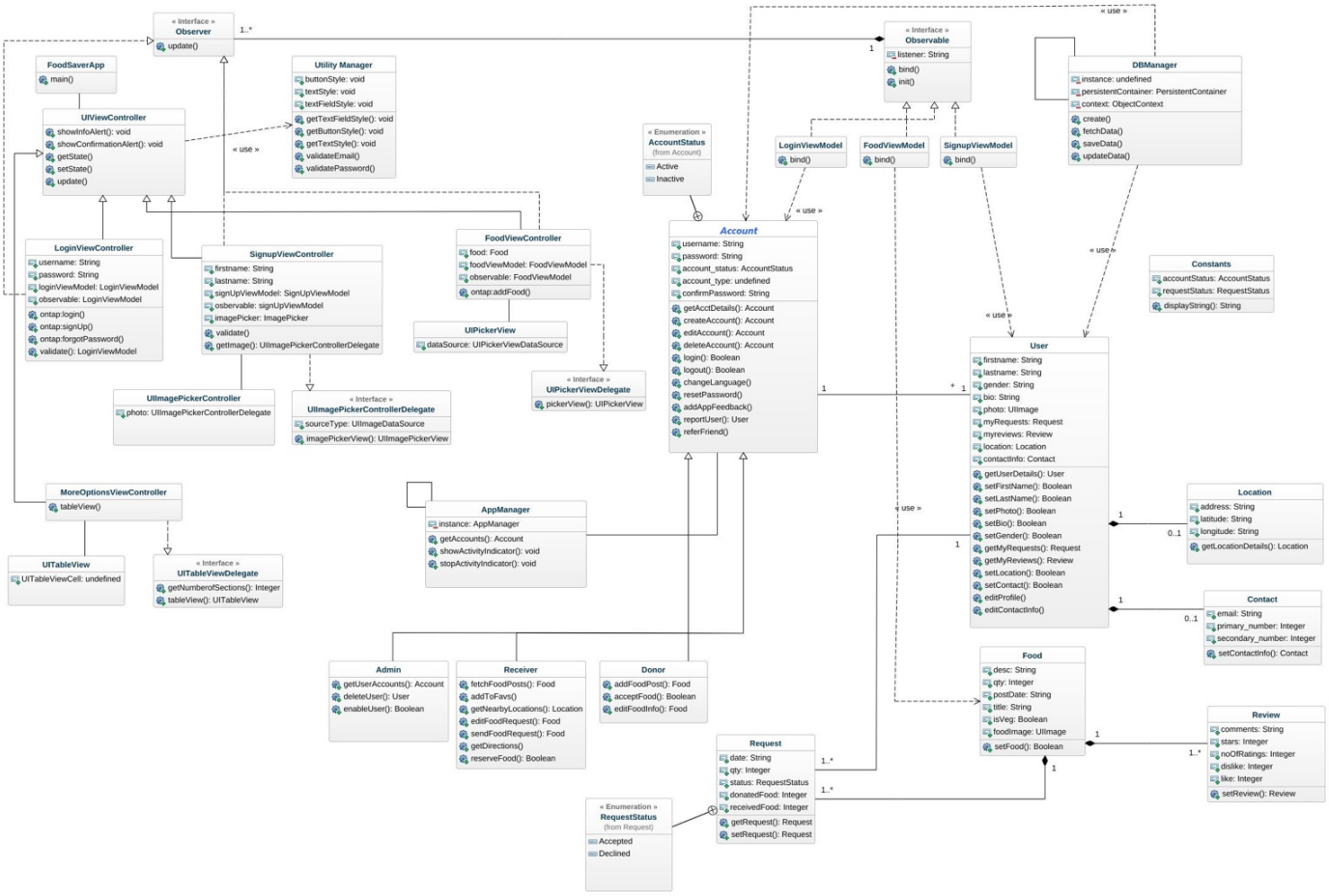
FoodSaver repo link: https://github.com/it426fall/foodsaver.git

4.4.2 Design

4.3.2.1 Class Diagram

The class diagram attached shows few modifications to our existing system. As seen in the diagram, there are additional things as follows:

MoreOptionsViewController – Controller class which handles additional functionalities like sharing the app details, etc.
UITableView – MoreOptionsViewController has UITableView which presents the data using several rows in a single column like a vertical view.
UITabelViewDelegate – MoreOptionsViewController uses this delegate interface to perform actions in a table view.
UIPickerView – Gives the option for the developer to show one or more set of values.
UIPickerViewDelegate - The interface for a picker view's delegate.



FoodSaver: Class Diagram 3

4.5 Deliverable 4

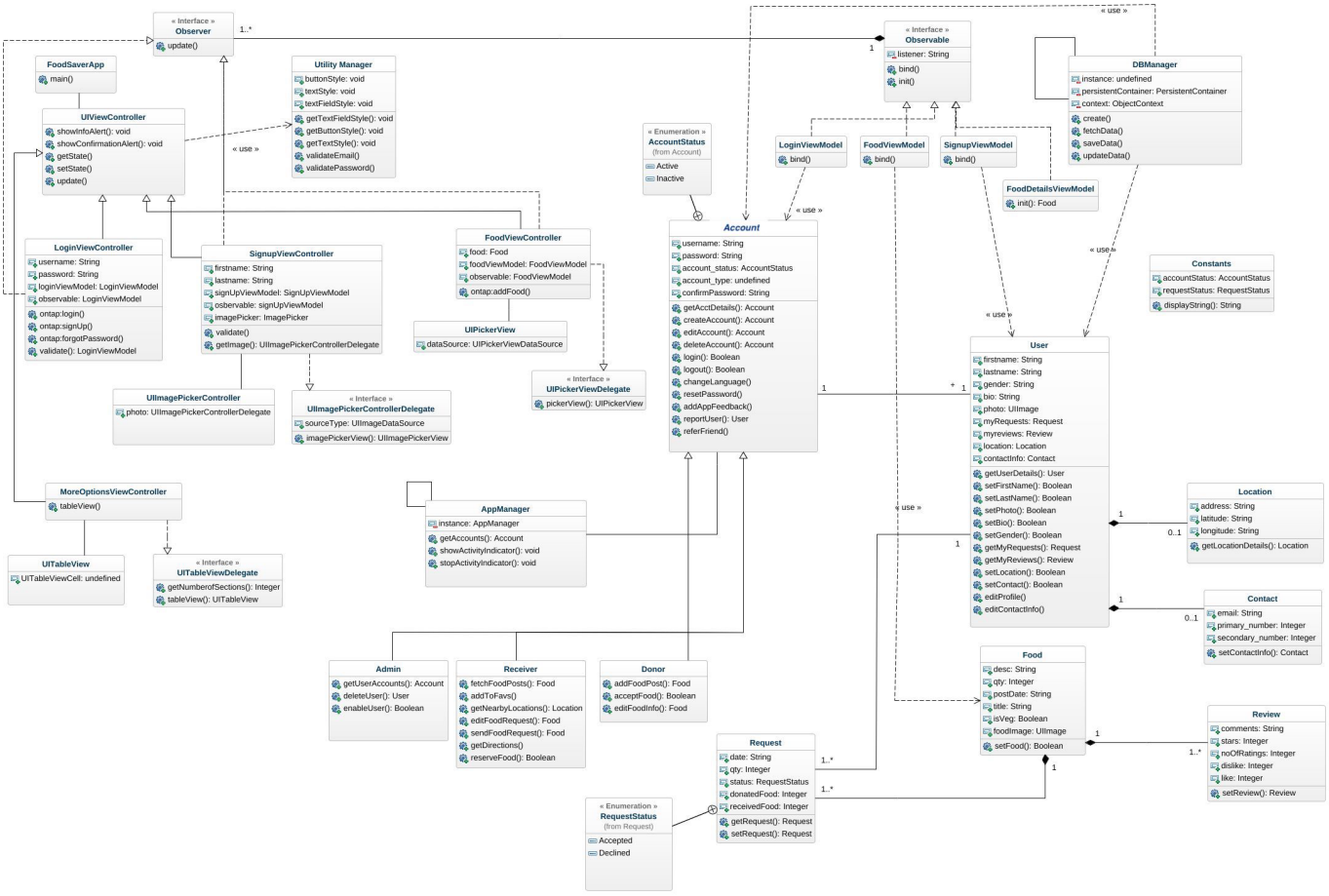
4.5.1 Description

During this sprint, our development team implemented user profile screen and favorites functionality. Also, made a few changes to our UI design.

FoodSaver repo link: https://github.com/it426fall/foodsaver.git

4.5.2 Design

4.3.2.1 Class Diagram



FoodSaver: Class Diagram 4