Grocery Shopping App Requirements

(Name coming soon!)

10/20/2019

Christopher Brown
Dane Emmerson
Evan Horne
Anthony Huynh
Briana Willems

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.
- 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
- 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

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
- 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 route optimization algorithm will run on the server to free up client device
- There should be different kinds of accounts.
    - User
    - 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
- Running route optimization algorithm should take no longer than 10 seconds for client to receive response from server
- 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

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)

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
- Data structure should be saved according to store location and “created by”, uploaded for other application users to select.
- Users default store should be set to created map.

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, loaded into user profile’s default store
- 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

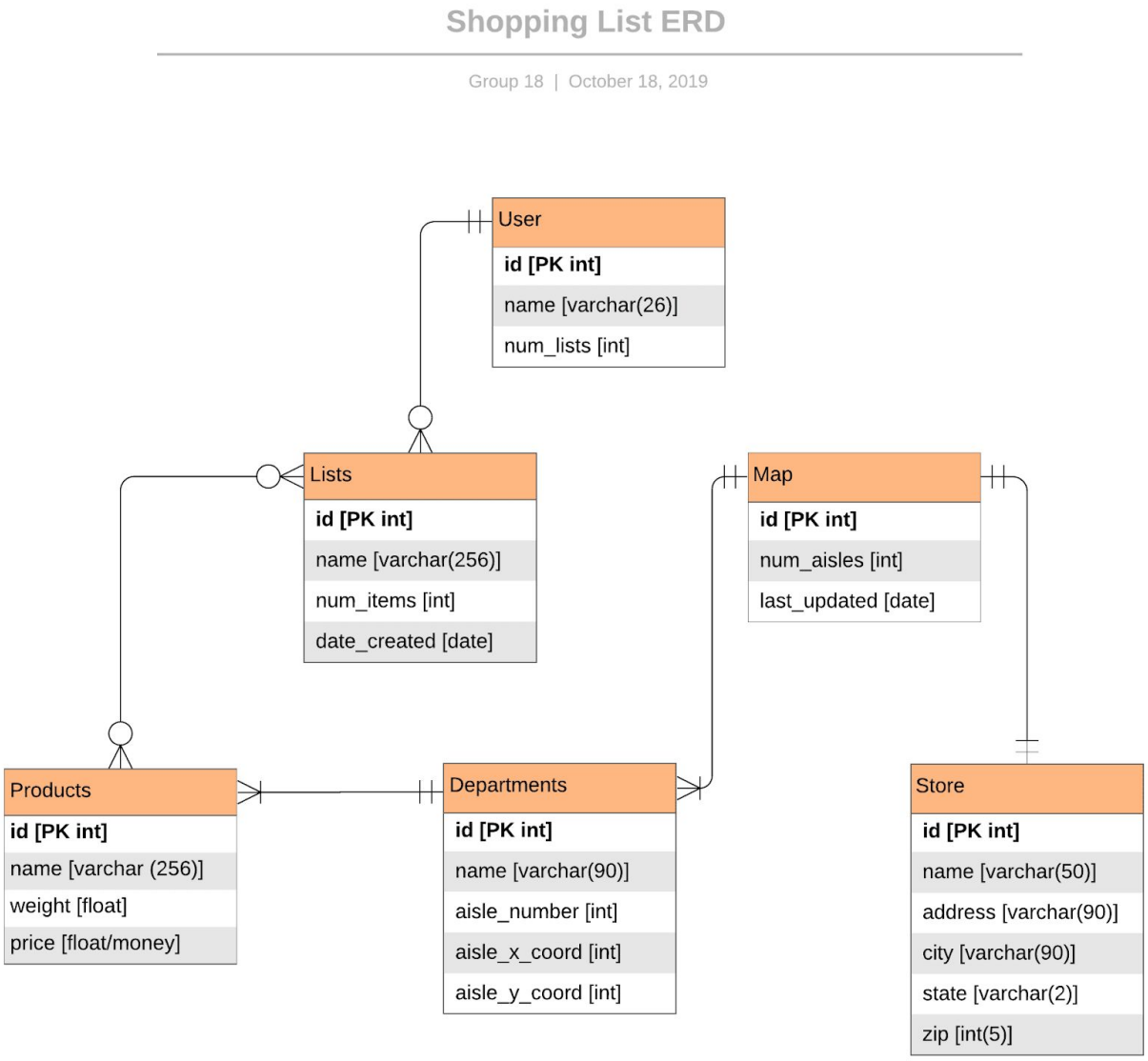
- Items were removed from grocery list in desired order (shortest route, heavy items last, cold items last)
    - In flow of events, user chose to pick up cold items last
- User has successfully collected all items on shopping list from grocery store in optimized order
- Customer/user is directed to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

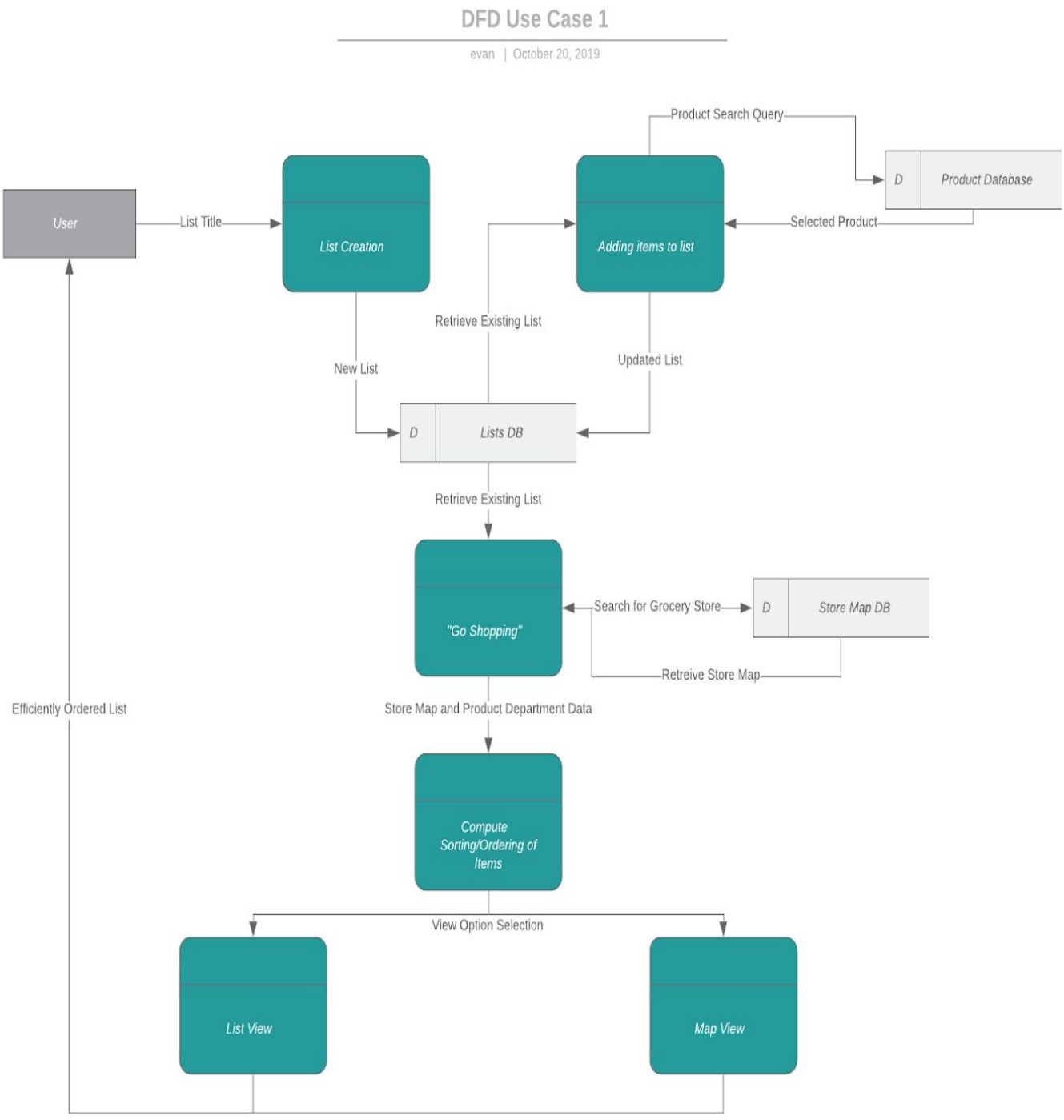
- Map is updated to show user next location in store (if user has map view selected)
- Grocery list is updated to show user next location in store in list view (if user has list view selected)

ERD:

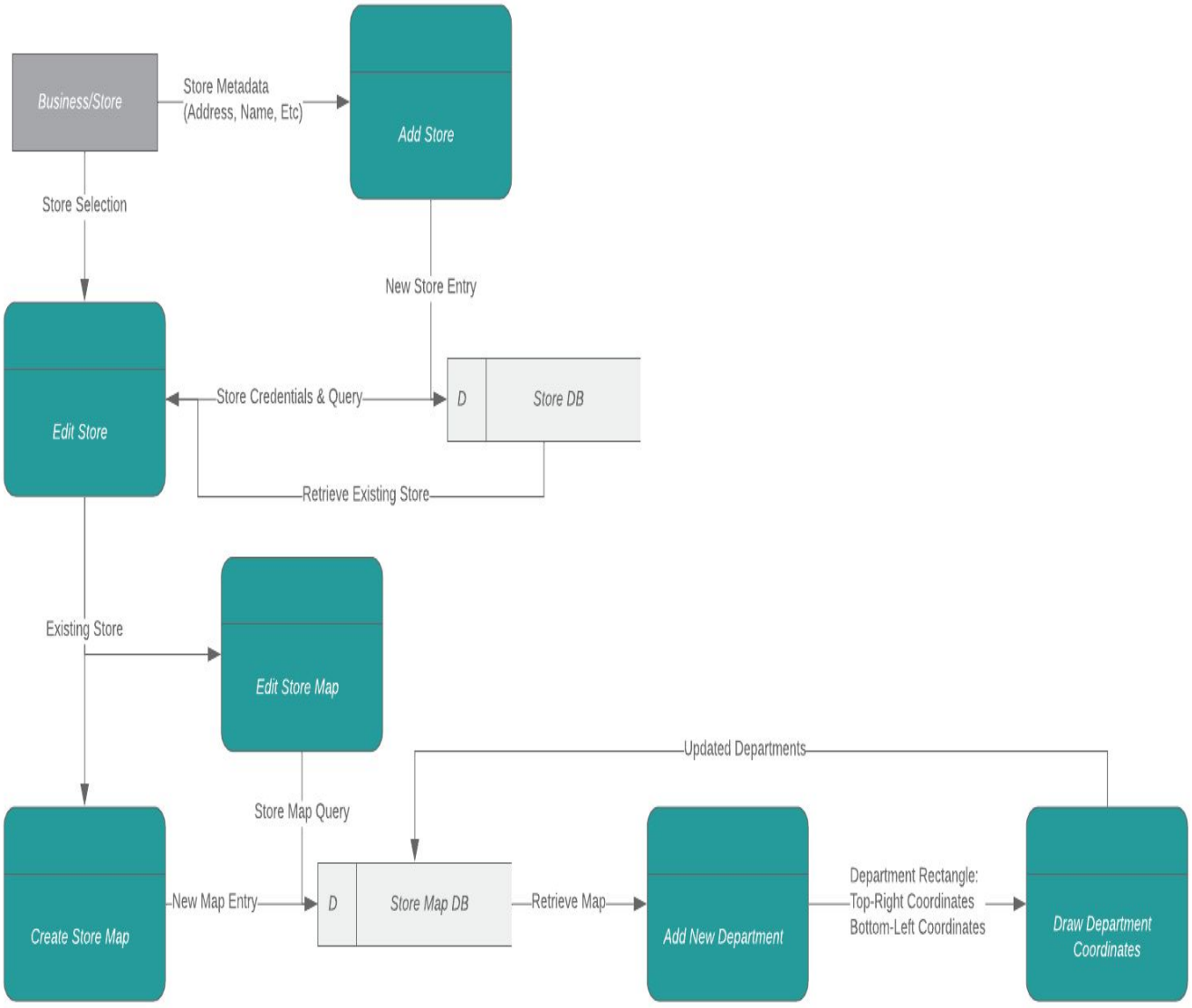

Shopping List ERD
Group 18 | October 18, 2019
User
id [PK int]
name [varchar(26)]
num_lists [int]
Lists
id [PK int]
name [varchar(256)]
num_items [int]
date_created [date]
Map
id [PK int]
num_aisles [int]
last_updated [date]
Products
id [PK int]
name [varchar (256)]
weight [float]
price [float/money]
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:

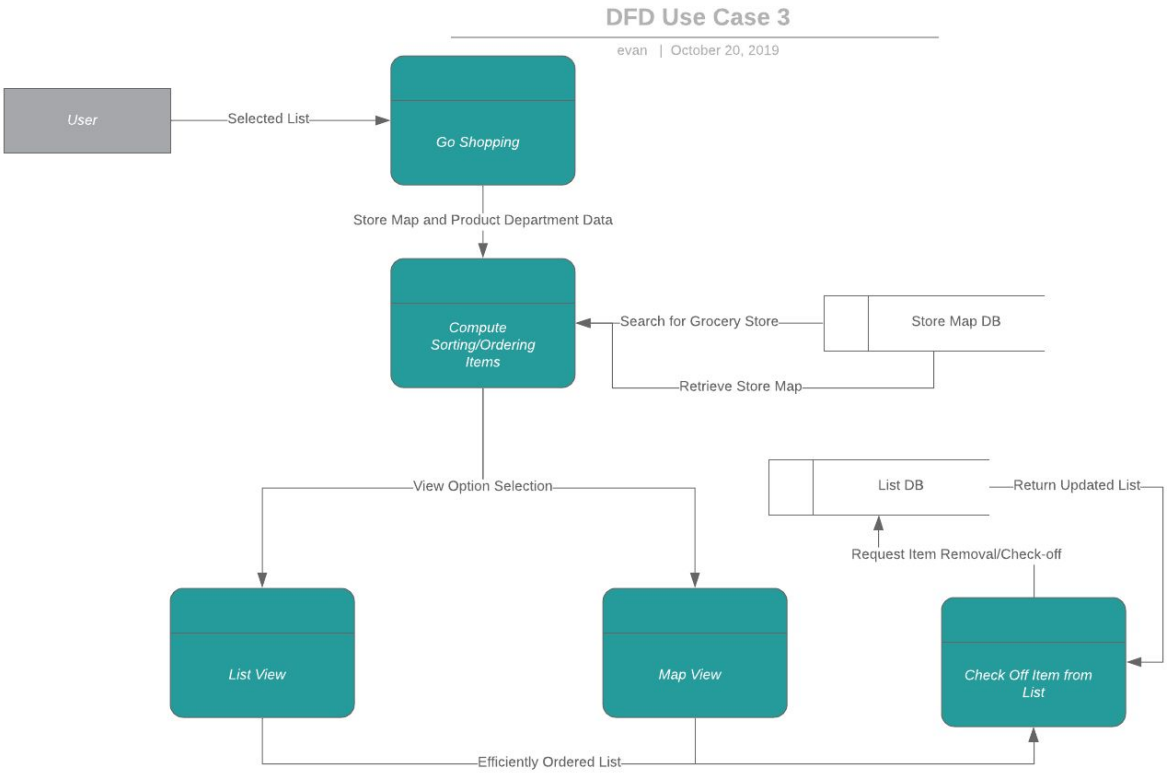
(Use Case 1):



(Use Case 2):

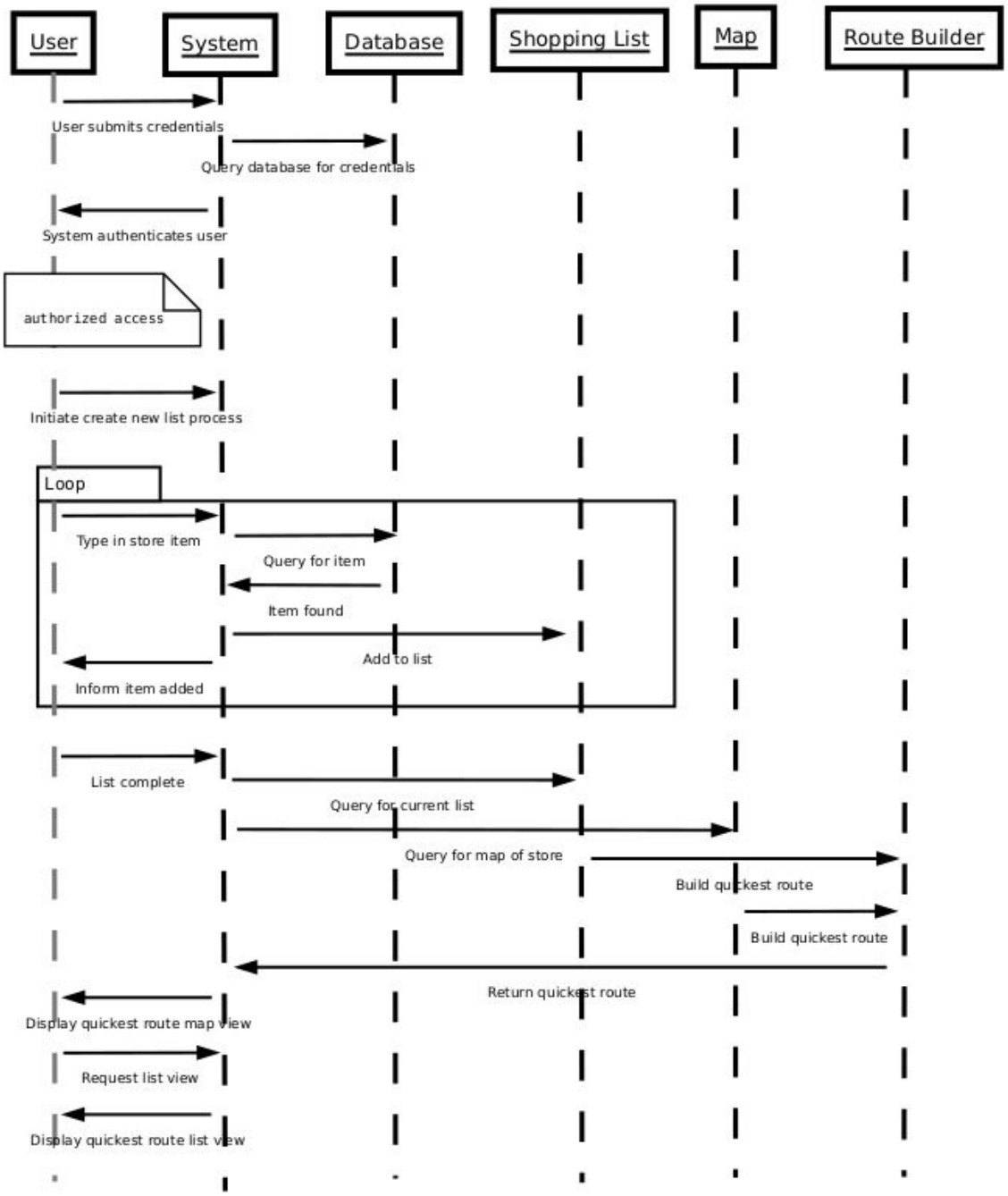

Business/Store
Store Metadata
(Address, Name, Etc)
Add Store
Store Selection
New Store Entry
Edit Store
Store Credentials & Query
D
Store DB
Retrieve Existing Store
Existing Store
Edit Store Map
Updated Departments
Store Map Query
Create Store Map
New Map Entry
D
Store Map DB
Retrieve Map
Add New Department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

(Use Case 3):

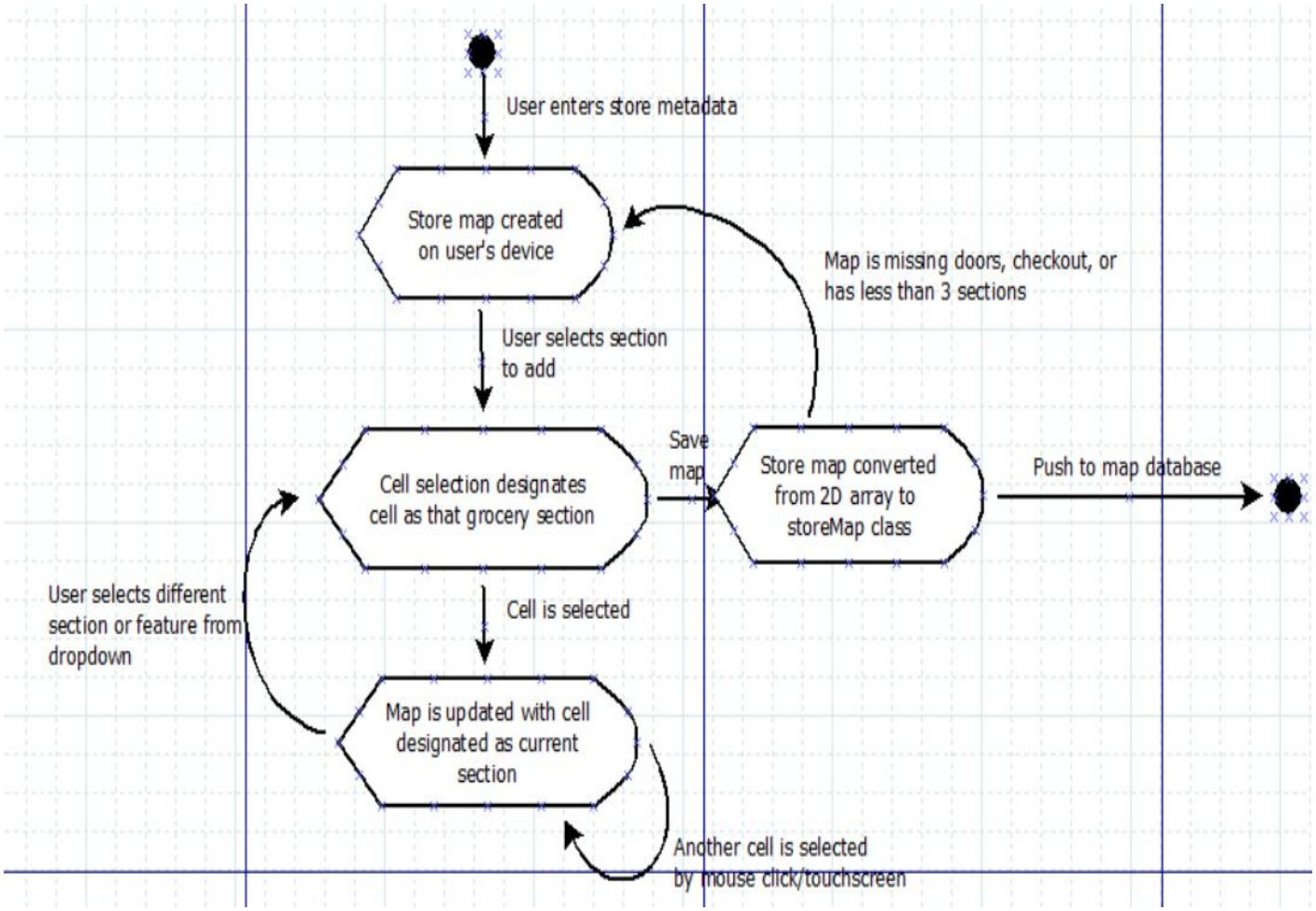

DFD Use Case 3
evan | October 20, 2019
User
Selected List
Go Shopping
Store Map and Product Department Data
Compute Sorting/Ordering Items
Search for Grocery Store
Store Map DB
Retrieve Store Map
View Option Selection
List DB
Return Updated List
Request Item Removal/Check-off
List View
Map View
Check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

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

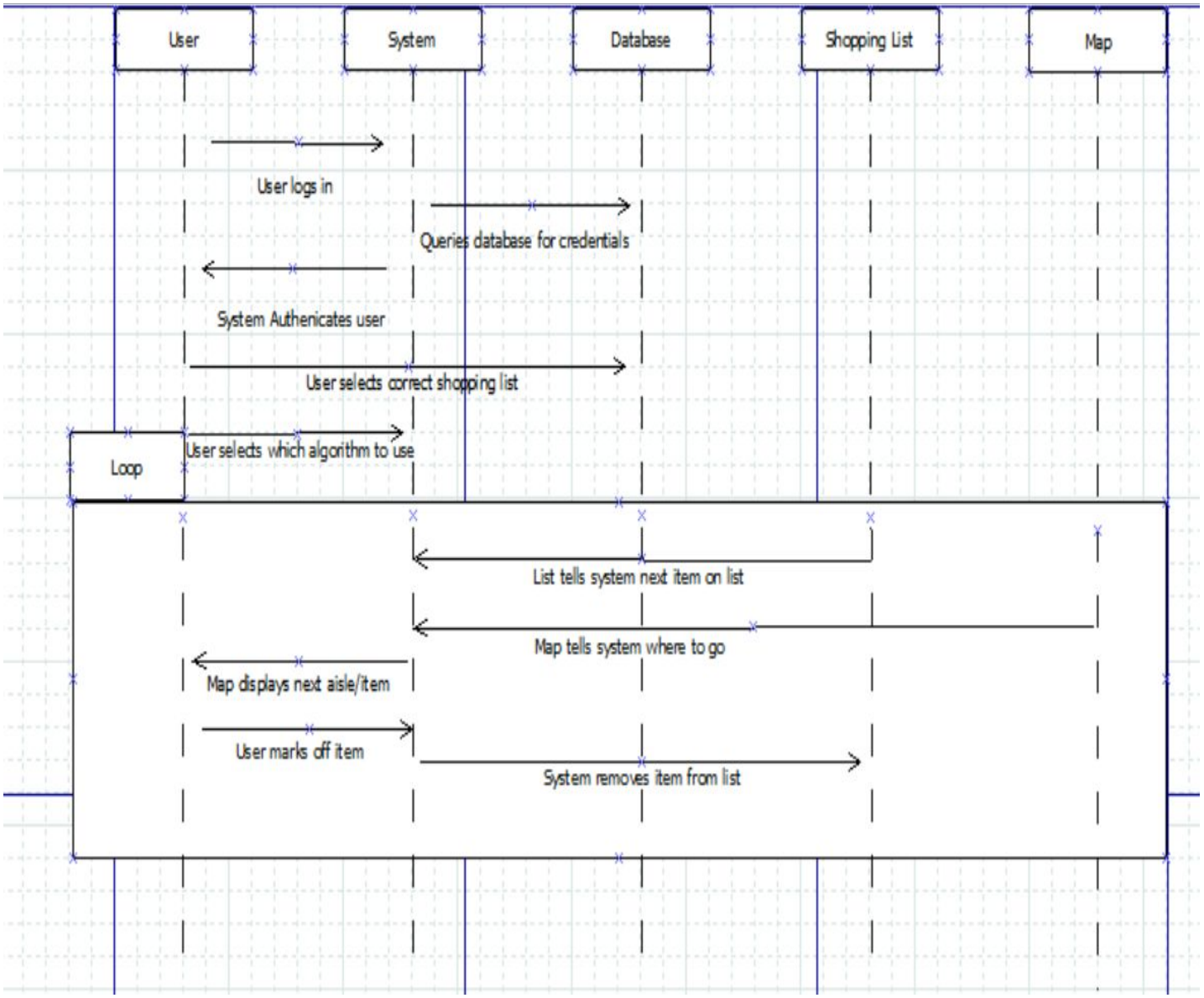
Message Sequence Diagram for Case 1 - Building a Grocery List



State Chart for Use Case 2 - Creating a store map



Use Case 3: Marking items off list



Contributions and Customer Interaction:

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)

We were able to meet with our customer, Kevin Davis, on the evening of Wednesday, October 16, to discuss the overall idea of the grocery shopping application and talk through our group's questions.