Methodology: AI-Powered Fatigue Detection System

1. Data Collection

Capture relevant data from existing hardware and software sources:

1. **Cameras**:
   - Use low-cost or existing cameras (e.g., workplace CCTV, laptop webcams).
   - Capture video at low resolution (e.g., 640x480) to reduce computational load.
   - Use libraries for video streaming:
     - **OpenCV** (for real-time video capture and processing).
     - **GStreamer** (for handling camera feeds in large systems).
2. **Desktop Activity**:
   - Track typing speed, errors, and mouse movement using:
     - **Pynput** (Python library for tracking mouse and keyboard activity).
     - **PyAutoGUI** (for GUI interaction tracking).
3. **Optional: Environmental Data**:
   - Measure ambient light levels from video frames.
   - Noise levels using microphones:
     - **PyAudio** (for capturing and processing audio data).

2. Preprocessing

Process raw input data to make it suitable for AI analysis.

1. **Posture Analysis**:
   - Use **pose estimation** to extract skeletal keypoints from video:
     - **MediaPipe** (Google's lightweight pose tracking library, fast and effective on low-end systems).
     - **OpenPose** (for more robust skeletal tracking but requires higher computational resources).
2. **Eye Gaze and Blink Detection**:
   - Detect eyes and track blinks using:
     - **Dlib** (for face and eye landmark detection).
     - **OpenCV** (for image preprocessing, such as grayscale conversion and histogram equalization).

3. **Task Monitoring Data**:
    - Normalize typing speed and mouse movement data:
        - Scale input data using **Scikit-learn** (for preprocessing functions like MinMaxScaler).
    - Smooth data using moving averages to remove noise.
4. **Environmental Features**:
    - Light levels: Compute brightness intensity from video using **OpenCV**.
    - Noise levels: Convert audio amplitude into a decibel scale with **NumPy**.

3. Feature Extraction

Extract meaningful features that correlate with fatigue.

1. **Posture Features**:
    - Extract:
        - Head tilt angle (indicates slouching or nodding off).
        - Spine curvature (slouching posture).
        - Movement frequency (reduced movement as a fatigue sign).
    - Tools:
        - **NumPy** and **Pandas** (for numerical feature calculations).
2. **Eye Features**:
    - Calculate:
        - Blink rate (blinks per minute).
        - Blink duration (average blink time).
        - Gaze deviation (focus vs. wandering).
    - Tools:
        - **OpenCV** for eye bounding box detection.
        - **SciPy** for temporal feature analysis.

3. **Behavioral Features**:
    - Analyze:
        - Typing speed variance.
        - Mouse trajectory patterns (straight vs. erratic movement).
    - Tools:
        - **Matplotlib** for visualizing behavioral patterns.
4. **Contextual Features**:
    - Aggregate environmental factors:
        - Low brightness levels as a fatigue indicator.
        - High noise as a stressor.

4. Model Training

Use supervised machine learning or deep learning models to classify fatigue levels.

1. **Data Preparation**:
    - Create labeled datasets:
        - Simulate fatigue by recording real-world scenarios (e.g., workers under fatigue and normal conditions).
        - Annotate datasets with fatigue levels (low, medium, high).
2. **Models**:
    - **Posture and Gesture Recognition**:
        - Use **Convolutional Neural Networks (CNNs)** for image data:
            - **MobileNet** (lightweight, pre-trained on COCO dataset).
        - Alternatively, use pose-based models like **Graph Convolutional Networks (GCNs)** for skeletal data.
    - **Eye Gaze Analysis**:
        - Use time-series models like **LSTMs** or **GRUs** to analyze blink and gaze data over time.
    - **Behavioral and Contextual Analysis**:
        - Train tree-based models like **LightGBM** on combined features.

3. **Tools**:

    - **TensorFlow** or **PyTorch** for deep learning models.
    - **Scikit-learn** for traditional machine learning models.

5. Integration and Inference

Combine model outputs for a comprehensive fatigue score.

1. **Multi-Model Fusion**:
    - Use an ensemble approach to combine predictions from:
        - Posture model.
        - Eye gaze model.
        - Behavioral model.
    - Weight each model's contribution based on validation accuracy.
2. **Real-Time Inference**:
    - Deploy models on lightweight systems:
        - **ONNX Runtime** (for optimized deep learning model inference).
        - **TensorFlow Lite** (for mobile or edge device deployment).
3. **Dashboard and Alerts**:
    - Visualize fatigue levels using:
        - **Streamlit** for building interactive dashboards.
        - Send alerts via notifications or email using **SmtpLib**.

6. Feedback and Recommendations

Provide actionable insights to users or supervisors.

1. **Feedback**:
    - Real-time alerts (e.g., “Take a break!” or “Adjust posture”).
    - Weekly summaries of fatigue trends.
2. **Recommendations**:
    - Suggest ergonomic interventions, breaks, or workload adjustments.

7. Evaluation and Optimization

Assess system performance and improve as needed.

1. **Metrics**:
   - Accuracy, precision, recall, and F1-score for model evaluation.
   - Use **TensorBoard** for tracking experiments.
2. **Deployment Testing**:
   - Simulate real-world conditions for validation.
   - Optimize latency and computational efficiency.