Grocery Delivery Services— Shared Order Management

Engineering Design Review

Author: Bingrui li
Student ID: 2509095

Report date: 30th October 2024

1 Introduction

In today's UK, smart home technology has become an essential part of household life. Student Smart Homes (SSH) is a startup focused on providing customized hardware and software solutions for student accommodations. SSH's products are not only directed toward students but also serve student housing providers. Given the specific nature of student accommodations, SSH's products must be adaptable to rental environments without requiring permanent modifications and must meet the needs of multiple users.

The smart home system offered by SSH aims to enhance the quality of shared living in student rental housing. In shared student accommodations, students often choose to place group grocery orders to save on delivery costs and reduce living expenses. This project aims to add a feature to the SSH Console Table and SSH App that allows students to collaboratively create and join a shared grocery order. Users can easily add items from partner supermarkets and view real-time price changes and promotional information, facilitating order management, total cost tracking, as well as individual spending details and cost-sharing for each member of the accommodation. This feature not only simplifies the shopping process but also makes bill management more convenient for housemates, significantly enhancing the user experience.

This feature will be integrated into SSH's existing smart home ecosystem, particularly working in tandem with SSH Hub - First Class, SSH Cloud, SSH App, and SSH Console Table to form a complete solution. By implementing the shared grocery order function, SSH can not only improve the user experience but also promote environmental sustainability by reducing delivery frequencies, while helping students save on costs, achieving a win-win for both economic and environmental responsibility.

2 Goals and non-goals

- **Goal:** Provide a multi-user collaborative shared order management feature, allowing multiple members to simultaneously view and edit the order.

- **Goal:** Enable real-time product information and price updates, order cost-sharing calculations, and support displaying each user's detailed bill information.
- **Non-goal:** Order delivery or supermarket inventory management.
- **Non-goal:** Personalized shopping recommendations.

3 Design overview

Shared Order Management: Students can manage the item list within the shared order via the SSH Console Table or SSH App.

- **Creating a Shared Order:** Students within a shared accommodation can initiate a shared order, allowing all participants to add desired items to the order.
- **Viewing a Shared Order:** Students can view details of the shared order, including each item's information, quantity, price, and the contributor. Item prices update in real time, with promotions or price changes highlighted on the interface.
- **Modifying a Shared Order:** Students can add items to the shared order or remove unwanted items. Each item entry records the quantity, price, and the user ID of the person who added it, enabling the system to track each user's actions.
- **Generate a bill:** Based on the user ID and quantities recorded for each item, the system calculates the total cost of the shared order, each member's individual cost, and generates detailed bill for users to review.

Both shared order and item information will require access to the backend database for queries, with the data then returned for display on the frontend.

3.1 Existing data

Users can access the shared order feature through the SSH Console Table or SSH App, where they can view and add items, check price changes, and see each member's share of the cost. The relevant data is as follows:

Table	Relevant fields	Relevance
Users	**user_id**, name, password, dormitory_number	The User Table records relevant user information.
Orders	**order_id**, ***user_id***, order_name, total_cost, created_by, created_at, finished_at	The Order Table records information about the shared orders, including a unique Order ID, the User ID of the order creator, the total cost of the order, the creation time, and the completion time.

Continued on next page

Table	Relevant fields	Relevance
Order_Items	**order_item_id**,***order_id, item_id,user_id***, quantity,current_price,added_by, added_at	The Order Items Table records information for individual items within an order. This includes a unique Item ID, the Order ID linked to the Orders Table, the Product ID linked to the Products Table, the User ID of the person who added the item, along with the item quantity, real-time price, the person who added it, and the time it was added.
Products	**item_id**,name,description, current_price, stock	The Products Table records information related to each product, including a unique Product ID, product name, current price, stock status, and the latest update time to track changes in price and stock availability.
Bills	**bill_id**,***order_id,user_id***, personal_total_cost, generated_at	The Bill Table records the bill details for a specific user, including a unique Bill ID, the Order ID linked to the Orders Table, the User ID of the billed user, the user's total cost, and the bill generation time.

3.2 Existing utility functions

Creating a Shared Order:

- `createSharedOrder(user_id, order_name)`: Creates a new shared order using the user ID and order name, returning the Order ID.

Viewing a Shared Order:

- `getOrderDetails(order_id)`: Retrieves detailed information on the items within a shared order.
- `checkAndUpdatePrices(order_id)`: Checks for price changes on items within the order and updates the order's price information if changes occur.

Modifying a Shared Order:

- `addItem(order_id, user_id, item_id, quantity)`: Adds an item to the shared order, recording the ID and quantity for the user who added it.
- `removeItem(order_id, item_id)`: Removes a specified item from the shared order.

Generating a bill.:

- `calculateTotalCost(order_id, user_id)`: Calculates the total cost of the shared order.
- `calculateIndividualCosts(order_id, user_id)`: Calculates individual costs based on items and prices each user added to the shared order.
- `generateBill(order_id, user_id)`: Generates a detailed bill, including the user's total spending and itemized details within the shared order.

3.3 Existing role

In the SSH system, specific user roles have been defined to ensure security and appropriate data access. For the shared grocery order feature, here is a description of the relevant roles:

Student Role

This role applies to all registered student users. They have access to the SSH App and SSH Console Table, allowing them to participate in the creation and management of shared grocery orders. The permissions for the student role are as follows:

- Register and log into the system.
- Create new shared orders.
- Add items to shared orders.
- View and modify orders they are participating in.
- View the total cost of an order and their own share of the expenses.

Admin Role

Administrators handle system management tasks and do not directly participate in order creation or management. However, they have permissions to view and manage all orders to assist students in resolving issues when necessary. The permissions for the admin role are as follows:

- Manage user accounts, including creating, modifying, and deleting users.
- Manage product information, including adding, updating, and deleting items.
- Add items to shared orders.
- Handle API integration with supermarkets and synchronize product data.

All shared order functionality interfaces will be controlled by role-based permissions, ensuring data security and privacy through role verification and authentication. This design ensures student users can only access and manage orders they are participating in, maintaining data privacy. Administrators can view or intervene in any order if needed, providing essential technical support in special situations.

4 Alternatives

Decentralized Management Approach

Allow each user to create orders and items on their own devices, with the system not centrally managing order data. Each member's order data is stored locally, and price updates and bill splitting are synchronized through user devices.

Advantages:

- Reduced Server Load: This approach minimizes reliance on the SSH Cloud backend, decreasing resource consumption on the server.

- Enhanced User Privacy: Local data storage reduces the need to store user data on the server, thereby lowering privacy risks.

Disadvantages:

- Difficulty in Ensuring Data Consistency: As each member manages their local order data, synchronizing information becomes challenging, especially during price and inventory changes, leading to significant data consistency issues.
- Limited Real-Time Collaboration: The decentralized management approach struggles to provide real-time synchronization of other members' shopping data, resulting in a user experience that is inferior to centralized solutions.

Conclusion: While the decentralized approach can alleviate server load, it presents notable shortcomings in data consistency, price synchronization, and real-time collaboration, making it unsuitable for shared order scenarios requiring multiple users to work together in real time.

5 Milestones

Milestone 1: Develop backend functionality for shared orders, supporting operations such as order creation, item addition and deletion, and order updates. Implement caching for product information to ensure efficient data access.

Milestone 2: Integrate data from partner supermarkets so the system can retrieve real-time product prices and promotions, with price change alerts displayed when users view the shared order.

Milestone 3: Develop the user interface on both the SSH Console Table and SSH App, allowing users to create, view, and modify shared orders. Display item details, quantity, price, and contributor information, and provide options to add or delete items.

Milestone 4: Implement a bill generation feature, where the system calculates individual expenses based on item contributor user ID and quantity, generates detailed bills, and provides an expense confirmation page for users to view each member's spending and cost-sharing breakdown.

Milestone 5: Gather user feedback to optimize the shared order feature's interface and backend processing logic, ensuring smooth real-time updates and enhancing user experience.

Milestone 6 (Optional): Add order history and basic expense data analysis features to help household members better manage shopping budgets and track spending trends.

6 Dependencies

- *UI Team*: Responsible for designing and developing the shared order interface on the SSH Console Table and SSH App.
- *Backend Development Team*: Responsible for the core development of the shared order feature, including order management, data display, price monitoring, cost calculation, and API integration.

- *Testing Team*: Conduct comprehensive testing of the shared order feature to ensure data consistency between the frontend and backend, as well as smooth interactions, and make optimizations based on user feedback.
- *Configuration Team*: Ensure the smooth transition of the shared order feature from the testing environment to the production environment, configuring and maintaining the system for optimal performance.
- *Legal Team*: Personal information involving users, such as shopping history and spending data, must be handled to ensure legal compliance and prevent privacy and security risks.

7 Cost

- API Call Costs: The shared order feature requires frequent retrieval of real-time prices and inventory information from supermarkets, which can incur costs, especially as the number of users increases and the frequency of calls rises significantly.
- Server and Infrastructure Costs: The shared order feature will increase the server demand for SSH Cloud to handle data storage, caching, and user requests. This includes database requirements for storing orders, product information, user bills, and other related data.

8 Privacy and security concerns

- Data Privacy Protection: The shared order feature requires the collection of basic user information, such as names, email addresses, and order history. This information may include sensitive data. It is essential to ensure that users are clearly informed about what data will be collected and how it will be used before they utilize the feature. A privacy policy and terms of use should be provided at the time of user registration or feature usage, along with obtaining user consent.
- Data Storage Security:
  - Data Encryption: All user information and order data stored in the database should utilize encryption technologies (e.g., AES encryption) to prevent unauthorized access and data breaches.
  - Access Control: Implement a strict access control mechanism that only allows authorized users and system components to access sensitive data. User role management should ensure that different roles have appropriate data access permissions.

9 Risks

Risk	**Mitigations(s)**
API Call Limits or Data Latency	Implement an effective caching strategy to reduce frequent calls to the supermarket API.
Data Privacy Leak	Strengthen data encryption. Implement strict access controls. Establish clear privacy policies.
The complexity of UI design increases	Conduct regular user testing to optimize the interface design.
System performance bottlenecks	Under high concurrency, the system may not be able to handle a large number of requests, leading to slower response times or service interruptions. Implement caching strategies to reduce direct access to the database.
Users may have low acceptance of new features, affecting the practical usage of the system.	Conduct user experience testing, collect feedback, and iterate improvements based on user needs. Promote and advertise the new feature to enhance user awareness and willingness to use it.

10 Supporting material

- Grandview Research. (2024, October 4). Smart Home Market Size, Share And Trends Report, 2030. Retrieved from https://www.grandviewresearch.com/industry-analysis/smart-homes-industry
- Baudier, P., Ammi, C., Deboeuf-Rouchon, M. (2018). Smart home: Highly-educated students' acceptance. Technological Forecasting and Social Change, 132, 83-94. https://doi.org/10.1016/j.techfore.2018.03.019
- Pressman, R. S. (2010). Software Engineering: A Practitioner's Approach (8th Edition). McGraw-Hill Education.