**REPUBLIC OF CAMEROON**

Peace-Work-Fatherland

**UNIVERSITY OF BUEA**

P.O. Box 63.

Buea, South West Region

CAMEROON

Tel: (237)33322134/33322690

Fax: (273)33322272



**FACULTY OF ENGINEERING AND TECHNOLOGY**

**DEPARTMENT OF COMPUTER ENGINEERING**

**COURSE**: INTERNET PROGRAMMING AND MOBILE PROGRAMMING. **CEF 440**

**COURSE COORDINATOR**: DR VALERY NKEMENI

**TASK 6:**

**Database design and implementation**

**GROUP 15 MEMBERS:**

1. EKANE CLINTON NTONGWE-FE22A198
2. ETUNDI ZAMBO JOSIANE-FE22A212
3. FAI NJI JR BAHTINYUY-FE22A215
4. NEBA NELLY ACHA-FE22A255
5. NGWA NATHAN NINJECK-FE22A268

Second semester Task 6 report, (B.ENG) Degree in Computer Engineering.

JUNE

2025

Contents

[Introduction 3](#_Toc200362051)

[Data elements 3](#_Toc200362052)

[ Primary Data Elements 3](#_Toc200362053)

[Conceptual Design 7](#_Toc200362054)

[ Database Architecture Overview 7](#_Toc200362055)

[ER Diagram 8](#_Toc200362056)

[ Entity Relationship Model 8](#_Toc200362057)

[ER Diagram Visualization 12](#_Toc200362058)

[Database Implementation 13](#_Toc200362059)

[ SQL Schema Creation 13](#_Toc200362060)

[**Backend implementation** 22](#_Toc200362061)

[Back-end Implementation Strategy for Attendance Management System 22](#_Toc200362062)

[System Architecture 22](#_Toc200362063)

[**Connecting our backend to our database** 26](#_Toc200362064)

[**Troubleshooting Common Issues** 28](#_Toc200362065)

[Conclusion 29](#_Toc200362066)

# Introduction

In today's data-driven environment, effective database design and implementation serve as the foundation for robust information systems that support organizational operations and decision-making processes. This report presents a comprehensive analysis of the database design and implementation project undertaken for [Your Project/Organization Name], detailing the systematic approach employed to create a scalable, efficient, and secure database solution.

The primary objective of this project was to design and implement a database system that addresses specific business requirements while ensuring data integrity, optimal performance, and future scalability. Through careful analysis of user requirements, thorough system design, and methodical implementation, this project demonstrates the practical application of database design principles and best practices.

This report documents the complete lifecycle of the database project, from initial requirements gathering and conceptual modeling through physical implementation and testing. Key areas covered include the analysis of existing systems, identification of functional and non-functional requirements, development of conceptual and logical data models, selection of appropriate database management systems, and the implementation process with accompanying performance evaluations.

The methodology employed follows established database design principles, incorporating entity-relationship modeling, normalization techniques, and consideration of ACID properties to ensure a robust and reliable database solution. Additionally, this report addresses critical aspects such as security implementation, backup and recovery strategies, and performance optimization techniques that were integrated throughout the development process.

# Data elements

Data Elements refers to **individual data units that comprise a dataset or database**. It represents specific attributes or fields within a record, including names, phone numbers, credit card numbers, dates, numerical values, or other discrete information.

## Primary Data Elements

**User Management Data**

**User ID**: Unique identifier for each system user

**Full Name**: Complete name of the user

**Email Address**: Primary contact and login credential

**Password Hash**: Encrypted password for authentication

**User Role**: Classification (Student, Lecturer, Admin)

**Registration Date**: Account creation timestamp

**Profile Image**: User's photograph for identification

**Status**: Account status (Active, Inactive, Suspended)

**Student-Specific Data**

**Student ID**: Institutional student identifier

**Program**: Academic program/course of study

**Level**: Academic level (100L, 200L, 300L, 400L, etc.)

**Department**: Academic department

**Faculty**: Academic faculty

**Lecturer-Specific Data**

**Staff ID**: Institutional staff identifier

**Department**: Teaching department

**Title**: Academic title (Dr., Prof., Mr., Mrs., etc.)

**Office Location**: Physical office address

**Attendance Data**

**Attendance ID**: Unique record identifier

**User ID**: Reference to the user marking attendance

**Session ID**: Reference to the attendance session

**Check-in Time**: Timestamp of attendance marking

**Location Data**: GPS coordinates at check-in

**Location Verified**: Boolean for geofence validation

**Face Verified**: Boolean for facial recognition success

**Attendance Status**: Present, Late, Absent

**Device Info**: Information about the device used

**Session Management Data**

**Session ID**: Unique session identifier

**Course Code**: Subject/course identifier

**Session Title**: Descriptive name for the session

**Start Time**: Scheduled session start time

**End Time**: Scheduled session end time

**Location**: Physical location of the session

**Geofence Coordinates**: Boundary coordinates for location validation

**Created By**: Lecturer who created the session

**Session Status**: Active, Completed, Cancelled

**Facial Recognition Data**

**Face ID**: Unique identifier for face data

**User ID**: Reference to the user

**Face Encoding**: Encrypted facial feature data

**Created Date**: When the face data was captured

**Updated Date**: Last modification timestamp

**Quality Score**: Confidence score of the face capture

**Geofencing Data**

**Geofence ID**: Unique geofence identifier

**Location Name**: Descriptive name (e.g., "Main Lecture Hall")

**Center Latitude**: Central GPS latitude coordinate

**Center Longitude**: Central GPS longitude coordinate

**Radius**: Geofence boundary radius in meters

**Created By**: Admin who defined the geofence

**Status**: Active, Inactive

* **System Data Elements**

**Audit Trail Data**

**Log ID**: Unique log entry identifier

**User ID**: User who performed the action

**Action Type**: Type of action performed

**Timestamp**: When the action occurred

**IP Address**: Source IP address

**Device Info**: Device information

**Success Status**: Whether the action succeeded

**Configuration Data**

**Config ID**: Configuration parameter identifier

**Parameter Name**: Name of the configuration setting

**Parameter Value**: Value of the configuration

**Description**: Description of what the parameter controls

**Modified By**: User who last modified the setting

**Modified Date**: When the setting was last changed

# Conceptual Design

Conceptual database design is the process of identifying the essential data elements, relationships, and [constraints](https://www.tutorialspoint.com/constraints-in-dbms) in a data model, which represents a particular organization's business requirements

## Database Architecture Overview

The Attendance Management App database follows a **relational database model** with the following design principles:

**Design Principles**

**Normalization**: Database is normalized to 3NF to eliminate redundancy

**Referential Integrity**: Foreign key constraints ensure data consistency

**Security**: Sensitive data is encrypted and access is role-based

**Scalability**: Designed to handle growth in users and attendance records

**Performance**: Optimized with appropriate indexing strategies

Core Entity Relationships

**Users** serve as the central entity with specializations for Students, Lecturers, and Admins

**Sessions** represent attendance opportunities created by Lecturers

**Attendance Records** link Users to Sessions with verification data

**Facial Recognition Data** provides biometric verification capabilities

**Geofences** define location boundaries for attendance validation

* **Data Flow Architecture**

**Input Data Flow**

**User Registration**: New users provide personal information and biometric data

**Session Creation**: Lecturers define attendance sessions with location and time parameters

**Attendance Marking**: Students check-in through location and facial verification

**Administrative Actions**: Admins manage users, sessions, and system configuration

**Processing Data Flow**

**Authentication**: User credentials are validated against encrypted stored data

**Location Verification**: GPS coordinates are checked against geofence boundaries

**Facial Recognition**: Captured images are processed against stored face encodings

**Attendance Logging**: Verified attendance is recorded with full audit trail

**Output Data Flow**

**Attendance Reports**: Generated for lecturers and administrators

**User Dashboards**: Personalized views of attendance history

**Analytics**: System usage and attendance pattern analysis

**Notifications**: Real-time alerts for attendance events

# ER Diagram

## Entity Relationship Model

**Entities and Attributes**

**USER**

user\_id (PK)

full\_name

email

password\_hash

user\_role

registration\_date

profile\_image\_url

status

created\_at

updated\_at

**STUDENT** (Inherits from USER)

student\_id (PK, FK)

institutional\_id

program

level

department

faculty

enrollment\_date

**LECTURER** (Inherits from USER)

lecturer\_id (PK, FK)

staff\_id

department

title

office\_location

hire\_date

**ADMIN** (Inherits from USER)

admin\_id (PK, FK)

admin\_level

permissions

**SESSION**

session\_id (PK)

course\_code

session\_title

start\_time

end\_time

location\_name

created\_by (FK → LECTURER)

geofence\_id (FK → GEOFENCE)

session\_status

created\_at

updated\_at

**ATTENDANCE**

attendance\_id (PK)

user\_id (FK → USER)

session\_id (FK → SESSION)

check\_in\_time

location\_latitude

location\_longitude

location\_verified

face\_verified

attendance\_status

device\_info

created\_at

**FACE\_DATA**

face\_id (PK)

user\_id (FK → USER)

face\_encoding

quality\_score

created\_date

updated\_date

**GEOFENCE**

geofence\_id (PK)

location\_name

center\_latitude

center\_longitude

radius\_meters

created\_by (FK → ADMIN)

status

created\_at

updated\_at

**AUDIT\_LOG**

log\_id (PK)

user\_id (FK → USER)

action\_type

timestamp

ip\_address

device\_info

success\_status

details

Relationships

**USER → STUDENT/LECTURER/ADMIN** (1: 0...1) – ISA relationship

**LECTURER → SESSION** (1 :M) - Lecturers create multiple sessions

**USER → ATTENDANCE** (1:M) - Users have multiple attendance records

**SESSION → ATTENDANCE** (1:M) - Sessions have multiple attendance records

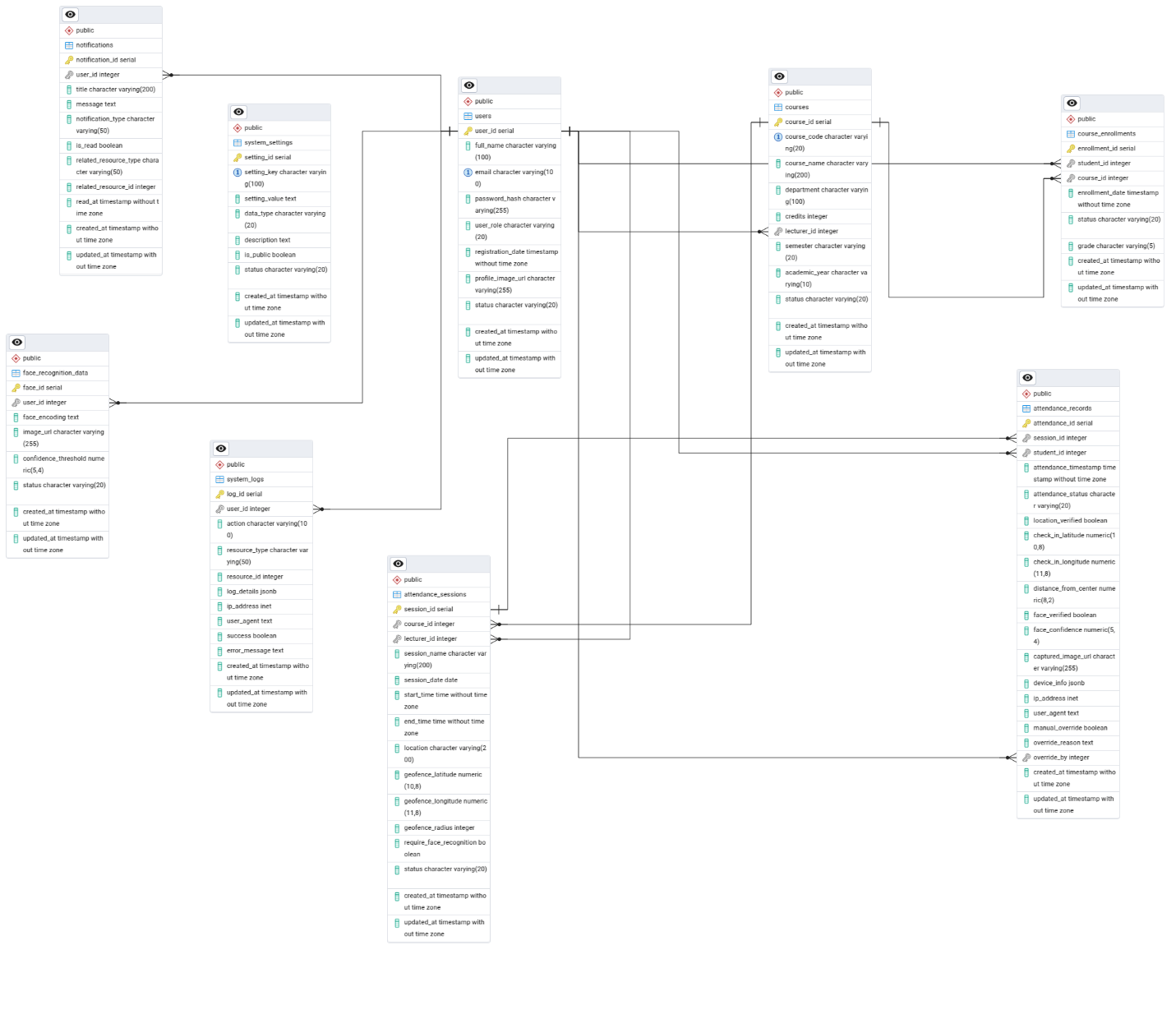
**USER → FACE\_DATA** (1:M) - Users can have multiple face captures

**GEOFENCE → SESSION** (1:M) - Geofences can be used by multiple sessions

**ADMIN → GEOFENCE** (1:M) - Admins create geofences

**USER → AUDIT\_LOG** (1:M) - Users generate multiple log entries

## ER Diagram Visualization



# Database Implementation

It refers to the process of designing, creating, and deploying a database system to meet specific application requirements. It involves translating conceptual data models into actual, functional database structures that can store, manage, and retrieve data efficiently.

## SQL Schema Creation

**Database Creation**

CREATE DATABASE attendance\_management\_db;

USE attendance\_management\_db;

Core Tables

**Users Table**

CREATE TABLE users (

user\_id INT PRIMARY KEY AUTO\_INCREMENT,

full\_name VARCHAR(100) NOT NULL,

email VARCHAR(100) UNIQUE NOT NULL,

password\_hash VARCHAR(255) NOT NULL,

user\_role ENUM('student', 'lecturer', 'admin') NOT NULL,

registration\_date TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

profile\_image\_url VARCHAR(255),

status ENUM('active', 'inactive', 'suspended') DEFAULT 'active',

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

updated\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP,

INDEX idx\_email (email),

INDEX idx\_role (user\_role),

INDEX idx\_status (status)

);

**Students Table**

CREATE TABLE students (

student\_id INT PRIMARY KEY,

institutional\_id VARCHAR(20) UNIQUE NOT NULL,

program VARCHAR(100) NOT NULL,

level VARCHAR(10) NOT NULL,

department VARCHAR(100) NOT NULL,

faculty VARCHAR(100) NOT NULL,

enrollment\_date DATE NOT NULL,

FOREIGN KEY (student\_id) REFERENCES users(user\_id) ON DELETE CASCADE,

INDEX idx\_institutional\_id (institutional\_id),

INDEX idx\_department (department),

INDEX idx\_level (level)

);

**Lecturers Table**

CREATE TABLE lecturers (

lecturer\_id INT PRIMARY KEY,

staff\_id VARCHAR(20) UNIQUE NOT NULL,

department VARCHAR(100) NOT NULL,

title VARCHAR(20) NOT NULL,

office\_location VARCHAR(100),

hire\_date DATE NOT NULL,

FOREIGN KEY (lecturer\_id) REFERENCES users(user\_id) ON DELETE CASCADE,

INDEX idx\_staff\_id (staff\_id),

INDEX idx\_department (department)

);

**Admins Table**

CREATE TABLE admins (

admin\_id INT PRIMARY KEY,

admin\_level ENUM('super', 'standard') DEFAULT 'standard',

permissions JSON,

FOREIGN KEY (admin\_id) REFERENCES users(user\_id) ON DELETE CASCADE

);

**Geofences Table**

CREATE TABLE geofences (

geofence\_id INT PRIMARY KEY AUTO\_INCREMENT,

location\_name VARCHAR(100) NOT NULL,

center\_latitude DECIMAL(10, 8) NOT NULL,

center\_longitude DECIMAL(11, 8) NOT NULL,

radius\_meters INT NOT NULL DEFAULT 50,

created\_by INT NOT NULL,

status ENUM('active', 'inactive') DEFAULT 'active',

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

updated\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP,

FOREIGN KEY (created\_by) REFERENCES admins(admin\_id),

INDEX idx\_location (center\_latitude, center\_longitude),

INDEX idx\_status (status)

);

**Sessions Table**

CREATE TABLE sessions (

session\_id INT PRIMARY KEY AUTO\_INCREMENT,

course\_code VARCHAR(20) NOT NULL,

session\_title VARCHAR(200) NOT NULL,

start\_time DATETIME NOT NULL,

end\_time DATETIME NOT NULL,

location\_name VARCHAR(100) NOT NULL,

created\_by INT NOT NULL,

geofence\_id INT NOT NULL,

session\_status ENUM('scheduled', 'active', 'completed', 'cancelled') DEFAULT 'scheduled',

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

updated\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP,

FOREIGN KEY (created\_by) REFERENCES lecturers(lecturer\_id),

FOREIGN KEY (geofence\_id) REFERENCES geofences(geofence\_id),

INDEX idx\_course\_code (course\_code),

INDEX idx\_start\_time (start\_time),

INDEX idx\_status (session\_status)

);

**Face Data Table**

CREATE TABLE face\_data (

face\_id INT PRIMARY KEY AUTO\_INCREMENT,

user\_id INT NOT NULL,

face\_encoding LONGBLOB NOT NULL,

quality\_score DECIMAL(3, 2) DEFAULT 0.00,

created\_date TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

updated\_date TIMESTAMP DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP,

FOREIGN KEY (user\_id) REFERENCES users(user\_id) ON DELETE CASCADE,

INDEX idx\_user\_id (user\_id)

);

**Attendance Table**

CREATE TABLE attendance (

attendance\_id INT PRIMARY KEY AUTO\_INCREMENT,

user\_id INT NOT NULL,

session\_id INT NOT NULL,

check\_in\_time TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

location\_latitude DECIMAL(10, 8),

location\_longitude DECIMAL(11, 8),

location\_verified BOOLEAN DEFAULT FALSE,

face\_verified BOOLEAN DEFAULT FALSE,

attendance\_status ENUM('present', 'late', 'absent') DEFAULT 'present',

device\_info JSON,

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

FOREIGN KEY (user\_id) REFERENCES users(user\_id) ON DELETE CASCADE,

FOREIGN KEY (session\_id) REFERENCES sessions(session\_id) ON DELETE CASCADE,

UNIQUE KEY unique\_user\_session (user\_id, session\_id),

INDEX idx\_user\_id (user\_id),

INDEX idx\_session\_id (session\_id),

INDEX idx\_check\_in\_time (check\_in\_time),

INDEX idx\_status (attendance\_status)

);

**Audit Log Table**

CREATE TABLE audit\_logs (

log\_id INT PRIMARY KEY AUTO\_INCREMENT,

user\_id INT,

action\_type VARCHAR(50) NOT NULL,

timestamp TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

ip\_address VARCHAR(45),

device\_info JSON,

success\_status BOOLEAN DEFAULT TRUE,

details TEXT,

FOREIGN KEY (user\_id) REFERENCES users(user\_id) ON DELETE SET NULL,

INDEX idx\_user\_id (user\_id),

INDEX idx\_action\_type (action\_type),

INDEX idx\_timestamp (timestamp)

);

4.2 Sample Data Population

Insert Sample Users

-- Insert sample admin

INSERT INTO users (full\_name, email, password\_hash, user\_role) VALUES

('System Administrator', 'admin@university.edu', '$2b$12$hash...', 'admin');

INSERT INTO admins (admin\_id, admin\_level) VALUES

(LAST\_INSERT\_ID(), 'super');

-- Insert sample lecturer

INSERT INTO users (full\_name, email, password\_hash, user\_role) VALUES

('Dr. John Smith', 'j.smith@university.edu', '$2b$12$hash...', 'lecturer');

INSERT INTO lecturers (lecturer\_id, staff\_id, department, title, office\_location, hire\_date) VALUES

(LAST\_INSERT\_ID(), 'STF001', 'Computer Engineering', 'Dr.', 'Block A, Room 205', '2020-01-15');

-- Insert sample students

INSERT INTO users (full\_name, email, password\_hash, user\_role) VALUES

('Alice Johnson', 'alice.j@student.edu', '$2b$12$hash...', 'student'),

('Bob Wilson', 'bob.w@student.edu', '$2b$12$hash...', 'student');

INSERT INTO students (student\_id, institutional\_id, program, level, department, faculty, enrollment\_date) VALUES

((SELECT user\_id FROM users WHERE email = 'alice.j@student.edu'), 'FE22A001', 'Computer Engineering', '400L', 'Computer Engineering', 'Engineering and Technology', '2022-09-01'),

((SELECT user\_id FROM users WHERE email = 'bob.w@student.edu'), 'FE22A002', 'Computer Engineering', '400L', 'Computer Engineering', 'Engineering and Technology', '2022-09-01');

Insert Sample Geofences

INSERT INTO geofences (location\_name, center\_latitude, center\_longitude, radius\_meters, created\_by) VALUES

('Main Lecture Hall', 4.1563, 9.2871, 100, 1),

('Computer Lab Block', 4.1565, 9.2875, 75, 1),

('Engineering Building', 4.1560, 9.2870, 150, 1);

Insert Sample Sessions

INSERT INTO sessions (course\_code, session\_title, start\_time, end\_time, location\_name, created\_by, geofence\_id) VALUES

('CEF440', 'Internet Programming Lecture 1', '2025-06-09 08:00:00', '2025-06-09 10:00:00', 'Main Lecture Hall', 2, 1),

('CEF440', 'Mobile Programming Lab', '2025-06-09 14:00:00', '2025-06-09 16:00:00', 'Computer Lab Block', 2, 2);

# **Backend implementation**

**It** refers to the server-side architecture and logic that powers an application or system. It encompasses all the behind-the-scenes functionality that users don't directly interact with, including server configuration, application logic, API development, database connectivity, authentication systems, and business rule processing. The backend acts as the bridge between the user interface (frontend) and the database, handling requests, processing data, and returning appropriate responses.

## Back-end Implementation Strategy for Attendance Management System

## ****System Architecture****

**Technology Stack**

**Framework**: FastAPI - excellent for async operations and auto-documentation

**Database**: PostgreSQL with SQLAlchemy ORM for complex relationships

**Authentication**: JWT tokens with role-based access control

**Face Recognition**: DeepFace + OpenCV

**Geofencing**: PostGIS extension for spatial queries

**Task Queue**: Celery with Redis for background processing

**Caching**: Redis for session management and frequent queries

**Database Schema Design**

**Core Tables**

**Users**: Enhanced with student\_id, department, phone number

**Courses**: Course code, name, instructor, semester, credits

**Class\_Sessions**: Scheduled classes with time, location, geofence data

**Enrollments**: Student-course relationships

**Attendance\_Records**: Core attendance tracking with timestamps

**Geofence\_Zones**: Campus boundaries with GPS coordinates

**Biometric\_Data**: Face embeddings linked to users

**Relationship Structure**

User → Multiple Enrollments → Multiple Courses

Course → Multiple Class\_Sessions → Multiple Attendance\_Records

User → One Biometric\_Data record

Class\_Session → One Geofence\_Zone

**API Endpoint Architecture**

**Authentication Module** (/auth)

POST /auth/register - Student/instructor registration

POST /auth/login - JWT token generation

POST /auth/refresh - Token refresh

GET /auth/profile - User profile management

PUT /auth/profile - Update profile information

**Course Management** (/courses)

GET /courses - List enrolled courses (students) or taught courses (instructors)

POST /courses - Create new course (admin/instructor)

GET /courses/{id}/sessions - Get class schedule

POST /courses/{id}/sessions - Schedule new class session

**Biometric Registration** (/biometrics)

POST /biometrics/enroll - Register face during onboarding

PUT /biometrics/update - Update facial data

DELETE /biometrics/remove - Remove biometric data

GET /biometrics/status - Check enrollment status

**Attendance Core** (/attendance)

POST /attendance/checkin - Primary attendance marking endpoint

GET /attendance/history - Student attendance history

GET /attendance/session/{id} - Attendance for specific class

PUT /attendance/manual - Manual attendance correction (instructor)

GET /attendance/report - Generate attendance reports

**Geofencing** (/location)

POST /location/verify - Verify if user is within campus bounds

GET /location/zones - Get available geofence zones

POST /location/zones - Create new geofence (admin)

**Core Business Logic Implementation**

* **Attendance Marking Flow**
* **Geofencing Implementation**
* **Face Recognition Pipeline**

**Security & Validation Layers**

**Multi-Factor Attendance Verification**

**Time Window**: Only allow attendance ±15 minutes from class start

**Location**: GPS coordinates must be within geofenced area

**Biometric**: Face match confidence > 85%

**Session Validation**: Student must be enrolled in the course

**Duplicate Prevention**: One attendance record per session per student

**Anti-Spoofing Measures**

**Liveness Detection**: Require blink detection or head movement

**Image Quality**: Minimum resolution and lighting requirements

**Rate Limiting**: Prevent rapid attendance attempts

**Device Fingerprinting**: Track device used for attendance

**Performance Optimization**

**Database Optimization**

**Indexing**: Multi-column indexes on (student\_id, session\_id, timestamp)

**Partitioning**: Partition attendance records by semester/year

**Caching**: Redis cache for frequently accessed course data

**Read Replicas**: Separate read/write operations

**Image Processing**

**Async Processing**: Use Celery for face embedding generation

**Image Compression**: Optimize storage with WebP format

**CDN Storage**: Store reference images in cloud storage

**Batch Processing**: Process multiple faces in single request

**API Performance**

**Connection Pooling**: Optimize database connections

**Response Compression**: Enable gzip compression

**Pagination**: Limit large result sets

**Background Tasks**: Move heavy operations to background queues

**Error Handling & Monitoring**

**Comprehensive Error Management**

**Logging & Analytics**

**Structured Logging**: JSON logs with correlation IDs

**Metrics Collection**: Track success rates, response times

**Alert System**: Notify on system failures or suspicious activity

**Audit Trail**: Log all attendance modifications

**Deployment & Scalability**

**Containerization**

**Docker**: Multi-stage builds for production optimization

**Docker Compose**: Local development environment

**Health Checks**: Application and database health endpoints

**Scalability Considerations**

**Horizontal Scaling**: Load balancer with multiple API instances

**Database Sharding**: Partition by semester or department

**Microservices**: Separate services for auth, biometrics, attendance

**Message Queues**: Decouple time-sensitive operations

This implementation provides a robust, scalable backend that handles the complexity of combining geofencing with facial recognition while maintaining security and performance standards.

# **Connecting our backend to our database**

* To do this, we made use of the following steps

**Step 1: Install Required Dependencies**

First, we created a requirements.txt file and install the necessary packages:

**Step 2: Set Up Database Configuration**

We then created our database configuration files:

**Step 3: Create Environment Configuration**

We created a .env file in our project root with the following information:

env

# Database Configuration

DATABASE\_URL=postgresql://your\_username:your\_password@localhost:5432/attendance\_db

# Security

SECRET\_KEY=your-super-secret-key-change-this-in-production

ALGORITHM=HS256

ACCESS\_TOKEN\_EXPIRE\_MINUTES=30

# Environment

ENVIRONMENT=development

DEBUG=true

**Step 4: Set Up PostgreSQL Database**

**- We Installed PostgreSQL**

**- We created Database and User**

- **Installed PostGIS** (for geofencing):

**Step 5: Set Up Database Migrations with Alembic**

* **We Initialized Alembic:**
* **Configured Alembic (alembic.ini):**

**sqlalchemy.url = postgresql://your\_username:your\_password@localhost:5432/attendance\_db**

* **Updated Alembic env.py:**

**from config.database import Base**

**from models import user, course, attendance**

**target\_metadata = Base.metadata**

* **Created and Ran Migrations**:

**Step 6: Test Database Connection**

To do this, we created a simple test script:

**Step 7: Run Your Application**

* **We then Started the FastAPI server** and **Tested the endpoints**:

Visit http://localhost:8000 for the root endpoint

Visit http://localhost:8000/docs for automatic API documentation

Visit http://localhost:8000/health to check database connectivity

# **Troubleshooting Common Issues**

**Connection refused**: Check if PostgreSQL service is running

**Authentication failed**: Verify username/password in DATABASE\_URL

**Database doesn't exist**: Create the database first

**PostGIS not found**: Install postgis extension in your database

**Import errors**: Ensure all model files are imported in main.py

Run the test script to verify everything is working correctly before proceeding with your API endpoints!

# Conclusion

This comprehensive database and backend design provides a robust foundation for the Attendance Management App with the following key features:

**Normalized Structure**: Eliminates data redundancy and maintains consistency

**Facial Recognition Support**: Secure storage and processing of biometric data

**Geofencing Integration**: Flexible location-based attendance validation

**Comprehensive Reporting**: Built-in views and procedures for analytics

**Maintenance Ready**: Includes archival, backup, and monitoring capabilities

This implementation aligns perfectly with the system architecture and provides a solid foundation for the mobile and web applications that will interact with this data layer.

our backend is now ready to handle the complex requirements of multi-factor attendance verification while maintaining the security, performance, and scalability standards outlined in our original architecture. The foundation we've built will support man concurrent users and can easily scale as our system grows.