Capstone Project Document

Introduction

Purpose

- What is the problem or the opportunity that the project is investigating?

The Go Fridge web app aims to address the common problem of food wastage caused by users forgetting about the expiration dates of items in their fridges or purchasing unnecessary duplicates.

- Why is this problem valuable to address?

Have you ever bought something, stored it in your fridge and then forgotten about it until your next Big Clean Day only to find it expired? Or have you purchased an item only to later discover the same thing already in your fridge?
The cost of living is so high nowadays, Go Fridge seeks to help users save money by preventing the waste of purchased items. The app aids in managing fridge contents efficiently, avoiding expired items, and generating shopping lists.

- What is the current state (e.g. unsatisfied users, lost revenue)?

Wastage occurs due to forgetfulness or unintentional duplicate purchases.

- What is the desired state?

Go Fridge will assist users in managing their fridges effectively, reducing waste, and providing features like shopping list creation.

- Has this problem been addressed by other projects? What were the outcomes?

Similar projects exist but lack widespread recognition and user-friendliness.

Stakeholders

- Who are the stakeholders? (be as specific as possible as to who would have access to the software)

Anyone who has a fridge.

- Why do they care about this software?

Go Fridge helps users reduce living costs by avoiding food wastage and unnecessary purchases.

- What are the stakeholders' expectations?

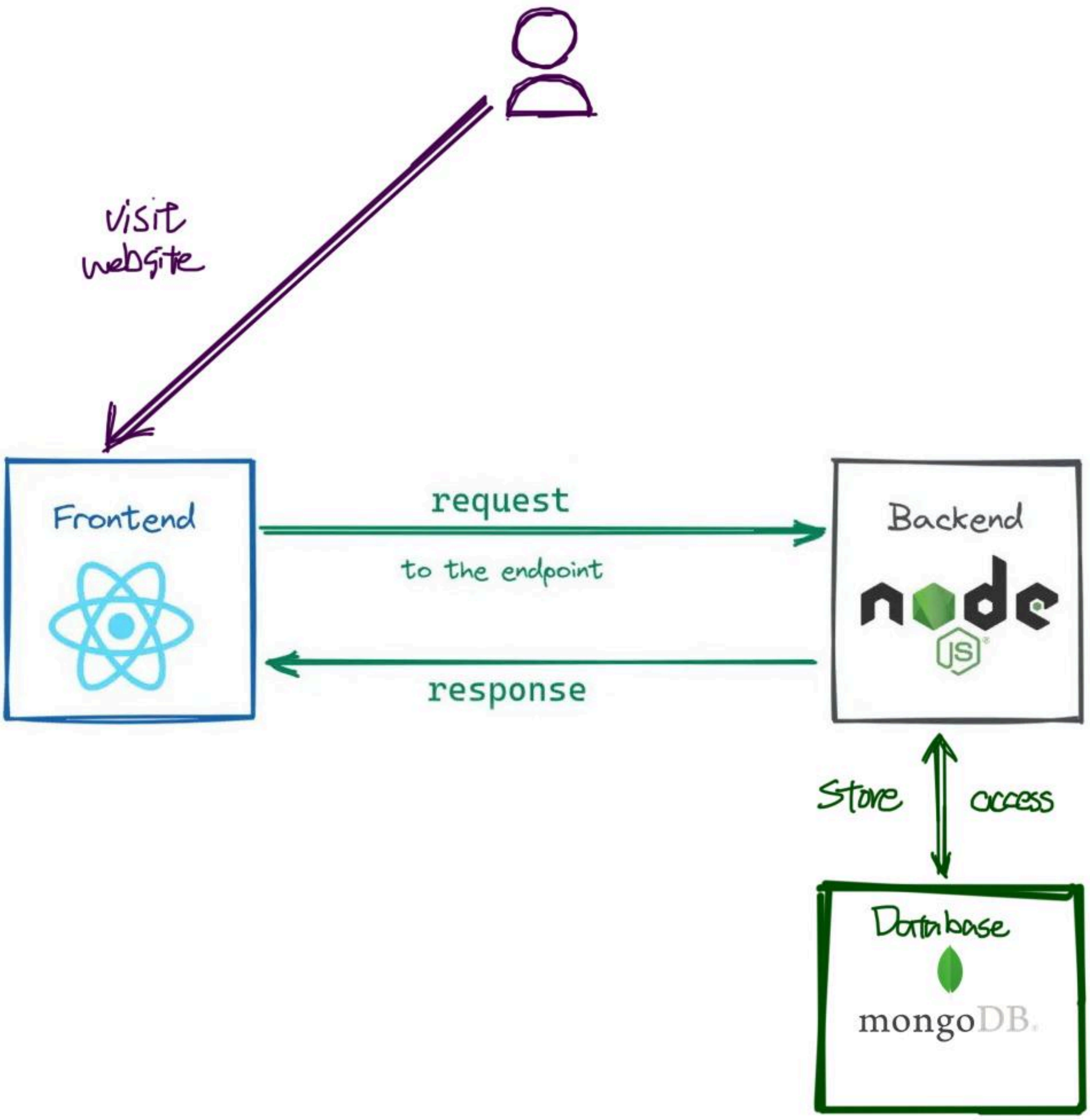
A succinct, clear and user friendly software, doesn't need to take too much time on learning how to use the app.

Product Description

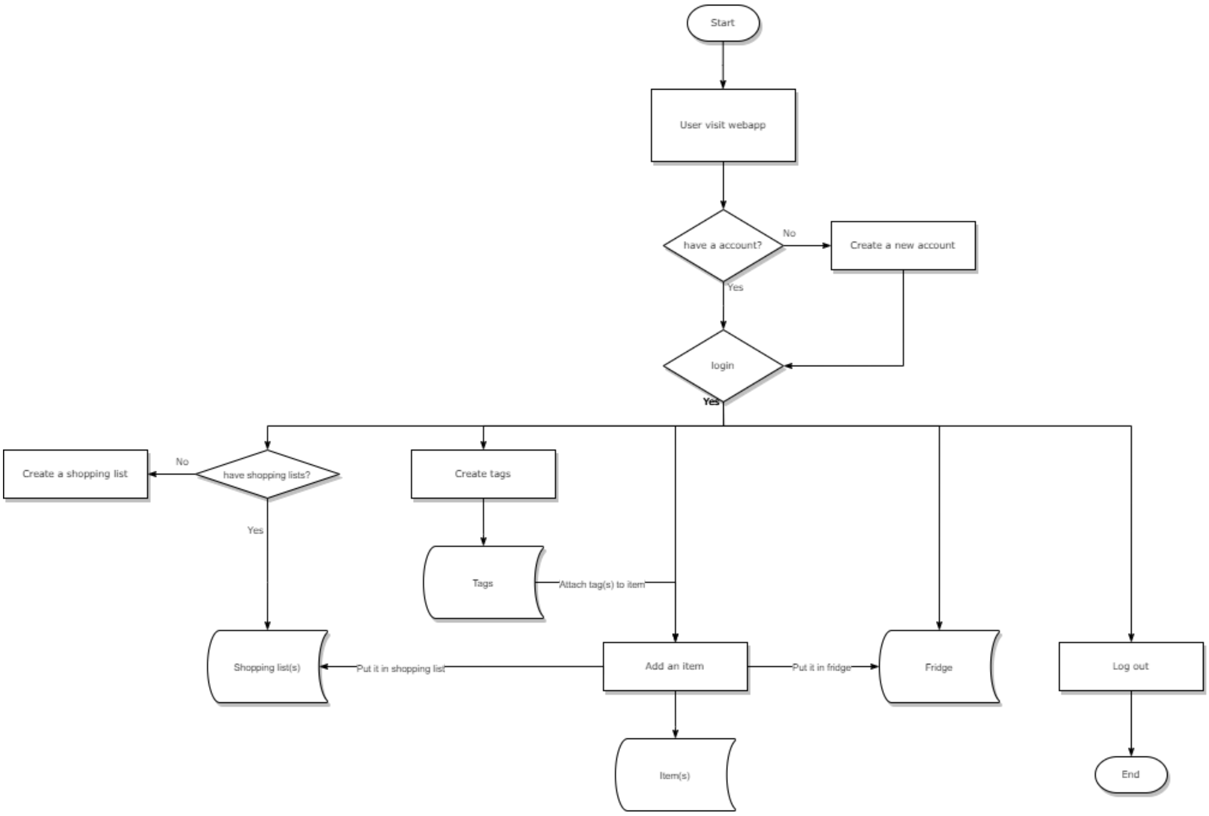
User Stories

User Story Title	User Story Description	Priority
User login	User login with registered account.	medium
User sign up	New users sign up for a new account that include email address, username and password.	medium
User logout	User logout, back to the signup/login page.	medium
Add new item	User adds a new item with quantity and tags, and also chooses to add it to their fridge or shopping list.	high
Update item's info	Users can update item's info to reflect the changes.	high
Add new tag	User adds a new tag in advance for later use.	medium
Search items	Users can search items by name in their fridge easily.	medium
Search tags	Users can search items by tags.	medium
Update shopping list	Users can update shopping list to reflect the changes.	medium
Responsiveness	User interface can adjust depending on what device that users are using while accessing the website.	high
Homepage	After login, users will redirect to homepage, shows fridge and shoppinglist to choose.	high
Best before date reminder	Highlight and alert the items that almost expire after user login/back to homepage.	medium

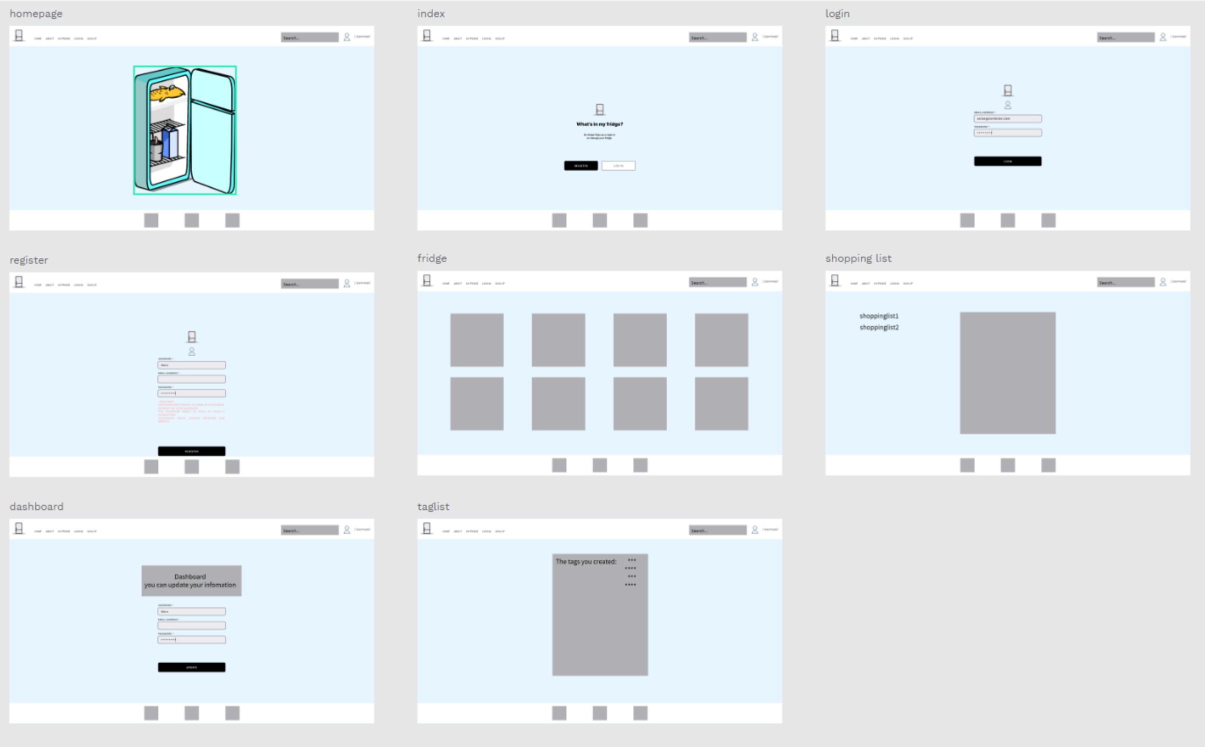
Architecture Design



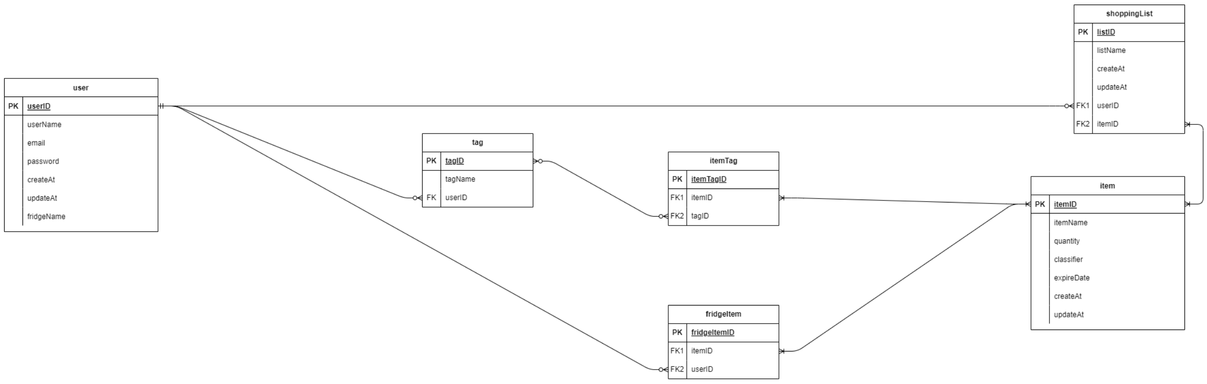
User Flow

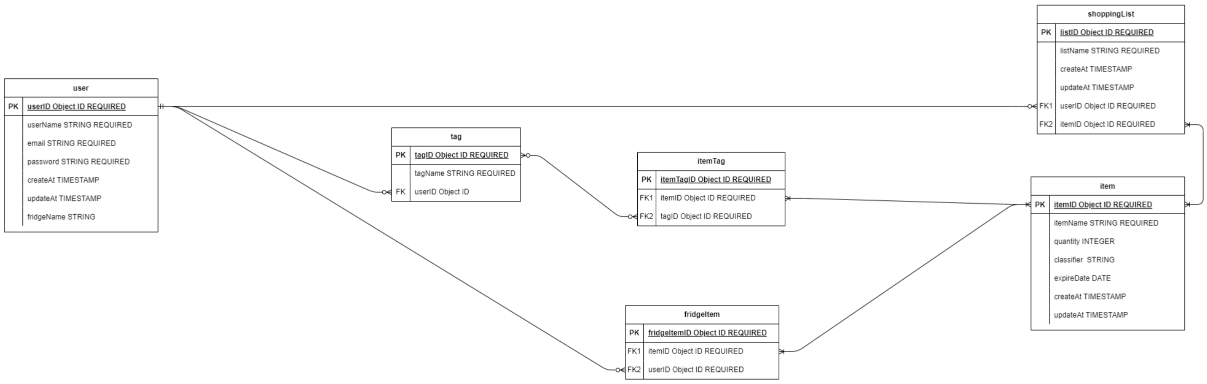

Start
User visit webapp
have a account?
No
Create a new account
Yes
login
Yes
Create a shopping list
No
have shopping lists?
Yes
Create tags
Tags
Attach tag(s) to item
Shopping list(s)
Put it in shopping list
Add an item
Put it in fridge
Fridge
Log out
Item(s)
End

Wireframe Design



Database Design





Open Questions/Out of Scope

- What features are considered out of scope?

1. Generate recipes based on items left in the fridge.
2. Input a recipe then compare the items in the fridge and generate a shopping list that includes the items that are needed but not in the fridge.
3. Host on AWS.

Non-functional Requirements

- What are the key security requirements? (e.g. login, storage of personal details, inactivity timeout, data encryption)

1. Password encryption.
2. Storage of user information.

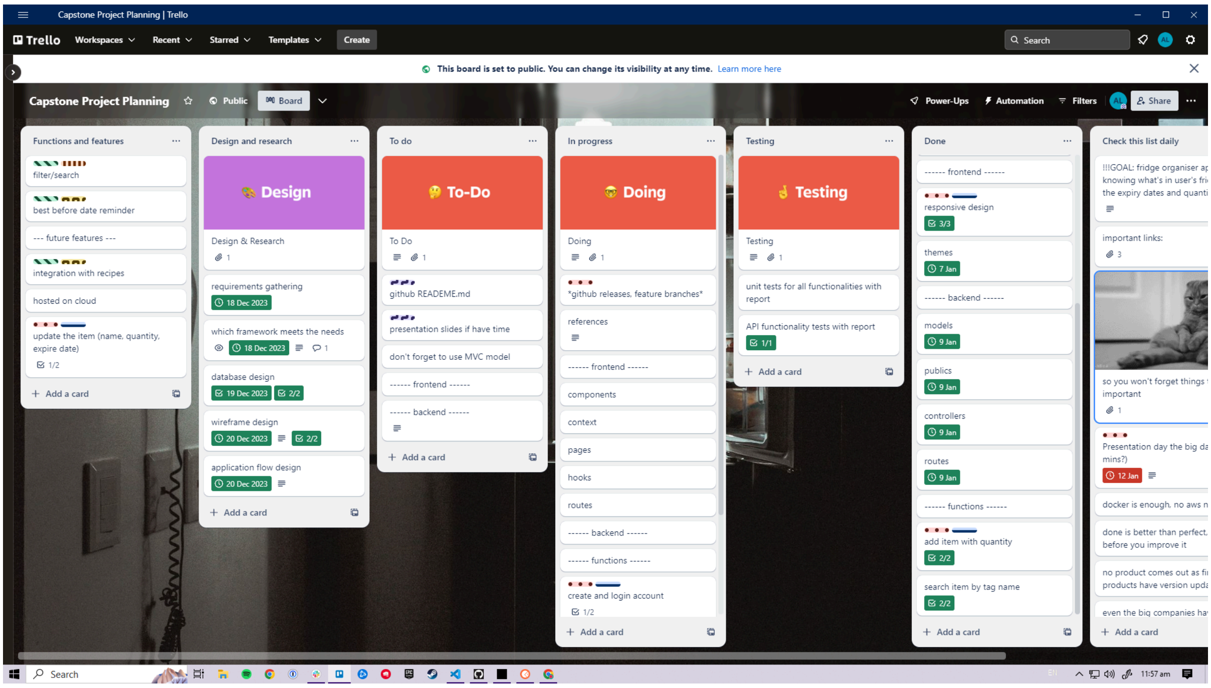
- How easy to use does the software need to be?

It should be simple enough that users know how to use it at first glance.

- Does the software conform to any technical standards to ease maintainability?

1. Code comments that help future me or developers to understand the code.
2. ReadME file that provides an overview of the project, instructions for setting up the development environment.
3. Frontend and backend separation in design and code.
4. Well-organised folder structure.
5. Clear and readable naming pattern for files, variables and functions.
6. Using GitHub for version control.

Project Planning



Capstone Project Planning | Trello
Trello
Workspaces
Recent
Starred
Templates
Create
Search
This board is set to public. You can change its visibility at any time. Learn more here
Capstone Project Planning
Public
Board
Power-Ups
Automation
Filters
Share
Add a card
11:57 am

Functions and features
- filter/search
- best before date reminder
- --- future features ---
- integration with recipes
- hosted on cloud
- update the item (name, quantity, expire date) 1/2

Design and research
- Design
- Design & Research 1
- requirements gathering 18 Dec 2023
- which framework meets the needs 18 Dec 2023 1
- database design 19 Dec 2023 2/2
- wireframe design 20 Dec 2023 2/2
- application flow design 20 Dec 2023

To do
- To-Do
- To Do 1
- github READEME.md
- presentation slides if have time
- don't forget to use MVC model
- ------ frontend ------
- ------ backend ------

In progress
- Doing
- Doing 1
- *github releases, feature branches*
- references
- ------ frontend ------
- components
- context
- pages
- hooks
- routes
- ------ backend ------
- ------ functions ------
- create and login account 1/2

Testing
- Testing
- Testing 1
- unit tests for all functionalities with report
- API functionality tests with report 1/1

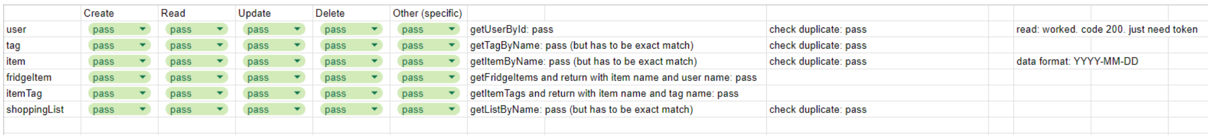
Done
- ------ frontend ------
- responsive design 3/3
- themes 7 Jan
- ------ backend ------
- models 9 Jan
- publics 9 Jan
- controllers 9 Jan
- routes 9 Jan
- ------ functions ------
- add item with quantity 2/2
- search item by tag name 2/2

Check this list daily
- !!!GOAL: fridge organiser ap knowing what's in user's fri the expiry dates and quanti
- important links: 3
- so you won't forget things important 1
- Presentation day the big da mins?) 12 Jan
- docker is enough, no aws n
- done is better than perfect, before you improve it
- no product comes out as fi products have version upda
- even the big companies ha

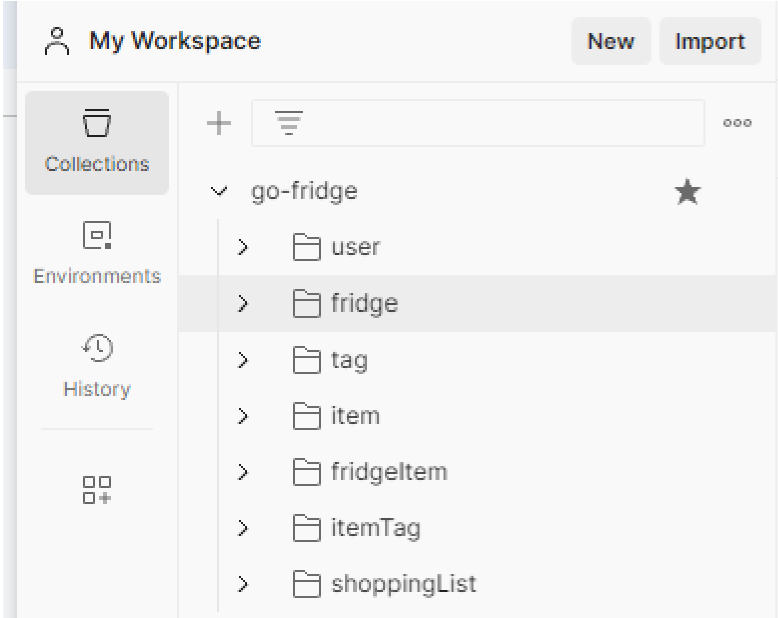
Trello was used to planning the project and tracking the process.

Testing Strategy

- Postman was used to test api CRUD operations. Status of each test was recorded.



	Create	Read	Update	Delete	Other (specific)			
user	pass	pass	pass	pass	pass	getUserById: pass	check duplicate: pass	read: worked. code 200. just need token
tag	pass	pass	pass	pass	pass	getTagByName: pass (but has to be exact match)	check duplicate: pass	
item	pass	pass	pass	pass	pass	getItemByName: pass (but has to be exact match)	check duplicate: pass	data format: YYYY-MM-DD
fridgeItem	pass	pass	pass	pass	pass	getFridgeItems and return with item name and user name: pass		
itemTag	pass	pass	pass	pass	pass	getItemTags and return with item name and tag name: pass		
shoppingList	pass	pass	pass	pass	pass	getListByName: pass (but has to be exact match)	check duplicate: pass	



My Workspace

New

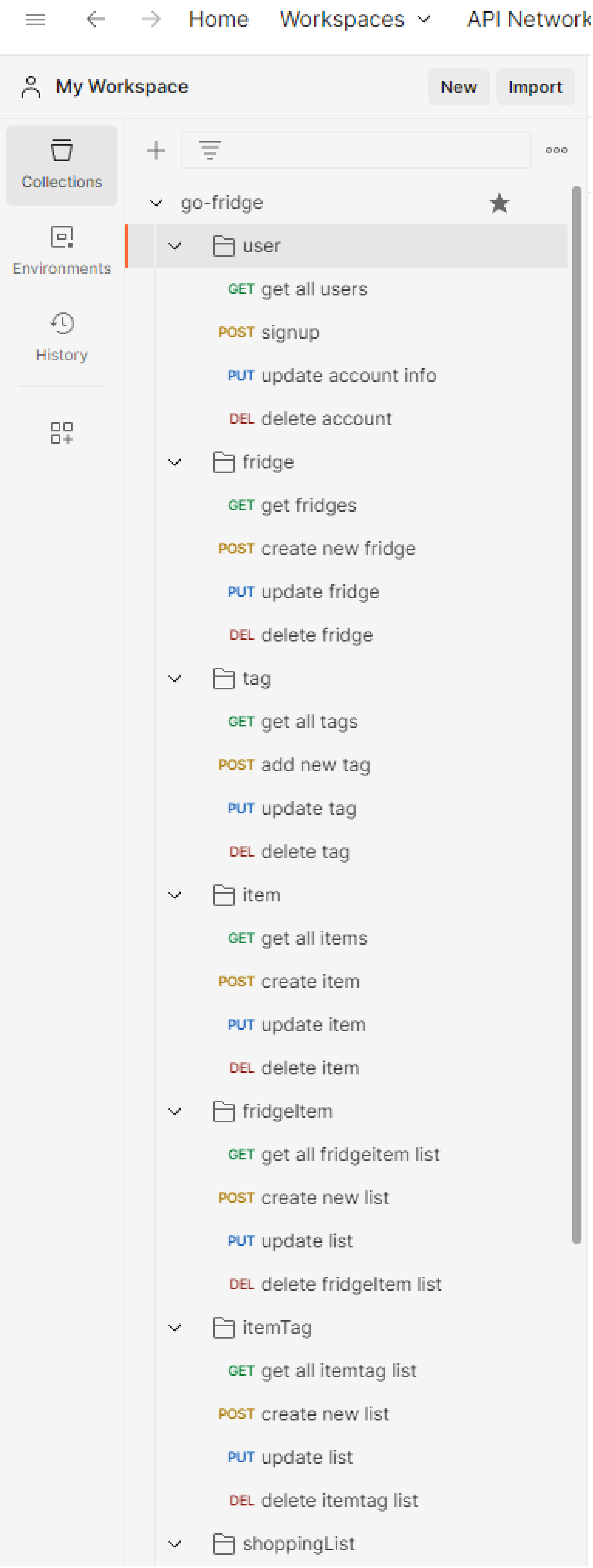
Import

Collections

Environments

History

- go-fridge
  - user
  - fridge
  - tag
  - item
  - fridgeItem
  - itemTag
  - shoppingList



go-fridge

user
- GET get all users
- POST signup
- PUT update account info
- DEL delete account

fridge
- GET get fridges
- POST create new fridge
- PUT update fridge
- DEL delete fridge

tag
- GET get all tags
- POST add new tag
- PUT update tag
- DEL delete tag

item
- GET get all items
- POST create item
- PUT update item
- DEL delete item

fridgeItem
- GET get all fridgeitem list
- POST create new list
- PUT update list
- DEL delete fridgeItem list

itemTag
- GET get all itemtag list
- POST create new list
- PUT update list
- DEL delete itemtag list

shoppingList

- Compare frontend with flowchart to make sure the logic is on track.

Edge cases handle:

1. Multiple console.logs were used to ensure expected results and catch odd results.
2. Inspect tool in browser was used to check consoles and elements.

Implementation

- What were the considerations for deploying the software?

Ideally, it would be hosted on the cloud so users can access the web app from anywhere.

End-to-end solution

- How well did the software meet its objectives?

All core functions are achieved in backend. However, most of them are not achieved in frontend.

General testing was done.

Originally planned to have it hosted on AWS but due to time and resource constraints, the goals could not be achieved. Added it to the future feature.

What I have learnt

- Be patient with the code, sometimes tiny problem will cause the big trouble :0
- It would be better to have a framework first and then fill in the features. Begin with the simplest functions and web pages so even if you can’t do all the features, you will at least have something to demonstrate.
- Don't be afraid to make mistakes, mistakes help you to learn. Nothing is perfect from the beginning.
- Writing comprehensive documentation and maintaining clean, well-commented code is important. It underscored how important it is to make software that not only works well but is also easy to maintain and understand for future developers. (at least it will help future you to understand what were you thinking when coding)

References

- Where is the code used in the project? (link to GitHub)

Code:

Go Fridge

- What are the resources used in the project? (libraries, APIs, databases, tools, etc)

Database:

MongoDB

Design Tools:

draw.io

- database design: database
- flowchart design: flowchart

Penpot

- wireframe design: Penpot

packages/libraries:

Free, Open API for Detecting Disposable Email Addresses (However, timeout during testing stage and can't be fixed. Code is comment out.)

jsonwebtoken - npm

bcryptjs - npm

cors - npm

dotenv - npm

express - npm

mongoose - npm

react - npm

react-dom - npm

react-router-dom - npm

axios - npm

@mui/material - npm

@mui/icons-material - npm

@emotion/styled - npm

@emotion/react - npm

vite - npm

project planning:

Go Fridge project kanban