

# An Alternative Approach to Scheduling: Time Slot Preference Ranking

---

**Erin Vuong ('22)**

evuong@princeton.edu

**Adviser: Jérémie Lumbroso**

lumbroso@cs.princeton.edu

*This paper represents my own work in accordance with University regulations.*

- Erin Vuong

## Abstract

*This paper details the design, development, and evaluation of Docketer (<https://docketer.herokuapp.com/>), an automated scheduling application that prioritizes scheduling preferences of the meeting host through time slot preference ranking. Existing scheduling applications fail to consider or communicate to guests who are scheduling host preferences, resulting in inconvenience and time fragmentation for the host. Docketer offers convenient external calendar integration and scheduling through one-time scheduling links, and prioritizes host preferences through time slot ranking and a redesigned scheduling UI. Evaluation using basic usability guidelines indicate that Docketer will be easy for real users to understand and use.*

## 1. Introduction

Professor Lumbroso proposed the idea for this project to me because he was frustrated with how allowing people to view his full availability and select the time that worked best for them when scheduling a meeting using an automated scheduling app (YouCanBookMe) had a tendency to turn

his time into “Swiss cheese”. Large blocks of free time became filled with “holes”, or unrelated meetings, as people scheduled here and there. Dr. Lumbroso, like myself and likely many others, requires some time to switch between tasks and ramp up to work. Having a meeting, then an hour of free time to work, and then perhaps another meeting instead of two meetings and then two hours of free time decreased his productivity. He also mentioned that he preferred completing meetings in the mornings if possible. However, without this even being communicated to guests through the scheduling software, much less enforced somehow, as one might expect his preferences were usually not respected. The goal of my project was to prototype a mobile web app that prioritizes the host’s scheduling preferences while also maintaining the convenience of automated scheduling requiring no back-and-forth.

A variety of scheduling software already exists to help people arrange meetings (YouCanBookMe, Calendly, WhenIsGood, etc.). Many of these options involve sending the guest a link to the host’s available time slots and letting them select the option that works best for them. While this is an easy and convenient way to quickly schedule meetings without back-and-forth, not all available time slots are created equal. Some people prefer to front load their day or week. Some people prefer to avoid switching back and forth between different tasks and meetings. Current scheduling solutions do not take these preferences into account, as explained in detail in [Section 2](#).

My application seeks to maintain the convenience of existing scheduling solutions by using the same schedule-by-link process, while redesigning the web page used for scheduling to incentivize picking slots preferred by the host. The redesigned scheduling UI ranks the preferability of available time slots according to parameters outlined in [Section 3.1](#), and then when a guest is scheduling presents the time slots one by one, from the host’s most preferred to least preferred, instead of presenting them all at once as existing scheduling software do. By making it slightly more inconvenient to pick a time that is more inconvenient for the host (a guest must click through more availability options), guests are encouraged to pick a meeting time more convenient for the host.

During my project I created a three-tiered web application named Docketer that enables users to

- If a host, login and import their availability from Google Calendar and optionally change their scheduling preference parameters from Section 3.1.
- If a host, to schedule with a guest select some options (valid days for scheduling, the start and end of the workday, meeting length, etc.) and generate a one-time scheduling link which can then be emailed to a guest to use for scheduling.
- If a guest, use a one-time scheduling link to pick a preferable time slot for the meeting, enter name and email address, and automatically schedule the meeting.
- For both guests and hosts, receive confirmation emails including the date, time, and instructions for further coordination, and for the host (and guest if using a Google email address) have the event automatically added to their Google calendar.

The web application was built mainly in Python and Javascript using Flask and React, and is deployed to Heroku from a GitHub repository. It uses a Heroku PostgreSQL database to store user and application data, in addition to accessing clients' external Google Calendars to fetch availability (see Sections 4.1 and 4.2 for details).

An additional goal I had during this project was to practice development best practices I had encountered previously in an industry environment or were mentioned in COS 333 but I had not had an opportunity to put to implement so far. This included developing an internal automated regression testing suite in Python (described in Section 5.1) and writing detailed documentation both of the frontend for users and of the backend for hypothetical future maintainers or developers (detailed in Section 4.4).

While evaluation with test users was not possible during the short time frame of this project due to the lengthy IRB approval process, the regression testing suite I developed provided assurance of correctness for my application's components. Additionally, I performed heuristic evaluation [51], a form of expert evaluation of the usability of UI/UX, in order to attempt to evaluate usability without test users. I found that my application successfully meets each of the ten metrics presented as a part of heuristic evaluation including error prevention, help and documentation, minimal design, and recognition rather than recall.

## 1.1. Definitions

In this subsection I will briefly define some terminology that will be helpful throughout the rest of this paper.

A *scheduling software* is piece of software (web app, downloadable application, etc.) designed to automate or otherwise expedite the scheduling process. A *scheduling solution* (*solution*) is the approach used by a scheduling software to automate or expedite the scheduling process (e.g. employing AI or using one-time scheduling links).

For any event or meeting that will need to be scheduled, there is one *host* and one or more *guests*. The words “host” and “guest” may have certain connotations pertaining to the relative levels of the two parties, and indeed in practice the host will usually be the person arranging the meeting or event, or perhaps an important individual who many people would like to schedule meetings with, but for the sake of clarity from the scheduling software’s perspective, such considerations are excluded from my definitions. I will define the *host* to be merely the direct client of the scheduling software software being used who triggers the scheduling process (e.g. generates a scheduling link), and a *guest* to be a party who participates in the scheduling process triggered by a host in order to schedule or RSVP to an event or meeting.

## 2. Market Research

Since the goal of my IW was to create an app to be used for personal scheduling, related work would be products already on the market for (personal) scheduling solutions. As such, I performed extensive market research on existing scheduling apps to ensure that my app will bring something new to the existing market, as well as to survey existing features and solutions that might have informed my design decisions. This section outlines the market research process, as well as summarizes the results and my conclusions.

## 2.1. Approach

My market research focused on a set of existing applications used for scheduling meetings and events (listed in Figure 1). I chose these applications by searching for scheduling applications on the internet, and also by considering ones I or acquaintances had used or interacted with in the past. The intent of choosing the applications this way was to emulate how a user might find and choose a scheduling app to fit their own needs, and also consider scheduling apps that are widely used enough that I, a user, was familiar with them.

1. *Calendly*
2. *YouCanBookMe*
3. *Wase*
4. *Meetingbird (now Front Scheduling)* (Front)
5. *Woven*
6. *Acuity*
7. *x.ai*
8. *Outlook Calendar/Microsoft Bookings* (Outlook)
9. *Google Calendar* (Google)
10. *StrawPoll*
11. *WhenIsGood*
12. *Rally*
13. *GoodTime*
14. *Greenhouse*

**Figure 1: List of apps considered during market research.**

The apps I considered represent a variety of types of scheduling solutions for various needs, including meeting scheduling and group event scheduling, and are also targeted at a variety of types of users, such as individuals and businesses small and large. While my IW is intended to create a personal scheduler with maximum benefit to individual users as opposed to organizations, it was

still useful to consider solutions to different yet related needs.

For each app I considered, I first visited their webpage and made notes about how they were marketing themselves, including which features they emphasized and which audience their webpage seemed to target. I also researched their pricing schemes. Then, I created an account if possible, and looked through the features available to me. Finally, I tested scheduling a meeting or event (from the perspective of both or all meeting attendees) and made notes about the experience. I also created several tags and labeled each app using them to more easily categorize them. These results are shown in Table 1.

## **2.2. Results**

While researching, two clear categories of scheduling software emerged. The first category includes apps aimed at helping users schedule meetings easily and automatically, often in conjunction with managing their calendar. All but StrawPoll, WhenIsGood, and Rally fall into this category (“Calender/Meeting organizer” in Table 1). This generally involves synching with one or more external calendars to quickly determine availability and allowing guests to quickly schedule a meeting at a time the host is available using a convenient link. This also updates the host’s external calendars automatically, automating and streamlining calendar management. Many also included their own calendar interface so users could view events easily in calendar format within the app itself, and supported email reminders to both scheduling parties.

Within this category, some subcategories also emerged. Greenhouse and GoodTime are services aimed exclusively at (large) businesses and intended to help improve the recruiting/onboarding process. Part of this includes scheduling interviews. They are priced for businesses. The remaining apps were also marketed to businesses to varying degrees (excluding Wase which is a service internal to Princeton), but also can be used by individuals, either for free or at a small monthly cost (Table 1: “Heavily targeted at businesses” and “Can use for personal scheduling”). x.ai and Acuity were marketed explicitly as automated personal assistants that used AI to easily schedule meetings (Table 1: “AI/Personal Assistant”).

		Scheduling Apps													
Tags		Calendly	YouCanBookMe	Wase	Front	Woven	Acuity	x.ai	Outlook	Google	StrawPoll	WhenIsGood	Rally	GoodTime	Greenhouse
	AI/Personal Assistant						✓	✓							
	Guest books with host	✓	✓	✓		✓	✓	✓		✓				✓	
	Host picks time from availability										✓	✓	✓	✓	✓
	Calendar/Meeting organizer	✓	✓	✓	✓	✓	✓	✓	✓	✓				✓	✓
	Scheduling group activity/poll	✓				✓	✓	✓*			✓	✓	✓		
	Can cause fragmentation	✓	✓			✓	✓	✓							
	Heavily targeted at businesses	✓	✓		✓		✓	✓	✓					✓	✓
	Can use for personal scheduling <sup>†</sup>	✓				✓		✓						✓	

**Table 1: Summary of common features and characteristics.**

For more detailed description of each tag, see Appendix Section A.1.

\*Assuming all parties scheduling have an account with x.ai.

†Can be summarized as has a free tier and allows for easy scheduling with people who do not have an account.

The second category includes apps intended to help groups of people decide on the best time for a group meeting or event. Some of the apps previously mentioned offer this functionality, but usually only in their paid versions. Generally this includes creating a poll on a set of available times which can be sent to participants via a link. The creator can then use the results to decide a time, but must schedule the meeting using some means external to the app used for polling. The three apps in this category (StrawPoll, WhenIsGood, and Rally) were all free and did not even require an account to use. As one might expect though, the features were significantly more limited than apps in the previous category.

Most of the existing scheduling apps offer different tiers of service at increasing prices. Higher tiers offered higher customizability including removing the software’s logo, integration with multiple external calendars, and video conferencing integration [12, 2, 76]. Seven of the apps offer a free account option (Rally is also free but does not support personal accounts), and nine target businesses explicitly in the language on their pricing pages (apps not marked N/A for “Per Head?” in Table 2) by referencing “price per head” or the maximum number of employees allowed under a certain pricing plan. For a full transcript of the market research notes on all apps, please see Appendix

App	Price (per month)	Per Head?‡	Notes
Calendly	Free, \$8, \$12	Y	Free tier offers lots of functionality
YouCanBookMe	\$10	Y	
Wase	N/A	N/A	Internal to Princeton
Front	\$19, \$49, \$99	N	Businesses only (price for entire Front software), max 10, 50, any users
Woven	Free, \$15/\$20	N/A	
Acuity	\$15/\$14, \$25/\$23, \$50/\$45	N	Max 1, 6, 36 users, custom pricing above that
x.ai	Free, \$10/\$8, \$15/\$12	Y	Instant scheduling if everyone has account
Outlook	\$5, \$12.50, \$20	Y	Annual commitment, price for Microsoft 365
Google	\$6, \$12, \$18, custom	Y	Google Workspace pricing
StrawPoll	Free	N/A	No account required to use
WhenIsGood	Free, \$20 per year	N/A	No account required to use
Rally	Free	N/A	Doesn't support logins
GoodTime	Free, starting at \$999	N	Businesses mainly (for recruiting)
Greenhouse	?	?	No prices listed, businesses only

**Table 2: Pricing details.** Pricing tiers are separated by commas, prices separated by / are *price billed monthly/priced billed annually*.

‡For business pricing, is the monthly price per employee?

## Section A.

### 2.3. Conclusions

During my research I was surprised by the similarities between scheduling solutions used by the different apps. The approaches I came across fell into three distinct types, polling for choosing availability with a large group, sending a link with your availability and letting people pick a slot, and asking a person for their ability through a link so you can choose a time that works for you (this type did not seem to be available in any of the free applications though). The first falls somewhat outside the main interest of my project so I will exclude such apps from the discussion from here, but was interesting nonetheless. The second and third approaches were solutions to the same scheduling issue my app will be mainly focused on: personal appointment scheduling and calendar management. Overall, all the apps used one or two of these solutions (mainly the second mentioned) with little variation in how the scheduling process worked, from creating a scheduling link to entering contact information. x.ai's approach to scheduling meetings - using AI to parse email messages and automatically schedule meetings - was the most innovative, but still essentially very similar to existing solutions.



I was also surprised by the extent to which most of the apps I tested (excluding the three polling ones) were targeted at businesses. They were either entirely or almost entirely targeted at businesses, even if it was implied that they could be used by individuals (essentially a single employee business) as well. This makes a lot of sense because a business is a much larger client than a single individual, and working with businesses can be very lucrative either through pay-per-head pricing schemes or even a large contract with a very large business that includes training and analytics.

I think that the uniformity of scheduling solutions among the non-polling apps and their efforts to target business clients are heavily related. Many possible business clients intend to use the scheduling software to help their own clients (guests) more easily schedule appointments. In this consumer-facing scheduling situation, where the business client (host) would generally be expecting to fill their day with appointments and likely are less worried about avoiding “Swiss cheese”-like calendars, or at least where streamlining the consumer’s experience outweighs personal convenience, it makes sense to use an approach that both manages calendars and automates scheduling, while also making the consumer’s experience most efficient. The preeminent approach to this type of scheduling, in my opinion, is the current most ubiquitous approach: sending a link to all availabilities from which a consumer can pick a time that works best for them and the meeting is scheduled automatically. The business generally need not do anything except check their calendar and prepare for appointments, while consumers can easily pick the available appointment that fits perfectly into their schedule, as well as cancel or reschedule with little to no back-and-forth. The apps using this approach even offer extra features that make scheduling more streamlined for the business, like requiring some advanced notice for scheduling (not offering availability too soon), or automatically creating buffers in between meetings.

Neither of the two existing approaches to meeting scheduling consider host scheduling preferences or fragmentation beyond meeting buffers and manually determined unavailable or available times, or even communicate them to guests. Therefore, an app that is focused specifically on automated *personal* appointment scheduling - scheduling for hosts who have more nuanced scheduling preferences - instead of hoping to attract businesses and serve their external scheduling needs, will

bring a new solution to automated scheduling to the market. In particular, no existing approach to scheduling meetings is remotely similar to the particular solution my project attempted, and more broadly no service I found is marketed heavily for helping individuals manage their schedule and make the most of their time by avoiding fragmentation.

One concern may be that by focusing on helping individuals schedule their time I am precluding myself from reaching the same business clients that existing apps are taking advantage of to be profitable. While my solution will likely be less desirable for business-external scheduling with consumers, I think that it would still be able to target businesses by focusing on helping employees schedule internal meetings with their colleagues in the middle of a busy project-based work day, and by targeting the same scheduling market that Greenhouse and GoodTime do: helping companies improve their recruiting by seamlessly scheduling interviews. The recruiters I personally worked with this year generally asked me for my availability before scheduling a meeting that worked best for my interviewer, who was generally not themselves in HR. It makes sense that in the case where I am interviewing with an interviewer who is not interviewing all day and whose time is valuable elsewhere, allowing me to pick from a swath of their availabilities would not be the best approach. This means that in addition to helping employees schedule meetings with their colleagues, I could also help employees schedule interviews into their busy workday according to their preferences. This may not work well for larger companies without some specializations though, as GoodTime and Greenhouse are both successful because they are highly specialized at all parts of the recruiting process, including interviewer training and candidate management.

### **3. Approach**

Existing scheduling apps minimize the back-and-forth communication generally required when scheduling a meeting by email or some other means because they provide a framework for exchanging the necessary information in as few steps and concisely as possible. While one may, following scheduling best practices, schedule a meeting in as few as three emails (host sends the initial invitation and outline your available times, the guest responds with a time or times that work

for them, host sends a final confirmation email), in practice humans are imperfect and rarely follow the most efficient or practical scheduling flow. Often one will offer several possible meeting times, then find none of them work and have to send one or more additional emails offering alternatives. On the other hand, as outlined in Section 2, the common scheduling flow enforced by scheduling apps involving a scheduling link and automatic scheduling reduces this to one email from the host, and abstracts away all further resolving availability and sending confirmations (and even reminders).

My scheduling application, much like existing applications, seeks to expedite the scheduling process by providing an information-exchange framework through a scheduling link and automatic scheduling. I use a single-use scheduling link, which is an intermediate compromise between manually booking on one's calendar or by email and making part or all of one's calendar fully available for direct booking, such as with Calendly and similar apps. Fully available means that the scheduling link being used is reusable, and anyone with the link can reuse the link with no limits to book at any available time. The single-use scheduling link offers similar convenience as fully available links (automatic scheduling without back-and-forth) without exposing hosts to the possibility of unwanted scheduling.

Existing scheduling apps take a binary approach to the host's availability: each block of the host's time is either available or unavailable. In reality, things are not really this simple: some available times may be more preferable or convenient than others to the host for a variety of reasons. Availability and preference are different metrics. As explained in Section 2.3, existing scheduling solutions do not really consider or enforce preferences over available times, something a host might be able to do themselves if they were scheduling by email ("While I am available all day tomorrow, I would prefer to meet in the morning if possible..."). Existing scheduling solutions let the guest choose the time that is most convenient for them, with no regard for the host's preferences. Even worse, the guest could unwittingly pick the host's most inconvenient time slot, when booking the host's most convenient slot might have been equally convenient for the guest.

The prototype I built does something none of the existing scheduling app do; not only does Docketer consider host preferences, it also dynamically tracks meetings that are scheduled and uses

this information to organize available time slots in order to minimize fragmentation caused by future scheduling. Then, it nudges guests towards a preferred behavior (picking a time slot that the host prefers and does not cause fragmentation) while still offering the host's full availability, thereby providing the same essential function as existing applications. To do this I redesigned the common scheduling page used by existing scheduling software to introduce friction to the scheduling process. I present available times to the guest sequentially instead of all at once, from most preferable to the host to least preferable. Additionally, I disallow returning backwards to slots a guest has passed on (they must go to the end and circle back, or if they are clever refresh the page and start from the beginning), thereby increasing the cost of passing on an availability that works for a guest. By doing so, I hoped to funnel guests towards accepting one of the earliest times they are shown. I incentivize picking preferable or convenient time slots, or perhaps more precisely I disincentivize picking inconvenient time slots. As Professor Lumbroso put it, if a guest is going to pick a time that is inconvenient for me, I will inconvenience them a little now while they are scheduling by forcing them to click through all the more convenient options.

Of course, determining a host's scheduling preferences with enough granularity to rank all possible time slots is a non-trivial task. The naive approach might be to just ask the host to rank all possible time slots according to their preferences, as this is guaranteed for obvious reasons to produce a correct ranking. However, scheduling apps including my own seek to make scheduling as painless and easy as possible by automating everything it can from determining availability using an integrated calendar (as opposed to asking the user to enter available or unavailable times) to sending confirmations and reminders. As one can imagine, asking a user to perform a task as tedious as ranking all possible time slots would have ultimately rendered my app completely unusable. Instead, my app attempts to estimate user preferences enough to provide improved preference awareness relative to existing scheduling apps. It does so by giving each possible available time slot a priority according to several parameters and then ranking the time slots according to those priorities. The rest of this section will provide a detailed description of the parameters and priority function, as well as the reasoning behind why the specific parameters were chosen.

### **3.1. Background: Preferences and Productivity**

In order to determine factors that might play a role in scheduling preferences and can be used as parameters when determining priorities, I researched factors of productivity. Preferences are evidently personal and cannot be generalized into a couple of parameters. However, as I only sought to estimate user preferences, it seemed reasonable to assume that productivity might play a significant role in when people prefer to schedule meetings and how they prefer to manage their time. It seemed promising, then, to look into research done on productivity as a roundabout way of determining possible factors of scheduling preferences.

There appeared to be three factors of productivity that were also relevant to the issue my project sought to address: time of day and week, and interruptions.

**3.1.1. Time of Day and Week** Research suggests that cognitive abilities increase in the morning and peak in the early afternoon before falling off rapidly throughout the rest of the day [22]. We are also generally happier in the morning, and mood similarly increases over the morning before peaking around noon [8]. These results seem to suggest that organizing meetings and handling more strenuous mental tasks earlier in the day can result in better performance. It seems reasonable to assume that a similar effect might be in play on the weekly scale. Anecdotally, everyone seems to dislike Mondays, and are likely less productive when ramping up after the weekend. Likewise, after Wednesday productivity might be expected to decrease as people anticipate the coming weekend. Some people are morning people, while others prefer to work later in the day. Some people may prefer to front load their week, while others prefer working later or even over the weekend.

**3.1.2. Interruptions** Interruptions and their effect on productivity have been studied widely in cognitive psychology, in particular as part of distraction theory. An interruption is defined as an “externally-generated, randomly occurring, discrete event that breaks continuity of cognitive focus on a primary task”. Interruptions generally reduce productivity because the interrupted party may need “a recovery period of up to 15 minutes” to re-concentrate on the primary task [63, p.4]. The recovery period is necessary because interruptions require task reconfiguration, or “the rearranging of physical or cognitive resources” needed to switch between tasks, which is “unanimously... a time

sink” [31]. Interestingly, research indicates that recent technological advances intended to increase productivity have often had the opposite effect as a result of interruptions. In particular, Rennecker and Godwin observe that in addition to interruption there is a second type of disruption to workflow, delay, where “a worker is unable to move forward with a task due to insufficient information”. Communication technology innovations have generally been targeted at decreasing communication delays, but “inadvertently contribute to an increase in work interruptions” [58, p.247] which in turn decrease productivity.

Meetings, especially when scheduled with little input from oneself aside from having no other meetings presently scheduled at that time, are essentially interruptions, even if they can be predicted well ahead of time. Put another way, meetings - and all other interruptions - cause fragmentation of working time, which is detrimental to productivity. Suppose a worker has several meetings that need to be scheduled in one day. For simplicity also assume that in addition to the meetings the worker has one large task to complete that will take up the remaining time of the workday. If the meetings are scheduled back to back (with whatever small break would be required to gather one’s thoughts and refresh in preparation for the next meeting), then the remaining time in the day is one or two large contiguous blocks. The worker will only face task reconfiguration once before each meeting, and once when returning to the original task. By contrast, if the meetings are spaced out during the day, task reconfiguration will be required both before and after each meeting. Or perhaps from an even simpler perspective: scheduling interruptions together and allowing the rest of the time to be a large contiguous block allows oneself more control over how to utilize the remaining time, and can be especially useful when one is faced with larger tasks, or when one prefers to focus for extended periods of time. Essentially, this is the crux of original inspiration for my project. We want to avoid “Swiss cheese”.

Interruptions clearly affect productivity, and while some people may be able to “hyper-refocus” [31] between tasks and therefore be less bothered by interruptions, in general I think it is fair to assume that 1) interruptions and their effect on productivity influence how people prefer to schedule their time, and 2) while some people may be more or less bothered by interruptions, in general

avoiding them is preferable for everyone.

### 3.2. Parameters of Preference

Given the previous context, I selected three parameters of scheduling preference that I thought would be sufficient to estimate an individual's scheduling preferences and rank available time slots: time of day, time of week, and interruption. Preferences are evidently very complex even for a single individual and vary from day to day. My goal was not to capture an individual's exact preferences, as stated before, but merely to estimate them to the extent required to improve on existing scheduling solutions. Consequently, these parameters are neither exhaustive nor exact.

**Time of day** ( $D$ ) can be understood as how preferred a time slot is relative to the user's scheduling preferences in terms of morning, afternoon, or evening. Quantitatively, this is measured as the distance between the start of the meeting ( $s$ ) and an optimal scheduling hour ( $s_o$ ).

$$D = |s_o - s|.$$

The optimal scheduling hour can be understood as the time at which the user would schedule a meeting or task to start given they only have one meeting or task to complete that day. Note that this does not take into account how long the task to be completed is, but is merely intended to get an idea of how much of a "morning" or "afternoon" person the user is. All other factors being constant, we expect a more preferred time slot will minimize this value.

Similarly, **time of week** ( $W$ ) can be understood as how preferred the time slot's day of the week is. Quantitatively, this is measured as the distance between the time slot's day of the week ( $d$ ) and an optimal day of the week ( $d_o$ ), which is the day a user would schedule an activity given they only have one activity to schedule in the week. All other factors being constant, we expect a more preferred time slot will minimize this value.

$$W = \min\{|d - d_o|, D_w - |d - d_o|\},$$

where  $D_w = 7$  is the number of days in a week. Note that  $d, d_0 \in \{0, 1, \dots, 6\}$ , and the term  $D_w - |d - d_0|$  is introduced to cover cases such as  $d = 6, d_0 = 0$ .

**Interruption** ( $I$ ) measures how much a given time slot breaks up an existing block of free time. Quantitatively, this will be calculated as the minimum of the time between the end of the last event ( $e_\ell$ ) and the start of the time slot ( $s$ ), and the end of the time slot ( $e$ ) and the beginning of the next meeting ( $s_n$ ).

$$I = \min\{s - e_\ell, s_n - e\}.$$

By definition  $s$  is always later than  $e_\ell$  and  $s_n$  is always later than  $e$ , so no absolute value is required. If there is no event before then the end time is taken to be the start of the workday (by default 9am), and likewise if there is no event after the time slot then the start time is taken to be the end of the workday (by default 5pm). Note that, we assume that all individuals prefer to minimize interruptions as much as possible. All other factors being constant, we expect a more preferred time slot will minimize this value.

### 3.3. Priority and Ranking Algorithm

The algorithm for ranking time slots is very simple. For each time slot I calculate a priority value ( $P$ ) using the following formula and the parameters above:

$$P = a \cdot D + b \cdot W + c \cdot I,$$

where  $a, b$ , and  $c$  are constants used to normalize the weight of the various parameters to approximately one. Then, I merely rank all the time slots in increasing order of priority.

## 4. Implementation

In this section I provide an overview of the implementation of my web app, including a description of technologies used, a system overview, use cases, and a summary of key steps and problems faced during implementation. I also describe other deliverables developed as part of the implementation



process.

#### **4.1. Technologies**

During the implementation of my project I created a three-tiered networked web application mainly intended for use on a computer. The app consists of a frontend tier, or the user interface (UI) that the client interacts with directly when they visit my website, a backend tier that handles the business logic of my application and serves client requests, and the database tier that stores persistent application data including user information.

The backend tier was implemented in Python using Flask [53] and Jinja2 [54]. I also used SQL Alchemy [6], a object relational mapper, for database management, as the information I needed to store (user profiles and scheduling link information) fit well into Python objects. The frontend tier was built in Javascript using React [56], with Bootstrap [57] and custom CSS for styling, and minimal HTML. The database tier consists of a Heroku PostgreSQL database [44] that holds user profiles and scheduling link info. My application also accesses clients' external Google Calendars to check availability using Auth0 for OAuth2 authentication [15] and the Google Calendar API [32]. The application was deployed to Heroku [43] from a private GitHub repository [33].

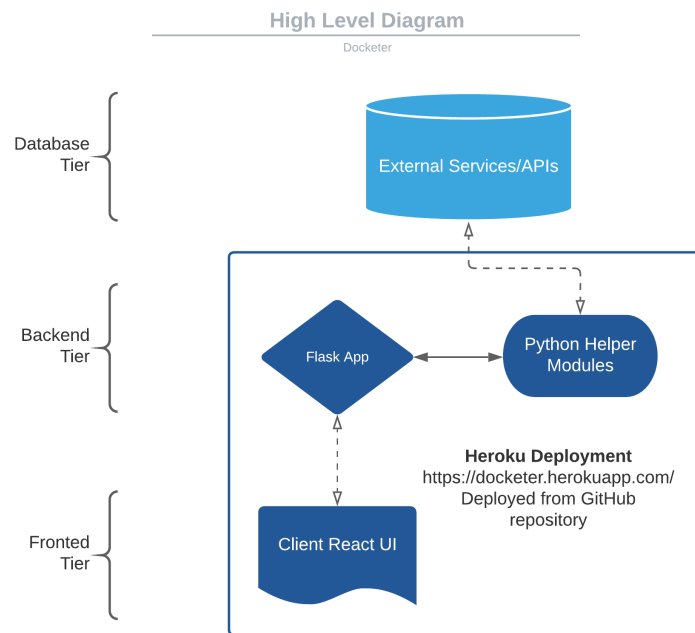
During COS 333 last semester, I worked on a three-tiered networked mobile web application using many of the same technologies I used in this project [69]. As a result, I was familiar with many of the technologies mentioned above, especially Flask, deploying to Heroku, and using a SQL database, and I was able to get started on this project quickly, as needed for this single-semester project.

However, I also worked with several new technologies, namely OAuth2 authentication through Auth0, the Google Calendar API, and most importantly React. React has a notoriously steep learning curve, and while it was introduced during COS 333 we were discouraged from using it on our final project as a result. I had previously performed updates on an existing React app during an internship, but had never built an entire React app with complex functionality. Through referencing many online resources I was able to teach myself these technologies while completing

my project [5, 32, 20]. I opted to use React in spite of possible difficulties because it is a very common framework for building user interfaces, and supports intuitive and modular UI design. The complexities that make React difficult to learn also make it a powerful tool for designing interfaces.

## 4.2. System Overview

As mentioned previously, Docketer’s implementation consists of three tiers: a database tier, a backend tier, and a frontend tier. Figure 1 provides a high-level visualization of which components compose each tier. Note that the blue outlined block indicates the code deployed to Heroku via a hosted GitHub repository.



**Figure 1: A high-level diagram of the application.** Made with Lucidchart [46].

As mentioned before, the frontend code was written in Javascript using React. I used `create-react-app` [16], a utility that quickly generates all the code and files needed for a simple single-page React application. This auto-generated code is located in the `/react-app` folder seen in Figure 2. Then, I altered the default configurations in order to integrate with and serve from a simple Flask application [50]. `/flask-server` contains the backend business logic as well as the Flask app that serves the frontend. As the GitHub repository consists of two main folders, one for frontend

development and one containing the Flask application, it was necessary to deploy a subtree of the directory (only the `/flask-server` subdirectory) to Heroku [86]. Updates to the React were made on Javascript files in `/react-app/src`. Then, the code was rebuilt using the `npm run build` command, which generated a simple HTML template, some CSS files, and minified Javascript files required to render the UI. These generated files are located in `/flask-server/static` and `/flask-server/templates`, and could be deployed to Heroku along with the Flask app.

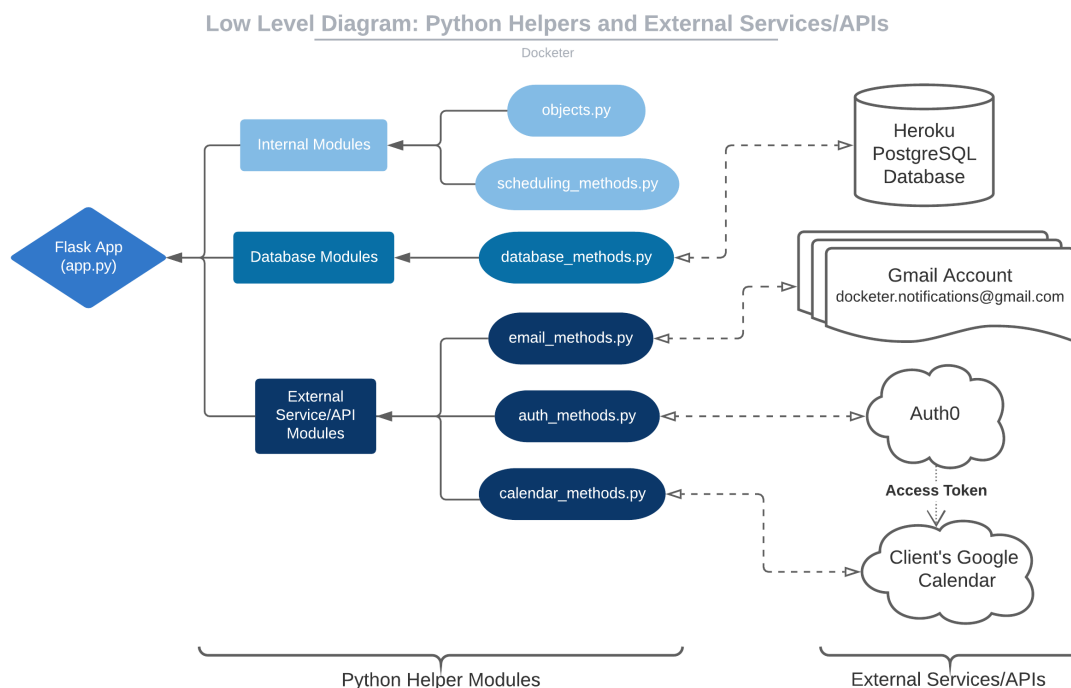
```
|-- /docketer
    |-- README.md
    |-- /flask-server                # BACKEND FUNCTIONALITY (deployed)
    |   |-- app.py                  # the main flask server app
    |   |
    |   |-- /test                   # TESTS (frontend/backend)
    |   |
    |   |                           # Built by React
    |   |-- /static
    |   |-- /templates
    |
    |-- /react-app                  # FRONT END DEVELOPMENT
        |-- /src                   # REACT JS FILES
            |-- /pages              # Pages
            |-- /components         # Other Components
```

**Figure 2: Project directory structure.** Abridged version of the detailed directory structure provided in the project README [67].

Figure 3 provides a low-level view of the Python files that make up the backend tier, as well as the external services and APIs that each file communicates with. To increase the modularity of the code, I opted to organize helper methods being used in the Flask app roughly by the functionality they were providing, which also aligned closely with which external service they were communicating with.

As the diagram indicates, the business logic files can be broken up into three categories: internal modules, database modules, and external API modules. The internal modules define Python objects used as common definitions throughout other files, in particular as table structures for the SQL

Alchemy found in the database modules. They also handle things like parsing availability data and generating time slots, as well as ranking and sorting the available time slot objects based on host preferences. The database module defines a set of common interactions with the database using SQL Alchemy; this includes things like adding a user, updating preferences, adding a link to a user, or deleting a link. The external API modules handle authentication with Auth0, emailing clients, and fetching information from Google Calendar.



**Figure 3: A low-level diagram of Python modules and the external services or APIs they use or communicate with. Made with Lucidchart [46].**

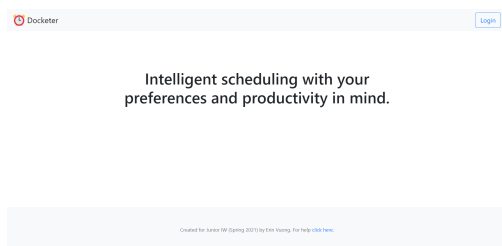
Auth0 is an external platform that abstracts logins and authentication for applications. When hosts login through my app they are redirected to a login page provided by Auth0. Once they have logged in, they are returned to Docketer through a callback URL. Auth0 attaches a profile variable to the Flask session containing some basic information including the user's id, name, and email address. Docketer only allows login using a Google account as it needs to be able to access a client's Google Calendar. However, logins through other services or even just with an application-specific username and password are also possible using Auth0 [5].

I opted to use Auth0 not only because it abstracts app logins, but also because it assists with

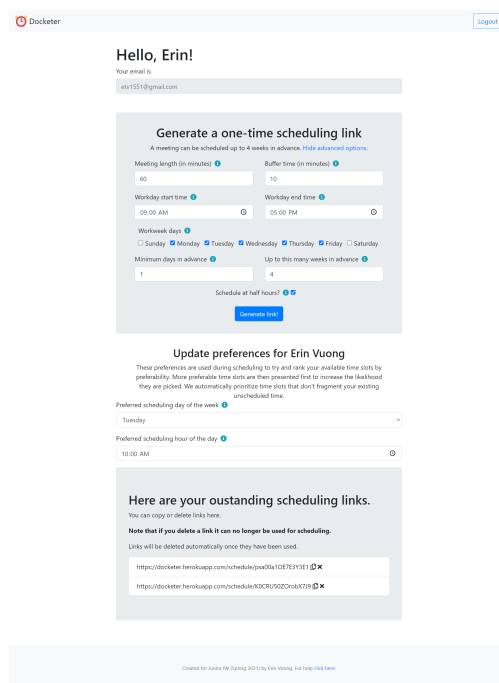
authentication for external Identity Service Provider APIs like the Google Calendar API. The Google Calendar API provides a helpful Python API for interacting with Google Calendars, including authenticating request from external services (my app), fetching availability, and adding events to a client's calendar. Requests to the Google Calendar API are authorized using access tokens and refresh tokens. Access tokens are short-term tokens that expire quickly. Refresh tokens are long-term tokens that may be used to request newer access tokens when past access tokens expire. Each token has a set of allowed scopes associated with it. Docketer requires both read and write permissions for host calendars in order to read available time and then add scheduled meetings. In Docketer, a host logs in at some point to generate a scheduling link, but requests to Google actually occur when the guest attempts to use the scheduling link at a later date, as we want to give the guest the host's most up-to-date availability. At this later date, any access token generated at the host's latest login may have already expired. For this reason, my application does not use access tokens but instead stores in the SQL database and uses refresh tokens to authorize Google Calendar API requests.

After a host logs in using Auth0, Docketer requests more private client information from an Auth0 Management API, which may include Google Calendar access and refresh tokens [3, 4]. The backend checks if a new refresh token has been provided by Auth0 and updates it as necessary; this occurs if the user is asked to re-approve required permission scopes during the login process. Docketer also checks that the refresh token on file still works by querying the host's calendar and checking that no error is returned. Refresh tokens should expire very rarely, but in the case that the token is invalid, the host will be alerted of the issue and asked to navigate to a specific link to fix the issue. This link forces the permissions approval prompt, which will ensure that when the user is logged back in the session profile provided by Auth0 contains a new valid refresh key. Checking the refresh token each time a host logs in is intended to minimize the chance an error will occur when a guest attempts to schedule. However, in the case that a host's refresh token is invalid when a guest attempts to schedule, they will see a specific error page telling them to wait until the problem has been fixed, and the host will receive an email with the same link described above to fix the issue.

The React frontend is comprised of over 20 smaller React components that fetch information from the Flask server where required and update dynamically as the user takes actions or makes changes. The UI is made up of three main pages, two for hosts and one for guests. For hosts, there is a simple homepage when logged out (Figure 4), and a page with several possible actions when logged in (Figure 5).



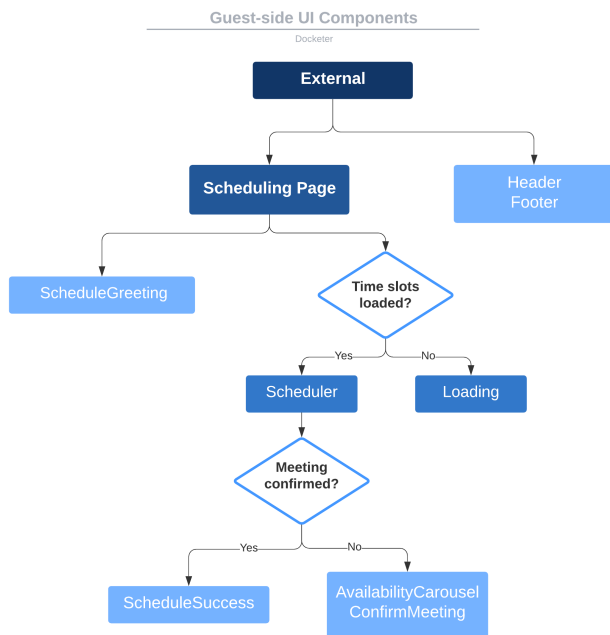
**Figure 4: Host view when logged out.**



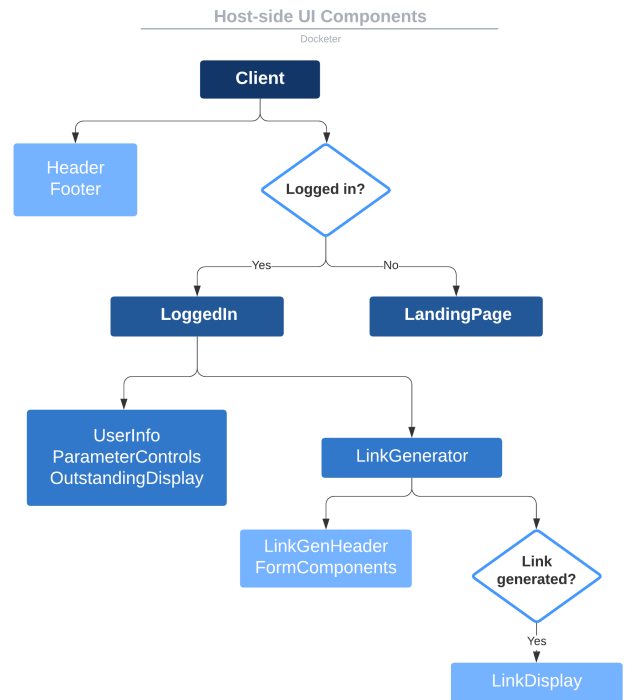
**Figure 5: Host view when logged in.**

Each section of the logged in host page (generally delineated by the gray jumbotrons) is its own smaller component, and each of those components are comprised of more smaller components, some of which are shared between several components (for example the information icons which are hoverable tooltips providing users with extra information on forms). Figure 7 provides a breakdown of the important components that make up each host-side page.

Guests interact with the scheduling page, seen in Figure 8. The component that renders the possible time slots takes a list of JSON objects, each representing a possible scheduling time. It renders one of the slots at a time in the gray button, starting from the beginning of the list, and increments the index rendered each time the guest selects the red button at the bottom. Once the

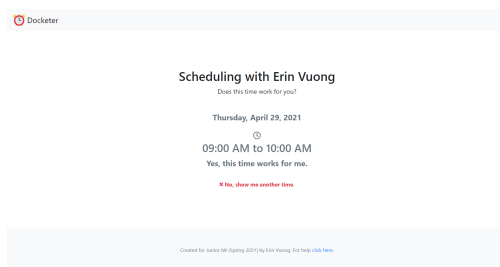


**Figure 6: A low-level diagram of the main React components that make up the guest-side UI. Made with Lucidchart [46].**

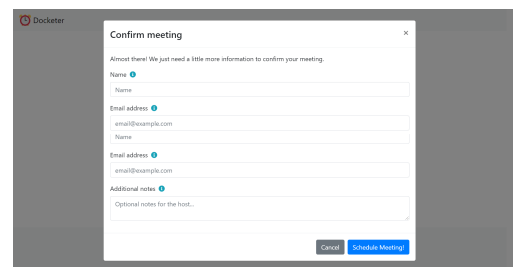


**Figure 7: A low-level diagram of the main React components that make up the host-side UI. Made with Lucidchart [46].**

guest reaches the end, they will have the option to return to the beginning, and the index will reset to zero. After selecting a time that works, guests will see the confirmation form shown in Figure 9. Name and email address are required fields because without them, the host will not know who has scheduled, and the guest will not be able to receive meeting details in an email. The required inputs are validated using React Bootstrap’s native form validation. The name must be a non-empty string, and the email must pass a default simple email regular expression. Figure 6 provides an overview of the important components that make up the scheduling page.

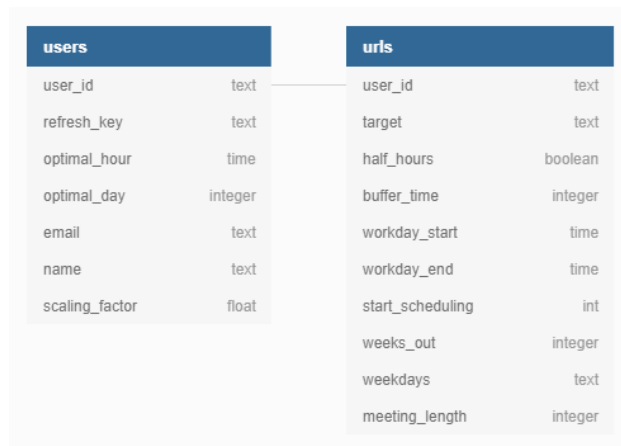


**Figure 8: Guest view of the scheduling page.**



**Figure 9: Meeting confirmation form.**

Figure 10 is a diagram of the layout of the Postgres SQL database. The database consists of two tables, a `users` table and a `urls` table. My app lent itself nicely to using three simple types of objects: `TimeSlots`, `Users`, and `SchedulingLinks` (URLs). All three are defined in `objects.py`. All three of the objects instance variables that store certain characteristics, methods to get those characteristics, and a method for returning those characteristics in a dictionary so they can easily be dumped to JSON and passed to the frontend. `TimeSlot` objects are used when generating and sorting time slots to make the code more readable and do not need to be stored. `Users` and `SchedulingLinks` however needed to be persistent. When users log in, we would like to be able to store (and later update) some information about them such as their name, email, and preferences. Likewise, when a scheduling link is generated we need to at minimum remember which host the given link is associated with, so when a guest tries to schedule we can schedule a meeting with the appropriate host. Figure 10 shows the characteristics that are stored for each user and scheduling link as fields in the corresponding table. I used SQL Alchemy as an object-relational mapper to easily add, edit, and delete objects from the database.



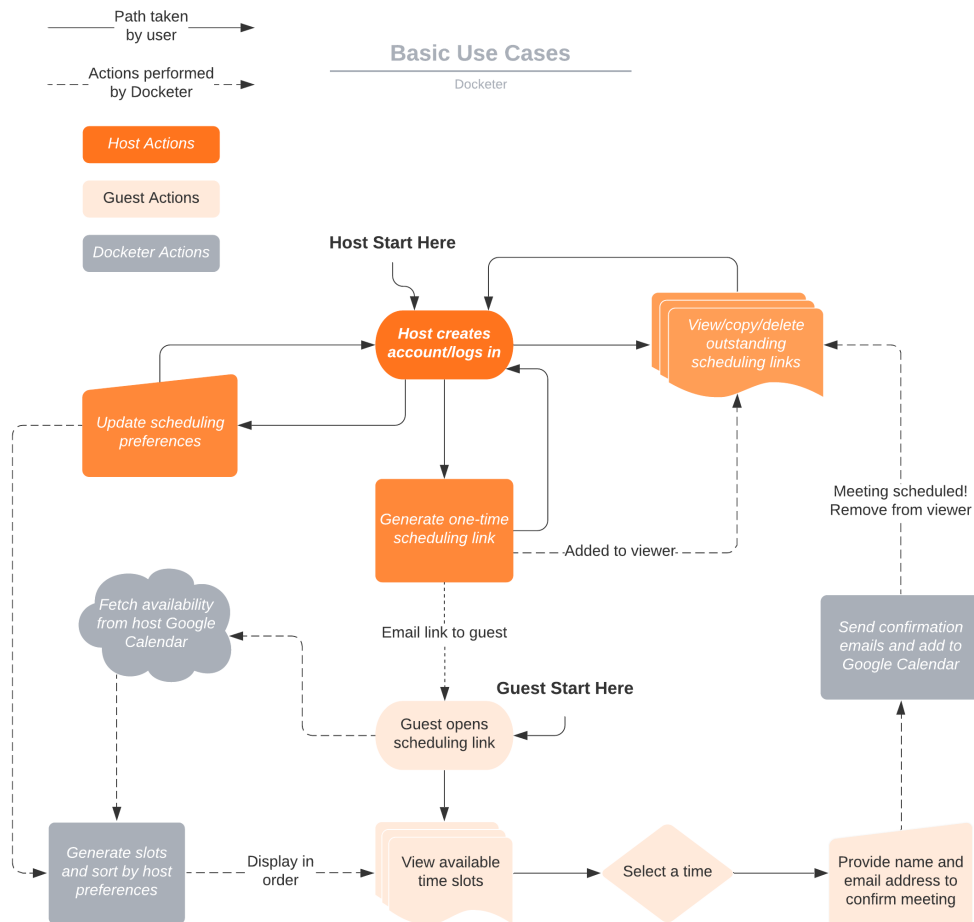
**Figure 10: A diagram of the SQL database.** Blue headers represent tables, other cells represent columns with name and type. Made with dbdiagram.io [18].

### 4.3. Use Cases

Docketer has one main function: using a one-time scheduling link to automatically schedule a meeting. This involves a host logging in and generating a link which the guest then uses to view



available times and confirm the meeting. Behind the scenes at the time of scheduling, Docketer fetches the host's availability from Google Calendar using stored access tokens, processes the available time, generates and sorts the time slots, and then displays them. Figure 11 provides a concise overview of the basic uses cases and actions possible to both users and guests, as well as what is done when on the backend.



**Figure 11: A flow diagram of the basic use cases.** Made with Lucidchart [46].

For more detailed information on use cases (including screenshots of the app) from the user's perspective, see the user guides mentioned in Section 4.4 [65, 66], as well as the video demonstration of the app [68].

## 4.4. Deliverables

The main deliverable for this project is of course the web app deployed at <https://docketer.herokuapp.com/>. Unfortunately due to the nature of Google authentication on un-verified apps, only approved test users can log in and view Docketer themselves.<sup>4</sup> To bypass this slightly I provide a video walk-through of the app at this link: <https://youtu.be/qdJrJkwTfbo>.

However, what is deployed to Heroku is only the tip of the iceberg. This project also involved developing significant amounts of code that are not deployed to Heroku (e.g. the testing suites described in Section 5), as well as documenting the project in code headers, and a README for future developers or maintainers. The actual GitHub repository used to deploy to Heroku must be kept private because it contains API client secret keys and other sensitive information. However, a sanitized version<sup>5</sup> has been made available here: <https://github.com/evuong483/docketer.git>. Note that the sanitized repository is not intended to be functional.

I created user guides for both hosts [66] and guests [65] to help with all possible functionality. The user guides provide detailed instructions of all use cases, including labeled screen shots of Docketer in various states. These are available to anyone, and can also be accessed from the README of the sanitized repository.

## 4.5. Key Steps and Problems Faced

The first step I took once I had decided on the purpose of this project was to decide what type of application would best fill the needs of my users. The intended users of my project are people who wish to have a convenient automated way to allow others to schedule meetings with them and quickly add those meetings to their existing calendar in a way that helps them maximize the productivity of their time. There were many possible types of applications that could meet the needs of my intended users and serve the intended purpose, including a (mobile) web app, a desktop application, or a native application for a phone or tablet. Given my previous experience in web

---

<sup>4</sup>Please contact me if you would like to view the app and have not already been added as a test user. Note that @princeton.edu email addresses will not work, likely due to Princeton-specific security measures.

<sup>5</sup>The actual repository can be made accessible to individuals upon request.

application development (COS 333 and an internship) I decided that it would be most realistic to build a web application in a single semester, as I already had an idea of how to get started and deploy an application. I also think that a web application is the most convenient for users, as all users who would be hoping to use such a service will have access to the internet through an internet browser, which is all that is required to access a web application. They do not need to download and store a separate application on their phone or computer. I also opted to focus on a web app intended for use on a computer that may or may not work well on a mobile device, instead of a mobile web app that also is guaranteed to work in mobile settings, in order to keep the scope of this project manageable and because mobile capabilities had no direct importance to the functionality of my application.

This three-tiered application was built using several different languages and libraries, and contains many moving parts. Initially the idea of building all the components myself and ensuring that they worked together properly in one semester seemed very daunting. Instead of tackling everything at once, I decided to first break the project up into smaller working parts, especially for parts I was unfamiliar with such as the React and working with authentication and the Google Calendar API. I created several small GitHub repositories: one for creating a simple React app, one for a Flask backend that could handle creating time slots and sorting them, and one for logging in users and making authenticated requests to Google for their available times. Then, once each part seemed to be working I started putting them together. I put the Flask and React together first, and once the Flask app was serving the UI properly went back and added the Python business logic. This was the largest and most difficult step in the implementation of my app; once the parts were combined it became much simpler to add new features and logic, partially because the framework was in place but also because I became more comfortable with the relevant technologies. Given more time it would have been better to write tests for each component before combining them, but I was anxious to make progress given the time constraints, and was lucky that I did not run into major issues on this step. Later, I was able to go back and write regression test suites for each component separately that should make future development and maintenance much easier.

## 5. Evaluation

### 5.1. Correctness

In order to ensure that all the components of my app were working as expected, I wrote a thorough automated regression testing suite with Python `unittest` tests for each component. The testing suite will also ensure that any bugs caused by future changes can be caught and dealt with easily and efficiently. The suite involves over 18 unit tests for Python backend business logic, 14 unit tests for the Flask app, and over 25 functional tests for the React frontend. In the `README` [67] I provided detailed instructions for running and updating the tests. Before each deployment to Heroku of new changes, I ensure that all tests are passing and that a quick manual walkthrough of a local deployment exposes no issues.

I was unable to write an effective test suite for authentication-based methods and routes in the Flask app, and therefore was unable to automate integration testing. Authenticating requests to Google Calendar in particular requires refresh tokens produced by Google and stored in the SQL database. These tokens are sometimes corrupted or cease to work. While I was able to determine this through manual testing and provide support for users who encounter this problem through alerts, warning emails, and instructions, it greatly complicates writing automated tests. It should be possible to create a Google account intended for testing and use Selenium webdriver to automate the login process, but this also creates more complications surrounding how availability is determined and how time slots are created. One would need to write a script to determine the current date, add certain events to the test user's calendar, and then formulate a specific request based on the current weekday, time, etc. It would also need to clear the test user's calendar after tests to allow re-running. Overall, I decided that the tests will require more maintenance and be more difficult to write than they will be helpful, especially because given the other frontend and Flask tests it is easy to verify that authentication is working by merely logging in once or twice manually. In place of automated tests for authentication I provided specific descriptions of how to manually test all uncovered backend code in the `README`.

As mentioned in section 4.4, Docketer is not Google-verified and as such can only be accessed by approved test users. Others who attempt to login will face an error, which in itself is not a bug but expected behavior. However, for some reason test users with @princeton.edu accounts may be added, but will not be able to access the app even after giving approval and granting the requisite permissions. This is likely due to security measures put in place for university accounts, although I do not know the exact reason.

## 5.2. Heuristic Evaluation

While I was not able to evaluate my app using real test users, I performed heuristic evaluation, a form of evaluation by experts. Heuristic evaluation was developed by Jakob Nielsen in 1990 and involves a set of “10 general principles for [human-computer] interaction design” intended to give UX/UI designers general guidelines for improving the usability of their applications. The ten heuristics are as follows (taken from Nielsen [51]):

- **Visibility of system status:** The design should always keep users informed about what is going on, through appropriate feedback within a reasonable amount of time.
- **Match between system and the real world:** The design should speak the users’ language. Use words, phrases, and concepts familiar to the user, rather than internal jargon. Follow real-world conventions, making information appear in a natural and logical order.
- **User control and freedom:** Users often perform actions by mistake. They need a clearly marked "emergency exit" to leave the unwanted action without having to go through an extended process.
- **Consistency and standards:** Users should not have to wonder whether different words, situations, or actions mean the same thing. Follow platform and industry conventions.
- **Error prevention:** Good error messages are important, but the best designs carefully prevent problems from occurring in the first place. Either eliminate error-prone conditions, or check for them and present users with a confirmation option before they commit to the action.
- **Recognition rather than recall:** Minimize the user’s memory load by making elements, actions, and options visible. The user should not have to remember information from one part of the

interface to another. Information required to use the design (e.g. field labels or menu items) should be visible or easily retrievable when needed.

- **Flexibility and efficiency of use:** Shortcuts — hidden from novice users — may speed up the interaction for the expert user such that the design can cater to both inexperienced and experienced users. Allow users to tailor frequent actions.
- **Aesthetic and minimalist design:** Interfaces should not contain information which is irrelevant or rarely needed. Every extra unit of information in an interface competes with the relevant units of information and diminishes their relative visibility.
- **Help users recognize, diagnose, and recover from errors:** Error messages should be expressed in plain language (no error codes), precisely indicate the problem, and constructively suggest a solution.
- **Help and documentation:** It's best if the system doesn't need any additional explanation. However, it may be necessary to provide documentation to help users understand how to complete their tasks.

For each of the metrics above I walked through my app several times and noted specific features where my app demonstrates good application of the metric, as well as places where my UI could improve. My conclusions are outlined below:

- **Visibility of system status:** Much of the information that is fetched for the user is provided quickly enough that it was not necessary to add loading messages and other indications of system status. One exception is that fetching available times, generating time slots, ranking them, and then returning them to the UI can take the Flask app a little bit of time. For this reason, when a guest is scheduling there is a spinning loading icon that runs while waiting for the time slots to load. Similarly, when a link is being generated a similar loading icon appears after the button is clicked and disappears when the link displays. Docketer allows users to copy links straight to their clipboard and even delete them by clicking on certain icons in the link generator and outstanding links display. Docketer notifies users that their copying was successful by updating the tooltip that appears above the copy icon when hovered from “Copy” to “Copied!”. Similarly

the icons to copy and delete disappear when a link is deleted, replaced by a red “Deleted” badge. Docketer could be improved by adding additional messages, such as notifying the host that they have successfully updated their scheduling preferences.

- **Match between system and the real world:** Docketer follows real-world conventions. For example, the login and logout buttons are located at the top right of the screen and labeled “Login” or “Logout” respectively, which are concepts most users will be very familiar with. Likewise, the link generator contains a “Show advanced options.” link which is a common pattern used in forms and for filtering search criteria. The vocabulary surrounding scheduling preferences (“Preferred scheduling day of the week” and “Preferred scheduling hour of the day”) may not be clear to users, so I provide tooltips with explanations. The vocabulary of Docketer could be refined so these tooltips are less necessary, but in general I use phrases that the users can likely deduce the meaning of from their existing understanding and perceptions. As a result of using the React Bootstrap library for components, the layout of my components will be familiar and intuitive to most users.
- **User control and freedom:** In general Docketer is very simple, containing one page for scheduling and two for hosts (logged in vs. logged out). However, it still provides user control and freedom. When scheduling, once a user confirms that a time works for them, a modal will open for them to enter some contact information to confirm the meeting. If the user changes their mind there are two buttons for them to close the modal, a “Cancel” button at the bottom and an “x” button at the top right, as is common. Once the meeting is confirmed, another modal with meeting details appears and once again there are two clear buttons to close the modal. One place where Docketer does not support user control is viewing previous time slots when scheduling. If a user indicates that they are not available at a certain time slot, they will be presented with another time slot. They will then have to cycle through all remaining time slots and go back to the beginning to view the passed time slot again. This is somewhat intentional, as the hope is that users will feel somewhat obligated to pick the first time slot that appears that they are available for. However, I might reconsider whether this is a good idea; it could be better to encourage users to go back

towards more preferable time slots to the host.

- **Consistency and standards:** Similar to in **Match between system and the real world**, Docketer is generally very simple and therefore naturally does not have many opportunities to be self-conflicting. Likewise, as a result of using the React Bootstrap library for components, the layout of my components will be familiar and intuitive to most users. Docketer has good internal and external consistency.
- **Error prevention:** One example of error prevention is the form validation used when scheduling. Guests must give a name and email address to confirm a meeting so that the host will know who scheduled which meeting and how to contact them. The form where guests input this information validates that the guest has provided something in each of these fields, and also that the email matches a simple regular expression for emails. This prevents the user from forgetting to fill in these fields properly and scheduling a meeting, as they will be unable to receive the meeting details email and the host may not know they have scheduled. For all other forms (link generation, host preferences), I use specific input types and provide default values so the user cannot provide improper types of values, and hopefully will find it harder to forget to fill out a field. I could improve error prevention by performing validation on the other forms as well.
- **Recognition rather than recall:** As with previous metrics, Docketer does a good job using recognition rather than recall because it has a simple interface and provides simple functionality. The host only interacts with one page while logged in. Newly generated links appear in both the outstanding links viewer and at the link generator and are easily recognizable. Similarly, the guest only interacts with one simple scheduling page and a confirmation form. After confirming the meeting, the meeting details are also reiterated in the confirmation message so the user can view them again. No recall is required.
- **Flexibility and efficiency of use:** Because all actions taken on Docketer are relatively simple, I do not provide any specific shortcuts for experienced users to speed up operations. However, I do provide one helpful customization that gives more experienced hosts flexibility when scheduling. Scheduling links are automatically generated using several default parameters (weekdays are



Monday through Friday, workday starts at 9:00 and ends at 5:00, etc.). The default link generator display shows a “Generate a one-time scheduling link” title and a “Generate link!” button. If one clicks the button directly, it generates a link with the default settings. However, an experienced user can click a “Show advanced options.” link which toggles the form to show inputs for each setting. The experienced host can update the various settings as they wish and generate a link accordingly. An improvement might be to allow the host to save different sets of settings as types of meetings and quickly auto-populate the link generator according to the desired type of meeting. I could also add more options for reminders, recurring meetings, etc.

- **Aesthetic and minimalist design:** Docketer is minimalist largely due to the fact that I was trying very hard to avoid working with complex styling and CSS while learning React. I opted to use the React Bootstrap library which provides nicely styled but very simple and minimalist components. The color scheme is simple (white/grey/blue). The guest page while scheduling contains only a title and two buttons. In the host page there are four main components, with two separated from the others by jumbotrons (a React Bootstrap component that mainly provides a gray background for the section). The outstanding link display (a display for links the host generated that have not yet been used) even disappears when there are no outstanding links. Ultimately, there was no extraneous styling. Every little thing on every page of my application serves a very specific purpose.
- **Help users recognize, diagnose, and recover from errors:** The main error that Docketer handles is dealing with corrupted Google refresh tokens. For the host, when they login the app automatically checks if their refresh token is working, and if not displays a clear red alert at the top of the host page. The alert is easily visible and red which matches traditional error messages. The alert informs the host that their “Google connection is broken” and instructs them to navigate to a link that will fix the refresh token (by logging them out and then logging them in again through a route that forces the permission approval prompt and will produce a new good refresh token). If a guest tries to schedule with a host whose refresh token is broken, they will see a simple message in plain language informing that a problem occurred and that the host has been

notified, and that they should wait for the host to fix the issue. The host will receive an email with instructions to fix the issue. All error messages give the user what went wrong without using jargon and offers a solution or instructions if a solution is not possible. Error recovery could be improved by finding more possible errors and creating error messages for them, as the main error case handled by the app is specifically the case with refresh tokens.

- **Help and documentation:** Overall Docketer is well documented. On all forms where the user is required to input or change information I added tooltips that when hovered over display information about what the given field is for. Tooltips are an excellent example of good documentation because they are extremely simple to use and are located right next to where the user would require it. They are also small and concise. I also provide more detailed user guides for both guests and hosts. The links to these guides could be made more obvious to users. The links are currently located in the footer as part of a “For help click here.” statement in relatively small font. While the page itself is simple so they are still easy to see, it might be helpful to either increase the visibility of the footer or reference user guides more explicitly in the footer and elsewhere. Within the user guides, I provide a table of contents with informative section titles and hyperlinks to the given section. This makes the documentation easy for users to search and find what they need. The guides also contain labeled screenshots and clear, concrete instructions so they should be very easy to follow.

Overall, Docketer does a good job with each of the ten usability heuristics. This is in part due to the relative simplicity of the app which automatically makes it easier for users to understand and use. However, certain features I created were particularly important for certain metrics, such as tooltips for documentation. While actual users interacting with the application will have different opinions and experiences, Nielsen’s ten heuristics for usability give me as the developer a good baseline to determine the usability of my app, especially without access to test users.

### **5.3. Original Plan: Evaluation With Test Users**

I had originally hoped to evaluate both the usability of my app and the effectiveness of my approach to automated scheduling using a moderately-sized batch of test users. It is impossible to accurately evaluate the usability of an application without asking an outside user to interact with the app and give feedback, as usability is very subjective. Similarly, my approach to scheduling relies heavily on human preferences, and as a result will require some sample of humans to evaluate its success. I had also hoped to compensate these test users somehow, and was fortunate enough to receive funding from the department to do so.

Research on human subjects has the potential to cause ethical issues, and as a result such research requires IRB approval. One of the earliest steps I took in my project was to request IRB approval for evaluation. The application involved writing a description of my project and answering various questions about privacy and consent, as well as providing a detailed description of the evaluation process. Even though I had barely begun designing my application, I was able to quickly design a detailed process for evaluating the results of my application's performance (in terms of user preferences) relative to existing solutions. The evaluation would have involved around 50 users, split into two test groups. In the scheduling phase, each user would have been sent several scheduling links and been asked to use the links to schedule meetings according to their real schedule. Half of these links would have been generated by Docketer, and half by an existing application for comparison. Then in the feedback phase users would have been presented with schedules generated during the scheduling phase and asked about their preferences and thoughts. I have included parts of the application, including a sample user survey, for reference in [Appendix B](#).

Unfortunately, my evaluation was not approved in time for me to perform it during this semester for this write up. However, if it is approved in the near future I will consider seeking separate funding to perform a user evaluation of this application and evaluate important features of automated scheduling solutions.

## 6. Summary

While analyzing existing scheduling applications as well as factors that might affect the preferability of certain meeting times, it became clear that existing scheduling applications face a convenience-fragmentation tradeoff. Existing applications prioritize convenience; they allow hosts and guests to schedule quickly with minimal back-and-forth, and they do so without asking the host to indicate specific times they are free, instead extrapolating available time from an integrated external calendar like Google Calendar. However, they do so at the cost of allowing ample fragmentation of the host's time. For some clients, such as businesses, this may not be an issue. For other clients though, existing solutions may not be ideal. In my project I sought to fill a hole in the existing scheduling software market by offering an application that tries to prioritize host preferences and avoid fragmentation during scheduling while still harnessing external calendar integration and automatic availability generation for convenience. Docketer attempts to prioritize host preferences and discourage fragmentation by ranking time slots from most preferred and least likely to cause fragmentation to least preferred and most likely to cause fragmentation. Then, by presenting slots one by one, it encourages guests to select better time slots rather than picking from all available slots.

During this project I was able to develop Docketer, test correctness through internal testing, and evaluate usability using Nielsen's heuristic evaluation framework [51]. I also developed extensive documentation and practiced writing readable code that would hopefully be able for maintainers or hypothetical colleagues to read and understand. I learned and practiced several new technologies including React, SQLAlchemy, and authentication using OAuth2, and feel that I have grown significantly as a developer.

Unfortunately, due to the limited time frame of this project and the human-research approval requirements, I was unable to perform evaluation with real test users. Test users would be required to make stronger conclusions about both the usability of my product and the effectiveness of Docketer's approach to prioritizing host preferences.

As such, future work on Docketer would likely involve performing usability tests with test users, gathering feedback, and making improvements accordingly. More generally, experiments could also be performed to evaluate the preference ranking approach's effectiveness at improving host satisfaction, especially in comparison to existing scheduling applications, such as the experiment described in Section 5.3. Through experimentation I might be able to find loopholes in the current protocol perhaps related to refreshing the page, and could consider ways to improve. For example, in the current protocol a guest might click through several options that work, then pick a later one that also works out of frustration. It might be better to continue showing some of the previous most preferred options, allowing users to pick previous preferable slots while still discouraging discovery of more (less convenient) slots [47]. I could also improve upon the existing preference ranking algorithm by using more complex models, finding new parameters, or even just optimizing the existing cost function.

One could also consider designing and evaluating other approaches to prioritizing host preferences, such as approximating the host's preferences and then indicating this visually to guests. One might imagine that guests, given an idea of the host's preferences rather than no idea at all, might voluntarily select more preferable spots, even while still picking based on their own preferences as well. It would be interesting to study if guests generally respect host preferences voluntarily, and also how effective it is compared to Docketer and other similar approaches. One could also imagine pricing slots according to their convenience to the host, where more inconvenient slots are more expensive. If such approaches prove successful, they could have major implications for the existing application scheduling market. This project would introduce an opportunity for innovation in the automated scheduling market, as the current preeminent approach may need to be rethought or improved to better meet users' needs.

## **Acknowledgements**

I would like to thank my adviser Professor Jeremie Lumbroso for giving me the idea for this project. It was extremely helpful to have a meaningful project idea that could also be appropriately scoped

for a one-semester IW. I am grateful that he agreed to be my advisor during this busy semester.

Thank you to the Accenture Independent Work Thesis Fund/SEAS for agreeing to fund my project beyond the normal funding limits.

I would like to thank all the online resources and tutorials that I relied on while completing my project: in particular React Bootstrap's component summary page [57] which was an ever-open tab on my computer, as well as Stack Overflow and Google for always being available to help me troubleshoot and answer any and every question that came to my mind.

I would like to thank sugar, and the many Princeton restaurants and shops that provided it in many forms, for fueling me this semester and helping me cope with the stress that came from working on this project.

Last but definitely not least, I would like to thank my friends and family for supporting me, listening to my ideas and frustrations, and always cheering me up. In particular, I would like to thank Danny for all of his help and support (and chocolate), and for putting up with me this semester.

## References

- [1] “Acuity scheduling: All you need to do is show up at the right time.” Available: <https://acuityscheduling.com/>
- [2] “Acuity scheduling: Pricing.” Available: <https://acuityscheduling.com/signup.php>
- [3] “Connect apps to google.” Available: <https://auth0.com/docs/connections/social/google>
- [4] “Identity provider access tokens.” Available: <https://auth0.com/docs/tokens/identity-provider-access-tokens>
- [5] L. Balmaceda, “Auth0 docs: Python quickstart.” Available: <https://auth0.com/docs/quickstart/webapp/python>
- [6] M. Bayer, “Sqlalchemy.” Available: <https://www.sqlalchemy.org/library.html>
- [7] “Bootstrap 5.0: Getting started.” Available: <https://getbootstrap.com/docs/5.0/getting-started/introduction/>
- [8] H. Brueck, “The best time of day to do everything at work, according to science,” *Business Insider*, Aug 2018. Available: <https://www.businessinsider.com/best-time-day-work-according-to-science-2018-5#so-it-might-be-best-to-schedule-important-meetings-and-earnings-calls-during-these-happier-morning-hours-4>
- [9] “Calendly.” Available: <https://calendly.com/>
- [10] “Calendly: Features.” Available: <https://calendly.com/pages/features>
- [11] “Calendly: Integrations.” Available: <https://calendly.com/pages/integrations>
- [12] “Calendly: Pricing.” Available: <https://calendly.com/pages/pricing>
- [13] “Calendly: Solutions.” Available: <https://calendly.com/pages/solutions>
- [14] “Calendly: Teams.” Available: <https://calendly.com/pages/teams>
- [15] M. Conradt, “Store and retrieve google refresh token.” Available: <https://community.auth0.com/t/store-and-retrieve-google-refresh-token/28973>
- [16] “Create react app.” Available: <https://create-react-app.dev/>
- [17] “datetime - basic date and time types.” Available: <https://docs.python.org/3/library/datetime.html>
- [18] “dbdiagram.io.” Available: <https://dbdiagram.io/home>
- [19] E. C. Dierdorff, “Time management is about more than life hacks,” *Harvard Business Review*, Jan 2020. Available: <https://hbr.org/2020/01/time-management-is-about-more-than-life-hacks>
- [20] R. Dondero, “Web Javascript Lecture 4 code handout,” COS 333 Fall 2020-2021.
- [21] “Emoji favicons: Alarm clock.” Available: <https://favicon.io/emoji-favicons/alarm-clock/>
- [22] S. Folkard, “Diurnal variation in logical reasoning,” *British Journal of Psychology*, vol. 66, no. 1, pp. 1–8, 1975. Available: <https://bpspsychub.onlinelibrary.wiley.com/doi/abs/10.1111/j.2044-8295.1975.tb01433.x>
- [23] “Fontawesome: copy.” Available: <https://fontawesome.com/icons/copy?style=regular>
- [24] “Sending emails with python.” Available: <https://realpython.com/python-send-email/>
- [25] “Fontawesome: info-circle.” Available: <https://fontawesome.com/icons/info-circle>
- [26] “Fontawesome: times.” Available: <https://fontawesome.com/icons/times>
- [27] “Fontawesome: undo.” Available: <https://fontawesome.com/icons/undo>
- [28] “Front scheduling: Why front.” Available: <https://frontapp.com/why-front>
- [29] “Front scheduling: Customer experience.” Available: <https://frontapp.com/solutions/customer-experience>
- [30] “Front scheduling: Pricing.” Available: <https://frontapp.com/pricing>
- [31] J. E. Gaskin and T. Skousen, “Time-chunking and hyper-refocusing in a digitally-enabled workplace: Six forms of knowledge workers,” *Frontiers in Psychology*, vol. 7, p. 1627, 2016. Available: <https://www.frontiersin.org/article/10.3389/fpsyg.2016.01627>
- [32] “Google calendar api.” Available: <https://developers.google.com/calendar>
- [33] “Github.” Available: <https://github.com/>
- [34] “Goodtime.” Available: <https://www.goodtime.io/>
- [35] “2021’s top automated interview scheduler | goodtime.” Available: <https://www.goodtime.io/interview-scheduling>
- [36] “Goodtime meet | meeting scheduler right from a link.” Available: <https://www.goodtime.io/meet>
- [37] “Pricing | goodtime.” Available: <https://www.goodtime.io/pricing>
- [38] “Use google calendar appointment slots.” Available: <https://support.google.com/calendar/answer/190998?co=GENIE.Platform%3DDesktop&hl=en>
- [39] “Google workspace pricing.” Available: <https://workspace.google.com/pricing.html>
- [40] “Applicant tracking system & recruiting software | greenhouse.” Available: <https://www.greenhouse.io/>
- [41] “Greenhouse product pricing.” Available: <https://www.greenhouse.io/pricing>
- [42] “Greenhouse recruiting software.” Available: <https://www.greenhouse.io/recruiting>
- [43] “Heroku.” Available: <https://www.heroku.com/home>
- [44] “Heroku postgres.” Available: <https://www.heroku.com/postgres>

- [45] “ISO 8601- effectively communicate dates and times internationally.” Available: <https://www.ionos.com/digitalguide/websites/web-development/iso-8601/#:~:text=According%20to%20the%20basic%20format,%3A%2012%3A07%3A22>
- [46] “Lucidchart.” Available: <https://www.lucidchart.com/pages/landing>
- [47] J. Lumbroso, “Slack correspondance,” 4 2021.
- [48] “Microsoft bookings want to help customers schedule meetings.” Available: <https://www.businessnewsdaily.com/10459-microsoft-bookings-wants-to-help-customers-schedule-meetings.html#does-microsoft-bookings-integrate-with-outlook>
- [49] “Microsoft bookings.” Available: <https://www.microsoft.com/en-us/microsoft-365/business/scheduling-and-booking-app>
- [50] nakatuddesuzan, “How to serve a react-app with a flask-server,” *Learning Dollars*, 2019. Available: <https://blog.learningdollars.com/2019/11/29/how-to-serve-a-reactapp-with-a-flask-server/>
- [51] J. Nielsen, “10 usability heuristics for user interface design.” Available: <https://www.nngroup.com/articles/ten-usability-heuristics/>
- [52] N. Patel, “How ‘flipping the funnel’ helped calendly hit double digit user growth for 24 months straight.” Available: <https://neilpatel.com/blog/flipping-the-funnel/>
- [53] T. P. Projects, “Flask.” Available: <https://palletsprojects.com/p/flask/>
- [54] T. P. Projects, “Jinja.” Available: <https://palletsprojects.com/p/jinja/>
- [55] “Rallly.” Available: <https://rallly.co/>
- [56] “React.” Available: <https://reactjs.org/>
- [57] “React bootstrap components.” Available: <https://react-bootstrap.github.io/components/alerts/>
- [58] J. Rennecker and L. Godwin, “Delays and interruptions: A self-perpetuating paradox of communication technology use,” *Information and Organization*, vol. 15, no. 3, pp. 247–266, 2005. Available: <https://www.sciencedirect.com/science/article/pii/S1471772705000114>
- [59] “RIA: Frequently asked questions.” Available: <https://ria.princeton.edu/human-research-protection/resources-and-quick-links/ohrp-frequently-asked-que-1>
- [60] “<select>: The HTML Select element.” Available: <https://developer.mozilla.org/en-US/docs/Web/HTML/Element/select>
- [61] “Strawpoll.” Available: <https://strawpoll.com/>
- [62] “Strawpoll: Schedule a meeting.” Available: <https://strawpoll.com/meetings/>
- [63] F. Tetard, “Fragmentation of working time and smarter is solutions,” in *Proceedings of the 33rd Annual Hawaii International Conference on System Sciences*, 2000, pp. 10 pp. vol.2–.
- [64] “unittest - unit testing framework.” Available: <https://docs.python.org/3/library/unittest.html>
- [65] E. Vuong, “Docketer help - guests.” Available: [https://docs.google.com/document/d/1FZGoeQps5YBUeevAp\\_zUig0guIGgjFvd-dx14NKvZY/edit?usp=sharing](https://docs.google.com/document/d/1FZGoeQps5YBUeevAp_zUig0guIGgjFvd-dx14NKvZY/edit?usp=sharing)
- [66] E. Vuong, “Docketer help - hosts.” Available: <https://docs.google.com/document/d/1uFg-jsFcZDis5LvXQ1Kj1aIRS7ndyUPSUJRIZSwnoBI/edit?usp=sharing>
- [67] E. Vuong, “Docketer: Sanitized github repository.” Available: <https://github.com/evuong483/docketer.git>
- [68] E. Vuong, “Docketer: Video demonstration.” Available: <https://youtu.be/qdJrJkwTfbo>
- [69] E. Vuong *et al.*, “Collaborative Campus Map,” COS 333 Fall 2020-2021 Project.
- [70] “Wase: Web appointment scheduling engine.” Available: <https://wase.princeton.edu/princeton/views/pages/login.page.php>
- [71] “Whenisgood.” Available: <https://whenisgood.net/>
- [72] “Whenisgood: Create event.” Available: <https://whenisgood.net/Create>
- [73] “Whenisgood: Pricing.” Available: <https://whenisgood.net/Pricing>
- [74] “Woven.” Available: <https://woven.com/>
- [75] “Woven: Features.” Available: <https://woven.com/features>
- [76] “Woven: Pricing.” Available: <https://woven.com/pricing>
- [77] “Woven: Scheduling links.” Available: [https://woven.com/features/scheduling\\_links](https://woven.com/features/scheduling_links)
- [78] “x.ai for teams.” Available: <https://x.ai/teams/>
- [79] “x.ai how it works.” Available: <https://x.ai/how-it-works/>
- [80] “x.ai plans & pricing.” Available: <https://x.ai/pricing/>
- [81] “You can book me.” Available: <https://youcanbook.me/>
- [82] “You can book me: Features.” Available: <https://youcanbook.me/features/>
- [83] “You can book me: How it works.” Available: <https://youcanbook.me/howitworks/>
- [84] “You can book me: Pricing.” Available: <https://youcanbook.me/pricing/>
- [85] “You can book me: Scheduling for teams.” Available: <https://youcanbook.me/scheduling-for-teams/>
- [86] S. Zhang, “Deploy git subdirectory to heroku,” *Medium*, Aug 2018. Available: <https://medium.com/@shalandy/deploy-git-subdirectory-to-heroku-ea05e95fce1f>



# Appendices

## A. Market Research Notes

This section provides a transcript of notes taken while performing market research, including detailed account of marketing strategies and lists of features provided. All of the apps where I created an account offered a free trial period, so I had access to more features than might be generally available at the free tier. This distinction was not always clear to me, so features listed may or may not be supported at the free tier.

### A.1. Tags

The following tags were used to categorize scheduling apps (results summarized in Table 1):

- *AI/Personal Assistant*: Marketed as using/uses AI to schedule meetings, acts as an automated personal assistant by offering more personalized scheduling help (daily briefings, active scheduling, etc.)
- *Guest books with host*: The guest views the host's (entire) availability and picks a time that works best for them
- *Host picks time from availability*: The host requests the guest(s)'s availability and then picks the meeting time
- *Calendar/Meeting organizer*: In addition to automating scheduling somehow, also provides calendar/meeting organizer functionality (e.g. having an in-app calendar representation)
- *Scheduling group activity/poll*: Good for coordinating with several guests, choosing a time that many people are available
- *Can cause fragmentation*: Is the host's time likely to be fragmented when using this app?, generally caused by "Guest books with host"
- *Heavily targeted at businesses*: The app markets itself as being useful for meeting businesses' scheduling needs, internal and external
- *Can use for personal scheduling*: Do the costs and features available make this suitable for a personal scheduling app (e.g. has a free tier, easy calendar management, makes scheduling one-on-one meetings easy)

### A.2. Calendly

**URL:** <https://calendly.com/>

**How is it marketed? What features do they say are offered?** "World's #1 Scheduling Tool", provides automated scheduling, offers different types of meetings (one-on-one, round robin, collective, group), can take ownership of your scheduling and time (notifications, buffers, daily limits, minimum scheduling notice, time zone detection, customizations), lots of integrations with external services (payment, API, embed in website, calendars) [9, 10, 11]

**Cost:** 3 tiers for individual users (free/\$8/\$12 per user per month), also have solutions for companies, ("for teams") options [12, 13, 14]

**Can create a free account?** Free tier has lots of options

**What features are available (through looking around as a user)?** Personal url, connect to calendar, set up availability, different types of events (one-on-one only for free tier), create different calendars, view integrations list, view scheduled events, get link for scheduling meeting

**How does meeting scheduling work from the guest's perspective?** Only one-on-ones available for free tier, received a link, view availability and pick one, give name, email, details, sends an invite (I think you can integrate zoom here?), sends invite and adds to calendar, can cancel or reschedule easily

**Thoughts/Impressions:** Pretty typical compared to other scheduling software, definitely great for the person scheduling the meeting and moderately convenient for the person being scheduled, same Swiss cheese problem

**Tags:** Guest books with host, calendar/meeting organizer, can cause fragmentation, heavily targeted at businesses, can use for personal scheduling (good for this)

### A.3. YouCanBookMe

**URL:** <https://youcanbook.me/>

**How is it marketed?** “Customer bookings straight to your calendar”, integration with external services, meeting reminders, Zoom integration, availability management, notifications and scheduling page style customization, team availability, calendar view/manage existing bookings [81, 83, 82]

**Cost:** \$10 per month, also has business pricing [85, 84]

**Can create a free account?** 14 day free trial (can sign up with Google calendar/Microsoft Outlook calendar and import availability)

**What features are available?** Connect to calendar, add info and company name, creating booking page w/ personal url, logo, title, decide duration of meetings, padding, if guest can choose duration (and min-max/default size), min and max advance notice

**How does meeting scheduling work from the guest's perspective?** Very similar to Calendly. Go to URL, click on availability, give name, email, reasons. Sends an email confirmation and adds meeting to my calendar. Can add or schedule easily **Thoughts/Impressions:** Very similar to calendly. **Tags:** Guest books with host, calendar/meeting organizer, can cause fragmentation, heavily targeted at businesses (more so than Calendly)

### A.4. Wase

**URL:** <https://wase.princeton.edu/princeton/views/pages/login.page.php>

**How is it marketed?** Princeton internal. Link to external calendar, limit who signs up (course

membership etc.), schedule events, multiple calendars [70]

**Cost:** With university membership (guest access also?)

**Can create a free account?** I have a university account

**What features are available?** Make appointment with existing calendar (usually have to have Wase account to make appointments with others' Wase calendar/be logged in), update reminder/sync preferences, make a calendar (set up info, view preferences, add blocks that people can pick), blocks can be broken into slots, can have multiple people in an appointment slot, can set max appointments per person, set up when the block repeats, can also make block "unavailable", deadlines for meeting being scheduled, access restrictions & labels

**How does meeting scheduling work from the guest's perspective?** Go to calendar/calendar link/search for person you are trying to book with, click on slot (can easily delete and choose a new one, view ones that are available/have person ahead/etc.), sends email reminder at time and one day before

**Thoughts/Impressions:** I think this works well for setting up recurring office hours or a swath of similar spots (for speaking test or something), can imagine it is inconvenient to have to pick blocks but also acceptable in the context that this is happening (you already have a specific block of time and size of slots in mind). More this is my fixed availability for this activity (or this is when this event is happening), come and choose/join if you want to/when you can

**Tags:** Guest books with host, calendar/meeting organizer (has "my appointments" feature)

#### **A.5. Front Scheduling (formerly Meetingbird)**

**URL:** <https://frontapp.com/>

**How is it marketed?** Scheduling arm of Front (business organization/communication solutions) [28, 29]

**Cost:** Targeted at businesses, 4 tiers of per user pricing (\$99/\$49/\$19 per person per month for businesses) [30]

**Can create a free account?** 7 day free trial

**What features are available?** Front itself is like a business email (shared inboxes, calendars, etc.), add an event to your synced calendar with guests, location, etc.

**How does meeting scheduling work from the guest's perspective?** Harder to do, think you might add guests to an event, they can accept? Unclear.

**Thoughts/Impressions:** More focused on business communications, was unclear how to use

scheduling, not good for personal

**Tags:** Calendar/meeting organizer, heavily targeted at businesses

#### A.6. Woven

**URL:** <https://woven.com/>

**How is it marketed?** One place to schedule all meetings, calendar that works for you. Smart templates for meeting types (customizable but efficient), integrated scheduling links (setting up group meetings through group polls), automatically find the best time to meet (availability sharing), analytics about your time. Integrate unlimited number of external calendars with woven (i.e. use woven to combine them), different calendars to separate your events vs info, ios integration (siri shortcuts, voice command) [74, 75, 77]

**Cost:** \$15 or \$20 per month (billed annually/monthly) [76]

**Can create a free account?** 21 day free trial

**What features are available?** Join with Google or Outlook account, add your calendar, designate calendars attending vs. not attending (busy during this time? Yes for personal calendar, no for holidays etc.), add video conference integration (can add custom Zoom link here), choose style. Has desktop app and in-browser app. Calendar in the app is very nice. Highlights events during your available times/to come, and fades past events. Has keyboard shortcuts to toggle different views (week/day/month/my time). Home screen shows analytics, events to come, meetings. Create templates, calendars, view analytics.

**How does meeting scheduling work from the guest's perspective?** To create a one-on-one meeting, set up link by selecting available times (adding specific times is kinda annoying have to add block by block, or can enter which is probably okay by typing). Ask for notification, allows you to tag for analytics purposes. Add people, location, conferencing, etc. Create link and send email/share. Nice thing is that it shows the options next to a calendar with the actual slots so things are very easy to visualize. Same for going to selection link. See list of slots in sidebar, and also on calendar. Can select in either, add people. When you select a slot you will give name and email, email you confirmation. Does not offer as easy rescheduling/canceling as some others (not sure if person who is coming can even cancel without having a Woven account?). For a polling one, same as before select availability (also can do no sooner, no later, buffer time), from pre-populated list, send/share link. People select on calendar or list what they're available for, send name and email. Desktop app has "planning" list of events so you can check on progress. Now shows best option/other options with who voted against/for. Planner selects option, schedules. Also have open invite option (can limit # of participants, set cutoff time for sign ups too)

**Thoughts/Impressions:** Very nice interface, nice features and while similar to the others I think is a very good option but again similar to the others. Good flexibility (on the premium trial). Slightly less directly targeted at businesses (implied but not excessively like some of the others)

**Tags:** Guest books with host, calendar/meeting organizer, scheduling group activity/poll, can cause fragmentation (depending on how you set up your availability beforehand)

### A.7. Acuity

**URL:** <https://acuityscheduling.com/>

**How is it marketed?** Personal assistant, control availability, organize schedule, keep track of clients, integration (payments, group scheduling, reminders, video conferencing, privacy, synch with calendars), customizable [1]

**Cost:** Targeted at businesses, tiers for pricing (not per-head as previous ones were) (\$14/\$23/\$45 per month for whole business 1/6/36 employees billed annually, \$15/\$25/\$50 if billed monthly) [2]

**Can create a free account?** 7 day free trial **What features are available?** Create account, enter business info, create appointment type (appointments have prices), force you to enter availability manually (annoying), then can sync with Google Calendar (can select specific Google Calendar calendars to block from), customize scheduling page (monthly - far out, vs daily - earliest convenient time), view calendar, clients, reports, edit business features (forms, appointment types, payment etc.), can customize link

**How does meeting scheduling work from the guest's perspective?** View site (as determined by above, share or email link to schedule, embedded), click slot (has options for selecting multiple times or setting a recurring meeting which is nice in certain contexts), enter information (name, phone, email), confirm appointment and get options to add it to calendars, reschedule, cancel, and gives you a QR code for app for managing appointments with this business, sends email to you and person you're scheduling with

**Thoughts/Impressions:** More focused on scheduling for businesses. I found UI was a little less user friendly (when logged in) but also more customizable, definitely for more heavy-duty scheduling. Recurring meetings/payment/customizability make it great for businesses, much less so for personal use. Again, similar to other things great for person booking, less so for the person they're booking with. Has confirmation for deleting an appointment.

**Tags:** Guest books with host, calendar/meeting organizer, can cause fragmentation, AI/Personal assistant, heavily targeted at businesses

### A.8. x.ai

**URL:** <https://x.ai/>

**How is it marketed?** AI SCHEDULING!! A!! (really emphasizes the use of AI). Automates tedious parts of scheduling for groups, targeted at businesses, intended to integrate into the emailing process of setting up a meeting (email the people you want to set up with and AI scheduler, AI will recommend times and make sure everyone is up to date), customizable/branded pages, round-robin

scheduling, team dashboard, payments, conference rooms, privacy, group meetings, embedding, questions from guests before meetings, daily email prepping for day (info about people you're meeting and conflicts), instant scheduling for groups that all have [78, 79]

**Cost:** Free plan (forever), individual \$8 per month billed annually (\$10 billed monthly), team \$12 per user per month billed annually (\$15 billed monthly) [80]

**Can create a free account?** Free account

**What features are available?** Integrate with existing calendar, set up info and availability, it gives you a list of people you've scheduled meetings with and asks who you schedule with regularly (if they also set up an account can schedule meetings instantly). Unlimited calendars, the option to copy text format of availability (customizable format) to send in an email, cc scheduler on email to set up meeting or send one-time link, can view all existing appointments (in rolling day-by-day calendar view). Send calendar page link, copy paste times, let x.ai send times, or create one-time link. Set up calendars with availability rules (comes with some defaults - lunch, coffee, etc.) (how long into the future, available hours, duration, lead time -> schedule not within 1 day, breather between meetings, slots per day, time breaks when they can begin, limit meetings per day/week (max), also with guest settings, privacy, questions, payment, follow-ups, post-meeting actions (paid)).

**How does meeting scheduling work from the guest's perspective?** The copy-pasted link is pretty cool -> formats it and adds links so they can click one and set up the meeting directly. Also includes link to calendar page, set up for the desired meeting (similar to one-time link). Different calendars have different types of meetings/preferences/etc. that determine your availability. Select from available meetings of a specific type or tells them to email you if none available. Asks for email, name, and then send email reminders. (also adds to person's calendar with the AI)

**Thoughts/Impressions:** I actually thought these tools were pretty cool. The "AI" abstracts a bit of the availability determination away from the client in a way that I would like to emulate, especially with the defaults. This app had the most nuanced view of availability. I also think the idea of having the scheduler respond to emails and send emails is pretty creative (was a little dubious at first but I can definitely see how this would work well, especially within a company). UI is not super helpful for the person setting the meeting up but I also don't think that's intended to be the selling point.

**Tags:** AI/Personal assistant, guest books with host, calendar/meeting organizer, scheduling group activity/poll (IF all attendees have an x.ai account can schedule automatically), can cause fragmentation, heavily targeted at businesses, can use for personal scheduling

## **A.9. Outlook Calendar/Microsoft Bookings**

**URL:** <https://www.microsoft.com/en-us/microsoft-365/business/scheduling-and-booking-app>

**How is it marketed?** Seems like they have a scheduling assistant/room finder that can help schedule meetings within a group of outlook accounts (help find a shared available spot). meetings appear on outlook calendar, can see who has RSVPed, give location (typical). Employees can check same calendar, embed in website, branding. [48]

**Cost:** Targeted at businesses, comes with other Microsoft business bundles or per user per month cost (\$5/\$12.50/\$20 per user per month for a year, comes with Microsoft 365) [49]

**Can create a free account?** No but previous experience

**What features are available?** Set up an event, check conflicts, sync with calendar, RSVP

**How does meeting scheduling work from the guest's perspective?** Set up event, add people who are coming, will see how they RSVP. To RSVP, when you receive the email notification click your response and it will add itself to your calendar (Outlook only). Microsoft bookings seems similar to Calendly, etc.

**Thoughts/Impressions:** (About Outlook) No frills but does do its job. Very simplistic though and while it is worth mentioning here that it's not really the type of scheduling app I'm looking at: it actually requires more scheduling outside of creating the event (by the time you actually get to the RSVP part all parties probably already agreed to the time and this is just to add it to your calendars), okay for big events (see who's going and everything but like Wase, should be planned ahead of time)

**Tags:** Calendar/meeting organizer, heavily targeted at businesses

## A.10. Google Calendar

### URL:

<https://support.google.com/calendar/answer/190998?co=GENIE.Platform%3DDesktop&hl=en>

**How is it marketed?** Appointment slots (create a block with slots right in your Google Calendar, send link people can use to book and they'll show up straight away). Sends invitation automatically (with RSVP like Outlook). Similarly can schedule events and add invitees who can accept or decline. [38]

**Cost:** Available for work or school accounts. \$6/\$12, \$18 per user per month or custom for businesses [39]

**Can create a free account?** Have a school account

**What features are available?** In Google Calendar, create appointment slots/events, send link for people to book (somewhat similar to Wase)



**How does meeting scheduling work from the guest's perspective?** Click a slot (have to be signed in with a Google account), will automatically book and send you an invitation, appear on your calendar so you can RSVP.

**Thoughts/Impressions:** Booking interface not pretty but very integrated with Google Calendar. All slots must be made manually. Very simplistic much like Outlook

**Tags:** Guest books with host, calendar/meeting organizer

#### A.11. StrawPoll

**URL:** <https://strawpoll.com/>

**How is it marketed?** Simple. Fast. Free. Create polls to schedule meetings/group events (also intended for polling on other topics). Offers some simple options (poll deadlines, supports emojis, API, free). User driven development. No integration with calendars [61]

**Cost:** Don't even need account to create poll. Free.

**Can create a free account?** Free account

**What features are available?** Create poll or meeting. Add meeting details, add days/times (adding days is easy - click on the calendar and it will automatically add to list of available times. can add simple limits and extend to all days. Good for big blocks of time across several days. Can also template i.e. every hour or something). Invite or share by link. Add your name, check days (available, yellow, or no check -> unavailable). If the granularity is smaller it's painful to click through them all if too many, but would not be helpful otherwise. Can edit responses. Person who created poll can view results, choose a time and schedule the meeting then. More for finding meeting time than actually scheduling it.

**How does meeting scheduling work from the guest's perspective?** Receive a link. Add your name, check days (available, yellow, or no check -> unavailable).

**Thoughts/Impressions:** Very simplistic.

**Tags:** Host picks time from availability, scheduling group activity/poll

#### A.12. WhenIsGood

**URL:** <https://whenisgood.net/>

**How is it marketed?** No account needed. Just for (group) scheduling. Pricing possible for no ads, slightly more respondent options, Excel exporting. [71, 72]

**Cost:** Don't even need account to create poll, free or \$20 per year [73]



**Can create a free account?** Didn't create account.

**What features are available?** Nothing beyond scheduling/viewing results (save/use an event code to check results)

**How does meeting scheduling work?** Click and drag on calendar to select polling times. Can pick slot lengths (only one size for all of them), and size of polling map. Can edit date, shown times, url, top text. Click and drag to highlight possible slots. Supports time zones. After creating the event are given a code to see results (I think without this if you have an account that might also work). Send the code to people. They will also paint over slots that are good, add their name and comments, and then send the response. They also get a link where they can update their responses. When viewing the responses you see greyed boxes where anyone can't attend + red dots for each person (hover to see who said they can't attend), green slots where all respondents are okay. You can delete responses or request an update, or you can select a spot and get an iCal link to add to calendars/create the invite.

**Thoughts/Impressions:** Very simplistic also but more specialized than straw poll, except only has available or unavailable. The painting over available times is really nice.

**Tags:** Host picks time from availability, scheduling group activity/poll

### **A.13. Rallyy**

**URL:** <https://rallyy.co/>

**How is it marketed?** Collaborate with friends. Create event page, discuss, and vote. Free to use, open source [55]

**Cost:** No account needed (in fact, cannot create account)

**Can create a free account?** Cannot create account but can use service

**What features are available?** Nothing beyond scheduling/viewing results

**How does meeting scheduling work?** Enter name, email, event details, select dates for event on calendar. Invite participants by entering their emails. Enable/disable notifications. Get a link to send to people. Person receives the link. Can add comments, add their name and select days they like. Granularity is fixed at days. After you see availability you can pick a day and schedule.

**Thoughts/Impressions:** Very simplistic. Similar to StrawPoll but only available/unavailable options and granularity at best days.

**Tags:** Host picks time from availability, scheduling group activity/poll

### **A.14. GoodTime**

**URL:** <https://www.goodtime.io/>

**How is it marketed?** Marketed as interview scheduling/management solution. Two products, GoodTime Meet (like Calendly -> people schedule with you), and interview scheduling, which is also marketed to be candidate schedules (and can reschedule) by viewing your availability. Automatic interviewer selection, coordinate followups, back-to-back, etc. Also offers diversity recruiting, interview training, virtual interviewing solutions, integrations. [34, 35, 36]

**Cost:** Free or expensive (for much larger businesses). Starting at \$999 a month for <200 employees. Targeted at large recruiting teams. [37]

**Can create a free account?** Yes

**What features are available?** Join with Google, select workdays and times/timezone, give common meeting length and type (video/in-person). Can make different meeting types with custom event descriptions (greet the attendee by their name, give them a specific link, etc. Buffers, meeting limits per day, search window length, allow rescheduling. Maximum number of reschedules, etc.

**How does meeting scheduling work from the guest's perspective?** Go to link, select from available days (some number of weeks out), more granular available slots are shown, pick one, give name and email, schedule automatically. I used this for scheduling one interview and the interviewer had me give my availability out of some options and then updated me with the time that had been selected. Did not schedule automatically.

**Thoughts/Impressions:** I can imagine this would be good and very customized for companies but depending on how it is used for candidates it can be annoying. When I used it (as a candidate) I had to select a bunch of times day by day which was tedious, but then if I wanted to go back and change anything I had to re-do everything again which was very annoying. No Swiss cheese because it literally can't be used for personal scheduling. **Tags:** Host picks time from availability, guest books with host, calendar/meeting organizer, heavily targeted at businesses, can be used for personal scheduling.

## A.15. Greenhouse

**URL:** <https://www.greenhouse.io/>

**How is it marketed?** Also interview scheduling/applicant management similar to GoodTime. Analytics, attracting talent, customize/integrate. Also for onboarding. Emphasizes diversity, scalable hiring. Find and track candidates, use equitable hiring by setting rubrics at the beginning to be consistent, etc. May also offer consulting? [40, 42]

**Cost:** Prices not listed but probably expensive. Businesses only. [41]

**Can create a free account?** No

**How does meeting scheduling work from the guest's perspective?** Also used this for scheduling an interview. They sent me an email with a place I could put in my availability (not select from

some existing slots, had to enter the start and end of each free block manually which was more annoying than selecting from existing spots), then internally they picked a time and let me know which.

**Thoughts/Impressions:** Similar to GoodTime. I assume it's valuable internally to the company but my experience was okay, not spectacular. No integration so I had to enter my availability for 3+ weeks by typing them in to various boxes.

**Tags:** Host picks time from availability, calendar/meeting organizer also, entirely targeted at businesses

## **B. IRB Approval Application**

As mentioned in Section 5, I had originally hoped to work with a group of test users to evaluate the usability of my product and potentially gain insight into whether or not this new approach to scheduling actually solved the issues I set out to solve. Below I include for reference a copy of relevant pages (21 pages) from the application I used to seek approval from the IRB, including the final version of each supporting document. We were asked to make revisions to the initial application from January in February to address some inconsistencies in the supporting materials and did so immediately. Then, in late April we were notified that the IRB would move forward with the revisions only once actions relating to another one of my adviser's IRB proposals had been completed. As these actions were due May 1st, it would not have been possible for me to complete the evaluation without doing so in non-compliance of the IRB (before receiving approval). I did end up receiving funding from the department for incentives but was unable to use it as intended.

## 2 General Information

---

Protocol Number 13648

Submission Number 13648-01

---

**Revise the default Study Title (2.1 below) NOW to reflect the study purpose - it may not be amended after submission.**

**\* (2.1) Study Title:**

An Alternative Approach to Scheduling: Time Slot Preference Ranking

**\* (2.2) Is your research being funded by any of the U.S. Department of Health and Human Services (DHHS) agencies listed [here](#)**

☐ Yes ☒ No

**\* (2.3) Select your study sponsor using the icons below:**

*First click the Add Sponsor link below. Then click the pencil icon to select your funder. Repeat as necessary if you have more than one funder. NOTE: If your study is internally funded by Princeton University, please choose "Princeton University" or "Internal" (either choice is fine).*

* Sponsor	Internal
-----------	----------

**\* (2.4) Type of Research:**

- ☐ Senior Thesis
- ☐ Faculty Research
- ☐ Postdoctoral Research
- ☒ Junior Project
- ☐ Graduate Research
- ☐ Other

**Please note, per the University Research Board, undergraduate students, graduate students, and postdocs CANNOT serve as the PI for an IRB submission.**

**Please refer to the University Research Board PI-eligibility [chart](#).**

**Please save your work before proceeding to the next section of this form. Otherwise, your work will not be saved.**

### 3 Research Personnel

[Add Personnel](#)

Name

Lumbroso, Jérémie



Principal Investigator (PI)

*The PI is the person who is ultimately responsible for the conduct of the study. For student-initiated research, the PI is typically the student's faculty advisor. However, the PI does not have to be your faculty advisor. For the eligibility criteria to serve as a PI at Princeton University, please see [this](#) document.*

Role in Research: Faculty Advisor

IRB Compliance Training

*This information is automatically populated based on data received from the [CITI Program](#). To allow the data to auto populate, you must affiliate yourself with Princeton in the CITI Program and use your Princeton email address in CITI.*

CertificationBeginEnd

- -

Name

Vuong, Erin



Principal Investigator (PI)

*The PI is the person who is ultimately responsible for the conduct of the study. For student-initiated research, the PI is typically the student's faculty advisor. However, the PI does not have to be your faculty advisor. For the eligibility criteria to serve as a PI at Princeton University, please see [this](#) document.*



Email Communication

*The IRB will send all communications to the PI. If you would like this person to also receive IRB communications about this study, please check this box.*

Role in Research: Performing IW

IRB Compliance Training

*This information is automatically populated based on data received from the [CITI Program](#). To allow the data to auto populate, you must affiliate yourself with Princeton in the CITI Program and use your Princeton email address in CITI.*

CertificationBeginEnd

- -

## 5 Study Design

---

The IRB asks the following questions because it must confirm that the study meets certain criteria for approval found in the federal regulations. If you are interested in learning more about these criteria, please visit [this page](#).

---

### Objectives

**\* (5.1) Describe the purpose of your [research](#) project such that members of the IRB who have expertise in areas other than yours can understand the proposed research. Include a description of the background and rationale for the study, explain the research design, research methodology, hypotheses and goal(s).**

A variety of scheduling software already exists to help people arrange meetings (You Can Book Me, Calendly, WhenIsGood, etc.). Many of these options involve sending the person you are booking with a link to your available time slots and letting them select the option that works best for them. While this is an easy way to quickly schedule meetings without back-and-forth, not all available time slots are created equal. Some people may prefer to frontload their day or week. Some people may prefer to avoid switching back and forth between different tasks and meetings by scheduling them all together. Current scheduling solutions do not take these preferences into account. My goal is to create a scheduling app that more intelligently considers time management preferences, and incentivizes the people scheduling with you to pick your more preferred slots.

Dr. Lumbroso (faculty advisor) first proposed the idea for this project to me after he became frustrated with how allowing people to view his full availability and select the time that worked best for them when scheduling a meeting (using YouCanBookMe) had a tendency to turn his time into "Swiss cheese". Large blocks of free time became filled with "holes", or unrelated meetings, as people scheduled here and there. Dr. Lumbroso, like myself and likely many others, requires some time to switch between tasks and ramp up to work. Having a meeting, then an hour of free time to work, and then perhaps another meeting instead of two meetings and then two hours of free time decreased the productivity of his time. He also said he preferred completing meetings in the mornings if possible, but without this even being communicated through the scheduling software much less enforced somehow, as might be expected his preferences were usually not respected.

The majority of my IW will consist of building a prototype for an app that solves the issue above by ranking the preferability of available time slots according to their time of day, day of week, and how much they interrupt existing blocks of unscheduled time, and then when someone is scheduling with a user to present the time slots one by one, from most preferred to least preferred, instead of presenting them all at once. By making it slightly more inconvenient for the person trying to schedule to pick a time that is more inconvenient for you (they must click through more availability options), they will be encouraged to pick a more convenient meeting time.

In order to evaluate how well my prototype actually solves the issue it is intended to, I am hoping to gather a group of human users and ask them to provide tasks and perform feedback. In both my app and an existing scheduling app I will create a test account for each user with some predefined somewhat randomly populated schedule to avoid asking users to use their own schedules. Then, I will ask all users to schedule meetings with several other users using both platforms to simulate how scheduling meetings among many people might happen in real life. Finally, I will ask each user to compare their satisfaction with the schedule produced by the existing app and my app, to evaluate whether my app actually succeeds in encouraging people to schedule meetings at times that are more preferable to my user, and if this overall is useful to people trying to both schedule meetings efficiently and manage their time. This will be more of an informal survey of user impressions than a rigorously methodological study.

My hypothesis is that automating scheduling in a way that considers my user's scheduling preferences will increase their satisfaction with how meetings are scheduled. My goal is to produce a prototype of one such scheduling app, and then informally determine if the idea seems promising and perhaps worth continuing in another IW next year by surveying a small group of users on their experience using it.

To clarify, this research involving human subjects is designed to develop or contribute to knowledge that is generalizable knowledge, namely to determine how factors that individuals value in schedules could be used to improve the automation of schedules. While drawing definitive conclusions on the subject may be beyond the scope of this specific one-semester project, I hope to, in a later iteration of this project, use the results collected now and further explore their generalizable implications.

I have included my written project proposal in the supporting documents section that contains more information and background on my project.

### Research Components

The research involves the following components (check all that apply):

\* **(5.2) Use of Zoom, Go-To-Meeting, or other virtual environment to conduct interviews or other interactions:**  
☐ Yes ☒ No

\* **(5.3) Questionnaires/Surveys/Interview Guide/Topics of Conversation:**  
☒ Yes ☐ No

*Upload the Questionnaires/Surveys/Interview Guide in section N ("Supporting Documents") of this form. The IRB cannot make a risk determination of the study without reviewing the proposed questions.*

*Proposals to survey current students will generally be limited to a random sub-sample of no more than 1000 students; requests to survey the entire student population must be accompanied by a strong methodological rationale for why this large sample size is required. See the IRB Guidelines for Survey Research found on the below resources page:*

[Human Research Regulations and Resources](#)

\* **(5.4) Ethnographic/Field Research/Participant Observation/Embedding in the field:**

*For the purposes of this form, "ethnography" means a study in which the investigator will be immersed or embedded in specific social settings.*

☐ Yes ☒ No

\* **(5.13) Analysis of tissues or specimens:**  
☐ Yes ☒ No

**(5.14) Recordings (check all that apply and revise the consent form accordingly):**

\* **(5.14.1) Audio**  
☐ Yes ☒ No

\* **(5.14.2) Video**  
☐ Yes ☒ No

\* **(5.14.3) Photography**  
☐ Yes ☒ No

\* **(5.15) Data Analysis Only** (the research is solely limited to data analysis- no interaction or intervention with the subjects):  
☐ Yes ☒ No

\* **(5.16) Clinical Trials:**

*A research study in which one or more human subjects are prospectively assigned to one or more interventions (which may include placebo or other control) to evaluate the effects of those interventions on health-related biomedical or behavioral outcomes. This definition encompasses phase 1 trials of FDA regulated drug and biological products, small feasibility studies of FDA-regulated device products, and studies of any intervention not regulated by the FDA, e.g., behavioral interventions. If your study is NIH-funded, the PI is encouraged to contact the NIH Program Official (scientific contact for the study) who will determine whether the study meets NIH's definition of a clinical trial.*

☐ Yes ☒ No

\* **(5.17) Hazardous Agents:**

Does this protocol involve the use of radiation, chemical, or biological agents (including recombinant DNA or BSL2)?

☐ Yes ☒ No

## Study Procedures

### \* (5.22) Describe in detail the procedures that will be used to achieve the objectives of the research project:

In order to survey user impressions of the performance of my application, I will:

1. Gather a group of 25-50 users who have access to a computer and the internet and who are willing to spend 1 hour over two different periods for evaluation, 1/2 an hour for performing scheduling tasks and 1/2 an hour for evaluating the results of the scheduling tasks
2. For each user I will create a pair of test scheduling accounts: a test account on my application and a test account on an existing scheduling application (free). I will pre-populate the availability on these two accounts semi-randomly for a two-week period for each user to simulate an actual schedule (both accounts will be pre-populated with the same availability). Note that the users will not have access to these accounts, and no specific accounts will be tied to a specific user.
3. For each test account I will generate 5-10 one-time meeting scheduling links. These will be used by other users in the test group to schedule meetings with the user for whom they were generated (no logging in required to schedule a meeting). The links represent meetings a user might need to schedule within a 1-2 week period. I will permute these meeting scheduling links and then distribute them among the users (I have not yet decided if a user can schedule with the same test account more than once, but it seems reasonable that this could occur as someone might have multiple meetings with the same person in one week). Each user will receive only links generated by my app or only links generated by the existing app to avoid being influenced by the other scheduling approach (a control group and a testing group of sorts).
4. Each user will use each link they received to schedule a one-hour meeting. I will suggest that they schedule the meetings according to their actual availability or preferences, and not at the same time for every link, but these will not be enforced in any way (i.e. I will not know anything about the user's actual schedule). In order to schedule a meeting a user usually is prompted for a name and email address. I will give all the test users a fake name and email address to use so they will not need to use any of their personal information outside of communications with me directly.
5. After all users have scheduled with one another, for each pair of testing accounts I will generate pictures of the schedules consisting of the pre-populated events (same for both) and the meetings that test users scheduled using the generated one-time scheduling links (likely will be different for each scheduling app). I will likely generate images similar to Google Calendar weekly-view, at least one for each account, perhaps two per account, one where all events are the same color and one where scheduled events are a different color to elicit specific feedback about the varying scheduling approaches (see the sample survey for example pictures).
6. For each user I will send them the images generated from one pair of testing accounts, along with a questionnaire about their impressions for them to fill out. I will compile their responses and analyze them for common themes, opinions, etc.

## Number of Subjects

### \* (5.23) Indicate the maximum number of subjects to be accrued across all sites. If a total number cannot be provided, please provide an estimate. If the activity is solely limited to data analysis, indicate the number of subjects' records that will be analyzed:

50

## Study Populations

*The regulations require that additional protections be given to the subject populations listed below. If you indicate Yes to any of these questions, the application will guide you to resources you should consult.*

### \* (5.24) Does your study involve children?



☐ Yes ☒ No

\* (5.25) Does your study involve prisoners?

☐ Yes ☒ No

\* (5.26) Does your study involve pregnant women, human fetuses or neonates?

☐ Yes ☒ No

### Inclusion and Exclusion Criteria

\* (5.27) Describe the criteria that define who will be included in your study and who will be excluded from your study:

Test users will be anyone with access to a computer and the internet that is willing and able to perform the scheduling tasks (navigate to a link, select a time, enter a fake name and email address, click submit) and then look at some pictures and fill out a questionnaire about their opinions at the end. The only true criterion for exclusion is the inability to perform the tasks above for any reason. Realistically, I will only be able to include users who I can contact quickly and who would be willing to perform evaluation, such as my peers or other members of the Princeton community.

\* (5.28) Will you collect information from subjects who are physically located in the European Economic Area (EEA)?

*The EEA includes the following countries: Austria, Belgium, Bulgaria, Croatia, Republic of Cyprus, Czech Republic, Denmark, Estonia, Finland, France, Germany, Greece, Hungary, Iceland, Ireland, Italy, Latvia, Liechtenstein, Lithuania, Luxembourg, Malta, Netherlands, Norway, Poland, Portugal, Romania, Slovakia, Slovenia, Spain, Sweden, and the UK. To avoid GDPR-applicability, you must exclude individuals who are physically located in the EEA.*

☐ Yes ☒ No

### Multi-Site Research

\* (5.29) Is this is a multi-site study?

*A multi-site study is research that takes place at more than one entity and each entity has an IRB. Please click on the above hyperlink for examples. Note that multi-site studies involving human subjects research funded by the federal government must decide upon a single Institutional Review Board ("sIRB"). Please contact the IRB office if you have questions.*

☐ Yes ☒ No

### Recruitment Methods

\* (5.32) Describe how you will locate, contact, and invite subjects to participate in your study. If you will be using recruitment materials such as advertisements or invitations (printed, audio, visual, online), submit them in the Supporting Documents section of this form:

I will locate subjects through my existing social networks and in the wider Princeton community. I will contact them through email and invite them to participate in the study. This may involve an email to the Princeton undergraduate listservs which can be used to reach a wider audience. I have included a draft of what an invitation email would look like in the supporting documents in "emai\_draft.pdf"

**\* (5.33) Describe the amount and timing of any payments, if any, to subjects:**

There will be no monetary payment for completion of evaluation for my users. I am hoping to provide a small compensation or incentive in the form of snacks for users who help in the evaluation of my product, but this will be dependent on receiving funding from the University for such incentives.

On the sample consent form and recruitment email draft provided (updated versions) I have covered both the case where I receive funding for an incentive ( indicates the snack incentive mentioned above), and where I do not receive funding for an incentive (on the email draft the relevant sentence is merely omitted, there is no alternative).

## Resources

**\* (5.34) Describe your process to ensure that all research personnel are adequately informed about the protocol, the research procedures, and about their duties and functions:**

There will be no personnel conducting the evaluation of my project outside of myself (Erin Vuong). I will be following the process outlined above.

**\* (5.35) Describe the availability of medical, psychological, or other resources that subjects may need as a result of participating in your study and explain how these resources will be supplied:**

No such resources should be needed as a result of my study.

**Please save your work before proceeding to the next section of this form. Otherwise, your work will not be saved.**

## 6 Risks to Subjects

---

**\* (6.1) List the reasonably foreseeable risks, discomforts, hazards, or inconveniences to the subjects related to the subjects' participation in the research:**

The largest foreseeable inconvenience to my subjects will be the hour or so it takes them to complete the required tasks. There are no foreseeable risks, discomforts, or hazards.

**\* (6.2) Describe the probability, magnitude, duration, and reversibility of the risks. Consider physical, psychological, social, legal, and economic risks:**

There should be no such risks from completing my study.

**\* (6.3) Describe any financial costs (for example, medical tests, tolls, parking) that subjects may be responsible for because of participation in the research:**

There should be no financial costs from completing my study. My subjects should already have access to all the tools required to complete the tasks (a computer and the Internet), so completing my study should pose no additional financial costs to the subject.

**\* (6.4) Please describe the medical treatment and compensation that will be provided to subjects in the event of an injury caused by your research. If medical treatment and compensation will not be provided to subjects, indicate that:**

No medical treatment and compensation will be provided.

**\* (6.5) Do you think the study poses more than Minimal Risk to subjects?**

☐ Yes ☒ No

**Please save your work before proceeding to the next section of this form. Otherwise, your work will not be saved.**

## 7 Potential Benefits

---

**\* (7.1) Describe the potential benefits, if any, that individual subjects may experience from taking part in the research: Include the probability, magnitude, and duration of the potential benefits.**

I do not anticipate there being any benefits to users who participate in my study.

**\* (7.2) What is the value of this study to society?**

As stated before, a variety of scheduling software already exists to help people arrange meetings and automate the scheduling process, but they do not take into account the preferences of the person who is being scheduled with and only focus on making the process efficient. If this study shows that my approach to scheduling that takes into account my users' preferences provides some discernable improvement over the existing scheduling solutions, then I may be able to continue work on this project in the future and produce a polished app that will hopefully both automate scheduling to make it more efficient while also helping people schedule in a way that allows them to be more productive and happier.

**Please save your work before proceeding to the next section of this form. Otherwise, your work will not be saved.**

## 8 Withdrawal of Subjects

---

**\* (8.1) Describe anticipated circumstances under which subjects may be withdrawn from the research without their consent:**

If a subject does not complete the first stage of the study (completing scheduling tasks), they may be withdrawn from the study to decrease unnecessary work for me (generating images, emailing them a questionnaire) if they seem unlikely to respond.

**Please save your work before proceeding to the next section of this form. Otherwise, your work will not be saved.**

## 9 Confidentiality

---

\* (9.1) Is your study federally funded and will you collect identifiable, sensitive information?

*Examples of "sensitive" information include research on mental health and research on the use and effect of alcohol and other psychoactive drugs.*

☐ Yes ☒ No

\* (9.2) Is your study funded by NSF? (National Science Foundation)

☐ Yes ☒ No

To complete the following sections, please refer to the following resources:

- [Research Data Security Guidelines](#)
- [Protecting Your Research Data -- Summary Table](#)
- [Research Data Management Guidelines](#)

Given the above guidelines, describe the procedures to secure and maintain the confidentiality of data in the following areas:

\* (9.4) Desktop and laptop computers used in the study:

No data will be stored locally on a computer or laptop. See Cloud storage services.

\* (9.5) File server(s) used in the study:

No file servers will be used in this study. See Cloud storage services.

\* (9.6) Removable media used in the study:

No removable media will be used in this study.

\* (9.7) Paper forms used in the study:

No paper forms will be used in this study.

\* (9.8) Cloud storage services used in the study:

Google web services (in particular Google Drive as provided with my PU Google account) will be used to gather and store the results in this study. I will be storing the names and email addresses of my subjects in order to contact them as needed to complete evaluation tasks. This information will be entirely confidential, available to only me (Erin Vuong).

\* (9.9) Encryption methods used in the study:

Princeton University Google Drive is automatically encrypted (<https://protectourinfo.princeton.edu/tools/store/google-drive>).

\* (9.10) Data transfer methods used in the study:

Email (my Princeton University Google account to subject's email account in order to give instructions and receive feedback). The evaluation results may be shared between my advisor and I using sharing through university Google Drive accounts. This will not include any PII, including parts of survey responses that may be used to infer the subject's identity. The data will be sanitized before being shared with anyone else.

**Please save your work before proceeding to the next section of this form. Otherwise, your work will not be saved.**

## 10 Privacy

---

\* (10.1) Describe the steps that will be taken to protect subjects' privacy interests.

*Privacy is about people and means respecting an individual's right to be free from unauthorized or unreasonable intrusion, including control over the extent, timing and circumstances of obtaining personal information from or about them. Click on the above link for examples.*

I will only collect the names and email addresses of tests subjects in order to contact them throughout the evaluation. This information will not be shared with anyone else at any time. Furthermore, the actual survey/questionnaire/evaluation responses will be carefully sanitized to ensure that no personally identifying information has been shared before making the responses available to anyone outside of myself. In my project writeup I may include quotations from certain responses, but will again ensure that no information can be used to infer the identity of any of my subjects.

**Please save your work before proceeding to the next section of this form. Otherwise, your work will not be saved.**

## 11 Consent

---

\* (11.1) Will subjects consent?

☒ Yes ☐ No

The IRB recommends that you use the Princeton University IRB consent form [template](#). The consent form template contains approvable language and its use will minimize delays in the IRB review process.

\* (11.3) Will any information about the study be withheld during the consent process (a.k.a., deception)?

☐ Yes ☒ No

\* (11.5) Will subjects sign the consent form?

☐ Yes ☒ No

\* (11.6) One of the below criteria must be met for the IRB to approve a waiver of written documentation of consent. Please justify how one of the below criteria is met.

- (i) That the only record linking the subject and the research would be the informed consent form and the principal risk would be potential harm resulting from a breach of confidentiality. Each subject (or legally authorized representative) will be asked whether the subject wants documentation linking the subject with the research, and the subject's wishes will govern;
- (ii) That the research presents no more than minimal risk of harm to subjects and involves no procedures for which written consent is normally required outside of the research context; or
- (iii) If the subjects or legally authorized representatives are members of a distinct cultural group or community in which signing forms is not the norm, that the research presents no more than minimal risk of harm to subjects and provided there is an appropriate alternative mechanism for documenting that informed consent was obtained.

The second criteria is met. Visiting five links, picking an available time, and entering some designated fake information provided to the subject at each, then looking at and comparing some photos according to the subject's preferences and providing feedback poses no more than minimal risk and does not generally require written consent. The participants will tacitly consent by performing the tasks provided to them, and will be able to revoke their consent at any time or to any portion of the study by merely not completing the given task(s).

\* (11.7) Are you obtaining consent of non-English-speaking subjects?

☐ Yes ☒ No

\* (11.12) Subjects are adult students or employees.

☐ Yes ☒ No

\* (11.14) Subjects are adults who are [cognitively impaired](#).

☐ Yes ☒ No

\* (11.16) Subjects are adults who are unable to consent.

☐ Yes ☒ No

Please save your work before proceeding to the next section of this form. Otherwise, your work will not be saved.



## 12 International Research

---

\* (12.1) Will this project be conducted in whole, or in part, at a location outside the United States?

☐ Yes ☒ No

**Please save your work before proceeding to the next section of this form. Otherwise, your work will not be saved.**

### 13 Conflict of Interest

---


\* (13.1) Does the PI or any research personnel have a financial or other potential conflict of interest affecting objectivity in the study? For more information, please review the [COI website](#) for details.


☐ Yes ☒ No


**Please save your work before proceeding to the next section of this form. Otherwise, your work will not be saved.**


## 14 Supporting Documents


Attach supporting documents here. Supporting documents include advertisements, surveys, consent documents, interview guides, questionnaires, and verification of human subjects training (if not completed via CITI and while at Princeton).


Document Type	Other
Document Description	Draft of invitation to participate in survey email
Upload	


Document Type	Consent Form/Script
Document Description	Draft of consent form
Upload	

Document Type	Survey
Document Description	Sample of the survey I will use to collect feedback (the final survey will be this simple)
Upload	

Document Type	Other
Document Description	A written project proposal containing more information about my project and background.
Upload	

Document Type	Consent Form/Script
Document Description	Draft of consent form (updated)
Upload	

Document Type	Other
Document Description	Cover letter for requested modifications
Upload	

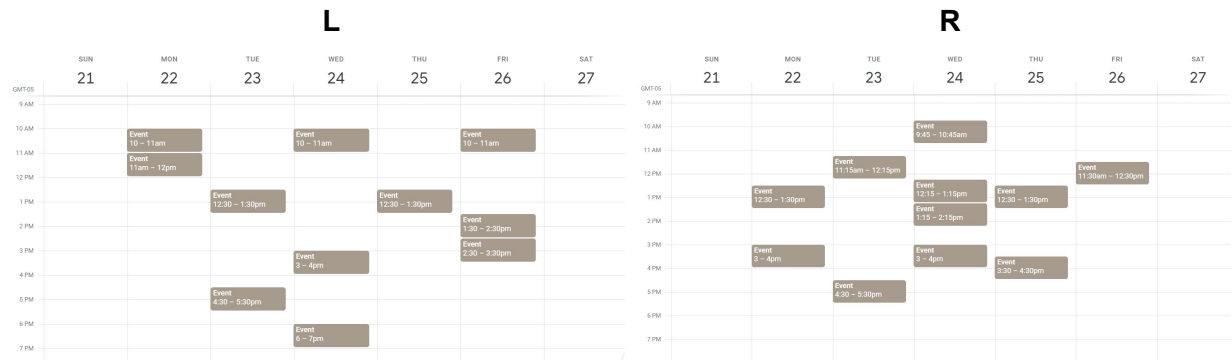
Document Type	Other
Document Description	Draft of invitation to participate in survey email (updated)
Upload	

**Please save your work before proceeding to the next section of this form. Otherwise, your work will not be saved.**

Sample Survey

Please follow the directions and answer the questions as instructed below.

Part 1: Below are two weekly schedules, left (L) and right (R).



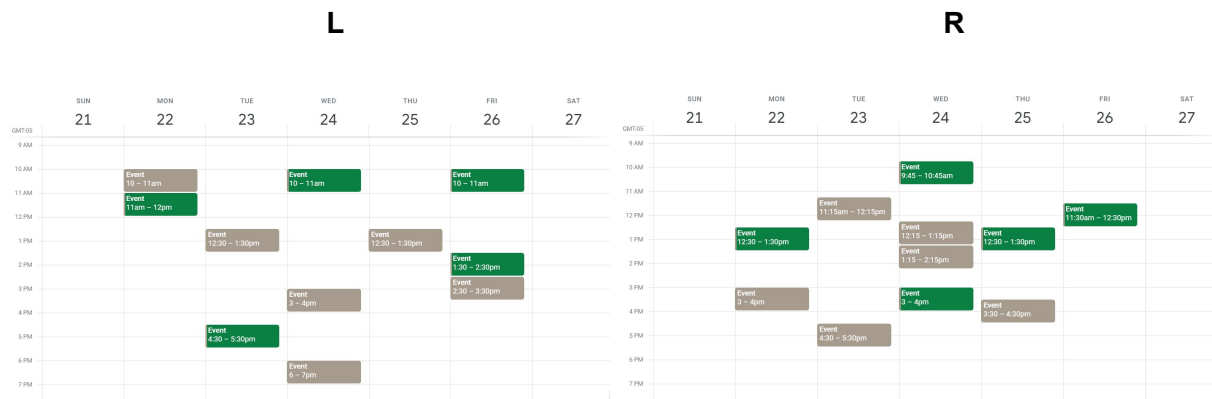
Q1: What differences do you notice between the two schedules?

Q2: Imagine that these are two possible schedules for your upcoming week. Which would you prefer? (Circle one) **A** or **B**

Q3: Why?

**Part 2:** Below are the same weekly schedules from Part 1, **L** and **R**. However, now the events have been colored in the following way:

- **Gray** events represent existing commitments or plans made well in advance (classes, recurring meetings, club meetings, personal plans, etc.)
- **Green** events represent meetings scheduled more recently (in the last week or so) using a scheduling app. You sent the person you wanted to schedule a meeting with, and they selected a time and the meeting was scheduled automatically and added to your calendar.



**Q1:** Given the new information, do you have any observations or comments?

**Q2:** How would you describe the process of scheduling that you used last week to schedule several meetings? What were your thoughts (if any)?



**ADULT CONSENT  
FORM PRIVATE  
PRINCETON UNIVERSITY**

TITLE OF RESEARCH: An Alternative Approach to Scheduling

PRINCIPAL INVESTIGATOR: Jeremie Lumbroso

RESEARCH PERFORMED BY: Erin Vuong

PRINCIPAL INVESTIGATOR'S DEPARTMENT: Computer Science

**Key information about the study:**

Your informed consent is being sought for research. Participation in the research is voluntary. You may choose to withhold your consent now by not participating in the research. Likewise, you may revoke your consent at any time during the research by ceasing participation in the research.

The purpose of the research: Evaluate the effect of two different approaches to automated scheduling on user satisfaction with scheduled events.

The expected duration of the subject's participation: 1 hour over a period of 2 weeks

The procedures that the subject will be asked to follow in the research: Visit several scheduling links where they will pick an available time from the provided options and enter information provided to them by the researcher to simulate scheduling meetings. Look at several images of various weekly schedules and give their preferences and opinions.

The reasonably foreseeable risks or discomforts to the subject as a result of participation: None

The benefits to the subject or to others, e.g., society that may reasonably be expected from the research: No immediate benefits to others are expected.

The alternative procedures, if any, that might be advantageous to the subject: N/A

**Additional information about the study:**

**Confidentiality:**

All records from this study will be kept confidential. Your responses will be kept private, and we will not include any information that will make it possible to identify you in any report we might publish. Research records will be stored securely on encrypted password-protected cloud storage requiring authentication. The research team will be the only party that will have access to your data, and the person performing the research will be the only party that will have access to any personally identifiable data (names, email addresses, etc.) Personal identifiers will be removed from survey responses, and after such removal, the information may be used for future research studies performed by the same research team as an extension of this project without additional informed consent from the subject.

**Compensation:**

*If funding is received for compensation:* Subjects will receive <incentive here> for their participation in this study.

*If no funding is received for compensation:* Subjects will not be compensated for their participation in this study.

**Who to contact with questions:**

Please direct any questions for the researchers to Erin Vuong ([evuong@princeton.edu](mailto:evuong@princeton.edu)).

Principal investigator: Jeremie Lumbroso ([lumbroso@cs.princeton.edu](mailto:lumbroso@cs.princeton.edu))

If you have questions regarding your rights as a research subject, or if problems arise which you do not feel you can discuss with the Investigator, please contact the Institutional Review Board at:

Assistant Director, Research Integrity and Assurance

Phone: (609) 258-8543

Email: [irb@princeton.edu](mailto:irb@princeton.edu)

**Summary:**

I understand the information that was presented and that:

My participation is voluntary, and the alternative is to not participate.

Refusal to participate will involve no penalty or loss of benefits to which I am otherwise entitled.

I may discontinue participation at any time without penalty or loss of benefits.

I do not waive any legal rights or release Princeton University or its agents from liability for negligence.

I hereby give my consent to be the subject of the research.

Dear <name of party being contacted>,

My name is Erin Vuong and I am a junior in the COS department completing IW this semester. My project involves creating an alternative approach to the scheduling process used by apps you may be familiar with (Calendly, YouCanBookMe) in which you receive a link that allows you to pick an available time that works for you and automatically schedule a meeting without emailing back and forth. I have included a blurb below with more information if you are curious. As part of my project's evaluation, I would like to have some users who are willing to perform two very simple tasks: schedule several meetings using scheduling links I send to them, and then look at pictures of some weekly schedules and compare them/provide their opinions. This should take no more than an hour over two different days, about half an hour each time (and will likely take way less time than that).

*If funding is received for compensation:* As a token of my appreciation for completing the tasks, users will receive <incentive here>.

Please let me know if you are interested and I can provide you with more information. I will be very grateful if you are able to participate in my evaluation.  
Best wishes,  
Erin Vuong ('22)

Elevator speech: A variety of scheduling software already exists to help people arrange meetings (You Can Book Me, Calendly, WhenIsGood, etc.). Many of these options involve sending the person you are booking with a link to your available time slots and letting them select the option that works best for them. While this is an easy way to quickly schedule meetings without back-and-forth, not all available time slots are created equal. You may prefer to frontload your day or week. You may prefer to avoid switching back and forth between different tasks and meetings. Current scheduling solutions do not take these preferences into account. My goal is to create a scheduling app that more intelligently considers your time management preferences, and incentivizes the people scheduling with you to pick more preferred slots.