

GrubSplit

Revised Design

**By: Amanda Zhou, Marcos Pertierra, Matthew Miklasevich, Jorrie
Brettin**

Motivation

Ordering in groups is currently a cumbersome process riddled with problems. Here's how a typical group order may take place. First, one person decides they'd like to order food, but not enough to meet any sort of delivery minimum. That person - we'll call them the creator - needs to get a group together. They go around their living space knocking on doors and asking if anyone would be interested in ordering. Eventually, the creator has gotten a group of others - we'll call them joiners - together. None of the joiners have ever ordered from the restaurant the creator has chosen, so the creator's computer must be passed around so each joiner can consult the menu. By the time all the joiners have put in their individual orders, the entire order is a jumbled mess with no way to identify who has purchased what. In response, the creator must verbally confirm which joiner has ordered what, figure out how much that joiner owes, then charge them (via Venmo, Square, PayPal, cash, etc.). With no easy way to keep track of who has and has not paid, mistakes can be made which are ultimately costly for the creator. Once the food arrives - upwards of an hour later - the process of rounding up the joiners must be done again. Furthermore, without a record of who ordered what, joiners may end up getting the wrong food, leading to disgruntled joiners and a botched group ordering experience for all.

GrubSplit is designed to streamline ordering in groups by allowing for parallel processing by multiple users as well as automating most of the process. GrubSplit will allow users to gather groups en masse with a single touch, eliminating the need for gathering people and convincing them to join your order. The food selection process can be done by all users simultaneously instead of each needing to look at a single menu. Calculating cost and charging will be an automated task instead of careful bookkeeping. Notifying everyone when the food arrives will be just one more button press away. Finally, ensuring that everyone pays and gets the right food will be simple - the creator of the order will have a checklist with the users' names, orders, and payment status. GrubSplit will save time, money, and hassle for everyone involved in a group order.

Concepts

Delivery - A delivery of food to a physical by a restaurant.

If you place a delivery, then food will be delivered to the location requested.

Grub - A group of food orders created by multiple users, managed by one. This is used to actually place a delivery order. One Grub represents an order from a single restaurant.

If you create a Grub, many people can order food from the same restaurant, thereby saving money on delivery charges and easily achieving delivery minimums.

SubGrub - A set of food from a single restaurant ordered by one user. One or more SubGrubs are part of a Grub. All SubGrubs of a single Grub must be from the same restaurant, as defined by the restaurant of the Grub.

If a user creates a SubGrub, they can pool their order with a group.

Invite - An invite allows a user to join a Grub.

If a user is invited to a Grub, they are granted permission to create a SubGrub in that Grub.

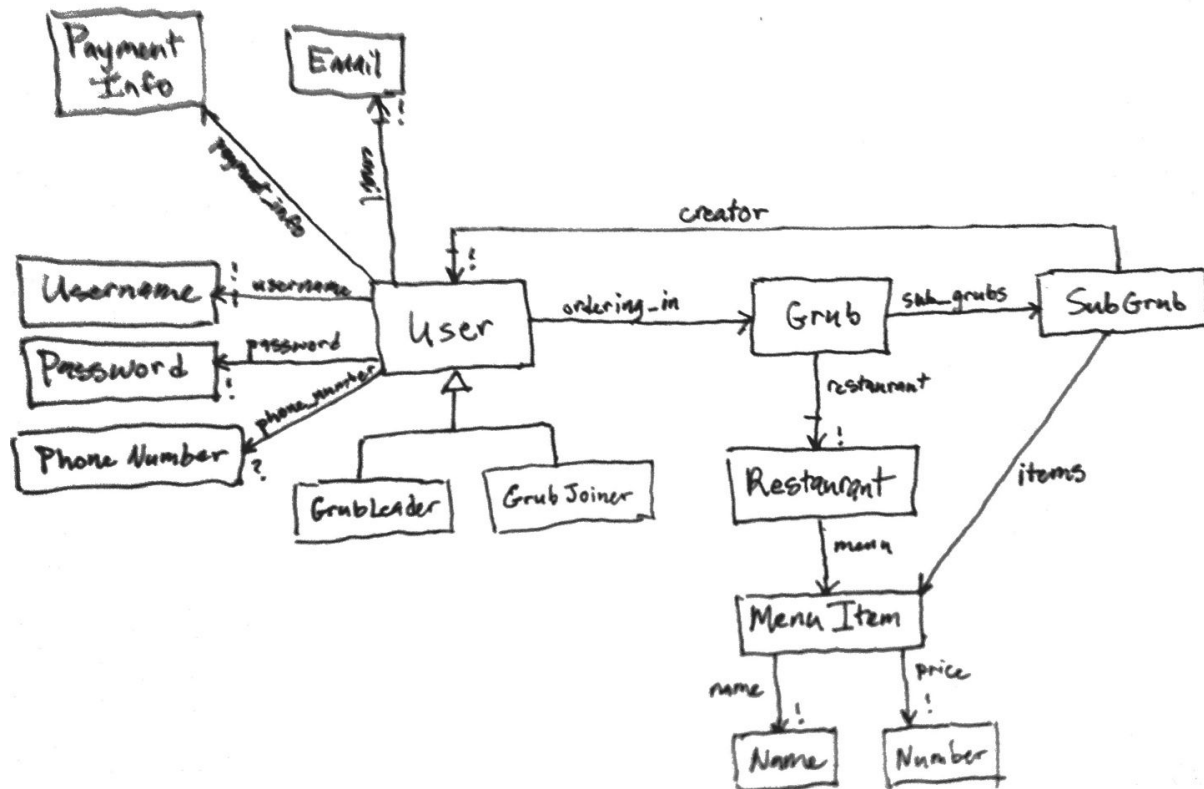
GrubLeader - Creates a Grub and has administrative power over it.

If you are a GrubLeader, you can decide who takes part in a Grub, which restaurant the Grub will be ordered from, and when the delivery will arrive.

GrubJoiner - Creates a SubGrub.

If you are a GrubJoiner, you can place a food order as a part of a Grub.

Data Model



Textual Constraints

The creator of a SubGrub must either be a GrubLeader or GrubJoiner of the Grub that the SubGrub belongs to

Each username must be unique

A SubGrub's items must belong to the same restaurant of its Grub

The GrubLeader of a particular Grub must be the same user that is leading the Grub

The GrubJoiner of a particular Grub must be the same user that is joining the Grub

A User can not be both a GrubLeader and a GrubJoiner for the *same* order.

Design Insights

Each Grub is associated with a single restaurant. If the GrubLeader wants to change the restaurant, he/she must cancel the Grub and create a new one.

In the current data model, menu items could be shared between different grubs. In practice, this means that different orders can contain the same item. This isn't an issue

because each menu item refers to the food's name and price, not the food itself.

Security Concerns

Overview

Each GrubSplit user's profile will contain a username, a password, and payment information. Payment information is incredibly sensitive and any exploits that can access user's bank accounts could be devastating, so security is absolutely critical.

Threat Model

We anticipate that malicious hackers will want to exploit GrubSplit for a number of reasons. If a user's account is compromised and logged into by an attacker, he could order food from restaurants and the unsuspecting user will end up paying for these meals. In the worst case scenario, an attacker could conceivably gain control of a user's payment information and steal money from their bank account.

We expect unauthenticated users to be able to construct requests in order to gain access to accounts or otherwise tamper with orders.

Mitigating Standard Attacks

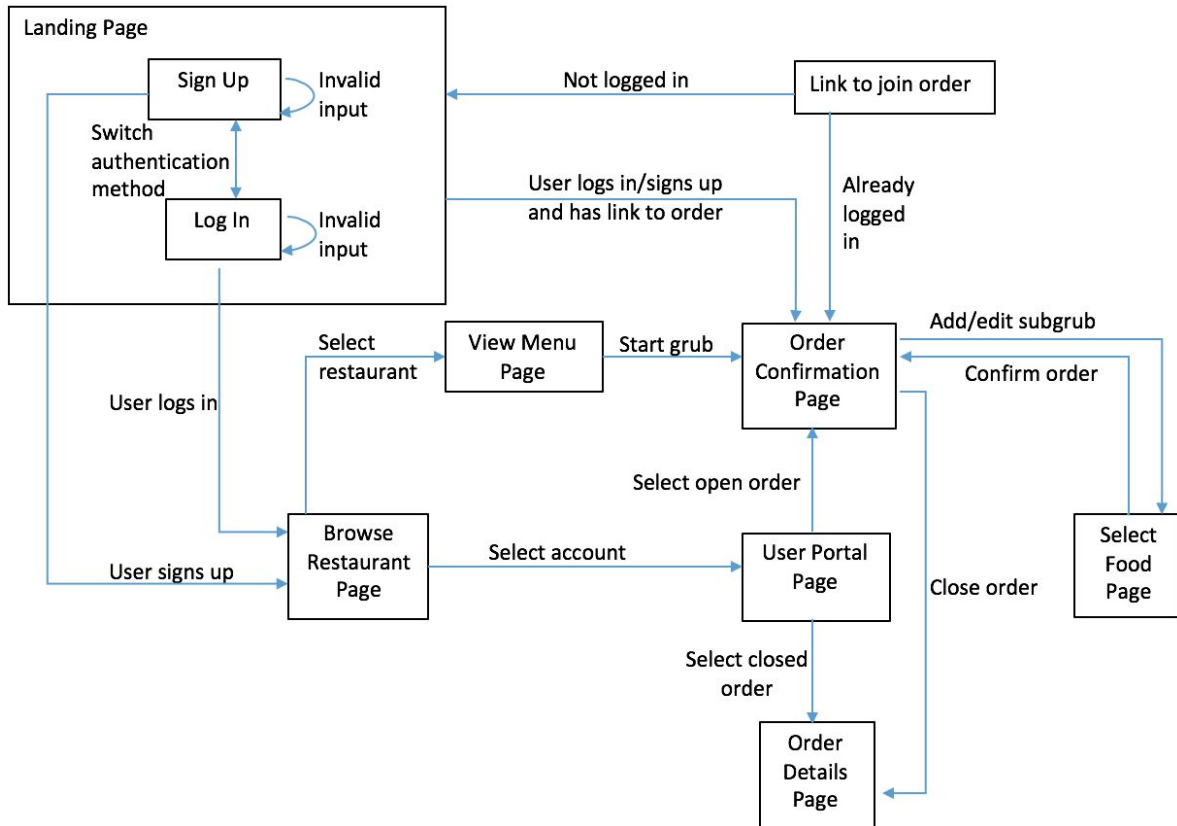
Passwords will be stored in a database, but GrubSplit will never store a user's plaintext password. Instead, we will salt the password and hash it using the bcrypt API. This will prevent attackers from gaining access to user's accounts by intercepting packets containing login information sent over the network. Furthermore, salting passwords prevent rainbow table attacks, where hackers build up a table of passwords and hash values then guess . Finally, we will require GrubSplit to be server over HTTPS rather than HTTP to ensure that all traffic is encrypted.

We aim to prevent XSS (cross-site scripting) by properly escaping all points of user-input. GrubSplit will only require user input through text fields when users register, login, submit payment information, invite friends to an order, and search for restaurants. This will prevent malicious users from running arbitrary scripts in GrubSplit to exploit vulnerabilities and gain access to otherwise secret data. It will also prevent hackers from entering database queries into text fields and gaining access to, modifying, or deleting someone else's data.

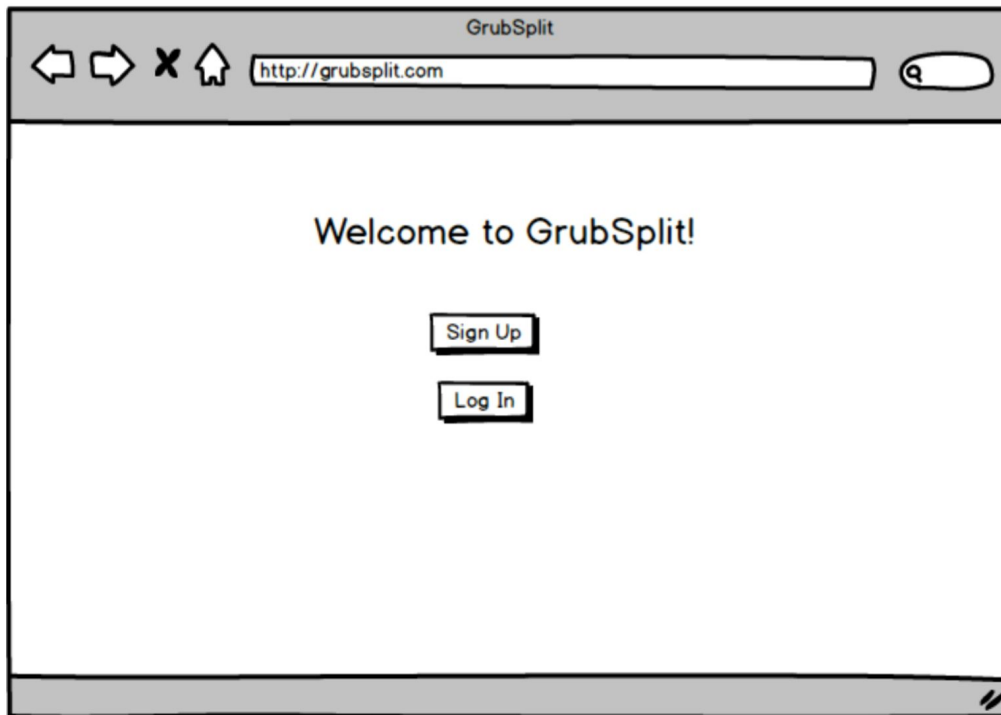
We aim to prevent CSRF (cross-site request forgery) by using hidden form tokens that are session-specific. This ensures that the request was sent by the correct, expected client. The security of this solution is enhanced by using HTTPS to encrypt all client-server traffic.

User Interface

Transition Diagram



User Landing Page



A hand-drawn mockup of a web browser window titled "GrubSplit". The address bar shows "http://grubsplit.com" with a search icon on the right. The main content area displays "Welcome to GrubSplit!" in the center. Below the text are two buttons: "Sign Up" and "Log In", stacked vertically. The browser window has a grey header and footer bar.

GrubSplit

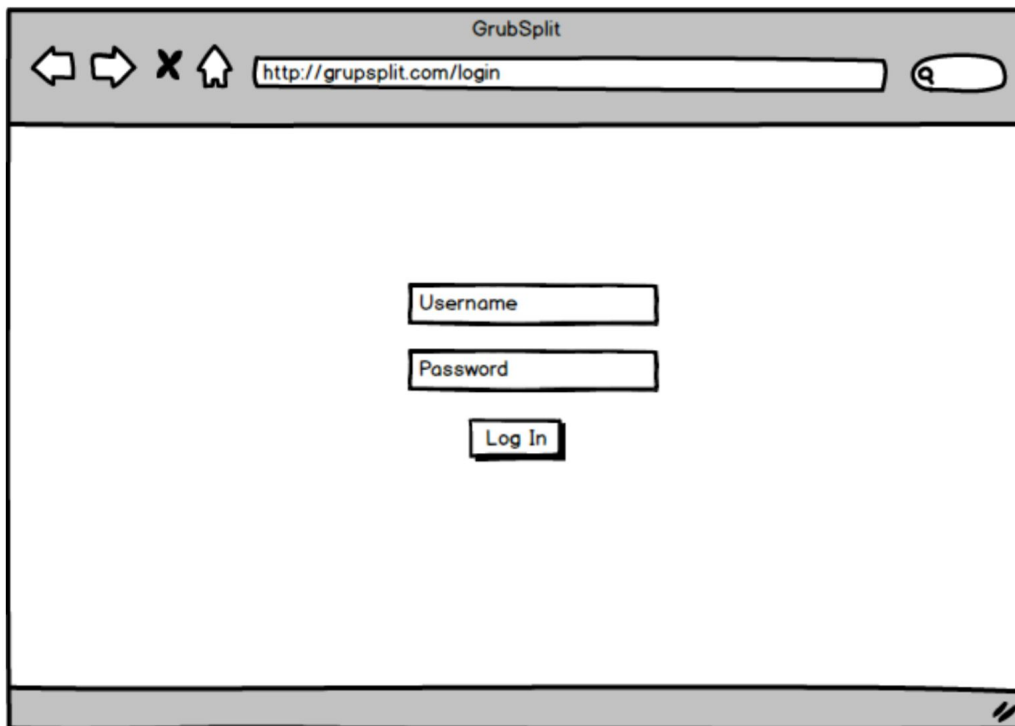
http://grubsplit.com

Welcome to GrubSplit!

Sign Up

Log In

Login Page



A hand-drawn mockup of a web browser window titled "GrubSplit". The address bar shows "http://grubsplit.com/login" with a search icon on the right. The main content area contains a login form with two input fields: "Username" and "Password", stacked vertically. Below the fields is a "Log In" button. The browser window has a grey header and footer bar.

GrubSplit

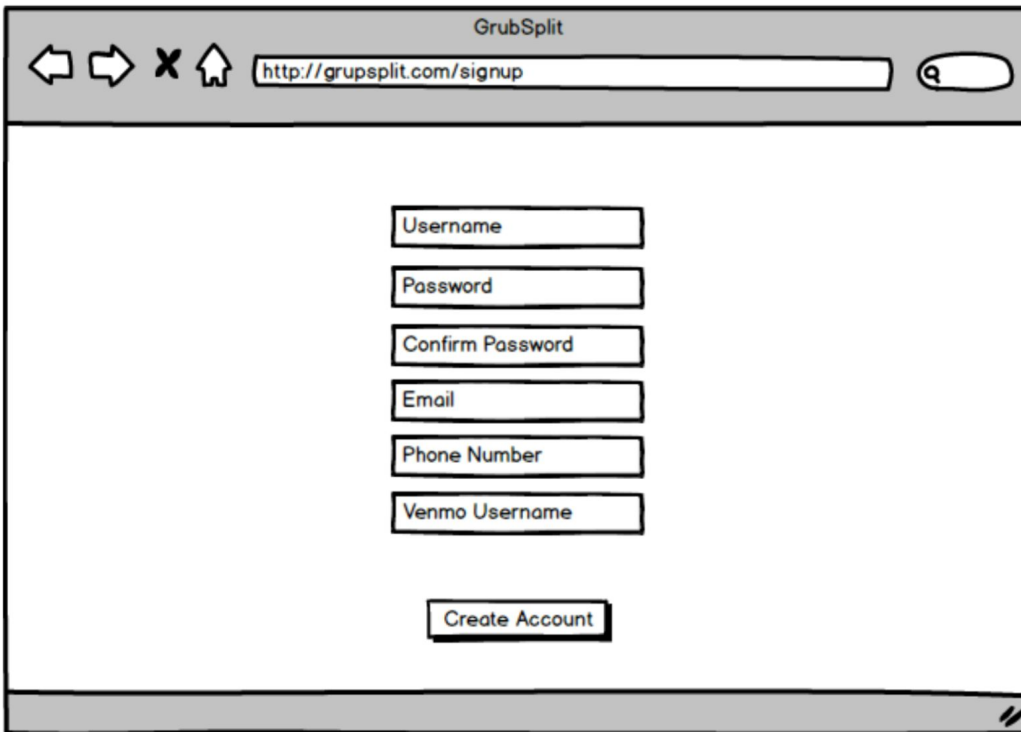
http://grubsplit.com/login

Username

Password

Log In

Sign Up Page



A hand-drawn sketch of a web browser window titled "GrubSplit". The address bar shows "http://grupsplit.com/signup". The page contains a vertical stack of input fields for "Username", "Password", "Confirm Password", "Email", "Phone Number", and "Venmo Username". Below these fields is a "Create Account" button.

GrubSplit

http://grupsplit.com/signup

Username

Password

Confirm Password

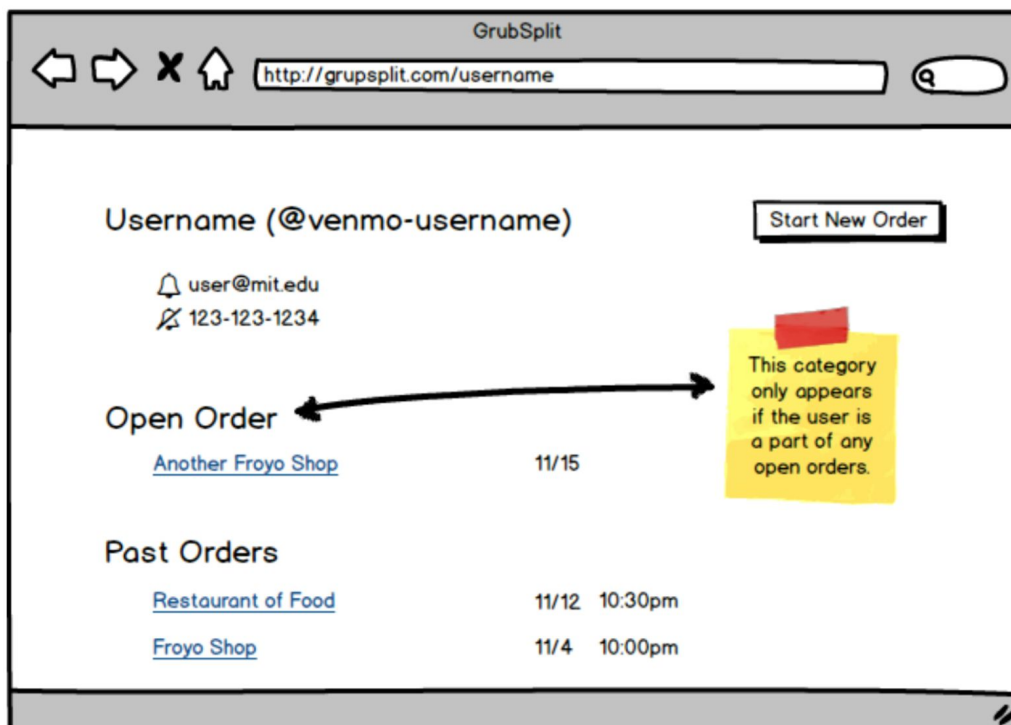
Email

Phone Number

Venmo Username

Create Account

User Portal Page



A hand-drawn sketch of a web browser window titled "GrubSplit". The address bar shows "http://grupsplit.com/username". The page displays user information: "Username (@venmo-username)", a bell icon with "user@mit.edu", and a key icon with "123-123-1234". There is a "Start New Order" button. Below this is an "Open Order" section with a link "Another Froyo Shop" and the date "11/15". A double-headed arrow points from "Open Order" to a yellow sticky note that says "This category only appears if the user is a part of any open orders." Below the "Open Order" section is a "Past Orders" section with two entries: "Restaurant of Food" (11/12 10:30pm) and "Froyo Shop" (11/4 10:00pm).

GrubSplit

http://grupsplit.com/username

Username (@venmo-username)

Start New Order

🔔 user@mit.edu

🔑 123-123-1234

Open Order

[Another Froyo Shop](#) 11/15

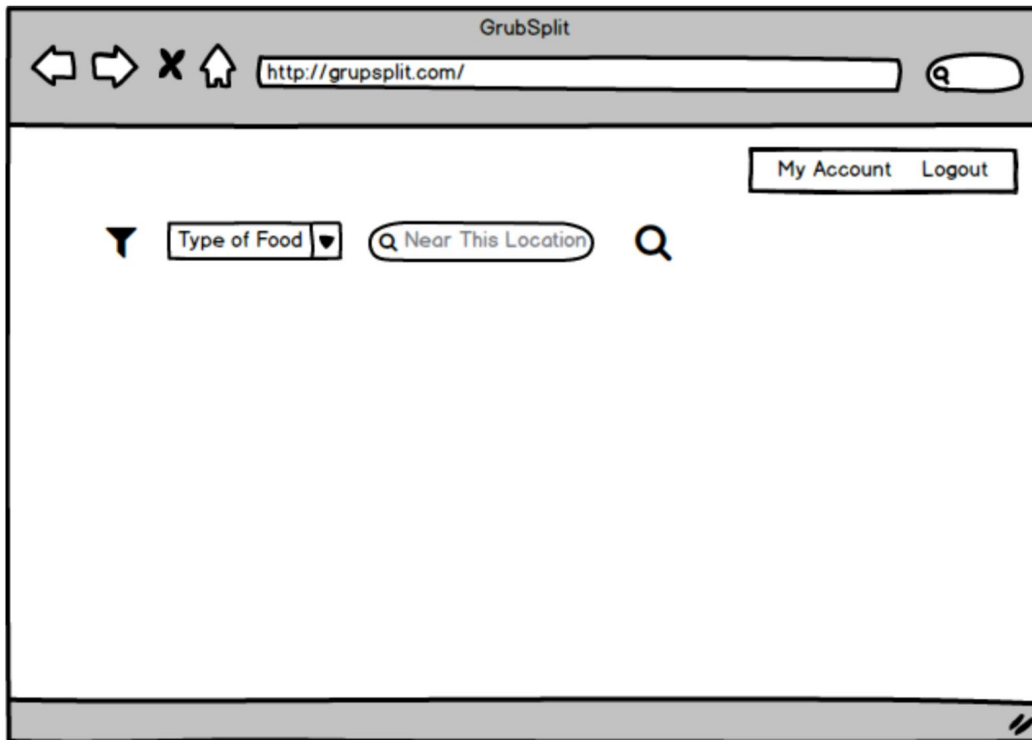
This category only appears if the user is a part of any open orders.

Past Orders

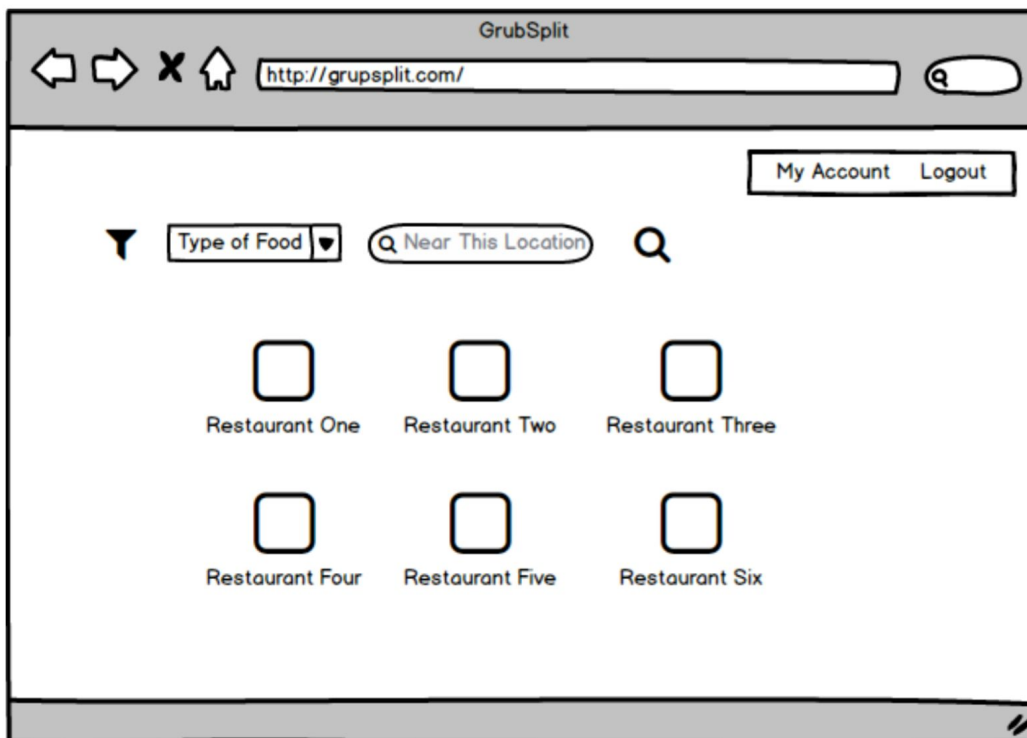
[Restaurant of Food](#) 11/12 10:30pm

[Froyo Shop](#) 11/4 10:00pm

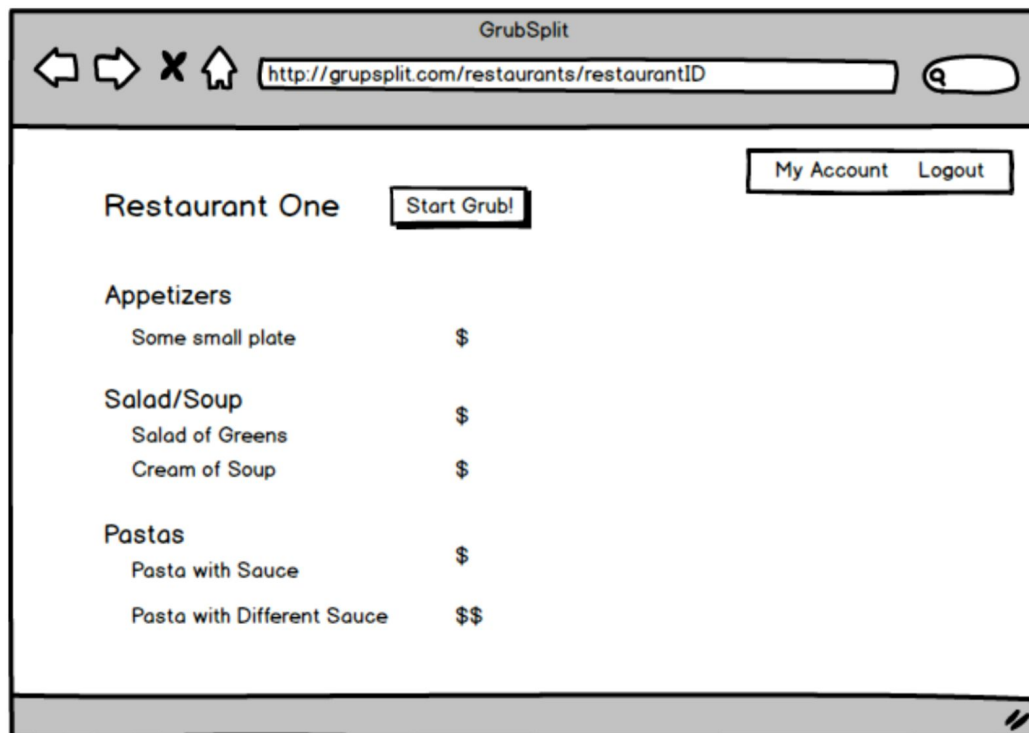
Home Page



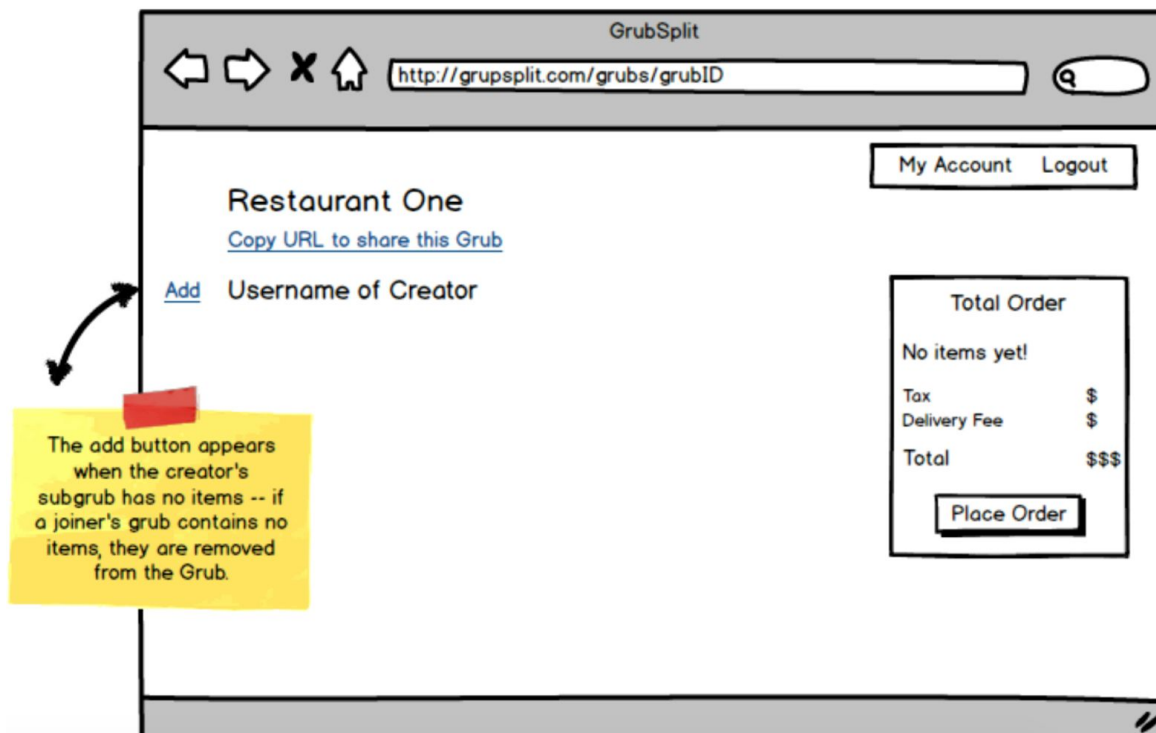
Browse Restaurant Page



View Menu Page



Order Confirmation Page (immediately after initiating grub)



Select Food Page

The screenshot shows a web browser window titled 'GrubSplit' with the URL 'http://grupsplit.com/subgrubs/grubID'. The page is for 'Restaurant One' and includes a 'My Account' and 'Logout' link. The main content is divided into two sections: a menu of food items and a 'Your Order' summary.

Restaurant One	
Appetizers	
Some small plate	\$
Salad/Soup	
Salad of Greens	\$
Cream of Soup	\$
Pastas	
Pasta with Sauce	\$
Pasta with Different Sauce	\$\$

Your Order	
Small Appetizer	\$\$
Pasta with Meat	\$\$\$
Froyo	\$
Total*	\$\$\$
* does not include tax/tip/other fees	
Place Order	

Order Confirmation Page

The screenshot shows the 'Order Confirmation Page' for 'Restaurant One'. It includes a 'My Account' and 'Logout' link. The page displays the order details for the creator and two participants. A yellow callout box points to the 'Edit' button next to the creator's subgrub, stating: 'The edit button appears next to the the user's subgrub (you cannot edit the order of others)'. Another yellow callout box points to the 'Place Order' button, stating: 'Only the creator has the power to place the order - a participant will see the total order but no button to place it.'.

Restaurant One	
Copy URL to share this Grub	
Edit	Username of Creator
Remove	Some small plate \$
	Pasta with Sauce \$\$
	Username of Participant One
	Small Appetizer \$\$
	Pasta with Meat \$\$\$
	Froyo \$
	Username of Participant Two
	Pasta with Sauce \$\$\$
	Cookie \$

Total Order	
Small Appetizer	\$\$
Some small plate	\$
Pasta with Meat	\$\$\$
Pasta with Meat (x2)	\$\$\$
Froyo	\$
Cookie	\$
Tax	\$
Delivery Fee	\$
Total	\$\$\$
Place Order	

Order Details Page

The screenshot shows the GrubSplit web interface for an order from 'Restaurant One'. The browser address bar shows 'http://grubsplit.com/grubs/grubID'. The page includes a 'My Account' and 'Logout' link, an 'Estimated Delivery' time of '11/14 10:30 pm', and a 'Notify Delivery' button. The order is divided into three sections: 'Paid?' (with a 'Remind' link), 'Username of Creator', and 'Username of Participant One' and 'Participant Two'. Each section lists items and their prices, including a total order summary on the right. Annotations highlight that only the order creator can see payment status and send reminders, and that delivery status (ETA or 'Delivered') is shown to all participants.

GrubSplit

http://grubsplit.com/grubs/grubID

My Account Logout

Estimated Delivery: 11/14 10:30 pm

Notify Delivery

Restaurant One

Paid?

☐ [Remind](#)

☒

Username of Creator	\$ including tip/tax/etc
Some small plate	\$
Pasta with Sauce	\$\$

Username of Participant One	\$ including tip/tax/etc
Small Appetizer	\$\$
Pasta with Meat	\$\$\$
Froyo	\$

Username of Participant Two	\$ including tip/tax/etc
Pasta with Sauce	\$\$\$
Cookie	\$

Total Order	
Small Appetizer	\$\$
Some small plate	\$
Pasta with Meat	\$\$\$
Pasta with Meat (x2)	\$\$\$
Froyo	\$
Cookie	\$
Tax	\$
Delivery Fee	\$
Total	\$\$\$

Delivery status will either show ETA or "Delivered"

Only the order creator can see which users have paid and can remind the users who haven't paid to pay.

The order creator can notify all participants of the delivery of the food - this gets sent out as an email/text (depending on the user's preferences)

Challenges

Problems

Relationship between Users, Grubs, and SubGrubs in data model

It was difficult deciding how exactly to map the relationship between a user, their individual SubGrubs, and the complete Grubs themselves. We decided to directly map users to their Grubs, with the relationship “joining” or “leading”. Then, there is a relationship between Grub and SubGrub where a Grub consists of multiple SubGrubs. Finally, a relation maps each SubGrub to its owner. Although it was tricky to describe and capture this behavior at first, clarifying this aspect of our data model helped us design and soon implement a more succinct, efficient user experience and UI.

We considered directly mapping the “joining” and “leading” relationships to SubGrubs, but it would have been difficult to determine or enforce who the single GrubLeader of the Grub actually is. With our current data model, it’s easy to display a list of orders that a user is a part of, and tell whether or not he is leading those orders. The Grub can easily show the included SubGrubs, as well as their owners.

Distinguishing order creators and joiners

From a conceptual standpoint, there is a clear difference between Creators and Joiners. However, when it came to the user interface, we had the decision of creating two order pages (one for Creators and one for Joiners) or just one. Since the Creators have “admin” access to the Grub, they should see all information regarding the SubGrubs. However, for the Joiners, the question was: should they only be able to see their SubGrub or be able to see the entire Grub? We chose to allow all users to see details of the entire Grub. This ensured continuity between the user experience, regardless of being a Joiner or Creator. Additionally, we saw no real benefit to showing users less information regarding the entire Grub. Displaying the entire Grub makes it explicit how the final cost was calculated (by seeing what others ordered, it is more clear why you were charged a certain amount of tax/tip/delivery fee) and may even help people decide what food to order in the future.