

read me

FocusFlow Pro

An Engineering Approach to Productivity and Visual Health

Version: 1.0.0

Status: Stable / Production Ready

License: MIT / Educational Use

Executive Summary

FocusFlow Pro is a specialized desktop web application designed to address two critical issues in the modern academic and professional landscape: **Cognitive Load Management** and **Digital Eye Strain**.

Built using **Python** and the **Streamlit** framework, this application engineers a digital solution to the famous Pomodoro Technique. Unlike generic timer applications, **FocusFlow Pro** is architected with a focus on data persistence, algorithmic state management, and ergonomic user interface design.

It allows users to break work into 25-minute focused intervals while the system automatically handles the complex logic of assigning short (5-minute) and long (15-minute) restorative breaks.

Furthermore, the application features an integrated analytics engine that visualizes historical performance data, transforming abstract time management into actionable productivity insights.

Project Objectives

The primary goals of this software engineering project are:

1. **Functional Utility:** To provide a robust tool that automates the cognitive rhythm of work/rest cycles without user intervention.
2. **Ergonomic Design:** To implement a "Dark-First" UI philosophy (Background Hex: #1e1e1e) that minimizes blue light exposure and reduces visual fatigue during extended use.
3. **Data Persistence:** To demonstrate the implementation of a local relational database (SQLite) for storing historical session data.
4. **Engineering Rigor:** To showcase a modular **Model-View-Controller (MVC)** architecture, separating business logic from presentation and data access layers.

Detailed Feature Specification

1. The Algorithmic Timer Engine

The core of FocusFlow Pro is a state-machine-driven timer located in the core/ module.

- **Automatic Phase Transitions:** The system intelligently switches between three distinct states: Deep Work, Short Break, and Long Break.
- **The "Fourth Cycle" Rule:** The engine tracks a session counter. Upon the completion of every fourth work session, the system overrides the standard break logic to enforce a 15-minute long break, promoting long-term cognitive endurance.
- **State Preservation:** Utilizing `st.session_state`, the application maintains timer progress even if the browser refreshes or the UI is redrawn, simulating a stateful desktop application environment within a stateless web framework.

2. Persistent Data Architecture

Data integrity is managed via a dedicated Data Access Object (DAO) pattern in the database/ module.

Zero-Config Storage: The app utilizes SQLite, a serverless, ACID-compliant database engine that requires no installation.

Session Logging: Every completed work cycle is time-stamped and committed to the sessions table.

CRUD Capabilities: Users have the ability to Create (finish timer), Read (view charts), and Delete (undo accidental entries) records, providing full control over their data.

3. Integrated Productivity Analytics

FocusFlow Pro moves beyond simple timekeeping by including a Data Analysis module.

- **Data Aggregation:** The system uses **Pandas** to query raw SQL data and aggregate it by date.
- **Visualization:** A **Matplotlib** rendering engine generates bar charts dynamically. These charts are programmatically styled to match the application's dark theme (dark grey backgrounds, light text), ensuring visual consistency.



System Architecture

The project codebase is refactored into a Modular Layered Architecture to ensure scalability and maintainability.

The MVC Pattern Implementation:

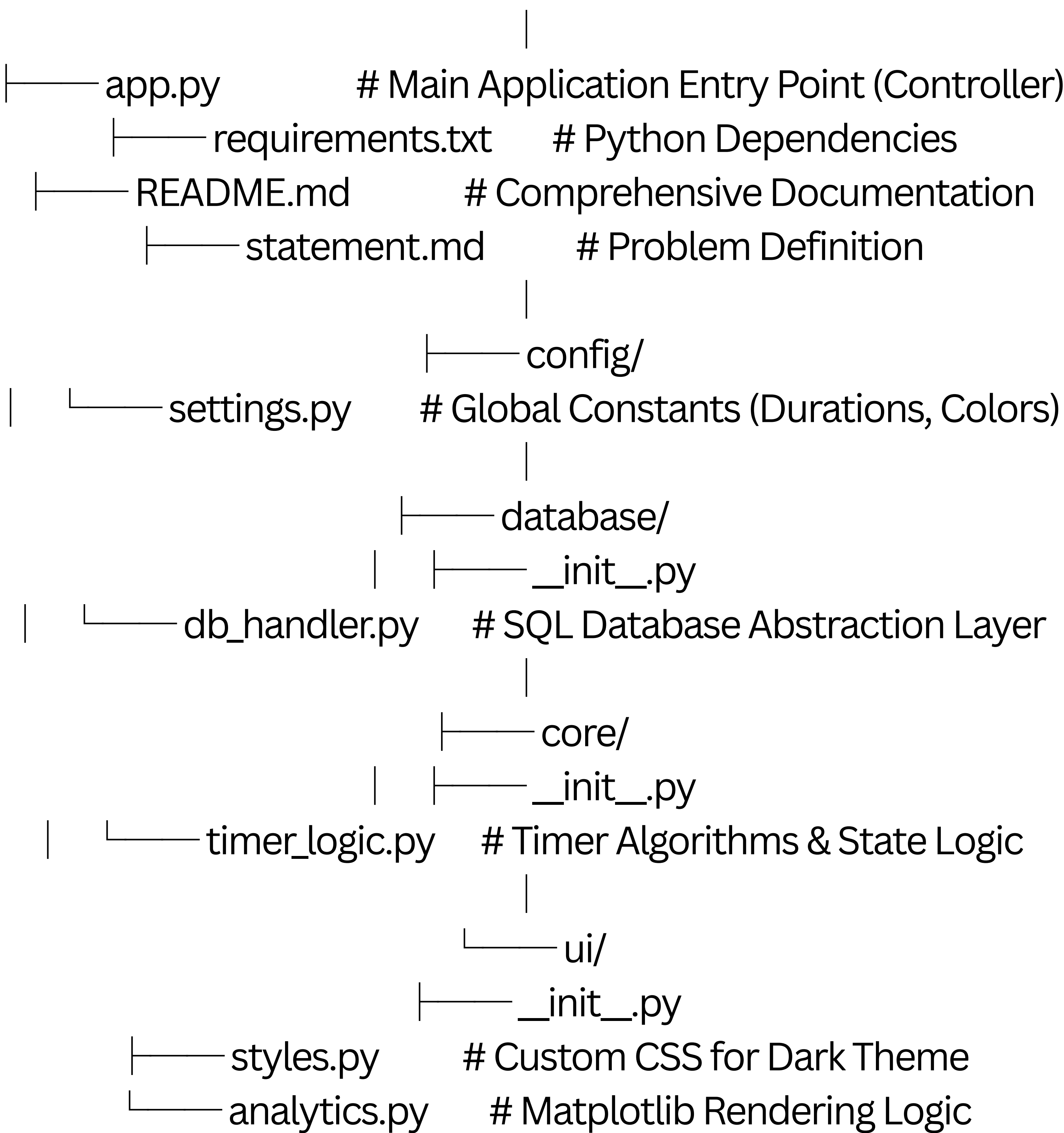
Model (Data Layer): Located in `database/db_handler.py`. This layer is responsible for all SQL interactions. It isolates the database schema from the rest of the application.

View (Presentation Layer): Located in `ui/`. This includes `styles.py` (CSS injection) and `analytics.py` (Chart rendering). It handles how data is displayed to the user.

Controller (Business Logic): Located in `core/timer_logic.py` and `app.py`. This layer processes user inputs (Start, Reset, Skip) and updates the Model and View accordingly.

.....

PomodoroFocus_Project/



.....

⚡ Installation & Setup Guide Prerequisites Python 3.8+ installed on your system. pip (Python Package Installer). Git (for version control). Step-by-

Step Installation Clone the Repository:

Bash

git clone cd PomodoroFocus_Project Environment Setup: It is best practice to use a virtual environment to isolate project dependencies. Windows: python -m venv venv then .\venv\Scripts\activate Mac/Linux:

python3 -m venv venv then source venv/bin/activate Install Dependencies: FocusFlow Pro relies on Streamlit, Pandas, and Matplotlib.

Bash

pip install -r requirements.txt Launch the Application: This command starts the local Streamlit server.

Bash

streamlit run app.py The application will automatically open in your default web browser at http://localhost:8501.

🩺 Testing Strategy The reliability of FocusFlow Pro was verified using a hybrid testing approach:

Unit Testing (Manual): Verified the db_handler correctly handles database locking and schema creation on fresh installs. Integration Testing: Validated that the "Undo Last" button correctly updates both the database count and the "Total Focus" metric in the UI simultaneously. UX/UI Testing: Confirmed that the dark theme renders correctly across different monitors and that charts remain legible without contrast artifacts. 🧠 Future Roadmap To further enhance the utility of FocusFlow Pro, the following features are planned for future iterations:

Cloud Synchronization: Migration to PostgreSQL/Firebase to allow users to sync session data across multiple devices. Task Categorization: Adding a tagging system to associate time slots with specific subjects (e.g., "Math", "Coding"). Gamification: Implementing "Streaks" and "Badges" to increase user motivation and retention. 📄 License & Acknowledgments License: MIT License. Methodology: Based on the Pomodoro Technique® by Francesco Cirillo. Tools: Built with Streamlit, Python, and SQLite.