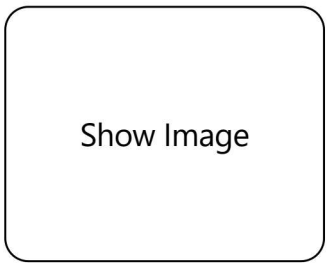
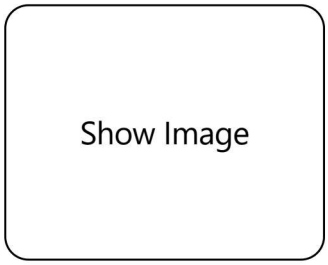
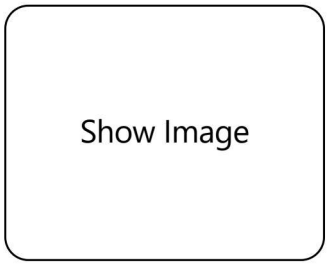
🥒 PickleJar Bot

A Discord bot that rewards users for pickle-related references and provides moderation and community engagement tools.







✨ Features

🥒 Pickle Tracking

- Automatically rewards users when they mention pickle-related words
- Tracks pickle counts per user and features a leaderboard system
- Daily rewards system with pickle coins

🛡️ Moderation Tools

- Ban, kick, mute, and warning systems
- Auto-punishment based on warning thresholds
- Comprehensive logging for moderation actions

👏 Community Recognition

- Allow users to recognize each other's contributions

- Recognition leaderboard to highlight active community members

🔧 Admin Tools

- Server statistics
- Bot ping and status commands
- Message purging
- Announcements

🤖 Custom Commands

- Create server-specific custom commands
- Simple interface for managing commands

🚀 Setup and Installation

Prerequisites

- Python 3.8 or higher
- PostgreSQL database (optional, but recommended for all features)

Step 1: Clone the repository

bash Copy

```
git clone https://github.com/yourusername/pickle_jar_bot.git
cd pickle_jar_bot
```

Step 2: Install dependencies

bash Copy

```
pip install -r requirements.txt
```

Step 3: Set up environment variables

Create a `.env` file in the root directory with the following:

Copy

```
DISCORD_BOT_TOKEN=your_discord_bot_token
DATABASE_URL=postgresql://username:password@hostname:port/dbname
```

Step 4: Create a Discord bot

1. Go to the Discord Developer Portal
2. Create a new application
3. Add a bot to your application
4. Copy the bot token to your `.env` file
5. Enable necessary intents (Presence, Server Members, Message Content)
6. Generate an invite link with appropriate permissions
7. Invite the bot to your server

Step 5: Start the bot

bash Copy

```
python main.py
```

🗄️ Database Setup

The bot uses PostgreSQL for data storage. The schema is available in `postgresql_schema_optimized.sql`.

Railway Deployment

If deploying on Railway:

1. Add the PostgreSQL plugin to your project
2. Set the `DATABASE_URL` environment variable to the connection string provided by Railway
3. Deploy your bot with the `railway up` command or through the web interface

Local PostgreSQL

1. Install PostgreSQL
2. Create a database for the bot
3. Run the schema file:

   bash Copy

   ```
   psql -U username -d database_name -f postgresql_schema_optimized.sql
   ```

4. Update your `.env` file with the connection string

📚 Available Commands

Pickle Commands

- `!pickles [user]` - Check pickle count for yourself or another user
- `!leaderboard` - Show the pickle leaderboard
- `!daily` - Claim your daily pickle coins

Moderation Commands

- `!ban <user> [reason]` - Ban a user
- `!kick <user> [reason]` - Kick a user
- `!warn <user> [reason]` - Warn a user
- `!warnings [user]` - Check warnings for yourself or another user
- `!clearwarnings <user>` - Clear all warnings for a user
- `!mute <user> [duration] [reason]` - Mute a user
- `!unmute <user> [reason]` - Unmute a user

Community Recognition

- `!recognize <user> <message>` - Recognize someone's contribution
- `!recognitions [user]` - Check recognition count
- `!recognition_leaderboard` - Show the most recognized users

Admin Tools

- `!ping` - Check bot latency
- `!stats` - Show bot statistics
- `!clear [amount]` - Clear messages
- `!reload <cog>` - Reload a specific cog
- `!announce <channel> <message>` - Send an announcement
- `!setup` - Initial server setup for the bot

Custom Commands

- `!addcmd <name> <response>` - Add a custom command
- `!delcmd <name>` - Delete a custom command

- `!listcmds` - List all custom commands

📦 Project Structure

Copy

```
pickle_jar_bot/
├── cogs/                  # Bot commands and features
│   ├── admin_tools.py     # Administrative tools
│   ├── community_recognition.py  # Recognition system
│   ├── custom_commands.py # Custom command system
│   ├── error_handler.py   # Global error handling
│   ├── moderation.py      # Moderation commands
│   └── pickle_tracking.py # Pickle tracking system
├── utils/                 # Utility modules
│   ├── config.py          # Configuration manager
│   ├── db_manager.py      # Database connection
│   └── logger.py          # Logging system
├── .env                   # Environment variables
├── config.json            # Bot configuration
├── main.py                # Main bot file
├── postgresql_schema_optimized.sql  # Database schema
├── railway.json           # Railway deployment config
├── README.md              # Documentation
└── requirements.txt       # Required packages
```

🔧 Configuration

The bot uses a `config.json` file for configuration:

json

Copy

```
{
  "pickle_rewards": {
    "words": ["pickle", "dill", "gherkin", "gherkins", "pickled"],
    "cooldown_seconds": 300,
    "reward_messages": [
      "🥒 {user} just got a pickle!",
      "Congrats {user}! You earned a pickle!",
      "One fresh pickle for {user}! 🥒",
      "Pickle acquired! {user} adds one to their collection!"
    ]
  },
  "moderation": {
    "default_warning_reason": "Breaking server rules",
    "auto_punish": false,
    "warning_thresholds": {
      "3": "mute",
      "5": "kick",
      "7": "ban"
    }
  }
}
```

🚂 Deploying on Railway

This bot is configured for easy deployment on Railway:

1. Create a new Railway project
2. Link your GitHub repository
3. Add the PostgreSQL plugin
4. Set environment variables in the Railway project settings
5. Deploy your application

The `railway.json` file is already configured for automatic deployment.

📝 Logs

Logs are stored in the `logs/` directory. The bot logs:

- Command usage
- Errors and exceptions

- Database operations
- Bot status changes
- Moderation actions

📋 To-Do List

- [ ] Add a shop system for spending pickle coins
- [ ] Implement role rewards based on pickle count
- [ ] Add scheduled announcements
- [ ] Create interactive pickle games
- [ ] Implement a web dashboard

🤝 Contributing

Contributions are welcome! Please feel free to submit a Pull Request.

1. Fork the repository
2. Create your feature branch: `git checkout -b feature/amazing-feature`
3. Commit your changes: `git commit -m 'Add some amazing feature'`
4. Push to the branch: `git push origin feature/amazing-feature`
5. Open a Pull Request

📜 License

This project is licensed under the MIT License - see the LICENSE file for details.

🙏 Acknowledgements

- discord.py for the Discord API wrapper
- Railway for hosting services
- PostgreSQL for database services