Full Stack Developer Challenge: The "Wellness at Work" Cloud-Synced Eye Tracker

Objective: Develop a well-designed, user-friendly, and cross-platform (MacOS and Windows) Laptop application that integrates a Python-based eye-blink tracker. This application must securely sync user data to a cloud backend, with the data being accessible via a WaW web platform. The entire system must be GDPR compliant, ensuring user privacy and data protection are paramount.

This exercise will assess your technical skills and organizational abilities. Try to complete the app to the best of your ability. You are allowed to use coding agents.

Python Application Reference: https://github.com/wellnessatwork/eye-tracker-share

---

Core Functional Requirements:

This project is broken down into three main components: the **Cross-Platform Desktop App**, the **Cloud Backend & Database**, and the **Web Platform**.

1. Cross-Platform Desktop App

- **Technology Choice:** The cross-platform application should be developed using **PyQt**. Keep the GUI design modern but simple with black, white and shades of Grey.
- **User Authentication.**
- **Real-time Blink Tracking:**
    - Integrate the provided Python eye-tracker script.
    - The app's UI must display the real-time blink count for the logged-in user.
- **Performance Monitoring:** Within the main application UI, display CPU Usage (%), Memory Usage (MB), Energy Impact / Power Usage**.**
- **GDPR Compliance:** Add your answer in Readme.md . What have you done or plan to do if provided more time?

2. Cloud Backend & Database

- **Technology Choice:** Deploy backend service on AWS using S3, RDS and others.
- **Database Schema:**
    - Design a database schema to store user profiles (e.g., user name, email, consent) and their associated blink data.
- **Secure API:**
    - Receiving and storing blink data on the cloud service. The application should handle being offline gracefully and sync on reconnection.

    - Exposing user data to be consumed by the Waw web platform (this endpoint must be protected)
- **Security:** Add your answer in Readme.md . What have you done or plan to do if provided more time?

3. Web Platform

- **Technology Choice:** Choose a modern web framework to build a simple web-based dashboard.
- **Functionality:** This is a "read-only" platform. The focus is on securely fetching and visualizing the data from the cloud backend of WaW's individual user.

Addition Challenges (Optional):

1. Implement device notification when the blink counter is below a certain value.
2. Design and implement the application using a Microservice Architecture.
3. Verify the application's icon appears and executes from the Windows system tray and the macOS menu bar upon launch.
4. Convert the blink counter logic from Python to C++ for performance optimization.
5. Define testing methodology for different Mac and Windows OS (latest) and hardware.

Deliverables

1. **Source Code:** In GitHub repository.
2. **README.md:** A comprehensive document that includes:
    - High-level architecture diagram showing how the components interact.
    - An explanation of GDPR and security requirements.
    - Define sample TestCases to be deployed on CI pipeline.
3. **CI/CD deployment on Github along with sample test cases and define structure.**
4. **Distribution**
    - **Windows:** Package the application as a standalone installer (MSIX) or an executable (**.**exe).
    - **macOS: (Choose any one)**
        i. Option A (Preferred): The application should be correctly signed, archived, and uploaded to TestFlight.
        ii. Option B: If TestFlight is not feasible, provide a sandboxed enabled and signed .app bundle for direct distribution. **The App Sandbox entitlement must be enabled.**
        iii. Option C: Share a dmg file to be installed locally on the device.
    - **Testers:** Send the TestFlight invitation or distribution files to ishaan80@gmail.com and mehul.bhardwaj@outlook.com.