NYU TANDON SCHOOL OF ENGINEERING

Computer Science and Engineering

Hunger Warrior

Software Design Description (SDD)

VERSION 1.1

Document Number: SDD-002

Project Team Number:	B14	
Project Team Members:	Gavin Senger (gjs374)	Sanidhya Sitaula (ss12081)
	Kori Vernon (ksv244)	Courtney Battieste (cbb389)

REVIEW AND APPROVALS

Team Members	Function	Date	Signature
Gavin Senger	Author	12/12/2021	Gavin Senger
Sanidhya Sitaula	Author	12/12/2021	Sanidhya Sitaula
Kori Vernon	Author	12/12/2021	Kori Vernon
Courtney Battieste	Author	12/12/2021	Courtney Battieste
Professor Strauss	Review/Approval		

REVISION LEVEL

Date	Revision Number	Purpose
10/15/2021	Version 1.0	Initial Release
12/12/2021	Version 1.1	Final Release

TABLE OF CONTENTS

1. INTRODUCTION

1.1 Purpose

The purpose of this System Design Description is to clearly define the system under development, namely Hunger Warrior. It will give detailed descriptions of the system's architecture, design, and implementation. The intended audience of this document includes the professor who will be reviewing our project, the team members of the project, fellow students in the class, and the end users of Hunger Warrior. Other intended audiences include the development team and members of the developing organization.

1.2 Scope

Hunger Warrior is a project that aims to assist with the fight against food insecurity and food waste in the United States of America by reallocating food to those that are in need. This is done by helping shelters get connected with grocery stores, allowing the grocery stores to donate food to the shelters that would've otherwise been thrown out. Grocery stores will be incentivized to limit excess food waste with increased tax incentives, and homeless shelters will gain access to more affordable food and access to more efficient distribution. Our goal is to make Hunger Warrior easy to implement into local communities, gradually allowing for expansion in the future. There are other products in the market that are similar in goals such as minimizing food waste and making food more attainable, but none that are similar in methods.
Hunger Warrior aims to connect grocery stores with homeless shelters in order to efficiently organize a reallocation and distribution of food. Hunger Warrior is intended to provide procedures for shelters that will enable requests and orders of food to grocery stores, while also allowing grocery stores to manage their inventory and provide delivery information. Hunger Warrior implements map functionality so that the grocery stores can

see the food requests, as well as documentation of orders and financial information within the website. The system will include the following functionality:

- Dashboard interface for both Grocery stores and Shelters
- Real time food requests and orders from Shelters in the local area that can be fulfilled by Grocery Stores
- Visualization of tracking and entity locations using a map
- Financials Documentation / Inventory Documentation

The goals of the system are:

- Connect Shelters and Grocery stores
- Reduce food waste when possible
- Provide benefits to participating Grocery stores and Shelters
- Be user friendly for employees and clients

However, the system will not be responsible for the specifics of the deliveries as it will all be coordinated by the grocery stores, nor will it be responsible for federally qualified food inspection.

1.3 Identification

Name: Hunger Warrior - System Design Description

Number: SDD-002

VERSION 1.1

1.4 Document Summary

The SDD will go over the reference documents that were used as reference for this document. It will then go over all of the requirements (business, specific, non-functional, functional), and then describe the component architectures, provide class diagrams, use cases, explain state logic, and depict the overall behavior under the analysis section. The document will also detail the test plans, qualification provisions, appendices, and

dictionaries. Overall, it describes the initial framework for the development of the project and provides a clear picture of the goals and obligations, which will also be updated throughout the course of the development period.

1.5 System Overview

Hunger Warrior is an application that strives to provide a reliable mode of communication between grocery stores and homeless shelters in order to ensure a consistent and seamless transfer of resources between both parties. It does this by utilizing a matching algorithm that will be able to take into account location, distance, resources requested, resource availability, etc.

2. REFERENCE DOCUMENTS

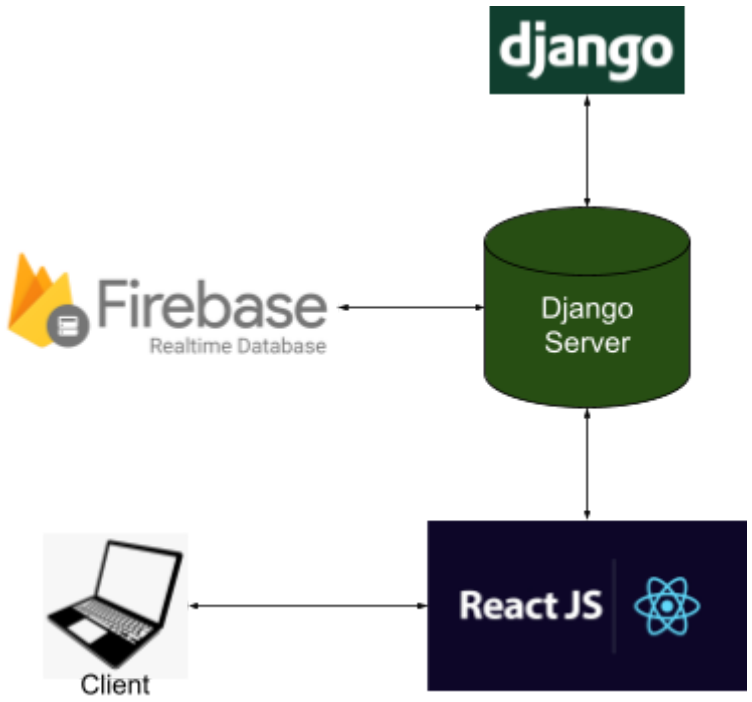
Battieste, Courtney., Senger, Gavin., Sitaula, Sanidhya., Vernon, Kori. “Project Proposal.” Version 2.0, September 2021.

Battieste, Courtney., Senger, Gavin., Sitaula, Sanidhya., Vernon, Kori. “System Requirements Specification (SRS).” Version 4.0, September 2021.

Battieste, Courtney., Senger, Gavin., Sitaula, Sanidhya., Vernon, Kori. “CS 4523 - Project Management Plan (SPMP).” Version 2.0, October 2021.

3. SYSTEM WIDE DESIGN DECISIONS

3.1 Software Component Architectural Design



3.2 Software Architecture General Description

In order to create the web application for Hunger Warrior to function, we will be using a Firebase realtime database with Django as the backend, and the front-end will be in React JS. The back-end of our software will be communicating with Firebase Realtime Database, which will act as the database for our project and store all relevant information to the project, such as locations, food items, and order history. Django will create a Django server that allows React JS to display information as well as take user input. This will then in turn, use user input in order to extract data from the firebase and return the necessary information.

3.3 Software Item Components

Firebase (DBMS) - A realtime database that can store data for the web application.
Django - A web framework that allows for the development of websites, it will also deploy a server (Django Server) that allows communication to the frontend (React JS)
React JS - A renderer and user interface that is able to display components supplied from Django and receive user input.
Client/User - The users that are interacting with the web application (usually denoted as Grocery Store or Shelter.)
Refer to Dictionaries for further detailed information.

3.4 Component Interface Identification

ClientToReact - the client interacting with ReactJS
ReactToDjango - ReactJS interacting with the Django Server
DjangoToFire - Django Server interacting with Firebase
DjangoToDjangoServer - Django Server interacting with Django
MapAPI - the map that will constantly be updated with locations
Shelter - a type of client that represents a homeless shelter
Grocery Store - a type of client that represents a grocery store

3.5 Software Component Concept of Execution

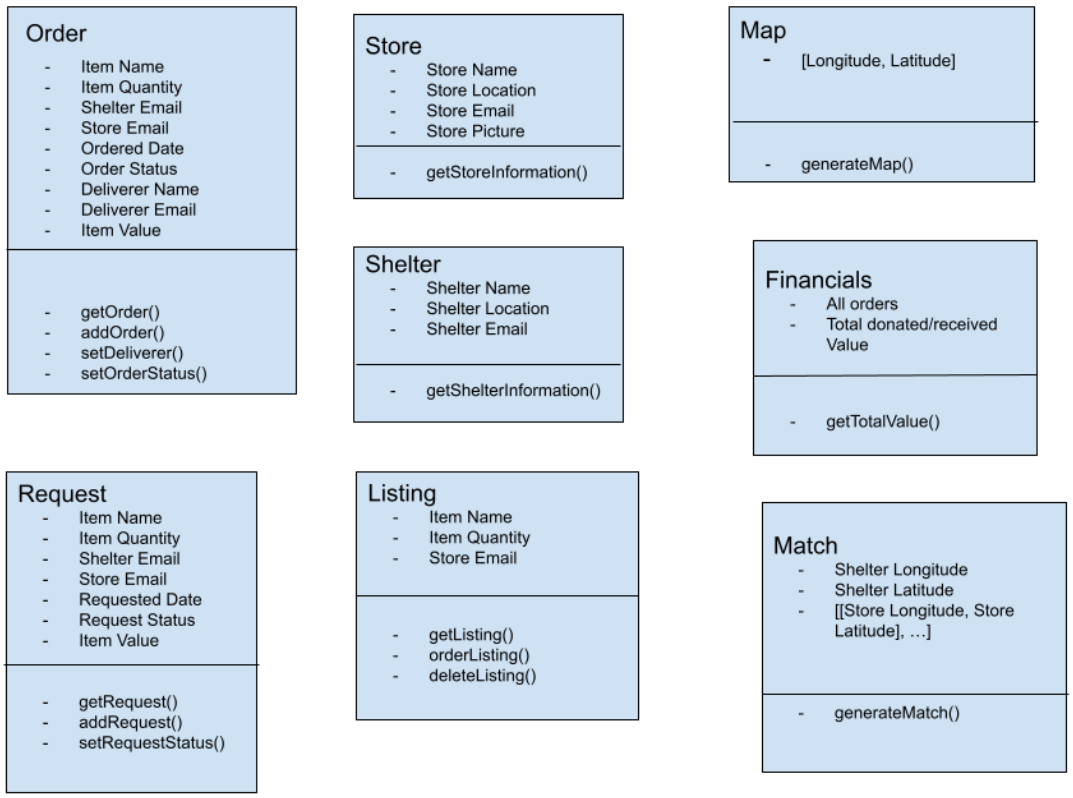
A client will interact with the website. The frontend (React JS) will display the necessary information to the client and will take in user input and will send/get this information from the Django server. If the client enters in information, React JS will send this information to the Django server which will then send it to Firebase. If a client requests to view information, that information will be requested from the Firebase through Django, the Firebase will then return the information to Django, which will then display it through React JS.

4. SOFTWARE ITEM DETAILED DESCRIPTION

4.1 Structure

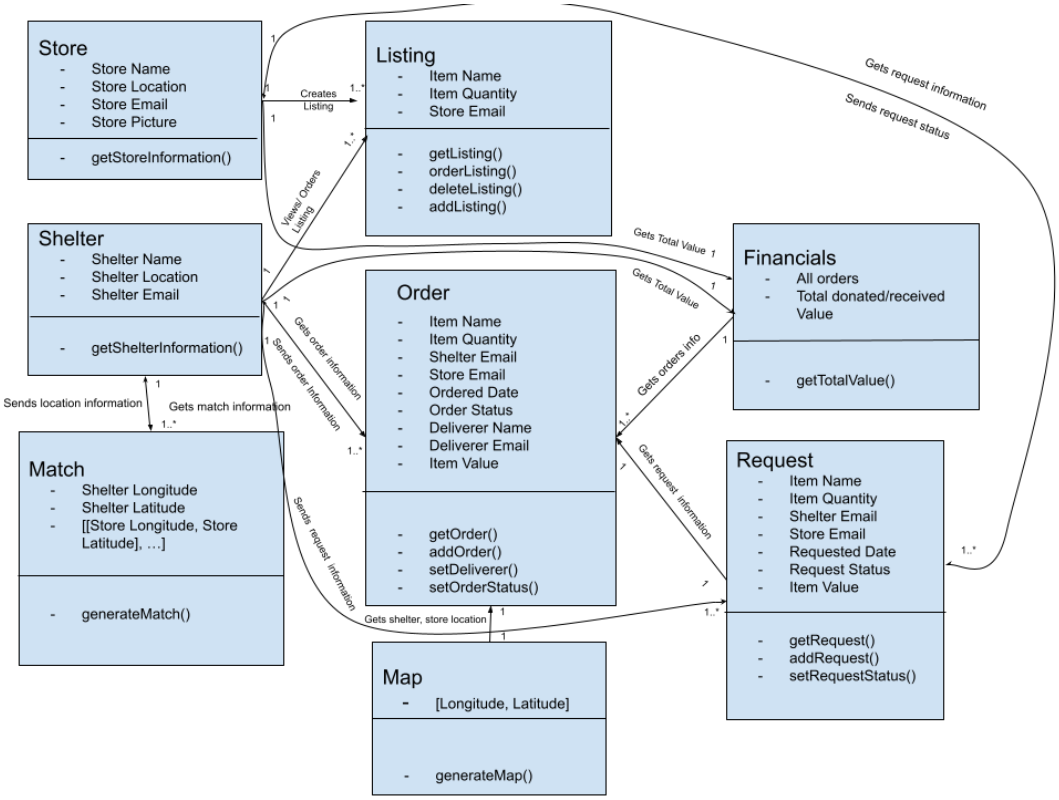
4.1.1 Software Unit Detailed Description

Class Diagram



4.2 Static Relationship of Software Unit

Class Interaction Diagram



4.2.1 Run-time Object Instances

Firebase Realtime Database manages multiple users accessing its database, which will employ the usage of threads.

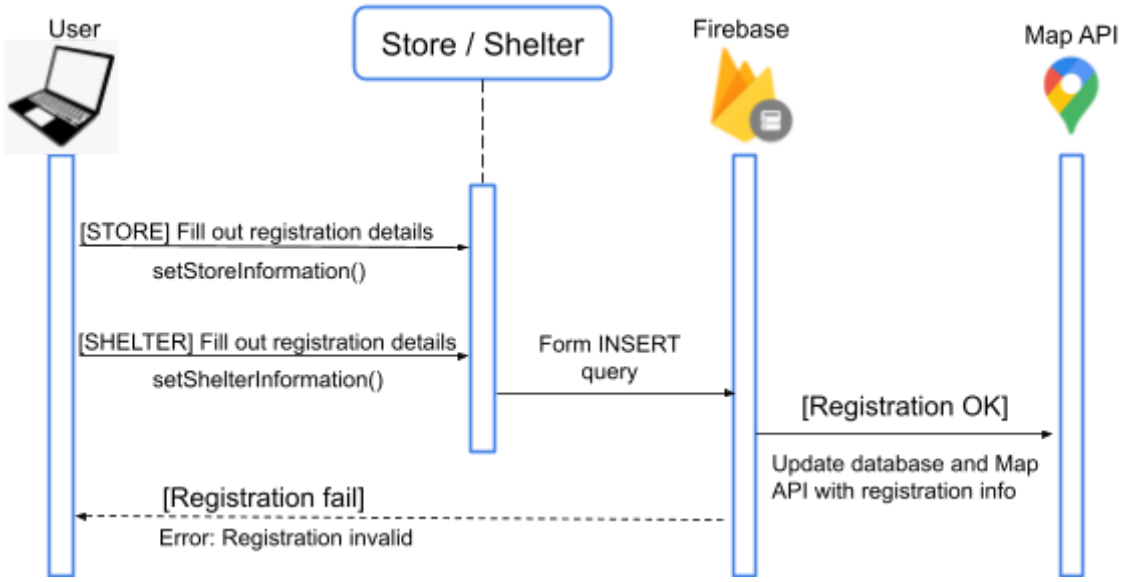
4.3 Behavior

Changes that occur frequently within the system are the number of grocery stores / shelters within the database, the requests that come from shelters, the current inventory and availability of resources within grocery stores, and delivery information. The app will have to frequently update in order to accommodate the changes that are happening. When a new shelter or grocery store is added to the system, a match algorithm will be

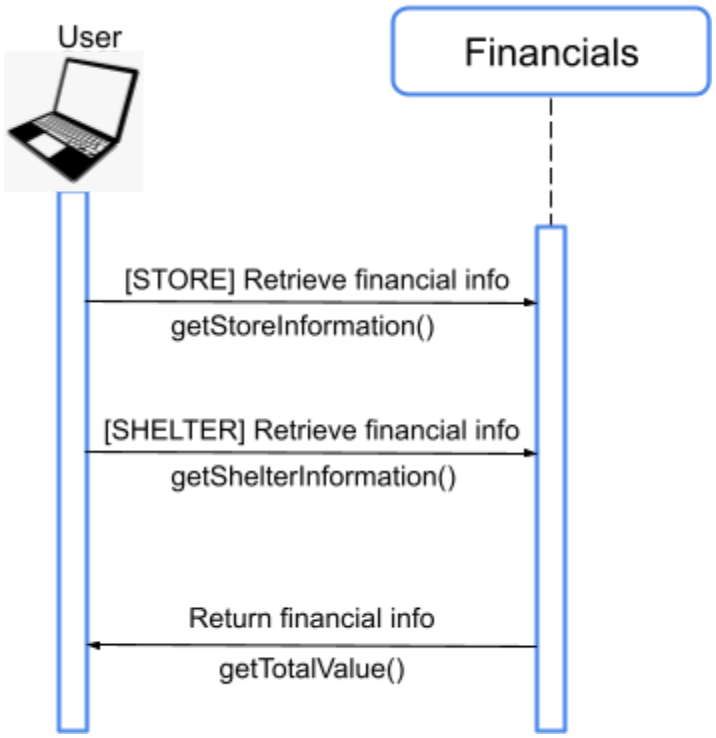
run in order to see if a more efficient relationship exists between the newly added entities and already existing entities within the system. Similar procedures will take place when a new request/order comes from a shelter or a grocery store's inventory is updated within the system. Delivery information will have to be constantly updated in order to make sure all parties are informed of delivery details.

4.3.1 Sequence Interaction Diagrams

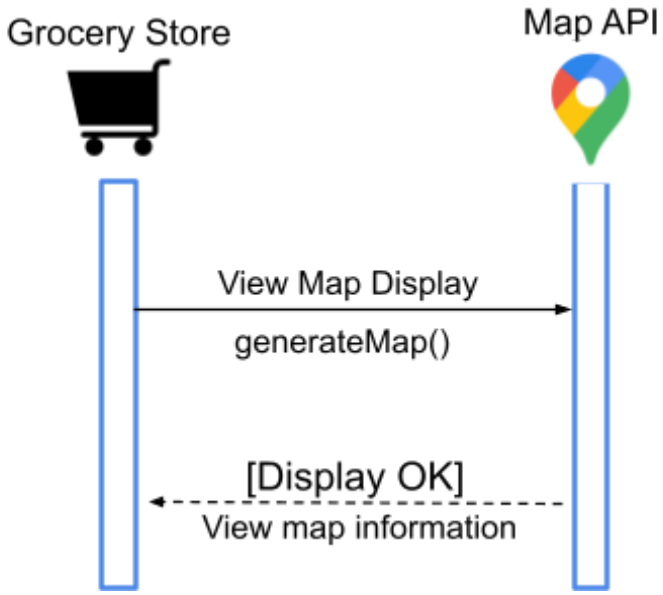
Register Location Information



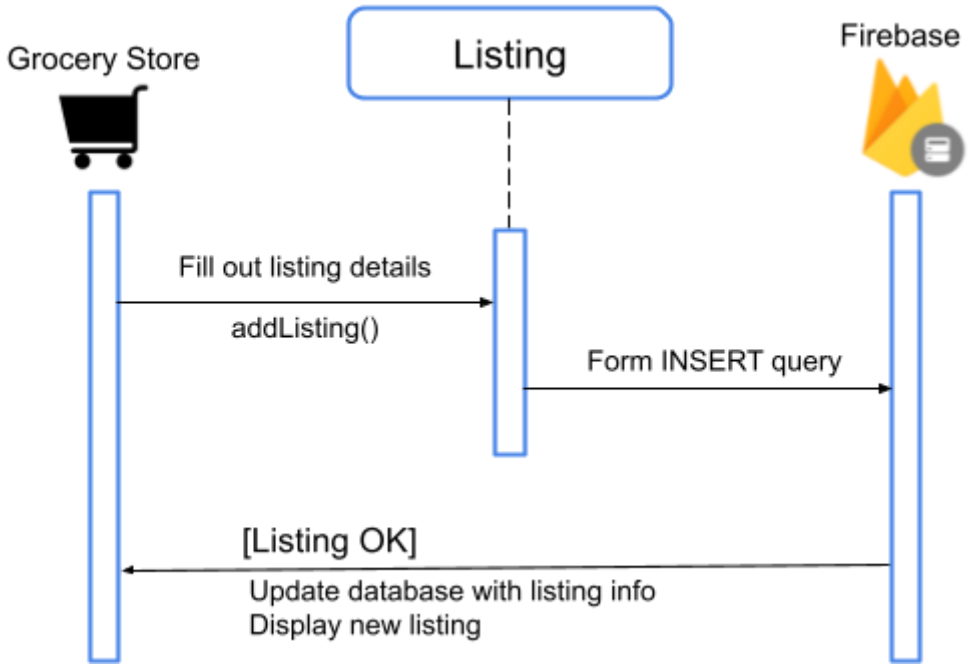
View financial information / delivery history



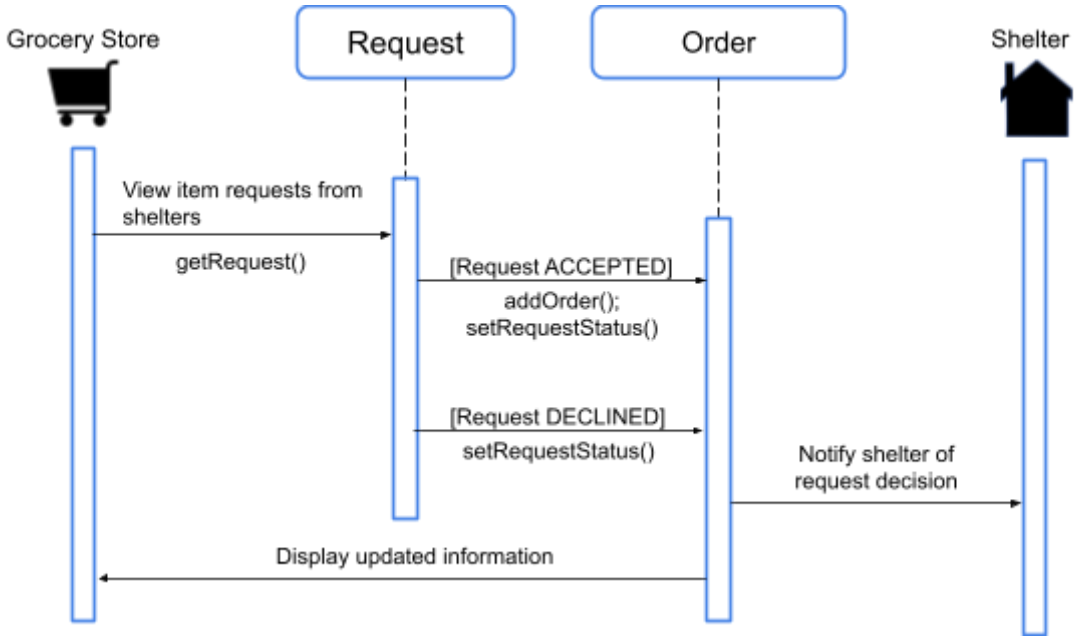
Navigate through overview map of local area



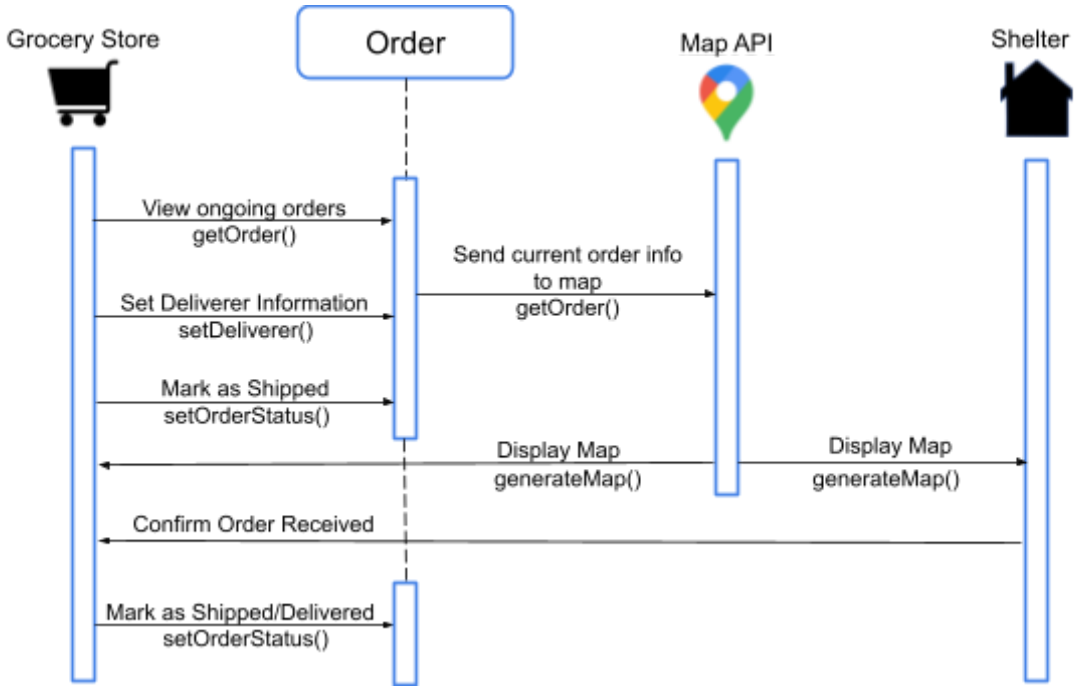
Publish Item Listings



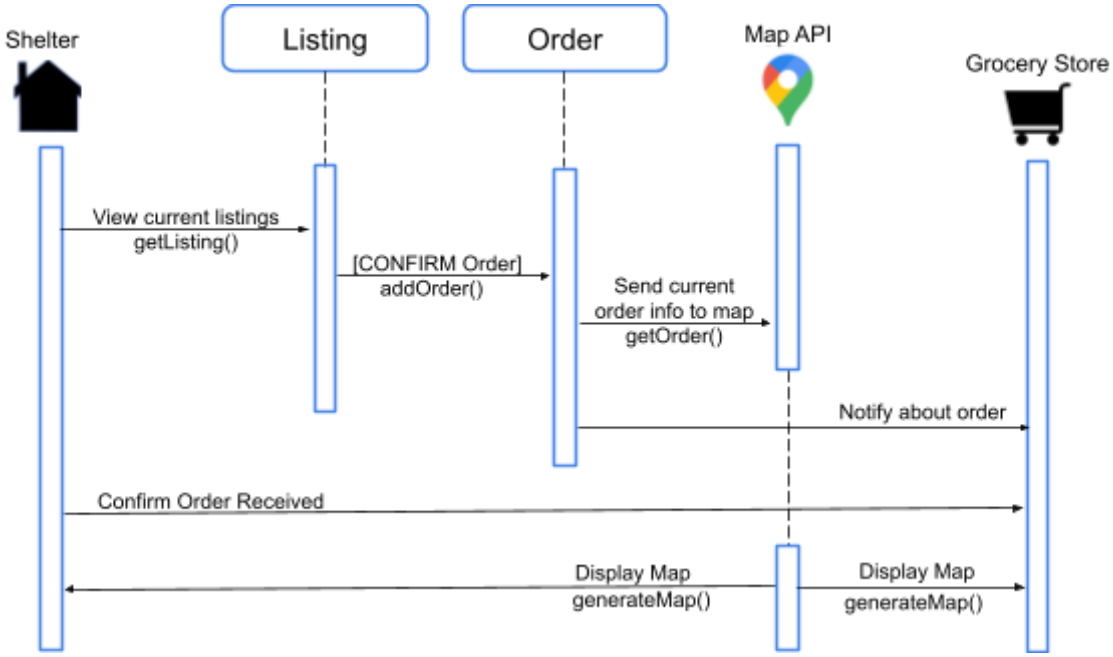
View item requests from shelters



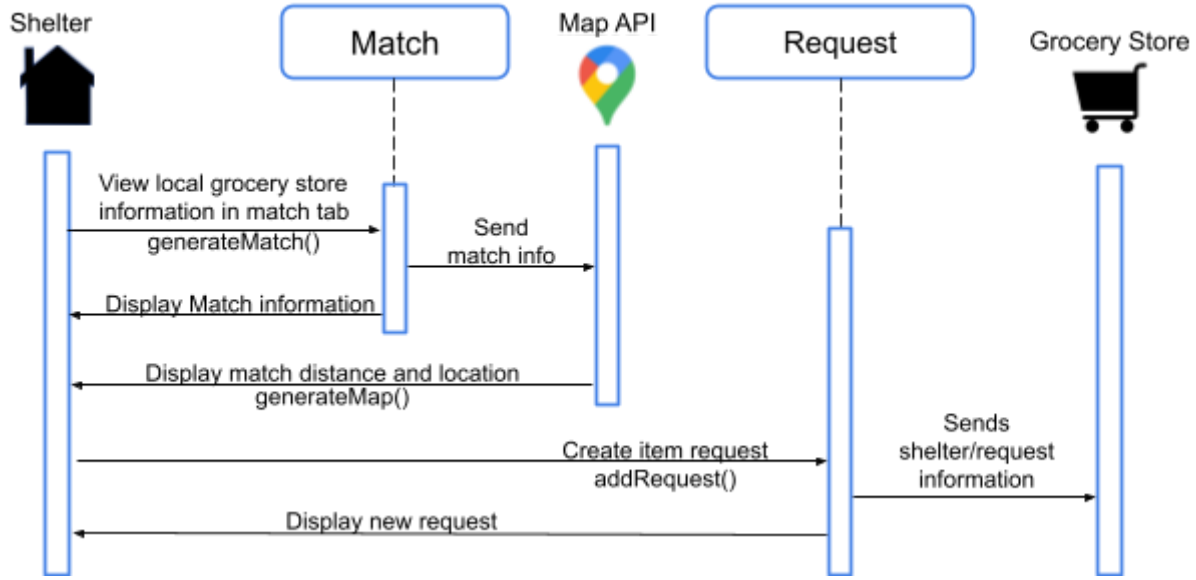
Manage order delivery information



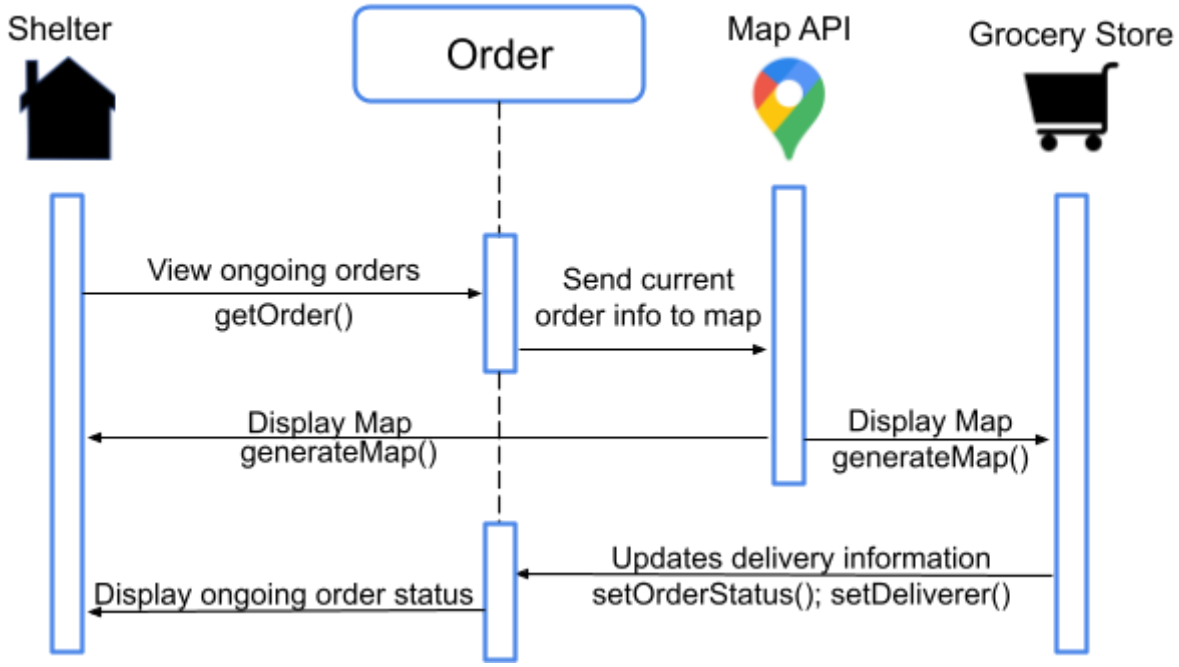
View/Select grocery store listings



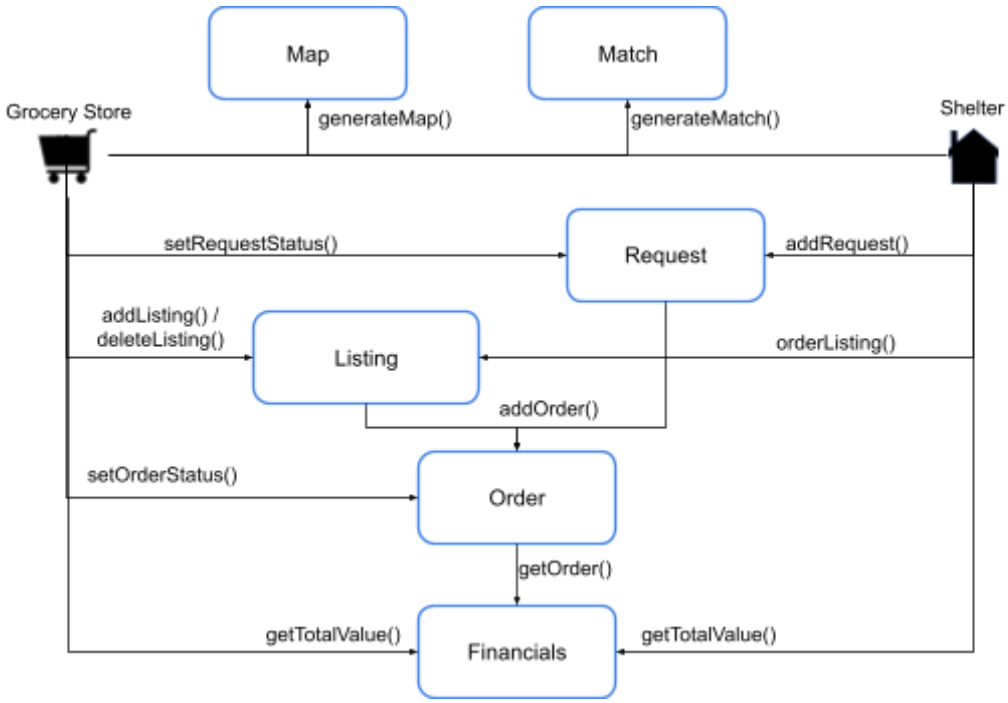
Make item requests



View order delivery information

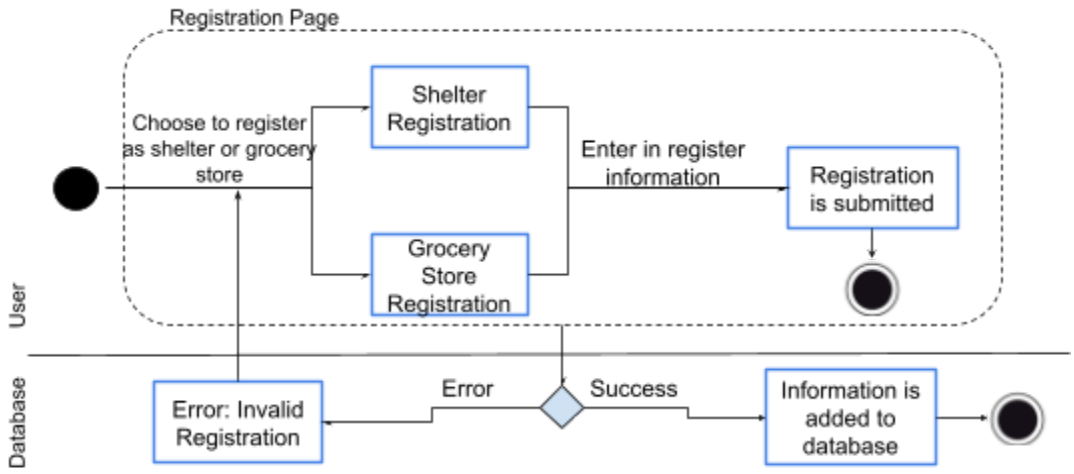


4.3.2 Collaboration Diagram

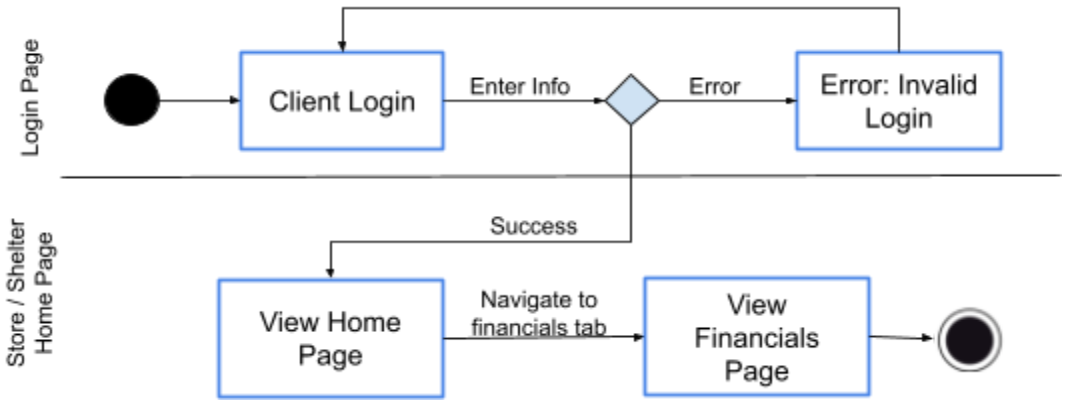


4.3.3 Activity Diagrams

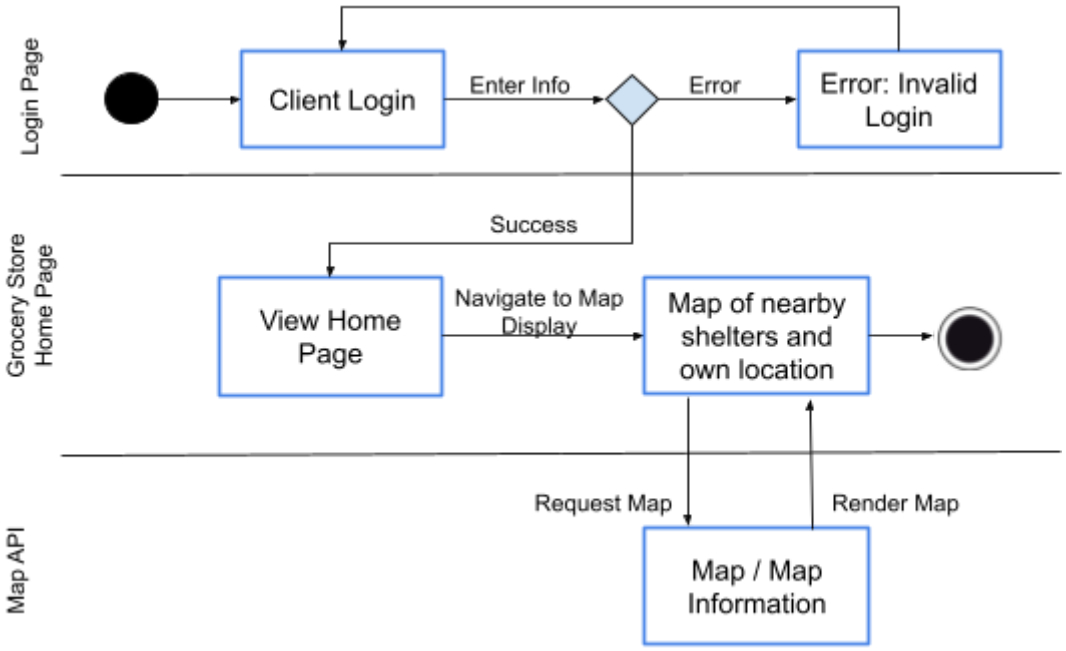
Register Location Information



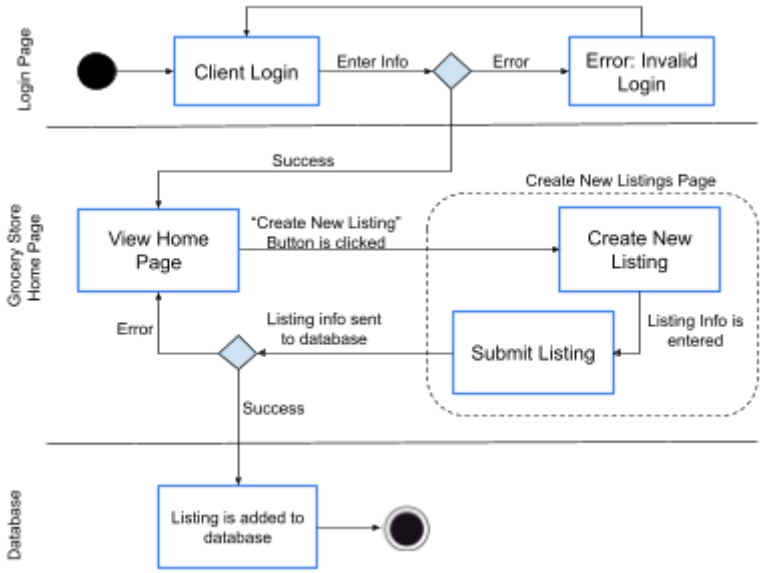
View financial information / delivery history



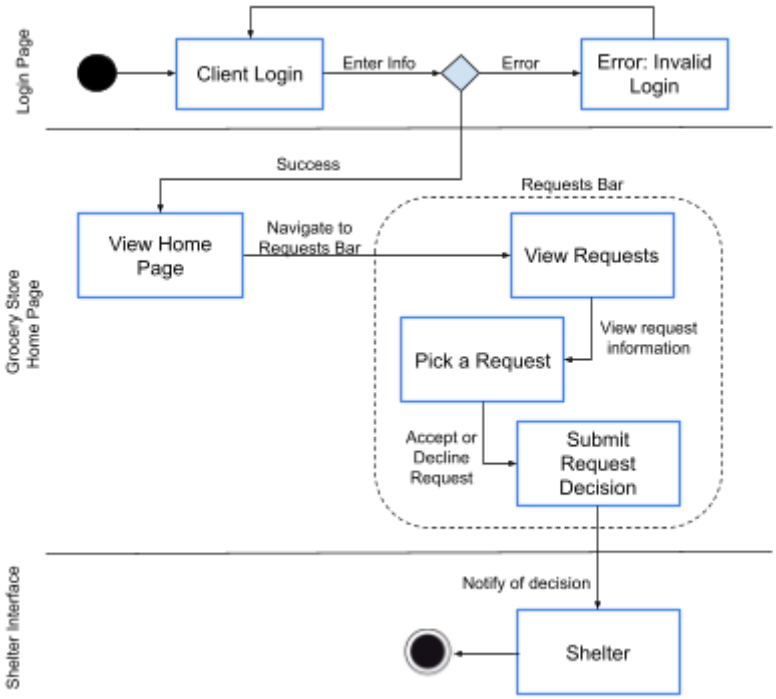
Navigate through overview map of local area



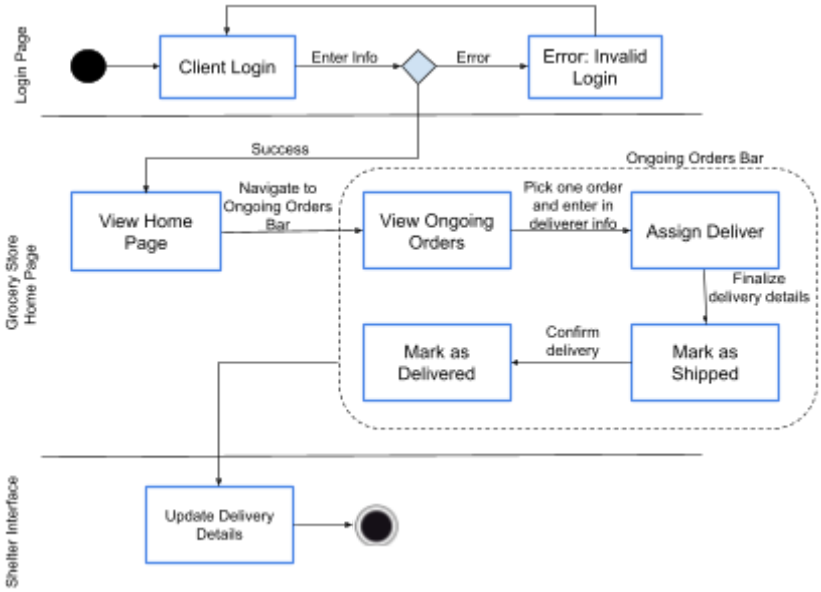
Publish Item Listings



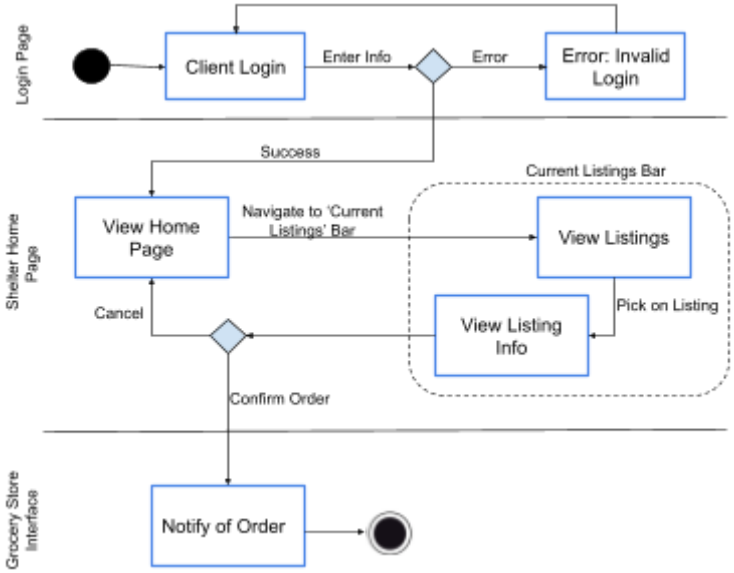
View item requests from shelters



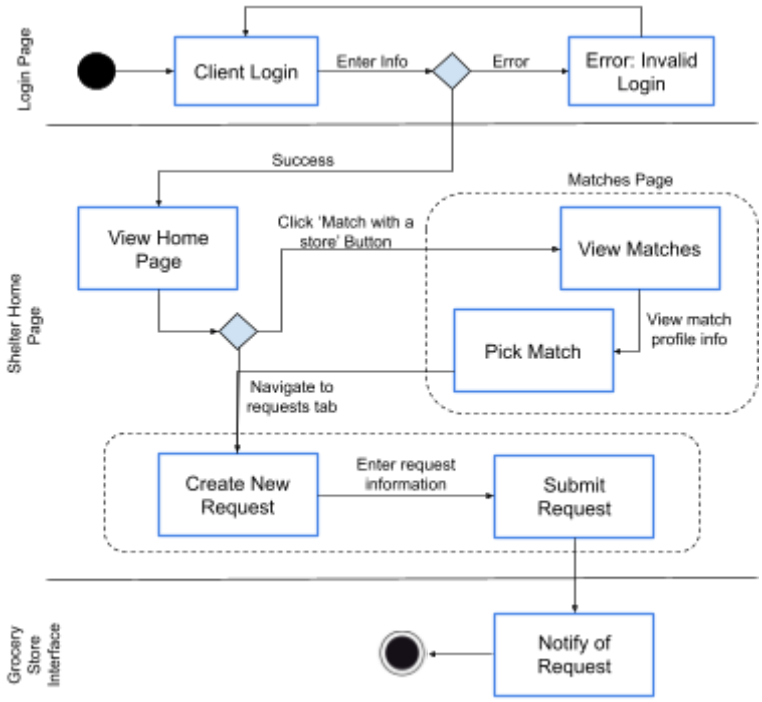
Manage order delivery information



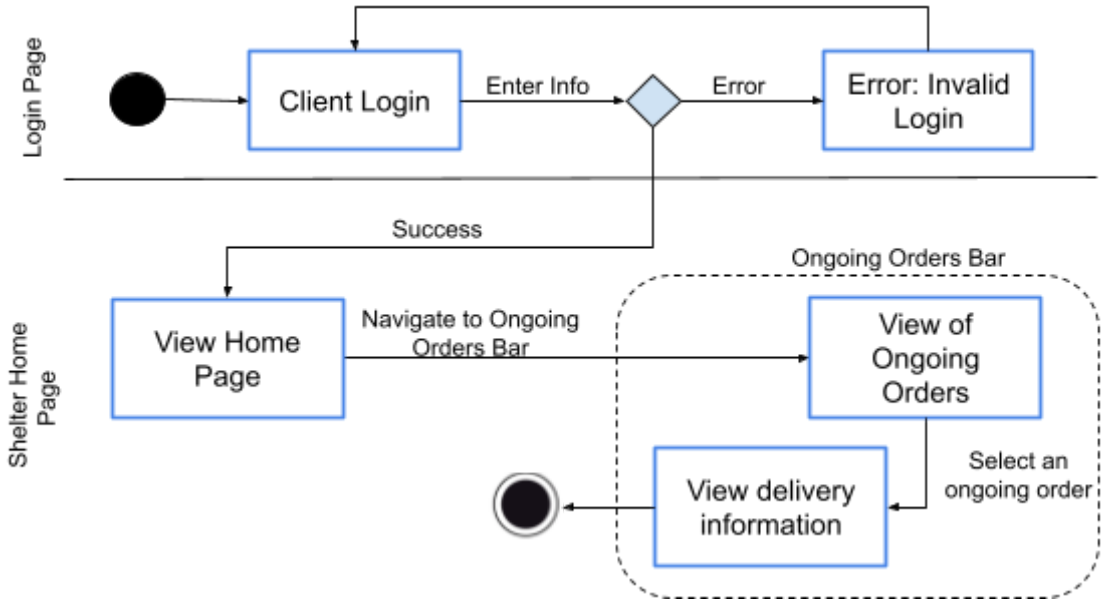
View/Select grocery store listings



Make item requests



View order delivery information

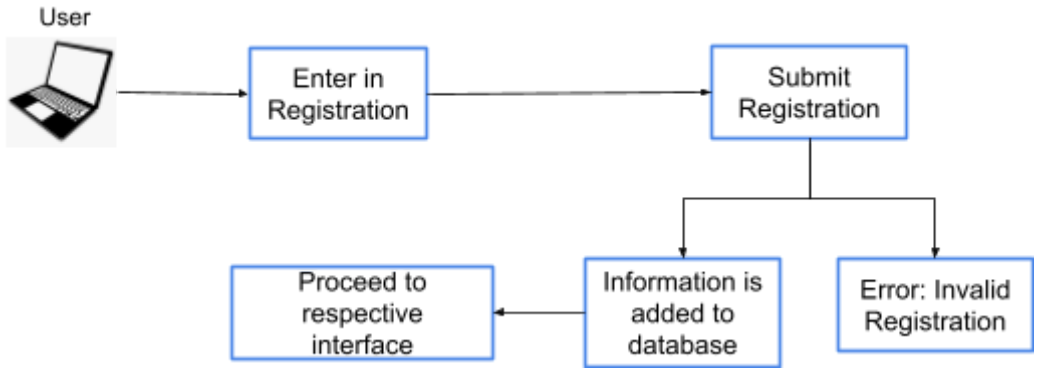


4.3.4

State Diagram

4.3.5 Event Diagrams

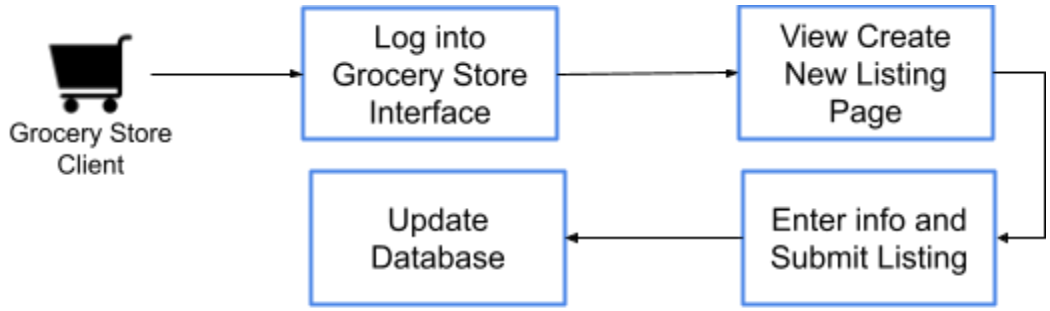
Register Location Information



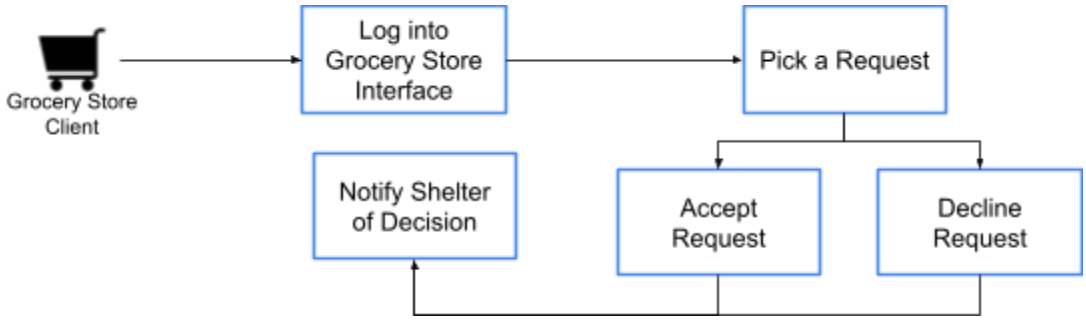
View financial information / delivery history

Navigate through overview map of local area

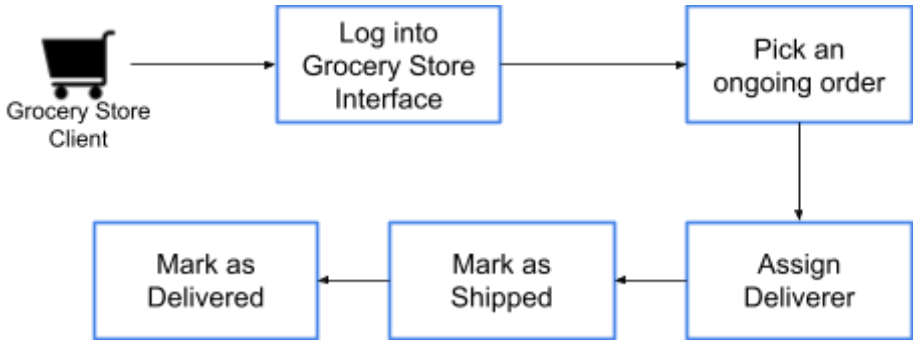
Publish Item Listings



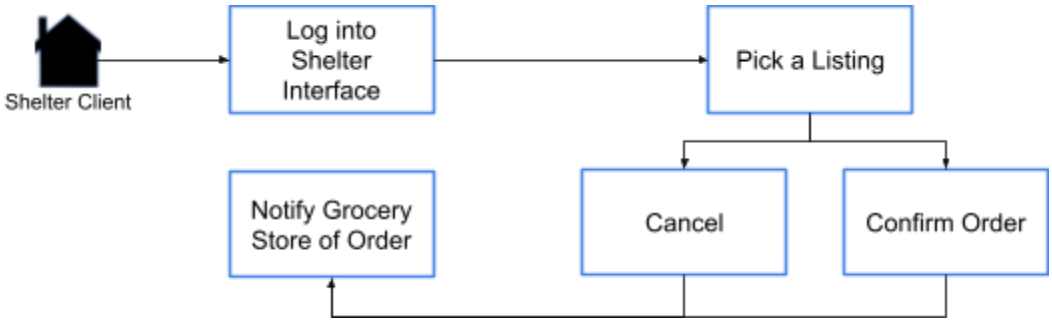
View item requests from shelters



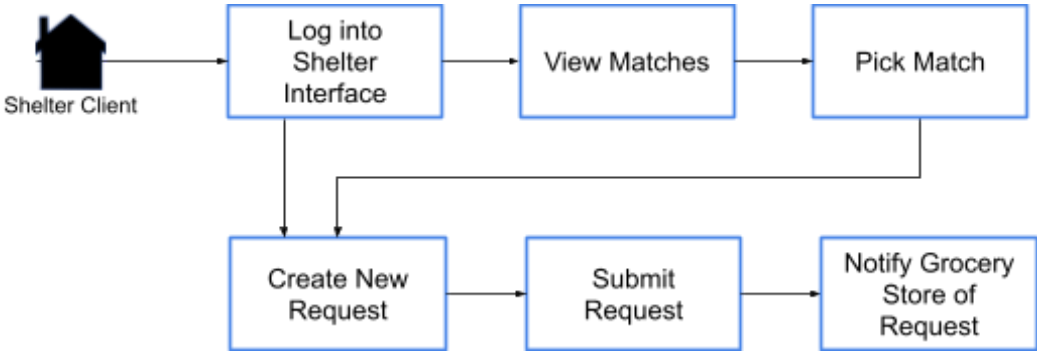
Manage order delivery information



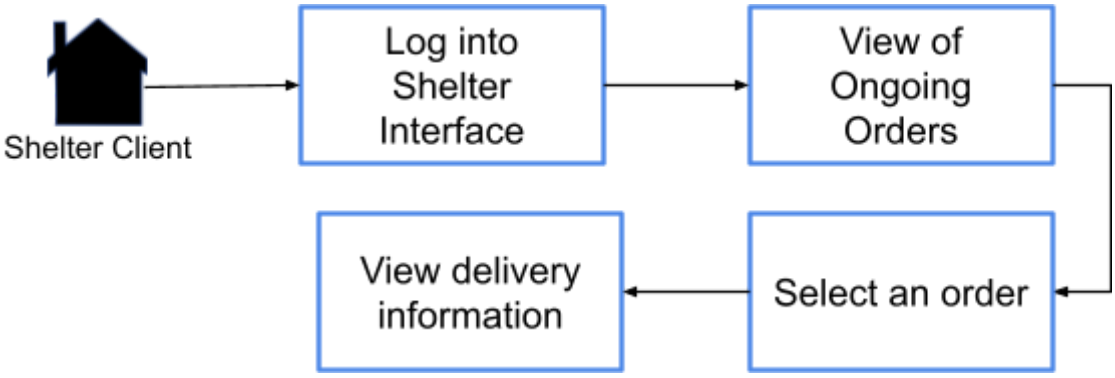
View/Select grocery store listings



Make item requests



View order delivery information



4.4 Concept of Execution

The user will be presented with the dashboard after accessing the Hunger Warrior URL and logging in to the system. The dashboard will contain navigation links to the user's financial information, to request/send items, and order-tracking services.

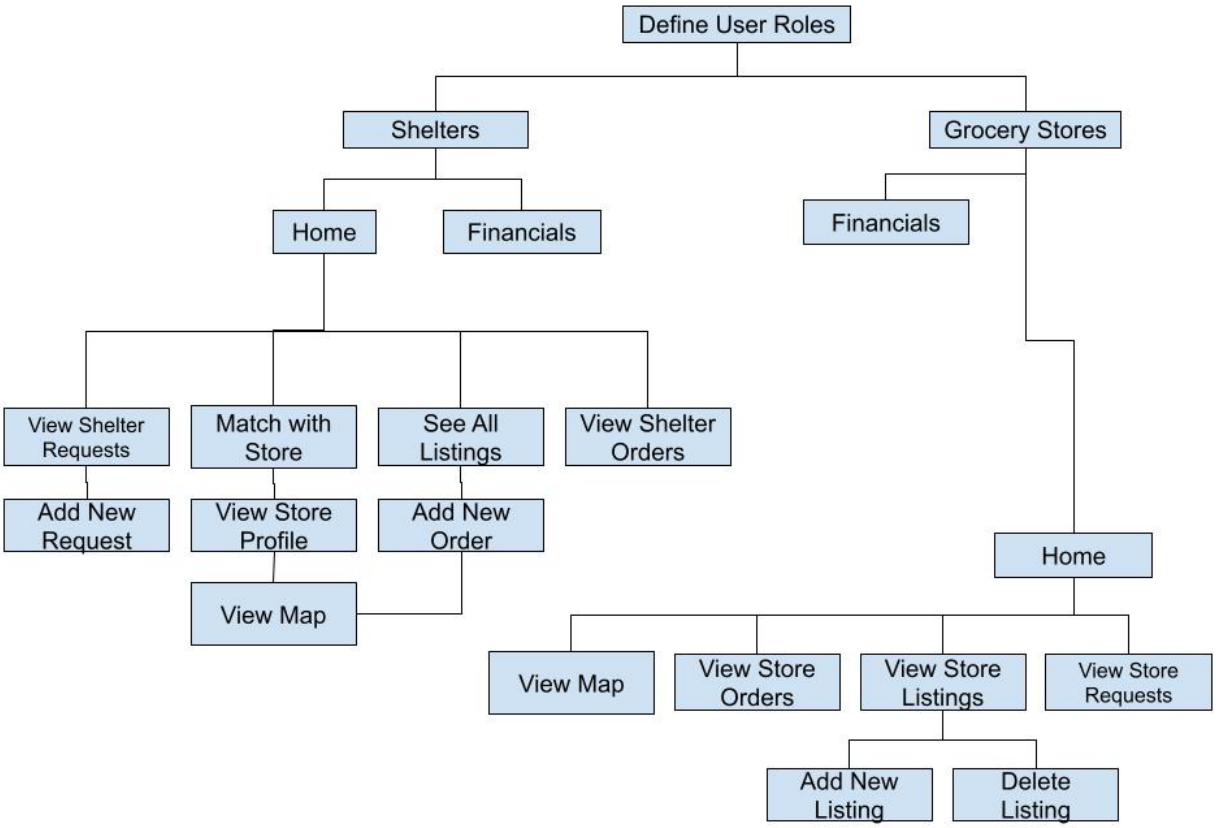
4.5 Interface Design

4.5.1 Unique Identifier of Interface

Refer to 4.1.1 for Class Diagrams

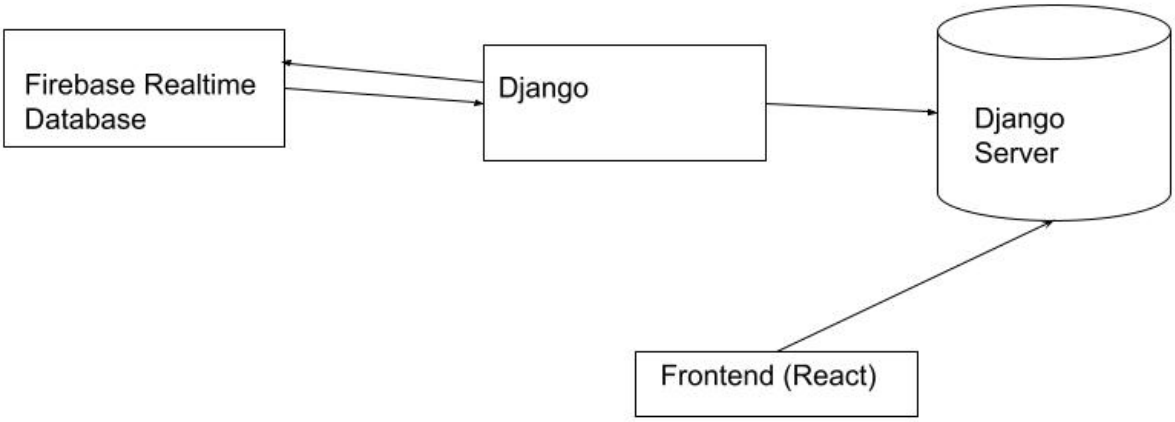
4.5.2 Interface Diagrams

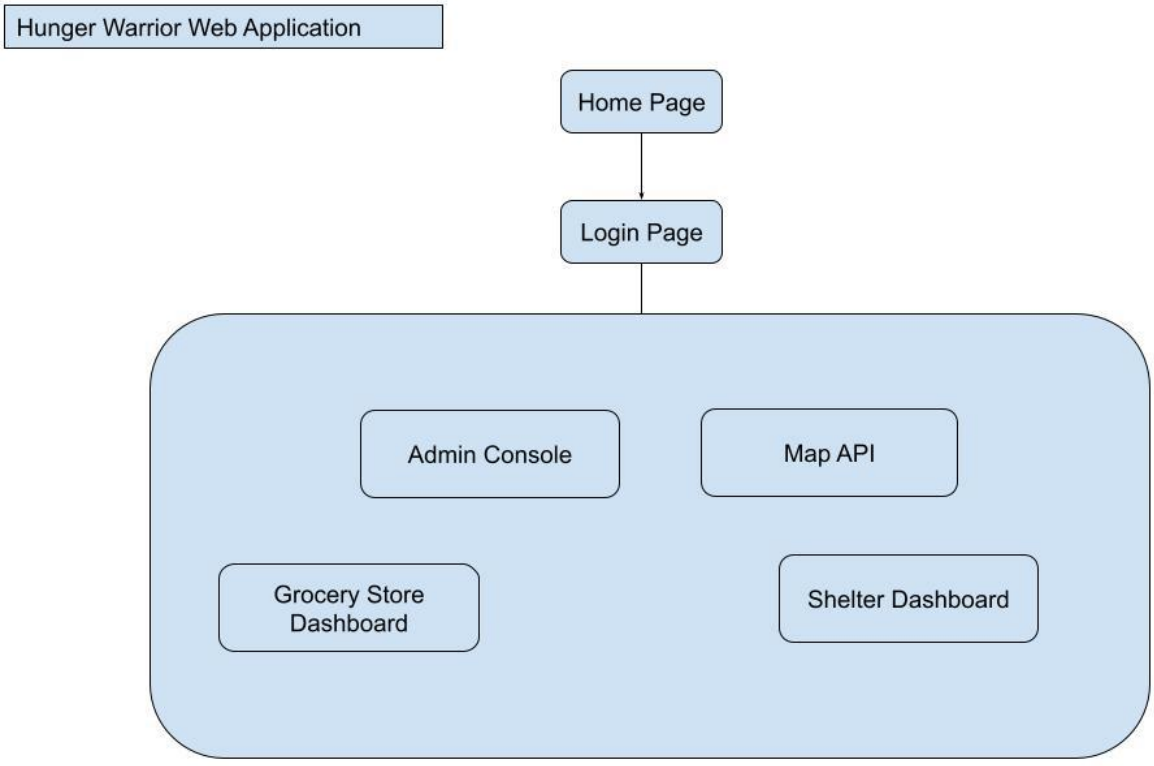
Refer to 4.1.1 for Class Diagrams



5. DEPLOYMENT ARCHITECTURE

5.1 Physical Deployment Architecture Diagram





6. DICTIONARIES

Classes

Name	Description	Methods	Attributes
Order	Food items that can be transferred from Grocery Stores to Shelters	getOrder() addOrder() setDeliverer() setOrderStatus()	Item Name, Item Quantity, Shelter Email, Store Email, Ordered Date, Order Status, Deliverer Name, Deliverer Email, Item Value
Shelter	Contains information about shelter	getShelterInformation()	Shelter Name, Shelter Location, Shelter Email
Grocery Store	Contains information about grocery store	getStoreInformation()	Store Name, Store Location, Store Email, Store Picture
Request	A food item that not within the store's inventory but is specifically requested by the shelter	getRequest() addRequest() setRequestStatus()	Item Name, Item Quantity, Shelter Email, Store Email, Requested Date, Request Status, Item Value
Financials	Shows financial information of client	getTotalValue()	All orders, Total donated/received value
Listing	Food item that is within a grocery store's inventory	getListing() orderListing() deleteListing()	Item Name, Item Quantity, Store Email
Match	Matches shelter with nearby grocery stores	generateMatch()	Shelter Longitude, Shelter Latitude, Store Longitude, Store Latitude
Map	Shows map information	generateMap()	Longitude, Latitude

Methods

Name	Description	Class	Arguments
getOrder()	Returns information of selected order	Order	order_id
addOrder()	Adds order information to the database	Order	item_name, item_quantity, item_value, ordered_date, ordered_by, ordered_from
setDeliverer()	Add deliverer information to the specific order	Order	deliverer_name, deliverer_phone
setOrderStatus()	Set status to specified order	Order	new_status
getStoreInformation()	Returns information of specified store	Store	store_email
getShelterInformation()	Returns information of specified shelter	Shelter	shelter_email
getRequest()	Returns information of selected request	Request	request_id
addRequest()	Adds request information to the database	Request	item_name, item_quantity, item_value, requested_by, requested_from
setRequestStatus()	Set status to specified request	Request	new_status
getListing()	Returns information of specified listing	Listing	listing_id
orderListing()	Creates order using information of specified listing	Listing	listing_id, item_name, item_quantity, store_email
deleteListing()	Removes listing information from the	Listing	listing_id

	database		
addListing()	Adds listing information to the database	Listing	item_name, item_quantity, store_email
getTotalValue()	Returns financial information of specified client	Financials	email
generateMap()	Generates and renders map using map information	Map	longitude, latitude
generateMatch()	Generates and creates matches using shelter and store information	Match	longitude, latitude

Attributes

Name	Description	C/S	Type	Size	Attributes
shelter_id	Unique Shelter Id	C	string	32 bits	None
store_id	Unique Store Id	C	string	32 bits	None
shelter_email	Unique Shelter Email	S	string	32 bits	None
store_email	Unique Store Email	S	string	32 bits	None
item_name	Item Name	S	string	32 bits	None
item_quantity	Item Quantity	S	int	64 bits	None
item_value	Item Value in Dollars	S	int	64 bits	None
latitude	Co-ordinate	C	float	64 bits	None
longitude	Co-ordinate	C	float	64 bits	None
ordered_from	Store Email	S	string	32 bits	None
ordered_by	Shelter Email	S	string	32 bits	None
requested_from	Store Email	S	string	32 bits	None
requested_by	Store Email	S	string	32 bits	None

order_status	Order Status	S	string	32 bits	None
request_status	Request Status	S	string	32 bits	None
deliverer_name	Deliverer Name	S	string	32 bits	None
deliverer_phone	Deliverer Phone	S	string	32 bits	None

Relationship

Name	Description	From Class	To Class	Optional/Mandatory	Cardinality
Create Listing	Creates a new store listing	Store	Listing	Mandatory	1 to 1
Views/Orders Listing	Views listing/creates a new order from listing	Shelter	Listing	Mandatory	1 to 1
Gets/sends order information	Gets information about the order	Shelter	Order	Mandatory	1 to many
Gets match info / sends location info	Gets a list of matchest ordered by distance from the shelter from least to highest	Shelter	Match	Mandatory	1 to many
Sends request info	Sends and creates a new shelter request	Shelter	Request	Mandatory	1 to many
Gets total value	Gets total value of donations from orders	Store	Financials	Mandatory	1 to 1
Gets total value	Gets total value of donations from orders	Shelter	Financials	Mandatory	1 to 1
Gets request	Gets request	Store	Request	Mandatory	1 to

info / Sends request status	information, sets request status (approved/denied)				many
Gets order info	Gets the list of all orders in order to calculate total value	Order	Financials	Mandatory	Many to 1
Gets request info	Gets request information to add as a new order	Order	Request	Mandatory	1 to 1
Gets shelter, store location	Gets locations of shelter and store to display in map	Order	Map	Mandatory	1 to 1

7. SOFTWARE ITEM COMPUTER RESOURCE UTILIZATION

Firebase Realtime Database is the Database that is used for the Hunger Warrior Application. We use Django and Firebase as the back-end to store and access the data from the front end which is built out in React JS. Each of these technologies serve a different purpose in order to accomplish the goal of building an interactive dashboard that is easy to use for employees and volunteers at shelters to acquire and distribute food-items.

8. REQUIREMENTS TRACEABILITY

8.1 Software Component-Level Requirements Traceability

Requirement	Description	Artifact first present in	Creation Date	Modification Date	Latest Artifact
Functional Requirements	Services the system should provide, how the system should react to particular inputs and situations. Part of user requirements.	SRS 2.0	March 16, 2021	September 27, 2021	SRS 4.0
Non-functional Requirements	Constraints on the services or functions offered by the system.	SRS 2.0	March 16, 2021	September 27, 2021	SRS 4.0
Business Requirements	Business drivers for the system	SRS 2.0	March 16, 2021	September 27, 2021	SRS 4.0

9. SYSTEM DESIGN TESTING

Unit tests will be run periodically to maintain the integrity of the web application. Using the testing environments specified below, the development team will maintain and update the product accordingly, taking into account user feedback and system requirements. The software will be developed using method techniques in order to

ensure security, usability, and quality service. Testing will be done continuously in order to mitigate defects within the product, with quality analysis reports being created to manage defects found and generate estimated defects. After every major update to the design, heavy testing will be done to ensure that the current update has integrated into the system accordingly and not produced any unforeseen errors.

Product Test

- A demo version of the website will be created for a few grocery stores and shelters. Feedback and errors will be reported.
- Shelters will test the log-in system, check if they are able to view their inventory, and verify that they are able to request and order food items.
- Grocery stores will test the log-in system, check if they are able to view their inventory, and verify that they are able to accept requests and view/edit orders.

Acceptance Testing

- Acceptance Criteria
    - Any reported bugs or system inconsistencies should be fixed
- Acceptance Tests
    - The tests below will be run with a demo version of the website. The demo will be released publicly if these tests run successfully.
    - Grocery stores and homeless shelters should be able to log in successfully with their email and password.
    - After logging in, shelters will be able to view their dashboard to see current orders, listings, requests, and all available stores. They should also be able to match with a store. Stores should be able to see current orders, listings, requests, and nearby shelters in a map.

- Shelters should be able to request and order food items to grocery stores of their choice and respective grocery stores should be able to view the requests and orders in real-time.
- Stores should be able to manage delivery status information.
- Stores should also be able to designate employees for delivery preparation. .

10. RATIONALE

11. NOTES

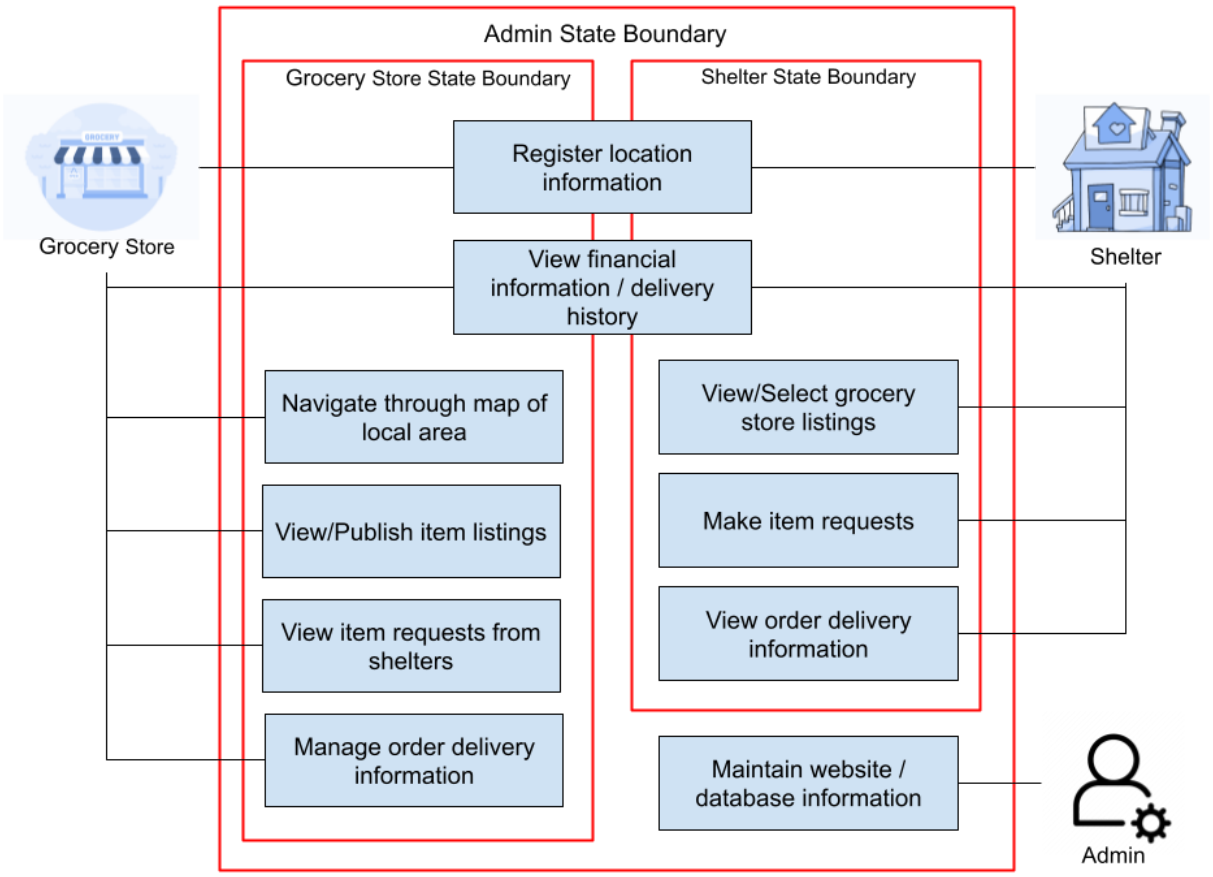
12. APPENDICES

12.1 Requirements Diagrams

Functional Requirement Definitions:

- Users shall be able to register as either a shelter or a grocery store
- Stores shall be able to accept/decline food requests made by a shelter. If accepted, the request should establish itself as a new order. If declined, it will remain in food requests until further notice.
- Stores shall be able to edit their item inventory / item listings.
- Stores shall be able to edit order delivery information.
- Stores shall be able to view all shelters nearest to them through a map display.
- Shelters shall be able to provide their location and get a list of grocery store profiles nearest them through a ‘match’ service.
- Shelters shall be able to request items not currently available in the store’s inventory.
- Shelters shall be able to order items currently available from a grocery store of their choice.
- Shelters shall be able to view and receive updates on order delivery tracking information.
- Shelters and Stores shall be able to view their order history and financial information.

Use Case Diagram



Use Case Descriptions

Register Location Information		
Description	Shelters / Grocery stores are able to register for Hunger Warrior and input necessary information	
Pre-Conditions	The web page must be loaded, connected to the internet, connected to the database, and a request form is generated.	
Flows	**Basic**	1. Fill out necessary information in order to complete registration 2. Confirm the registration clicking the ‘Sign up’ Button
Post Conditions	Once the registration is complete, the application will convert the address into geo-coordinates and store it in the users database.	
Special Requirements	1. When necessary, admin will be able to register entities on their end in case any problems arise. 2. The stores/shelters must provide a valid address.	
Extension Points	None	

View financial information / delivery history		
Description	Shelters / Grocery stores are able to view financial information regarding how much they have donated / been donated to.	
Pre-Conditions	The webpage must be loaded, the client must be logged into their respective dashboard, and the financials tab must be clicked on.	
Flows	**Basic**	1. Click the financials tab and scroll in order to view details
Post Conditions	Shelters / Grocery stores are able to view their financial information.	

Special Requirements	None
Extension Points	None

Navigate through map overview of local area		
Description	Grocery stores can navigate through all nearby shelters through the Google Maps API	
Pre-Conditions	The web page must be loaded, connected to the internet, connected to the database, and must show a visualization of the local area map.	
Flows	**Basic**	1. Scroll vertically and horizontally through the local area map 2. Drag map finger or computer mouse in any direction 3. Pins populate with shelters in the local area.
Post Conditions	The map will look different based on user changes.	
Special Requirements	When necessary, the map will regularly be updated to support the latest version of Google Maps API and bugs on the page will be fixed.	
Extension Points	None	

View / Publish Item Listings		
Description	Grocery stores are able to post available inventory listings that are ready for deployment	
Pre-Conditions	The web page must be loaded, connected to the internet, connected to the database, and a grocery store must be logged in.	
Flows	**Basic**	1. A form will be presented to the store 2. Stores can fill up and submit the form in order to register a new listing.

Post Conditions	The local shelters will be able to see these listings and will be able to choose them as they see fit.
Special Requirements	When necessary, bugs on the page will be fixed.
Extension Points	None

View Item Requests from Shelters		
Description	Grocery stores will be able to see requests that have been made from local shelters and be able to either accept them or deny them.	
Pre-Conditions	The web page must be loaded, connected to the internet, connected to the database, and a grocery store must be logged in.	
Flows	**Basic**	1. A list of all requests made to the store will be displayed. 2. Grocery stores can view the details of a particular request. 3. Grocery stores can accept/deny the request.
Post Conditions	If accepted, the request will be added to the orders database and will be displayed in the “orders” section. If rejected, the request will stay in the database as a record.	
Special Requirements	When necessary, bugs on the page will be fixed.	
Extension Points	None	

Manage order delivery information	
Description	Grocery Stores will be able to set a deliverer to a particular order.

Pre-Conditions	The web page must be loaded, connected to the internet, connected to the database, and a grocery store must be logged in.	
Flows	**Basic**	1. The store will be able to click on a particular order, and will then be presented with the order details. 2. The store will be able to set delivery status. 3. The store will also be presented with a form to set deliverer information. 4. The store can fill up and submit the form.
Post Conditions	The store will be able to see the order update in real time.	
Special Requirements	When necessary, bugs on the page will be fixed.	
Extension Points	None	

View / Select grocery store listings		
Description	A shelter should be able to view or select a particular listing.	
Pre-Conditions	The web page must be loaded, connected to the internet, connected to the database, and a grocery store must be logged in.	
Flows	**Basic**	1. The shelter will be provided with a list of all available listings made from a store. 2. The shelter may choose to order the listing.
Post Conditions	If shelter orders the listing, the listing details will be added to the orders database.	
Special Requirements	When necessary, bugs on the page will be fixed.	
Extension Points	None	

Make item requests		
Description	A shelter should be able to make new item requests that aren't already available in any of the store's listings.	
Pre-Conditions	The web page must be loaded, connected to the internet, connected to the database, and a grocery store must be logged in.	
Flows	**Basic**	1. The shelter will be provided with a form to make a new request.
Post Conditions	When submitted, the request will be added to the requests database.	
Special Requirements	When necessary, bugs on the page will be fixed.	
Extension Points	None	

View order delivery information		
Description	Shelters are able to view delivery information regarding their specific order. This information includes the status of the order and the contact information of the deliverer.	
Pre-Conditions	The webpage must be loaded, the client must be logged into their respective interface, an order must've been placed with a grocery store, and the specific order must be clicked on in order to be navigated to the ongoing order detail page.	
Flows	**Basic**	1. Click on the specific ongoing order in order to view specific details of that specific order
Post Conditions	The shelter will be updated with the delivery information provided by the grocery store	
Special Requirements	None	
Extension Points	None	

Maintain website / database information		
Description	Admin, whenever necessary, will be able to perform any other use case in order to maintain the website and database information	
Pre-Conditions	An authenticated user from the Hunger Warrior Development team will log in with their credentials and gain special access to the Hunger Warrior software.	
Flows	**Basic**	1. Admin users will go in and make necessary changes / updates to potential bugs within the Hunger Warrior software.
Post Conditions	The website and database will be reloaded and all updates will be implemented	
Special Requirements	None	
Extension Points	None	

12.2 Schedule Tracking

Artifact or Deliverable	Who	Estimated	Actual	Difference
SDD-Initial	Gavin Senger	6 hours	5 hours	1 hour(s)
	Sanidhya Sitaula	6 hours	5 hours	1 hour(s)
	Kori Vernon	3 hours	5 hours	2 hour(s)
	Courtney Battieste	3 hours	3 hours	0 hour(s)
	Team Summary	18 hours	18 hours	4 hour(s)

Artifact or Deliverable	Who	Estimated	Actual	Difference
SDD-Final	Gavin Senger	3 hours	3 hours	0 hour(s)
	Sanidhya Sitaula	3 hours	4 hours	1 hour(s)
	Kori Vernon	3 hours	3 hours	0 hour(s)
	Courtney Battieste	3 hours	4 hours	1 hour(s)
	Team Summary	12 hours	14 hours	2 hour(s)

Artifact or Deliverable	Who	Estimated	Actual	Difference
Project Code-Final	Gavin Senger	6 hours	5 hours	1 hour(s)
	Sanidhya Sitaula	6 hours	8 hours	2 hour(s)
	Kori Vernon	6 hours	8 hours	2 hour(s)
	Courtney Battieste	6 hours	5 hours	1 hour(s)
	Team Summary	24 hours	26 hours	6 hour(s)

Artifact or Deliverable	Who	Estimated	Actual	Difference
Project Presentation	Gavin Senger	2 hours	3 hours	1 hour(s)
	Sanidhya Sitaula	2 hours	2 hours	0 hour(s)
	Kori Vernon	2 hours	3 hours	1 hour(s)
	Courtney Battieste	2 hours	4 hours	2 hour(s)
	Team Summary	8 hours	12 hours	4 hour(s)

	Who	Estimated	Actual	Difference
Cumulative	Gavin Senger	17 hours	16 hours	1 hour(s)
	Sanidhya Sitaula	17 hours	19 hours	2 hour(s)
	Kori Vernon	14 hours	19 hours	2 hour(s)
	Courtney Battieste	14 hours	16 hours	2 hour(s)
	Team Summary	62 hours	70 hours	8 hour(s)

12.3 Defect Tracking

Artifact or Deliverable	Who	Estimated	Actual	Difference
SDD-Initial	Gavin Senger	2 hours	2 hours	0 hour
	Sanidhya Sitaula	2 hours	2 hours	0 hour
	Kori Vernon	2 hours	2 hours	0 hour
	Courtney Battieste	2 hours	2 hours	0 hour
	Team Summary	8 hours	8 hours	0 hour

Artifact or Deliverable	Who	Estimated	Actual	Difference
SDD-Final	Gavin Senger	1 hour	1 hour	0 hour
	Sanidhya Sitaula	1 hour	1 hour	0 hour
	Kori Vernon	1 hour	1 hour	0 hour
	Courtney Battieste	1 hour	1 hour	0 hour
	Team Summary	4 hours	4 hours	0 hour

Artifact or Deliverable	Who	Estimated	Actual	Difference
Project Code-Final	Gavin Senger	1 hour	1 hour	0 hour
	Sanidhya Sitaula	1 hour	1 hour	0 hour
	Kori Vernon	1 hour	1 hour	0 hour
	Courtney Battieste	1 hour	1 hour	0 hour
	Team Summary	4 hours	4 hours	0 hour

Artifact or Deliverable	Who	Estimated	Actual	Difference
Project Presentation	Gavin Senger	1 hour	1 hour	0 hour
	Sanidhya Sitaula	1 hour	1 hour	0 hour
	Kori Vernon	1 hour	1 hour	0 hour
	Courtney Battieste	1 hour	1 hour	0 hour
	Team Summary	4 hours	4 hours	0 hour

	Who	Estimated	Actual	Difference
Cumulative	Gavin Senger	5 hours	5 hours	0 hour
	Sanidhya Sitaula	5 hours	5 hours	0 hour
	Kori Vernon	5 hours	5 hours	0 hour
	Courtney Battieste	5 hours	5 hours	0 hour
	Team Summary	20 hours	20 hours	0 hour

12.4 Project Schedule

Gantt Chart: