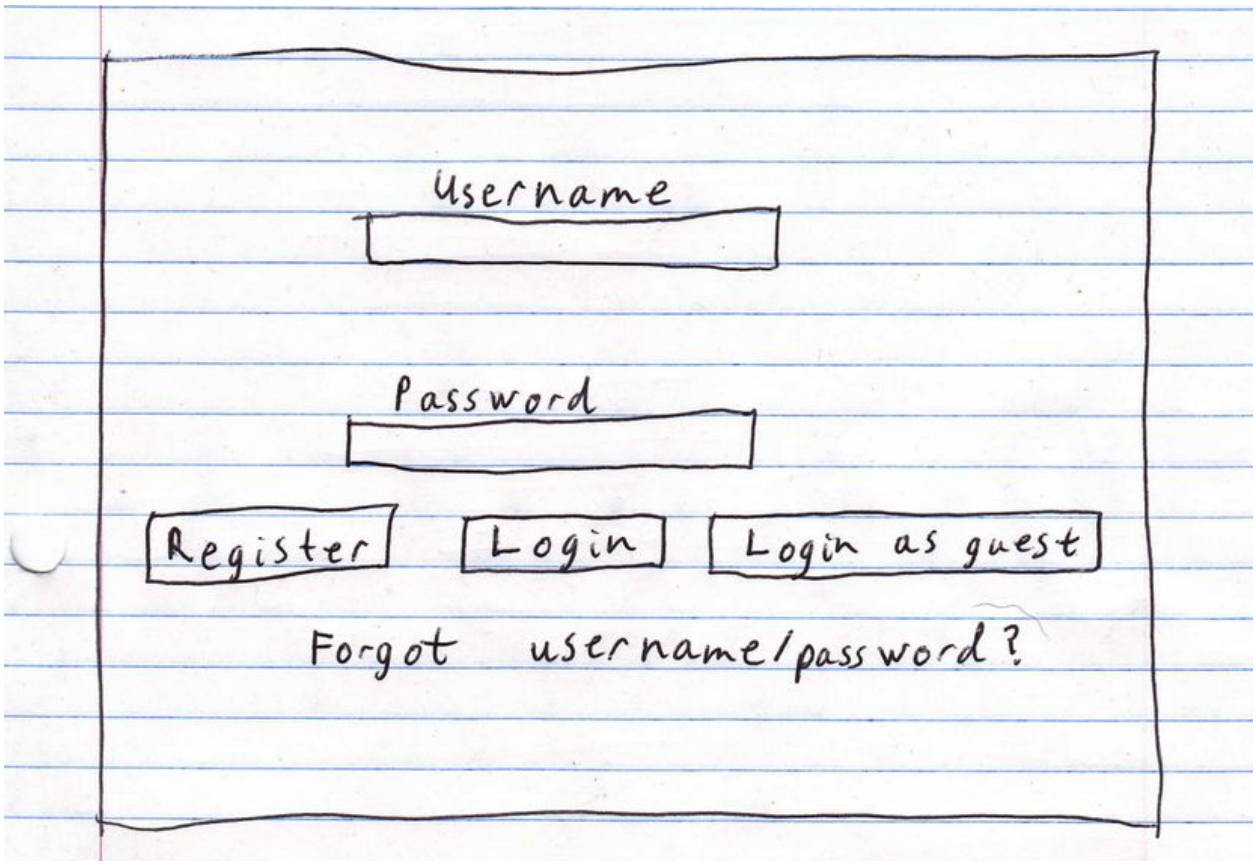
Grocery Shopping App Requirements
(Grocery Guide)

10/27/2019

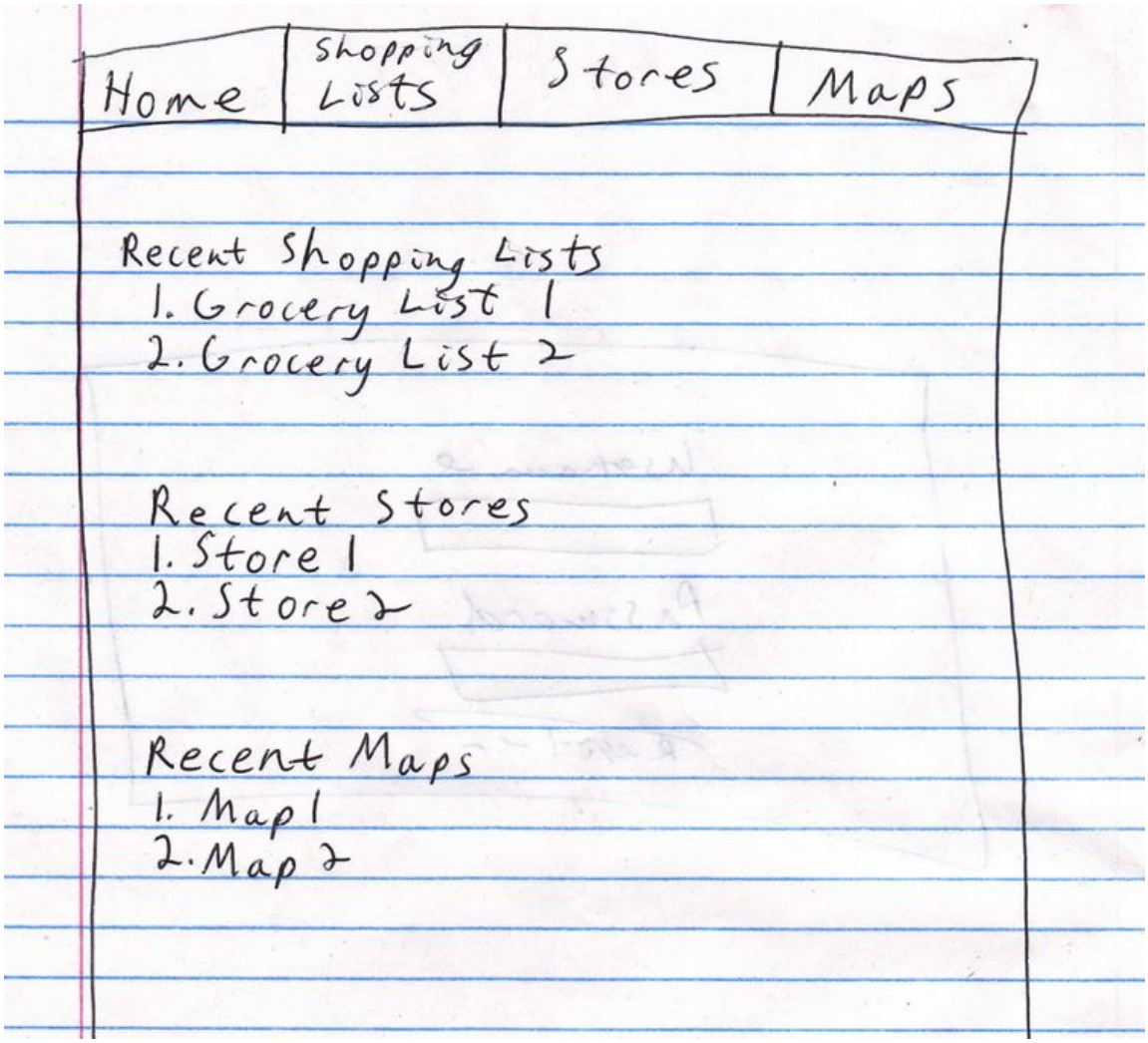
Christopher Brown
Dane Emmerson
Evan Horne
Anthony Huynh
Briana Willems

Login Page



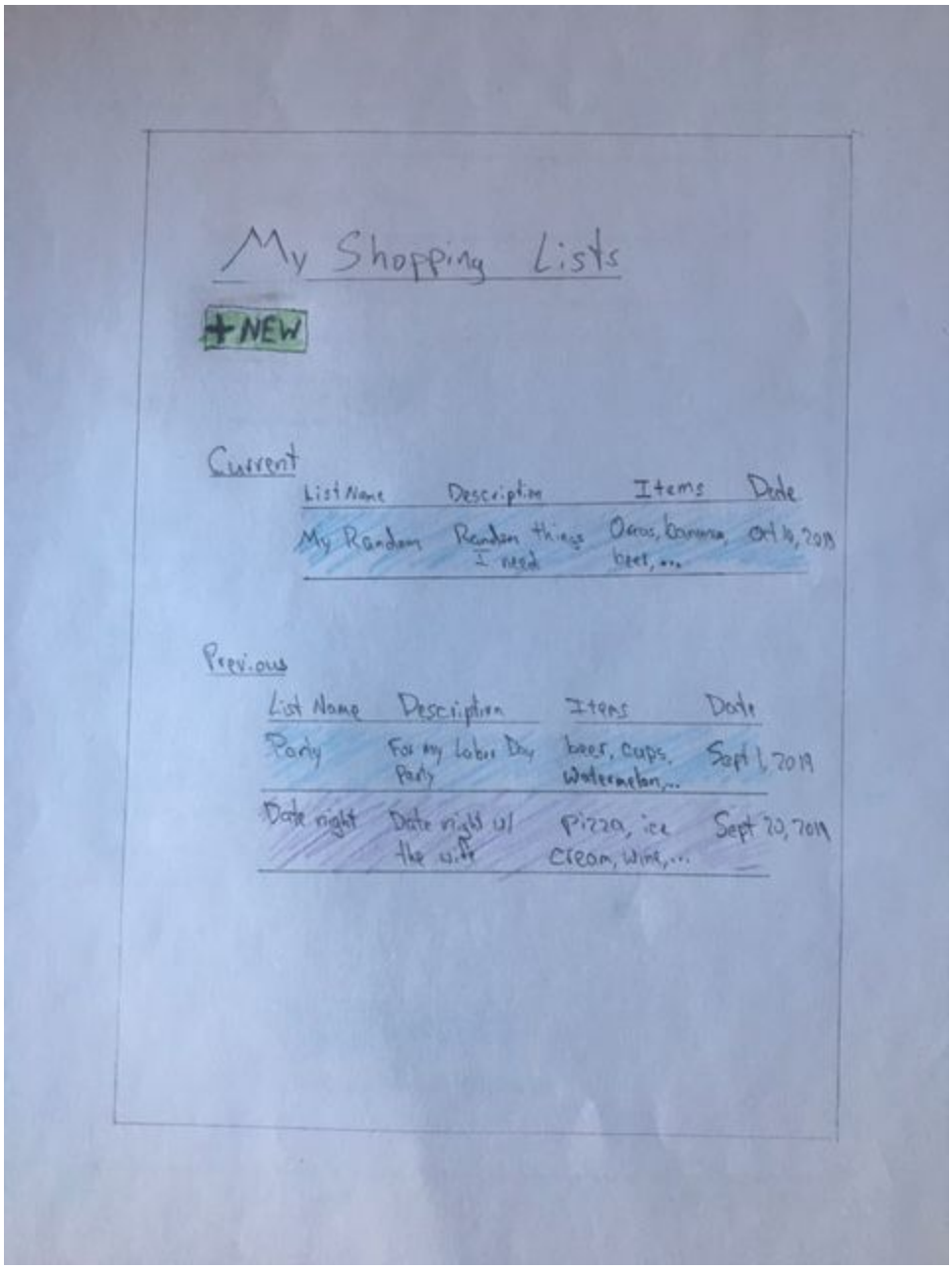
The user should be able to login as guest if they do not wish to create an account.

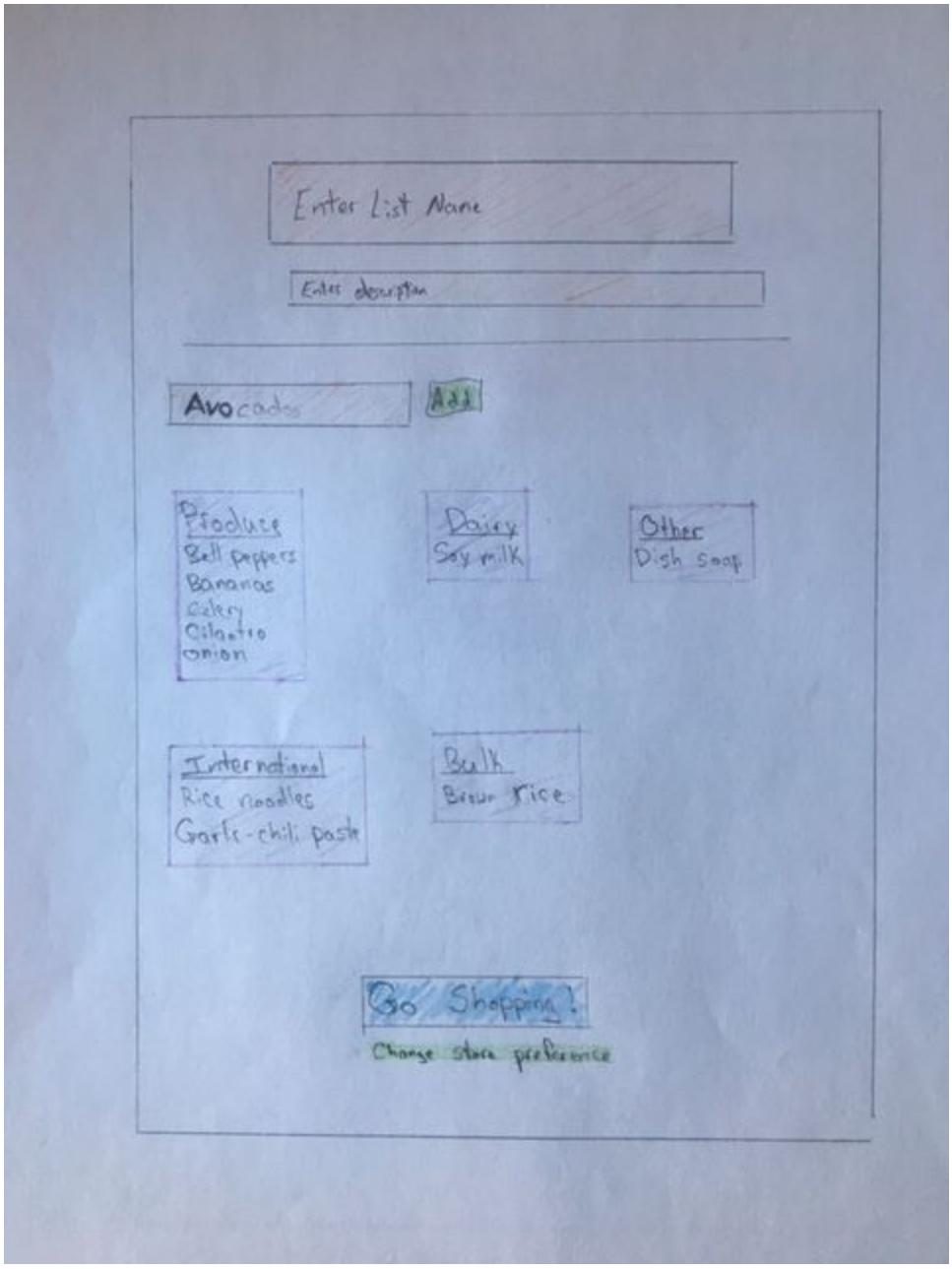
Landing Page



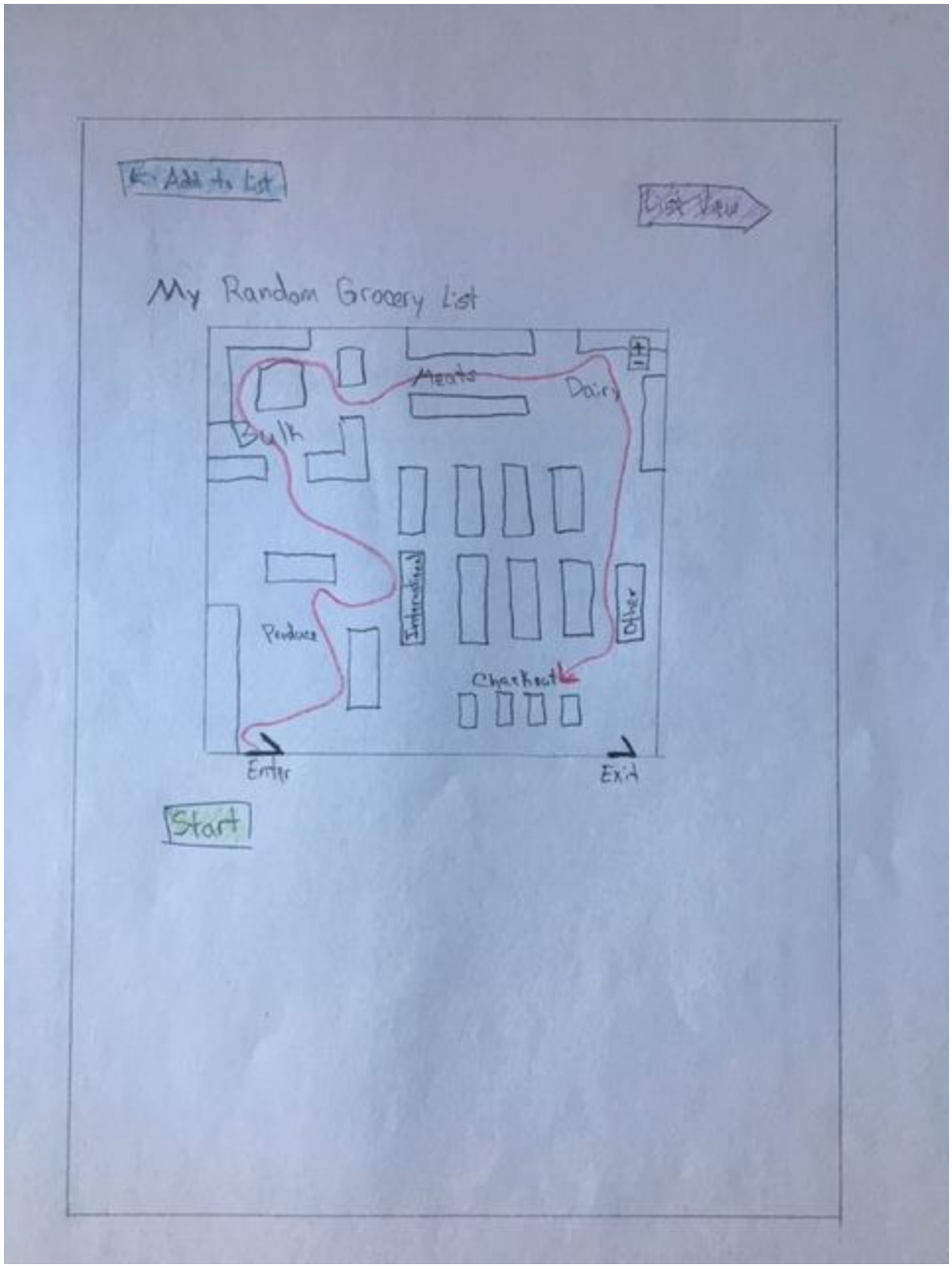
The most recently used shopping lists, stores, and maps are available from the landing page.

Use Case 1 - Create New Shopping List

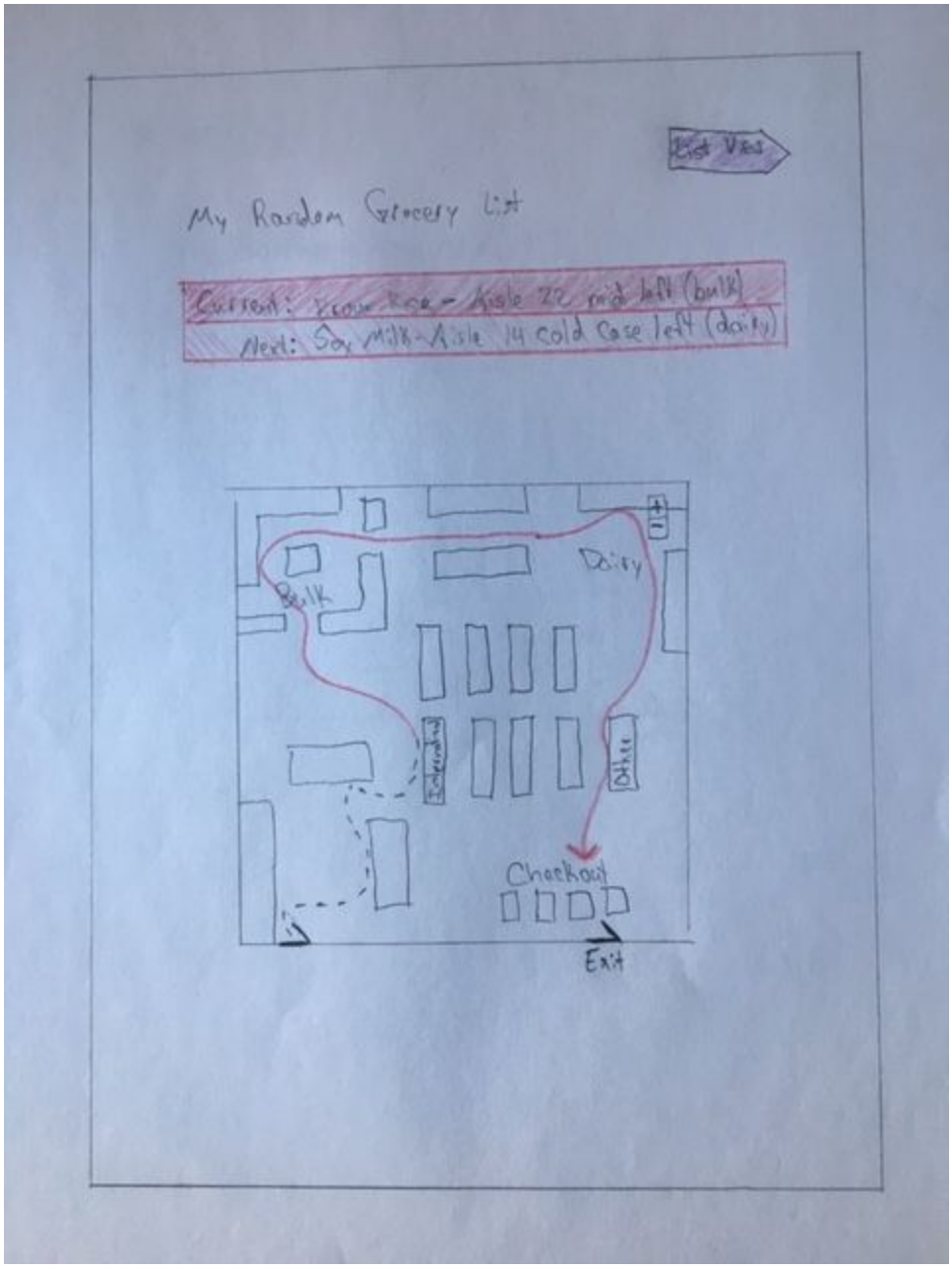




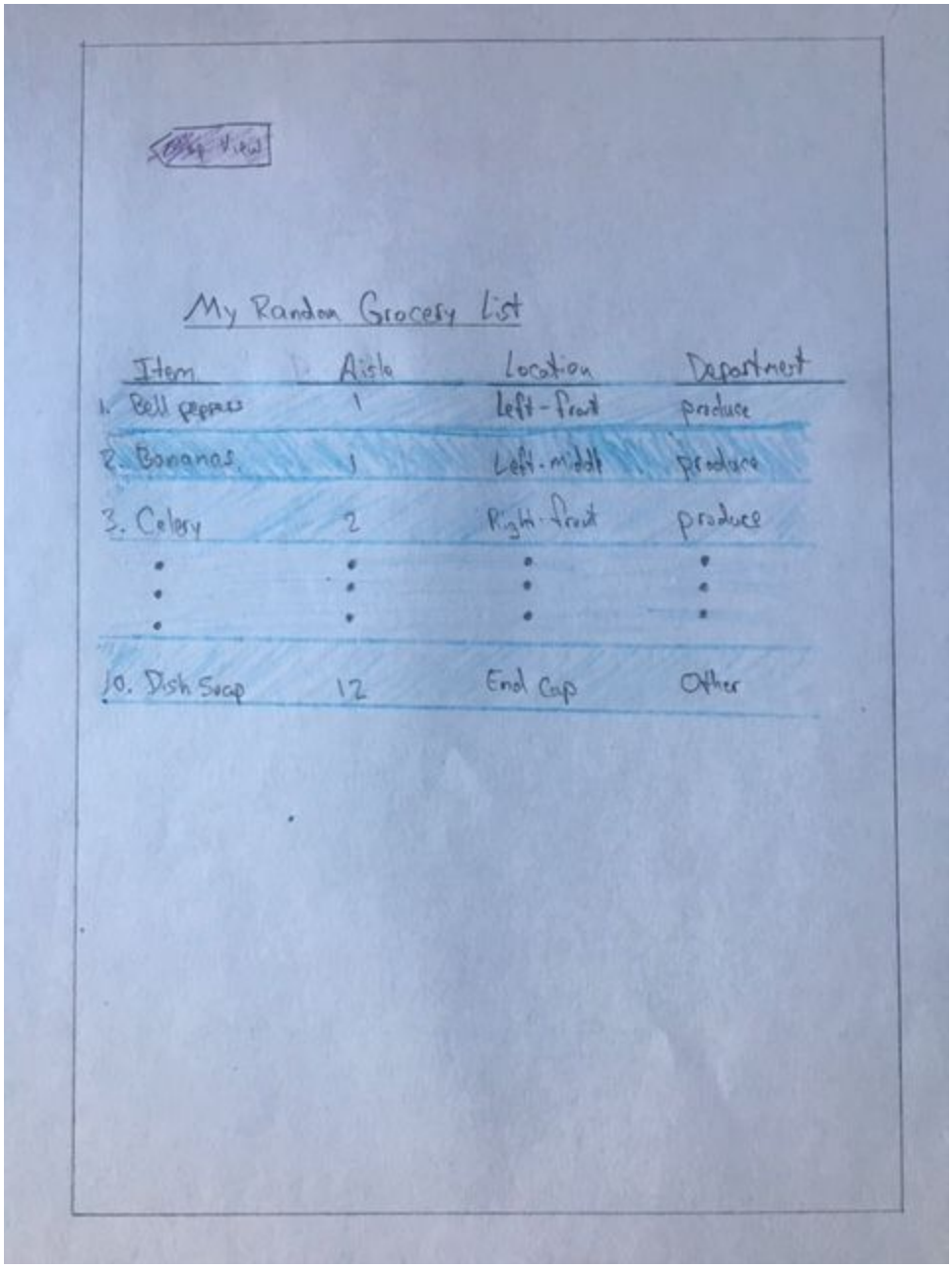
As user types in items, text autopoluates. When user selects “Add”, items are groups categorically together. User may enter in “List Name” and “List Description”. List autosaves as user works through adding list. User may select “Change store preference” to select a different store that their default store. User may also select “Go Shopping” to go to the map view of shopping.



User is now shown a map of the store with their route overview. User may choose to see list view, or user may select “Start”



User is partway through picking up their groceries. The current item that the user is picking up is shown, followed by the next item on the list, along with their respective locations in the store. On the map, the user's path becomes a dotted line as the user picks up each item.

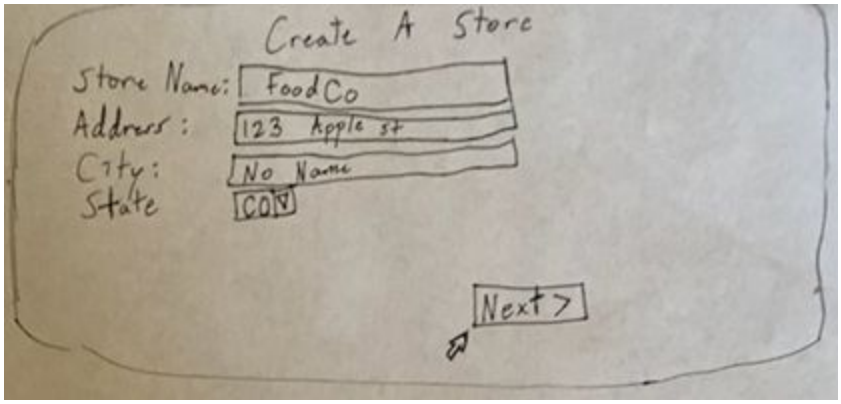


My Random Grocery List

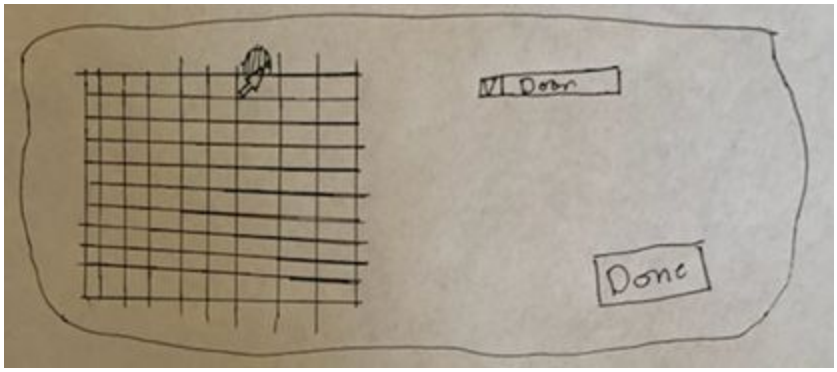
Item	Aisle	Location	Department
1. Bell peppers	1	Left-front	produce
2. Bananas	1	Left-middle	produce
3. Celery	2	Right-front	produce
.	.	.	.
.	.	.	.
.	.	.	.
10. Dish Soap	12	End Cap	Other

User is now looking at the list view of their shopping list route.

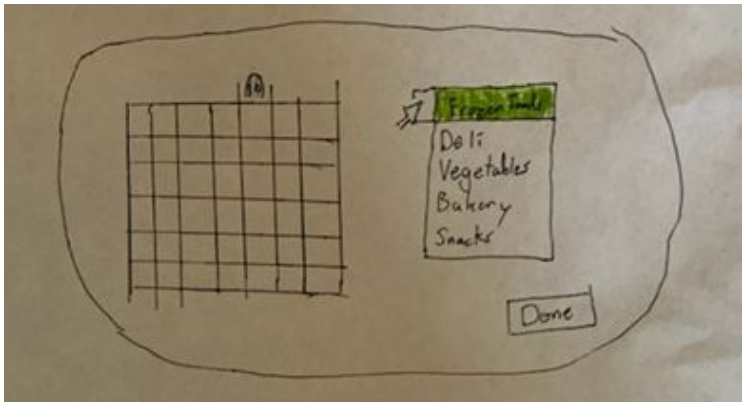
Use Case 2 - Creating a map



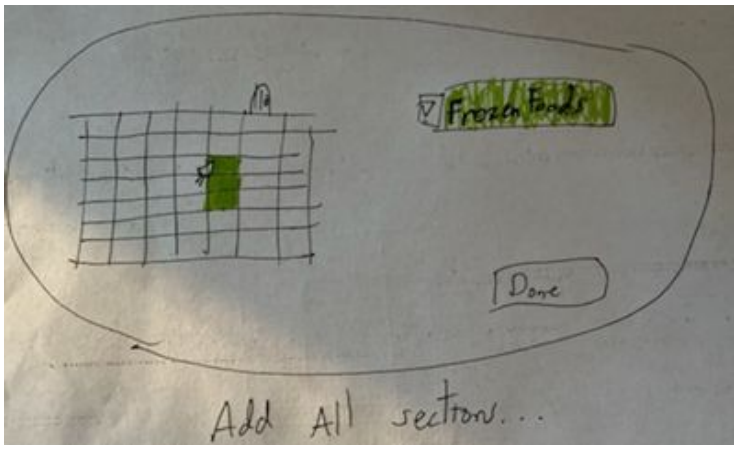
User is in the store creation screen. This contains store metadata so other application users can use the maps others have created. After group and customer discussions, there was a decision to add a size feature for the store on this screen that would affect the grid size proceeding.



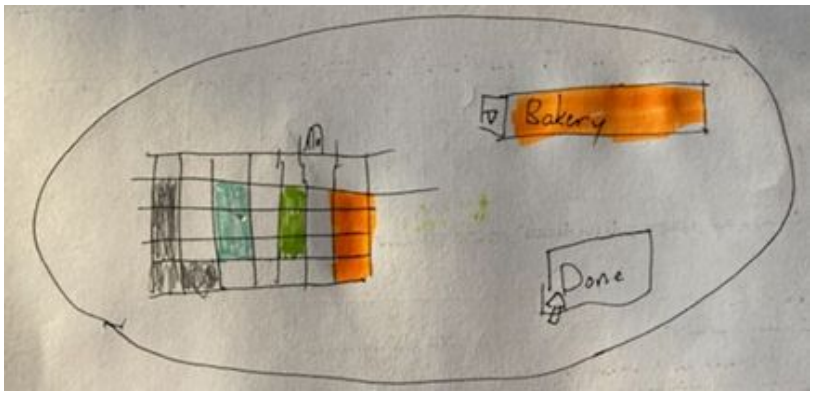
This is the first screen after a store has been created. There is now a 2D matrix hosted on the user's device being edited by cell selection. The drop-down menu on the right chooses the grocery section that clicking a cell will designate. There should also be an option for removing items from cells.



Here the drop-down is displayed, with a defined list of grocery section options. Choosing an option from the list causes clicking a cell to be designated as that section.

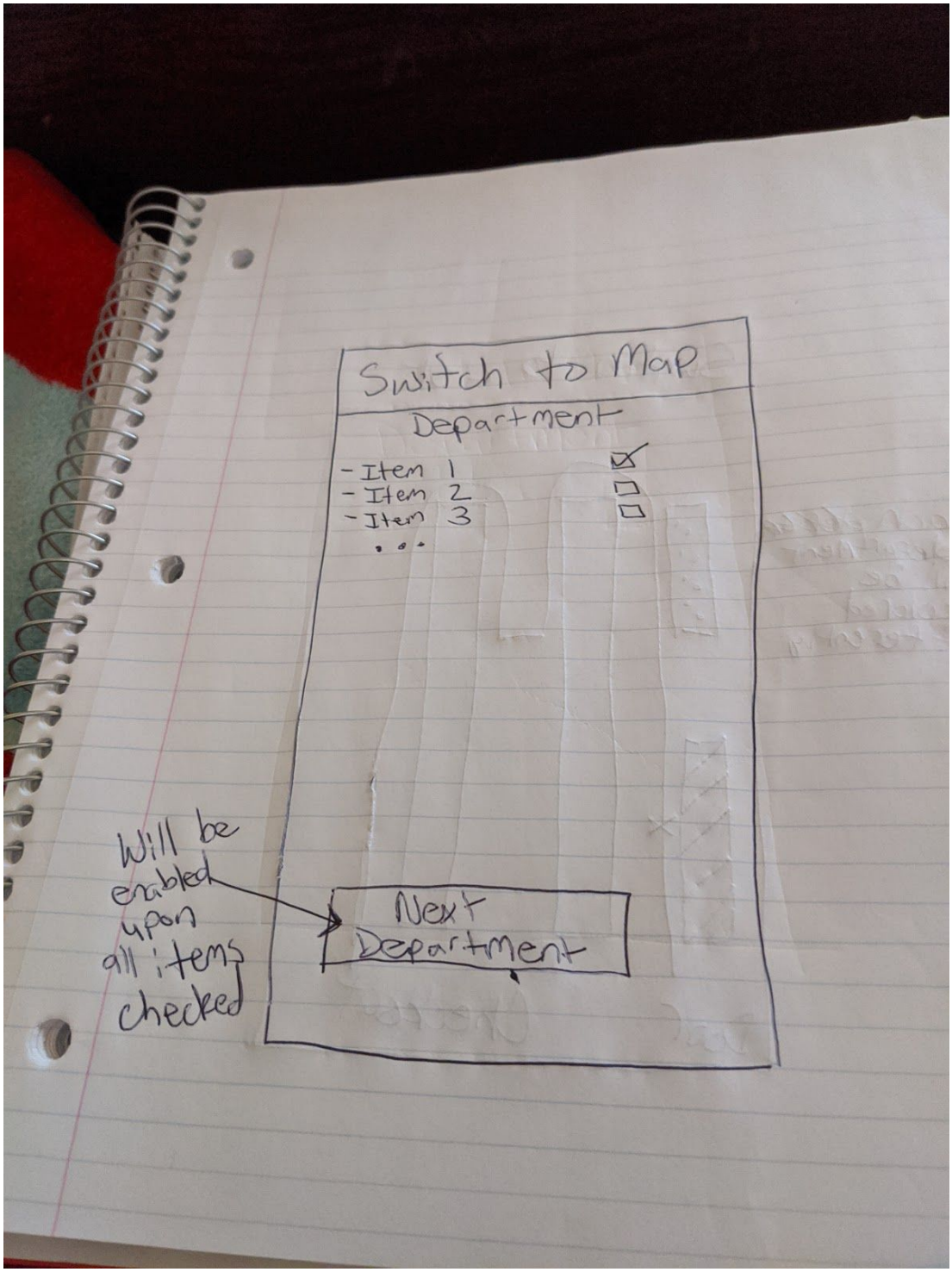


This demonstrates how selecting a cell designates that cell as the colored option. When surveying potential other users and the customer, they said they would prefer pictures or emojis to just a color if possible. This process is repeated until the map is completed.

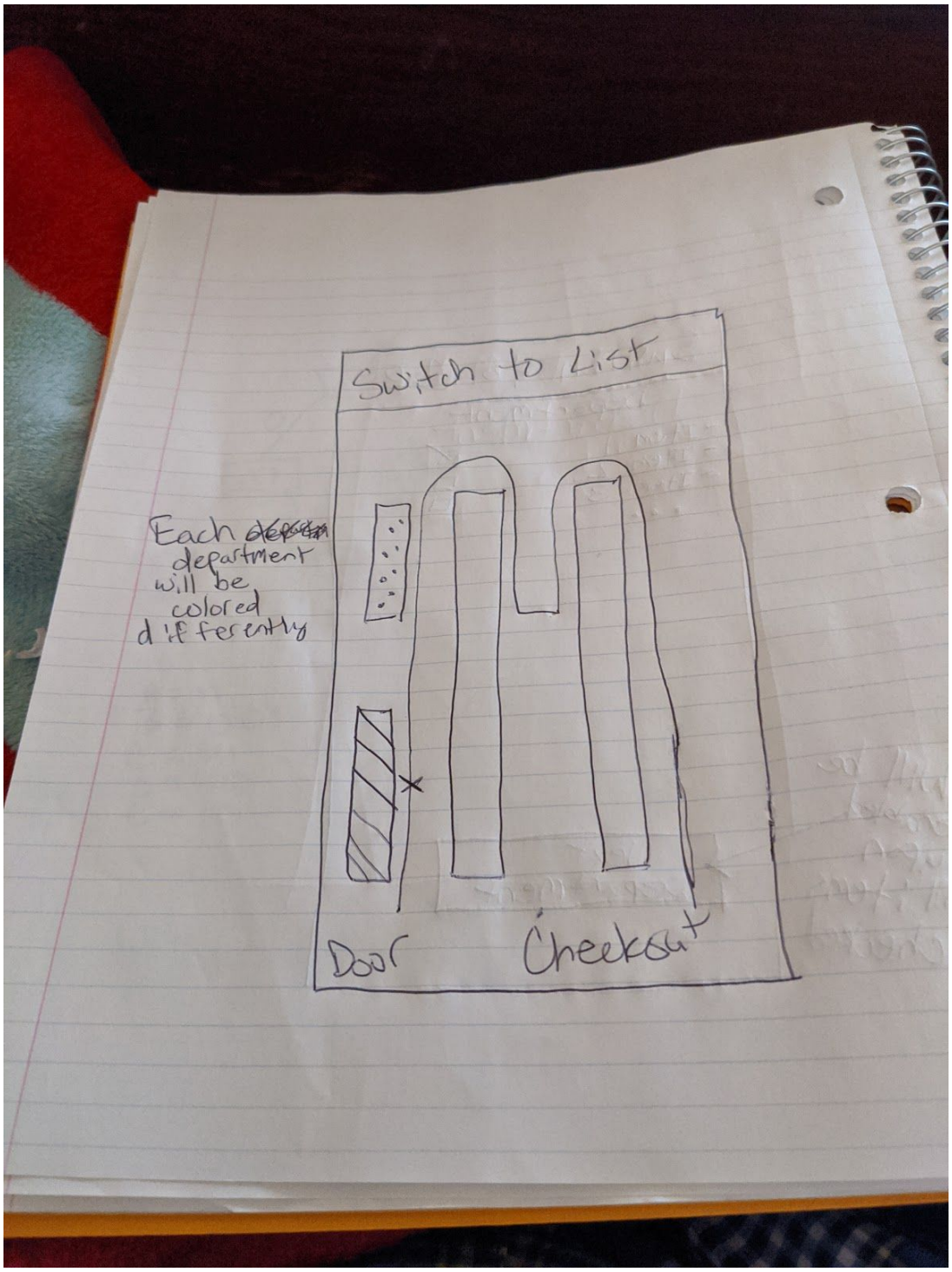


Once the map is completed, it can be pushed to the remote database by clicking “Done”.

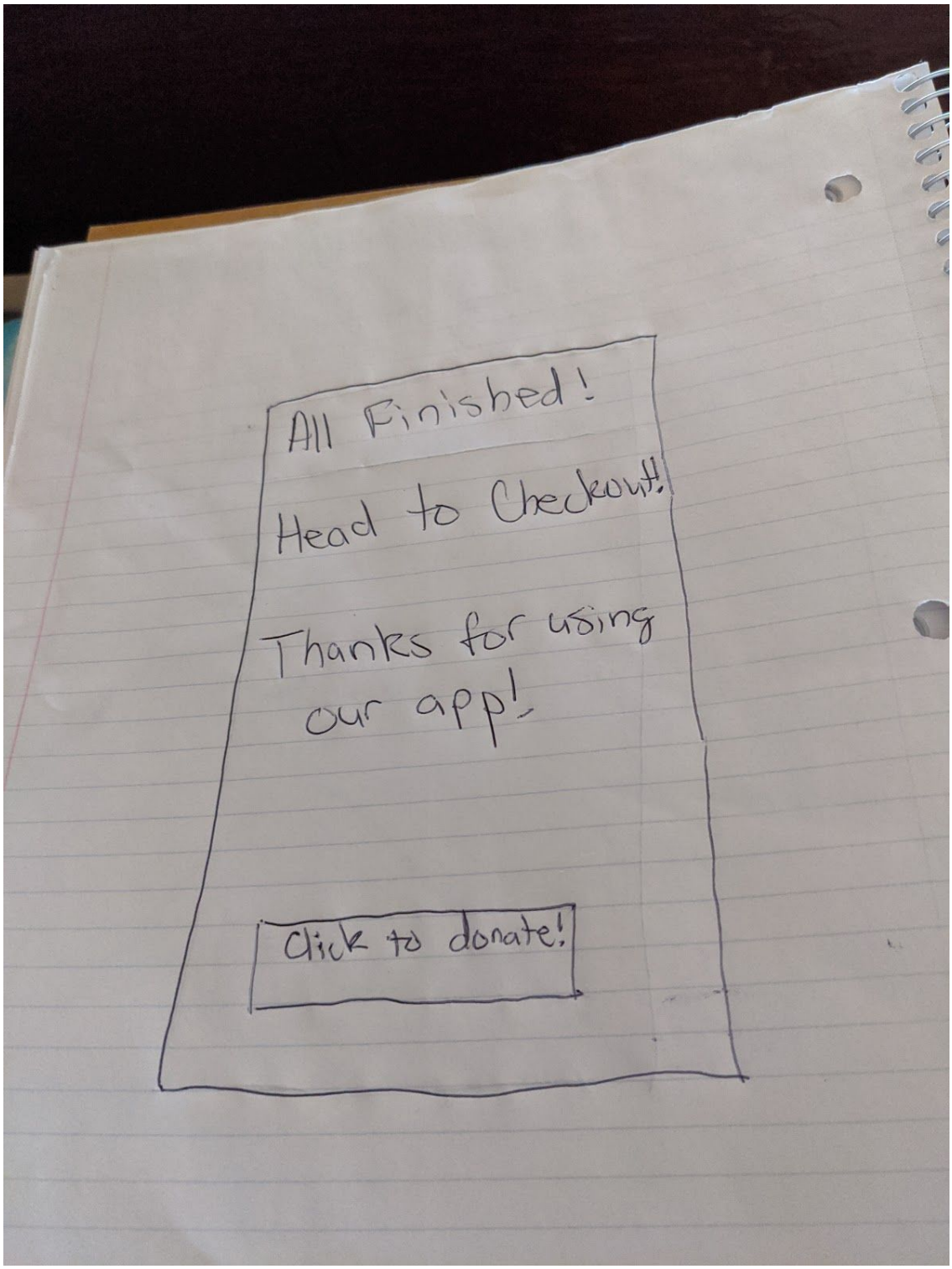
Use Case 3 - Marking items off List



User will be shown items in the department that the user is in, and will be able to mark items off as they grab them. The next department button will only be clickable upon checking all items on the list. The user can also switch to a map view of the store.



This will display a map view of the store. The map will display the optimal route and where the user currently is (with an ‘x’) as well as the departments. Each department will be color coded a certain color.


All Finished!
Head to Checkout!
Thanks for using our app!
Click to donate!

Functional Requirements (Environmental):

- The user should be able to create a categorized grocery list.
- The grocery list should have items that are in the same area put together.
- The user should be able to print off maps if they wish
- The user should be able to use a desktop version to create maps more easily.
- When creating store maps, there should be standard templates that the user can choose from as a starting point for the store map creation, and then further customize these templates to meet the needs of the user's grocery store.
- The user should be able to create and save shopping lists.
- The user should be able to obtain an optimized route through their desired grocery store using maps uploaded to the app in conjunction with their shopping list.
- The user should have options on what algorithm they want to use when creating their route:
    - Picking up the frozen food last.
    - Picking the fastest route.
    - Choosing the heaviest loads first or last.
- While shopping with the app, the user should be given an approximate location in the store for each item on the user's shopping list in the order of the optimized path
- The various locations in the grocery store should be represented by emojis rather than using just colors to represent different departments of the store to allow simple user recognition.
- Stores should be able to create the official store maps for their grocery stores
- The store should be able to update their official store maps
- The store should give more information about their store to improve the maps.
- The app should be able to have commercials to bring in revenue.
- The app should know what store departments certain grocery items will be.
- The app should not lead the user through the same section more than once.
- The app should have auto complete options when adding items to the list.
- The app should allow the user to view their optimized route in two ways:
    - Map view, where the user can see a map of the store with their optimized route laid out. The user is given only the department/aisle number/location in aisle of next item on the list.
    - List view, where the user is not given a map but instead sees each item on their shopping list in the optimized route order with directions from each item to the next
- The app should allow the user to check off items off their shopping list and then guide the user to the next item on their shopping list

Non-Functional Requirements (Environmental):

- The map should generate in a reasonable amount of time (less than 30 seconds)
- The map should be easy to read - standardized modeling system to build and display maps back to user. For example, the produce section looks the same on every map, regardless of who submits the map.
- The app should be convenient.

- The app should make visits to the store more convenient.
- The app should help save time in the grocery store compared to someone shopping without using the app to navigate the store
- The app should be easy to use.
- The app shouldn't cause a bunch of extra work.
- Stores should find it useful to keep their information updated.
- Stores should find it easy to keep their information updated.

Functional Requirements (System interfaces):

- The user should be able to use their phone or desktop.
- The user should be able to look at different stores through the app
- The user should be able to have an account on the app, stored in database, but the user shall not have to account to use the app. Certain features may not be available if the user does not have a registered account with the app, such as saved shopping lists.
- Privacy standards should be kept and carefully observed.
- The users should be informed on what data is gathered and for what purpose.
- The user should be able to switch between using their phone, laptop, or desktop with their information.
- It should be easy for users to see their own data.
- There should be a database that has information about the different stores, items in those stores, maps of stores, and the users stored data, including username/password, shopping lists, etc.
- Sensitive data should be encrypted in the database - in particular, passwords must be hashed
- Upon attempting to login, user credentials are verified with information stored in database
- While shopping, as user checks items off list, user's shopping list is updated to reflect which items user has picked up already
- The user should be able to manipulate the map, using touch, and/or mouse depending on what device is being used.
- As user/stores builds maps, the map information is automatically autosaved in database to prevent loss of information
- As user builds shopping lists, the information is automatically autosaved in database with no need for user to manually save
- The user should be able to build manually save "base maps" with associated stores to build the current list off of however
- There should be different kinds of accounts.
    - User
    - Store
- The algorithms for creating shopping routes within the store shall be sent to the user's device to allow user to switch views regardless of whether or not user has cell service inside store.

Non-Functional Requirements (System interfaces):

- User shall be informed and explicitly asked consent prior sending data to server for the first time - for example, upon creating first shopping list, user should be informed that this information is going to our server and be asked consent.
- Users should be shown their own data (maps, shopping lists, etc) within 5 seconds after logging in
- Sensitive data should be securely stored with current hashing algorithms
- Querying the database on the server should take no longer than 5 seconds for client to receive response from server
- Route optimization should be hosted on user device
- Upon user typing text while creating a shoppingl list, user's list will automatically save every 5 seconds
- Upon user or stores creating maps, user's/store's input will automatically save every 10 seconds
- The user shall be able to switch between the different algorithm routes (eq heavy items last vs cold items last) without having internet connection.

Use Case 1: Make grocery List

Actors
Application user/average grocery shopper

Preconditions
- User has at least four or five items they want to add to the grocery list
- User has a smartphone with grocery application already installed
- User has basic knowledge of how to use their smartphone
- User already has an account on application
- The user's preferred grocery store already has a store map uploaded/created in the application

Postconditions
- User successfully input their grocery list at their preferred grocery store
- User is able to view a map of their preferred grocery store with most efficient route to pickup the items on their shopping list
- User is able to toggle between the route selector to see most efficient routes when choosing to (a) pick up heaviest items last, (b) pick of cold/frozen items last, etc

Flow of Events
- Login - user opens up grocery app on smartphone and enters in their username/password
- Authentication - application authenticates user and takes user to the dashboard screen
- New shopping list - user selects button to create a new shopping list
- Add items to shopping list
    a. User begins typing name of first item on list
    b. App begins searching database as user types each letter, displaying list of possible items that user is searching for
    c. After typing three or four letters of item, user selects item which they are looking for
    d. Item is added to the shopping list and user repeats a-c for remaining items on list
- Shopping map view - user is now able to select "Go shopping" which displays a map of their preferred grocery store and shows them the quickest route through store
    a. Frozen items last map view - user is able to select a "frozen items last" filter which alters the user's route through the grocery store to the most efficient route that ensures the user picks of frozen items last
    b. Heaviest items last map view - user is able to select "heaviest items last" filter which alters the user's route through the grocery store to the most efficient route that ensures the user picks up the heaviest items last (add additional option to allow user to change weight limit for what constitutes heavy items?)
- Shopping list view - user now has option on the "Go shopping" page to select "List view" which displays an ordered list of their grocery list at their preferred grocery store in the order which the user should pick up items to follow the most efficient route

    a. Frozen items last list view - user is able to select a “frozen items last” filter which alters the user’s route through the grocery store to the most efficient route that ensures the user picks of frozen items last
    b. Heaviest items last list view - user is able to select “heaviest items last” filter which alters the user’s route through the grocery store to the most efficient route that ensures the user picks up the heaviest items last (add additional option to allow user to change weight limit for what constitutes heavy items?)
- User is all set to go to the grocery store!

Use Case 2: Creating map of store

Actor

Customer or business manager wanting a map for their local store

Preconditions

- User has a smartphone or desktop to create the map with, and some basic familiarity with computers
- User has account registered on application
- Application is installed
- Store is fairly typical in layout and items (deli, bakery, international, snacks, etc)
- Store layout/inventory does not change dramatically often (i.e. farmer’s markets, overstock stores)
- Standard store templates are already loaded into application for user to choose from as a starting point for creating custom store

Postconditions

- Created data structure of nodes each representing a section of the store, and their spatial relationships with one another.
- Map should be displayed in easy to understand format, similar to what was submitted in project proposal
- The map should be uploaded to a database for others to use.

Flow of Events

- User logs in/creates account on desktop or smartphone
- User selects “Create new map for my store” option
- User enters store metadata (name, address, city, state, ZIP, username autofilled)
- Go to map creation screen, create map locally on phone. Blank grid map appears on screen)
- User selects grocery section or feature from the list of options, adds squares to map to represent location in store
- User repeats until they are satisfied with map
- User clicks to save map

- Grid is converted into data structure for pushing to database on user's phone, tagged with metadata and pushed.
- Backend store database receives store map, makes available to other users if desired
- Metadata on store map is tracked on backend database (date last updated, # of get requests, reviews, etc)

Use Case 3: Check Items off Grocery List as User is Shopping

Actor
Grocery shopper using app

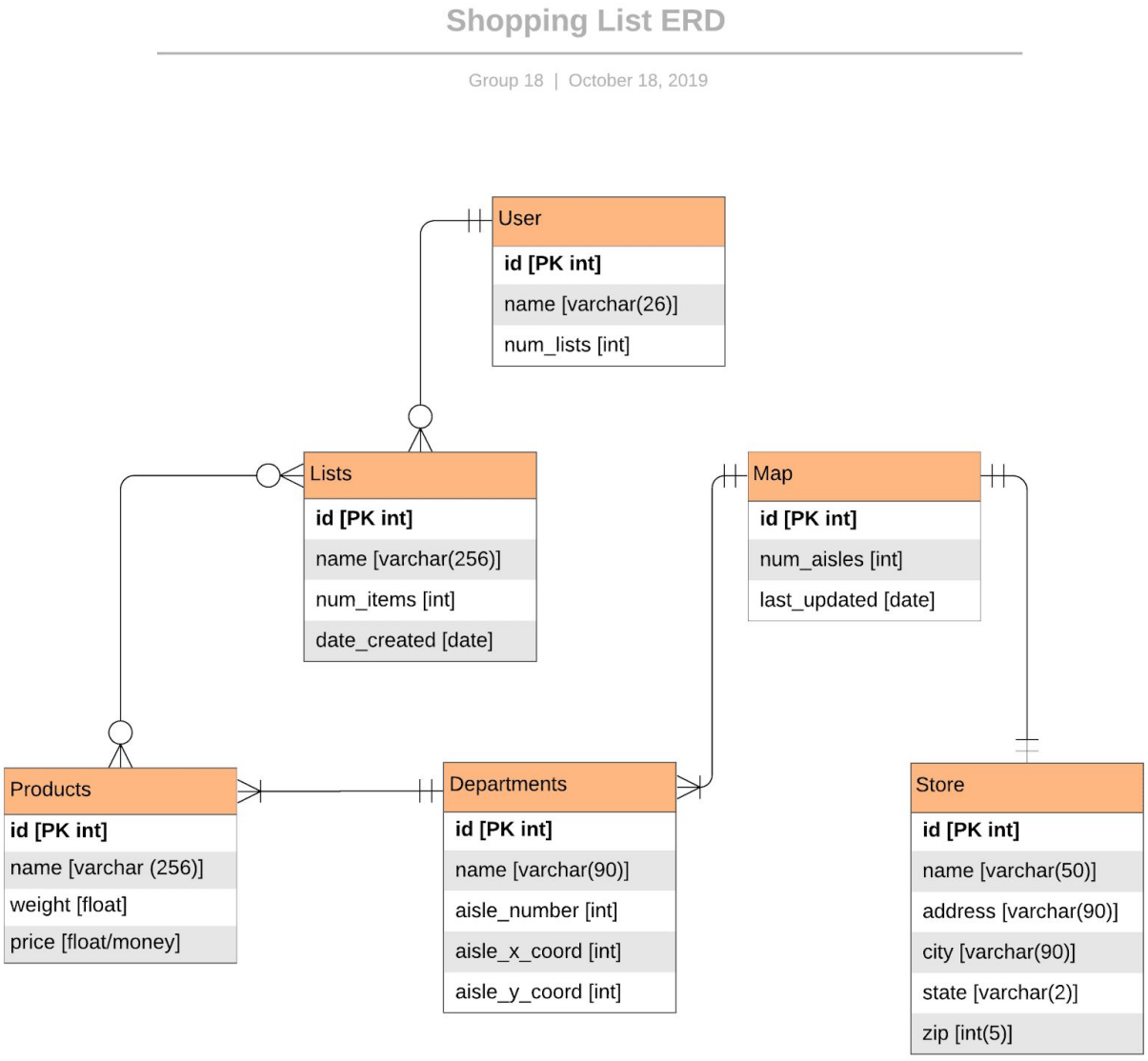
Preconditions
- User has existing grocery shopping list
- A grocery store map has already been created in the application for the user's grocery store of choice
- Customer has account registered
- Grocery list consists of items that would be expected to be found at target store
- Grocery shopper user is physically at the grocery store location where they will begin shopping using the application

Postconditions
- Items were removed from grocery list in desired order (shortest route, heavy items last, cold items last)
    - In flow of events, user chose to pick up cold items last
- User has successfully collected all items on shopping list from grocery store in optimized order
- Customer/user is directed to checkout with all items on list in shopping cart

Flow of Events
- User logs into application
- User selects correct grocery list
- User selects filter option to pick up cold items last
- User selects begin shopping
- User is directed by application where to start in the store
    - User is shown map view of store and application tells user where exactly to head to in the store (department, aisle number,and location in aisle)
    - User may choose "List view" which removes the map from screen and instead shows user each item on shopping list, in the order that user should pick up each item, with department, aisle number, and location in aisle of each item
    - User is able to change sorting algorithm to pickup heavy items last or cold items last, or just the fastest route, without having internet during this shopping process
- User proceeds to first department/aisle as determined by application
- User grabs first item off store shelf
- User removes item from list by selecting next on user interface
- List updates to show next item on list and tells the user which aisle to head to
    - If department change, will inform user
    - Map is updated to show user next location in store (if user has map view selected)
    - Grocery list is updated to show user next location in store in list view (if user has list view selected)
- User continues to repeat the previous four steps until all items are checked off list, except cold items

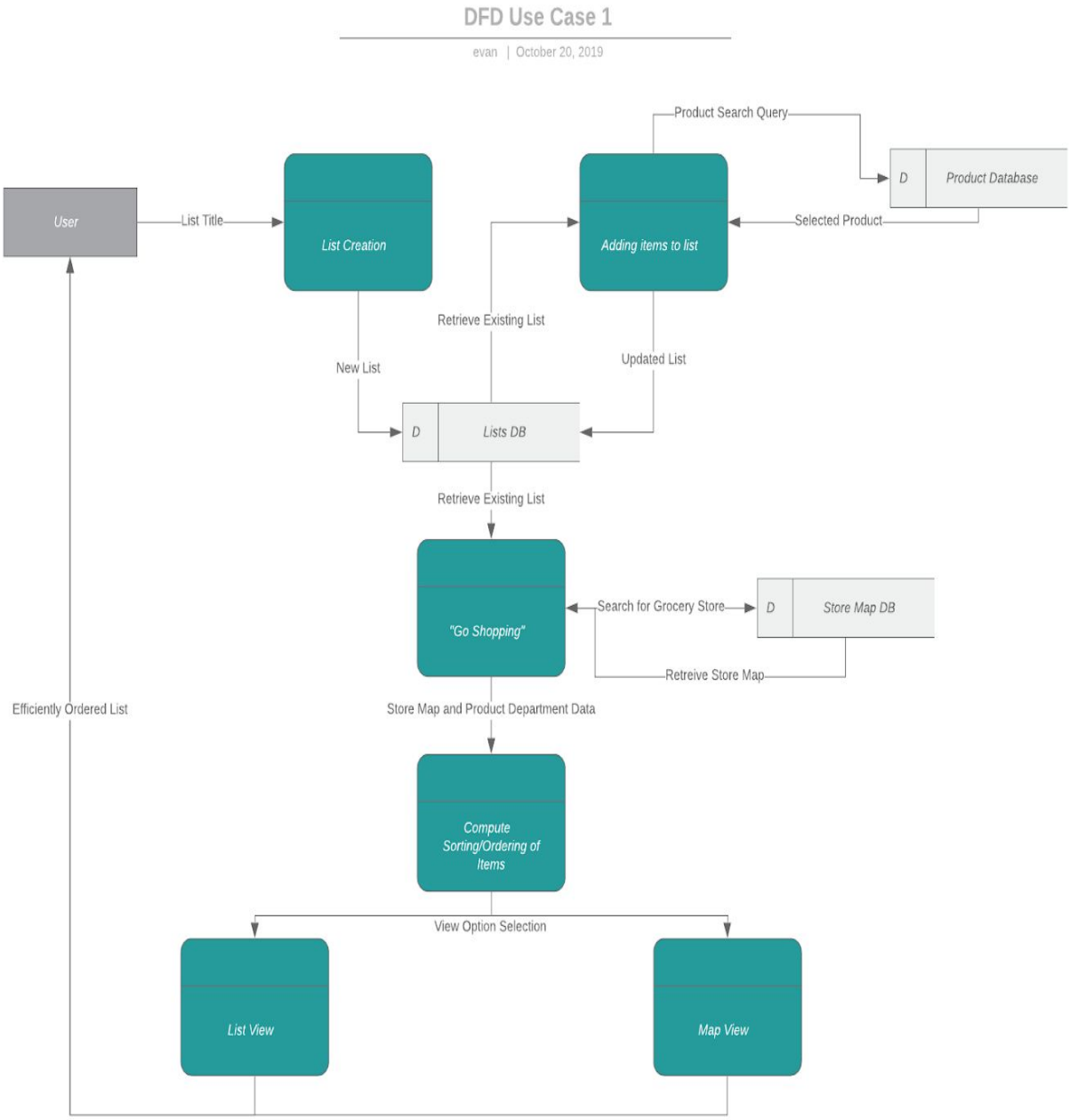
- User is now directed by application to go to cold/frozen section of store
- User is given aisle number/aisle location of first cold item that user should pick up first
- User grabs item out of cold case
- User removes item from list by selecting next on user interface
- List updates to show next cold item on list and tells the user which aisle to head to
    - Map is updated to show user next location in store (if user has map view selected)
    - Grocery list is updated to show user next location in store in list view (if user has list view selected)

ERD:

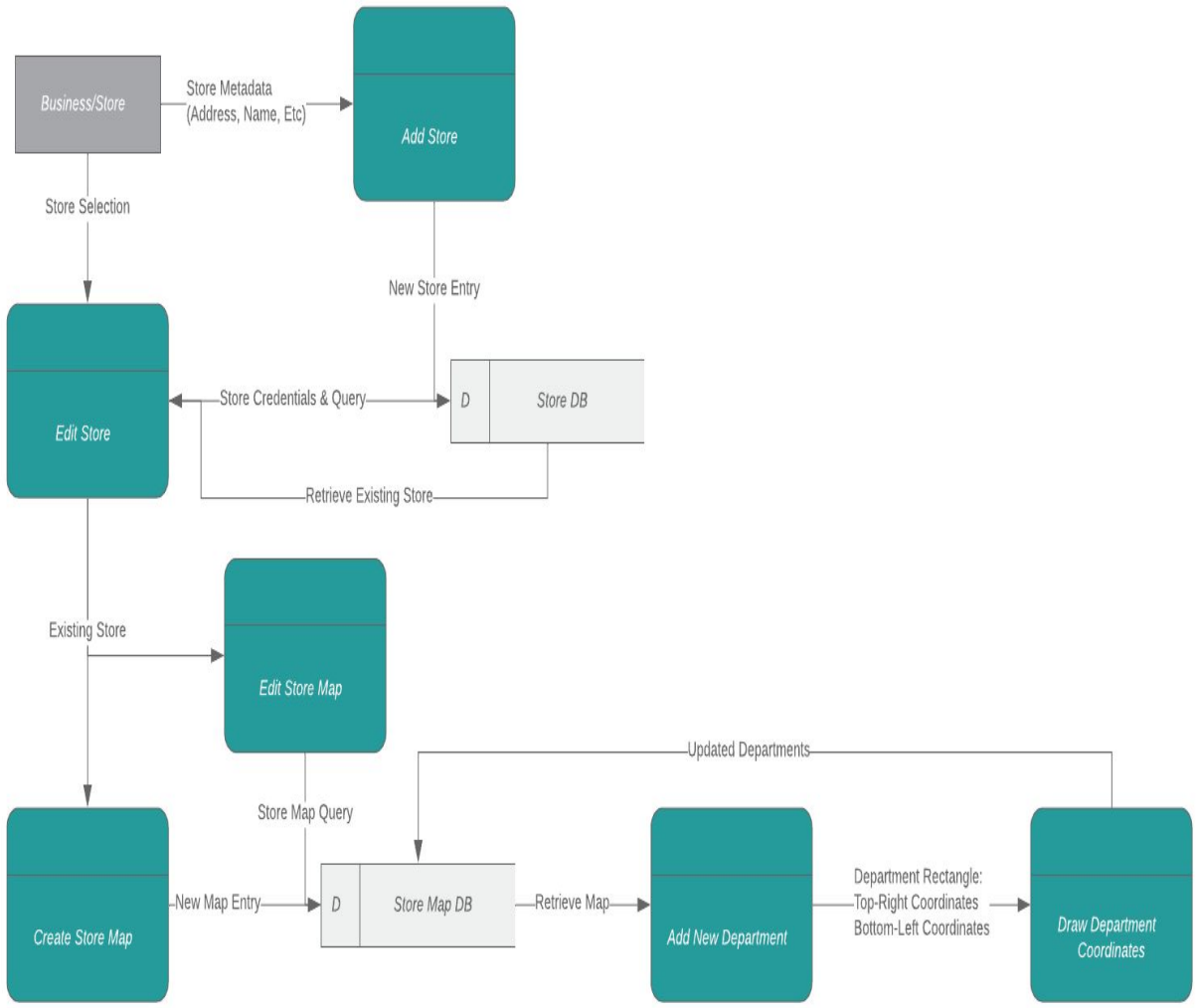

Shopping List ERD
Group 18 | October 18, 2019
User
id [PK int]
name [varchar(26)]
num_lists [int]
Lists
id [PK int]
name [varchar(256)]
num_items [int]
date_created [date]
Map
id [PK int]
num_aisles [int]
last_updated [date]
Products
id [PK int]
name [varchar (256)]
weight [float]
price [float/money]
Departments
id [PK int]
name [varchar(90)]
aisle_number [int]
aisle_x_coord [int]
aisle_y_coord [int]
Store
id [PK int]
name [varchar(50)]
address [varchar(90)]
city [varchar(90)]
state [varchar(2)]
zip [int(5)]

Data Flow Diagrams:

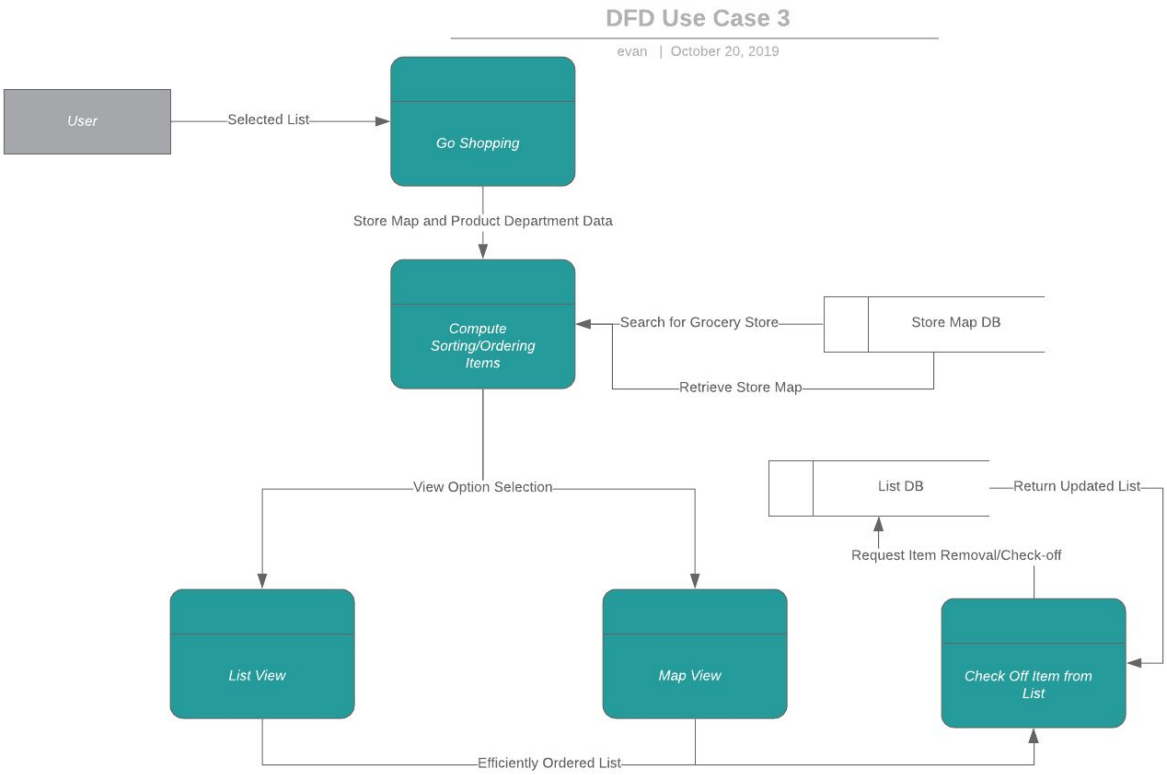
(Use Case 1):



(Use Case 2):

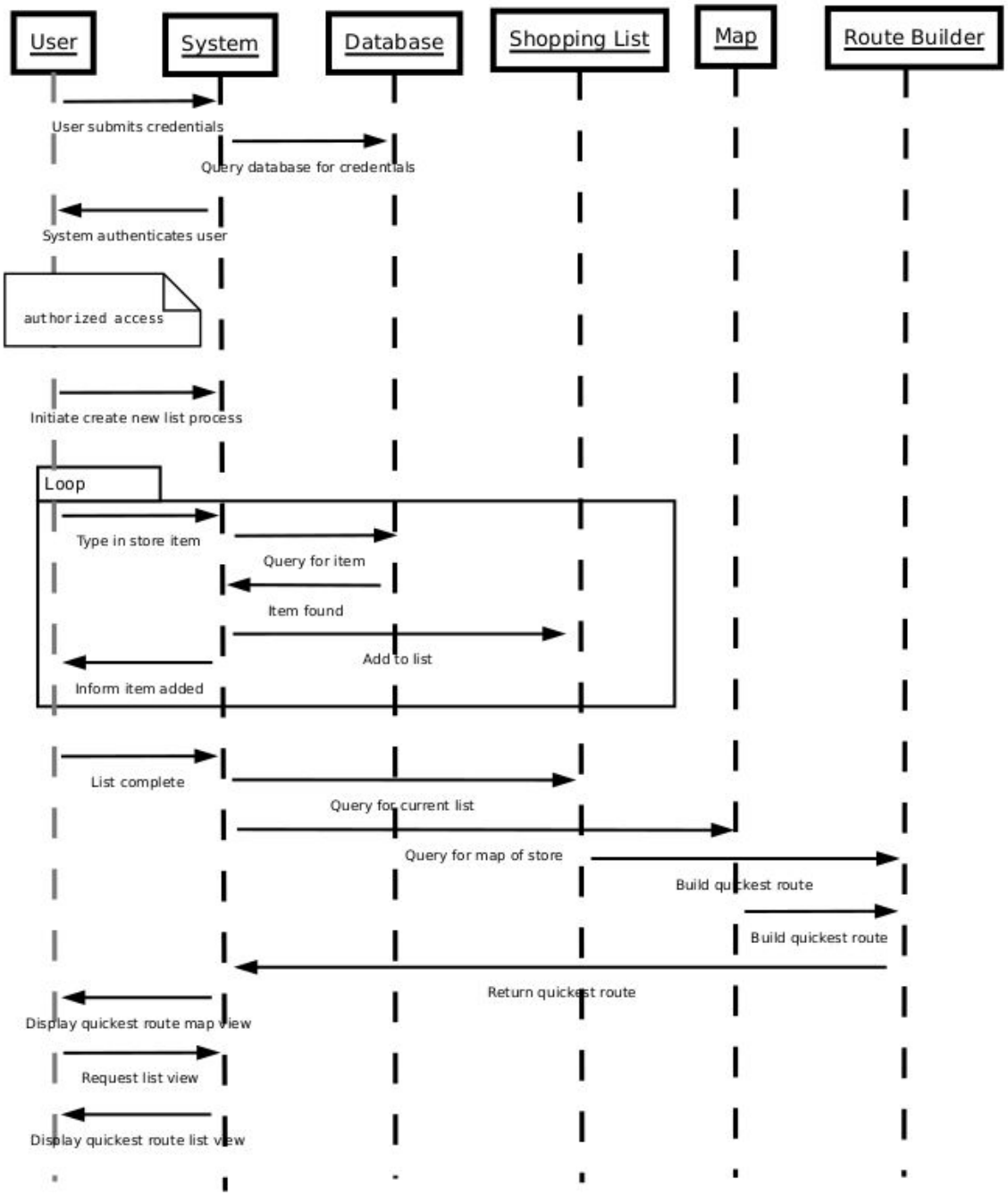

Business/Store
Store Metadata
(Address, Name, Etc)
Add Store
Store Selection
New Store Entry
Edit Store
Store Credentials & Query
D
Store DB
Retrieve Existing Store
Existing Store
Edit Store Map
Updated Departments
Store Map Query
Create Store Map
New Map Entry
D
Store Map DB
Retrieve Map
Add New Department
Department Rectangle:
Top-Right Coordinates
Bottom-Left Coordinates
Draw Department
Coordinates

(Use Case 3):

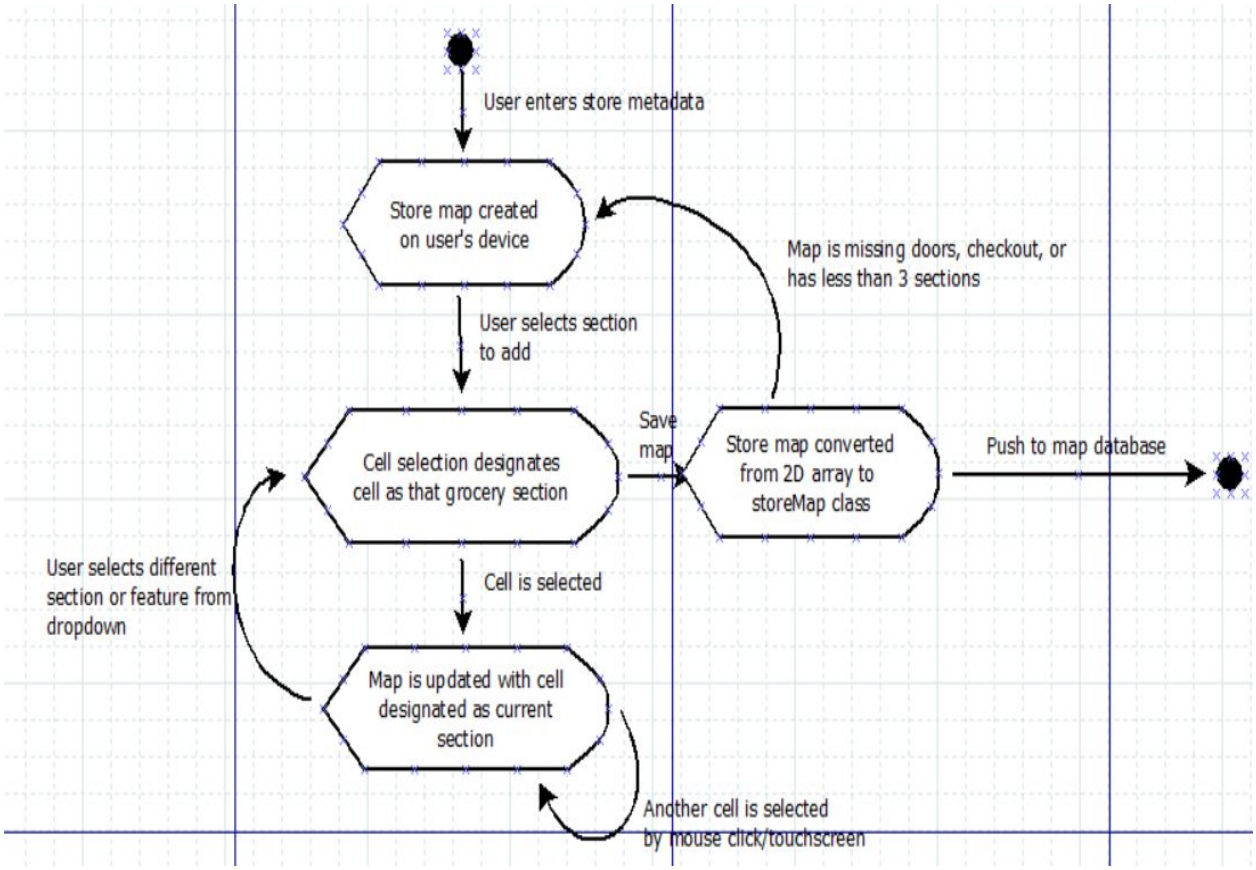

DFD Use Case 3
evan | October 20, 2019
User
Selected List
Go Shopping
Store Map and Product Department Data
Compute Sorting/Ordering Items
Search for Grocery Store
Store Map DB
Retrieve Store Map
View Option Selection
List DB
Return Updated List
Request Item Removal/Check-off
List View
Map View
Check Off Item from List
Efficiently Ordered List

Message Sequence/State Chart (Dane/Chris/Anothony):

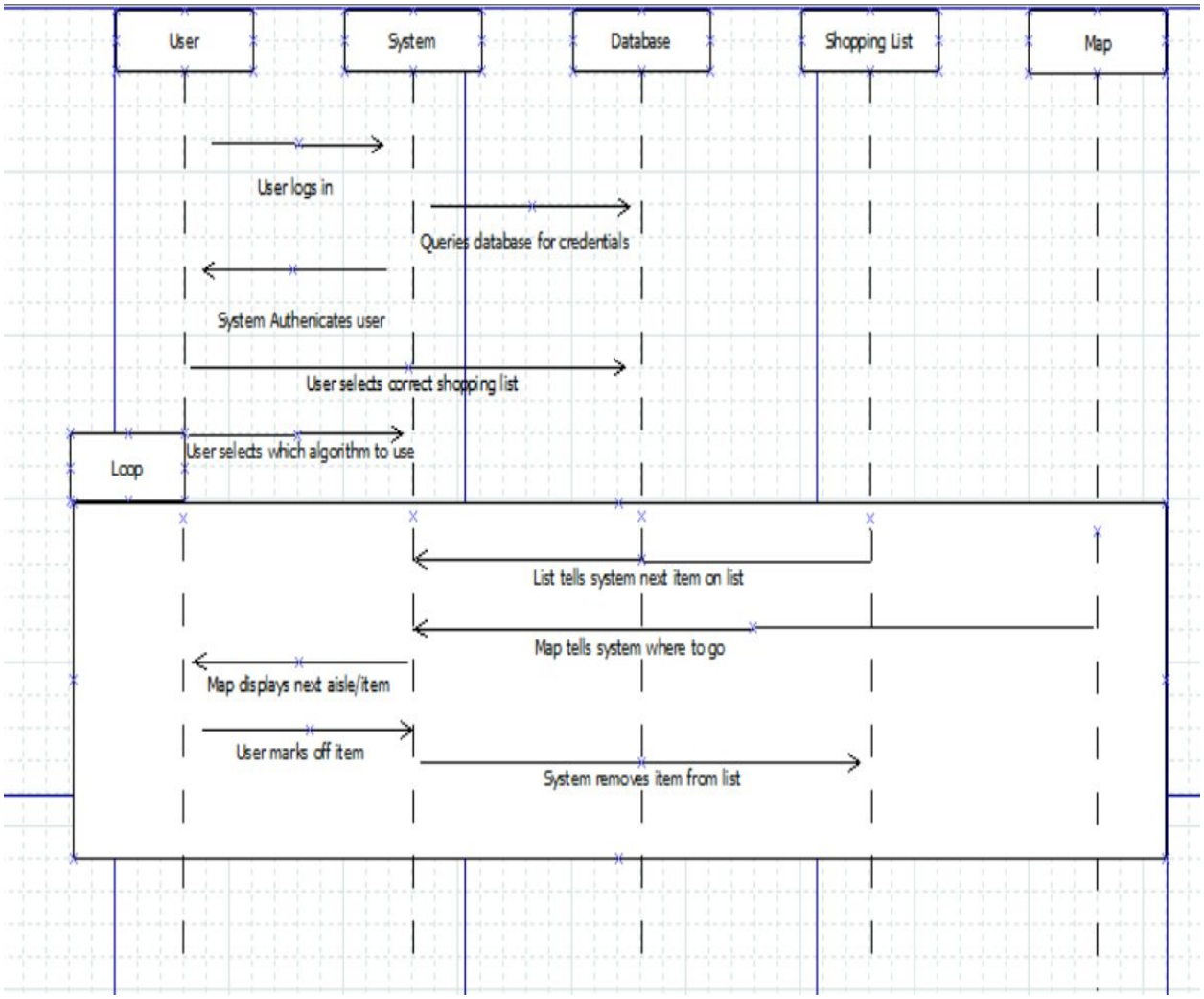
Message Sequence Diagram for Case 1 - Building a Grocery List



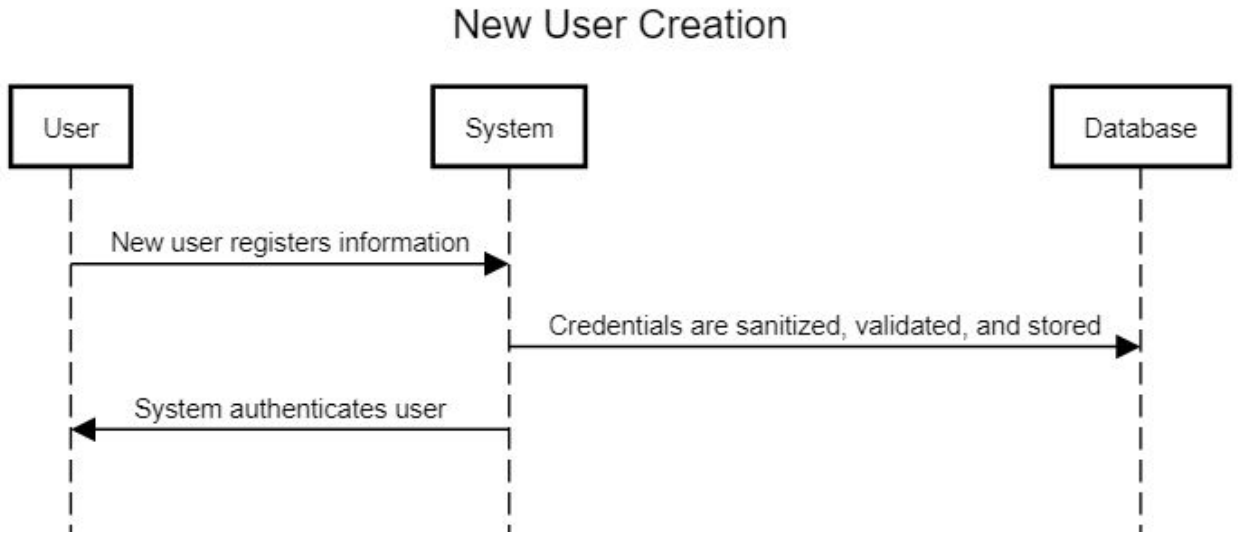
State Chart for Use Case 2 - Creating a store map



Use Case 3: Marking items off list



User Creation Message Sequence Diagram:



Requirement Definition and Specification Changes

· When creating a store map, there should be multiple sizes to pick from, representing common store sizes. This would relate to a larger grid on the screen, requiring another metadata tag on maps along with various scrolling changes. These different store sizes would represent a starting point for the user, and the user would be allowed to add/remove from the store size in order to fully customize their grocery store, as described in the specifications. A grid bar on the map creation process will allow the user to scale the different parts of their store appropriately.

· The customer and user feedback suggested the need for an option to toggle between a list and map view when picking through a list (use case #3). The list view should show a small list of items to grab from the current section, with the next section moving up from the bottom of the list as items from the current section are ticked off. This allows the user to choose the optimal short-path route within a department. The map view should show the user the route through the store, with the next item displayed above the map and brief directions for where the user to head to pick up the next item on their grocery list.

· The customer also liked the idea of saving lists and associating them with stores for later use. Presumably this information would be associated with the user account and stored on their device. Users would then be able to see their old shopping lists, as shown in our paper prototypes, and re-select these old shopping lists to be used as their current shopping list. The user could take an old shopping list, add or remove items off that list, and then click “go shopping” to be able to reuse this old shopping list as a template.

· User feedback, however, desired an option to use the application as a guest. The customer agreed that we should not chase away potential users by requiring accounts, but those without an account should not be able to save maps for others. We have added in our requirements that we will allow the use of this application without an account, but many features, such as saved grocery lists, may not be available if the user does not have an account with which to associate their shopping lists.

· The customer iterated that he wants minimal whitespace in the UI of the product (non-functional requirement). We will make sure to take this into consideration during the later stages of the application development.

· The customer and user feedback both preferred pictures or emojis representing areas of the map over colors. For example, rather than just representing the produce section of the store by a solid color, something like a head of broccoli and a carrot will be used on the map view to convey to the user that this is the produce section of the store. This will help the user quickly identify sections of the store without needing to remember something like “green means produce”. While this is a mostly trivial change, it could mean the map will render differently on various mobile OS’s. In future steps of this project, we will keep this in mind.

· The customer explicitly desires map and route-building algorithms being hosted on the user’s device. This will help free up the servers to serve more people, as well as allow the user to change the routing method inside a store, where the user may or may not have cell service. This will ultimately improve the user experience.

· We explicitly specified that the app should allow the user to checkoff items while shopping, and that the app should guide the user to the next item in the store as the user checks off items off the shopping list. While this is a central feature of this application, we had not explicitly stated this in our initial requirements.

· Additionally, we updated our use cases to incorporate the changes that were made to the requirements. For example, in the use case where the user is creating a store, the user is able to select from the standard store templates, and then further customize this template to create their grocery store.

· Finally, we added in a message sequence diagram for the user creation process to make this process more explicit

Contributions and Customer Interaction:

Login and Landing Page Paper Prototypes (Briana):

User Case 1 Paper Prototypes (Dane):

User Case 2 Paper Prototypes (Chris):

User Case 3 Paper Prototypes (Anthony):

Functional and Non-Functional Requirements for environment and system (Briana):

Data Flow Diagram (Evan):

3 use cases:

1. Make grocery list (Dane)
2. Make store maps (Chris)
3. Check items off grocery list (Anthony)

Requirement Definition and Specification Changes (All of group):

We were able to meet with our customer, Kevin Davis, on the evening of Friday, October 25, to discuss the paper prototypes of the grocery shopping application and talk through our group's questions.