Problems BOT
Installation and Setup Guide

December 3, 2025

1 Overview

Problems BOT is a Discord bot that posts a new problem every day at 12:00 PM (IST) and lets users submit answers via direct messages.It stores problems and submissions in a local SQLite database `quiz_bot.db` and exposes several slash commands for Curators and solvers inside Discord.The scoring system uses a base of 1000 points per problem, with time decay and an optional penalty for wrong answers.

2 Prerequisites

2.1 Discord Application

1. Open the Discord Developer Portal: `https://discord.com/developers/applications`.
2. Create a new application and add a *Bot* user.
3. Enable the "Message Content Intent" and "Server Members Intent" if needed.
4. Copy the bot token; it will be used as `DISCORD_TOKEN`.

2.2 Python Environment

1. Install Python 3.9 or newer.
2. Install dependencies in a virtual environment:

```
python -m venv .venv
source .venv/bin/activate   # Windows: .venv\Scripts\activate
pip install -r requirements.txt
```

3. Ensure `discord.py` (2.x), `python-dotenv` and `sqlite3` are available.

3 Configuration

3.1 Environment Variables

Create a file named `.env` in the project root:

```
DISCORD_TOKEN=your_bot_token_here
```

The bot code uses `python-dotenv` to load this file at startup.If `DISCORD_TOKEN` is not set, the bot exits with an error.

3.2 Timezone and Scheduling

In `bot.py` the timezone and daily posting time are configured as:

```
IST = ZoneInfo("Asia/Kolkata")
DAILY_POST_TIME = time(hour=12, minute=0, tzinfo=IST)
```

By default, the bot posts one problem per day at 12:00 PM India Standard Time in the configured guild and channel.

3.3 Guild and Channel IDs

Set the IDs of the guild and channel where the automatic daily problem should be posted:

```
AUTO_GUILD_ID   = 123456789012345678
AUTO_CHANNEL_ID = 234567890123456789
```

Use Discord's developer mode to copy these IDs from your server and channel.

3.4 Curator Role

Define a role in your Discord server named `Curator`.Members with this role can:

- Manually post today's problem using `/post_today`.
- Create new problems using `/create_problem`.
- Reset scores for a problem using `/unscore_problem`.

The role name must match the `CURATOR_ROLE` constant in `bot.py`.

4 Database Initialization

On first run, the bot initializes the SQLite database:

- Table `problems`: stores problem metadata, statement, answer and timing.
- Table `submissions`: stores each DM submission, correctness flag and points.
- Table `problem_ratings`: stores 1–5 star ratings for solved problems.

The function `init_db()` is called in the `if __name__ == "__main__":` block before the bot starts.

5 Running the Bot

5.1 Local Run

From the project root:

```
python bot.py
```

If the token and intents are configured correctly, the bot logs in and synchronizes its slash commands to the configured guild.

5.2 Systemd / Long-Running Process

For a server or VPS, run the bot under `tmux`, `screen`, a process manager, or a systemd unit.Ensure the working directory contains `bot.py`, `.env` and `quiz_bot.db`.

6 Commands Overview

6.1 Solver Commands

- `/leaderboard overall`: show global solver leaderboard by total points and solves.
- `/leaderboard today`: show today's leaderboard for the currently active problem.
- `/rate_problem code:<code> rating:1-5`: rate a problem after solving it correctly.

Users submit answers by sending a direct message to the bot in the format:

```
<problem_code> <answer>
```

6.2 Curator Commands

- `/create_problem`: create a new problem (limited to 1 per 24 hours per user).
- `/post_today code:<code>`: manually post a specific problem in the current channel.
- `/unscore_problem code:<code> user:[optional]`: clear scores for a problem globally or for a single user.
- `/list_problems`: list all problems with difficulty and active status.
- `/curator_leaderboard`: show a leaderboard of problem authors by count and rating.

7 Scoring Model

The scoring configuration in `bot.py` is:

```
BASE_POINTS = 1000
DECAY_INTERVAL_SECONDS = 120   # 1 point lost every 2 minutes
WRONG_PENALTY = 50             # points deducted per wrong answer
```

For a correct submission at time t seconds after the problem opens, the raw score is

$$\max\bigl(0,\ \text{BASE\_POINTS} - \bigl\lfloor \frac{t}{\text{DECAY\_INTERVAL\_SECONDS}} \bigr\rfloor\bigr).$$

Each wrong attempt inserts a submission row with $-\text{WRONG\_PENALTY}$ points, and leaderboards sum points across all submissions for a user.