1. **INTRODUCTION**

**1.1 Document Purpose**

This Software Requirements Specification (SRS) defines the functional and non-functional requirements of a Python-based Attentiveness Monitoring Tool. The system analyzes facial features, especially eye movements, to detect and monitor the attention level of individuals in real-time using a video feed. It utilizes computer vision and machine learning techniques to generate attention reports. This document provides a complete understanding of the system requirements for developers, testers, and stakeholders.

**1.2 Product Scope**

The Attentiveness Monitoring Tool is designed to assess the attentiveness of individuals during sessions such as online classes, exams, or meetings. The system captures real-time video, detects the facial region, tracks eye movements, and calculates the attention level based on head orientation and eye direction. At the end of the session, it generates a detailed report indicating attentive and non-attentive durations. Future extensions may include attendance automation, live dashboards, and exit-entry monitoring.

**1.3 Intended Audience**

This document is intended for the project development team, academic mentor, testers, and future project evaluators. It serves as a guideline for understanding system design, implementation, and testing procedures.

**1.4 Definitions, Acronyms, and Abbreviations**

* **YOLOv8**: Object detection model
* **mediapipe**: A machine learning pipeline framework used for face and eye detection.
* **OpenCV**: Library for real-time computer vision tasks.
* **NumPy, Pandas**: Python libraries for data processing and analysis.

**1.5 Document Conventions**

* Fonts used: Times New Roman 12pt
* Metrics mentioned in standard units
* Technical terms capitalized when referenced

**1.6 References**

* MediaPipe official documentation
* OpenCV official documentation
* NumPy and Pandas library documentation
* IEEE Standard for Software Requirements Specifications

1. **OVERALL DESCRIPTION**

**2.1 Product Overview**

The system is a Python-based real-time attentiveness monitoring tool designed to operate through a camera feed. It identifies the facial region, tracks eye movements, and analyzes head orientation to assess attentiveness. By the end of a session, it generates a report detailing the periods of attentiveness and inattentiveness. This system is intended to assist instructors and supervisors in monitoring participants' focus levels without manual intervention.

**2.2 Product Functionality**

* Detect and track the user’s face and eyes in real-time using MediaPipe.
* Analyze eye movements and head orientation to determine attentiveness.
* Calculate attentive and non-attentive durations based on real-time data.
* Generate a session report summarizing the user’s attentiveness throughout the monitored period.

**2.3 Design and Implementation Constraints**

* The system will be implemented using Python.
* Libraries used include MediaPipe, OpenCV, NumPy, Pandas, and other necessary libraries.
* The system requires a high-definition camera with consistent frame capture capability.
* GPU acceleration is recommended but not mandatory for small-scale testing.
* All data processed should comply with general data protection norms, avoiding long-term storage of video feeds.

**2.4 Assumptions and Dependencies**

* A working camera feed is available throughout the session.
* Sufficient lighting is present for accurate face and eye detection.
* The user’s face remains mostly visible to the camera.
* MediaPipe and OpenCV models are pre-trained and operational.
* The session is recorded or monitored in real-time, and no face registration is required for attention detection.

1. **SPECIFIC REQUIREMENTS**
   1. **External Interface Requirements**

**3.1.1 User Interfaces**

The system will operate with a basic console or GUI-based interface. It will display the real-time status of attentiveness and offer options to start monitoring, stop monitoring, and generate the report. The report will be generated in a human-readable format like CSV or PDF or PNG.

**3.1.2 Hardware Interfaces**

The tool requires:

* A high-definition camera for video feed
* A computing system capable of running MediaPipe and OpenCV smoothly, preferably with GPU support for real-time analysis

**3.1.3 Software Interfaces**

The system uses:

* MediaPipe for facial landmark detection
* OpenCV for video capture and frame processing
* NumPy and Pandas for data handling and report generation
* Optional: PDF or CSV libraries for report export

**3.2 Functional Requirements**

* The system shall capture and process the real-time video feed to detect the user’s face and eyes.
* The system shall analyze eye movements and head position to determine if the user is attentive or not.
* The system shall log the attentive and non-attentive durations throughout the monitoring session.
* The system shall generate a session report indicating the total attentive and non-attentive periods.

**3.3 Use Case Description**

Use Case: **Real-Time Attentiveness Monitoring and Report Generation**

* Actor: System User (Instructor, Supervisor, or Automated System)
* Purpose: Monitor a person’s attentiveness during a session and generate a report at the end.
* Preconditions: The system is running with an active camera feed.
* Postconditions: An attentiveness report is generated summarizing attentive/non-attentive durations.
* Basic Flow:
  1. The system starts capturing the video feed.
  2. The user’s face and eyes are detected.
  3. Real-time attentiveness is analyzed based on facial orientation and eye position.
  4. The system logs attentive and non-attentive time.
  5. After completion, a report is generated.

1. **OTHER NON-FUNCTIONAL REQUIREMENTS**

**4.1 Performance Requirements**

* The system should process video frames in real-time with at least 10 FPS performance on standard hardware.
* Report generation should not exceed 30 seconds post-session.

**4.2 Safety and Security Requirements**

* The system must ensure that any sensitive visual data is not stored permanently unless explicitly required.
* Access to generated reports should be restricted to authorized users only.

**4.3 Software Quality Attributes**

* Reliability: The system must provide stable detection without frequent crashes during operation.
* Usability: The system interface should be simple enough for a non-technical user to start monitoring and generate reports.
* Maintainability: The architecture should allow future enhancements like adding attention scoring algorithms or integrating a dashboard.
* Flexibility: The system should support easy integration with additional features like attendance monitoring in the future.

1. **OTHER NON-FUNCTIONAL REQUIREMENTS**

**5.1 Performance Requirements**

* The system must maintain real-time video processing at a minimum of 10 frames per second (FPS) for a single subject with standard hardware.
* Report generation for a complete session (1 hour) must not exceed 30 seconds.
* The system should handle continuous monitoring sessions of up to 3 hours without crashing or memory leakage.

**5.2 Safety Requirements**

* The system must not store video footage unless explicitly configured.
* The system must handle hardware failures like camera disconnection by safely terminating the session and logging the event.
* Attention data should only be collected during authorized sessions.

**5.3 Security Requirements**

* Data logs and reports must be accessible only to authorized users.
* The system should avoid the permanent storage of facial video data, focusing on temporary memory operations.

**5.4 Software Quality Attributes**

* **Maintainability:** System architecture must support updates like adding dashboard modules or improving detection models.
* **Usability:** The tool will offer an intuitive interface to start/stop monitoring and generate reports.
* **Flexibility:** Designed to support integration with future modules like attendance, entry/exit tracking, and real-time dashboards.

**5.5 Business Rules**

* The system operates only during predefined session timings to prevent unnecessary monitoring.
* Reports are to be generated at the end of each session and optionally saved for future analysis.

1. **OTHER REQUIREMENTS**

* The system must support modular expansion to integrate attendance and entry/exit monitoring.
* Future versions should support dashboard visualization of real-time attention metrics.
* The system should adhere to institutional policies on privacy and data protection.

**APPENDIX A - DATA DICTIONARY**

**Libraries/Dependencies Used:**

* **Python (version: 3.7-3.10)**
* **MediaPipe:** Face and eye detection
* **OpenCV:** Video processing
* **NumPy:** Numerical calculations
* **Pandas:** Data analysis and report generation
* **Matplotlib/Seaborn:** For visual graphs in reports

**Key Data Elements:**

* **Face Coordinates:** (x, y, z) positions of face landmarks
* **Eye Status:** Open or closed detection for attention
* **Time Stamps:** Start and end time of attention/non-attention phases
* **Session Duration:** Total monitoring time
* **Attention Metric:** Percentage of time attentive vs. total time

**APPENDIX B - GROUP LOG / DEVELOPMENT LOG**

* **Initial Phase:** Defined the project scope — attentiveness monitoring (Week 1)
* **Research Phase:** Selected MediaPipe over YOLOv8 due to better eye tracking for attentiveness (Week 2)
* **Development:** Built real-time detection and tracking system using OpenCV and MediaPipe (Week 3-4)
* **Testing:** Validated accuracy and system performance on real-time streams (Week 5)
* **Report Module:** Implemented report generation in png (Week 6)
* **Future Planning:** Proposed live dashboard, entry/exit monitoring, and automated attendance (Week 7)

**APPENDIX C - TO BE DETERMINED (TBD) LIST**

* Integration of real-time **live dashboard** displaying attention status (TBD in future versions)
* Addition of **attendance and entry/exit detection modules** (TBD)
* **User authentication module** to secure report access (TBD)
* **Extended multi-camera support** for larger classroom environments (TBD)
* Integration with **cloud database storage** for large-scale deployment (TBD)