CEN4010 – Principles of Software Engineering (Spring 2025)

FINAL PROJECT

Group: Network Ninjas
Members: < Dkartiganer2024@fau.edu , jadynreid2024@fau.edu, dweiss2022@fau.edu >
Discord Bot: Simple Bott

1. Solution Overview

We are developing a 'Study Companion' Discord Bot – a tool designed to assist students with scheduling, study habits, and test preparation through interactive discord commands.

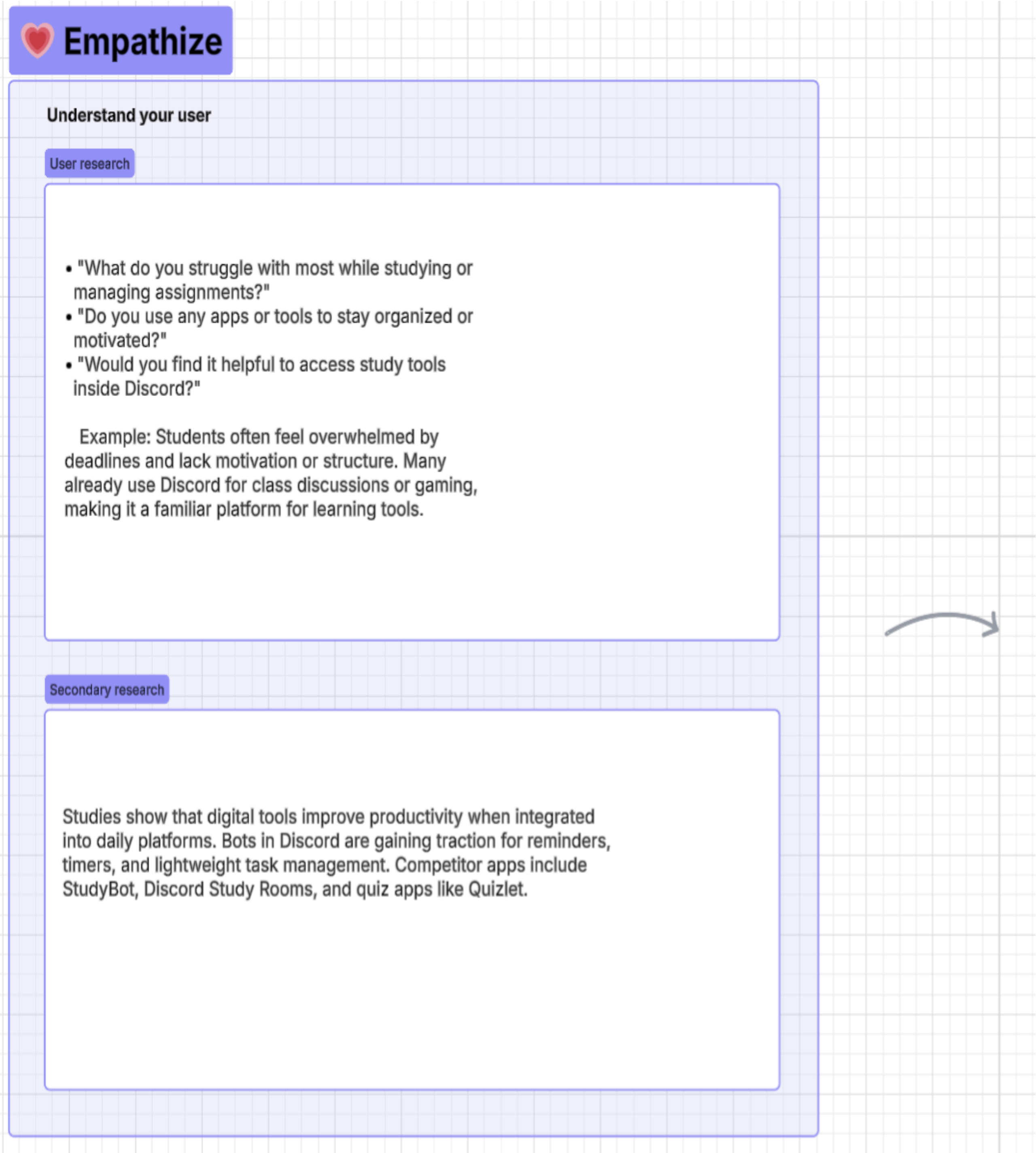
Key features:

- Create flashcards based on study topics.
- Quiz students using created flashcards.
- Create and list study events for student collaboration.
- Set, list, and delete reminders for tasks and deadlines.

The solution works by:

- Accepting commands like !setReminder, !quizme.
- Prompts users for input to generate flashcards, reminders, or events.
- Stores information and uses it later to support the user.
- Generates quizzes using stored flashcards.
- Ensures fast responses for smooth interactions.

2. Design Thinking Process



Empathize

Understand your user

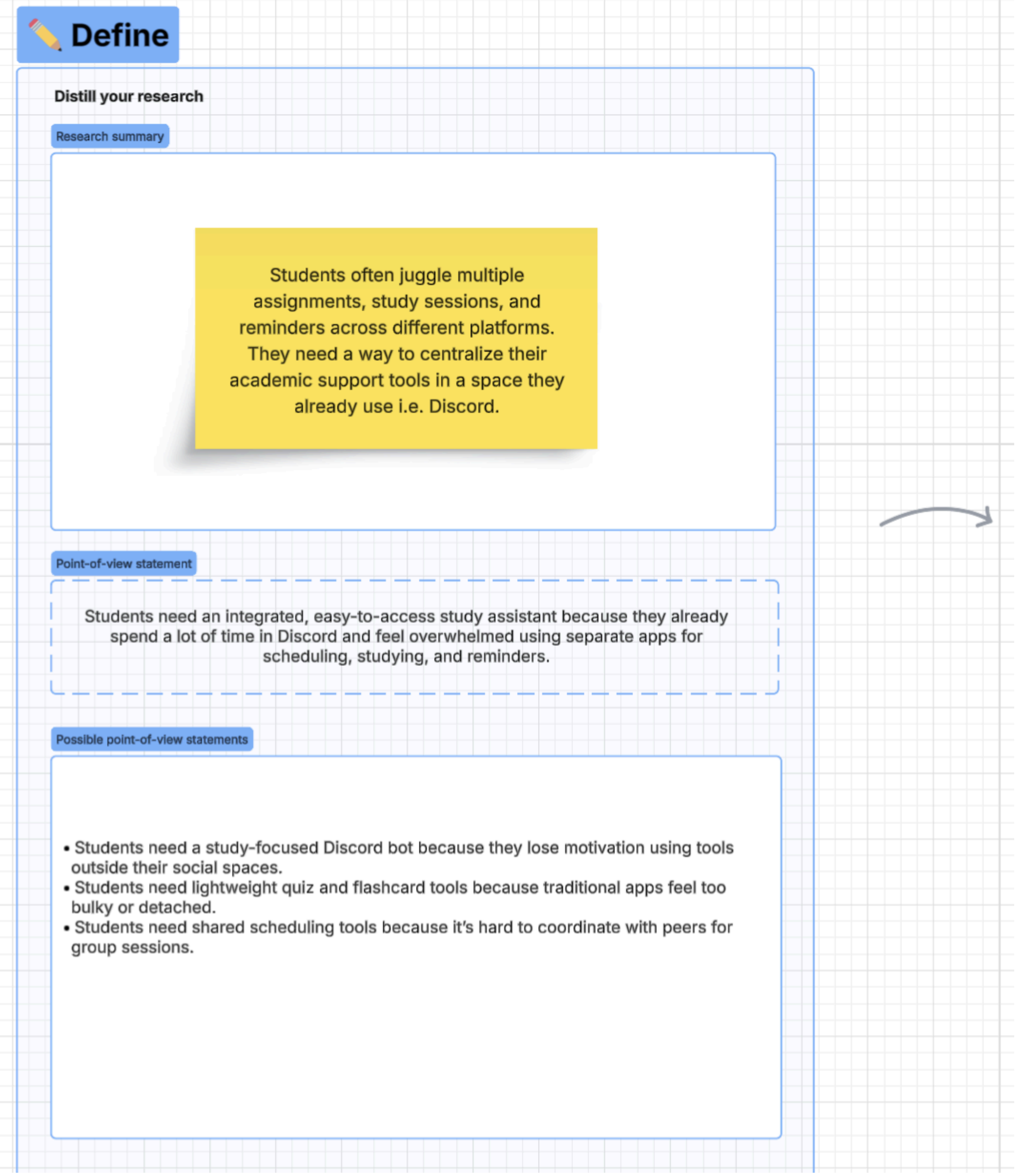
User research

- "What do you struggle with most while studying or managing assignments?"
- "Do you use any apps or tools to stay organized or motivated?"
- "Would you find it helpful to access study tools inside Discord?"

Example: Students often feel overwhelmed by deadlines and lack motivation or structure. Many already use Discord for class discussions or gaming, making it a familiar platform for learning tools.

Secondary research

Studies show that digital tools improve productivity when integrated into daily platforms. Bots in Discord are gaining traction for reminders, timers, and lightweight task management. Competitor apps include StudyBot, Discord Study Rooms, and quiz apps like Quizlet.



✏️ Define

Distill your research

Research summary

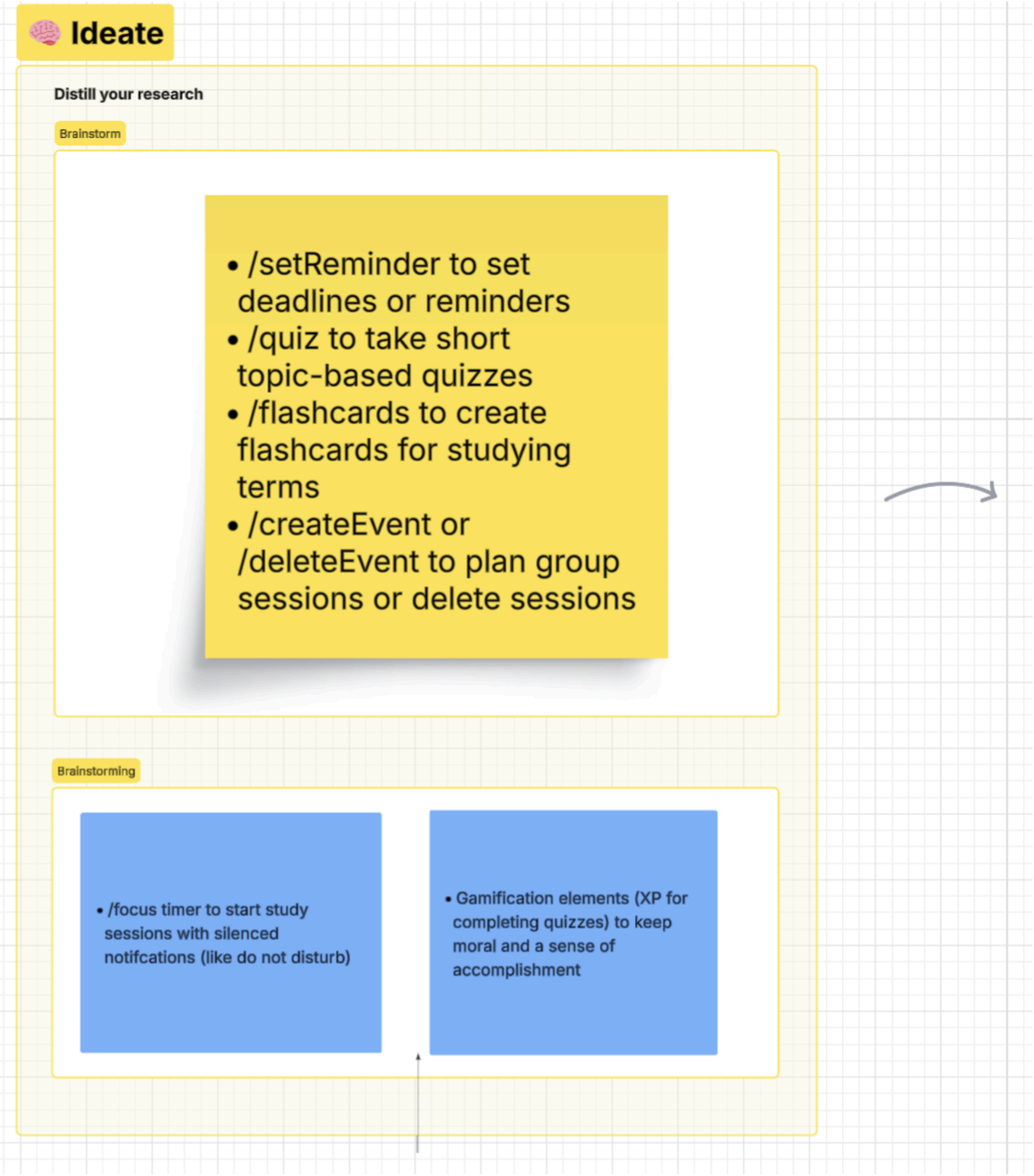
Students often juggle multiple assignments, study sessions, and reminders across different platforms. They need a way to centralize their academic support tools in a space they already use i.e. Discord.

Point-of-view statement

Students need an integrated, easy-to-access study assistant because they already spend a lot of time in Discord and feel overwhelmed using separate apps for scheduling, studying, and reminders.

Possible point-of-view statements

- Students need a study-focused Discord bot because they lose motivation using tools outside their social spaces.
- Students need lightweight quiz and flashcard tools because traditional apps feel too bulky or detached.
- Students need shared scheduling tools because it's hard to coordinate with peers for group sessions.



🧠 Ideate

Distill your research

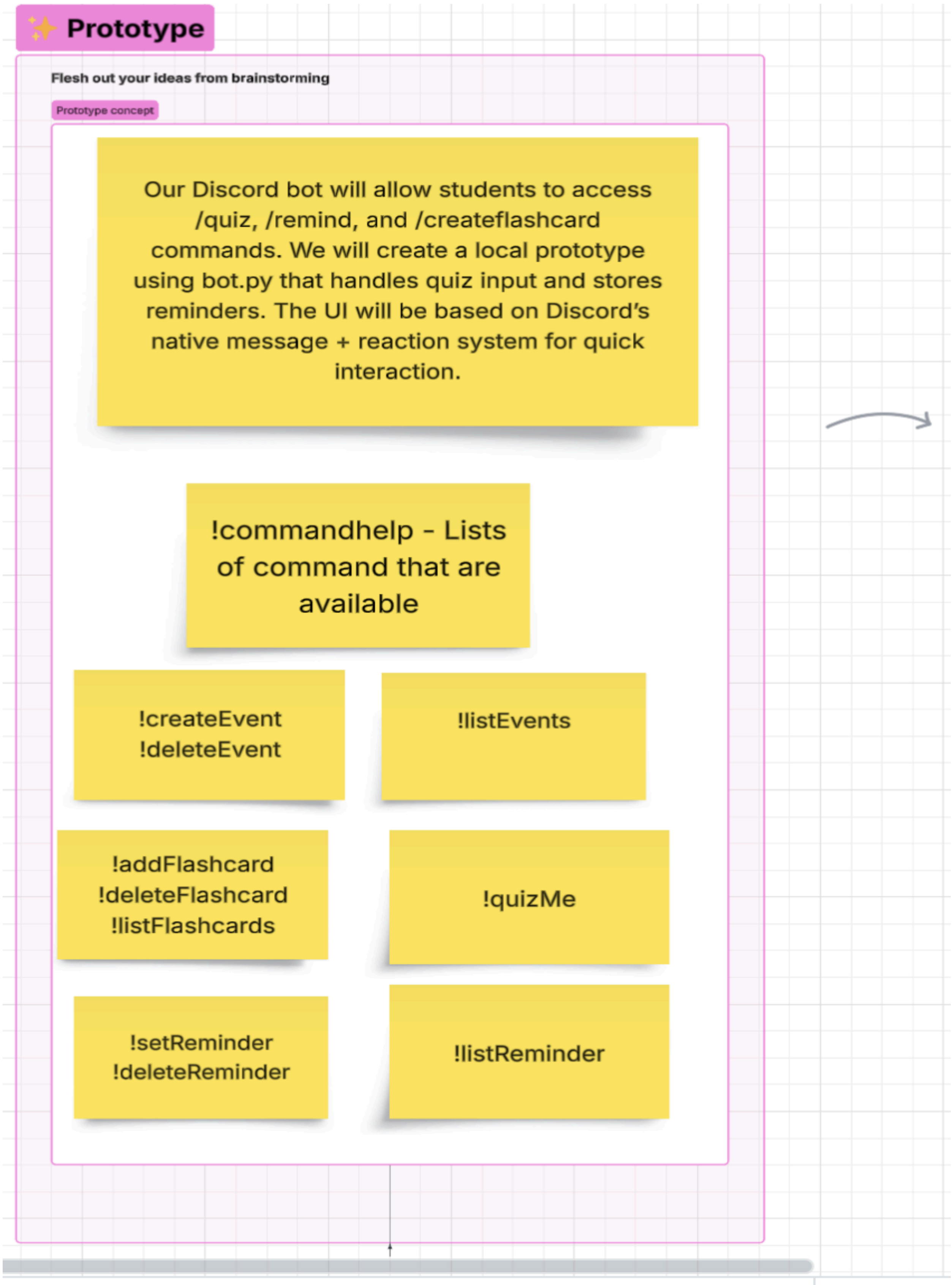
Brainstorm

- /setReminder to set deadlines or reminders
- /quiz to take short topic-based quizzes
- /flashcards to create flashcards for studying terms
- /createEvent or /deleteEvent to plan group sessions or delete sessions

Brainstorming

- /focus timer to start study sessions with silenced notifcations (like do not disturb)

- Gamification elements (XP for completing quizzes) to keep moral and a sense of accomplishment



Prototype

Flesh out your ideas from brainstorming

Prototype concept

Our Discord bot will allow students to access /quiz, /remind, and /createflashcard commands. We will create a local prototype using bot.py that handles quiz input and stores reminders. The UI will be based on Discord's native message + reaction system for quick interaction.

!commandhelp - Lists of command that are available

!createEvent
!deleteEvent

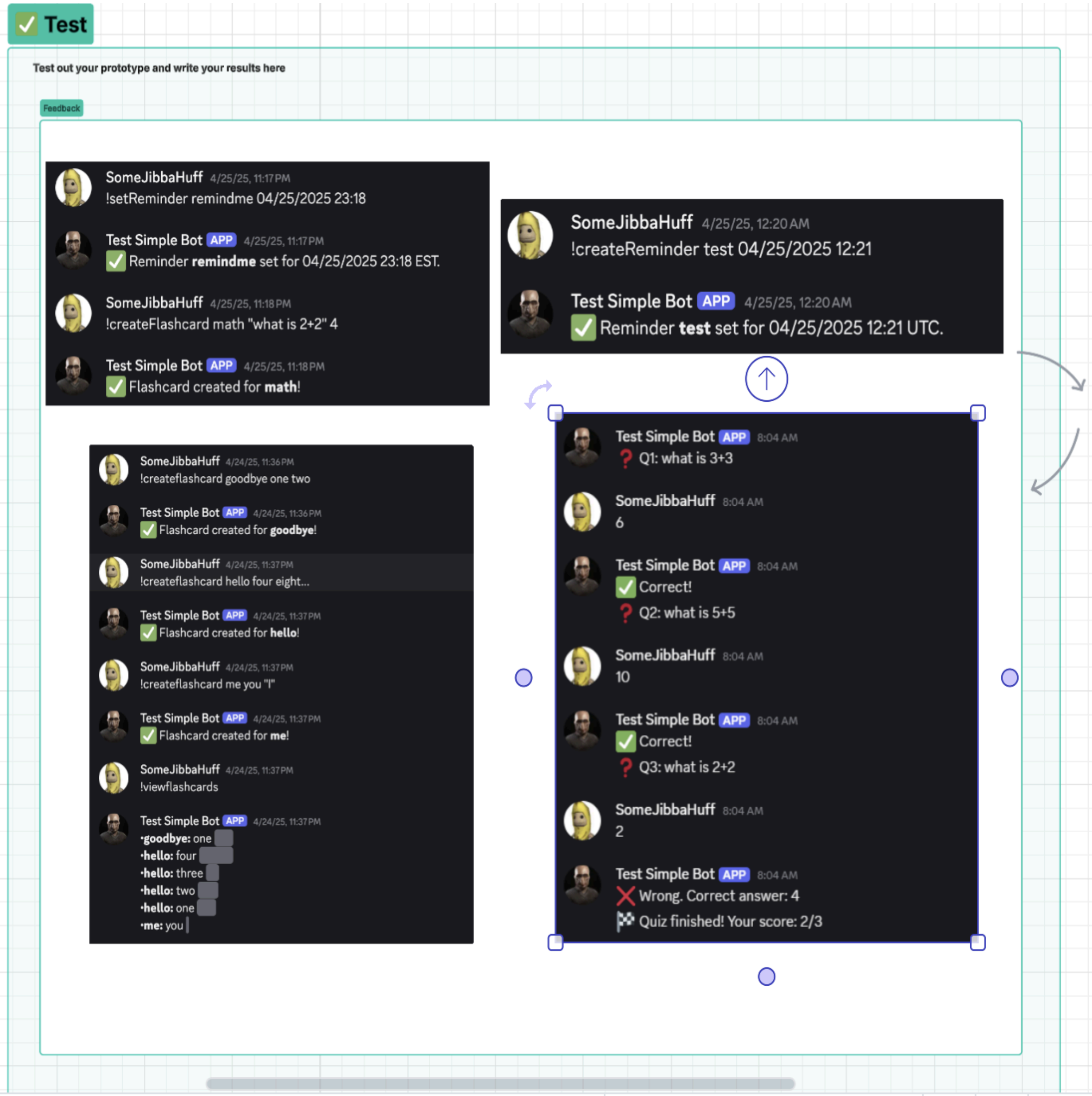
!listEvents

!addFlashcard
!deleteFlashcard
!listFlashcards

!quizMe

!setReminder
!deleteReminder

!listReminder



✅ Test

Test out your prototype and write your results here

SomeJibbaHuff 4/25/25, 11:17 PM
!setReminder remindme 04/25/2025 23:18

Test Simple Bot APP 4/25/25, 11:17 PM
✅ Reminder **remindme** set for 04/25/2025 23:18 EST.

SomeJibbaHuff 4/25/25, 11:18 PM
!createFlashcard math "what is 2+2" 4

Test Simple Bot APP 4/25/25, 11:18 PM
✅ Flashcard created for **math**!

SomeJibbaHuff 4/24/25, 11:36 PM
!createflashcard goodbye one two

Test Simple Bot APP 4/24/25, 11:36 PM
✅ Flashcard created for **goodbye**!

SomeJibbaHuff 4/24/25, 11:37 PM
!createflashcard hello four eight...

Test Simple Bot APP 4/24/25, 11:37 PM
✅ Flashcard created for **hello**!

SomeJibbaHuff 4/24/25, 11:37 PM
!createflashcard me you "I"

Test Simple Bot APP 4/24/25, 11:37 PM
✅ Flashcard created for **me**!

SomeJibbaHuff 4/24/25, 11:37 PM
!viewflashcards

Test Simple Bot APP 4/24/25, 11:37 PM
- **goodbye:** one
- **hello:** four
- **hello:** three
- **hello:** two
- **hello:** one
- **me:** you

SomeJibbaHuff 4/25/25, 12:20 AM
!createReminder test 04/25/2025 12:21

Test Simple Bot APP 4/25/25, 12:20 AM
✅ Reminder **test** set for 04/25/2025 12:21 UTC.

Test Simple Bot APP 8:04 AM
❓ Q1: what is 3+3

SomeJibbaHuff 8:04 AM
6

Test Simple Bot APP 8:04 AM
✅ Correct!
❓ Q2: what is 5+5

SomeJibbaHuff 8:04 AM
10

Test Simple Bot APP 8:04 AM
✅ Correct!
❓ Q3: what is 2+2

SomeJibbaHuff 8:04 AM
2

Test Simple Bot APP 8:04 AM
❌ Wrong. Correct answer: 4
🏁 Quiz finished! Your score: 2/3

3. Requirement Analysis

Functional Requirements:

(10-20 functional requirements aligned with your Solution Overview)

Requirement Description	Priority	Complexity	Expected Outcomes (What the system should do as a result? How to validate it?)	How to implement it? (What is the technique / function / config require to deliver this requirement?
The system should be able to send messages	High	Low	Bot responds to commands with Discord server messages and direct messages	discord.py library to implement Discord API, server.send()
The system should create “flashcards” through commands	High	Medium	Bot creates text string with a given topic, question, and answer and stores it	Create dictionary with topic, question, and answer keys, then stores them in a list
The system should allow users to view their flashcards through commands	High	Low	Bot responds to command with a list of all of the flashcards they made	Create a string by appending all flashcards, then add it to a message and send it
The system should format flashcards so that the answer is hidden until revealed	High	Low	When a flashcard is viewed, the answer is concealed until the user clicks on it	Add spoiler tags to the answer part of the flashcard, then append it to the message
The system should allow users to filter flashcards based on topic	Low	Medium	When viewing flashcards, the bot only lists the flashcards that have the given topic	Check the topic and only append the flashcards with the given topic parameter
The system should allow users to delete flashcards	High	Low	Bot removes flashcard with given topic and question from the list, preventing it from being viewed	Remove flashcard from stored list and delete it
The system should create quizzes through commands	High	High	When command is inputted, a quiz should begin with questions based on a topic, allowing users to answer the questions	Creates quiz using flashcards, sends message with a question and waits for user to answer before sending next question
The system should send an error message if a quiz cannot be created	Medium	Low	If there are no valid flashcards with the given topic, bot sends an error message	Check if there are any flashcards with the topic first before creating the quiz, send error message if not
The system should randomize the order of questions in a quiz	Low	Medium	When a quiz is created, the order of the questions is not necessarily the same as the order of the flashcards	Random function to generate random order of questions in a quiz
The system should score a user’s quiz based on how many correct answers they got	High	Low	At the end of a quiz, the bot sends a message indicating the correct answers over the total questions	Keep track of correct answers and number of questions in quiz, send message with both as a fraction
The system should create server events through commands	High	Low	Bot creates server event with a given name and datetime	discord.py library to implement Discord API, guild.create_scheduled_event()
The system should allow users to view their created server events through commands	Low	Low	Bot displays a list of server events that were created on the server	discord.py library to implement Discord API, guild.fetch_scheduled_events()
The system should create reminders through commands	High	Medium	Bot creates reminder with a given name and datetime and stores it	Create dictionary with name and datetime keys and add it to the list
The system should validate the datetime before creating a reminder or event	High	Low	If the datetime is invalid, the bot will not create that reminder/event and send error message	Check if current datetime is before the reminder datetime and check that the datetime is formatted correctly
The system should frequently check if the datetime for a reminder has been reached	HIgh	Medium	Bot can recognize when a reminder’s datetime has been reached	Bot regularly checks the datetimes of reminders and compares them to the current datetime
The system should execute reminders by directly messaging a user	High	Low	When the datetime of the reminder is reached, the bot messages the user with the name of the reminder and removes it from the list	discord.py library to implement Discord API, user_id.send(), then remove reminder from the list
The system should allow users to view their upcoming reminders through commands	Medium	Low	Bot displays a list of reminders that the user created	Retrieve stored list of reminders, append each name and datetime to a message and send it

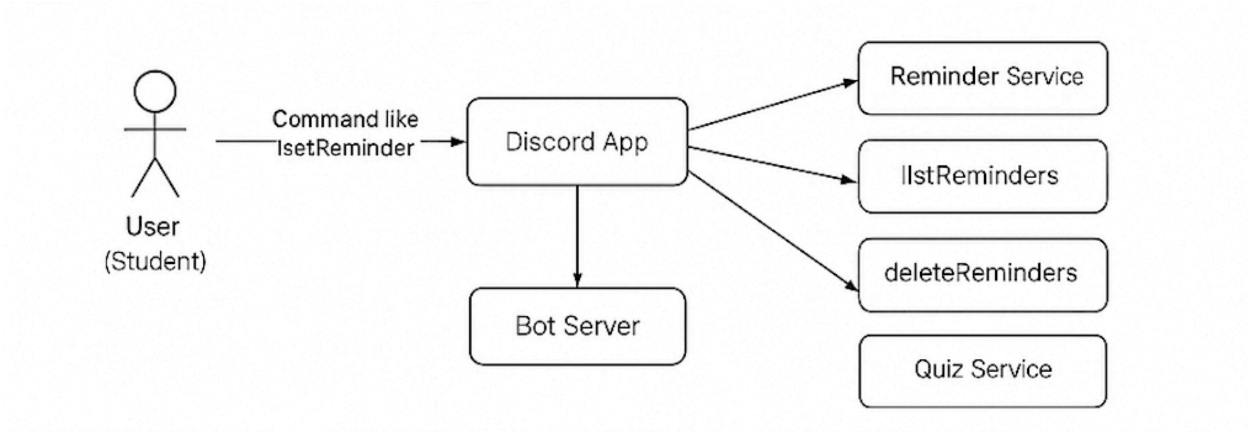
The system should allow users to delete reminders through commands	Medium	Low	Bot removes reminder from the list and does not message the user at the datetime	Remove reminder from stored list and delete it

Non-Functional Requirements

(5-10 non-functional requirements for your Solution)

Requirement Description	**Priority**	**Complexity**	**Expected Outcomes** **(What the system should do as a result? How to validate it?)**	**How to implement it?** **(What is the technique / function / config require to deliver this requirement?)**
The system should ensure question order feels random over multiple quizzes	Low	High	When two quizzes are created for the same topic, the questions should be ordered differently in each of them	Keep track of previous order for topic, random function to randomize order, randomize again if order is the same
The system should be lenient in how commands are inputted	Low	Low	Commands should allow any capitalization and still execute properly	Discord API, case_Insensitive = true
The system should format lists so they are easily readable	Medium	Low	When a list is viewed with commands, each item is separated by a line break and a bullet point for readability	Create strings with bullet points and line breaks, and append them to the message
The system should ensure quiz questions stand out compared to other messages	Low	Low	When a quiz question is asked, the text stands out with a number	Create strings with a number corresponding to the index of the question, then send the message
The system should be efficient when executing reminders	Medium	High	The bot sends a direct message at the reminder's set datetime with at most a second's delay	Sort the reminder list by datetime so only the first datetime needs to be frequently checked, then immediately remove it from the list when the datetime is reached
The system should ensure speedy quiz creation	Medium	High	The bot creates a quiz and sends the first question with minimal delay	Ensure that the flashcard list is sorted by topic so that retrieval is faster, then randomize and send from that smaller list
The system should ensure that lists are not cut off by Discord's character limit for readability	Low	High	When a list is longer than 2,000 characters, an additional message is sent containing the remainder of the list	Count the number of characters in the string each time an item is appended, then when the limit is about to be reached, send a new message containing the rest of the items and restart the character count

4. Solution Architecture



<u>Explanation:</u>

The system is structured as follows:

- **User (Student):** Sends commands (e.g., !createEvent)
- **Discord App:** Sends commands to the bot
- **Bot Server (Simple Bott):** Processes commands
    - **Command Parser:** Reads the command (like !quizStart) and figures out what to do.
    - **Event Service:** Sets and keeps track of reminders like “Study at 5PM”.
    - **Quiz Service:** Sends quiz questions based on flashcards to help students study.

Flow:

- User → Discord → Bot → Service → Response → User

Why?

- We need all these parts because:
    - Students need a place to ask for help.
    - Discord is the app that is already widely used amongst students.
    - The bot server does the hard work.

    - The database saves everything so it’s not lost and users can come back to it whenever needed.

Prompt:

Create a solution architecture diagram for Simple Bott, a Discord bot that helps students with academic tasks. Include the following:

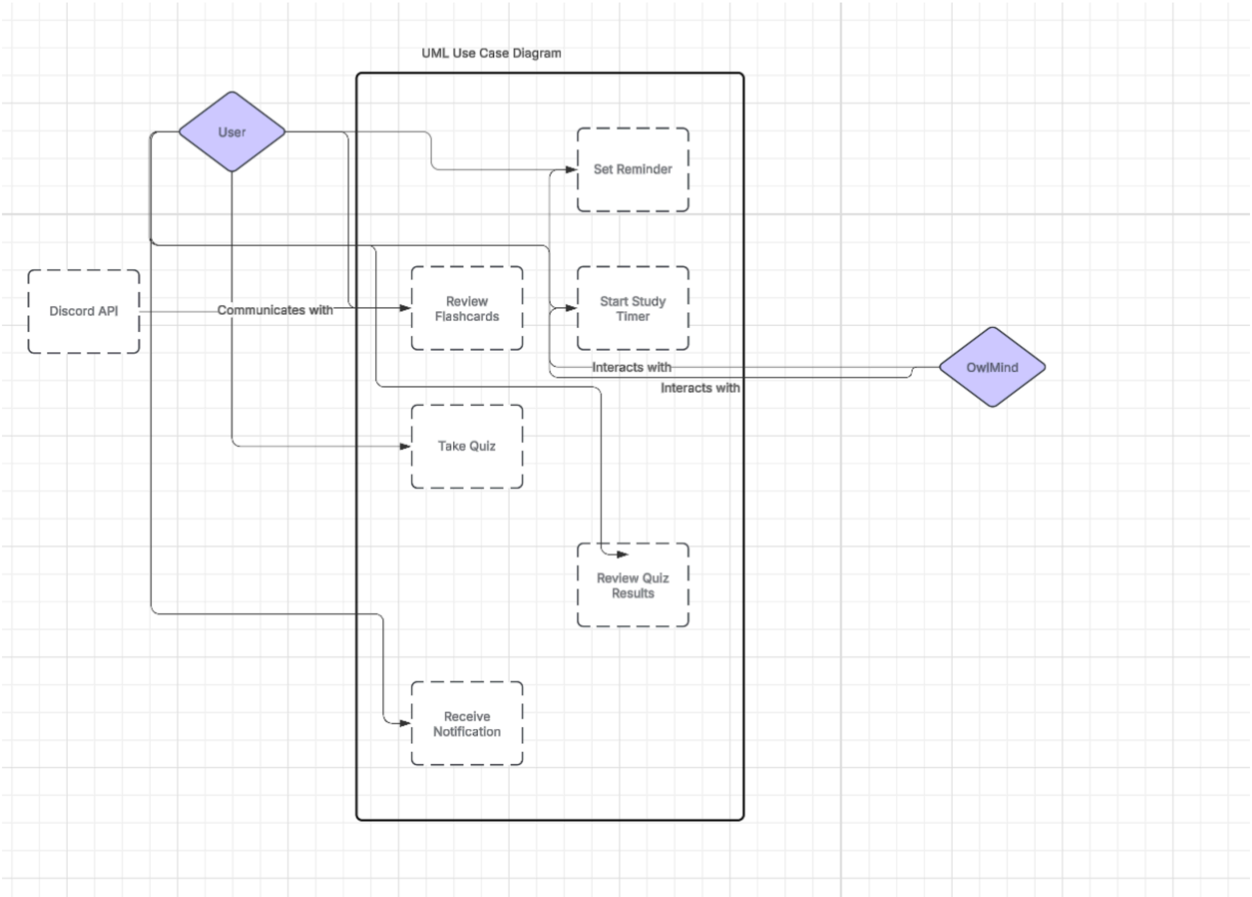
1. User (Student) – Sends commands like !setReminder.
2. Discord App – Receives commands and sends them to the bot server.
3. Bot Server (Simple Bott) – Processes commands and includes:
    - Command Parser – Handles commands like !setReminder, !listReminders, etc.
    - Reminder Service – Manages reminders.
    - Quiz Service – Handles quizzes.

Connections:

- User → Discord App (Sends command)
- Discord App → Bot Server (Processes command)
- Bot Server → Services (Handles tasks)
- Services → Database (Stores/retrieves data)
- Bot Server → Discord App → User (Sends response)

5. System Modeling

5.1 UML Use Case Diagram



Explanation :

This diagram shows six use cases that Simple Bott supports to help students manage their academic tasks:

1. Set Reminder is there to provide a way for users to keep track of important tasks or deadlines by scheduling reminders that the bot will later deliver.
2. Start Study Timer is there to provide a tool for focused studying.
3. Review Flashcards is there to provide an easy way for students to study and memorize key terms or concepts by going over flashcards they created earlier.
4. Take Quiz is there to provide a self-testing feature that quizzes students using the flashcards they made, helping them prepare for exams or review topics.
5. Review Quiz Results is there to provide feedback after a quiz, so the student can see what they got right or wrong and improve.

6. Receive Notification is there to provide timely alerts and reminders, so the student doesn’t miss any events, reminders, or deadlines.

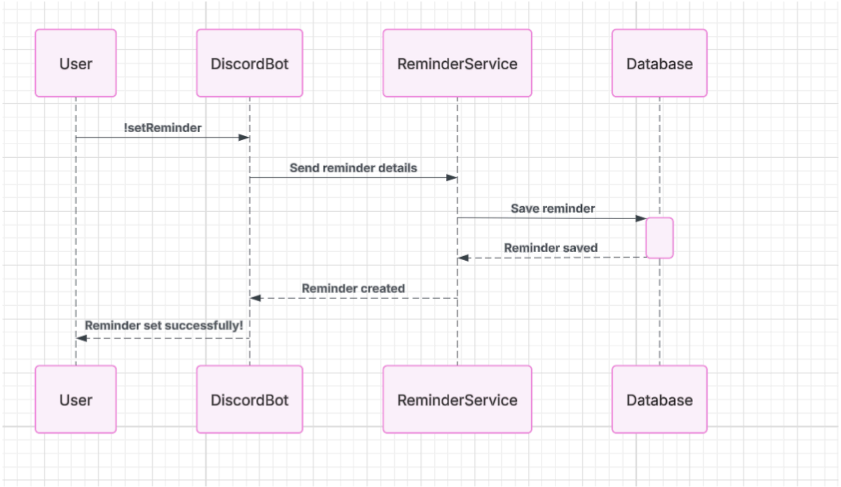
These use cases are all accessed by the User, and the Discord API handles the communication between the user and the bot. The entire system is managed within the Simple Bott boundary. It can optionally connect with OwlMind services in the future for AI enhancements.

Prompt:

- Create a use case diagram for or Simple Bott, a Discord bot that helps students with academic tasks. Include:
- Actors: User, Discord API, Owlmind
- System Boundary
    - Label: OwlMind Bot
    - Place inside:
    - Set Reminder
    - Start Study Timer
    - Review Flashcards
    - Take Quiz
    - Review Quiz Results
    - Receive Notification
- Connections:
    - User → all six use cases (solid lines)
    - User → Discord API (“Communicates with”)
    - OwlMind Bot → OwlMind (“Interacts with”)

-

5.2 UML Sequence Diagram(s)



Explanation:

This interaction works as each part of the bot has a specific role to help the system run smoothly. First, the user sends the !setRemindercommand through Discord. The Discord bot receives the command and recognizes it needs to create a reminder. It then forwards the reminder details (such as the task and time) to the ReminderService, which is responsible for managing reminders. The ReminderService saves the reminder into the database. Once the database confirms the reminder has been successfully stored, the ReminderService notifies the bot. Finally, the bot responds back to the user with a message saying "Reminder set successfully!" Each component plays a distinct role: the user initiates the action, the bot parses and routes the request, the ReminderService handles the logic, and the database ensures persistence. This design keeps the system organized, easy to maintain, and ensures a fast and reliable user experience.

Prompt:

Create a UML sequence diagram for the command `!setReminder` in a Discord bot.

Participants:

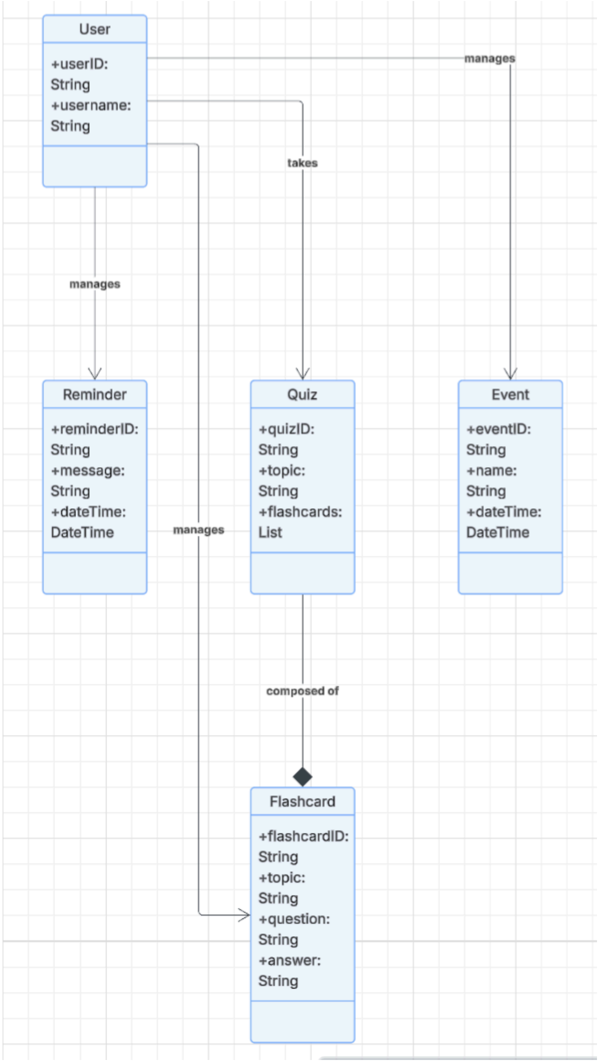
- User
- Discord Bot
- ReminderService

- Database

Flow:

1. User sends the command `!setReminder` through Discord.
2. Discord Bot receives the command.
3. Bot sends the reminder details to the ReminderService.
4. ReminderService saves the reminder to the Database.
5. Database confirms the reminder is saved.
6. ReminderService notifies the Bot that the reminder is created.
7. Bot sends a message back to the User: "Reminder set successfully!"

5.3 UML Class Diagram



Explanation:

This class diagram reflects how Simple Bott organizes and manages user data behind the scenes. Each class represents a key element of the bot's functionality. The User class stores the user's ID and username, linking all actions back to the student using the bot. The Reminder class contains the details of scheduled tasks, including the message and date/time. The Event class handles group study events, structured similarly to reminders but intended for collaborative scheduling.

The Flashcard class holds individual study cards, each with a topic, question, and answer. These flashcards are then grouped by the Quiz class, which generates quizzes based on a chosen topic. The Quiz class maintains a list of associated flashcards.

The relationships between classes are straightforward: a single User can have many Reminders, Events, Flashcards, and Quizzes, while each Quizis composed of multiple Flashcards. This structure keeps data organized, scalable, and makes it easy for each bot feature to interact with the others in a consistent way.

Prompt:

Prompt for Lucid : Create a UML class diagram for a Discord bot that helps students manage reminders, events, flashcards, and quizzes.

Classes:

1. User
    - userID: String
    - username: String

2. Reminder
    - reminderID: String
    - message: String
    - dateTime: DateTime

3. Event
    - eventID: String
    - name: String
    - dateTime: DateTime

4. Flashcard
    - lashcardID: String
    - topic: String
    - question: String
    - answer: String

5. Quiz
    - quizID: String
    - topic: String
    - flashcards: List<Flashcard>

Relationships:

- A User can have multiple Reminders

- A User can have multiple Events
- A User can have multiple Flashcards
- A User can take multiple Quizzes
- A Quiz is composed of multiple Flashcards

6. Prototype Implementation Plan

To build ‘Simple Bott’, the plan was to use Python along with the discord.py library to handle user interaction inside Discord servers. Our bot works by listening for specific keyword-based commands from users. When a command is recognized, the bot prompts the user for additional information if needed, processes the input, stores it in memory, and then sends back a response.

Each user interaction follows a simple pattern:

> User command → Bot interprets → Bot prompts (if needed) → User input → Bot processes → Bot stores → Bot responds

These are some of the commands we implemented:

- !commandHelp -> To display the command list
- !createReminder -> To ask for task and date
- !quizme -> Prompt for topic, then start quiz
- !createFlashcard -> Asks for topic, question, answer
- !listFlashcards -> Shows all user flashcards
- !createEvent -> Asks for event name, date and time

All reminders, events, flashcards, and quizzes are currently stored in memory (in lists or dictionaries), meaning they are temporary and reset when the bot restarts. The bot is designed to respond quickly to user inputs and provide simple, easy-to-follow prompts for a smooth experience.

The bot handles validation for user inputs, such as checking if a date and time are in the correct format before creating reminders or events. For quizzes, the bot randomizes the order of flashcards within the selected topic to create a new experience every time.

At this stage, Simple Bott is fully functional for managing study reminders, events, flashcards, and quizzes using Discord commands.

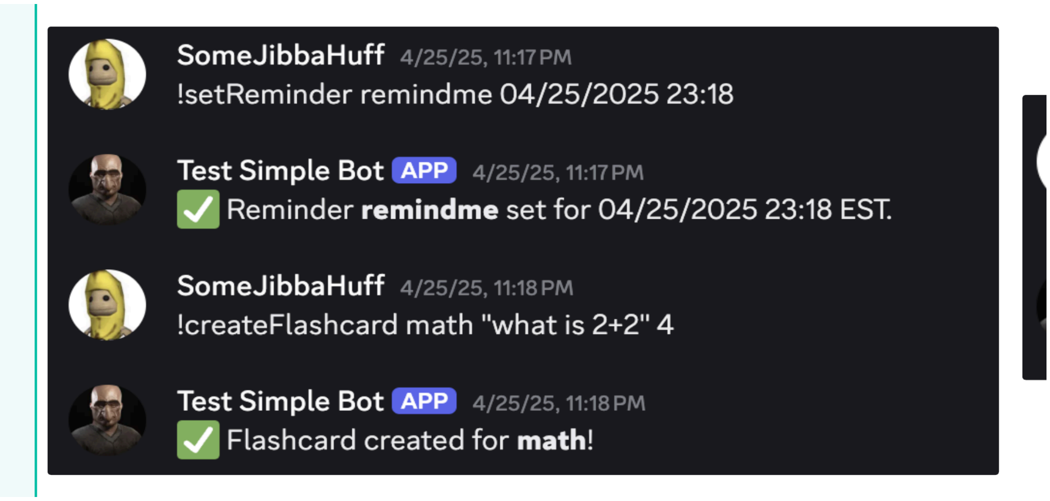
Possible future ideas: AI Integration

While our current bot works using simple commands, in the future we might like to make it more advanced by adding AI features. This would allow Simple Bott to automatically create flashcards or quizzes from class notes or textbook content that users upload. That would give the possibility to explore Retrieval-Augmented Generation (RAG), which would help the bot find useful information and turn it into study material. This would make studying easier by reducing how much users need to type or enter manually.

7. (Optional) Experimental Prototype

In case you built a prototype 🎉🎉🎉

- Link to Github - https://github.com/Dkartiganer2024/Simple-Bott/tree/main
- Screenshots- More on bottom of page. -

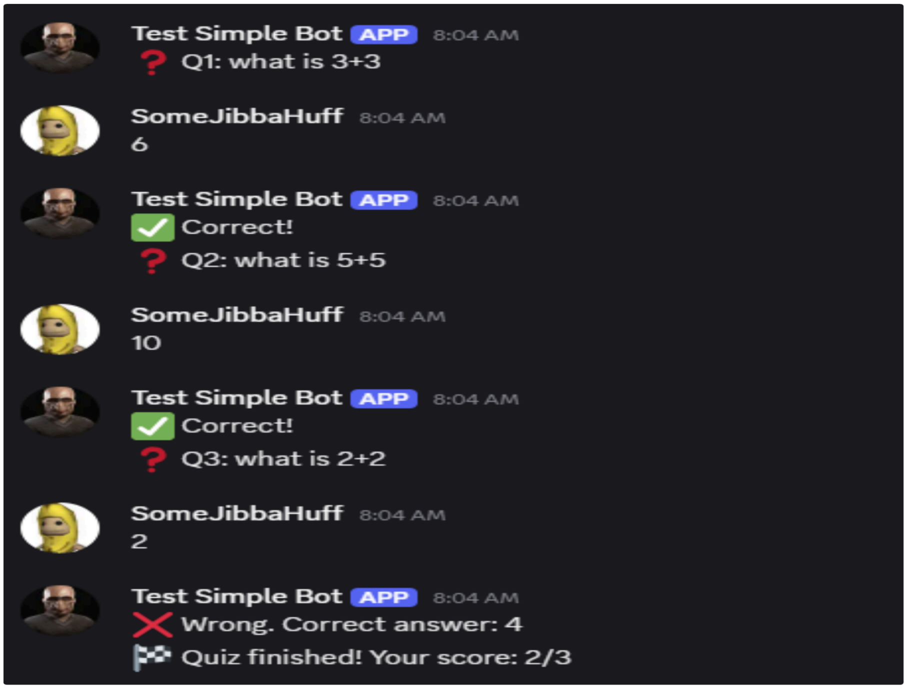


SomeJibbaHuff 4/25/25, 11:17 PM
!setReminder remindme 04/25/2025 23:18

Test Simple Bot APP 4/25/25, 11:17 PM
✅ Reminder **remindme** set for 04/25/2025 23:18 EST.

SomeJibbaHuff 4/25/25, 11:18 PM
!createFlashcard math "what is 2+2" 4

Test Simple Bot APP 4/25/25, 11:18 PM
✅ Flashcard created for **math**!



Test Simple Bot APP 8:04 AM
❓ Q1: what is 3+3

SomeJibbaHuff 8:04 AM
6

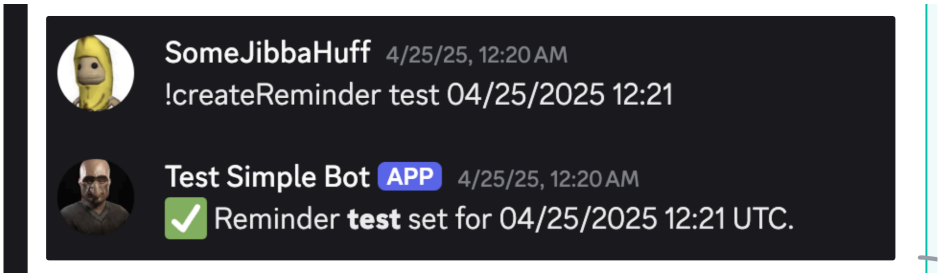
Test Simple Bot APP 8:04 AM
✅ Correct!
❓ Q2: what is 5+5

SomeJibbaHuff 8:04 AM
10

Test Simple Bot APP 8:04 AM
✅ Correct!
❓ Q3: what is 2+2

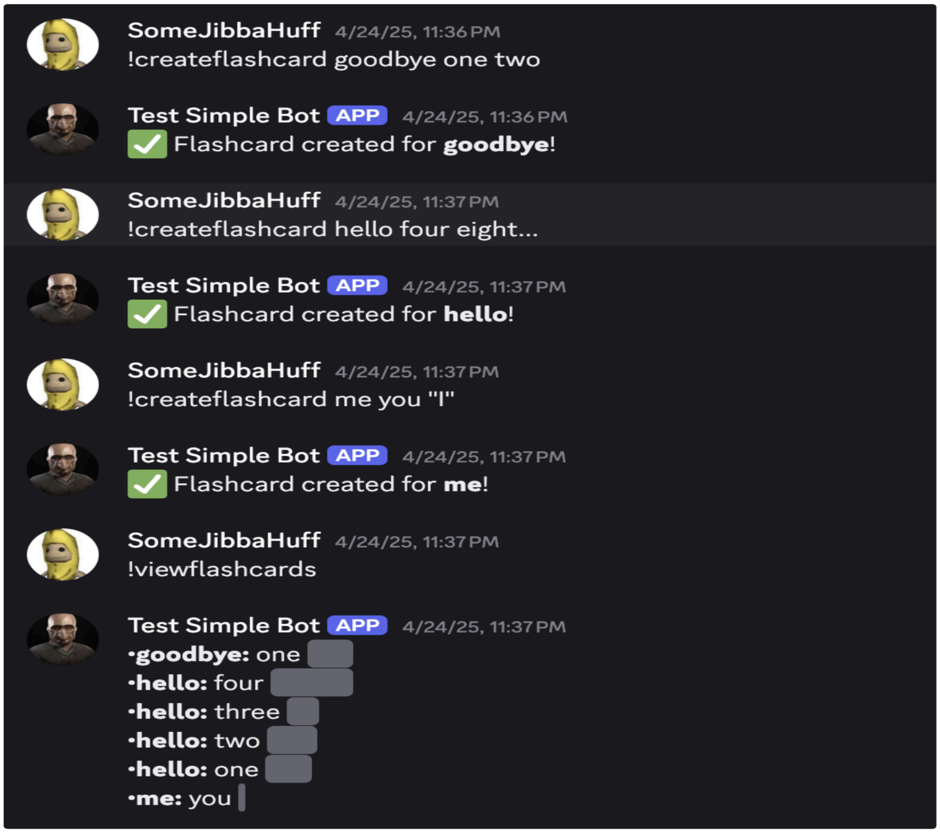
SomeJibbaHuff 8:04 AM
2

Test Simple Bot APP 8:04 AM
❌ Wrong. Correct answer: 4
🏁 Quiz finished! Your score: 2/3



SomeJibbaHuff 4/25/25, 12:20 AM
!createReminder test 04/25/2025 12:21

Test Simple Bot APP 4/25/25, 12:20 AM
✅ Reminder **test** set for 04/25/2025 12:21 UTC.



SomeJibbaHuff 4/24/25, 11:36 PM
!createflashcard goodbye one two

Test Simple Bot APP 4/24/25, 11:36 PM
✅ Flashcard created for **goodbye**!

SomeJibbaHuff 4/24/25, 11:37 PM
!createflashcard hello four eight...

Test Simple Bot APP 4/24/25, 11:37 PM
✅ Flashcard created for **hello**!

SomeJibbaHuff 4/24/25, 11:37 PM
!createflashcard me you "I"

Test Simple Bot APP 4/24/25, 11:37 PM
✅ Flashcard created for **me**!

SomeJibbaHuff 4/24/25, 11:37 PM
!viewflashcards

Test Simple Bot APP 4/24/25, 11:37 PM
- **goodbye:** one
- **hello:** four
- **hello:** three
- **hello:** two
- **hello:** one
- **me:** you

- How to Make It Available

You can make Simple Bot available by:

- Running locally: Simply run `python your-bot-file.py` on your machine. (Bot stays online while your computer is on.)

- Cloud hosting (recommended): Deploy it to a free service like [Render](https://render.com/), [Railway](https://railway.app/), or [Replit](https://replit.com/) for 24/7 uptime.

- VPS hosting: Host it on a private server (AWS, DigitalOcean) for more control and reliability.

For small projects, free cloud services are usually enough.