SSH – Group Grocery Order Feature

Engineering Design Review

Author: Harvey Nicholson
Date: 28 October 2024

Introduction

In some student houses, individuals place online grocery orders with supermarkets, which are then delivered. Currently, supermarkets don't offer a system where a single order can be created and added to by multiple people. Consequently, in student houses, if multiple housemates wish to order groceries online, they have two options. The first is for one housemate to place the order for the whole house, and other housemates pay them back for their items. This requires a great deal of trust and cooperation, which housemates may not have. The alternative is that they place separate orders, incurring multiple delivery fees for orders to the same address. This also makes it harder for housemates to coordinate which groceries they are going to share, potentially leading to duplicates of a given item. This places extra strain on already limited student budgets. Whilst Student Smart Homes (SSH) already has offerings aimed at student houses, including the SSH Console Table, which already has the capabilities to split the bill for third party purchases, there is currently no support for making group grocery shopping orders, or splitting the bill in an uneven way to correspond to what each housemate added to the order.

Therefore, we propose adding a group grocery shopping feature to the SSH App and SSH Console Table, to make and view one combined order with a partner supermarket, delivering all ordered items in a single delivery, with one delivery charge, saving participating users money. By helping students economise, this feature provides more value for existing SSH customers, and helps attract new users.

Goals and Non-Goals

- **Goal:** Add a new section to the SSH App and SSH Console Table for users to create a group order from a supermarket. Within one year of release, at least 75% of sampled users should indicate that the feature saved them money.
- **Goal:** Within one year of release, in at least 50% of student houses using SSH technologies, have at most one delivery from each supermarket (at a given time) on order to each house.
- **Non-goal:** Targeting product offer recommendations based on previous orders.
- **Non-goal:** Change customer behaviour to increase the amount being spent per person in an online grocery order.

Design Overview

User Interface (SSH App and SSH Console Table)

The group grocery order feature consists of an item browsing and selection view, and a basket view.

The item browsing and selection view features a search bar, an offers section, and a list of categories.
The search bar enables users to look for specific items to add to the group basket.
The offers section showcases items, line-by-line, that the supermarket is promoting at a lower price. Each line has an image of the item, its name, its base-price, its on-offer price, and a button to add the item to the group basket.
The list of categories (e.g. Dairy, Frozen, Pantry, etc.) provides an alternative to *searching* for items.

The basket view has a breakdown tab, and an items tab.
The breakdown tab shows the total price of the basket and a cost-per-housemate list. Each list item shows the housemate's SSH profile picture, their name, the total cost of the items they added, and a

button that, when clicked, shows the items that housemate added to the basket, with quantities and prices of each item.
The items tab uses the same data as the breakdown tab, but takes an *item-oriented* approach to presenting the data (in contrast to the breakdown tab's *housemate-oriented* approach), showing the order's total price and a list of all items in the basket. Each list row shows an image of the item, its name, its current price (base price, or base *and* offer prices if applicable), the quantity in the basket, and which housemate added the item to the basket.
In both tabs, there is a button for the user to approve the order as-is. If the order is later changed by another housemate, the user is prompted (via notification) to approve the changed order.
We propose that the same UI elements be used for both the SSH App and SSH Console Table implementations, with layout adaptations for the console table's (assumed) larger screen.

Supporting Data
The data drawn on by the group grocery shopping feature are primarily stored in a relational database on the SSH Hub, and includes:

Table	Attributes	Relevance
`House`	`house_id` (PK), `house_address`	Provides the delivery address for the order, and differentiates each house that places group orders with supermarkets.
`Housemate`	`housemate_id` (PK), `housemate_forename`, `housemate_surname`, `housemate_img`, `house_id` (FK)	Key to tracking (with `housemate_id`) which items are added by each housemate to the group basket.
`Store`	`store_id` (PK), `store_name`	Used for differentiating product lists and orders from each partner supermarket (assuming there is more than one).
`Category`	`category_id` (PK), `store_id` (PK, FK), `category_name`	Facilitates sub-sections of a supermarket's inventory so users can discover items *without* using the search feature.
`Item`	`item_id` (PK), `store_id` (PK, FK), `item_name`, `item_img`, `item_base_price`, `item_offer_price`, `is_item_in_stock`, `category_id` (FK)	Allows for relevant item information to be shown to the user when searching for/browsing items, and provides information required to submit orders to supermarkets (i.e. `item_id`).
`Basket`	`basket_id` (PK), `store_id` (FK), `house_id` (FK)	Required by SSH Cloud to differentiate baskets submitted for order from each household, as well to identify the items submitted in orders.
`BasketItem`	`basket_id` (PK, FK), `item_id` (PK, FK), `store_id` (PK, FK), `housemate_id` (FK), `item_quantity`	Stores all items in each basket, so baskets can be viewed on SSH Console Tables and in the SSH App, and then submitted by SSH Cloud for order from supermarkets.
`BasketApproval`	`basket_id` (PK), `housemate_id` (PK), `approval_status`	Tracks which housemates have approved the order, so that only unanimously approved items are ordered. (See below for more detail on this mechanism).

Dataflow
We propose that for participating households, supermarket item lists and group baskets are stored in SSH Cloud, using a relational database containing all mentioned (*see above*) tables.

The `House` and `Housemate` tables are maintained by the Sales and Registration team, ensuring that only active housemates can use the group grocery shopping feature.
The `Store`, `Item` and `Category` tables are propagated daily by fetching from the (assumed to be provided) supermarket APIs that query 'Item' and 'Category' tables in their database. We assume that item prices and availability change at most once-per-day.
The `Basket` and `BasketItem` tables grow when a user adds an item. `BasketApproval` is initialised with non-approval for each housemate when a basket is created.

When clients (SSH App instances or SSH Console Tables) are started, up-to-date versions of the database are fetched from SSH Cloud. After initial power-on, we assume that SSH Hubs and Console Tables are always-connected devices, and thus propose that SSH Hubs synchronise with SSH Cloud when changes are made. SSH Hubs and Console Tables (*in the same house*) synchronise with each other in the same way. Each SSH Hub only fetches records relevant to the house it runs in, preventing any privacy issues arising from storing data from users in other houses.

For each order, at its deadline, order creation is initiated. SSH Cloud combines unanimously approved (through the SSH App or Console Table) items from `BasketItem` that correspond to the `Basket` record in question into a single order. This order is then placed by SSH on behalf of the users in the house, through the (assumed to be provided) 'Order Placing' API. To facilitate this, the existing console table feature for making purchases and evenly splitting a bill, will be extended to run in SSH Cloud and support different contribution amounts from each housemate paying towards the bill. This provides feature support on *both* the app and console table, and ensures housemates pay only for *their* grocery items, and *their* share of the delivery charge.
If, at the order deadline, there are housemates that have not approved the basket, their items are removed, and the order proceeds with items added and approved by other housemates.

Alternatives

Manage basket data locally on SSH Hubs, rather than in SSH Cloud
Store `Basket`, `BasketItem` and `BasketApproval` on SSH Hubs, rather than in SSH Cloud, reducing the amount of data stored by SSH Cloud. This avoids the cost of adding and maintaining data storage capacity to the existing server solution for basket data.
Furthermore, in a short-term SSH Cloud outage, users would still be able to view and amend their orders with their SSH Console Table, whereas centralised data storage would lead to a full outage.

However, infrastructure improvements would be required even without needing to add extra storage capacity to SSH Cloud for basket data. Indeed, increased storage is required for SSH Cloud to store the rest of the new database tables, and thus we argue that adding storage for basket information would make sense to include in the already proposed infrastructure upgrade.
Furthermore, storing basket information on SSH Hubs would require SSH App to request SSH Cloud to fetch and return basket data when loading the app, rather than SSH Cloud providing it directly. This would lead to a performance degradation, which may lead to users abandoning the app [1].

A housemate places the final order, rather than SSH placing the order on behalf of the house
Nominate an individual in the house to take lead for the group order, removing SSH as the middle-entity between the housemates and the supermarket.
This would remove SSH's responsibility to resolve any issues with order or payment details, reducing business overhead in supporting the feature.

However, it would increase the friction in the order process for the nominated individual, which we believe would reduce the usage of the feature. Housemates may also not trust each other to the point which an individual could take sole responsibility for the order. Furthermore, it would also enable the supermarket to track and sell shopping data [2], creating a privacy risk for the nominated individual.

Not having an approval button

Instead of having an explicit approval button, use an implicit approval system, where any item added to a group basket by any housemate is ordered, without further approval.
As users only pay for their items in the basket, and their share of the delivery charge, they would not be financially worse-off from this change, and it would also reduce the required amount of user interaction, in turn reducing risk of burning-out users.

However, it would reduce the coordination of sharing items, increasing duplicate ordering, and may introduce the issue of some individuals being uncomfortable with certain items in the order – none of which align with SSH's goals.

Milestones

- **Milestone 1:** Add the new tables to the SSH Cloud database, and design, implement, and test routines that populate them from the supermarket APIs. This milestone ensures that the mechanisms that provide essential data work before working on dependent systems.
- **Milestone 2:** Design how the new feature is incorporated into the existing SSH Console Table and SSH App UIs. Solicit and then action feedback on the design from colleagues, product managers, and focus groups.
- **Milestone 3a:** Create logic for the SSH App and the SSH Hub to fetch, update and push shopping item, house/housemate, and basket data from/to SSH Cloud.
- **Milestone 3b:** Create routines for the SSH Console Table to fetch, update, and push basket information from/to the SSH Hub.
- **Milestone 4:** Payments team design, implement and test the SSH Cloud version of the payment splitting system. Code reviews are particularly emphasised in this milestone.
- **Milestone 5:** Integration testing on the basket and payment systems, testing full run-throughs and edge cases to catch bugs.
- **Milestone 6:** Feature rollout to a limited number of households, with monitoring from all development teams for issues and bugs.
- **Milestone 7:** SSH Cloud infrastructure team scales up server capacity to handle increased load from payment processing, house/housemate and supermarket item storage, and increased volume of service requests from the SSH Hub and SSH App clients.
- **Milestone 8:** Contingent on the success of the limited rollout in Milestone 6 and adequate server resilience from Milestone 7, full release to all houses using SSH technologies.

Dependencies

- **UI team:** Needs to design the UI for the SSH App, and then adapt its layout for the larger screen of the SSH Console Table.
- **Consumer product logic team:** Needs to collaboratively create routines for data synchronisation between SSH App and SSH Cloud, SSH Console Table and SSH Hub, as well as SSH Hub and SSH Cloud. They also need to link the backend data to the app and console table frontends created by the UI team.
- **SSH Cloud infrastructure team:** Needs to scale up the SSH Cloud server architecture, ensuring it *is*, and *remains* secure and resilient.
- **SSH Cloud logic team:** Needs to collaboratively create dataflow routines for synchronisation between SSH App and SSH Cloud, as well as SSH Hub and SSH Cloud.
- **Database team:** Needs to add new tables to the SSH Cloud database.

- **Payments team:** Needs to extend the existing payment splitting system to work in SSH Cloud and support different contribution amounts for each person.
- **Sales and Registration team:** Needs to maintain records of each house and housemate using SSH technologies.

Cost

We anticipate that extra cost will be incurred in setting up and maintaining more storage capacity and increasing request processing throughput for the SSH Cloud architecture. However, we expect that this will be offset by increased revenue from SSH Console Table and SSH Hub sales, as well as increased SSH App revenue (assuming it has a revenue-generation model, such as advertising).

Privacy and Security Concerns

In SSH Cloud and on SSH Hubs, data is stored that could be used to track individuals' shopping behaviour. Supermarkets could sell this data to advertisers or to other entities, which could negatively impact users, depending on how entities use the data. Hence, we propose that SSH obfuscates individual shopping habits from supermarkets by placing the combined order on their behalf. Furthermore, we propose dedicating time for consumer product logic and SSH Cloud infrastructure teams to implement multi-faceted data-security protections, mitigating the risk of theft by attackers.

Another concern is the increased scope of payment processing undertaken by SSH, creating more opportunities for attack. To address this, payment information will be encrypted at every step in the chain, and all payment steps should be securely logged.

Risks

Risk	Mitigation(s)
Student engagement with the feature may be low.	Regular marketing campaigns to drive awareness of the feature, as well as taking and actioning regular feedback from users of the feature.
An SSH Cloud outage would prevent users from being able to view and amend their orders.	SSH Console Tables already update their databases whenever a change is made (either on device or in SSH Cloud). SSH App could be altered to fetch up-to-date data in the background. These measures would enable users to view and amend orders in the event of a short-term outage.
The approval mechanism may discourage use of the group order feature.	A user-facing setting could be added to change the app to only notify the user to approve the order near the order deadline. If this mitigation fails, and user retention drops, the explicit approval mechanism could be replaced with the implicit approval mechanism described previously.
A lot of development is required before significant results are produced.	Development milestones have been ordered so as to work on critical and uncertain aspects of the system first, to try and mitigate risk. Furthermore, feedback is sought and actioned at key steps, ensuring feature delivery aligns with product manager expectations.

Supporting Material

1. https://support.google.com/googleplay/android-developer/answer/10911598?hl=en-GB
2. https://www.theguardian.com/business/2023/dec/13/sainsburys-boss-sell-customers-nectar-card-data-tesco