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. Discord is popular among software engineers because of its support for 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 since many people already use it for calls and messages. 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 where not everyone is required to attend meetings, as it does not have the functionality to poll users for their availability. Apollo not being able to poll users for times they’re available is a huge negative because the ability to poll users helps streamline the scheduling process, increase activity and engagement, and reduce scheduling conflicts and cancellations. Knowing the availability of users makes it easier to find the perfect time to meet, as you only need to find a time when every or most members can attend. 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 for every recurring event. This may not be ideal if an individual’s schedule varies between
meetings, as everyone has to fill out the schedule again 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 the link which sends them to the Schedgy website.
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 navigate to the Event Creation page(see Figure 2). The Event Coordinator will then enter custom event details, including the name, id, and description.
4. From there, 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
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.

5. In the Event Coordinator’s Homepage the event details will populate the table on the left (see Figure 3). The Event Coordinator can then add any other Member to the event by clicking the Send Invite button (see Figure 4).

6. Each group member will need to sign in with Discord once redirected to the website.

7. After signing in they will see that they have been added to an event.

8. From here the Member will navigate through the View Event button to select their available times (see Figures 4 and 5). The Member will click “Submit Time” once they have selected available times, then the Member will click “Confirm Submission” to add their times to the database.

9. Once a number of users equal to or greater than the specified minimum number have filled out their availability, a user can use the command “/finalizeevent” (see Figure 6) to send out what time a meeting will take place based on the overlap of every users’ availability. If automatic creation fails due to there not being a good time overlap to meet or some to other error, the creator will be notified.

UI & Final Results

Figure 1 shows how a user can use the command “/createevent” to get a link to the Schedgy website. On the website the user will first be prompted to sign in using Discord and then they can either view or create events on the homepage.
Figure 2: Selecting a Range of Dates During Event Creation

Figure 2 is a representation of event creation. The Event Coordinator will navigate to this page from their Schedgy Homepage. Here, the Event Coordinator will fill out all boxes required, and click “Create Event” to create the event.
Figure 3: Event Invitation

This is the page an Event Coordinator can navigate to in order to add people to the event they created. The Event Coordinator simply enters the Event’s unique ID and the ID of the user they wish to add to their event. From here the invited users home page will then populate the new event and allow them to input their availability.

Figure 4: Schedgy Homepage

Figure 4 shows the homepage of the Schedgy website. On the homepage, the user can view the events they’ve created on the left side of their screen, and events that they’ve been invited to are displayed on the right side of the screen. From here a user can also create a new event using the “Create New Event” button, view events using the “View Events” button, and send out invites to an event to other users using the “Send Invite” button.
Figure 5: Filling Out Availability

Figure 5 shows a user filling out their time availability, in this scenario there are only 4 possible time slots a user can pick from. A user can click on a white box shown on the matrix at the left side of the screen if they are available at that time. The matrix on the right side of the screen shows how many users can attend a time slot, the example shows that currently, no one can show up for 6:00 AM or 7:00 AM meetings on May 15 or May 17 of this year. Once a user selects their time they can click the giant “Submit Time” button on the bottom of the screen to confirm their availability.
Figure 6: Finalize Event Bot Command

Figure 6 shows a user using the bot command “/finalizeevent” to see what time the meeting will take place. The time and date of this meeting are determined by the overlap of users’ availability selection. If members have still not responded to the event with their availability the following message will be sent instead: “Some members of your event have not yet responded. You can ping them with /pingunresponded to let them know”. If no members have times that line up the following message will be sent instead: “No members of your event have times that line up.”

Figure Generation

Instructions on how to create a polling chart:

1. Download the Schedgy source code from
   https://github.com/GarbageCan622/Schedgy/tree/main
2. Unzip the folder and rename the folder to "Schedgy"
3. Go to https://www.apachefriends.org/ to download Xampp for your appropriate system
4. Run the Xampp installer, downloading all components when you reach the "Select Components" window
5. Finish the setup and run Xampp
6. Locate the newly added "Xampp" folder in file explorer
7. Navigate to the "htdocs" subfolder inside the "Xampp" folder
8. In "htdocs" drag and drop the folder containing the Schedgy source code
9. In the Xampp application under Actions click Start on Apache and MySQL
10. In your browser, type "localhost/Schedgy/SQL/index.php"
11. Create an account by entering an id and a username
12. Enter the id for the account just created and login
13. Click on Create New Event
14. Type in an event name, event id, and description for the event
15. Click on the date range under "Select Specific Dates" to pick the dates for the event and select a range of times for the event under "What times might work?"
16. Click the create event button then click on event homepage to return to the homepage menu
17. Click view events
18. You can now see the event. On the left side of the screen is the personal availability chart where the user can enter their own availability, on the right side is the group availability.

You should now be presented with the event’s polling chart, displaying the times that would be filled out by users. To see an example of the polling chart generated, reference figure 4.

Technology

Figure 6: Overview of how Libraries are used in the Architecture

Figure 6 shows how the architecture of Schedgy works. The Node.js library is used for both the Schedgy Bot and the website while the PHP library is used only for the website.

The Schedgy website uses 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 used throughout the project to a minimum, our group used 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 uses to store meeting data.

Class Diagram
## Schedgy Bot

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>create_event() : BigInt</td>
<td>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.</td>
</tr>
<tr>
<td>view_events()</td>
<td>Schedgy sends a message listing all the names and ID’s of all events you own.</td>
</tr>
<tr>
<td>ping_attendees(event_ID : BigInt)</td>
<td>Schedgy sends a message notifying all users in an event using Discord’s @ feature. Only the creator of the event can use this command.</td>
</tr>
<tr>
<td>ping_unresponded(event_ID : BigInt)</td>
<td>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.</td>
</tr>
<tr>
<td>finalize_event(event_ID: BigInt): bool</td>
<td>Schedgy sends a message containing what time and date your event can take place, using the longest length of time where every member of your event is available. If there are members of your event who have not responded it will say in the message. The message will also say if no time where all members are available is found. The message also reacts with a thumbs up and down, to allow your members to confirm their attendance one last time.</td>
</tr>
<tr>
<td>Event Property</td>
<td>Description</td>
</tr>
<tr>
<td>----------------</td>
<td>-------------</td>
</tr>
<tr>
<td>event_ID : BigInt</td>
<td>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.</td>
</tr>
<tr>
<td>creator_ID : string</td>
<td>Unique Discord ID of the creator of this event.</td>
</tr>
<tr>
<td>attendee_IDs : string*</td>
<td>Unique Discord IDs of the attendees of this event.</td>
</tr>
<tr>
<td>start_time : date</td>
<td>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.</td>
</tr>
<tr>
<td>end_time : date</td>
<td>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.</td>
</tr>
<tr>
<td>polling_chart : Chart</td>
<td>An object representing the availability of attendees at different dates and times. The chart object is specified below.</td>
</tr>
</tbody>
</table>

**Functions**

- **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**

<table>
<thead>
<tr>
<th>Chart Variables</th>
<th>Description</th>
</tr>
</thead>
</table>
| 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.

<table>
<thead>
<tr>
<th>Functions</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mark_available(date: timeslot)</td>
<td>Users mark themselves as available for the specified date and time slot. This is done using the webpages’ chart UI.</td>
</tr>
<tr>
<td>mark_unavailable(date: timeslot)</td>
<td>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.</td>
</tr>
</tbody>
</table>

**Project Results**

During the semester, our team successfully created a functional website with Schedgy that enables users to create, view, and join events, invite others, and view their invitations. Our Discord bot has been set up with commands that allow users to send the link to the Schedgy homepage, ping attendees to a meeting that they own (required a meeting ID), list event IDs, list the people involved in an event, finalize an event once everyone has responded, and check who has responded to a meeting. Most of our planned objectives for the semester were achieved successfully. However, we faced some setbacks due to our hosting service’s inability to support server-side programming. As a result, our complete project can only be run locally as of now. We were also unable to send notifications automatically to a Discord server based on if certain events happened on the website such as an event being created. We plan to incorporate these features in the future using Discord Webhooks [12], which is a technology that enables automated messages to be sent with a bot to a Discord server when an event occurs on a website. Aside from those setbacks, we were able to create a scheduling system that’s seamlessly integrated with Discord.
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.

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


Appendix

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 use 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.

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