Design Studio 2 (part 2)

Team 15 Design Studio 2-1 (ZotFeeds)

1. **Audience and other stakeholders:**
    - **Audience:**
        - Irvine Residents:
            - Individuals and residents of Irvine will operate the application on their local device and create/log in to a food donor account to request food pick up from a nonprofit organization.
        - UCI Dining Services:
            - A big source of food waste from Irvine comes from UCI dining services. Instead of throwing away excess food, they can feed many other residents of Irvine as well.
        - Nonprofits that will organize the reception of the food:
            - These are the organizations that look to supply themselves with food to give for free for individuals who need it. All the food processed through the app will be distributed to the varying organizations' food is food drive signed up.
        - Irvine Restaurants:
            - Restaurants are a major supplier for food banks, they will be able to donate their excess food through the app.
        - Food Donors:
            - Any individual who would casually donate their excess food through the app.
        - Irvine Grocery Stores:
            - Grocery Stores are responsible for a big percentage of food waste. The app will allow them to donate their excess food.
    - **Stakeholders:**
        - Food banks:
            - The food banks in Irvine would be affected and would benefit greatly by knowing of the availability of food that the application provides.
        - The city of Irvine:
            - The city of Irvine will be affected by the application because it will bring new ideas to combat food insecurity.
        - Homeless People:
            - This application will increase food donations to non-profit organizations and local food banks, therefore, increasing the food supply for the homeless who benefit from these organizations.
        - Irvine Recycling and Waste services:
            - The city of Irvine can be directly affected by the reduction of solid waste such as trash, litter, and garbage items. Due to the reduction of waste, Irvine as a city will experience less pollution in their

        local environments and diminish the workload on these types of services.
    - Environment:
        - The environment as a whole would greatly benefit from less food waste as a result of the software's goal to reduce food waste in Irvine.
    - The software developers:
        - The software developers are responsible for creating the software and helping with the design.
    - The software maintainers :
        - The people responsible for maintaining the software would have an effect on the future and current development and maintenance of the software and in turn, would affect the design of the software.

2. **Goals, constraints, assumptions for the overall design solution**
    - **Goals:**
        - Reduce the levels of malnutrition within the Irvine Community.
            - The application will help reduce hunger and malnutrition in the Irvine community by increasing donations to food banks and other non-profit organizations that combat this issue.
        - Reduce food and other disposable waste within Irvine Community.
            - The app will take in requests from food donors. This food will be distributed to nonprofit organizations, thus reducing food waste.
        - The software shall provide good usability.
            - The app will have an intuitive and accessible interface for all users of the app.
        - Speed up the process of the retrieval and delivery of the food to people in need.
            - Connects sources of food to the people that are going to deliver the food together to ensure that everything gets set up faster so the food will get to the people that need it as quickly as possible.
        - Make the process easier for the non-profit organization to collect, process, and distribute donations.
            - With features like direct communication, pickup and delivery confirmations, and estimated arrival times, donation collection and distribution by the non-profit organization will be much more intuitive and efficient.
        - Make the process easier for donors to get rid of their excess food.
            - Traditionally, donors wanting to donate food would need to search for non-profit organizations in their area, get in touch with them, and figure out how/ when/ where to deliver the food. With the app, donors can simply request for food to be picked up and the accepting organization will handle pick-up.
        - The application shall be accessible via different smartphones.

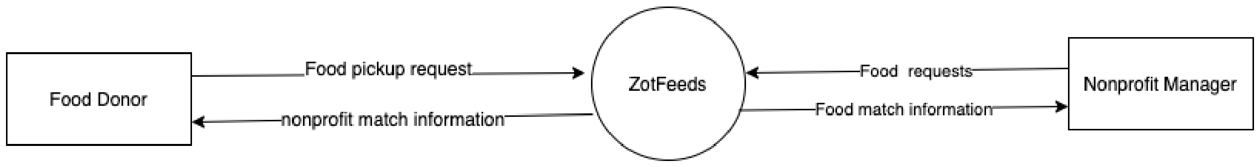
    - Focusing on mobile devices, the app should be downloadable from the Apple and Android app stores so we can reach a wider audience.
  - Ensure the excess food will go to a legitimate and professional nonprofit organization that will deliver to people in need.
    - The application aims to ensure the safe and correct delivery of food to its rightful recipients, who are people who know where to give the food to people in need.

- **Assumptions:**
  - All Nonprofit organizations will have access to the internet and the software.
    - For the organizations to use or download the application, they will need to have internet access.
  - There will always be a need for food.
    - Food is a daily necessity for non-profit organizations. This will mean all food put up for pickup by donors will have a match ready and willing to accept it.
  - All donors will have access to the internet and the software.
    - The donors will need to have internet access to use and download the application.
  - Assume both organizations and food donors will input correct information when submitting a request.
    - If the correct information is given, it reduces the time and effort that the system must do to make changes to those human errors.
  - The organization will be in charge of picking up the food.
    - The app will only focus on connecting the parties, the pickup of the food will be organized by the non-profit organizations.
  - The application will benefit and be used mostly by non-profit organizations
    - With non-profits as the main user, the application will be designed to adhere to that statement.
  - The application is an open-source project, but we shall be able to control and define how the project should be developed.
    - There is a need for restrictions on the project to complete and satisfy the goals of the group. By making this assumption our group will be able to define features for our system.
- **Constraints:**
  - The application can't always guarantee the request of providing food to the nonprofit organizations
    - Due to the limited supply of extra food, the need for food cannot always be fulfilled especially when the demand is higher than the supply.
  - Individuals can <u>not</u> put in a request for food, only non-profit organizations.

        - Requests are exclusive to non-profit organizations because they are the main distributors of donations to those affected by food insecurity.
    - Non-profit organizations can not request again until their current order is fulfilled.
        - This will ensure those who are in the queue will also be considered for the donations accordingly.
    - There will be no budget for designing this application.
        - As this project is done as charitable work to help address an important societal problem, it will not be funded by anyone and the software developers and maintainers will have to work voluntarily.
    - The project completion date is due by Fall 2021.
        - As this project is done as charitable work to help address an important societal problem, launching it as soon as Fall 2021 would be a tight timeline to work through.
    - The software is an open-source project and is reliant on community developers to contribute to it.
        - The software development will need developers, if there are not enough contributors the project will not be able to be completed by the projected completion date.

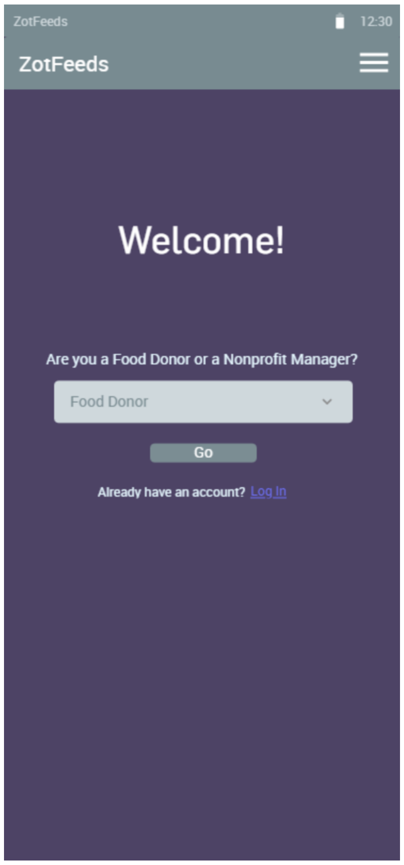
3. **Main design**
    - **Application Design:**
        - The application shall require a signup process for two different users with different functionalities for each.
            - The application will have two types of users, the food donor user and the non-profit organization manager user, each with different functionalities.
        - The application shall allow the food donor user to put in a request for the pickup of the excess food and input the information regarding the food.
            - When the food donor puts in the request they must input the information regarding the food which includes: the amount of food (estimate of the number of people the food is available for), the type of food, how long the food will be good for (an expiration date), and the location of the pickup of food. All information regarding the food will be inputted when the food donor requests the pickup for food.
        - The application shall allow the nonprofit manager user to request food and get notified when a food pickup request matches their food request.
            - When the nonprofit manager puts in the request for food, they must input information regarding the request such as how much food is need (estimate of how many people need the food) and they must choose the level of urgency for the request; low being that the nonprofit has other options and resources (other than the application) to request food from and high being that the application is the last resort.

- The application shall contain a priority system that will determine who gets food first based on need.
    - Many factors make your request more important than other organizations like how many other sources of food they have and how many people they are projected to feed. The level of urgency will help determine where the food goes.
- The application shall allow both non-profit manager types and food donor types to contact each other via a phone/contact number.
    - Both users will have provided their phone/contact number to communicate with one another through the food delivery process in case of any miscommunication or misinformation of the requests.
- The application shall have a matching feature between food donors and nonprofit organizations.
    - Food donors and nonprofit organizations shall be matched so that a communication and acceptance process can proceed forward between the two groups.
- The application shall have a verification process for nonprofit organizations to solidify legitimacy.
    - The nonprofit organizations need to be verified to ensure that food is being delivered to a proper location and that the system is not being taken advantage of.
- The request by non-profits and food donors shall have the option to be canceled whenever necessary.
    - Nonprofit organizations will have the option to cancel their request at any time. The application shall notify the donor if the request is canceled.
    - Food donors will have the option to cancel their donation before it gets matched by the algorithm.
- The application shall record/keep track of the current request of a Nonprofit organization, and ensure that it has no more than one request.
    - There should be only one request per nonprofit organization so that the software will not be overloaded
    - Anyone organization shall not be able to request all the available food and therefore take all the resources.
- The application shall allow the nonprofit organization manager to enter the estimated time of pickup for the food donor to expect.
    - After being matched, the nonprofit organization manager should notify the food donor of the estimated time of pickup that they will arrange for the food after using the location to estimate it. If any updates, the food donors’ contact information shall be available to the organization.



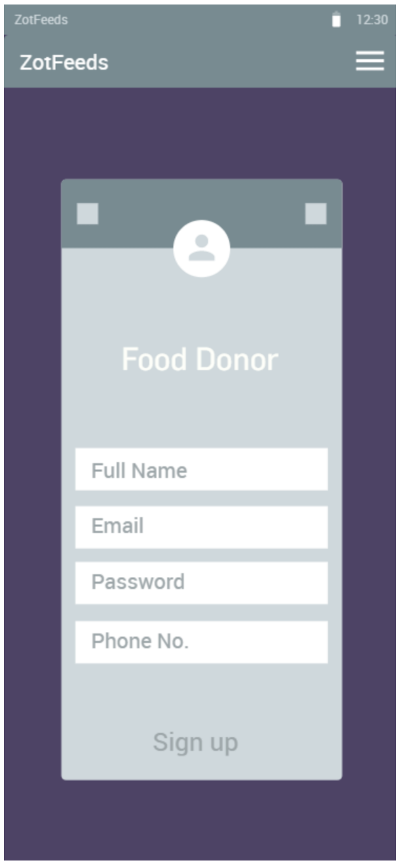
- **Interaction Design:**
  - The application shall have two user signup ("Food Donor" or a "Nonprofit Manager") options that they can choose from at the homepage.
    - Due to the different functionalities of the two users, the user log-in will determine the different homepages and operations that are available to each.
    - When the "Food Donor" is chosen, the application shall present the user text boxes to fill out their name, phone number, and email address.
    - When the "Nonprofit Manager" is chosen, the application shall ask the user for the same information for "Food Donor", and will also ask for the nonprofit organization the user is working for and it will go through the verification process that is required of the nonprofit.
    - If the user already has an account, they can click the login button instead of signing up again, where they will be able to choose whether they are a food donor or a nonprofit manager. They will be able to log in with their account information that they used when they first signed up with.
  - When a "Food Donor" user logs in, they will see a button to request for pickup of excess food.
    - On the homepage, there will be a button, "Pickup Request," that users can click to look for a request. The button will open a different page to fill up the request information.
    - The pickup request will show a questionnaire to be filled.
    - The request will be added to the home page with a status label (pending or accepted).
  - When a "Nonprofit Manager" user logs in, they will be able to request to pick up food.
    - On the homepage, there will be a button, "Food Request," that users can click to look for a request that best fits their needs. The button will open a different page to fill up the request information.
    - The food request will show a questionnaire to be filled.
    - The level of urgency they need the food for will be a dropdown menu with the options: (low, medium, high) (low if they have other resources to get the food from others, high if they do not).

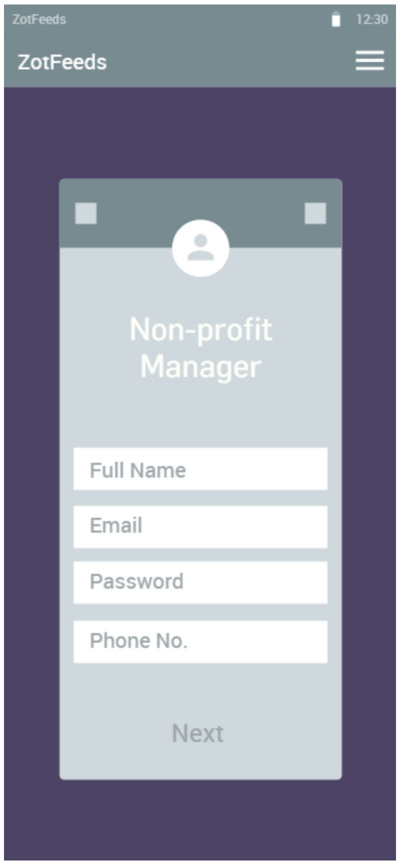
- Donors can fill out how fresh the food they are donating is through a calendar system.
    - The donors will be able to choose a date on the calendar that shows when their food will expire.
    - Based on what date the donors choose, if the food will expire too soon, then their request to get their food picked up cannot be processed.
- Once a match has been found between a food donor and a nonprofit organization, both parties will receive a notification on their device.
    - The notification text for food donors will say that they have found a nonprofit organization that will come pick up your food.
    - The notification will only appear for nonprofit managers once the food donor has filled out the location of where they want their food to be picked up. The notification text will contain the food donor's location.
    - Both users can click on the notification and they will be sent into the app to view the match.
    - The nonprofit manager will be able to view the location of the food donor so that they know where they are supposed to go to pick up the food.
    - The nonprofit manager will be prompted to send the food donor the estimated time of pickup for them to see.
    - The food donor will be able to view the estimated time it takes for the nonprofit to reach them.
- Users can use the "Cancel" button to cancel their requests.
    - This button will only allow canceling the donation or the request depending on the user. Food donors can cancel before their request gets matched (changes from pending to accepted). Nonprofit Managers will be able to click the Cancel Button always.
- Food donors and nonprofit managers can view their past transactions through the dropdown menu of the application homepage.
    - Food donors can click the dropdown menu and see all the past transactions they had picked up. They can see the date and time of when their food got picked up and what food they donated.
    - Nonprofit managers can click the dropdown menu and view all the past transactions they picked up. They can view when and where they picked up the food from and also what food the transaction contained.

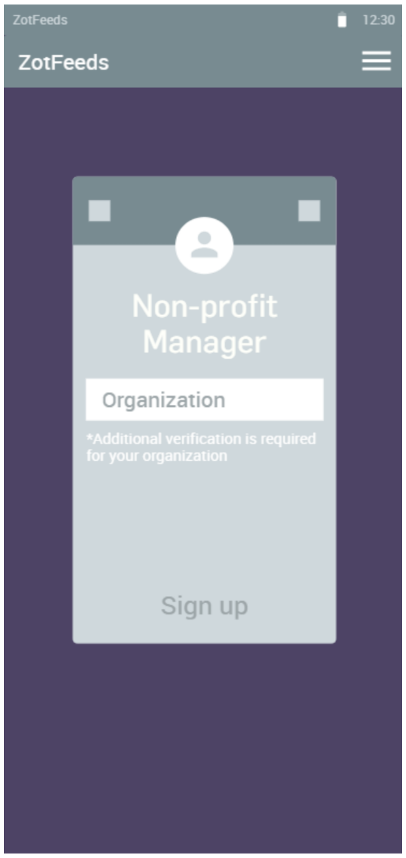


Are you a Food Donor or a Nonprofit Manager?

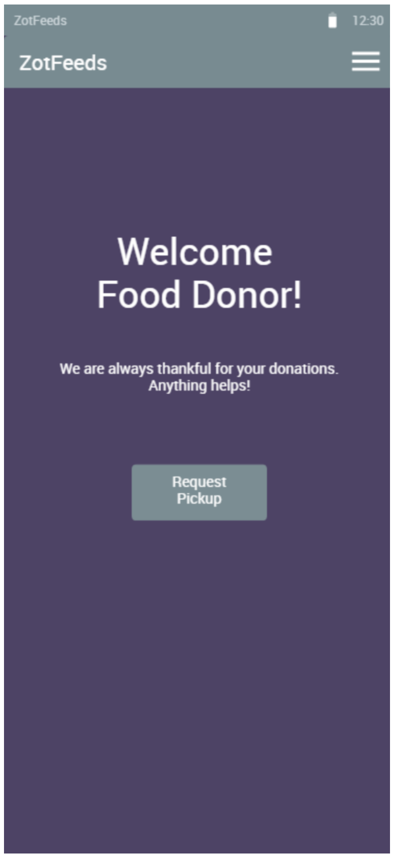
Already have an account? Log In



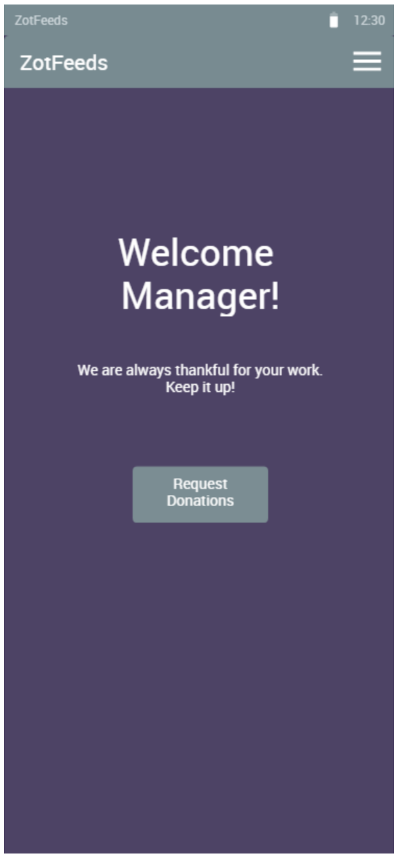




*Additional verification is required for your organization



We are always thankful for your donations. Anything helps!



We are always thankful for your work. Keep it up!

4. **Architecture Design:**
    - The application will follow a server-client architecture design, where the client will alternate between the food donor user and the nonprofit organization user.
        - The server will provide services, where the food donor requests food pickup, the server, in this case, is the nonprofit, will provide the food donor with the service of sending someone to pick up the food and vice versa in the case that the nonprofit puts in the request for food.
    - The application will incorporate an algorithm that determines who receives food first based on an importance level standardized by our system.
        - The application will review and read the request for food by nonprofits, taking information such as priority/urgency level and how many requests the non-profit has made. Based on these numbers the algorithm will sort the virtual queue by greatest need.
    - The application shall use a database that holds two types of accounts, food donors and nonprofit organizations.
        - The database is necessary to remember users who want to frequently donate food and to ensure that nonprofit accounts are verified and legitimate organizations. This database will also hold necessary information about each account which will help when submitting requests for food or food pickup.
    - The application shall use an array within the user class that holds all current/active requests for that user.
        - The requests for the user are stored in an array. The virtual queue will get a reference to each new request for easy access and update.
        - The array remembers all completed requests to maintain a history that will be used to display requests by a user and is account-linked so that an individual can view and monitor their previous actions.
    - The application shall make use of a virtual queue for nonprofit requests that are sorted by their urgency, number of individuals in need of food, and time submitted.
        - The virtual queue is an important element in distributing food equally and efficiently. By creating a virtual queue that holds requests in a specific sorted order from the database, there will always be a current request in need so long as the database for the current request is not empty.
    - The application will create a new class object for each request, both from food donors and nonprofit organizations, which will hold the specific parameters needed for it to be created.
        - Upon a user submitting a request for food or a request for food to be picked up, it will create a new request class object for that submission which will be stored and saved into the array of the user class holding all current and active requests. Non-profit requests will be sent to the Sorted virtual queue while the food donors' requests will be sent to the Food request queue.
    - The application shall only allow one request from each organization on the virtual queue.

    - Upon submission of a request, the software shall search through the database of current/active requests checking for any request class object that has the same class object name. The class object name is determined upon the creation of one's account, and that name will be used throughout the database whenever a new object from that user is created.
  - The application will use a matching algorithm to connect food donors to nonprofit organizations.
    - Using the database that holds both requests by food donors and nonprofits, it will search and find the best match based on the amount of food that can be donated, the amount of food requested, and use of the virtual queue of the next nonprofit available to receive food.
  - The application shall make use of a verification process to ensure that receivers of food are legitimate nonprofit organizations.
    - Upon creating an account for nonprofit organizations, the verification process will make use of the Charity Directory of Irvine website. This process will ensure that the information input matches with and goes beyond public viewable material to promise donors that their food is being sent to professional systems.
  - The application will use the React Native framework to implement the application to be available on both IOS and Android application stores.
    - ZotFeeds will solely be a mobile application that is available on both IOS and Android mobile devices. The user will be able to perform all functionalities and use of the software system on the ZotFeeds mobile application.

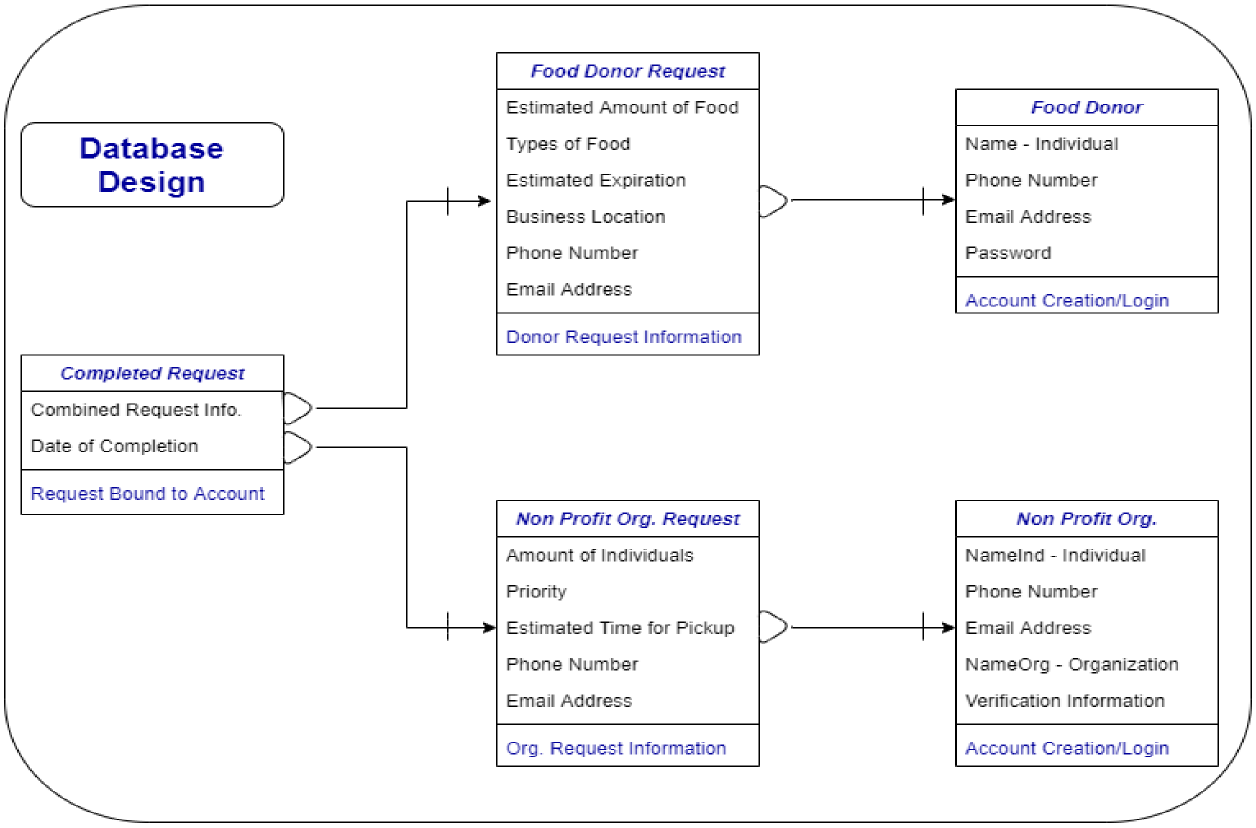
The section below describes the different artifacts used in the architecture design of ZotFeeds. The artifacts are the framework used, database architecture, client-server architecture, object-oriented architecture design, the system's data flow, and the two major concepts used to make the system work which are the virtual queue and the matching algorithm.

Framework:

We decided to use React Native as the framework for the following reasons:

- React Native uses javascript which is a language most of us are familiar with.
- React Native is open source which gives us more tools available and third-party features to build the application faster and efficiently.
- Similar apps using React Native: UberEats, Instagram, skype.
- React Native is cross-platform. This means we only need to have one base code for IOS and Android. It helps reduce development time and spending. It also increases the audience reached to multiple platforms.
- React Native makes it easy to maintain apps. It has a sync update feature that should update the code for patches and updates within minutes to the source code.
- The available screen containers in React Native are ones that are adequate for ZotFeeds as it contains features such as the sign-in containers, verification page and order container.

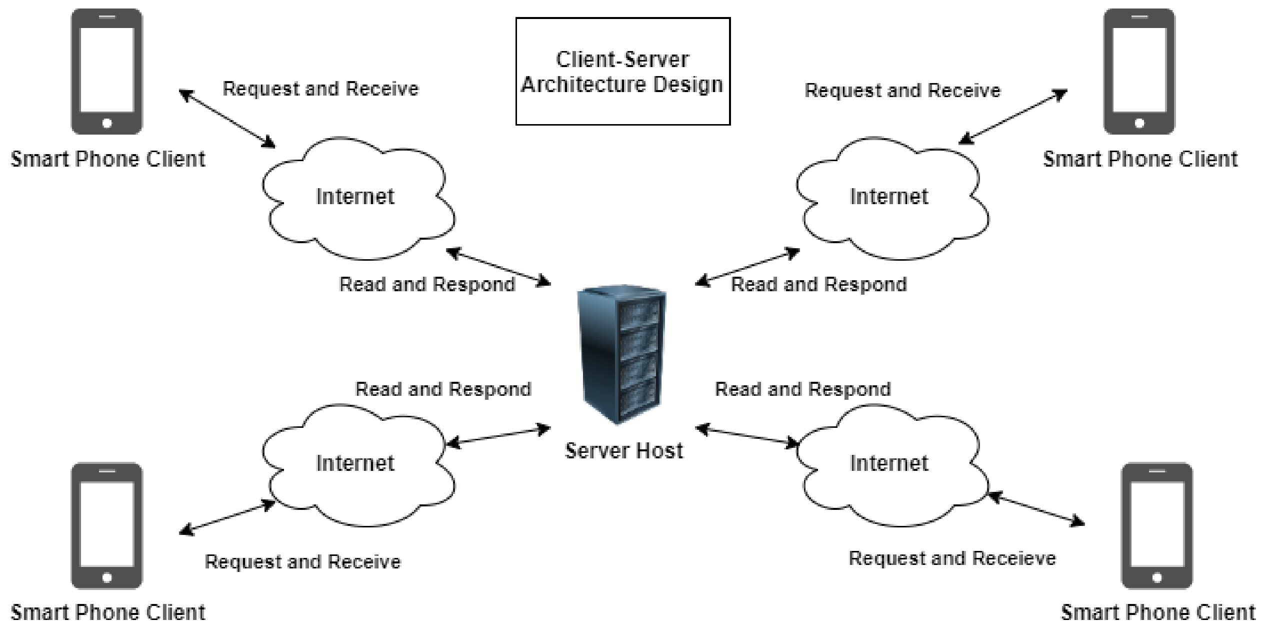
<u>Database Architecture:</u>



<u>Database description:</u>

______This database follows a key-value format where the key will be the name and the value will be a class object either a Nonprofit Organization or Food Donor. This will be used for the authentication of a user when they try to log in. When new users sign in, the database will be updated to have the new user inserted in the database. The user class will have access to an array of all requests the user has made, this will not be part of the database but rather referenced by the user class.

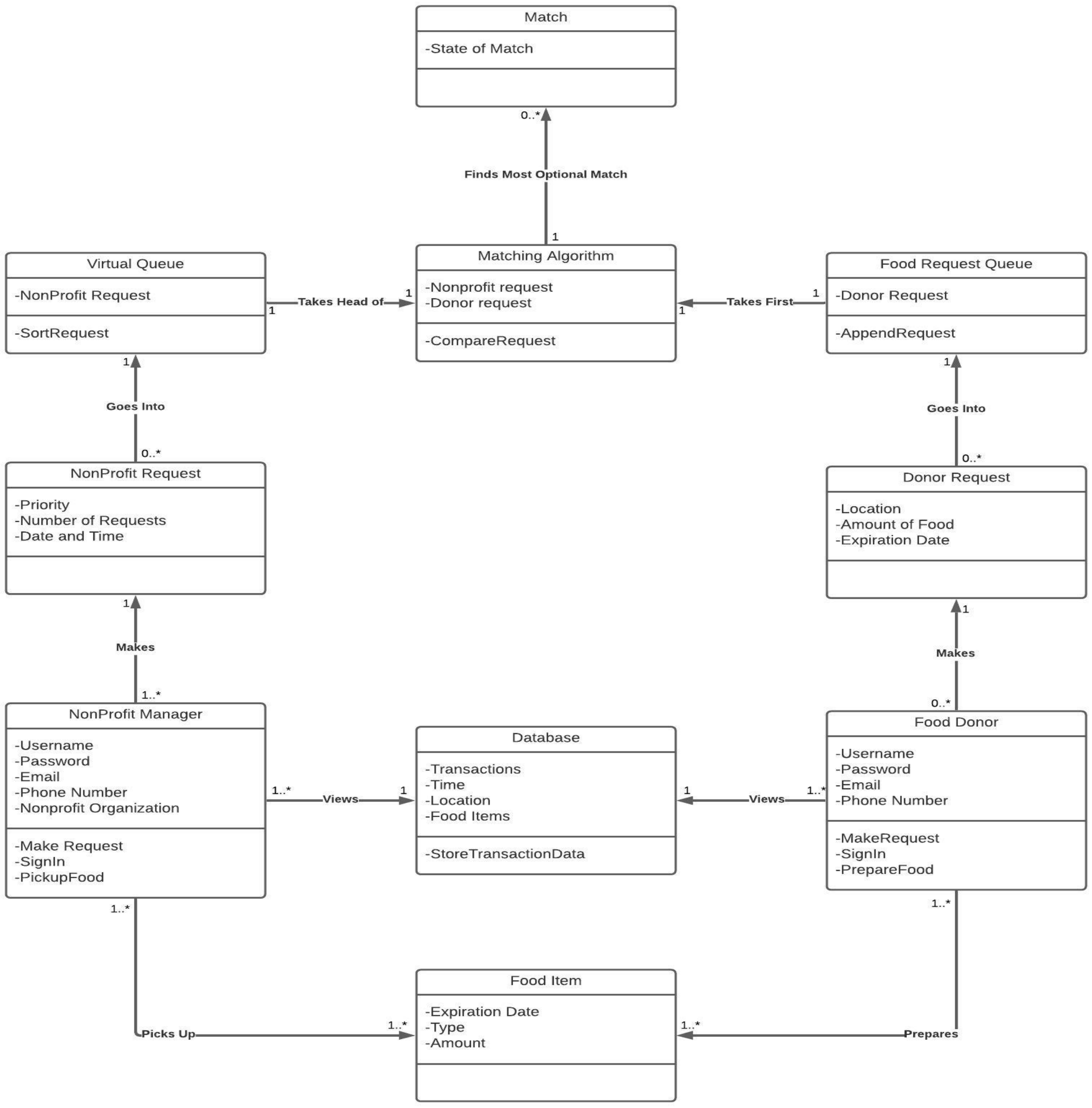
Client-Server Architecture:



Client-Server description:

ZotFeeds will follow a client-server architecture that will provide services and functionalities for its users based on the account type when logged into the application, food donors, or nonprofit organizations. The client-server follows a TCP connection meaning that the data that is requested and received is passed through five layers: application layer, transport layer, network layer, link layer, and physical layer. These five layers work together in breaking data down into segments, transporting and routing them to their source destination, and putting the segments back together before arrival to the source destination. To generalize this, users will communicate with the ZotFeeds server host to be able to receive the proper data requested. From this point, users will be able to view and interact with what is displayed on their device

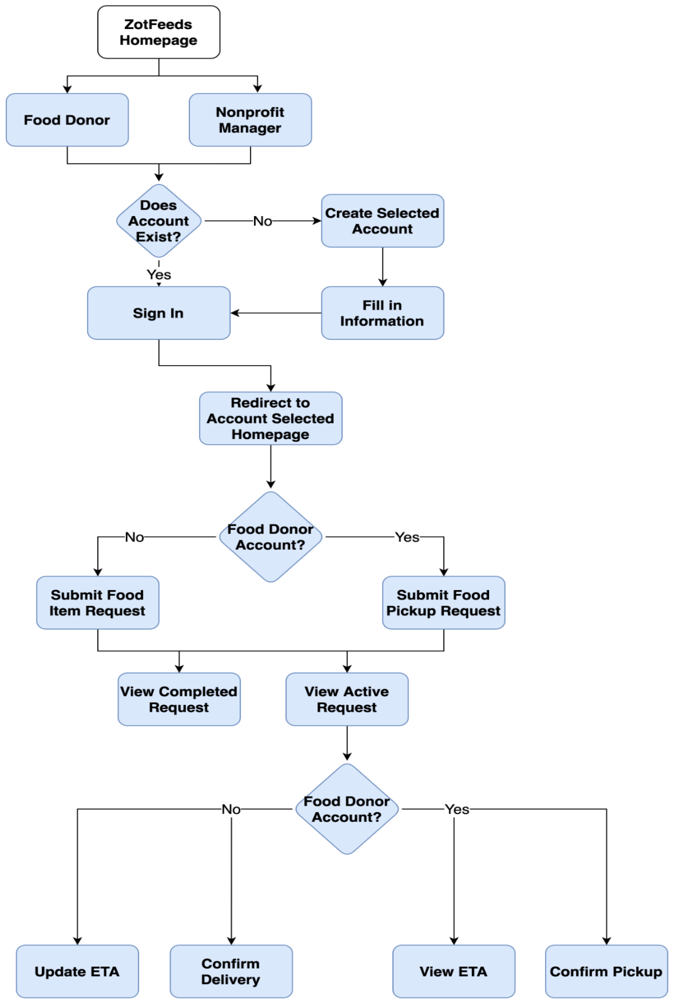
Object-Oriented Architecture Design:



Object-Oriented Architecture Design Description:

The UML Class Diagram has Non-Profit Manager and Food Donor as the two main actors of the system. They can view a database that stores the history of past transactions that they have made, make their appropriate requests, and pick up/prepare food. Once either actor has made a request, their request will go into their respective queue system. The virtual queue for the NonProfit requests will automatically sort the requests by priority, number of requests, and date and time respectively. The Food Donor queue system will just append the requests to the end of the queue. Then, the matching algorithm will take the first request from each queue and try to find the best match possible.

System Data Flow:



Data flow description:

The data flow structure describes the process and the flow of data that both a food donor or nonprofit organization will follow when opening and using the ZotFeeds application. The user will begin by selecting the type of account that they already have or that they want to be created. From there users can either select to sign in if an account exists or create an account filled with the necessary information to the selected account type. Upon finishing the sign-up process or sign-in process the user will be directed to their selected account type's home directory page. Food donors shall be able to submit requests for food pick and nonprofits shall be able to submit requests for food. Both types of accounts are able to view their completed request through a history query, however, viewing their active request differs slightly. Food donors will be able to view the estimated time of arrival and confirm the pickup when completed, while nonprofits can update their estimated time of arrival and confirm when the delivery for food has been picked up.

Sorted Virtual Queue:

As stated above the application shall make use of a virtual queue for nonprofit requests that are sorted by their urgency, number of individuals in need of food, and time submitted. This linked list will store the nonprofit requests and sort them based on the following factors: Highest Priority (high, medium, low), Least number of requests, and Earliest Timestamp. The highest priority goes first on the queue (closest to the head), if the same priority, then the least number of requests goes first, if they share the same number, then the oldest timestamp goes first. A function will be called when the nonprofit user makes a request. This function will send a linked list node with a reference to the request to be added into the sorted Virtual Queue.

Virtual Queue Pseudocode:

```
LL = Linked list storing the sorted nonprofit food request item
Item = nonprofit food request item incoming
Current  =  current item in the LL
While Item has not been inserted:
     #check priority
     If item's priority level is greater than
     the current's priority level do:
          Insert item node in front current node.
     Elif item's priority level is equal to the current's
     priority level do:
          #check number of request
          If the item number request is less than the current's
          number request do:
               Insert item node in front of current node.
          Elif the item number request is equal to the current
          number request do:
               #check timestamp
```

```
            If the item’s timestamp is earlier than the
            current’s timestamp do:
                Insert item node in front of current node.
            Else:
                Iterate to the next node, set current to the
                next node.
        Else: (greater than)
            Iterate to the next node, set current to the next
            node.
    Else (less than):
        Iterate to the next node, set current to the next
        node.
```

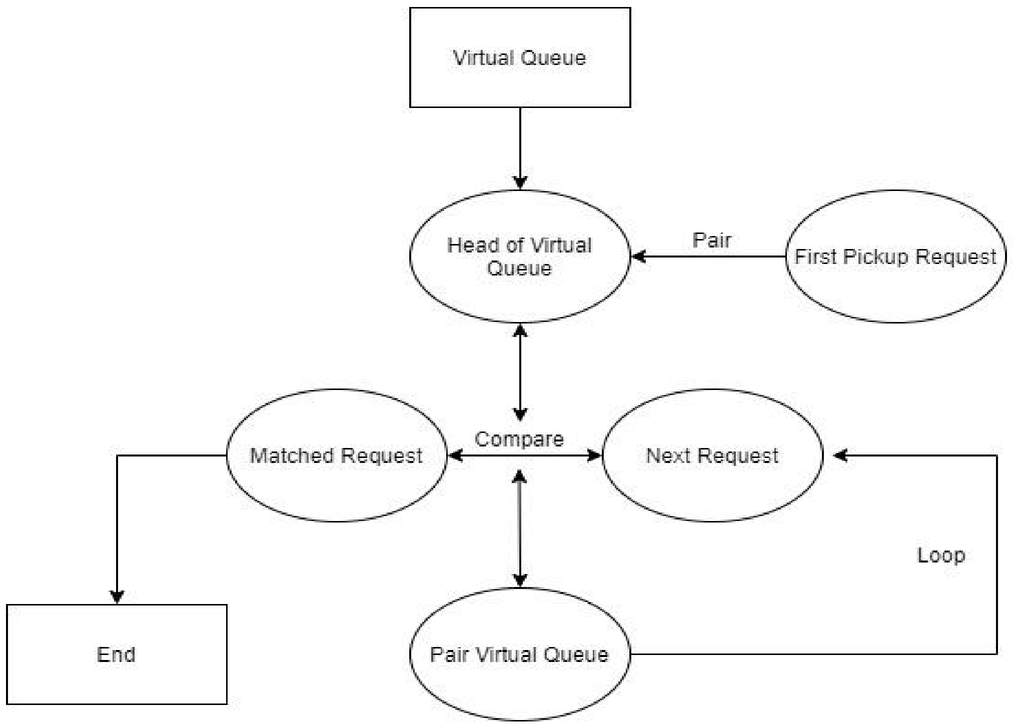
Virtual Queue Pseudocode Description:

When a request is made by the nonprofit user, the virtual queue function above is called and the following steps take place. When a new item is added to the list, the algorithm will iterate and compare each item to its priority starting at the head to insert the node in the right place. “Current” is the node that is currently being compared to the incoming node in the loop. Insert the incoming node before the current node if it has a higher priority. If priority is less, keep traversing. If it has the same priority, compare both item’s request numbers. If the incoming item has a lesser number than the current, insert it before the current. If it is greater, keep traversing. If they have the same request number, compare both items' timestamps. If the incoming item has an earlier timestamp than the current, insert the item before the current, else keep traversing. When the item is added to the linked list, the function ends and waits to be called again when another item is added.

Matching Algorithm:

The matching algorithm in ZotFeeds is responsible for matching the list of nonprofit's requests to the limited number of available food donors' requests. This matching algorithm is made easier due to most of the work being done by the virtual queue, where the requests are already sorted by the desired priority. The matching algorithm will simply take the head from the virtual queue and compare the nonprofit's request to a list of food donors' requests to find the best match. The food request queue is a linked list that will store references to Food requests made by food donors on a first come first serve basis. Each new request made by food donors will get added to the end of the link list, with the consideration of the food item’s expiration date. Food requests that are expired will be popped off the list by the matching algorithm. The diagram below shows the linear search process of the matching algorithm and the virtual queue.

Linear Search Diagram:



Matching Algorithm Pseudocode:

```
Take the head of Virtual Queue
If first request estimated expiration >= current date
    Pop first request from Virtual Queue and set firstRequest.matched = true
Else, take first request and pair with head
Loop:
    Make X = next request
    If X estimated expiration >= current date
        Pop X from Virtual Queue and set X.matched = true
    Else, compare paired request priority with X’s
        If X priority > paired priority, pair X with head
        If same priority, compare number of requests
            If X requests < paired requests, pair X with head
            If same number of requests, compare time/date
```

```
                    If X time/date < paired time stamp, pair X
          with head
End Loop
```

Matching Algorithm Pseudocode Description:

The matching algorithm takes the head (the first node) of the virtual queue. Then it takes the first pickup request and checks if the estimated expiration date is greater than or equal to the current date. If the check is true, the first pickup request is removed from the virtual queue and set its matched boolean to true. If the check is false, the algorithm pairs the virtual queue head and the first pickup request. Then it will start a loop until it reaches the end of the pickup requests list. In this loop, it will take the next request and for now, call it variable X. X's expiration date will be checked if it's greater than or equal to the current date. If true, X will be removed from the virtual queue and set its matched boolean as true. X will then be compared to the current paired request through priority. If X has a greater priority, then it will be matched with the head. If X has the same priority as the paired, it will compare the number of requests. If X has fewer requests, it will be paired with the head. If X has the same amount of requests as the paired, it will compare their time stamps submitted. If X was submitted earlier, it will be paired with the head.

Updates to the System's Design:

During the Design Studio 2-2 assignment, our group took notice of three main updates to the system: Framework for the software, LinkedList data structures, and design for the matching algorithm.

Framework:

As our team made decisions we found one area that was left undiscussed in Design Studio 2-1 was the framework that was going to be used to create the software. Our research led us to two popular frameworks such as Flutter and React Native which both had their pros and cons in app development for food-related applications. Collectively as a group we decided that React Native was the better fit and framework to follow for our application because of its benefits to specifically creating software for IOS and Android devices. We wanted to limit the application to be strictly for mobile devices and React Native was the most optimal in accomplishing that goal efficiently and cost-effectively. React Native also is an open-source mobile application framework and our group aimed for ZotFeeds to be an open-source project where contributors could help whenever they pleased. This grabbed our attention as a group because React Native was geared towards our initial motives of how and who we wanted to create the ZotFeeds software. The use of the React Native framework would cause the system to restrained to using the javascript programming language which is now updated in the system design.

The Use of Linked List Data Structure:

We did not originally account for the Food Request from food donors to be in a separate array. We wanted to have both types in a virtual queue that would match them from there, however, we encountered problems with the design when trying to have one virtual queue to hold both types of requests. We decided to implement the virtual queue separately, and have this unsorted linked list to hold the food request. The reason why this is unsorted is that we can assume there will always be a need for food, but we cannot assume that there will always be food donors.

Matching Algorithm:

In Design Studio 2-1, we decided that we will try to relieve food insecurity through an application that will take food donations and pickup requests. We knew that these requests were the most essential part of the software, but we did not know how we would process, fulfill and match the donation and pickup requests. In Design Studio 2-2, we decided to create criteria for donation requests to properly determine a fitting match. We added "Priority", "Least Number of Requests" and "Earliest Timestamp" sub-values to each request to give the requests their prioritization. Next, we realized that we can use a search algorithm in part with the priority criteria to create a matching algorithm for ZotFeeds. We decided to use a linear search algorithm to pair up donation requests with pickup requests.

Databases:

We originally anticipated a database for all completed and active requests. However, we thought that just storing them in a list in the user class will be more efficient. This would help the accessing of those requests and would not need to be updated as often as the original idea would. It also saves space since we do not need two separate databases to hold these objects.

5. **Alternatives considered**
   - **Alternative #1:** This alternative would help reduce the user's spending by providing a resource for coupons and offers.
     - **Description of Alternative #1**
       - This alternative software would aim to help reduce the amount of money the user spends on food and groceries to help food insecurity and availability for people in need.
       - An application that collects and stores coupons for food in a database that the user can use to look up their groceries.
       - This application will allow the users to find coupons for food and other groceries. The coupons will allow users to buy groceries and food for cheap.
       - The application will actively look for coupons online and collect them to create a database of nearby stores' coupons.
       - The application will have a feature to add your own collected coupons to the application database.

        a. Additionally, users will have an option to add sale or clearance events for nearby users to get notified.
        b. The events must have proof, such as a picture, a description of the event, and location.
    - An algorithm will act as a moderator to remove fake or expired coupons and events. Users will be able to downvote events or coupons so they get removed by this algorithm.
    - The application would potentially only use the location of the user while using the application to customize what coupons are shown.
    - The app would open directly to a directory of coupons for stores nearby.
    - There will be filtering and search features to allow users to make customized searches.
  - **Comparison to ZotFeeds:**
    - Trying to address the same societal problem of food insecurity, this alternative's approach helps and eases the issue to a certain extent, but it does not do so aggressively. This approach would not be fit for everyone, it wouldn't be broad enough to consider people who need food but do not have the money to use the coupon or the offers.
- **Alternative #2:** This alternative would follow the same approach as ZotFeeds but would have a third-party service to handle the delivery of the food from the food donors to the nonprofits.
  - **Description of Alternative #2**:
    - This feature would add a Driver user (third party) to the login and signup tabs.
    - This Driver user will get assigned to a request for pickup and delivery from a nonprofit that lacks pickup drivers.
    - This feature would allow the driver to get the information about the pickup request, such as pickup and delivery directions, food type and amount, etc.
    - These Driver users will get paid through the application based on the trip length and time.
    - The food donors and nonprofit organizations will have live tracking on the driver to see the progress of the current transaction taking place with estimated times of pickup and delivery.
    - The application will allow drivers to use a map feature to get directions to pickup and delivery locations.
    - The drivers can get rated on a 1-5 scale of how reliable and fast they get the food delivered and picked up.
  - **Comparison to ZotFeeds:**
    - This approach takes upon the same roles as ZotFeeds, but with a change on how the delivery and pickup of the food would be handled. In our original system, ZotFeeds was aimed to have the nonprofit organizations handle the pickup and delivery of an accepted food request. In this alternative idea, however, our team

brainstormed of having a third-party service that could handle the times when nonprofits were unable to pick up the order. This plan highlighted this issue and also illuminated more ideas that could be implemented into the system. For example, a more accurate tracking system would be implemented with up-to-date information on the arrival of one's order. However, we ran into issues with the cost of these services. As a team, we decided that our system ZotFeeds should be a system to connect these two groups, rather than being a delivery service between the two. By taking this approach, we could better handle the problem of food security and reduce any additional costs for nonprofit organizations that are already in need of support.

- **Comparison of all the alternative approaches to ZotFeeds**
  - The first alternative, the Coupon application had many constraints and failed to be feasible in many ways, some of these are that it relies on users to operate and actively engage with the application. Another constraint is that the application will only be able to show general coupons, these are coupons that are not attached to one sale and can be used by multiple users. Additionally, this application was not feasible to make, specifically, the algorithm that finds coupons on the internet would need a high level of coding to find a coupon. Finally, the application does not tackle the main population affected by food insecurity. The application's audience would be people who have cell phones and exclude the most vulnerable to food insecurity (the homeless and the poor). This approach compared to the ZotFeeds approach, barely scratches the surface in trying to address the food security problem, as it is not inclusive of its users that suffer from the problem (homeless people and people who aren't able to afford food). The approach we took, in contrast to this application, is directly helping the most affected by food insecurity which also works hand in hand with reducing food waste.
  - The second alternative approach of adding a third-service party service to ZotFeeds would be not feasible and cumbersome to implement as it would add the payment options of the drivers and adding them as different types of users would add unnecessary attention to the functionality of the driver, which would, in turn, derail from the main concern of the application with is to match food donors to nonprofits and help reduce food insecurity. Looking at these alternatives above, we tried to stick to our goal of keeping this election simple and useful, focusing solely on the problem at hand. With addressing and trying to tackle such a significant problem, we must always put our audience (people in need of food and people with food insecurity) first and think about what would be feasible, not only for us as designers but also for the users and how this software would be available for them and how it would help them if it isn't.

6. Ethical Analysis / Values Statement

- Security
  - ZotFeeds aims to help the problem of food security within the Irvine, California community. By connecting food donors and nonprofit organizations, it is one small step towards bettering the lives of others and reducing the effect food waste has on the environment.
- Privacy
  - ZotFeeds needs to be secure and protect confidential information that is submitted by a user whether that be during the account creating process or a request for food and food pickup.
- Honest
  - ZotFeeds wants to promote honesty amongst its users including both food donors and nonprofit organizations. By promoting honesty it will ensure that food is being sent to legitimate organizations and that no one organization is receiving too much based on their submission requests.
- Helpful
  - ZotFeeds aims to help individuals with excess food connect with nonprofit organizations to ensure that food is not going to waste. By connecting the two parties, there can always be a transaction of these items having a positive outcome on the lives of those who need food and on the environment.
- Responsible
  - ZotFeeds takes it upon itself to connect these two parties. It attacks the problem of the lack of communication between these groups by connecting and allowing these parties to interact with one another through the software.
- Healthy
  - ZotFeeds aims to help the individuals and nonprofit organizations within the Irvine, California community by taking a step towards resolving the food security issues. By connecting food donors and nonprofits, it will be able to allow for easier distribution of food to those in need and helping their dietary needs.
- Successful
  - ZotFeed’s main goal is to help reduce the problem of food security. It aims to help the individuals and organizations in need so that they can continue everyday life equally amongst society. Reducing one more issue can help contribute to that.
- Influential
  - ZotFeeds aims to influence and encourage more individuals and parties to become food donors to reduce the problem of both food waste and food security.
- Equality

        - ZotFeeds aims to distribute food equally amongst its parties that request food through its software. It wants to ensure that individuals' worries about where their next meal will be are reduced so that they can focus on the more important things in their lives.
    - Capable
        - ZotFeeds should be accessible by any party that is willing to offer food/resources or any nonprofit organization within the Irvine, California community. This would allow for maximum potential to attack the problem of food waste and food security through its connectivity features.

    - **Could this software marginalize, create a barrier for, or embody bias against any particular segment of the population? How can we mitigate this?**

        For users unfamiliar with the English language, it prevents them from being able to read and use the current state of the software. This issue can be mitigated by implementing different language translations. Another issue that could create a barrier for users is if they are unfamiliar with software that is similar to ZotFeeds. This issue can be resolved by providing a tutorial option that can guide and give knowledge on how to use the ZotFeeds software.

    - **What are some potential ways the existence of this software may cause harm to people or the environment? How can we mitigate this?**

        The current state of the desired system has no verification process of food donor accounts. This puts the potential risk that when food is being picked up by a nonprofit organization, it may put them in a dangerous position. This issue can be resolved by having a background check taken on food donors so that the software can ensure the safety of both parties.

    - **Could this software be used in a nefarious way to harm people? How can we mitigate this?**

        The software can be used in a nefarious way by submitting false requests for food pickup. This ultimately, if a nonprofit organization is matched to the false donor, can lead to a waste of time, effort, and money for the organization. This issue can be mitigated by again including a background check on individuals who want to become food donors and also requiring the food donor to upload an image of the food with the current date marked in the photo.