Black Bear Foodshare

System Requirements 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

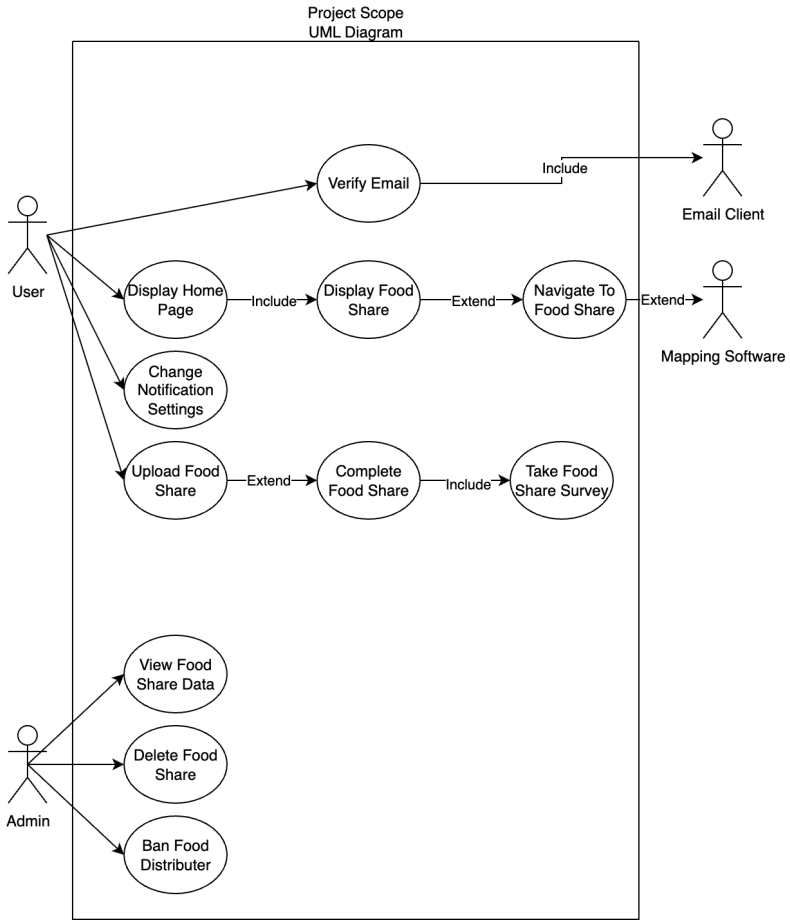
The Black Bear FoodShare application will provide a platform for verified university-affiliated food servers ("Hosts") to post announcements about excess food, including details and photographs. It will allow university students ("Users") to view these posts and opt-in to receive push or SMS notifications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

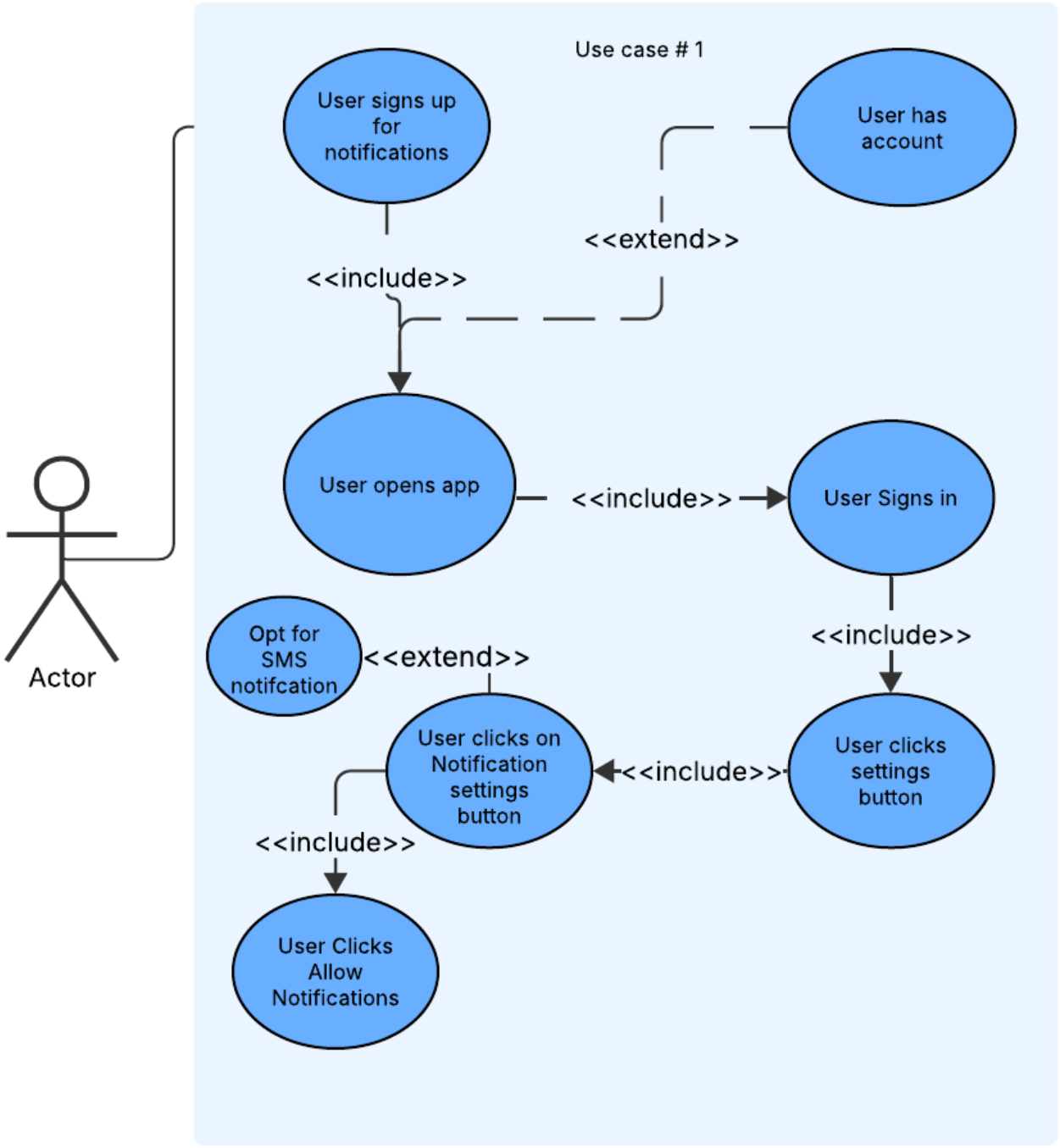
Out-of-Scope:

- 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

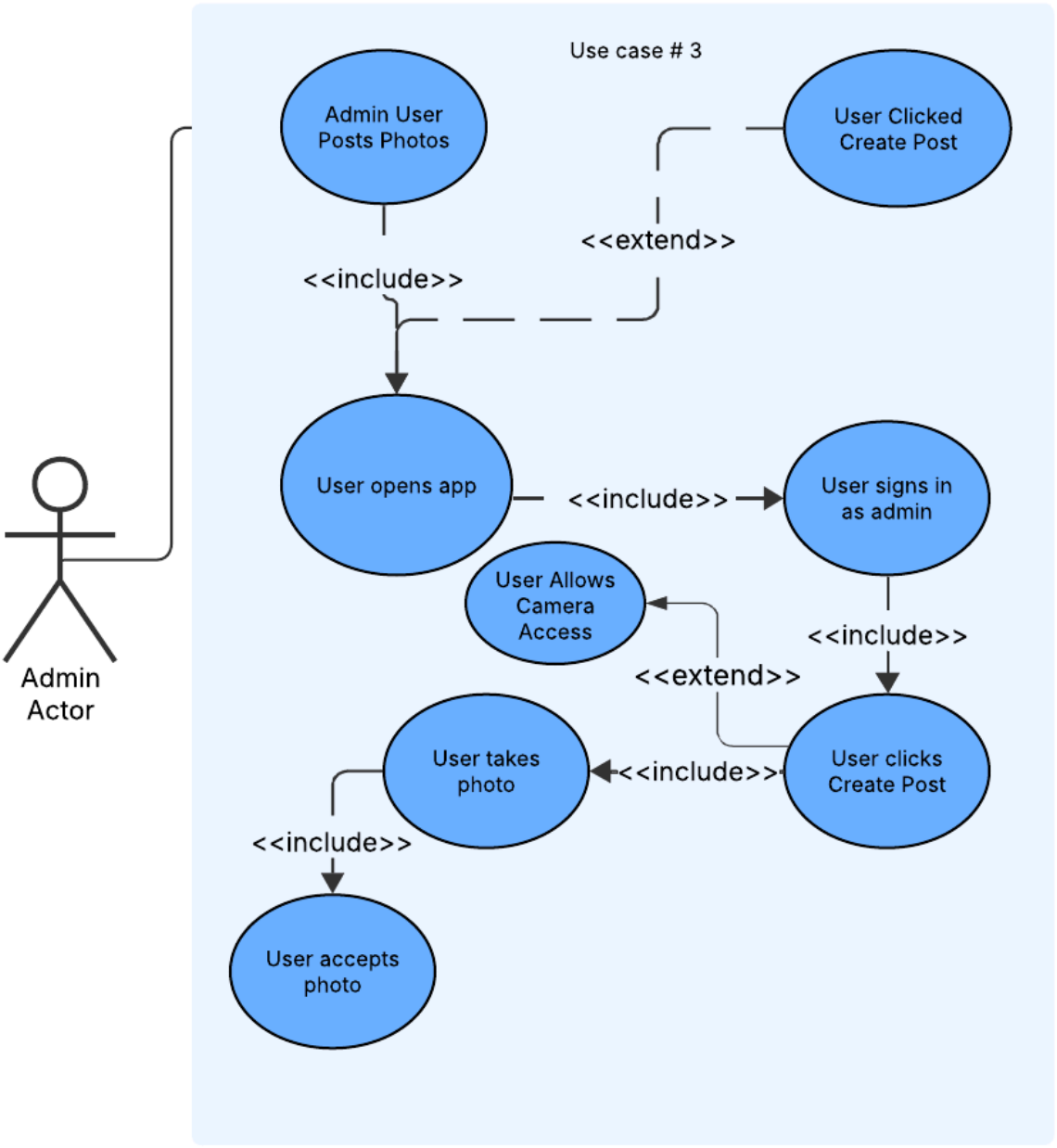
2. Functional Requirements

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 Distributer	
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 distributer" 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	Measure response time	5

	provide directions to the location of the food with Google Maps within 3 seconds.	from when the user requests directions to 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	Digitally: pdf, docx Hard copy
System Design Document	Digitally: 11/17/25	Digitally: pdf, docx Hard copy

User Interface Design Document	Digitally: 12/3/25	Digitally: pdf, docx Hard copy
Critical Design Review Document	Digitally: 12/17/25	Digitally: pdf, docx
User Manual	TBD	Digitally: pdf, docx Hard copy
Administrator Manual	TBD	Digitally: pdf, docx Hard copy
All Biweekly Status Reports	TBD	Digitally: pdf, docx Hard copy
(All Source code)	TBD	Git
The Executable program	TBD	Git
(All other software required for installation and execution of delivered program)	TBD	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

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