**📚 MAIgnus\_CAIrlsen - Comprehensive System Documentation**

**🔹 Overview**

**MAIgnus\_CAIrlsen** is an AI-powered chess performance management system. It:

1. Fetches your latest completed games from Chess.com.
2. Analyzes game performance with OpenAI’s GPT models.
3. Identifies your biggest mistake/blunder.
4. Generates a board diagram of the mistake.
5. Sends a detailed email report with analysis and visuals.
6. Runs automatically on a schedule via Windows Task Scheduler.

**🔹 Core Features**

| **Feature** | **Description** |
| --- | --- |
| **Chess.com API Integration** | Automatically fetches latest completed games in PGN format. |
| **Automated PGN Analysis** | Uses OpenAI GPT (via API) to analyze your game. |
| **Blunder Identification** | Highlights key mistakes (by move number) in your games. |
| **Board Diagram Generation** | Generates a PNG showing the board position before a blunder. |
| **Email Reporting** | Sends a polished, automated report (text + image) by email. |
| **Task Scheduling** | Runs automatically up to 4x per day without user input. |

**🔹 Folder Structure**

bash

CopyEdit

MAIgnus\_CAIrlsen/

│

├── data/ # Stores downloaded PGN game files

│ └── seanr87\_xxxxx.pgn

│

├── images/ # Stores generated board images of blunders

│ └── blunder\_position.png

│

├── logs/ # Logs for tracking workflow actions and errors

│ ├── checker.log

│ ├── analyzer.log

│ └── email\_sender.log

│

├── reports/ # Game analysis text files

│ └── game\_analysis.txt

│

├── src/ # All source code scripts

│ ├── chess\_api.py

│ ├── game\_checker.py

│ ├── game\_analyzer.py

│ ├── blunder\_analyzer.py

│ ├── email\_sender.py

│ └── maignus\_bot.py

│

├── .env # Environment variables (excluded from repo)

├── .gitignore # Prevents secrets from being committed

├── requirements.txt # Python dependencies

└── README.md # High-level project overview

**🔹 Environment Setup**

1. **Python Version**
   * Python 3.12
2. **Install Dependencies**
   * Create and activate a virtual environment (optional but recommended):

bash

CopyEdit

python -m venv .venv

.venv\Scripts\activate

* + Install requirements:

bash

CopyEdit

pip install -r requirements.txt

**🔹 Dependencies**

**1. Python Packages (requirements.txt)**

| **Package** | **Version (recommended)** | **Purpose** |
| --- | --- | --- |
| requests | >=2.31 | HTTP requests to Chess.com API |
| openai | >=1.0 | GPT-3.5/4 API client |
| yagmail | >=0.15 | Simplified email sending via Gmail |
| python-chess | >=1.999 | Handling PGN files and chess boards |
| cairosvg | >=2.7 | Convert SVG chessboards to PNG images |
| markdown | >=3.4 | Convert markdown game analysis to HTML |
| python-dotenv | >=1.0 | Load sensitive configs from .env file |

**2. GTK and Cairo Runtime (for cairosvg to work)**

* **GTK3 Runtime for Windows**
  + Installed at: C:\Program Files\GTK3-Runtime Win64
  + bin folder added to the **PATH** system environment variable.

**🔹 Environment Variables (.env)**

Create a .env file in the root directory (never commit it to GitHub).

dotenv

CopyEdit

# OpenAI

OPENAI\_API\_KEY=sk-...

# Gmail SMTP

SENDER\_EMAIL=your\_email@gmail.com

EMAIL\_APP\_PASSWORD=your\_app\_password

RECEIVER\_EMAIL=receiver\_email@gmail.com

**🔹 Scripts Overview**

**1. chess\_api.py**

* Pulls player profiles and archives from Chess.com’s public API.
* Basic test script for profile and game fetching.

**2. game\_checker.py**

* Checks for new games.
* Downloads and stores PGN files in /data/.
* Maintains a log of already downloaded games (based on filenames).

**3. game\_analyzer.py**

* Reads the latest PGN file.
* Sends the PGN to OpenAI’s GPT for analysis.
* Generates a game\_analysis.txt report.

**4. blunder\_analyzer.py**

* Parses PGN using python-chess.
* Identifies the move where you blundered (simple logic now, expandable later).
* Uses python-chess to render the board at the move before the blunder.
* Converts SVG board to PNG using cairosvg.
* Outputs blunder\_position.png in /images/.

**5. email\_sender.py**

* Reads game\_analysis.txt.
* Converts analysis from Markdown to HTML (markdown library).
* Composes an HTML email:
  + Embedded blunder diagram.
  + Key Insight.
  + Full analysis.
* Sends via Gmail with yagmail.

**6. maignus\_bot.py (the orchestrator)**

* Runs the full workflow:
  1. Check for new games.
  2. Analyze the latest game.
  3. Generate blunder insights and diagram.
  4. Send the summary email.

**🔹 Task Scheduler**

The maignus\_bot.py script is scheduled using Windows Task Scheduler.

**Task Configuration**

* **Program/script**: C:\Users\soreill5\AppData\Local\Programs\Python\Python312\python.exe
* **Arguments**: maignus\_bot.py
* **Start in**: C:\Users\soreill5\MAIgnus\_CAIrlsen\src
* **Triggers**: Every 6 hours (or customize).
* **Conditions**: Run whether user is logged in or not.

**🔹 Critical Architectural Decisions**

1. **Modularity**: Each step (check, analyze, blunder, email) is separate for scalability and debugging ease.
2. **Stateless**: No database. Game files and filenames track progress. Simple and easy to rebuild.
3. **Logging**: Every script logs events and errors for troubleshooting.
4. **Secrets**: .env file for sensitive data, and .gitignore to exclude it from GitHub.
5. **HTML Emails**: Inline image embedding and Markdown-to-HTML conversion for clean reports.

**🔹 How to Rebuild From Scratch**

1. **Clone the Repo**

bash

CopyEdit

git clone https://github.com/seanr87/MAIgnus\_CAIrlsen.git

1. **Set Up Python**
   * Install Python 3.12.
   * Create virtual environment.
   * Install requirements.txt.
2. **GTK3 Runtime for Windows**
   * Download GTK3 Runtime.
   * Install to C:\Program Files\GTK3-Runtime Win64.
   * Add C:\Program Files\GTK3-Runtime Win64\bin to your PATH.
3. **Create a .env file**
   * Add API keys and email credentials.
4. **Verify Folder Structure**

bash

CopyEdit

/data

/images

/logs

/reports

1. **Test Manually**

bash

CopyEdit

python maignus\_bot.py

1. **Automate via Task Scheduler**

**🔹 Future Expansion Ideas**

1. **Categorize Blunders**
   * Opening, Midgame, Endgame errors.
   * Tactical vs. Positional mistakes.
2. **Database Integration**
   * SQLite/Postgres to track long-term performance trends.
3. **Front-end Dashboard**
   * Visualize performance trends over time.
4. **Stockfish Integration**
   * Deeper engine analysis for move quality.