Final Project for SW Engineering Class CSC 648-848 Section 05 Fall 2022

Team 3

The Virtual Farmers' Market

12/15/2022

http://ec2-13-52-221-26.us-west-1.compute.amazonaws.com/

Name	Role
Seth Pavlicek spavlicek@sfsu.edu	Team Leader
Alexander Bjeldanes	Database
Armando Partida	Backend
Michael Abolencia	
Angel Antunez	Github Master
Igor Tsygankov	Frontend

History

Version	Date
Milestone 1 Version 1	9/22/2022
Milestone 1 Version 2	10/7/2022
Milestone 2 Version 1	10/21/2022

Milestone 2 Version 2	10/31/2022
Milestone 3 Version 1	11/10/2022
Milestone 3 Version 2	12/1/2022
Milestone 4 Version 1	12/1/2022
Milestone 4 Version 2	12/14/2022
Milestone 5	12/15/2022

Index

Product Summary

Name: The Virtual Farmers

Priority 1:

1. Registration:

    1.1 A general user shall be able to enter their data and sign up for our services.

2. Registered Customer Account:

    2.1 A general user shall be able to set up a registered customer account.
    2.2 A registered customer shall only purchase items.

3. Registered Vendor Account:

    3.1 A general user shall be able to set up a registered vendor account.
    3.2 A registered vendor account shall only sell items.

4. Search:

    4.1 A general user or registered customer shall be able to search for items.
    4.2 A general user or registered customer shall be able to search for registered vendor accounts.

5. Filtering:

    5.1 A general user or registered customer shall be able to use filters when searching to narrow down their selection of items.

7. Purchase:

    7.1 A general user or a registered customer account shall be able to purchase items from registered vendor accounts.

9. Recurring Deliveries:

9.1 A registered customer account shall be able to create a recurring delivery in order to receive items at given time intervals.

9.2 A registered customer account shall be able to cancel their recurring delivery.

10. Revising an Order:

10.1 A general user or a registered customer shall be able to change an order within a certain time frame.

11. Contacting Vendors:

11.1 A general user or registered customer shall be able to access a registered vendor's contact information to ask about their items.

13. Nutrition Information:

13.1 A general user or a registered customer shall be able to view the nutrition information of an item.

15. Price Comparison:

15.1 A general user or a registered customer shall be able to compare the prices of the same item sold by different registered vendors.

16. Vendor Proximity:

16.1 A general user or a registered customer shall be able to search for registered vendors closest to a given address

17. Vendor Pricing:

17.1 A registered vendor shall be able to set the price for the items they are selling.

19. Previous Purchases:

19.1 A registered customer shall be able to view their purchase history.

20. Alternatives:

20.1 A general user or a registered customer shall be able to pick a different registered vendor if their first choice ran out of stock.

23. Editing Customer Information:

23.1 A registered customer shall be able to edit their information.

24. Total Price:

24.1 A general user or a registered customer shall be able to see the total price of all the items in their shopping cart before finalizing the transaction.

26. Sort:

26.1 A general user or a registered customer shall be able to sort their search results based on given criteria.

27. Tracking ID:

27.1 A registered customer shall be given a tracking ID for their orders.

33. Cart:

33.1 A general user or a registered customer shall be able to see items they added to their cart.

33.2 A general user or a registered customer shall be able to add and remove items or change the quantity of an item in their cart.

35. Delivery/Pickup:

35.1 A general user or a registered customer shall be able to pick up or have their items delivered to them.

38. Item Description:

38.1 A general user or a registered customer shall be able to read a clear description of an item.

41. Login/Logout:

41.1 A general user shall be able to login and logout of their registered customer and registered vendor accounts.

42. Editing Vendor Information:

42.1 A registered vendor shall be able to edit their business contact information.

43. Selecting a Vendor:

43.1 A general user or a registered customer shall be able to select a registered vendor that they will solely buy items from.

Unique about the product:

Our product differs from competing products in a few ways. First and foremost we strived for a product that was simple to navigate in order to prevent any clients from being overwhelmed looking for any of our services. Everything that a client would need access to in order to use our site will not take more than a few seconds of searching nor will said components be hidden in unintuitive places. Everything is clearly labeled with nothing obscuring a user's visibility. One advantage of our product is that we allow customers the option to filter out our content based on dietary restrictions and personal tastes which results in a faster, and safer time searching for the desired products hosted to our site.

Adding on to this, clients can either search by a product's name, or they can search by using an address in order to help them locate nearby vendors. These filters will always be present during the search so that customers can access them at any point during their search. All searches will be conducted using the same search bar in order to prevent any unnecessary confusion. On the other hand, we made sure that joining our site as a vendor is as simple as joining as a customer. Vendors will be able to add and sell their products on our site. All users will be able to make informed decisions about their purchases by viewing an item's description. In addition, vendor contact information will be readily accessible for our users should they wish to make inquiries about vendor stocks and products.

Link to product: http://ec2-13-52-221-26.us-west-1.compute.amazonaws.com/

Milestone 1

Index

Executive Summary

The Virtual Farmers' Market (TVFM) is an online grocery shopping service similar to other delivery services like Amazon Fresh or Instacart. However, TVFM can provide produce and other food straight from local farmers' markets. This means that a user can receive the freshest and highest quality ingredients quickly. If they have allergies or strong preferences then TVFM makes it very easy to find the food they like.

Team 3 was inspired to make this project because we all like food. However, everyone wants the highest quality food quickly. How would we solve this problem? We decided that farmers' markets have very high-quality produce. Now how would someone get it quickly? Simple, a service that would deliver it straight to the customer.

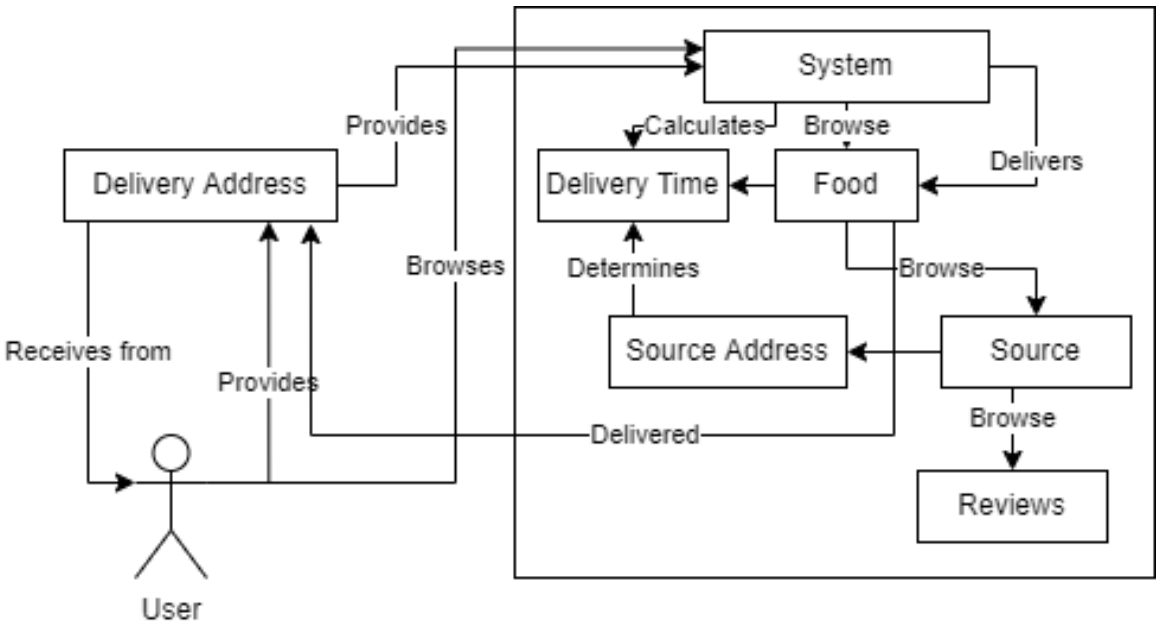
Main Use Cases

Use Case 1: Need Groceries

Actors: Eric (User)

Description: Eric is hungry and wants some groceries. However, he is very busy and cannot make it to the store to get his own groceries. He decides that he should order his food online.

Solution: Eric goes to The Virtual Farmer's Market to order his groceries. From there he can decide when and from which source he wants his food. Eric can decide if he wants a certain food from a certain source based on the price and quality from his own experience or from reviews on that source. Eric can also specify where he wants his groceries to be delivered.

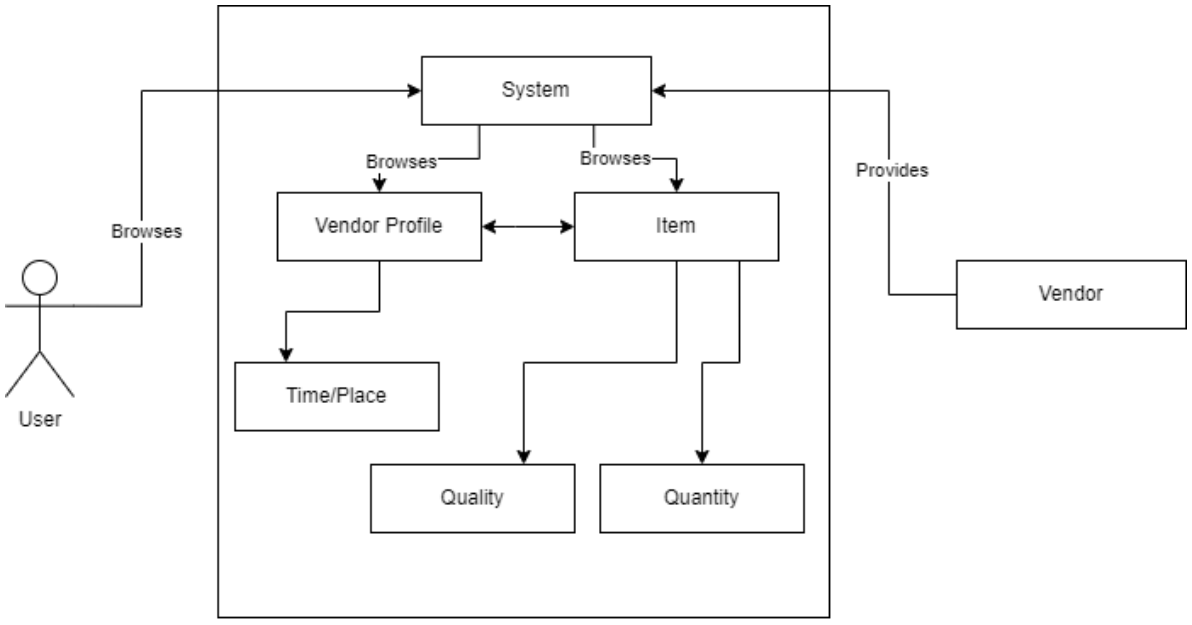


Use Case 2: Vendor Inventory

Actors: Jake (Vendor), Customer (User)

Description: Jake gets many business related messages and phone calls related to inventory at upcoming farmer's markets. Jake's current system for updating customers is by individually replying to each inquiry. Both Jake and the customer seek a more organized solution that is more convenient.

Solution: Jake goes to the Virtual Farmer's Market to update his upcoming inventory. Virtual Farmer's Market Database has a list of inventory that Jake can update and display. The customer can better track what inventory will be for sale in the upcoming farmer's market. This saves Jake and his customers significant amounts of time and energy. Jake can also specify if certain discounts will be applied to items.

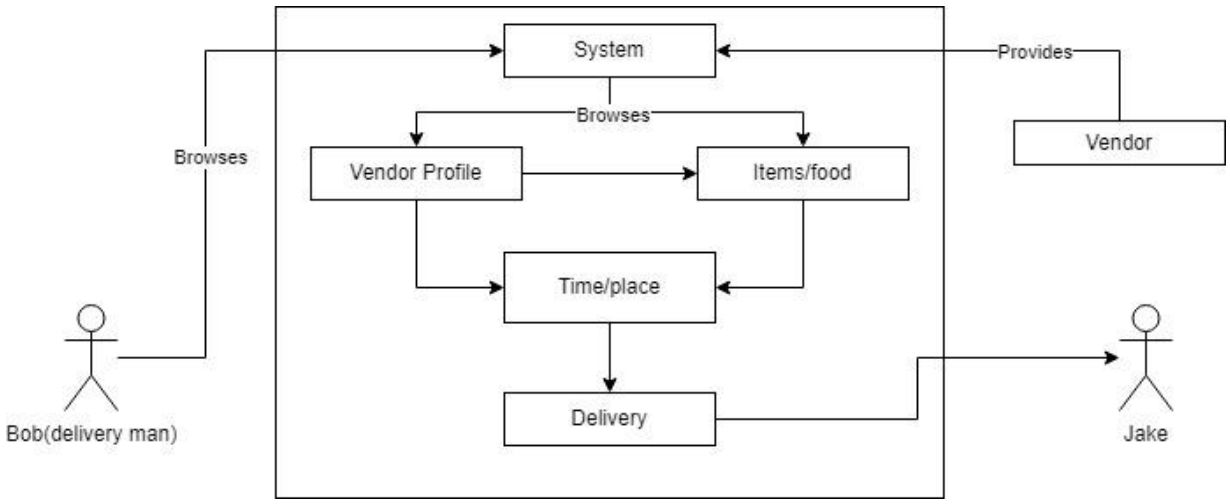


Use Case 3: Delivery Man

Actors: Bob (Delivery Man), Jake (User), Vendor

Description: Bob wants to make extra money during his free time. He decides to become a delivery man because of the flexible hours. He decides to become a delivery man for a certified farmers market.

Solution: As a delivery man, Bob will choose what items he would like to deliver. Bob decides to deliver items that Jake ordered on the farmers market website. He will go to the farmer's markets to meet the vendors. Bob will ask for the items that he has to deliver and be on his way. Bob will earn money by delivering these items to Jake.

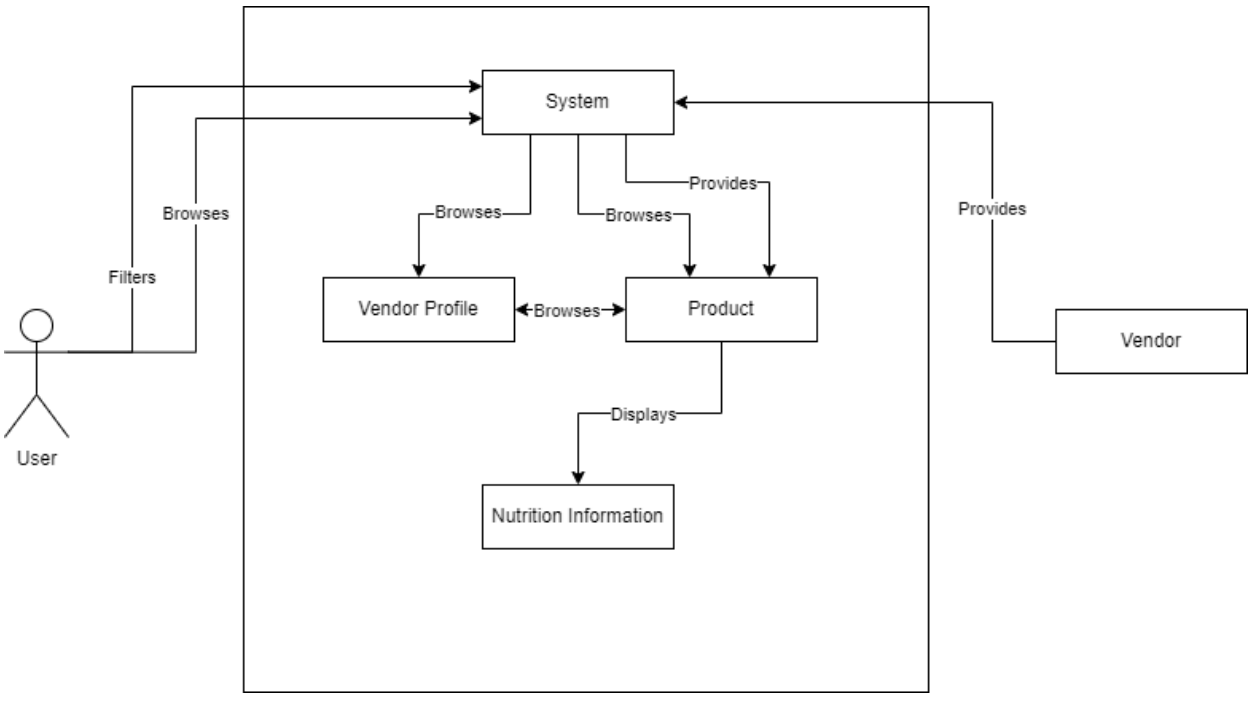


Use Case 4: Health Conscious Individual

Actors: Tom (User), Vendor

Description: Tom wanted to make a change in his life and he decided to start by eating better. Unfortunately, fresh produce is hard to come by in Tom's city. Farmer's Markets are non-existent and the nearest farm is miles away.

Solution: Tom can browse through TVFM's inventory and order fresh produce sourced from reliable farmers delivered to him via TVFM. Tom can filter through TVFM's inventory based on whatever his dietary needs are. In addition, Vendors will provide nutritional information that Tom can access whenever he needs to. Furthermore, TVFM will locate the nearest vendor to Tom's delivery address in order to shorten the delivery time and ensure he can get the freshest products possible.

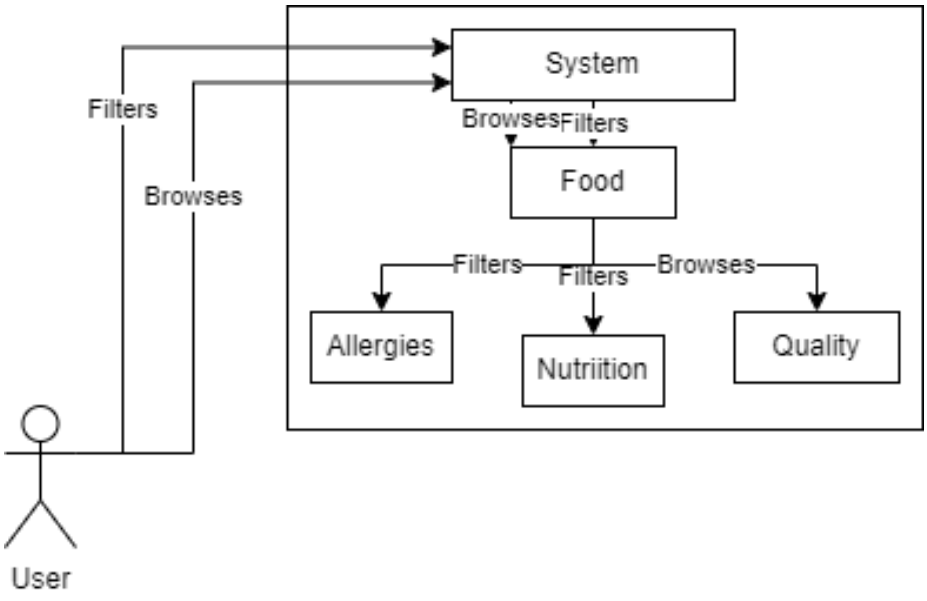


Use Case 5: Dietary Requirements

Actors: Pedro (User)

Description: Pedro has various allergies and dietary restrictions. Pedro also has kids that are very picky about their food. His kids demand nothing but the highest quality ingredients in their food. Pedro wants to ensure that he can buy groceries that fit his diet, allergies, and picky children. He could check the allergens, nutritional facts, and quality reviews from previous customers for each item. However, he would like to avoid this because it would be a very time-consuming process.

Solution: Through TVFM, Pedro can filter out certain products that do not fit his dietary requirements. Similarly, he can filter out certain products by their allergens. To meet his kid's standards Pedro can check the reviews for products that he selects and can quickly find out which ones are reliably high-quality produce.

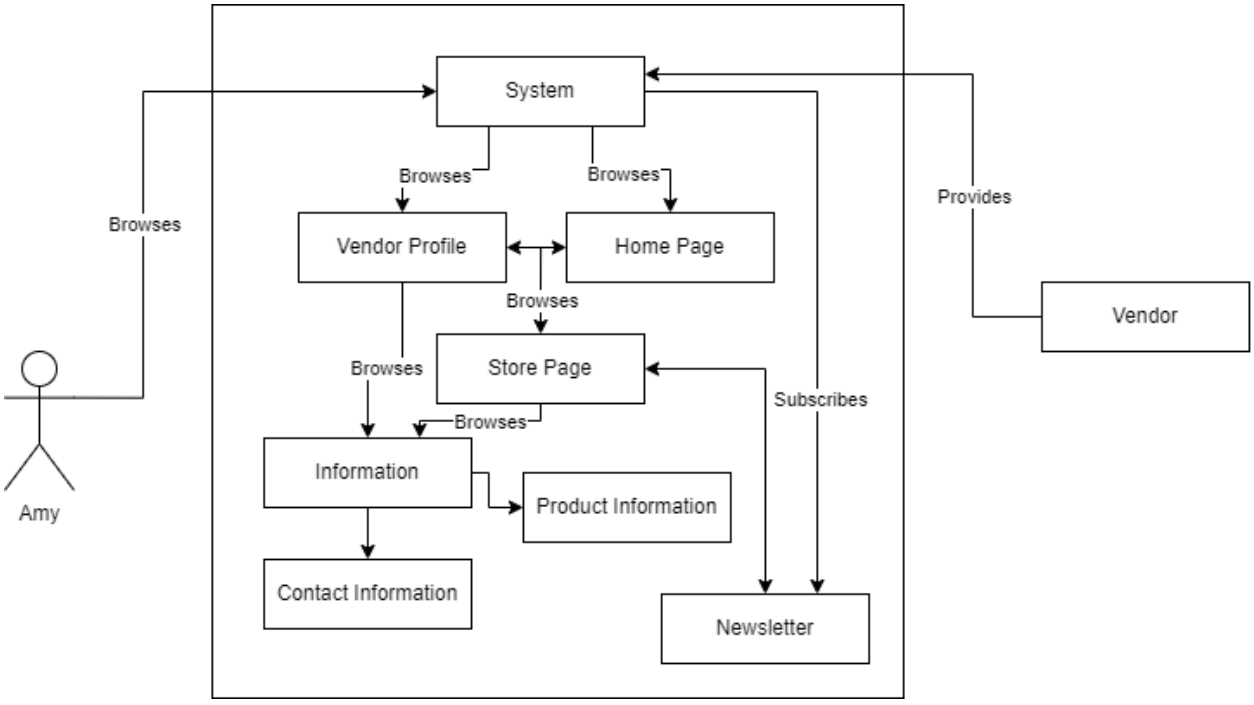


Use Case 6: Small Scale Farmer

Actors: Amy (User), Vendor

Description: Amy runs a small family farm. She wants to help grow her family business and expand her consumer base. She can't afford the same level of marketing that big brands can do.

Solution: TVFM can help Amy grow her business by providing a platform for commerce and giving exposure to her business ventures. Amy can create and manage a store page on our site. We will also feature vendors on the homepage in order to help them attract customers and gain exposure. Lastly, we will send out weekly newsletters that help promote businesses on our site.

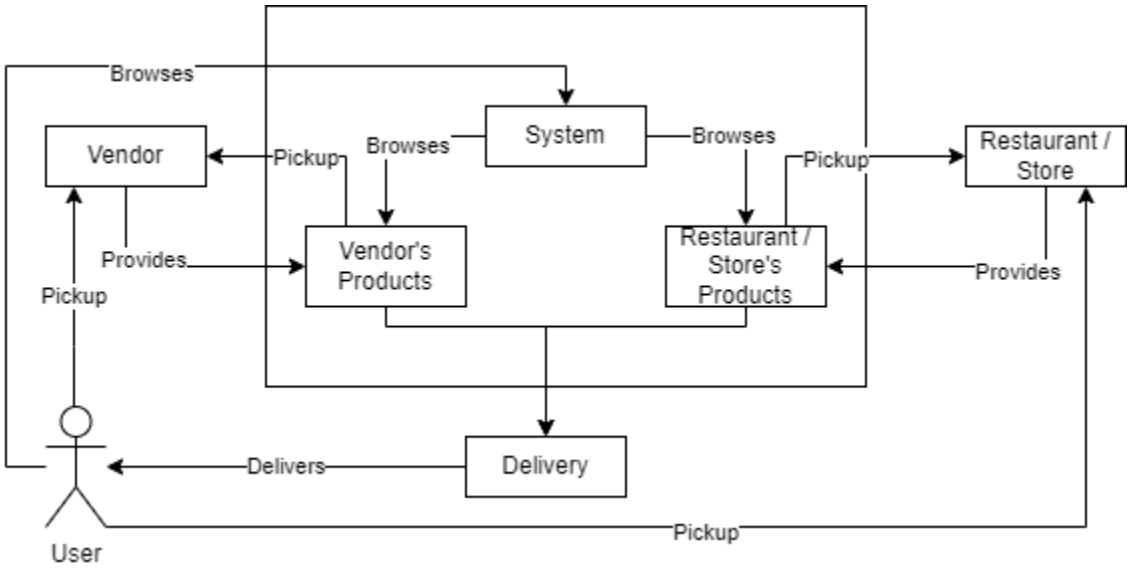


Use Case 7: Quick Lunch

Actors: Bob (User), Vendor

Description: Bob is a busy person that wants to buy a fresh meal. However, because Bob has a busy schedule, he does not have the time to wait for a fresh meal to be prepared.

Solution: Bob can order a fresh meal through The Virtual Farmer's Market. It can be sourced directly from a farmers market vendor, through a store, or restaurant. Bob can select from the variety of fresh meals and place an order in which he will be able to get it delivered or can quickly pick it up at a physical restaurant or store at a specified time.

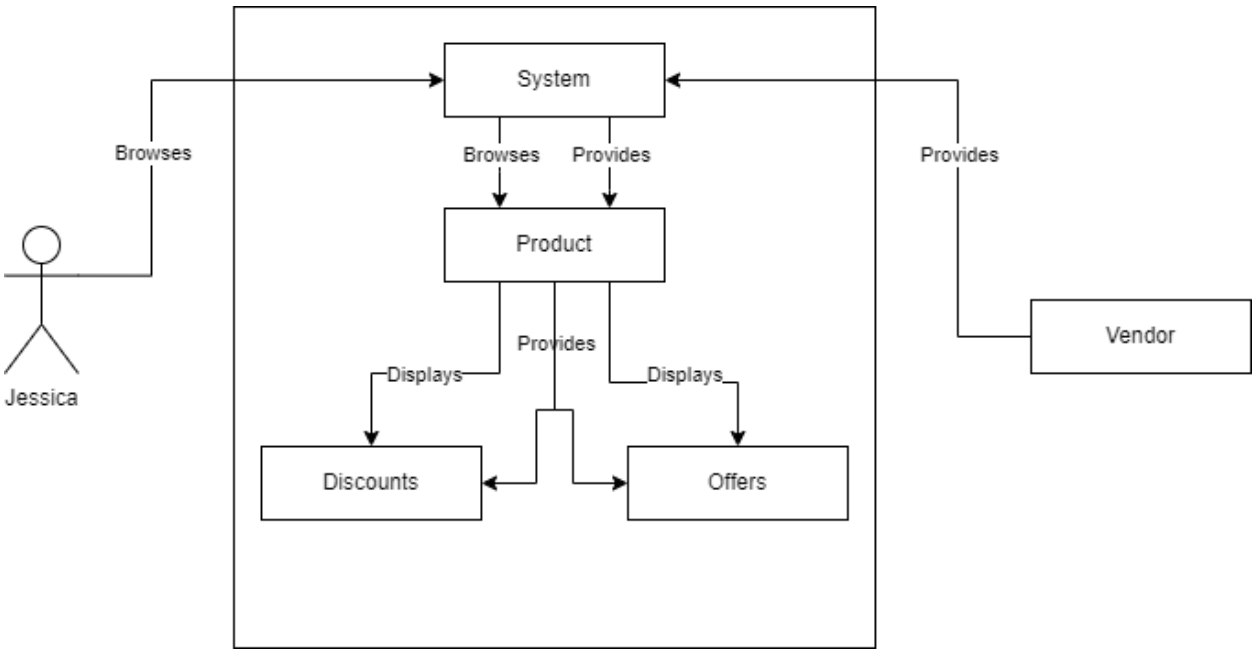


Use Case 8: Affordable Produce

Actors: Jessica (User), TVFM (Store)

Description: Jessica doesn't have a lot of money for her meals and she would also like to buy quality food where she can buy cheap and healthy products. She would like to know if there is any store where she can compare products or where she can find food offers since she would like to buy cheaper ones.

Solution: In TVFM, Jessica can compare products with other stores and can find out how cheap it is to buy on our web page without forgetting the quality of the products. The store also offers a page full of discounts and products that at the moment have a reduced price.

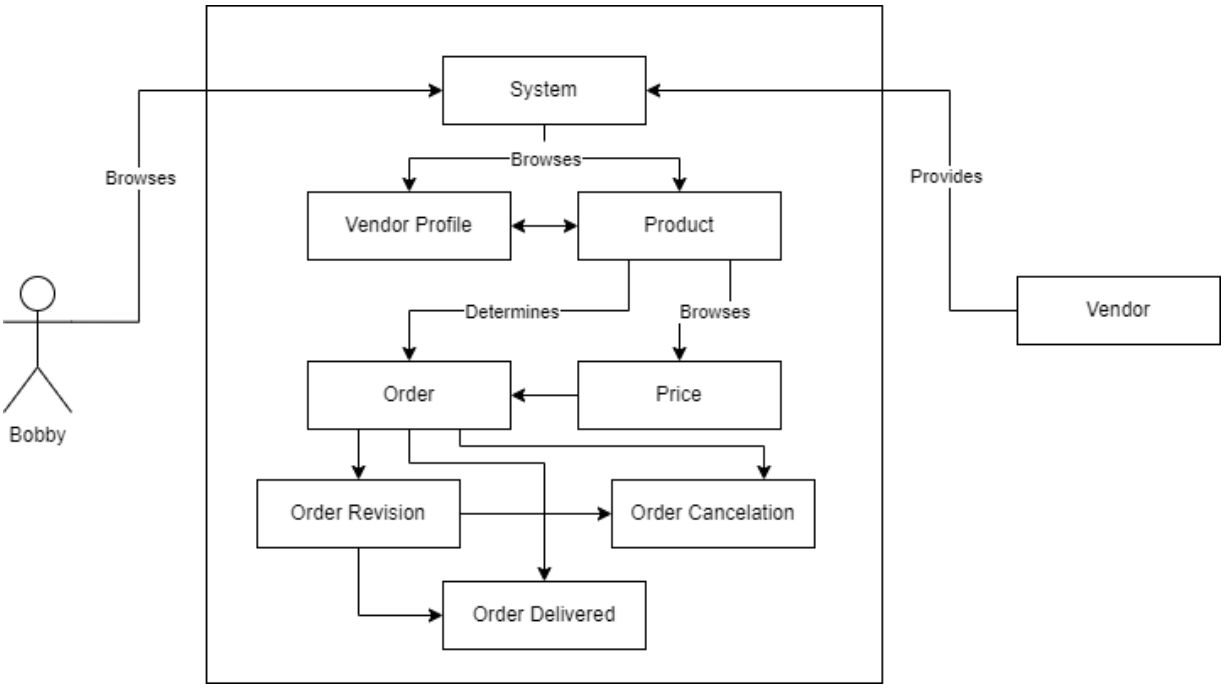


Use Case 9: Revising an Order

Actors: Bobby (User), Vendor

Description: Bobby realizes that he needs to buy groceries because he has run out. Instead of going to buy them at the store, Bobby decides to order his things in the virtual farmers market. After submitting his order, he realizes that he picked some of the wrong items.

Solution: After submitting his order and realizing that he bought the wrong items, he has a limited time to either cancel his order or revise it before the vendor confirms the order and hands it off to the delivery man.

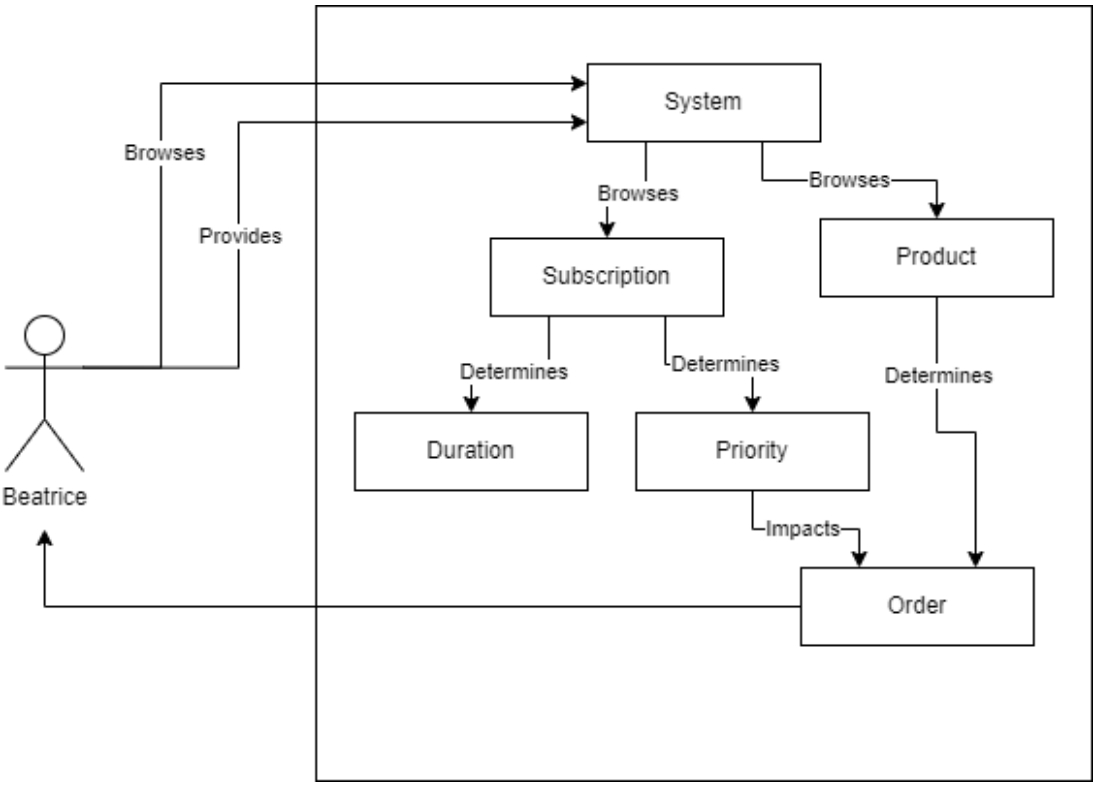


Use Case 10: Catering

Actors: Beatrice (User)

Description: Beatrice owns a restaurant that offers catering services. She needs a vast amount of ingredients on a weekly basis and local grocery stores sometimes have trouble keeping up with her needs.

Solution: TVFM offers subscription services. Beatrice can select products that she wants as well as the length of her subscription. For customers that rely on a constant supply of food such as Beatrice, TVFM also has a premium subscription service that gives both expedited deliveries, as well as priority on all of her deliveries.

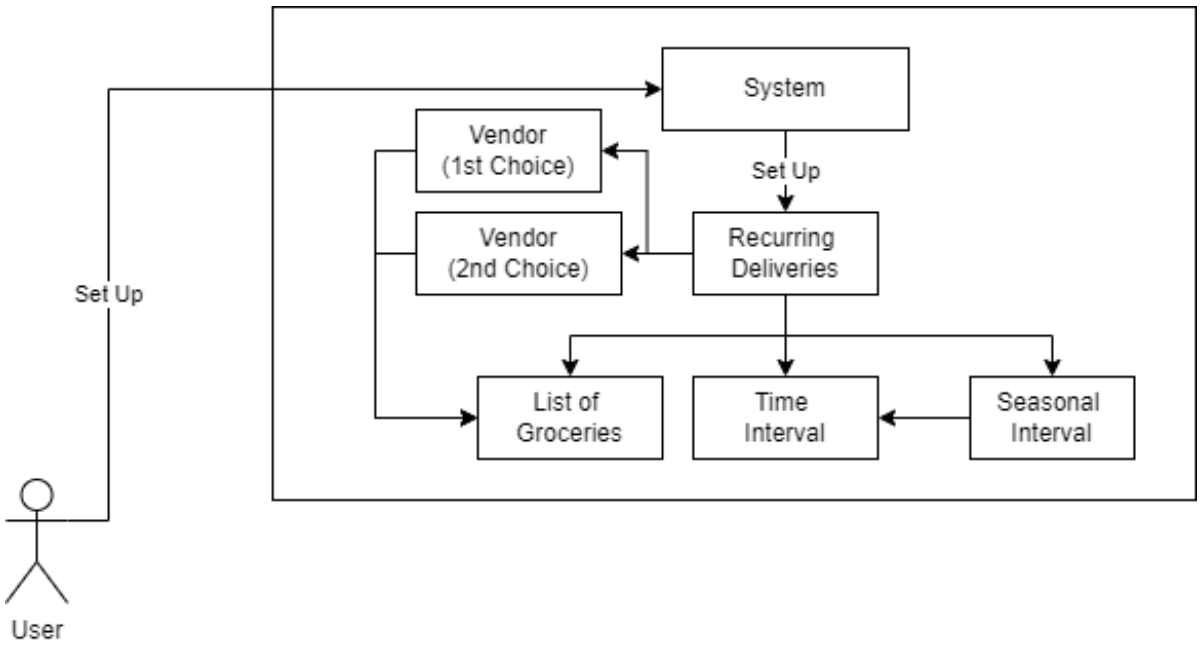


Use Case 11: Meal Planning

Actors: Jasmin (User)

Description: Jasmin likes to plan everything in advance including her meals. Therefore she would like a recurring service that can reliably deliver certain essential groceries to her house.

Solution: Jasmin can set up recurring deliveries through The Virtual Farmer's Market. She can specify which groceries should be delivered at which times and how much. She may also set up seasonal deliveries for certain produce that only grows during certain seasons. She can also specify a second source to choose produce from if the first source is out of stock.

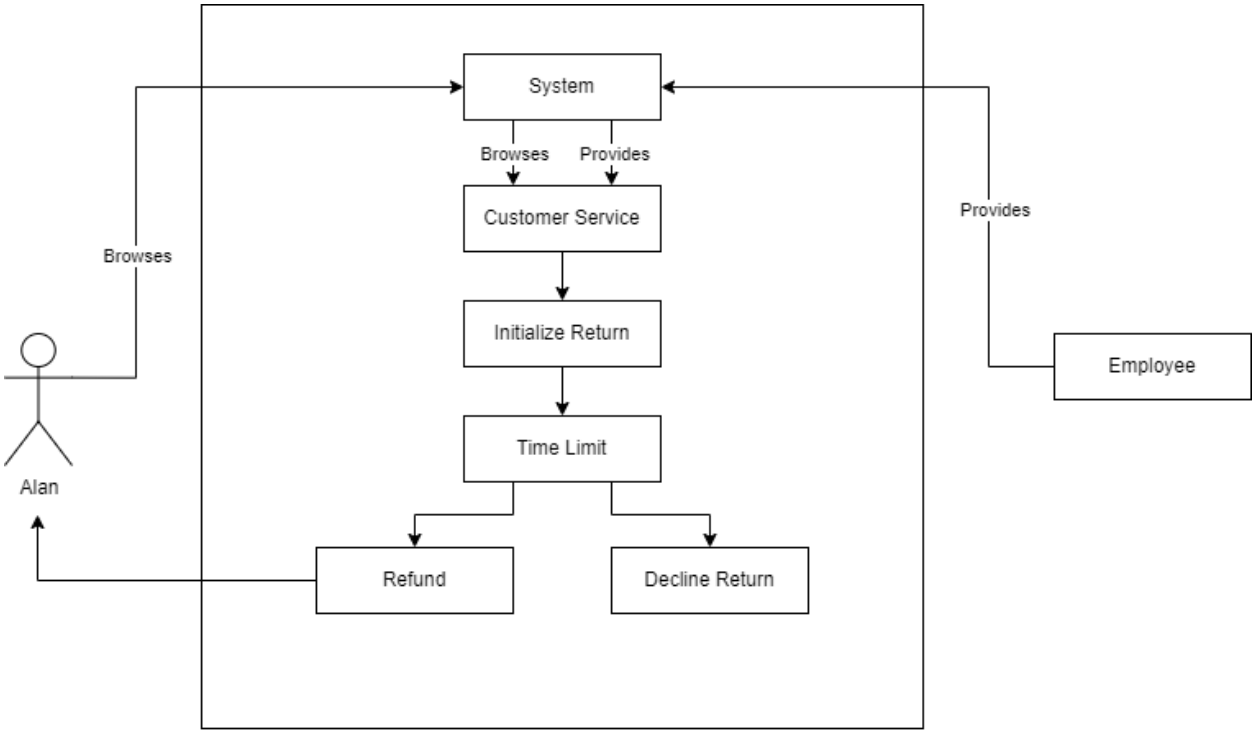


Use Case 12: Returning Items

Actors: Alan (User), Customer Service

Description: Alan is determined to buy food for his best friend. He goes to the virtual farmers market and buys food for his friend. When he arrives home and gives the food to his friend, he sees that he refuses the food because he is allergic to certain products. Alan wants to know if he can return the purchased products.

Solution: Alan has to go to customer service at a nearby store to return the products he bought. The store offers a period of time in which customers can return the purchased products, if Alan returns items within the time limit, Alan must provide the products he bought and the employee takes care of the returning process so that Alan gets his money back and is satisfied with the customer service. Otherwise, the return process is declined.

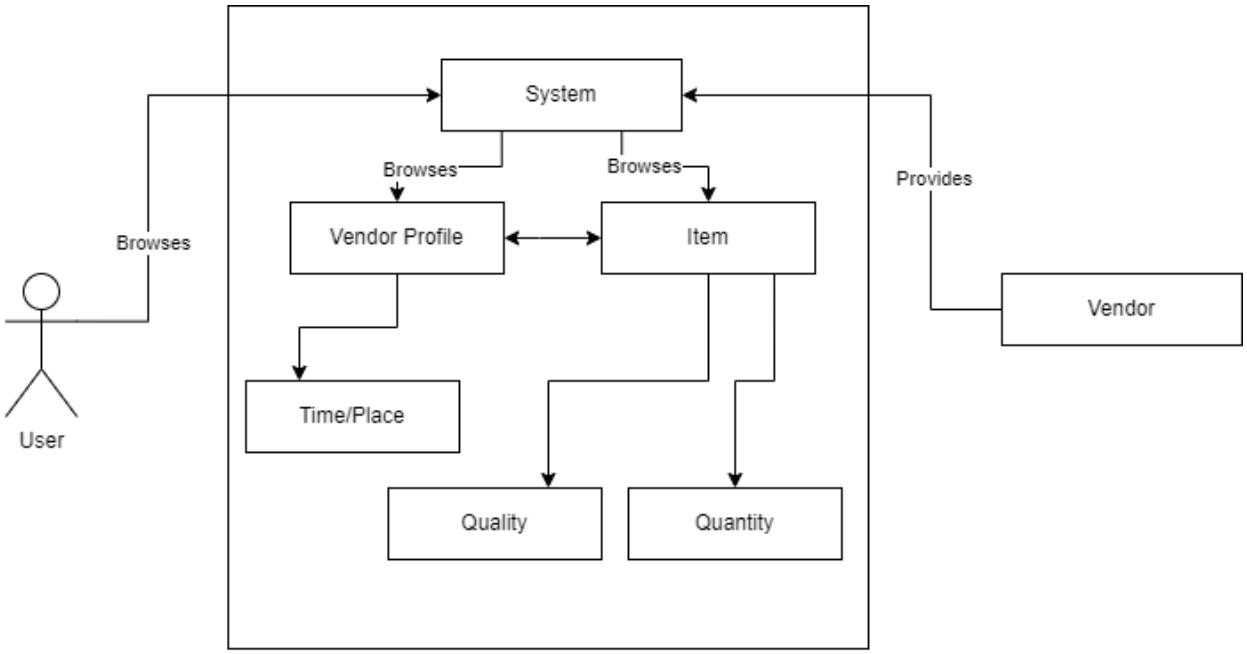


Use Case 13: Contact Information

Actors: Jill (Customer), James (Vendor)

Description: Jill wants to contact James. James doesn't have his business information published in any easily accessible manner. James also wants his business information to be easily found on platforms that are relevant to farmer’s markets. Both the customer and vendor want a system that is both convenient and simple to use.

Solution: James has his business contact information listed on TVFM. His information is listed on his profile. James has the option to update or remove that information whenever. Jill can easily find this information, as well as look at other vendors.



Main data items and entities

1. User

1.1 A user shall be able to browse food.

1.2 A user shall be able to purchase food.

1.3 A user shall be able to have their food delivered at a specified location.

1.4 A user shall be able to browse for food using filters.

2. Vendor

2.1 A vendor shall be able to sell food.

2.2 A vendor shall provide the nutritional facts and allergens for their food.

2.3 A vendor shall provide their contact information.

2.4 A vendor shall be able to deliver to at least one location.

3. Food

3.1 Food shall be either an ingredient or a meal.

3.2 Food shall contain 0 or more allergens.

3.3 Food shall have nutritional facts.

4. Filters

4.1 A filter shall describe a product’s allergen information

4.2 A filter shall describe a product’s nutritional facts.

4.3 A filter shall describe a product's available quantity.

4.4 A filter shall describe a product's price.

4.5 A filter shall describe what category a product falls in.

4.6 A filter shall describe a product's seasonality.

4.7 A filter shall describe dietary preferences and restrictions.

5. Recurring Delivery

5.1 A recurring delivery shall be delivered to one location.

5.2 A recurring delivery shall source its food from one or more vendors.

Initial list of functional requirements

1. Registration

    1.1.1. General users shall be able to enter their data and sign up for our services

2. Registered Customer Account

    2.1.1. A general user shall be able to set up a registered customer account.

    2.1.2. A registered customer shall only purchase items.

3. Registered Vendor Account

    3.1.1. A general user shall be able to set up a registered vendor account which will allow them to sell their items on our site.

    3.1.2. A registered vendor account shall only sell items.

4. Search

    4.1.1. A general user or registered customer shall be able to search for items.

    4.1.2. A general user or registered customer shall be able to search for registered vendor accounts

5. Filtering

    5.1.1. A general user registered customer shall be able to use filters when searching to narrow down their selection of items.

6. Purchase

    6.1.1. A general user or a registered customer account shall be able to purchase items from registered vendor accounts.

7. Recurring Deliveries

    7.1.1. A registered customer account shall be able to create a recurring delivery in order to receive items at given time intervals.

    7.1.2. A registered customer account shall be able to cancel their recurring delivery whenever they want.

8. Contacting Vendors

    8.1.1. A general user or registered customer shall be able to access a registered vendor’s contact information to ask about their items.

9. Nutrition Information

    9.1.1. A general user or a registered customer shall be able to view the nutrition information of an item.

10. Price Comparison

    10.1.1. A general user or a registered customer shall be able to compare the prices of the same item sold by different registered vendors.

11. Vendor Proximity

    11.1.1. A general user or a registered customer shall be able to search for registered vendors closest to a given address

12. Vendor Pricing

    12.1.1. A registered vendor shall be able to set the price for the items they are selling.

13. Alternatives

    13.1.1. A general user or a registered customer shall be able to pick a different registered vendor if their first choice ran out of stock.

14. Editing Customer Information

    14.1.1. A general user or a registered customer shall be able to edit their information.

15. Total Price

    15.1.1. A general user or a registered customer shall be able to see the total price of all the items in their shopping cart before finalizing the transaction.

16. Sort

    16.1.1. A general user or a registered customer shall be able to sort their search results based on given criteria.

17. Tracking ID

    17.1.1. A general user or a registered customer shall be given a tracking ID for their orders.

18. Cart

    18.1.1. A general user or a registered customer shall be able to see items they added to their cart.

18.1.2. A general user or a registered customer shall be able to add and remove items or change the quantity of an item in their cart.

19. Delivery/Pickup

19.1.1. A general user or a registered customer shall be able to pick up or have their items delivered to them.

20. Item Description

20.1.1. A general user or a registered customer shall be able to read a clear description of an item.

21. Login/Logout

21.1.1. A general user shall be able to login and logout of their registered customer and registered vendor accounts.

22. Editing Vendor Information

22.1.1. A registered vendor shall be able to edit their information.

23. Selecting a Vendor

23.1.1. A general user or a registered customer shall be able to select a registered vendor that they will solely buy items from.

24. Reviews

24.1.1. A registered customer shall be able to post reviews for an item.

25. Revising an Order

25.1.1. A general user or a registered customer shall be able to change an order within a certain time frame.

26. Discounts

26.1.1. A general user or a registered customer shall be able to view discounts for certain items.

26.1.2. A registered vendor shall be able to provide discounts for their items.

27. Previous Purchases

27.1.1. A general user or a registered customer shall be able to view their previous purchases of items.

28. Recommended Products

28.1.1. A general user or a registered customer shall be recommended items that may interest them based on previous purchases.

29. Following a Vendor

29.1.1. A general user or a registered customer shall be able to follow a registered vendor for updates about a vendor’s items.

30. Discounting Bulk Purchases

30.1.1. A general user or a registered customer shall receive a discount if the total price of items in their cart exceeds a certain amount.

31. Customer Service

31.1.1. A general user or a registered customer shall be able to contact customer service if they have any issues.

32. Rewards

32.1.1. A general user or a registered customer shall be able to see the rewards page.

32.1.2. A general user or a registered customer shall receive rewards points for every purchase.

33. Returns

33.1.1. A general user or a registered customer shall be able to return their purchased items within a certain time.

34. Newsletter

34.1.1. A general user or a registered customer shall receive a newsletter notifying them about deals, new products, and popular vendors.

34.1.2. A general user or a registered customer shall opt-in to the newsletter and be able to opt-out at any time.

35. Careers

35.1.1. A general user shall be able to apply for working positions at TVFM.

36. Social Media

36.1.1. Anyone shall be able to see offers, features, discounts, events on the social media account of TVFM.

37. Advertisement

37.1.1. A general user or a registered customer shall receive advertisements about items they may be interested in.

37.1.2. A registered vendor shall be able to pay to have an advertisement about their item shown to a general user or a registered customer.

List of non-functional requirements

Coding Standards:

1. All variables shall use camelCase.
2. Space between ‘if’ and the condition i.e if (condition).
3. Space between parentheses and the curly brace i.e () {}.
4. Space before and after “=”.
5. Comment above each function describing its purpose.

Performance:

1. The home page shall load in at most 500 ms.

Compatibility:

1. Current and previous version of Google Chrome.
2. Current and previous version of Mozilla FireFox.
3. Current and previous version of Microsoft Bing.
4. Current and previous version of Safari by Apple.

Scalability:

1. Application will be able to accommodate multiple users.
2. Application will be able to scale up and down depending on device.

Portability:

1. Website will function properly on different devices.

Reliability:

1. Application will consistently perform without failure.

Security:

1. User's personal information to be kept confidential and not distributed.
2. User's passwords to be encrypted.
3. Users may not be granted access till a strong password is created.
4. System will lock the user out after a specific number of login attempts.

Usability:

1. Interface is easy to use, user-friendly.

Availability:

1. Support during business hours.
2. Common question solutions will be available for non-business hours

Regulatory:

1. Application will comply with all regulations and laws.

Data Integrity:

1. System shall have backups of all updates to the database for every transaction.

Cost:

1. Developmental costs will be kept at a min by using free services provided online.

Testability:

1. Application will have a suitable testing approach for each feature.

Organization:

1. Company name and logo on the top-left corner of each web page.
2. Files and folders will be easily accessible and not confusing

Competitive analysis

	Instacart	**Walmart**	**Farm fresh to you**	**Safeway**	**Amazon Fresh**
Strengths	-12 stores to choose from - Estimated delivery time/ same day delivery - Can give specific instructions to the person picking up your food - Easy partnership	- Wide Selection - Affordable for everyone - Worldwide organization - Reward system	- Appealing to the public - Customizable options - Third parties can associate easily with the company	- Large organic and non-organic selection - Unlimited free delivery over order of $30 - Reward system	- Large selection of products - Same day Delivery - Great prices
Weaknesses	- Dependency on third parties - Sustainability, decrease of workforce due to pandemic - Mismatch on items delivered and customer choice	- It is significantly disadvantaged against premium retailers - Expensive if the service is not in use	- Limited to bay area - Mixed reviews about delivery	- Prices are above average - Constantly out of stock - Delivery issues - Not financially viable for families below 3-4 people	- Must have Amazon Prime Subscription -Not appealing to the public -Delivery fee for orders under $35
Pricing	Various fees per order + Original Cost of food	$12.95/Month $98/year	Various fees per order + Original Cost of food	$12.99/Month $99/year	Must be Amazon prime member (14.99/month)
User Experience	Good organization of products (Mostly) Not easy to understand at first since the page contains a lot of options.	Decent Great to find what you are looking for. But you can be overwhelmed by a lot of products and poor descriptions.	Moderate. Great number of steps. You can easily lose what you are looking for due to the great number of options.	Decent Easy to find products. Appealing to the user. But, you can feel overwhelmed by all the information.	Moderate There is a lot going on in the home page and you can feel overwhelmed by all the products. - Not that appealing to the user

Social Media	-Instagram -Facebook -Twitter	-Facebook -Twitter -Instagram	-Facebook, -Twitter, -Instagram, -Youtube	-Instagram, -Facebook, -Twitter	-Facebook, -Twitter, -Instagram

Features	**InstaCart**	**Walmart**	**Farm fresh to you**	**Safeway**	**Amazon Fresh**	**The Virtual Farmers' Market**
Shopping Cart	(+)	(++)	(+)	(++)	(+)	(+)
Searching	(+)	(+)	(+)	(+)	(+)	(+)
Vendors	(+)	(+)	(++)	(+)	(++)	(+)
Payment System	(+)	(+)	(+)	(+)	(+)	(+)
Functionality (easy to use?)	(+)	(+)	(+)	(+)	(+)	(++)
Contact Info/ Social Media	(+)	(+)	(+)	(++)	(+)	(+)
Filters	(-)	(++)	(-)	(+)	(+)	(+)
Product descriptions/ distribution of content	(+)	(+)	(+)	(+)	(+)	(++)
Reward System	(+)	(+)	(-)	(+)	(+)	(++)

Competitive Analysis Summary

On our site we are trying to make it appealing to the public so that users can find the highest quality food at the lowest price. We seek to compete with our competitors' delivery times. Similarly, we want our page to be much easier to use than our competitors so that users can quickly navigate and locate the products they want. We also want our customers to feel safe with their purchases. Therefore, we offer filters that make it easier to search for products for people with allergies, illnesses, or simply people who are looking for something specific from the nutritional facts. This is something that none of our competitors offer since the information about it is contained in the description of the products. Finally, we want to promote the rewards system and our social networks, so that users can see the weekly promotions and earn points for their next purchase.

High-level system architecture and technologies used

Server Host: AWS

Operating System: Ubuntu 22.04

Server Database: MySql 8.0

Web Server: Apache 2.4

- Server-Side Framework: Spring Boot 2.7.3
- Language: Java 17

Web Framework: React

IDE: IntelliJ, Visual Studio

Checklist

To Do	Done/ On Track/ Issues
Team found a time slot to meet outside of the class	Done
Github master chosen	Done
Team decided and agreed together on using the listed SW tools and deployment server	Done
Team ready and able to use the chosen back and front end frameworks and those who need to learn are working on learning and practicing	Done
Team lead ensured that all team members read the final M1 and agree/ understand it before submission	On Track
Github is organized as discussed in class (e.g. master branch, development branch, folder for milestone documents etc.)	On Track

Milestone 2

Index

Data Definitions

General User:

- Name

Registered Vendor:

- Address
- Email
- Name
- Phone Number
- Items Currently Selling

Registered Customer:

- Address
- First Name
- Last Name
- Email
- Phone Number

Item:

- Vendor Source
- Meal
    - Name
    - Price

        - Ingredients
        - Nutrition
            - Allergens
            - Calories
            - Sugar
            - Carbohydrates
            - Fat
            - Protein
- Produce
    - Name
    - Price
- Packaged Food
    - Name
    - Price
    - Weight
    - Nutrition
        - Allergens
        - Calories
        - Sugar
        - Carbohydrates
        - Fat
        - Protein

Filters:

- Item Category
- Nutrition
    - Allergens
    - Calories
    - Sugar
    - Carbohydrates
    - Fat
    - Protein

Recurring Delivery:

- Address to deliver to
- Items to deliver
- Frequency
- Duration
    - Seasonal

Address:

- Street Number
- Street
- City
- County
- State

- Zip code

Prioritized Functional Requirements

Priority 1:

1. Registration:

    1.1 A general user shall be able to enter their data and sign up for our services.

2. Registered Customer Account:

    2.1 A general user shall be able to set up a registered customer account.

    2.2 A registered customer shall only purchase items.

3. Registered Vendor Account:

    3.1 A general user shall be able to set up a registered vendor account.

    3.2 A registered vendor account shall only sell items.

4. Search:

    4.1 A general user or registered customer shall be able to search for items.

    4.2 A general user or registered customer shall be able to search for registered vendor accounts.

5. Filtering:

    5.1 A general user or registered customer shall be able to use filters when searching to narrow down their selection of items.

7. Purchase:

7.1 A general user or a registered customer account shall be able to purchase items from registered vendor accounts.

9. Recurring Deliveries:

9.1 A registered customer account shall be able to create a recurring delivery in order to receive items at given time intervals.

9.2 A registered customer account shall be able to cancel their recurring delivery.

11. Contacting Vendors:

11.1 A general user or registered customer shall be able to access a registered vendor's contact information to ask about their items.

13. Nutrition Information:

13.1 A general user or a registered customer shall be able to view the nutrition information of an item.

15. Price Comparison:

15.1 A general user or a registered customer shall be able to compare the prices of the same item sold by different registered vendors.

16. Vendor Proximity:

16.1 A general user or a registered customer shall be able to search for registered vendors closest to a given address

17. Vendor Pricing:

17.1 A registered vendor shall be able to set the price for the items they are selling.

20. Alternatives:

20.1 A general user or a registered customer shall be able to pick a different registered vendor if their first choice ran out of stock.

23. Editing Customer Information:

23.1 A registered customer shall be able to edit their information.

24. Total Price:

24.1 A general user or a registered customer shall be able to see the total price of all the items in their shopping cart before finalizing the transaction.

26. Sort:

26.1 A general user or a registered customer shall be able to sort their search results based on given criteria.

27. Tracking ID:

27.1 A registered customer shall be given a tracking ID for their orders.

33. Cart:

33.1 A general user or a registered customer shall be able to see items they added to their cart.

33.2 A general user or a registered customer shall be able to add and remove items or change the quantity of an item in their cart.

35. Delivery/Pickup:

35.1 A general user or a registered customer shall be able to pick up or have their items delivered to them.

38. Item Description:

38.1 A general user or a registered customer shall be able to read a clear description of an item.

41. Login/Logout:

41.1 A general user shall be able to login and logout of their registered customer and registered vendor accounts.

42. Editing Vendor Information:

42.1 A registered vendor shall be able to edit their business contact information.

43. Selecting a Vendor:

43.1 A general user or a registered customer shall be able to select a registered vendor that they will solely buy items from.

Priority 2:

6. Reviews:

6.1 A registered customer shall be able to post reviews for an item.

10. Revising an Order:

10.1 A general user or a registered customer shall be able to change an order within a certain time frame.

14. Discounts:

14.1 A general user or a registered customer shall be able to view discounts for certain items.

14.2 A registered vendor shall be able to provide discounts for their items.

19. Previous Purchases:

19.1 A registered customer shall be able to view their purchase history.

21. Recommended Products:

21.1 A general user or a registered customer shall be recommended items that may interest them based on previous purchases.

22. Following a Vendor:

22.1 A general user or a registered customer shall be able to follow a registered vendor for updates about a vendor’s items.

25. Discounting Bulk Purchases:

25.1 A general user or a registered customer shall receive a discount if the total price of items in their cart exceeds a certain amount.

31. Customer Service:

31.1 A general user or a registered customer shall be able to contact customer service if they have any issues.

37. Rewards:

37.1 A general user or a registered customer shall be able to see the rewards page.

37.2 A general user or a registered customer shall receive rewards points for every purchase.

Priority 3:

8. Returns:

8.1 A general user or a registered customer shall be able to return their purchased items within a certain time.

12. Newsletter:

12.1 A general user or a registered customer shall receive a newsletter notifying them about deals, new products, and popular vendors.

12.2 A general user or a registered customer shall opt-in to the newsletter and be able to opt-out at any time.

29. Careers:

29.1 A general user shall be able to apply for working positions at TVFM.

30. Social Media:

30.1 Anyone shall be able to see offers, features, discounts, events on the social media account of TVFM.

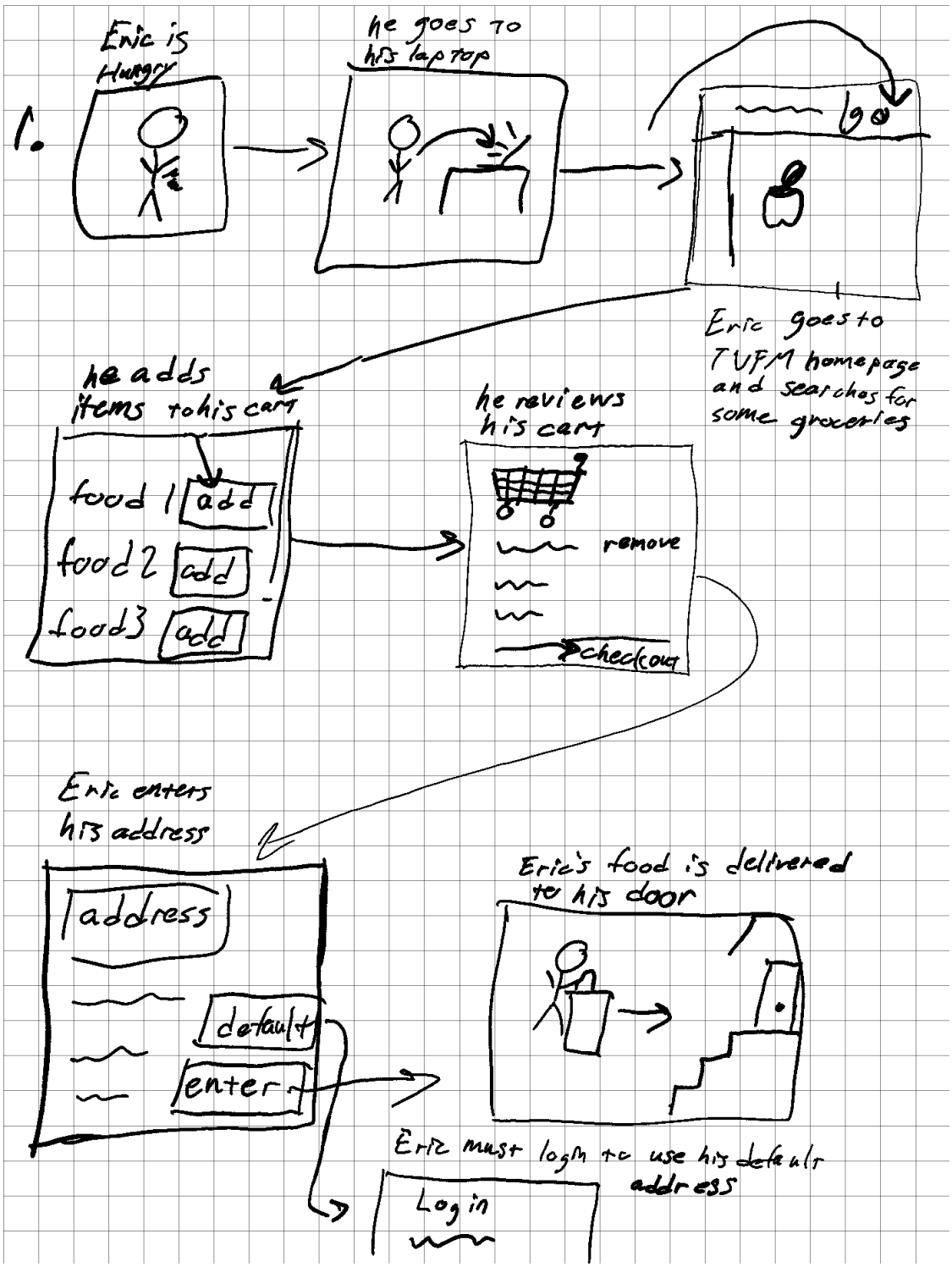
39. Advertisement:

39.1 A general user or a registered customer shall receive advertisements about items they may be interested in.

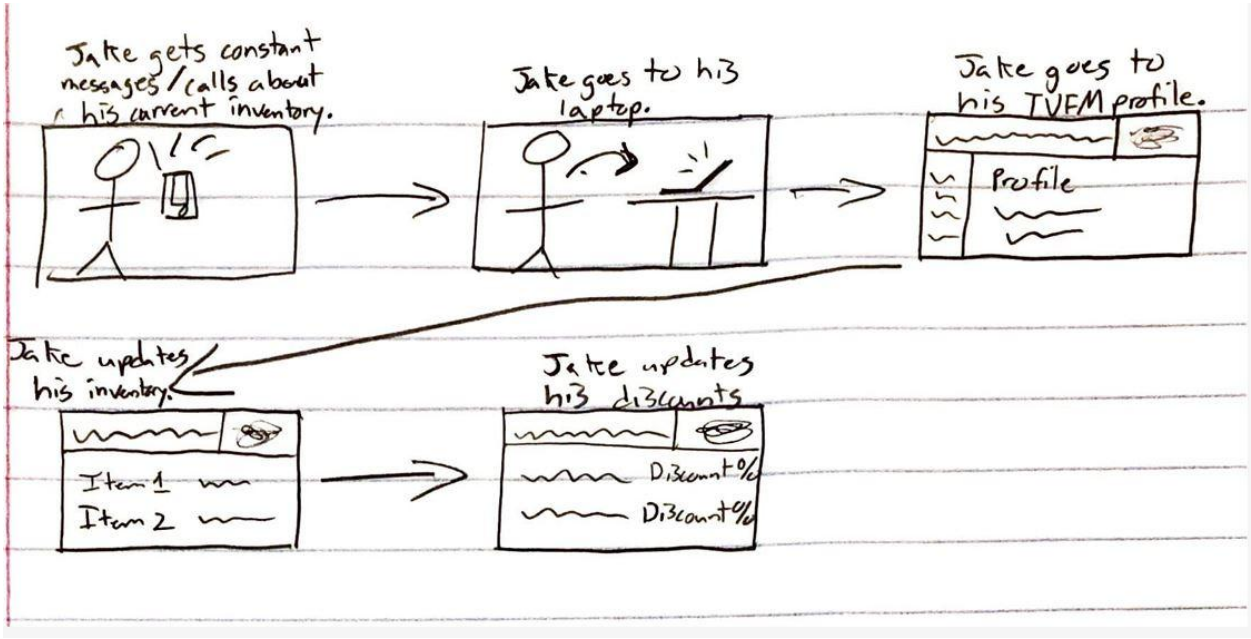
39.2 A registered vendor shall be able to pay to have an advertisement about their item shown to a general user or a registered customer.

UI Mockups and Storyboards

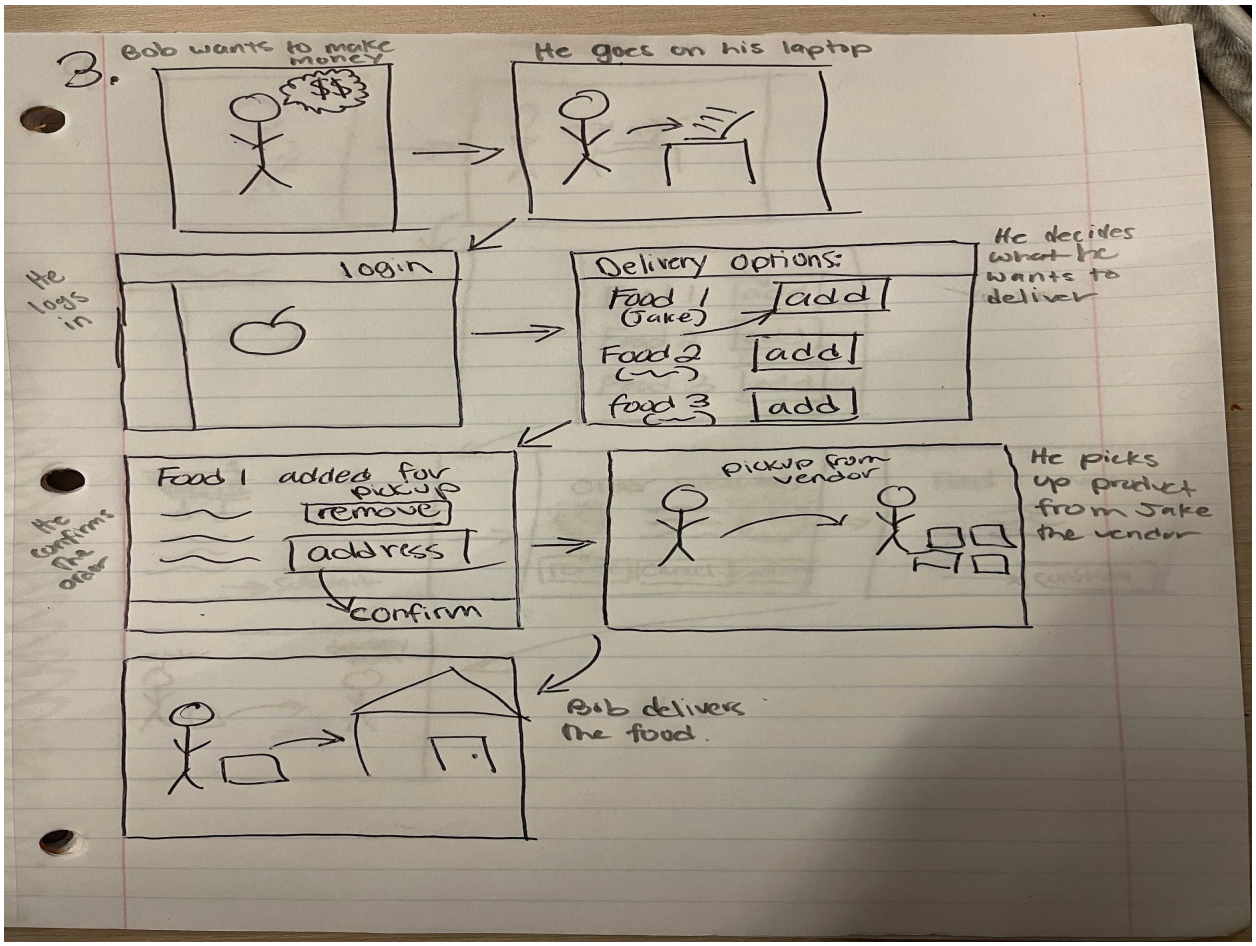
Use Case 1: Need Groceries



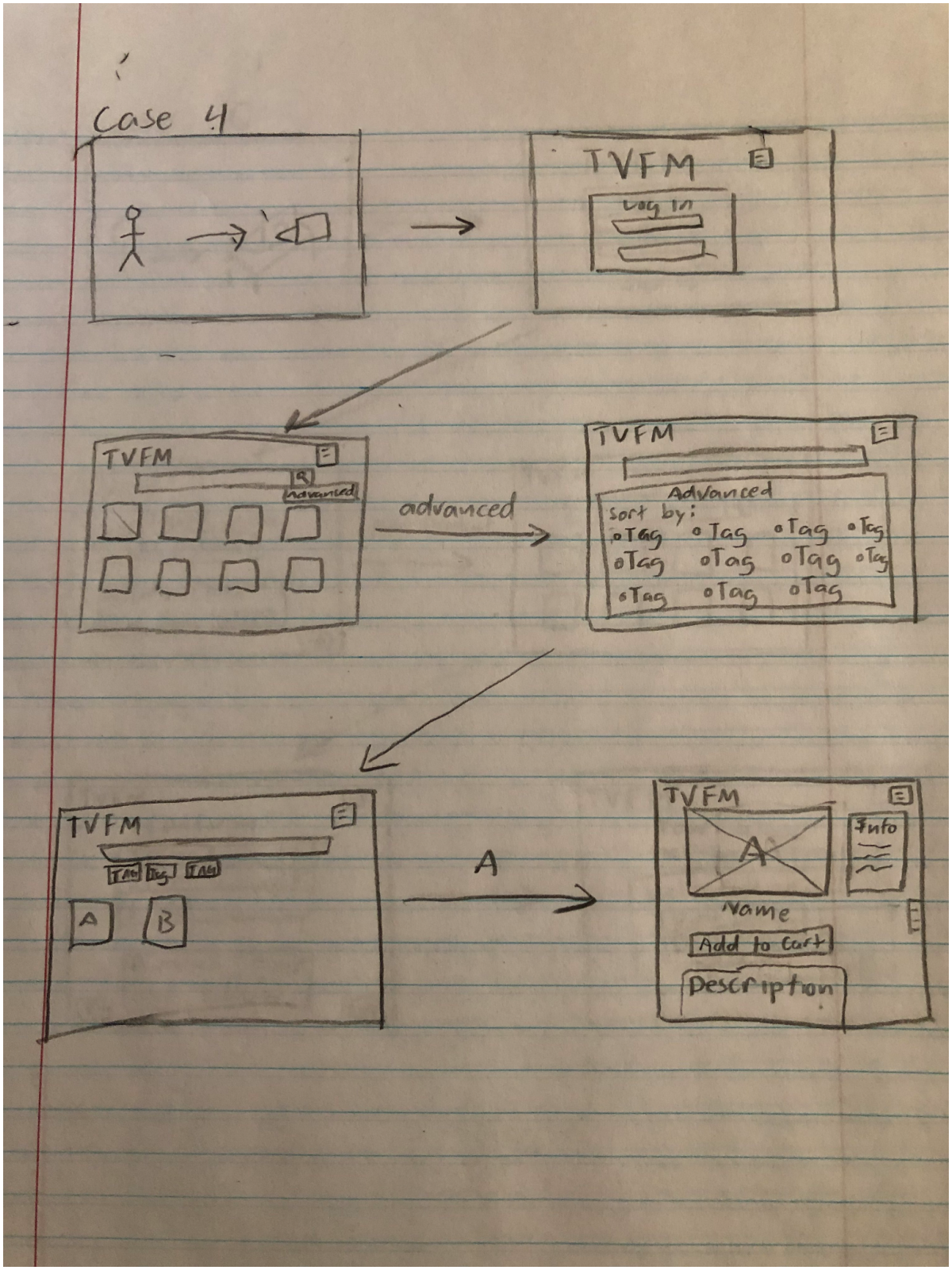
Use Case 2: Vendor Inventory



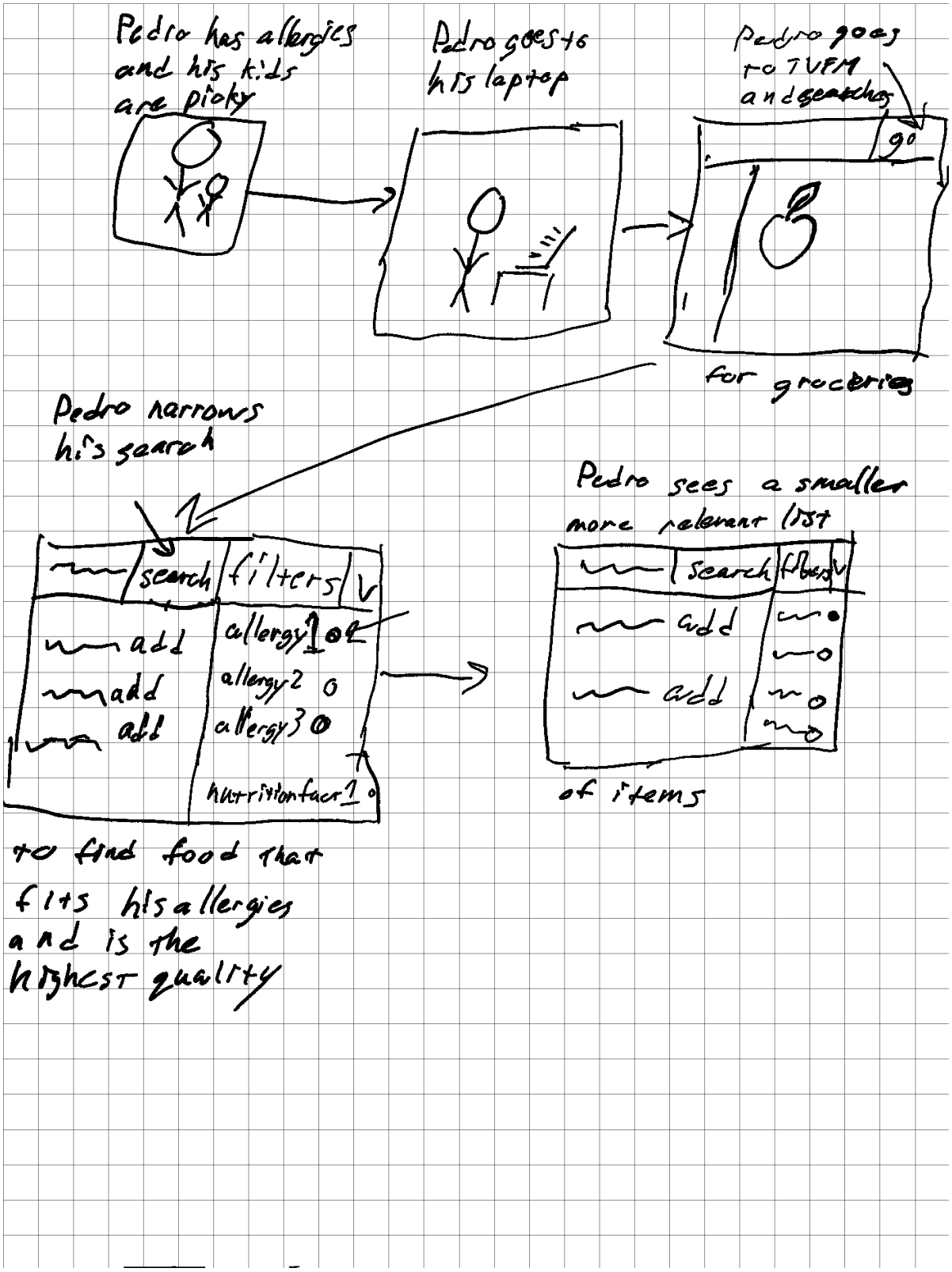
Use Case 3: Delivery



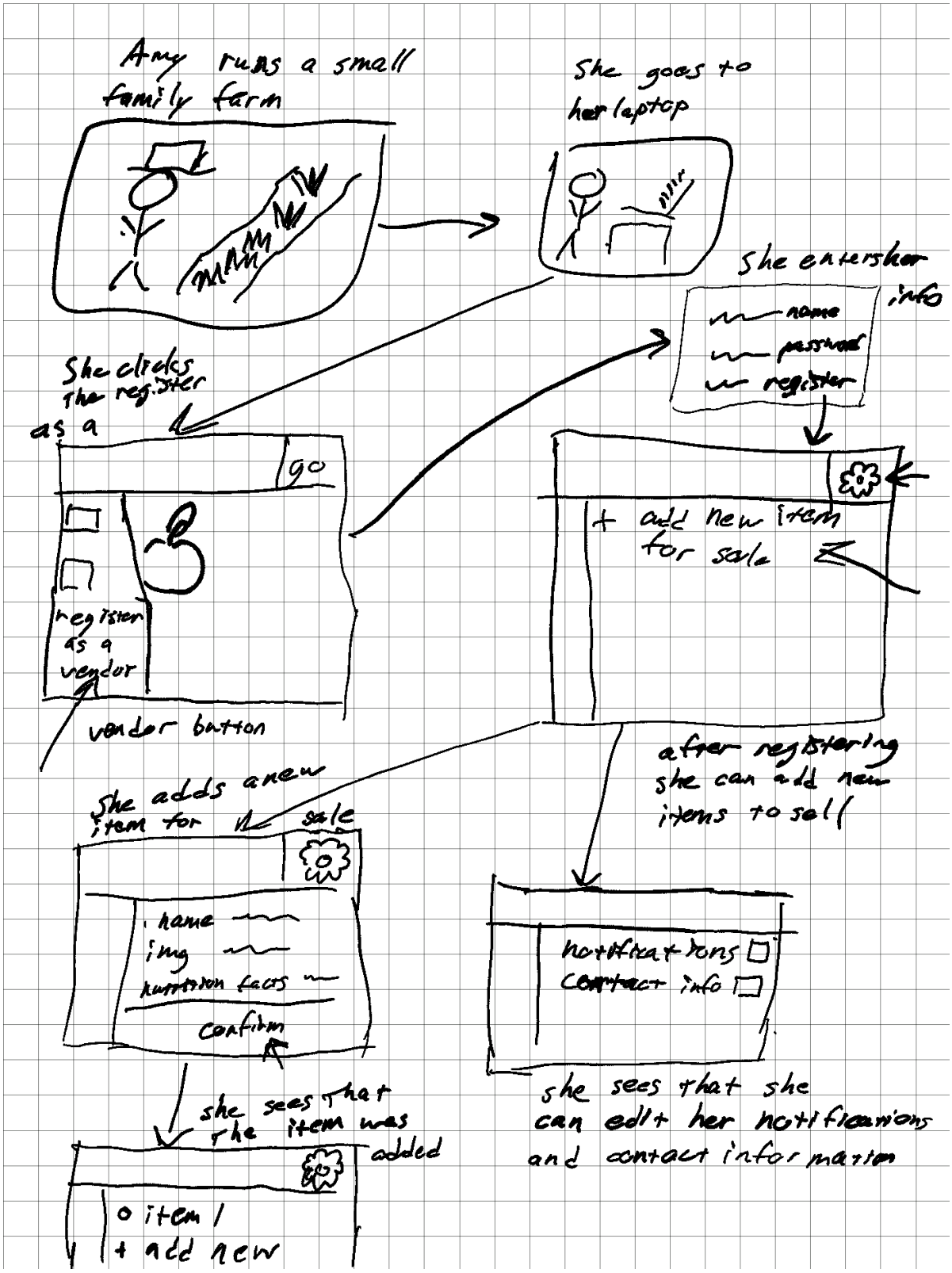
Use Case 4: Health Conscious Individual



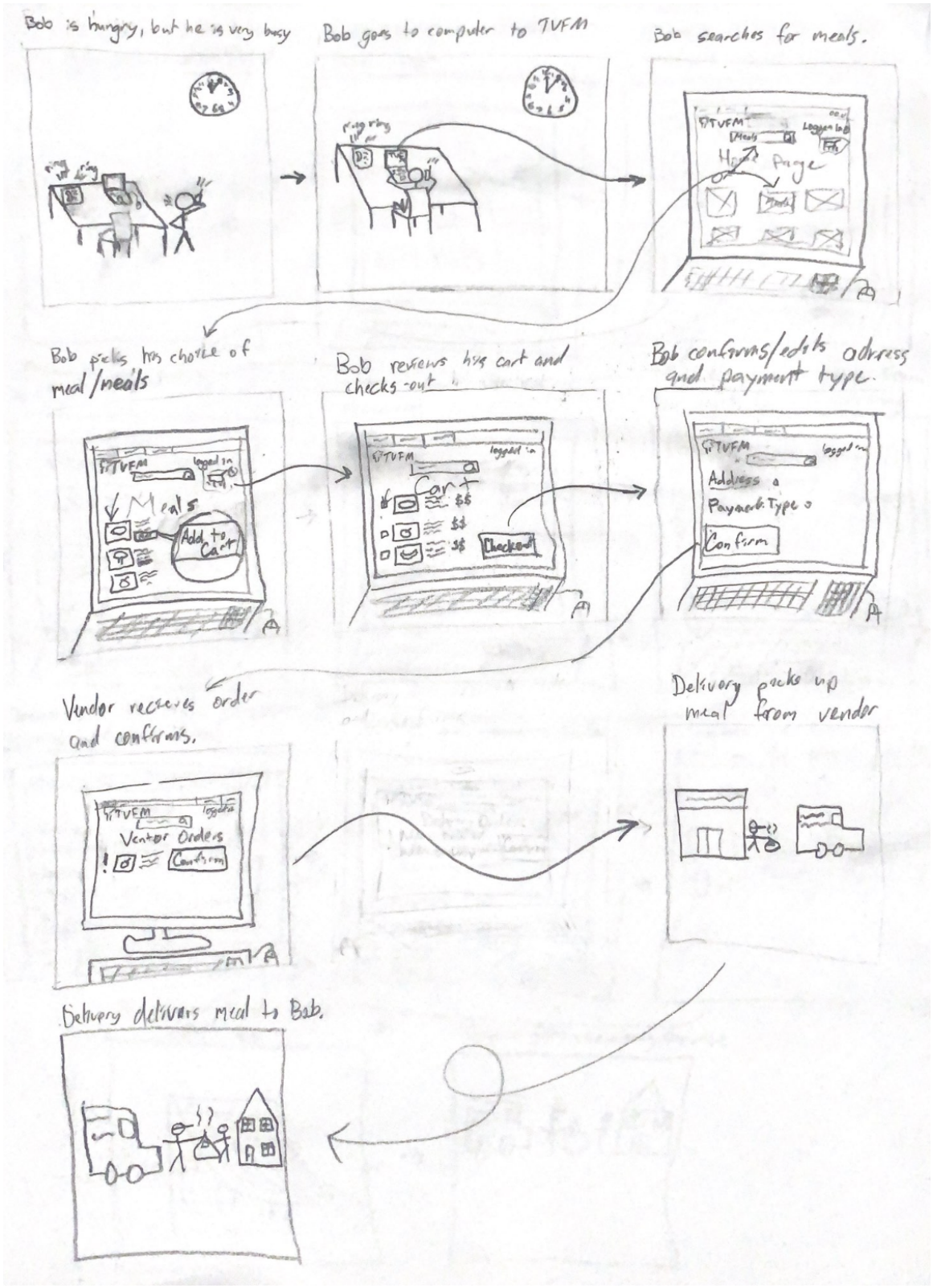
Use Case 5: Dietary Requirements



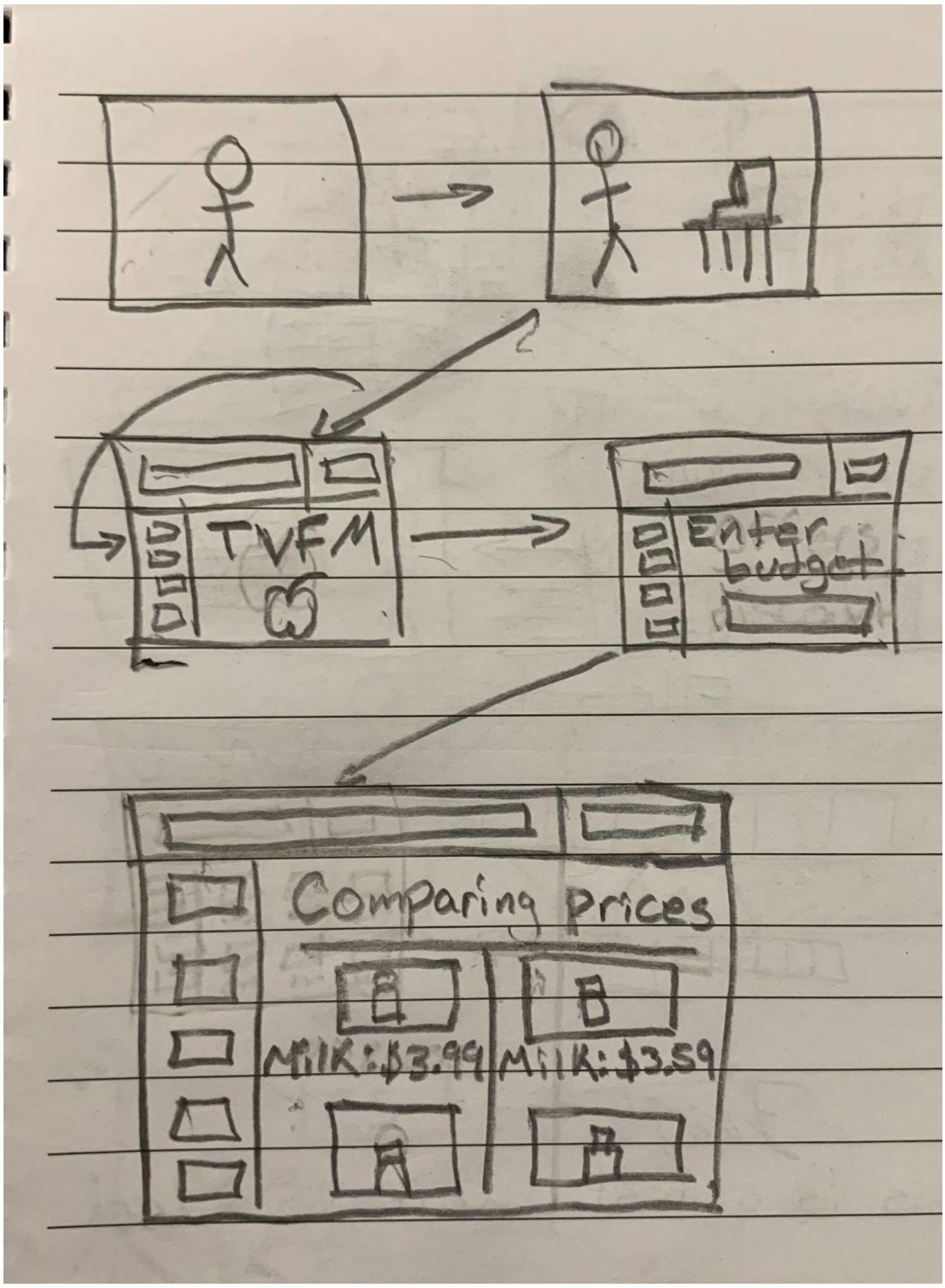
Use Case 6: Small Scale Farmer



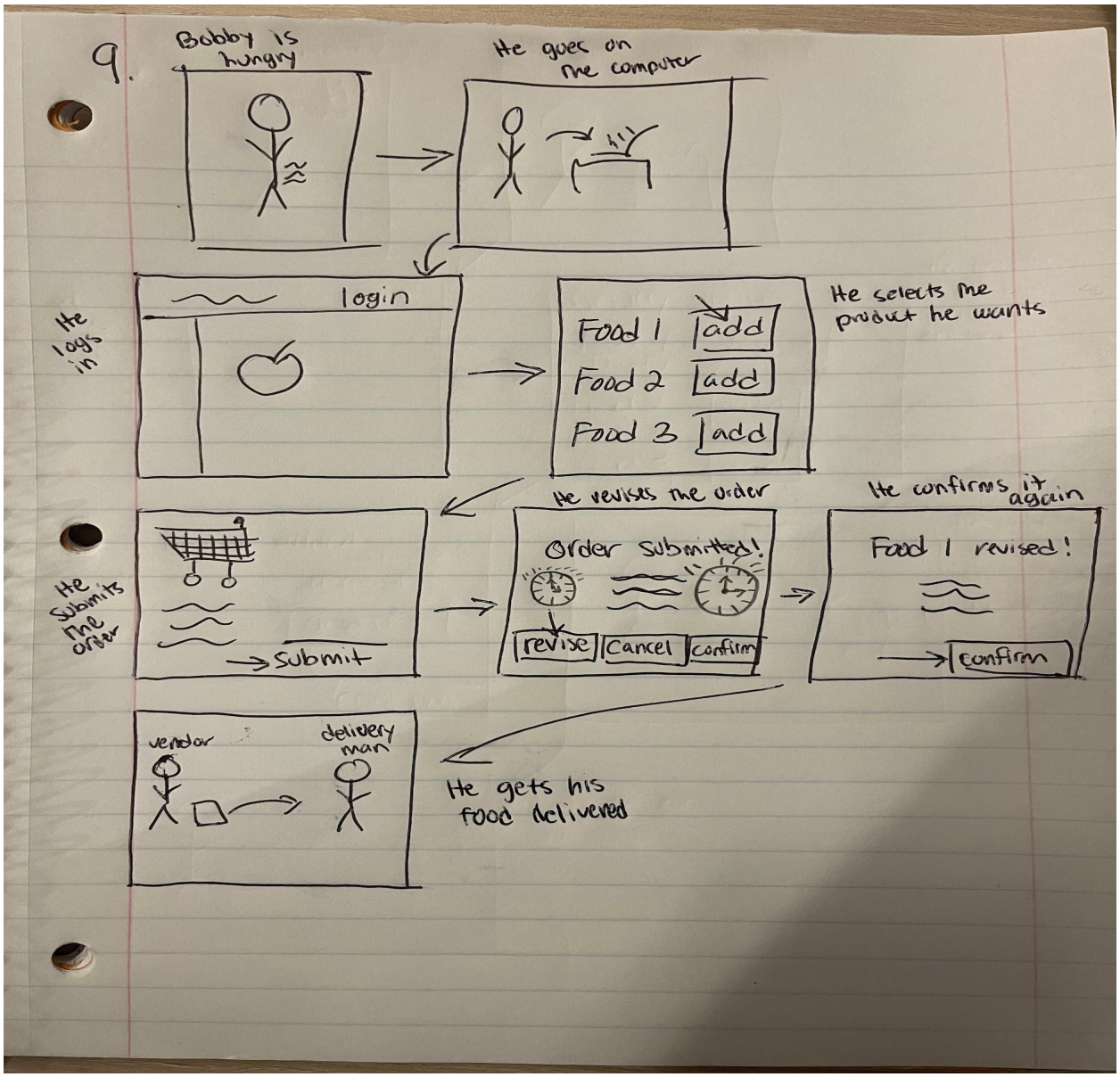
Use Case 7: Quick lunch



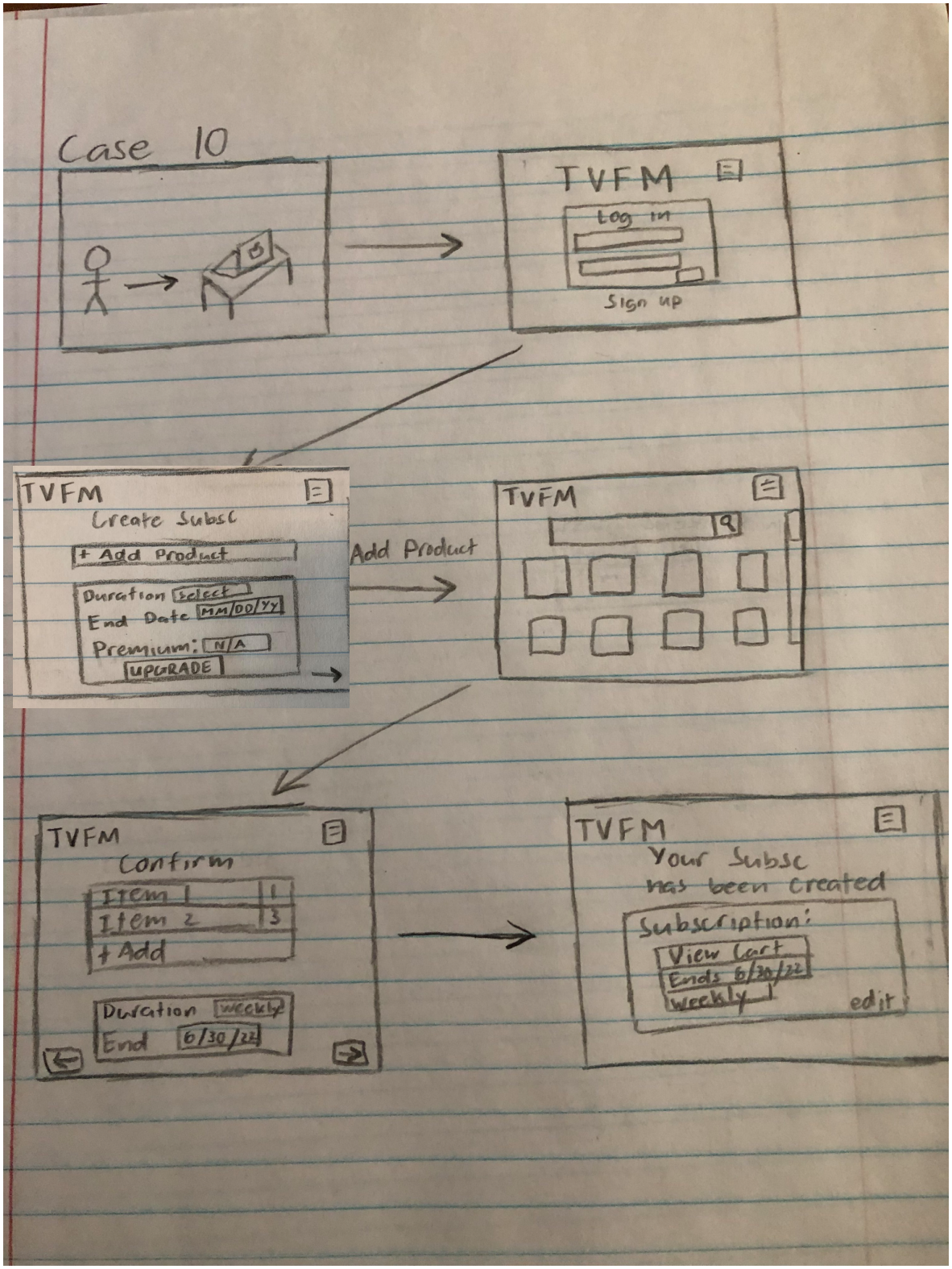
Use Case 8: Affordable produce



Use Case 9: Revising an Order

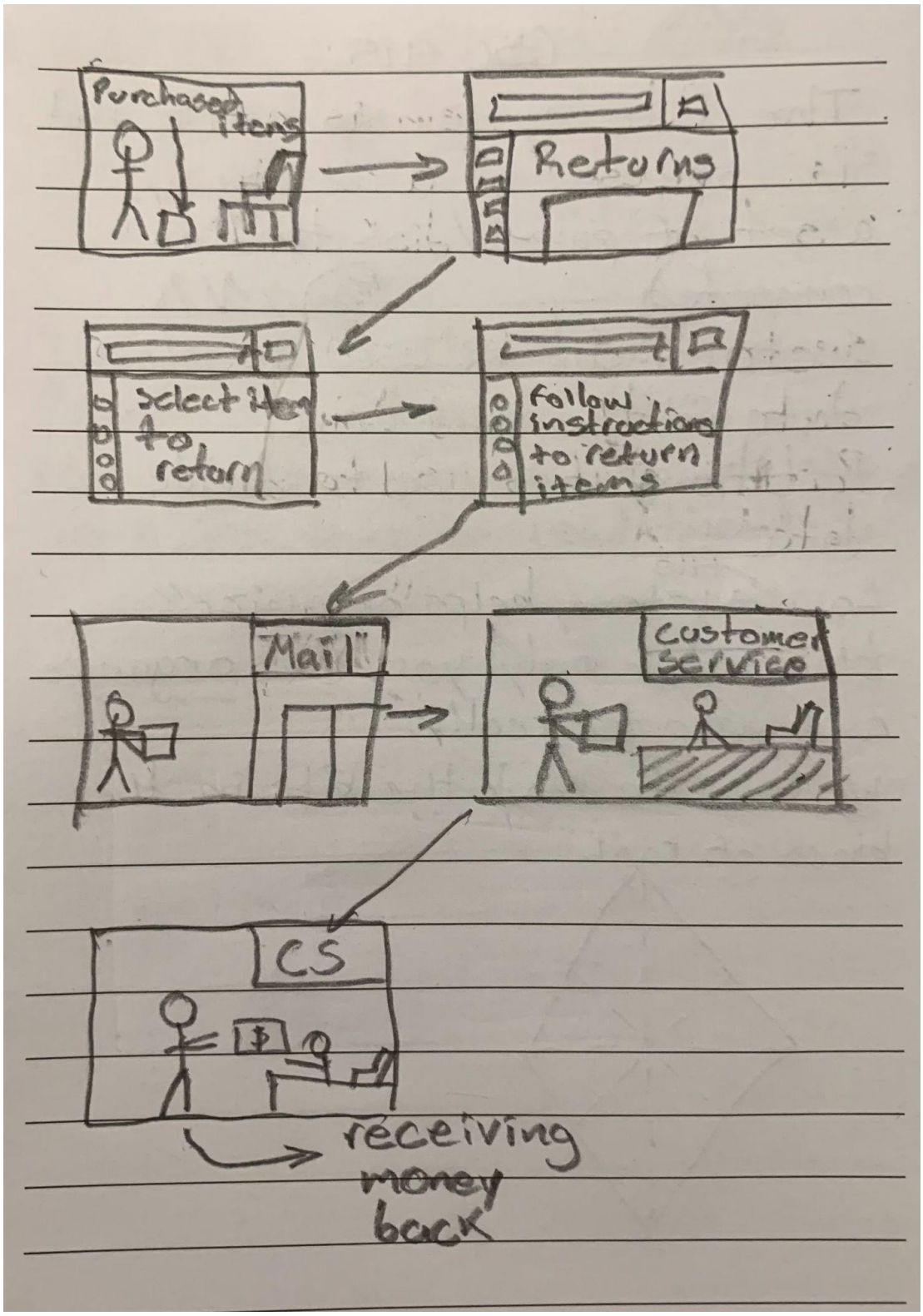


Use Case 10: Catering

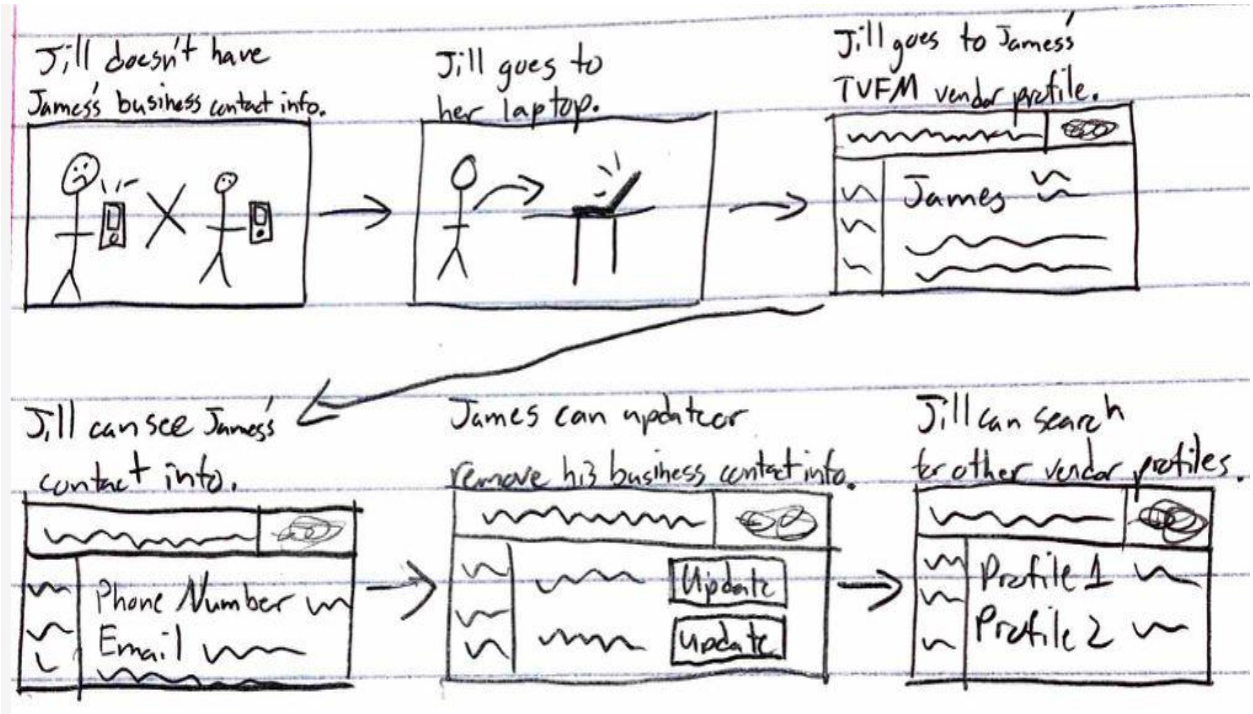


Use Case 11: Meal planning

Use Case 12: Returning items



Use Case 13: Contact Information



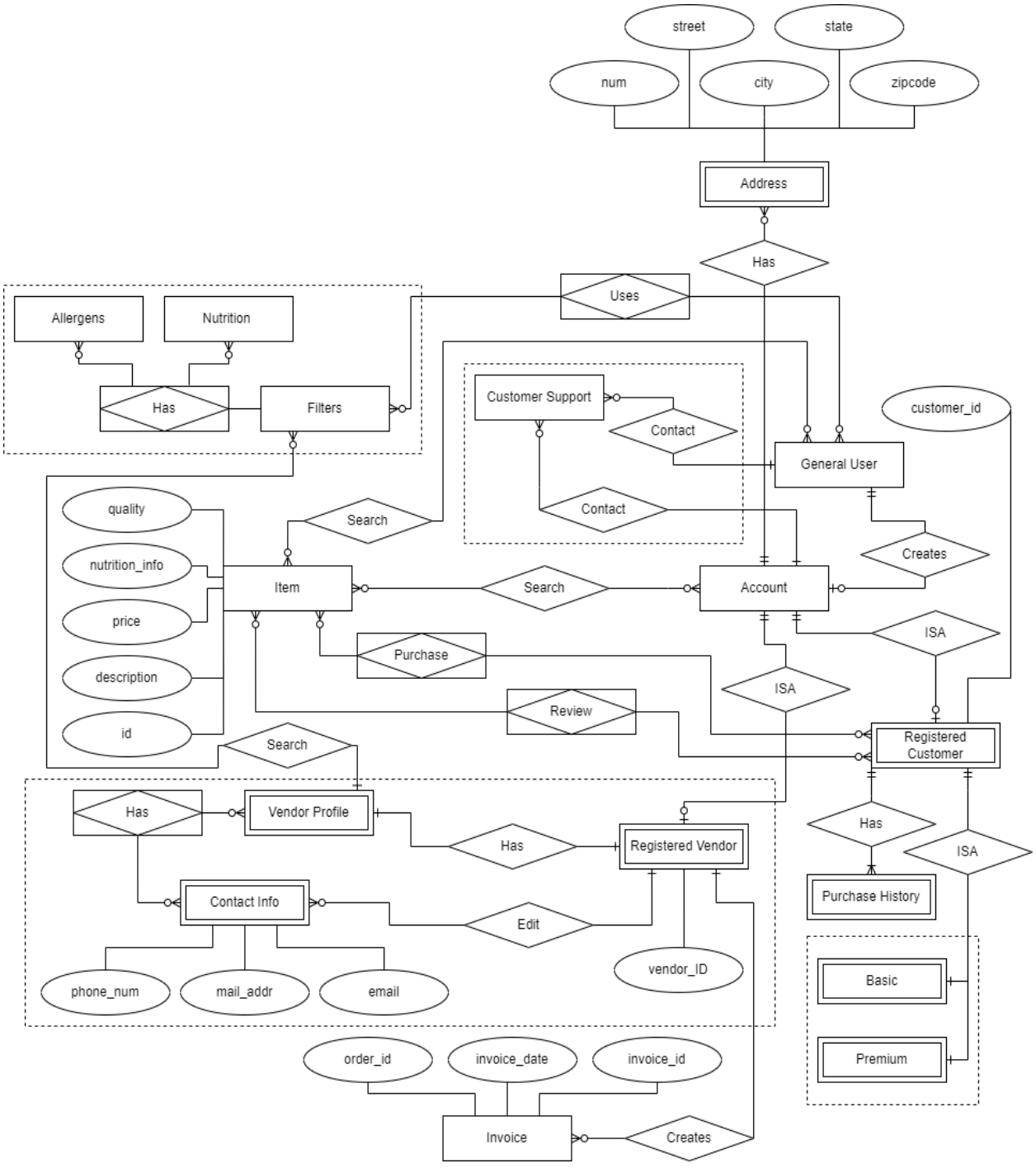
High level Database Architecture and Organization

1. A general user shall be able to enter their data and sign up for our services.
    a. Entities: General User, Account
    b. Relationship: Creates
    c. Abstract Quantitative Properties: One and Only One <-> Zero or One
2. A general user shall be able to set up a registered customer account.
    a. Entities: Account, Registered Customer (Weak Entity)
    b. Relationship: ISA
    c. Abstract Quantitative Properties: One and Only One <-> Zero or One
3. A general user shall be able to set up a registered vendor account which will allow them to sell their items on our site.
    a. Entities: Account, Registered Customer (Weak Entity)
    b. Relationship: ISA
    c. Abstract Quantitative Properties: One and Only One <-> Zero or One
4. A registered customer shall be able to purchase items.
    a. Entities: Registered Customer, Item
    b. Relationship: Purchase (Associative Table)
    c. Abstract Quantitative Properties: Zero to many <-> Zero to many
5. A registered customer shall be able to post reviews for an item.
    a. Entities: Registered Customer, Item

    b. Relationship: Review (Associative Table)
    c. Abstract Quantitative Properties: Zero to many <-> Zero to many
6. A registered customer shall be able to view their purchase history.
    a. Entities: Registered Customer, Purchase History
    b. Relationship: Has
    c. Abstract Quantitative Properties: One and Only One <-> One to many
7. A general user shall be able to contact customer service if they have any issues.
    a. Entities: General User, Customer Support
    b. Relationship: Contact
    c. Abstract Quantitative Properties: Zero to many <-> Zero to many
8. A registered customer shall be able to contact customer service if they have any issues.
    a. Entities: Registered Customer, Customer Support
    b. Relationship: Contact
    c. Abstract Quantitative Properties: Zero to many <-> Zero to many
9. A registered vendor shall be able to edit their contact information.
    a. Entities: Registered Vendor, Contact Information
    b. Relationship: Edit
    c. Abstract Quantitative One <-> Zero to many

10. A general user or a registered customer shall be able to view the nutrition information of an item.
    a. Entities: Filters, Nutritional Information
    b. Relationship: Has (Associative Table)
    c. Abstract Quantitative Properties: Zero to many <-> Zero to many

ERD:


street
state
num
city
zipcode
Address
Has
Uses
Allergens
Nutrition
Has
Filters
Customer Support
Contact
customer_id
General User
Contact
quality
Search
Creates
nutrition_info
Item
Search
Account
price
ISA
Purchase
description
ISA
Review
id
Search
Registered Customer
Has
Vendor Profile
Has
Registered Vendor
Has
ISA
Contact Info
Edit
Purchase History
vendor_ID
Basic
phone_num
mail_addr
email
Premium
order_id
invoice_date
invoice_id
Invoice
Creates

High Level APIs and Main Algorithms

Home Page:

-

Vendor Registry Page:

-

Customer Registry Page:

-

Delivery Registry Page:

-

Login:

- Body should contain email and hashed password.

Search:

- Body should contain filters and what the general user searched for.

Checkout:

- Body should contain all items the user wants.

Vendor Profile:

- Body should contain the vendor name/identifier.

Update Products:

- Body should contain the product identifier and what to change.

Register Vendor:

- Body should contain all attributes of a new vendor.

Add New Product:

- Body should contain products and all attributes.

Revise Order:

- Body should contain an order identifier and what items are removed/changed.

Start Recurring Order:

- Body should contain all items to deliver and how often.

Add Review:

- Body should contain an item identifier and the review.

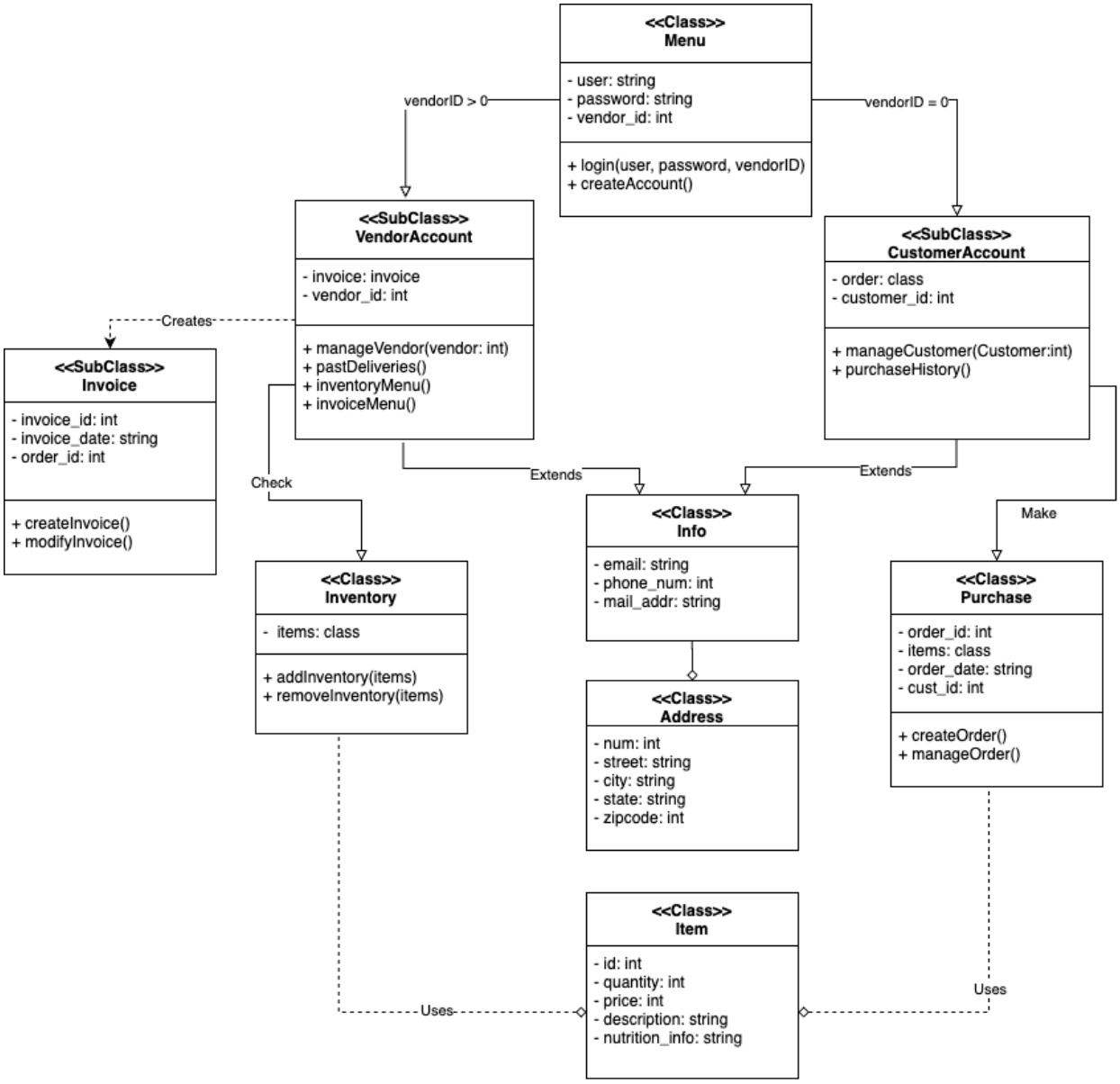
Searching Algorithm:

- Will search for an item within a certain radius of a zip code or address.

Time Estimation Algorithm:

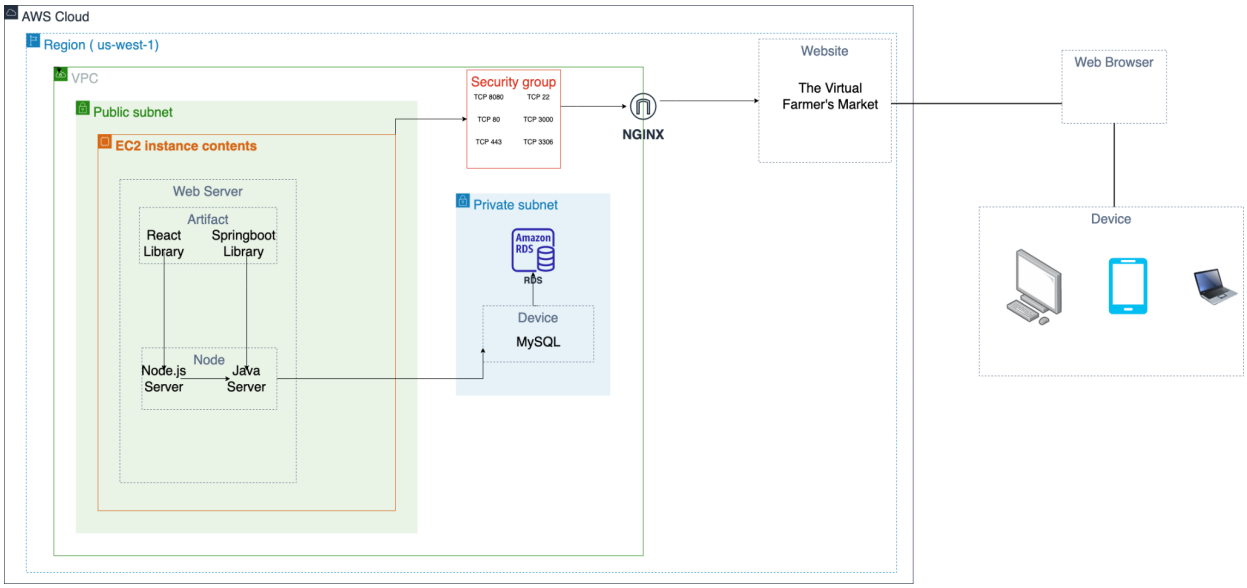
- Will estimate delivery time of an item based on distance and traffic between the two locations (vendor and customer).

High Level UML Diagram

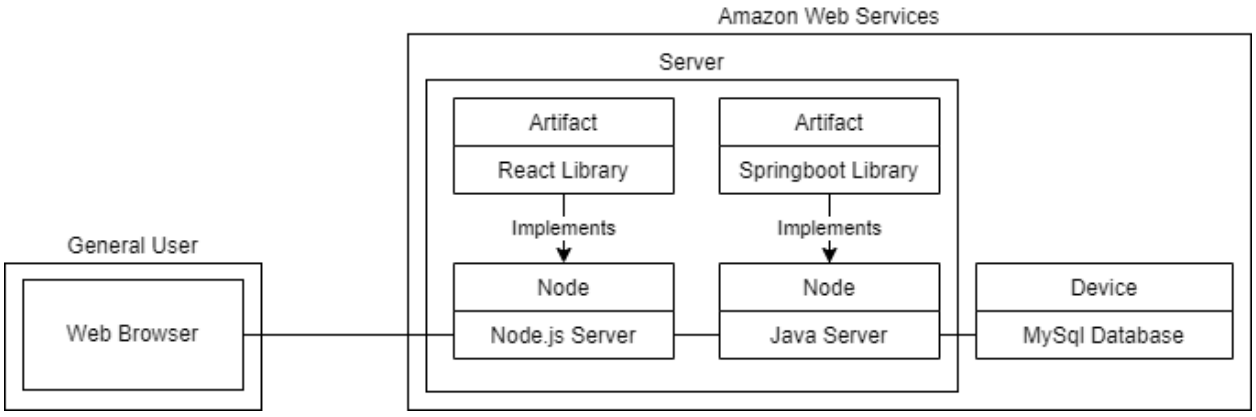


High Level Application Network Diagram and Deployment Diagram

Network Diagram:



Deployment Diagram:



Key Risks

Skill:

- No one in our team has done a UML diagram before. So it is a very new concept to us.
    - We will look at the slides provided in iLearn in order to learn more about UML diagrams.

Schedule:

- There are no known scheduling risks.

Technical:

- There are no known technical risks.

Teamwork:

- There are no known teamwork risks.

Legal/Content:

- There are no known legal/content-related risks.

Project Management

Using Trello I will assign tasks to everyone. Tasks will generally be due by the next meeting. Smaller tasks will be due the day after they are assigned. In the future, I will use Trello to manage tasks and I will use the same due-date system.

We meet every Monday and Friday for about 30 minutes. Also before the due date of the next milestone, we meet on Wednesday.

List of team contributions

Seth Pavlicek:

- Data Definitions
- High Level API
- Deployment Diagram
- UI Mockups: 1, 5, 6

Alex Bjeldanes: 9/10

- UML and ERD Diagrams
- Database Requirements
- UI Mockups: 2, 13
- Milestone 2 Document Cleanup

Armando Partida: 8/10

- UML and ERD Diagrams
- Backend Setup
- UI Mockups: 8, 12

Angel Antunez: 6/10

- Register Page
- UI Mockups: 3, 9

Igor Tsygankov: 9/10

- Application Networks Diagram
- Header for Frontend
- UI Mockups: 7, 11
- AWS Setup

Michael Abolencia: 6/10

- Login Page
- UI Mockups: 4, 10

Milestone 3

Index

Data Definitions

General User:

- Name

Registered Vendor:

- Address
- Email
- Name
- Phone Number
- Items Currently Selling

Registered Customer:

- Address
- First Name
- Last Name
- Email
- Phone Number

Item:

- Vendor Source
- Meal
    - Name
    - Price

        - Ingredients
        - Nutrition
            - Allergens
            - Calories
            - Sugar
            - Carbohydrates
            - Fat
            - Protein
- Produce
    - Name
    - Price
- Packaged Food
    - Name
    - Price
    - Weight
    - Nutrition
        - Allergens
        - Calories
        - Sugar
        - Carbohydrates
        - Fat
        - Protein

Filters:

- Item Category
- Nutrition
    - Allergens
    - Calories
    - Sugar
    - Carbohydrates
    - Fat
    - Protein

Recurring Delivery:

- Address to deliver to
- Items to deliver
- Frequency
- Duration
    - Seasonal

Address:

- Street Number
- Street
- City
- County
- State

- Zip code

Prioritized Functional Requirements

Priority 1:

1. Registration:

    1.1 A general user shall be able to enter their data and sign up for our services.

2. Registered Customer Account:

    2.1 A general user shall be able to set up a registered customer account.

    2.2 A registered customer shall only purchase items.

3. Registered Vendor Account:

    3.1 A general user shall be able to set up a registered vendor account.

    3.2 A registered vendor account shall only sell items.

4. Search:

    4.1 A general user or registered customer shall be able to search for items.

    4.2 A general user or registered customer shall be able to search for registered vendor accounts.

5. Filtering:

    5.1 A general user or registered customer shall be able to use filters when searching to narrow down their selection of items.

7. Purchase:

7.1 A general user or a registered customer account shall be able to purchase items from registered vendor accounts.

9. Recurring Deliveries:

9.1 A registered customer account shall be able to create a recurring delivery in order to receive items at given time intervals.

9.2 A registered customer account shall be able to cancel their recurring delivery.

10. Revising an Order:

10.1 A general user or a registered customer shall be able to change an order within a certain time frame.

11. Contacting Vendors:

11.1 A general user or registered customer shall be able to access a registered vendor's contact information to ask about their items.

13. Nutrition Information:

13.1 A general user or a registered customer shall be able to view the nutrition information of an item.

15. Price Comparison:

15.1 A general user or a registered customer shall be able to compare the prices of the same item sold by different registered vendors.

16. Vendor Proximity:

16.1 A general user or a registered customer shall be able to search for registered vendors closest to a given address

17. Vendor Pricing:

17.1 A registered vendor shall be able to set the price for the items they are selling.

19. Previous Purchases:

19.1 A registered customer shall be able to view their purchase history.

20. Alternatives:

20.1 A general user or a registered customer shall be able to pick a different registered vendor if their first choice ran out of stock.

23. Editing Customer Information:

23.1 A registered customer shall be able to edit their information.

24. Total Price:

24.1 A general user or a registered customer shall be able to see the total price of all the items in their shopping cart before finalizing the transaction.

26. Sort:

26.1 A general user or a registered customer shall be able to sort their search results based on given criteria.

27. Tracking ID:

27.1 A registered customer shall be given a tracking ID for their orders.

33. Cart:

33.1 A general user or a registered customer shall be able to see items they added to their cart.

33.2 A general user or a registered customer shall be able to add and remove items or change the quantity of an item in their cart.

35. Delivery/Pickup:

35.1 A general user or a registered customer shall be able to pick up or have their items delivered to them.

38. Item Description:

38.1 A general user or a registered customer shall be able to read a clear description of an item.

41. Login/Logout:

41.1 A general user shall be able to login and logout of their registered customer and registered vendor accounts.

42. Editing Vendor Information:

42.1 A registered vendor shall be able to edit their business contact information.

43. Selecting a Vendor:

43.1 A general user or a registered customer shall be able to select a registered vendor that they will solely buy items from.

Priority 2:

6. Reviews:

6.1 A registered customer shall be able to post reviews for an item.

14. Discounts:

14.1 A general user or a registered customer shall be able to view discounts for certain items.

14.2 A registered vendor shall be able to provide discounts for their items.

21. Recommended Products:

21.1 A general user or a registered customer shall be recommended items that may interest them based on previous purchases.

22. Following a Vendor:

22.1 A general user or a registered customer shall be able to follow a registered vendor for updates about a vendor's items.

25. Discounting Bulk Purchases:

25.1 A general user or a registered customer shall receive a discount if the total price of items in their cart exceeds a certain amount.

31. Customer Service:

31.1 A general user or a registered customer shall be able to contact customer service if they have any issues.

37. Rewards:

37.1 A general user or a registered customer shall be able to see the rewards page.

37.2 A general user or a registered customer shall receive rewards points for every purchase.

Priority 3:

8. Returns:

8.1 A general user or a registered customer shall be able to return their purchased items within a certain time.

12. Newsletter:

12.1 A general user or a registered customer shall receive a newsletter notifying them about deals, new products, and popular vendors.

12.2 A general user or a registered customer shall opt-in to the newsletter and be able to opt-out at any time.

29. Careers:

29.1 A general user shall be able to apply for working positions at TVFM.

30. Social Media:

30.1 Anyone shall be able to see offers, features, discounts, events on the social media account of TVFM.

39. Advertisement:

39.1 A general user or a registered customer shall receive advertisements about items they may be interested in.

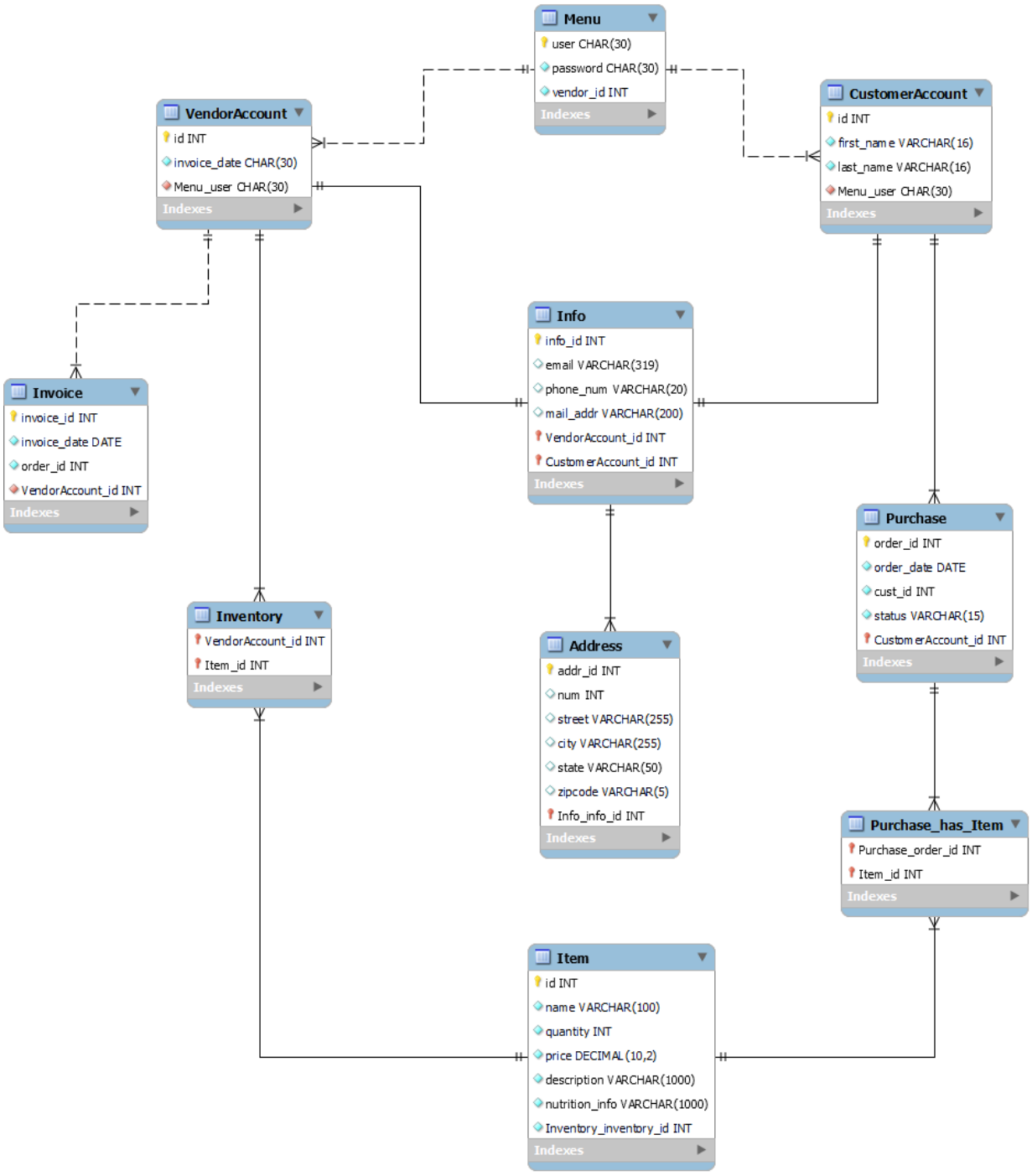
39.2 A registered vendor shall be able to pay to have an advertisement about their item shown to a general user or a registered customer.

Wireframes Based on your Mockups/Storyboards

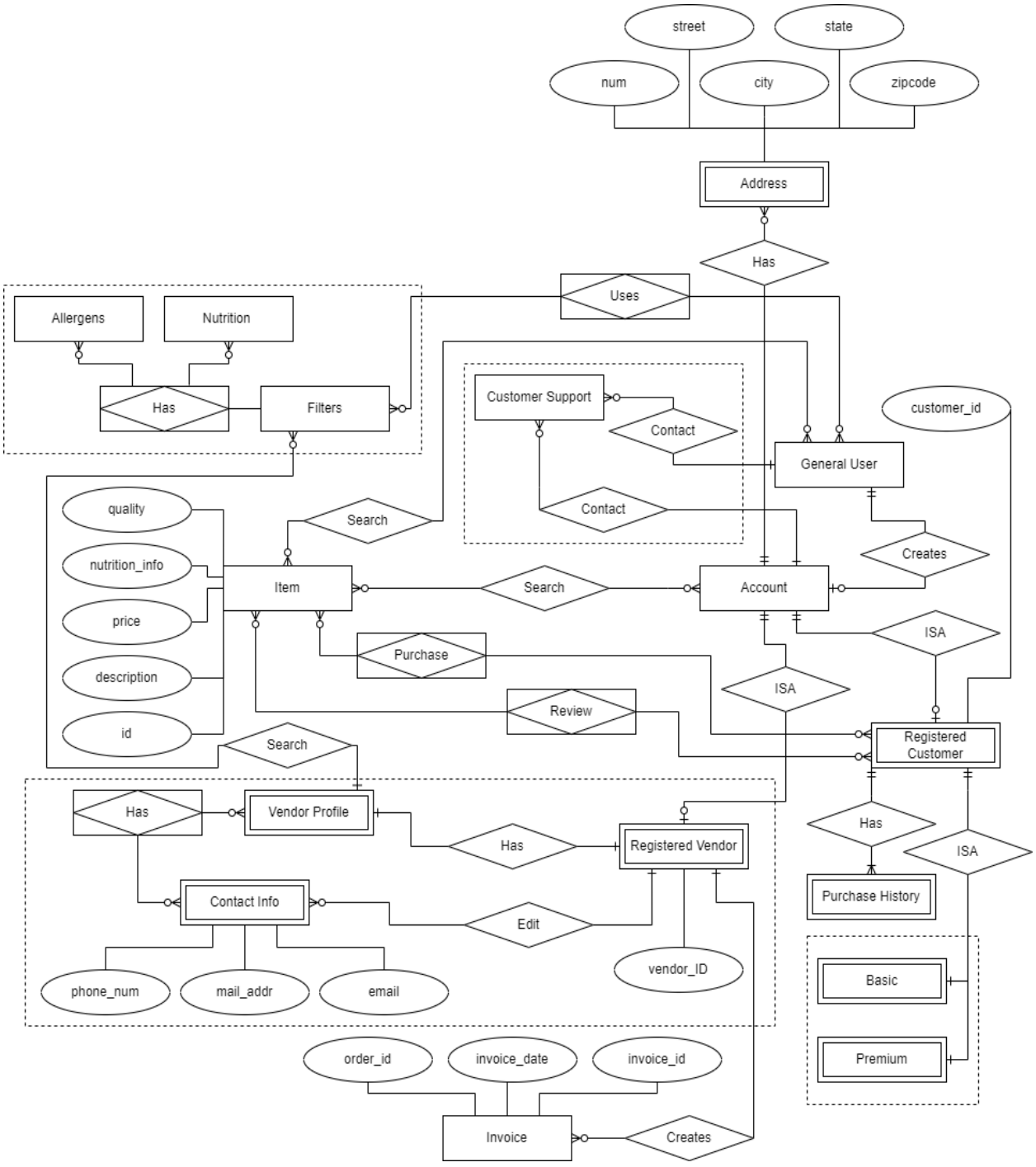
https://www.figma.com/file/4I7PluemvtUdCm3O3YVvAM/TVFM-WireFrames?node-id=0%3A1

High level database Architecture and Organization

EER: MySql

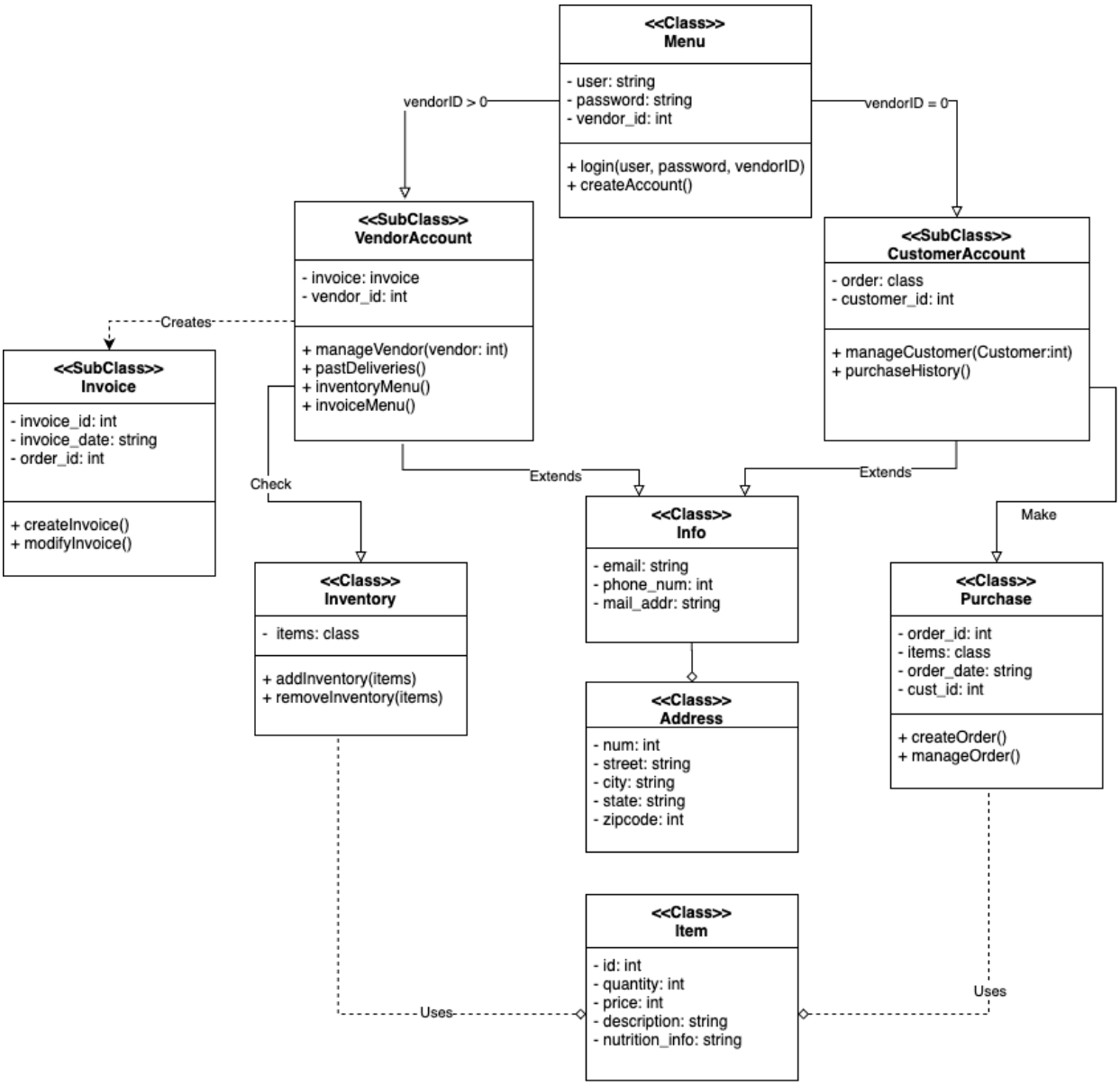


ERD: MySql

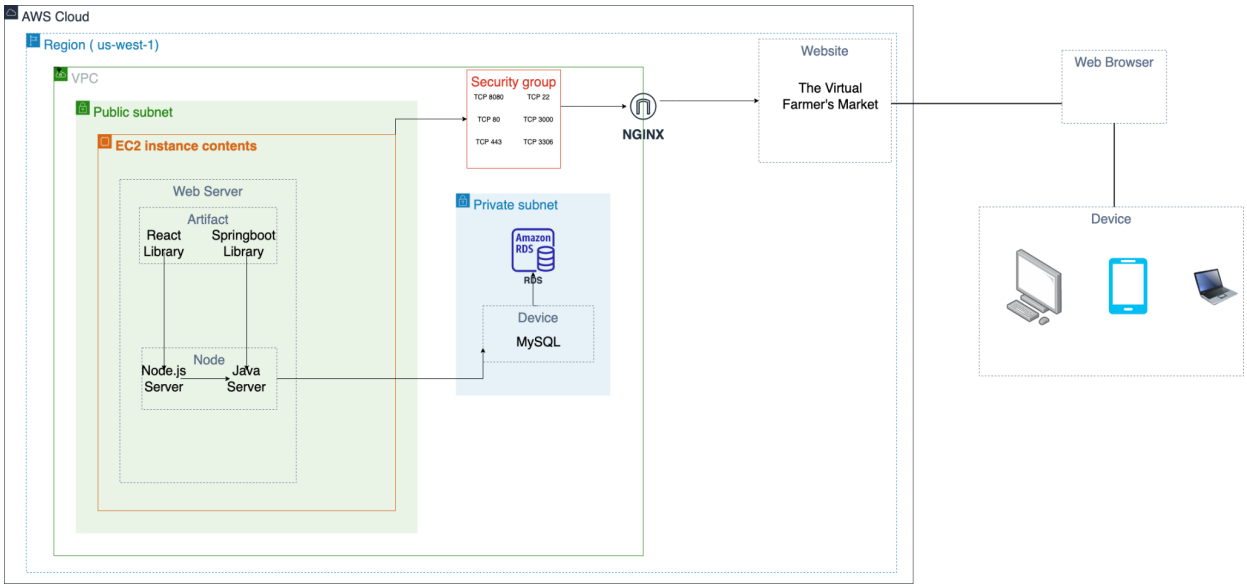

street
state
num
city
zipcode
Address
Has
Uses
Allergens
Nutrition
Has
Filters
Customer Support
Contact
customer_id
General User
Contact
quality
Search
nutrition_info
Item
Search
Account
Creates
price
Purchase
ISA
description
ISA
Review
id
Search
Registered Customer
Has
Vendor Profile
Has
Registered Vendor
Has
ISA
Contact Info
Edit
Purchase History
vendor_ID
Basic
phone_num
mail_addr
email
Premium
order_id
invoice_date
invoice_id
Invoice
Creates

High-Level Diagrams

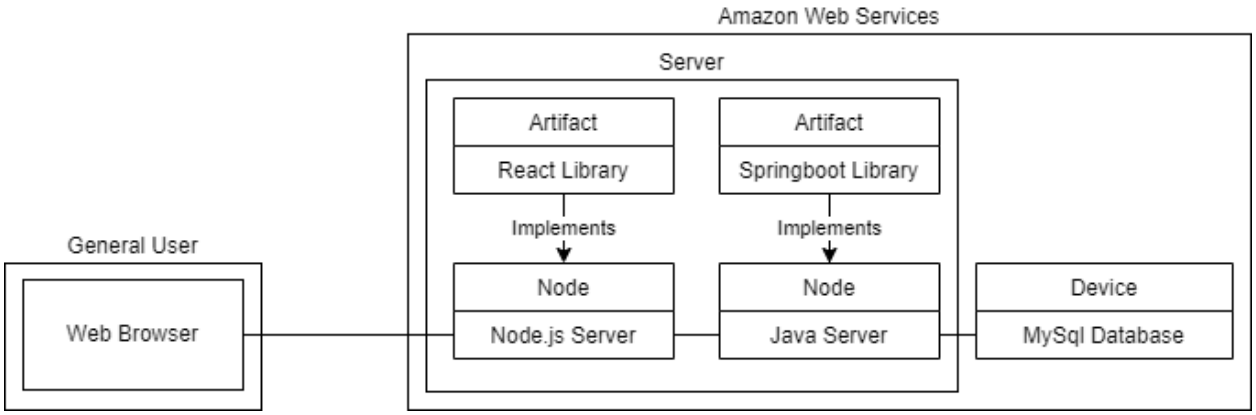
UML Diagram:



Network Diagram:



Deployment Diagram:



Detailed List of Contributions

Seth Pavlicek:

- Header Wireframe
- Homepage Wireframe
- Recurring Delivery Wireframe
- Home Page Hook

Alex Bjeldanes: 9/10

- EER Diagram

Armando Partida: 9/10

- Backend Classes from UML
- Initial Search Query

Angel Antunez: 9/10

- Footer in Frontend
- Search Page
- Cart Wireframe
- Cart Page

Igor Tsygankov: 10/10

- Header in Frontend
- Settings Wireframe
- Filter Wireframe
- Footer Wireframe
- Search Page Filters
- Recurring Delivery Page

Michael Abolencia: 9/10

- Register Pages
- Register Wireframes
- Product Information Page
- Privacy Policy Page

Milestone 4

Index

Product Summary

Name: The Virtual Farmers

Priority 1:

1. Registration:

    1.1 A general user shall be able to enter their data and sign up for our services.

2. Registered Customer Account:

    2.1 A general user shall be able to set up a registered customer account.

    2.2 A registered customer shall only purchase items.

3. Registered Vendor Account:

    3.1 A general user shall be able to set up a registered vendor account.

    3.2 A registered vendor account shall only sell items.

4. Search:

    4.1 A general user or registered customer shall be able to search for items.

    4.2 A general user or registered customer shall be able to search for registered vendor accounts.

5. Filtering:

    5.1 A general user or registered customer shall be able to use filters when searching to narrow down their selection of items.

7. Purchase:

    7.1 A general user or a registered customer account shall be able to purchase items from registered vendor accounts.

9. Recurring Deliveries:

9.1 A registered customer account shall be able to create a recurring delivery in order to receive items at given time intervals.

9.2 A registered customer account shall be able to cancel their recurring delivery.

10. Revising an Order:

10.1 A general user or a registered customer shall be able to change an order within a certain time frame.

11. Contacting Vendors:

11.1 A general user or registered customer shall be able to access a registered vendor's contact information to ask about their items.

13. Nutrition Information:

13.1 A general user or a registered customer shall be able to view the nutrition information of an item.

15. Price Comparison:

15.1 A general user or a registered customer shall be able to compare the prices of the same item sold by different registered vendors.

16. Vendor Proximity:

16.1 A general user or a registered customer shall be able to search for registered vendors closest to a given address

17. Vendor Pricing:

17.1 A registered vendor shall be able to set the price for the items they are selling.

19. Previous Purchases:

19.1 A registered customer shall be able to view their purchase history.

20. Alternatives:

20.1 A general user or a registered customer shall be able to pick a different registered vendor if their first choice ran out of stock.

23. Editing Customer Information:

23.1 A registered customer shall be able to edit their information.

24. Total Price:

24.1 A general user or a registered customer shall be able to see the total price of all the items in their shopping cart before finalizing the transaction.

26. Sort:

26.1 A general user or a registered customer shall be able to sort their search results based on given criteria.

27. Tracking ID:

27.1 A registered customer shall be given a tracking ID for their orders.

33. Cart:

33.1 A general user or a registered customer shall be able to see items they added to their cart.

33.2 A general user or a registered customer shall be able to add and remove items or change the quantity of an item in their cart.

35. Delivery/Pickup:

35.1 A general user or a registered customer shall be able to pick up or have their items delivered to them.

38. Item Description:

38.1 A general user or a registered customer shall be able to read a clear description of an item.

41. Login/Logout:

41.1 A general user shall be able to login and logout of their registered customer and registered vendor accounts.

42. Editing Vendor Information:

42.1 A registered vendor shall be able to edit their business contact information.

43. Selecting a Vendor:

43.1 A general user or a registered customer shall be able to select a registered vendor that they will solely buy items from.

Unique about the product:

Our Product differs from competing products in a few ways. First and foremost we strived for a product that was simple to navigate in order to prevent any clients from being overwhelmed looking for any of our services. Everything that a client would need access to in order to use our site will not take more than a few seconds of searching nor will said components be hidden in unintuitive places. Everything is clearly labeled with nothing obscuring a user's visibility. One advantage of our product is that we allow customers the option to filter out our content based on dietary restrictions and personal tastes which results in a faster, and safer time searching for the desired products hosted to our site. Adding on to this, clients can either search by a product's name, or they can search by using an address in order to help them locate nearby vendors. These filters will always be present during the search so that customers can access them at any point during their search. All searches will be conducted using the same search bar in order to prevent any unnecessary confusion. On the other hand, we made sure that joining our site as a vendor is as simple as joining as a customer. Vendors will be able to add and sell their products on our site. All users will be able to make informed decisions about their purchases by viewing an item's description. In addition, vendor contact information will be readily accessible for our users should they wish to make inquiries about vendor stocks and products.

Link to product:

http://ec2-13-52-221-26.us-west-1.compute.amazonaws.com/

Usability test plan

Order food

This is testing how easy it is to order food from vendors from our website. We are testing this because it is one of the main functions of our website. This should test many components such as the cart and the search bar. This is intended for anyone who wants to purchase food from their home. The tester should start from the home page. The user may decide to register and login but that is unnecessary. With or without logging in, the tester should start searching for food they want to eat. Then they should add food to their cart. When they decide that they have found enough food they should go to their cart and check out their food.

http://ec2-13-52-221-26.us-west-1.compute.amazonaws.com/

This test is going to measure the effectiveness and efficiency of the cart and checkout processes. The search bar will be tested more thoroughly in the search test.

Search

This test is to ensure that anyone is able to search for anything that is related to the site. This is to include unregistered users, customers and vendors. The things related to the site that can be searched for will include products, customers, vendors, and certain pages from the site. Anyone should be able to search on any page and be redirected to the physical search page on the site and the page will display everything that is related with what was searched, if it is

in the system. If nothing is found, it will notify the user with a message that nothing was found and will give suggestions related to what was searched.

Add Item (Vendor)

This test is meant to show us how much effort it takes a vendor to add an item to our system. Since our product is an online commerce platform, it is important for vendors to be able to add their products otherwise there will be nothing for customers to purchase. This is intended for vendors that would like to see their products on our website. From the Homepage, A vendor will login to their account. If they do not have one, then they will register and sign in. Once they are signed in, then the vendor will navigate to their settings page and add an item to their inventory.

Revise Order

Testing ‘revising an order’ is important since customers should be able to add products to the cart and should be able to view what they have in it, in case they want to remove what they do not want or if they want more of that product. This test is simple, the user should be able to see the products he/she added, also the quantity of the product, and the price before going to the checkout page. We are testing this because it is really important since if the customer is currently in this part of the process to buy our products, she/he should be assured of what they are buying.

Objectives: Ensure that customers can view the items that they are going to order or what they ordered.

Setup: When they are in the checkout, the customer should be able to see the items they are going to order. The database should save the order of the customer when they make the purchase so when they want to check their order, the system can retrieve the order.

The intended users would be anyone who wants to buy from our webpage, it can be an adult, student, etc.

Change Personal Information:

Testing the change personal information function is very important for this project since customers and vendors should be able to easily change their information if needed. This function being fully implemented and working will make sure that purchases and deliveries are made to the right people. The testing for this function is very simple, once a customer or vendor is logged in, they should be able to navigate to the settings page where they will have the option to edit account information. We want to test this because account information is a big part of this project, it fully working will show that our page is able to process new information and update accordingly.

Effectiveness table:

Number	Test case	Description	Errors	Comments
1	Order food	The user is able to order food.	No errors.	No confirmation of order.
2	Search	The user is able to search for food	Picture did not load.	Picture did not load correctly, a few items. No dollar sign.
3	Add Item (Vendor)	The user is able to add items for sale	Page not linked correctly	Not clear how to add items for sale.
4	Revise Order	The user is able to change their order before submitting	No errors	Cart removes items.
5	Change Personal Information	The user is able to change their personal information	Page not linked.	Cannot go to settings.

Efficiency table:

Number	Test case	Description	Time completed
1	Order food	How quickly was the user able to order food	~10 seconds
2	Search	How quickly the user was able to search for food.	~5 seconds
3	Add Item (Vendor)	How quickly the user was able to add an item for sale	N/A
4	Revise Order	How quickly the	~1 second

		user was able to revise their order	
5	Change Personal Information	How quickly the user was able to change their personal information	N/A

User satisfaction:

Fill in the box that most closely describes how you felt regarding each statement.

	Strongly Disagree				Strongly Agree
I would use this site over other websites that serve similar services.	1 ✓	2	3	4	5
It was simple to navigate the website.	1	2 ✓	3	4	5
It was easy to find the products I was looking for.	1	2 ✓	3	4	5
The website was consistent between different pages.	1	2 ✓	3	4	5
I would recommend this website to other people wanting food deliveries.	1 ✓	2	3	4	5
It was easy to filter out certain products I did not want.	1	2 ✓	3	4	5
It was easy to find a vendor's contact information.	1 ✓	2	3	4	5
It was easy to modify my order before purchasing it.	1	2	3	4	5 ✓
It was easy to see my previous orders.	1 ✓	2	3	4	5
It was simple and quick to create a new account.	1	2 ✓	3	4	5

QA test plan

Home Page shall load in at most 500 ms

The time that it takes to load the webpage is critical since it is our first step to hook the user to the page. We want to make sure that when we open the page, it does not take too long to open since the customer can leave the page. We need to focus on this because we can lose a lot of customers. The test is just simple, open the page and measure the time it takes to load and improve that load time.

Objectives: Make sure the home page is optimized so it can open in the shortest possible time.

Setup: look at the processes the page is doing at the time the user is loading the web page and ensure that it does not use more processes than needed to load the page.

The intended users would be anyone who wants to buy from our webpage, it can be an adult, student, etc.

User satisfaction:

QA test plan

Number	Description	Test Input	Expected Output	Pass/Fail
1	The web page should load in at most 500 ms	http://ec2-13-52-221-26.us-west-1.compute.	The home page of our website should load in at most 500 ms	pass

		amazonaws.com/		

User’s Password shall be encrypted

Objectives: Ensure that passwords stored in the database are encrypted

Setup: Register a user by directly contacting the backend through postman or by using the frontend. Then connect to the database and look at the passwords for the newly registered user. If the password was encrypted it should not look the same as the entered password.

Tested Feature: Password Encryption

QA Test Plan:

Number	Description	Test Input	Expected Output	Pass/Fail
1	Password encryption test for: “”	“”	d41d8cd98f00b204e9800998ecf8427e	pass
2	Password encryption test for: “short”	“short”	4f09daa9d95bcb166a302407a0e0babe	pass
3	Password encryption test for: “wHat)_anawsomePassw0rd”	“wHat)_anawsomePassw0rd”	b997419ac83d8299fc2426d97eb62f6f	pass

System will lock the user out after a specific number of login attempts.

Test objectives: To ensure that the system will react to excessive login attempts by a given user.

Setup: The user will be on the login/out page for this test. The user will interact with the frontend and will interact with the backend to process the request. The backend will interact with the database and check if the given username and password are in the database. If they aren't, then the login attempt has failed. If there is a match, then the system will begin processing the login.

Number	Test case	Description	Errors	Comments
1	User locked out of login	The website should disable after a certain amount of logins	None	Website correctly locked user out
2	User can login on first attempt	The website allows the user to login on first attempt	None	Website allowed user to login on first attempt with correct credentials
3	Credentials are correct/incorrect	The website should allow the user to know if their credentials are correct/incorrect	None	Website correctly informed the user of credential status.

Features to be tested: Login attempt limit

QA Test plan:

Number	Description	Test Input	Expected Output	Pass/Fail
1	System shall lock the user's account after 3 attempts.	Incorrect login credentials into the login page.	Will not allow the user to attempt another login.	Pass
2	System shall not lock the user out after the first login attempt.	Correct login credentials into the login page.	Login successful	Pass
3	Users shall know if their credentials are incorrect.	Incorrect login credentials into the login page.	Login attempt failed, please try again.	Pass

Current and previous version of Google Chrome

This test will check if the web page can be run in any versions of google so the customer will not have to worry to get the correct version of google to use or web page.
Objectives: Ensure that customers can view the web page in different google versions.
Setup: We will open the web page in the current version of google and record if it loads correctly and then we will open the web page in the previous version of google and record if it loads without problems.
The intended users would be anyone who wants to buy from our webpage, it can be an adult, student, etc.

Number	Test case	Description	Errors	Comments
1	Google current version	The website opens in the current version of google	None	The website loads as expected.
2	Google previous version	The website opens in the previous version of google	The website may not load correctly or may have display errors.	Users may need to update their browser before viewing the website properly.
3	Other browser current version	The website opens in the current version of the other browser	The website may not load correctly or may have display errors.	If not fully compatible, users should switch to google chrome.

User satisfaction:

QA test plan:

Number	Description	Steps	Expected Output	Status

1	The web page should load in the current version of google	Revise if the current version of google is installed. Open the web page.	The web page should load the home page without problems	Pass
2	The web page should load in the previous version of google	Revise if the previous version of google is installed. Open the web page.	The web page should load the home page without problems	Pass
3	The web page should load in other browsers	Open the web page using another browser	The web page should load the home page without problems	Pass

User’s personal information to be kept confidential and not distributed

This test is very important since we will save the credentials of the people and we want to have them confidential. The test is simple, we are going to receive the credentials of a user, in this case an email and a password, and we are going to hash them so in the database they will not be recognizable to the user.

Objectives: Ensure that the customers credentials are secured in the database.

Setup: When a user is registering to our web page we want to send the credentials to a hashing function and return an unrecognizable string before storing those strings in the database.

The intended users would be anyone who wants to buy from our webpage, it can be an adult, student, etc.

Number	Description	Steps	Expected Output	Status
1	Hash the user email	Send the user’s email to the hashing function.	Return an unrecognizable string	Pass
2	Hash the user password	Send the user’s password to the hashing function.	Return an unrecognizable string	Pass
3	Store the unrecognizable string in the database	Send the string to the database with the corresponding user information	In the database, you should see the string of the hash function	Pass

Code Review

Code Review for TvfmAPI.java

Overall Comments

The code looks good overall. The variable names are meaningful and the code is organized properly. Additionally, the various sections of the code are easy to understand. The readability of the code is also good, I see functions being reused. However, there are some minor issues related to code commenting, exception handling, removing unwanted code, etc. See below for a detailed review

Detailed Review

```
1  package com.Team03.TVFM.api;
2
3
4  import com.Team03.TVFM.model.*;
5  import org.springframework.beans.factory.annotation.Autowired;
6  import org.springframework.util.Assert;
7  import org.springframework.web.bind.annotation.*;
8  import org.springframework.web.servlet.view.RedirectView;
9
10 import java.util.LinkedList;
11 import java.util.List;
12 import java.util.Random;
13 import java.time.format.DateTimeFormatter;
14 import java.util.Date;
15
16 import java.security.MessageDigest;
17 import java.security.NoSuchAlgorithmException;
18 import java.text.SimpleDateFormat;
19
20 import javax.persistence.Query;
21 import javax.transaction.Transactional;
22
```

It’s great that you have separated imports from the package definition. One improvement that can be made here is to sort the imports.

```
22
23  @RestController
24  @RequestMapping("/")
25  public class TvfmAPI {
26
27      @Autowired
28      private CustomerRepo CustomerRepo;
29      @Autowired
30      private VendorRepository VendorRepository;
31      @Autowired
32      private ItemsRepo ItemsRepo;
33      @Autowired
34      private PurchasesRepo PurchasesRepo;
35
36      private List<Customer> isCustomer;
37      private List<Vendor> isVendor;
38      Vendor loginVendor = new Vendor();
39      Customer loginCustomer = new Customer();
40
```

No major concerns about this. I am assuming the properties defined from line 36 to 39 are for test purposes only - it's usually not a good idea to have data shared among endpoints.

```
42      //------Simple password hashing-------//
43
44      public String passwordHashing(String password){
45          String passwordToHash = password;
46          String generatedPassword = null;
47
48          try {
49              MessageDigest md = MessageDigest.getInstance("MD5");
50              md.update(passwordToHash.getBytes());
51              byte[] bytes = md.digest();
52
53              StringBuilder sb = new StringBuilder();
54              for (int i = 0; i < bytes.length; i++) {
55                  sb.append(Integer.toString((bytes[i] & 0xff) + 0x100, 16
                        ).substring(1));
56              }
57
58            generatedPassword = sb.toString();
59          } catch (NoSuchAlgorithmException e) {
60          e.printStackTrace();
61          }
62          //System.out.println(generatedPassword);
63
64          return generatedPassword;
65      }
```

You can remove the commented-out code. Instead of printing the stack trace, it’s usually better to raise the exception and handle it in the endpoint code returning a proper response from the server.

```
69      @PostMapping(value = "/registration")
70      public String registration(@RequestBody Registration user){
71          String status = "registration failed";
72
73          String newPassword = passwordHashing(user.getPassword());
74          String hashEmail = passwordHashing(user.getEmail());
75
76          if(user.getVendor() == 0){
77              Customer customer = new Customer();
78
79              customer.setId(user.getId());
80              customer.setEmail(hashEmail);
81              customer.setPassword(newPassword);
82              customer.setName(user.getName());
83              customer.setLastname(user.getLastname());
84
85              CustomerRepo.save(customer);
86              status = "registered as a customer";
87          }
```

It’s usually better to have your constants defined in a special module or at the top of the time.

```
105     @Transactional
106     @GetMapping(value = "/search")
107     public List<Items> searchItem(@RequestBody Items item){
108         List<Items> itemRepoHolder;
109 
110         if(item.getId() instanceof Long){
111             itemRepoHolder = ItemsRepo.findItembyID(item.getId());  //to
                    look with an id
112         }
113         else if(item.getName() instanceof String){
114             itemRepoHolder = ItemsRepo.findItem(item.getName());  //to
                    look with a name, description, nutrition
115         }
116         else{
117             System.out.println("Item not found");
118             itemRepoHolder = null;
119         }
120 
121         //for a recommendation, may have to change it later
122         if(itemRepoHolder.isEmpty()){
123             Random rand = new Random();
124             int num = rand.nextInt((ItemsRepo.maxID() - 1) + 1) + 1;
125 
126             itemRepoHolder = ItemsRepo.findItembyID(num);
127         }
```

On line 124, it would be better to have a comment that explains this seemingly weird computation.

```
135     @PostMapping(value = "/login")
136     public String login(@RequestBody Registration user){
137         String status;
138 
139         String getPassword = passwordHashing(user.getPassword());
140         String hashEmail = passwordHashing(user.getEmail());
141 
142         isCustomer = CustomerRepo.findCustomer(hashEmail, getPassword);
143         isVendor = VendorRepository.findVendor(hashEmail, getPassword);
144 
145         if(!(isCustomer.isEmpty())) {
146             status = "login as a customer succesful";
147             loginCustomer.setEmail(user.getEmail());
148         }
149         else if(!(isVendor.isEmpty())){
150             status = "login as a vendor succesful";
151             loginVendor.setEmail(user.getEmail());
152         }
153         else {
154             status = "login failed";
155         }
156 
157         return status;
158     }
```

Probably a detailed reason for login failure would be helpful.

```
251     @DeleteMapping(value = "/admin/deleteCustomer/{id}")
252     public String deleteCustomer(@PathVariable long id) {
253         Customer customer = CustomerRepo.findById(id).get();
254         CustomerRepo.delete(customer);
255         return "deleted";
256     }
```

This will raise an exception if a customer with this ID is not found. Shouldn't you handle this case properly? I am seeing this behavior in a number of other places as well.

Code Review for SearchPage.java

Overall Comments

The code is readable. The code is styled properly, variable names are meaningful and indentation is properly followed. Additionally, the various sections of the code are easy to understand. However, there is some room for improvement. Error scenarios can be handled in a better way. There are some React practices that are not being followed properly like assigning a unique key to each element in the list.

Detailed Review

```
1  import React from "react";
2  import Header from "./header";
3  // import {Link} from 'react-router-dom';
4  import '../css/searchPage.css';
5  import { useState } from "react";
6
7  import Footer from "./footer.js";
8  import { json } from "react-router-dom";
9  import VendorSignUp from "./VendorSignUp";
10
11
12
13
14
15 const SearchPage=(search)=>{
```

You can remove the unused import. The line spacing can be fixed. There is too much space wasted, this can be cleaned out.

```
const SearchPage=(search)=>{

    const [items, setItems] = useState(false);


    const getItems = event => {
        const body = {
            name : search.name
        }
        const settings = {
            method : "post",
            body : JSON.stringify(body)
        }
        fetch('/search', settings).then(data => setItems(data))
    }

```

What if the API fails to respond? I think the failure scenarios need to be handled.

```
    return(
        <div>
            <Header/>
            <h2>Search Page will be fully implemented in the future</h2>
            <div className="search-page">
                <div className="filter">
                    <div className="filter-price">
                        <h4>Price</h4>
                        <div className="price-box">
                            <input type="text" placeholder="min"></input>
                        </div>
                        <div className="price-box">
                            <input type="text" placeholder="max"></input>
                        </div>
                    </div>
                    <div>
                        <h4>Exclude</h4>
                    </div>
                    <div className="filter-exclude">
                        <div className="filter-nuts">
```

I like the code indentation. It's easy to understand the hierarchy of components.

```
                                <label htmlFor="1">☆</label>
                            </div>
                        </div>
                        <div className="search-page-container">
                            {items.map (item => {
                                return (
                                    <div>
                                        <p>{item.name}</p>
                                        <p>Price: {item.price}</p>
                                        <p>Description: {item.description}</p>
                                    </div>
                                )
                            })}
                        </div>
                    </div>
                    <Footer/>
                </div>
    );
    };

```

In React, you should set a key property in each element of a list. This helps React in optimizing the page re-renders.

Overall, this file (TvfmAPI.java) needs more comments for each path describing what the path needs and does. Also, it is missing a header. The naming scheme is ok. It might want to be modified a little. For instance, in the update items path, the item should just be called item. The sub-headers are very descriptive and help to differentiate the different paths.

```
package com.Team03.TVFM.api;
import com.Team03.TVFM.model.*;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpHeaders;
import org.springframework.util.Assert;
import org.springframework.web.bind.annotation.*;
import org.springframework.web.servlet.view.RedirectView;
import java.util.*;
import java.time.format.DateTimeFormatter;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;
import java.text.SimpleDateFormat;
import java.util.concurrent.ConcurrentHashMap;
import java.util.regex.Matcher;
import java.util.regex.Pattern;
import javax.persistence.Query;
import javax.transaction.Transactional;
@RestController
@RequestMapping("/")
public class TvfmAPI {
   @Autowired
   private CustomerRepo CustomerRepo;
   @Autowired
   private VendorRepository VendorRepository;
   @Autowired
   private ItemsRepo ItemsRepo;
   @Autowired
   private PurchasesRepo PurchasesRepo;
   private List<Customer> isCustomer;
   private List<Vendor> isVendor;
   Vendor loginVendor = new Vendor();
   Customer loginCustomer = new Customer();

   //------Utility Functions-------//
   // Checks if the email is a valid email (does not look for
conflicts in the db)
```

```
    public static boolean emailValidation(String email) {
        Pattern emailReg =
Pattern.compile("/^[\\w-\\.]+@([\\w-]+\\.)+[\\w-]{2,4}$/");
        Matcher emailMatch = emailReg.matcher(email);
        return emailMatch.find();
    }
    // Checks if the password is at least 8 long has a capital
number and letter
    public static boolean passwordValidation(String password) {
        Pattern numReg = Pattern.compile("[0-9]+");
        Pattern capReg = Pattern.compile("[A-Z]+");
        Matcher numMatch = numReg.matcher(password);
        Matcher capMatch = capReg.matcher(password);
        if(password.length() >= 8 &&
                numMatch.find() &&
                capMatch.find()) {
            return true;
        }
        return false;
    }
    public String passwordHashing(String password){
        String passwordToHash = password;
        String generatedPassword = null;
        try {
            MessageDigest md = MessageDigest.getInstance("MD5");
            md.update(passwordToHash.getBytes());
            byte[] bytes = md.digest();
            StringBuilder sb = new StringBuilder();
            for (int i = 0; i < bytes.length; i++) {
                sb.append(Integer.toString((bytes[i] & 0xff) +
0x100, 16).substring(1));
            }
          generatedPassword = sb.toString();
        } catch (NoSuchAlgorithmException e) {
        e.printStackTrace();
        }
        //System.out.println(generatedPassword);
        return generatedPassword;
    }
    //------Registration path-------/
    @PostMapping(value = "/registration")
    public String registration(@RequestBody Registration user){
        String status = "registration failed";
        if (passwordValidation(user.getPassword()) == false) {
            return status;
        }
        String newPassword = passwordHashing(user.getPassword());
```

```
        String hashEmail = passwordHashing(user.getEmail());
        if(user.getVendor() == 0){
            Customer customer = new Customer();
            customer.setId(user.getId());
            customer.setEmail(hashEmail);
            customer.setPassword(newPassword);
            customer.setName(user.getName());
            customer.setLastname(user.getLastname());
            CustomerRepo.save(customer);
            status = "registered as a customer";
        }
        else{
            Vendor vendor = new Vendor();
            vendor.setId(user.getId());
            vendor.setEmail(hashEmail);
            vendor.setPassword(newPassword);
            vendor.setName(user.getName());
            vendor.setLastname(user.getLastname());
            VendorRepository.save(vendor);
            status = "registered as vendor";
        }
        return status;
    }
    //----------Search-----------//
    @Transactional
    @PostMapping(value = "/search")
    @CrossOrigin
    public List<Items> searchItem(@RequestBody Items item){
        List<Items> itemRepoHolder;
        if(item.getId() instanceof Long){
            itemRepoHolder =
ItemsRepo.findItembyID(item.getId());  //to look with an id
        }
        else if(item.getName() instanceof String){
            itemRepoHolder = ItemsRepo.findItem(item.getName());
//to look with a name, description, nutrition
        }
        else{
            System.out.println("Item not found");
            itemRepoHolder = null;
        }
        //for a recommendation, may have to change it later
        if(itemRepoHolder.isEmpty()){
            Random rand = new Random();
            int num = rand.nextInt((ItemsRepo.maxID() - 1) + 1) +
1;
            itemRepoHolder = ItemsRepo.findItembyID(num);
```

```
        }
        return itemRepoHolder;
    }
    //-------Login--------//
    Map<String, Integer> loginMap = new ConcurrentHashMap<>();
    @PostMapping(value = "/login")
    @CrossOrigin()
     public String login(@RequestBody Registration user,
@RequestHeader("User-Agent") String header){
         String status  = "login failed";
        if(loginMap.get(header) >= 3) {
            return status;
        }
        isCustomer = CustomerRepo.findCustomer(user.getEmail(),
user.getPassword());
        isVendor = VendorRepository.findVendor(user.getEmail(),
user.getPassword());
        if(!(isCustomer.isEmpty())) {
            status = "customer";
//            loginCustomer.setEmail(user.getEmail());
            loginCustomer = isCustomer.get(0);
        }
        else if(!(isVendor.isEmpty())){
            status = "vendor";
            loginVendor.setEmail(user.getEmail());
        }
        else {
            //insert the person in the map or increment their
number of failed logins
            loginMap.merge(header, 1, Integer::sum);
            status = "login failed";
        }
        return status;
    }
    //still testing
    public RedirectView vendorPage(){
        return new RedirectView("https://www.google.com/");
    }
    //------previous purchases--------//
    @PostMapping(value = "{limit}/purchases")
    public List<Purchases> getPurchases(@PathVariable long limit,
@RequestBody Customer customer) {
        // get all the purchases
        List<Purchases> retval =
PurchasesRepo.findPurchasesByCustomerID(customer.getId(),
limit);
        // start looking for the items in the purchases
```

```
        for(Purchases purchase : retval) {
            // get the id of the items
            List<Integer> itemIDs =
PurchasesRepo.findItemIDInPurchase(purchase.getId());
            List<Items> items = new LinkedList<>();
            for (Integer id: itemIDs) {
                items.addAll(ItemsRepo.findItembyID(id));
            }
            // add all the items to the purchase
            purchase.setItems(items);
        }
        return retval;
    }
    //-------get vendor info----------//
    @GetMapping(value = "/vendorInfo/{id}")
    public String getVendorInfo(@PathVariable long id){
        String info = ItemsRepo.findVendor(id);
        return "This email of the vendor: " + info + "\n Name: "
+
        VendorRepository.vendorName(info) + " Lastname: " +
VendorRepository.vendorLastname(info);
    }
    //-------place an order---------//
    //Still working on this
    @PostMapping(value = "/cart/placeOrder")
    public String placeOrder(@RequestBody Items items) {
        Purchases makeOrder = new Purchases();
        String status;
        SimpleDateFormat formatter = new
SimpleDateFormat("dd/MM/yyyy");
        Date date = new Date();
        if(!(isCustomer.isEmpty())){
            if(PurchasesRepo.maxID() == 0){
                makeOrder.setId(PurchasesRepo.maxID());
            }
            else{
                makeOrder.setId(PurchasesRepo.maxID() + 1);
            }
            makeOrder.setOrder_date(formatter.format(date));
            long id =
CustomerRepo.findCustomerID(loginCustomer.getEmail());
            makeOrder.setCustomer_id(id);
            makeOrder.setStatus("Order Placed");
            makeOrder.setItem(items);
            status = "Order Placed";
            PurchasesRepo.save(makeOrder);
        }
```

```
        else{
            status = "you need to login to use this function";
        }
        return status;
    }
    //-------current orders---------//
    @GetMapping(name = "/admin/orders")
    public List<Purchases> allOrders() {
        return PurchasesRepo.findAll();
    }
    //-------Admin permissions-------//
    @GetMapping(value = "/admin/allCustomers")
    public List<Customer> getAllCustomers() {
        return CustomerRepo.findAll();
    }
    @GetMapping(value = "/admin/allVendors")
    public List<Vendor> getAllVendors() {
        return VendorRepository.findAll();
    }
    @DeleteMapping(value = "/admin/deleteCustomer/{id}")
    public String deleteCustomer(@PathVariable long id) {
        Customer customer = CustomerRepo.findById(id).get();
        CustomerRepo.delete(customer);
        return "deleted";
    }
    @PutMapping(value = "/admin/updateVendor/{id}")
    public String updateVendor(@PathVariable Long id,
@RequestBody Vendor vendor) {
        Vendor updateVendor =
VendorRepository.findById(id).get();
        updateVendor.setEmail(vendor.getEmail());
        updateVendor.setPassword(vendor.getPassword());
        VendorRepository.save(updateVendor);
        return "updated";
    }
    @DeleteMapping(value = "/admin/deleteVendor/{id}")
    public String deleteVendor(@PathVariable long id) {
        Vendor vendor = VendorRepository.findById(id).get();
        VendorRepository.delete(vendor);
        return "deleted";
    }
    @GetMapping(value = "/admin/allItems")
    public List<Items> getAllItems() {
        return ItemsRepo.findAll();
    }
    @PostMapping(value = "/admin/addItems")
    public String addItems(@RequestBody Items item) {
```

```
        String status = "no items added";
        if(!(isVendor.isEmpty())){
            item.setVendor(loginVendor.getEmail());
            ItemsRepo.save(item);
            status = "Saved";
        }
        else{
            status = "you need to login to use this function";
        }
        return status;
    }
    @DeleteMapping(value = "/admin/removeItem/{id}")
    public String removeItem(@PathVariable long id){
        Items item = ItemsRepo.findById(id).get();
        ItemsRepo.delete(item);
        return "deleted";
    }
    @PutMapping(value = "/admin/updateItem/{id}")
    public String updateItem(@PathVariable Long id, @RequestBody
Items item) {
        Items updateItem = ItemsRepo.findById(id).get();
        updateItem.setName(item.getName());
        updateItem.setQuantity(item.getQuantity());
        updateItem.setPrice(item.getPrice());
        updateItem.setDescription(item.getDescription());
        updateItem.setNutrition(item.getNutrition());
        ItemsRepo.save(updateItem);
        return "updated";
    }
    //-------Vendor paths--------//
    @PostMapping(value = "/vendor/addItems")
    public String addItemsVendor(@RequestBody Items item) {
        ItemsRepo.save(item);
        return "Saved";
    }
    @DeleteMapping(value = "/vendor/removeItem/{id}")
    public String removeItemVendor(@PathVariable long id){
        Items item = ItemsRepo.findById(id).get();
        ItemsRepo.delete(item);
        return "deleted";
    }
    @PutMapping(value = "/vendor/updateItem/{id}")
    public String updateItemVendor(@PathVariable Long id,
@RequestBody Items item) {
        Items updateItem = ItemsRepo.findById(id).get();
        updateItem.setName(item.getName());
        updateItem.setQuantity(item.getQuantity());
```

```
        updateItem.setPrice(item.getPrice());
        updateItem.setDescription(item.getDescription());
        updateItem.setNutrition(item.getNutrition());
        ItemsRepo.save(updateItem);
        return "updated";
    }
    @GetMapping(value = "/vendor/allItems")
    public List<Items> getAllItemsVendor() {
        return ItemsRepo.findAll();
    }
    //------ testing paths -------//
```

```
    @GetMapping(value = "test")
    public String test(){
        return "Test is working";
    }
    @PostMapping(value = "/add/customer")
    public String addCustomer(@RequestBody Customer customer) {
        CustomerRepo.save(customer);
        return "Saved";
    }
    @PostMapping(value = "/add/vendor")
    public String addVendors(@RequestBody Vendor vendor) {
        VendorRepository.save(vendor);
        return "Saved";
    }
    @GetMapping(value = "/customer")
    @CrossOrigin
    public Customer getCustomer(){
        return loginCustomer;
    }
    @GetMapping(value = "/logout")
    @CrossOrigin
    void logoutUser(){
        loginCustomer = null;
    }
}
```

Self-Check on best practices for security

Major assets that are protected

- Customer's password
- Vendor's password

```
18 •  SELECT * FROM dbmodel.customer;
```

Result Grid | Filter Rows: | Edit: | Export/Import:

id	name	lastname	email	password
19	Jacob	Barns	22b7a95b8fdcf41aa3464da3a0cd5be9	94e8cde4...
20	Jimmy	Brown	6812b3df0afbdbae36ef3aa5a115e8c9	7adad95...
21	York	Blake	a88f873a621b501a514679949bd2a4e3	9fa5adf3...
22	Johnathan	Lively	74088342311ee9a4d557c443f112fb01	60167ad...
23	Lil	Wayne	ebe7b324764c1bf376efd915c2314a28	f946d8b1...
NULL	NULL	NULL	NULL	NULL

```
15 •  SELECT * FROM dbmodel.vendor;
```

Result Grid | Filter Rows: | Edit: | Export/Import:

id	name	lastname	email	password
9	Masion	Gibson	a829547725b60...	94e8cde4612b3fd390677d42e7b22002
10	James	Hill	895613f33bdfa3...	ff4cac7d4874299d83f8be280d1cbb77
11	Lily	Ryan	9592cba3a8341...	af65872d9bdcc4e0071f11cb5ebadb2f
12	Sierra	Tang	61c9ccbf8199bc...	cfe3b870f18dec98de461639bbe8da08
13	Hungry	Lion	e5c01b825a950...	8b1a7220fac6825195becf17ebf92174
NULL	NULL	NULL	NULL	NULL

Confirm Input data validation (list what is being validated and what code you used) – we request you validate search bar input;

Self-Check adherence to Original non-functional specs

Coding Standards:

1. All variables shall use camelCase. On Track
2. Space between ‘if’ and the condition i.e if (condition). On Track
3. Space between parentheses and the curly brace i.e () {}. On track
4. Space before and after “=”. On Track
5. Comment above each function describing its purpose. On Track

Performance:

1. The home page shall load in at least 500 ms. Done

Compatibility:

1. Current and previous version of Google Chrome. Done
2. Current and previous version of Mozilla FireFox. On Track
3. Current and previous version of Microsoft Bing. On Track
4. Current and previous version of Safari by Apple. On Track

Scalability:

1. Application will be able to accommodate multiple users. On Track
2. Application will be able to scale up and down depending on device. On Track

Portability:

1. Website will function properly on different devices. On Track

Reliability:

1. Application will consistently perform without failure. On Track

Security:

1. User’s personal information to be kept confidential and not distributed. Done
2. User’s passwords to be encrypted. Done
3. Users may not be granted access till a strong password is created. Done
4. System will lock the user out after a specific number of login attempts. On Track

Usability:

1. Interface is easy to use, user-friendly. On Track

Availability:

1. Support during business hours. Issue
2. Common question solutions will be available for non-business hours On Track

Regulatory:

1. Application will comply with all regulations and laws. On Track

Data Integrity:

1. System shall have backups of all updates to the database for every transaction. On Track

Cost:

1. Developmental costs will be kept at a min by using free services provided online.

   On Track

Manageability:

Testability:

1. Application will have a suitable testing approach for each feature. On Track

Organization:

1. Company name and logo on the top-left corner of each web page. Done
2. Files and folders will be easily accessible and not confusing. On Track

Detailed list of contributions

Seth Pavlicek:

- Password Validation Backend
- Email Validation Backend
- Login Lockout Backend
- Code Review for other team
- Code Review within team
- Password Hashing QA Test Plan

Alex Bjeldanes: 7/10

- Database Updates
- CSS Rework
- Code Review for other team

Armando Partida: 8/10

- Password Hashing
- Revise Order Usability Test Plan
- Backend Paths

Angel Antunez: 9/10

- Password Validation Frontend
- Changing Personal Info Test Plan
- QA Test plans

Igor Tsygankov: 10/10

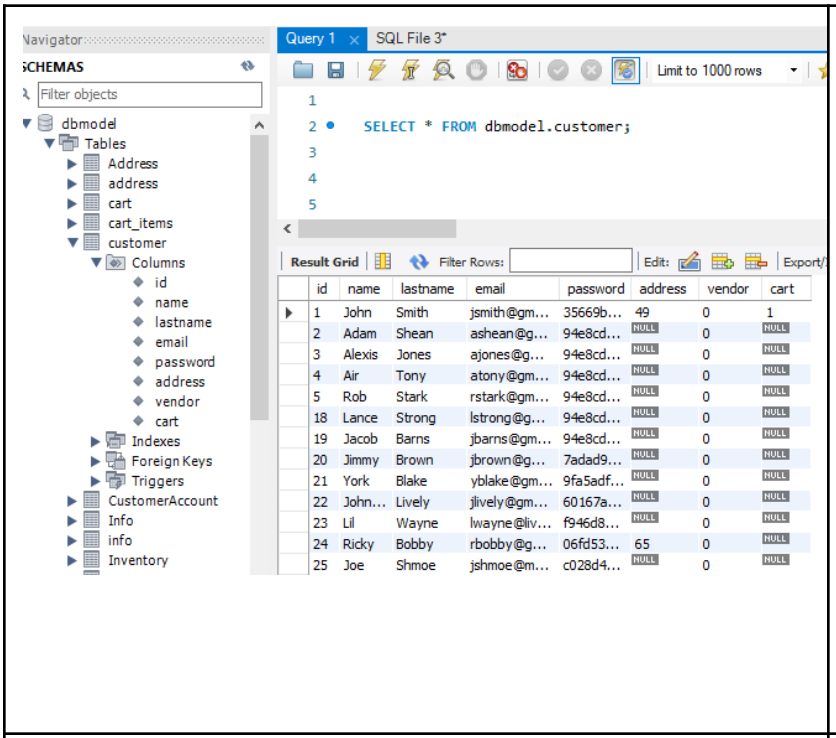
- AWS Updates
- Header change on login
- Search Usability Test Plan
- Login Lockout Frontend

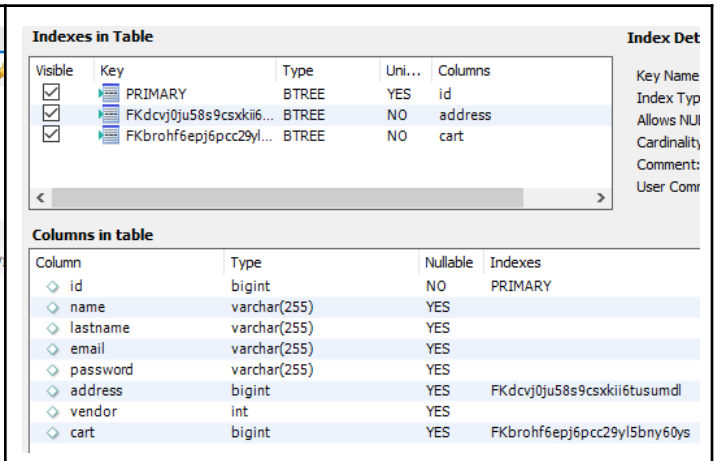
Michael Abolencia: 9/10

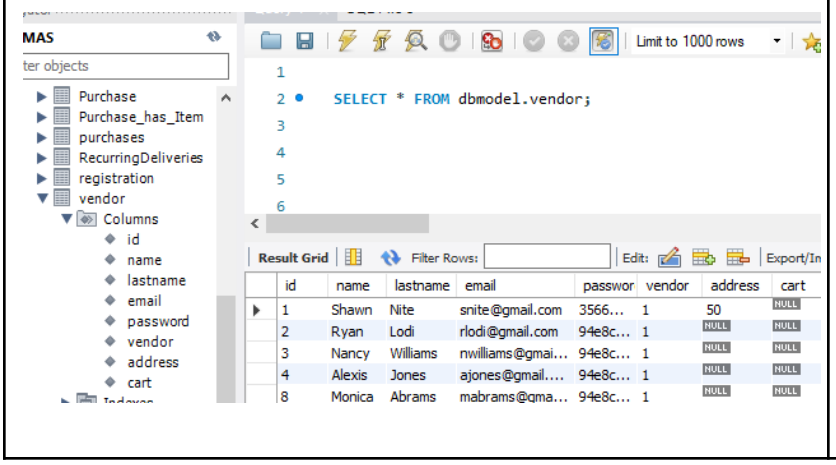
- Search Page Integration

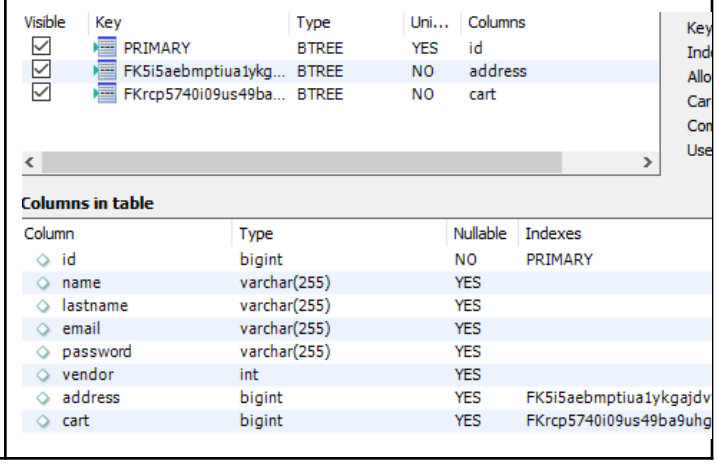
- Order Submission Page
- Add Item Usability Test Plan

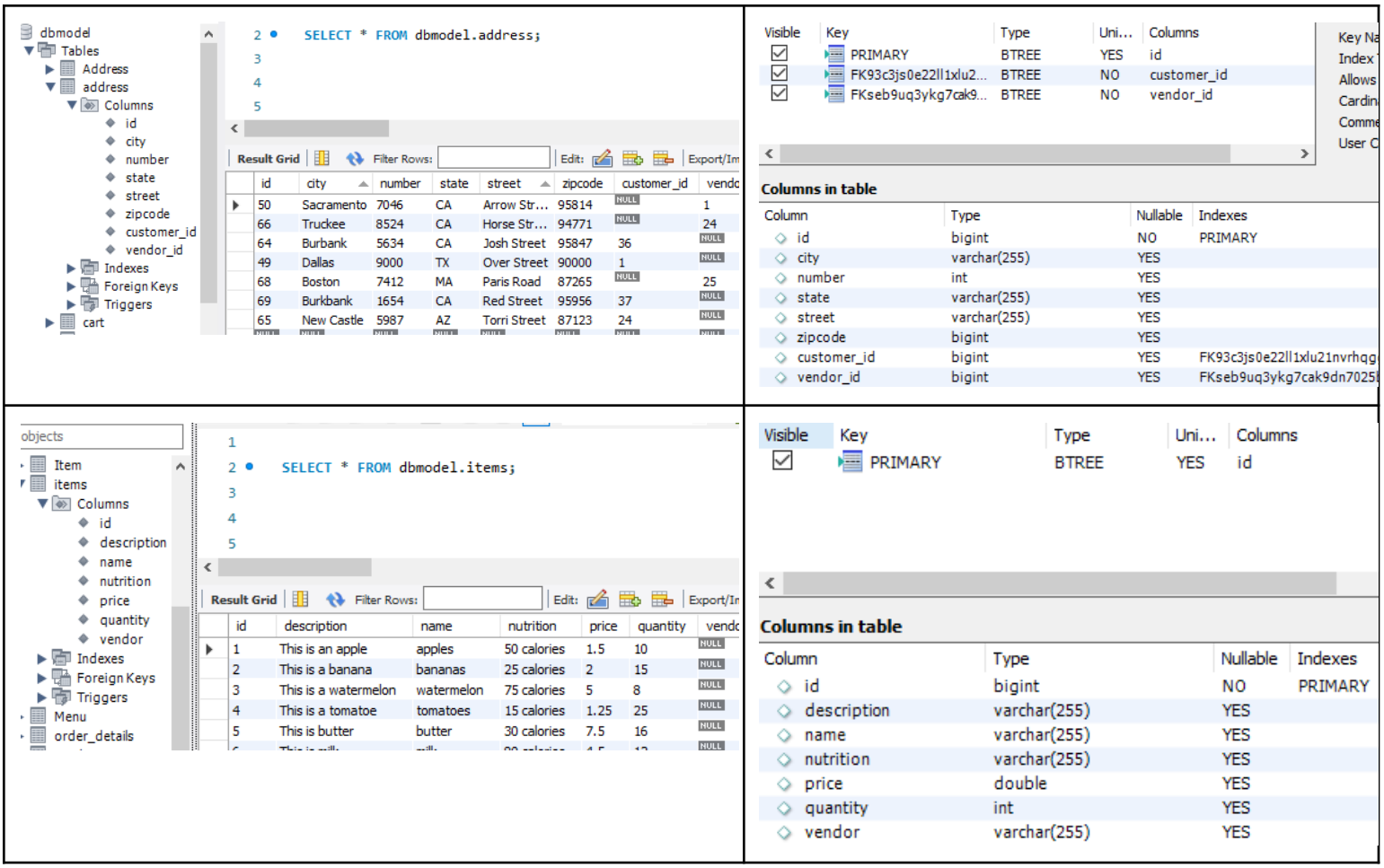
Screenshots of key DB tables



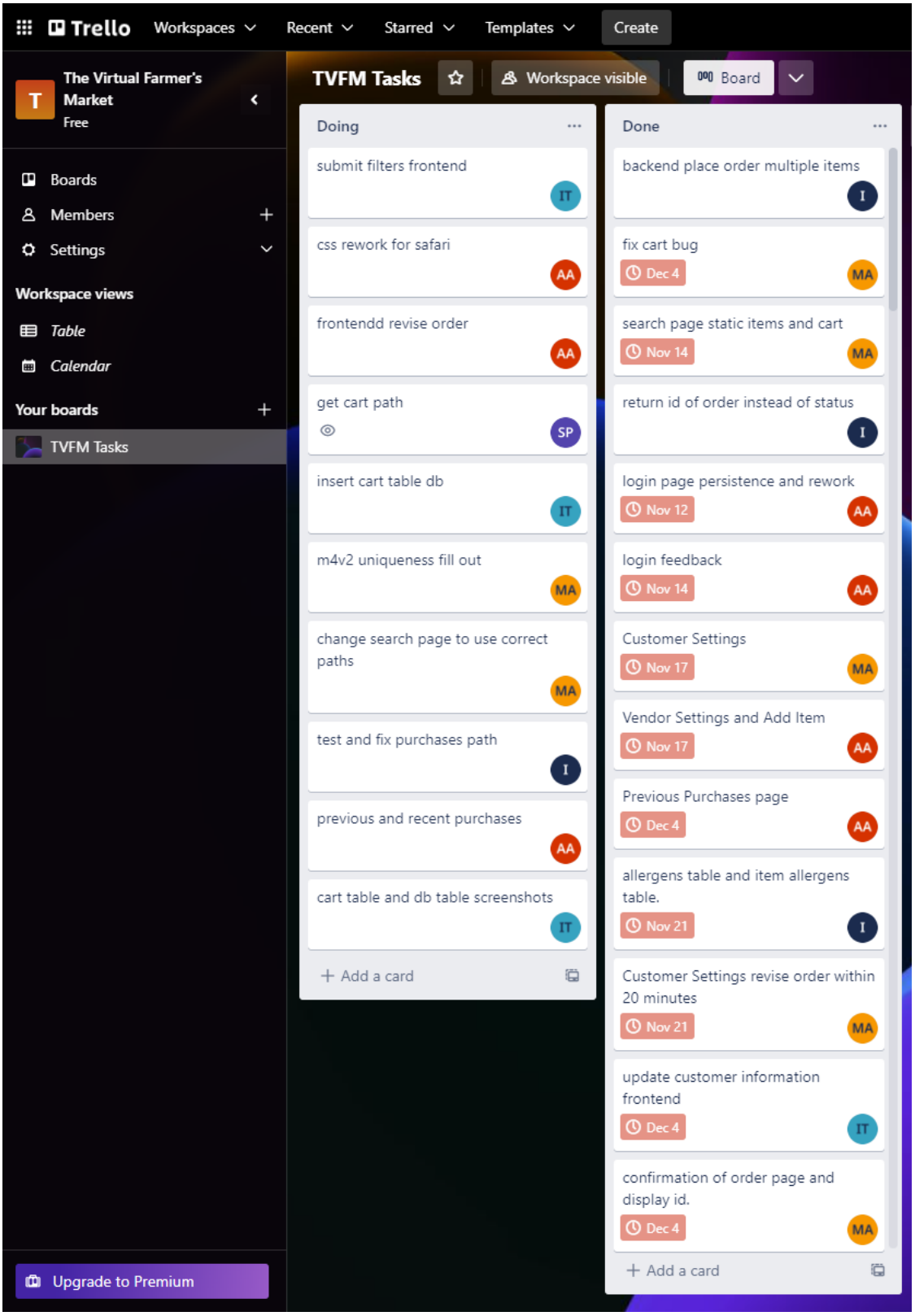







dbmodel
Tables
Address
address
Columns
id
city
number
state
street
zipcode
customer_id
vendor_id
Indexes
Foreign Keys
Triggers
cart
SELECT * FROM dbmodel.address;
Result Grid
Filter Rows:
Edit:
Export/Im
id city number state street zipcode customer_id vendc
50 Sacramento 7046 CA Arrow Str... 95814 NULL 1
66 Truckee 8524 CA Horse Str... 94771 NULL 24
64 Burbank 5634 CA Josh Street 95847 36 NULL
49 Dallas 9000 TX Over Street 90000 1 NULL
68 Boston 7412 MA Paris Road 87265 NULL 25
69 Burkbank 1654 CA Red Street 95956 37 NULL
65 New Castle 5987 AZ Torri Street 87123 24 NULL
Visible Key Type Uni... Columns
PRIMARY BTREE YES id
FK93c3js0e22ll1xlu2... BTREE NO customer_id
FKseb9uq3ykg7cak9... BTREE NO vendor_id
Key Na
Index
Allows
Cardin
Comme
User C
Columns in table
Column Type Nullable Indexes
id bigint NO PRIMARY
city varchar(255) YES
number int YES
state varchar(255) YES
street varchar(255) YES
zipcode bigint YES
customer_id bigint YES FK93c3js0e22ll1xlu21nvrhqg
vendor_id bigint YES FKseb9uq3ykg7cak9dn7025
objects
Item
items
Columns
id
description
name
nutrition
price
quantity
vendor
Indexes
Foreign Keys
Triggers
Menu
order_details
SELECT * FROM dbmodel.items;
Result Grid
Filter Rows:
Edit:
Export/In
id description name nutrition price quantity vendc
1 This is an apple apples 50 calories 1.5 10 NULL
2 This is a banana bananas 25 calories 2 15 NULL
3 This is a watermelon watermelon 75 calories 5 8 NULL
4 This is a tomatoe tomatoes 15 calories 1.25 25 NULL
5 This is butter butter 30 calories 7.5 16 NULL
Visible Key Type Uni... Columns
PRIMARY BTREE YES id
Columns in table
Column Type Nullable Indexes
id bigint NO PRIMARY
description varchar(255) YES
name varchar(255) YES
nutrition varchar(255) YES
price double YES
quantity int YES
vendor varchar(255) YES

Screenshot of our task management system



Team Member Contributions

Seth Pavlicek:

- Setup backend and frontend servers locally
- Executive Summary
- Coding Standards in non-functional requirements
- Use Case 1, 5, 11
- Researched Instacart
- Data Definitions
- High Level API
- Deployment Diagram
- UI Mockups: 1, 5, 6
- Header Wireframe
- Homepage Wireframe
- Recurring Delivery Wireframe
- Home Page Hook
- Password Validation Backend
- Email Validation Backend
- Login Lockout Backend
- Code Review for other team
- Code Review within team
- Password Hashing QA Test Plan

Alex Bjeldanes: 9/10

- Finalized and reviewed use cases
- Formatted milestone 1 document
- Researched Safeway
- Use Case 2, 13
- UML and ERD Diagrams
- Database Requirements
- UI Mockups: 2, 13
- Milestone 2 Document Cleanup
- EER Diagram
- Database Updates
- CSS Rework
- Code Review for other team
- Milestone 4 V2 QA rework

Armando Partida: 8/10

- Combined research into competitive analysis
- Researched Instacart
- Use Case 8, 12
- UML and ERD Diagrams
- Backend Setup
- UI Mockups: 8, 12
- Backend Classes from UML

- Initial Search Query
- Password Hashing
- Revise Order Usability Test Plan
- Backend Paths

Angel Antunez: 8/10

- Non-Functional requirements and Functional requirements
- Researched Farm Fresh to You
- Use Case 3, 9
- Register Page
- UI Mockups: 3, 9
- Footer in Frontend
- Search Page
- Cart Wireframe
- Cart Page
- Password Validation Frontend
- Changing Personal Info Test Plan
- QA Test plans

Igor Tsygankov: 10/10

- Setup backend and frontend on AWS
- Researched Amazon Fresh and Walmart
- Use Case 7
- Application Networks Diagram

- Header for Frontend
- UI Mockups: 7, 11
- AWS Setup
- Header in Frontend
- Settings Wireframe
- Filter Wireframe
- Footer Wireframe
- Search Page Filters
- Recurring Delivery Page
- AWS Updates
- Header change on login
- Search Usability Test Plan
- Login Lockout Frontend

Michael Abolencia: 8/10

- Functional requirements
- Researched Misfits
- Use Case 4, 6, 10
- Login Page
- UI Mockups: 4, 10
- Register Pages
- Register Wireframes
- Product Information Page
- Privacy Policy Page

- Search Page Integration with Backend
- Order Submission Page
- Add Item Usability Test Plan

Post Analysis

Overall, we made a working project. We made many mistakes along the way. Some of our issues were with our tech stack, some were with our time constraint. For our backend framework we chose to use Springboot. All of us had to learn the proper syntax and best practices to use our framework. This ate some of our time that could have been used to improve the project.

This class is crammed into one semester which means we have to push out milestones very quickly. This time constraint definitely reduced the quality of our project.

If we were to try to complete this project again we would pick a backend framework that we are more familiar with. We would also take more classes that would prepare us for such a broad project such as database classes and web development classes.