**📚 PTCGP Discord Bot - User Documentation**

**Overview**

The PTCGP (Pokémon Trading Card Game Pocket) Discord Bot is a comprehensive bot for managing reroll activities, tracking god packs, and analyzing user performance in PTCGP.

**File Structure and Purpose**

**Core Files**

1. **main.py** - The main bot entry point
   * Initializes the Discord bot
   * Loads all components
   * Manages background tasks
   * Handles commands and events
2. **config.py** - Configuration management
   * Loads environment variables
   * Validates bot permissions
   * Manages feature flags
   * Handles all bot settings
3. **.env** - Environment configuration (create from .env.example)
   * Store sensitive data (tokens, IDs)
   * Configure features
   * Set rate limits

**Database Management**

1. **database\_manager.py** - SQLite database operations
   * User management
   * God pack tracking
   * Heartbeat storage
   * Test result management
2. **db\_setup.py** - Database initialization
   * Creates required tables
   * Sets up indexes
   * Validates database structure

**Core Functionality**

1. **core\_utils.py** - Core utility functions
   * User statistics
   * Server reports
   * Data management
   * Legacy XML support
2. **utils.py** - General utilities
   * String formatting
   * Discord helpers
   * Math functions
   * File operations
3. **xml\_manager.py** - XML data management (legacy support)
   * User data storage
   * Server data management
   * Backward compatibility

**Enhanced Features**

1. **enhanced\_bot\_commands.py** - Slash commands and bot commands
   * Probability calculations
   * User status management
   * System administration
   * Enhanced rate limiting
2. **probability\_calculator.py** - God pack probability calculations
   * Bayesian probability updates
   * Test result processing
   * Confidence calculations
3. **heartbeat\_analytics.py** - Activity analysis
   * Run detection
   * Performance metrics
   * Anomaly detection
   * Leaderboards
4. **plotting\_system.py** - Data visualization
   * Activity charts
   * Server statistics
   * Probability trends
   * Text fallback for no matplotlib
5. **expiration\_manager.py** - God pack expiration tracking
   * Automatic archiving
   * Expiration warnings
   * State management
6. **google\_sheets\_integration.py** - Google Sheets sync
   * User data export
   * God pack tracking
   * Statistics logging

**Support Files**

1. **upload\_utils.py** - Data upload utilities
   * GitHub Gist integration
   * Pastebin support
   * File uploads
2. **miss\_sentences.py** - Response messages
   * Contextual messages
   * Encouragement texts
   * Celebration messages
3. **enhanced\_gp\_test\_utils.py** - God pack testing utilities
   * Test data management
   * Probability calculations

**🚀 Setup Guide**

**Prerequisites**

* Python 3.8+
* Discord Bot Token
* Discord Server (Guild) ID
* Required Discord channel IDs

**Installation Steps**

1. **Clone/Download the bot files**

bash

*# Create bot directory*

mkdir ptcgp-bot

cd ptcgp-bot

*# Copy all files to this directory*

1. **Install dependencies**

bash

pip install -r requirements.txt

1. **Configure environment**

bash

*# Copy example environment file*

cp .env.example .env

*# Edit .env with your values*

nano .env *# or use any text editor*

1. **Required .env settings**

env

# Discord Configuration (REQUIRED)

DISCORD\_TOKEN=your\_bot\_token\_here

DISCORD\_GUILD\_ID=your\_guild\_id\_here

# Channel IDs (REQUIRED)

CHANNEL\_ID\_COMMANDS=channel\_id\_for\_commands

CHANNEL\_ID\_USER\_STATS=channel\_id\_for\_stats

CHANNEL\_ID\_HEARTBEAT=channel\_id\_for\_heartbeat

CHANNEL\_ID\_WEBHOOK=channel\_id\_for\_webhooks

# Optional but recommended

ENABLE\_AUTO\_BACKUP=true

MAX\_GLOBAL\_REQUESTS\_PER\_MINUTE=500

1. **Set up database**

bash

python db\_setup.py

1. **Configure bot permissions**
   * Required permissions:
     + Send Messages
     + Embed Links
     + Attach Files
     + Read Message History
     + Add Reactions
     + Manage Threads
     + Create Public Threads
     + Use Slash Commands
2. **Start the bot**

bash

python main.py

**📋 Command Reference**

**Basic Commands (Prefix: !)**

* !help - Show all available commands
* !test - Test bot functionality
* !ping - Check bot latency
* !status - Show bot status

**User Commands**

* !stats - Show server statistics
* !mystats - Show your personal statistics
* !setpack <pack\_name> - Set preferred pack
* !mypack - Show pack preferences
* !active - Set status to active
* !inactive - Set status to inactive
* !farm - Set farming status
* !leech - Set leeching status

**Slash Commands (if enhanced features available)**

* /probability <gp\_id> - Calculate god pack probability
* /miss <gp\_id> - Add miss test
* /noshow <gp\_id> <slots> <friends> - Add no-show test
* /plot\_user [user] [days] - Generate activity plot
* /plot\_status - Check plotting system status

**Admin Commands**

* !forcestats - Force statistics update
* !cleanup CONFIRM - Clean inactive users
* !backup - Create data backup
* !health - System health check
* /system\_status - Enhanced system status
* /create\_backup - Create manual backup
* /rate\_limit\_stats - View rate limiting stats

**🔧 Feature Configuration**

**Enable/Disable Features**

Edit .env file:

env

# Feature Flags

FEAT\_ADV\_PROBABILITY=true

FEAT\_PLOTTING=true

FEAT\_EXPIRATION=true

FEAT\_ANOMALY=true

# Google Sheets (requires credentials.json)

ENABLE\_SHEETS=false

GOOGLE\_CREDENTIALS\_FILE=credentials.json

**Rate Limiting Configuration**

env

# Adjust these based on server size

MAX\_GLOBAL\_REQUESTS\_PER\_MINUTE=500

MAX\_USER\_COMMANDS\_PER\_5MIN=100

MAX\_HEAVY\_COMMANDS\_PER\_HOUR=20

COMMAND\_COOLDOWN=2

**Plotting Configuration**

env

# If matplotlib not available, enable fallback

ENABLE\_PLOTTING\_FALLBACK=true

TEXT\_CHART\_WIDTH=40

TEXT\_CHART\_HEIGHT=10

**🛠️ Troubleshooting**

**Bot won't start**

1. Check Discord token in .env
2. Verify guild ID is correct
3. Check Python version (python --version)
4. Review error messages in console

**Commands not working**

1. Check bot permissions in Discord
2. Verify channel IDs in .env
3. Check command prefix setting
4. Look for errors in logs/bot.log

**Features unavailable**

1. Check !status command output
2. Install missing dependencies
3. Review feature flags in .env
4. Check component availability

**Performance issues**

1. Adjust rate limits in .env
2. Enable auto-cleanup features
3. Check database size
4. Monitor memory usage

**📊 Maintenance**

**Regular Tasks**

1. **Backups**: Automatic daily or use !backup
2. **Cleanup**: Automatic or use !cleanup CONFIRM
3. **Updates**: Check for updates regularly
4. **Logs**: Review logs in logs/ directory

**Database Maintenance**

bash

*# Check database integrity*

sqlite3 data/ptcgp\_unified.db "PRAGMA integrity\_check;"

*# Vacuum database (reduces size)*

sqlite3 data/ptcgp\_unified.db "VACUUM;"

**Monitoring**

* Use !health command regularly
* Check /system\_status for detailed info
* Monitor rate limit stats with /rate\_limit\_stats
* Review logs for errors

**🆘 Support**

**Common Issues**

1. **"Module not found"** - Install requirements: pip install -r requirements.txt
2. **"Permission denied"** - Check bot Discord permissions
3. **"Channel not found"** - Verify channel IDs in .env
4. **"Rate limited"** - Adjust rate limits or wait

**Getting Help**

1. Check error messages in console
2. Review logs/bot.log for details
3. Use !status to check system health
4. Verify configuration in .env

**🎯 Best Practices**

1. **Start with basic features** - Don't enable everything at once
2. **Test in a test server first** - Before production deployment
3. **Regular backups** - Enable auto-backup feature
4. **Monitor performance** - Use health check commands
5. **Update regularly** - Keep dependencies updated
6. **Secure your tokens** - Never share .env file
7. **Set appropriate rate limits** - Based on server size
8. **Use slash commands** - For enhanced features
9. **Configure channels properly** - Separate channels for different functions
10. **Review logs regularly** - Catch issues early

**📈 Performance Tips**

1. **Database optimization**:
   * Run VACUUM monthly
   * Keep retention period reasonable
   * Use cleanup features
2. **Rate limiting**:
   * Start with default values
   * Adjust based on usage
   * Monitor rate limit stats
3. **Memory management**:
   * Enable memory monitoring
   * Use plotting fallback if needed
   * Regular bot restarts (weekly)
4. **Feature optimization**:
   * Disable unused features
   * Adjust sync intervals
   * Use caching where available