SSH-grocery delivery services

Engineering Design Review

Author:Saibo Guo
Date: 30 October 2024
Status: Draft

Introduction

In some student apartments, students use grocery delivery services to have groceries sent to their homes. Students have busy schedules, balancing academics, social activities, and part-time jobs. Grocery delivery services can save them time traveling to supermarkets. Furthermore, data shows that online grocery shopping has become a trend; for instance, from 2015 to 2017, the number of people choosing online grocery shopping in EU countries saw significant growth(Radka Bauerová 2018). Although students can already make third-party purchases and split bills automatically using the SSH App and SSH Console Table, different supermarkets may require separate applications or websites for purchases, leading students to spend more time searching for products and comparing prices to find cost-effective options, which is often overlooked.

We suggest expanding the SSH app and SSH Console Table to include a new feature aimed at making grocery delivery more convenient for students. This feature would be open to all students using SSH. The company could partner with several supermarkets to provide product information and current prices. Students would be able to search for products within the feature interface and learn about prices at different supermarkets. Additionally, if a supermarket runs promotions or if there are price changes for items already added to students' orders, notifications would be sent to them. Students could place orders together, clearly seeing who added what items and the total costs for everyone. Adding this feature would make grocery shopping more convenient for students, helping them save time and energy for studying and socializing. At the same time, students would be more inclined to choose SSH-equipped student dormitories, which is important for students, clients, and the company.

Goals and non-goals

Goals: The new purchasing interface consolidates all product information provided by supermarkets and offers a search bar for students. At least 50% of SSH-using students participate in orders during the first quarter after launch.

Goals: All users view their individual cost-sharing situations clearly and reduce average delivery costs through shared ordering.

Non-goal: The project does not address specific logistics or delivery quality issues and focuses on order generation and cost-sharing.

Non-goal: The project does not involve quality control of supermarkets and concentrates on real-time pricing of products and order generation.

Design Overview

The goal of this design is to utilize the existing functionalities of the SSH Console Table and SSH application to expand their capability to support shared grocery orders.The system integrates real-time product and price information from partner supermarkets through SSH Cloud, allowing student users to view, add products, and track cost allocation. The SSH Console Table and SSH application serve as the primary interaction interfaces, supporting product search, order management, and cost-sharing functions. SSHHub-First Class connects various devices, ensuring real-time synchronization of order information across multiple SSH devices in the home. SSH Cloud acts as the core backend, managing interactions with supermarket APIs, data storage, and price updates.

Product Search and Addition Module

The SSH application and SSH Console Table allow users to connect to partner supermarket APIs via SSH Cloud to search for product information on their application or desktop device. Users can filter products by name or category and add them to a shared order. SSH Cloud communicates with supermarket APIs to retrieve current product information and prices. When adding products, the system stores the product information and the ID of the user who added it in the SSH Cloud database for order tracking and cost allocation.

Real-Time Price Update Module

SSH Cloud periodically pulls the latest price information from supermarket APIs to ensure the accuracy of product prices in shared orders and pushes price change notifications to the SSH application and SSH Console Table. To reduce data request load, price data can be cached for a period when products are added and updated regularly or synchronized with price changes upon user request. Products with updated prices receive alerts on the order interface.

Order Management Module

The SSH Console Table and SSH application allow residents to view shared orders, including product lists, information about who added each product, product prices, and the total order amount. Users can delete or modify products in the order when necessary. SSHHub-First Class helps synchronize order status across devices. Each time an order updates, the information syncs to other user devices via SSH Cloud or WebSocket, ensuring consistency in order information.

Cost-Sharing Module

The SSH Console Table and SSH application provide a cost-sharing view for each resident, showing total costs and individual expenses. Whenever a user adds or removes a product, the system records that user's cost, and SSH Cloud automatically updates the cost-sharing situation for all residents. Users can also choose to automatically generate bills and notify each person for payment

Databese table design

Table	Relevant field	Relevance
user_information	user_id, user_password, user_address, user_name, user_phoneNumber	Store users account and relevant information
commodity	item_id, item_price, item_number, item_date, item_type,	Item_id stores the unique ID of the product, used for order queries related to the product. Item_price stores the current price of the product. Item_number stores the remaining quantity of the product. Item_date stores the expiration date of the product.item_type stores the category of the product.
order	order_id, order_time, user_id, user_address,user_name, user_phoneNumber total_cost,person_cost	Store relevant information about the order and the user who placed the order for retrieval.
order_item	Item_id, Item_price, order_number, total_cost, user_id, person_cost	Query the order item price using Item_id, Item_price, order_number, and user_id, store the ordered items quantity, and calculate the personal expenditure based on the personal ID, storing it in person_cost. Store the total sum of all person_cost values in total_cost.
item_search	Item_id, Item_price,item_type, search_type	search_type stores the product category input by the user, and then uses Item_id, Item_price, and item_type to list the matching products and their prices.

Alternatives

In this proposal, the SSH system independently operates the shared order module on each user's device (SSH App and SSH Console Table) using a distributed database for data synchronization. Each user's order data is partially stored on their own device, but it automatically synchronizes between household members' devices, ensuring that everyone can view the latest order status.

Pro: Even if one device goes offline, other devices can still view and edit the order, making it suitable for environments with unstable networks.

Pro: Data synchronization occurs only within the home network, reducing reliance on SSH Cloud and lowering the risk of data leakage.

Con: In cases where multiple users edit the order simultaneously, handling data conflicts and consistency issues increases development complexity.

Con: Since the solution does not rely on SSH Cloud, users cannot access orders or perform remote management when outside the home.

Milestones

Milestone 1: Implement backend logic to integrate and cache product data from various supermarkets. This allows for functionality verification and performance evaluation on test data.

Milestone 2: Create a new job type for the existing backend job system to regularly pull data updates from supermarket APIs and integrate the cache. Testing can occur in a non-production environment to validate performance and stability with real data.

Milestone 3: Develop a new interface for the shared order functionality in the SSH App and SSH Console Table. The system will prioritize fetching data from the cache when users view the shared order page.

Milestone 4: Compose notification content in collaboration with the notifications team to draft alerts and updates regarding product information. Integrate notification sending functionality into the backend jobs, utilizing the existing email system for dispatch.

Milestone 5: Conduct UI design and accessibility reviews. The design team will create the shared order page, ensuring optimal display on both mobile and desktop platforms, and perform accessibility checks. Present the final design to a select group of users for feedback.

Milestone 6: Localize notification and page content to support users in different languages. The new functionality will officially launch in the next version of the SSH system.

Milestone 7 (Optional): Allow student dormitories or management organizations to customize settings on the shared order page, such as calculations for delivery cost-sharing amounts and individual payment limits.

Milestone 8 (Optional): Add new API endpoints for other services to retrieve order summary data and create related documentation for developers.

Dependencies

Backend Development Team: Implement the core logic for the shared order functionality, including pulling data from partner supermarket APIs, integrating product information, designing and implementing the caching mechanism, and developing periodic backend jobs.

Frontend Development Team: Develop the user interface for the SSH App and SSH Console Table, enabling users to view, add, and manage shared orders, while ensuring seamless interaction with the backend logic.

Notifications and Messaging Team: Implement and manage the notification system, including drafting messages to remind order participants and developing the sending mechanism for bill notifications.

Design and User Experience (UI) Team:Design the shared order page and notifications, focusing on layout, accessibility, and cross-device compatibility testing, ensuring visual appeal and usability that meet user needs.

Localization and Content Management Team: Provide multilingual support for the shared order functionality and notification content, ensuring a consistent experience for users speaking different languages.

Partner Integration and Business Development Team: Collaborate with supermarkets and third-party vendors to ensure the system can access the latest product and pricing information and update it as required.

Cost

Introducing real-time data aggregation and cache updates will increase cloud computing costs due to regular data synchronization with supermarket APIs. Every data pull for prices and other updates requires storage and processing resources, which can lead to higher demands on the database and servers. To maintain service stability and meet performance requirements, it may be necessary to scale up the current cloud infrastructure, particularly storage and processing capacity.

Privacy and security concerns

The shared order feature collects information on students' shopping habits and order history, such as the items each student adds and individual spending totals. This data qualifies as sensitive personal consumption information, so treating it responsibly is essential.

Encrypt order records and user behavior data during storage and transmission with protocols like SSL/TLS, which secures data in transit and protects against unauthorized access.Encrypt cached product data and order records to prevent any data leaks.Implement strict authentication mechanisms within the SSH App and SSH Console Table to ensure only authorized student users access the shared order functionality.

Risk

Risk	Mitigation(s)
Data Privacy Breach	To prevent privacy breaches, encrypt data in storage and during transmission. Implement strict access control policies to regulate who can view and edit data. Require users to consent to data collection, and conduct regular privacy audits to review compliance.
Unauthorized Users Changing Shared Orders	Unauthorized changes can undermine user trust and system reliability. To address this, implement role-based access control to restrict editing privileges, ensuring that only authorized users can modify shared orders. Use an order change log to track modifications and maintain transparency.
Performance Lag Due to High Data Volume:	High volumes of data can slow system performance, especially for updated items like product prices. Optimize caching and database queries to handle such data efficiently, and perform regular scalability testing under high usage conditions to ensure smooth functionality.

Supporting material

Radka Bauerová (2018). Consumers' Decision-Making in Online Grocery Shopping: The Impact of Services Offered and Delivery Conditions.Acta Universitatis Agriculturae et Silviculturae Mendelianae Brunensis,vulume66,page 124,Retrieved from https://www.researchgate.net/publication/328591072_Consumers%27_Decision-Making_in_Online_Grocery_Shopping_The_Impact_of_Services_Offered_and_Delivery_Conditions