



# Computer Science and Engineering

---

## EcoFoodie

## Software Design Description (SDD)

**Version 2.0**

Document Number: SDD-002

Team Number: B23

Team Members:	Nathan Cantu	snc387
	Vijay Kallem	vrk248
	Jonathan Xu	jx990

---

## REVIEW AND APPROVALS

Printed Name and Title	Function (Author, Reviewer, Approval)	Date	Signature
Nathan Cantu	Author, Reviewer, Approval	12/14/2022	Nathan C
Vijay Kallem	Author, Reviewer, Approval	12/14/2022	Vijay K
Jonathan Xu	Author, Reviewer, Approval	12/14/2022	Jonathan X

---

**REVISION LEVEL**

Date	Revision Number	Purpose
Fall 2022	Version 1.0	Initial Release
Fall 2022	Version 2.0	Final Release

---

**Table of Contents**

<b>1. INTRODUCTION</b>	<b>1</b>
1.1 PURPOSE	1
1.2 SCOPE	1
1.3 IDENTIFICATION	1
1.4 DOCUMENT SUMMARY	1
1.5 SYSTEM OVERVIEW	2
<b>2. REFERENCE DOCUMENTS</b>	<b>2</b>
<b>3. SYSTEM WIDE DESIGN DECISIONS</b>	<b>2</b>
3.1 SOFTWARE COMPONENT ARCHITECTURAL DESIGN	2
3.2 SOFTWARE ARCHITECTURE GENERAL DESCRIPTION	2
3.3 SOFTWARE ITEM COMPONENTS	3
3.4 COMPONENT INTERFACE IDENTIFICATION	3
3.5 SOFTWARE COMPONENT CONCEPT OF EXECUTION	3
<b>4. SOFTWARE ITEM DETAILED DESIGN</b>	<b>4</b>
4.1 STRUCTURE	4
4.2 STATIC RELATIONSHIP OF SOFTWARE UNIT	6
4.3 BEHAVIOR	6
4.4 CONCEPT OF EXECUTION	15
4.5 INTERFACE DESIGN	15
<b>5. IMPLEMENTATION ARCHITECTURE (NOT REQUIRED)</b>	<b>16</b>
5.1 ALL ACTIVE AND PASSIVE CLASSES ASSIGNED TO COMPONENTS	16
5.2 DIAGRAMS OF PHYSICAL PACKAGING OF LOGICAL COMPONENTS	16
<b>6. DEPLOYMENT ARCHITECTURE</b>	<b>16</b>
6.1 PHYSICAL DEPLOYMENT ARCHITECTURE DIAGRAM	16
<b>7. DICTIONARIES</b>	<b>17</b>
<b>8. SOFTWARE ITEM COMPUTER RESOURCE UTILIZATION</b>	<b>25</b>
<b>9. REQUIREMENTS TRACEABILITY</b>	<b>25</b>
9.1 SOFTWARE COMPONENT-LEVEL REQUIREMENTS TRACEABILITY	25
<b>10. SYSTEM DESIGN TESTING</b>	<b>26</b>
<b>11. RATIONALE</b>	<b>27</b>
<b>12. NOTES</b>	<b>27</b>
<b>13. APPENDICES</b>	<b>28</b>
13.1 DICTIONARIES	28
13.2 UML DIAGRAMS	28
13.3 REQUIREMENTS DIAGRAMS	29
13.4 SCHEDULE TRACKING	37
13.4 DEFECT TRACKING	39
13.5 PROJECT SCHEDULE	41

---

## 1. INTRODUCTION

---

### 1.1 Purpose

The purpose of this document is to define the contents of the Software Design Description (SDD). The SDD documents the System Architecture and the Detailed Design. The document is used to communicate overall quantitative and qualitative system characteristics to the software development team, technical support, training, and operators.

### 1.2 Scope

This document may describe software-only products, hardware, or a combination of each. However, hardware only products are not described in this standard. This standard is applicable to all forms of software products media, including application, infrastructure, embedded systems, and operational scripts. The SDD is formally delivered as part of the software product release package.

### 1.3 Identification

Software Design Description, SDD-001 Version 1.0

### 1.4 Document Summary

- Title Page
- Review/Approval Signatures
- Table of Revisions
- Table of Contents
- Introduction
- Reference Documents
- System Wide Design Decisions
- Software Item Detailed Design
- Deployment Architecture
- Dictionaries
- Software Item Computer Resource Utilization
- Requirements Traceability
- System Design Testing
- Appendices
- Schedule/Defect Tracking

## 1.5 System Overview

The frontend part of our system will be done with React Native to seem like a forum-type interface and all the information will be stored in a PostgreSQL database through Django using pgAdmin4. We will be utilizing the google maps API in order to do location-based searching. Most of the hardware usage will come from the user's smartphone for internet connection, and GPS tracking.

## 2. REFERENCE DOCUMENTS

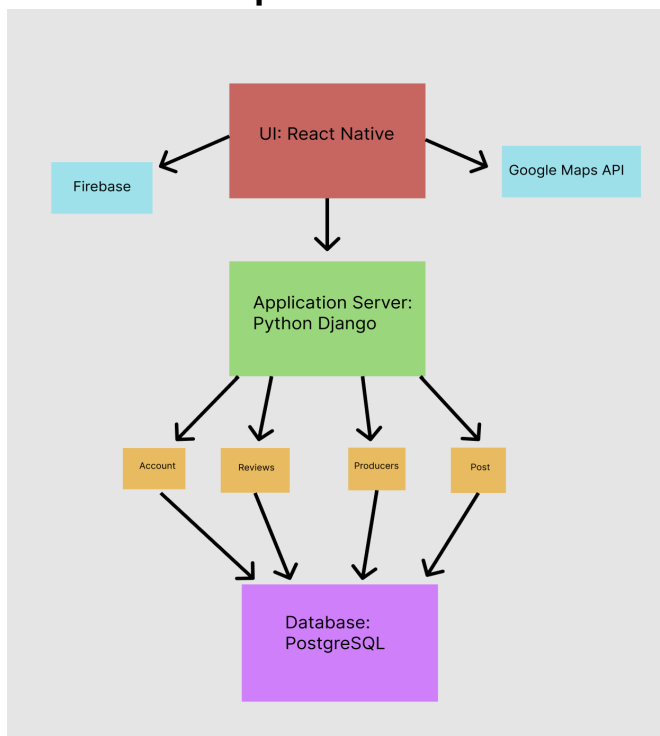
---

- Eco-Foodie, B23
- Project Team Selection Form Version 1.0: 9/14/22
- Project Proposal Version 1.0: 9/21/22
- Software Analysis/Requirements Specification (SRS) Version 1.0 : 9/26/22
- Software Project Management Plan (SPMP) Version 1.0: 10/11/22
- Project Information/Description Form Version 1.0 : 10/12/22

## 3. SYSTEM WIDE DESIGN DECISIONS

---

### 3.1 Software Component Architectural Design



### 3.2 Software Architecture General Description

The front-end UI shall consist of React Native Components. It will be composed of

login and account creation form pages, account pages, social media style timelines, terms and conditions page, review pages and have the functionality of GPS tracking, camera and microphone access, and a messaging service for group chat conversations. The backend Django framework shall consist of data being formed into four primary classes: Reviews, Users, Producers, and Posts. The data being sent from and to Django shall be stored in PostgreSQL using pgAdmin4.

### 3.3 Software Item Components

React Native: Login and Account Creation Forms, Social Security Validation, GPS tracking, camera and microphone access, timer, current date and time, email verification, password encryption, social media style post timeline page, terms and conditions page, group chat page, account review pages, account pages, settings page, and a help page.

Django: Models and Serializers for data grouped into the categories of Reviews, Users, Producers, and Posts.

PostgreSQL: Tables for Users, Producers, Photos, Videos, Groups, Posts, Messages, Reviews, and Friends

### 3.4 Component Interface Identification

Front Connection: React Native -> Django

Back Connection: Django -> PostgreSQL

Account Data: Login and Account Creation Forms -> Django

TrackingAccess: GPS tracking-> Django

MediaAccess: Camera and microphone access -> Django

Date/Time: Current date and time -> Django

AcceptPost: Customers -> Post

AddGroupMember: Account -> Group

AddFriend: Account -> Account

DeleteGroupMember: Account -> Group

DeleteFriend: Account -> Account

SendGroupAlert: Post -> Group

SendAlert: Post -> Customers

CreatePost: Account -> Post

CreateReview: Account -> Post

HelpMe: Account -> Admin

DeletePost: Admin -> Post

DeleteUser: Admin -> Account

### 3.5 Software Component Concept of Execution

The application Django server gets launched on localhost with access to the PostgreSQL database through the IP address, React Native fetches data from the application server through the fetch method returning a Promise of the data requested

## 4. SOFTWARE ITEM DETAILED DESIGN

---

### 4.1 Structure

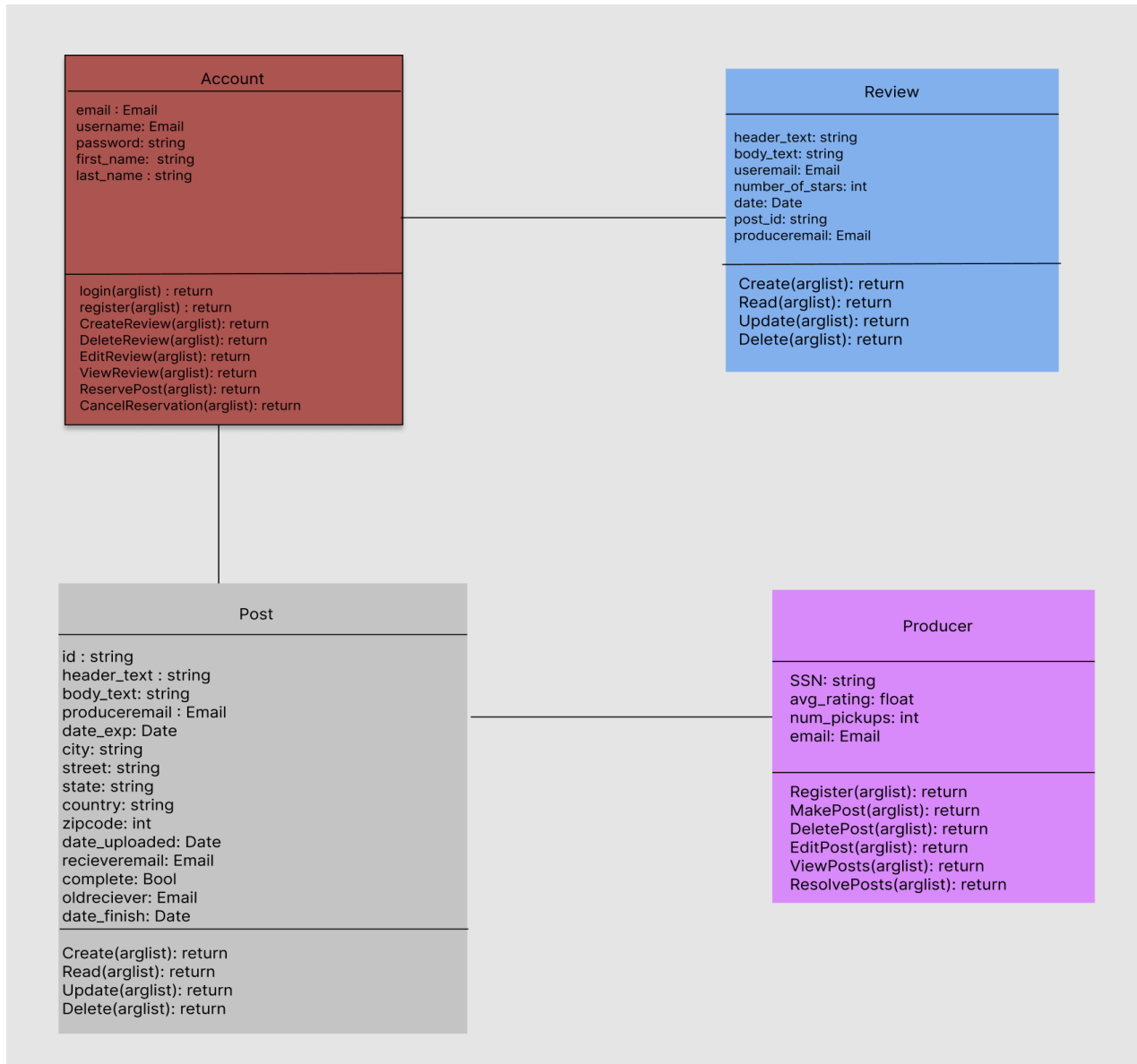
Front End: React Native with .js files acting as UI components and Screens to display different pages

Back End: Django .py files as Models and Serializers to add, delete, edit, and retrieve data stored in the database and convert it to a JSON format that is then sent when a fetch request is called (categorized as four classes: Reviews, Users, Producers, and Posts.)

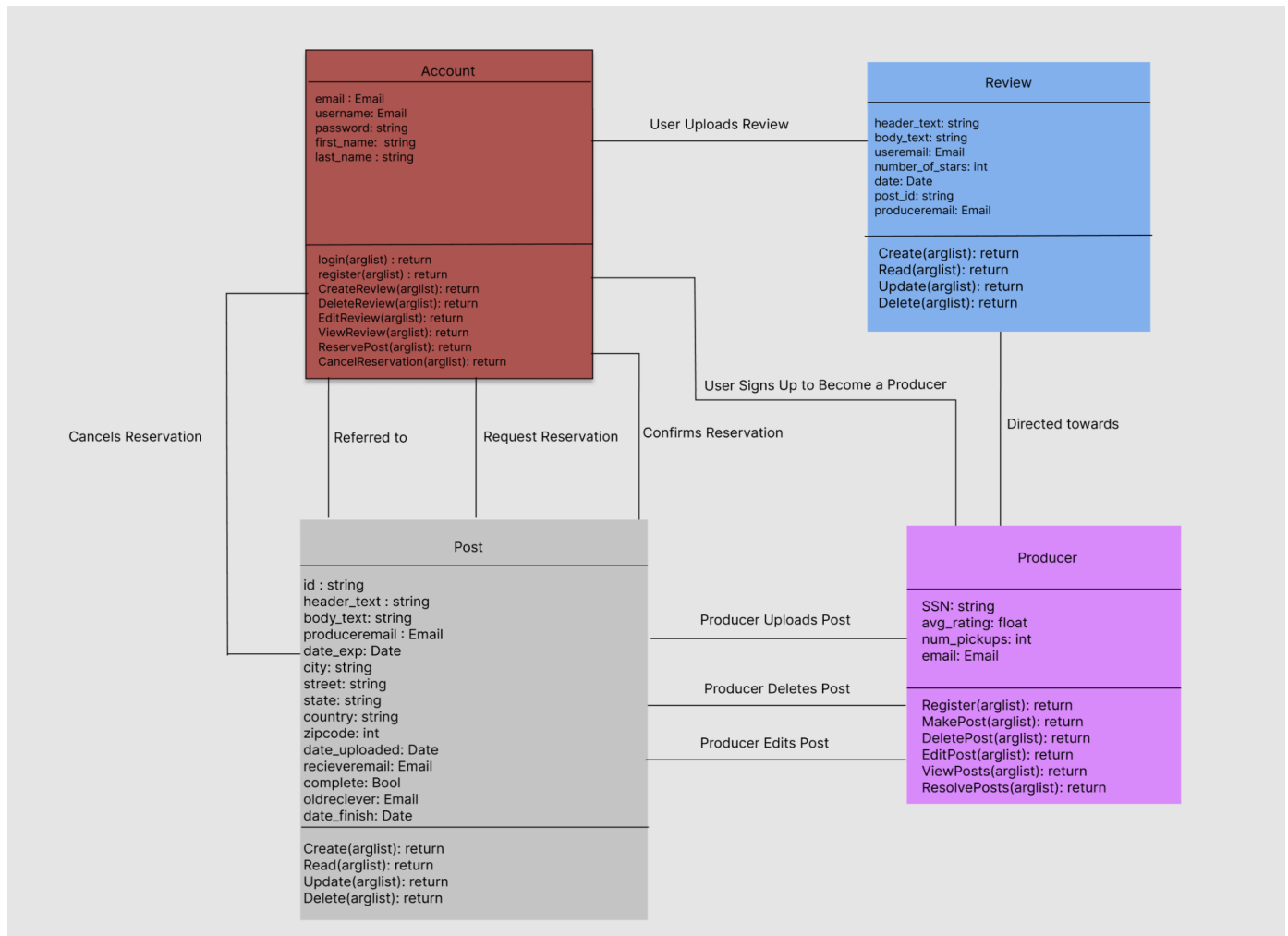
Database: PostgreSQL using pgAdmin4 management tool to keep tables of all of the data



### 4.1.1 Software Unit Detailed Design



## 4.2 Static Relationship of Software Unit



### 4.2.1 Run-time Object Instances

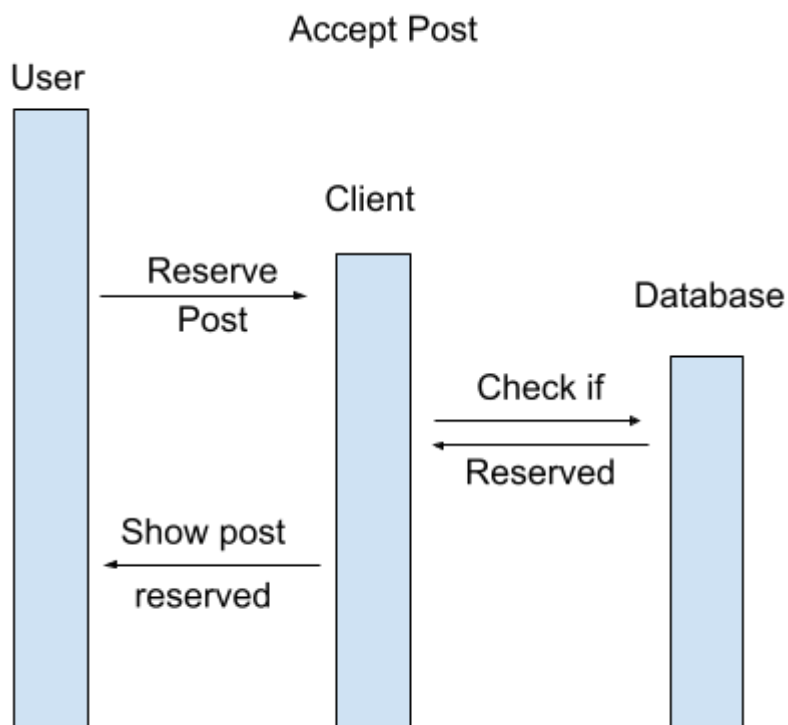
The Django Application Server gets instantiated once we start the server on a localhost. This will retrieve data automatically from the database through the IP

address. React Native will be instantiated once the project is run, in which then it will call fetch requests from and to the application server through the application url

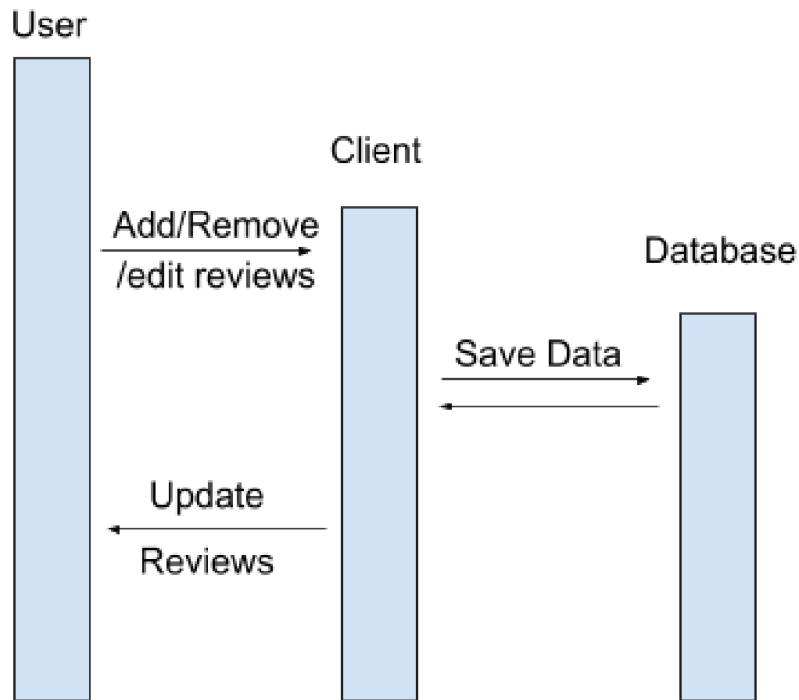
## 4.3 Behavior

Users can create posts that then notify other users of the listing and that listing will appear in the user's feed. Once a user creates a review for either a post or an account, it will get shown in the account's public review section and the score will be averaged amongst all reviews and shown in the account. Friends can be added and users can invite friends to groups, which then creates a group chat for those users. Posts are deleted after a set timeout or after a customer picks up the food. Admin can delete posts or accounts if they are flagged for being harmful.

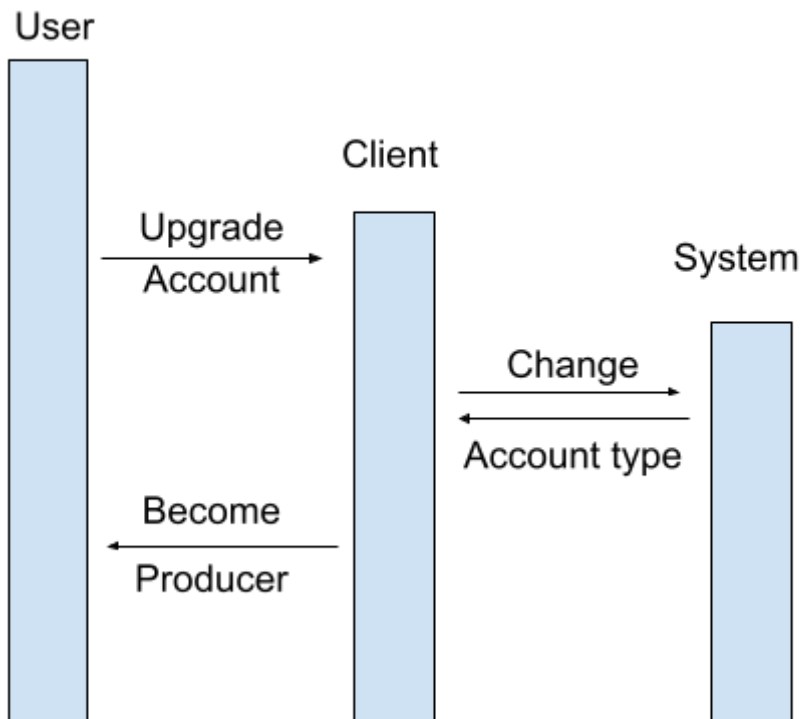
### 4.3.1 Sequence Interaction Diagrams



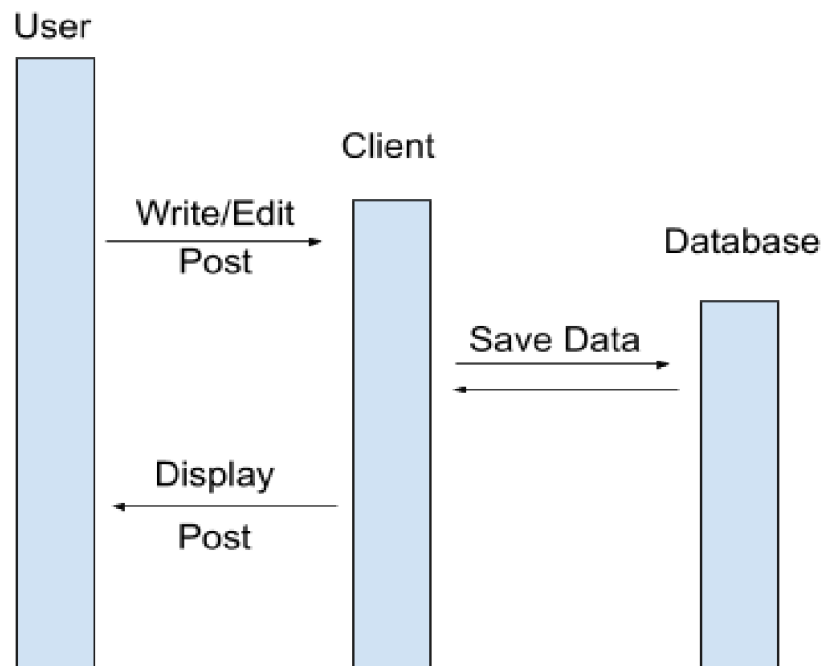
## Manage Reviews



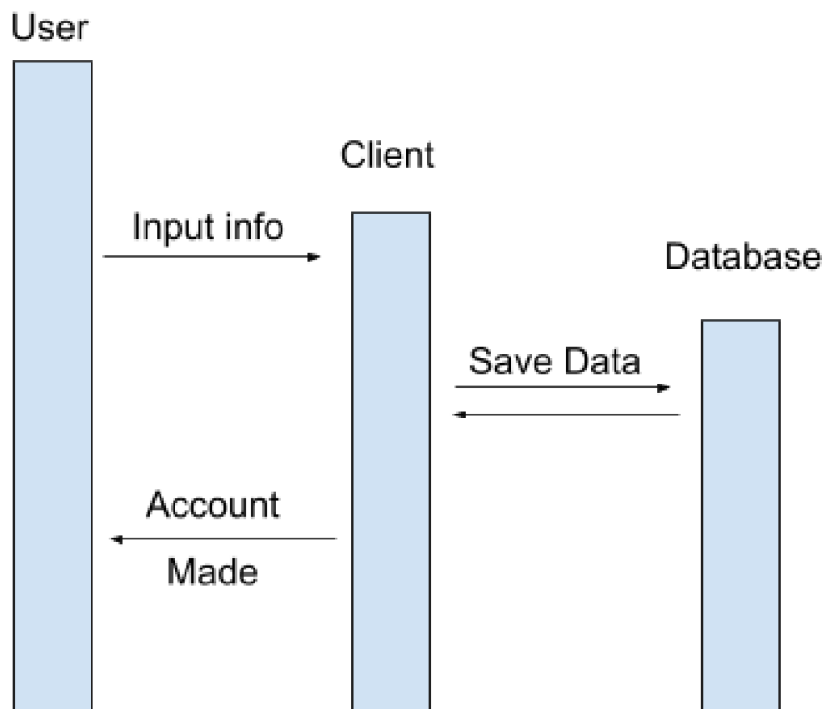
## Become Producer

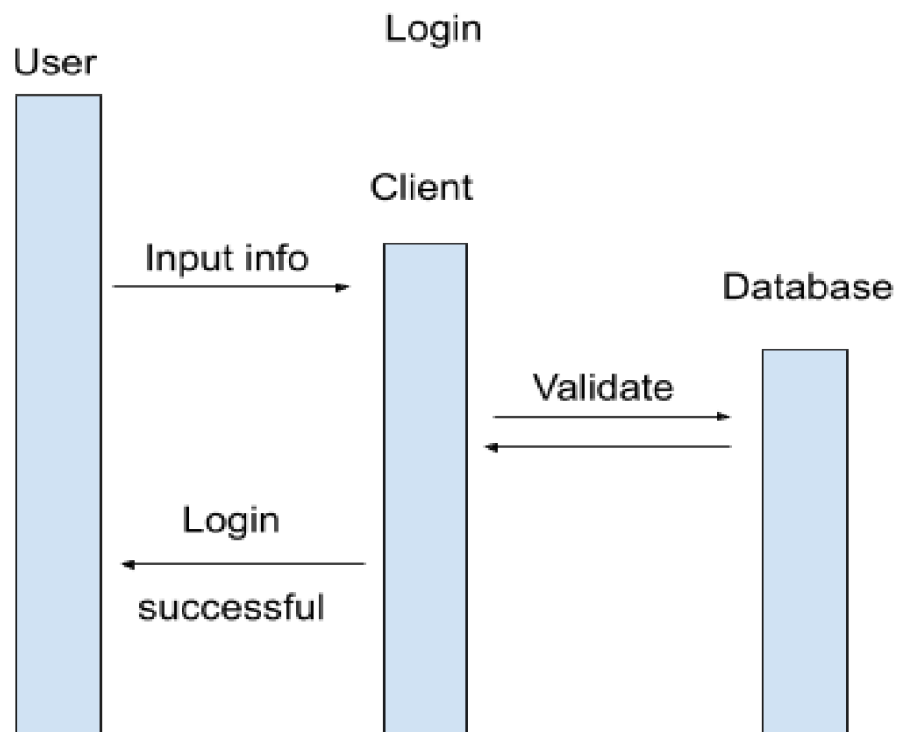


## Make/Edit Posts

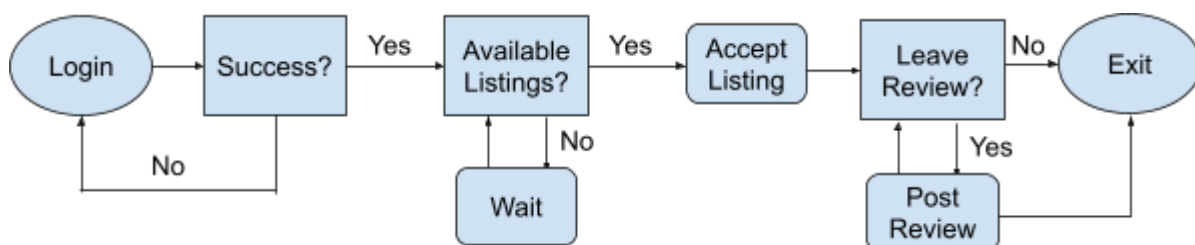


## Create Account

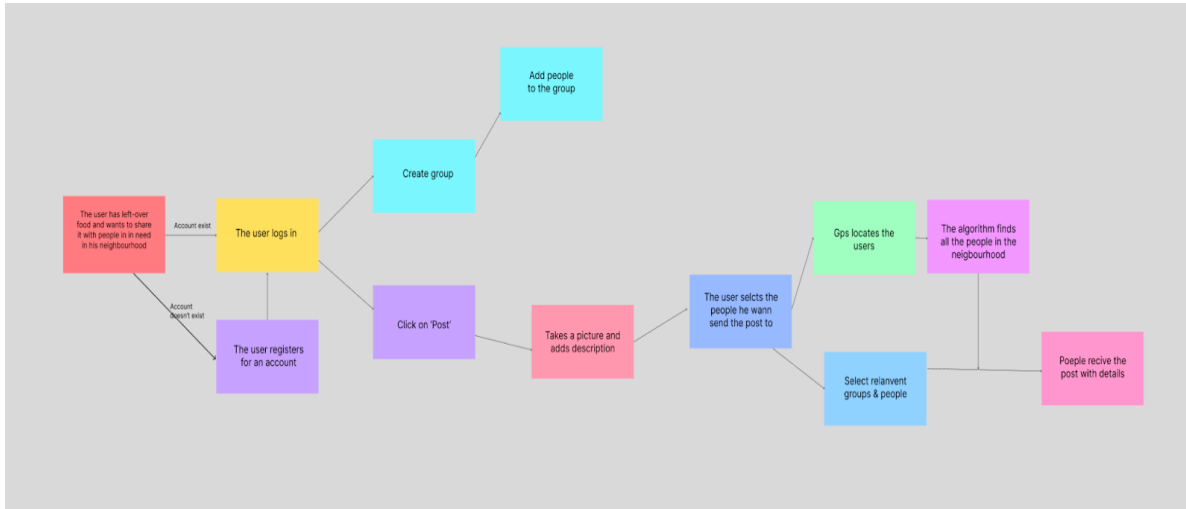




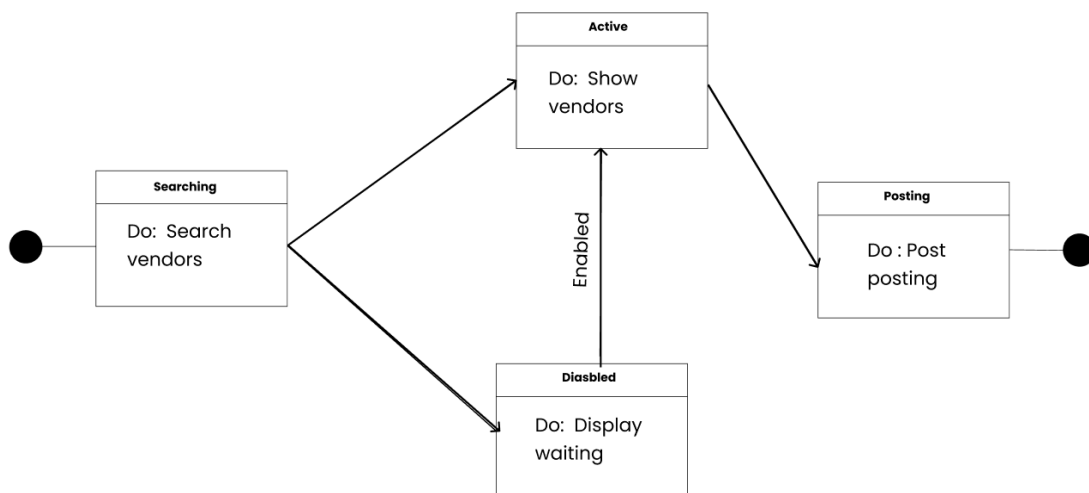
#### 4.3.2 Collaboration Diagrams



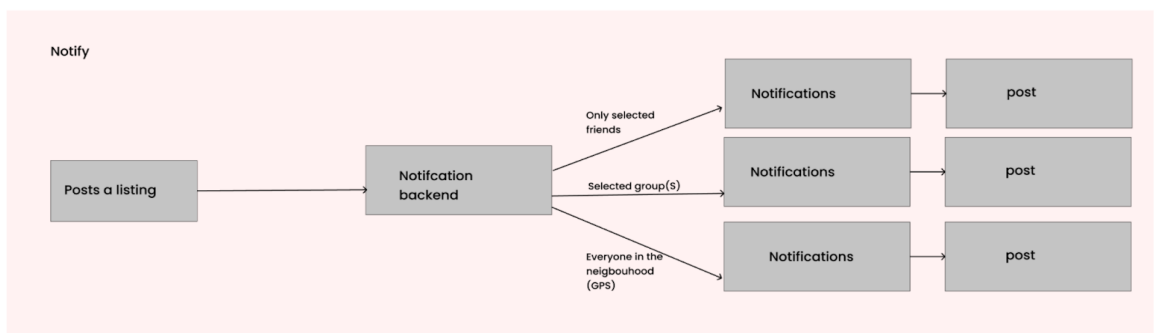
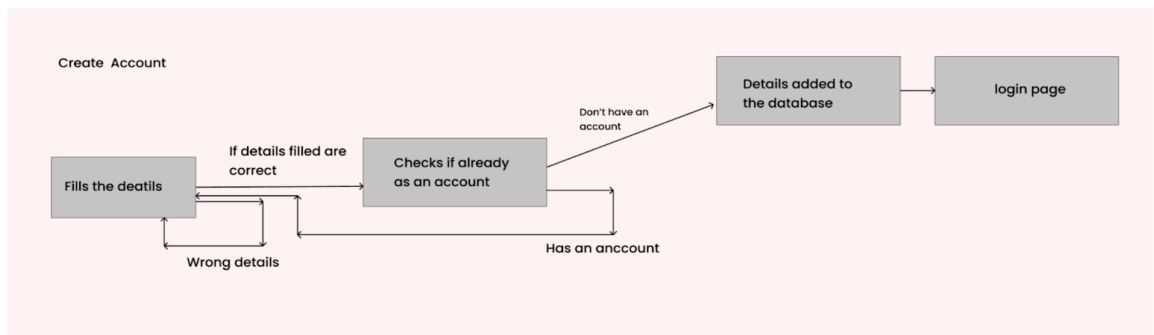
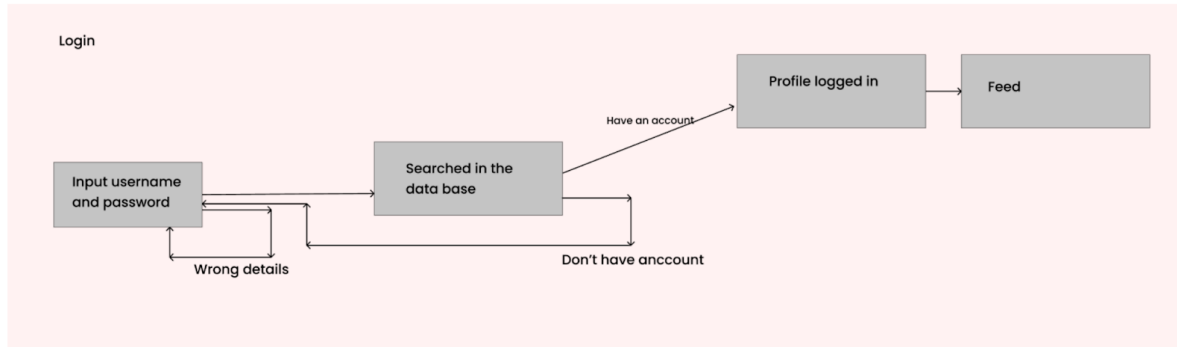
### 4.3.3 Activity Diagrams



### 4.3.4 State Diagrams

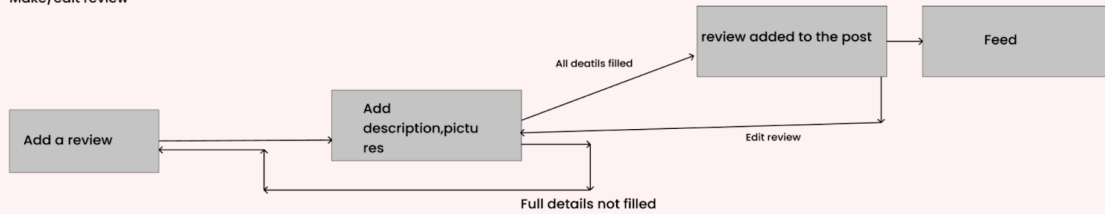


### 4.3.5 Event Diagrams

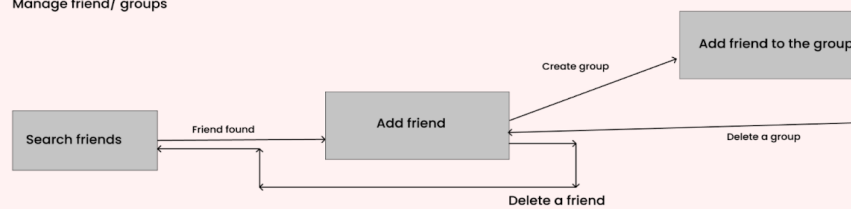




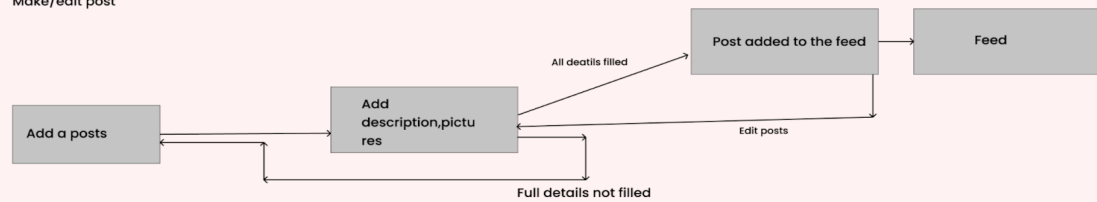
## Make/edit review



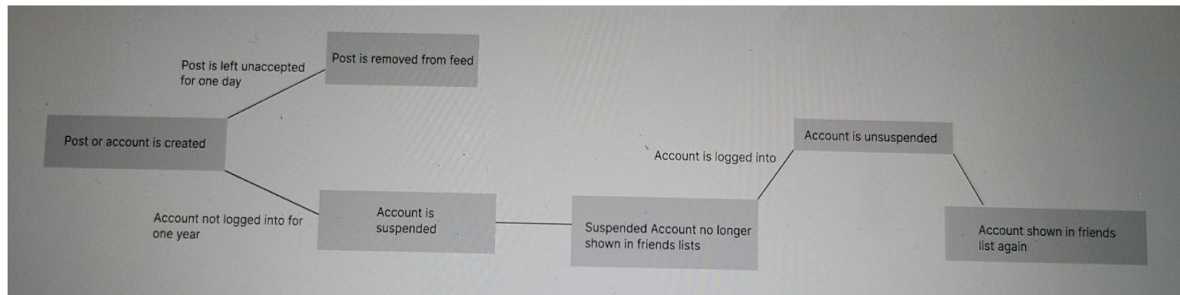
## Manage friend/ groups



## Make/edit post



## Suspension/Deletion of Account/Post



---

## 4.4 Concept of Execution

The application Django server gets launched on localhost with access to the PostgreSQL database through the IP address, React Native fetches data from the application server through the fetch method returning a Promise of the data requested. That data is then sent to be visualized on the current page.

## 4.5 Interface Design

Account Creation Page will lead to a page asking the user if they want to sign up to become a producer. If so, they will be redirected to a SSN validation page. If not, they'll be taken to the home page, which is a social media style listing page showing all of the listings that are available within that user's specific location. From the home page, users can access their settings, the listing options, the terms of service page, and their group chats. Producers can access these as well as the create listing page.

### 4.5.1 *Unique identifier of Interface*

AccountSignUp: Account Creation Page -> Producer Prompt

SSNValidation: Producer Prompt -> SSN Validation Page

Login: Login Page -> Social Media Listing Page

ToSettings: Social Media Listing Page -> Account Settings

ToTerms: Social Media Listing Page -> Terms and Service Page

ToChats: Social Media Listing Page -> Group Chats

ToChat: Group Chats -> Group Chat

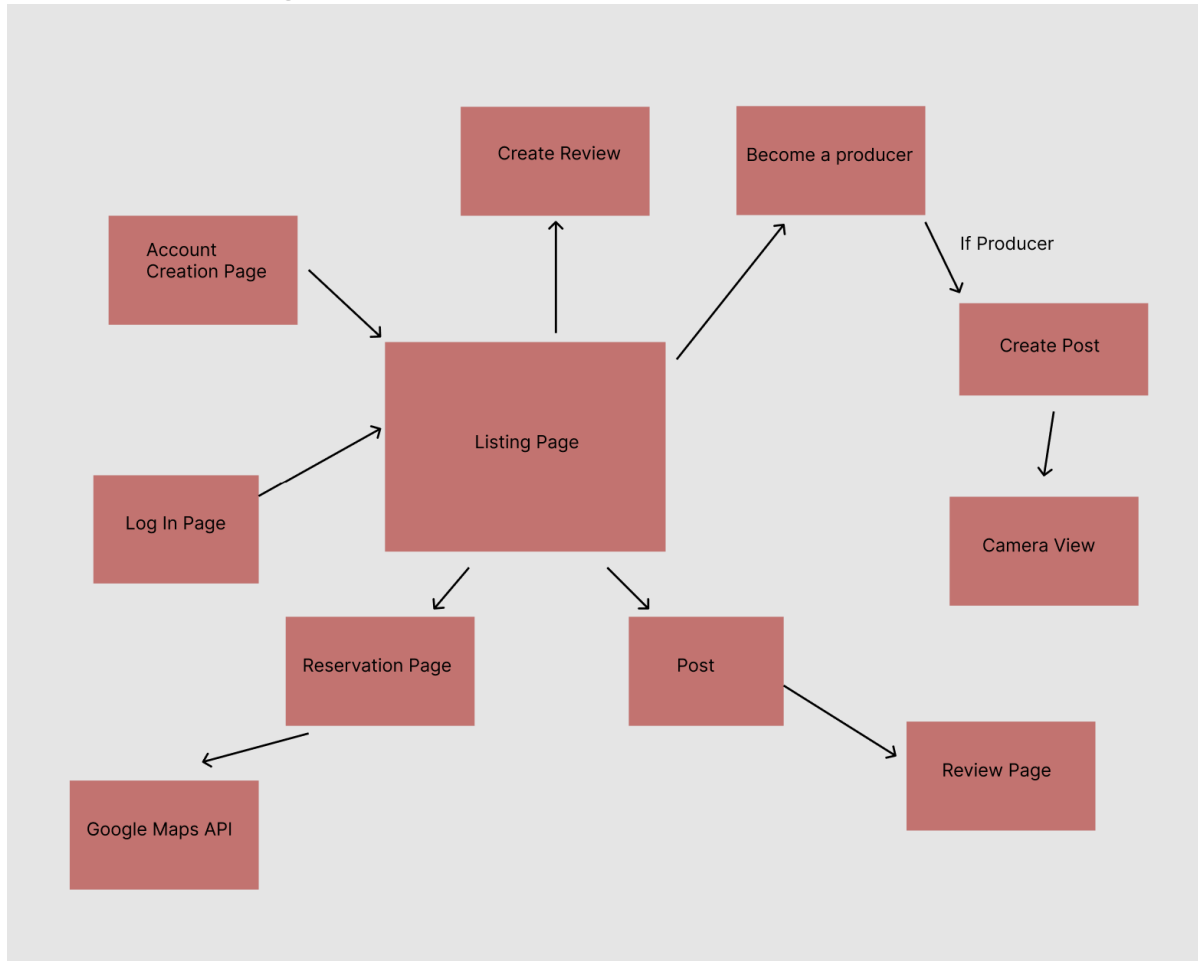
ViewListing: Social Media Listing Page -> Listing Page

AcceptListing: Listing Page -> Google Maps Popup Tracking

CreateListing: Social Media Listing Page -> Create Listing Page

ViewCamera: Create Listing Page -> Camera View

### 4.5.2 Interface Diagrams



## 5. IMPLEMENTATION ARCHITECTURE (NOT REQUIRED)

---

### 5.1 All Active and Passive Classes Assigned to Components

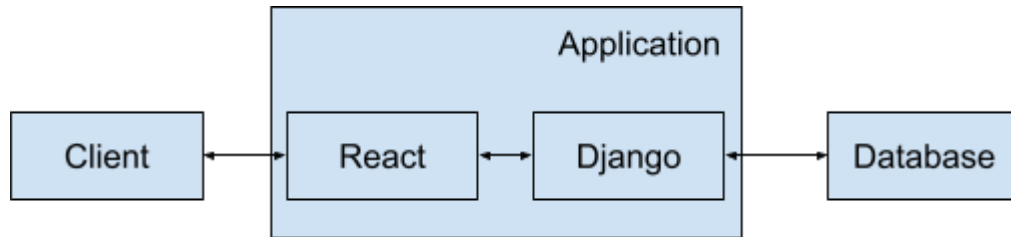
Includes all files (.CPP, Header, DLL, EXE, etc.) and middleware

### 5.2 Diagrams of Physical Packaging of Logical Components

## 6. DEPLOYMENT ARCHITECTURE

---

### 6.1 Physical Deployment Architecture Diagram



## 7. DICTIONARIES

---

Classes:

Name	Description	Methods	Attributes
Account	User account information	login(arglist) : return register(arglist): return Send_request(arglist) : return accept_request(arglist) : return	user_id : integer password: string full_name : string user_email : string contact_no : integer SSN : integer friends_list : list of Account objects rating_percentage : integer is_producer : bool group_list : list of Group objects
Group	Group of friends that the user creates that the user can send notifications to instead of the general public	add_member(arglist) : return delete_member(arglist) : return	group_list : list of Account objects names_list : list of strings
Post	Publication of a listing made by a producer	summary(arglist) : return cancel_reservation(arglist) : return	user_id : integer full_name : string user_email : string contact_no : integer Rating_percentage : integer pickup_time : integer pickup_place : string Reserved : bool

Customer near address	A user within the same vicinity as the public listing is notified, customer can request the listing	make_reservation(arglist) : return cancel_reservation(arglist) : return	user_id : integer full_name : string user_email : string contact_no : integer has_reservation : bool
-----------------------	---	--	---

Admin	Administrative office that has access to every Account object, validates SSN to promote them to supplier	validate_ssn(arglist) : return	user_id : integer password: string full_name : string user_email : string SSN : integer contact_no : integer friends_list : list of Account objects Rating_percentage : integer
-------	--	-----------------------------------	--

## Methods:

Name	Description	Class	Arguments
------	-------------	-------	-----------

login()	Validates the user using email and password and gives the user access to their personal data	Account	User inputted email User inputted password
---------	--	---------	---

register()	Creates a new Account object	Account	User inputted email User inputted password
------------	------------------------------	---------	---

Send_request()	Sends a friend request to another user	Account	Account object friends_list
----------------	--	---------	--------------------------------

Add_member()	Adds a member to the group only if that member is in the friends list	Group	Account object Friends_list
--------------	---	-------	--------------------------------

delete_member()	Deletes a member from the group	Group	Account object
-----------------	---------------------------------	-------	----------------

validate_ssn()	Validates the users SSN	Admin	SSN user_id Full_name is_producer
----------------	-------------------------	-------	--

Summary()	Presents a summary of the listing information to be printed onto the listing	Post	Full_name Contact_no User_email Rating_percentage Pickup_time Pickup_place
-----------	--	------	---

make_reservation()	Accepts listing made by producer	Customer near address	User_id User_email Full_name Contact_no
--------------------	----------------------------------	-----------------------	--

cancel_reservation()	Producer or user cancels reservation	Customer near address	Reserved User_id user_email
----------------------	--------------------------------------	-----------------------	-----------------------------------

Accept_request()	Accepts or declines a friend request sent by another user	Account	Whether or not the user accepts
------------------	---	---------	---------------------------------

## Attributes:

Name	Description	Type	Complexity
------	-------------	------	------------

user_id	User identification number	Integer	Simple
password	User login identification sequence of characters	String	Simple
full_name	Name that user wants other users to call them by	String	Simple
user_email	User contact and log in identification email address	String	Simple
Contact_no	User phone number for contact and log in identification	Integer	Simple
SSN	User's social security number used to hold the user liable for wrongdoing	Integer	Simple
Friends_list	List of users that this user has either sent or accepted a friend request	List of Account Objects	Complex
rating_percentage	The percentage number calculated through reviews made on the user's account	Integer	Simple
Is_producer	Whether or not this user is certified as a producer	Bool	Simple



Group_list	List of users who are members of the same group	List of Group objects	Complex
------------	---	-----------------------	---------

Names_list	List of people's names in the Group	List of strings	Complex
------------	-------------------------------------	-----------------	---------

Pickup_place	Address that the listing is located	String	Simple
--------------	-------------------------------------	--------	--------

Pickup_time	Time that the listing will expire by	Integer	Simple
-------------	--------------------------------------	---------	--------

Reserved	Whether or not this listing is reserved	Bool	Simple
----------	---	------	--------

Has_reservation	Whether or not the user has the reservation to pick up the food	Bool	Simple
-----------------	---	------	--------

## Relationships:

Name	Description	From Class	To Class
------	-------------	------------	----------

Accessing information	Admin has access to every Account object's information	Account	Admin
-----------------------	--	---------	-------

Posting Listings	Producers can post listings	Account	Post
------------------	-----------------------------	---------	------

Refer public	Notifies those within the same area of a new listing	Post	Customer near address
--------------	--	------	-----------------------

Refer group	Notifies those within the same group of a new listing	Post	Group
-------------	---	------	-------

Creating Group	User creates a group of friends	Account	Group
----------------	---------------------------------	---------	-------

Public requests	User requests reservation	Customer near address	Post
-----------------	---------------------------	-----------------------	------

Group requests	Group member requests reservation	Group	Post
----------------	-----------------------------------	-------	------

Confirms public	Producer confirms reservation to user	Post	Customer near address
-----------------	---------------------------------------	------	-----------------------

Confirms group	Producer confirms reservation to group member	Post	Group
----------------	---	------	-------

## Key Events:

Name	Description	Motive	Action	Pre-Conditions	Post-Conditions	State Change
------	-------------	--------	--------	----------------	-----------------	--------------

User becomes producer	A user submits their SSN for verification in order to become a producer	User wants to start posting listings	User submits SSN	SSN field is empty Is_producer is false	SSN field is full Is_producer is true	User becomes a producer
-----------------------	---	--------------------------------------	------------------	--	--	-------------------------

User joins group	A user is added to a group by the user who created it	Friend wants to join exclusive rights to notifications	User adds a friend to the group	Group_list initial Names_list initial	Account object is added to the group_list Friend's name added to the names_list	Friend joins group
------------------	---	--	---------------------------------	--	--	--------------------

Member leaves group	A user is deleted from the group	User does not want to have exclusive rights to notifications anymore	User deletes either themselves or another user from the group	Group_list initial Names_list initial	Account object is removed from the group_list Friend's name removed from the names_list	User leaves group
---------------------	----------------------------------	--	---	--	--	-------------------

Customer gets listing reservation	A user requests reservation	User wants to acquire listing contents	User requests the listing	Has reservation is false Reserved is false	Has reservation is true Reserved is true	User acquires reservation
-----------------------------------	-----------------------------	--	---------------------------	---	---	---------------------------

Customer loses reservation A user cancels reservation	A user cancels reservation or a producer cancels the listing	User does not want to acquire listing contents anymore	User cancels reservation Producer cancels listing	Has reservation is true Reserved is true	Has reservation is false Reserved is false	User loses reservation
--	--	--	--	---	---	------------------------

**SOFTWARE ENGINEERING STANDARD****SOFTWARE DESIGN DESCRIPTION  
SDD- 001**

Timer cancels reservation	The window that the user who has the reservation has for getting to the location is over and the listing becomes open again	Since user probably won't be coming to get the food another user can acquire it	Timer cancels the reservation	Has reservation is true Reserved is true	Has reservation is false Reserved is false	User loses reservation
---------------------------	---	---	-------------------------------	--	--	------------------------

---

## 8. SOFTWARE ITEM COMPUTER RESOURCE UTILIZATION

---

CPU: cannot 40% maximum

Memory: RAM/Disk around 30% minimum, cannot exceed 80%

---

## 9. REQUIREMENTS TRACEABILITY

---

### 9.1 Software Component-Level Requirements Traceability

Requirement:

13.3.1:

Forward: Requirements 13.3.6, 13.3.12

Backward: Requirements 13.3.2, 13.3.3, 13.3.7, 13.3.8, 13.3.9,  
13.3.11

13.3.2:

Forward: Requirements 13.3.1, 13.3.4

Backward: Requirements 13.3.5

13.3.3:

Forward: Requirements 13.3.9, 13.3.1

Backward: Requirements 13.3.1, 13.3.4

13.3.4:

Forward: Requirements 13.3.6, 13.3.12

Backward: Requirements 13.3.6, 13.3.12

13.3.5:

Forward: Requirements 13.3.6

Backward: Requirements 13.3.1, 13.3.2, 13.3.3

13.3.6:

Forward: Requirements 13.3.9, 13.3.7

Backward: Requirements 13.3.4, 13.3.5

13.3.7:

Forward: Requirements 13.3.9

Backward: Requirements 13.3.6

13.3.8:

Forward: Requirements 13.3.9,

Backward: Requirements 13.3.7

13.3.9:

Forward: Requirements 13.3.10, 13.3.11, 13.3.12, 13.3.13

Backward: Requirements 13.3.8, 13.3.7

13.3.10:

Forward: Requirements 13.3.1, 13.3.2, 13.3.9, 13.3.12, 13.3.13

Backward: Requirements 13.3.3, 13.3.7, 13.3.8, 13.3.11

13.3.11:

Forward: Requirements 13.3.9, 13.3.10

13.3.12:

Forward: Requirements 13.3.1, 13.3.7

Backward: Requirements 13.3.10

13.3.13:

Forward: Requirements 13.3.1

Backward: Requirements 13.3.7

13.3.14:

Forward: Requirements 13.3.16, 13.3.17

Backward: Requirements 13.3.15, 13.3.18

13.3.15:

Forward: Requirements 13.3.14, 13.3.16, 13.3.17

Backward: Requirements 13.3.18

13.3.16:

Forward: Requirements 13.3.17

Backward: Requirements 13.3.15

13.3.17:

Backward: Requirements 13.3.16

13.3.18:

Forward: Requirements 13.3.14

## 10. SYSTEM DESIGN TESTING

---

### Software Quality Assurance Testing Process:

1. If the app launches properly with all the features and functionality
2. If the user can register/login into the app
3. If the user can see the feed of all the pickups available
4. If multiple users can access the app simultaneously
5. If the user interface is seamless
6. If all the content on the app are properly aligned and well managed

### Product Testing:

1. Test out the GPS by simulating an order
2. If the system sends out local notifications once an order is posted
3. Check if the user is able to set up profiles with their

- names and any additional information they want to include in a text box
4. Simulate friend request to see if it works
  5. If the system allows the users to create multiple groups and add friends to those groups
  6. Simulate and check if the system allows the users to post the order to only selective groups or friends
  7. If the users are able to post by checking their verification status
  8. If the system can only allow the post to be published if all the information is filled
  9. If the application can access the camera and can take pictures and upload them to the post.
  10. If the system only allows to upload photos/videos that are under certain reserved storage
  11. If the system allow to post one listing at a time
  12. If the user able to delete or edit their listings after publication
  13. Once the user picks up the food, is the user able to give a review to the user who posted it.
  14. If the user able to submit a complaint about the order
  15. If the app is directly users to the complaint box if he gives a 1 star review
  16. If the post gets deleted if the account is suspended or deleted by the developers.
  17. Check if the app has all the required content.

**Acceptance Testing:**

1. This testing is conducted as part of customer sign off on the finished software
2. If the user is satisfied after using the app or in other words user does not find it difficult to use the application
3. If the user is able to use it multiple times in a day
4. If the user likes the features and functionalities of the application

**11. RATIONALE**

---

**12. NOTES**

---

## **13. APPENDICES**

---

### **13.1 Dictionaries**

The dictionaries are included in section 7

### **13.2 UML diagrams**

Included in section 4.2



---

## 13.3 Requirements diagrams

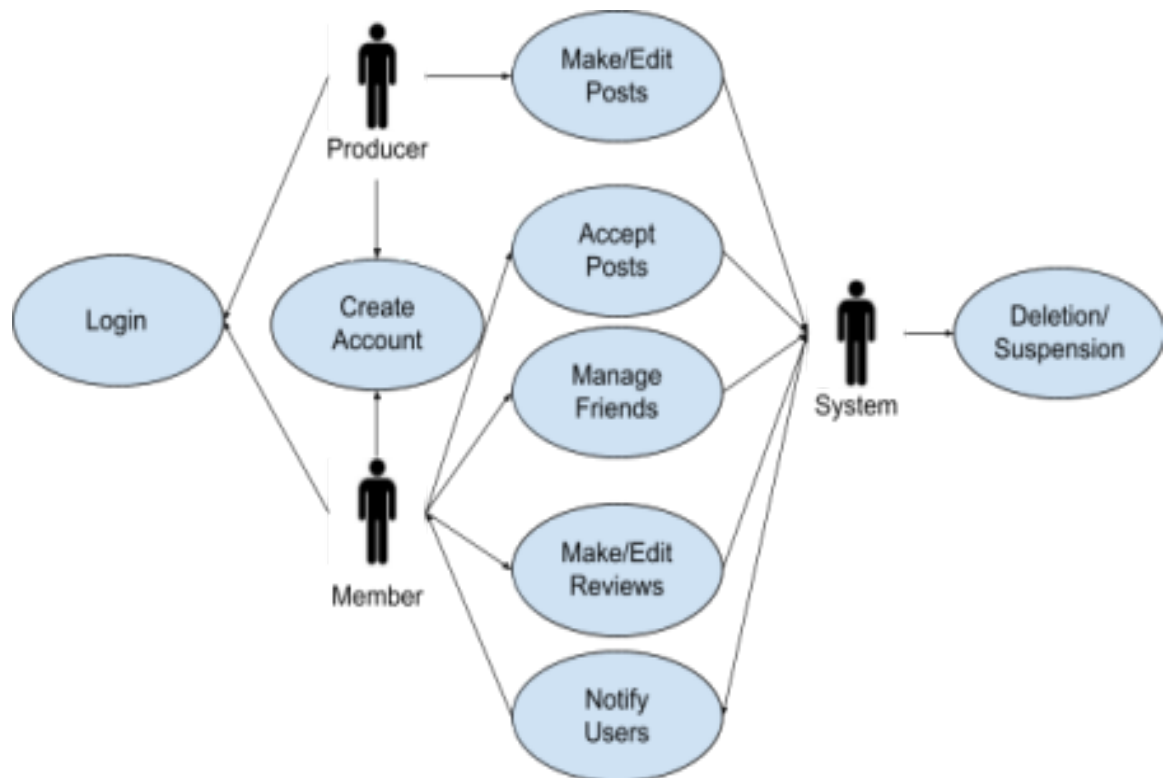
Functional Requirement (at least all the requirement definitions)

1. The system shall send out local notifications to devices within NYC boundaries
2. The system shall allow for access to the GPS location of a users device in order to send out local alerts to every user in the area (currently within NYC is the locality for all posts)
3. The system shall allow users to set up profiles with their name and any additional information they want to include in a text box
4. The system shall allow users to send friend requests and accept friend requests from other users
5. The system shall allow users to create multiple groups consisting of some or all of the user's friends
6. The system shall allow users the option on each post before publication to only notify selected groups or group
7. The system shall only allow users to be able to publish posts if they are registered as a producer and their SSN has been registered and verified
8. The system shall interact with the Social Security Number Verification Service (SSNVS) in order to check a users SSN verification
9. The system shall check that photos, an address, a video, a recipe, and allergy information are all included on each individual post before it is allowed to be published
10. The system shall have interactions with the devices camera system in order to upload pictures or videos and a timer system to track the listing take down reupload schedule
11. The system shall have reserved storage for file uploads
12. The system shall only allow one user to reserve a listing at a time
  - a. The system shall take down the listing temporarily and reupload the listing after a designated amount of time if the producer has not marked it as complete
  - b. The system shall remove the user who had reserved the order from the reservation list upon timer completion
13. The system shall allow producers to delete or edit their listings after publication
14. The system shall allow users to leave public reviews

available to every user on a producer's account with a star rating system out of 5 and the ability for the user to publish text along with the star rating

15. The system shall include a complaint box page available to all users
  - a. The page is a text box where you can send an email to the customer support team
  - b. Other text boxes include the account the user wants to complain about and the order or orders they wish to complain about
16. The system shall direct users who leave a 1 star review to the complaint box page
17. The system shall allow for a post to be deleted by and for an account to be suspended or permanently deleted by the developers
18. The system shall include a terms of service page available to every user detailing the rules, regulations and conduct standards that every producer must abide by

#### Use Case Diagrams and Use Case Descriptions



Log in		
<b>Description</b>	The user/producer puts in their username and password in order to access their account.	
<b>Pre- Conditions</b>	Must already have an account.	
<b>Flows</b>	Basic or Normal Flows	<ol style="list-style-type: none"> <li>1. User inputs account username/password</li> <li>2. Profile loads up</li> </ol>
	Alternative Flows	<ol style="list-style-type: none"> <li>1. Incorrect Information is put in</li> <li>2. User is prompted to retry</li> <li>3. Correct info is finally entered</li> </ol>
<b>Post Conditions</b>	Can now view profile and see if any posts are out	
<b>Special Requirements</b>	Must access database to confirm account information	
<b>Extension Points</b>	Users are taken to a main page where they can view posts, manage friends, etc	

Create Account		
<b>Description</b>	User fills in their personal information and username/password and chooses whether they are making a producer/member account. Producers have to put in extra information to ensure that they are trustworthy and don't have ill intentions.	
<b>Pre- Conditions</b>	Must have access to the application	
<b>Flows</b>	Basic or Normal Flows	<ol style="list-style-type: none"> <li>1. User chooses member account</li> <li>2. User fills out member information</li> <li>3. Information is saved to database</li> </ol>
	Alternative Flows	<ol style="list-style-type: none"> <li>1. User chooses producer account</li> <li>2. User fills out producer information</li> <li>3. Information is saved to database</li> </ol>
<b>Post Conditions</b>	If a producer account is created, they will have some extra privileges. Can now sign in.	

<b>Special Requirements</b>	Checks with Social Security Number Verification Service when creating a producer account.
<b>Extension Points</b>	Now will be sent to the login page

<b>Make/Edit Posts</b>		
<b>Description</b>	User makes a post about leftovers that they have ready, taking pictures and putting down information on the food that has been prepared	
<b>Pre- Conditions</b>	User must be in producer account and have valid information	
<b>Flows</b>	Basic or Normal Flows	<ol style="list-style-type: none"> <li>1. User prepares the food</li> <li>2. User takes picture of food</li> <li>3. User fills out information on the post</li> </ol>
	Alternative Flows	<ol style="list-style-type: none"> <li>1. User makes mistake in post</li> <li>2. User edits post</li> <li>3. User re uploads post</li> </ol>
<b>Post Conditions</b>	Member users can now see the post and react to it	
<b>Special Requirements</b>	System ensures that all requirements of the posts are met. GPS location is tracked to see where the close members are	
<b>Extension Points</b>	The post then goes out to the system which activates the notification	

<b>Accept Posts</b>	
<b>Description</b>	User can accept the post if they decide to take the leftovers from the producer/vendor
<b>Pre- Conditions</b>	Must have member account

<b>Flows</b>	Basic or Normal Flows	<ol style="list-style-type: none"> <li>1. User looking through posts</li> <li>2. User sees a post that catches his eye</li> <li>3. User hits accept and gets their food</li> </ol>
	Alternative Flows	<ol style="list-style-type: none"> <li>1. User accepts a post and changes their mind</li> <li>2. User revokes the claim</li> <li>3. Post goes back out into the system</li> </ol>
<b>Post Conditions</b>	Post disappears so other members can't accept it	
<b>Special Requirements</b>	Activates GPS to see where the food is located	
<b>Extension Points</b>	Post is then removed from the system and later user is prompted to make a review	

Manage Friends		
<b>Description</b>	User can add other users as friends as well as delete and can create group chats with them	
<b>Pre- Conditions</b>	Must have account already	
<b>Flows</b>	Basic or Normal Flows	<ol style="list-style-type: none"> <li>1. User looks up another user's username</li> <li>2. Checks to confirm profile is the one they want to add</li> <li>3. Add person as friend and they can decide whether or not to accept</li> </ol>
	Alternative Flows	<ol style="list-style-type: none"> <li>1. Can highlight various friends from friends list</li> <li>2. Pick the group option</li> <li>3. New group is created with all the highlighted friends</li> </ol>
<b>Post Conditions</b>	Either have a new friend, new friend group, or less friends.	
<b>Special Requirements</b>	Database must be accessed and new friends list/group must be updated	

<b>Extension Points</b>	Updated friends list is shown in the system and the user has the option to make changes again
-------------------------	---

<b>Make/Edit Reviews</b>		
<b>Description</b>	After accepting a post from any producer, the user has the option to either write a good or bad review describing the experience as well as the food quality	
<b>Pre- Conditions</b>	Must be a member and have accepted a post	
<b>Flows</b>	Basic or Normal Flows	<ol style="list-style-type: none"> <li>1. User finishes food and enjoys it</li> <li>2. User writes a good review for the producer</li> <li>3. Future users get to see the review and producer has a higher chance of getting posts accepted</li> </ol>
	Alternative Flows	<ol style="list-style-type: none"> <li>1. User finishes food and dislikes it</li> <li>2. User write a very negative review for the producer</li> <li>3. If review was bad they are also given the option to file a complaint</li> </ol>
<b>Post Conditions</b>	Producer's profile will be updated to show the average rating since their last post	
<b>Special Requirements</b>	If a complaint is filed, the contents of the complaint get emailed to a support team that will handle their case	
<b>Extension Points</b>	After making the review, get sent back to the main menu where you have the option to accept posts, manage friends, etc.	

Notify Users		
<b>Description</b>	Once a post is made, the system immediately sends out a notification to all users in the area	
<b>Pre- Conditions</b>	A post had to have been made in order for the notification to appear	
<b>Flows</b>	Basic or Normal Flows	<ol style="list-style-type: none"> <li>1. Producer makes a post</li> <li>2. System notices and checks the gps location</li> <li>3. All member users in the general location get notified</li> </ol>
	Alternative Flows	<ol style="list-style-type: none"> <li>1. Producer makes a post, but it was canceled</li> <li>2. System notifies, but then the notification says canceled</li> <li>3. Users see that it's canceled and look for other posts</li> </ol>
<b>Post Conditions</b>	Notification disappears quickly as well as the post if it was canceled. Otherwise, notification appears to everyone in the area	
<b>Special Requirements</b>	All members in the area are affected	
<b>Extension Points</b>	New post appears on the main page when application opens	

Deletion/Suspension		
<b>Description</b>	Once a post is left untouched for long enough, it will get automatically deleted by the system. The same goes for accounts, except they first get suspended and would require the user to put in their information to ensure nothing changed	
<b>Pre- Conditions</b>	Must have account/post that has been out for too long (1 year for account, 1 day for post)	
<b>Flows</b>	Basic or Normal Flows	<ol style="list-style-type: none"> <li>1. Post left unaccepted for 1 day</li> <li>2. System automatically deletes the post</li> <li>3. Post is no longer shown on users' home pages</li> </ol>
	Alternative Flows	<ol style="list-style-type: none"> <li>1. Account not logged into for at least a year</li> <li>2. System automatically suspends it</li> </ol>

		3. User no longer shown on friends lists or has privileges
<b>Post Conditions</b>	Post will no longer be shown and account will no longer be useable until user verifies all the information on the profile if the user ever comes back	
<b>Special Requirements</b>	If need be, the post/account could also be deleted directly by developers	
<b>Extension Points</b>	Once the account is suspended (from inactivity), user is taken back to sign in page to verify information if they come back to use the application	



## 13.4 Schedule Tracking

Time (hours)

Artifact or Deliverable	Who (individual and team)	Estimated	Actual	Difference
SPMP	Nathan Cantu	5 hours	6 hours	1 hours
	Jonathan Xu	4 hours	6 hours	2 hours
	Vijay Kallem	4 hours	4 hours	0 hour
	Summary for entire team	13 hours	16 hours	3 hours

Artifact or Deliverable	Who (individual and team)	Estimated	Actual	Difference
SRS - Final	Nathan Cantu	6 hours	10 hours	4 hour
	Jonathan Xu	5 hours	8 hours	3 hours
	Vijay Kallem	5 hours	6 hours	1 hour
	Summary for entire team	16 hours	24 hours	8 hours

Artifact or Deliverable	Who (individual and team)	Estimated	Actual	Difference
SDD - Initial	Nathan Cantu	8 hours	10 hours	2 hour
	Jonathan Xu	5 hours	7 hours	2 hours
	Vijay Kallem	4 hours	5 hours	1 hour
	Summary for entire team	17 hours	22 hours	5 hours

Artifact or Deliverable	Who (individual and team)	Estimated	Actual	Difference
SDD Final	Nathan Cantu	2 hours	1 hours	1 hour

	Jonathan Xu	1 hours	30 minutes	30 minutes
	Vijay Kallem	1 hours	30 minutes	30 minutes
	Summary for entire team	4 hours	2 hours	2 hours

**Cumulative**

Who (individual and Team)	Estimated	Actual	Difference
Nathan Cantu	21 hours	27 hours	6 hour
Jonathan Xu	15 hours	21 hours 30 minutes	6 hours 30 minutes
Vijay Kallem	14 hours	15 hours 30 minutes	1 hour 30 minutes
Summary for entire team	50 hours	64 hours	14 hours

## 13.4 Defect Tracking

### Defect Counts

Artifact or Deliverable	Who (individual and team)	Estimated	Actual	Difference
SPMP	Nathan Cantu	3 per page	5 per page	2 per page
	Jonathan Xu	3 per page	4 per page	1 per page
	Vijay Kallem	5 per page	5 per page	0 per page
	Summary for entire team	11 per page	14 per page	3 per page

Artifact or Deliverable	Who (individual and team)	Estimated	Actual	Difference
SRS - Final	Nathan Cantu	5 per page	3 per page	2 per page
	Jonathan Xu	4 per page	3 per page	1 per page
	Vijay Kallem	3 per page	8 per page	5 per page
	Summary for entire team	12 per page	14 per page	8 per page

Artifact or Deliverable	Who (individual and team)	Estimated	Actual	Difference
SDD - Initial	Nathan Cantu	4 per page	2 per page	2 per page
	Jonathan Xu	3 per page	4 per page	1 per page
	Vijay Kallem	4 per page	7 per page	3 per page
	Summary for entire team	11 per page	13 per page	2 per page

Artifact or Deliverable	Who (individual and team)	Estimated	Actual	Difference
SDD - Final	Nathan Cantu	2 per page	1 per page	1 per page
	Jonathan Xu	1 per page	2 per page	1 per page
	Vijay Kallem	2 per page	3 per page	1 per page
	Summary for entire team	5 per page	6 per page	1 per page

**Cumulative**

Who (individual and team)	Estimated	Actual	Difference
Nathan Cantu	14 per page	11 per page	3 per page
Jonathan Xu	11 per page	13 per page	2 per page
Vijay Kallem	36 per page	23 per page	13 per page

Team summary	61 per page	47 per page	18 per page

## 13.5 Project Schedule

[illegible]

Number	Task Name	Assigned to	Start	Finish	Duration	Completion	Resource Names	30	2	4	6	8	10	12	14	16	18	20	22	24	26	28	30	1	3	5	7	9	11	13	15	17	19	21	23
4.1	Discussed Potential Changes	Nathan Cantu, Vijay Kallem, Jonathan Xu	Sat 10/1/22	Sat 10/1/22	1 day	100%	Discord																												
4.2	Group Edited the SPMP	Nathan Cantu, Vijay Kallem, Jonathan Xu	Thu 10/6/22	Tue 10/11/22	4 days	100%	Google Docs																												
4.3	Submitted SPMP	Nathan Cantu	Tue 10/11/22	Tue 10/11/22	1 day	100%	NYU Brightspace																												
5	Started Downloading Technologies for Our Tech Stack	Nathan Cantu, Vijay Kallem, Jonathan Xu	Sun 10/2/22	Sun 10/2/22	1 day	100%	Google, Visual Studio																												
TA3	TA meeting	Nathan Cantu, Vijay Kallem, Jonathan Xu	Fri 10/7/22	Fri 10/7/22	1 day	100%	Zoom																												
TA3.1	Discussed Project Status	Nathan Cantu, Vijay Kallem, Jonathan Xu	Fri 10/7/22	Fri 10/7/22	1 day	100%	Zoom																												
6	Project Information/Description Form	Nathan Cantu, Vijay Kallem, Jonathan Xu	Tue 10/11/22	Wed 10/12/22	2 days	100%	Google Docs, NYU Brightspace, Discord																												
6.1	Editing the Form	Nathan Cantu, Vijay Kallem, Jonathan Xu	Tue 10/11/22	Wed 10/12/22	2 days	100%	Google Docs																												
6.2	Submitting the Form	Nathan Cantu	Wed 10/12/22	Wed 10/12/22	1 day	100%	NYU Brightspace																												
7	System Design Description	Nathan Cantu, Vijay Kallem, Jonathan Xu	Thu 10/13/22	Mon 10/17/22	3 days	100%	Visual Studio, Google Docs, NYU Brightspace, Discord																												
7.1	Starting Initial Code	Nathan Cantu, Vijay Kallem, Jonathan Xu	Thu 10/13/22	Mon 10/17/22	3 days	100%	Visual Studio																												
7.2	Group Editing the Design Description	Nathan Cantu, Vijay Kallem, Jonathan Xu	Thu 10/13/22	Mon 10/17/22	3 days	100%	Google Docs																												
7.3	Submitting the Description	Nathan Cantu	Mon 10/17/22	Mon 10/17/22	1 day	100%	NYU Brightspace																												
TA4	TA meeting	Nathan Cantu, Vijay Kallem, Jonathan Xu	Fri 10/14/22	Fri 10/14/22	1 day	100%	Zoom																												
TA4.1	Discuss Project Status	Nathan Cantu, Vijay Kallem, Jonathan Xu	Fri 10/14/22	Fri 10/14/22	1 day	100%	Zoom																												
8	Implementation	Nathan Cantu, Vijay Kallem, Jonathan Xu	Wed 10/19/22	Mon 11/28/22	29 days	0%	Visual Studio, Google, Google																												
8.1	Develop Code Base	Nathan Cantu, Vijay Kallem, Jonathan Xu	Wed 10/19/22	Mon 11/28/22	29 days	0%	Visual Studio																												

