Pencil Me In
Caroline Chin | Rebecca Corcillo | Kwame Efah | Abraham Quintero
Updated Design Document
11/18/15

Design Overview

Motivation

Brief description of system to be built

- The system will allow a group of users to find mutually available times based on their google calendar schedules.
- A host user will create a meeting on PencilMeIn and invite others to the event using their gmail accounts.
- Each user will be presented with availabilities determined by their google calendar. Users can choose to modify their availabilities according to their preferences. (See wire frames for a more detailed explanation)
- The webapp then determines a meeting time that matches as many participants' schedules and preferences as possible

Deficiencies of existing solutions (if relevant)

- Overview of Existing Solutions
    - The current workflow for scheduling a meeting between several people consists of Three main steps:
        - The meeting host creates a doodle, whenisgood or an event on a different scheduling platform and gives all of the invitees a set of choices for when the meeting will be held
        - Each invitee has to manually input their availabilities for the specified time ranges of the event, referencing their calendars to see when they are free.
        - The invitee responds to the host.
    - We believe that there is a better way to do this since the invitee's calendar is most likely already in a machine readable format. Why not have the scheduling app take their google calendar into account?
- Deficiencies of existing solutions
    - Email
        - Cannot integrate with personal calendar
        - Poor organization of day of week, date and time
    - Doodle, Congregar, and WhenIsGood
        - Cannot integrate with personal calendar
    - Outlook/Company-Wide Scheduling
        - Not used outside of large companies

Key purposes (what problems does it solve? why should it exist?) , Each purpose summarized in a short sentence and then explained

- To find the optimal mutually available time among a group of users with minimal manual user input

    - why should it exist / problems that it solves:
        - The current methods for determining free time for a group of people requires a significant amount of work from each user.
        - Most scheduling applications also require the creator of the event to manually determine the times for which all users are available.
        - This process creates many opportunities for human error and is extremely inefficient.
    - explanation:
        - Automating this process will reduce both the human error problem and the inefficiency problem to create a better and more integrated user experience for setting up meetings
- To automatically put group meetings for users in their google calendars without them having to do it manually.

    why should it exist / problems that it solves:
        - Given a scheduled group meeting (date/time,location), each member of the group has to manually put that meeting in their respective google calendars for them to be reminded of it / have it in their calendar.
        - explanation:
        - Currently, the general procedure for scheduling a group meeting consists of 1) manually deciding what time to schedule the meeting and 2) each member manually putting that meeting into their individual google calendars so they don't forget/are reminded of it.
        - Since our first purpose automates the first part of this procedure, it makes sense that we should also automate the second part of the procedure, so the entire group scheduling process is as automatic as possible.
        - Automating this process will reduce human error (misscheduling a meeting) and also reduce manually unnecessary work each user must do.
- To schedule a meeting during suboptimal time ranges if necessary
    - why should it exist / problems that it solves
        - The ability for users to specify suboptimal time ranges for which they are willing to meet if necessary makes it more likely that a mutually available time range can be found among a large group of people.
        - Currently, some scheduling applications allow users to set priorities on availabilities. In these cases, the creator of the event manually determines the optimal time for an event based on these priorities
        - In other cases (for example, in When2Meet), a participant has no way of specifying that they can attend a meeting at a certain time if necessary, reducing the number of potential time ranges for meetings.
    - explanation
        - The application will allow users to specify whether they are willing attend the meeting during specific time ranges if necessary. It will only schedule a meeting at this time if there are no other available times mutually available times between the meeting participants.

        - This will allow for ease and flexibility in scheduling an event with many people.

Design essence

Concepts

List of key concepts with brief definitions

- In:
    - Purpose:
        - To find an optimal mutually available time among a group of users with minimal manual user input
    - Operational Principle:
        - After all users submit their preferences for times, the app intelligently schedules a single optimized “In”, a time range to hold the meeting during which all users are available
        - When an “In” is created, the meeting will be automatically inserted into each user’s google calendar at its corresponding time.
- Availability:
    - Purpose:
        - To allow users to accurately broadcast the most updated version of their schedules with minimal user input
    - Operational Principle
        - When a user enters preferences for a specific meeting they create or are invited to, their schedules from Google Calendar will be automatically imported to the PencilMeIn platform.
        - A user will be able to click and drag on an interface overlaid over their google calendar to specify their available and occupied time ranges (with a granularity of 30 minutes starting at either on the hour or on the half hour).
- Squeeze:
    - Purpose:
        - To schedule a meeting during suboptimal time ranges if necessary
    - Operational Principle:
        - A user indicates “if need be” time ranges for which they would rather not have a meeting, but can if necessary. If there are no overlapping time ranges that are marked completely available for all users, the algorithm will schedule a “squeeze”, a meeting that overlaps some part/all of an “if need be” range.

Security concerns

- Security Policy:

    - Google is secure
    - Only allow access with a google account.
    - Account recovery is done through google.
    - We do not store any event titles from users, only ranges that they are unavailable
        - This will limit the information that we can leak.
    - A user should not be able to see another user's availability; they should only be able to see the "In" that the application has scheduled for the whole group.
- Threat model:
    - Assumptions:
        - Google accounts are secure; attackers cannot compromise google accounts.
            - Google is known to have a very secure infrastructure.
            - There is nothing we can do to protect a user's information once an attacker has compromised their google account.
        - Attacker will not have physical access to servers:
            - If the attacker has physical access to the machine, there is very little we can do to protect against attacks such as reading off of the ram.

Vulnerabilities	**Mitigations**
User's google account can be compromised	use Passport.js, a well known library to handle sessions and authentication. We also use google's authentication libraries to talk to google, so that we are even more protected
User's availability or general preferences could be stolen	A user has to be authenticated before they are able to see any availability or general preferences. The user is then only able to see their own data. This is enforced by ensuring that a user owns the availability or general preferences they are about to access in the database.
Another user's meeting could be modified	Only the owner of the meeting is allowed to make changes
Attacker could gain access to url to join meeting	We store the email of each invitee given a url. When the url is visited, we check to see if the visiting user had previously been invited to the

	meeting.

- Mechanism:
    - Use OAuth2 to authenticate with google using Google's nodejs npm packages.
        - Removes the need for a user to directly pass credentials to our application.

- App Permission Matrix

	Modify Meeting	See their own Availability	Sign in with google account
Meeting Creator	yes	yes	yes
Meeting Invitee	no	yes	yes
No account	no	no	no

- Attacker Identities:
    - Meeting creator
    - Meeting invitee
    - Someone who wants to compromise google accounts
    - Someone who specifically targets the meeting creator or invitee's schedule

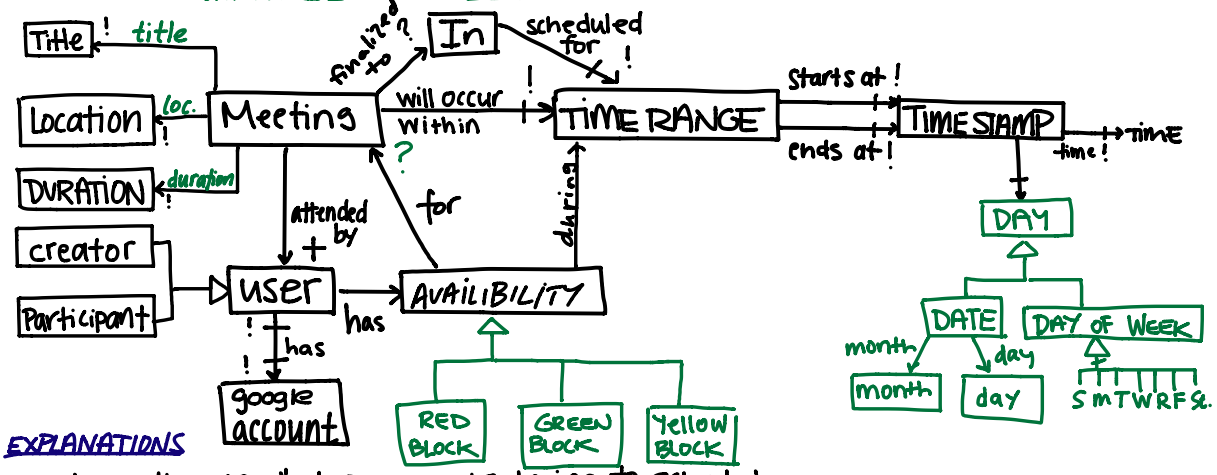
Standard web attacks:

- We're protecting against web attacks by using a standard framework (node.js with express and mongodb)
- Protect from xss by sanitizing inputs
    - The application requires very few text inputs:
        - Location
        - Meeting title
    - Use security reviewed code to sanitize input:
        - santize html npm plugin based on Google caja code that has been code reviewed and is widely used https://www.npmjs.com/package/sanitize-html
- Use CSRF tokens to stop CSRF attacks:
    - Using csurf express module
        - https://github.com/expressjs/csurf
- Use google libraries to interface with google api to protect against OAuth replay attacks:
    - google-auth-library
    - https://developers.google.com/google-apps/calendar/quickstart/nodejs

Pencil Me In

UPDATED ☺

Data Model (UPDATES IN GREEN)



EXPLANATIONS

→ A meeting is what our app is trying to schedule

→ The meeting creator sets up the meeting with a location, duration and time window in which the meeting can actually occur.

→ The algorithm looks at information for users involved in this meeting and schedules it for a single "In"

→ An "In" is an optimally scheduled time range for a meeting. We chose to make an In its own object to highlight the fact that it was optimally scheduled.

→ Users can set up Availibilities for each meeting, which consists of time ranges they want to exclude/re-include for the scheduling algorithm

→ Users log in with their google account which imports their google calendar information which implicitly tells the algorithm which time windows are free/taken before being possibly manually overwritten

→ A meeting has only one permanent creator who sets that meetings TITLE, DURATION and LOCATION.

→ users have both general and meeting specific Availibilities Meeting Specific Availibilities combine google calendar info w/ meeting specific submission info. General Availibilities are not meeting specific and are for general days of the week (and free/busy times) v.s. meeting specific Availibilities which are date specific

→ GREEN BLOCKS are free for meetings, RED are not, yellow are neutral + only used FOR AN IN IF NO completely free block is shared.

→ Time ranges of blocks have a minimum time granularity of 30 minutes.

→ MEETING SPECIFIC AVAILIBILITIES OVERRIDE MANUALLY SELECTED Meeting Specific availibilities.

INSIGHTS

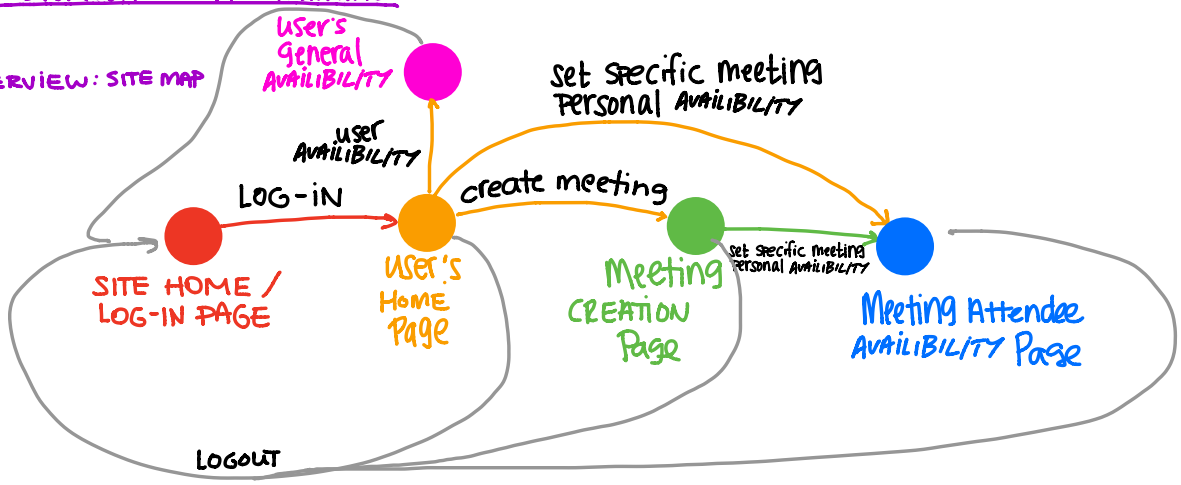
→ If there is no avaible overlapping time that everyone can attend, our algorithm will schedule the In so the most people can attend.

→ If a user changes their google calendar before the In is scheduled, our app will reimport their calendar information again right before it schedules the In via the google calendar API refresh token. Any manually set Availibilities will override the refreshed calendar.

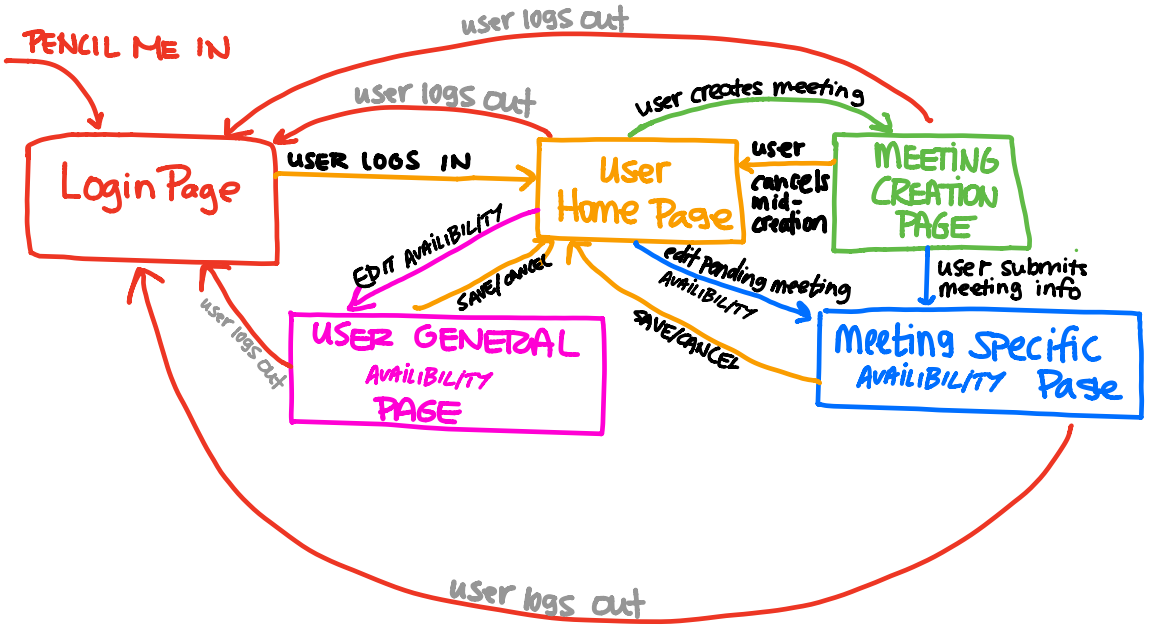
→ Before the In is finalized, users can go back and view/change their existing preferences.

UI: Overview + Wireframing

OVERVIEW: SITE MAP



OVERVIEW: MEDIUM DETAIL



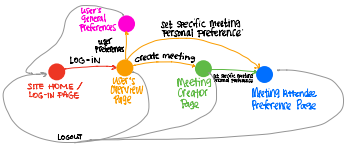
NOTE ON ERROR HANDLING:

SINCE USERS HAVE A VERY LIMITED WAY TO INPUT INFORMATION TO OUR SITE, WE WILL CHANGE THE FORM FIELDS AS USERS INPUT INFORMATION TO PREVENT THEM FROM INPUTTING ILLEGAL THINGS (LIKE ENTERING A LATEST END TIME BEFORE AN EARLIEST START TIME, THOSE TIMES WON'T BE CLICKABLE), OR TRYING TO SELECT OR ENTER INVALID DATA WILL DO NOTHING AND SHOW AN ERROR MESSAGE VIA AJAX (FOR EXAMPLE, YOU CAN'T INVITE SOMEONE WITH A NON VALID GMAIL ACCOUNT.

ALSO, GOOGLE ACCOUNT VALIDATION WILL TAKE CARE OF FAILED LOGINS AND OUR SITE DOESN'T NEED TO. WE WILL ALSO MAKE A CUSTOMIZED 404 PAGE IF A USER TRIES TO GO TO A INVALID OR OFF LIMITS URL THEY AREN'T LOGGED IN TO GO.

PAGE DETAILED WIREFRAMES

SITE HOME/LOG-IN PAGE



Welcome to Pencil Me In,
We make scheduling meetings easy, as it should be.

All you need is a google account

[g+ Sign in with Google]

Don't have a google account?

[g+ make Google acct.]

→ USERS MUST LOGIN WITH A GOOGLE ACCOUNT.

User's Overview/Home Page

Hello, Abraham!

[YOUR PREFERENCES] [Log-out]

[Schedule a new Meeting]

Your Pending Meetings

- 6.170 Team Design Meeting [Edit AVAILIBILITY]
- 6.034 STUDY GROUP QUIZ 3 [Edit AVAILIBILITY]
- Friend Workout [Edit AVAILIBILITY]

Recently Scheduled

- 6.806 STUDY GROUP QUIZ 2 Tue, Nov 17 3PM

→ REDIRECTS USER TO THEIR GENERAL AVAILIBILITY PAGE

→ Logs user out + redirects to LogIn Page

→ SENDS USER TO MEETING CREATION PAGE

→ EACH REDIRECTS USER TO MEETING AVAILIBILITY PAGE SPECIFIC TO MEETING BUTTON IS NEXT TO

→ THIS IS JUST FOR AVAILIBILITY, SO A USER CAN SEE ALL OF THEIR UPCOMING MEETINGS OUR APP SCHEDULED FOR THEM

(EACH SCHDULED IN WILL SEND USERS AN EMAIL W/ THE INFO + PUT THE EVENT IN THEIR CALENDAR FOR THEM TOO)

Meeting Creation Page

Create a meeting!

[HOME] [Log-out]

Meeting Title : [ ]

Location : [ ]

Time Range

Earliest start : [11/8/15] [03:00PM]

Latest End : [11/14/15] [04:00PM]

Invitees

[@gmail.com] [Invite]

caroline@gmail.com ✗

rcorcillo@gmail.com ✗

[SUBMIT] [CANCEL]

→ cancels + redirects to user home

→ ON CLICK, CALENDAR TOOL POPS UP

→ ON CLICK, TIME SELECTION TOOL POPS UP

→ ADDS gmail user to tentative invitee list below

→ REMOVES INVITEE FROM TENTATIVE LIST

→ ADDS FINAL INVITEES AS MEMBERS OF THE MEETING. (ALSO SENDS E ILS TO ALL)

→ REDIRECTS CREATOR TO MEETING SPECIFIC AVAILIBILITY PAGE

USER GENERAL AVAILIBILITY PAGE

Your General AVAILIBILITY

HOME | Log-out

Earliest start Time: 09:00 AM

Latest End Time: 04:00 PM

[POTENTIALLY ADDITIONAL SETTINGS]

SUBMIT

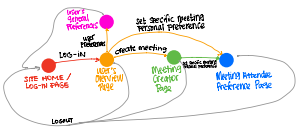
CANCEL

ON CLICK, TIME SELECTION TOOL POPS UP

12 AM
⋮
12 PM

SAVES USER AVAILIBILITY AND REDIRECTS TO USER HOME PAGE

Cancels + redirects to user home



MEETING AVAILIBILITY PAGE

Scheduling AVAILIBILITY FOR 6.170 Team Design Meeting

BETWEEN Nov. 8 10am — Nov. 14 10pm

HOME | Log-out

10am	10 – 11 6.034lec		10 – 11 6.034lec		10 – 11 6.034lec
11am	11 – 12:30p 6.858lec		11 – 12:30p 6.858lec		
12pm		12p – 1p 6.034rec		12p – 1p 6.170rec	
1pm		1p – 2:30p CMS.309lec		1p – 2:30p CMS.309lec	
2pm					
3pm	2:30p – 4p 6.170lec		2:30p – 4p 6.170lec		
4pm					
5pm					
6pm					
7pm					
8pm					
9pm	9p – 10p 858 lecture quiz	9p – 10p 6.170 lecture quiz	9p – 10p 858 lecture quiz	9p – 10p 6.170 lecture quiz	

SUBMIT

CANCEL

REDIRECTS TO USER HOME PAGE + SAVES AVAILIBILITY

NEW BACKEND ALGORITHM RUNS TO SCHEDULE MEETING WHEN THE LAST INVITEE TO SUBMIT DOES SO

CLICKING ON A TIME TOGGLES IT FROM GREEN TO RED OR VICE VERSA

CLICK+SHIFT+SECOND CLICK SELECTS THE SPAN OF TIME GIVEN IT IS A SPAN IN A SINGLE DAY.

REDIRECTS TO HOME PAGE + DOESN'T SAVE

Challenges

Design challenges

List of problems to resolve in concepts, data model or user interface
For each problem: options available, evaluation, which chosen

1. Some people don't put all of their commitments in their google calendar calendar, or don't necessarily need to attend all events that are in their calendars. In order for our scheduling application to work, we need an accurate representation of the events users plan to/need to attend so that it has the correct information when finding mutually available times.
    a. Options:
        i. Don't use google calendars at all and have users fill out all of their commitments in directly in our app.
            1. Evaluation: Although they would enter commitments knowing that the meeting will be directly scheduled around what the put in, this is a lot of unnecessary work since we assume that at least a majority of a person's regular commitments will be in their google calendars.
        ii. Import a person's google calendar and use it directly, disregarding possible misfits.
            1. Evaluation: In the worst case (where someone doesn't use their calendar accurately), this will make it very hard to schedule meetings where everyone can attend, whether because people have commitments our algorithm didn't know about and they aren't really available for the meeting it schedules, or because there are no known overlapping free times and the scheduled the meeting excludes people who could have actually attended if their calendars knew which commitments weren't absolutely mandatory.
        iii. Import a person's google calendars with an overlaying editable availability layer. The import automatically puts green blocks where there is free time in the google calendar and red blocks over commitments in the calendar, allowing users to manually change the color of blocks if their current color doesn't reflect reality. Red signifies hard commitments, green signifies completely free time, and yellow signifies a suboptimal time that can be used for a meeting if no completely green overlapping times are available.
            1. Evaluation: This option allows users to do the least amount of work to accurately represent their schedules, and even allow them to indicate a sense of importance that can be used by the algorithm to schedule the meeting in a way that everyone can attend if possible.
    b. Final decision:

        i. Option iii, because it is the best of both worlds. Again, it is the easiest way for users to accurately represent their availabilities without having to repeat work they've already done to make their google calendars.

2. If one user takes much longer to fill out their form than everyone else, then the availability data for the rest of the group might be stale- they might have scheduled new events and we wouldn't know about it.
    a. Options:
        i. Ignore the new data- this is the way that doodle works currently, it does not update due to new constraints.
            1. Evaluation: Although this lacks a nice feature, it is the standard for our market so users will not expect our app to contain this feature.
        ii. Keep listening to user's calendars and update the meeting based on new constraints.
            1. Evaluation: It might be difficult to keep the connection to the google APIs open for such a long period of time such that we can keep listening to new calendar events that a user has.
    b. Final decision:
        i. We are choosing to ignore the new data for the initial MVP. This is because we think that there are significant technological risks involved in implementing the other option. This is a feature that we might choose to implement later on.
3. If there is no single time that all users can meet:
    a. Options:
        i. We can throw away the meeting and tell users that there are no viable times.
            1. Evaluation: This seems inconvenient because we are not giving our users any information that can help them to make a decision.
        ii. Choose the In so that the host and the maximum number of invitees can attend, then warn everyone that not all invited can attend.
            1. Evaluation: Seems like a better way of dealing with this problem because at least we give our users some information about the largest subset of people that can attend their meeting.
        iii. Let the creator choose which invitees they care the most about and schedule an in just for that subset.
            1. Evaluation: We want to help the user make decisions with a small amount of user interaction, but this particular feature would require a lot of user interaction with little payoff.
    b. Final Decision:
        i. We chose option ii because it allows us to give users the most information while requiring the least user input.
4. How to let users show their preference of times:
    a. Options:

i. Users can click on a block and set their preference for that block by indicating. Preferences have to be manually set for each block.
    1. Evaluation: We can reduce this issue by having a default preference for blocks.
ii. Don't let users show their preference of times:
    1. Evaluation: This is an important feature that allows users to select an optimal time. It seems very limiting to not include this feature.

b. Final decision:
    i. We chose option i because we believe that it is important to allow users to set preferences and we believe this is an easy feature to implement.

5. Granularity of availability
    a. Options:
        i. Every minute
            1. Evaluation: This will make the algorithm for determining meeting times slower with minimal value added to the user. Most users do not schedule their meetings and plans by the minute so this level of granularity is unnecessary
        ii. 15 minutes
            1. Evaluation: Similarly, many users do not schedule their days in 15 minute intervals, so for the purposes of our application, this level of granularity is also unnecessary
        iii. 30 minutes
            1. Evaluation: This choice makes our app more compatible with Google Calendar, which currently schedules events in thirty minute time blocks.
    b. Final decision:
        i. Option iii. We chose this option because it provides the granularity necessary to schedule most meetings

6. How to manage constraints on Squeeze
    a. Options
        i. Set no priority levels
            1. Evaluation: This will increase the likelihood that there will not be a common meeting time for all participants in the meeting
        ii. Set more than three priority levels
            1. Evaluation: Too many priority levels will make the user interface unnecessarily confusing
        iii. Limit to 3 priority levels: 1 (available), 2 (neutral), 3 (not available)
            1. Evaluation: Does not introduce significant algorithmic complexity and allows for flexibility to setting priorities
    b. Final Decision

        i. Option iii: We made this decision because it gives users the flexibility of setting priorities on time blocks without providing an excessive number of options
7. How to ensure data is up to date.:
    a. Options
        i. Pull the latest Google Calendar events right before scheduling/running the algorithm.
            1. Evaluation: Ensures that we only perform computations on the most up to date data from Google
        ii. Pull new google calendar data every certain number of minutes
            1. Evaluation: Reduces computational overhead of running the algorithm, but increases the probability that the algorithm is run on stale data
    b. Final Decision
        i. Final Decision
            1. Option i: We made this decision because it is more important to us to ensure that calendars are up to date, because just one user's stale calendar could greatly affect the final result of the algorithm.

8. How to express general preferences:
    a. Options
        i. Soft General Preferences: Users can set the general time of day they would prefer to have group meetings (e.g. I like to have my meetings around 10am during the week and around 1pm on weekends). The algorithm would try to weigh these preferences to maximize happiness.
            1. Evaluation: Allows users to have some say on when meetings are scheduled without preventing a meeting to be by saying they have hard commitments.
        ii. Hard General Preferences: Users can specify concrete time ranges on certain days of the week they always can or always can't have meetings. E.g No meetings before 11AM on a Saturday or after 5PM on Tuesdays.
            1. Evaluation: Allows users to specify hard time ranges for which they are not willing to attend meetings. However, it does not afford the flexibility for users to set more nuanced preferences.
        iii. Mix using 3 priority levels: Users set time ranges during certain days of the week to specify whether they have hard commitments (aka are completely available/not available) or have commitments which they can miss if necessary to schedule the meeting. Any general preference are overridden by meeting specific preferences.
            1. Evaluation: Gives users flexibility to express both hard and soft preferences that will be automatically imported with their google calendar for every meeting.
    b. Final Decision

i. Option iii: Mix using 3 priority levels
    1. We chose this because it is consistent with how meeting specific availabilities are structured and is the best balance between saving users time, giving users availability to express day specific scheduling preferences, and ease of implementation.

Data design choices and their justifications

- Time range for a meeting is immutable
    - If it weren't immutable, the creator would be able to change the original time range for the meeting such that there is no overlap between the new and original time ranges. If so, all participants will have to fill in their time preferences again.
- A user must have a google account
    - Our application relies on using google calendar, which a user only has if they have a google account