Pocket Planner Design Doc

Amanda Liu (ajliu), Everardo Rosales (erosales),
Samantha Briasco-Stewart (erosolar), Sin Kim (kimsin98)
November 13, 2015

Quick links to the different parts of the Design Doc:

Overview

Motivation

We are going to make a system to simplify the process of event planning. To do this, the system will have a centralized place to keep track of what needs to be done in order to plan the event. You will be able to create a new event and record the different steps that have been done and still need to be done by creating To-Dos that will send periodic reminders. These To-Dos will be grouped in Categories that will help the organize the different action items into higher level tasks. Furthermore, it can also keep track of attendees who have expressed interest in attending this event so they can be notified of any updates or changes in the event itself.

The key purposes that this application will address are:

- Organize all of the action items that need to be completed for an event.
- Keep track of and recording/updating important details for an event.
- Notifying event attendees of important updates relating to the event itself.

The main ‘existing solution’ is plain to-do apps. Our app adds on to these to-do apps by bundling to-dos into events, and can give ordering and deadlines to them. In addition, it stores information about the event that is unrelated to to-dos, which a to-do app cannot.
Another existing solution are event social media, such as Facebook events and Eventbrite. However, these applications focus on the publicization and invitations, whereas we will focus on the logistics of planning a great event.

Design Essence

Concepts

Event

A gathering of people at some specific location and time.
Purpose: Categorize and track all of the information associated with organizing a gathering of people.
Operational Principle: If a person wants to organize a gathering of people, then they can create an Event to keep track of information pertaining to that gathering.

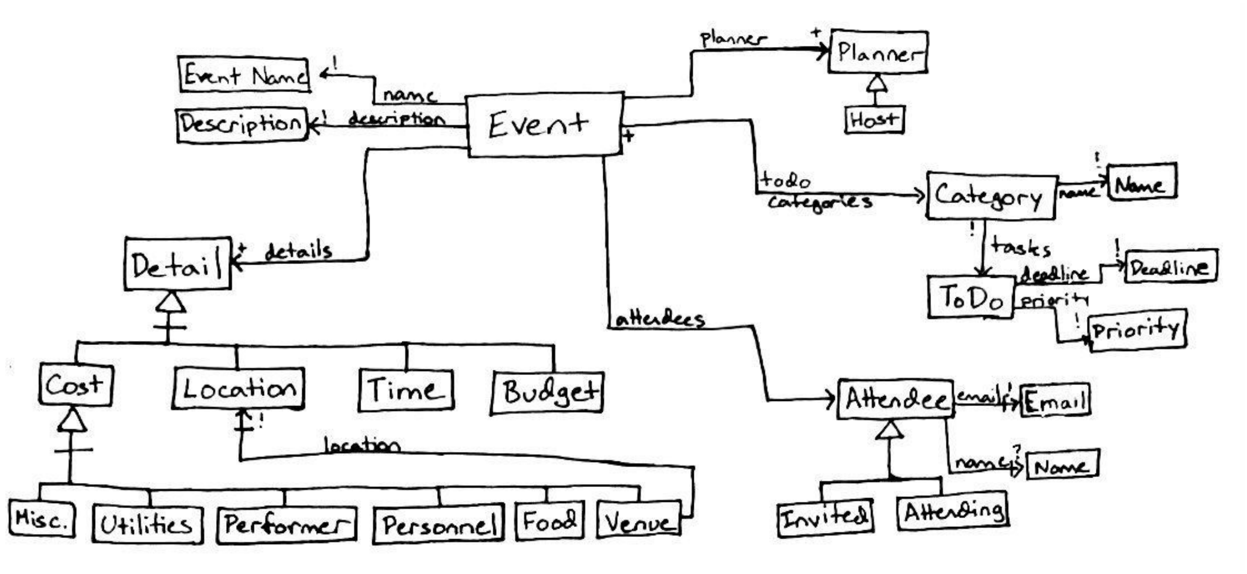
Category

Collection of action items.
Purpose: to group relevant action items together.
Operational Principle: If the organizers have several action items that they want to keep track of that are all related to securing a venue, them they can create a category labeled venue and move the action times into the category to better track the progress of securing a venue.

To-Do

An action item for the event organizer.
Purpose: To keep track of something that needs to be done for the event.
Operational Principle: If the organizer wants to keep track of event logistics such as contacting a catering service, then they can create a To-Do that needs to be accomplished.

Data Model



Textual Constraints

- An event must have exactly one Host Planner.
- An event must have (at least) one Time Detail.

Explanations

Budget vs. Cost

Budget refers to the total amount of money the planner(s) have at their disposal to use on the event. Cost refers to the amount of money spent on a specific part of planning the event (eg. on food, or on a venue). In general, Budget is a positive amount of money, and Cost is a negative amount of money (and if Budget - Costs < 0, you've got a problem!)

Attending vs. Invited

An Invited Attendee is one that has been invited to the event, but has not yet responded yes or no. An Attending Attendee is one that has responded yes, they will come to the event. (Note that we intentionally don't keep track of non-attending attendees, they're just removed entirely from the list.)

Insights

Every ToDo belongs to exactly one category, and we do not allow categories inside categories. We decided upon this model of ToDo organization to make our ToDo's user-friendly, since it closely resembles non-digital to-do lists drawn on paper, blackboards, etc. we have seen before in event planning. The tradeoff is that we limit the scope of ToDo organization, but we do not foresee any practical use case where our model is inadequate.

It should be noted that the atoms pertaining to Events address all of “who, when, where, and what.”

Who: Attendee, Planner
When: Time
Where: Location
What: Cost, Budget

Security Concerns

Security Requirements/Policy

Creators of an event and event planners can have access over editing and deleting an event as well as reading the To-Dos for a given event.
This can be addressed by requiring strict account validation on all pages that are under the event planning side. Moreover, have the planning page and the attendee summary pages under different routes will allow us to handle the difference in authentication easily.

Only event planners (or creators) can send invitations to the event to other people.
Again, having strict account validation on requests that tell the server to forward event updates to attendees would ensure that only planners of the event can forward the email to the attendees.

For a private event, only users that are invited can view the details about the event.
When accessing a private events details page we can require a user to specify an access code, email, or other kind of identifier or secret to make sure that people who aren't supposed to have access to an event don't.

Users can accept or decline only their own attendance to the event.
We are planning on having users use a personalized attendance link when marking their attendance which will be sent

Threat Model

Unauthenticated user can construct requests to tamper with the details of an event

Unauthenticated user can impersonate an event planner to get details of an event before invitations are sent out.

Mitigating Standard Web Attacks

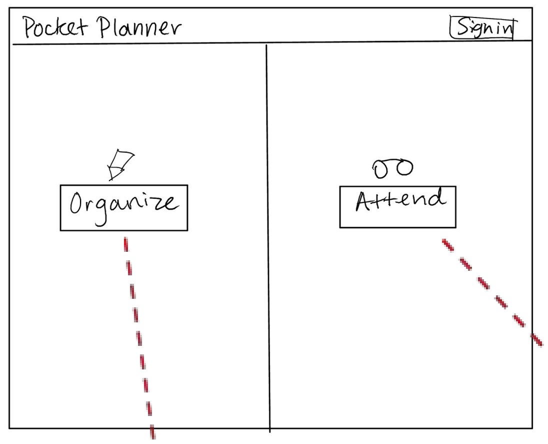
First and foremost, event planner's accounts must be secured and remain secured. We can do this by using a more secure third party for account login and registration. Namely this could be done with Passport or Mongoose-auth if we are building the application with Node.js or using Meteor accounts should we choose to make our application with Meteor.js. These packages would allow us to leverage the security of third party Oauth services without needing to reinvent the wheel. Furthermore, it would also provide a less error-prone way of ensuring that certain pages are only accessed by those who have the correct access privileges.

Regarding resisting XSS attacks, there exist tools to automate XSS testing of webpages. We will use these tools to verify that we are correctly escaped and validator user inputs.

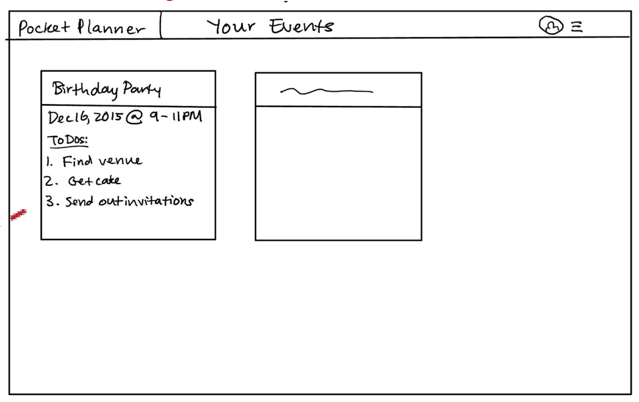
Furthermore, we can also use a tool like validator for Node.js that will make it easy for use to validate inputs. We can use this with all of the inputs on different web pages to make sure that we are getting valid email address, strings and not getting unescaped script tags or javascript.

To mitigate CSRF attacks, we can make use of SSL or TLS encryption when sending request back and forth from the client to the server. As well as attach some session and/or IP specific information. This way messages can’t be easily replayed or forged.

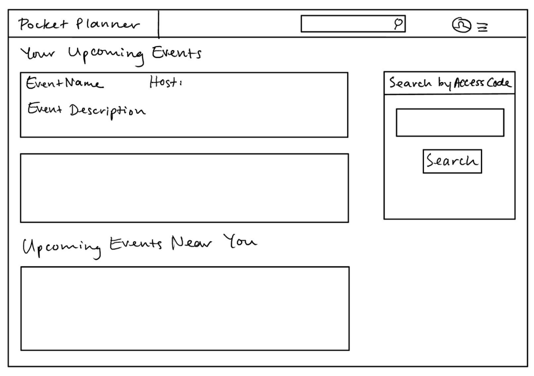
User Interface



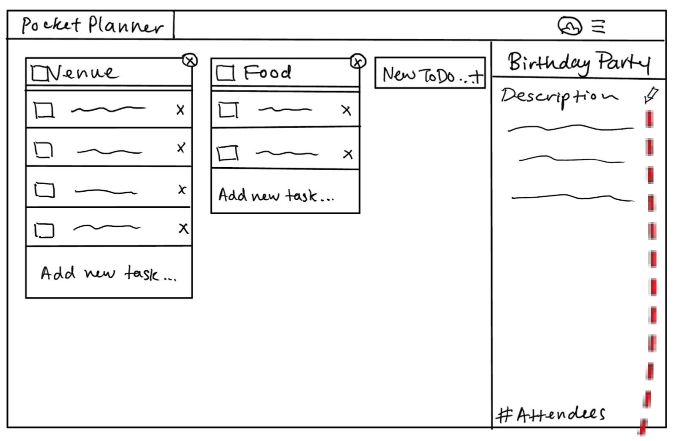
This is the homepage of our app, what a non-signed-in user sees when they load our website. Upon clicking “Organize”, the user will be prompted to sign in.



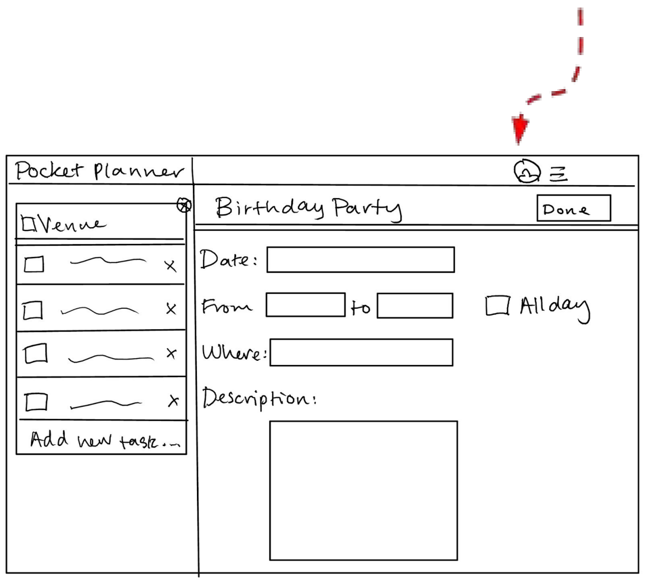
The event organizer main page, with a brief summary of all events this organizer is planning.



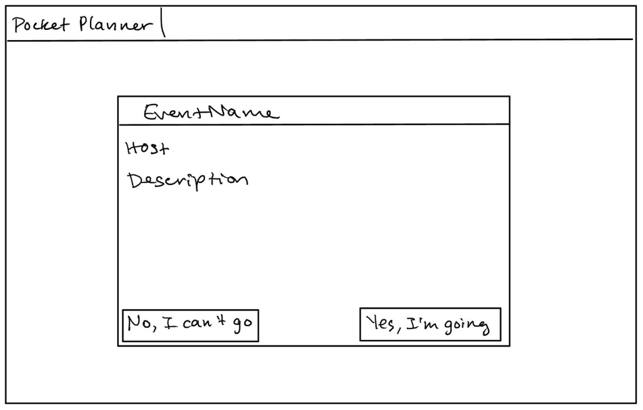
The events gallery, showing upcoming events, giving priority to those a user is attending (if they’re signed in). Also has a hovering box (that will follow as the user scrolls) to enter an access code to RSVP for a specific event.



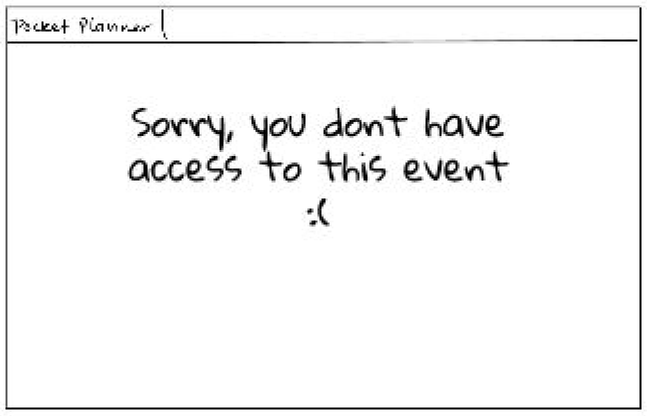
Status page for a particular event. On the right, there is a short, editable summary of the details of the event. On the left, a set of To-Do categories (here, Venue and Food), each containing some To-Dos that have not been checked off yet. Also of note is the “# Attendees” in the bottom right corner, telling the organizer how many people have responded and are attending.



When a user clicks the edit button (where the red arrow is coming from), the summary expands to the left to form this screen. There is enough space left on the left side of the screen to view one of the To-Do categories, and scrolling left or right in that section of the screen will change which one is visible (so it's easy to cross-reference information while editing the event).



Lastly, this is the screen that an invited user sees when they click on the link provided in their email. It displays the event and a short description (including date/time and location), and allows the user to RSVP yes or no.



When trying to gain unauthorized access to a event planning page or when an unauthenticated user is accessing a

Challenges

Design Challenges

Problems to resolve

Spamming

A malicious user uses uses *the app to send unsolicited emails to people.*
Mitigation options:

- Have users provide identification information, such as email, when signing up. This would (hopefully) encourage users to not misuse the app, knowing that we have their personal information.
- Have a spam reporting system, where email recipients can click on a link in an email from pocket planner if they think it is spam. If a significant portion of recipients marks an email coming from a particular event as spam, we could then remove the event, and send a warning to the user that created it.

We're actually going to implement both of these, because they fix different aspects of the problem, and will most likely work better in concert than either would individually.

Over-Attendance

An event attendee clicks the RSVP link more than once, giving the event planner an incorrect number for the amount of attending people.
Mitigation options:

- Send out unique, one-time-use URLs as RSVP links in the email, so that if a potential attendee clicks the link more than once nothing will happen.
- Require potential attendees to enter their emails, or some other bit of identifying information, when they RSVP, so that it will be more annoying and difficult for someone to register multiple times.

Again, we're going to implement both of these, for the same reason as above. They both fix different aspects of the problem, and will most likely work better together than either would individually.

Invite-Only Events

An attendee tells his friends about a (supposed-to-be) private event, thus letting unidentified people into the party.
Mitigation options:

- Unique, one-time-use URLs - so that a user can't just forward the email to his friends and have them respond as well.

- Require attendees to enter their names, creating a guest list for the event planner to use in order to keep unknown people out of the party.

We're going to focus on implementing the first one of these, as it is also something we're implementing for the second challenge. The second is probably also useful but doesn't solve this problem as well, and so we'd like to have that as an option for the user (and as a stretch goal for us) as a useful feature .

Data Design Choices

Declined Attendees, and why it is their fate to be deleted

We decided that it would be best to not keep track of potential attendees that declined an invitation to an event.

Why Venue is a cost, and not part of Location (a detail)

We decided to have this distinction because an event can be at a location that doesn't require acquiring and/or paying for a venue - for example, the internet.

Why Attendees have names

We wanted to give the event organizers an easy way to make a guest list to police attendance at the door (eg. for private events). Giving the Attendees a name field was a very simple way to do this, and when implemented will make the creation of the guest list a lot easier.