Schedgy Project Report

Ethan Lennon, Jay Rathore, Seattle Bourassa, James Reeve, Cliff Landry

GitHub Repository Link: https://github.com/GarbageCan622/Schedgy

Overview

Schedgy is a scheduling tool seamlessly integrated with Discord[1], an excellent tool for communication that has voice chat, instant messaging functionality, and the ability to create servers where users can collaborate. Discord has been rising in popularity over the past few years. Many software engineers use Discord because it can support video calls, screen sharing, and the ability to send code snippets. While these features are very appealing, it does not have functionality for scheduling meetings between users. This feature would be a valuable addition to Discord because many people already use it for calls/meetings. The scheduling tool will be set up in a way that it finds the time slots where all group members can attend a meeting.

Alternatives

Currently, there are websites to set up meetings such as When2Meet[2], the Apollo Discord bot[3], and Chronicle Bot[8] on Discord. Apollo allows a person to create an event with a timespan and send it out to a group of people where they can mark themselves as attending, declining, or tentative. Apollo is a valuable tool for larger-scale events rather than smaller groups where everyone must attend the meetings, as it has no functionality to poll users for the times they are available. When2Meet is different from Apollo as it's not a part of Discord and allows the creator of an event to choose a timeframe of dates and specific hours. The creator can then send out the link to their group members and select which time slots they're available. As the chart of times is filled out, users can see the overlap of available times and pick times when everyone or almost everyone can attend. When2Meet's main issue is that a new link is required every time for a recurring event. This isn't ideal if an individual's schedule is variable between meetings, as everyone has to redo the schedule every week. Since When2Meet isn't integrated into Discord there are no direct notifications to group members. This means that the event's organizer must check the polling chart of When2Meet manually, creating unnecessary work for them and creating the opportunity for human error. Finally, Chronicle Bot is also a popular

Discord scheduler, but its only function is integrating a personal Google calendar into Discord's in-built events. It does not support polling attendance like Apollo or offer any sort of coordination planning such as When2Meet.

Methodology

The primary goal of Schedgy is to streamline the process of finding meeting times and then scheduling the meeting, in a way that alternatives do not offer. While other scheduling applications do exist, these tools often have many different problems. The first problem that many scheduling apps have is that it does not communicate with the users when an event can occur. We can see this in When2Meet, where, even if all users are available for an event, the website has no means of communicating this with its users. Having to regularly monitor the schedule to see when an event is available is unappealing to many users, and Schedgy solves this with Discord integration.

By integrating with Discord, Schedgy simplifies the process of finding a time that works for everyone. The app sends notifications to the entire group when all members are available to meet, and provides the organizer with specific reminders and notifications before the event is to take place or if an event will still occur despite some participants being absent. In addition, Schedgy's Discord integration allows for the sending of polling and event messages to individual members or separate channels, making it easier for users to keep track of upcoming events most related to them.

Schedgy stands out from other Discord scheduling bots like Apollo by always attempting to offer the most feasible time when all members can attend the event, rather than relying on the organizer to hopefully have the greater insight to know others' availability or personal responsibility that may interfere. This is especially useful for small group meetings, where all members are expected to attend, but still applicable for larger events with a dedicated lead speaker where the organizer wants to maximize those who can attend.

One of Schedgy's distinct advantages is the ability to set recurring events. Upon initial setup of an event and users have individually specified their optimal times, the event can be set to meet on a set schedule utilizing the best times for everyone. This allows groups to quickly and effectively set up their needed collaboration time without the hassle of needing to remember to set the next daily, weekly, or monthly event ahead of time.

Any group looking for a time to meet up can use the application Schedgy if they already utilize Discord. Groups will be able to meet consistently with less effort than other applications by using the function to set available times, message users when an event is approaching, and schedule recurring events. Schedgy can be used by friends trying to catch up, friends looking for a time to meet, gaming groups looking for a time to play, and school projects looking for a good time to get together, along with countless other event ideas coming in all sizes.

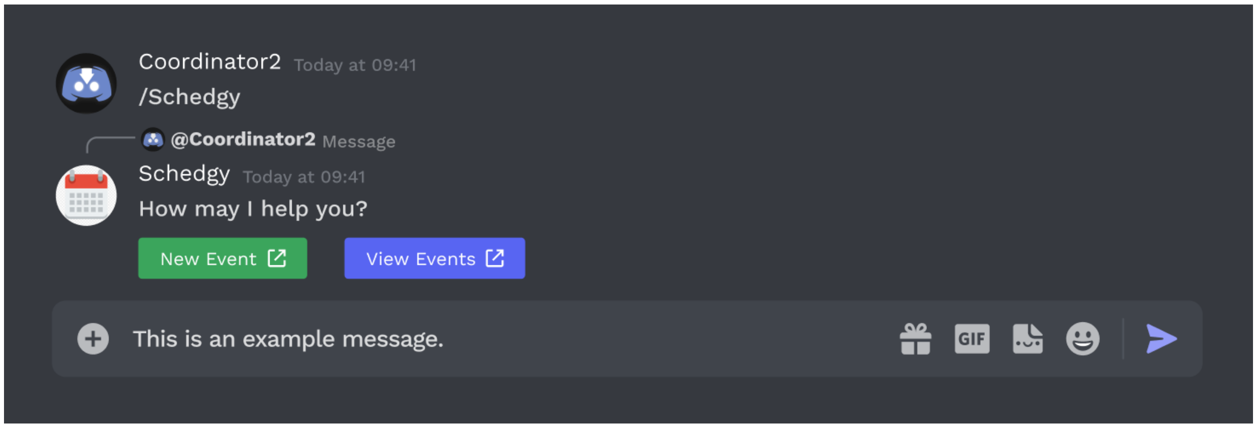
Event Creation Instructions

Instructions on how an event can be made by Schedgy:

1. The Event Coordinator, a person designated to create an event (it can be anyone in the Discord server), interacts with Schedgy (see Figure 1), and then clicks on “New Event” to create a new event.
2. The Event Coordinator gets directed to a website where they first have to sign in to Discord.
3. After signing in, the Event Coordinator can select the timeframe of their event. They can select either a range of dates, using a start and end date, or select the days of the week the event might take place during. The coordinator can then select a time frame in which a meeting can take place. An example of this step would be a coordinator choosing Monday to Friday with 9 AM as the earliest time and 9 PM being the latest time per day a meeting can take place. The website would then create a time interval between 9 AM and 9 PM for Monday to Friday with time slots that can be selected that are each an hour long (Users will be able to select 9 AM, 10 AM, 11 AM,...9 PM for their availability). The time frame also can’t be split up into multiple intervals. (See Figures 2 and 3 for more information) There are also additional parameters they can fill out, such as making the event repeat automatically or for Schedgy to try and automatically create a Discord event.
4. A notification with a link to a polling chart gets sent out to all specified members or roles within the server after the time frame has been confirmed by the Event Coordinator.
5. Each group member will need to sign in with Discord once redirected to the website
6. After signing in they can fill out the times they’re available (see Figure 3) on the calendar view on the left-hand side of the screen. Members can look at other members’ availability by looking at the calendar view on the right-hand side of the screen.

7. Once a number of users equal to or greater than the specified minimum number have filled out their availability, Schedgy will notify the creator of the event. If the creator has allowed Schedgy to attempt to automatically create a Discord event, it will do so. If the number of users available are available for long enough, a Discord event will be created, and the creator notified. If automatic creation fails, the creator will also be notified.

UI & Initial Results

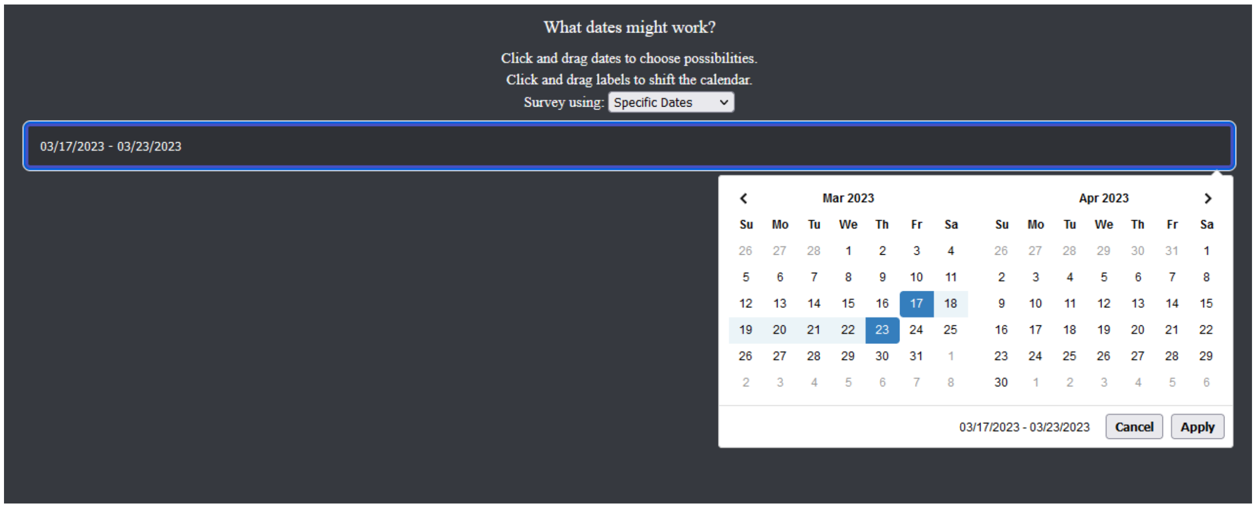


Coordinator2 Today at 09:41
/Schedgy

@Coordinator2 Message
Schedgy Today at 09:41
How may I help you?

Figure 1: Schedgy Discord Interaction

Figure 1 shows how a user can tag the Schedgy bot on Discord by using the command “/Schedgy”. This command leads to the bot giving the user the option of creating a new event or viewing other events in the group. The command can be performed by any member in the Discord server.



What dates might work?

Click and drag dates to choose possibilities.
Click and drag labels to shift the calendar.
Survey using: Specific Dates

03/17/2023 - 03/23/2023

Figure 2: Selecting a Range of Dates During Event Creation

Figure 2 is a representation of event creation. Here the user has chosen that a range of dates would be used for the polling chart. When selected, an interactive calendar appears, allowing the user to choose the start and end dates needed for the event.

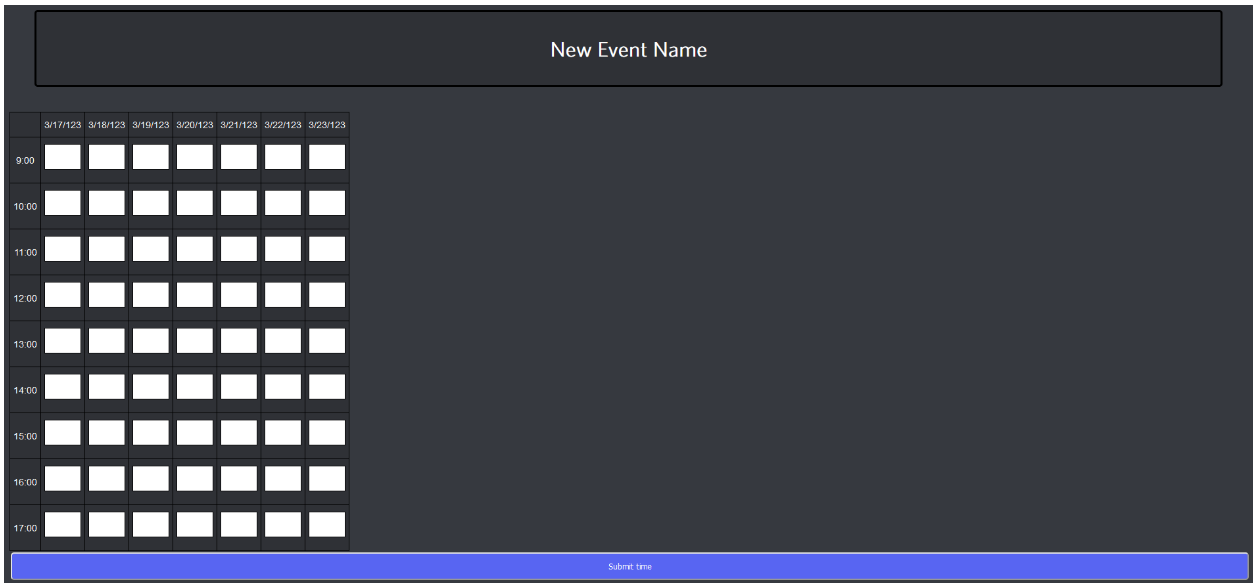
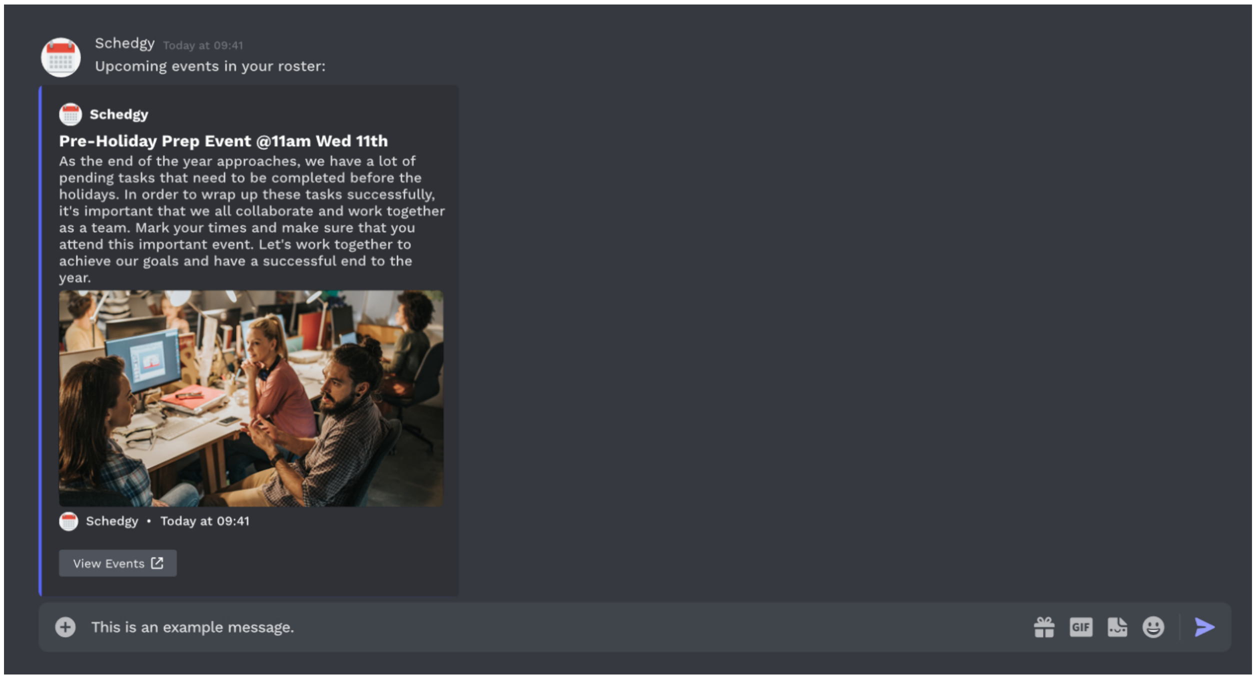


Figure 3: Select Availability

Figure 3 shows the main web page UI. This is from the view of any user accessing the link created by Schedgy. The top hosts the picture and name of the user viewing the window. The event name is below the user name. The calendar on the left is interactive so the user can select the time they want to show they are available. The calendar on the right shows the total count of all users that have responded and the times they responded to. The calendar that shows the availability of other members doesn't include the user currently filling out the polling until they click on the "Confirm button". A user will be able to see which user signed up for each slot by hovering over a time slot on the right-hand side calendar with their mouse. Above this calendar lists all members who have responded. A confirm button is located on the bottom right of the page for the user to verify their times for Schedgy.


Schedgy Today at 09:41
Upcoming events in your roster:

Schedgy
Pre-Holiday Prep Event @11am Wed 11th
As the end of the year approaches, we have a lot of pending tasks that need to be completed before the holidays. In order to wrap up these tasks successfully, it's important that we all collaborate and work together as a team. Mark your times and make sure that you attend this important event. Let's work together to achieve our goals and have a successful end to the year.

Schedgy • Today at 09:41

View Events

This is an example message.

Figure 4: View Upcoming Events

Figure 4 is an example of a Discord event Schedgy creates. In this case, it sends out a link that holds all your upcoming events/meetings within the server.

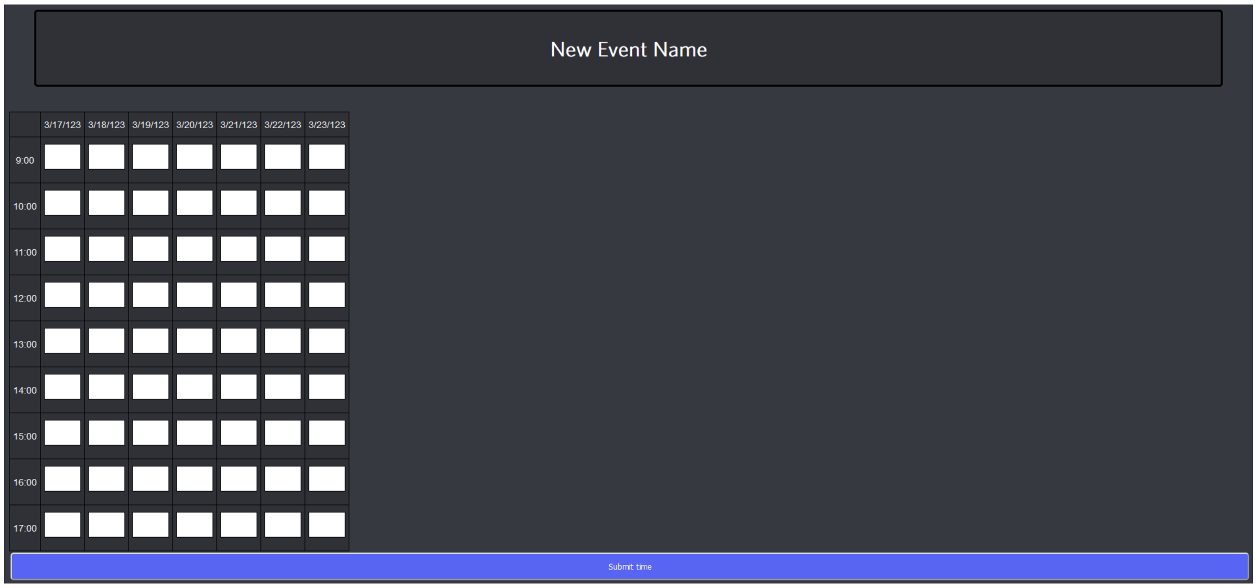
Figure Generation

Instructions on how to create a polling chart:

1. Add Schedgy to a Discord server you own using this link, if you do not have it already.
2. In a Discord channel Schedgy can access, type “/sendlink” to receive a link to Schedgy’s homepage.
3. From here you can log into Discord, which will give you access to the polling prototype page.
4. On the polling page, you then set a name for your event in the “New Event Name” textbox.
5. You can then choose whether you want a “Specific Date” event or a “Days of the Week” event.
    a. If “Specific Date” is chosen, click on the dates shown to pull up the calendar and select a date range for the event to occur
    b. If “Days of the Week” is chosen, click on the days of the week you intend to have the event occur on

6. In the "No earlier than:" and "No later than:" sections, enter the hour range in which you want the event to take place.
7. Click “Create Event”

You should now be presented with the event’s polling chart, displaying the times that would be filled out by users.

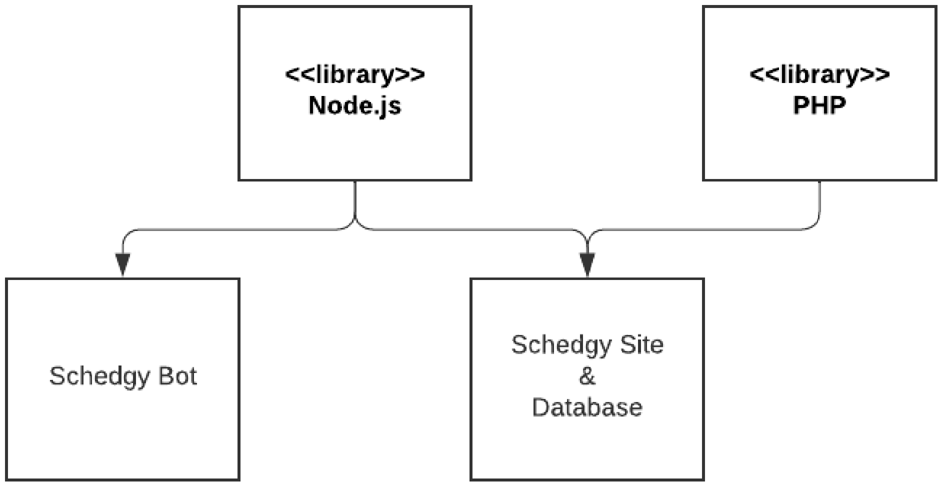


	3/17/123	3/18/123	3/19/123	3/20/123	3/21/123	3/22/123	3/23/123
9:00							
10:00							
11:00							
12:00							
13:00							
14:00							
15:00							
16:00							
17:00							

Figure 5: Example figure generation

Figure 5 is an example of a polling chart generated by the site. Because this is an early build it lacks some features and polish. The end goal of this page is to recreate our prototype diagram in Figure 3.

Technology

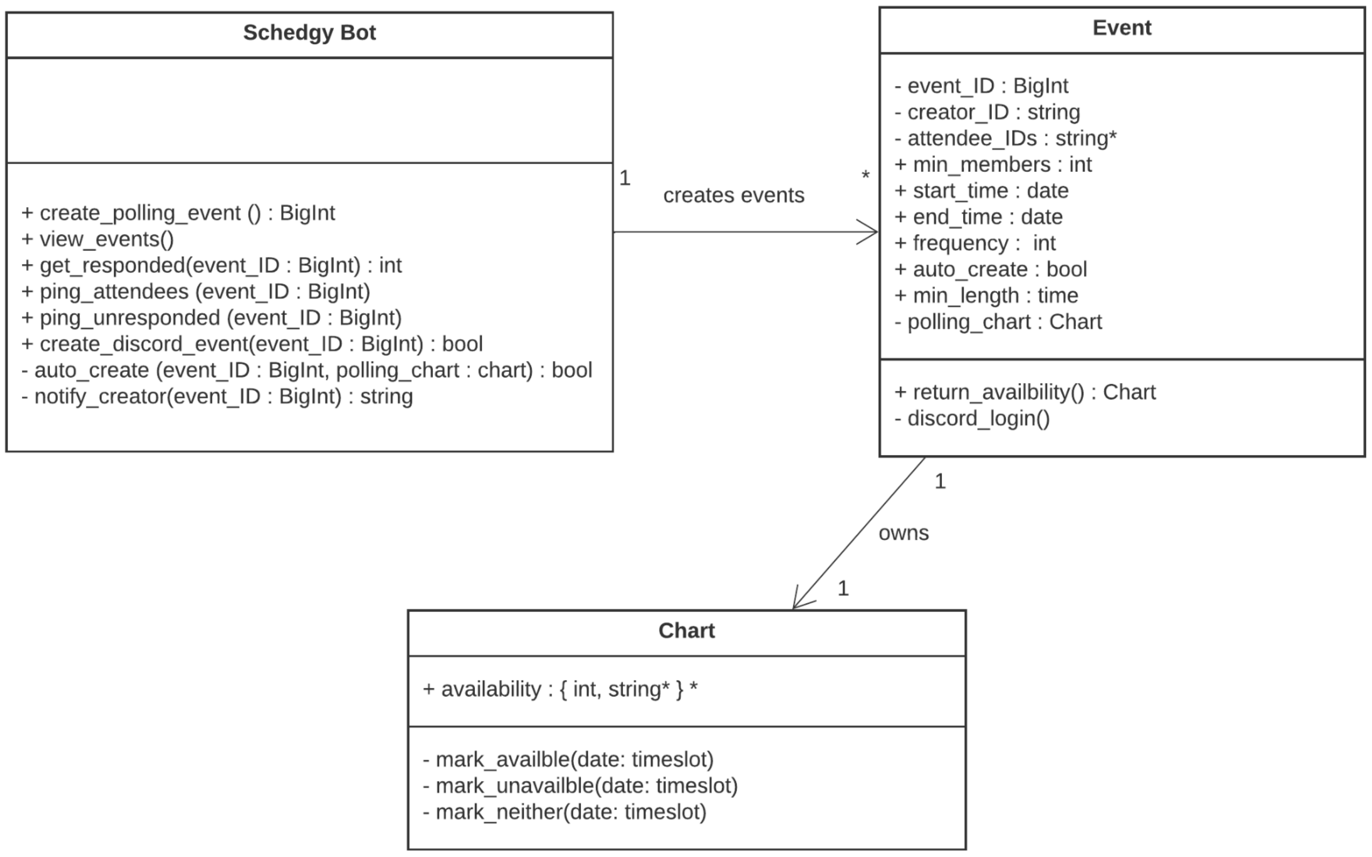


For the creation of the website, Schedgy utilizes Node.js [4]. This is a powerful JavaScript Runtime Environment for executing JavaScript code with a specialty in server-side programming and client-side programming. This feature set enables JavaScript to fulfill the great bulk of our project's architecture needs for the web interface and the Discord bot, minus the minor inclusion of PHP for the database storage of events. Node.js also allows us to integrate OAuth2 [6]. OAuth2 is used by Discord to allow access to the Discord API and Discord authorization and requires no additional libraries. Working with OAuth2, users will be able to log into the website using their Discord login information, have their Discord account linked to the website, and then allow the website to communicate with the Discord bot and issue commands to it.

A Discord bot must be registered through Discord's online interface to function on the platform, but the bot itself may implement the Discord API across multiple different environments. In an effort to keep the number of languages utilized throughout the project to a minimum, our group will be utilizing Node.js to build Schedgy. Node.js also has a library available to help with Discord API integration called Discord.js [5]. Discord.js allows access to the Discord API easier by using an object-oriented approach, enabling the creation of commands and other functionality of the bot. To host the bot we used SparkedHost[7]. SparkedHost is a server hosting service that specializes in hosting Discord bots and video game servers.

SparkedHost also offers to host MySQL[11] databases in addition to hosting the Discord bot, which the website utilizes to store meeting data.

Class Diagram


Schedgy Bot
+ create_polling_event () : BigInt
+ view_events()
+ get_responded(event_ID : BigInt) : int
+ ping_attendees (event_ID : BigInt)
+ ping_unresponded (event_ID : BigInt)
+ create_discord_event(event_ID : BigInt) : bool
- auto_create (event_ID : BigInt, polling_chart : chart) : bool
- notify_creator(event_ID : BigInt) : string
1
creates events
*
Event
- event_ID : BigInt
- creator_ID : string
- attendee_IDs : string*
+ min_members : int
+ start_time : date
+ end_time : date
+ frequency : int
+ auto_create : bool
+ min_length : time
- polling_chart : Chart
+ return_availbility() : Chart
- discord_login()
1
owns
1
Chart
+ availability : { int, string* } *
- mark_availble(date: timeslot)
- mark_unavailble(date: timeslot)
- mark_neither(date: timeslot)

Data Dictionary

Schedgy Bot	
Function	**Description**
create_polling_event() : BigInt	Schedgy creates a link to a webpage containing the event's polling chart and sends this link to the creator's direct messages. The function returns an internal ID used to differentiate events from each other. The ID is generated with Date.now(), which returns how many milliseconds have passed since 1/1/1970, which assures every ID is unique.

view_events()	Schedgy sends a direct message containing a link to the event, for every event the user has created.
get_responded(event_ID : BigInt) : int	Schedgy returns the number of attendees who have responded to the event given in its parameter, out of the number of expected members, as a reply to the command message calling this function.
ping_attendees(event_ID : BigInt)	Schedgy sends a message notifying all users in an event using Discord's @ feature. Only the creator of the event can use this command.
ping_unresponded(event_ID : BigInt)	Schedgy sends a message notifying all users in an event using Discord's "@" feature that have not responded to the polling chart yet. Only the creator of the event can use this command.
create_discord_event(event_ID: BigInt): bool	Creates a Discord event using the attributes from the associated event. Schedgy calls this function automatically when using auto_create, otherwise, it is called when the event's creator uses the associated command. Returns true if event creation succeeds, false if it fails.
auto_create(event_ID : BigInt, polling_chart : Chart) : bool	If the associate event is set to auto-create events, Schedgy will attempt to create a Discord event, using the event's availability chart, expected number of members, and event length attributes. Returns true if an event can be created, and false otherwise.
notify_creator(event_ID : BigInt) : string	This is called when certain conditions are met, such as when the polling chart has been responded to by all attendees, or when auto_create returns false. This returns a string with information about the event's current status, such as why the event creation failed..

Event	
Variables	**Description**
event_ID : BigInt	Unique internal ID of the event, used to differentiate it from other events. This is stored as a big integer, a primitive data type of Javascript, which can store larger non-decimal numerical values than the number type.
creator_ID : string	Unique Discord ID of the creator of this event.
attendee_IDs : string*	Unique Discord IDs of the attendees of this event.
min_members : int	Minimum number of attendees (including creator) needed for the event to take place. Used in auto event creation.
start_time : date	Stores the earliest date and time of day that the event can take place. This value must be less than end_time. The date data type contains values for year, month, days, and hours, as well as smaller units of time. For our purposes, only the four values mentioned by name are used to generate polling charts.
end_time : date	Stores the latest date and time of day that the event can take place. This value must be greater than start_time. The date data type contains values for year, month, days, and hours, as well as smaller units of time. For our purposes, only the four values mentioned by name are used to generate polling charts.
frequency : int	Determines the frequency of how often the event repeats, if at all. The integer value corresponds to a set frequency (ex. 1= weekly, 2 = biweekly…), with a value of 0 representing an event that does not repeat.
auto_create : bool	Determines if Schedgy will try to automatically create a Discord Event once all attendees have responded.

min_length : time	Minimum length of an event (ie. 1 hour, 2 hours…). Used in auto event creation, to assure events meet user’s needs.
polling_chart : Chart	An object representing the availability of attendees at different dates and times. The chart object is specified below.
Functions	**Description**
return_avilability() : Chart	Returns the polling_chart, used to send this information from the web page to Schedgy.
discord_login()	Uses OAuth to allow users to log in to the webpage with their Discord account.

Chart	
Variables	**Description**
availability :{int, string*}*	The polling chart and availability of different users is represented as an array of tuples. Each element of the array chronologically corresponds to a single time slot of the chart’s days and times. The first value in the tuple is an integer storing how many users are available for that time slot. The second value is an array of strings, storing the user’s Discord IDs so that who is available is known as well.
Functions	**Description**
mark_availble(date: timeslot)	Users mark themselves as available for the specified date and time slot. This is done using the webpages’ chart UI.
mark_unavailble(date: timeslot)	Users mark themselves as unavailable for the specified date and time slot. This is done using the webpages’ chart UI. When a user marks themselves as unavailable, the number of users available for that slot becomes -1.
mark_neither(date: timeslot)	Users mark themselves as neither available nor unavailable. This is the default state of timeslots and is used to signify a user isn’t unavailable, but may not be available.

Success, Risks, and Solutions

With the rise of remote work and the increased popularity of Discord as a platform for social and professional interaction on team projects, the need for an efficient scheduling tool is all the more in demand. With the success of Schedgy, we expect to have a significant impact on the way teams schedule events and meetings through Discord. With the current lineup of limited scheduling options available, like the previously mentioned Apollo bot and off-platform solutions such as When2Meet for event times, our tool provides a near-complete replacement for them as we fully integrate Discord's built-in event functionality and collaborative nature with the added ease of use with tools like When2Meet. Schedgy also expands upon what was previously available with the added ability to create recurring events and leverage the additional Discord features such as roles and user notifications. This means that teams will no longer have to depend on unintegrated third-party services, which can be time-consuming and inconvenient, by streamlining the scheduling process with Schedgy.

However, there certainly were risks associated with the project as we began our journey, particularly in terms of our team's inexperience in the creation of Discord bots and interaction with Discord’s API. The web-side client was done using JavaScript, HTML/CSS, and PHP, which our team members had varying degrees of experience with. Our dedicated team members who had prior experience with these languages assisted in getting the group up to speed and delegated specialized tasks to those most capable. From this, we were confident that the risks could be mitigated with this collaboration and team learning. From the current successful status of our project, it has become self-evident that our confidence was not unfounded.

In terms of architecture, the project relies on several external factors to support our goal. Discord, of course, is the backbone of our project and without the continued support of the Discord service, our main point of user interaction would no longer be functional. As it is unlikely Discord will be closing its doors entirely, the primary risk of developing on its platform is the API remaining open and backward compatible with developed applications. So far Discord has had a pretty good track record of maintaining compatibility with bots and services developed on its platform only killing off some API features related to music streaming. This shines a good light on the lifespan of Schedgy as it is unlikely that Discord would remove or deprecate support for Schedgy in the future. This also meant that when we planned and began creating Schedgy it was very unlikely that we would need to change our implementation plans based on changes from Discord.

In addition, the hosting of our platform on SparkedHost[7] is a point of concern as we are at the whim of their service options and closure, but due to the standard format of Discord Bots and our development in JavaScript, we should have no issues migrating platforms. Alternatives to SparkedHost include but are not limited to Zluqe[9] and BOTSHARD[10], both being free bot-hosting platforms. This provides a simple mitigation for any potential risks that could be encountered in the long term with SparkedHost as migration can be accomplished with minimal impact to costs or performance.

With the team's success bringing Schedgy to its full functionality, we see this project having the potential to gather wide adoption by Discord communities around the world wide if deployed as an official bot. As Schedgy addresses some of the significant pain points in the scheduling process shared by small friend groups and professional team projects alike. Additionally, considering the team's implementation of Schedgy with its time-saving and hassle-free features, our platform has the potential to generate revenue through the future development of premium features. Despite the challenges faced by some of the risks associated with our project, we do see the potential benefits of a successful launch far outweighing any risks to the development process.

Spending

Schedgy being a Discord bot means that all the resources needed to build it have come from the internet meaning that we've spent no additional money on it. Since we're using a web page alongside the Schedgy bot we spent $1 per month for Sparked Host. The $1/month option provided adequate storage space for our application.

Testing

Throughout development, our team has created tests to analyze the success of our work. We tested our webpage's polling chart using a mock object of user inputting dates for an event. The Javascript takes the object that we pass and creates a chart based on the dates and times. We tested this to ensure that our main user interface works the way it was intended, along with keeping usability high.

We have also created tests for HTML. These tests check to make sure that each page renders a heading element or page title. We wanted to ensure that none of our pages were blank and eliminated user confusion. It would be unfortunate if a user did not know what page of the website they were accessing at the current moment.

By implementing large checks spanning multiple cooperating functionalities in midterm and final tests we were able to identify any potential integration issues early, before concluding a module is complete, even if it passes all individual tests.

Feedback

- Free alternatives to SparkedHost were suggested, however, their services were lacking in terms of storage when compared to SparkedHost. As such, even though SparkedHost requires payment, it better suits our needs.
- With the understandable concern over the potential risk for malicious actors to utilize Schedgy to spam users with events and notifications, it is worth noting that Discord's own in-server moderation tools are fully capable of limiting access to the bot commands to specific users/roles. This allows for mitigation to the spam concerns by standardized means.

Lessons Learned

This semester taught us the importance of creating a full plan and a full schedule for development. This created weekly goals for us and allowed others on the team to know what everyone else should be doing. Having a schedule also created points to talk about during our meetings as having a list of tasks and rough estimates of how long the tasks should take allowed us to know who needed help and who could work on their own. We did not stay entirely on schedule, however, it served as a great guideline and reminder for future work.

More research should have been done about the technologies we chose to use to avoid misconceptions and assumptions. For example, while Discord.js provided the functionality we needed, mocking the library to use in unit tests was not something we could set up. This is largely due to recent changes to Discord’s API, and a lack of up-to-date resources to aid in test-driven development.

This semester taught us about the importance of dividing labor and creating different modules of our code. By dividing the team into 2 members working on the discord bot, 2 members creating the webpage, and one member helping with both, we were able to efficiently create the project. This led to everyone being able to focus on their part of the project and not have their attention divided.

References

[1] Discord, “Discord — A New Way to Chat with Friends & Communities,” Discord, 2022. https://discord.com/

[2] “Apollo Discord Bot,” Apollo Discord Bot. https://apollo.fyi/

[3] “When2meet,” www.when2meet.com. https://www.when2meet.com/?About

[4] Node.js Foundation, “Node.js,” Node.js, 2019. https://nodejs.org/en/

[5] “Discord.js,” discord.js.org. https://discord.js.org/#/

[6] “OAuth 2.0 — OAuth,” Oauth.net, 2020. https://oauth.net/2/

[7] “Sparked Host - Minecraft Server Hosting,” Sparked Host — Minecraft Server Hosting. https://sparkedhost.com/

[8] “ChronicleBot,” Discord Bots. https://discord.bots.gg/bots/903380664336928798

[9] “Zluqe | Free Bot Hosting,” zluqe.com. https://zluqe.com/

[10] “BOTSHARD | FREE DISCORD BOT HOSTING,” botshard.com. https://botshard.com/

[11] “MySQL,” MySQL. https://www.mysql.com/

Appendix

Member Responsibilities

- Discord Bot / Discord Integration
  - Ethan
  - Jay
- Webpage
  - Cliff
  - Seattle
- Communication between bot and webpage / Documents
  - James

Weekly Schedule

Week 7 - 02/26

- Webpage with Discord login functionality complete and tested
- Bot functionality of sending the polling webpage link and notifications to Discord channels/user DMs complete and tested

Week 8 - 03/05

- UI of polling chart implemented into the webpage
  - Chart will be fillable by users with access to the link
  - Chart will display other users' availability to all users
  - Chart will use placeholder values for dates and times

Week 9 - 03/12

- Testing of page creation and polling chart fill-ability complete

Week 10 - 03/19

- Functionality to allow time frame selection complete and tested
  - Timeframe would be either days of the week or calendar date(s), as well as times of day
  - Replaces placeholder values on the webpage

Week 11 - 03/26

- Repeating event functionality tested and complete

    - During event poll creation, the creator selects if the event is repeating or one off, as well as how often and when it repeats
    - Event would reuse previous event's availability times but allow users to modify their availability at any time

Week 12 - 04/02

- Automatic Discord event creation completed and tested
    - During polling event creation, the creator can set conditions, that if met, enable the bot to automatically create a Discord event
    - Creator will be alerted if an event could not be created, as well as what conditions weren't met

Week 13 & 14 - 04/09 & 04/16

- Testing of bot and webpage functionality and integrity up until the submission date