**GST application:**

**6.(a) Brief Elaboration & Requirements Analysis**

**Scenario Overview**

* GST (General Science and Technology) admission system helps students avoid traveling to multiple universities for admission.
* After 4th or 5th migration, open calls require universities to manually verify thousands of students via **document verification** and **viva**, which is time-consuming.
* An ML-based system is proposed to automate **student registration and verification**.
* Students **register with live pictures and signatures**.
* On verification day, students provide a **live image & signature**, which is matched with stored data for **automated attendance**.

**Functional Requirements**

1. **Student Registration**
   * Capture student details, live photo (multiple angles), and signature.
   * Store them securely with **face recognition and signature verification model training**.
2. **Authentication & Verification**
   * On appearance day, capture **real-time image and signature**.
   * Match them with stored records to verify student identity.
3. **Attendance System**
   * If verification is successful, mark attendance.
   * If failed, notify officials for manual checking.
4. **Admin Panel**
   * View student details, verification results, and attendance logs.
   * Manage system configurations.
5. **Security Features**
   * Prevent **spoofing attacks** (e.g., printed photos, videos).
   * Ensure **data encryption** and GDPR compliance.

This is a great project that leverages machine learning and image processing to automate student verification for GST university admissions. Below is a structured approach to address your requirements:

## ****(b) Activity & Swimlane Diagrams****

### ****Activity Diagram Steps****

1. Student enters merit-based registration.
2. The system asks for **live photo and signature**.
3. The system **trains the ML model** for recognition.
4. On physical appearance day:
   * The student provides a **real-time image & signature**.
   * The system **compares and verifies identity**.
   * If matched → **Attendance is recorded**.
   * If not matched → **Manual verification required**.

### 

### ****Swimlane Diagram****

* **Actors**: Student, System, Admin
* **Lanes**:
  + **Student** → Register, upload documents, appear for verification.
  + **System** → Store data, process ML verification, mark attendance.
  + **Admin** → Monitor records, intervene if issues arise.

## ****(c) State & Sequence Diagram****

### ****State Diagram****

* **States**:
  1. **Registering** (Student enters data)
  2. **Training Model** (System processes images)
  3. **Verification Pending** (Waiting for student to appear)
  4. **Verification Process** (Live check-in process)
  5. **Matched** → Attendance recorded
  6. **Not Matched** → Manual verification

### ****Sequence Diagram****

* **Actors**: Student, Camera, System, Admin
* **Flow**:
  1. Student **registers and submits photo & signature**.
  2. System **stores data & trains ML model**.
  3. On verification day, student provides **live image & signature**.
  4. System **matches data**.
  5. If **match found**, mark attendance.  
     Else, send **manual verification request to admin**.

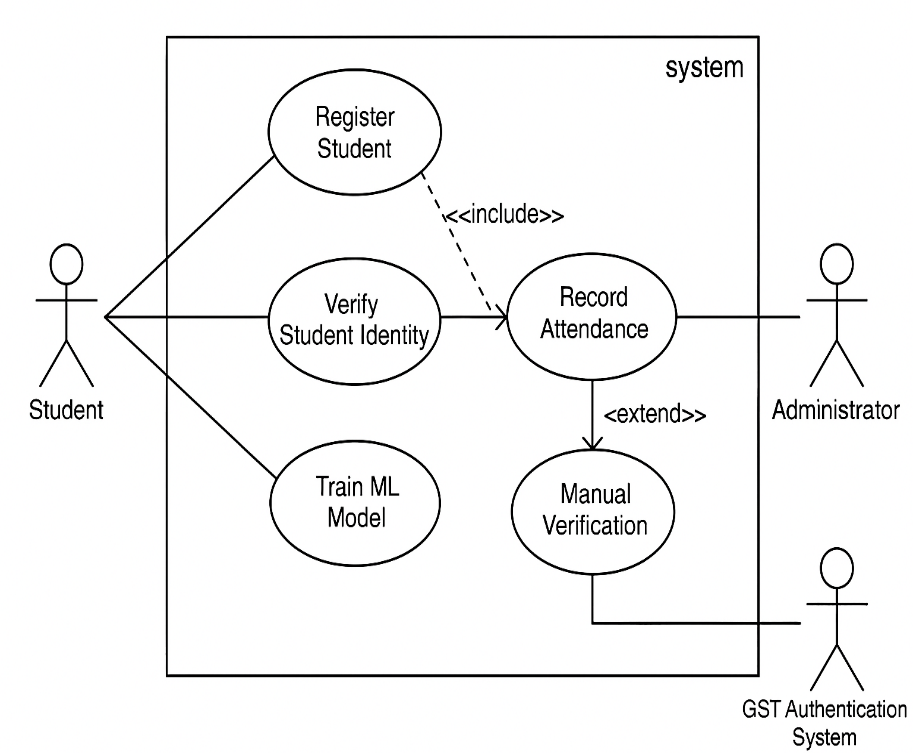
## ****7.(a) Use Case Diagram****

### Possible Use Cases:

1. **Student Registration** - The student provides their details, live pictures, and signature.
2. **System Learning** - The system processes and learns the registered students' images and signatures.
3. **Student Verification** - The system verifies students using live camera feed and signature input.
4. **Attendance Recording** - The system marks attendance for verified students.
5. **Manual Verification** - If the system cannot verify a student, a manual verification process is triggered.
6. **Administrator Management** - Admins oversee the registration, verification, and attendance records.
7. **Data Synchronization** - The system synchronizes data with the GST authentication system.

**Use Case Diagram Components:**

* **Actors:** Student, System, Administrator
* **Relationships:** Students interact with the registration, verification, and attendance modules, while the system manages authentication and data processing.



## ****(b) Software Testing Types with Justification****

Testing Type Justification

1. Accuracy Testing Verify ML model's face/signature recognition accuracy (FP/FN rates)

2. Load Testing Simulate 3000+ concurrent registrations/open calls to check scalability

3. Security Testing Pen-test biometric data storage and GST authentication to prevent breaches

4. Usability Testing Ensure intuitive UI for students/admins (e.g., easy document upload)

5. Integration Testing Validate GST system ↔ ML application data sync

Tools: Selenium (UI), JMeter (load), OWASP ZAP (security), TensorFlow Model Analysis (ML accuracy).

## ****(f) Data Flow Diagram (DFD)****

### ****Level 0****

* **Student → Register → System**
* **System → Train ML Model**
* **Student → Verification → System → Attendance Log**

### ****Level 1****

* **Registration Module** → Takes inputs, stores data.
* **Processing Module** → Trains ML model.
* **Verification Module** → Matches images & signatures.
* **Attendance Module** → Records results.

## ****(g) Class Design (CRC Cards)****

|  |  |  |
| --- | --- | --- |
| **Class** | **Responsibilities** | **Collaborators** |
| Student | Submit docs, Attend call | RegistrationSystem |
| RegistrationSystem | Validate data, Assign unit | Student, Admin |
| BiometricVerifier | Match face/signature | MLModel |
| AttendanceLogger | Record verification | Database |

## ****(h) Possible Test Cases****

| **Test Case** | **Expected Outcome** |
| --- | --- |
| Register student with valid data | Student successfully registered |
| Register student with blurry image | System prompts for a clear image |
| Verification with correct face & signature | System grants access & marks attendance |
| Verification with incorrect face | System denies access |
| Fake photo (printed) attempt | System detects fraud & rejects |
| System downtime during verification | Admin is notified |

## ****(i) ER Diagram****

### ****Entities & Relationships****

* **Student** (StudentID, Name, MeritRank)
  + Has **one-to-many** **Photos & Signatures**
* **Verification** (StudentID, Date, Status)
* **Admin** (AdminID, Name)

An **ER Diagram** visually represents these relationships.

Which diagram I am not included here, It’s present in my personal paper that is drawn by my own hand.