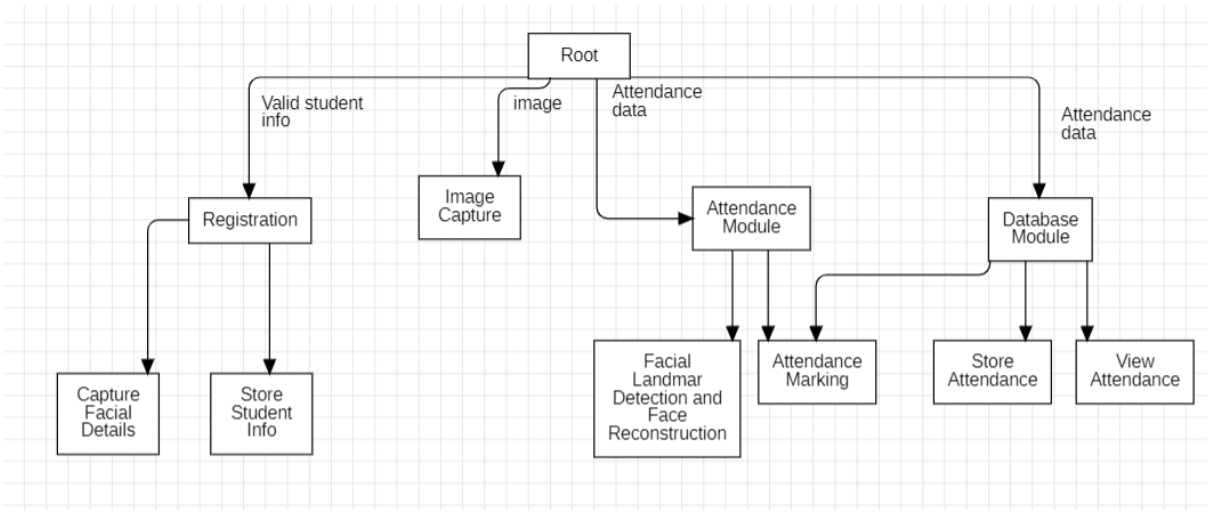
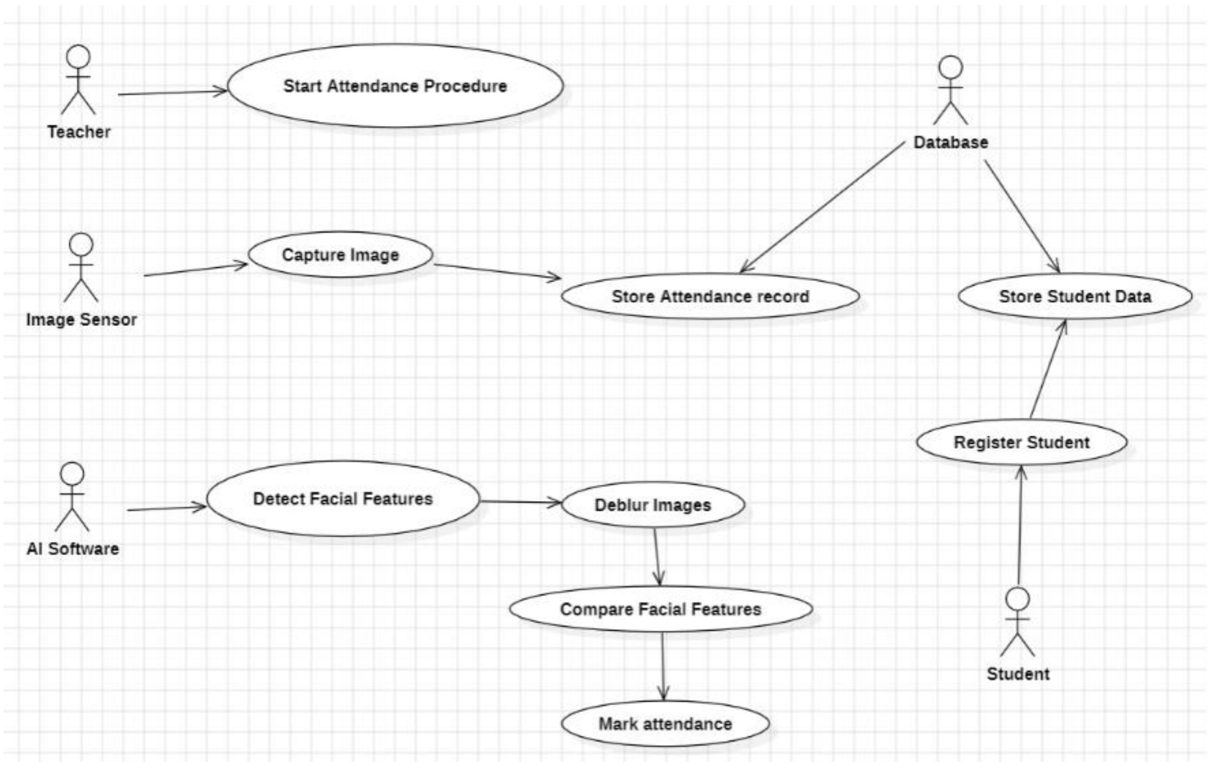
SOFTWARE ENGINEERING LAB.

ASSESMENT 4

Structure Chart



Use Case Diagram (Properly display the include and extend relationship among use cases)



Use Case Description for all the use cases (Use the appropriate Template – for each use case shown in the Use Case Diagram)

Use Case ID:	AttendanceSystem-UC001		
Use Case Name:	Start Attendance Procedure		
Created By:	Haneesha	Last Updated By:	Haneesha
Date Created:	20/03/2024	Date Last Updated:	21/03/2024

Actor:	Teacher
Description:	Initiates the process to take attendance using the facial recognition system.
Preconditions:	- Teacher logged in - System initialized and operational
Postconditions:	- Attendance session started
Priority:	High
Frequency of Use:	Daily
Normal Course of Events:	1. Teacher selects "Start Attendance" 2. System activates cameras for capture | 3. Teacher confirms to start session
Alternative Courses:	none
Exceptions:	- Camera failure - System error
Includes:	Capture Image Detect Facial Features Compare Facial Features Mark Attendance
Special Requirements:	Functional cameras, clear view of faces
Assumptions:	Students are in camera view
Notes and Issues:	Timely start and end are important

Use Case ID:	AttendanceSystem-UC002		
Use Case Name:	Capture Image		
Created By:	Mounika	Last Updated By:	Mounika
Date Created:	20/3/2024	Date Last Updated:	21/3/2024

Actor:	Image Sensor
Description:	Camera captures images during attendance session for facial recognition.
Preconditions:	- Cameras operational - Attendance session in progress
Postconditions:	- Image captured and stored
Priority:	High
Frequency of Use:	Continuous
Normal Course of Events:	1. System activates cameras 2. Cameras capture images 3. Images stored in database
Alternative Courses:	None
Exceptions:	- Camera failure - Image storage error
Includes:	None
Special Requirements:	Functional cameras, image storage
Assumptions:	Cameras positioned for clear images
Notes and Issues:	Image quality crucial for recognition

Use Case ID:	AttendanceSystem-UC003		
Use Case Name:	Detect Facial Features		
Created By:	Ananya	Last Updated By:	Ananya
Date Created:	20/3/2024	Date Last Updated:	21/3/2024

Actor:	AI Software
Description:	Facial recognition system detects facial landmarks and features for subsequent processing.
Preconditions:	- Images captured and stored - System initialized and running
Postconditions:	- Facial features detected
Priority:	High
Frequency of Use:	Continuous
Normal Course of Events:	1. System processes captured images 2. Facial landmarks detected \ 3. Features used for comparison
Alternative Courses:	none
Exceptions:	- Detection failure
Includes:	none
Special Requirements:	Accurate facial detection algorithms
Assumptions:	Clear and unobstructed facial images
Notes and Issues:	Accuracy crucial for recognition

Use Case ID:	AttendanceSystem-UC004		
Use Case Name:	Deblur images		
Created By:	Ananya	Last Updated By:	Ananya
Date Created:	20/3/2024	Date Last Updated:	21/3/2024

Actor:	AI Software
Description:	The system performs a deblurring procedure on captured images that exhibit blurriness.
Preconditions:	- Images with blurriness available - System initialized and running
Postconditions:	- Deblurred images ready for processing
Priority:	High
Frequency of Use:	As needed
Normal Course of Events:	1. System detects blurry images 2. Apply deblurring algorithm 3. Deblurred images prepared
Alternative Courses:	None
Exceptions:	- Deblurring failure
Includes:	None
Special Requirements:	Effective deblurring algorithm
Assumptions:	Some images may require deblurring
Notes and Issues:	Image quality affects recognition

Use Case ID:	AttendanceSystem-UC005		
Use Case Name:	Compare Facial Features		
Created By:	Mounika	Last Updated By:	Mounika
Date Created:	20/3/2024	Date Last Updated:	21/3/2024

Actor:	AI Software
Description:	System compares facial features of captured images with stored templates for attendance matching.
Preconditions:	- Facial features detected - Stored templates available
Postconditions:	- Matching results for attendance
Priority:	High
Frequency of Use:	Continuous
Normal Course of Events:	1. System retrieves stored templates 2. Compare facial features 3. Determine similarity for matching
Alternative Courses:	None
Exceptions:	- Comparison failure
Includes:	None
Special Requirements:	Effective facial recognition algorithm
Assumptions:	Templates are up to date
Notes and Issues:	Threshold for similarity is important

Use Case ID:	AttendanceSystem-UC006		
Use Case Name:	Mark Attendance		
Created By:	Haneesha	Last Updated By:	Haneesha
Date Created:	20/3/2024	Date Last Updated:	21/3/2024

Actor:	AI Software
Description:	System marks the attendance of recognized individuals based on comparison results.
Preconditions:	- Comparison results available - System initialized and running
Postconditions:	- Attendance recorded
Priority:	High
Frequency of Use:	Continuous
Normal Course of Events:	1. System receives comparison results 2. Determine if match surpasses threshold 3. Mark as present or absent accordingly
Alternative Courses:	None
Exceptions:	- Attendance marking error
Includes:	Store Attendance Record
Special Requirements:	Accurate comparison results
Assumptions:	Matching criteria are well defined
Notes and Issues:	Accuracy of matching is crucial

Use Case ID:	AttendanceSystem-UC007		
Use Case Name:	Store the Attendance Record		
Created By:	Mounika	Last Updated By:	Mounika
Date Created:	20/3/2024	Date Last Updated:	21/3/2024

Actor:	Database
Description:	System stores the attendance record including student ID, time, and other details in a secure database.
Preconditions:	- Attendance marked for individuals - System initialized and running
Postconditions:	- Attendance records stored securely
Priority:	High
Frequency of Use:	Continuous
Normal Course of Events:	1. System receives attendance details 2. Store in secure database
Alternative Courses:	None
Exceptions:	- Database storage error
Includes:	None
Special Requirements:	Secure database storage
Assumptions:	Access control for database
Notes and Issues:	Security is critical for attendance records

Use Case ID:	AttendanceSystem-UC008		
Use Case Name:	Store student data		
Created By:	Ananya	Last Updated By:	Ananya
Date Created:	20/3/2024	Date Last Updated:	21/3/2024

Actor:	Database
Description:	System stores student information and corresponding facial templates during registration.
Preconditions:	- Student enrolled and registered - System initialized and running
Postconditions:	- Student data and templates stored
Priority:	High
Frequency of Use:	Occasional
Normal Course of Events:	1. System prompts for student details 2. Capture facial data for enrollment 3. Store student information
Alternative Courses:	None
Exceptions:	- Data capture error - Database storage error
Includes:	None
Special Requirements:	Secure storage of student data
Assumptions:	Enrollment session has clear images
Notes and Issues:	Data accuracy and security are crucial

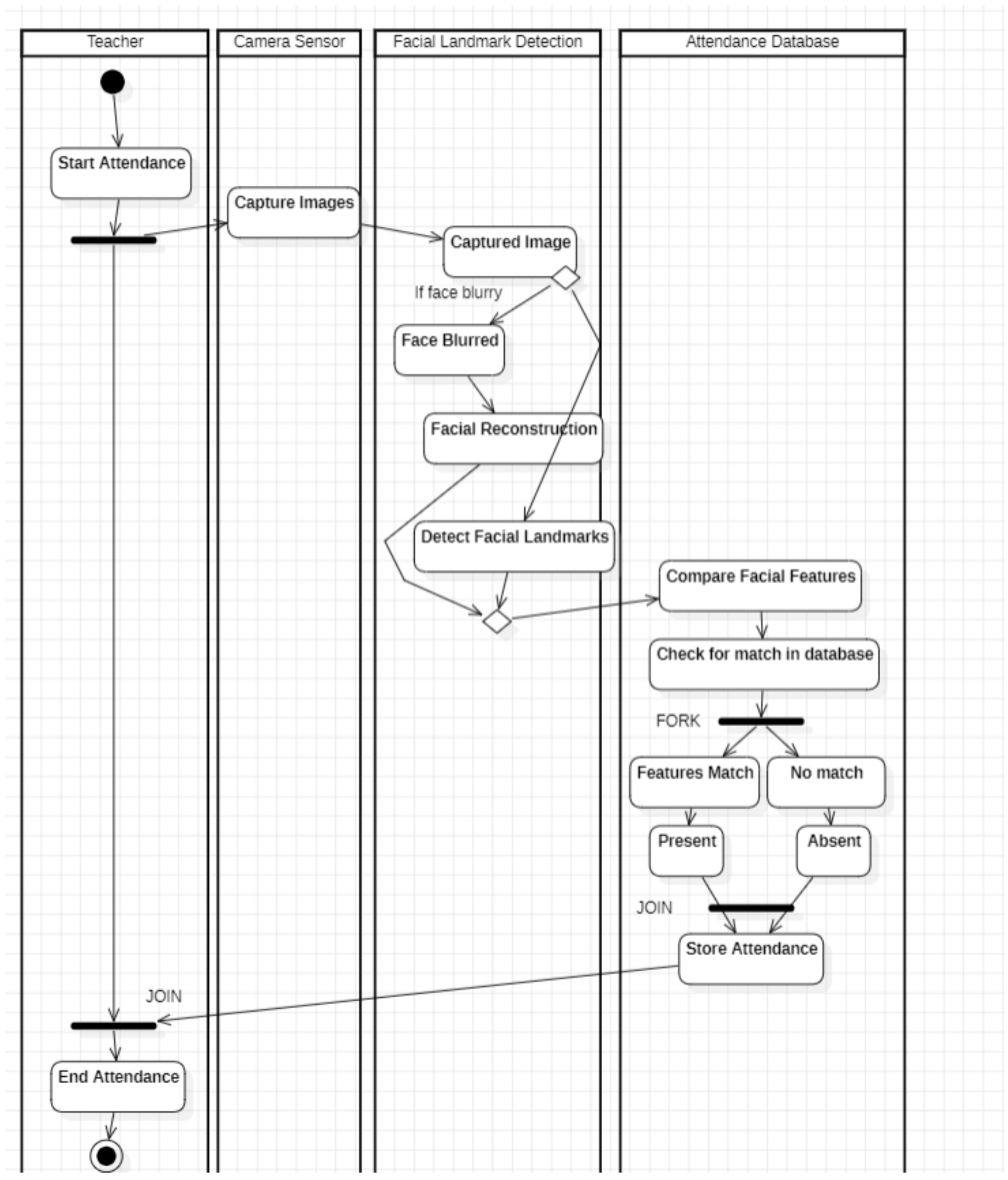
Use Case ID:	AttendanceSystem-UC008		
Use Case Name:	Store student data		
Created By:	Ananya	Last Updated By:	Ananya
Date Created:	20/3/2024	Date Last Updated:	21/3/2024

Actor:	Database
Description:	System stores student information and corresponding facial templates during registration.
Preconditions:	- Student enrolled and registered - System initialized and running
Postconditions:	- Student data and templates stored
Priority:	High
Frequency of Use:	Occasional
Normal Course of Events:	1. System prompts for student details 2. Capture facial data for enrollment 3. Store student information
Alternative Courses:	None
Exceptions:	- Data capture error - Database storage error
Includes:	None
Special Requirements:	Secure storage of student data
Assumptions:	Enrollment session has clear images
Notes and Issues:	Data accuracy and security are crucial

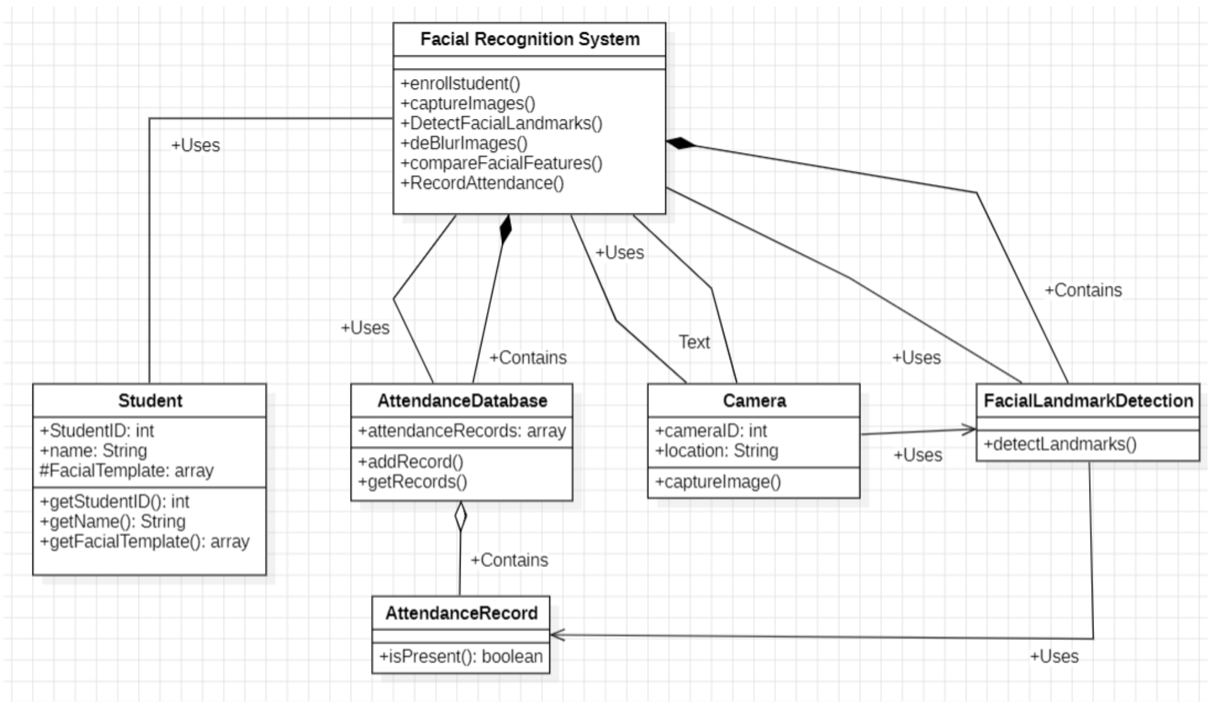
Use Case ID:	AttendanceSystem-UC009		
Use Case Name:	Register Student		
Created By:	Ananya	Last Updated By:	Ananya
Date Created:	20/3/2024	Date Last Updated:	21/3/2024

Actor:	Student
Description:	Administrator registers a student for the facial recognition system.
Preconditions:	- Administrator logged in - System initialized and running
Postconditions:	- Student enrolled and template stored
Priority:	High
Frequency of Use:	Occasional
Normal Course of Events:	1. Administrator selects "Register" 2. System prompts for student details 3. Capture facial data for enrollment
Alternative Courses:	None
Exceptions:	- Data capture error - Template generation error
Includes:	Store Student Data
Special Requirements:	Reliable facial data capture
Assumptions:	Students willingly participate
Notes and Issues:	Template quality is critical

Activity Diagram (Use the concept of swim-lane)



Class Diagram (Carefully associate the different classes using proper relationship and display the multiplicity values)



CRC card (for each class shown in the class diagram)

Class name: Facial Recognition System	Super Class: -	Sub Classes: Student, Attendance DB, Camera, Facial Landmark Detection
Responsibilties: ● Enroll Student ● Capture Image ● Detect Facial Landmarks ● Deblur image ● Compare Facial Features ● Record Attendance	Collaborations: ● Student ● Attendance DB ● Camera ● Facial Landmark Detection	

Class name: Student	Super Class: Facial Recognition System	Sub Classes: -
Responsibilties: • Getting Student ID • Getting student name • associating these details with their face template	Collaborations: • Facial Recognition System	

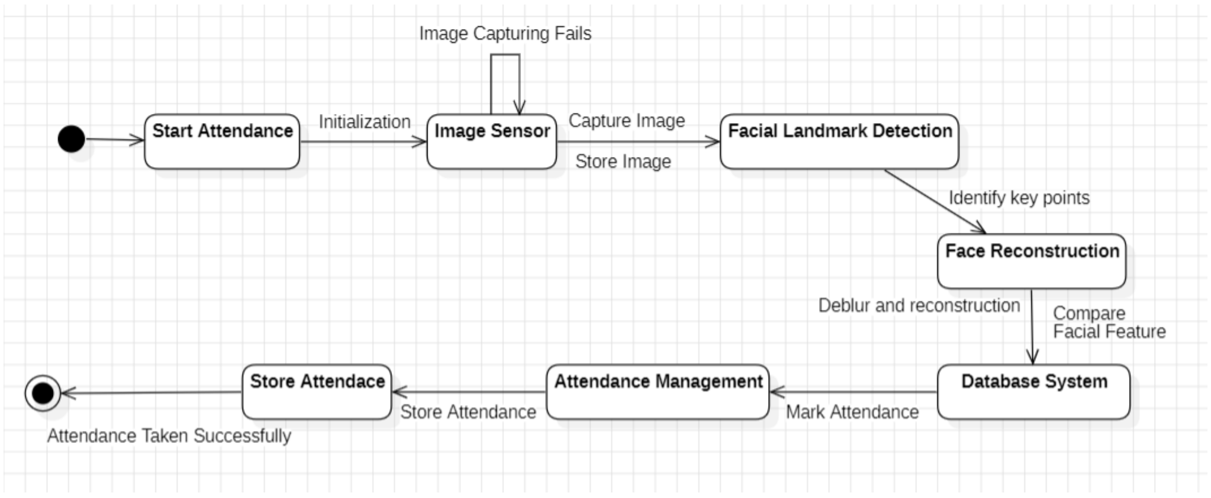
Class name: Attendance Database	Super Class: Facial Recognition System	Sub Classes: Attendance Record
Responsibilties: • add record • get record	Collaborations: • Facial Recognition System • Attendance Record	

Class name: Camera	Super Class: Facial Recognition System	Sub Classes: -
Responsibilties: • capture the image • send to facial landmark detection	Collaborations: • Facial Recognition System • Facial Landmark Detection	

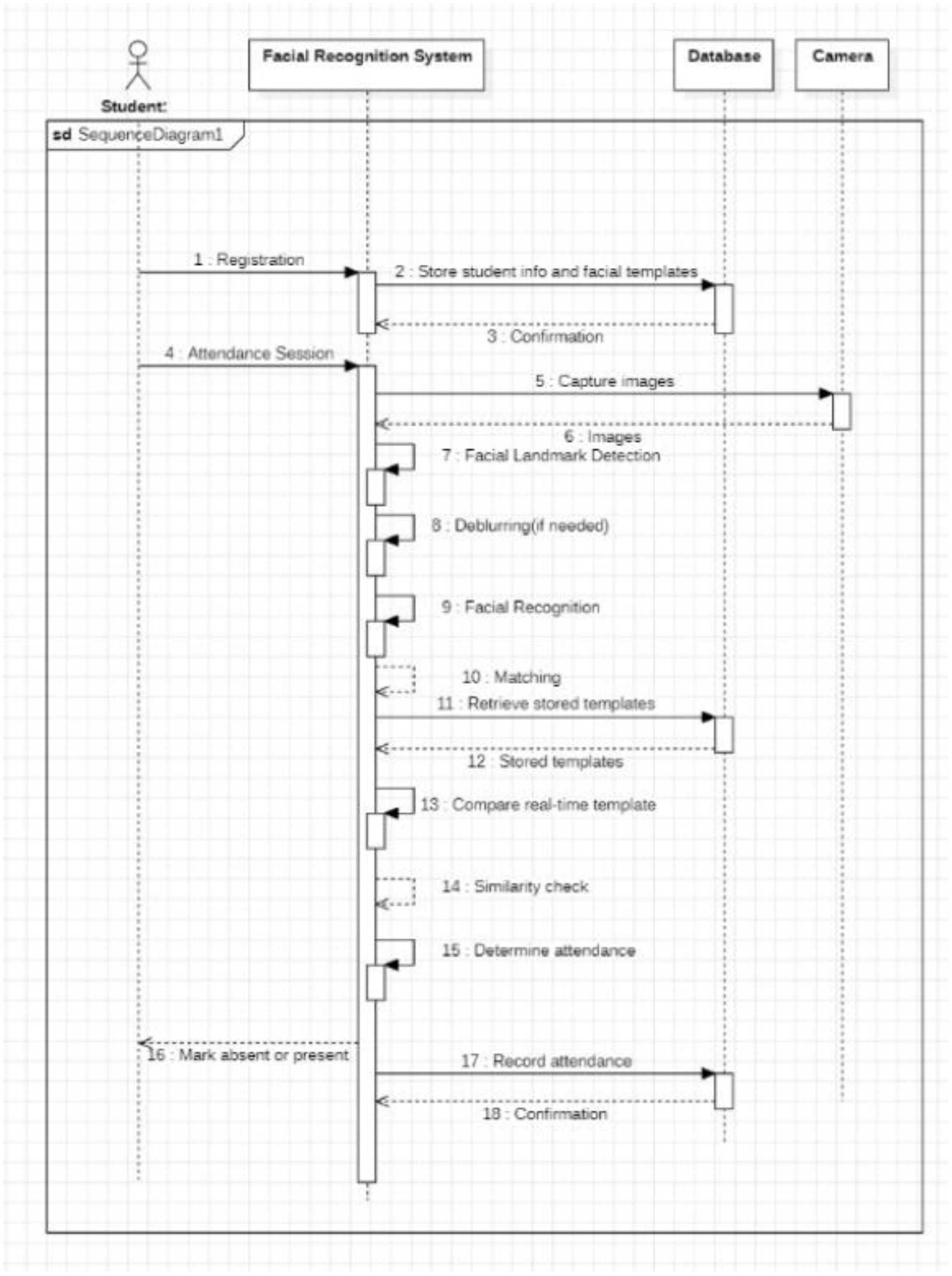
Class name: Attendance Record	Super Class: Attendance Database	Sub Classes: -
Responsibilties: • Record the Data	Collaborations: • Attendance Database • Facial Landmark Detection	

Class name: Facial Landmark Detection	Super Class: Facial Recognition System	Sub Classes: -
Responsibilties: • Detect Landmark • send signal if attendance is marked	Collaborations: • Facial Recognition System • Attendance record	

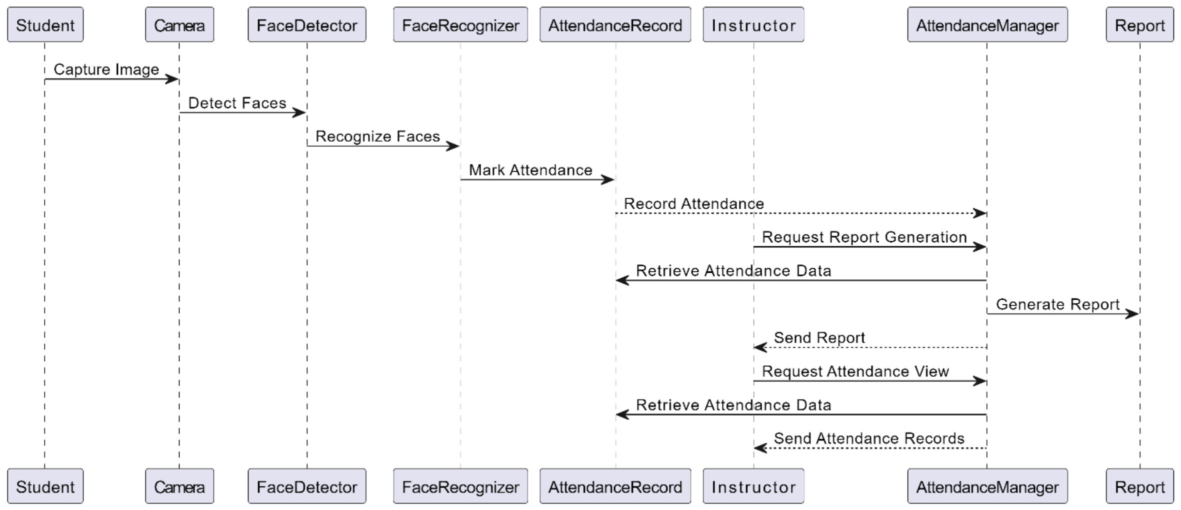
State Chart Diagram (Displaying the all the states and transition among states by firing an event)



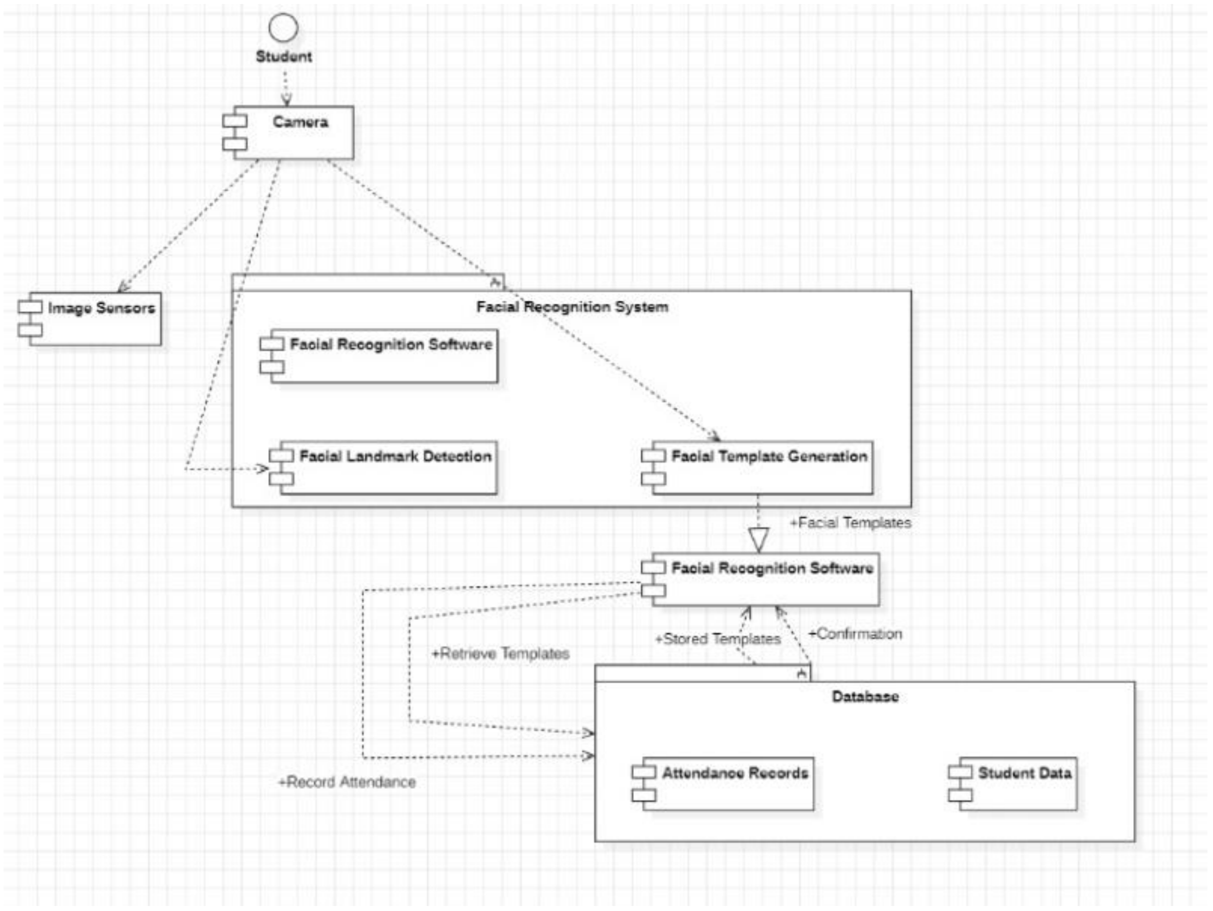
Sequence Diagram (Display the timelines (lifelines) of each objects properly)



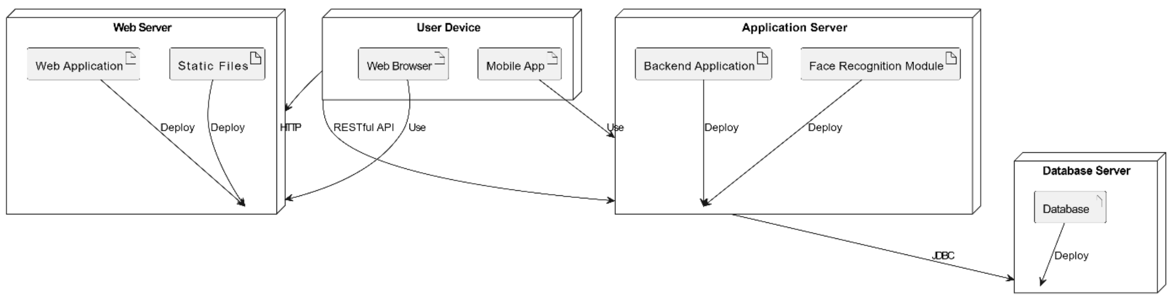
Collaboration / Communication Diagram



Component Diagram (Join the components through proper interfaces)



Deployment Diagram (Display the relevant Nodes and the interconnection among Nodes as well as artifacts)



Generate the code as per your developed language. (If for your language, the plugins are not available, then generate the code in JAVA). Display your output as File Name and File Content for each file generated.

Attendance Database:

```
import java.util.*;


/**
 *
 */
public class AttendanceDatabase {


    /**
     * Default constructor
     */
    public AttendanceDatabase() {
    }


    /**
     *
     */
    public array attendanceRecords;


    /**
     *
     */
    public void addRecord() {
        // TODO implement here
    }

    /**
     *
     */
    public void getRecords() {
        // TODO implement here
    }

}
```

Attendance Record:

```
import java.util.*;

/**
 *
 */
public class AttendanceRecord {

    /**
     * Default constructor
     */
    public AttendanceRecord() {
    }

    /**
     * @return
     */
    public boolean isPresent() {
        // TODO implement here
        return false;
    }

}
```

Camera:

```
import java.util.*;

/**
 *
 */
public class Camera {

    /**
     * Default constructor
     */
    public Camera() {
    }

    /**
     *
     */
    public int cameraID;

    /**
     *
     */
    public String location;
    /**
     *
     */
    public void captureImage() {
        // TODO implement here
    }

}
```

Facial Recognition System:

```
import java.util.*;

/**
 *
 */
public class Facial Recognition System {

    /**
     * Default constructor
     */
    public Facial Recognition System() {
    }

    /**
     *
     */
    public void enrollstudent() {
        // TODO implement here
    }
```

```
/**
 *
 */
public void captureImages() {
   // TODO implement here
}

/**
 *
 */
public void DetectFacialLandmarks() {
   // TODO implement here
}

/**
 *
 */
public void deBlurImages() {
   // TODO implement here
}

/**
 *
 */
public void compareFacialFeatures() {
   // TODO implement here
}

/**
```

```
     *
     */
    public void RecordAttendance() {
        // TODO implement here
    }

}
```

Facial Landmark Detection:

```
import java.util.*;

/**
 *
 */
public class FacialLandmarkDetection {

    /**
     * Default constructor
     */
    public FacialLandmarkDetection() {
    }

    /**
     *
     */
    public void detectLandmarks() {
        // TODO implement here
    }

}
```

Student:

```
import java.util.*;

/**
 *
 */
public class Student {

    /**
     * Default constructor
     */
    public Student() {
    }

    /**
     *
     */
    public int StudentID;

    /**
     *
     */
    public String name;

    /**
     *
     */
    protected array FacialTemplate;
    /**
     * @return
     */
    public int getStudentID() {
        // TODO implement here
        return 0;
    }

    /**
     * @return
     */
    public String getName() {
        // TODO implement here
        return "";
    }

    /**
     * @return
     */
    public array getFacialTemplate() {
        // TODO implement here
        return null;
    }

}
```