**Software Requirements Specification (SRS)**

**For Project:** *EyeD – AI-powered Attendance System with Face Recognition & Liveness Detection*

**1. Introduction**

**1.1 Purpose**

The purpose of this document is to define the **Software Requirements Specification (SRS)** for *EyeD*, an AI-powered attendance system that leverages **face recognition with liveness detection** to ensure reliable, transparent, and proxy-free attendance tracking. This document is intended for developers, project stakeholders, and academic evaluators. It specifies system behavior, functional and non-functional requirements, and environmental constraints.

**1.2 Document Conventions**

* IEEE 830 standard has been followed.
* Requirements are numbered as **FR#** for functional requirements.
* Non-functional requirements are referenced as **NFR#**.
* Terminology:
  + *User* → student/employee marking attendance.
  + *Admin* → teacher, HR, or manager overseeing attendance.
  + *System* → EyeD software solution.

**1.3 Intended Audience & Reading Suggestions**

* **Developers:** Use Sections 2, 3, and 4 for implementation guidance.
* **Administrators/Stakeholders:** Refer to Section 2 (Overall Description) and Section 6 (Non-Functional Requirements) to understand scope and constraints.
* **Testers:** Focus on Section 2.2 (System Features) and Section 7 (Testing Considerations).

**1.4 Project Scope**

Traditional attendance methods (manual registers, RFID cards, or fingerprint scanners) are prone to inaccuracies, inefficiency, and fraud. *EyeD* aims to provide a cost-free, open-source, and transparent solution by integrating **face recognition** with **liveness detection (blink verification)**. The system provides:

* Self-registration using selfies.
* Real-time recognition with liveness detection.
* Automatic logging into attendance sheets.
* A dashboard for visualization, analytics, and gamified insights.
* Local execution with hybrid deployment (local + cloud demo).

The system is **not intended** to replace enterprise-grade biometric systems or support very large-scale enterprise deployments (>10k concurrent users initially).

**1.5 References**

* IEEE 830-1998: IEEE Recommended Practice for Software Requirements Specifications.
* OpenCV Documentation (<https://docs.opencv.org/>)
* DeepFace Library (<https://github.com/serengil/deepface>)
* Streamlit Documentation (<https://docs.streamlit.io/>)

**2. Overall Description**

**2.1 Product Perspective**

*EyeD* is a **standalone application** with two components:

1. **Local Demo Application** (face recognition + liveness + logging).
2. **Streamlit Dashboard** (attendance visualization, analytics, gamification).

It integrates multiple open-source libraries:

* **OpenCV** → video capture, face detection.
* **DeepFace** → face embeddings & recognition.
* **Dlib** → facial landmarks & blink detection.
* **Streamlit + Plotly** → interactive dashboard.
* **CSV/JSON** → lightweight data storage.

**2.2 Product Functions**

The key functions include:

* User registration with selfie (stored as embedding vector).
* Face recognition using embeddings.
* Blink-based liveness detection.
* Attendance logging into CSV with metadata (confidence, timestamp).
* Streamlit dashboard with logs, filters, and analytics.
* Gamified elements (badges, emojis, timelines).
* Cloud deployment (logs + analytics with video demo).

**2.3 User Classes & Characteristics**

* **End-users (students/employees):** Casual users, minimal technical expertise, only interact during registration and attendance marking.
* **Administrators (teachers, HR):** Medium technical expertise, interact with dashboard for analytics and validation.
* **Developers/Evaluators:** Technical expertise, concerned with source code, deployment, and demo.

**2.4 Operating Environment**

* **Hardware:** Laptop/desktop with webcam.
* **OS:** Windows/Linux/macOS.
* **Programming Language:** Python 3.8+.
* **Libraries:** OpenCV, Dlib, DeepFace, Pandas, Numpy, Streamlit, Plotly.
* **Deployment:**
  + Localhost (recognition + liveness).
  + Streamlit Cloud (logs, analytics, demo video).

**2.5 Design & Implementation Constraints**

* Must use only **free, open-source resources**.
* Time limit: **16 days modular roadmap**.
* Lightweight storage (CSV/JSON).
* Performance requirement: recognition ≤ 1s per frame.

**2.6 Assumptions & Dependencies**

* Assumes stable webcam and lighting conditions.
* Assumes access to modern web browsers for dashboard.
* Depends on Python libraries (OpenCV, Dlib, DeepFace).
* Depends on GitHub (version control) and Streamlit Cloud (hosting).

**3. System Features**

**3.1 Core Setup & Face Registration**

* **FR1:** Provide a structured repository with folders for data, modules, dashboard, demos.
* **FR2:** Enable user registration by uploading/capturing a selfie.
* **FR3:** Extract embeddings (DeepFace MobileNet) and store in faces.json / faces.csv.

**3.2 Recognition & Liveness**

* **FR4:** Detect and recognize user’s face in real-time.
* **FR5:** Display recognized name + confidence score.
* **FR6:** Detect blink events using Eye Aspect Ratio (EAR).
* **FR7:** Mark user as “Verified Live” only if both recognition + blink detection succeed.

**3.3 Attendance Logging**

* **FR8:** Log attendance with Name, ID, Date, Time, Status, Confidence, Liveness Verified.
* **FR9:** Ensure unique daily entries per user.
* **FR10:** Provide real-time confirmation message upon successful logging.

**3.4 Dashboard Development**

* **FR11:** Sidebar menu with sections → Dashboard, Logs, Analytics, Registration.
* **FR12:** Filterable attendance logs.
* **FR13:** Attendance analytics (percentages, trends, late arrivals).
* **FR14:** Registration form in dashboard (uploads image + updates embeddings).
* **FR15:** Gamification → badges, emojis, timelines.

**3.5 Deployment & Demo**

* **FR16:** Provide demo video under /demos/.
* **FR17:** Deploy Streamlit dashboard on Streamlit Cloud with video-upload support.

**4. External Interface Requirements**

**4.1 User Interfaces**

* Streamlit-based dashboard with sidebar navigation.
* Minimal, modern UI (charts, tables, emojis, badges).
* Webcam capture for registration (local app).

**4.2 Hardware Interfaces**

* Webcam: Required for recognition and blink detection.

**4.3 Software Interfaces**

* DeepFace: Face recognition and embeddings.
* Dlib: Facial landmark detection.
* Pandas/CSV: Attendance logging.
* Streamlit/Plotly: Dashboard visualization.

**4.4 Communication Interfaces**

* Localhost: For local demo.
* HTTPS (Streamlit Cloud): For remote dashboard access.

**5. Non-Functional Requirements**

**5.1 Performance**

* Recognition latency ≤ 1 second per frame.
* Dashboard load time < 3 seconds.

**5.2 Security**

* Attendance logs stored locally.
* Face embeddings not shared externally.
* Admin-only dashboard access (future enhancement).

**5.3 Reliability & Availability**

* Local recognition works offline.
* Streamlit Cloud uptime ≥ 95%.

**5.4 Scalability**

* Supports 50–100 users initially.
* Can scale to thousands by migrating to relational databases (future).

**5.5 Usability**

* Registration → single image upload.
* Attendance → automatic once camera detects face.
* Dashboard → intuitive, visually engaging, gamified.

**6. Other Requirements**

**6.1 Legal/Compliance**

* Privacy: Biometric data stored only locally.
* GDPR/FERPA alignment: No cloud storage of raw face data.

**6.2 Testing Requirements**

* Unit tests for recognition, blink detection, logging.
* Integration testing for full flow (registration → recognition → logging → dashboard).
* Usability testing for dashboard.

**6.3 Documentation**

* **README.md** with setup instructions.
* **User Manual** for registration, attendance, dashboard usage.
* **Developer Documentation** for modules (face\_db, recognition, dashboard).

**7. Appendices**

* **Glossary:**
  + EAR – Eye Aspect Ratio.
  + Embeddings – numerical vector representing face features.
  + Proxy – marking attendance on behalf of another user.
* **Future Enhancements:**
  + Mobile app support.
  + OTP/email authentication for admin login.
  + Cloud database integration.
  + Advanced anti-spoofing (iris tracking, head movement).