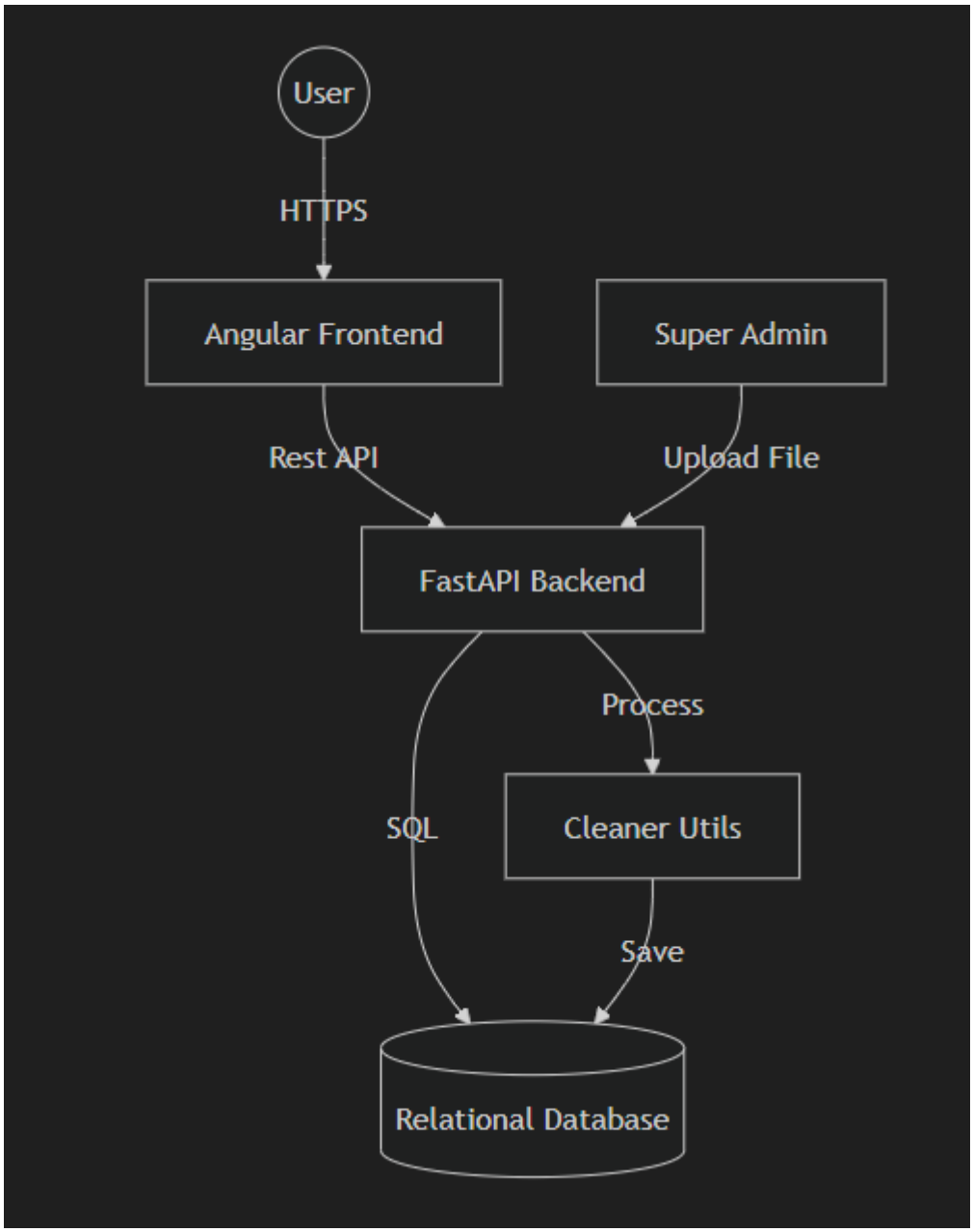
Biometric Attendance System - Production-Ready Architecture

This document outlines the evolutionary design of the Attendix platform, transitioning it from a stateless processing tool to a scalable, persistent, and secure office management system.

🏭 High-Level System Architecture

The system follows a classic 3-tier architecture for modularity and scalability.



Components:-

1. **Frontend (Angular)**: Responsible for visualization, role-based routing, and interactive reports.
2. **Backend (FastAPI)**: Handles authentication, file processing logic (Pandas), and database orchestration.
3. **Database (PostgreSQL/MySQL)**: Stores employee records, attendance history, and audit logs.

🔐 Role-Based Access Control (RBAC) Strategy

The system enforces strict data isolation based on roles.

Role	Permissions	Data Scope
Super Admin	Full CRUD, User Management, File Upload	Entire Company
HR User	Read-Only Access, Reporting	Entire Company
Employee	Read-Only Access	Self-only (Record filtered by EmpID)

Authorization Mechanism

- **JWT (JSON Web Tokens)**: Securely transmit claims (Role, EmpID) between client and server.
- **Middleware**: Backend decorators to verify permissions before executing sensitive operations (e.g., @require_role('SUPER_ADMIN')).
- **Data Query Isolation**: Queries for 'Employee' role will always append a WHERE emp_id = :session_id clause to prevent unauthorized data access.

🗄 Database Design

A relational design ensures data integrity and historical accuracy.

1. Employee Master Table

Stores identity and authentication details.

- id (PK)
- employee_id (Unique, used in biometric logs)
- name
- email
- password_hash
- role (ENUM: SUPER_ADMIN, HR, EMPLOYEE)
- status (Active, Inactive)
- created_at, updated_at

2. Attendance Daily Logs (Aggregated)

Stores daily processed data for quick reporting.

- id (PK)
- employee_id (FK)
- date (Date)
- first_in (Time)
- last_out (Time)
- total_duration (Interval/Minutes)
- status (Present, Absent, Half-Day)
- source_file_id (FK for traceability)

3. File Upload Audit

Tracks source files for accountability.

- id (PK)
- filename
- uploaded_by (FK)
- upload_timestamp
- report_type (Type A or Type B)

🗺 Normalization & Indexing

- **Composite Index**:

  (employee_id, date) on logs to speed up monthly report generation.

- **Partitioning**: If scaling to thousands of employees, partition the attendance table by month/year.

🔄 Data Lifecycle & ETL Process

1. **Extract**: Super Admin uploads raw

   .xls/.csv.

2. **Transform**:
   - Backend parses file using cleaner_utils.py.
   - Data is validated (Detects missing IDs, malformed timestamps).
   - Deduplication logic ensures same day records aren't duplicated.
3. **Load**: Records are persisted to the database.
4. **Correction Flow**: If an Admin modifies a record, the original biometric data is preserved, but a manual_correction flag and corrected_by column are updated.

Risks, Edge Cases & Future Extensions

Risks & Edge Cases

- **Overlapping Uploads**: Handling cases where an admin uploads the same month twice (Idempotency).
- **Timezone Conflicts**: Ensuring server and biometric device timestamps are synchronized (UTC recommended for storage).
- **EmpID Mismatch**: Handling raw files where EmpID doesn't exist in the Employee Master.

Future Extension Ideas

- **Mobile Integration**: Geo-fencing for remote check-ins.
- **AI Forecasting**: Predictive analytics for employee absence trends.
- **Multitenancy**: Supporting multiple offices or sub-companies in one instance.
- **Leave Management Integration**: Automatically marking "Present" vs "Leave" by syncing with HRMS.

> Verification Plan

This design will be verified against the requirements through structural review.

Structural Verification

- ☑ Ensure all 3 roles have distinct defined paths.
- ☑ Validate that the database schema supports "Correction Traceability".
- ☑ Confirm the architecture supports "Historical Data Preservation" via snapshots/persistence.