

Chipln



Team 2

Kayoon Koh,
Chelsea Lee,
Joylyn Pan,
Lilian Zhao

Index

Index.....	1
Purpose.....	2
Functional Issues.....	2
Non-Functional Issues.....	4
Response Time.....	4
Scalability.....	5
Usability.....	5
Security.....	5
Design Outline.....	6
High Level Overview.....	6
Sequence of Events Overview.....	7
Design Issues.....	9
Functional Issues:.....	9
Nonfunctional Issues:.....	11
Design Details.....	13
Database Design.....	13
Class Diagram.....	13
Schema.....	14
Sequence for Creating ChipIn Account.....	17
Sequence for Joining a Household Group.....	18
Sequence for Manually Adding Item to Grocery List.....	19
Sequence for Moving Grocery Item to Purchased List & Updating Expenses.....	20
Sequence for Generating AI Recipes.....	21
Navigation Flow Chart.....	22
UI Mockups.....	23
Sign up/ Log in process.....	23
Dashboard.....	24
Household Info Page.....	25
My Expenses Page.....	26
Recipes Page.....	27
User Profile/Settings.....	28

Purpose

Running a household with several roommates is difficult; one major challenge is maintaining an accurate inventory of shared items. Whether it's personal or shared groceries, cleaning supplies, or pet items, keeping track of what products need to be bought, replaced, or discarded prevents unnecessary purchases and reduces trips to the store. Splitting costs on top of bookkeeping for many housemates presents even more of a struggle. Unlike traditional financial management tools like Splitwise and shared grocery list apps like AnyList, ChipIn not only tracks expenses, but also offers useful insights into the actual shared products within households, streamlining both cost-sharing and inventory management.

Functional Issues

1. As a user, I would like to be able to register for a ChipIn account.
2. As a user, I would like to be able to log in and manage my ChipIn account.
3. As a user, I would like to be able to log in via Gmail and connect associated accounts.
4. As a user, I would like to be able to change/reset my password.
5. As a user, I would like to be able to create a user profile.
6. As a user, I would like to be able to change my name, username, and profile picture on my profile.
7. As a user, I would like to be able to delete or log out of my ChipIn account.
8. As a user, I would like to be able to choose to receive notifications for low inventory and nearing expiration dates of household items.
9. As a user, I would like to be able to modify default user settings involving display, when to give payment reminders, and notifications.
10. As a user, I would like to be able to create and/or join a household.
11. As a user, I would like to be able to invite people to my household in-app, via an invite link, or email.
12. As a user, I would like to be able to leave a household I'm no longer part of.
13. As a user, I would like to be able to view my grocery list and list of purchased items.
14. As a user, I would like to be able to view cards with an item's name, price, shared/unshared, and expiration date.
15. As a user, I would like to be able to select the category of items (e.g., groceries, cleaning supplies, pet items) when adding them to the grocery list / purchased list.
16. As a user, I would like to be able to search for a product in both the purchased items list and the grocery list.
17. As a user, I would like to be able to sort purchased items by their expiration dates.
18. As a user, I would like to be able to filter purchased items based on the housemates they are shared with and by category (e.g., groceries, cleaning supplies, pet items).

19. As a user, I would like to be able to have an AI automatically categorize the items in my grocery list.
20. As a user, I would like to be able to view what items are about to expire or are expired.
21. As a user, I would like to be able to add items to my grocery list.
22. As a user, I would like to assign myself as the purchaser for a grocery list.
23. As a user, I would like to be able to select which roommates will share an item with me.
24. As a user, I would like to be able to click a 'purchased' button to move an item from the grocery list to the household's purchased items list, as well as a 'repurchase' button which will add the item back to the grocery list.
25. As a user, I would like to be able to add purchased items directly to the shared inventory display.
26. As a user, I would like to be able to delete purchased items directly in the shared inventory display as items will not be automatically removed.
27. As a user, I would like to be able to scan a receipt that will automatically populate additional item cards and add them to the purchased items list.
28. As a user, I would like to be able to edit any item card in the purchased items list in the case that the scanned receipt yields incorrect items.
29. As a user, I would like to be able to remove an item from the purchased list if I choose to return a product and recalculate owed amounts accordingly.
30. As a user, I would like to be able to expand a card to view more detailed payment/item information.
31. As a user, I would like to be able to do uneven splits for specific items.
32. As a user, I would like to be able to view a separate dashboard of how much money I owe each member of my household.
33. As a user, I would like to be able to have the owed amounts updated automatically with every purchase.
34. As a user, I would like to be able to see how much each person sharing an item contributes to the price.
35. As a user, I would like to be able to click a 'paid all' button that will clear the amount owed to one or more roommates, a 'paid custom amount' button that will allow me to pay back part of my debt, and a 'paid' button that will log that I paid someone back for a specific item.
36. As a user, I would like to be able to use the item function for expenses like water/electricity bills.
37. As a user, I would like to be able to select ingredients and items I currently have for the AI recipe suggestion tool.
38. As a user, I would like to be able to add additional ingredients and items to the AI suggestion tool.

39. As a user, I would like to be able to view different meal ideas suggested by the AI.
40. As a user, I would like to be able to click on the specific meal idea and have the recipe expanded in a pop-out card.
41. As a user, I would like to be able to add a meal idea and its respective recipe to a 'saved meals' page.
42. As a user, I would like to be able to enter a specific item into the AI deal finder tool.
43. As a user, I would like to be able to view the best deals for that item.
44. As a user, I would like to be able to view the store name, proximity, and more product information.
45. As a user, I would like to be able to export expiration dates as a csv file to upload to calendar apps that accept imports e.g. Google/Outlook.
46. As a user, I would like to be able to view statistics on frequently purchased items and prices.
47. As a user, I would like to be able to use ChipIn on a mobile browser.
48. As a user, I would like to be able to send nudges to my roommates when I want them to pay me back.
49. As a user, I would like to be able to add notes to a shared corkboard for special requests on grocery trips, allergy reminders, etc.
50. As a user, I would like to be able to view a leaderboard displaying who pays everyone back the fastest. (if time allows)
51. As a user, I would like to be able to message my roommates through the app. (if time allows)
52. As a user, I would like to be able to customize the theme of my dashboard. (if time allows)
53. As a user, I would like to be able to receive badges if my household doesn't let items expire. (if time allows)

Non-Functional Issues:

Response Time

Since ChipIn will be web app based, the response time will be under 600 ms for all features. Logging into the web app and navigating the households to get to the dashboard should take no more than 200 ms. In addition, viewing the shared inventory and total balance should also take 200 ms. There could be some minimal lag when it comes to the ChipIn recalculating the total balance and recalculating the individual prices that each person owes to other roommates. The process of viewing each item card should also be smooth and if there are too many items in the purchase list, there also might be minimal lag when it comes to scrolling through that list.

Scalability

ChipIn should be able to support a baseline of around 100 users simultaneously without performance issues. This ensures that for small to medium sized households, our platform can handle activities like updating multiple users' grocery lists, sending reminders about shared bills, or splitting shared expenses without any noticeable slowdowns. Our goal is for ChipIn to eventually be able to serve thousands of users. We plan to build the app's frontend and backend separately to avoid compatibility issues and ensure ease of development. Additionally, using MongoDB to manage our database will allow us to easily scale our database both vertically and horizontally.

Usability

ChipIn's dashboard will be designed so users can easily view important information and statistics in one centralized space, such as the amount owed to each housemate and the inventory/status of shared/personal household items. The web app will be compatible with all browsers and screen sizes.

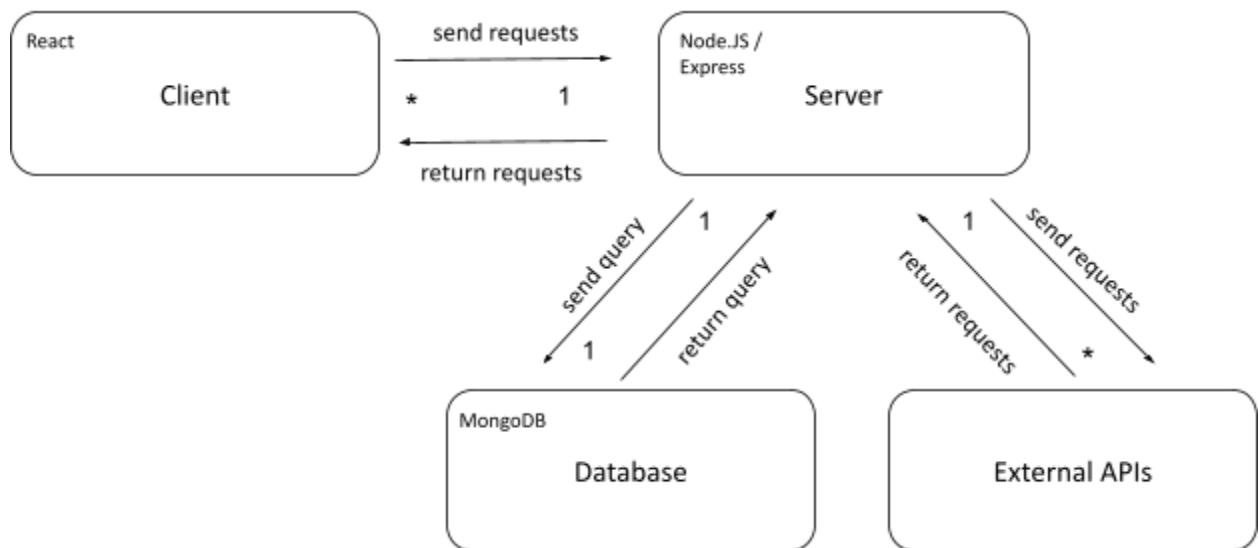
Security

To ensure users can safely upkeep their grocery lists, ChipIn will use password hashing and Json Web Tokens for user authentication. This will not only maintain the stateless nature of the app, but also reduce the number of database queries with information stored in the tokens themselves. Additionally, sanitizing user input with mongo-sanitize will help prevent database injection so each household can only view their own details.

Design Outline

High Level Overview

Chipin will be a client-server web application that has four major components: the client, the server, the database, and external APIs. On the client side, users will be allowed to interact with the user interface and will be able to make requests that will be sent to the server. For example, users will be able to select different items and add it to their grocery list. The server will update these changes and other client requests accordingly in the database. The server will also accept certain client requests and will send those requests to external APIs. The API will process those requests and further send relevant data back.



1. Client
 - a. The client will provide a user interface for users to organize grocery items and facilitate costs.
 - b. The client sends requests to the server.
 - c. The client will receive back requests from the server and will update the user interface with these results.
2. Server
 - a. The server will receive and handle requests from the client.
 - b. The server will establish a connection with the database and will construct a query based on the client's request.
 - c. The server will send queries to the database.
 - d. The server will also create an API endpoint and send a request through HTTP?

3. Database

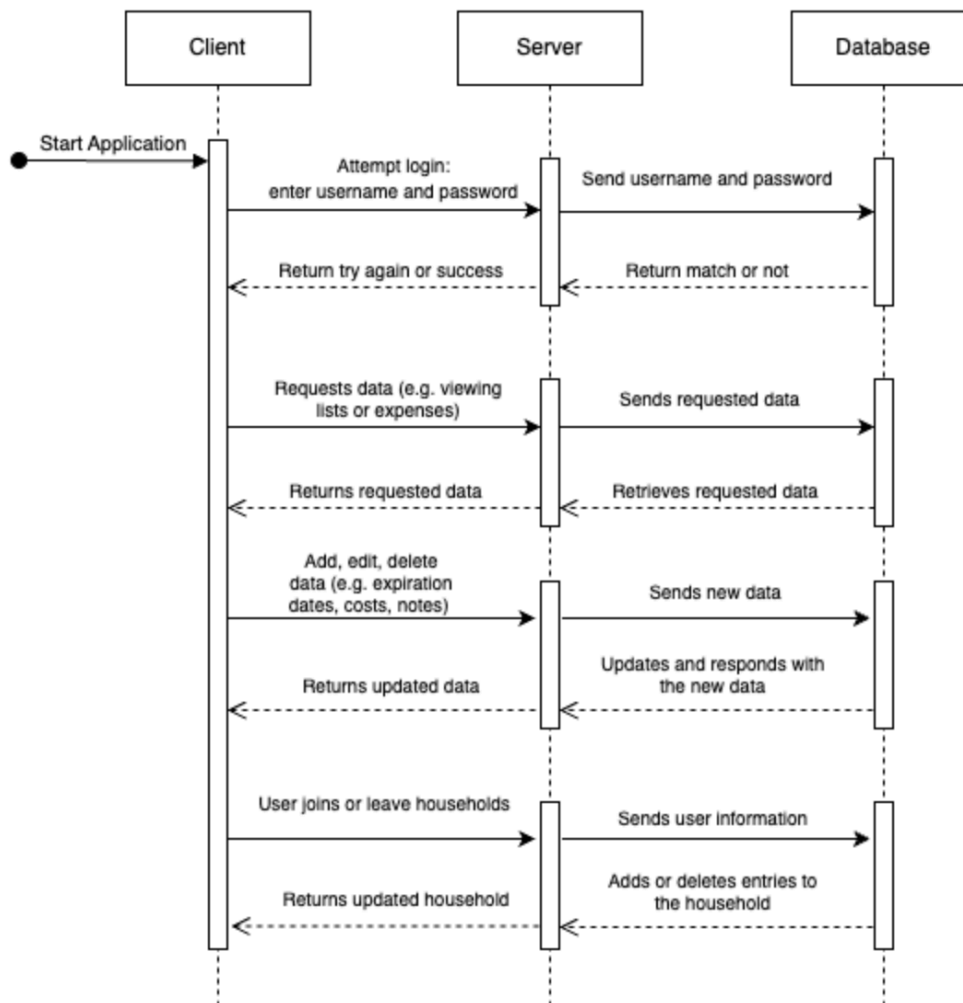
- a. A MongoDB NoSQL database will store all collections including a household collection, items collection, notes collection, and users collection.
- b. Within these collections will store more specific documents pertaining to that category.
- c. This database will process and execute queries sent by the server and update any records.

4. External APIs

- a. An OCR API will be used when users want to upload and scan their receipts into the grocery list.
- b. A GPT API will be used when users want to receive recipe ideas based on the existing ingredients they have.
- c. The APIs will receive requests from the server and will process and return results back to the server.

Sequence of Events Overview

The sequence of events below illustrate the general flow between the client, server, and database. The user will first be presented with the login page and will login using a unique username and password. When the user logs in, the client will send a request to the server which will then send that request as a query to the database. The database will search for the username and password and will return a result to the client. After logging in, the user will then be brought to all of the households that they reside in and once they choose their household, the user will be brought to the dashboard page. Once again, the client will send all of these requests to the server to process. The server will send back a response for the client to update the user interfaces. The client will allow users to perform other actions like adding items to the grocery list, adding items to the purchased list, and viewing owed expenses to other roommates. The server will send queries to the database which will update or retrieve results from it.



Design Issues

Functional Issues:

What information is necessary for a user to create an account?

- Option 1: Name, Username, Password
- Option 2: Name, Username, Password, Email
- Option 3: Name, Username, Password, Email, Phone Number

Solution: Option 2

The second option is ideal for both simplicity and security. Opting for creating a username and password will allow for quick and easy login and including a name will make each user's account more personalized. In addition, users will be able to link their personal email to the account for verification reasons and also receive ChipIn notifications through their email. Including a phone number could complicate the sign-up process unnecessarily, adding potential issues with SMS verification or messaging without significant benefits for the core functionality.

How can users add grocery items to their purchased list?

- Option 1: User manually logs items purchased
- Option 2: User can click a 'purchased' button to move an item from the grocery list to the household's purchased items list
- Option 3: User can scan receipts through Optical Character Recognition (OCR) technology

Solution: Options 1, 2, 3

Users should be able to move grocery items directly into their purchased list, bypassing the need to manually log repetitive items. For longer lists, the OCR-powered receipt scanning feature automates the process by extracting item names and details, making it faster and more convenient. Additionally, if users need to modify any item details—such as price, quantity, or who contributed—they can manually adjust the information. Offering all three options ensures flexibility, efficiency, and accuracy in tracking purchased items, and adapting to different user preferences and needs.

How does ChipIn compute splits between roommates?

- Option 1: Even split between all roommates for all purchased items
- Option 2: Allow users to customize split percentages for specific items

Solution: Options 1, 2

Having both features available gives roommates flexibility in situations where shared items may not be used equally, while also accommodating situations where an even split is preferred for common items. For example, during a grocery trip, there may be a mix of personal items, which only one roommate should pay for, and shared items, which are split among roommates. By offering both options, ChipIn ensures that costs are distributed fairly and transparently in any scenario, accommodating different living arrangements and purchasing habits.

How do we display the item lists?

- Option 1: Vertical list displaying item information
- Option 2: Table with rows containing item information
- Option 3: Individual cards displaying item information

Solution: Option 3

Using individual cards to display item information provides a cleaner, more intuitive visual layout, making it easier for users to quickly browse and understand details at a glance. Each card can highlight key information, such as the item name, price, shared status, and expiration date, with clear visual separation between items. This approach is also more responsive for desktop views, allowing for better flexibility when adjusting to different screen sizes. Cards can easily accommodate interactive elements like buttons for editing or marking items as purchased, making the interface more user-friendly and visually engaging.

Nonfunctional Issues:

What frontend framework should we use?

- Option 1: React
- Option 2: Angular
- Option 3: Vanilla Javascript + HTML/CSS

Solution: Option 1

React is widely used, well-documented, and has a large community, making it a strong choice for building modern, dynamic, and responsive user interfaces. Its component-based architecture promotes reusability, which will streamline the development of features like grocery lists, expense tracking, and custom input forms for the ChipIn app. Additionally, React's support for state management and seamless integration with various libraries makes it flexible for scaling and future enhancements.

What backend framework should we use?

- Option 1: Spring Boot
- Option 2: Node.JS
- Option 3: Flask
- Option 4: PHP

Solution: Option 2

Node.JS is fast, lightweight, and scalable, making it a good fit for ChipIn's real-time and event-driven features. Its ability to handle multiple connections simultaneously is ideal for a household app where several users will interact with shared lists and data. Additionally, using JavaScript for both the frontend (React) and backend (Node.JS) ensures consistency in the development stack, reducing complexity and allowing for smoother team collaboration.

What database service should we use?

- Option 1: MySQL

- Option 2: PostgreSQL
- Option 3: MongoDB
- Option 4: Firebase

Solution: Option 3

MongoDB's NoSQL, document-based structure offers flexibility for storing varied data such as grocery items, user preferences, and expense tracking, without the need for a rigid schema. It allows fast read and write operations, essential for real-time updates in ChipIn, like grocery list changes and expense tracking. MongoDB scales easily with sharding, ensuring performance as the app grows. Security is robust with features like authentication, encryption, and role-based access control, protecting sensitive data. Its compatibility with Node.JS makes it a good fit for the backend, enabling secure, efficient communication.

What tool should we use for receipt scanning?

- Option 1: OCR (Optical Character Recognition)
- Option 2: API-based receipt scanning (Google Vision, Microsoft Cognitive Services, etc.)
- Option 3: Manual upload with template parsing

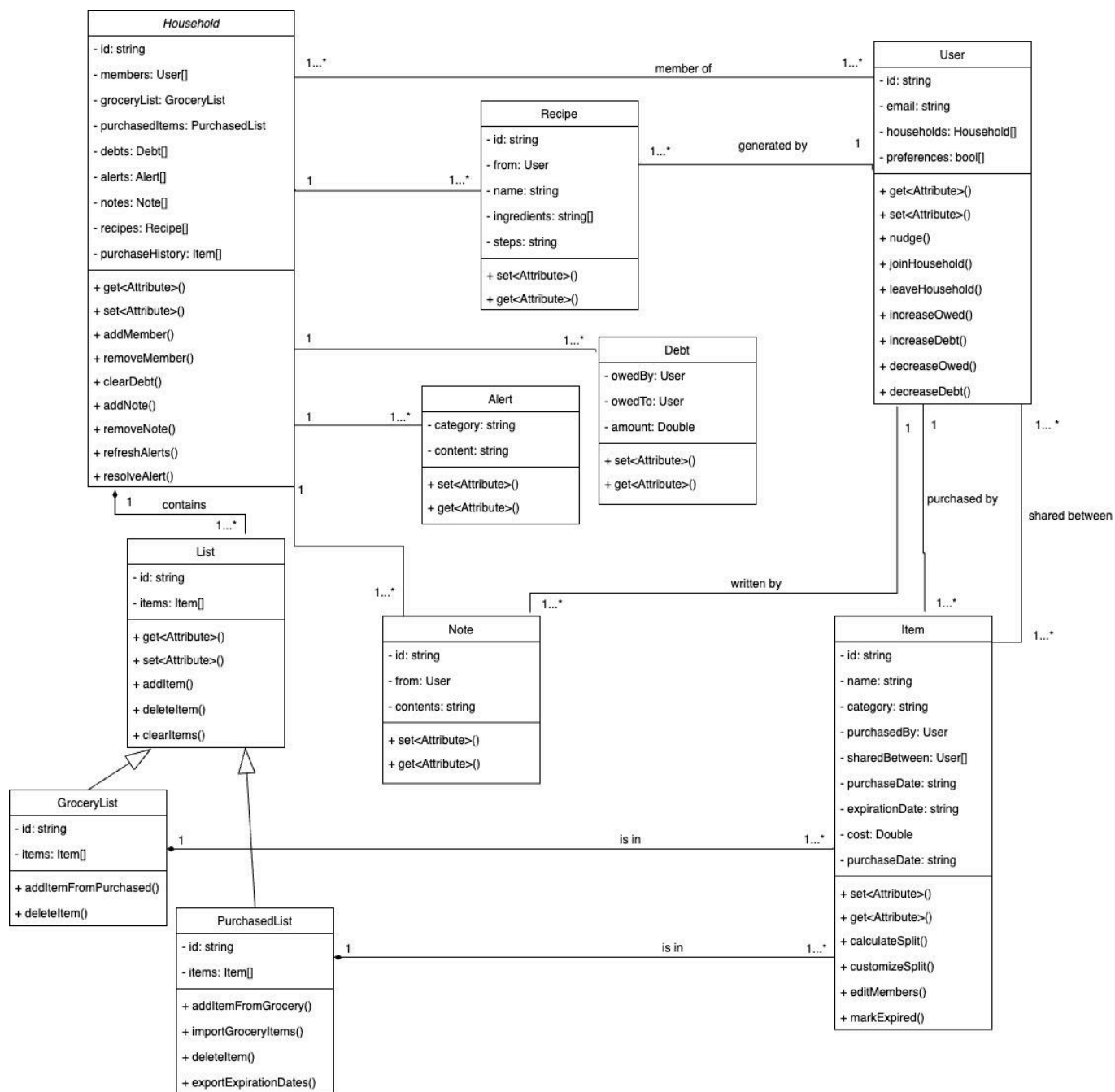
Solution: Option 1

We chose to use OCR (Optical Character Recognition) for ChipIn's receipt scanning feature due to its simplicity and flexibility. We plan to use the open-source OCR tool EasyOCR, which can be easily integrated into the app and provides support for multiple languages, ensuring broad compatibility with different receipt formats. By automating the text extraction process, users will save time and effort in logging purchases, while ensuring accuracy in tracking grocery items.

Design Details

Database Design

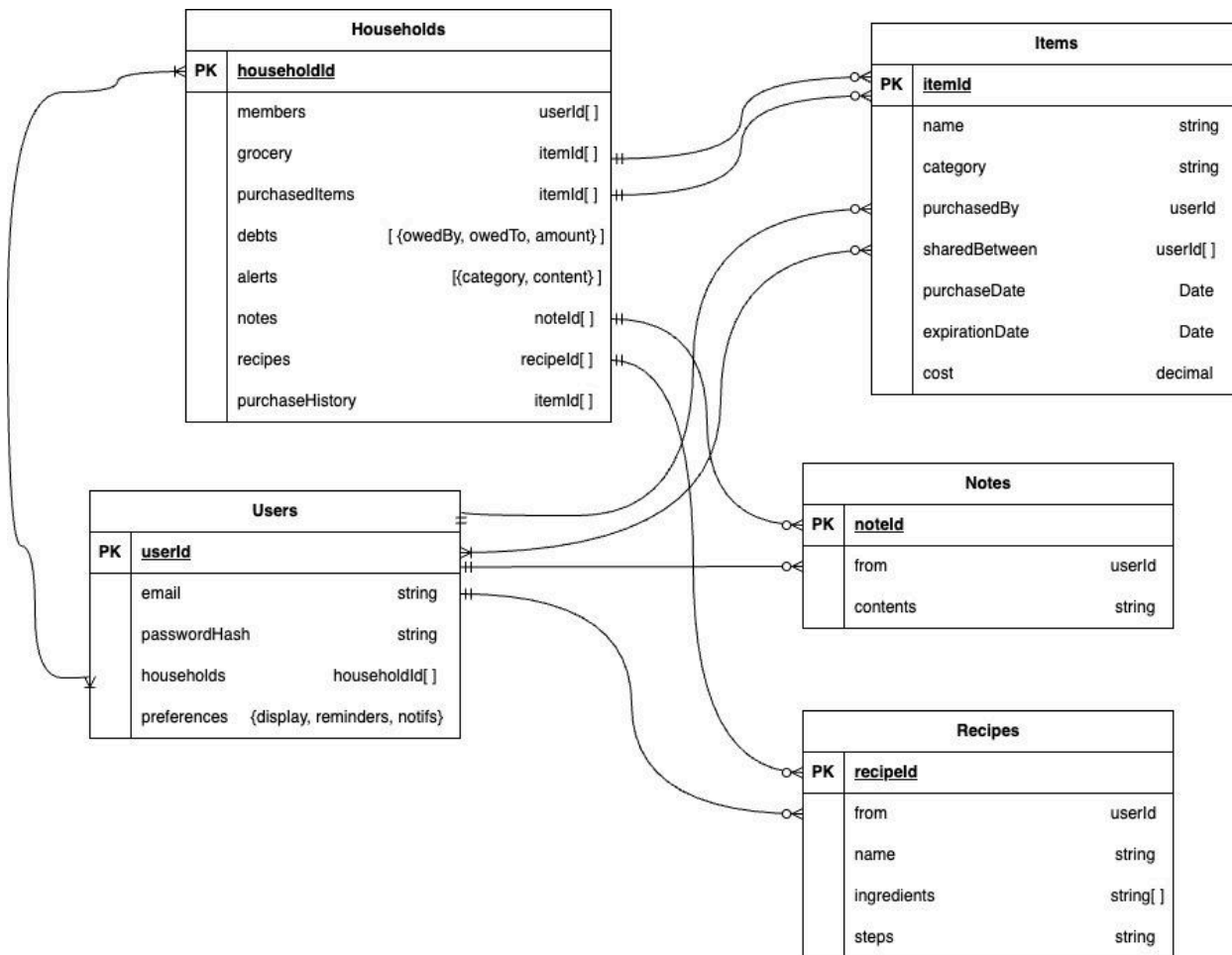
Class Diagram:



- **User**
 - Users are created when someone creates an account
 - Usernames are unique and used as user ID's
 - Users will login via username/email and password
 - Each user will be able to join one or more households
 - Users can set their site preferences
- **Household**
 - Households are created by users
 - Contains
 - list of Users that live together
 - a GroceryList
 - a PurchasedList
 - list of Debts that keep track of the amount each User in a household owes another
 - list of Alerts displayed on dashboard
 - a shared notepad where Users can make notes/reminders
 - saved recipes AI generated recipes
 - a list of archived items for stats use
- **List**
 - Generic list class with functions for editing list
 - **GroceryList**
 - List for users to add items to buy; items will only have name/category/shared fields at this point
 - Additional function to add an item from purchased list back to grocery list—creates copy of purchased item with blank fields to represent new distinct item
 - Items deleted from grocery list are removed from database
 - **PurchasedList**
 - List for items after they are purchased; remaining item fields are filled
 - Items can be added directly or moved from grocery list
 - Adding an item to the purchased list automatically splits the cost of the item between the roommates that share it i.e. updates the amounts owed between the user that purchased it and their roommates
 - Items deleted from purchased list are no longer displayed but not deleted from the database for statistics purposes
- **Item**
 - Items are created by Users and exist in either a household's grocery or purchased list

- Uniquely identified by ID
- Grouped by categories: Food, Cleaning Supplies, Pet Supplies, Medication, Kitchen Supplies, Miscellaneous
- Users can select which roommates an item is shared between
- The purchase date is recorded (the day an item is added to the purchased list by default, or a custom date)
- An item will generate an alert when an expiration date is nearing
- **Note**
 - Notes are created when a user adds a note to the shared notepad
 - Author of a note is displayed
- **Alert**
 - Alerts are automatically created when an item is nearing expiration or if a user's amount owed goes above a number of their choosing
 - Users can 'nudge' a roommate to pay them back, which creates an alert that will display on the recipient's dashboard
- **Debt**
 - Debt objects are automatically created when a user joins a household
 - $2 \times n$ debts for n roommates to record owed amounts for every relationship
 - Users can record paying someone back in full, a specific amount, or for an item
- **Recipe**
 - Recipes are created when a user uses the AI recipe generator
 - Recipes are comprised of a dish name, ingredients, and instructions
 - A user can choose to save a recipe, which is then visible to everyone in the household

Schema:



Users

```

{
  "id": "string/username",
  "email": "string",
  "passwordHash": "string",
  "households": ["household1 id", "household2 id" ...],
  "preferences": {
    "display": "string",
    "reminders": "string",
    "notif": "boolean"
  }
}

```

Households

```

{

```

```

    "id": "primary key",
    "members": ["user1 id", "user2 id" ...],
    "groceryList": ["item1 id", "item2 id" ...],
    "purchasedItems": ["item3 id", "item4 id" ...],
    "debts": [
      {
        "owedBy": "user1 id",
        "owedTo": "user2 id",
        "amount": "Decimal128"
      }
    ],
    "alerts": [
      {"category": "string"}
    ],
    "notes": ["note1 id", "note2 id" ...],
    "recipes": ["recipe1 id", "recipe2 id" ...]
    "purchaseHistory": ["item3 id", "item4 id" ...]
  }

```

Items

```

{
  "id": "primary key",
  "name": "string",
  "category": "string",
  "purchasedBy": "user id",
  "sharedBetween": ["user1 id", "user2 id" ...],
  "expirationDate": "Date",
  "purchaseDate": "Date",
  "cost": "Decimal128"
}

```

Notes

```

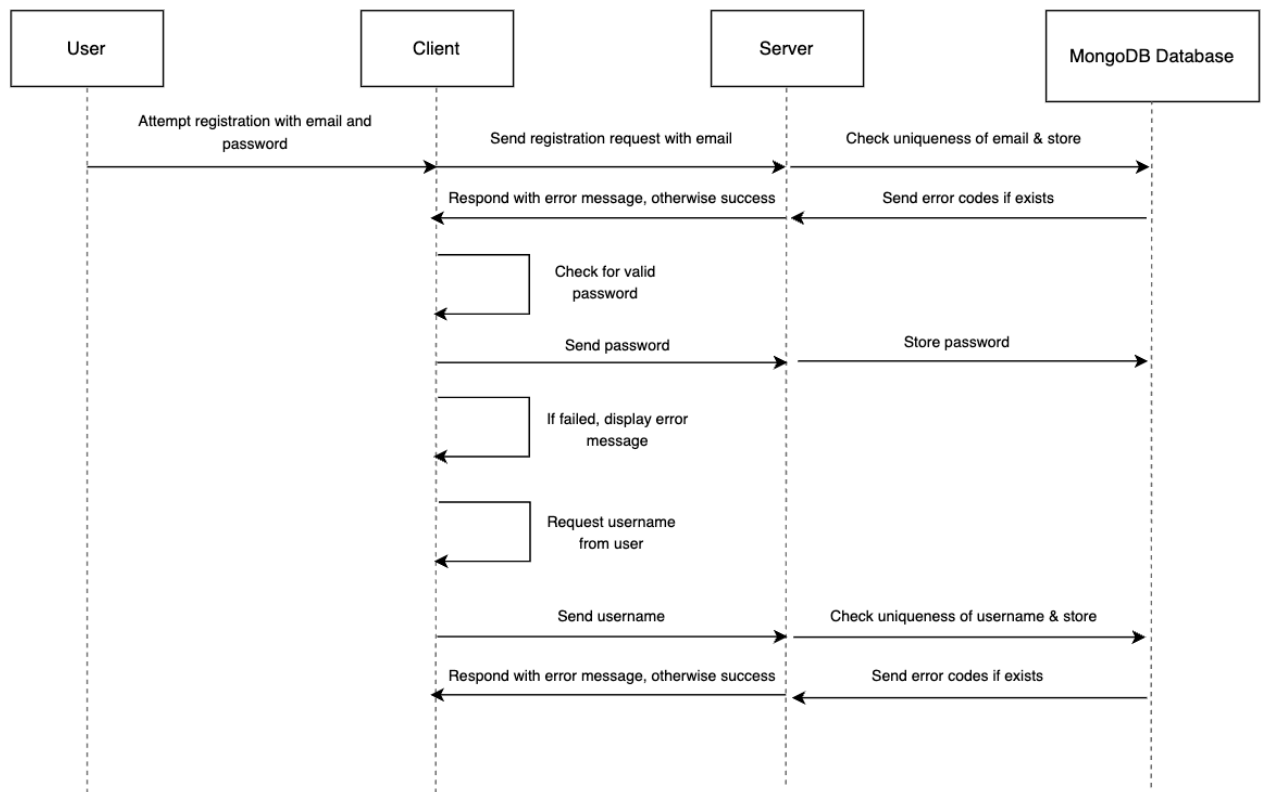
{
  "id": "primary key",
  "from": "user id",
  "contents": "string",
}

```

Recipes

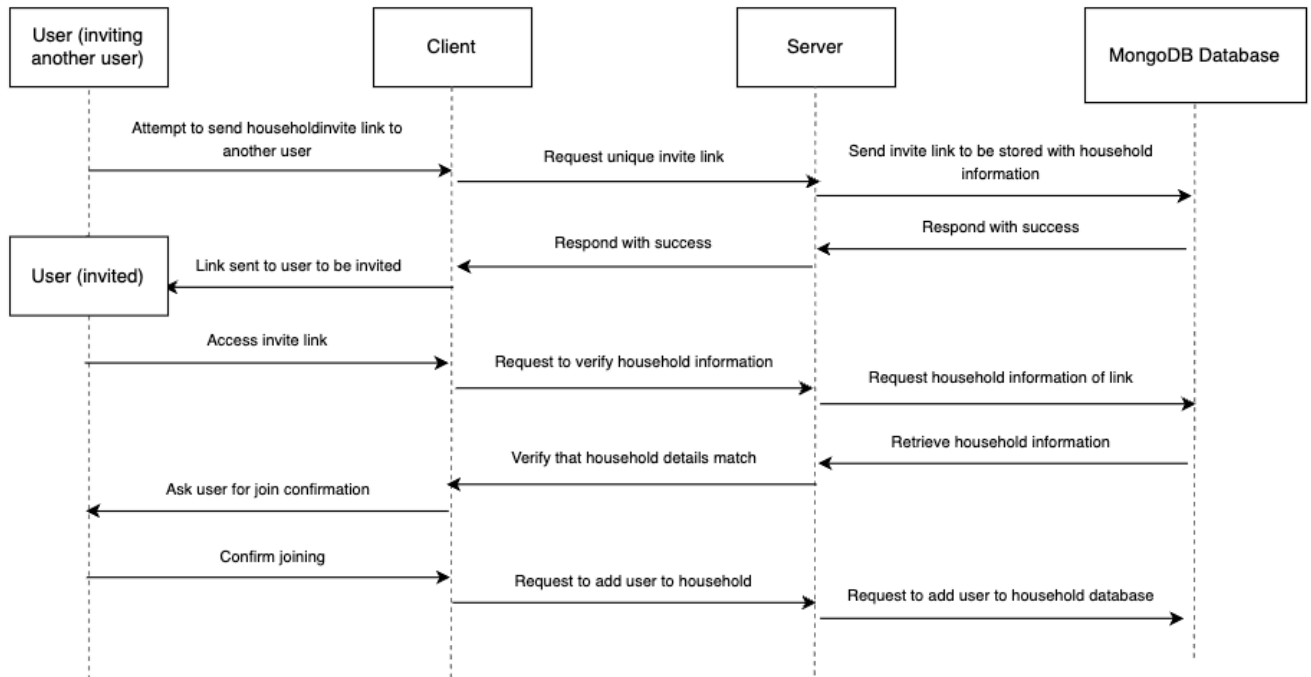
```
{  
  "id": "primary key",  
  "from": "user id",  
  "name": "string",  
  "ingredients": ["string1", "string2" ...],  
  "steps": "string"  
}
```

Sequence for Creating ChipIn Account



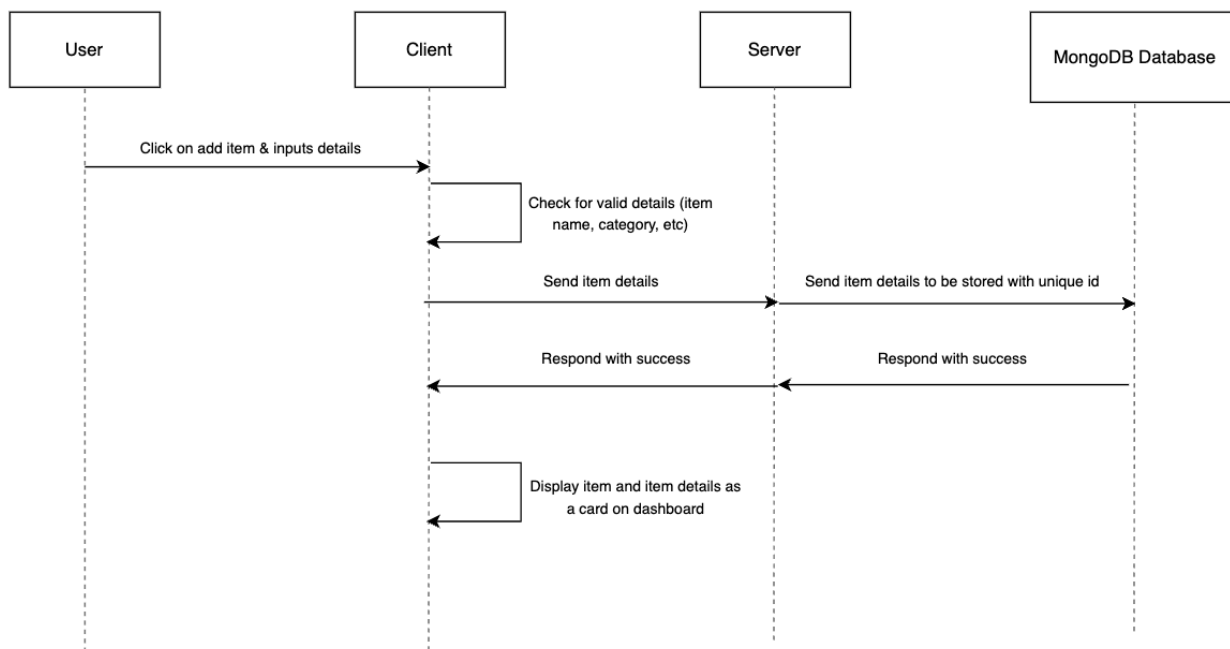
This is a sequence diagram for creating a ChipIn account. The user begins attempting to create an account by submitting an email and password, which the client sends to the server. The server checks the email's uniqueness via the database and returns success or an error if the email exists. The client then validates the password and, upon success, sends it to the server for storage. Next, the user is prompted for a username, which the server again checks for uniqueness in the database, as users will be identified by their username. The process concludes with either a success message or an error if any validation fails.

Sequence for Joining a Household Group



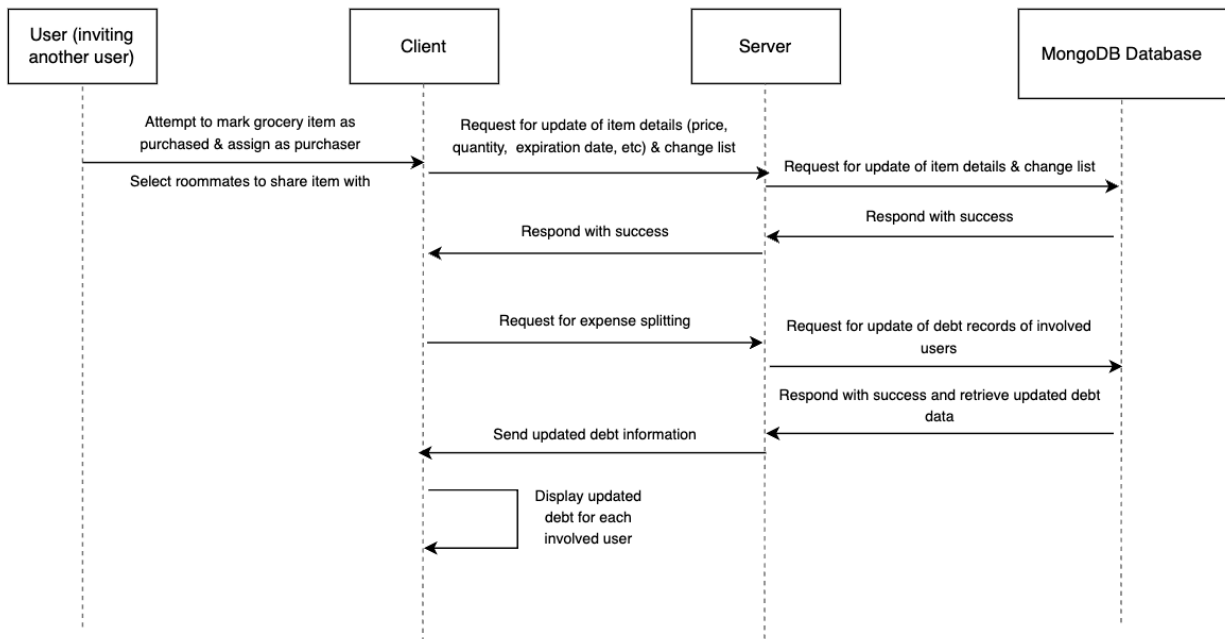
This is a sequence diagram of a user joining a household group. To do so, an existing user will invite another to join a household in Chipln. The inviting user initiates the process by sending a request for a unique invite link, which the client sends to the server. The server stores the link along with household details in the MongoDB database and returns it to the client. The link will then be sent to the invited user. Upon accessing the link, the client will request verification of the household information from the server, which retrieves and verifies the details from the database. Once confirmed, the invited user is asked to confirm joining the household and after confirmation, the client sends a request to the server to add the user to the household, which is updated in the database, completing the process.

Sequence for Manually Adding Item to Grocery List



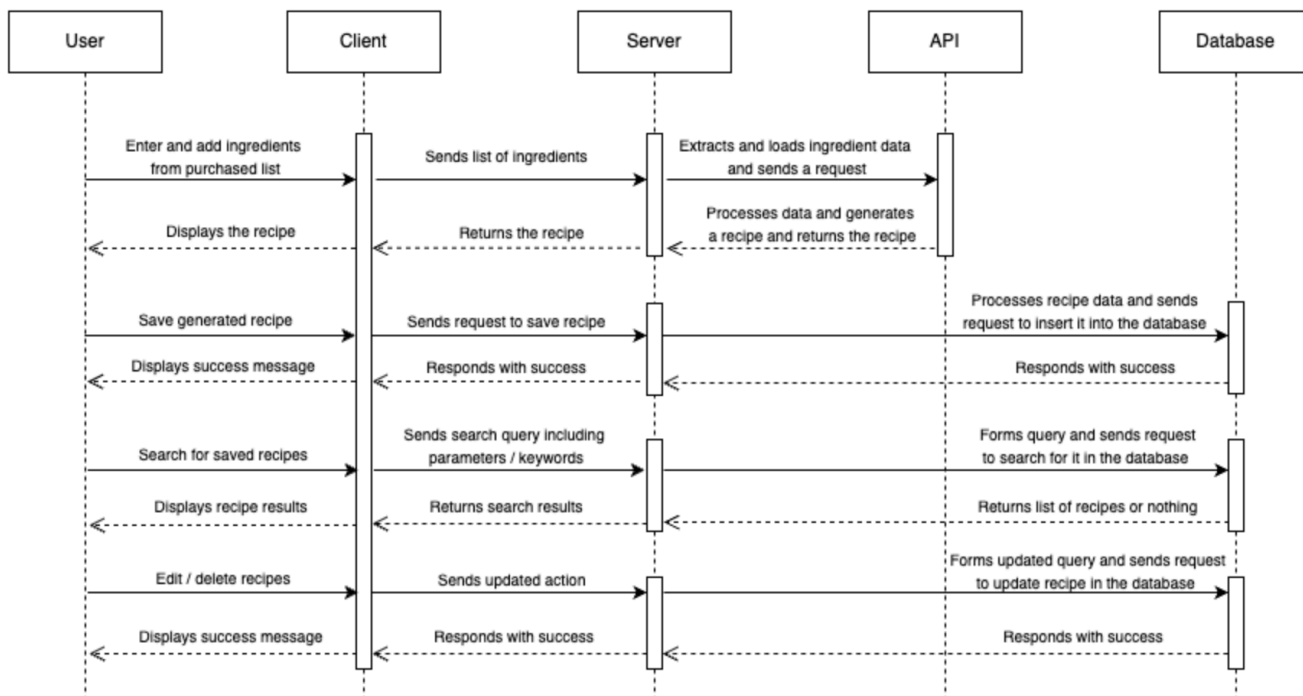
The sequence diagram illustrates the process of manually adding an item to the grocery list, which is one of two ways a user can do so. The user initiates the process by clicking on an "add item" button and entering relevant details such as the item name and category. The client checks if the entered details are valid and, if successful, sends the item details to the server. The server forwards the information to the MongoDB database, which stores the item with a unique identifier. Upon successful storage, the server responds to the client, which then displays the item details on the user's dashboard. The process concludes with a successful response to the user.

Sequence for Moving Grocery Item to Purchased List & Updating Expenses



This sequence diagram depicts the flow of users moving items from the grocery list to the purchased list. After moving the item to the purchased list, the database will update the purchased list accordingly. The user can also request to update the shared expenses once the purchase list is updated. The client will send this request to the server and the database will update the debt records before sending the data back to the client. Users can also choose to share items with other roommates and request uneven splits, in which the MongoDB database will update the debts of involved users and the client will display the new results to the user.

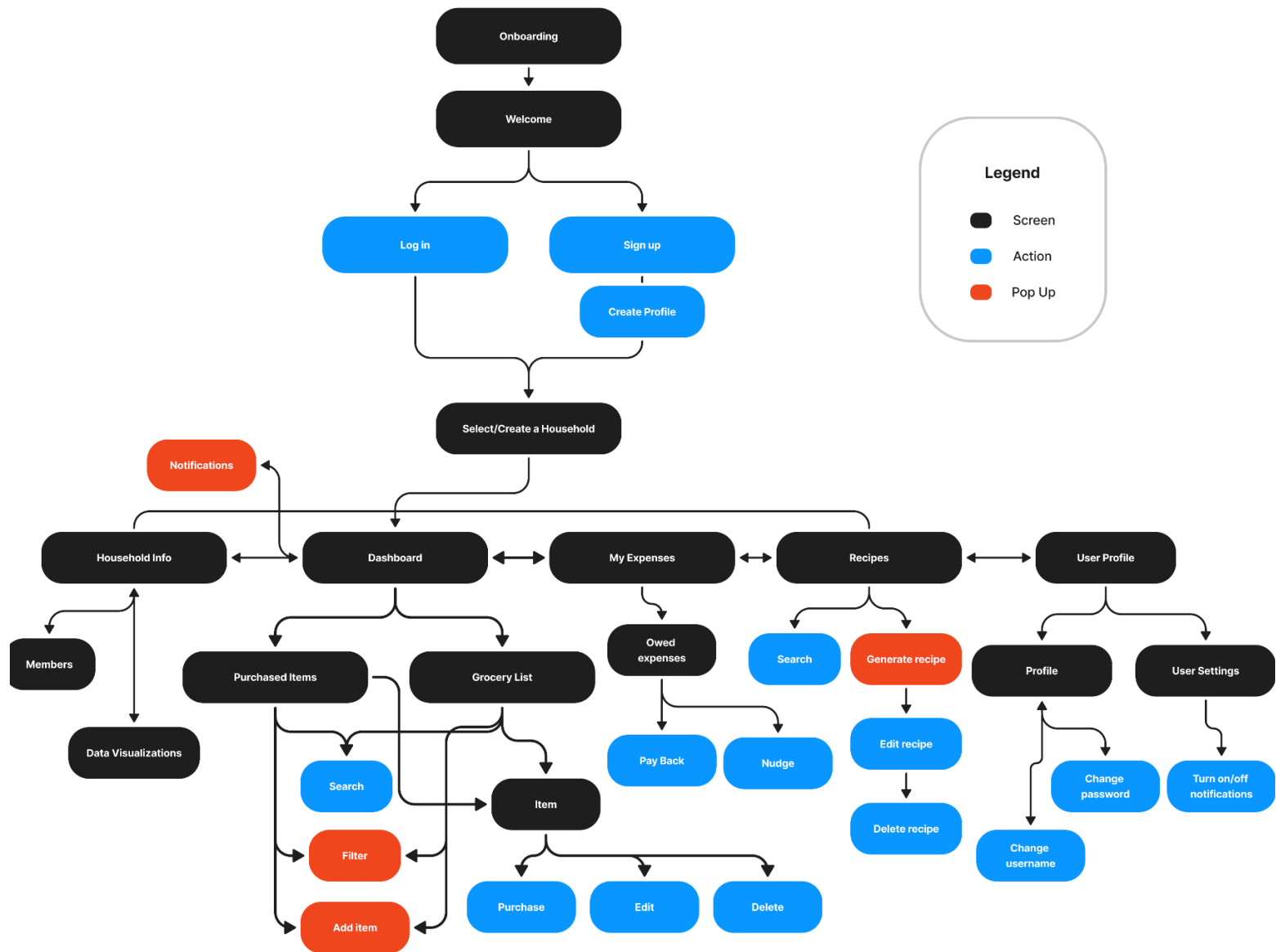
Sequence for Generating AI Recipes



This sequence diagram illustrates the flow of a user generating an AI recipe, a user saving the respective generated recipe, a user searching for any saved recipe, and a user editing and deleting recipes. When a user requests to generate an AI recipe by selecting the ingredients they want from the purchased list, the client will connect with the server which will connect to an API. When a user wants to search for recipes, save generated recipes, or edit and delete recipes, the client will connect with the server which will connect to the MongoDB database.

Navigation Flow Chart



Our user flow will prioritize seamless navigation between pages, along with an intuitive interface that makes it easy for the user to find the function they need. First, the user is prompted to log in or make an account. They are then prompted to join a household if they are not part of one already. After adding a household, the user will be able to access the household's dashboard and all its associated inventory items, expense tracking, and recipe generating functionality. They may also receive and view notifications relevant to the household via pop ups on the site. The sidebar acts as the main navigation component for our page—the user can use it to navigate between the pages specific to their selected households, switch between households, or access their user profile and settings.



UI Mockups

Sign up/ Log in process

LOGO



ChipIn

Create an Account

Email



Username

Password

Sign Up

Already have an account? [Sign in here.](#)

LOGO



ChipIn

Welcome back!

Username


Password

Sign In

[Forgot username or password?](#)

[Don't have an account? Sign up here.](#)

Dashboard

 Basement Dwellers
6 members

Dashboard

My Expenses

Recipes



Settings


HOUSEHOLDS +

Baba Yaga

McDonald's

Basement Dwellers

 Lettuce the Great

Basement Dwellers

Corkboard ^

CLEAR ALL ✎

ADD NOTE +

Alert: Pile of Dirt is about to expire in 1 day!

Note: Purdue Pete is allergic to poison.

Search items

Sort by ▾

Inventory ^

EXPORT EXPIRATION DATES ↗

ADD ITEM +

SCAN RECEIPT +

Fruit

Banana \$100

Shared by chelusa, thy mother, julio, muffin man, lettuce

Added by chelusa

Expires 12/25

Dairy

Milk \$1.69 x5

Shared by chelusa, test454, lettuce

Added by chelusa

Expires 12/25

Misc

Pile of Dirt \$235

Shared by chelusa, Purdue Pete, muffin man, lettuce

Added by chelusa

Expires 12/25

Bills

Wifi \$9999

Shared by chelusa, julio, muffin man, lettuce

Added by chelusa

Expires 12/25

Fruit

Grape \$1

This item is not shared with you.

Added by chelusa

Grocery List ^

ADD ITEM +

Milk 1 + ✎ 🗑


More Milk 1 + ✎ 🗑


Even More Milk 1 + ✎ 🗑


Household Info Page





My Expenses Page

**Basement Dwellers**
6 members


 Dashboard


 **My Expenses**


 Recipes


 Settings


HOUSEHOLDS

 Baba Yaga

 McDonald's

 **Basement Dwellers**



**Lettuce the Great**


My Expenses

Total owed: \$312Total spent: \$999

\$56 was added to your balance for Rice.

You paid back \$6.87 to smbdylsss.


PAY ALL +PAY SELECTED +

**Lettuce the Great**
@letttttuce
lettuce@gmail.com

Owes you \$5You owe \$90

Pay back


Nudge

**Lettuce the Great**
@letttttuce
lettuce@gmail.com

Owes you \$5You owe \$90

Pay back


Nudge

**Lettuce the Great**
@letttttuce
lettuce@gmail.com

Owes you \$5You owe \$90

Pay back

Nudge


**Lettuce the Great**
@letttttuce
lettuce@gmail.com


Owes you \$5You owe \$90


Pay back


Nudge


Recipes Page

**Basement Dwellers**
6 members


 Dashboard


 My Expenses


 **Recipes**


 Settings


HOUSEHOLDS


 Baba Yaga

 McDonald's


 **Basement Dwellers**






**Lettuce the Great**



Recipes



GENERATE RECIPE +

Vegan Dinner   




Fried Rice 5 ingredients

Added by lettuce
lettuce@gmail.com

- Rice
- Olive oil
- Tomato
- Mushroom
- Spinach

EXPAND ▾




COLLAPSE ▴

Vegan Dinner   

Fried Rice 5 ingredients

Added by lettuce
lettuce@gmail.com

EXPAND ▾


Vegan Dinner   


Fried Rice 5 ingredients


Added by lettuce
lettuce@gmail.com


EXPAND ▾


User Profile/Settings


**Basement Dwellers**
6 members


 Dashboard


 My Expenses


 Recipes


 Settings


HOUSEHOLDS 



 Baba Yaga


 McDonald's

 **Basement Dwellers**



**Lettuce the Great**



**Lettuce the Great**
Joined 9/17/2024


Payment Summary



\$9999 spent



\$567 paid back

53 items bought


\$66 owed



Profile 

DELETE ACCOUNT  EDIT PROFILE 


Display Name  



Lettuce the Great

CHANGE NAME 


Username  


@lettuce



CHANGE USERNAME 

Password  



.....


CHANGE PASSWORD 



Settings 


DELETE ACCOUNT  EDIT PROFILE 

Notifications

Household Alerts  



Payment Reminders  



Currency

USA Dollar (\$)

CHANGE CURRENCY 