Black Bear Foodshare

Critical Design Review Document.

Olive Food Solutions
Dr. Scott Marzilli
Team Members: Dumas Wesley, Kaulenas Corey, Moody-Broen Makai, Sima Denis, Sholler Jakob

Executive Summary

This project was proposed by Dr. T. Scott Marzilli with the goal of creating a judgment-free platform that allows students to easily access surplus food from campus events. Many events at the University of Maine generate leftover food that often goes unused, and this project aims to reduce waste while supporting students who may benefit from additional food access. Our team's objective is to develop a mobile application that enables users to share information about available leftover food and quickly find it across campus.

Over the course of this semester, we have carefully identified and documented the project requirements, designed the overall system architecture, and developed the user interface with accessibility and ease of use in mind. In addition, we have built a small working prototype that demonstrates the core functionality of the client. At this stage, the project is well prepared for full-time development and testing next semester, and it remains on schedule with all established deadlines.

Preface

The Black Bear Foodshare application was designed with the special circumstances of food insecurity in college students in mind, and how significant the effect of food insecurity can be on a student's ability to perform in college. Food insecurity is often viewed with stigma, so the application was designed in such a way that allows anyone to use it, so that no one has to put themselves out there to get access to more nutrition.Significant preparation has been put into ensuring that the application has as low of a barrier to entry as possible for both the hosts and the users. We wanted to ensure that a host creating a new foodshare took as little time as possible to make it easy for the hosts to help people. We also wanted to ensure that the barrier of use for the users were low as well, making it easy to log in to the application, and easy to view each foodshare and the associated foods and allergens present. The hope for the Black Bear Foodshare application is that it will serve as a practical tool that allows for students to get access to more food with a low barrier of use that allows all students the ability to use it. With the low barriers of use it will allow students who are afraid of reaching out for help the ability to get help and to stay full.

Table of Contents

List of Figures

List of Tables

1. Introduction

1.1 Background

Food insecurity and food waste are pressing issues on college campuses. The Black Bear Foodshare project was initiated by Dr. T. Scott Marzilli to create a judgment-free platform where students can access surplus food from campus events, thereby reducing waste and supporting those in need.

1.2 Purpose

This CDRD provides a comprehensive review of the system design, ensuring that all requirements are met and that the proposed solution is feasible, testable, and ready for implementation.

1.3 Objectives.

Confirm alignment between SRS, SDD, and UIDD
Validate technical architecture and UI/UX design
Establish a clear testing and deployment plan
Secure stakeholder sign-off before development begins

2. Method

We're building this step-by-step. First, we listened and wrote down what the app should do (SRS). Then we figured out how to make it happen (SDD). After that, we sketched what it should look like (UIDD). Now, in this document, we're putting it all together to make sure nothing's missing. We meet every week, track tasks in GitHub, and run everything by our client along the way.

3. Design

Here's how the app is put together:

- Frontend: iOS app built in Swift. Clean, familiar, and easy to navigate.
- Backend: Python scripts that handle posts, users, photos, and filters.
- Database: Firestore fast, scalable, and good with real-time updates.
- Server: Hosted on Digital Ocean. Reliable and affordable.

We're using the Model-View-Controller (MVC) pattern to keep things organized. The design supports everything from posting food to filtering by allergies, and it's built to grow if the university wants to expand it later.

4. Equipment

This Section documents all tools and platforms used in the development through deployment of the Black Bear Foodshare IOS App.

4.1 Development tools.

- Mac Computer / Mac OS
  - Primary device(s) used for development. Required for creation of IOS applications. All Swift development, ios upload and verification has to be done on Mac OS.
- Xcode
  - Apple's Official Integrated development environment for IOS applications. Utilized to build, test, and debug IOS applications.
- Swift

    - Apple's Official programming language for iOS, iPadOS, macOS, tvOS, and watchOS.

- GitHub/ Git
    - Utilized for central repository for Our codebase and for its version control. Gits Version Control ensures that the team can keep track of any changes made to the central repository, in addition branching allows for easier collaboration.

4.2 Backend tools

- Docker
    - Utilized for its containerization.
- Digital Ocean
    - Hosts the backend and provides stable and affordable server infrastructure. Utilized for its

4.3 Database tools

- Firebase
    - Utilized for its scalability and affordability, with support of 50k monthly active users and 1gb of storage, in addition it does also support Google pricing for cloud storage if more data is needed.

4.4 Api's

- Apple Development API
    - Mapkit
    - User notification
- Firebase
    - Firestore
    - Authentication

4.5 Documentation / Communication tools

- Google Docs
    - The Main Application used for writing of documents for this project.
- Github Project Board
    - Utilized for Task breakdown assigning and tracking.
- Discord
    - Main Application used for Communication throughout the process.

5. Deliverables, Milestones, and Timeline.

This Section outlines all of the deliverables and milestones Olive Food Solutions have achieved on the project of Black Bear Foodshare. It also provides a Timeline of the project up to current Day.

5.1 Deliverables

Title of Deliverable	Purpose of Deliverable	Date Delivered	How it was Delivered (Hard copy? Zip? Git?)
System Requirement Specification	Defines Functional and Non-functional Requirements for Entire Project.	Digitally : 10/20/25 Physically: TBD	Digitally: pdf, docx Hard copy
System Design Document	Provides the architecture and a detailed design template for implementation	Digitally: 11/17/25 Physically: TBD	Digitally: pdf, docx Hard copy
User Interface Design Document	Specifies the layout and interactions of the projects User Interface	Digitally: 12/3/25 Physically: TBD	Digitally: pdf, docx Hard copy
Critical Design Review Document		Digitally: 12/17/25	Digitally: pdf, docx
User Manual	Guides Users on installing and utilizing the system effectively	TBD (Spring Semester)	Digitally: pdf, docx Hard copy
Administrator Manual	Instructs administrators on Administrative duties.	TBD (Spring Semester)	Digitally: pdf, docx Hard copy
All Biweekly Status Reports (1-5)	Tracks Progress,	Digitally 12/19/25 Physically : TBD	Digitally: pdf, docx Hard copy
(All Source code)	Hand off Project to client, so that they can maintain with other developers in future	TBD (Spring Semester)	Git

Title of Deliverable	Purpose of Deliverable	Date Delivered	How it was Delivered (Hard copy? Zip? Git?)
	years.		
The Executable program	Executable version of the Project for testing and final use.	TBD (Spring Semester)	Git
(All other software required for installation and execution of delivered program)	Lists dependencies and tools required for the project to run successfully.	TBD (Spring Semester)	Git

5.2 Milestones

Milestone	**Description**	**Date Completed**
Initial Client Meeting	**First Meeting Between Client and Team**	**10/8/25**
System Requirement Document	**Completion of Requirements Document**	**10/20/25**
System Requirement Review	**Review with Client of Requirements document**	**10/22/25**
System Design Document	**Completion of Design Document**	**11/17/25**
System Design Review	**System Design Review with Client**	**11/5/25**
User Interface Design Document	**User Interface Design Complete**	**12/3/25**
User Interface Review	**User Interface Review with Client**	**12/3/25**
Critical Design Document Completion	**Critical Design Completed**	**12/19/25**
Critical Design Review	**Critical Design Review with Client and Class**	**12/8/25**
Demo / Prototype of application	**Demo/ Prototype Completed**	**12/7/25**

Final Signing of CDRD	**When the CDRD is signed**	**12/19/25**

5.3 Timeline

Fall Semester

- Weeks 1–4: Discovery & Requirements
    - During this period, the team met with the client to understand the project more, and gauge the scope of the project, its constraints, and the goal of the project.
    - Communications was also established on Discord as the main communication application.
    - Roles were assigned of who would be client liaison, meeting times were determined, received proper channels to setup meetings with client
    - Github was also initialized for future use in the project.
    - Used all tools together along with meeting information to create the system Requirements Document.
    - Reviewed Requirements Document with the Client.
- Weeks 5–8: Prototyping & Architecture
    - Utilizing the project proposal to help guide the System Design Document, outlining the system and all of its components.
    - Researched tools to meet project proposal requirements.
    - Meeting with Client to Review System Design Document
    - Development of User Interface begun
- Weeks 9–12: Initial Implementation
    - Finished User Interface Design
    - Held User Interface Design meeting with Client
    - Created Demo For Critical Design Review
    - Prepared to present project for Critical Design Review
    - Held Presentation on Blackbear Foodshare

Spring Semester : This hasn’t happened yet.

- Weeks 13–16: Hardening & Analytics
    -
- Weeks 17–20: Pilot Prep & UAT
    -
- Weeks 21–24: Pilot & Reporting
    - Application has released

6. Conclusions

Overall, our team has completed all planning required for the project to start coding and testing. We know our requirements, design, and interface. We also have prototypes ready, one of which has been reviewed. For our next steps, we need to start prototyping more of the client, and start building the server and database. Additionally, we need to have tests for our product running. Overall, we are on track to complete the project.

7. Recommendations

Based on our conceptual designs and identified requirements, there are many recommendations proposed to guide the development and deployment of the Black Bear Foodshare application. First, it is recommended that the application is initially released and tested by a small number of students, to allow for the identification of usability issues and bugs. The feedback generated from this should be used to improve upon the system before its official release. Second, it is recommended that the application goes through usability testing before release to ensure that the navigation is intuitive and that the core features can be easily accessed by users. Finally, the data collected from the application should be analyzed to determine the impact it has upon food accessibility, and to monitor its usage among students. The information generated should be used to evaluate the effectiveness of the system and its alignment with the project's goals.

Bibliography

Olive Food Solutions. (2025, October). System Requirements Document.
System Requirements Specification

Olive Food Solutions (2025, November) System Design Document
System Design Document

Olive Food Solutions (2025, November) User Interface Document
User Interface Design Document.docx

Olive Food Solutions (2025, December) Critical Design Review Document
Critical Design Review Document
Olive Food Solutions (2025, October) Team Olive Github
https://github.com/MoodyMakai/OliveCapstone

Olive Food Solutions (2025, November) BlackBear Foodshare Github
https://github.com/Athenaphilia/BlackBearFoodShare

Apple Inc. (n.d.). *Swift*. Apple Developer Documentation.
https://developer.apple.com/documentation/swift

Apple Inc. (n.d.). Xcode documentation. Apple Developer Documentation.
https://developer.apple.com/xcode/

Google. (n.d.). Cloud Firestore documentation. Firebase.
https://firebase.google.com/docs/firestore

DigitalOcean. (n.d.). DigitalOcean documentation.
https://docs.digitalocean.com/

Docker Inc. (n.d.). Docker documentation.
https://docs.docker.com/

Python Foundation. (n.d.). Welcome to flask. Welcome to Flask - Flask Documentation (3.1.x).
https://flask.palletsprojects.com/en/stable/

Acknowledgement

We would like to thank Dr T. Scott Marzilli and his team for bringing this project to us and providing a clear vision to work towards.

Appendix A - System Requirement Specification.

Black Bear Foodshare

System Requirement Specification

Olive Food Solutions

Dr. Scott Marzilli
Team Members: Dumas Wesley, Kaulenas Corey, Moody-Broen Makai, Sima Denis, Sholler Jakob

Black Bear FoodShare
System Requirements Specification

Table of Contents

1. Introduction

This is (Black Bear FoodShare) capstone project for Dr. Scott Marzilli, in partial fulfillment of the computer science BS degree for the University of Maine. The purpose of this project is to develop an application for various food servers on campus to easily post excess food to the students of UMaine. These posts will include the food, photos of the food, the location and pictures of the location. The goal of this project is to reduce waste and increase sustainability on campus, while also complying with regulations to make sure the excess food is safe to consume.

1.1 Purpose of this Document

This document specifies the required functions and metrics of the Black Bear FoodShare application. The SRS is intended to create a foundation for both the Olive Food Solutions (OFS) and the UMaine Dining team to agree upon and work from. This document seeks to specify the functional requirements (what this application is meant to do and how it will do it), the non-functional requirements (performance speed, user capacity, etc.), and provide the groundwork for UI, deliverables, and open issues.

1.2 References

Food Safety and business - cooperative extension: Food & health - university of maine cooperative extension. Cooperative Extension: Food & Health. (2025, June 22). https://extension.umaine.edu/food-health/food-safety/

Fowler, M. (2018). UML distilled : a brief guide to the standard object modeling language. Addison-Wesley.

1.3 Purpose of the Product

Dr. Marzilli has 12 years of experience in student success and innovation. During this time, Dr. Marzilli noticed two things. First was the excess food waste from various hosts on the campus, and second there are students who don’t eat as much as they should. This birthed the idea of Black Bear FoodShare. The goal of Black Bear FoodShare is to provide the hosts on campus an easy way to post the excess food they have to the students of the university. While at the same time allowing any student who would like to be able to sign up for push notifications to see these posts as they are created to obtain some of the excess food. Reducing food waste, and feeding students so that they can continue to focus on their studies.

1.4 Product Scope

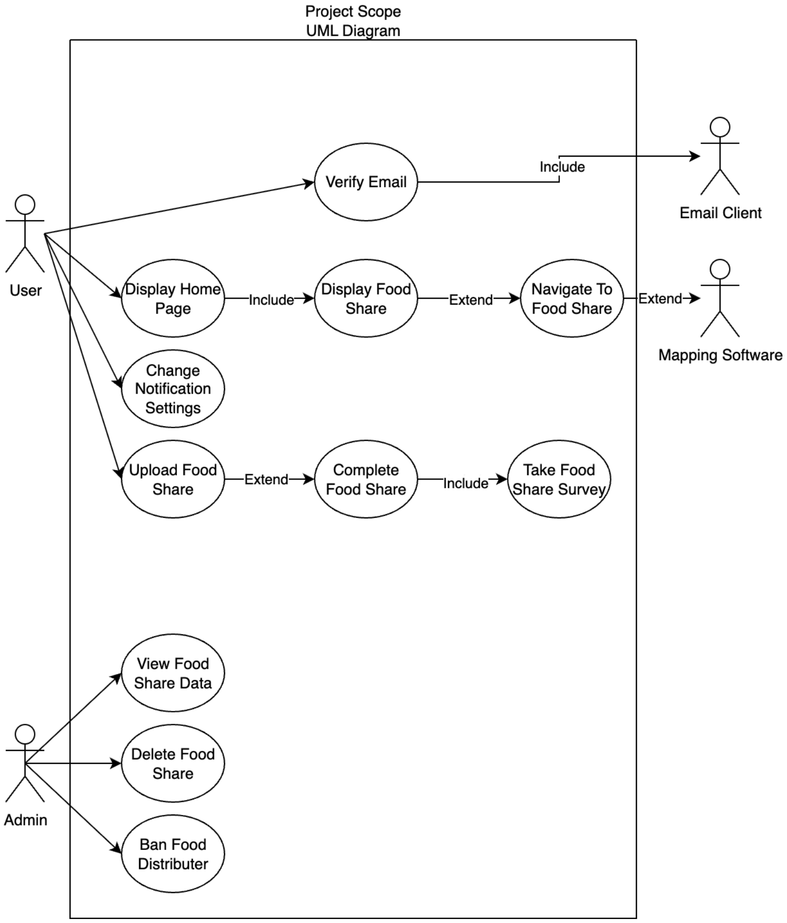
The Black Bear FoodShare application will provide a platform for verified university-affiliated food servers ("Hosts") to post announcements about excess food, including details and photographs. It will allow university students ("Users") to view these posts and opt-in to receive push or SMS notifications for new posts.

In-Scope:

- Hosts food shares for anybody with a UMaine email
- Letting hosts make posts with food and location pictures.
- Sending alerts to students for new posts.
- A short survey for hosts when they end a food share.
- The system and database that runs the app.

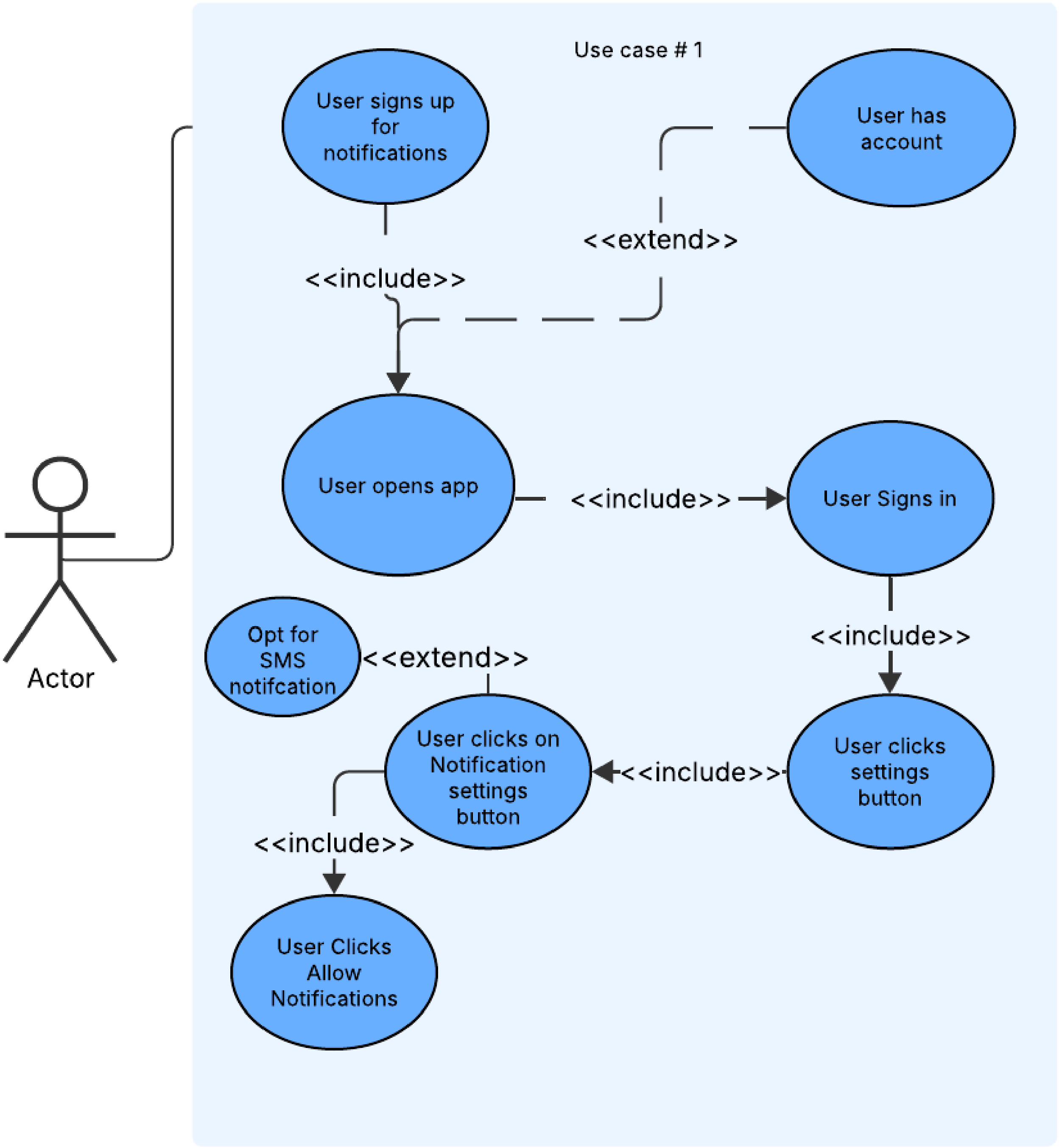
Out-of-Scope:

- No money or payments.
- No chat between users and hosts.
- No food delivery.
- No user reviews or star ratings.
- No sharing to Facebook or other apps.


Project Scope
UML Diagram
User
Verify Email
Include
Email Client
Display Home Page
Include
Display Food Share
Extend
Navigate To Food Share
Extend
Mapping Software
Change Notification Settings
Upload Food Share
Extend
Complete Food Share
Include
Take Food Share Survey
Admin
View Food Share Data
Delete Food Share
Ban Food Distributer

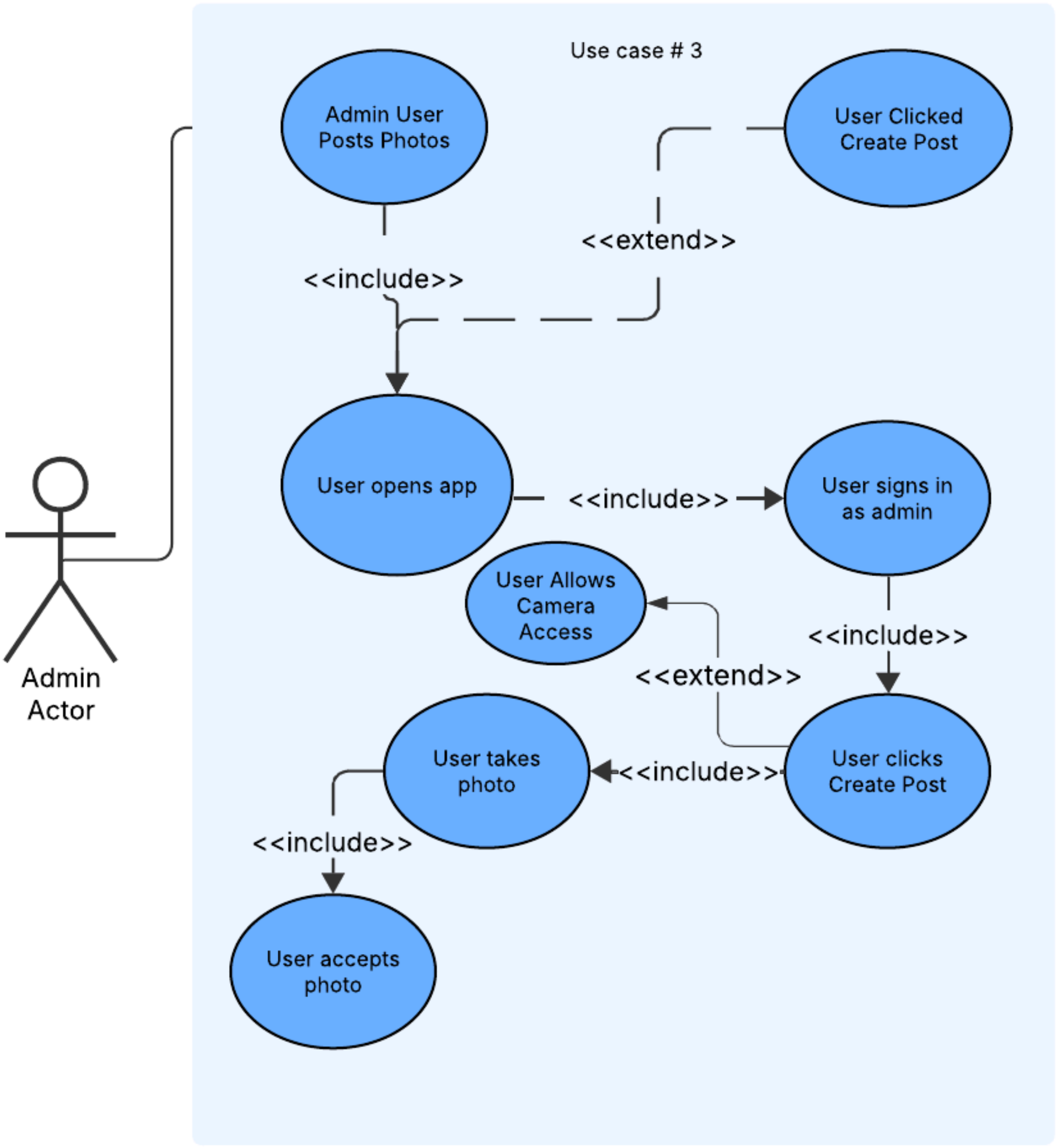
2. Functional Requirements

Number	UC-001	
Name	Users shall be able to sign up for push notifications.	
Summary	Allows the user to opt in to push notifications to get live updates on excess food being shared around campus.	
Priority	5	
Preconditions	User has the app open. User must have a verified email.	
Postconditions	User now receives either push notifications or SMS notifications.	
Primary Actor	User	
Secondary Actors	Client OS	
Trigger	User clicks on “notifications” in account settings.	
Main Scenario	**Step**	**Action**
	1	User navigates to account settings.
	2	User navigates to notification settings.
	3	User opts in for push notifications.
Extensions	**Step**	**Branching Action**
	3b	User opts in for SMS notifications. User enters their phone number.
Open Issues		


Use case # 1
User signs up for notifications
User has account
<<extend>>
<<include>>
User opens app
<<include>>
User Signs in
Actor
Opt for SMS notifcation
<<extend>>
<<include>>
User clicks on Notification settings button
<<include>>
User clicks settings button
<<include>>
User Clicks Allow Notifications

Number	UC-002	
Name	Create food share	
Summary	Allows a verified user to create a food share.	
Priority	5	
Preconditions	User has app open User must have a verified email	
Postconditions	The food share is created and viewable on the app.	
Primary Actor	User	
Secondary Actors		
Trigger	User clicks “create food share” button.	
Main Scenario	**Step**	
	1	The user is presented with inputs for “event name”, “add picture”, “end time”, “location”.
	2	The user enters this info.
	3	The user clicks “accept”.
Extensions	**Step**	
	3a	User attempts to submit without filling necessary fields, is presented with an error message.
Open Issues	N/A	

Number	UC-003	
Name	Hosts shall be able to upload and post images inside their posts.	
Summary	Allows for hosts to upload two photos into their posts; 1. a Picture of the food and 2. a picture of the location the food is at.	
Priority	4	
Preconditions	Host has app open and has verified email Create post is open	
Postconditions	Images have been added to the post.	
Primary Actor	Host	
Secondary Actors	Client OS	
Trigger	Host clicks “add photo”(either food or location).	
Main Scenario	**Step**	**Action**
	1	Host clicks “add photo”.
	2	Host is prompted and allows camera access.
	3	Host takes a photo.
	4	Host’s photo is added to post.
	5	Repeat steps 1-4 for other photo (Food or Location).
Extensions	**Step**	**Branching Action**
	2b	Host has already given permission for camera access Jump to Step 3.
Open Issues	N/A	


Use case # 3
Admin User Posts Photos
User Clicked Create Post
<<extend>>
<<include>>
User opens app
<<include>>
User signs in as admin
User Allows Camera Access
<<include>>
<<extend>>
User takes photo
<<include>>
User clicks Create Post
<<include>>
User accepts photo
Admin Actor

Number	UC-004	
Name	View current Food Shares	
Summary	Allows a user to view the current food shares around them on the homepage of the app.	
Priority	4	
Preconditions	User must have the app open and have a verified account.	
Postconditions	A list of food shares must be listed on the screen.	
Primary Actor	User	
Secondary Actors	N/A	
Trigger	The user either opens the app, or clicks the home page.	
Main Scenario	**Step**	**Action**
	1	The user navigates to home.
	2	App refreshes and User is presented with active food shares.
Extensions	**Step**	**Branching Action**
Open Issues	N/A	

Number	UC-005	
Name	Ending food shares	
Summary	When a host ends a food share, they will be asked to complete a survey before the food share is deleted.	
Priority	5	
Preconditions	A host must have created a food share, and has clicked on the “end food share” button. The host must also have a verified email.	
Postconditions	The food share must be deleted, and the host must have taken a survey.	
Primary Actor	User	
Secondary Actors	N/A	
Trigger	The host clicks the “end food share” button.	
Main Scenario	**Step**	**Action**
	1	The host clicks the “end event” button.
	2	The host is presented with survey
	3	Host answers roughly how many students attended the event, and roughly how much food was taken
	4	Host submits survey
	6	Host is presented with "Successfully deleted message”
Extensions	**Step**	**Branching Action**
	2a	User submits without completing survey
Open Issues	N/A	

Number	UC-006
Name	User Verifies Email
Summary	User gets emailed to verify they are a UMaine student.
Priority	4

Preconditions	User has opened the app and the current device is not linked to a verified Email.	
Postconditions	User's device is linked to a verified account.	
Primary Actor	User	
Secondary Actors	Email Client	
Trigger	User opens the app for the first time.	
Main Scenario	**Step**	**Action**
	1	Client shows email input.
	2	User inputs email.
	3	Client sends request to server with email.
	4	Server sends verification email.
	5	User opens email in email client.
	7	User verifies through HTTPS link.
	8	Client sends request to server to verify.
	9	Server marks email as verified.
	10	Server sends confirmation information to client.
Extensions	**Step**	**Branching Action**
	9a	If the email isn't in the maine.edu domain, return an invalid email error.
Open Issues		

Number	UC-007	
Name	User Filters Food Shares by Allergy.	
Summary	User applies filters to remove food shares containing potential allergens.	
Priority	3	
Preconditions	User has verified account and opened homepage.	
Postconditions	User is presented with only food shares which meet criteria.	
Primary Actor	User	
Secondary Actors	N/A	
Trigger	User navigates to home	
Main Scenario	**Step**	**Action**
	1	User clicks "drop down menu" button.
	2	User clicks "filter" option.
	3	User clicks which criteria they would like to filter by.
	4	User submits.
	5	Page refreshes.
	7	User is presented with matching food shares.
Extensions	**Step**	**Branching Action**
	4a	If no food shares match criteria, return "No available food shares".
Open Issues		

Number	UC-008
Name	Host Adds Allergen warning.
Summary	User selects allergens included in food share before posting.

Priority	3	
Preconditions	Host is in the process of creating a food share.	
Postconditions	Included allergens are listed in food share.	
Primary Actor	Host	
Secondary Actors	N/A	
Trigger	Host has created a food share and clicks “post”.	
Main Scenario	**Step**	**Action**
	1	Host clicks on “edit food share”
	2	Client sends request to server asking for food share info
	3	Server receives and returns food share
	4	Client presents options to change the allergens and options in food share
	5	Host inputs information and clicks “confirm”
	6	Client sends request to edit food share with required credentials
	7	Server receives request, verifies it, and edits the database accordingly
Extensions	**Step**	**Branching Action**
	7a	Request is invalid, so there server returns an error code
Open Issues		

Number	UC-009	
Name	User accesses location map service	
Summary	User clicks on “food share address” which navigates the user to their default mapping software.	
Priority	2	
Preconditions	User has clicked on food share.	
Postconditions	User is viewing mapping app which shows steps to reach the food share’s location.	
Primary Actor	User	
Secondary Actors	Mapping client	
Trigger	User accesses food share location.	
Main Scenario	**Step**	**Action**
	1	User clicks on hyperlinked address in food share information.
	2	User is navigated to the system's default mapping software.
	3	User is presented with the desired location in mapping software.
	4	User clicks “accept”.
	5	User is presented with directions for the destination.
Extensions	**Step**	**Branching Action**
	4a	The user does not accept and returns to the Foodshare application.
Open Issues		

Number	UC-010	
Name	View Food Share	
Summary	User views entire food share when clicking on it in the home page	
Priority	3	
Preconditions	User is on the home page	
Postconditions	User sees food share in its own page	
Primary Actor	User	
Secondary Actors		
Trigger	User clicks on food share	
Main Scenario	**Step**	**Action**
	1	Client sends request to server for food share
	2	Server returns food share data
	3	Client displays food share
Extensions	**Step**	**Branching Action**
Open Issues		

Number	UC-011	
Name	Admin Views Food Share Data	
Summary	Admin views the current data about all food shares, current and past	
Priority	2	
Preconditions	Admin is on the admin page with proper credentials	
Postconditions	The food share data is downloaded as a zip file	
Primary Actor	Admin	
Secondary Actors		
Trigger	Admin clicks on “export data”	
Main Scenario	**Step**	
	1	Client sends request for food share data
	2	Server packages the database and returns it
	3	Client downloads the zip file
Extensions	**Step**	
Open Issues		

Number	UC-012	
Name	Admin Deletes Food Share	
Summary	Admin deletes a specific running food share	
Priority	3	
Preconditions	Admin is on the admin page with proper credentials	
Postconditions	The food share is deleted from the database	
Primary Actor	Admin	
Secondary Actors		
Trigger	Admin clicks on “delete food share”	
Main Scenario	**Step**	
	1	Client sends request to server to delete with proper credentials

	2	Server receives and checks request before deleting it from the database
	3	Server returns response
	4	Client displays response
Extensions	**Step**	
	3a	If food share couldn’t be deleted, return error code, else success
Open Issues		

Number	UC-013	
Name	Admin Bans Food Distributor	
Summary	Admin bans food distributor from creating more food shares	
Priority	4	
Preconditions	Admin is on the admin page with proper credentials	
Postconditions	The account is banned and their food shares are marked as inactive	
Primary Actor	Admin	
Secondary Actors		
Trigger	Admin clicks on “ban food distributor” for a recent food share	
Main Scenario	**Step**	
	1	Client displays confirmation of ban
	2	Admin confirm or deny
	3	Client sends request to server for deletion with the credentials
	4	Server marks users as banned, and marks all of their food shares for deletion, then returns response
	5	Client displays response
Extensions	**Step**	
	2a	If admin denies, stop process
	4a	If an error occurred, return error code, else success
Open Issues		

1.1 Use Case Tests

Number	UCT-001
Name	Push Notification Test.
Test For	UC-001
Step	**Description**
1	Start server, and set up dummy food share.
2	Create a verified account with no notifications set up.
3	Call functions to send notification to client.
4	Set account notification preference to “yes”.
5	Call functions to send notification to client.
Test on Step	**Description**
3a	Notification should fail, function return error message.

5a	Function returns notification with food share info.

Number	UCT-002
Name	Create Food Share Test
Test For	UC-002
Step	**Description**
1	Start server.
2	Call create food share function with various permutations.
2.1	Completely valid food share.
2.2	End time less than current time.
2.3	Empty name.
Test on Step	**Description**
2.1a	Food share should be in the database, and the function returns success.
2.2-2.3a	Function returns an error specifying what is wrong.

Number	UCT-003
Name	Add Picture to Food Share Server.
Test For	UC-003
Step	**Description**
1	Start server.
2	Call create food share function with picture.
Test on Step	**Description**
2.1a	Food share should be in the database, with a link to photo in the database.
2.1b	The photo should be in the database, with expiration date 2 weeks from current date.

Number	UCT-004
Name	View Food Shares
Test For	UC-004
Step	**Description**
1	Start Server
2	Create multiple dummy food shares
Test on Step	**Description**
2a	Check to see that food shares are returned when calling the “get all food shares” function

Number	UCT-005
Name	Check if food shares end
Test For	UC-005
Step	**Description**
1	Start Server
2	Create dummy account 1 and 2

3	Create food share linked to dummy account 1
4	Attempt to end the food share as both accounts
Test on Step	**Description**
4a	Should fail when dummy account 2 attempts it
4b	Should succeed, with no food share displayed, (but still in database) when dummy account 1 tries

Number	UCT-006
Name	Server Creates Email
Test For	UC-006
Step	**Description**
1	Start Server
2	Call function to create email with dummy account
3	Call function to verify account
Test on Step	**Description**
2a	See if function returns an email format that links to the proper validation address
3a	Check if the account is now verified

Number	UCT-007
Name	Filter Food Shares
Test For	UC-007
Step	**Description**
1	Start Server
2	Create food shares that have different allergens
3	Call function to filter food shares by every allergen added in step 2
Test on Step	**Description**
3a	Check the contents of the return for every permutation such that they only display the contents with respect to the filter

Number	UCT-008
Name	Add Allergy
Test For	UC-008
Step	**Description**
1	Start Server
2	Create dummy food shares and user 1
3	Call function to edit food share with additional allergens
Test on Step	**Description**
3a	Check that the changes are reflected properly within the database, as well when the food share is viewed individually.

Number	UCT-009
Name	Navigates to Food Share
Test For	UC-009

Step	**Description**
1	Start Server and Client
2	Create dummy food share with location
3	Have client call navigate to food share function
Test on Step	**Description**
3a	Check that client is linked to the proper google/apple maps location

Number	UCT-010
Name	View food share
Test For	UC-010
Step	**Description**
1	Start Server
2	Create dummy food share
3	Call function to view food share
Test on Step	**Description**
3a	Check that food share is in proper format, and has the correct data

Number	UCT-011
Name	Admin Get Food Share Data
Test For	UC-011
Step	**Description**
1	Start Server
2	Create dummy food shares
3	Call function to get food share data
Test on Step	**Description**
3a	Check that all food shares are in the data returned, and with correct data, as well as a zip was returned

Number	UCT-012
Name	Admin Delete Food Share
Test For	UC-012
Step	**Description**
1	Start Server
2	Create dummy food share
3	Call function to delete food share
Test on Step	**Description**
3a	Check that food share is no longer in the database

Number	UCT-013
Name	Admin Ban User
Test For	UC-013
Step	**Description**
1	Start Server

2	Create dummy verified user and food shares linked to them
3	Call function to ban user
Test on Step	**Description**
3a	Check that user is marked as banned, and that their food shares are no longer active

3. Non Functional Requirements

Number	Requirement	Test	Priority (1-5)
1.	Black Bear Foodshare shall be able to sustain a user base of 2,000 individuals without suffering in performance.	Use a load testing tool to measure performance with a large number of simulated users.	4
2.	All food postings shall be viewable with a maximum latency of 5 seconds after posting.	Measure latency after a post across several devices	3
3.	Map pins shall display relative location on map with building-level accuracy	Compare locations displayed on an application map to another mapping software, such as Google Maps.	5
4.	It should take a user 30 seconds or less to post a food share.	Create a dummy actor that takes an average amount of time to input info, then time that actor as it “creates” a food share.	2
5.	The system shall not allow unverified users to post food shares.	Have a normal user and a verified user try to post a food share, and see if it works.	5
6.	The system shall have an uptime of 95% per month.	Verify that the uptime is 95% over a monitoring period.	3
7.	The system shall provide directions to the location of the food	Measure response time from when the user requests directions to	5

	with Google Maps within 3 seconds.	when the map opens.	
8.	Only authenticated admins shall be able to access the admin dashboard.	Verify that the admin dashboard cannot be reached without an authenticated account	3
9.	The directions button shall be available within one click from the food event page.	Verify that a user can get directions within one click.	5
10.	The posts shall be automatically removed after the expiry time without manual intervention.	Verify that posts are removed after the expiry time.	2
11.	The app shall load the home screen in under 2 seconds under normal network conditions.	Measure average load time across multiple devices.	4
12.	The system shall ensure data backups occur daily for user and food share data.	Verify backup logs when simulating days	4

4. User Interface

Please see User Interface Design Document for Black Bear FoodShare.

5. Deliverables

Title of Deliverable	Date Delivered	How it was Delivered (Hard copy? Zip? Git?)
System Requirement Specification	Digitally : 10/20/25 Physically: TBD	Digitally: pdf, docx Hard copy
System Design Document	Digitally: 11/17/25 Physically: TBD	Digitally: pdf, docx Hard copy
User Interface Design Document	Digitally: 12/3/25 Physically: TBD	Digitally: pdf, docx Hard copy

Critical Design Review Document	Digitally: 12/17/25	Digitally: pdf, docx
User Manual	TBD (Spring Semester)	Digitally: pdf, docx Hard copy
Administrator Manual	TBD (Spring Semester)	Digitally: pdf, docx Hard copy
All Biweekly Status Reports (1-5)	Digitally 12/19/25 Physically : TBD	Digitally: pdf, docx Hard copy
(All Source code)	TBD (Spring Semester)	Git
The Executable program	TBD (Spring Semester)	Git
(All other software required for installation and execution of delivered program)	TBD (Spring Semester)	Git

6. Open Issues

1. What questions will be on the "end food share" survey?
2. What technology will we use for the app and notifications?
3. What is the exact UI design of the app?
4. What architecture will we use for the internal design of our app?

Appendix A - Agreement Between Customer and Contractor

The customer and team agree that the document represents a finalized version that meets all agreed upon criteria and requirements. Both parties have thoroughly reviewed the document and all of its contents and agree that the information and structure meet the standards of the requirements. Any feedback or revisions have been addressed and remedied. The document is considered complete and ready for use.

If changes to the System Requirements Specification are required, the team will begin by preparing a draft of the document with the changes. After the draft has been prepared, each team member will review the draft and note any problem with it they might have. Once all the problems have been addressed, the team will review the document once more and sign off on it, to show that they are satisfied with the state of the document. Once that has been completed, we will submit the draft to the client for review and approval.

Wesley Dumas
Signature: Wesley Dumas________________________Date: 10/29/25

Corey Kaulenas
Signature: Corey Aras Kaulenas_____________________Date: 10/29/25

Makai Moody-Broen
Signature: Makai Moody-Broen______________________Date: 10/29/25

Denis Sima
Signature: Denis Sima____________________________Date: 10/29/25
DS

Jakob Sholler
Signature: Jakob Sholler________________________Date:10/29/25

Dr. Scott Marzilli
Signature: ______________________________________Date: __10/29/2025_______
Comments:

Appendix B - Team Review Sign Off

All team members have thoroughly reviewed this document and reached full agreement on all of its content. No major concerns have been raised, and any minor concerns or clarifications raised by individuals are noted in the comments below their signature. Additionally, all members of the team have agreed upon the formatting and presentation of this document with no major complaints.

Wesley Dumas
Signature: Wesley Dumas________________________Date:10/29/25
Comments:

Corey Kaulenas
Signature: Corey Aras Kaulenas________________________Date: 10/29/25
Comments:

Makai Moody-Broen
Signature: Makai Moody-Broen______________________Date: 10/29/25
Comments:

Denis Sima
Signature: Denis Sima__________________________Date: 10/29/25
Comments:

Jakob Sholler
Signature: Jakob Sholler_______________________________Date:10/29/25
Comments:

Appendix C - Document Contributions

Sholler Jakob, Contributions:

- Introduction
- Deliverable Table Outline
- Functional requirements 1, 2, 6
- Purpose of Product

Dumas Wesley, Contributions:

- Non-functional requirements 1, 2, 6, 7, 8, 9, 10
- Appendix A, B

Moody-Broen Makai, Contributions:

- Purpose of Document
- Non-functional requirements 1, 2, 3
- Functional requirements 6, 7, 8, 9

Kaulenas Corey, Contributions:

- Functional requirements 2, 4, 5, 10, 11, 12, 13
- Non-functional requirements 4, 5, 6
- Product Scope UML Diagram
- All Functional requirement use case tests

Sima Denis, Contributions:

- 1.2 References
- 1.4 Product Scope
- 6. Open Issues
- Functional requirements 5
- Non-functional requirements 12

Appendix B - System Design Document

Black Bear Foodshare

System Design Document

Olive Food Solutions

Dr. Scott Marzilli

Team Members: Dumas Wesley, Kaulenas Corey, Moody-Broen Makai, Sima Denis, Sholler Jakob

Black Bear Foodshare
System Design Document

Table of Contents

1 Introduction

This is a capstone project for Dr. T Scott Marzilli, Associate Provost of Student Success & Innovation, in partial fulfillment of the computer science BS degree for the University of Maine. Dr. Mazilli places great emphasis on the importance of providing the necessary resources and opportunities for students to succeed at the University of Maine. The impetus of this project comes from the passion Dr. Marzilli and our team share in providing judgement free opportunities for students dealing with food insecurity, as well as furthering the elimination of food waste on our campus. The purpose of this project is to develop an application to allow hosts of catered events to easily inform students of excess food on campus. These posts will include the food, photos of the food, and the location. The goal of this project is to reduce waste and increase sustainability on campus, while also complying with regulations to make sure the excess food is safe to consume.

1.1 Purpose of This Document

The purpose of this document is to inform our team, client, and interested parties of the intended design and functionality of the Black Bear Foodshare application.

1.2 References

- Olive Food Solutions, System Requirements Specification
  System Requirements Specification

2 System Architecture

This section of the System Design Document contains section 2.1 Architectural Design, and section 2.2, Decomposition Description. This section outlines the necessary technologies that will be implemented in the design of the Foodshare application. Section 2.2 specifies the functions and methods that will be used to develop the application.

2.1 Architectural Design

Our system shall operate as an IOS only mobile application. It will be developed in XCode, using the proprietary IOS object oriented development language Swift. For the backend we will be using Firestore as our development tool. The backend will be programmed with Python as our language of choice. Digital Ocean will be the hosting service for the corresponding webserver, which will be programmed with JavaScript. Docker will be used as the container service for the application. Depending on time and technological constraint, an

Android service may be developed alongside the IOS service. Due to the functionality of cross platform development, our app would have to be simultaneously and separately programmed in Kotlin and Swift, because of our time constraints we are prioritizing IOS development. If the aforementioned constraints permit Android development, the Android application will be programmed with Kotlin.

- Frontend - Swift

  The frontend is responsible for displaying all of the visual portions of the application. Such as food posts, surveys, and account information. If possible the user interface will mimic the current Umaine theme/aesthetic. The frontend will also be communicating with the backend to manage user actions and data flow. If the user clicks on the map details it will send info to the mapping client. If the user clicks for a post, checks an account etc it will communicate with the backend.

- Backend - Python Firestore

  Utilizing Firestore for its high reliability, easy scalability and speed. Our backend needs scalability for future growth, speed for handling the logic behind the various subsections of code, (Uploading Posts, Viewing Posts, updating account settings, etc.).

- Database - Firestore

  Firestore was chosen for the database due to its scalability and affordability, with support of 50k monthly active users and 1gb of storage. It covers more than enough to cover more potential users than planned. With the minimal data that is being saved, 1gb of data is enough, it does also support Google pricing for cloud storage if more is needed. Data such as user account information, Survey Information, While foodshares and their photos will be stored, they will only be temporary. Being removed after time to save on space and costs. Surveys and Foodshare data stored will be utilized for the improvement of the application.

- Server - Digital Ocean

  Digital Ocean is what our system uses to host the webserver for the project.

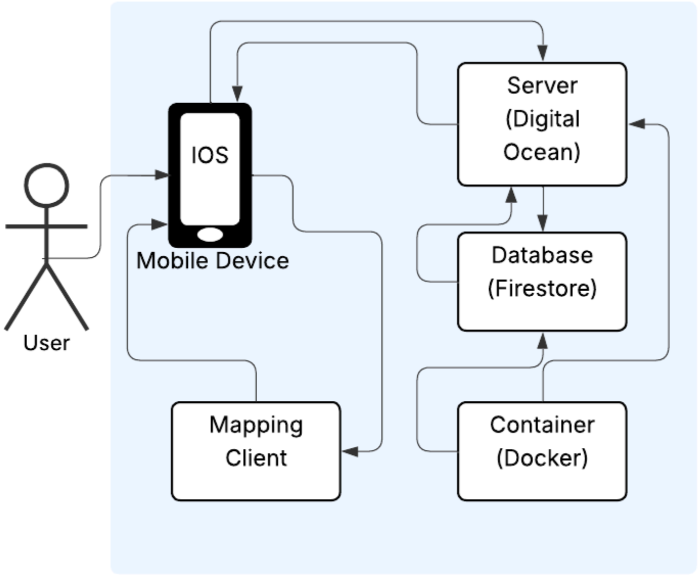


Fig 1. Technology Architecture Diagram

2.2 Decomposition Description

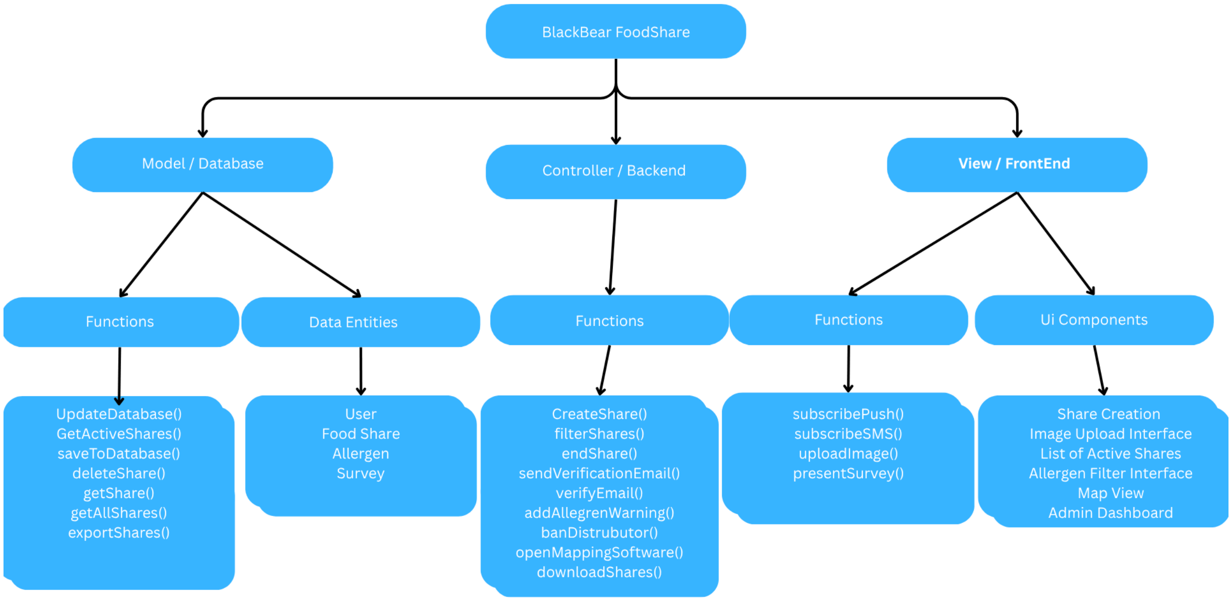


Fig. 2 Class Diagram

Our System follows an object-oriented design and utilizes the Model-View-Controller pattern. The pattern is split into three layers.

- Model Layer - Database; utilizing firestore.

  The model layer is our database; it is where data will be stored, and any functions will be needed to modify or view the data from the database.
  The data entities are the different types of data that will be in the backend, there will be user data such as accounts, foodshare data, which will be the posts sharing food. Allergen data which is for safety, and Survey data which will be used for analytics

- View Layer - Frontend; utilizing swift for IOS

  The View layer is the front-end visuals where the user will be. Anything that needs to be visualized will be in the view layer such as Ui Components. There will also be functions that reflect user actions such as signing up for notifications, or uploading images for food shares.

- Controller Layer - Backend; utilizing python and firestore.

  The Controller layer handles the data processing between the view layer which is where the user will be, and the model layer or database. Any data processing such as filtering food shares, email verification, creating food shares, opening a map for finding the food share, and adding allergens to a food share are all things the controller layer handles.

3 Persistent Data Design

This section shall highlight the relationship and format of the data we store. It will show what fields are exactly stored in the database in section 3.1, and what file formats the system will work with in 3.2.

3.1 Database Descriptions (if you use a database)

Our system will use a NoSQL database. We will have fields for the User, Foodshare, Picture, and Survey. The User, Foodshare and Survey will be permanent, while the pictures will be deleted from the database after two weeks.

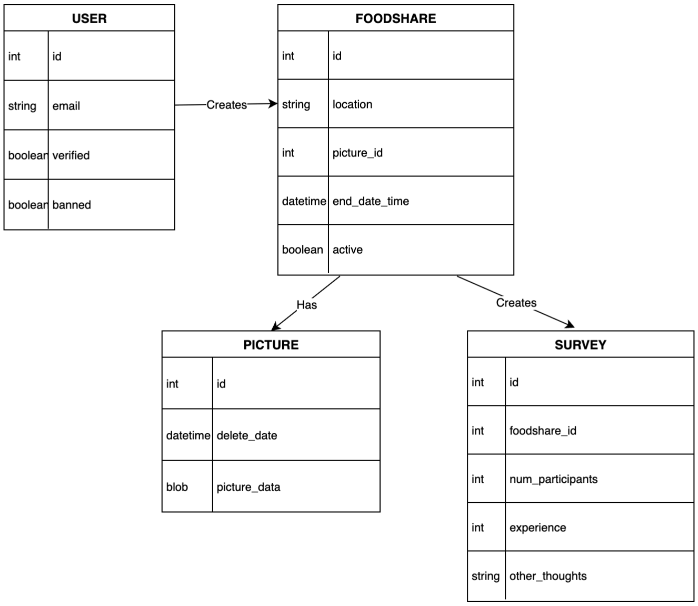


Fig. 3 Database Description

3.2 File Descriptions

The database will use standard file types like jpeg, zip, json, and csv. Our system will use no non-standard file structure.

4 Requirements Matrix

The requirements matrix shows how each functional requirement in the System Requirements Specification corresponds with the components in the System Design Document. Its purpose is to ensure that every requirement is addressed in the system design and has methods or functions to complete the requirements. It will be referenced during implementation to determine the details of how each of these different requirements should be implemented in practice to work together to create the overall product. A 'user' is anyone who is using the application, a 'host' is someone who creates a food share, which any 'user' is able to do, and

Functional Requirement use case number	Functional Requirement name	System Component

UC-001	Users shall be able to sign up for push notifications.	subscribePush()
		subscribeSMS()

UC-002	Create food share	createShare()
		updateDatabase()

UC-003	Hosts shall be able to upload and post images inside their posts	uploadImage()
UC-004	View current food shares	getActiveShares()

UC-005	Ending food shares	endShare()
		presentSurvey()

UC-006	User Verifies Email	sendVerificationEmail()
		verifyEmail()

UC-007	User Filters food shares by Allergy.	filterShares()

UC-008	Host Adds Allergen warning.	addAllergenWarning()
		saveToDatabase()

UC-009	User accesses location map service	openMappingSoftware()
UC-010	View food share	getShare()

UC-011	Admin Views food share Data	getAllShares()
		exportShares()
		downloadShares()

UC-012	Admin Deletes food share	deleteShare()
		updateDatabase()

UC-013	Admin Bans Food Distributor	banDistributor()
		updateDatabase()

Appendix A – Agreement Between Customer and Contractor

The customer and team agree that the document represents a finalized version that meets all agreed upon criteria and requirements. Both parties have thoroughly reviewed the document and all of its contents and agree that the information and structure meet the standards of the requirements. Any feedback or revisions have been addressed and remedied. The document is considered complete and ready for use.

If changes to the System Design Document are required, the team will begin by preparing a draft of the document with the changes. After the draft has been prepared, each team member will review the draft and note any problem with it they might have. Once all the problems have been addressed, the team will review the document once more and sign off on it, to show that they are satisfied with the state of the document. Once that has been completed, we will submit the draft to the client for review and approval.

Wesley Dumas
Signature: *Wesley Dumas* Date: 11/17/25

Corey Kaulenas
Signature: *Corey Aras Kaulenas* Date: 11/17/25

Makai Moody-Broen
Signature: *Makai Moody-Broen* Date: 11/14/25

Denis Sima
Signature: *Denis Sima* Date: 11/17/25

Jakob Sholler
Signature: *Jakob Sholler* Date: 11/16/25

Dr. Scott Marzilli
Signature: ______________________________________ Date: 11/17/25
Comments:

Appendix B – Team Review Sign-off

All team members have thoroughly reviewed this document and reached full agreement on all of its content. No major concerns have been raised, and any minor concerns or clarifications raised by individuals are noted in the comments below their signature. Additionally, all members of the team have agreed upon the formatting and presentation of this document with no major complaints.

Wesley Dumas
Signature: *Wesley Dumas* Date: 11/17/25
Comments:

Corey Kaulenas
Signature: *Corey Aras Kaulenas* Date: 11/17/25
Comments:

Makai Moody-Broen
Signature: *Makai Moody-Broen* Date: 11/14/25
Comments:

Denis Sima
Signature: *Denis Sima* Date: 11/17/25
Comments:

Jakob Sholler
Signature: Jakob Sholler Date:11/16/25
Comments:

Appendix C – Document Contributions

Sholler Jakob, Contributions:
Decomposition description
2.1 bullet points

Dumas Wesley, Contributions:
Requirements Matrix
Appendix A, B

Moody-Broen Makai, Contributions:
Introduction
Purpose
Architectural design intro and 2.1 paragraphs
Logo

Kaulenas Corey, Contributions:
Database Description
File Description

Sima Denis, Contributions:

Appendix C - User Interface Design Document

Black Bear Foodshare

System Design Document

Olive Food Solutions
Dr. Scott Marzilli
Team Members: Dumas Wesley, Kaulenas Corey, Moody-Broen Makai, Sima Denis, Sholler Jakob

Table of Contents

Page 2

1. Introduction

This is a capstone project for Dr. T Scott Marzilli, Associate Provost of Student Success & Innovation, in partial fulfillment of the computer science BS degree for the University of Maine. Dr. Mazilli places great emphasis on the importance of providing the necessary resources and opportunities for students to succeed at the University of Maine. The impetus of this project comes from the passion Dr. Marzilli and our team share in providing judgement free opportunities for students dealing with food insecurity, as well as furthering the elimination of food waste on our campus. The purpose of this project is to develop an application to allow hosts of catered events to easily inform students of excess food on campus. These posts will include the food, photos of the food, and the location. The goal of this project is to reduce waste and increase sustainability on campus, while also complying with regulations to make sure the excess food is safe to consume.

1.1 Purpose of This Document

The Purpose of this user interface design document is to establish a clear set of guidelines for how the application's interface components will be organized, displayed, and developed. In addition to presenting prototypes for the UI elements, this document outlines the underlying data structures and identifies any physical reports that could be generated, such as surveys for Black Bear Foodshare.

1.2 References

- Olive Food Solutions, System Requirements Specification
  System Requirements Specification
- Olive Food Solutions, System Design Document
  System Design Document

2. User Interface Standards

This section establishes the design standards that will guide the development of Black Bear Foodshare's user interface. These standards ensure that developers maintain a consistent, intuitive, and coherent experience across all components of the system.

- Design & Principles
  - This app will follow the UMaine colors and design. There will be changes in the final iteration to create a more cohesive and finished look that the diagrams do not fully capture.
  - The fonts and colors will follow the UMaine principles for accepted designs.
- Header
  - The header will be consistent across all pages and have buttons for home or settings
  - At any point the user should be able to access the home screen through the Header navigation bar
- Log In
  - Will only be accessed if the device is not yet affiliated with an account, and will ask for a UMaine email address
  - A verification email will be sent, the log in page will remain static until verification is confirmed, whereupon it will redirect to the homepage
- Homepage
  - Our homepage will serve as the central hub and primary page for all users
  - The homepage will display active foodshares
  - The homepage will display options for foodshare creation
  - The homepage will display a settings option
  - The homepage will display a filter option

- Foodshare
  - Each foodshare will have its own viewable page which can be accessed from the homepage
  - Each foodshare will display location
  - Each foodshare will display images (if provided by host)
  - Each foodshare will display allergens
  - Each foodshare will display types of food available
  - Each foodshare will display start time and end time

- Foodshare creation
  - The foodshare creation option will be accessed through the homepage
  - The foodshare creation page will contain fields for location, images, allergens, types of food, and start/end times
  - The foodshare creation page will display a button to end foodshare, which will remove foodshare from homepage
  - The foodshare creation page will display a two question survey upon completion (estimated total attendees, estimated percentage of food taken)

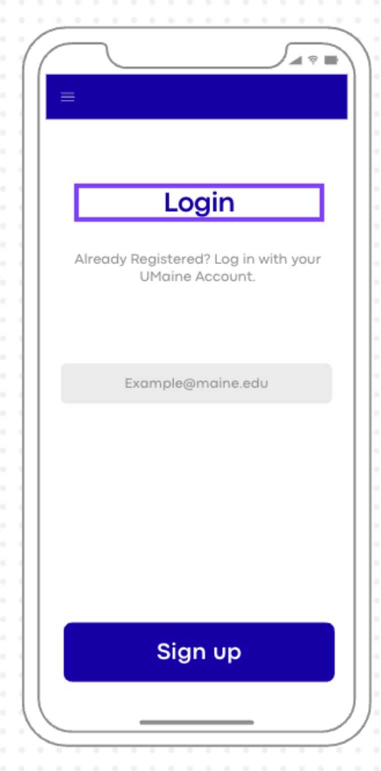
- ❖ Settings
  - ➢ The settings page will be a drop down display accessed from the homepage
  - ➢ The settings page will display a log out option
  - ➢ The settings page will display a contact option

- ❖ Log out
  - ➢ The logout button can be accessed from settings
  - ➢ When pressed a confirmation message will be displayed, which will then redirect to log in page

- ❖ Contact
  - ➢ Contact page can be accessed through setting dropdown
  - ➢ Contact page will contain a field with which any queries or issues can be reported and sent to the relevant contacts.

3. User Interface Walkthrough

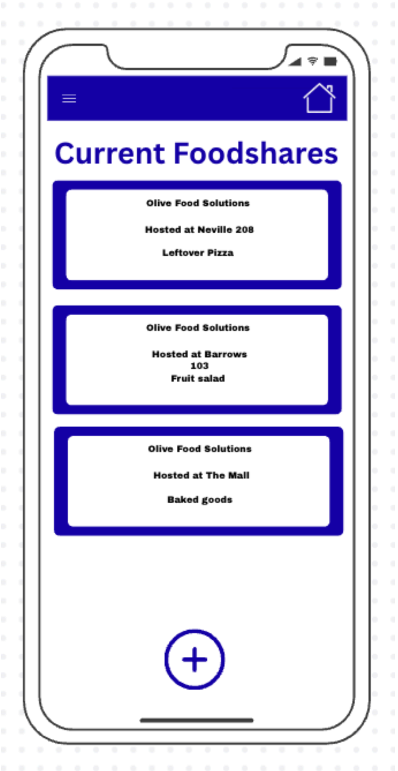
This section provides a comprehensive walkthrough of Black Bear Foodshare's user interface, guiding readers through the structure and flow of the system's screens.



Login

Already Registered? Log in with your UMaine Account.

Fig.1



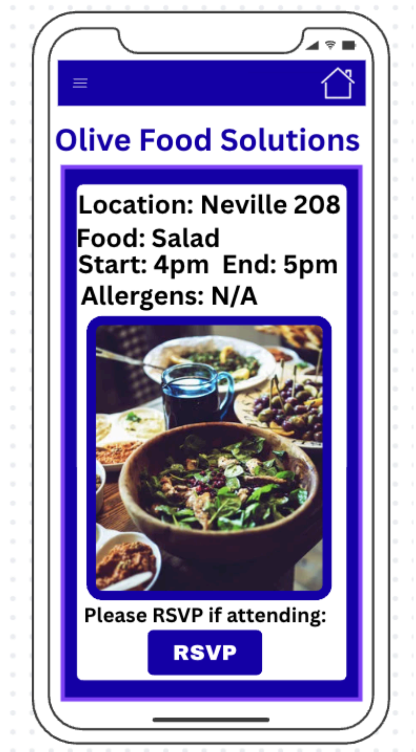
Current Foodshares

Olive Food Solutions
Hosted at Neville 208
Leftover Pizza

Olive Food Solutions
Hosted at Barrows 103
Fruit salad

Olive Food Solutions
Hosted at The Mall
Baked goods

Fig. 2



Olive Food Solutions

Location: Neville 208
Food: Salad
Start: 4pm End: 5pm
Allergens: N/A

Please RSVP if attending:

Fig. 3

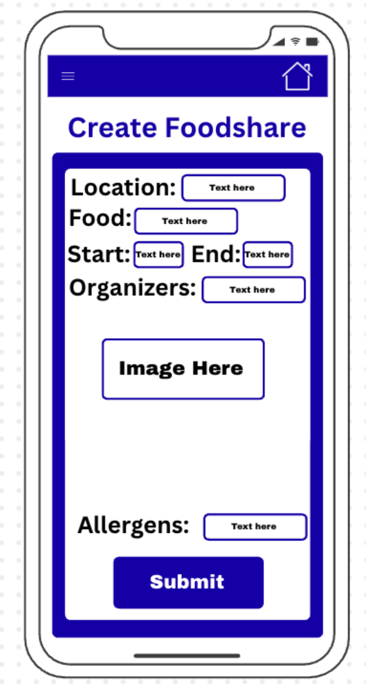


Fig. 4

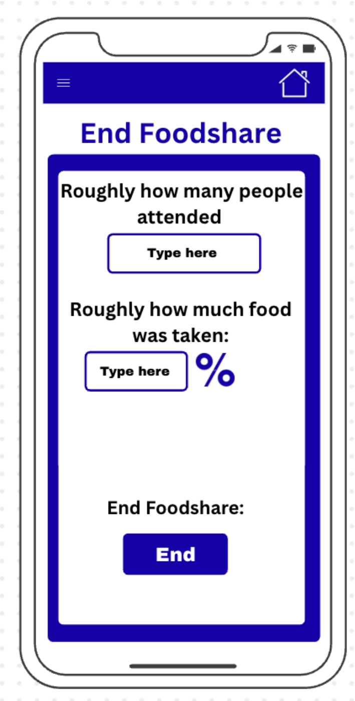
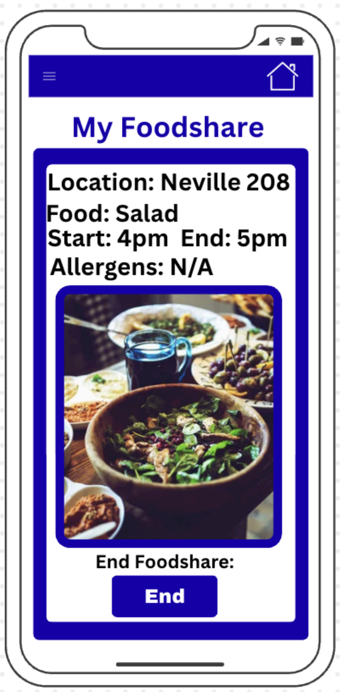

End Foodshare
Roughly how many people attended
Type here
Roughly how much food was taken:
Type here
%
End Foodshare:
End

Fig. 5



Location: Neville 208
Food: Salad
Start: 4pm End: 5pm
Allergens: N/A

Fig. 6

In the figures above the layout for all user interfaces is depicted. Fig 1 will only be displayed when the app is first opened, and after any subsequent log outs.

Fig 2. shows a very simple version of the foodshares and the information that will displayed in the home page

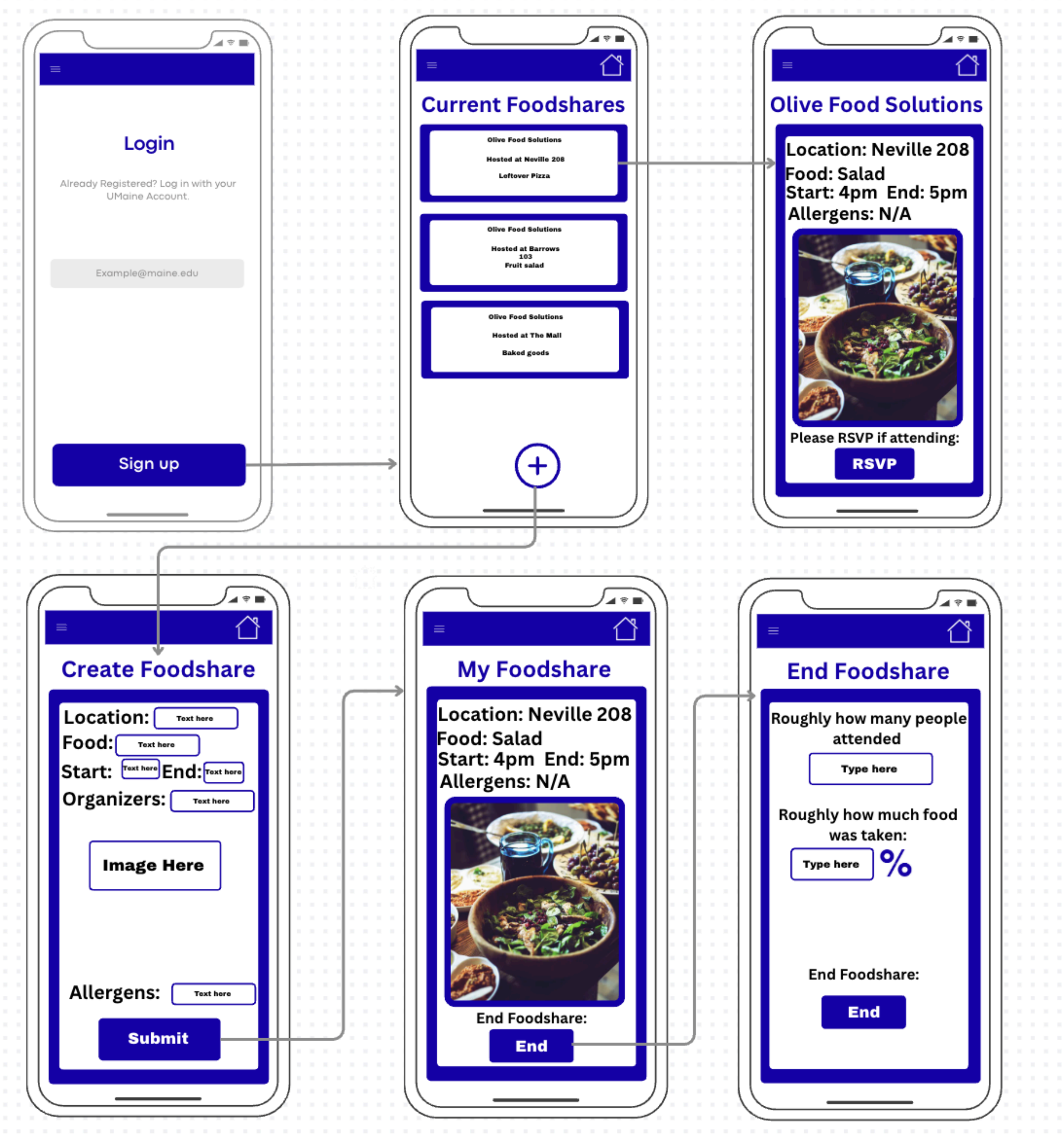
Fig 3. illustrates the display that will be accessed by clicking on a foodshare in the homepage.

Fig 4. Contains all of the relevant fields for a foodshare host. These will be accessed by clicking the plus on the homepage. A host can either choose a photo from their library or take a picture with their camera.

Fig 5. shows the end of the foodshare survey that will appear once a host ends their foodshare. This will be an optional survey with the aim of providing UMaine with data.

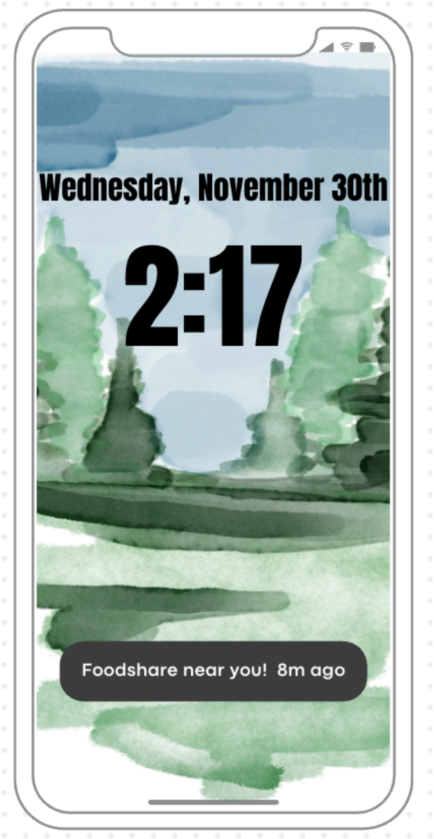
Fig 6. Is what a host will see when viewing their own foodshare. This will have the option for them to end the foodshare.

Fig 7. Walkthrough Diagram:



The Above figure depicts the walkthrough for the Black Bear Foodshare application. This will be the typical navigation steps for users or hosts. At any point a user can click upon the home button which will return them to the home page (Fig. 2)

Fig 8. Notification:



Wednesday, November 30th

2:17

Foodshare near you! 8m ago

The above figure demonstrates an example notification.

4. Data Validation

This section outlines the data validation standard that governs all user-entered information within the Black Bear Foodshare system. It provides a detailed description of each data item, including its type, limits, and required formats. Ensuring that inputs are consistent and accurate.

Data Item	Data Type	Limits	Allowable Format(s)
Login Email (Fig. 1)	string	512 characters	Email address
Images (Fig. 4)	image	Size less than 10MB	PNG, JPEG, HEIC
Location (Fig. 4)	string	512 characters	Address
Food (Fig. 4)	list of strings	25 elements, of no more than 50 characters each	text
Allergens (Fig. 4)	list of strings	25 elements, of no more than 50 characters each	text
Start Time (Fig. 4)	time	Cannot be before current time	Valid time in the format of hh:mm
End Time (Fig. 4)	time	Must be after Start Time	Valid time in the format of hh:mm
Organizers (Fig. 4)	string	512 characters	text
People Attended (Fig. 5)	int	64 bit integer limit	Integer
Food Taken Percent (Fig. 5)	double	64 bit float limit	Decimal number

Appendix A – Agreement Between Customer and Contractor

The customer and team agree that the document represents a finalized version that meets all agreed upon criteria and requirements. Both parties have thoroughly reviewed the document and all of its contents and agree that the information and structure meet the standards of the requirements. Any feedback or revisions have been addressed and remedied. The document is considered complete and ready for use.

If changes to the User Interface Design Document are required, the team will begin by preparing a draft of the document with the changes. After the draft has been prepared, each team member will review the draft and note any problem with it they have. Once all the problems have been addressed, the team will review the document once more and sign off on it, to show that they are satisfied with the state of the document. Once that has been completed, we will submit the draft to the client for review and approval.

Wesley Dumas
Signature: *Wesley Dumas* Date: 12/3/25

Corey Kaulenas
Signature: Corey Date: 12/3/25

Makai Moody-Broen
Signature: *Makai Moody-Broen* Date: 11/24/25

Denis Sima
Signature: *Denis Sima* Date: 12/3/25

Jakob Sholler
Signature: . Jakob Sholler Date: 12/3/25

Dr. Scott Marzilli
Signature: ______________________________________ Date: 12/3/25
Comments:

Appendix B – Team Review Sign-off

All team members have thoroughly reviewed this document and reached full agreement on all of its content. No major concerns have been raised, and any minor concerns or clarifications raised by individuals are noted in the comments below their signature. Additionally, all members of the team have agreed upon the formatting and presentation of this document with no major complaints.

Wesley Dumas
Signature: *Wesley Dumas* Date: 12/3/25
Comments:

Corey Kaulenas
Signature: Corey Date: 12/3/25
Comments:

Makai Moody-Broen
Signature: *Makai Moody-Broen* Date: 11/24/25
Comments:

Denis Sima
Signature: *Denis Sima* Date: 12/3/25
Comments:

Jakob Sholler
Signature: Jakob Sholler Date:12/3/25

Comments:

Appendix C – Document Contributions

Sholler Jakob, Contributions:
Introduction(s)
Purpose

Dumas Wesley, Contributions:
Data Validation

Moody-Broen Makai, Contributions:
2 User interface standards
3 User interface walkthrough

Kaulenas Corey, Contributions:
Data Validation
Editing

Sima Denis, Contributions:

Appendix D - Requirement Matrix.

The requirements matrix shows how each functional requirement in the System Requirements Specification corresponds with the components in the System Design Document. Its purpose is to ensure that every requirement is addressed in the system design and has methods or functions to complete the requirements. It will be referenced during implementation to determine the details of how each of these different requirements should be implemented in practice to work together to create the overall product. A 'user' is anyone who is using the application, a 'host' is someone who creates a food share, which any 'user' is able to do, and

Functional Requirement use case number	Functional Requirement name	System Component

UC-001	Users shall be able to sign up for push notifications.	subscribePush()
		subscribeSMS()

UC-002	Create food share	createShare()
		updateDatabase()

UC-003	Hosts shall be able to upload and post images inside their posts	uploadImage()
UC-004	View current food shares	getActiveShares()

UC-005	Ending food shares	endShare()
		presentSurvey()

UC-006	User Verifies Email	sendVerificationEmail()
		verifyEmail()

UC-007	User Filters food shares by Allergy.	filterShares()

UC-008	Host Adds Allergen warning.	addAllergenWarning()
		saveToDatabase()

UC-009	User accesses location map service	openMappingSoftware()
UC-010	View food share	getShare()

UC-011	Admin Views food share Data	getAllShares()
		exportShares()
		downloadShares()

UC-012	Admin Deletes food share	deleteShare()

		updateDatabase()

UC-013	Admin Bans Food Distributor	banDistributor()
		updateDatabase()

Appendix E - Agreement Between Customer and Contractor

The customer and team agree that the document represents a finalized version that meets all agreed upon criteria and requirements. Both parties have thoroughly reviewed the document and all of its contents and agree that the information and structure meet the standards of the requirements. Any feedback or revisions have been addressed and remedied. The document is considered complete and ready for use.

If changes to the User Interface Design Document are required, the team will begin by preparing a draft of the document with the changes. After the draft has been prepared, each team member will review the draft and note any problem with it they have. Once all the problems have been addressed, the team will review the document once more and sign off on it, to show that they are satisfied with the state of the document. Once that has been completed, we will submit the draft to the client for review and approval.

Wesley Dumas
Signature: Wesley Dumas Date: 12/19/25

Corey Kaulenas
Signature: Corey Date: 12/19/25

Makai Moody-Broen
Signature: *Makai Moody-Broen* Date: 12/17/25
Denis Sima
Signature: *Denis Sima* Date: 12/19/25

Jakob Sholler
Signature: Jakob Sholler Date: 12/ 18/25

Dr. Scott Marzilli
Signature: Date: 12/19/25
Comments:

Appendix F - Team Review Sign-Off

All team members have thoroughly reviewed this document and reached full agreement on all of its content. No major concerns have been raised, and any minor concerns or clarifications raised by individuals are noted in the comments below their signature. Additionally, all members of the team have agreed upon the formatting and presentation of this document with no major complaints.

Wesley Dumas
Signature: Wesley Dumas Date: 12/19/25
Comments:

Corey Kaulenas
Signature: Corey Date: 12/19/25
Comments:

Makai Moody-Broen
Signature: *Makai Moody-Broen* Date: 12/19/25
Comments:

Denis Sima
Signature: *Denis Sima* Date: 12/19/25
Comments:

Jakob Sholler
Signature: Jakob Sholler Date:12/19/25
Comments:

Appendix G - Document Contributions.

Sholler Jakob, Contributions:

- Document Template
- Sections 4, 4.1, 4.2, 4.3, 4.4, 4.5
- Section 5, 5.1, 5.2, 5.3
- Bibliography
- List of Figures
- List of tables

Dumas Wesley, Contributions:

- Preface
- Recommendations

Moody-Broen Makai, Contributions:

- UIDD revisions

Kaulenas Corey, Contributions:

- Executive Summary
- API Section
- General editing

Sima Denis, Contributions:

- Sections 1.1,1.2, 1.3
- Section 2
- Section 3