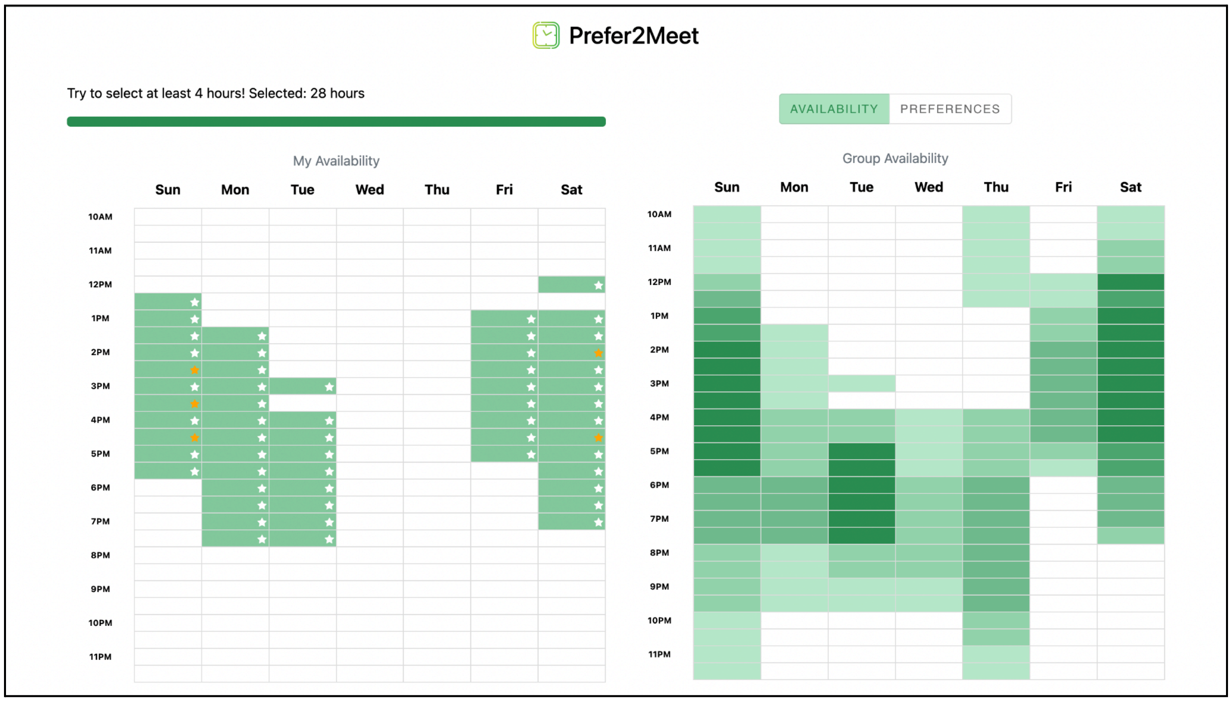
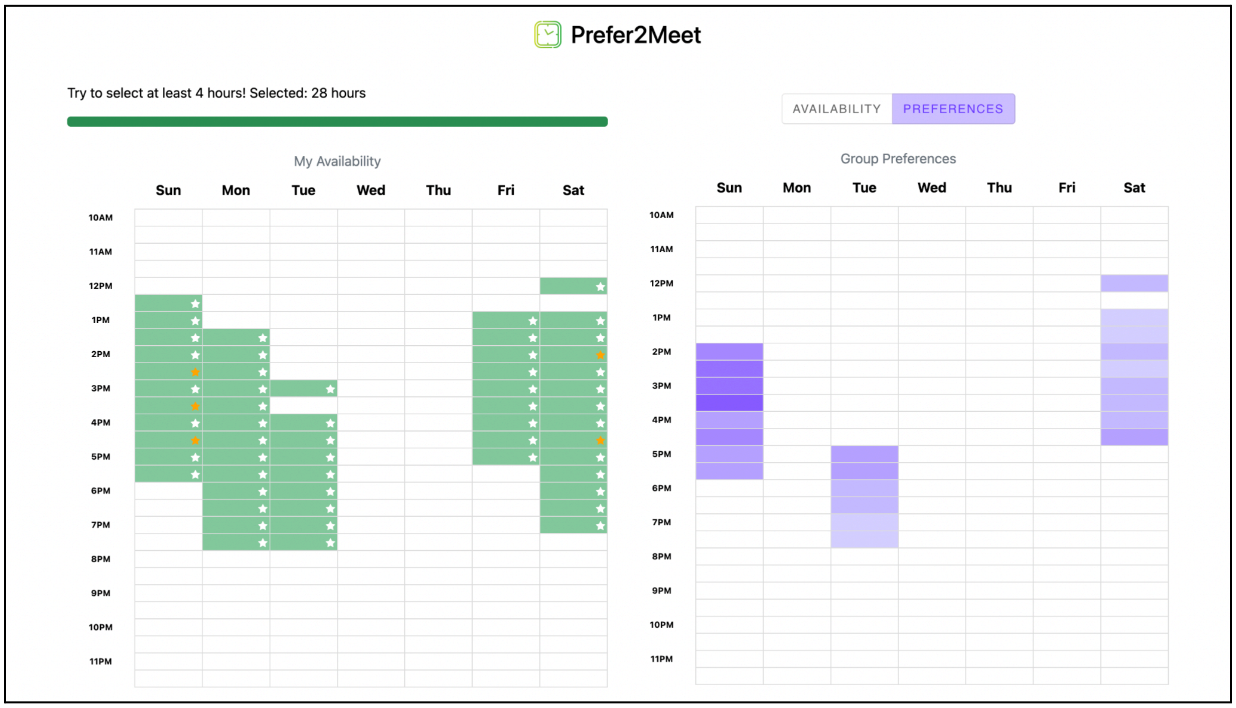
Chris Barber, Jessica Li, and Sylvia Cruz-Albrecht

Input Pair Project Reflection

1. Overview

Tutorial: [demo link] | Code: [github repo] | Figma: [figma link]





2. Concepts

For our Input Project, we created Prefer2Meet – an enhanced, more user-friendly version of When2Meet that incorporates two new concepts in order to help users find a consensus more easily and better cater to everyone's schedules. These two concepts are *minimum availability* and *preferences*.

Minimum availability encourages users to input at least a certain number of hours of availability – set by the event creator. In our interface, we dynamically show users how far they are from the goal through a progress bar. Our motivation is to nudge users to be amenable and input more availability, increasing the probability of finding a common meeting time. By not making this a strict requirement, we aim to avoid cases where users are actually unavailable for a set amount of time.

Preferences allows users to mark blocks of their availability as preferred by starring specific time blocks. We incorporate user preferences into the UI through the 'Group Preferences' view on the right calendar. Here, only time blocks where all users are available are colored and the shade is determined by the proportion of users who have marked that time block as preferred. This is informed by the scenario where users have several times of overlap in availability and it is unclear how to choose between tied time blocks.

The Doodle reading discussed how people's mental model of availability is not binary (available/not available). As a result, individuals often mark less of their availability to protect their time and optimize their preferences – to varying degrees ranging on the individualistic tendency of cultures – which can make it harder to find times that work for everyone. Our motivation was to make it easier to find times that work for *and* are preferred by everyone. By both encouraging users to provide more availability and allowing users to express finer-grained preferences that matter in the selection of the final time, these new concepts work together to enable users to find the optimal meeting time.

3. UI modifications

Our UI modifications include a de-cluttered interface with larger input tables than When2Meet and a fresh color scheme for the Preferences display. We took inspiration from Schej, which is essentially a nicer version of When2Meet with Google Calendar integration.

4. Svelte Framework

We used Svelte to implement our interface because of its reactivity and component-based nature. Reactivity allowed us to provide immediate UI rendering to the user as they dragged their availability. It also allowed us to access the current user's input alongside other users' data (hard-coded) in order to reflect in both the user and group views in real time. In addition, store variables made it seamless to share data between our components.

The Components concept of Svelte was a good fit for our app, which relies on three tables: “Availability,” “Group Availability,” and “Group Preferences.” We developed a reusable parent <Table /> component and used props, such as title={“Availability”}, and conditional logic, such as {#if isWriteable}, to adapt this table to each distinct view.

However, one of the major drawbacks with Svelte is its smaller developer community, compared to other JavaScript frameworks like React. We expected there to be more open source projects using Svelte that we could learn from or use in the project – instead, we ended up building most of our app from scratch. It also made for a more challenging debugging experience.

5. Playtesting

Playtesting was highly rewarding and offered insightful perspectives on the concepts that we tried to implement. Broadly speaking, the feedback helped us better understand our end user and their user scenarios, as well as challenging our assumptions about our app’s usability. For instance, we learned from the playtesting that it is not immediately obvious that the starred time blocks indicate preferred times. After these results, we think adding more communication on the app – through tooltips, instructions, and/or a legend – would improve its user experience.

We found that our concept of preferences did not immediately align with our playtesters and would benefit from clarification that it is not everyone’s preferences but rather times when everyone is available and colored by preferences. Some scenarios left our users confused, such as when the preferences tab went blank after users filled out availability (due to no overlap with our fake data). The scenario we envisioned was for the preferences view to aid in resolving ties – but as one of our playtesters pointed out, it may be unrealistic for everyone to converge on a time if groups are large. With more time, we would definitely modify our preferences tab to accommodate more of the overlap times, such as a slider that allows you to choose what threshold of overlap you want to display.

6. Attribution

We used some open source components from Svelte MUI like the toggle between ‘Group Availability’ and ‘Group Preferences’. We also got some of the code for mouse dragging from a Stack Overflow post.

As a team, we worked together to conceptualize our project and then split up the programming tasks. Sylvia implemented the minimum availability counter and progress bar, logic to make only ‘My Availability’ editable, and a lot of the UI as well as coming up with the logo and name. Jessica implemented the data structure for holding available and preferred time blocks, fake data, the toggle, the logic for the group availability/preferences views, and some UI improvements. Chris implemented the table, dragging to select cells on the calendar, the star/preferred buttons, and sharing data between components. And we all worked on the writeup!