Shopping & Panty Management App

Context Prompt

For context, the purpose of the app is to provide a complete shopping and pantry management solution with barcode scanning, inventory management, recipe suggestion, and shopping list features. It offers multi-user support and data import/export options, and is built with a tech stack consisting of Visual Studio Code (VS Code) as the Integrated Development Environment (IDE), JavaScript and Handlebars for front-end development, Node.js with the Express framework and a SQLite database for back-end development, and QuaggaJS for barcode scanning functionality.

My Epic

Introducing the ultimate shopping and pantry management solution! With our app, you'll never have to worry about running out of items or forgetting what you need at the store again. Our app allows you to easily scan barcodes with your device's camera to keep track of everything in your pantry, reducing food waste by helping you use up items before they expire.

Our powerful recipe suggestion feature provides you with meal ideas based on what you already have on hand, maximizing the value of your grocery purchases and reducing the need for last-minute store runs. Our "preferences and allergies" feature takes away the hassle of meal planning for those with particular dietary needs and supports those with nutritional preferences like those following a specific diet or diabetics who need carb-specific meal information.

Organization is a breeze with our pantry inventory, which allows you to categorize items for easy searching and keeps track of expiration dates so you never have to throw out spoiled food again. Our shopping list feature makes it simple to add items to your list by scanning barcodes or manually entering information, and you can even transfer your list directly to an online store for seamless shopping, helping you save time and money.

Multi-user support means that families or roommates can share a pantry and shopping list, making it easier than ever to coordinate grocery shopping and reduce the likelihood of duplicate purchases. And with data import/export options, you can easily transfer your information from other apps or share it with others.

Our chat feature allows users to communicate in real-time, so you can easily coordinate shopping trips or share recipe ideas. User management features, including the ability to edit account information and delete your account if needed, ensure that your experience with our app is smooth and hassle-free. With error handling in place, you can be sure that our app will provide relevant messages in case of any issues during the process.

In short, our app offers a complete solution for all your shopping and pantry management needs, helping you reduce food waste, maximize your grocery dollars, simplify your life, and support your dietary needs. Try it out today and start enjoying the benefits!

Development Environment

- Use Visual Studio Code (VS Code) as the Integrated Development Environment (IDE) for coding and building the app. - Run the app on a local development server during development using the localhost environment.

Build Stack

- Front-end: JavaScript and the Handlebars templating engine to build the user interface and handle user interactions. - Back-end: Node.js with the Express framework and a SQLite database to store user information and pantry data. - Barcode scanning functionality: QuaggaJS, an open-source barcode scanning library for the web.

Development Process

1. Set up the development environment and build the user authentication system, including registration, login, and password reset functionality. 2. Implement the barcode scanning feature, including the ability to scan barcodes and store the information in the database. 3. Develop the pantry inventory feature, including the ability to add items to the pantry, view the list of items, and update the quantity of items. 4. Add the recipe suggestion feature, including the ability to suggest recipes based on items in the pantry and provide nutritional information for each recipe. 5. Implement the shopping list feature, including the ability to add items to the list, mark items as bought, and transfer the list to an online store. 6. Add multi-user support, including the ability for multiple users to share a pantry and shopping list. 7. Implement the chat feature using the observer pattern, including the ability for users to subscribe to the chat service and receive real-time updates when new messages are received. 8. Implement data import/export functionality to allow users to import/export their pantry, shopping list, and chat data. 9. Thoroughly test the app to ensure that it is working as expected and fix any bugs or errors. 10. Deploy the app to a hosting environment such as Heroku, which is optimized for Node.js-based web applications.

Requirements

1. User Authentication:
    a. Provide a way to store user information such as name, email, and hashed password.
    b. Implement password reset functionality.
2. Scanning Barcodes:
    a. Allow users to scan barcodes using the device's camera.
    b. Store the barcode information in the database.
    c. Upon scanning a barcode, the app should automatically check the pantry inventory to see if the item is already in the pantry.
    d. If the item is not found in the pantry, the user should be prompted to add the item to the pantry inventory.
    e. When adding a new item to the pantry inventory, the following information should be collected and stored:
        i. Barcode
        ii. Item name
        iii. Quantity
        iv. Perishable Y/N
        v. Best Before Date
        vi. Unit of measurement
        vii. Image of the item
        viii. Timestamp of when the item was added to the pantry inventory.
3. Pantry Inventory:
    a. Allow users to keep an inventory of items in their pantry.
    b. Provide the ability to add items to the pantry inventory by scanning barcodes or manually entering information.
    c. Show a list of items in the pantry inventory, sorted by expiration date, with items that are about to expire appearing first.
    d. Provide the ability to update the quantity of items in the pantry.
    e. Allow users to categorize items in their pantry for easier organization and searching.
    f. Provide a "preferences and allergies" feature that takes away the hassle of meal planning for those with particular dietary needs and supports those with nutritional preferences like those following a specific diet or diabetics who need carb-specific meal information.
4. Recipe Suggestions:
    a. Based on the items in a user's pantry, the app can suggest recipes that can be made using those ingredients.
    b. Provide nutritional information for each recipe.
    c. Allow users to save their favorite recipes for later use.
    d. Maximize the value of the user's grocery purchases and reduce the need for last-minute store runs.
5. User Management:
    a. Implement group functionality:
        i. Provide a way for users to create or join groups that represent flat-mates or people living in the same place who want to combine their pantry.
        ii. Each group should have a name, description, and list of members.
        iii. Users should be able to search for and join groups based on various criteria such as location, interests, or pantry preferences.
    b. Group pantry management:

i. Once users are part of a group, they should be able to view and manage the group's pantry inventory.
ii. The group pantry should be a separate entity from each user's individual pantry, but users should be able to add, update, and delete items from the group pantry.
iii. Users should also be able to view the group pantry's inventory levels and expiration dates to help with meal planning and shopping.

c. Group shopping list:
    a. Users in a group should be able to create and manage a shared shopping list that reflects the needs of the entire group.
    b. Users should be able to add, update, and delete items from the shopping list, as well as mark items as bought.
    c. The shopping list should also be linked to the group pantry, so users can easily add items to the shopping list based on what is currently available in the pantry.

d. User Permissions:
    a. Implement different levels of user permissions to ensure that each user has appropriate access to the group's pantry and shopping list.
    b. Group administrators should have full access to add, update, and delete items from the group pantry and shopping list, while regular users may only have permission to view and add items.

e. Notifications:
    a. Implement notification functionality to keep all group members informed about pantry inventory levels and shopping list updates.
    b. Users should be able to choose their preferred notification method, such as email or push notification, and specify the types of updates they want to receive.
    c. For example, a user may want to receive a notification when an item in the group pantry is about to expire or when a new item is added to the shopping list.

1. Shopping List:
    a. Allow users to add items to their shopping list by scanning barcodes or manually entering information.
    b. Provide the ability to mark items as 'bought' by scanning the barcode in-store or manually marking the item.
    c. Show a list of items that have not been bought yet, sorted by category.
    d. Provide the option to transfer the list to an online store for shopping.
    e. Provide the ability to delete items from the shopping list.

6. Chat Feature:
    a. Implement the chat feature using the observer pattern.
    b. Allow users to subscribe to the chat service and receive real-time updates when new messages are received.
    c. Provide an interface for users to send messages to the chat room.
    d. Store chat messages in the database and display them in the chat interface.

7. Preferences and Allergies:
    a. Allow users to specify their dietary preferences, such as vegetarian, vegan, or gluten-free, as well as any food allergies or intolerances.
    b. When suggesting recipes, the app should take into account the user's dietary preferences and allergies, and only suggest recipes that are suitable for them.

c. Provide a way for users to filter recipes based on their dietary preferences and allergies.
d. When adding items to the pantry inventory or shopping list, the app should provide alerts for any items that contain allergens specified by the user.

8. Error Handling:
    a. Provide appropriate error handling throughout the app to handle unexpected situations and user errors.
    b. Implement user-friendly error messages to guide users in resolving any issues they encounter.
    c. Handle errors related to database connections, network connections, and other potential sources of failure.
9. User Interface:
    a. Design an intuitive and user-friendly interface that is easy to navigate.
    b. Ensure that the interface is responsive and works well on both desktop and mobile devices.
    c. Provide clear and concise instructions for using the app's various features.
    d. Use a consistent design language and color scheme throughout the app.
    e. Ensure that the app is accessible to users with disabilities by following best practices for accessibility, such as providing alt tags for images and using ARIA attributes where appropriate.
10. Performance:
    a. Optimize the app for performance to ensure that it loads quickly and operates smoothly.
    b. Minimize the app's use of system resources to conserve battery life and prevent unnecessary strain on the device.
    c. Implement caching mechanisms to reduce server load and improve response times.
    d. Ensure that the app is scalable and can handle increased traffic and usage as the user base grows.
11. Security:
    a. Implement best practices for security throughout the app to protect user data and prevent unauthorized access.
    b. Store user passwords securely using industry-standard encryption techniques.
    c. Use HTTPS to encrypt all communication between the app and server to prevent eavesdropping and man-in-the-middle attacks.
    d. Implement measures to prevent SQL injection attacks and other common security vulnerabilities.
    e. Regularly perform security audits and penetration testing to identify and address potential security weaknesses.

Database Schema - Base Table Descriptions

---

app_users: This table stores information about the users of the app. The table contains information such as the user's username, email, password, created_at, and updated_at timestamps, name, authToken, pwdResetToken, and pwdResetToken_expiration. Each user is identified by a unique id. The table also contains a foreign key to the groups table, allowing users to belong to a specific group.

---

shopping_list: This table stores information about the shopping lists created by users. The table contains information such as the user_id that created the list, the created_at and updated_at timestamps, and a name for the list. Each shopping list is identified by a unique id.

---

pantry_items: This table stores information about the items in a user's pantry. The table contains information such as the user_id that the pantry item belongs to, the created_at and updated_at timestamps, the item's barcode and name, the quantity of the item in the pantry, and the item's cost, unit of measure, item description, image, purchase date, expiration date, used flag, and recipe_id. Each pantry item is identified by a unique id. The table contains foreign keys to the app_users and items tables, allowing pantry items to be associated with a specific user and item.

---

items: This table stores information about the items available in the app. The table contains information such as the item_id, created_at and updated_at timestamps, the item's barcode, name, description, cost, unit of measure, quantity, and image. Each item is identified by a unique id.

---

password_reset_tokens: This table stores information about the password reset tokens issued by the app. The table contains information such as the user's email, token, and expiration date for the token. The user's email is used as the primary key for this table.

---

chat_rooms: This table stores information about the chat rooms available in the app. The table contains information such as the room's id, name, created_at and updated_at timestamps. Each chat room is identified by a unique id.

---

chat_messages: This table stores information about the messages sent in the chat rooms. The table contains information such as the message's id, user_id of the message sender, chat_room_id of the chat room where the message was sent, message text, and created_at timestamp. Each chat message is identified by a unique id. The chat_room_id and user_id fields have foreign key constraints that reference the chat_rooms and app_users tables, respectively.

---

groups: This table stores information about the groups in the app. The table contains information such as the group_id, group_name, and group_size. Each group is identified by a unique id. The app_users table contains a foreign key to this table, allowing users to belong to a specific group.

---

group_preferences: This table stores information about the preferences of the groups in the app. The table contains information such as the group_id, shopping_frequency, and min_stock_level. Each group preference is identified by a unique id. The table contains a foreign key to the groups table, allowing preferences to be associated with a specific group.

Database Schema

shopping_list

Field Name	Data Type	Nullable	Description
id	INTEGER	No	Unique identifier for the shopping list
user_id	INTEGER	No	User's identifier that the shopping list belongs to
created_at	DATETIME	Yes	Timestamp of when the shopping list was created
updated_at	DATETIME	Yes	Timestamp of when the shopping list was last updated
name	TEXT	Yes	Name for the shopping list

password_reset_tokens

Field Name	Data Type	Nullable	Description

Field Name	Data Type	Nullable	Description
email	TEXT	No	Email address associated with the password reset token
token	TEXT	No	Password reset token
expires_at	DATETIME	No	Timestamp of when the password reset token expires

PRIMARY KEY(email)

chat_rooms

Field Name	Data Type	Nullable	Description
id	INTEGER	No	Unique identifier for the chat room
name	TEXT	No	Name for the chat room
created_at	DATETIME	Yes	Timestamp of when the chat room was created
updated_at	DATETIME	Yes	Timestamp of when the chat room was last updated

PRIMARY KEY(id)

chat_messages

Field Name	Data Type	Nullable	Description
id	INTEGER	No	Unique identifier for the chat message
user_id	INTEGER	No	User's identifier that sent the chat message
chat_room_id	INTEGER	No	Chat room's identifier where the message was sent
message	TEXT	No	Content of the chat message
created_at	DATETIME	Yes	Timestamp of when the chat message was created

PRIMARY KEY(id),
FOREIGN KEY(user_id) REFERENCES app_users(id),
FOREIGN KEY(chat_room_id) REFERENCES chat_rooms(id)

groups

Field Name	Data Type	Nullable	Description
group_id	INTEGER	No	Unique identifier for the group
group_name	TEXT	No	Name for the group

Field Name	Data Type	Nullable	Description
group_size	INTEGER	No	Size of the group

PRIMARY KEY(group_id)

group_preferences

Field Name	Data Type	Nullable	Description
group_id	INTEGER	No	Group's identifier that the preferences belong to
shopping_frequency	INTEGER	No	Frequency of group's shopping
min_stock_level	INTEGER	No	Minimum stock level for the group

PRIMARY KEY(group_id),
FOREIGN KEY(group_id)
REFERENCES groups(group_id)

app_users

Field Name	Data Type	Nullable	Description
id	INTEGER	No	Unique identifier for the user
username	TEXT	No	Username for the user
email	TEXT	Yes	Email address for the user
password	TEXT	No	Password for the user
created_at	DATETIME	Yes	Timestamp of when the user account was created
updated_at	DATETIME	Yes	Timestamp of when the user account was last updated
deleted_at	DATETIME	Yes	Timestamp of when the user account was deleted
name	TEXT	Yes	Name of the user
authToken	TEXT	Yes	Authentication token for the user
pwdResetToken	TEXT	Yes	Password reset token for the user
pwdResetToken_expiration	DATETIME	Yes	Timestamp of when the password reset token expires
group_id	INTEGER	Yes	Group's identifier that the user belongs to

PRIMARY KEY(id),
FOREIGN KEY(group_id)
REFERENCES groups(group_id)

items

Field Name	Data Type	Nullable	Description
item_id	TEXT	No	Unique identifier for the item
created_at	DATETIME	Yes	Timestamp of when the item was created
updated_at	DATETIME	Yes	Timestamp of when the item was last updated
barcode	TEXT	No	Barcode for the item
item_name	TEXT	No	Name for the item
item_description	TEXT	Yes	Description for the item
item_cost	NUMERIC	Yes	Cost for the item
item_UOM	TEXT	No	Unit of measurement for the item
item_quantity	INTEGER	No	Quantity of the item
item_img_path	TEXT	Yes	File path for the item's image
item_img_blob	BLOB	Yes	Binary data for the item's image

PRIMARY KEY(item_id)

pantry_items

Field Name	Data Type	Nullable	Description
id	TEXT	No	Unique identifier for the pantry item
user_id	INTEGER	No	User's identifier that the pantry item belongs to
created_at	DATETIME	Yes	Timestamp of when the pantry item was created
updated_at	DATETIME	Yes	Timestamp of when the pantry item was last updated
barcode	TEXT	No	Barcode for the pantry item
name	TEXT	No	Name for the pantry item
quantity	INTEGER	No	Quantity of the pantry item
item_id	TEXT	Yes	Identifier for the item in the pantry
purchase_date	DATETIME	Yes	Date of when the pantry item was purchased
expiration_date	DATETIME	Yes	Date of when the pantry item will expire
used	BOOLEAN	Yes	Indicates whether the pantry item has been used
recipe_id	INTEGER	Yes	Identifier for the recipe that uses the pantry item

PRIMARY KEY(id),

Field Name	Data Type	Nullable	Description
FOREIGN KEY(`user_id`) REFERENCES `app_users`(`id`), FOREIGN KEY(`item_id`) REFERENCES `items`(`item_id`), FOREIGN KEY(`recipe_id`) REFERENCES `recipes`(`recipe_id`)			

item_measure_types

Field Name	Data Type	Nullable	Description
`id`	INTEGER	No	Unique identifier for the item measure type
`name`	TEXT	No	Name of the item measure type
`default_measure`	TEXT	No	Default unit of measurement for the item measure type
PRIMARY KEY(`id`)			

Parking Lot:

QuaggaJS is an open-source barcode scanning library that uses HTML5 and JavaScript to scan different types of barcodes, including EAN, UPC, Code 128, Code 39, and others. The accuracy of the scanning functionality depends on factors like lighting conditions, barcode quality, and camera quality. However, QuaggaJS provides a robust set of tools to adjust scanning settings and improve the accuracy of the scans.

Users can import and export data from the app using different formats, such as CSV, JSON, or XML files. The app can also integrate with other services like Google Sheets, Dropbox, or OneDrive, using APIs or third-party tools like Zapier or IFTTT.

The app allows multiple users to create and manage their own profiles, which can be linked to a shared shopping list or pantry inventory. Users can invite others to join their profile and collaborate on the same lists or inventory.

Yes, the app could potentially use machine learning or AI-based features to provide personalized recipe suggestions, based on factors like a user's dietary preferences, previous recipe searches, or ingredient inventory.

The app uses robust security and privacy measures to protect user data, such as encryption of sensitive data, secure data storage, and user authentication. The app could also comply with relevant data privacy laws, such as GDPR or CCPA.

Testing the app with real users is essential to improving the user experience and identifying potential bugs or issues. User feedback can be gathered through various methods, such as surveys, user testing sessions, or beta testing programs. This feedback can be used to prioritize feature development, improve UI/UX design, and ensure that the app

meets the needs and expectations of its target audience.

There are different ways to monetize the app, depending on the business model and target market. Some options could include offering a freemium model with ads or in-app purchases, charging a one-time fee for the app, or offering a subscription-based model with premium features and services. It's important to consider the pricing strategy carefully, based on factors like user demand, competition, and the app's unique value proposition.

User Registration: This sequence diagram should illustrate the flow of events that occur when a user registers for a new account, including input validation, hashing the user's password, and creating a new user record in the database. User Login: This sequence diagram should illustrate the flow of events that occur when a user logs in to their account, including input validation, checking the user's credentials against the database, and creating a new session for the user. Password Reset: This sequence diagram should illustrate the flow of events that occur when a user initiates a password reset, including input validation, generating a password reset token, and sending the token to the user's email address. Barcode Scanning: This sequence diagram should illustrate the flow of events that occur when a user scans a barcode with the app, including processing the barcode image, querying the database for the corresponding item, and prompting the user to add the item to their pantry inventory if it is not already there. Pantry Inventory Management: This sequence diagram should illustrate the flow of events that occur when a user adds, removes, or updates items in their pantry inventory, including input validation, updating the database, and refreshing the UI to reflect the changes. Recipe Suggestion: This sequence diagram should illustrate the flow of events that occur when a user requests recipe suggestions based on the items in their pantry inventory, including querying the database for relevant recipes, calculating nutritional information, and displaying the results to the user. Shopping List Management: This sequence diagram should illustrate the flow of events that occur when a user creates, modifies, or completes items on their shopping list, including input validation, updating the database, and refreshing the UI to reflect the changes. Chat Feature: This sequence diagram should illustrate the flow of events that occur when a user sends or receives messages in the chat room, including input validation, updating the database, and refreshing the UI to display the new messages. Data Import/Export: This sequence diagram should illustrate the flow of events that occur when a user imports or exports data from the app, including input validation, processing the data file, and updating the database.