

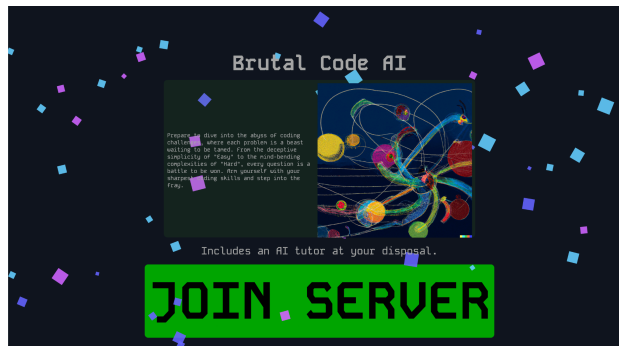
StoryCode Docs

By: John Yehia, Yogesh Sampathkumar, Sivamsh Indukuri, Kshiraj Gupta

Front Page site with included discord link:

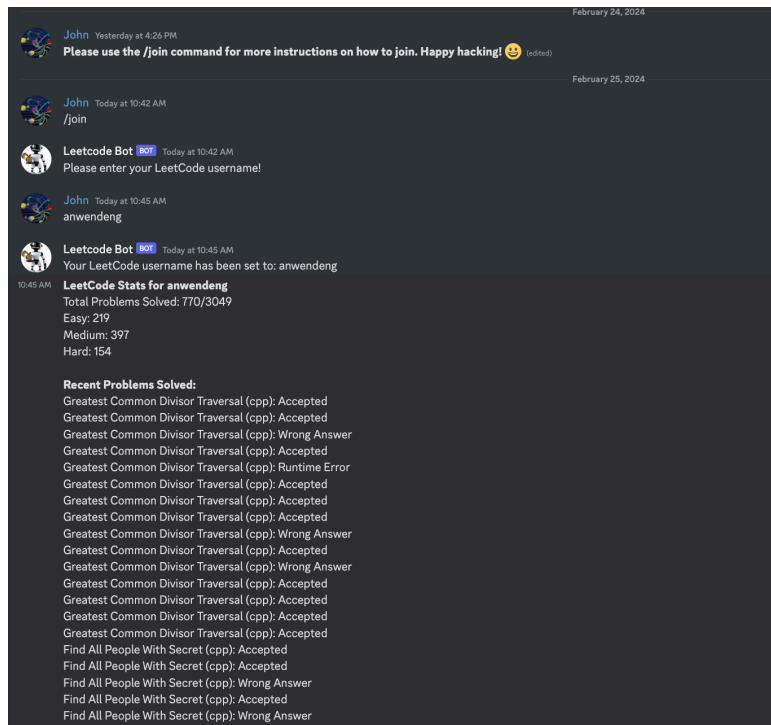
<https://65dafc1cc56064b3baba2c9b--monumental-rugelach-7cdb76.netlify.app/>

Site preview:

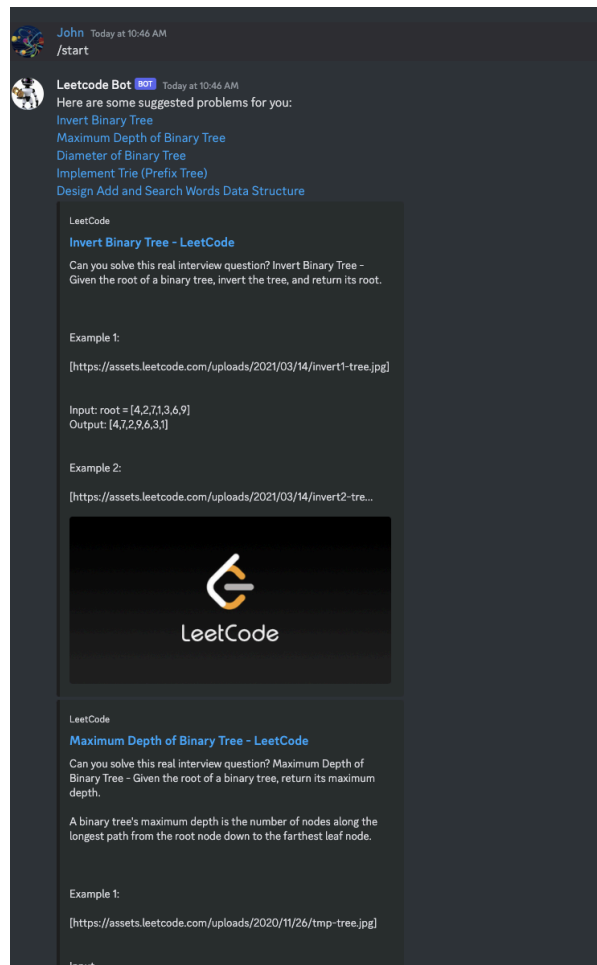


Finished Bot has these Main commands:

/join - Initiates the process to link a LeetCode account with the Discord user.



/start - Provides the user with problem suggestions based on their skill level.



/status - Checks and displays the user's progress on the suggested problems for the current day.

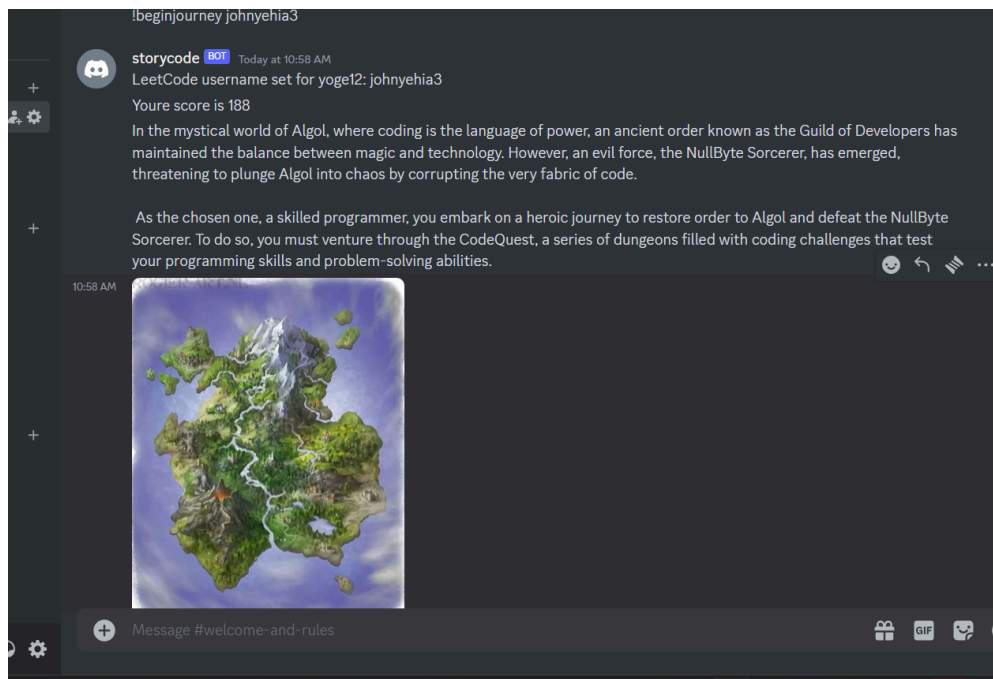


To use the AI Tutor:

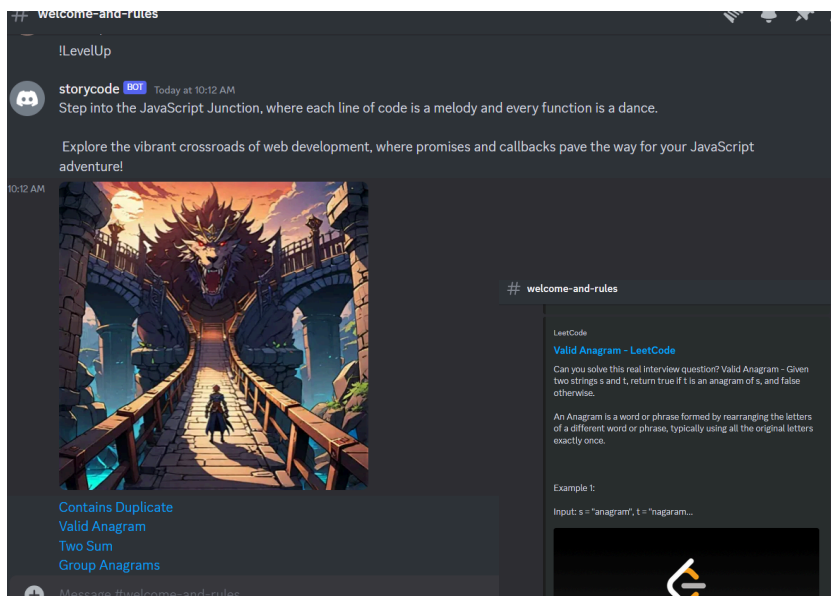
Simply navigate to the [# -ai-tutor-](#) chat in the StoryCode discord server [link above].

Walkthrough of Story Main Commands → Unfinished but We Plan to Implement the Bot in the Future

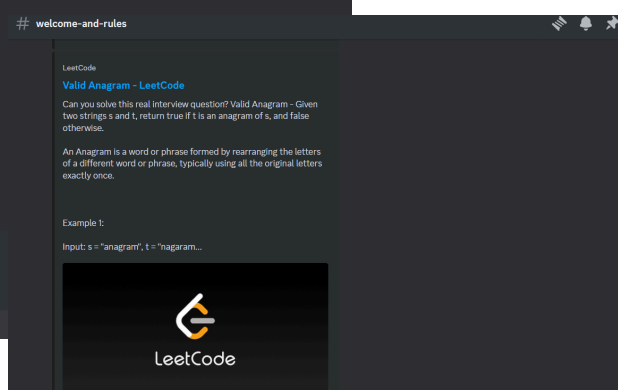
!beginJourney



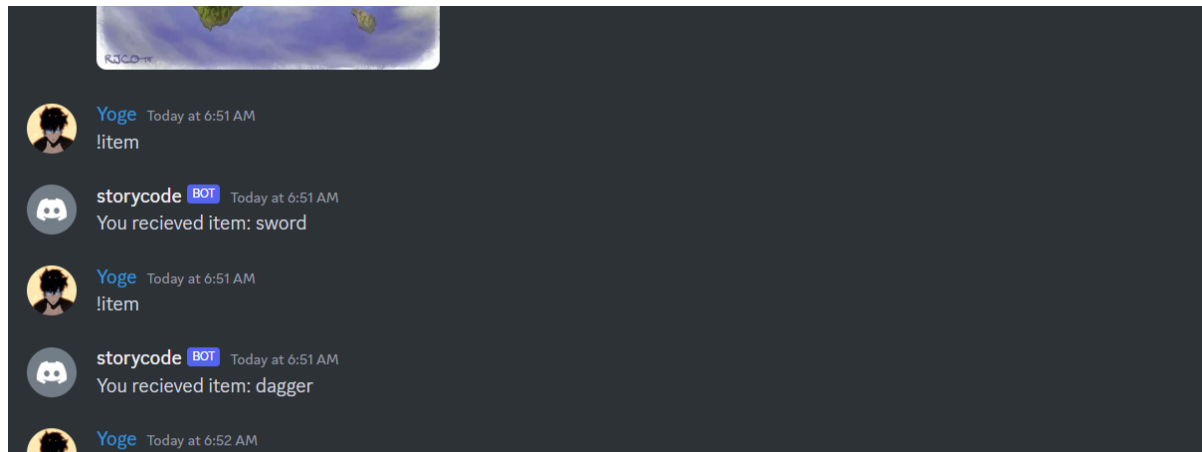
!LevelUp



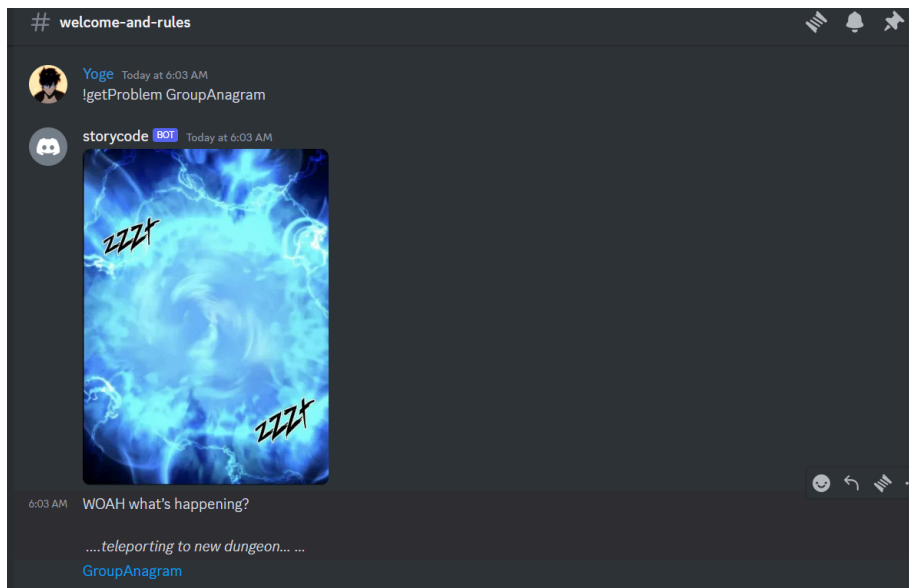
(Example of link given by Bot)



!item



!getProblem (with a link)



Code Implementation of Story

JSON File Website Scraper:

```
1 class Webscraper:
2
3     def createArray(self):
4         LeetUser = input("Input your LeetCode Username ")
5         my_url = "https://leetcode-api-faisalshohag.vercel.app/" + LeetUser
6
7         client = uReq(my_url)
8
9         page_html = client.read()
10
11         page_html
12
13         page_soup = soup(page_html, features= "html.parser")
14
15         page_soup.getText
16
17         text = page_soup.getText() + ""
18
19         completedNum= int(text[15:text.index(",")])
20
21
22         questionArray=list()
23         while text.index("\"title\":")+9 != text.rindex("\"title\":")+9:
```

Description: Parses through a JSON File with the recent data of a certain user (through their username) and creates an array that contains the title of the problem, the timestamp for how recently they completed a Leetcode problem, and if they had successfully completed the problem or not. This array is the main array implemented to other files and classes that is used for the XP and Rank system

Tier List and XP on User

```

de Blame 44 lines (33 loc) · 1.52 KB Code 55% faster with GitHub Copilot
15
16     user_score = 0
17
18     for problem in user_completed_problems:
19         for tier, tier_problems_list in tier_problems.items():
20             if problem in tier_problems_list:
21                 user_score += tier * 10
22                 break
23
24     tierEnd = False
25
26
27     def get_user_tier(user_problems):
28         global tierEnd
29
30         greatest_tier = 0 # Initialized with the lowest tier
31
32         for tier, tier_problems_list in tier_problems.items():
33             if set(tier_problems_list).issubset(set(user_problems)):
34                 greatest_tier = max(greatest_tier, tier)
35                 tierEnd = True
36             elif tierEnd:
37                 return greatest_tier
38
39         return greatest_tier
40
41     user_tier = get_user_tier(user_completed_problems)
42     print(f"The users score is {user_score}")
43     print(f"The user is in Tier {user_tier}")
44

```

Description: This user takes in the array and compares them to a python dictionary of problem names with different tier (ranks) based on these questions. It iterates through a double for loop and multiplies whatever the tier is by 10 and adds it to the user's score (XP0 if the completed problem name is in tier problem list already created. Additionally, it takes the greatest tier that the user is in and returns it as the Rank of the user

Point System based on Time

```

36     class pointSystem(WebScraper):
46         # return points
47         newArray = WebScraper.createArray()
48         points = 0
49     def sendReminder(self, newArray):
50         timeStamp = newArray[0].returnTime()
51         dt_object = datetime.datetime.fromtimestamp(timeStamp)
52         currentTime = datetime.now()
53         time_difference = currentTime - dt_object
54         days_difference = time_difference.days
55         reminder = "Its been 24 hours. Why not practice some LeetCode?"
56         if(days_difference >= 1):
57             return reminder
58
59         #After a day call this method
60         count = 0
61     def pointsOnTime(self, points, count, newArray):
62         timeStamp = newArray[0].returnTime()
63         dt_object = datetime.datetime.fromtimestamp(timeStamp)
64         currentTime = datetime.now()
65         time_difference = currentTime - dt_object
66         days_difference = time_difference.days
67         if(days_difference <= 1 ):
68             count = count + 1
69             if(count == 3):
70                 count = 0
71                 points = points + 1
72         return points
73
74
75     print(sendReminder(newArray))

```

Description: The two methods in this file called `sendReminder` and `pointsOnTime` utilizes the variable in the JSON file that is based on most recent time data that the user has done a leetcode problem. The `sendReminder` method will send a reminder to the user every 24 hours if and only if the user hasn't completed a problem in the last 24 hours and offers them a problem to complete. The `pointsOnTime` method will give an additional few points to the user for completing a problem every day for a 3 day streak. It uses the array from parsing the JSON file and the `datetime` library to convert the timestamp to a more useable format

Leetcode Bot Overview

The Progress Bot is designed to engage and motivate learners by tracking their progress on coding challenges, specifically tailored for platforms like LeetCode. It categorizes users based on their performance and suggests problems suited to their skill level.

Key Features

Problem Categorization: Problems are categorized into different topics and difficulties, making it easier for users to find suitable challenges.

User Categorization: Based on the number of problems solved, users are classified into beginner, intermediate, or advanced categories.

Problem Suggestions: The bot suggests problems to users based on their categorization, helping them progress at a pace suited to their skill level.

LeetCode Integration: By fetching data from the LeetCode API, the bot provides up-to-date statistics on users' progress and recent problem-solving activities.

→ Working Features of the Story Walkthrough:

!beginJourney → starts the journey of the story in the Guild of Coders after joining the server, creates the beginning map and outlines the story ahead for the user ✓

!marketplace → allows for an interactive marketplace where the user can use their XP for various actions (booster XP limited time, mentorship by asking higher level tiers and paying XP, and acclaim various badges)

!checkXP → checks the current XP (points) of the user from completing leetcode and from maintaining timely streaks in the server ✓

!checkRank → checks the current rank or tier of the user at that certain moment ✓

!getProblem (gets the link) → gets the link of the leet code problem in that tier of the user, the user will be able to click on the link and solve the problem ✓

!profile → returns the profile of the user (a message that has the format of an ID card, profile picture of the user, Rank and XP) ✓

!levelUp → if the user has leveled up to a higher dungeon, the bot will return the image and the welcome prologue of the dungeon. Additionally it will provide a handful of links to leetcode problems that the user can click on ✓

→ Future Features

!leaderboard → gives back the current leaderboard of the entire server based on the current XP of each user

General Features: Creating a different storyline in the future with more dungeons and various coding challenges. Additionally, implementing a way to verse other discord users in a timed competition for XP and a Rank boost.

No Language Barriers: Anyone in the world will be able to prepare for coding interviews regardless of region

A Website that Implements Code: Users will be able to track, learn, and solve leetcode problems regardless of where they live. Discord isn't supported everywhere, so we've decided to implement a web app in the future so that there aren't platform barriers associated with the region.

Usage

Joining the Bot: Users start by typing `/join` in a channel where the bot is active. They will then be prompted to enter their LeetCode username.

Fetching Stats: Once the username is set, the bot fetches and displays the user's LeetCode statistics, including total problems solved, and a breakdown by difficulty.

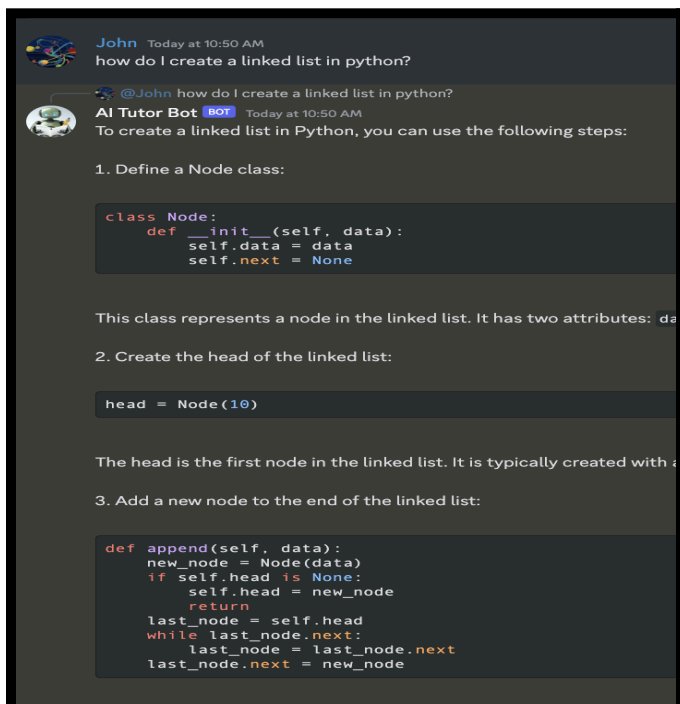
Getting Suggestions: Users can type `/start` to receive problem suggestions tailored to their skill level.

Commands

`/join` - Initiates the process to link a LeetCode account with the Discord user.

`/start` - Provides the user with problem suggestions based on their skill level.

`/status` - Checks and displays the user's progress on the suggested problems for the current day.



AI Tutor using the Google Gemini API

The AI Tutor is a conversational assistant powered by Google's Generative AI, designed to help users with programming-related queries and learning resources within the Discord environment.

Key Features

Conversational Interactions: Users can ask programming-related questions or request explanations on various topics, and the AI Tutor will generate responses in real-time.

Content Generation: Leveraging Google Generative AI, the tutor can create explanations, examples, and even code snippets to aid in learning.

Adaptive Responses: The AI Tutor can handle a wide range of queries, from simple definitions to complex problem-solving techniques.

Interacting with the Tutor

Users can simply type their queries in the designated channel, and the AI Tutor will respond. This can range from asking for explanations on specific concepts to seeking advice on debugging code.

Long Responses Handling: If the AI's response exceeds Discord's character limit, the bot is designed to split the content into manageable chunks, ensuring that users receive complete answers.

Commands

There are no specific commands to interact with the AI Tutor. Users engage with it by typing their questions or requests directly into the chat.

Conclusion

Both the Progress Bot and AI Tutor are designed to enhance the learning experience within the StoryCode Discord community. By providing personalized problem suggestions and real-time assistance, these tools aim to foster a supportive and interactive learning environment for coding enthusiasts at all levels.