WEB-BASED FACIAL AUTHENTICATION SYSTEM

High Level Design of the proposed solution

1. SYSTEM DESIGN OVERVIEW

1.1 HIGH-LEVEL ARCHITECTURE

Client Side

Web Browser Interface:
Webcam Integration -> Captures live video
User Interaction:

- Shows live video feed.
- Displays messages like “Face Detected” / “No Face Detected.”
- Displays access results (e.g., “This is Alex, 20 years old, a student.”)

Server Side

API Layer:

- Receives face images (frames) from the frontend.
- Routes requests for:
  - Face Detection
  - Face Recognition
  - Database queries

Face Detection & Recognition Engine:

- Detects if the image contains a face.
- Extracts features (embeddings) from the face.
- Compares to stored face embeddings (database).

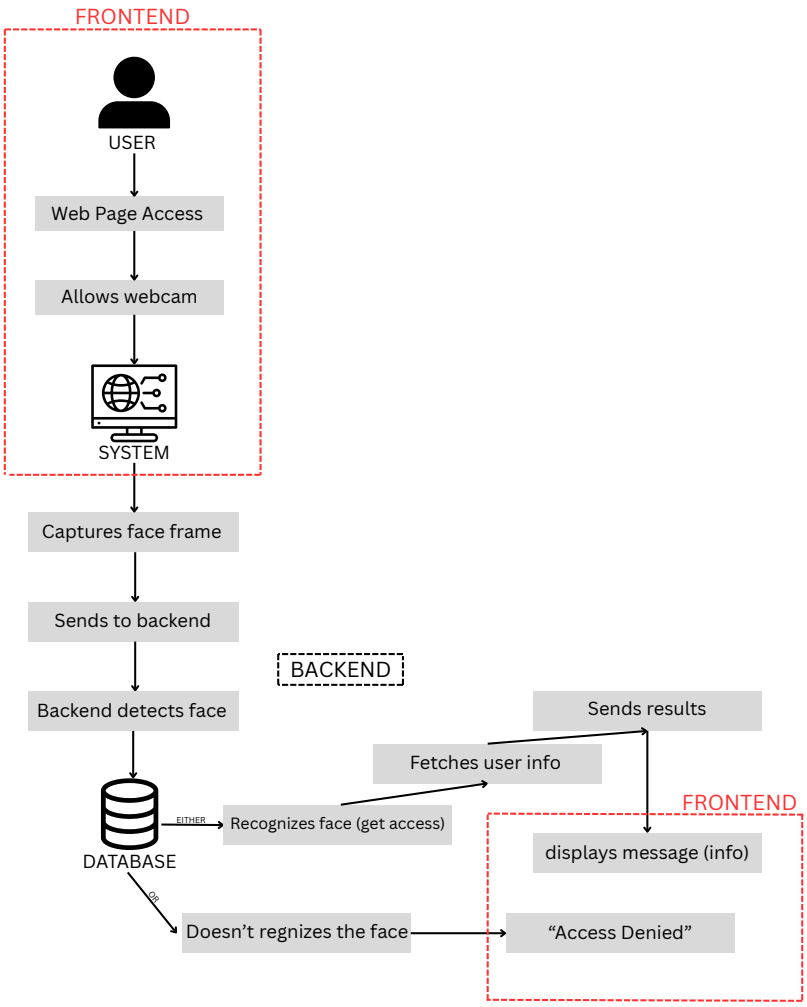
Database

User Profiles Table
Logs Table:
Records each access attempt (time, result, face detected or not).

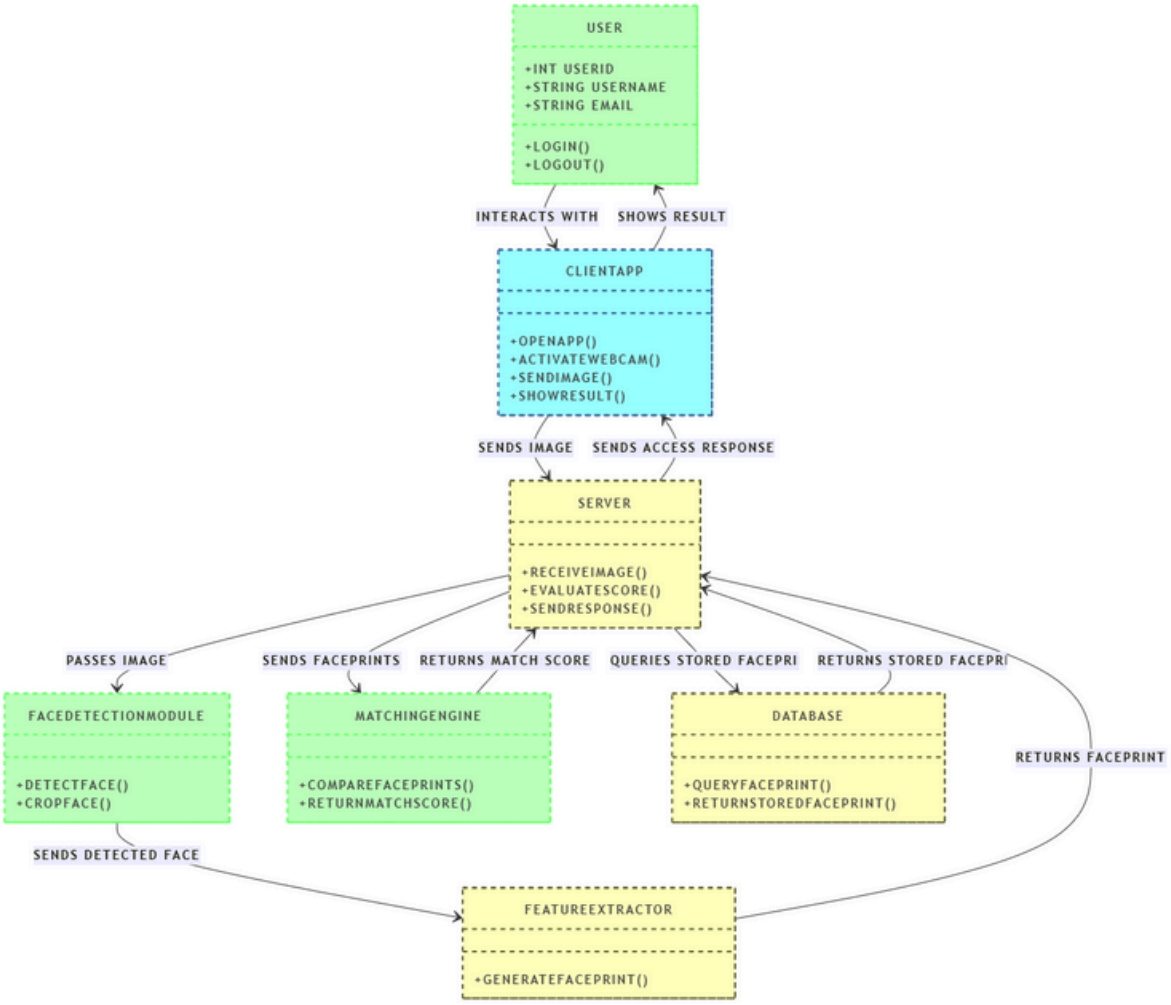
Authentication & Security

Secure API: HTTPS + token-based authentication for safe data transfer.
Privacy: Encrypt sensitive data (especially face embeddings).

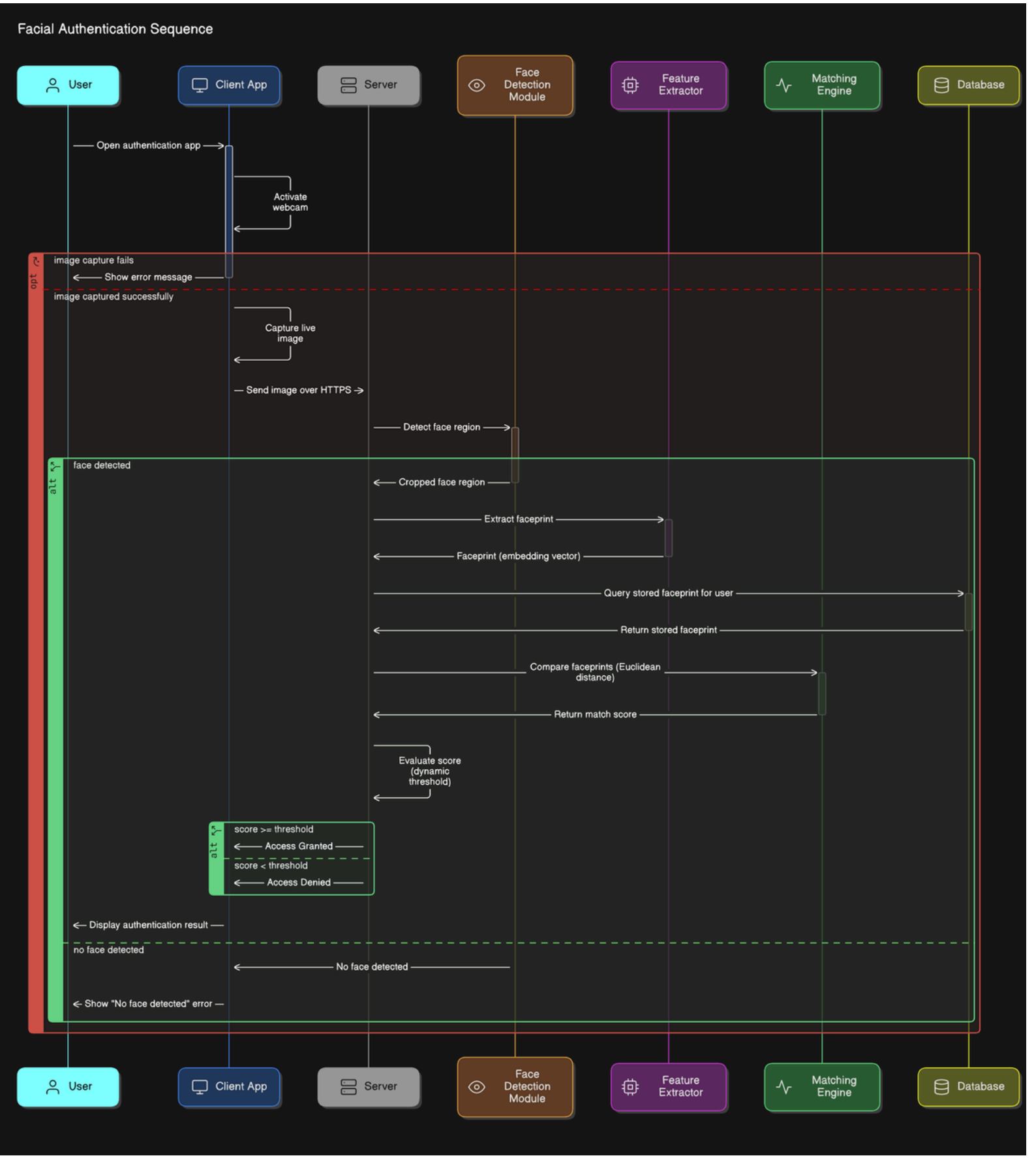
1.2 SYSTEM ARCHITECTURE DIAGRAM

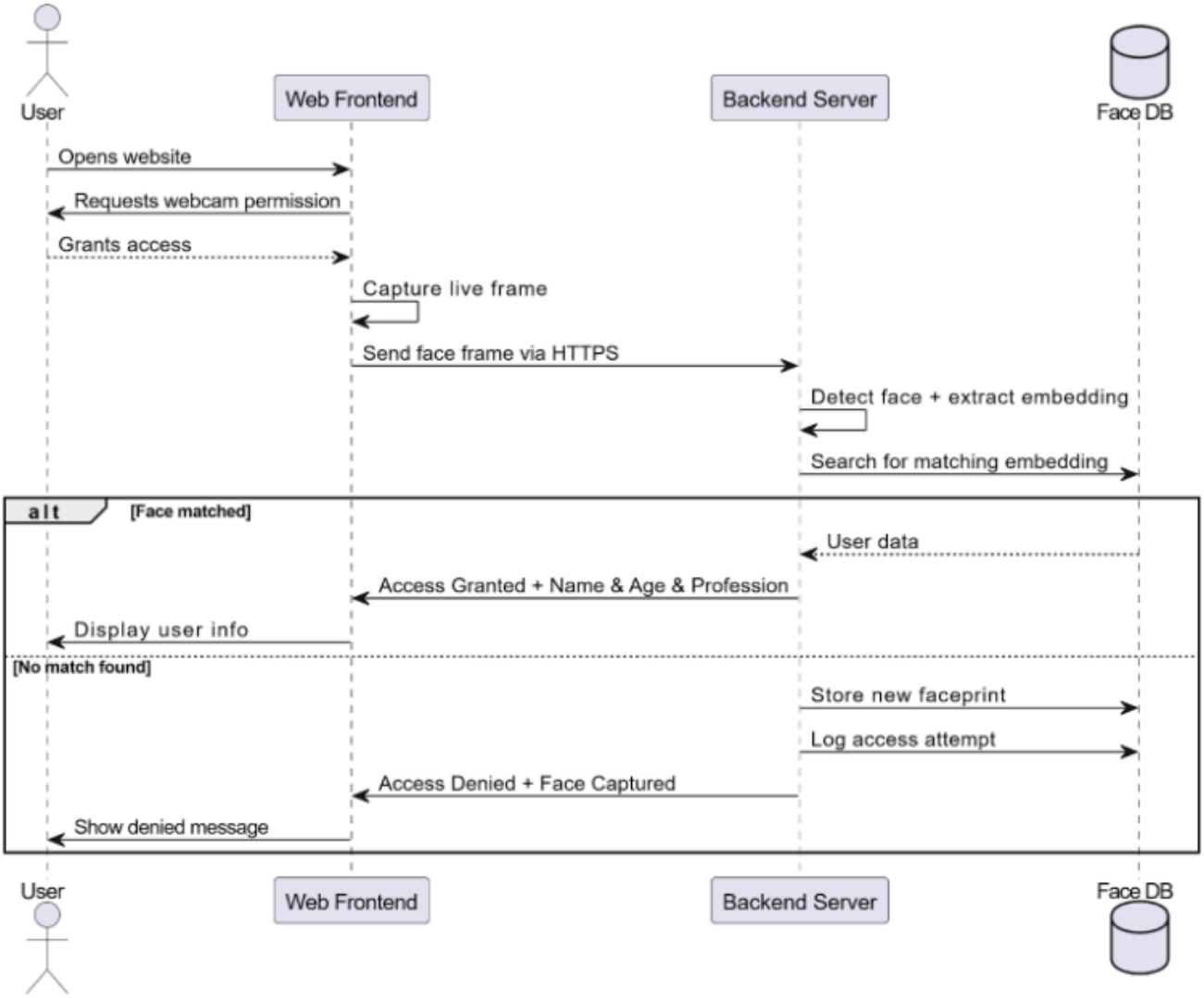


1.3 CLASS DIAGRAM

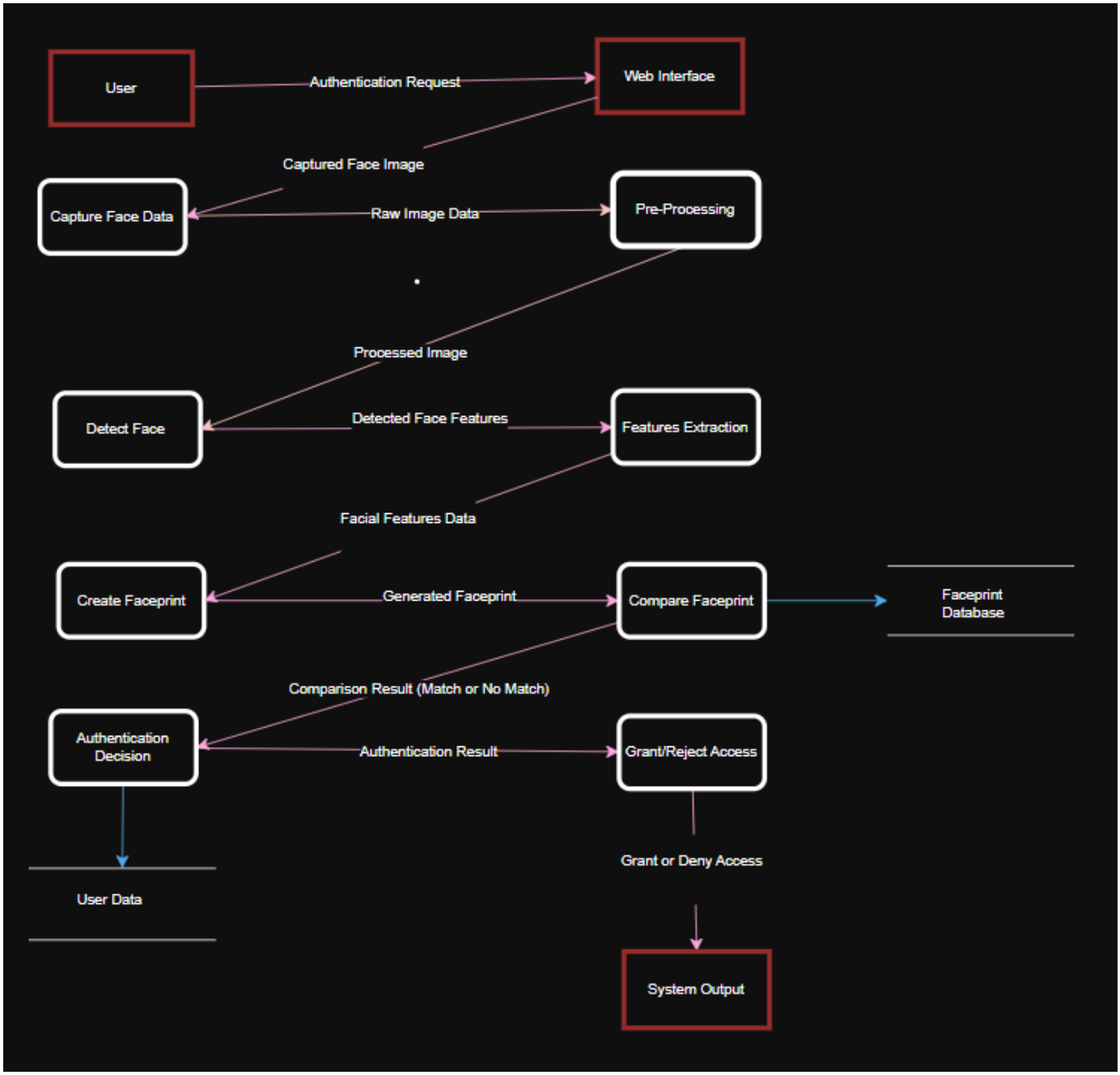


1.4 SEQUENCE DIAGRAM




User
Web Frontend
Backend Server
Face DB
Opens website
Requests webcam permission
Grants access
Capture live frame
Send face frame via HTTPS
Detect face + extract embedding
Search for matching embedding
alt
[Face matched]
User data
Access Granted + Name & Age & Profession
Display user info
[No match found]
Store new faceprint
Log access attempt
Access Denied + Face Captured
Show denied message
User
Web Frontend
Backend Server
Face DB

1.5 DATA FLOW DIAGRAM

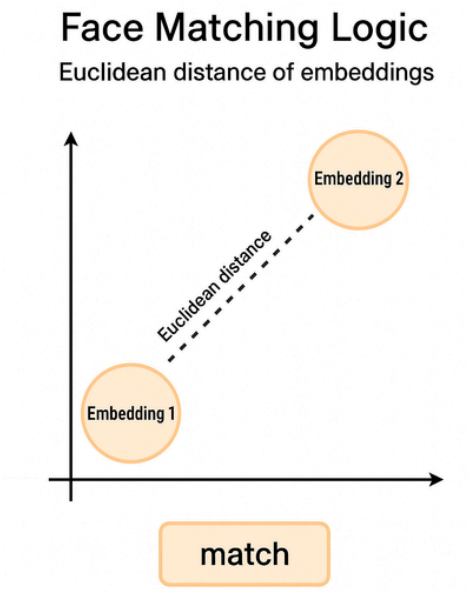


2. SYSTEM COMPONENTS

The facial authentication system is composed of several key components, each responsible for a specific role in the authentication pipeline. The interaction between these components enables secure, accurate, and efficient identity verification:

Component	Role
Camera Module	Captures live facial images or video streams for enrollment and login.
Face Detector	Detects and extracts the facial region using models like MTCNN or Haar.
Feature Extractor	Generates facial embeddings using deep learning models (e.g., FaceNet).
Matching Engine	Performs face matching via Euclidean distance of feature vectors.
Database	Stores encrypted faceprints and user metadata.
Liveness Detector	Detects live faces and prevents spoofing with motion/eye-blink checks.

Each of these components communicates with others through secure APIs and is designed to be modular for easy updates or integration with additional security layers.



3. WORKING FLOWS/ MESSAGE EXCHANGE

This section describes how different components of the system interact when users try to access the site, and how data moves between the frontend, backend, and database.

3.1 Authentication Flow (Live Camera Recognition)

When someone opens the website, it immediately asks for permission to access their webcam. If the system recognizes the face, it fetches the user's data (like name and age and job) and displays it. If not, the system denies access but automatically captures the face and stores it for future use.

Step-by-Step Process:

1. The user visits the website.
2. The browser requests camera access.
3. If granted, the webcam shows a live feed on the webpage.
4. The frontend detects when a face appears and sends the frame to the server (via secure API).
5. The backend:
    - Runs face detection and feature extraction (embedding).
    - Checks the database for a matching faceprint.
6. If found:
    - Fetches user metadata (e.g., "This is Alex, 20 years old").
    - Sends "Access Granted" + user info back to the client.
7. If not found:
    - Stores the faceprint and logs it.
    - Sends "Access Denied" + "New face captured" message.

3.2 Optional Admin Metadata Update Flow

Sometimes, unknown users are automatically added to the database without name/age/profession. An admin can later log in to label these entries (e.g., assign names to new faceprints).

Steps:
1-Admin logs in via secure portal (with password & MFA).
2-Admin views list of recently added unknown users.
3-Admin selects a user and updates metadata (name, age, profession).
4-Server updates the entry in the encrypted database.

3.3 Notes on Secure Message Exchange

Security and privacy are a top priority in all interactions. Here are the key protections in place:

Secure Communication
-All data (face images, API calls) is sent over HTTPS to prevent eavesdropping.
-Token-based authentication is used for API access control.

Data Storage
-Only face embeddings are stored — raw images are discarded after processing.
- All faceprints are encrypted in the database.
-Logs are maintained for every access attempt (success or failure).

Privacy and Control
-Users can request deletion of their data.
-Admins must go through role-based scopes and MFA for sensitive actions.

4. Roles and Permissions

This section defines the different roles in the facial authentication system and outlines the permissions associated with each. Clear separation of responsibilities helps ensure system integrity, security, and accountability.

4.1. Role Granularity

Role	Description	Permissions
User	The end-user who authenticates to access a service or system.	- Authenticate using face recognition - Enroll (if allowed by admin policy)
Admin	A privileged role responsible for user management and system supervision.	- Manually enroll new users - Upload user images - View authentication logs - Remove or update faceprints
System	The backend system performing all facial processing tasks.	- Detect and extract facial features - Compare faceprints for authentication - Log access results - Securely store and retrieve data

4.2. Critical security additions

- Just-In-Time (JIT) Admin Access:
  - Admins request temporary elevated privileges (e.g., 1-hour window) approved via MFA.
- Role-Based API Scopes:
  - Example: POST /enroll requires admin:write, while GET /logs requires auditor:read.
- Break-Glass Protocol:
  - Emergency superuser role (e.g., root) with manual approval and full activity tracing.

4.3. User privacy controls

- Right to Erasure: Users can request faceprint deletion (GDPR compliance).
- Transparency: Allow users to see when/why their face data was accessed (e.g., via audit logs).

🛡️ Security Note:

Admin access is password-protected and restricted to trusted personnel only. System components enforce strict API-based access control to avoid unauthorized usage.