Possible improvements

Current project: https://github.com/M-Aaliyan/grocery-app

List sharing with family members

To improve user experience, we could implement list sharing with family members so that they can view and edit the list together. This feature would allow multiple users to manage the same list in real time. To implement this feature, we would need to make the following changes:

1. Implement User Authentication
    a. Create user login page where user can sign in using email and password or OAuth
    b. Could be implemented using firebase or made custom
    c. Create a users table to track user information such as which lists they have access to
    d. Update api endpoints to include user specific access using JWT
2. List ownership / Family Access
    a. Create an association between lists and user accounts to determine which users have access to which lists
    b. Can be implemented by creating a new table called lists and giving each list a unique id
    c. Items will be associated with lists they exist in, lists will be associated with users that have access to them
3. Real time updates
    a. Use websockets to update the state of the current list across multiple users
4. List invites
    a. Add the ability to invite family members to join a list via email or sharable link
    b. Would need to implement user flow for new users to sign up and join the list

Scaling to a million users

To manage a million users using this app, we would need to implement scaling patterns such as horizontal scalability, performance optimization and infrastructure resilience to handle the load reliably. Here's how this could be done:

1. Use a scalable hosting provider
    a. Instead of using a docker container locally, we could use a cloud platform such as AWS or GCP to manage multiple instances as needed
2. Api scaling and rate limiting
    a. Use a load balancer to distribute requests across multiple backend instances
    b. Implement caching (localstorage, redis, etc.) for data that does not change frequently such as some user info or item presets

    c. Implement rate limits per user to prevent users from abusing the service
    d. Rate limit using tokens that can be used within a time limit or timeouts to prevent multiple calls too quickly
3. Database optimizations
    a. Use indexes for columns that are frequently used in queries
    b. Implement pagination for history and list fetching and lazy load the on frontend
4. Monitor and logging
    a. Track various aspects of the app such as api performance and uptime to find bottlenecks and usage trends

Offline support

When a user is in a grocery store, they may not have access to the internet which would not work for the current implementation of this app. To allow users to use the app without internet access we could implement some of the following changes:

1. Local storage
    a. When offline, we can store items and history in the browser using localstorage or IndexeDB if we want to store larger and more structured data
2. Sync local data
    a. Changes that are made offline are stored locally and we can create a queue for these offline updates
    b. When a user comes back online the queue is processed to sync local changes with the server
    c. Need to take conflicts between local and server data into consideration