Discord SET Bot

Midterm Project Report

Alex Algazi
asa10@hood.edu

4/4/22

What is Discord?

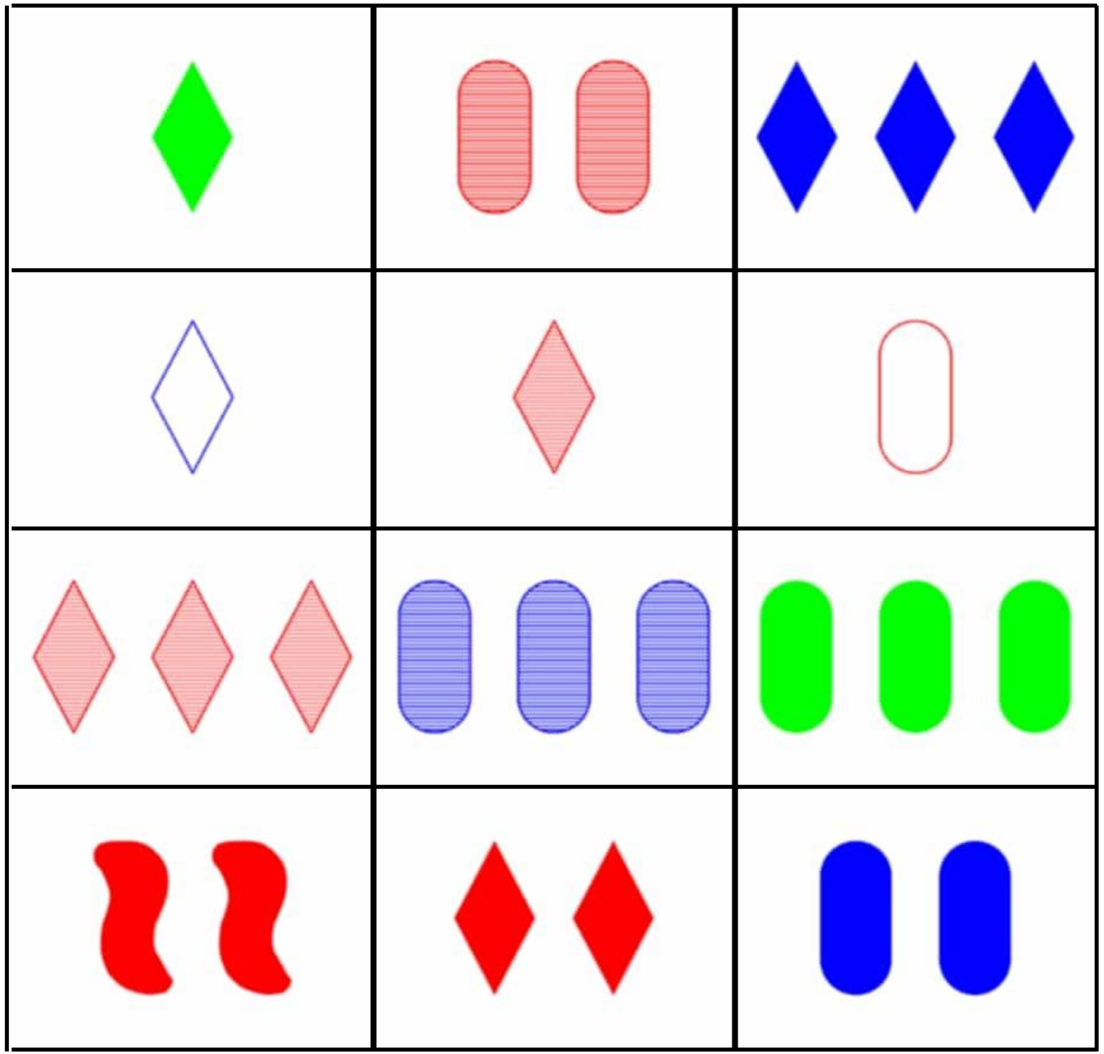
- VoIP and instant messaging platform released in May of 2015
- Users congregate in communities called guilds, or “servers”
- Includes an extensive API for developing bots for the platform
  - The capabilities of bots range from simple call-and-response text messaging to bots that facilitate music streaming into voice channels
- Initially popular among video game enthusiasts, Discord is now regarded as an all-purpose communication platform by many millennials and Gen Z-ers

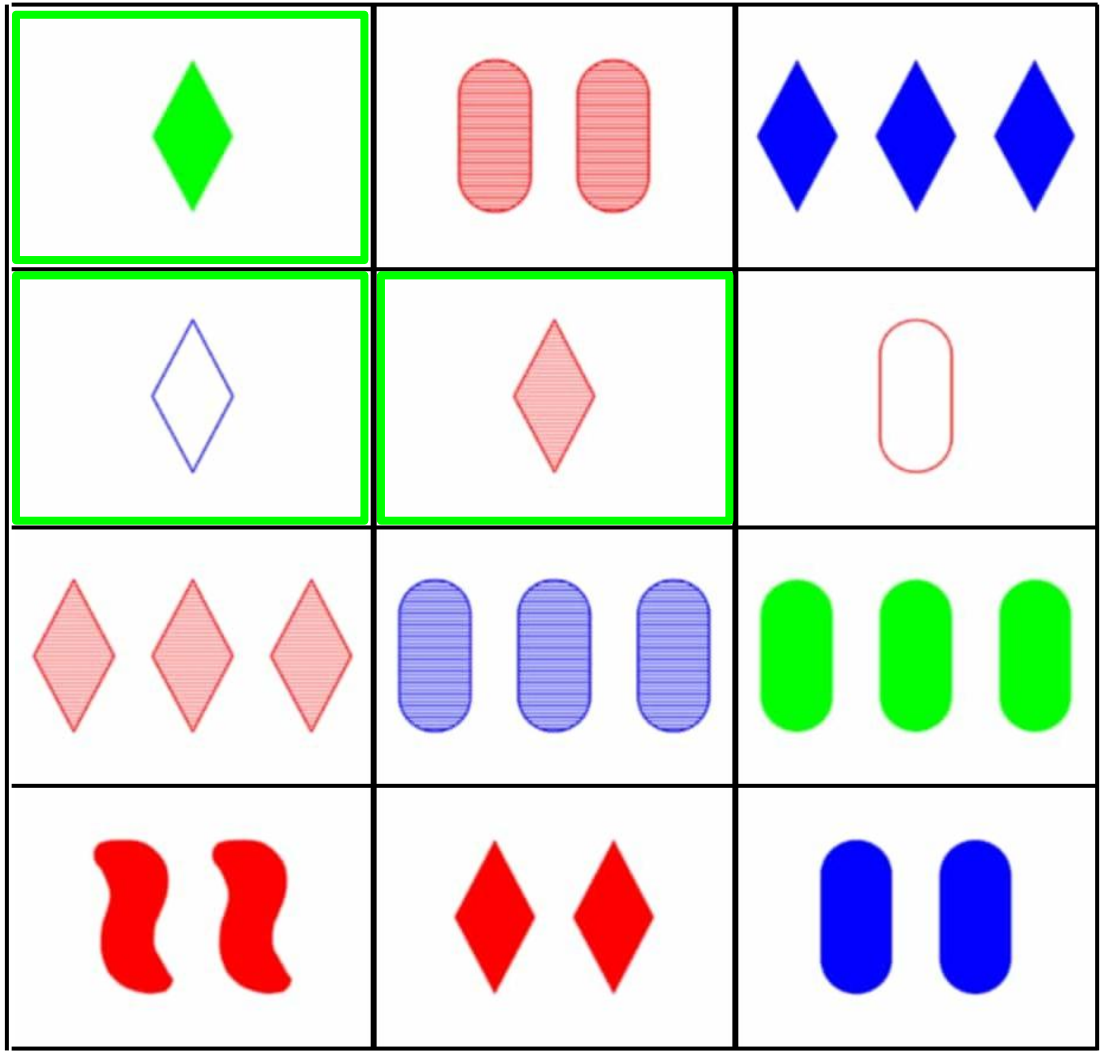
What is SET?

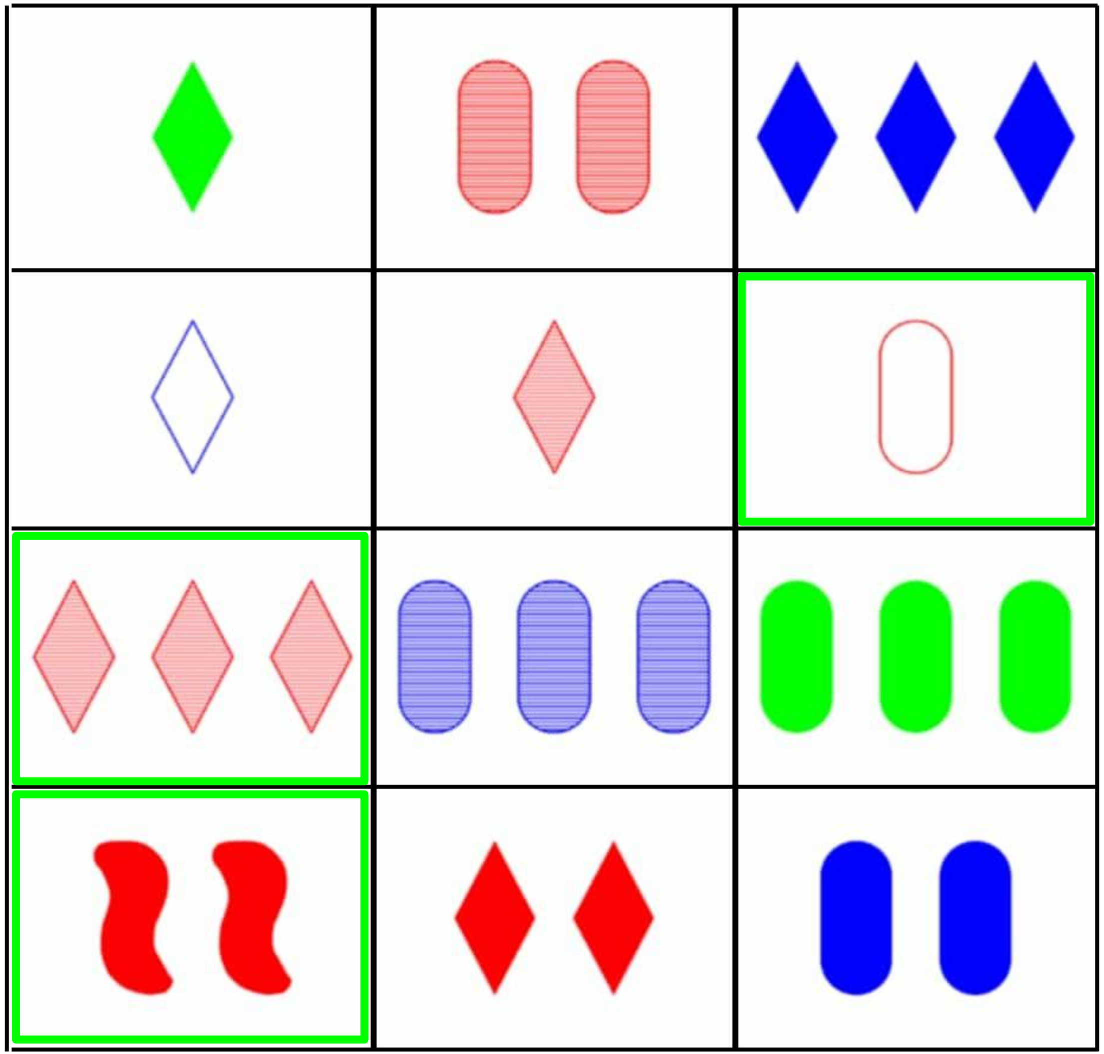
- Pattern-recognition card game invented by geneticist Marsha Falco in 1974
- Each card has four attributes: number, shading, color, and shape
- There are three possibilities for each attribute:

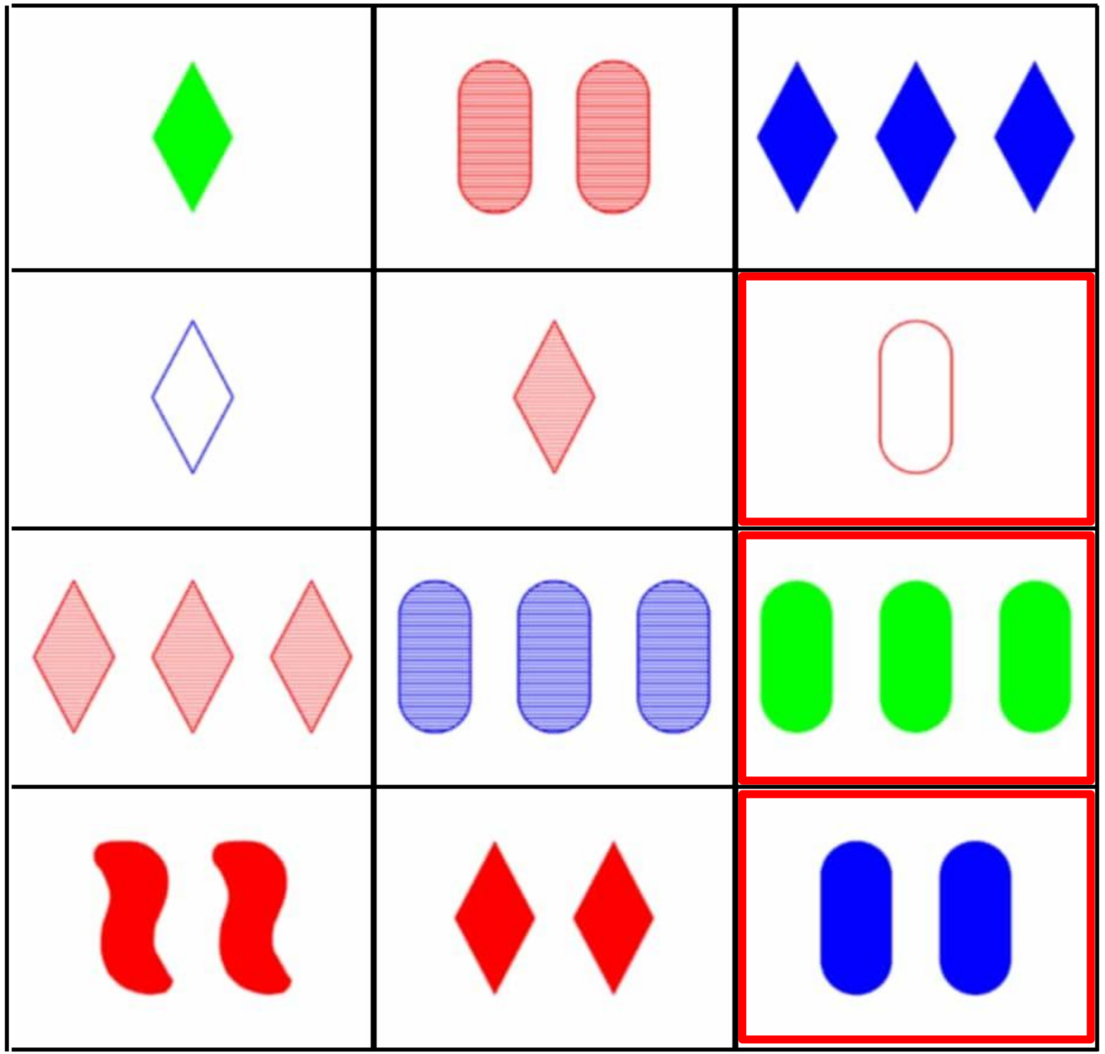
Number:	One, Two, Three
Shading:	Solid, Striped, Open
Color:	Red, Green, Blue
Shape:	Oval, Squiggle, Diamond

- There are 81 cards in the deck, one for each possible combination (3^4)
- A set is a group of three cards where each attribute is **either** the same for all three cards **OR** different for all three cards
- Players compete to find sets on a face-up group of cards called a board









Objectives

1. Generate a board of 12 cards face-up, such that all users in a guild can see
2. Recognize a correct 3-card set using arithmetic
3. Systematically check a board for sets, and add 3 cards if no set is found
4. Take user inputs using Discord emoji reactions
5. Regenerate the board after a set is found, such that the board always has at least 12 cards
6. Show user scores in an ordered leaderboard when the game ends
7. Track relevant statistics in each guild so users can review their performance alongside other users in their guild

Background

- Bots have existed on the platform since December 2015, when Discord launched their unofficial API
- Common use cases for bots at that time included an early music streaming bot, a dice-rolling bot, and bot which scans the internet for funny images
- In recent years, bots have been made for all sorts of purposes, including games. Notable examples of game bots include:
  - Pokétwo - a bot that emulates the experience of playing a Pokémon game
  - EPIC RPG - a bot which guides users through a role-playing game with dungeons, an in-game economy, and player-versus-player interactions
  - AniGame - a collectible card game where users can assemble teams of fighters to fight through randomly generated dungeons, or battle it out against other players
  - Gamebot - a collection of games for Discord guilds, including Chess, Connect 4, Cards Against Humanity, and various others
- However, no one has ever attempted to make a bot for the SET card game…

Project Plan

- Create a logic loop that will enable the game to be played until there are no cards remaining in the deck
- Encode the cards such that proposed sets can be checked mathematically
- Stitch images of the cards together to assemble randomly generated rows
- Use reaction collectors to record user inputs, and run set-checking on each user’s last 3 selected cards
- Check the board for sets in an efficient manner
- Store game data in a JSON file that can be accessed at any point
- Send game data to a database for later retrieval
- Host the bot on Hood’s “pluto” server

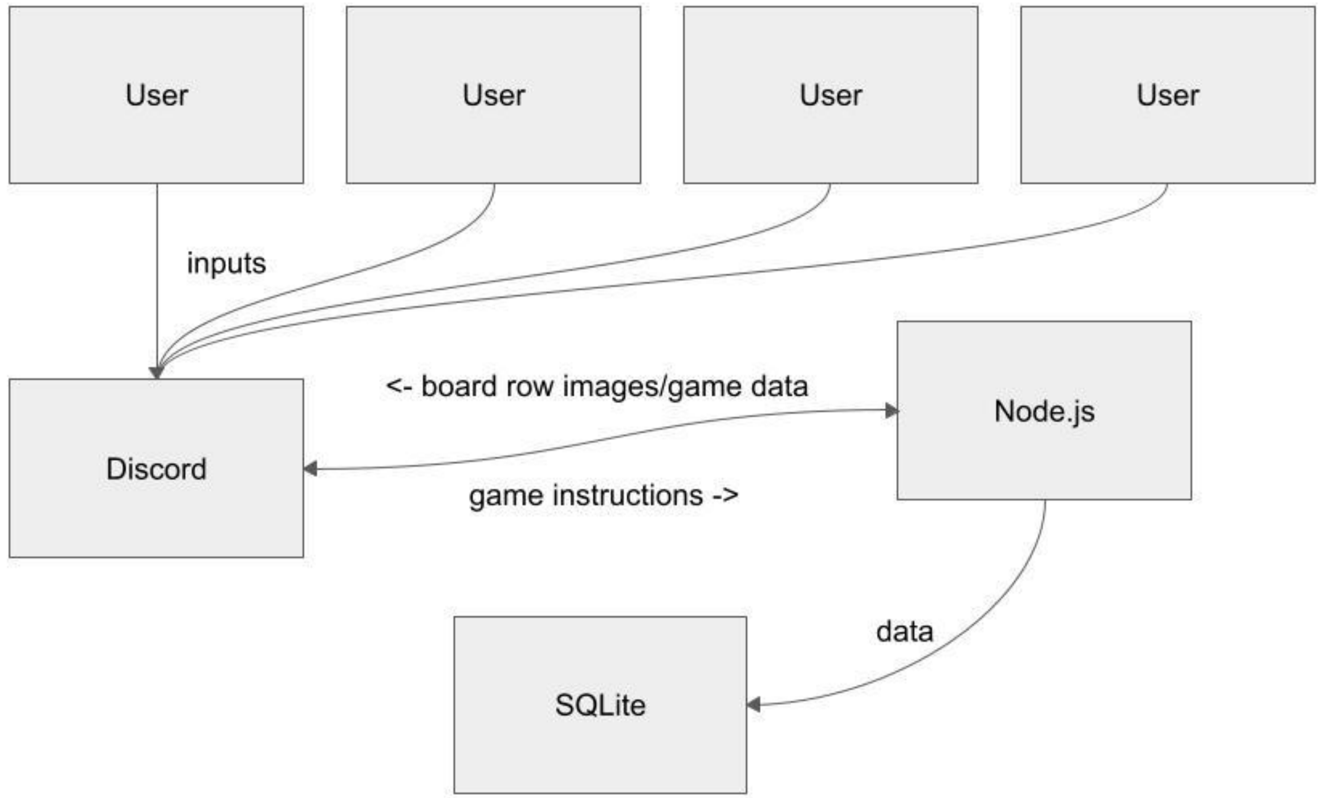
Tech Stack

- JavaScript, for the game logic
- Node.js, a server-side JavaScript runtime environment
- Discord.js, a Node framework for using Discord API calls
- SQLite, to store game data as well as player scores in a discrete file
- pluto@hood.edu, as a host for the bot and all its data

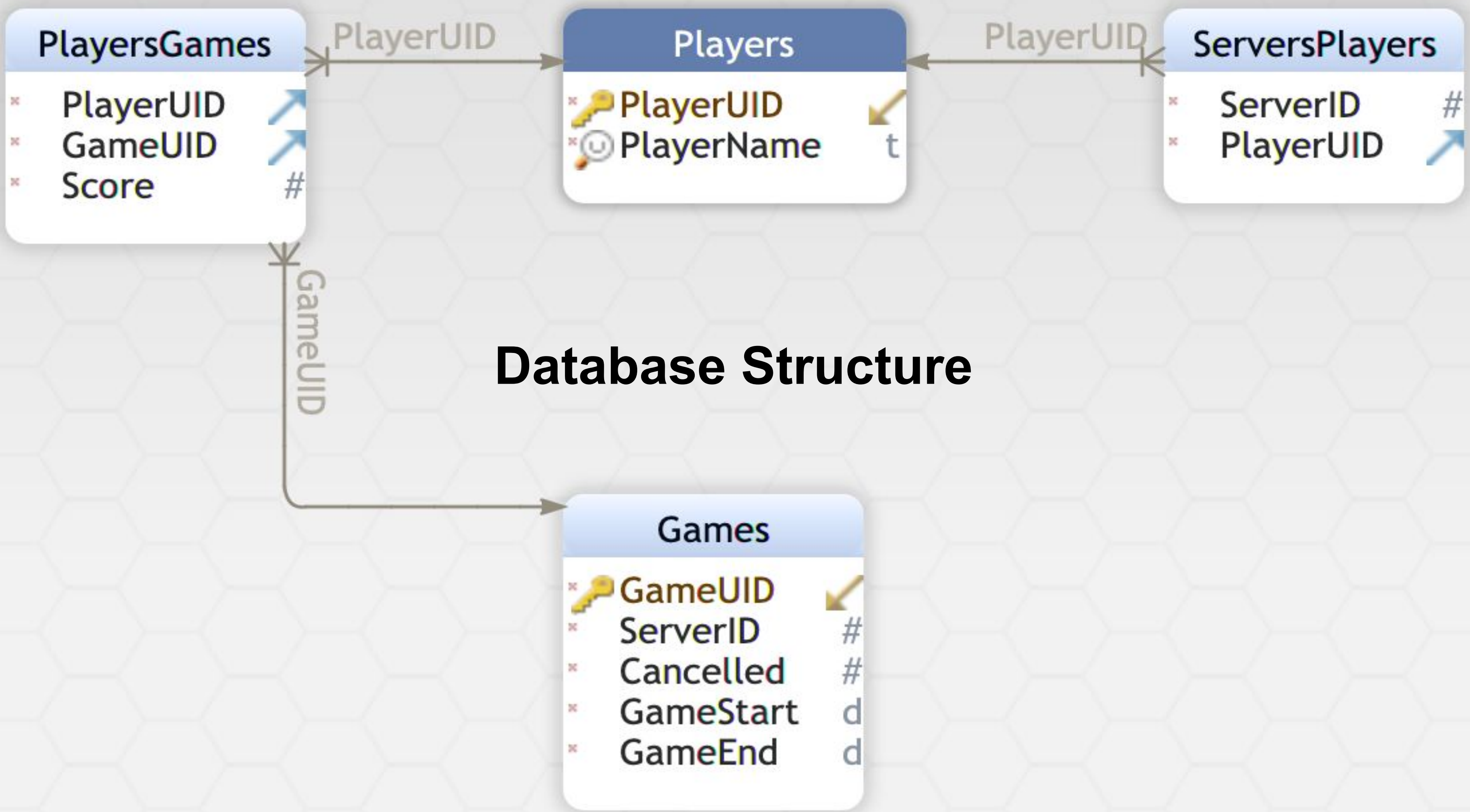
Required Functionality/Features

- "/newgame" command will start the game, generating a new random board
- Players will be able to select cards from the board, and when a set is found, a new board will be generated with the 3 selected cards replaced
- Play will continue until the deck is empty and there are no sets left on the board, or until the user who started the game presses the cancel button
- Scores will be displayed, and stats will be recorded about each game when they end or are cancelled

Architecture

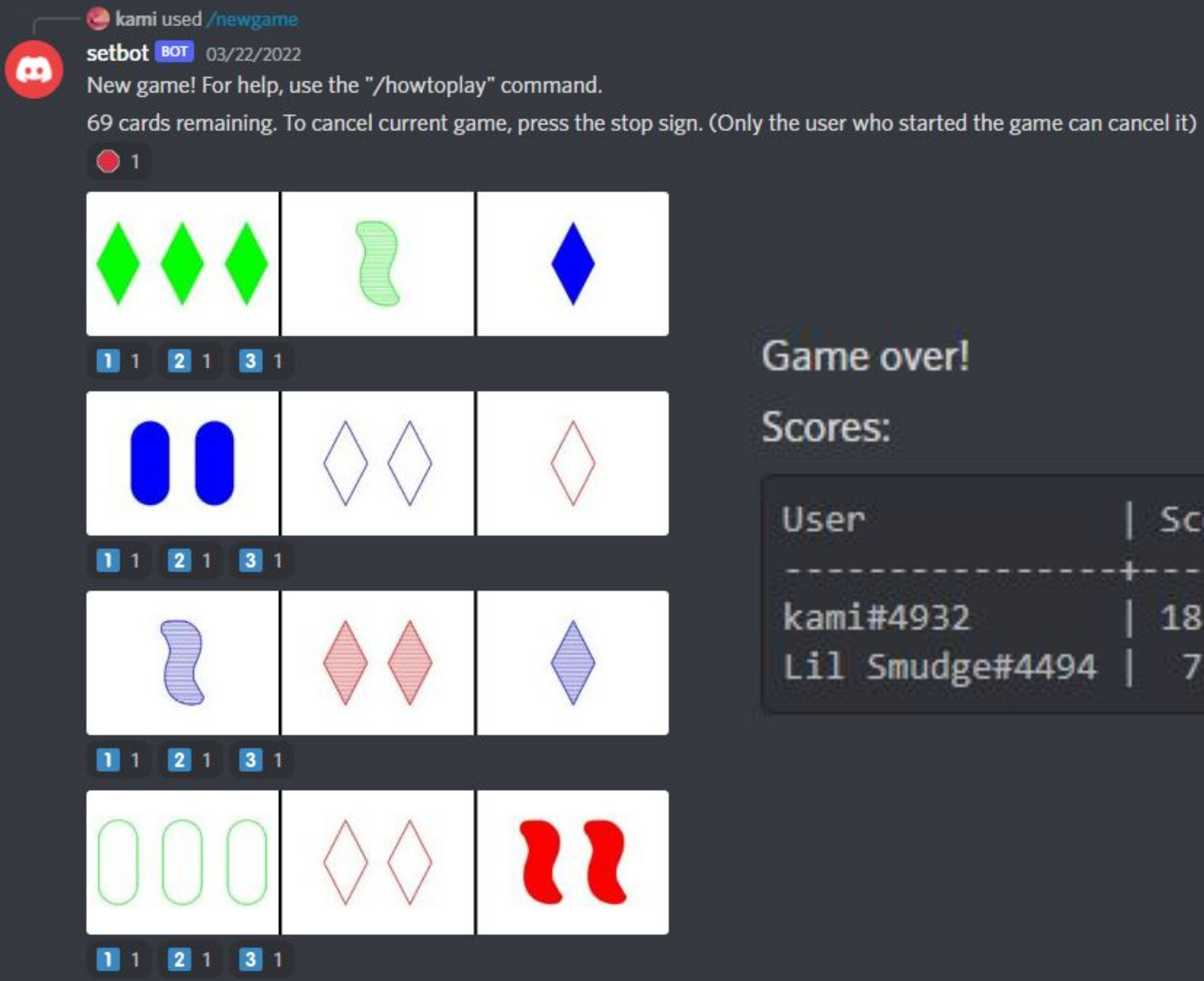


Database Structure



Accomplished Work

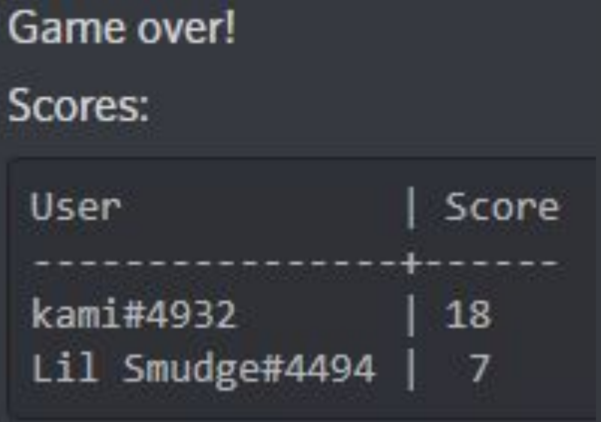
- Besides database integration and statistics, all functionalities described in this document have been successfully implemented
    - Game loop is achieved through recursion of the “continueGame()” method
    - Cards are encoded using four digits, with values 1 2 and 3 for card properties
    - Images are composited using images.js, a Node.js framework for manipulating image files
    - Six identical reaction collectors have been made, one for each possible row
    - Boards are checked using an optimum pair-check algorithm as described in the paper “On the Complexity of the Game of SET,” by K. Chaudhuri et al.
- The bot has yet to be hosted on pluto@hood.edu, but we have developed a Unix service file that will allow for uninterrupted operation of the program
- The main logic file, “newgame.js”, contains over 1000 lines of code
- The project is absolutely living up to our initial expectations, and in some cases even exceeding them


kami used /newgame

setbot BOT 03/22/2022

New game! For help, use the "/howtoplay" command.

69 cards remaining. To cancel current game, press the stop sign. (Only the user who started the game can cancel it)


Game over!

Scores:

```
User              | Score
------------------+------
kami#4932         | 18
Lil Smudge#4494   |  7
```

Effort

Team Member Name	Responsibilities/ Accomplishments	Hours Spent
Alex Algazi	Game loop structure General encoding Image generation Game data storage Unix service	25
Diana Teka	Board set-checker Player scores display Help command “/howtoplay” Database structure	18

Effort cont.

- The team communicated using Discord primarily, and using email when the other member could not be reached on Discord
- We used clockify to record the number of hours worked on each task
- We used Jira to track our overall schedule and keep track of individual tasks

Overall, this style of communication (using Discord) was very effective, and we were able to meet and work on the project remotely using Discord's VoIP calls

My only regret is that I wish we could have used the time in class more efficiently. Since the bot only has one unique ID, only one person could be testing features at a time, which slowed down development in some instances.

Lessons Learned

- When you have a development environment with only one user permitted to test at a time, responsibilities must be delegated such that one team member can always be working on something, such as the database backend, that does not require immediate testing.
- Much of our time in late February/early March was spent redesigning the overall structure of the program to account for a timing discrepancy in the generation of the board’s elements. In retrospect, being first-time discord developers, we could not have foreseen this complication. However, in the future, were I to develop another Discord application, I would try to better account for processing overhead from the beginning, rather than having to patch it up later.

What’s Next?

- While the database schema has been finalized, we still need to include logic for inserting records into our tables in the program. This will be accomplished by Tuesday April 26th
- After that, we need to finally migrate the bot over to pluto@hood.edu, in such a way that the program will be automatically executed at runtime on the server. This will be completed by the final deliverable date, Tuesday May 5th

In loving memory of Diana Teka