

Integrated Project Plan

James Boehm, Marcus Jones, David Markowski

Ray Gutierrez, Andrew Delgado, David Batdorf

December 15, 2019

CMSC 495

Professor Terrence Mentzos

## Table of Contents

<b>Introduction .....</b>	<b>Page 3</b>
<b>Overview .....</b>	<b>Page 3</b>
<b>Integrated Project Plan .....</b>	<b>Page 4</b>
<b>Requirements Specification .....</b>	<b>Page 17</b>
<b>System Specification .....</b>	<b>Page 18</b>
<b>User Guide .....</b>	<b>Page 20</b>
<b>Test Plan and Results .....</b>	<b>Page 26</b>
<b>Design and Alternate Designs .....</b>	<b>Page 40</b>
<b>Development Histroy .....</b>	<b>Page 44</b>
<b>Conclusions.....</b>	<b>Page 45</b>
<b>Resourses .....</b>	<b>Page 47</b>

## 1. Introduction

The “Food 4 Friends” web application was designed to combat two large issues that face society today, by bringing people together. The first issue is food waste. According to the USDA somewhere between 30 and 40 percent of the US food supply is lost to waste. The second issue is hunger, which according to USDA 11.1% of Americans lacked enough food to sustain themselves at some point last year. That means over 36 million Americans lack access to an adequate and stable food supply.

We at KISS are developing the “Food for Friends” web application to coordinate individuals to combat these societal issues. In the future the application will use geolocation services to pair individuals, givers, with those who are in need of a food, recipients. Currently the service is limited to a single static test market, UMGC’s Adelphi MD campus. Users will be required to create an account, with authentication provided through Google’s API. Once a user is authenticated, they can set the app to either giver, or recipient mode. Giver mode will allow the user to post available food to users in the recipient mode. Recipient mode allows users to view a list of locally available food offerings.

## 2. Overview

The “Food for Friends” application is browser-based application, built upon Vue.js framework. The project contains three separate coding languages. The graphical user interface was designed in HTML, with JavaScript controlling the user’s interactions with the individual parts of the page. To meet the design requirement of allowing users interact with each other SQL was employed to communicate between the application and a remote SQLite database that stores transaction data, and user profiles.

The six-man team working on this project consists of the Project Manager David Batdorf, the Technical Writer Ray Gutierrez, the User Interaction Engineer Andrew Delgado, the Software Engineer David Markowski, the Integration Engineer Marcus Jones, and finally the Test Engineer James Boehm. Each member contributed significantly to the final product, and allowed the idea to come to fruition.

David Batdorf, the team’s project manager, coordinated the entire project, making major decisions, while leaving minor aspects as the responsibilities of the subject matter experts. He also created, and updated the project plan through each sprint cycle. Next David coordinated with team members to distribute workload, and ensure that all deliverables were completed in a timely manner. Finally, David compiled submission documents to accompany each code deliverable.

Ray Gutierrez, the team’s Technical Writer has played an instrumental role in ensuring that each document adheres to standards, and flows in a logical manner. Ray has also created documents from the data provided from the subject matter experts, including the test plan, the coding phase documents and a significant portion of the final report.

Andrew Delgado, the team's User Interaction Engineer, has taken the lead in not only defining how the application will look, but also directing how the application will flow. David introduced the team to an application called Figma, that allowed him to create flows, and mock user interfaces. This tool has been extremely helpful for team collaboration to visualize the user interface, before it has been implemented into the code base. The tool also greatly aided the team in staying with in requirements, because the user interface was designed before any coding took place.

David Markowski, the team's Software Engineer, has put in a large amount of work to ensure that the application meets the required standards each sprint, while also keeping the Integration Engineer, and the Test Engineer apprised of any changes, or new requirements. David has done an excellent job of communicating with the entire team during the development process, and has ensure that the majority of the development efforts have been completed on time.

Marcus Jones, team's the Integration Engineer, has worked closely with the Software Engineer, and the Project Manger to ensure that external interfaces, and project frameworks meet all project requirements, and will smoothly interface with the existing code base. Marcus also designed the database connectors, and the database itself. Finally Marcus also aided the software developer in creating code for the project.

James Boehm, the team's Test Engineer created a comprehensive plan to test the "Food for Friends" application, and then regularly executed the test plan throughout the development phase of the project. James also significantly contributed to error correction after his tests had completed. James also acted as the team's security expert, guiding the design and coding in a secure manor.

### 3. Integrated Project Plan

#### 3.1.0 Introduction

This project is designed to tackle two large issues that face society today, by bringing people together. The first issue is food waste. According to the USDA somewhere between 30 and 40 percent of the US food supply is lost to waste. The second issue is hunger, which according to USDA 11.1% of Americans lacked enough food to sustain themselves at some point last year. That means over 36 million Americans lack access to an adequate and stable food supply.

We at KISS are developing the "Food for Friends" web application to coordinate individuals to combat these societal issues. The application will use geolocation services to pair individuals, givers, with those who are in need of a food, recipients. Users will be required to create an account, with authentication provided by Google. Once a user is authenticated, they can set the app to either giver, or recipient mode. Giver mode will allow the user to post available food to users in the recipient mode. Recipient mode allows users to view a local map with pins denoting the recently posted food offerings.

This document is designed to clearly communicate with both the project team and the stakeholders. This document will outline how to communicate with the project team, and provide a development timeline. The document will also define the project risk, along with change procedures and control procedures for cost, and time.

### 3.2.0 Project Contacts and Roles

Role	Name	Organization/Department	Phone/E-mail Address
Project Manager	David Batdorf	Project Management	afdave5124@gmail.com
User Interaction Engineer	Andrew Delgado	Requirements	andrewdelgado88@gmail.com
Technical Writer	Ray Gutierrez	Requirements	raygoot@outlook.com
Software Engineer	David Markowski	Development	markowskidavid77@gmail.com
Integration Engineer	Marcus Jones	Development	marcus.njones2713@gmail.com
Test Engineer	James Boehm	Test	boehm22@hotmail.com

### 3.3.0 Communications Management

This table lists the different communication items needed for this project. A communication item may be a Word document, an e-mail, a meeting, or anything called for by another plan.

What	When	How	Responsible	Audience
Project Kickoff	24 October 2019	Virtual Meeting	Project Manager	Request comments
Team Progress Sync	Weekly	Virtual Meeting	Project Manager	Update Management on deliverable progress
Team Deliverable Review	Weekly	Virtual Meeting with trello Kanban board collaboration	Project Manager	Review and critique weekly deliverable.

### **3.4.0 Risk Management**

#### **3.4.1 Risk Management Strategy**

- Risks will be identified both initially during the planning, and analysis phases of the development lifecycle. Secondly risk will be identified during weekly team syncs, where project management will review weekly work plans.
- The risks associated with this project vary, both in the type and severity of the risk. First the risk exists to users when they meet strangers via the application. The users also have risk from potentially malicious users, such as poisoned food. Risk exists when the application interfaces with outside modules, such as geolocation, authentication and database interface. Risk also exists to the database that contains user data, and current food posts since the database must accept user inputs. Finally, risk exists in the inability to test the full variety of commercially available computing devices.
- The risk associated with the large variety of commercial computers is one that simply must be accepted due to budgetary constraints. The risk to users, both in meeting strangers and accepting food items will be mitigated. The risk associated with external interfaces will be mitigated, and remaining risk, while minimal must be accepted. The risk to the database will be full mitigated to prevent external tampering.

#### **3.4.2 Mitigation and Contingency Plans**

To mitigate the risk to users via meeting and accepting food from strangers, a rating system will be implemented. This system will allow users to not only rate the person they interact with, but they can also leave feedback. The risk via external dependencies will be mitigated by choosing a reliable source. The risk associated with database access will be fully mitigated by implemented extremely stringent data input parsing, to detect and 5.0 Change Management

The project will follow the standard Software Development Life Cycle change management convention. Any changes made to the project will also require a full team code review, before any changes are committed to the code base.

### **3.5.0 Skillset Management**

Team members were placed into the best role suited for their skill sets. Several team members had existing skills for their original roles. Two members, Marcus Jones and David Markowski switched roles to better suit their skill sets. David Markowski is now the Software Engineer, and Marcus Jones is now the Integration Engineer. All other team members have existing professional or academic experience in their respective work roles.

The skills necessary vary for each position within the project workgroup. The skills needed for a project manager include, talent management, and risk management. The User Interface member must have skills that include graphical generation, and the ability to accurately anticipate user needs. The Technical Writer must have the technical ability to understand the project, along with the writing skills to accurately, and logically document the project. The member in the role of software developer must have a strong grasp of JavaScript, and HTML web design. the integration engineer must have a strong knowledge of external systems, such as databases and third-party authentication modules. Finally, the member in the role of Test Engineer must understand both functionality and security testing, along with having strong written skills to document their efforts.

### 3.6.0 Time and Cost Management

- The most important aspect to cost control is limiting product changes during deployment.
- Time and costs will be controlled by defining clear, and concise requirements.
- Any changes to requirements require Project Manager Approval.
- In the advent of required changes, additional personal will be moved from other roles to ensure a timely development schedule.

#### 3.6.1 Sprint schedule

Sprint Cycle	Deliverable	Date Range	Assignees
1	GIT setup Trello setup	October 21 – October 27	Project Manager
	Project Direction Plan		All Team Members
2	Project Plan	October 28 – November 3	Project Manager
3	Peer Review #1	November 4 – November 10	All Team Members
	Test Plan		Test Engineer Technical Writer
4	Project Design	November 11 – November 17	Project Manager Technical Writer Integration Engineer UX/HCI

5	Peer Review #2	November 18 – November 24	All Team Members
	Phase #1 Source		Project Manager UX/HCI Software Engineer Integration Engineer
6	Phase #2 Source	November 25 – December 1	Project Manager UX/HCI Software Engineer Integration Engineer
7	Phase #3 Source	December 2 – December 8	Project Manager UX/HCI Software Engineer Integration Engineer
8	Peer Review #3	December 9 – December 15	All Team Members
	Final Product Submission		

### 3.6.2 Gantt Diagrams and Feature Backlog

**Figure 3.6.2.1 Sprint Function Goals and Backlog List**

Sprint	Planned Features	Successfully Implemented Features	Feature Backlog
Week 5	Web application navigation	Web application navigation using hard coded data.	None
Week 6	Database Integration  Ability to post food offerings into the database	Database Integration  Ability to post food offerings into the database  Ability to claim food offerings	Interaction rating system  Ability to acknowledge food claims and send the offering to the completed transactions



	Ability to claim food offerings into the database  Ability to cancel food offerings into the database  Interaction rating system	into the database  Ability to cancel food offerings into the database	list.
Week 7	Google sign in  Google Maps integration  Remote web access  User interaction heatmapping	Google sign in  Ability to acknowledge food claims and send the offering to the completed transactions list.	Interaction rating system  Google Maps integration  Remote web access  User interaction heatmapping

Figure 3.6.2.2 Gantt Charts for Development effort

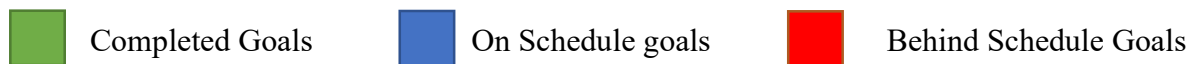


Figure 3.6.2.3 Project Management Gantt Chart

Goals	Sprint 1	Sprint 2	Sprint 3	Sprint 4	Sprint 5	Sprint 6	Sprint 7	Sprint 8
Initialize collaboration tools, including GIT and Trello.	■							
Define project Goals	■							
Define project requirements	■	■						
Create project plan		■						
Compile and update project budget		■	■	■	■	■	■	■
Analyze goal against weekly progress	■	■	■	■	■	■	■	■

Figure 3.6.2.3 Testing Gantt Chart
















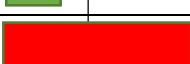








Goals	Sprint 1	Sprint 2	Sprint 3	Sprint 4	Sprint 5	Sprint 6	Sprint 7	Sprint 8
Convert project requirements into test goals								
Compile test plan								
Conduct test plan against Phase I code								
Conduct test plan against Phase II code								
Conduct test plan against Phase III code								
Compile all test result into final document								

Figure 3.6.2.4 Development Gantt Chart

Goals	Sprint 1	Sprint 2	Sprint 3	Sprint 4	Sprint 5	Sprint 6	Sprint 7	Sprint 8
Convert requirements and testing metric into project design								
Layout sprint development goals								
Create navigable GUI using the Vue.JS framework								
Database Integration for food offerings and user IDs								
Ability to post food offerings								
Ability to claim food offerings								
Ability to cancel food offerings								
Interaction rating system								
Google sign in								
Google Maps integration								
User interaction heatmapping								

**Figure 3.6.2.5 Technical Writing Gantt Chart**

Goals	Sprint 1	Sprint 2	Sprint 3	Sprint 4	Sprint 5	Sprint 6	Sprint 7	Sprint 8
Test guide document								
User guide								
Design document								
Phase I document								
Phase II document								
Phase III document								
Final project document								

### 3.7.0 Quality Management Plan

Software quality assurance is crucial to the software development lifecycle and must exist during every step of the lifecycle. Deliverable reviews, and project management will be directly part of each step of the development processes to ensure that specifications are followed and potential errors are minimized

#### 3.7.1 Defect Resolution

A key part to any software development project is defect resolution. In order to identify, and resolve any issues, the software will go through three development phases. After each phase it will undergo rigorous testing from the team Test Engineer. Any identified bugs will be patched, and retested for the following release. Post deployment, a bi-weekly deployment schedule will be used for most defect resolution, with the option for same day deployments for critical fixes.

Defects, when discovered will be classified in one of three ways, based on there potential to harm the applications image. The first and lowest priority classification will be “Low”. These defects are minor and do not risk damage to the application or users. These defects will be corrected as developer time allows. The next tier is “Medium” grade defects. The development team will aim to fix these defects within 1 standard sprint cycle. These defects present medium risk to the applications image and may hinder usage. The final classification of defect is “High”, these defects present extreme risk to the application, or the user. These defects will be priority tasks to fix, and will be patched via an unscheduled application patch.

### 3.8.0 Procurement Management

All external procurements will be vetted by program management, and assessed for stability, security, and cost viability compared to market competitors. External procurement procedures will be used not only for physical procurements such as database servers, but also for digital procurements such as external code libraries.

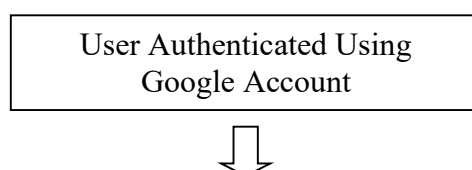
### 3.9.0 Scope Management

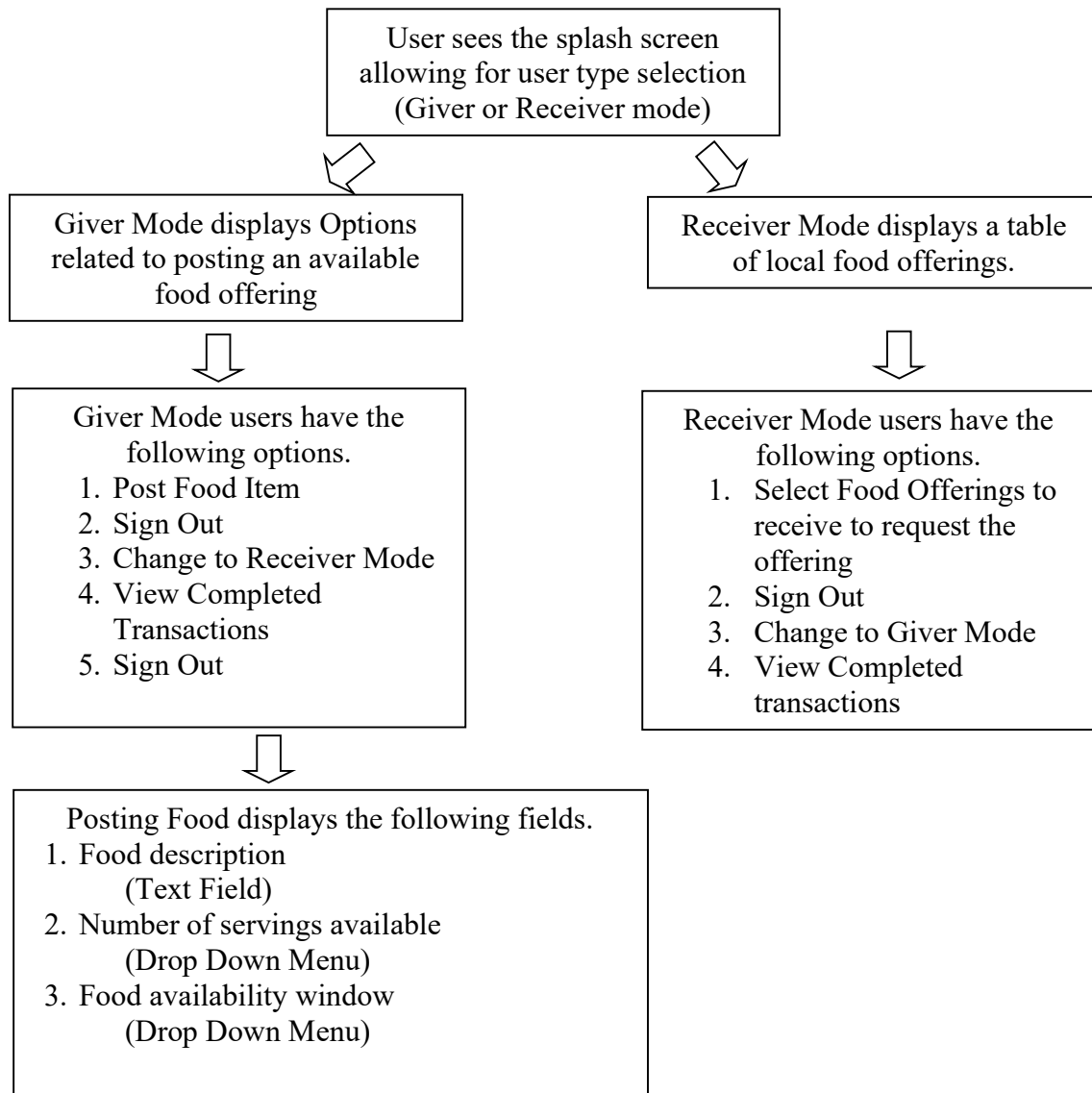
Managing the scope of a project is integral to ensuring that time and costs are kept in check. The requirements for the project fall into two categories. First is the requirements for the application that define what it will do. The second category is the requirements that the user must adhere to, in order to be able to use the application

Each deliverable will be subject to a full team review 3 days prior to submission to ensure that it meets all requirements for the deliverable, without exceeding the scope of the deliverable. Requirements for each deliverable will be fully defined before any part of the creation of that deliverable begins. For the application itself, a full user guild, and test plan will be created before the application is coded. This is done to ensure that the final product meets the initial design fully. This also prevents requirements creep, by limiting new features to only those that will be tested. Team members will be only tasked with creating deliverables that directly relate to their position on the project team. Team members may be the secondary for roles that are less aligned with their role to ensure a timely deliverable delivery.

In order to reach as broad of a user base as possible the technical requirements to run the web application are very general, and the team will be using code that is as portable as possible. For a user to be able to use the web application “Food for Friends” they must have a computer, or smart phone, capable of running the latest version of either Firefox, Chrome, Edge, or Safari. Users must have a google account for authentication. Finally, users must allow the Web Application to access their current location. Users will have some limitation on how they may use the application to ensure that it is only used to give away unused food to those in need. All user entries will be based off drop down menus, not user text entry. This design decision was made to ensure that only intended items may be posted on the application. Initial launch of the “Food for Friends” product will be limited to a single metropolitan area in order to gauge consumer usage, and to acutely compare usage to resource usage. After initial release the product will be deployed to surrounding areas, and a rate of one new area per 2 months. The deployment rate will be reevaluated after one year.

### 3.10.0 Project Data Flow





### 3.10.1 Sample User Interaction

1. Kate signs into the web application “Food for Friends”
2. Kate selected Giver Mode
3. Kate Selects Post Food Item
4. Kate Enters the following information
  - a. Food Description: Meat Free Chili

- b. Number of Servings Available: 2 (from drop down menu)
- c. Food availability window: 3 hours (from drop down menu)
- 5. Kate clicks the post button
- 6. John signs into the application
- 7. John selects Receiver Mode
- 8. John sees Kates nearby post.
- 9. John selects Kates post from the available offerings
- 10. Kate receives a notification to meet John at the UMGC main entrance
- 11. The application moves the transaction to completed transactions.

### 3.11.0 Cost Estimation

The following hourly Rates were retrieved from Glassdoor and will be used to calculate estimated project costs. The rates are based on the Maryland locality that UMGC is based in. The cost estimate for developing the application will be separated into 1-week sprints during initial development, then during system maintenance phase, sprints will be two weeks long. Values for the maintenance phase are estimated averages over one year.

**Table 1.**  
**Original Cost and Time Schedule**

Sprint	Project Manager hours	User Interaction Engineer hours	Technical Writer hours	Software Engineer hours	Test Engineer hours	Integration Engineer hours	Total Cost (hourly rates from table 4)
1	3	1	1	1	1	1	\$364.18
2	5	1	3	1	1	2	\$589.03
3	3	5	3	3	5	2	\$932.35
4	3	3	2	6	2	2	\$827.79
5	3	3	3	6	3	4	\$1,007.40
6	3	3	3	6	3	4	\$1,007.40
7	3	3	3	6	3	4	\$1,007.40
8	5	3	5	1	5	2	\$929.15
Total	28	22	23	30	23	21	\$6,664.70
Maint-enance	2	1	1	2	2	1	\$303.95

**Table 2.**  
**Code Phase II Cost and Time Schedule**

Sprint	Project Manager hours	User Interaction Engineer hours	Technical Writer hours	Software Engineer hours	Test Engineer hours	Integration Engineer hours	Total Cost (hourly rates from table 4)
1	3	1	1	1	1	1	\$364.18
2	5	1	3	1	1	2	\$589.03
3	3	5	3	3	4	1	\$841.65
4	3	3	2	4	2	2	\$727.89
5	3	4	3	7	4	5	\$1,188.65
6	3	4	3	7	3	4	\$1,097.95
7	3	2	3	6	3	5	\$1,013.75
8	5	2	5	1	5	1	\$841.6
Total	28	22	23	30	23	21	\$6,664.70
Maint- enance	2	1	1	2	2	1	\$303.95

**Table 3**  
**Code Phase III Cost and Time Schedule**

Sprint	Project Manager hours	User Interaction Engineer hours	Technical Writer hours	Software Engineer hours	Test Engineer hours	Integration Engineer hours	Total Cost (hourly rates from table 4)
1	3	1	1	1	1	1	\$364.18
2	5	1	3	1	1	2	\$589.03
3	3	5	3	3	4	1	\$841.65
4	3	3	2	4	2	2	\$727.89
5	3	4	3	7	4	5	\$1,188.65
6	3	4	3	7	3	4	\$1,097.95
7	4	2	4	10	5	8	\$1530.85
8	5	2	5	1	5	1	\$841.60
Total	29	22	24	34	25	24	\$7181.80
Maint- enance	2	1	1	2	2	1	\$303.95

**Table 4.**  
**Hourly Rates for Project team**  
**based on glassdoor.com average salaries for**  
**Washington DC locality.**

Role	Expected hourly cost
Project Manager	\$46.99
User Interaction Engineer	\$40.60
Technical Writer	\$31.96
Software Engineer	\$49.95
Test Engineer	\$43.75
Integration Engineer	\$46.95

The charts above are based on estimated personnel time and the average salaries in the Adelphi, Maryland local area, obtained from glassdoor.com. According to the initial estimates the creation of the project should require a total of 147-man hours, totaling a cost of \$6,664.70. An amount of \$8,000 will be budgeted for the project, this will allow an approximately 20% overage from the estimates before going over budget. It is being estimated that reoccurring maintenance, and continued geographical roll out will cost \$303.95 per two-week sprint. The second expense will be the offsite database. The project will utilize an AWS database server with an estimated cost of \$ 243.61 per month. This brings the post development monthly cost to \$851.51. At project completion the project man hours costed \$7,181.80, from a total of 158 man-hours which was above the estimated cost, but still within the budgeted \$8,000. The overages occurred in sprints 6 and 7, due to the complexity of integrating

## 4. Requirements Specification



## 4.1 How To Get There

### Food 4 Friends Web Application Requirements

Our Program Manager (David B) orchestrated a perfect Opus of assigning tasks to meet this seemingly impossible task. His insight to success was evident from the first week's bi-weekly meeting in which the newly-minted group of individuals quickly formed into a cohesive team. For this team was not a dictatorship but special unit upon which all ideas, big or small, were routed through the consensus process. From the onset it was clear we would follow Agile development schema over WaterFall development. As this was a very compact period of performance (POP), using a Waterfall development schema was subject to its limitations. According to Guru99, the Waterfall schema was not conducive where the requirements are not clear at the beginning; difficult to move back to makes changes in the previous phases; since testing process starts once development phase has ended there is high chances of bugs to be found later in development where they are expensive to fix (Guru99, 2019).

## 4.2 What Vehicle Do We Use

The Requirements (Andrew & Ray) ensemble was more of the collective team brainstorming to quickly amass viable options to fulfill in the POP. It was soon realized that a web browser application with plenty of versatility was needed. In this discussion, this application wasn't going to be a mere Android, or iPhone app(s) but could harness the use of any Apple, Windows, Linux, etc... browser thus eliminating any one-hit-wonders. No limiting factor like, "Only available on your friendly iTunes site for download!"

## 4.3 What is the Application

With scalpel-like precision, it was agreed that this application should meet a social need, only thought about during the holidays...fighting hunger! Countless tons of excess food goes to waste every day because people do not an easy to use way post you are looking receive or give your excess food before it goes to waste. Other web applications are out there like Craigslist or Backdoor but you need to wade through epic pages of used crusty couches and cheeky romantic rendezvous. The application should be tailored to create an avenue for an individual or group (Giver) with any excess food a central location to post the availability of their free food. The submitted food will be posted in a user-friendly environment, upon which personnel (Recipient) can navigate based on either availability, gastronomic preference, or location. Additionally, since the principle tenets were the keystones of the application, two precepts to include application security, and application integrity will be required to ensure the application is vested. Finally, the Giver and/or Recipient application flow path should be user friendly.

## 4.4 UX Design Decisions

The color selections we made were based on background image which illustrates a home cooked meal with a picnic style presentation. Using color theory, we chose green which signifies the freshness of the food and the picnic table red cloth represents a communal aspect to sharing

food. Since we are using Google Authentication for user login, we utilized their logo which has blue, green, yellow and red. We used the same hexi-decimal colors they use in the logo to show consistency throughout the application. We used font size that would fit the width and height of each button or surface area and made them big enough for the user to easily read on any medium. The font type itself was primarily “Open Sans” which is a professionally acclaimed font by most UX designer standards. The grey fonts are used for column headers which are helpful pieces of data to understand how we organized the data and used black font for text that is conveying specific details about individual transactions. We are currently not supporting section 508 standards for this release.

## 4.5 Development

As is the norm for Agile, we would be testing early and testing often. The project embodied the Lean Kanban Agile development methodology. The team employed Trello, Git and discord to ensure communication flowed continually throughout the project. The Development Trio of Integration (Marcus), Testing (James), and Development Engineer(s) made the Requirement Ensembles (Andrew, Ray, David B.) vision into reality. They set to institute Systems Specification plan into an incrementally success by testing for functionality as well as security throughout the process.

# 5. System Specification

## 5.1 System Specifications

The Food 4 Friends mantra was to fill the void of minimizing food waste by connecting individuals with excess food to others looking eat whether truly hungry or looking avoid the waste of food. Albeit, many web applications are out in the wild where people can post items such as Craigslist or Facebook Marketplace, but the issue is it inundated with countless other posting not specific to minimizing food waste or hunger. The web application was instituted to be run across a multitude of browsers so as not limit the viability of the target audience.

## 5.2 User Specifications:

- Access to internet capable device
- Internet access
- Email to register

## 5.3 Development Specifications:

- Visual Studio Code
- npm version 6.12.0
- NPM packages for server:

- body-parser @1.19.0
- cors @2.8.5
- express @4.14.1
- mysql-ssh @1.0.6
- nodemon @2.0.1 as dev
- NPM packages for client:
- "dependencies":
  - "axios": "^0.19.0",
  - "core-js": "^3.4.2",
  - "vue": "^2.6.10",
  - "vue-google-signin-button": "^1.0.4",
  - "vue-router": "^3.1.3",
  - "vue2-google-maps": "^0.10.7"
- "devDependencies": {
  - "@vue/cli-plugin-babel": "^4.0.0",
  - "@vue/cli-plugin-eslint": "^4.0.0",
  - "@vue/cli-service": "^4.0.0",
  - "babel-eslint": "^10.0.3",
  - "eslint": "^5.16.0",
  - "eslint-plugin-vue": "^5.0.0",
  - "vue-template-compiler": "^2.6.10"

## 6. User's Guide

### 6.1 Installation Guide

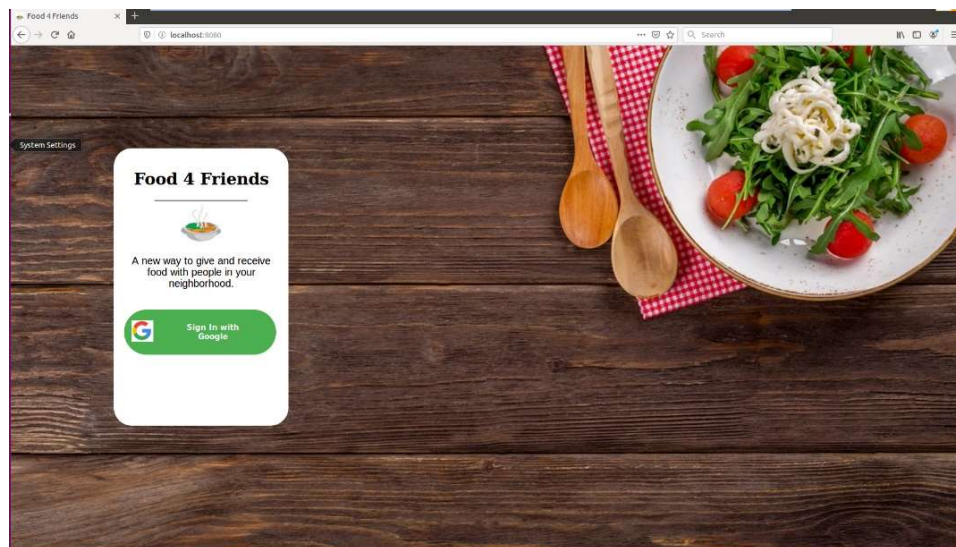
The application is still in development. It currently runs as a local service utilizing Node.js. Therefore, installation of Node.js is a prerequisite. These steps are to be preformed by a system administrator ahead of time to set up the service. Users will follow the user guide starting at step 1 of the use guide.

- Visit [www.nodejs.org/en](http://www.nodejs.org/en) and download and install the current LTS version.
- Extract the zip file to a destination folder of your choice.
- Open terminal or PowerShell window at the project root folder.
- Type the command `npm run serve`
- Navigate to the backend server root folder
- Type the command `npm run start`
- The application will compile and become hosted at <http://localhost:8080> or [http://you\\_private\\_ip:8080](http://you_private_ip:8080)

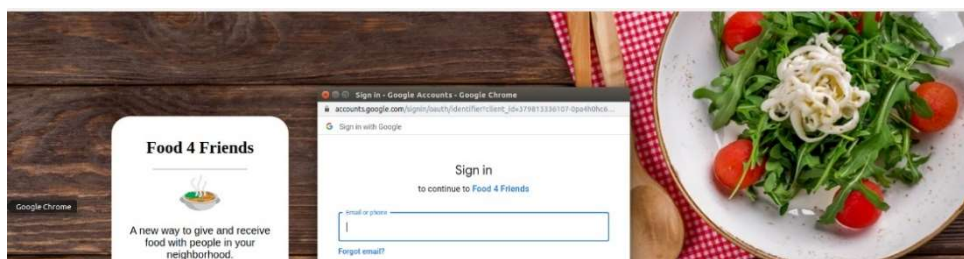
## 6.2 End User Guide

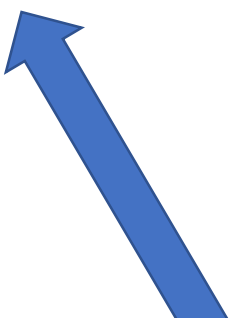
The Food for Friends web application is designed to be easy to use, for users of all technical skill. After initial installation by a system administrator the user is ready to run the application. Due to database limitations, all interactions take a few seconds to run, please do not click multiple times.

1. First the user will visit the local webpage page “localhost:8080”. The user will then see the following screen



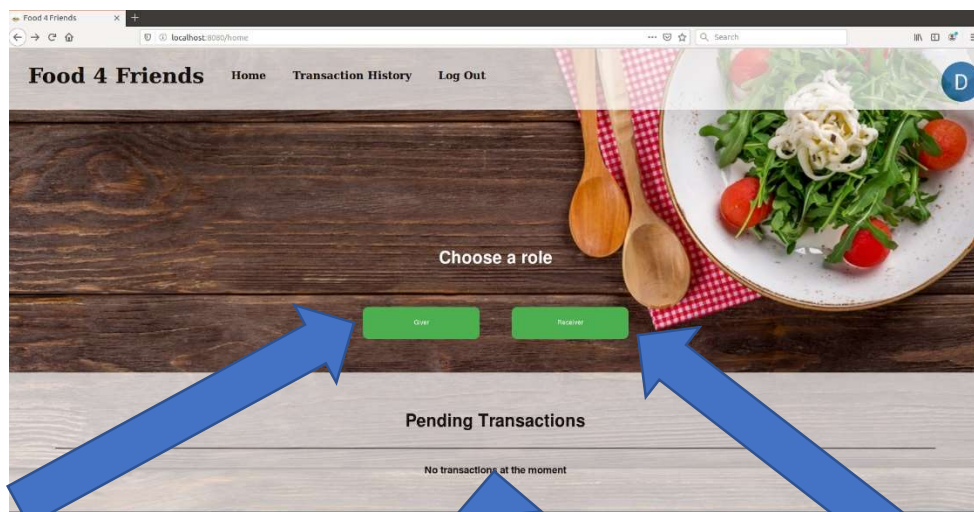
2. Next the user will click on the green “Sign in with Google” button. If the user is not already signed in with a Google account, they will be prompted to sign in.





Use Your Google credentials to log in!

3. In this new window the user will sign in using their Google credentials. Once signed in they will be directed to the “Food for Friends” home page.

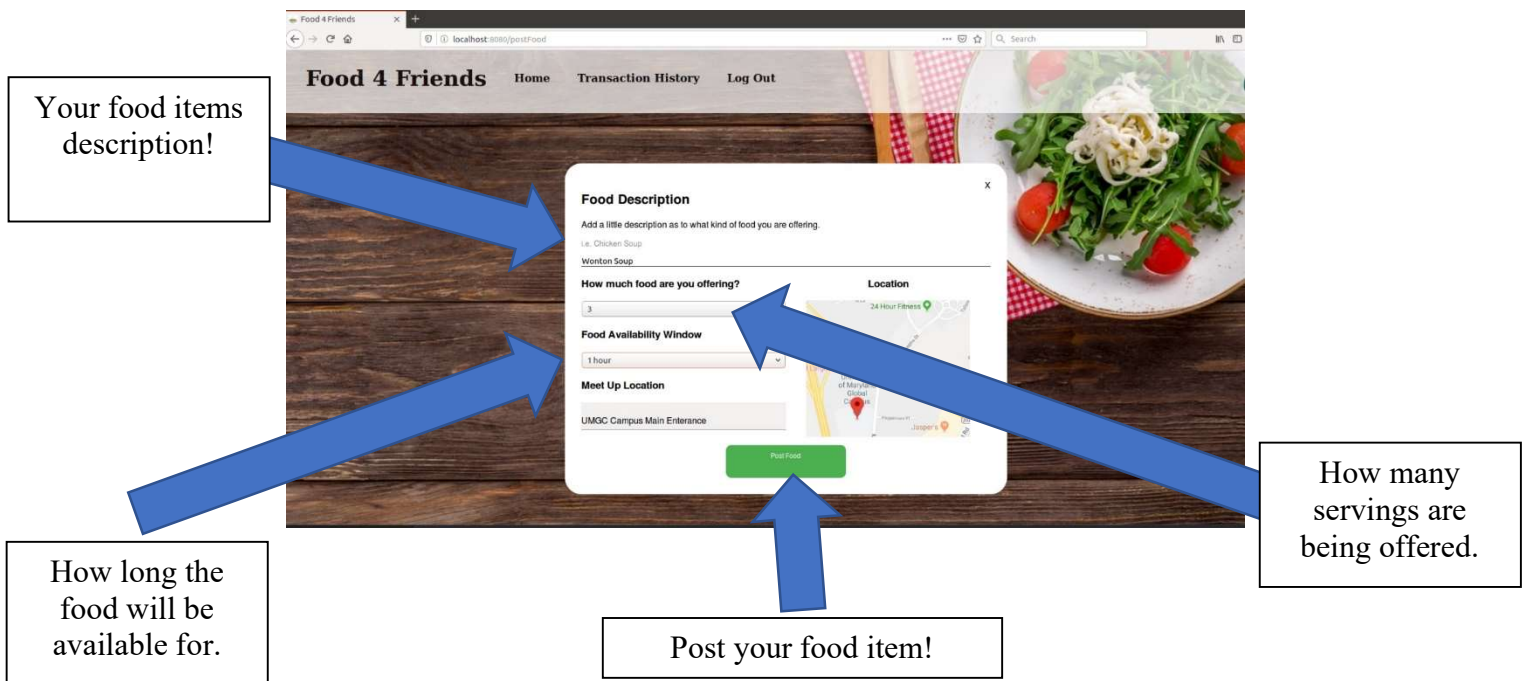


Click here for “Giver” Mode

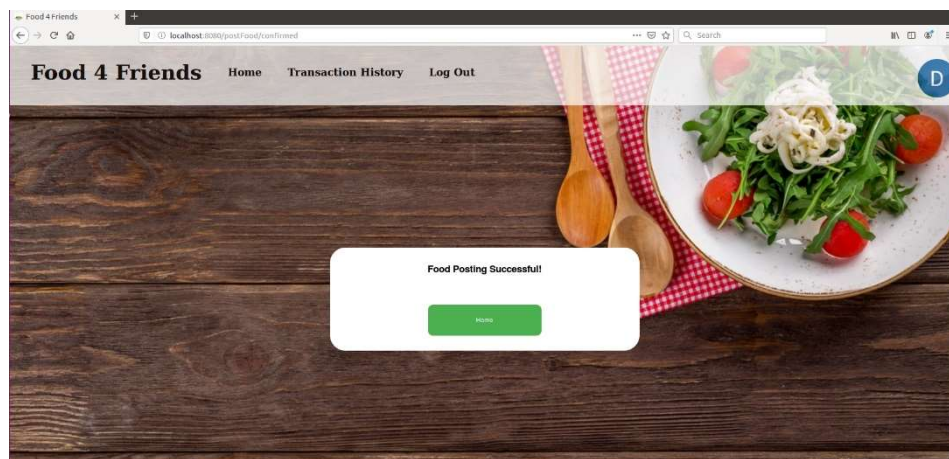
Pending transactions appear here!

Click here for “Receiver” Mode

4. (See Above) On this screen the user will be able to view any of their pending transactions, and select “Giver” or “Receiver” mode. If you are not going to use “Giver” mode skip to step 10.
5. To post a food item select the green “Giver” button, visible above and you will be taken to the post a food item page.

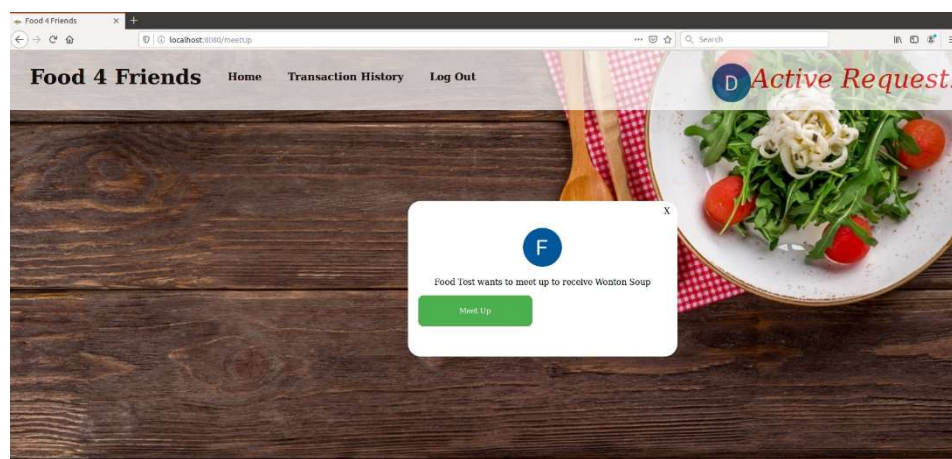
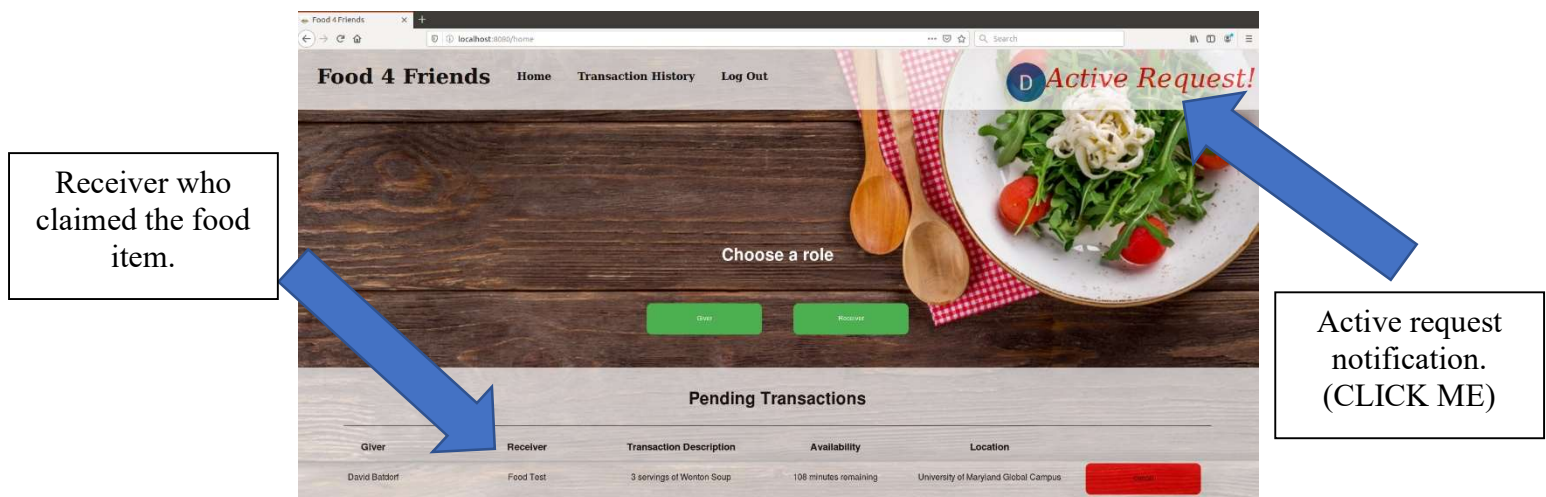


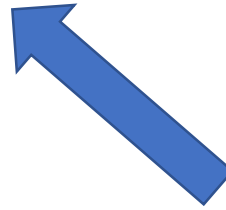
6. The user must add a description of their food offering, along with a number of servings and how long the food will be available, in hours. The meet up location is currently limited to UMGCs Adelphi MD campus at the main entrance.
7. Once the user posts the food offering a confirmation popup appears. The popup contains a button to return the user to the home page from step 3, where they will be able to see their newly posted food item, along with its remaining availability time. The user also has the option to cancel the food item from this page. The user will not be able to view their own food postings from the “Receiver” page.





8. At any time after you have posted your food, refresh the page to see if someone claimed it. Clicking on the Active Request notification allows givers to confirm the exchange, and send the request to the transaction history.





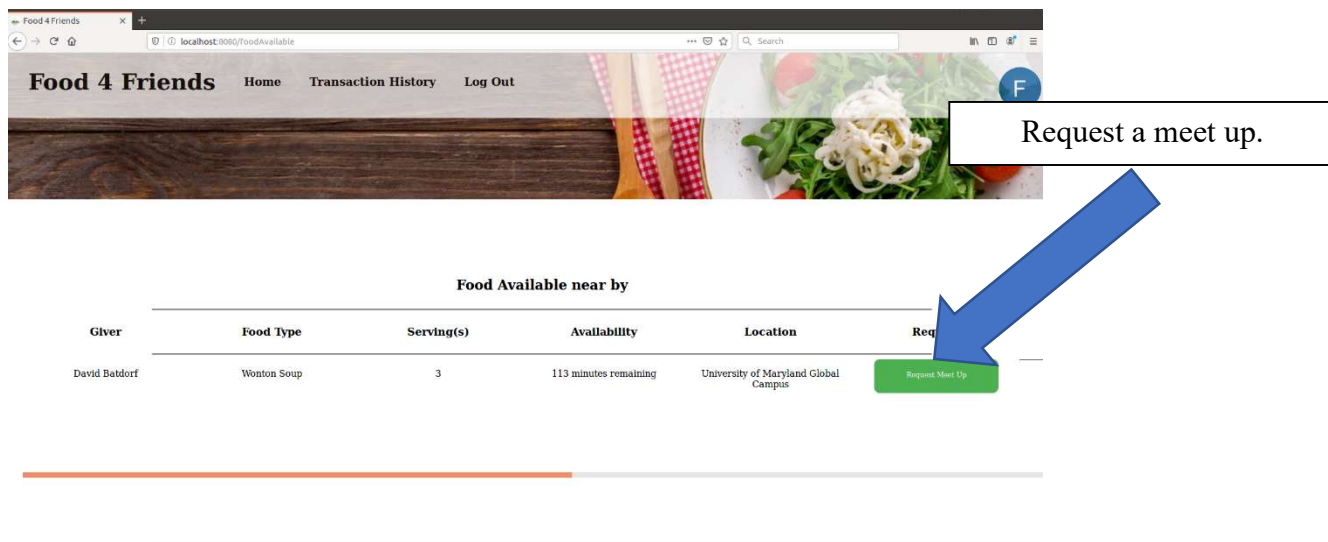
Click here to confirm the offering and send it to transaction history.

9. Once a “Receiver” initiates a meet up, the “Giver” receives a notification to accept the meet up, and they are expected to proceed to the meet up location within 30 minutes, currently only UMGC’s front entrance in Adelphi, MD.

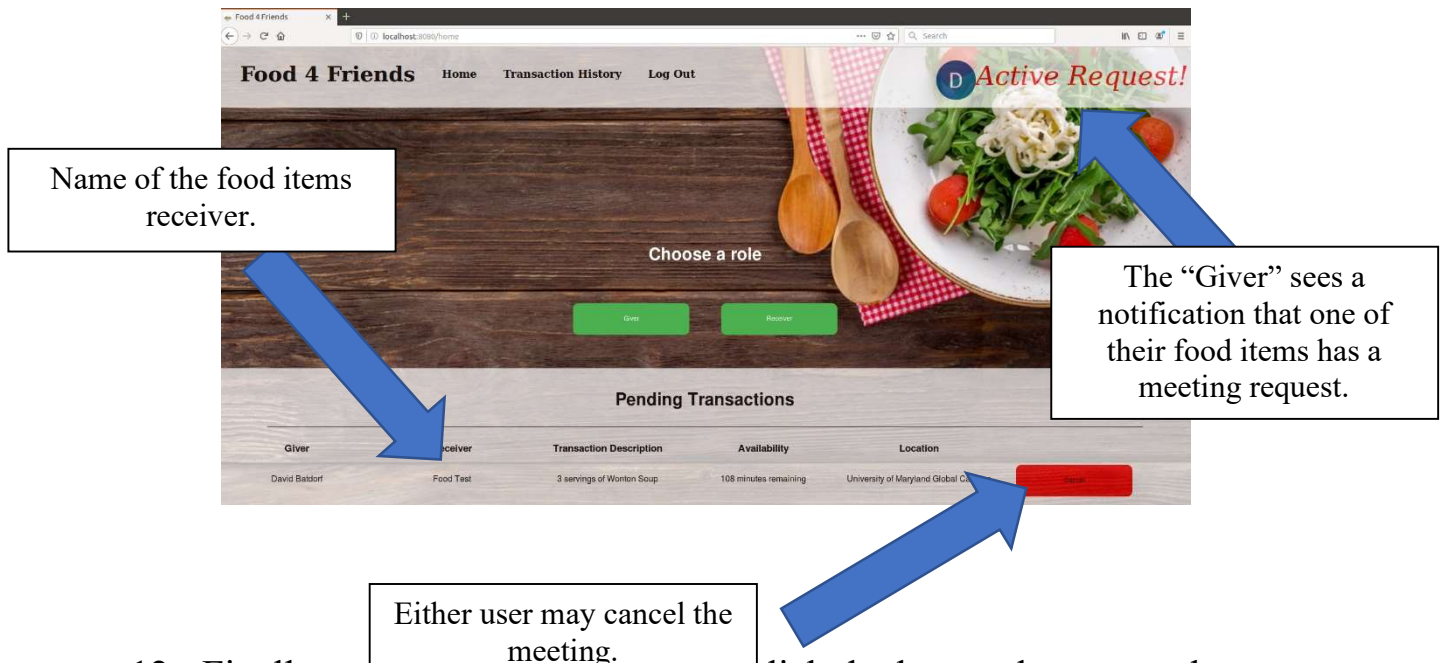


10. If a “Receiver” initiates a meet up, meet ups are expected to happen at the defined location within 30 minutes of the “Receiver” claiming the food at the location shown above.

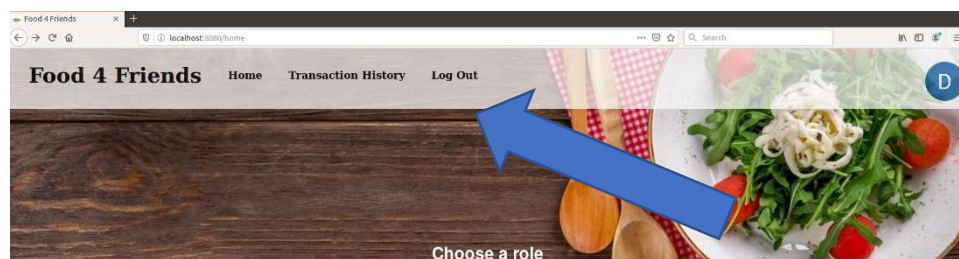




11. If either party cancel's the meeting the food will need to be reposted, to ensure other individuals have adequate time to claim the food, thus preventing food waste.



12. Finally, users can click the log out button on the top center of the page to return to the start, to end their current usage, or change accounts.



Log out to return to the  
sign in page.

## 7. Test Plan and Results.

### 7.1 Test Plan Introduction

#### Food 4 Friends Web Application Test Plan

As we looked to have a flawless product launch, it was evident that a well-planned test plan was pivotal to introduce our Food 4 Friends web application. To develop a solid functioning application, we executed a test plan using an Agile Model. Ambler stated the key to software development success is keep all project stakeholders engaged in two-way communication to strive to develop the simplest solution possible that meets all of your needs (2018). We at the KISS group, have coupled a product test plan to compliment the project plan. A simple test plan lays a road map for how we will test the Food 4 Friends successfully validates requirements criteria.

Two principle tenets and a few precepts were established to validate under this test plan as users navigate application. First principle tenet, the application should be tailored to create an avenue for an individual or group (Giver) with any excess food a central location to post the availability of their free food. Next principle tenet, the submitted food will be posted in a map-like environment, upon which personnel (Recipient) can navigate based on either availability, gastronomic preference, or location. Additionally, since the principle tenets were the keystones of the application, two precepts to include application security, and application integrity will be required to ensure the application is vested. Finally, the Giver and/or Recipient application flow path should be user friendly. Please refer to Figure 1 below of simplified flow path.

### 7.2 High Level Cumulative Interaction

1. Jane (Giver) signs into the web application “Food 4 Friends”

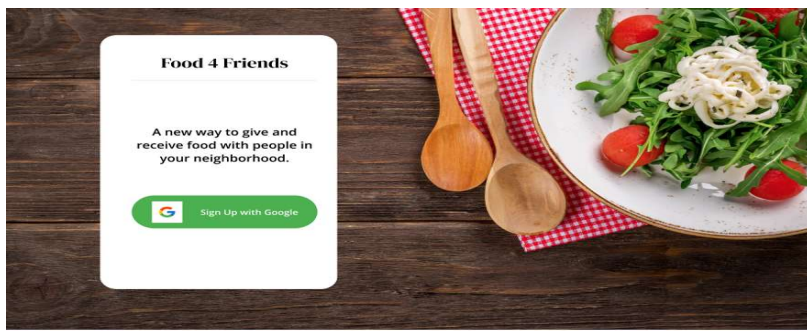


Figure 7.1. Entering site.

2. Signing into is done via Google

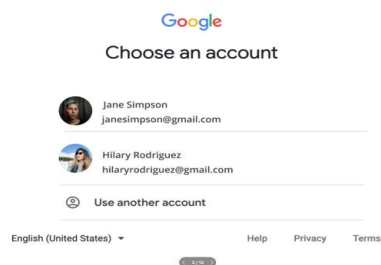


Figure 7.2. User selects account

3. Upon successful entry, user role prompt is displayed for Jane (Giver)
  - a. Selects 'Giver' mode
  - b. Does not select 'Receiver'

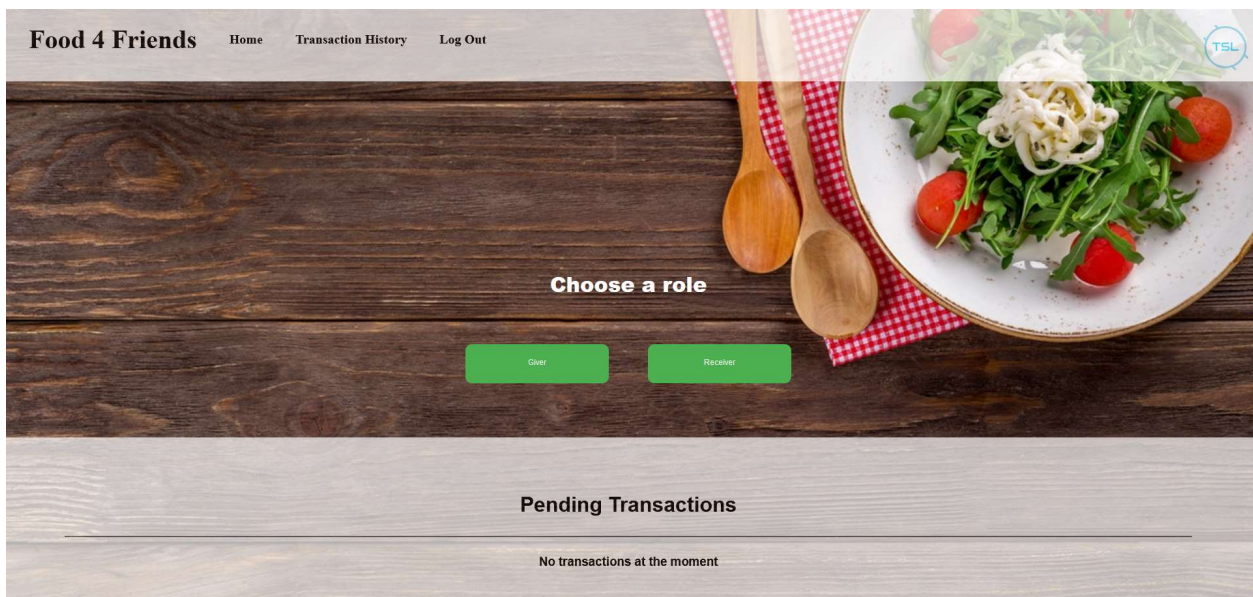


Figure 7.3. Choose a role prompt

4. Follow up data entry page displays the following entry fields:
  - a. Message: Food Description

- b.
  - i.e. Chicken Soup
  - i. Manual entry field
- 5. Message: How much food are you offering?
  - a. Number of Servings (#)
  - b. Drop down restricted
  - c. Message: Food Availability Window
    - i. Drop down restricted
  - d. Message: Type Address
  - e. Example: Rockville Library
  - f. Message: Location
    - i. Intermediate sized locator map
    - ii. Red pin to indicate location of food

**Food 4 Friends** Home Transaction History Log Out

**Food Description** X

Add a little description as to what kind of food you are offering.

i.e. Chicken Soup

**How much food are you offering?**

**Food Availability Window**

**Meet Up Location**

UMGC Campus Main Entrance

**Location**

24 Hour Fitness

University of Maryland Global Campus

Jasper's

Post Food

Figure 7.4. Food selection entry form

- 6. Jane (Giver) finishes data entry then to complete transaction selects Post Food button
- 7. Jane (Giver) views a confirmation status pop-up
  - a. Message: Food Posting Successful
  - b. Button: To close prompt Home button is depressed



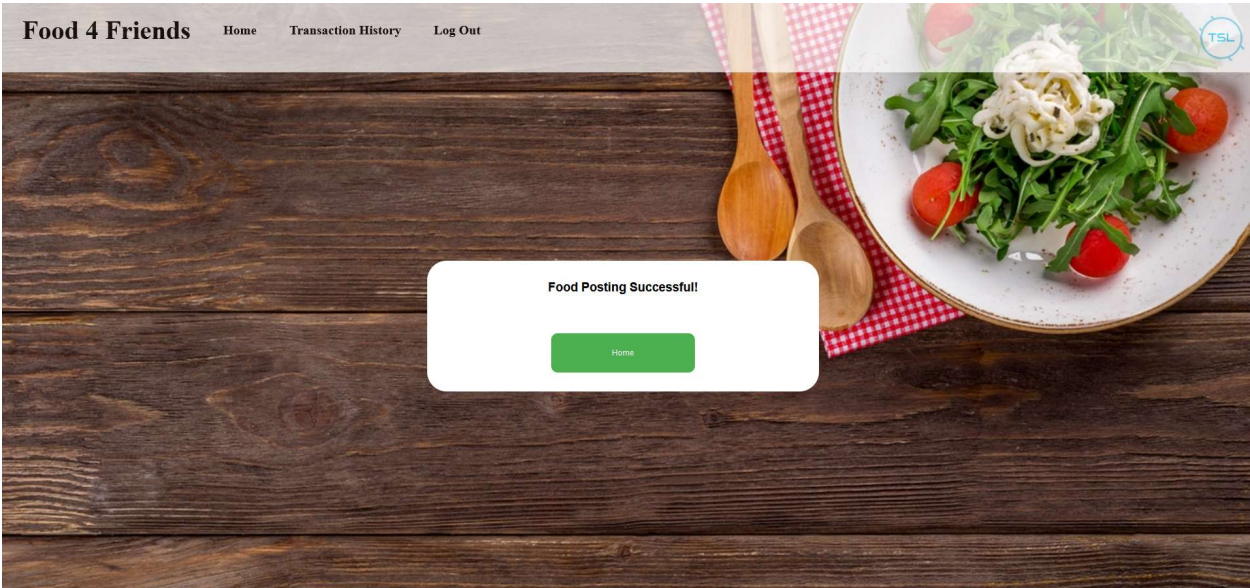


Figure 7.5. Confirmation pop-up

- 6. Hilary (Recipient) signs into the application
- 7. Hilary selects Receiver Mode

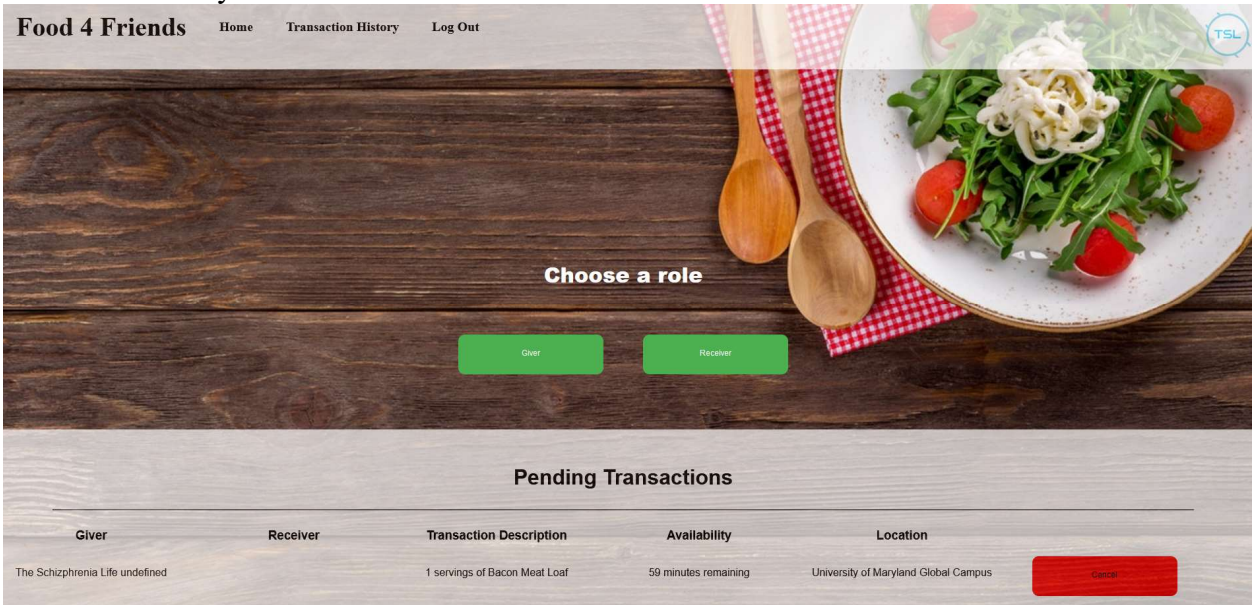


Figure 7.6. Hilary selects Receiver role

8. Hilary (Recipient) sees Jane (Giver) nearby post as located 5 miles away on follow up page.
  - a. Selection is performed by selecting 'Request Meet Up' button

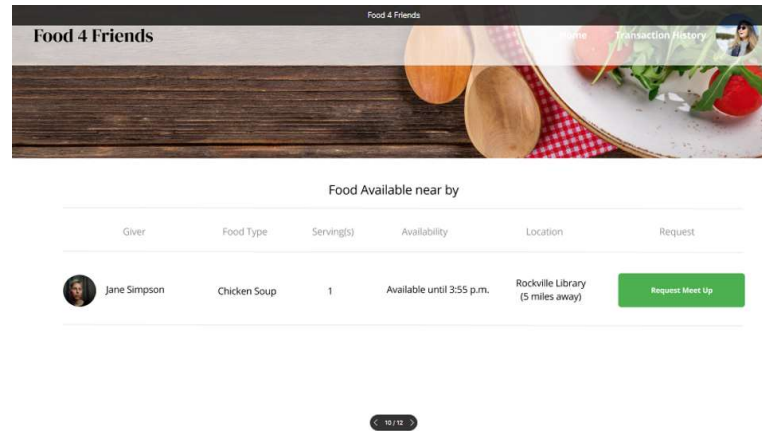


Figure 7.7. Hilary selects Request Meet Up

9. Hilary (Recipient) selects Jane (Giver) post via a pin on her local map
10. Dual message go out:
  - Jane (Giver) receives the notification of Hilary (Recipient) offering and prompted for action
    - a. Pop-up prompt:
      - i. Message: Hilary wants to meet up to receive Chicken Soup.
      - ii. Button: Selects 'Meet Up' button to continue transaction

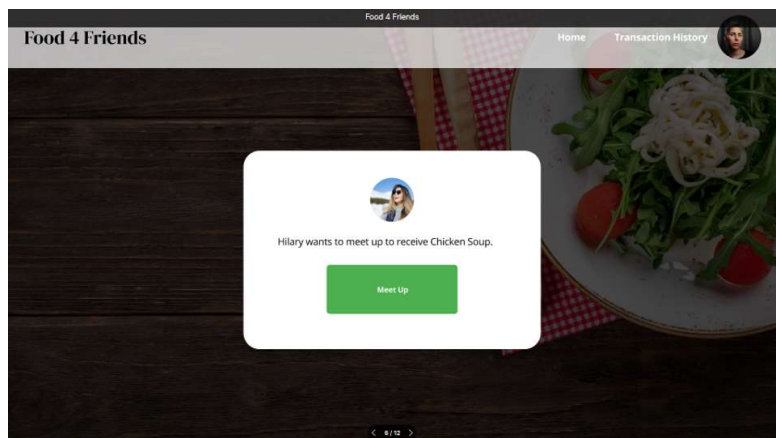


Figure 7.8. Two parties linked prompt - Giver

- Hilary (Recipient) receives the notification of Jane (Giver) offering and prompted for action
  - a. Pop-up prompt:
    - iii. Message: Jane wants to meet up to give you Chicken Soup.
    - iv. Button: Selects 'Meet Up' button to continue transaction

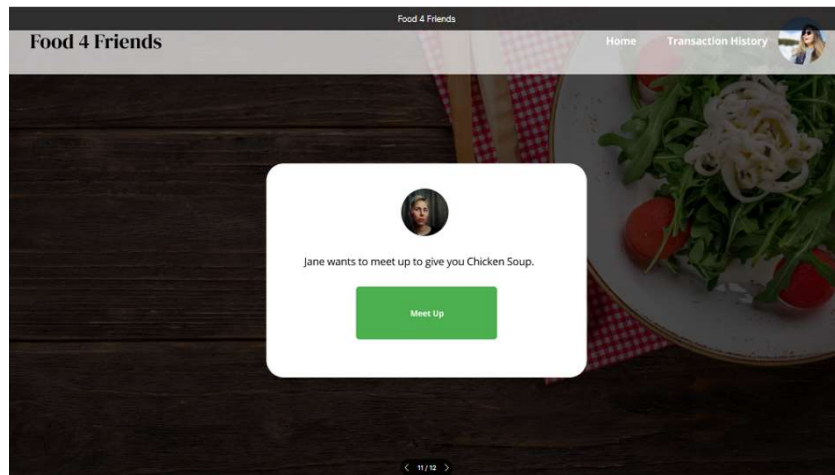


Figure 7.9. Two parties linked prompt - Recipient

11. Internally the application links Hilary (Recipient) and Jane (Giver), allowing them to rate the interaction.
- a. Message: Did you meet up with Hilary or Jane?
  - b. Message: If so, how was your experience?
  - c. Stratification schema: Selectable 5 star ranking of interaction
  - d. Transaction summary available upon completion

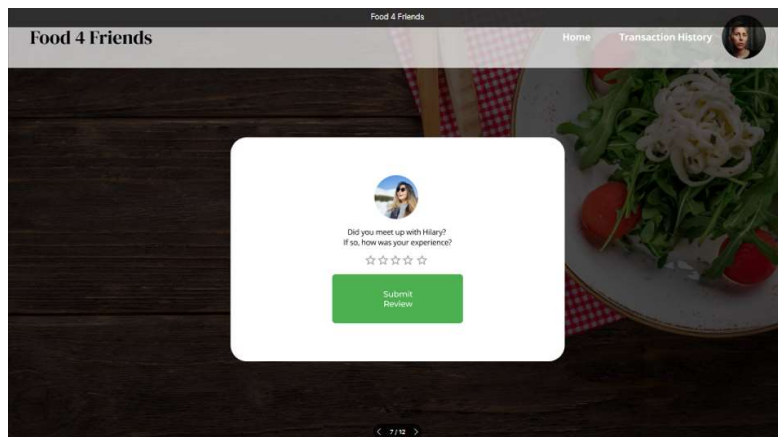


Figure 7.10. Jane is provided a transaction feedback path

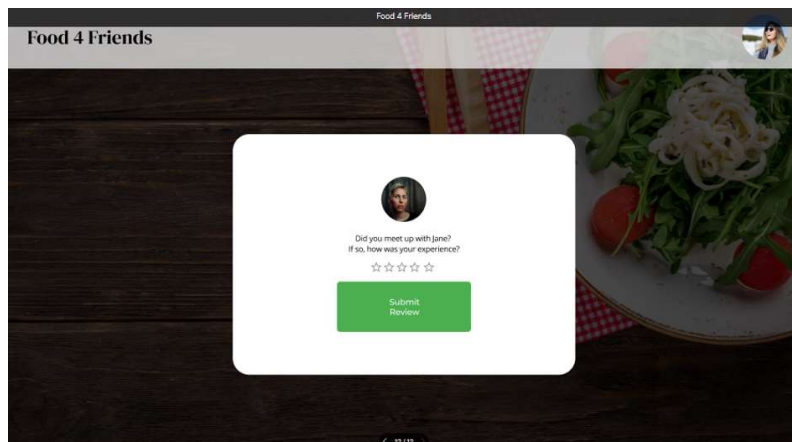


Figure 7.11. Jane is provided a transaction feedback path

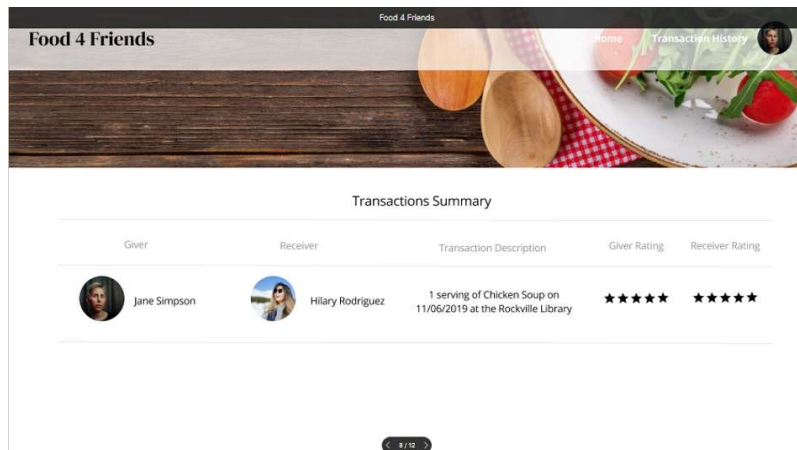


Figure 7.12. Transaction Summary



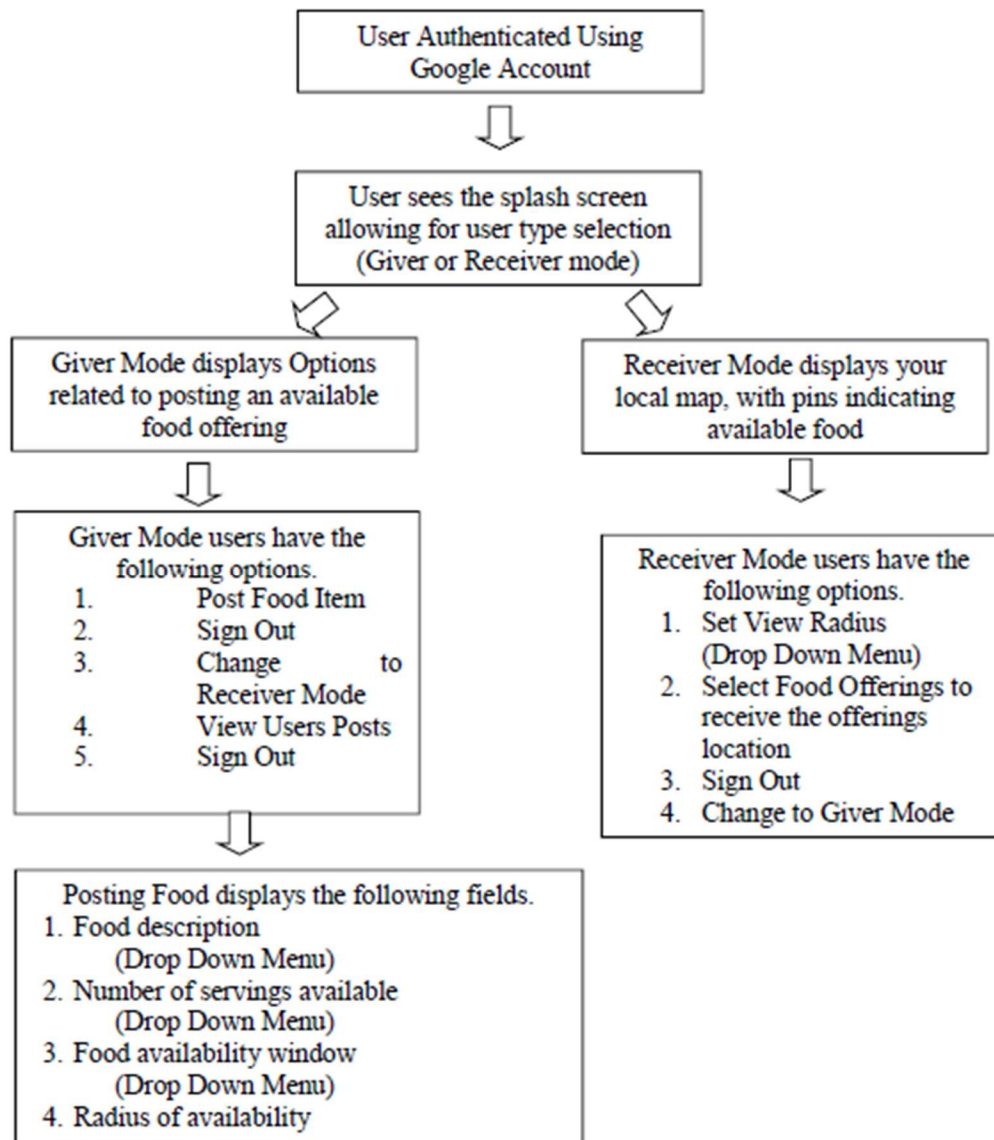
**7.3 Giver or Recipient Flow Path**

Figure 7.13. Giver or Recipient Flow Path

## 7.4 Sign Up Interaction 1

Test Scenario ID	Sign Up Interaction 1				Test Case ID	SU_1.1
Test Case Description	Sign Up - Positive Test Case				Test Priority	Medium
Pre-Requisite	Access to Computer with Internet Accessed Browser				Post-Requisite	N/A
Test Execution Steps:						
Scenario Ordinal	Action	Inputs	Expected Outputs	Actual Outputs	Test Comment	
SU_1.1a	Initial user interfaces with the New Account Registration page	Selects Google Choose an Account	Can see-->  • New account registration hyperlink bring up choosing separate account	Can see-->  • New account registration hyperlink bring up choosing separate account	Pass	
SU_2.1b	New page opens to access new account registration	Views Google Choose an Account	Is content there:  Sees site content: • Language Selection • If applicable, select a different account	Is content there:  Sees site content: • Language Selection • If applicable, select a different account	Pass	

## 7.5 Sign Up Interaction 2

Test Scenario ID	Sign Up Interaction 2			Test Case ID	SU_2.1
Test Case Description	Sign Up - Negative Test Case			Test Priority	High
Pre-Requisite	Access to Computer with Internet Accessed Browser			Post-Requisite	N/A
Test Execution Steps:					
Scenario Ordinal	Action	Inputs	Expected Outputs	Actual Outputs	Test Comment
SU_2.1a	Initial user interfaces with the New Account Registration page	Selects Google Choose an Account	Can see-->  • New account registration hyperlink fails to open	Can see-->  • New account registration hyperlink fails to open	Pass
SU_2.1b	New page opens to access new account registration	Views Google Choose an Account	Is content there:  Does not see site content or fields will not populate : • No content linked	Is content there:  Does not see site content or fields will not populate : • No content linked	Pass

## 7.6 Giver Interaction 1

<b>Test Scenario ID</b>	Giver Interaction 1			<b>Test Case ID</b>	GI_1.1
<b>Test Case Description</b>	Giver Interaction - Positive Test Case			<b>Test Priority</b>	Low

Pre-Requisite	• Access to Computer with Internet Accessed Browser • User account created previously			Post-Requisite	N/A
Test Execution Steps:					
Scenario Ordinal	Action	Inputs	Expected Outputs	Actual Outputs	Test Comment
GI_1.1a	Login Page -- user interfaces with it	Selects Google Account Login Page hyperlink	Successfully enters account into --> Google Account	Successfully enters account into --> Google Account	Pass
GI_1.1b	Mode selection page -- fork in path to one of two modes	Selects Giver mode button	Giver mode properties enabled	Giver mode properties enabled	Pass
GI_1.1c	Giver Mode selection page -- Posting Food	Selects Post Food Item mode button	Selects Post Food Item fields enabled and ready to go	Selects Post Food Item fields enabled and ready to go	Pass
GI_1.1d	Mode selection page -- Sample Food Entry	Giver enters the following information: a. Message: Food Description ii. Manual entry field of Chicken Soup b. Message: How much food are you offering? i. Number of Servings (#) ii. Enters 5 cups in Drop down c. Message: Food Availability Window i. Drop down selects 2 hours d. Message: Type Address i. Lists Rockville Library e. Message: Location i. Intermediate sized locator map	Verify the following information: a. Message: Food Description ii. Manual entry field of Chicken Soup b. Message: How much food are you offering? i. Number of Servings (#) ii. Enters 5 cups in Drop down c. Message: Food Availability Window i. Drop down selects 2 hours d. Message: Type Address i. Lists Rockville Library e. Message: Location i. Intermediate sized locator map	Verify the following information: a. Message: Food Description ii. Manual entry field of Chicken Soup b. Message: How much food are you offering? i. Number of Servings (#) ii. Enters 5 cups in Drop down c. Message: Food Availability Window i. Drop down selects 2 hours d. Message: Type Address i. Lists Rockville Library e. Message: Location i. Intermediate sized locator map	Pass
GI_1.1e	Giver Mode selection page -- Posting Food	After selecting food above, Selects Post Food Item button	Selects Post Food Item now posted on map grid	Selects Post Food Item now posted on map grid	Pass
GI_1.1f	Giver and Recipient transaction	Giver Mode selection page -- Giver and Recipient transaction selectable	Giver Mode selection page -- Giver and Recipient transaction ensues	Giver Mode selection page -- Giver and Recipient transaction ensues	Pass
GI_1.1g	Post Giver and Recipient transaction	Post Giver and Recipient transaction a feedback message appears	Post Giver and Recipient transaction a feedback message appears and can be completed	Post Giver and Recipient transaction a feedback message appears and can be completed	Pass

## 7.7 Giver Interaction 2

Test Scenario ID	Giver Interaction 2			Test Case ID	GI_2.1
Test Case Description	Giver Interaction - Negative Test Case			Test Priority	Low
Pre-Requisite	• Access to Computer with Internet Accessed Browser • User account created previously			Post-Requisite	N/A
Test Execution Steps:					
Scenario Ordinal	Action	Inputs	Expected Outputs	Actual Outputs	Test Comment
GI_2.1a	Login Page -- user interfaces with it	Selects Google Account Login Page hyperlink	Entering your Google Account fails	Entering your Google Account fails	Pass
GI_2.1b	Mode selection page -- fork in path to one of two modes	Selects Giver mode button	Giver mode properties Not enabled and/or wrong mode resulted	Giver mode properties Not enabled and/or wrong mode resulted	Pass
GI_2.1c	Giver Mode selection page -- Posting Food	Selects Post Food Item mode button	Selects Food Item fields disabled or SQL injection vulnerable	Selects Food Item fields disabled or SQL injection vulnerable	Pass
GI_2.1d	Mode selection page -- Sample Food Entry	Giver enters the following information: a. Message: Food Description ii. Add symbols !@\$% b. Message: How much food are you offering? i. enter manually Enters 5 cups in Drop down c. Message: Food Availability Window i. enter manually 2 hours d. Message: Type Address i. Lists Rockville Library e. Message: Location i. Intermediate sized locator map shows Egypt	Giver can't select the following information: a. Food Description: !@\$%^ (manual input) b. Number of Servings Available: 5 (manual input) c. Food availability window: 3 hours (manual input) d. Radius of availability: 10 miles (manual input) Intermediate sized locator map shows Egypt	Giver can't select the following information: a. Food Description: !@\$%^ (manual input) b. Number of Servings Available: 5 (manual input) c. Food availability window: 3 hours (manual input) d. Radius of availability: 10 miles (manual input) Intermediate sized locator map shows Egypt	Pass
GI_2.1e	Giver Mode selection page -- Posting Food	After selecting food above, Selects Post Food Item button	Selects Post Food Item now does not display on map grid	Selects Post Food Item now does not display on map grid	Pass
GI_2.1f	Giver and Recipient transaction	Giver Mode selection page -- Giver and Recipient transaction selectable	Giver Mode selection page -- Giver and Recipient transaction can't connect if out of desired range	Giver Mode selection page -- Giver and Recipient transaction can't connect if out of desired range	Pass

GI_2.1g	Post Giver and Recipient transaction	Post Giver and Recipient transaction a feedback message appears	Post Giver and Recipient transaction a feedback message never appears	Post Giver and Recipient transaction a feedback message never appears	Pass
---------	--------------------------------------	---	---	---	------

## 7.8 Recipient Interaction 1

Test Scenario ID	Recipient Interaction 1			Test Case ID	RI_1.1
Test Case Description	Recipient Interaction - Positive Test Case			Test Priority	Low
Pre-Requirement	• Access to Computer with Internet Accessed Browser • User account created previously			Post-Requirement	N/A
Test Execution Steps:					
Scenario Ordinal	Action	Inputs	Expected Outputs	Actual Outputs	Test Comment
RI_1.1a	Login Page -- user interfaces with it	Selects Login Page hyperlink	Successfully enters account into --> • User Login field • Password field	Successfully enters account into --> • User Login field • Password field	Pass
RI_1.1b	Mode selection page -- fork in path to one of two modes	Selects Recipient mode button	Recipient mode properties enabled	Recipient mode properties enabled	Pass
RI_1.1c	Recipient Mode selection page -- Posting Food	Selects Post Food Item mode button	Selects Post Food Item fields enabled and ready to go	Selects Post Food Item fields enabled and ready to go	Pass
RI_1.1d	Mode selection page -- Sample Food Entry	Recipient looks for available food	Recipient sees and selects the following information:  a. Food Description: Chicken Soup	Recipient sees and selects the following information:  a. Food Description: Chicken Soup	Pass
RI_1.1e	Recipient Mode selection page -- Posting Food	After selecting food above, Selects Post Food Item button	Selects Post Food Item now posted on map grid	Selects Post Food Item now posted on map grid	Pass
RI_1.1f	Recipient and Giver transaction	Recipient Mode selection page -- Recipient and Giver transaction selectable	Recipient Mode selection page -- Recipient and Giver transaction ensues	Recipient Mode selection page -- Recipient and Giver transaction ensues	Pass
RI_1.1g	Post Recipient and Giver transaction	Post Recipient and Giver transaction a feedback message appears	Post Recipient and Giver transaction a feedback message appears and can be completed	Post Recipient and Giver transaction a feedback message appears and can be completed	Pass

## 7.9 Recipient Interaction 2

<b>Test Scenario ID</b>	Recipient Interaction 2			<b>Test Case ID</b>	RI_2.1
<b>Test Case Description</b>	Recipient Interaction - Negative Test Case			<b>Test Priority</b>	Low
<b>Pre-Requisite</b>	<ul style="list-style-type: none"> <li>• Access to Computer with Internet Accessed Browser</li> <li>• User account created previously</li> </ul>			<b>Post-Requisite</b>	N/A

Test Execution Steps:					
Scenario Ordinal	Action	Inputs	Expected Outputs	Actual Outputs	Test Comment
RI_2.1a	Login Page -- user interfaces with it	Selects Login Page hyperlink	Entering your account info fails at Google sign in	Entering your account info fails at Google sign in	Pass
RI_2.1b	Mode selection page -- fork in path to one of two modes	Selects Recipient mode button	Recipient mode properties Not enabled and/or wrong mode resulted	Recipient mode properties Not enabled and/or wrong mode resulted	Pass
RI_2.1c	Recipient Mode selection page -- Posting Food	Selects Post Food Item mode button	Selects Food Item fields disabled or SQL injection vulnerable	Selects Food Item fields disabled or SQL injection vulnerable	Pass
RI_2.1d	Mode selection page -- Sample Food Entry	Recipient looks for a sofa	Recipient can't select the following information: sofa are not allowed	Recipient can't select the following information: sofa are not allowed	Pass
RI_2.1e	Recipient Mode selection page -- Posting Food	After selecting food above, Selects Post Food Item button	Selects Post Food Item now does not display on map grid	Selects Post Food Item now does not display on map grid	Pass
RI_2.1f	Recipient and Giver transaction	Recipient Mode selection page -- Recipient and Giver transaction selectable	Recipient Mode selection page -- Recipient and Giver transaction can't connect if out of desired range	Recipient Mode selection page -- Recipient and Giver transaction can't connect if out of desired range	Pass
RI_2.1g	Post Recipient and Giver transaction	Post Recipient and Giver transaction a feedback message appears	Post Recipient and Giver transaction a feedback message never appears	Post Recipient and Giver transaction a feedback message never appears	Pass

## 7.10 Security Testing

The security test will be done once the application is complete. It starts with manual code review looking for the top vulnerabilities in web applications. Once complete an automated review is done by hosting the application and crawling through it with different apps looking for problems. Once both are done the results are compared to each other to develop a plan on how to fix the bigger issues like CSS or SQL Injection. Normally the review itself it rather quick the fixing of those issues is what takes time. These types of reviews should be done multiple times before the release of an app. Otherwise an application could release with major problems that hackers could exploit resulting in monetary loss or data loss. Every few years

(OWASP, 2019) comes out with list showing the top ways hackers are exploiting current applications. A good starting place is making sure none of those issues are in your application.

Below are the top current areas where problems arise according to (OWASP, 2019) listed from highest priority(M1) to lowest(M10):



## 8. Design and Alternate Designs

### 8.1 Initial Speciation Diagrams

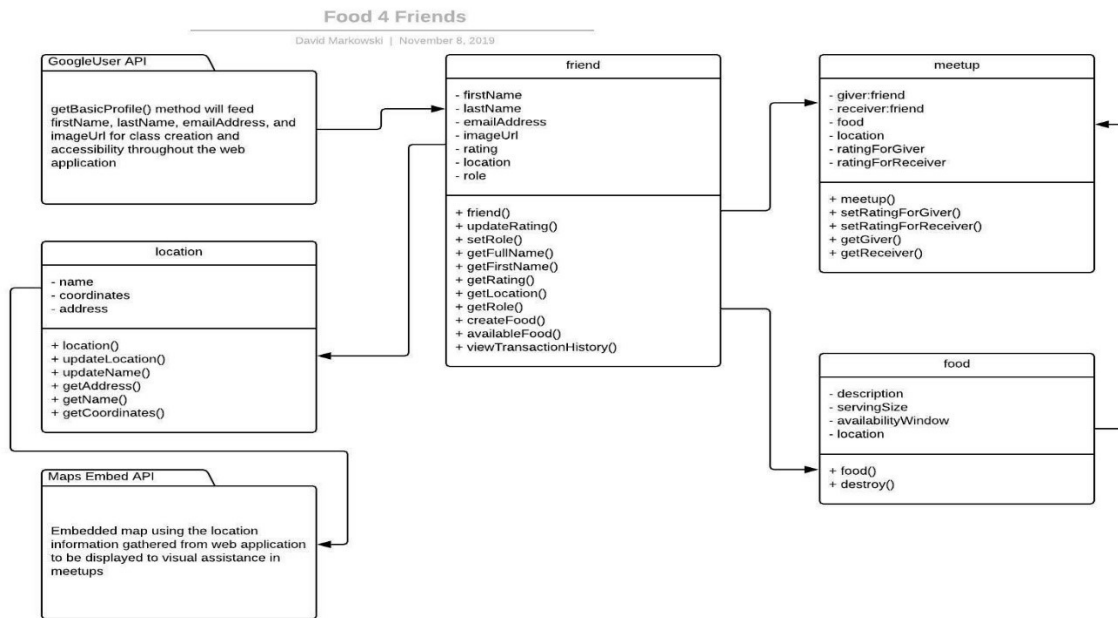


Fig 8.1: initial Classes, Methods, and Fields Design

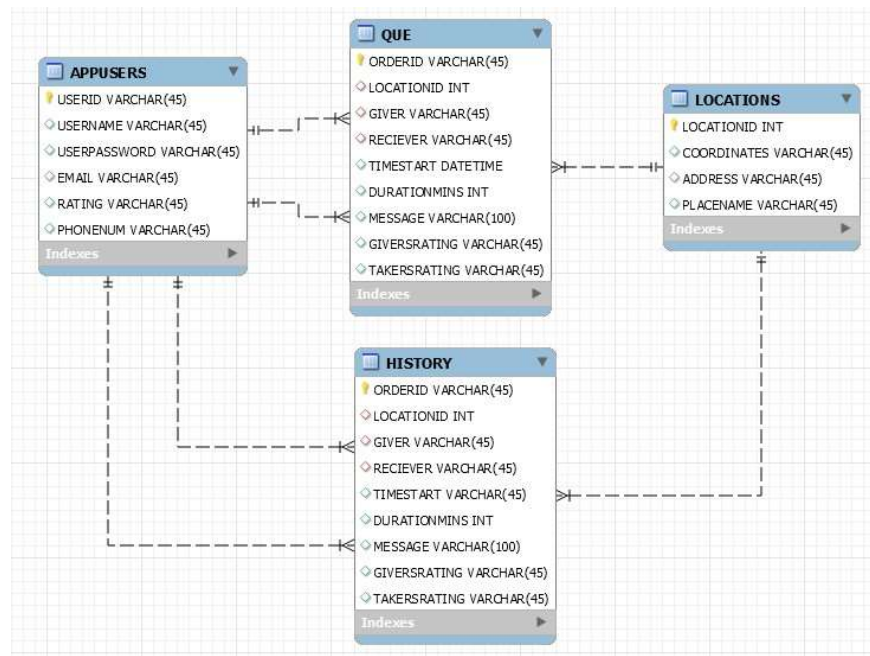


Fig 8.2: Original Database Data Structure Design



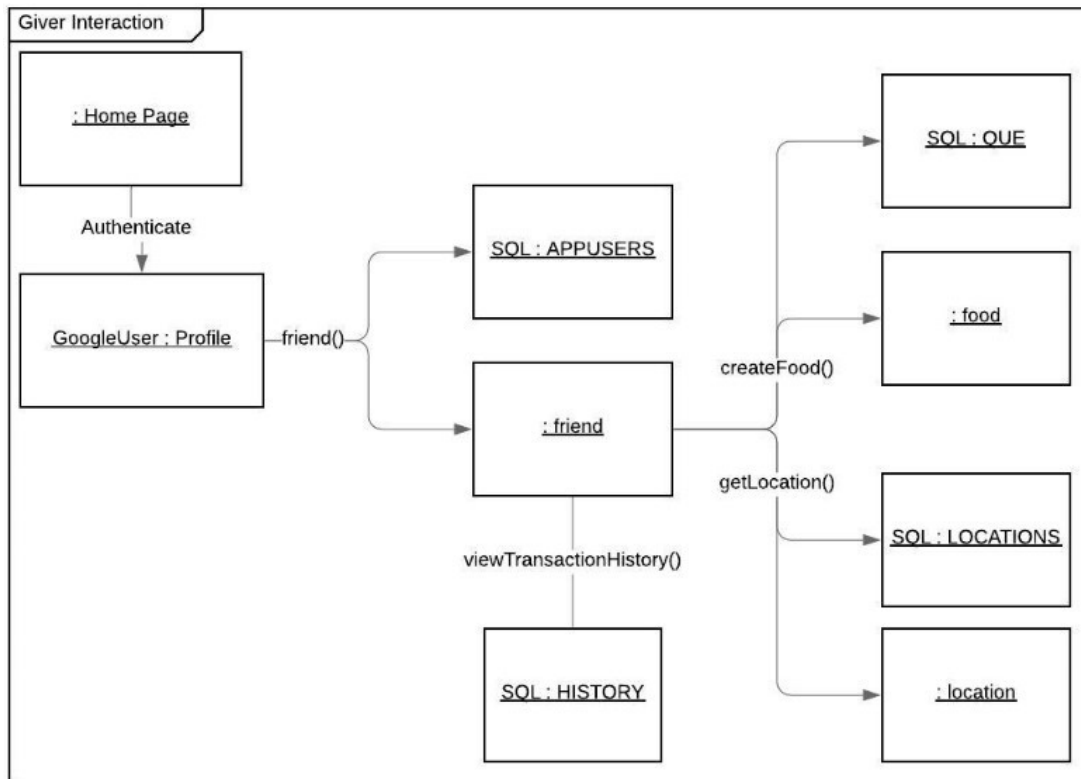


Fig 8.3: Original Giver Communication Flow Design

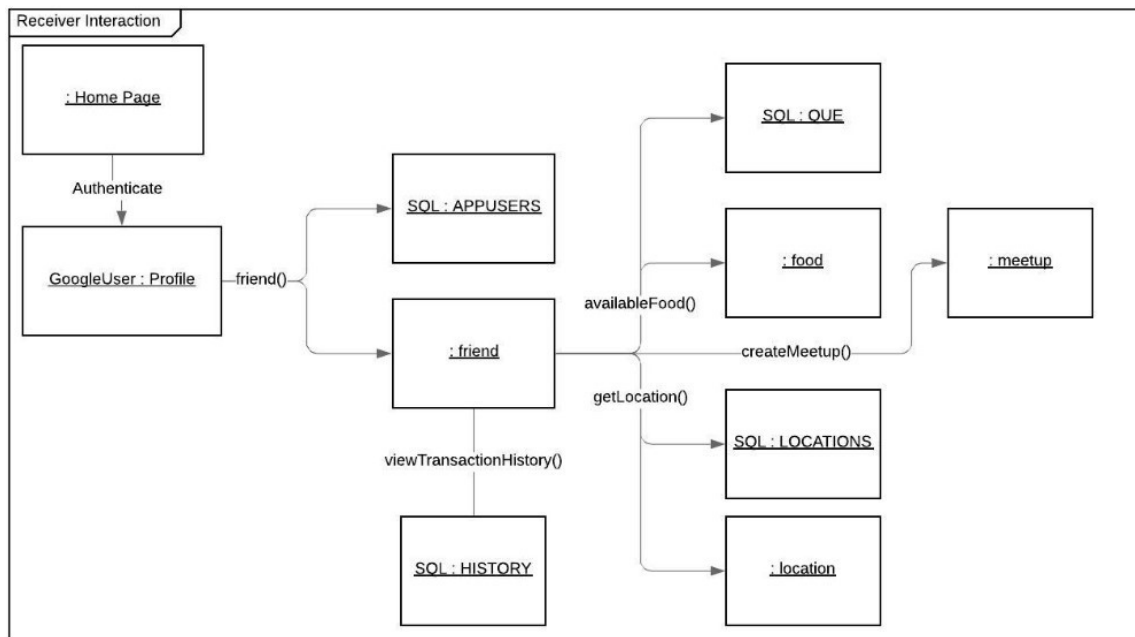


Fig 8.4: Original Receiver Communication Flow Design

## 8.2 Speciation Diagram Updates

Final UMLs are available as separate files within the zip for improved readability.

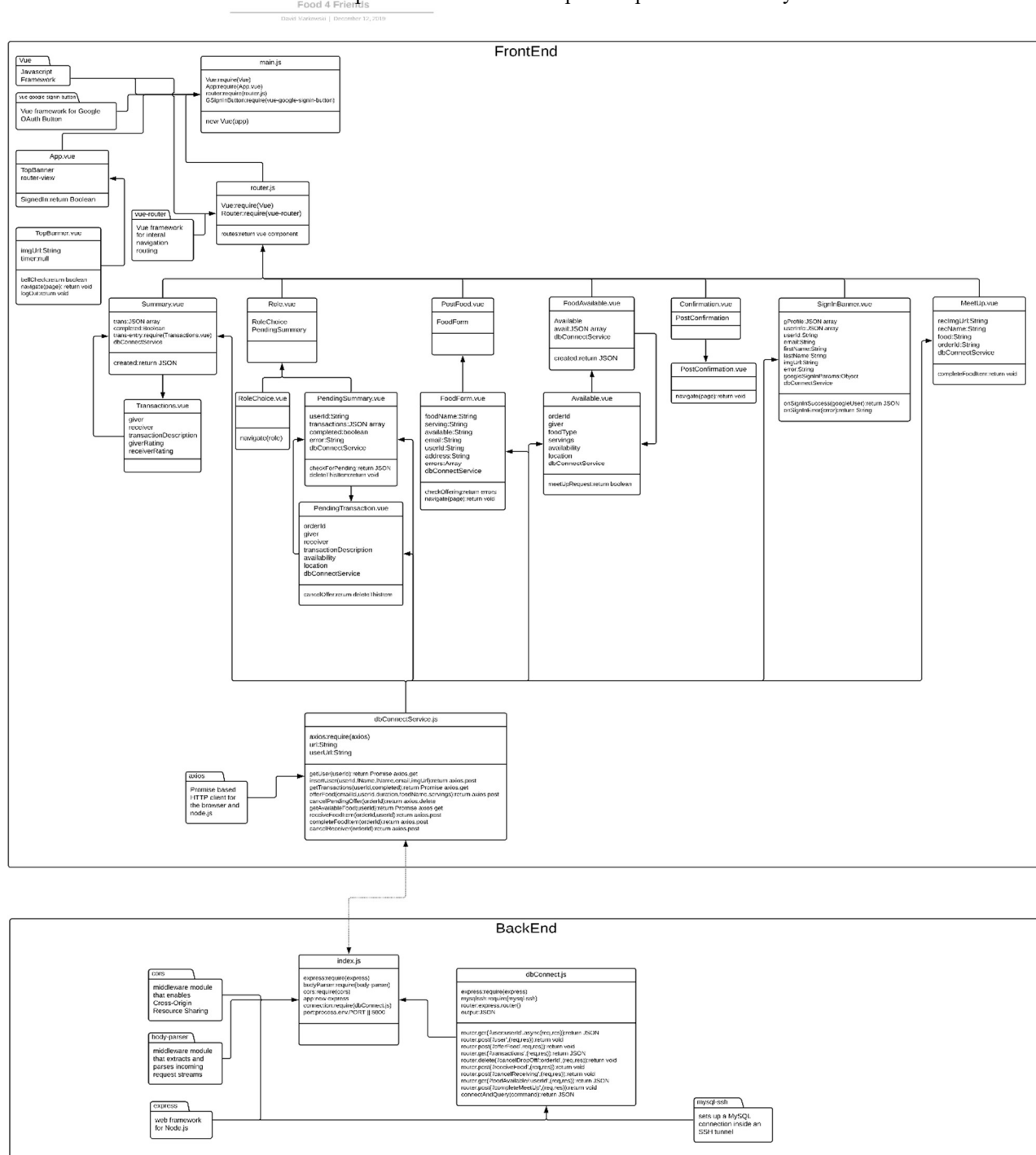


Fig 8.5: Final Class UML diagram  
(Food\_4\_Friends\_-\_Class\_UML.pdf)

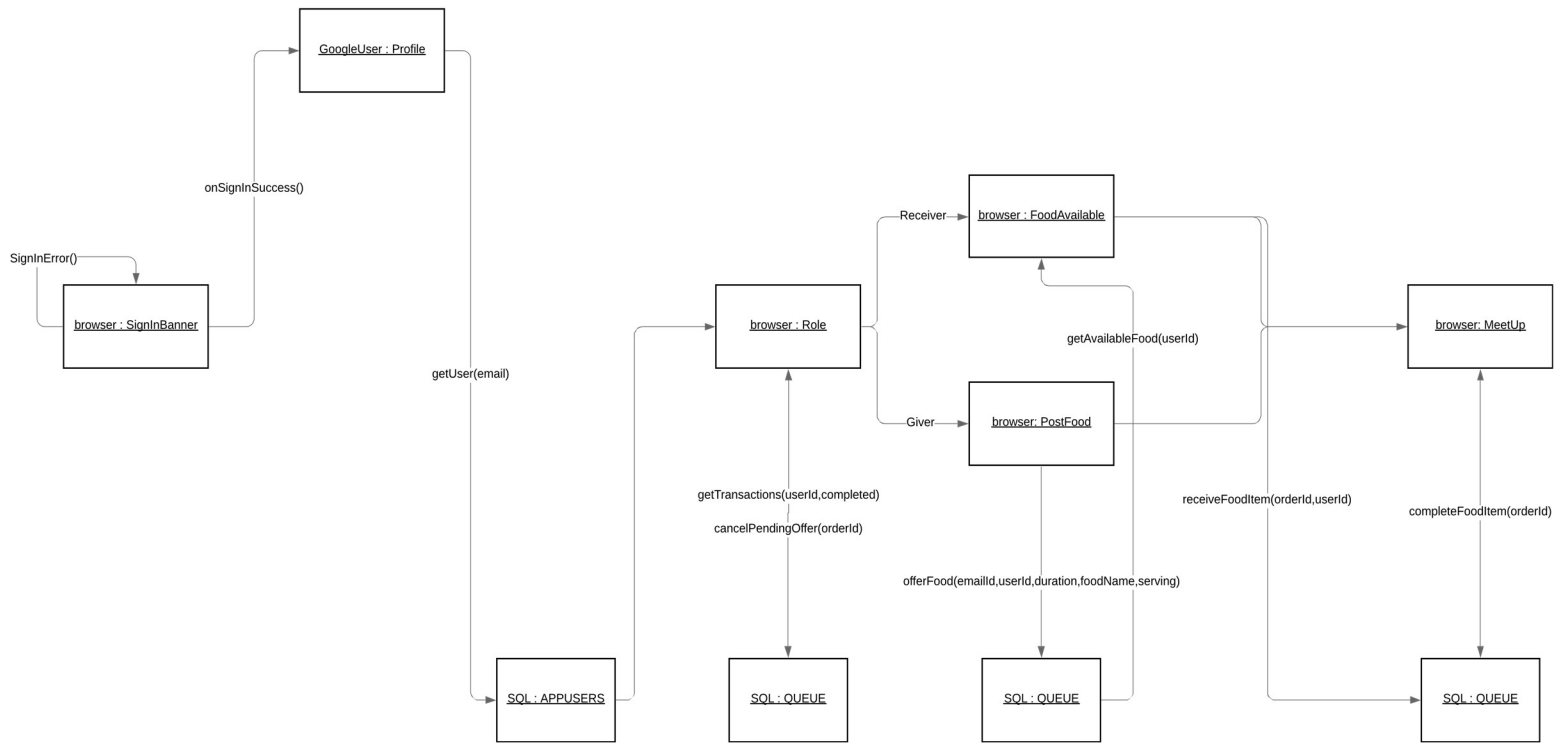


Fig 8.5: Final Communication UML diagram  
(Food\_4\_Friends\_\_Communication\_UML.pdf)

## 9. Development History

Sprint Week	Development Milestones
#1	Initial Project Idea Creation
#2	JavaScript and HTLM selected as program languages.  Initial project requirements list generated.
#3	Vue.js Framework selected.  User interface and basic project flow defined via Figma flow diagrams.
#4	Vue.js project created.  Translated Figma diagrams into HTLM interfaces.  Created Project Design document, outlining how the classes will store data, and interacted. This document also defined planned data structures, and sample user interactions.
#5	Created application sign in page  Created application role selection page  Created giver mode page  Created receiver mode page  Created transaction history page
#6	Integrated remote database into all existing pages to allow connectivity between users.  Corrected navigation risk discovered by Test Engineer.
#7	Integrated Google sign in API into the application

	Added meet up acceptance by “Giver” user.
	Corrected food item deletion risk discovered by Test Engineer
	Corrected error where canceling a food item caused database instability
	Updated user and food item database key naming convention

## 10. Conclusions

In conclusion this was an extremely challenging and extremely rewarding project. As a team we worked together extremely well, meeting twice weekly to ensure that all members are aware of changes, and difficulties with the project progress. We ran into many challenges throughout the development effort, and we were able to overcome the majority of them.

As a team we all learned many lessons from this project. We all learned to work as a team, and how to accept we each had a part to play. For many of us this was our first experience working as a team, as other University of Maryland Global Campus are entirely based on solo assignments. As a team we all learned that building any application using a framework beyond the standard language is initially significantly more difficult. In the case of this project we used Vue.js as our framework. The initial set-up was much more complicated, but once we got the project set up, expanding upon the project was much simpler than a project without a framework. The nature of the framework also forced the development team to use correct software engineering principles, including modularity and information hiding. David Markowski, the team’s software engineer, also learned about the importance of documentation. While implementing the Google sign in API several system requirements, that were undocumented by Google caused him development setbacks. The final thing we learned is exactly how complicated simply looking web sites are. Getting end simple navigation was very difficult, and more complex concepts such as database integration, Google API integration, and real time updates very quickly caused us to go over our project budget.

The strength of the “Food for Friends” design lies in three areas. The first and most important is usability. The Graphical user interfaces are designed to be as simple and user friendly as possible. Andrew Delgado’s introduced the team to Figma which is a collaborative interface design tool. With this tool we had a viable user interface before coding began. Secondly the application was designed to run on a large variety of systems. With a simple install the application will run through a web browser on Linux, Windows and OS X. In the future the application will be fully remotely hosted and will be able to run on any web capable devices. The final and greatest strength of the project is its ability to bring people together and make lives better. The application has the ability, with proper guidance to help tackle both hunger, and food waste.

The current iteration of the “Food for Friends” web application currently has several limitations. The first major limitation is the lack of an external web hosting service. Without external hosting the product will be limited to systems that have the application installed. The second limitation is due to the way the database implemented, via an ssh shell, it is prone to disconnects and complete crashes. The project is also limited in deployment area. This limitation was a planned limitation to introduce the project to a small test market initially to gauge usage, estimate hardware expenditures and promote the applications usage. Due to the application only being distributed to a single locality as a market test, Google’s location service was not implemented. In the future Google’s map location service will be implemented to allow users over a much broader area to arrange meetings and food exchanges.

The “Food for Friends” application is still very clearly a work in progress with several parts still able to be implemented, and several bugs requiring solutions. The first issue being the lack of a web host. With an external webhost, and a URL, which is required by non-local applications using Google sign in API, the application would be able to reach a much broader audience. The second improvement that could be made to the project would with the database connection. Currently we send SQL commands to a SQLite database via a SSH command. This is not the most stable system currently. To improve upon this, we have considered a few options. The first would be a que-based system to ensure that only a single command is sent at a time. The second and better option, once remote hosting is possible would be to have both the database and web server local to each other and use a connector in a different language such as java. In the future the team would also like to implement a rating system between users. This system would allow both the giver and receiver to rate the interaction, and users would be able to view a user’s rating before meeting for a food exchange. Finally, once the application is distributed to a larger geographic area Google’s location and mapping service will be implemented to allow user control over meeting locations, while retaining the simplistic user interface philosophy.

## 11. Resources

Ambler, S. (2019). Agile Design. Retrieved from <http://www.agilemodeling.com/essays/agileDesign.htm>.

Company Salaries. (n.d.). Retrieved from <https://www.glassdoor.com/Salaries>.

Frequency of Food Insecurity. (n.d.). Retrieved October 31, 2019, from <https://www.ers.usda.gov/topics/food-nutrition-assistance/food-security-in-the-us/frequency-of-food-insecurity/>.

Food Waste FAQs. (n.d.). Retrieved October 31, 2019, from <https://www.usda.gov/foodwaste/faqs>.

Guru99. (2019). Waterfall Vs. Agile: Must Know Differences. Retrieved from <https://www.guru99.com/waterfall-vs-agile.html>.

OWASP. (2019). Mobile Top 10 2016-M2-Insecure Data Storage. Retrieved from [https://www.owasp.org/index.php/Mobile\\_Top\\_10\\_2016-M2-Insecure\\_Data\\_Storage](https://www.owasp.org/index.php/Mobile_Top_10_2016-M2-Insecure_Data_Storage)

TutorialsPoint.Com. (2019). UML - Use Case Diagrams. Retrieved from [https://www.tutorialspoint.com/uml/uml\\_use\\_case\\_diagram.htm](https://www.tutorialspoint.com/uml/uml_use_case_diagram.htm).