

# Potluck Design Doc

Group: ARMR

Authors: Anna Frederich, Rachel Rotteveel, Maddie Severance, Ryan Stuntz

## Overview

[Brief description of system to be built](#)

[Key purposes](#)

[Deficiencies in Existing Solutions](#)

## Design Essence

[Concepts](#)

[Data Model](#)

[Security concerns](#)

[User Interface](#)

[Wireframes](#)

## Challenges

[Design Choices](#)

[Design Risks](#)



# Overview

## Brief description of system to be built

We are building a party planning application to better organize and centralize the important aspects of a party, namely the location, host, guests, and supplies. Furthermore, we want to ensure that everyone contributes equally to the party, whether that be in terms of supplies or money. This application will notify users of the time and location of the party as well as their individual contributions in order to prevent users from forgetting the party or their assigned supplies.

## Key purposes

The main purposes of our app are three-fold. First, we want to make this app an easy way to invite friends and family to attend an event and notify/ remind them of its happening. In other words, the host will input the emails or numbers of all of the guests and then they will be sent an email or text notifying them of their invitation. Second, we want to evenly distribute the bringing of supplies to the event. The host will make a list of all supplies required for the party, and then invited users can logon and volunteer to bring an item. Finally, we want to ensure that the cost of the event is split evenly as well, so if the host buys all of the supplies for the event, they can easily keep track of what they have bought in addition to how much each item costs, and then after the party, each attendee will be notified of how much money they owe the guest.

## Deficiencies in Existing Solutions

Some existing solutions for our problem include creating Facebook events or inviting people in a traditional group chat. Facebook events allow you to invite guests, write a description of the event, and post in the event. However, there is no way for people to coordinate who is bringing what or do equal cost-sharing. Group chats are also inefficient, in that all of the coordination happens in a chain of texts, and you have to read and write dozens of texts to make plans. Tilt is an existing crowdsourcing app for events, but it doesn't help people decide who is bringing what, and doesn't divide costs equally. Our app is different than existing solutions because it aims to achieve all three of our purposes in an integrated way.



# Design Essence

## Concepts

### **Party:**

- *Purpose:* To create the event you will be hosting
- *Operational Principle:* If you click 'Create Party', then you will become the host of the event and be prompted to invite guests who can join your event and help identify supplies essential for your event (add to the supplies list). After the date of the party has passed, the host can 'close out' a party and notifications will be sent to all attendees of the party to let them know how much money they need to pay and to whom. The amount each person owes is based on how much everyone has paid in total for their contributions and what the difference is to make the costs even. A user can indicate on the party whether they have paid someone that they owe, and that party will disappear when everyone has been paid back. If there are no payments to be made, the party disappears once it has been closed by the host.

### **Invite:**

- *Purpose:* To notify a person that they have been invited to an event.
- *Operational Principle:* If a host adds a person to their invite list, then the person will be notified via email that they are invited to a party which they can then view, RSVP for, and sign up to bring items to via a link included in the email. The host and guests can see who has accepted their invites to a party. Invites can be accepted up until the date of the party, and when accepted, that user shows up on the "Attending" list.

### **Contribution:**

- *Purpose:* To assign how guests will equitably provide supplies or money for a party
- *Operational Principle:* When a guest responds to a party's invite, he or she is either held responsible to a monetary contribution to the host if they host is providing all of the supplies, or he or she must make a contribution to the party by bringing one of the items on the supply list. In order to bring one or more items on the supply list, a user can click the "Claim" button next to an item. When a user claims an item, an input will appear next to the item for them to input how much of that item they want to bring and the cost of their contribution. A guest does not have to bring all of the quantity specified next to an item (i.e. if 200 cups are listed on the supply list, a guest can choose to only bring 100, so the supply list notes that 100 are claimed and 100 remain to be claimed). A user can input the cost until the party has been closed out by the host. Then all costs are totaled, and people are notified how much money they owe and to whom (as stated earlier). If a contribution is an item, it is brought to the party. If the bringer decides that item should be factored into cost-splitting, then the bringer notes the item's cost on the application before the host closes the party. If there are any monetary contributions to be made



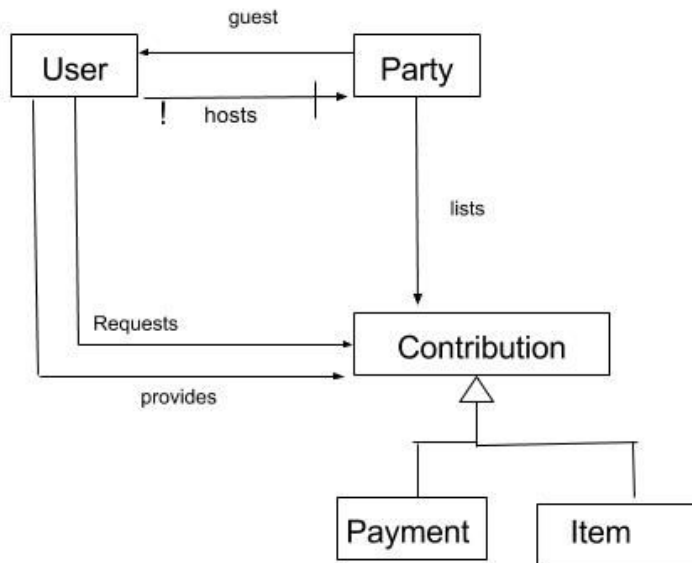
after the closing of the party, each monetary contribution's payer and payee sees the amount to pay or be paid on their homepage. The contribution disappears from the page once the payee verifies receipt of payment.

### Item:

- *Purpose:* To represent a need of the party in order for the party to be well supplied.

- *Operational Principle:* If an item is added to the supply list, then it can be claimed by either an attendee or the host. Items have a name, quantity and units. For example, an item on the supply list could be 3 bags of chips or 5 bags of ice.

## Data Model



**Textual Constraints:** A user who is hosting a party is automatically a guest of that party as well.

**Explanations:** A user can request contributions for a party, in terms of the party's supplies or money to pay for such supplies. A user requesting an item contribution would like to see that item at the party. A user requesting a payment contribution would like to be paid for an item they are bringing. Items and payments are both subsets of contributions because each guest is expected to contribute equally, whether that's in terms of items they bring or payments they provide to other guests. Thus, when attending a party, a user provides items and/or payments.



**Insights:** In terms of users providing contributions, if an item is not claimed before the start of the party, the host has discretion as to whether or not he or she would like to provide that item. If the host provides the item, he or she can decide whether or not to include that item in calculating monetary contributions. If the host does not provide the item, it does not appear at the party. All guests are expected to provide contributions, in the form of payments or items, that amount to roughly the same value. It is up to the guests to determine the best mode of payment and to fulfill promises to one another.

## Security concerns

In all web applications, there are many points of entry for attackers. In our application, we want to try and limit the number of different attacks that are possible in order to best prevent the attacks. In the attacks that we will discuss, we are assuming that attackers do not have access to the server. The attacks that the user may be able to make would include adding items that may be dangerous to either our database or that would attempt to run code on our server. In order to mitigate these attacks, every call to our database will just pass in parameters instead of being created on the fly. In addition, we will not use `eval()` so that no arbitrary code will be able to be run on the server, and will make sure that no other parameters will be used in such a way so that they can execute arbitrary code passed in.

Another thing that could go wrong is users getting access to parties that they should not have access to. In order to mitigate that, we will be making sure that every user is logged in before they ever see a party. By doing so, we can make sure that users never see a party that they should not be allowed to see.

In order to mitigate cross site scripting and request forgery, we will institute a few different requirements. In order to prevent cross site scripting, we will make sure that no input from the user will be used in a context where the html will not be escaped. The way that this will be done is through React. React automatically escapes everything unless you use a `"dangerouslySetInnerHTML"` method which we will not be using in our code.

In addition, to prevent CSRF attacks, we will make sure that we do not allow requests coming from other domains. This will be done through Node by using the `csrf` package which will allow us to ensure that every request is actually coming from our website. This package will take care of most of the details and implementing it as middleware will allow us to be safe on every page.

In order to mitigate security attacks on payment information, we have decided not to store any user's payment information nor handle any monetary



transactions. We will calculate how much each user should pay and to whom and make that known to the payer and payee. We believe our target users would most likely use Venmo to handle monetary transactions, and unfortunately the Venmo API is no longer available to developers. Therefore, we have decided with the resources available to us, limited time for this project, and severity of risks associated with storing payment information, that it would be best for us to forgo the implementation of payments. This design decision will also allow for the flexibility of payers and payees to determine the best mode of payment, whether that be cash, Venmo, PayPal, etc.

We will use the passport.js middleware to handle user authentication. This will allow us to authenticate users based on their email and password and specify routes for authentication. Passport.js manages the necessary sessions and user information required to securely handle requests. In addition we will use bcrypt to hash and salt passwords.

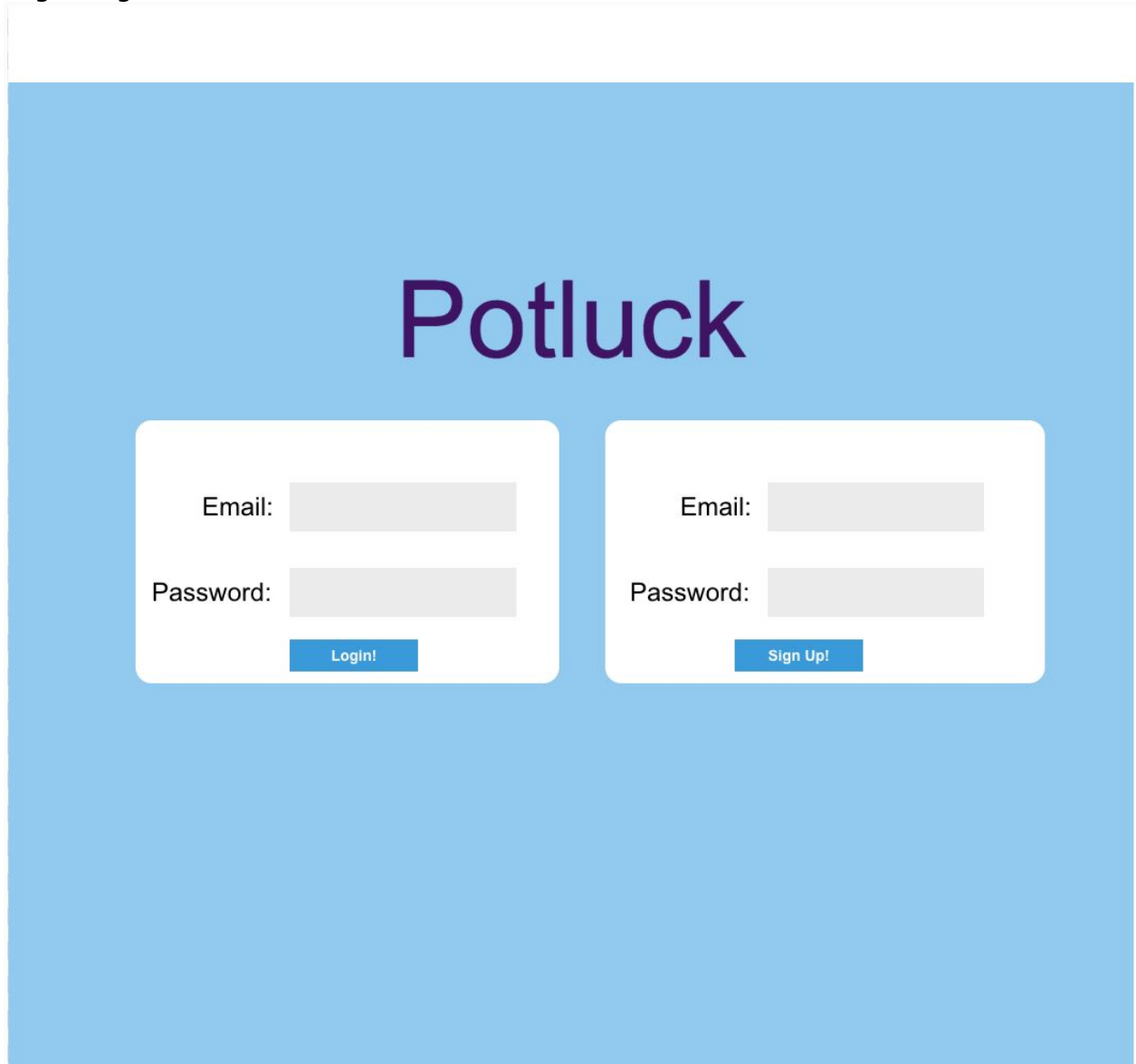
These specified security measures in addition to our design choice not to store secure payment information, will help to protect our application from attacks.



# User Interface

## Wireframes

Login Page:



The wireframe shows a login page with a light blue background. At the top, the word "Potluck" is displayed in a large, dark purple font. Below the title, there are two white rounded rectangular boxes. The left box contains a login form with labels "Email:" and "Password:" next to input fields, and a blue "Login!" button at the bottom. The right box contains a sign-up form with labels "Email:" and "Password:" next to input fields, and a blue "Sign Up!" button at the bottom.



User home page:

Anna Frederich

+

Logout

Past Parties:

Summer BBQ

9-17-2016 1pm

You owe *maddog@gmail.com* 13\$  
Have You payed yet? 

Yes

Upcoming Parties:

Anna's 21st:  
You're hosting

11-30-2016 9pm

587 Beacon St.

View Less

Guest List

Edit

Invited

NameEmail

Anna Frederich  
Ryan Stuntz  
Chelsea Clinton

arfred@mit.edu  
rstuntz@mit.edu  
cc@gmail.com

Supply List

Edit

Item

Qnt.

Contrib

Cost

Cups

100 cups

arfred@mit.edu

\$10

Cups

100 cups

Claim

Tacos

8 tacos

cc@gmail.com

\$80

Secret Santa:  
RachelRo@mit.edu is hosting

12-1-2016 9pm

Stata Center

View Less

Guest List

Supply List

Invited

NameEmail

Rachel Rot  
Anna Frede  
Maddie S

Attending

Anna Frederich  
Chelsea Clinton

arfred@mit.edu  
cc@gmail.com

Quantity:

Enter Item

candy canes


OK



For clarification purposes: The guests are not visible on Holiday Mixer, as this party's "View More" button has not been clicked. When it is, the full view, as seen in Secret Santa, will be visible with both guests and supplies.

Create new party:

Anna Frederich

Logout 

## Create New Party

Title:

Date:

Location:

Description:

### Guest List

Guest Name

Guest Email

Add Guest

Name	Email
Ryan Stuntz	rstuntz@mit.edu

### Supply List

Item Name

Item Quantity

Item Unit

Add Item

Item	Quant	Unit
Cups	200	Cup

Create Party!



# Challenges

## Cost Sharing

Cost sharing is one of the most important pieces of our design and explaining it warrants its own section.

## UI Cost Sharing

To start, we will not be handling money directly through our application. Doing so would open up a bucket of security concerns that would make finishing this application in the time needed significantly more difficult. Instead, when finishing out a party, each user will be notified of whom they owe money to. In addition to the email notification which will occur during finishing out a party, when a user logs onto the website, they will see a different party view (shown in the wireframes section) where they can see who they owe the money to. Once they have completed that obligation (choosing their own method of repayment), they can click the button saying that they have payed and then the party will disappear from their view. If you are owed money by other people, the UI will show that, and you can click to say that they have paid you.

## Algorithm Explanation

In order to figure out the cost sharing functionality, we must discuss how the costs will be split up. After claiming an item, the user will have the opportunity to input the cost of the item. This will be recorded and when the party is finished out, we will run an algorithm that will output how much and who each user owes. Each user will either owe people money, or will be owed money by others, but never both. The reason that it can not be both is if a user is in the situation with both, then the money coming in could just be redirected to the people where the money is going out. This will end up in a case where either the money coming in was larger in the beginning (and the user will just be owed money) or it was smaller (and the user will owe others money). Thus an user can not be in both categories. This algorithm will run through each person, and make sure that their contributions plus the monies owe/d afterwards will equal the total cost of the party divided by the number of attendees.



## Algorithm Pseudocode

1. Sum up all the contributions to get a total, then divide by the total number of attendees to get a per person cost.
2. Sort the list from least contributed to most.
3. Take the person who paid the least, and increase their payment up to the per person cost, giving the amount to the person who has paid the most (up until the point where initial contribution - payment = per person) and moving down to the person paying the second most and so on if the total payment does not yet equal the per person cost. Each of these payments is recorded in the list of payments to be outputted.
4. Remove any people from the list whose updated totals are equal to the per person cost.
5. Continue steps 2-4 until there are no more people left in the list.
6. Output the payments that were found in step 3.

## Design Choices

The first design challenge that we came across was based on the method of authenticating users. Specifically, we discussed how we wanted to implement the signup process for new users. When a guest gets invited to a party via **email** message, should that link go straight to the party page or not? Both sides had valid arguments, but we decided to redirect to a login/signup page instead. The main reason behind this was security concerns as making sure that each person should have access to the party that they are looking at is paramount for confidentiality. By making each user have to have an account before looking at a party, we make sure that only the users who should be able to see a party can. However, the idea behind going straight to the party had some arguments that were intriguing. For one, by doing so, less users would be lost in-between the **email** getting to them and actually signing up for something to bring to the party. In order to speed line this process, we will be populating the sign-up field with the information that the host has already given us about the user (name, email/phone number) so that it will be very quick to sign up.

Another challenge is deciding how the supply list is created, and how that factors into cost sharing. Originally, we decided that the host could either create and buy all of the items on the supply list, and each person could reimburse him/her equally, or each guest could bring one item to the party that was either suggested by the host or by a guest. Ideally, we would like to allow both the host to create an initial supply list, and allow guests to contribute new items to the list if they see a need. However, if the host brings multiple items from their initial list, we would also have to incorporate cost-sharing into that party, even if each guest is bringing an



item as well (so the host doesn't find buying a majority of the supplies to be a burden). Therefore, we can't have black and white "cost-sharing" or "supply-sharing" party types. A party can become a mixture of the two, which might be a challenge in our design. **How contributions and cost-splitting are handled in this mixture case is described in detail in Cost Sharing and Data Model sections.**

## Design Risks

### *Anticipated Misfits:*

1. Making sure that the invite accounts for everyone you want at the party.

Response: We could allow the host to invite additional guests later, not just in the initialization of the party. If we have enough time to implement, we could allow guests to recommend other people they want to see at the party to the host. The host could approve additional guests at their discretion.

2. Determining if supplies people are bringing are evenly distributed.  
Determining whether contributions are defined solely by the host, or whether a guest can choose his or her contribution.

Response: We would allow both parties to define contributions, so guests feel like they have the same say in what items are at the party as the host. For most parties, we anticipate that supplies being bought would most likely be evenly distributed among users. **This is because a user is expected to contribute to the party in some way, whether that's in terms of supplies or reimbursements. Users have the option of inputting costs for items they provide, so that the event's total cost is divided amongst all guests and all guests pay an equal amount. If items are left on the supply list without a provider, the host must decide whether to provide those items or not and whether to be reimbursed for them.** We would also allow a scenario where the host suggests and buys a majority or all of the items, but would allow the host to choose whether to use cost reimbursement for this scenario.

3. Items still being forgotten when creating/adding items to the supply list

Response: We could in the future include a "suggested item" concept where when the supply list is initially created, items would be suggested to be added to the list based on other items added already, number of people being invited, and the type of party.

