Business Proposal

for

Discord Calendar/Reminder System

Version 1.0 approved

Prepared by Akshay Gopalakrishnan, Nguyen Vi Cao, Sovanna Yoo, Skyla Tran, Adam Chhor

CSS 370 B

June 5, 2023

Table of Contents

Executive Summary

As of the current version of Discord (as of the date this document is released), there is no unified way for members of Discord servers and/or groups to plan or create events. This feature, the Discord Calendar/Reminder System, provides a service to users that increases their user experience with the platform by introducing a new, useful feature.

The purpose of the Discord Calendar/Reminder System is to create an additional feature for Discord users. The Discord Calendar/Reminder System is able to be adopted and integrated into both Discord users' accounts and owned servers, providing a calendar interface and functionality such as scheduling events and viewing upcoming events.

The Discord Calendar/Reminder System has two main viewings for Discord users to interact with: the personal calendar and the owned/joined Discord server's calendar. Both calendar views display a typical calendar interface, with current and upcoming events scheduled on specified days. Events can be created by Discord users and viewed on both the Discord Server's calendar and also their personal calendars. Discord server calendars display the events of the Discord server, while personal calendars display all of the events the Discord user has "joined" across all Discord servers they are a part of.

Events created within a Discord server are displayed on the Discord server's calendar. These events, server events, are displayed to all current members of the Discord server. Discord server members are able to view the title, description, time, and date of the event which was specified by the Discord user who created the event. The displayed event also has a button/option to "join" the event. These server events can be added to a Discord user's personal calendar by joining the event through that previously mentioned button. Events are also able to be deleted by the Discord user who created the event, or "un-joined" through a button similar to the "RSVP" one. In addition to creating, joining, and removing events from Discord server calendars and Discord user personal calendars, a reminder can be sent out to Discord users when an event they have joined is coming up (both server and personal events).

In terms of desirability, the Discord Calander/Reminder System satisfies Discord users' desires in having an intuitive and efficient way of sharing events. Many times, Discord servers are used to facilitate a variety of groups outside of the Discord platform such as university courses, friend groups, and professional teams. This means Discord users often desire the ability to create and share events pertaining to group meetings for a variety of purposes.

In terms of feasibility, there are a variety of resources and systems similar to the Discord Calander/Reminder System which can be utilized during development. The Discord Calander/Reminder

System remains viable through user donations and membership plans. Free users are offered limited use of resources such as a limited number of published events at a time and a limited number of RSVPs to events. To gain access to unlimited features, Discord users can pay for membership.

The development cost is ~$20,000 and takes 4 months. With other similar projects reaching profitability and the team's knowledge of Discord, we believe that the feasibility of the Calendar/Reminder feature is achievable.

Detailed Project Proposal

Purpose

The purpose is to create a Calendar/Reminder API that connects with Discord. The Calendar/Reminder feature allows users to have a physical calendar to add, edit, and remove events. There is also a reminder feature to send out notifications to the users based on the calendar.

Document Conventions

The standard convention for this document is that the main body font is in Times New Roman 11-point font. All headers are bolded and adhere to the following sizing: H1 is 32-point font, H2 is 16-point font, and H3 is 14-point font. For requirements, we followed the MoSCoW prioritization convention.

Intended Audience and Reading Suggestions

This document intends to provide a design overview of the system architecture and modules representing the Calendar/Reminder API and its uses. This document is intended for individuals and groups looking for a high-level understanding of the product system. This includes:

- Stakeholders
- Project Managers,
- Higher-Ups
- Developers
- Discord users

Project Justification

Justifying Projects

Desirability

Certain customers are targeted; therefore, we are focusing on having an extremely smooth and pleasant end-to-end experience. Users like a smooth end-to-end experience.
Ex: Clubs would be able to notify members when the next general meeting is easily and clearly by using this feature. This would enable club administrators to keep using this feature for future meetings and events.

Feasibility

When knowing our customer's expectations and goals for the product/feature (calendar reminder) we can easily adapt only focusing on those aspects of the feature without having to add more unnecessary things.

Viability

We can have a standard version of our calendar available for free. The more advanced version which includes automatically detecting events and adding it to the calendar requires fees (or already included if subscribed to Discord Nitro). If we can make the feature more advanced or more desirable and charge for that version, users are then more willing to pay to use the feature.
Ex: Whenever the Professor sends out an email about a class meeting, the calendar can automatically detect it via email and add the event on the Discord calendar.

Quantifying "cost" and "benefit"

Hassle

A hassle for teachers and professors on Discord is that they want an easy way to organize office hours so that their students know when they can meet. They do not have a way of doing that, and it makes them frustrated because their students are missing out on getting help for important assignments and projects. As a result, students get lower scores, making both teachers and students frustrated.

Net Benefit

The main forms of capital that teachers and professors get from the calendar/reminder system are social and intellectual capital. With a more organized way of keeping track of events, more students will come to office hours. Before, on average per 100 students, only 20 students would come to office hours. However, after the software is implemented, the number of students come to office hours is 60. This is an increase of 40 students.

In terms of intellectual capital, student grades improve. Before the feature is implemented, the average grade is 80%. However, after the feature is implemented, the average grade improves to 86%. An improvement of 6% in the grades impresses school administrators and superintendents.

Aggregate Impact

The improvement would vary from school to school and college to college, because they have different systems and scheduling. Dividing the scheduling between semester and quarterly, for a semester-based system the improvement in students attending office hours for a teacher is 40 students * 3 classes * 2 semesters in a year. This equates to an improvement of 240 students attending office hours. For the quarterly system the improvement in students attending office hours for a teacher is 40 students * 3 classes * 4 quarters in year. This equates to an improvement of 480 students attending office hours.

For grade improvement, the improvement cost is based upon the scheduling system. With an increase of 6%, the number of students that pass with a 3.0 should increase as well. On average a professor has 30 students in class, with about 60% passing with a 3.0 or above. This means that 18 students are passing with a 3.0. If there is a 6% increase in the average grades, then that means that the number of students with 3.0 or above increases by around 4 students. For a semester-based schedule, 4 students * 3 class * 2 semesters = 24 more students passing with a 3.0 in a year. In terms of a quarterly based schedule, 4 students * 3 classes * 4 quarters = 48 more students passing with a 3.0 in a year.

Range of Estimates

Figure 1: Range of Students Coming to Office Hours for Each Schedule Type in a Year

Schedule Type	Conservative Estimate	Planned Estimate	Optimistic Estimate
Semester	120 Students	240 Students	720 Students
Quarterly	240 Students	480 Students	1440 Students

Figure 2: Range of Students Passing with a 3.0 for Each Schedule Type in a Year

Schedule Type	Conservative Estimate	Planned Estimate	Optimistic Estimate
Semester	12 Students	24 Students	72 Students
Quarterly	24 Students	48 Students	144 Students

Tangible versus Intangible Benefits

Figure 3: Tangible and Intangible Benefits

Tangible	Intangible
● More students attending office hours ● Students' grades improve	● Increased organization of events ● Increased usability for various types of users on a server

Cost Estimation

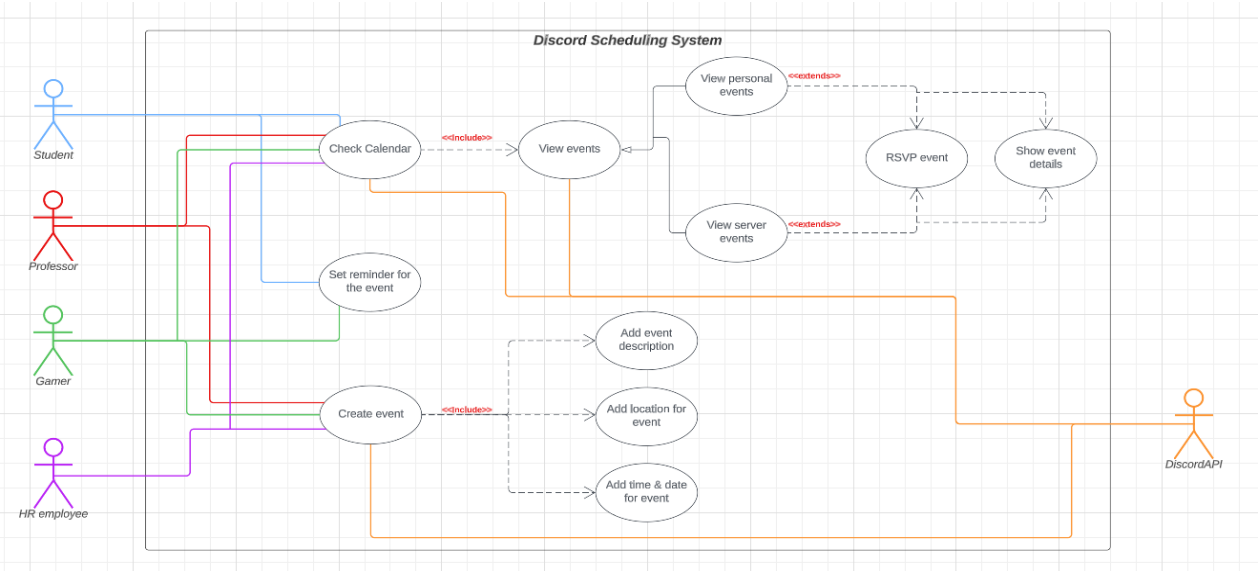
Methods for Software Development Estimation

Basic Methodology

We expect around 1000 LOC which includes class and object design, database setup, and API setup. We also expect our project to have 50 functions to complete the task. The object classes include Event, Reminder List, Calendar, Database, API.

To help determine the important parts of the feature we create the use case diagram shown in Figure 4.

Figure 4: Discord Calendar/Reminder System Feature Use Case Diagram



We then use the use case diagram to create the user stories shown in Figure 5, which helps determine the system requirements. Each user story has points. The higher the points, the more relevant the user stories are to our target product.

Figure 5: User stories

ID	User Story	Story Points	INVEST Critique
1.1.1	As a Student, I want to see what events are on my calendar, so that I know what my schedule looks like.	10	• **I:** This user story is independent • **N:** Negotiable within boundaries • **V:** Valuable to users • **E:** Estimable • **S:** Small enough to be completed within a sprint • **T:** Easily testable
1.1.2	As a student, I want to schedule events, so that I can notify my peers about study groups and get-togethers	10	• **I:** This user story is independent • **N:** Negotiable within boundaries • **V:** Valuable to users • **E:** Estimable • **S:** Small enough to be completed within a sprint • **T:** Easily testable
1.1.3	As a club moderator, I want to edit the event time and location for the offline event on the channel's calendar so that everyone in the channel can see it.	10	• **I:** This user story is independent • **N:** Negotiable within boundaries • **V:** Valuable to users • **E:** Estimable • **S:** Small enough to be completed within a sprint • **T:** Easily testable
1.1.4	As a Gamer, I want to be notified before the event starts, so that I do not miss it.	10	• **I:** This user story is independent • **N:** Negotiable within boundaries • **V:** Valuable to users • **E:** Estimable • **S:** Small enough to be completed within a sprint • **T:** Easily testable
1.1.5	As a professor, I want to be able to mark my office hours on my class schedule so that my students can see my available time slot.	7	• **I:** This user story is independent • **N:** Negotiable within boundaries • **V:** Valuable to users • **E:** Estimable • **S:** Small enough to be completed within a sprint • **T:** Easily testable

ID	User Story	Story Points	INVEST Critique
1.1.6	As an HR employee, I want to view who is coming to office parties so that I can plan and prepare the itinerary.	5	• **I:** This user story is independent • **N:** Negotiable within boundaries • **V:** Valuable to users • **E:** Estimable • **S:** Small enough to be completed within a sprint • **T:** Easily testable
1.1.7	As an HR employee, I want to cross off events in my list and calendar, so that I can order my events in priority.	5	• **I:** This user story is independent • **N:** Negotiable within boundaries • **V:** Valuable to users • **E:** Estimable • **S:** Small enough to be completed within a sprint • **T:** Easily testable
1.1.8	As a Manager, I want to be able to assign my employees to a meeting and have it added to their calendar so that they know it is required for them to come.	5	• **I:** This user story is independent • **N:** Negotiable within boundaries • **V:** Valuable to users • **E:** Estimable • **S:** Small enough to be completed within a sprint • **T:** Easily testable

Risk Factors Used to Determine Project Size

New Factors

Working with Discord's API, connecting backend development, Discord's API, as well as the Database are new technologies aspects that our team needs to overcome. Upon that, our team has just been formed recently so the risk of a new team chemistry is also worth noticing.

We are integrating a Calendar feature to Discord which aims to enhance Discord's users experience in the convenient wise. However, Calendar feature is not a new feature so there are lots of sources that we could refer to for our feature; hence, the subject domain is not a big concern for the team.

New software engineering methods are a medium concern for our team. Since each member in the team are all Entry-level in the field. However, that is also an advantage since we can apply what we have learned in school as well as being more open to new concepts and engineering methods.

Constraints

All members in the team are full time students so the tight schedule is a worth noticing constraint. Since we work on a tight to no budget. The budget constraint should also be notified. We do not need to worry about legacy technology or infrastructure because we plan on building our new database, API, classes, and functions.

We believe that the team size (5 members) is perfect for the size of this project. Each member specializes in a specific aspect of the project which benefits team productivity and capabilities. The team members entered this project with the same goal. Hence, we believe that we have full commitment to the project.

Project Factors

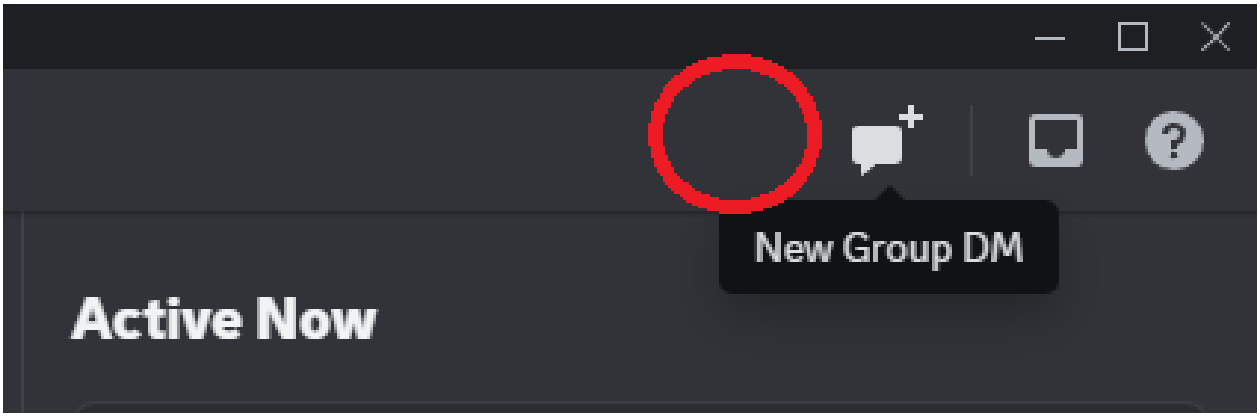
For stability aspect, we aim to build our feature to be able to handle:

- The system should have an uptime of at least 99.999%.
- The system should have less than the rate of occurrence of failure of 1/5000.
- The system could be able to handle 100,000 requests at once.

We expect only one UI to be implemented, which is the drop-down view of the Calendar. However, the button is placed in two different scenarios:

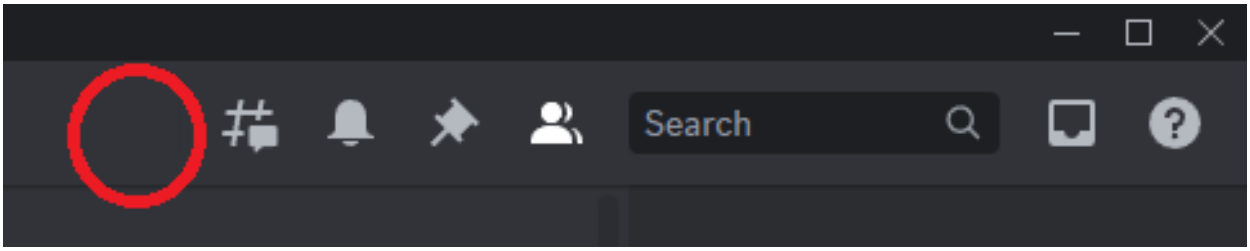
- For individual calendar usage, a calendar icon button is placed next to the “New Group DM” button located on the top right of the user home page of Discord. Figure 6 depicts this location.

Figure 6: **Location of the Individual Calendar Icon Button**



- For server/group calendar usage, a calendar icon button is placed next to the “Threads” button located on the top right of the server channel of Discord. Figure 7 depicts this location.

Figure 7: **Location of the Server Calendar Icon**



The drop-down view of the Discord Calendar/Reminder System's calendar is inspired by the interface illustrated in Figure 8. Note the hourly breakdown of the days which can be used to show specifically what time events are occurring. For each event, a bot sends out a reminder a defined time before the event occurs.

Figure 8: **Google Calendar Interface Inspires the Discord Calendar/Reminder System Calendar**

Domain Factors

The domain that has to deal with the most crucial regulation is our database. It must secure the users' information from attackers as well as provide service up to 10,000 requests at the same time without being interrupted.

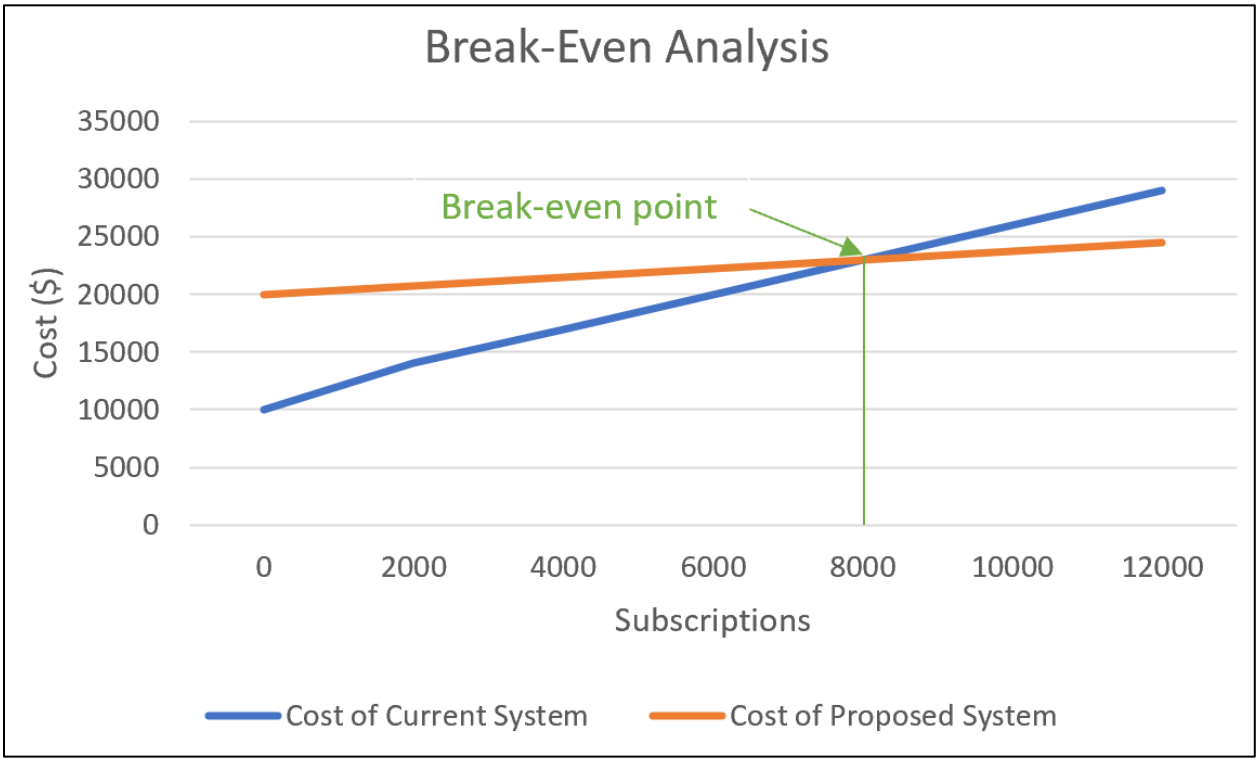
Project Size and Team Effort

We expect this project to take about 4 months to be accomplished and each team member is required to work around 20 hrs/week. The salary for the team is $2,240/month ($28/hr). The hardware cost, facility cost, operating cost, and supply cost are estimated around $2,760/month. This brings the total to $5,000/month. This is $20,000 for the entire project.

Formal Economic Analysis / Cost-Benefit Analysis

Break-Even Analysis

Figure 9: Break-Even Analysis Graph



Description

The cost of the current system overtakes the cost of the proposed system after 8000 subscriptions are made. This point is called the break-even point and shows how the proposed system is worth it. There is an initial cost that needs to be paid, but the initial cost is worth it in the future in terms of profits and money made.

Payback Analysis

Figure 10: Payback Analysis Graph

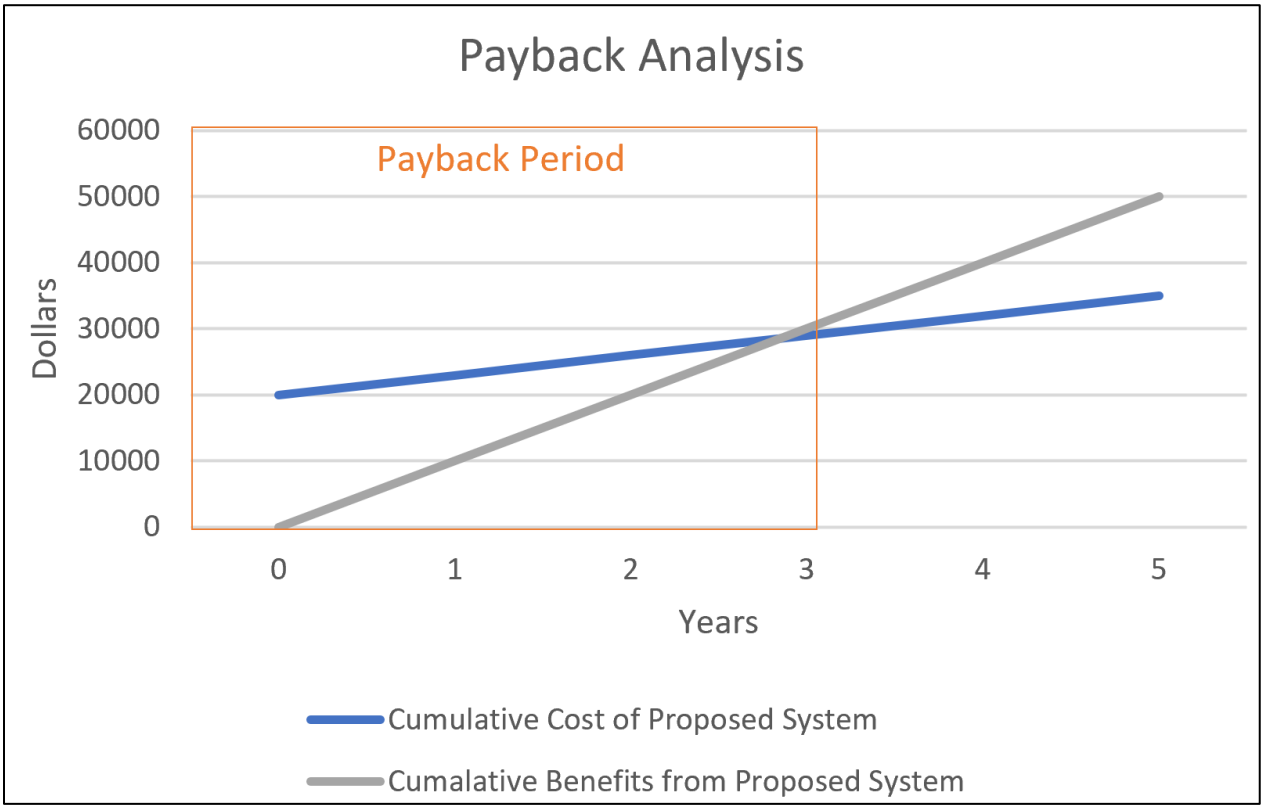


Figure 11: Payback Analysis Chart

Year	Cost($)	Cumulative Cost($)	Benefits($)	Cumulative Benefits ($)
0	20000	20000	0	0
1	3000	23000	10000	10000
2	3000	26000	10000	20000
3	3000	29000	10000	30000
4	3000	32000	10000	40000
5	3000	35000	10000	50000

Description

The overall cost of developing the system is estimated at around $20,000. Each year of maintenance is about $3,000. The benefit received in the first year is $0, because the feature is free for users to see and use. After the first year, there are subscriptions that cost certain amounts of money. The estimated benefit for each year is $10,000. After 3 years past the deployment date, the total benefit is worth more than the total cost.

Return on Investment (ROI) / Total Cost of Ownership (TCO)

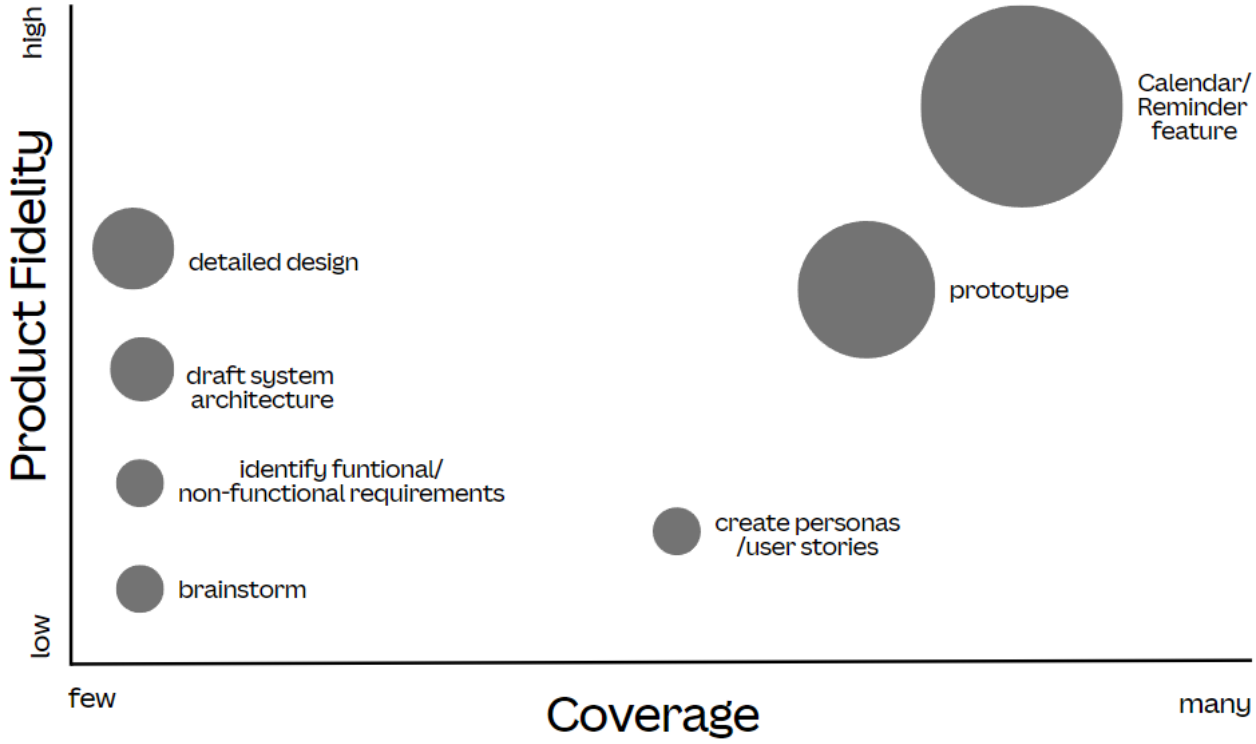
ROI = (benefit – cost) / cost = (50,000 – 3,000) / 3,000 = 15.67

TCO = budget * duration (months) = 4,240 * 60 = 254,400

ROI / TCO = 15.67 / 254,400 = 0.00006159591

- ROI is calculated for a time period like 5 years but expressed in terms of annualized rate.
- We include MPV in calculations of total costs and total benefits.
- TCO is calculated for a time period like 5 years.
- TCO includes wages, development, infrastructure, and operational costs.

Figure 12: **Minimal Viable Product (MVP) Analysis of the Calendar/Reminder Feature**



For the MVP analysis of our Calendar/Reminder feature, we created a 2-Dimensional graph contains two axes: the Coverage represents the number of reached customers on the x-axis, and the Product Fidelity which represents how close the product at that point to the final product. Each dot represents a single process that our team has gone through over the development cycle. The length of each feedback cycle/iteration is presented by the size of the dot. For instance, the brainstorming process only happens between the developers, and not reaching to any users; the dot size is also small since the team can get instant feedback. On the other hand, the prototype/demo of the product can be reached by a certain

number of users that are willing to test the feature as well as the dot size also changed since it requires more time to get feedback from the users.

Stakeholder Cost/Benefit Analysis

The three main types of stakeholders that are analyzed are the purchaser, seller, and advertiser. For the purchaser the main benefits are increased usability because events are organized more. There is no real cost for the purchaser.

In terms of the seller, the main benefits are that increased usability from the app/service of the calendar/reminder system leads to even more users leading to more profits for the business. The cost is that the initial post- deployment phase might have some deficits which could lead to some cost cutting practices to be enabled.

Finally, the benefits for the advertiser of the app/service are that they have more users to see their products, leading to more sales and profits for their own products. There is really no cost associated with the advertiser.

Risk Management

Risk Appetite

Description

In assessing the decision to save the Discord calendar feature created by college students, we need to consider the changes in popularity and usage. If the feature's popularity and usage decline significantly, it may indicate that users no longer find it beneficial in comparison to other Calendar/Reminder applications available in the market. At this point, it becomes important to evaluate the risk tolerance among stakeholders. If stakeholders demonstrate a high tolerance for risk and have a willingness to explore innovative solutions, there may be an opportunity to gather support to revive the feature. However, to respect stakeholders who have a low-risk tolerance and prioritize stability, if they prefer to discontinue the feature due to market demands and user preferences, it is best to cut further negotiations. The decision to save or abandon the Discord calendar feature depends on all stakeholders involved. It is important to carefully weigh all factors and respect each stakeholder’s choice on either reviving the feature or suspending it.

Risk Analysis

Figure 13: **Table of Risk Type Analysis on Probability, Impact and Exposure**

Risk	Probability (Scale)		Impact	Exposure
	Generic	Precise		
Hiring college students	Medium	75%	High because the code might not be maintainable, so debugging and adding new features to the app might in the long-run cost more money than hiring experienced programmer.	This causes additional schedule impacts due to the delay in the feature to be released at a required date due to the code not being maintainable. It may need consistent testing before release.
Offering the product completely free for the first year of launch and then going on a "Freemium" model	Medium	70%	Medium because the product has no revenue for the first year it degrades not only product quality but satisfaction.	This causes additional cost impacts because there can also be a decrease in customer satisfaction with the product, which needs to be upgraded before releasing a full-cost version of the product.
Launching a minimum value product (MVP)	Medium	65%	Medium because as MVP may not fully meet the needs and expectations of the users because of a limited set of features and lack of quality.	This causes additional cost impacts because the lack of quality on the product decreases customer satisfaction and furthermore product revenue.
Discord implements a Calendar and Reminder feature natively.	Low	30%	High because adopting the Calendar and Reminder feature may be a struggle for users since Discord is solely known for communication gaming purposes which can further delimit customer satisfaction.	This causes additional cost impacts because if customers do not want to natively adopt the Calendar and Reminder feature cost revenue depreciates.

Reliance on the Discord API	Low	25%	High because the Discord API may later have restrictions which conflict with customer satisfaction. It may also introduce security risks that affect the quality of the software, which need to be properly tested.	This causes schedule impacts. Calendar/Reminder feature has updates or fixes within Discord, potentially causing delays in the release scheduling for updated versions, because of it still being integrated in the Discord infrastructure.

Using Processes to Address Risk

Figure 14: **Table of Risks and Process Types that May Influence Probability and Cost**

Risk	Process	Probability	Impact	Potential Cost
Hiring less experienced programmers	Integrated Risk and Opportunity Analysis was used as the team weighed the pro for cheaper development, while the con of lower quality of code.	Medium because college students do not have the same experience compared to professional software programmers.	High because the code might not be maintainable, so debugging and adding new features to the app might in the long-run cost more money than hiring experienced programmer.	60%

Risk	Process	Probability	Impact	Potential Cost
Offering the product completely free for the first year of launch and then going on a "Freemium" model	Integrated Risk and Opportunity was used to determine the pro of potential growth and con of not making any revenue in the first year.	Medium because users would be accustomed to using the product for free, so having users pay for additional features makes some of them leave.	Medium because the product does not have any revenue for the first year.	40%
Launching a minimum value product (MVP)	Integrated Risk and Opportunity Analysis was used as the team saw the pros of launching the product earlier outweighed the con of the product not having all the features	Low because since the feature is a live service, developed minimally and lacks sufficient quality there would be more maintenance in developing and testing.	Medium because as MVP may not fully meet the needs and expectations of the users because of a limited set of features and lack of quality.	40%
Discord implements a Calendar and Reminder feature natively.	Integrated Risk Opportunity Analysis because we acknowledge that Discord could implement it, but the opportunity outweighs the risk.	Low because Discord has shown no interest in developing a Calendar and Reminder system as their focus is on communication gaming purposes.	High because users use the natively built-in feature; however, we could switch our product to be built on top of the natively built one, adding additional features.	70%

Risk	Process	Probability	Impact	Potential Cost
Reliance on the Discord API	The Resource Allocation and Monitoring Approach was used to determine this risk because we are relying on Discord API. However, there are bug fixes and version updates with Discord application, so our software needs to adapt to this.	Low because the API was first introduced in December 2016[6], it is widely used, and Discord has been proactive in their support with their API.	High because the Discord API may later have restrictions. It may also introduce security risks that need to be properly tested.	30%

System Description Artifacts

Lean Canvas

Figure 15: Lean Canvas for the Discord Calendar/Reminder System

<table>
<tr>
<td>Problem:
- The clock needs to be automated.

- Another thing that needs to be automated would-be events/meetings that are repeated.

- Automatically recognize events from email, discord message, etc.</td>
<td>Solutions:
The new system includes a calendar feature which can help schedule and remind people of events/meetings.</td>
<td>Unique Value Proposition:
- Tailored for discord users specifically

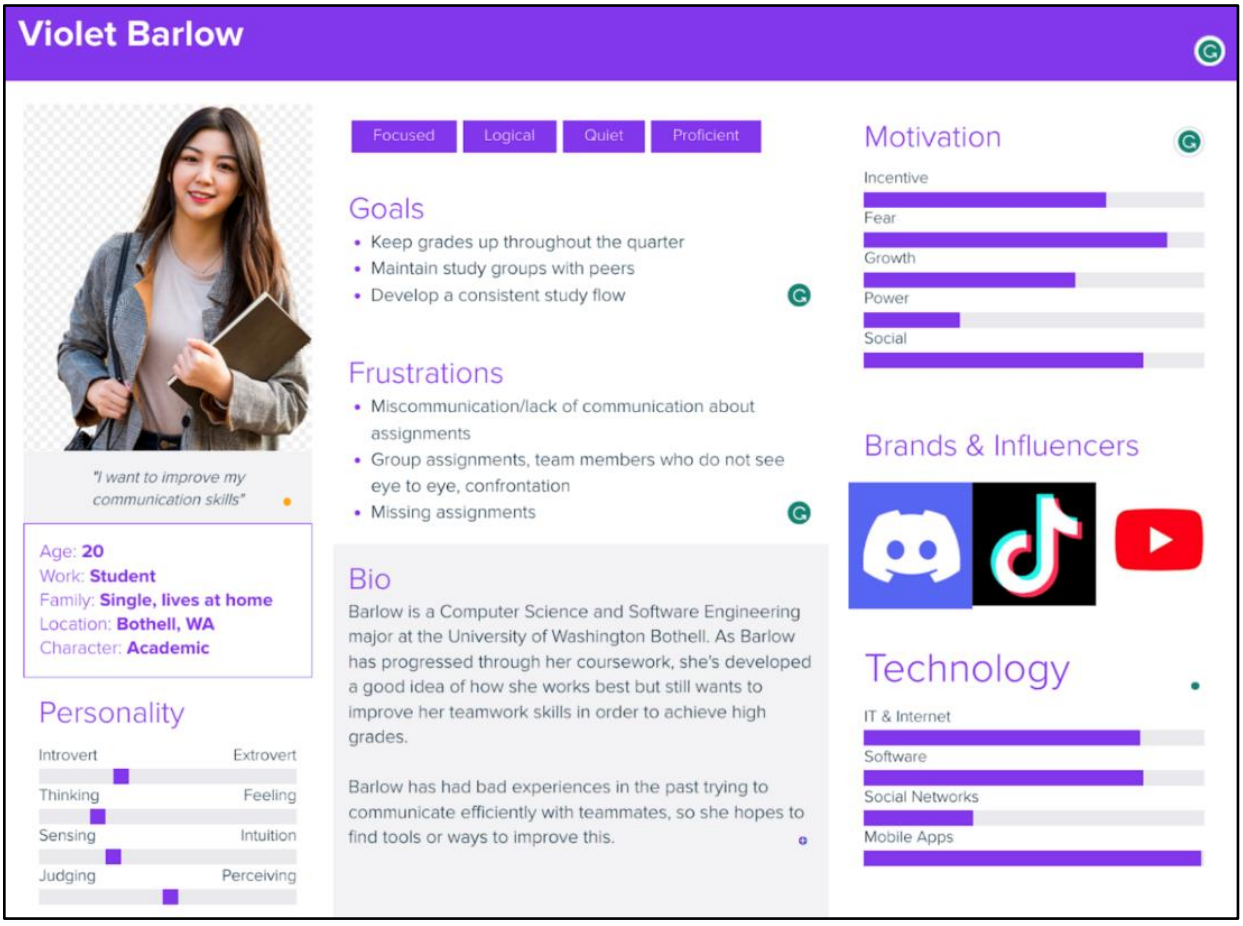
- Can be connected to your personal calendar

- Used with a common communication platform</td>
<td>Unfair Advantage:
First to market in the Discord bot ecosystem</td>
<td>Customer Segments:
- Gamers
-Academia
-Business</td>
</tr>
<tr>
<td></td>
<td>Key Metrics:
- Number of users/servers
- Satisfaction of consumers
- Number of events/reminders created</td>
<td></td>
<td>Channels:
-Social Media
-YouTube
-Word of Mouth
-Discord Servers</td>
<td></td>
</tr>
<tr>
<td colspan="3">Costs:
- Database Fee: cost depends on the number of users and information.
- Licensing (patent and trademark): cost depends on the price and uniqueness of the feature.
- Website: (Hosting and Domain free)</td>
<td colspan="2">Benefits:
- Customers pay a subscription fee to have additional features.
- Donations/Patrons</td>
</tr>
</table>

Artifact Description: The purpose of this artifact is to communicate the concept and idea of the Discord Calendar/Reminder System quickly and visually. This artifact was used near the beginning/middle of our development processes in order to create a well-rounded and clear idea of the product. Key building blocks of this artifact are connected back to the Discord Calendar/Reminder System specifically.

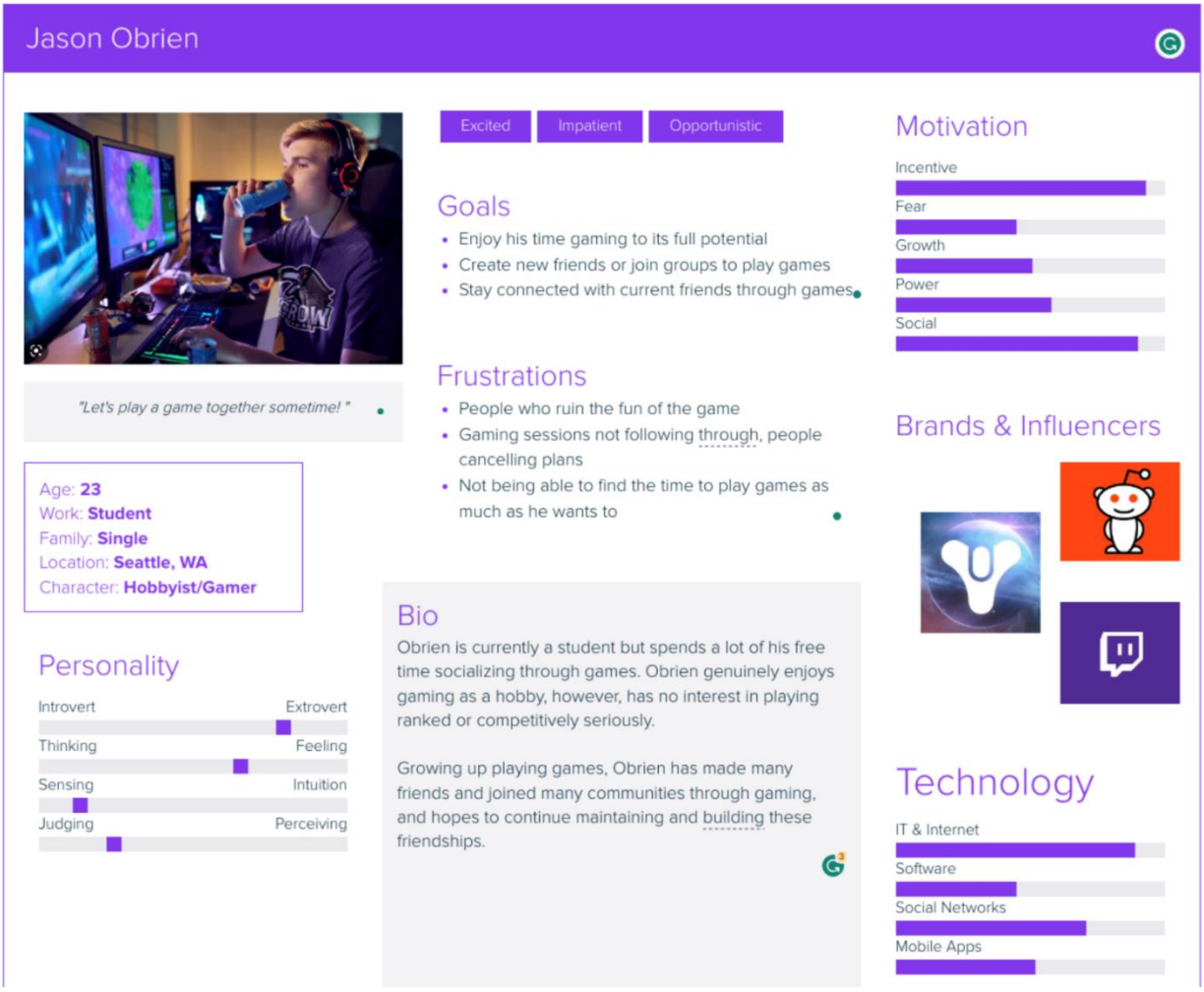
Personas

Figure 16: Academic Persona Representing a User of the Discord Calendar/Reminder System



Artifact Description: After identifying key customer segmentations, customer personas were created to further connect with the audience and understand their needs and desires. This artifact, the Violet persona, played a crucial role during development as our product is targeted significantly towards students. The Violet persona is based on the student character type, which falls under the academia segmentation.

Figure 17: Gamer/Hobbyist Persona Representing a User of the Discord Calendar/Reminder System



Artifact Description: After identifying key customer segmentations, customer personas were created to further connect with the audience and understand their needs and desires. This artifact, the Jason persona, played a secondary role during development. Although our primary audience is academic-focused individuals, we still recognize gamers and hobbyists were the original target audience for the Discord platform in general. Due to this, we targeted the gamer/hobbyist segmentation in a secondary way also.

SPICIER Scenarios

Figure 18: SPICIER Scenario Violet's Group Study Session

Violet is a Computer Science and Software Engineering student at the University of Washington. As a first-generation university student, she is determined to succeed in her academic career, which includes keeping her grades up and developing meaningful connections. In hopes of achieving these goals, she wishes to form a weekly study group with peers in her class. To create this study group, she opens a discord server to invite all her peers. Once the server is created though, she finds it frustrating to spread the word about her study session, as the announcements get lost in a sea of other messages within the channels. Nobody seems to be responding to her announcement due to the lack of visibility and structure of the invite, causing Violet to feel unheard of and dejected.

<magic happens>

Violet can easily schedule an event for her study session, which is proudly displayed above the channels where it is visible to everyone on the server. Violet can add all the information she needs to the event (time, date, location, description) so others can easily decide whether or not they can attend. Violet is relieved that she has reassurance everyone in the channel is able to view her study session event and is excited to begin connecting with more people from her courses. In addition, Violet can now view who and how many people are interested in her study sessions, allowing her to prepare more adequately for the study session via snacks, scratch paper, and study room sizes!

Artifact Description: The main domain-specific concern we were targeting when creating this scenario was the struggle to share an event within a Discord server. Since this is a SPICIER scenario, it was motivated by a function requirement, specifically inviting members of a discord server to an event. This scenario adheres to the SPICIER template, with each trait having a specific rationale. The story narrative of the customer's (Violet) experience makes it easy for others to understand. The story shares the personal details of the customer, mainly her academic background as a first-generation student, to help readers understand her motivation and desire for success. The story remains implementation free to focus on the customer's needs and desires, which helps stakeholders understand the need for this feature. Insight into the customer's desires can also be extracted from this story by stepping into her shoes and understanding the desire for this feature, mainly more usability with the Discord servers for event planning. Emotional words such as frustrated, relieved, and excited are used to demonstrate the reaction this customer feels in relation to the lack of/presence of our Discord Calendar/Reminder feature.

The domains of specific knowledge represented in this scenario are academia-related (student, first-generation student, university, courses, grades, study groups) and Discord-related (servers, channels, messages).

Figure 19: SPICIER Scenario Phil's Car Meet

Phil is a 19 years-old college student in the Bay Area California. His main interests are cars and technology. One day, when he was surfing through social media, he found a Car Group in his area, and it also includes a Discord link to a server for the participants to have a place to meet virtually. After joining several voice channels, and making some friends and connections, Phil excitedly proposes an idea to have an offline car meet. People in the group agreed with his proposal.

There are several tools on the market to create a shared calendar such as Google Calendar, Apple Calendar, etc. However, those tools all share some disadvantages or inconveniences for all users which can cause annoyance and confusion, especially when users may use different platforms depending on their technological background. With all the different types of platforms out there (Google Calendar, Apple Calendar, etc.) it can be difficult for individuals to synch leaving Phil discouraged.

<magic happens>

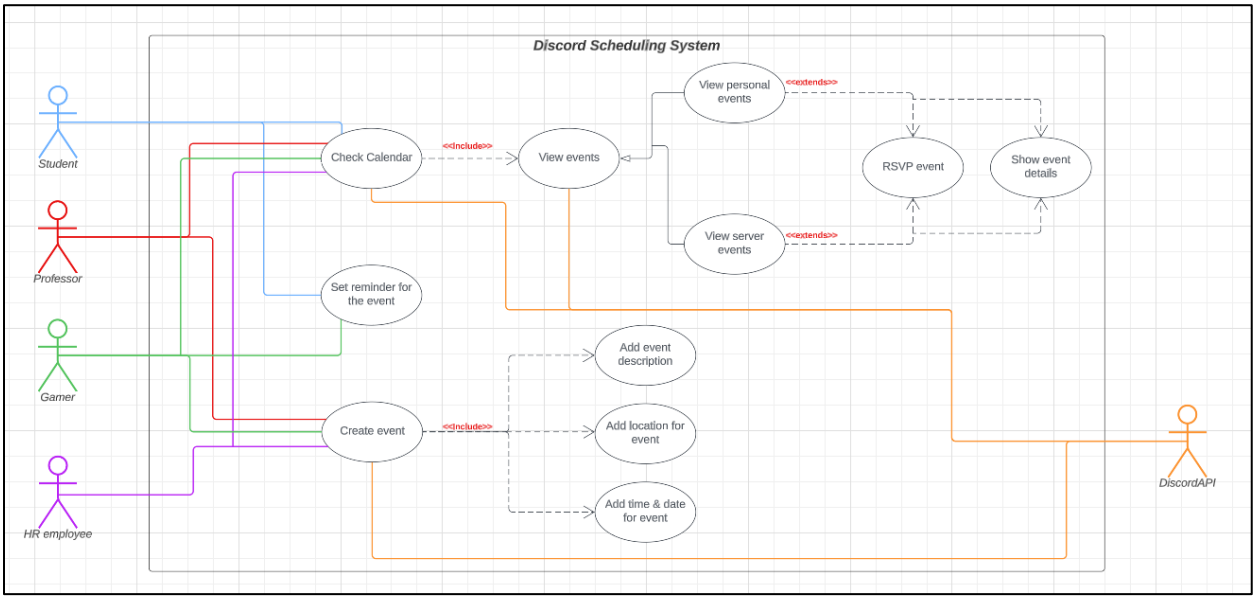
Instead of having to create another calendar tool and share the link to every group member, they use the Calendar/Reminder feature on Discord. The calendar/reminder feature allows people to mark if they are able to attend, view and share the event on the discord server calendar, and most importantly have one centralized place everyone in the server is able to plan events within. Phil is excited again for his car meet!

Architectural Description: The main domain-specific concern we were targeting when creating this scenario was the struggle to share an event to a wide audience of individuals within a hobbyist group. Since this is a SPICIER scenario, it was motivated by a function requirement, specifically inviting members of a discord server to an event.

This scenario adheres to the SPICIER template, with each trait having a specific rationale. The story narrative of the customer's (Phil) experience makes it easy for others to understand. The story shares the personal details of the customer, mainly his desire to meet people in his area who share the same interests. The story remains implementation free to focus on the customer's needs and desires, which helps stakeholders understand the need for this feature. Insight into the customer's desires can also be extracted from this story by stepping into his shoes and understanding the desire for this feature, mainly a way to gather people of all technical backgrounds and platforms. Emotional words such as excited, annoyed, confused, and discouraged are used to demonstrate the reaction this customer feels in relation to the lack of/presence of our Discord Calendar/Reminder feature. The domains of specific knowledge represented in this scenario are hobbyist-related (Car group, car meet) and Discord-related (link, servers, voice channels).

Use Case Diagram

Figure 20: Use Case Diagram of the Discord Calendar/Reminder System



Artifact Description: The use case diagram of the Discord Calendar/Reminder System provides visual representation of the relationship between actors (users, external systems) and the system itself. This artifact was used to map out the different functionalities needed within the system, which customer segmentations would use what, and how the overall system would interact. As an example, it was found that almost all the actors would need the same three main functionalities (check calendar, set reminder for the event, create event) in order to do any tasks regardless of their motives. This artifact also allowed us to begin thinking about the structure of the Discord Calendar/Reminder System through the include and extends arrows.

Use Case Scenarios

Figure 21: Use Case Scenario for an Off-Line Meeting on a Discord Server

Use Case	Off-line club event
Primary Actor	Club user
Goal In Context	Use Calendar/reminder for an off-line car club event
Preconditions	1. Must have a Discord account 2. User is already a member of the channel
Trigger	Participants in the channel agreed on a date to do an off-line event
Scenario	1. Car enthusiast saw a link to a Car Group Discord link on social media 2. The person joins the group 3. After several online meeting via Discord, the group participants decided to have an off-line meeting on a specific date 4. Anyone who wants to participate uses the Calendar in the Discord group to mark available dates. 5. The group admin then choose a date every 3 months that have meets the largest amount of availability to mark that date to be the off-line date 6. The calendar feature allows the group admin to mark the date, add notes under that date (time, location, rules, etc.) 7. The calendar feature has different options for the group admin to choose to remind anyone who is coming 24-hr before the event
Exceptions	The event has already passed

Artifact Description: In this Use Case Scenario, we focus on the Creating Event feature of our project. We decided to choose a more casual approach by creating a table for the Use Case Scenario. With this decision, it is clearer for the reader to distinguish between the Use Case Elements on the left-hand side of the table, and the Description for each Elements on the right-hand side of the table.

This specific Use Case Scenario targeted a Discord user that participates in a car club channel. He wants to create an off-line event for the group and for convenience create, edit, and send notifications to all participants about the event inside the group's main communication medium, which is Discord.

Figure 22: Use Case Scenario Focusing on Scheduling/Creating an Event

Use Case	Schedule an event
Primary Actor	Professor
Goal In Context	To schedule an event for their next office hours
Preconditions	1. The date and time of the office hours have been predetermined 2. The professor has Discord installed and is hosting a server 3. The professor has Discord Reminder System
Trigger	The professor wants to share their office hours with students via their class Discord server
Scenario	1. The professor logs into Discord 2. The professor opens their Discord server 3. The professor selects "Add New Event" from the Discord Reminder System add-on 4. The professor references the time and date they want to hold their office hours 5. The professor selects the date they want to hold their office hours and inputs the time range of the office hours 6. The professor titles their event 7. The professor may optionally input a description for their event 8. The professor confirms the information and publishes the event 9. A new event has been created 10. The event is now viewable and interactable by members of the Discord server
Exceptions	The professor cannot log into their Discord account

Artifact Description: In this Use Case Scenario, we focus on the Creating Event feature of our project. We decided to choose a more casual approach by creating a table for the Use Case Scenario. With this decision, it is clearer for the reader to distinguish between the Use Case Elements on the left-hand side of the table, and the Description for each Elements on the right-hand side of the table. This specific Use Case Scenario follows a professor creating an event for their office hours.

SRS Requirements (Functional/Non-Functional)

Figure 23: Functional Requirements of the Discord Calendar/Reminder System

Req. ID	Requirement
FR-1	The system must display the calendar to users.
FR-2	The system must provide a list of events to the users within a certain timeframe.
FR-3	The system must send notifications to users about events.
FR-4	The system must allow users to schedule events.
FR-5	The system must allow users to input information about events (when, where description).
FR-6	The system should allow users to RSVP for events.
FR-7	The system should allow for multiple events to be scheduled.
FR-8	The system should allow the user to check who is coming for an event.
FR-9	The system should allow users to cancel/decline events.
FR-10	The system could allow users to provide a rationale as to why they canceled the event.

Artifact Description: The functional requirements listed above are specifically created from the Discord Calendar/Reminder System thinking about the desires and fears of the identified customer segmentations. The top five functional requirements are highlighted in yellow to visually show their importance to the system. The highlighted functional requirements mainly focus on the core functionalities of the Discord Calendar/Reminder system which the remainder of the other artifacts focus on as well, such as displaying the calendar and events, sending our reminders for events, and creating events.

Figure 24: Non-functional Requirements of the Discord Calendar/Reminder System

REQ. ID	Requirement	Rationale
NFRQ-1	The system must allow multiple different operating systems to interact.	Users of a discord server may all be interacting with the server using different platforms.
NFRQ-2	The system must ask for the user's permission to have access to their personal information such as email to detect events and add them to the calendar.	Users want the calendar to automatically detect events from a convenient standpoint, but they also need some permission to agree on if the calendar wants to access the information.
NFRQ-3	The system should process and display newly created events within 15 milliseconds.	The system should notify users if something went wrong within a small amount of time to avoid confusion
NFRQ-4	The system should retrieve the details of an event within 10 milliseconds	Users on Discord want to access information as quickly as possible.
NFRQ-5	The system should notify the user with an error if an event could not be created within 15 milliseconds.	The system should notify users if something went wrong within a small amount of time to avoid confusion
NFRQ-6	The system should encrypt the data it transmits.	Users want their information and data within their system to be secured.
NFRQ-7	The system should have an uptime of at least 99.999%.	Users should be able to always access the calendar as the system being down defers customers from using the feature.
NFRQ-8	The system should have less than the rate of occurrence of failure of 1/5000.	Users need to rely on the system.

NFRQ-8	The system should only grant access to the calendar feature to authorized users and should require verification before allowing users to add events to the calendar.	Users want clear and organized information.
NFRQ-10	The system could be able to handle 100,000 requests at once.	The system should be able to handle lots of users at once.

Artifact Description: The non-functional requirements listed above are specifically created from the Discord Calendar/Reminder System thinking about the desires and fears of the identified customer segmentations. The non-functional requirements identified are used to address concerns in safety, security, and software-quality categories. As examples, security concerns are addressed by NFRQ-5, NFRQ-6, and NFRQ-10. The ten non-functional requirements can also be categorized by various attributes such as interoperability (NFRQ-1), usability (NFRQ-2, 3, 4, 9), reliability (NFRQ-5, 6, 10), availability (NFRQ-7), and correctness (NFRQ-8).

Product / Project Backlog

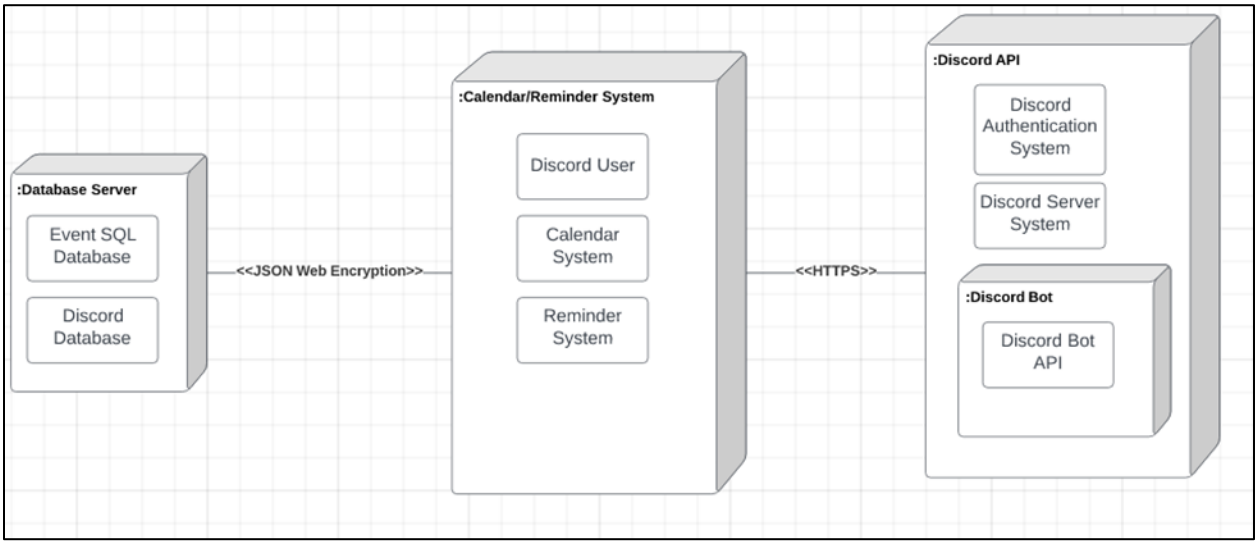
Figure 25: Project Backlog Organized by Sprint Number

Req. ID	Requirement	Sprint
FR-6	The system should allow users to RSVP for events.	1
NFRQ-4	The system should retrieve the details of an event within 10 milliseconds	1
FR-7	The system should allow for multiple events to be scheduled.	2
FR-8	The system should allow the user to check who is coming for an event.	2
FR-9	The system should allow users to cancel/decline events.	3
NFRQ-3	The system should process and display newly created events within 15 milliseconds.	4
NFRQ-5	The system should notify the user with an error if an event could not be created within 15 milliseconds.	4
NFRQ-6	The system should encrypt the data it transmits.	5
FR-10	The system could allow users to provide a rationale as to why they canceled the event.	6

Artifact Description: The project backlog is visualized as a table of functional and non-functional requirements that are not of top priority at the current moment of this document's release. These requirements fall under the "should", "could", and "won't" prioritizations of the MoSCoW convention, and therefore are backlogged for future development sprints. Each requirement is small enough to be completed within a single sprint or multiple can be done in one. When moving through the backlog, requirements which are classified as "should" are of higher priority than "could", meaning future sprints should focus on those higher priority requirements first.

Deployment Diagram

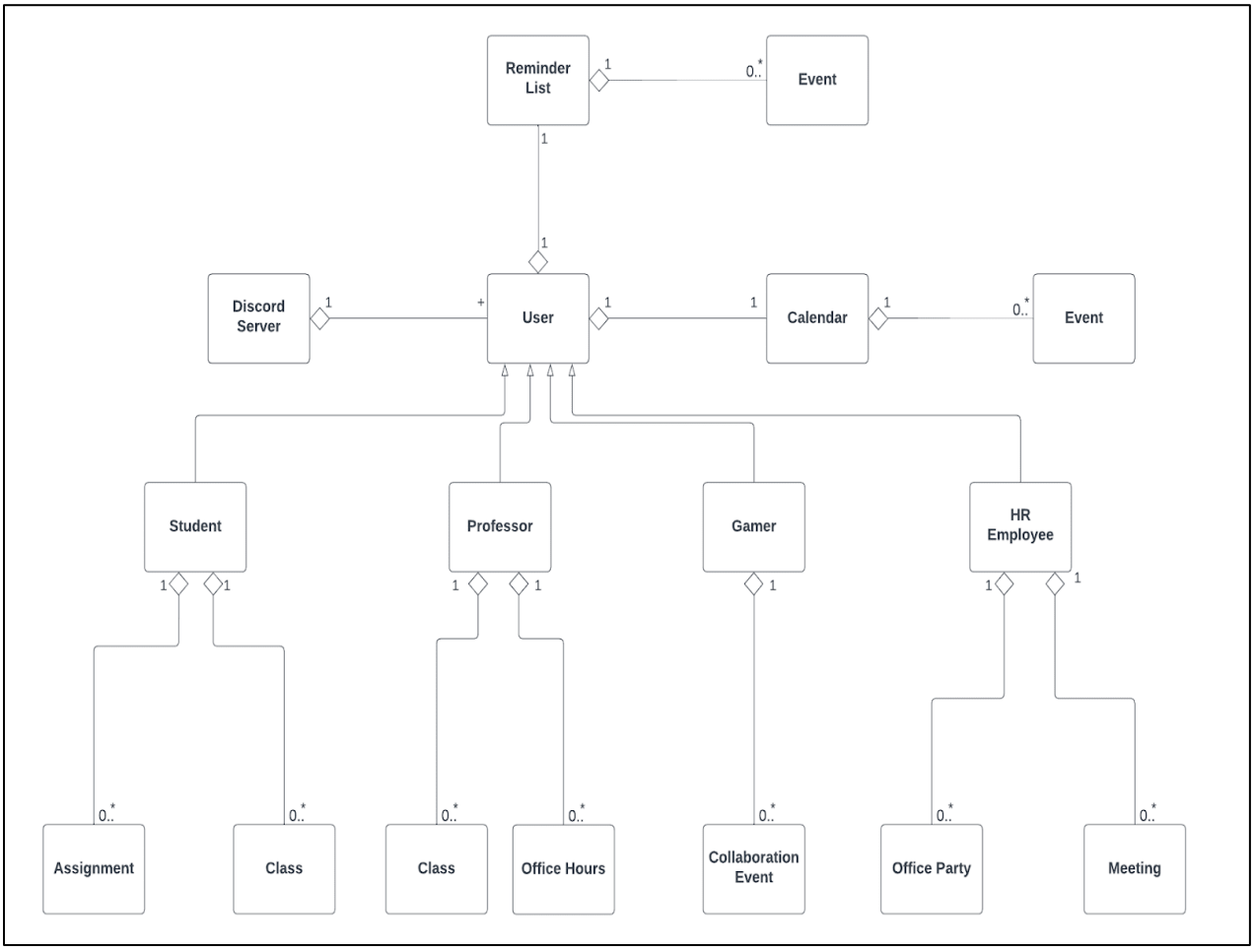
Figure 26: Ad Hoc Deployment Diagram of the Discord Calendar/Reminder System



Artifact Description: The purpose of this Ad Hoc Model is to show the different components' functionality and how each component interacts with each other and what communication protocols are being used. The Discord API component is built by Discord so the systems we cannot alternate them, so all the logic is being down by our Calendar/Reminder System. The decision to use JSON Web Encryption and HTTPS, as from our groups research, is because they are standard protocols, and it ensures that the data being sent between the different components is encrypted. This protects against potential sensitive information from being interrupted and read by black actors.

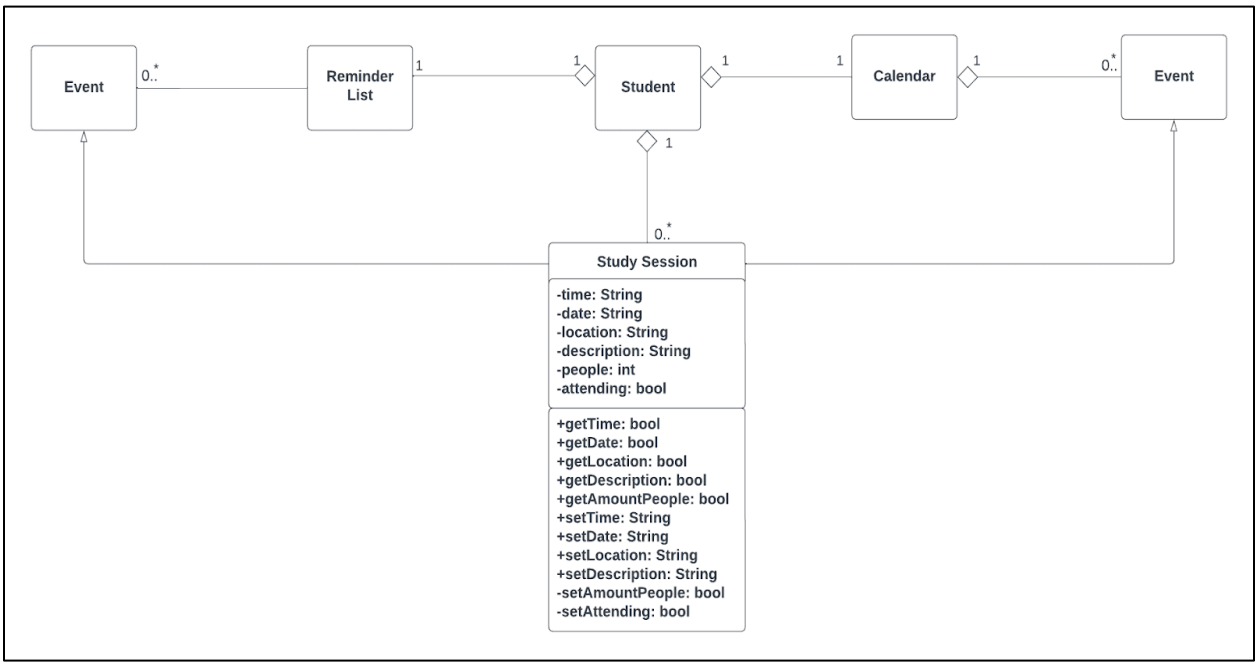
Domain Diagram / Class Diagram

Figure 27: Domain Diagram of the Discord Calendar/Reminder System



Artifact Description: This model addresses the usability of the calendar/reminder list feature. This shows the organization of the calendar/reminder list feature within the Discord Server amongst different types of users. These types of users are specifically students, professors, gamers, and HR employees. Each of these users has different types of things that they have in respect to their character type and role. One key architectural decision was to have events for both the calendar and reminder list. This is to keep both types parallel and consistent in terms of their composition, making both features usable.

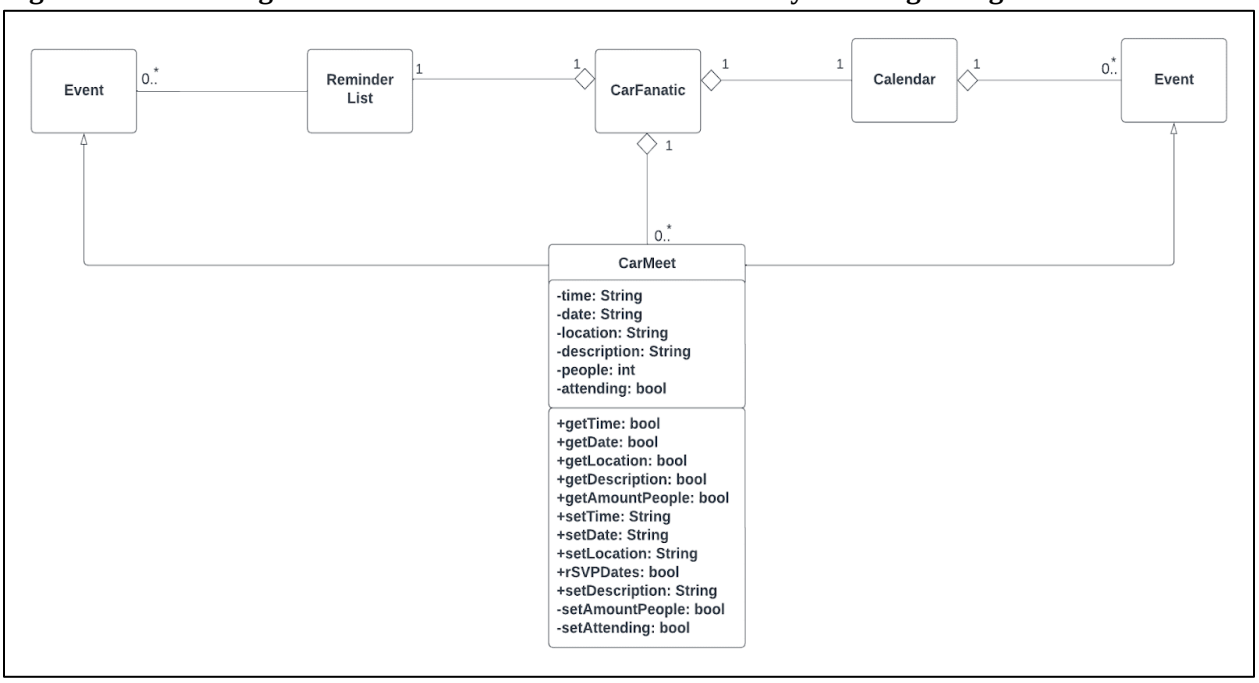
Figure 28: Class Diagram of the Discord Calendar/Reminder System Regarding Violet's Study Session



Artifact Description:

This diagram shows the various attributes and functions of a study session, and how it is organized within a Discord Server. It shows how a study session is a type of event which is within a Reminder List and Calendar. The study session attributes are mainly for the information about the study session. The functions are mostly for setting and getting various attributes from the study session which is useful for the user interface of the system.

Figure 29: Class Diagram of the Discord Calendar/Reminder System Regarding Phil's Car Meet

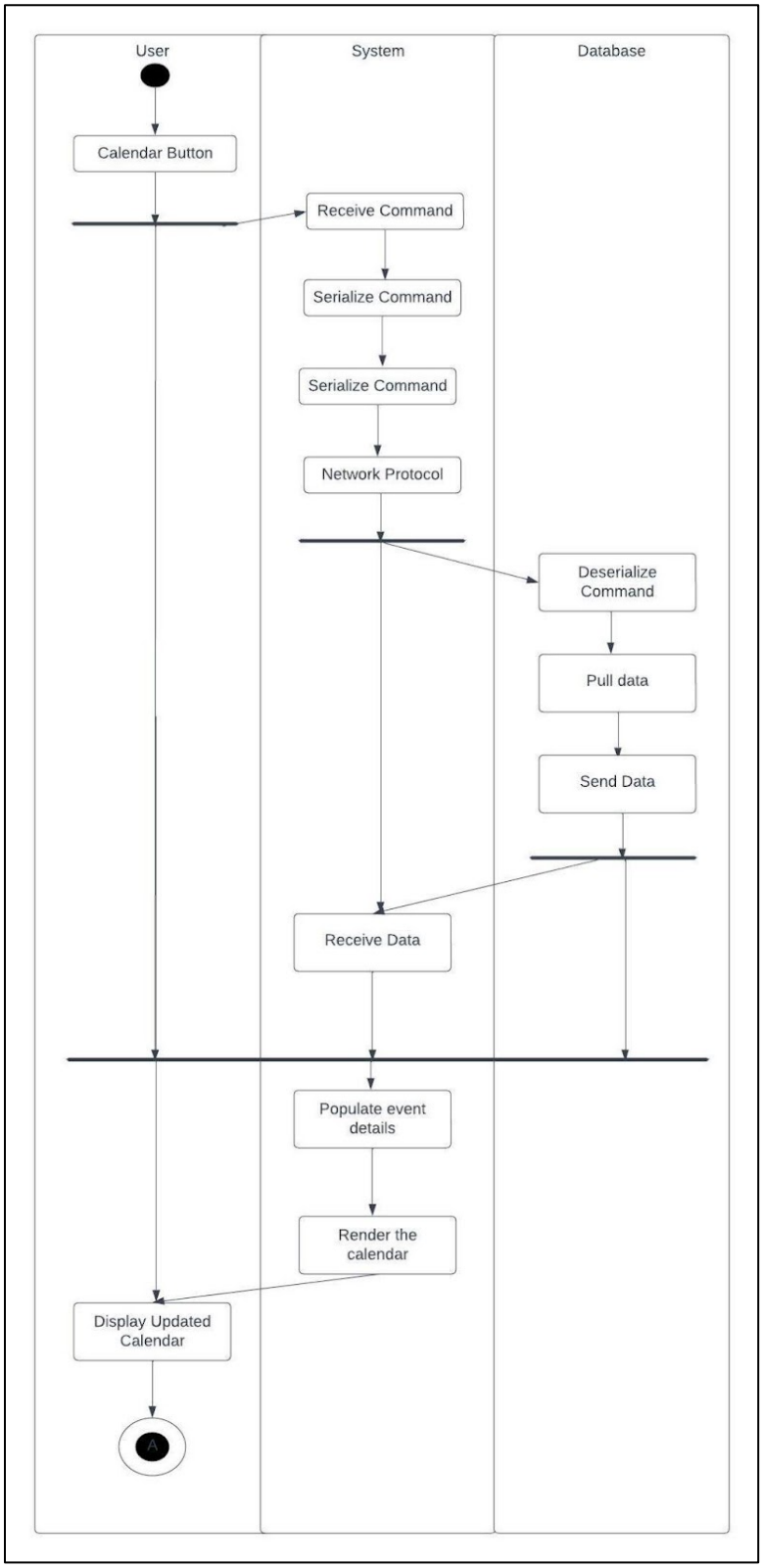


Artifact Description:

This diagram shows the various attributes and functions of a car meet, and how it is organized within a Discord Server. It shows how a car meet is a type of event which is within a Reminder List and Calendar. The car meet attributes are mainly for the information about the event. The functions are mostly for setting and getting various attributes from the car meet which is useful for the user interface of the system. One important function is the RSVP, which helps users indicate whether or not they are coming to the meeting.

Activity Diagram

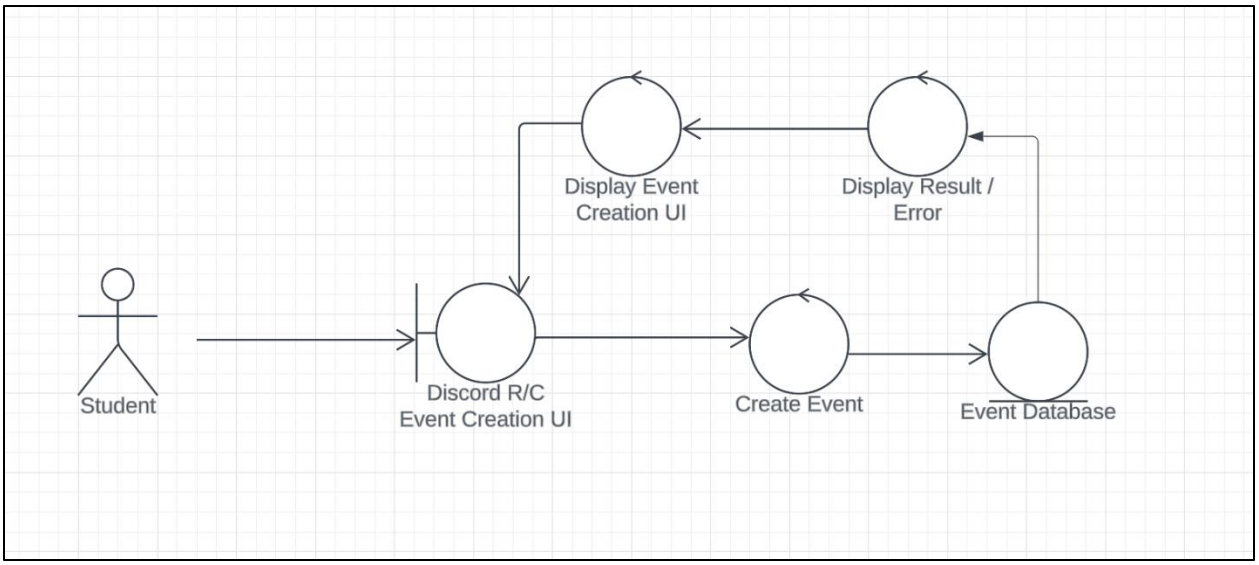
Figure 30: Activity Diagram of the Discord Calendar/Reminder System



Architectural Description:
The diagram focuses on Discord and how the Discord Calendar Button is processed once the user clicks on the button. The specific concern to create the revised form of the activity diagram was to establish traceability and reliability when a user clicks on the calendar button within the system. It visually outlines the flow of activities during the process and how they interact with different system parts, including the database, to understand the entire process. One key architectural decision was to understand how much computational power the system needs to process the requests and update the requests on the calendar. This is important because to determine the optimal response time for the user before the user recognizes a stall in the system, one must know the amount of computer resources the system uses to determine the entire system's performance, and how this can affect cost.

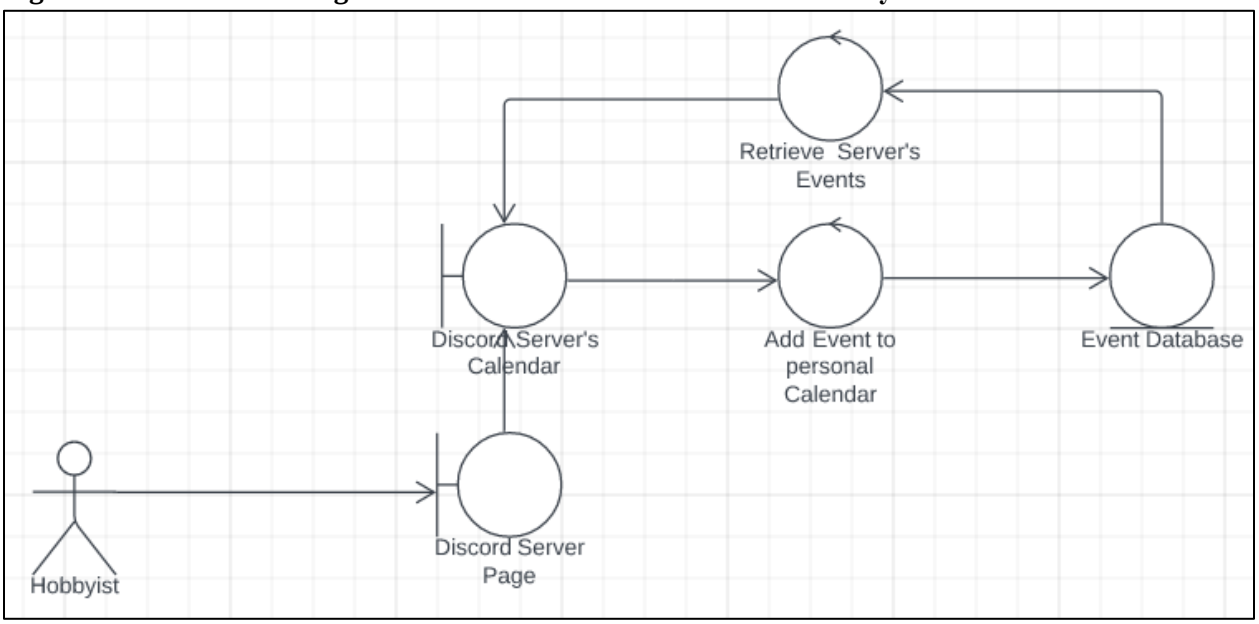
Robustness Diagrams

Figure 31: Robustness Diagram of the Discord Calendar/Reminder System for Violet's Study Session



Artifact Description: This robustness diagram was used to help develop a high-level view of the relationships and interactions between the Discord Calendar/Reminder System's components/processes regarding the Violet persona's use case scenario. By breaking down that use case scenario using a robustness diagram, it is clearer to see how the system behaves. The student only interacts with the Discord Calendar/Reminder (R/C) Event Creation UI in order create the event from their end. However, based on the input of the student the system runs a process to create the actual event object and store it within the event database entity. From there, there a result is created in another process (successful/unsuccessful) and then displayed to the user through that same UI.

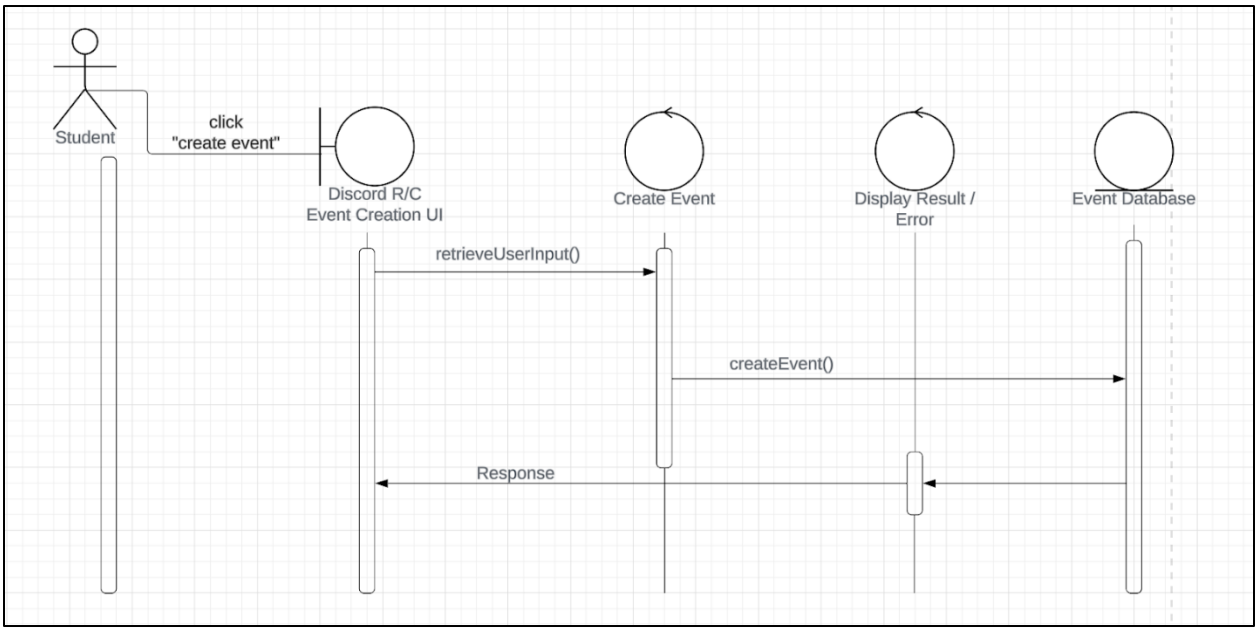
Figure 32: Robustness Diagram for the Discord Calendar/Reminder System for Phil's Car Meet



Artifact Description: This robustness diagram was used to help develop a high-level view of the relationships and interactions between the Discord Calendar/Reminder System's components/processes in regards to the Phil persona's use case scenario. By breaking down that use case scenario using a robustness diagram, it is clearer to see how the system behaves. The Hobbyist must only interact with the Discord Server Page and select the calendar button. From there the Hobbyist is brought to another UI displaying the calendar, there processed such as add event runs depending on the Hobbyist's interactions with the Discord Server's Calendar UI.

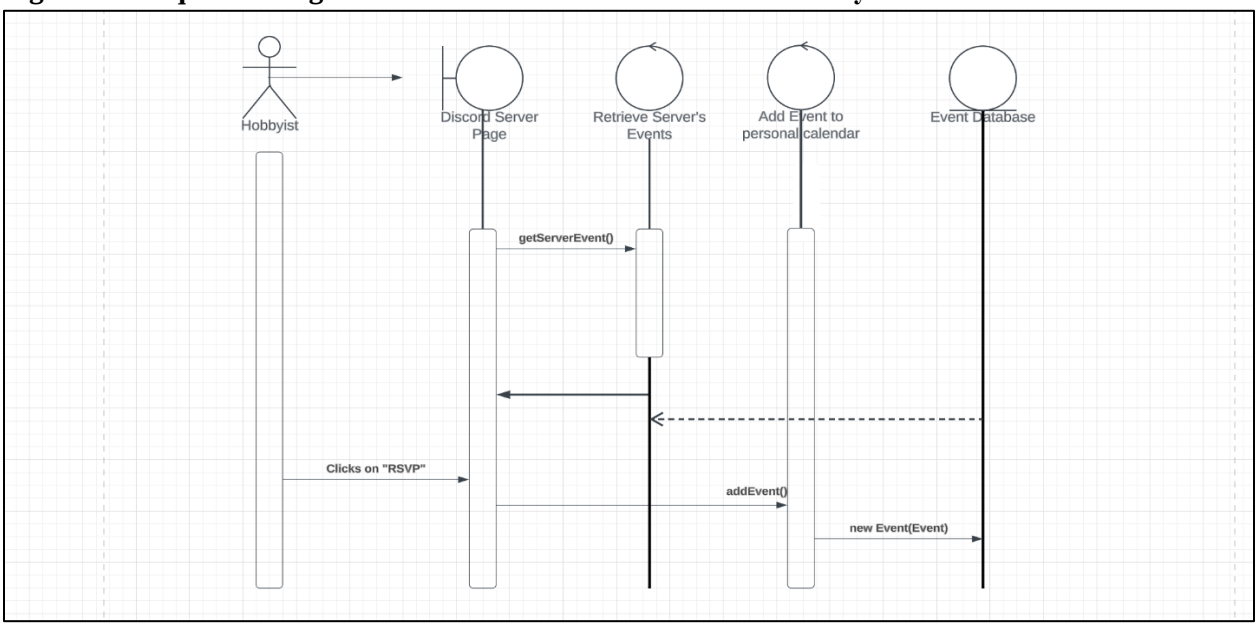
Sequence Diagrams

Figure 33: Sequence Diagram for the Discord Calendar/Reminder System for Violet's Study Session



Artifact Description: This sequence diagram visually represents the interfaces Violet may encounter, the processes the system runs, and the entities that are called upon during her interaction with the Discord Calendar/Reminder System based on her SPICIER scenario and robustness diagram. This sequence diagram is broadened to be about the student segmentation and focuses on the create event functionality of the system.

Figure 34: Sequence Diagram for the Discord Calendar/Reminder System for Phil's Car Meet

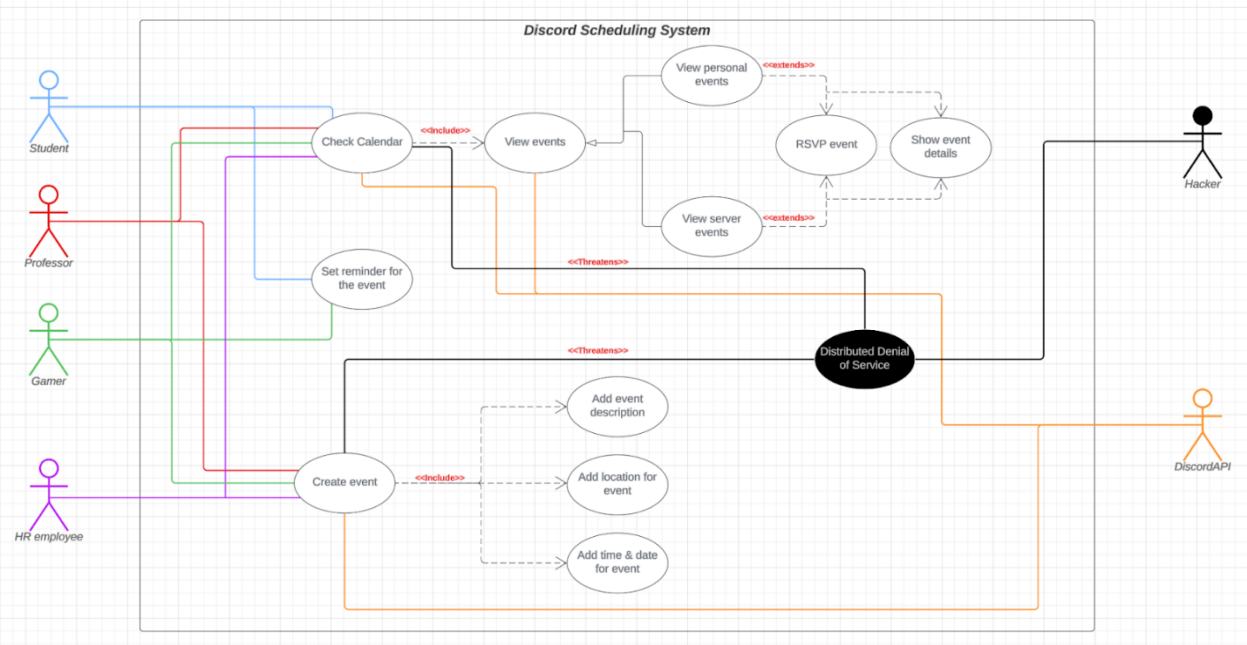


Artifact Description: This sequence diagram visually represents the interfaces Phil may encounter, the processes the system runs, and the entities that are called upon during his interaction with the Discord Calendar/Reminder System based on his SPICIER scenario and robustness diagram. This sequence diagram is broadened to be about the hobbyist segmentation and focuses on the RSVP functionality of the system.

Threat Model Analysis

Misuse Case

Figure 35: **Misuse Case of the Calendar/Reminder System**

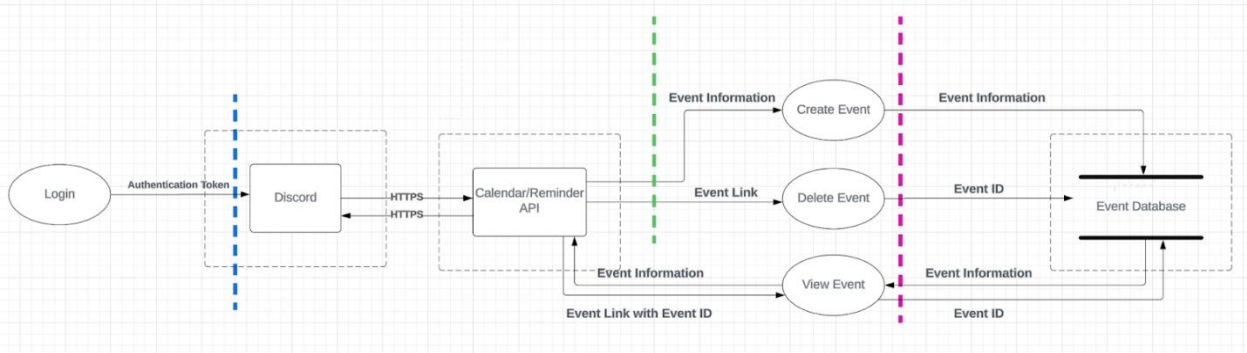


Our threat model above showing a misuse case, involves a potential DDOS attack that exploits the actions of retrieving and writing to the calendar database. This attack is critical because a hacker can exploit DDOS in the system by making spam requests to either view the calendar or creating an event which requires data to be retrieved from the database. This can also cause a server outage which causes monetary loss.

This security model helps our team understand critical points where potential instances can occur in the Calendar Reminder and fix damages that can heavily impact the monetary value. The security checks in these diagrams help influence the cost estimation with other stakeholders wanting to implement the Calendar Reminder infrastructure based on their domain. There should not be too much time spent on little details, because they might get modified later in future iterations.

Threat Model

Figure 36: **Threat Model for Calendar/Reminder System**



Blue Privilege Boundary

Spoofing: The adversary disguises themself as an authenticated user (could be through social engineering, which would circumvent any software precautions) gaining access to different actions only authenticated users should be able to perform.

Tampering: The adversary may alter event information using those actions (create event and delete event) which should only be available to authenticated users. This threat may occur if adversaries are able to circumvent authentication.

Repudiation: If an adversary can obtain the login information of an authorized user, or circumvents the authentication process, they may be able to claim all the actions performed were not done by them, but rather by the Discord user they were impersonating or some other individual.

Information Disclosure: Private information contained within event details can be compromised if an adversary obtained access to the view event process. This event may occur if an adversary somehow obtains this information through either human, software, or hardware flaws (social engineering, hacking, stealing physical laptops).

Denial of Service: The adversary may be able to spam event creations to overwhelm the server the authorized user is a part of, which would cause both the discord server and calendar to become cluttered/unusable. This should not crash the platform; however, it would deny service to the server for a small amount of time.

Elevation of Privilege: Adversaries may climb the discord privilege rankings by first spoofing, then taking on the identity of the authorized individual. The authorized individual may have sensitive information about other users, including those with higher powers, which can be exploited by the adversary into gaining more privilege (blackmail, or another spoofing attack).

Pink Privilege Boundary

Spoofing: An adversary might be able to utilize some type of malware or remote exploit to gain access to the database if there are some weaknesses in the actual implementation of the database management. This would allow the adversary to take on the role of an authenticated entity.

Tampering: By circumventing the pink trust boundary, adversaries would have direct access to the database where all event and calendar information is stored. This means adversaries would have the ability to alter any data within the database affecting user calendars and events.

Repudiation: If an adversary can cover their tracks after spoofing, they may have the ability to refute any claims they tampered with or gained access to restricted data, especially since data within the database would be altered often as users add, remove, and edit their events.

Information Disclosure: By having access to the database, all of the users whose information is stored in that database are exposed to the adversary. Sensitive information includes the names of people attending events (possibly their account information), where events are being held, and the specific details of events. This would cause a huge data leak as the whole database is exposed to an unauthorized entity.

Denial of Service: The adversary may disable, destroy or make changes to the database in some way once gaining access to it, causing the Calendar/Reminder API to become unavailable to all users.

Elevation of Privilege: Gaining access to the database means the adversary would have access to a plethora of sensitive information, including both user account information and event information. This could lead to the adversary either using blackmail or another spoofing attack to gain more privilege within the system.

Transferring Discord Token Hijacking

We decided to transfer this threat due to its high threat total calculated by the DREAD evaluation. Additionally, this threat should be easily transferred to the actual Discord platform as users of our system, Discord Calendar/Reminder API, would only be authorized Discord users. Meaning they have gone through Discords preventative and authentication measures.

Transferring Alteration of Individual's Information

Like transferring unauthorized access to individual user accounts, we decided to transfer the threat of alteration of individual account information as it goes with the previously mentioned threat. It shares both a high-risk total calculated by the DREAD evaluation and can easily be transferred to Discord which has its own security measures in place.

Avoiding Unauthorized Access to the Database

We chose to avoid the threat of unauthorized access to the database due to the potential consequences. Despite the score of 22 when utilizing the DREAD evaluation, this threat, although rare, cannot happen as

potential sensitive information in the database can be leaked, leading to a total loss of trust in our users. Additionally, since the database is considered a critical asset, we would like to maintain full control over it rather than transferring the power to a third party.

Mitigating Denial of Service Rooted in an Attack on the Database

We chose to mitigate the possibility of a denial-of-service attack on the database since these attacks are super simple and well published, our system is bound to experience them. The best option is to lessen the likelihood our system is overwhelmed with the traffic by having measures in place, as if the attack is successful in denying our service, it affects all customers.

Figure 37: Threat Prioritization Table

Vulnerability	D	R	E	A	D	Total	Decision
Discord Token Hijacking	5	10	10	2	8	35	Transfer
Denial of Service Rooted in an Attack on the Database	0	5	10	10	10	35	Mitigate
Alteration Individual Event Information	3	10	8	0	8	29	Transfer
Database Breaching	10	3	2	10	1	26	Avoid
Gaining Excessive Privilege through a User Account	5	5	5	3	6	24	Mitigate
Alter Database Information	10	2	0	10	1	23	Mitigate
Unauthorized Access to the System's Database	10	2	0	10	0	22	Mitigate

Reference

[1] "Free Trial Might Be Dangerous." Neil Patel, 2023, neilpatel.com/blog/free-trial-might-be-dangerous/.

[2] "Minimum Viable Product (MVP)." Productboard Glossary, Productboard, 2023, www.productboard.com/glossary/minimum-viable-product-mvp/

[3] "Pros and Cons of API-Driven Architecture." Insights for Professionals, 2023, www.insightsforprofessionals.com/it/software/pros-cons-of-api-driven-architecture

[4] "Pros and Cons to Recruiting College Students and Recent Graduates." HR Daily Advisor, 22 Apr. 2021, https://hrdailyadvisor.blr.com/2021/04/22/pros-and-cons-to-recruiting-college-students-and-recent-graduates/

[5] "SQL Tutorial." W3Schools Online Web Tutorials, 2021. https://www.w3schools.com/sql/.

[6] Discord. "API Docs for Bots and Developers." Discord Developer Portal. Available: https://discord.com/developers/docs/intro.

[7] "DiscordPy API Reference." Read the Docs, 2021.https://discordpy.readthedocs.io/en/stable/discord.html.

[8] Discord. "Discord Guidelines." Available at: https://discord.com/guidelines. Accessed on: Apr. 27, 2023.

[9] Discord. "Safety and Trust." Available at: https://discord.com/safety. Accessed on: Apr. 27, 2023.

[10] Discord. "Setting up Two-Factor Authentication." Available at: https://support.discord.com/hc/en-us/articles/219576828-Setting-up-Two-Factor-Authentication. Accessed on: Apr. 27, 2023.

[11] Discord. "Tips Against Spam and Hacking." Available at: https://discord.com/safety/360044104071-tips-against-spam-and-hacking. Accessed on: Apr. 27, 2023.

[12] J. Vass, "How Discord Handles Two and Half Million Concurrent Voice Users Using Webrtc," Discord, 24-Aug-2021. Available: https://discord.com/blog/how-discord-handles-two-and-half-million-concurrent-voice-users-using-webrtc.

[13] Lucidchart. "370 Use Case Diagram." Lucidchart. Available: https://lucid.app/lucidchart/1ef4690f-636b-4c22-9b65-a72be9a6495b/edit?invitationId=inv_357a55e9-c951-4552-83e2-68a24b678e12&page=8nUP7UGRwm9t#.