**MAY 2024**

**UNIVERSITY OF BUEA**

**FACULTY OF ENGINEERING AND TECHNOLOGY (FET)**

**DEPARTMENT OF COMPUTER ENGINEERING**

**COURSE: INTERNET PROGRAMMING AND MOBILE PROGRAMMING (CEF440)**

**REPUBLIQUE DU CAMEROUN**

***PAIX – TRAVAIL – PATRIE***

**MINISTERE DE L’ENSEIGNEMENT SUPERIEUR**

**REPUBLIC OF CAMEROON**

***PEACE – WORK – FATHERLAND***

**MINISTRY OF HIGHER EDUCATION**



**PRESENTED BY : GROUP 19**

|  |  |  |
| --- | --- | --- |
| **NAME** | **MATRICULE** | **SPECIALTY** |
| NKEMCHOU PIANKE OLIVIER | FE21A275 | SE |
| DJITUE TOGUE BRINDA SPACHELLE | FE21A171 | SE |
| TEGUE NONO MIKEL MODEIRO | FE21A321 | SE |
| REOUTADE ROLAND | FE21A301 | SE |
| AMARACHUKWU GODLOVE AHANONU | FE21A136 | SE |

TASK4: SYSTEM MODELLING AND DESIGN

**INSTRUCTOR : Dr. NKEMENI VALERY**

# 

Table of content

[1. INTRODUCTION 2](#_Toc167724728)

[1.1 Definition 2](#_Toc167724729)

[1.2 How the system functions 2](#_Toc167724730)

[2. CONTEXT DIAGRAM 3](#_Toc167724731)

[2.1 Key Elements of a Context Diagram 3](#_Toc167724732)

[2.2 Purpose of a context diagram 3](#_Toc167724733)

[2.3 Biometrics student’s attendance app context diagram 3](#_Toc167724734)

[3. USE CASE DIAGRAM 5](#_Toc167724735)

[3.1 Use Case1: Administrator Panel 5](#_Toc167724736)

[3.2 Use Case 2: Instructor Panel 8](#_Toc167724737)

[3.3 Use Case 3: Student Interactions 10](#_Toc167724738)

[4. SEQUENCE DIAGRAM 12](#_Toc167724739)

[4.1 Components of a Sequence Diagram 12](#_Toc167724740)

[4.2 Description of the Sequence Diagram 12](#_Toc167724741)

[4.2.1 Scenario: User Registration with Biometric Authentication and 2FA 12](#_Toc167724742)

[4.2.2 Scenario: Attendance Tracking 16](#_Toc167724743)

[4.2.3 Scenario: Instructor Login to Course 22](#_Toc167724744)

[5. CLASS DIAGRAM 26](#_Toc167724745)

[5.1 Components of a Class Diagram 26](#_Toc167724746)

[5.2 Biometrics student’s attendance app class diagram 27](#_Toc167724747)

[5.2.1 Key Entities (Classes) And Relationships 28](#_Toc167724748)

[6. DEPLOYMENT DIAGRAM 31](#_Toc167724749)

[6.1 Key Elements of a Deployment Diagram 31](#_Toc167724750)

[6.2 Purpose of Deployment Diagrams 31](#_Toc167724751)

[6.3 Biometric student’s attendance app Deployment diagram 32](#_Toc167724752)

[7.0 CONCLUSION 35](#_Toc167724753)

# INTRODUCTION

## Definition

System modeling is the process of creating abstract representations of a system to better understand, analyze, and communicate its structure, behavior, and functionality. These models help in visualizing and specifying the components and interactions within a system, facilitating the identification and resolution of issues early in the development process. System modeling is a crucial step in systems engineering, software development, and other technical fields.

## 1.2 How the system functions

**Phase 1:** User registration

* For students: At the beginning of each academic year, the student’s information are recorded during the submission of documents to the faculty. This information include; name, matricule number, department, email and biometric information.
* For instructors: At the launch of the system the information of each instructor is recorded, and each time the faculty recruits a new instructor his information is recorded into the system. This information include; name, email and biometric information.
* Course registration: At the beginning of each academic year the administrator creates the course for that academic year and attributes their respective lecturers.

**Phase 2:** Attendance tracking

* Instructor logs in course: Instructors enters the class and picks the mobile device. Then selects his course, after selecting the course the system prompts his fingerprint and checks if they match.
* Once the instructor logged in, he can start the attendance process by clicking the button for that purpose.
* The device goes round, each student enters his matricule and waits for the system to verify the validity of the matricule. Once the matricule verified, the system prompts his fingerprint and check the match. This process is repeated for the number of student.
* Once all the students’ attendance recorded, the system prompts the instructor’s fingerprint to validate the list and the list is stored.

# CONTEXT DIAGRAM

A system context diagram, also known as a high-level diagram, provides a top-level view of a system and its external environment. It focuses on the system as a single process or entity surrounded by external entities such as users, systems, or other stakeholders. This diagram deliberately simplifies the complexity of the system by highlighting only the most important interactions and data flows between the system and its external elements. As a high-level representation, it aims to give stakeholders a quick and clear understanding of the system's context without going into detailed technical specifications. This helps in defining the system's scope, boundaries, and key relationships at a glance.

## 2.1 Key Elements of a Context Diagram

**Process:** This is the central element and the most crucial element of the context diagram which represents the overall system, it is represented by a circle.

**External entities:** These are people, organizations, other systems, or external data sources that interact with the system. It is represented by rectangles.

**Data flow:** These are arrows connecting the central process and the external entities. These arrows are used to show the movement of data between the system and external entities.

## 2.2 Purpose of a context diagram

* Defining system boundaries: Clearly delineates what is inside the system and what is outside, helping to define the system's boundaries
* Identify external interactions: Identify all external entities that interact with the system and also the way they interact.
* Simplify complex systems: Provides a high-level view of the system, abstracting away internal complexities and focusing on external interactions

## 2.3 Biometrics student’s attendance app context diagram

**Description:** The biometric student attendance mobile application is the system we want to build so it will be the main process and interacts mainly with users (students, instructors and administrator) and data flows between the main system and these users.

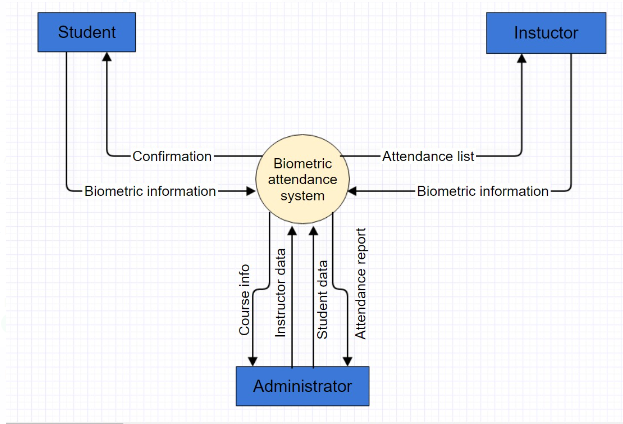
For our biometric attendance system we came out with the following entities, process and dataflow.

**Process:** The process is simply the system represented by the central circle of the diagram

**External Entities:** Our system mainly interact with the students, instructors and administrator which represent the external entities of the system

**Dataflow:**

* **With respect to student:** The student supplies the system with his biometric information and receives confirmation during attendance marking
* **With respect to instructor:** Instructor supplies his biometric information to the system and can receive the attendance list.
* **With respect to the Administrator:** The administrator enters the student and instructor’s data into the system and can receive the attendance report and also the course information.

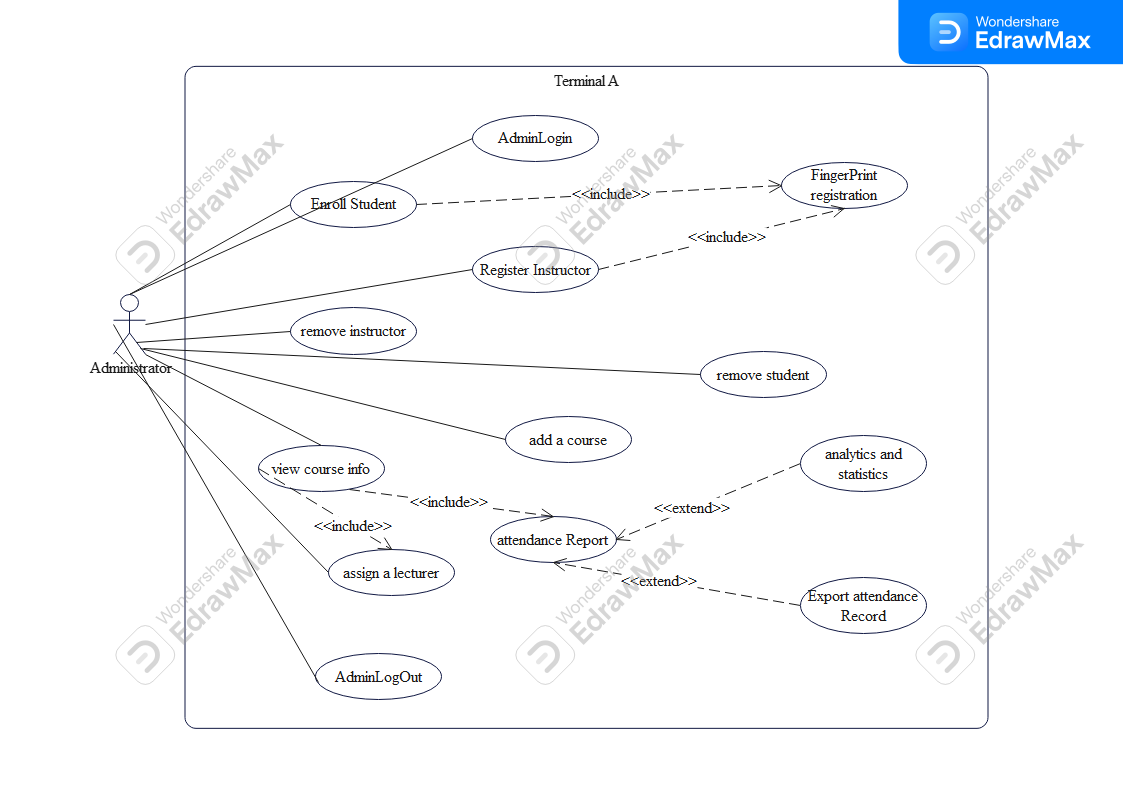


# USE CASE DIAGRAM

A use case diagram is a visual representation of the interactions between users (actors) and a system, illustrating how different users engage with the system to achieve various goals. It helps in capturing the functional requirements of a system and provides a clear understanding of the system’s behavior from an end-user perspective. The key components of a use case diagram include actors, use cases, and the relationships between them.

* Actors: Entities that interact with the system (can be users or other systems).
* Use Cases: Specific functions or activities that the system performs in response to actor inputs.
* Relationships: The connections between actors and use cases, showing how actors participate in use cases.
* System Boundary: A rectangle that encapsulates all use cases, defining the scope of the system

## **3.1 Use Case1: Administrator Panel**

****

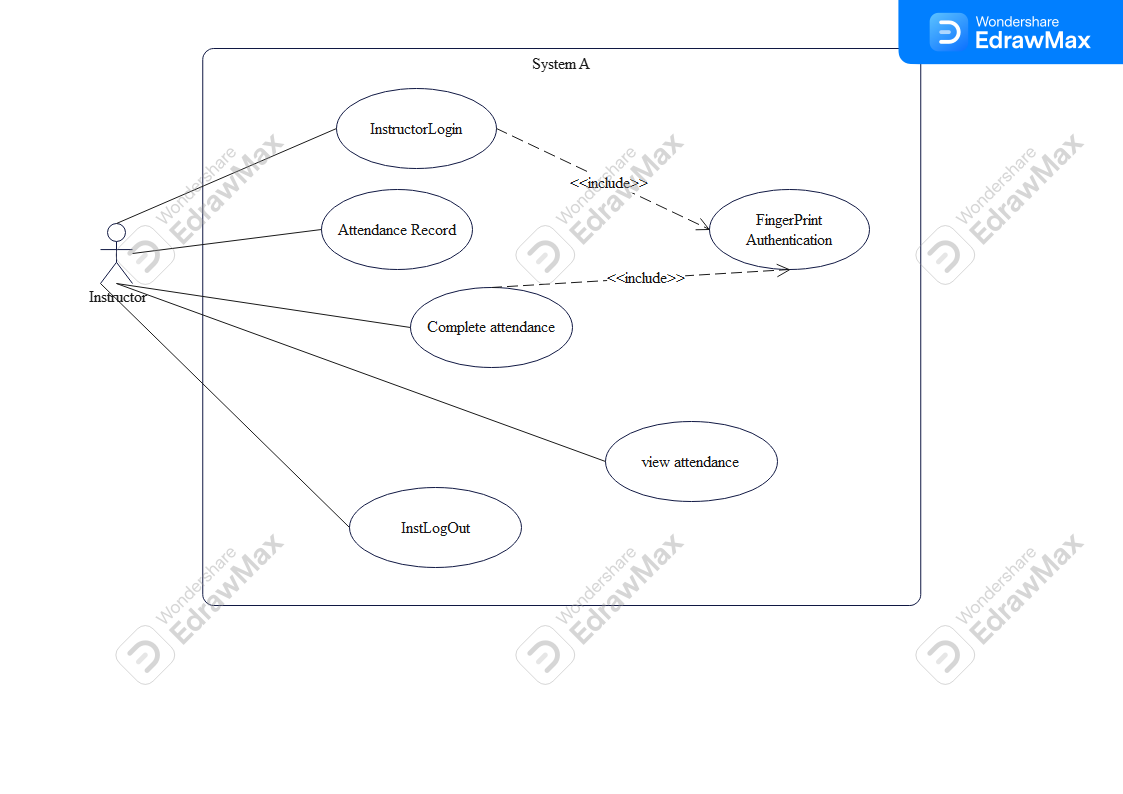
**Actors:**

Administrator: The primary user responsible for managing the system’s operations.

**Use Cases and Scenarios:**

1. AdminLogin
   * **Description**: The administrator logs into the system.
   * **Scenario**: The administrator enters credentials to access the system’s functionalities.
2. FingerPrint registration
   * **Description**: Registers the fingerprint of users (students or instructors) into the system.
   * **Scenario**: After logging in, the administrator uses this feature to capture and store fingerprints for authentication purposes.
   * **Relation**: Included in both **Enroll Student** and **Register Instructor** use cases as a necessary step.
3. Enroll Student
   * **Description**: Enrolls a new student into the system.
   * **Scenario**: The administrator inputs student details and captures their fingerprint for attendance purposes.
   * **Relation**: Includes **Fingerprint registration** as part of the enrollment process.
4. Register Instructor
   * **Description**: Registers a new instructor in the system.
   * **Scenario**: The administrator inputs instructor details and captures their fingerprint for system access.
   * **Relation**: Includes **Fingerprint registration** as part of the registration process.
5. Remove Instructor
   * **Description**: Removes an instructor from the system.
   * **Scenario**: The administrator selects an instructor and deletes their records from the system.
6. Remove Student
   * **Description**: Removes a student from the system.
   * **Scenario**: The administrator selects a student and deletes their records from the system.
7. Add a Course
   * **Description**: Adds a new course to the system.
   * **Scenario**: The administrator inputs course details such as course name, description, and schedule.
8. View Course Info
   * **Description**: Views details of the courses available in the system.
   * **Scenario**: The administrator can search and view information about the courses.
9. Assign a Lecturer
   * **Description**: Assigns a lecturer to a course.
   * **Scenario**: The administrator selects a course and assigns an instructor to it.
   * **Relation**: Included in the **View Course Info** use case as part of course management.
10. Attendance Report
    * **Description**: Generates reports on student attendance.
    * **Scenario**: The administrator can generate and view attendance reports for different courses and time periods.
    * **Relation**: Extends **View Course Info** and is necessary for generating detailed attendance reports.
11. Export Attendance Record
    * **Description**: Exports attendance records to a file or external system.
    * **Scenario**: The administrator exports attendance data for analysis or record-keeping.
    * **Relation**: Extends **Attendance Report** use case for data export functionality.
12. Analytics and Statistics
    * **Description**: Provides analytics and statistics on attendance data.
    * **Scenario**: The administrator views various metrics and statistics derived from attendance records.
    * **Relation**: Extends **Attendance Report** to provide detailed insights and analytics.
13. AdminLogOut
    * **Description**: Logs out the administrator from the system.
    * **Scenario**: The administrator ends the session and logs out of the system securely.

## **3.2 Use Case 2: Instructor Panel**

****

**Textual Description**

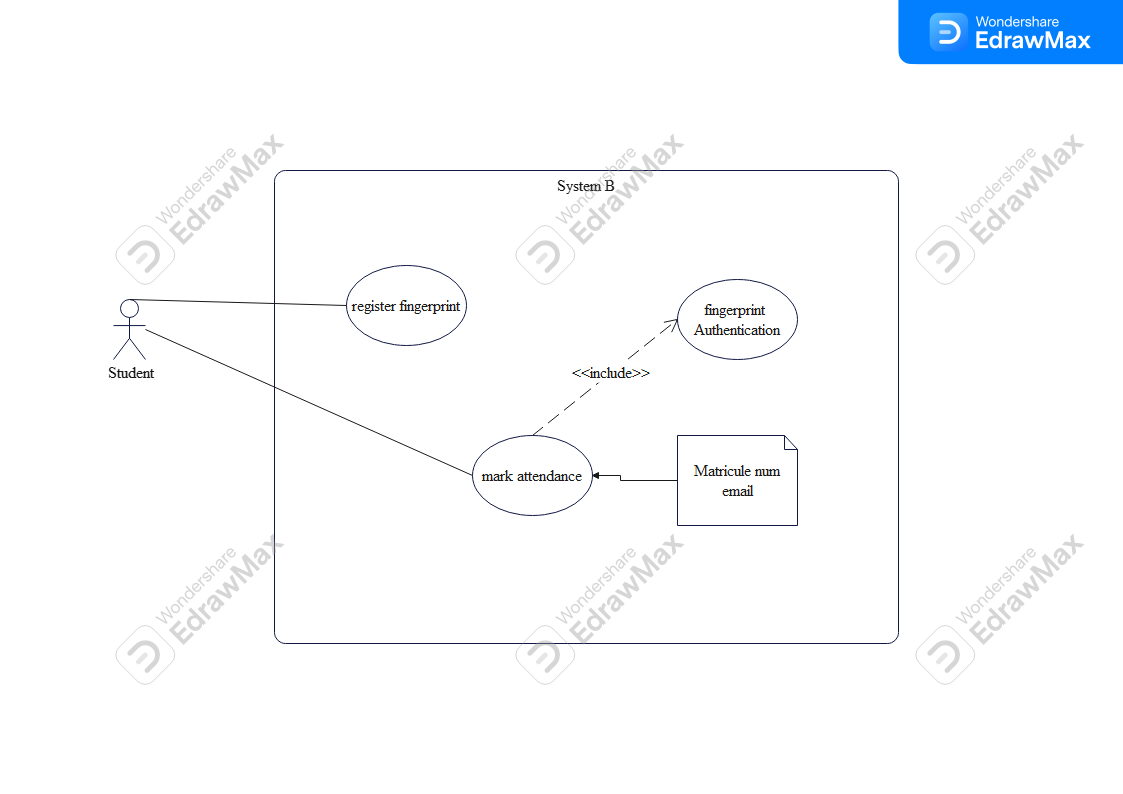
**Actors:**

Instructor: This actor represents the user (instructor) who interacts with the system to manage attendance records and view attendance data.

**Use Cases:**

1. InstructorLogin:
   * **Description**: This use case allows the instructor to log into the system.
   * **Scenario**: The instructor enters their credentials to access the system’s functionalities. The login process includes fingerprint authentication.
2. Complete Attendance:
   * **Description**: This use case allows the instructor to terminate the attendance tracking process, thereby confirming this action using fingerprint data ensuring integrity of attendance record
   * **Scenario**: The instructor selects the complete option and validate the completion of attendance tracking for the given course session by using fingerprint authentication, ensuring that he is the only one having the permission to validate and approve the attendance recorded
3. View Attendance:
   * **Description**: This use case allows the instructor to view the attendance records of students.
   * **Scenario**: The instructor selects a course to view detailed attendance records, including dates, statuses (present, absent), course title and code, session, and attendance time-stand for each student along with their names and matricules
4. Attendance Record:
   * **Description**: This use case involves maintaining and accessing the attendance records.
   * **Scenario**: The system automatically updates the attendance records as the instructor marks attendance. The records can be accessed for viewing and reporting purposes.
5. Fingerprint Authentication:
   * **Description**: This use case is included in the login and attendance marking processes to verify the identity of the instructor.
   * **Scenario**
     + The instructor places their finger on the biometric scanner, and the system verifies the fingerprint before granting access to further functionalities.
     + A students place their finger on the biometric scanner, and the system verifies the fingerprint before updating the attendance status of the student to present
6. InstLogOut:
   * **Description**: This use case allows the instructor to log out of the system.
   * **Scenario**: The instructor clicks the logout button to securely end the session and exit the system.

## **3.3 Use Case 3: Student Interactions**



**Detailed Description of the Use Case Diagram**

**Actors:**

Student: This actor represents the student who interacts with the system to register their fingerprint and mark attendance.

**Use Cases:**

1. Register Fingerprint:
   * **Description**: This use case allows a student to register their fingerprint with the system.
   * **Scenario**: The student provides their fingerprint, which is scanned and stored in the system for future authentication. This step is necessary for the student to be able to mark their attendance using the fingerprint in subsequent steps.
2. Fingerprint Authentication:
   * **Description**: This use case is included in the processes where the student needs to authenticate their identity using their fingerprint.
   * **Scenario**: Whenever the student needs to mark their attendance, they must authenticate their identity by placing their finger on a biometric scanner. The system verifies the fingerprint against the stored fingerprint data.
3. Mark Attendance:
   * **Description**: This use case allows the student to mark their attendance using their fingerprint.
   * **Scenario**: The student goes to the attendance terminal and authenticates their identity using their fingerprint. Once authenticated, the system records the attendance along with the student’s matriculation number and email for verification and record-keeping purposes.

System Boundary:

The system boundary encapsulates all the functionalities (use cases) managed by the Biometric Student Attendance Mobile Application. The actor (Student) interacts with the system through various use cases to register their fingerprint and mark their attendance securely and efficiently.

# SEQUENCE DIAGRAM

A sequence diagram is one of the types of interaction diagrams in UML (Unified Modeling Language). It shows how objects interact in a given situation. Sequence diagrams are typically used to model the interactions between components in a system, focusing on the sequence of messages exchanged between these components over time.

## 4.1 Components of a Sequence Diagram

1. Actors: Represent entities outside the system that interact with the system (e.g., users or external systems). Depicted as stick figures.
2. Objects: Instances of classes or other elements that participate in the interaction. Represented by rectangles.
3. Lifelines: Vertical dashed lines that indicate the life span of an object during the interaction.
4. Messages: Horizontal arrows between lifelines representing communication between objects (method calls, returns, etc.).
5. Activation Bars: Narrow rectangles on lifelines showing when an object is active and performing an action.
6. System Boundary: Denotes the scope of the sequence diagram, showing which components are inside the system and which are external.

## 4.2 Description of the Sequence Diagram

## 4.2.1 Scenario: User Registration with Biometric Authentication and 2FA

This sequence diagram models the interaction involved in registering a user (either an instructor or a student) in a biometric student attendance mobile application, including the use of two-factor authentication (2FA) for instructors.

Components

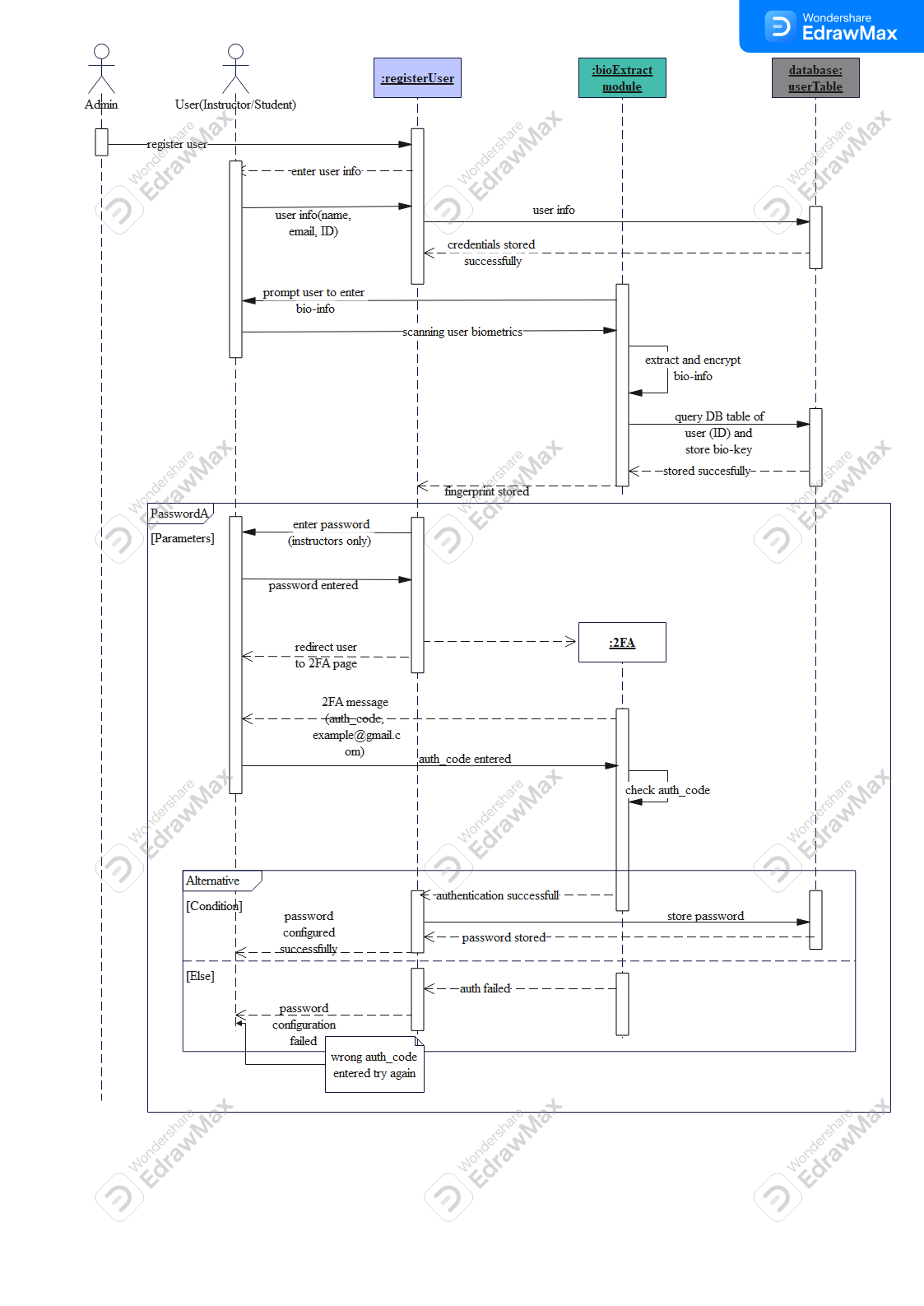
1. **Actors**:
   * Admin: The administrator who initiates the registration process.
   * User (Instructor/Student): The individual being registered.
2. **Objects**:
   * **: registerUser**: The component responsible for handling the registration process.
   * **: Bio Extract Module**: The component responsible for extracting and encrypting biometric information.
   * **database: userTable**: The database table where user information and biometric keys are stored.
   * **PasswordA**: The module handling password entry for instructors.
   * **2FA**: The module responsible for handling two-factor authentication.

Message Flow and Interactions

1. **Admin** interacts with **: registerUser**
   * **Message**: **register user**
   * **Description**: The admin initiates the user registration process.
2. **: registerUser** sends a message to User (Instructor/ Student)
   * **Message**: **enter user info**
   * **Description**: The user is prompt to provides his/her basic information (name, email, ID).
3. **stores the user information:**
   * **Message**: user info (name, email, ID)
   * **Response**: credentials stored successfully
   * **Description**: The system acknowledges that the user's credentials have been successfully stored.
4. **: bioExtractModule** prompts **User (Instructor/Student)** to provide biometric information:
   * **Message**: prompt user to enter bio-info
   * **Description**: The system requests the user to enter their biometric data.
5. **User (Instructor/Student)** interacts with **: bioExtractmodule**
   * **Message**: **scanning user biometrics**
   * **Description**: The user provides their biometric information (e.g., fingerprint).
6. **: bioExtractmodule** processes the biometric data:
   * **Message**: extract and encrypt bio-info
   * **Description**: The biometric information is extracted and encrypted.
7. **: bioExtractmodule** interacts with **database: userTable**:
   * **Message**: query DB table of user (ID) and store bio-key
   * **Response**: stored successfully
   * **Description**: The encrypted biometric key is stored in the database associated with the user’s ID.
8. **: registerUser** prompts **User (Instructor)** to enter a password (instructors only):
   * **Message**: enter password (instructors only)
   * **Description**: The system requests instructors to provide a password as an alternative authentication method.
9. **User (Instructor/Student)** enters the password:
   * **Message**: password entered
   * **Response**: **redirect user to 2FA page**
   * **Description**: The system redirects the user to the 2FA page.
10. **2FA** sends a message to **User (Instructor)**:
    * **Message**: 2FA message (auth\_code\_example@gmail.com)
    * **Description**: The 2FA system sends a two-factor authentication code to the user's email.
11. **User (Instructor/Student)** enters the authentication code:
    * **Message:** Auth code entered
    * **Description**: The user enters the authentication code received via email.
12. **2FA** performs a checking of the code sent against the code provided by the user ensuring integrity
    * **Message**: **check auth\_code**
    * **Description**: The 2FA system checks the entered authentication code.

Alternatives

* **Alternative 1**: Successful Authentication
  + **Condition**: authentication successful
  + **Response**: **password stored**
  + **Description**: If the authentication code is correct, the password is stored successfully.
* **Alternative 2**: Authentication Failed
  + **Condition**: auth failed
  + **Message**: wrong auth\_code entered try again
  + **Description**: If the authentication code is incorrect, the user is prompted to try again.



## 4.2.2 Scenario: Attendance Tracking

It consists of two main scenarios related to attendance tracking using biometric authentication. The participants include an **Instructor**, **Student**, and several system components such as the **attendance Tracking** system, **student table** database, and **attendance report** module. Additionally, it involves a **complete Attendance** process and **Auth\_module**

Components of the Sequence Diagram

1. **Actors:**
   * Instructor: The person who initiates the process to track attendance.
   * Student: The individual whose attendance is being tracked.
2. **Objects:**
   * **: attendanceTracking**: The component responsible for handling the attendance tracking process, and responsible for authenticating biometric information.
   * **student\_table: dataBase**: The database table where student information and biometric keys are stored.
   * **: AttendanceReport**: The module where attendance records are stored.
   * **: completeAttendance**: The component responsible for completing the attendance process.
   * **Authentication:auth\_module**: component responsible for authenticating biometric information of instructors to complete attendance
   * **inst\_table: dataBase**: The database table where instructor information and biometric keys are stored.
   * **PasswordA**: The module handling password entry for instructors.
   * **2FA**: The module responsible for handling two-factor authentication.

Message Flow and Interactions

First Scenario: Tracking Attendance

1. **Instructor interacts with : attendanceTracking**
   * **Message:** track attendance
   * **Description:** The instructor initiates the process to track attendance.
2. **: attendanceTracking sends a message to Student**
   * **Message:** prompt student to input matricule
   * **Description:** The system requests the student to input their matricule number.
3. **Student interacts with : attendanceTracking**
   * **Message:** student inputs matricule
   * **Description:** The student provides their matricule number and the module uses this data to query the database and check if the student record exists
   * **Return message:** return the digital key of the student to be used in authentication process later
4. **: attendanceTracking prompts Student to provide biometric information**
   * **Message:** Enter Biometric Data
   * **Description:** The system requests the student to enter their biometric data.
5. **Student interacts with : attendanceTracking**
   * **Message:** biometric input
   * **Description:** The student provides their biometric information.
6. **The module : attendanceTracking processes the biometric data**
   * **Message:** biometric input
   * **Description:** The system receives and processes the biometric input, extracting the biometric details, and encrypt it to obtain the participant’s digital key in order to perform a match against the stored data on registration
7. **: attendanceTracking matches the biometric data**
   * **Message:** Extract biometric details, encrypt to obtain digital key, and match with ***S\_key***
   * **Description:** The system extracts the biometric details, encrypts them to obtain a digital key, and matches it with the ***S\_key*** from the database.

**Alternatives**

* **Alternative 1**: Successful Authentication
  + **Condition**: authentication successful
* **update the attendance record in : AttendanceReport**
  + **Message:** Student is marked present
  + **Description:** If authentication is successful, the student is marked present in the attendance report.
* **Alternative 2**: Authentication Failed
  + **Condition:** authentication failed
  + **Message:** check matricule number and try again
  + **Description:** If authentication fails, the system prompts the student to check the matricule number and try again.

Second Scenario: Completing Attendance with Authentication

* **Instructor interacts with : completeAttendance component**
  + **Message:** complete attendance
  + **Description:** The instructor initiates the process to complete attendance.
* **: completeAttendance component sends a message to Instructor**
  + **Message:** Request Bio-input
  + **Description:** The system requests the instructor to provide their biometric data.
* **Instructor interacts with : completeAttendance**
  + **Message:** Biometric inputs
  + **Description:** The instructor provides their biometric information.
* **: completeAttendance processes the biometric data**
  + **Message:** Biometric details extracted
  + **Description:** The system extracts the biometric details.
* **: completeAttendance interacts with Authentication: Auth\_module**
  + **Message:** Biometric details extracted
  + **Description:** The system sends the extracted biometric details to the authentication module.
* **Authentication: Auth\_module interacts with inst\_table: dataBase**
  + **Message:** Request Key Stored
  + **Response:** return Key
  + **Description:** The authentication module requests the stored key from the instructor database, and the database processes the request and returns the stored key.
* **Authentication: Auth\_module performs matching of biometric data**
  + **Message:** Perform matching
  + **Description:** The authentication module performs matching of the biometric details with the stored key.

Alternative 1

* + **Condition:** Auth Successful
  + **Response:** Attendance completed
  + **Description:** If authentication is successful, the attendance process is completed.

Alternative 2

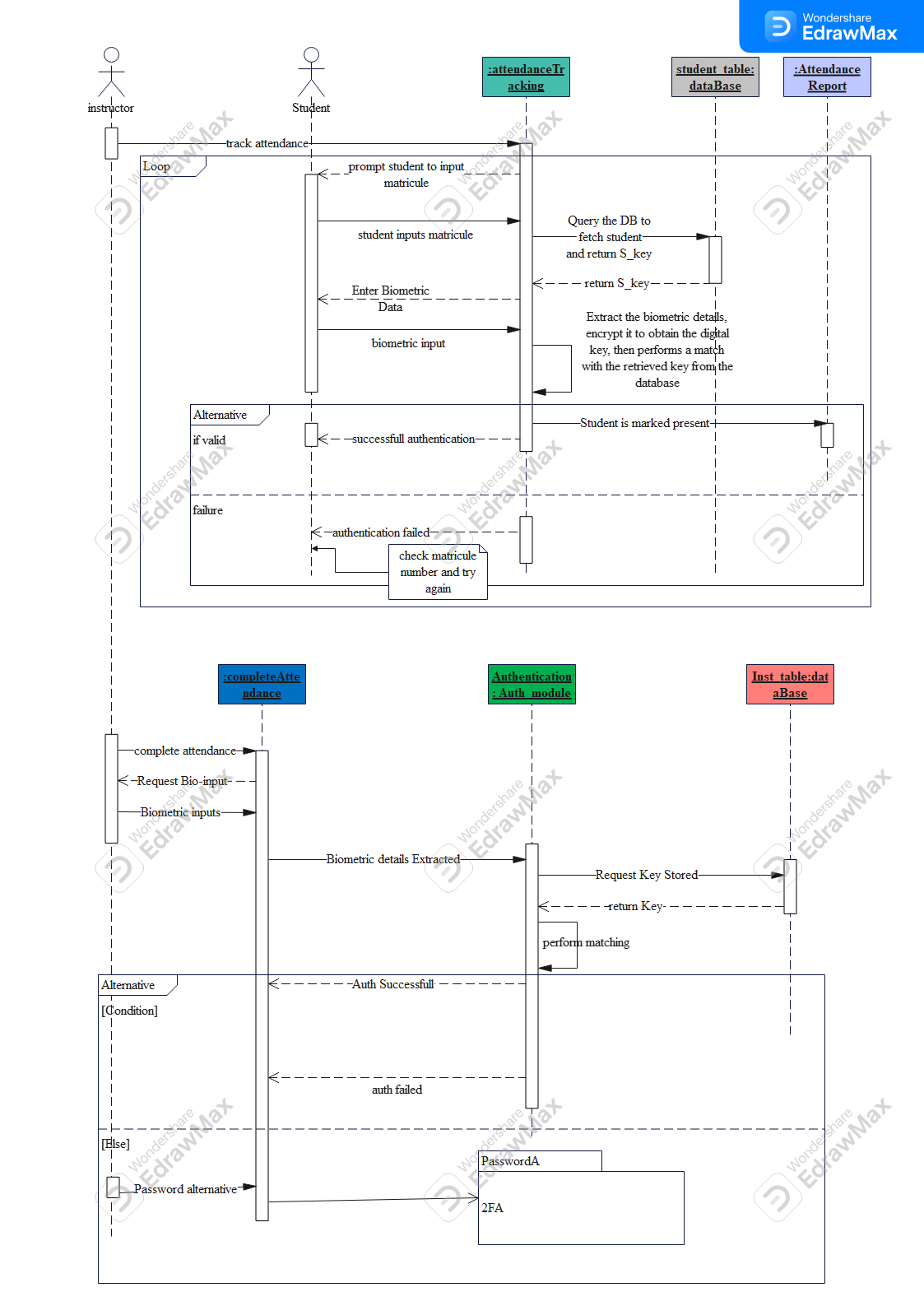
* + **Condition:** Auth failed
  + **Message:** Password alternative or 2FA
  + **Description:** If authentication fails, the system prompts for an alternative password or two-factor authentication (2FA).
* **PasswordA prompts instructor to enter a password**
  + **Message:** Enter Password
  + **Description:** The system requests the instructor to provide a password.
* **Instructor enters the password**
  + **Message:** Password entered
  + **Response:** Redirect user to 2FA page
  + **Description:** The system stores the entered password performs a matching of the provided password against the stored password during registration, in case of match and redirects the instructor to the 2FA page.
* **2FA sends a message to Instructor**
  + **Message:** 2FA message (auth\_code@example.com)
  + **Description:** The 2FA system sends a two-factor authentication code to the instructor's email.
* **Instructor enters the authentication code**
  + **Message:** Auth code entered
  + **Description:** The instructor enters the authentication code received via email.
* **2FA checks the authentication code**
  + **Message:** Check auth\_code
  + **Description:** The 2FA system checks the entered authentication code.

Alternative 2a

* + **Condition:** Authentication successful
  + **Response:** 2FA successful
  + **Description:** If the authentication code is correct, the attendance process is completed and Instructor can export attendance report generated

Alternative 2b

* + **Condition:** Auth failed
  + **Message:** Wrong auth\_code entered, try again
  + **Description:** The authentication code is incorrect



## 4.2.3 Scenario: Instructor Login to Course

Components of the Sequence Diagram

1. **Actors:**
   * Instructor: The person who initiates the process to log into a course and track attendance.
2. **Objects:**
   * **dashboard: ins\_dashB**: The component responsible for handling the instructor's dashboard operations.
   * **course\_table: dataBase**: The database table where course information is stored.
   * **match\_pattern: authentication**: The component responsible for matching biometric patterns for authentication.
   * **instructor\_table: dataBase**: The database table where instructor information and biometric keys are stored.
   * **PasswordA**: The module handling password entry for instructors.
   * **2FA**: The module responsible for handling two-factor authentication.

Message Flow and Interactions

1. **Instructor interacts with dashboard: ins\_dashB**
   * **Message:** input course code
   * **Description:** The instructor inputs the course code to select the course they want to log into.
2. **: dashboard: ins\_dashB sends a message to Instructor**
   * **Message:** prompt user to enter biometric data
   * **Description:** **dashboard: ins\_dashB** uses the course code to retrieve the course info form **course\_table: dataBase**, and prompts the instructor to provide their biometric data for authentication.
3. **Instructor interacts with : dashboard: ins\_dashB**
   * **Message:** biometric inputs
   * **Description:** The instructor provides his/her biometric data.
4. **: dashboard: ins\_dashB interacts with match\_pattern: authentication**
   * **Message:** biometric details
   * **Description:** The system sends the extracted biometric details to the authentication module.
5. **match\_pattern: authentication interacts with instructor\_table: dataBase**
   * **Message:** request for key stored
   * **Response:** return key
   * **Description:** The authentication module requests the stored biometric key from the instructor database.
6. **instructor\_table: dataBase processes the request**
   * **Message:** request for key stored
   * **Response:** return key
   * **Description:** The database processes the request and returns the stored key.
7. **match\_pattern: authentication performs matching of biometric data**
   * **Message:** perform match
   * **Description:** The authentication module performs matching of the biometric details with the stored key.

Alternative 1

* + **Condition:** authentication true
  + **Response:** redirect to attendance tracking
  + **Description:** If authentication is successful, the instructor is redirected to the attendance tracking module.

Alternative 2

* + **Condition:** authentication fail
  + **Message:** error message - Select the right course and try again
  + **Description:** If authentication fails, the system displays an error message prompting the instructor to select the correct course and try again.

Password Alternative

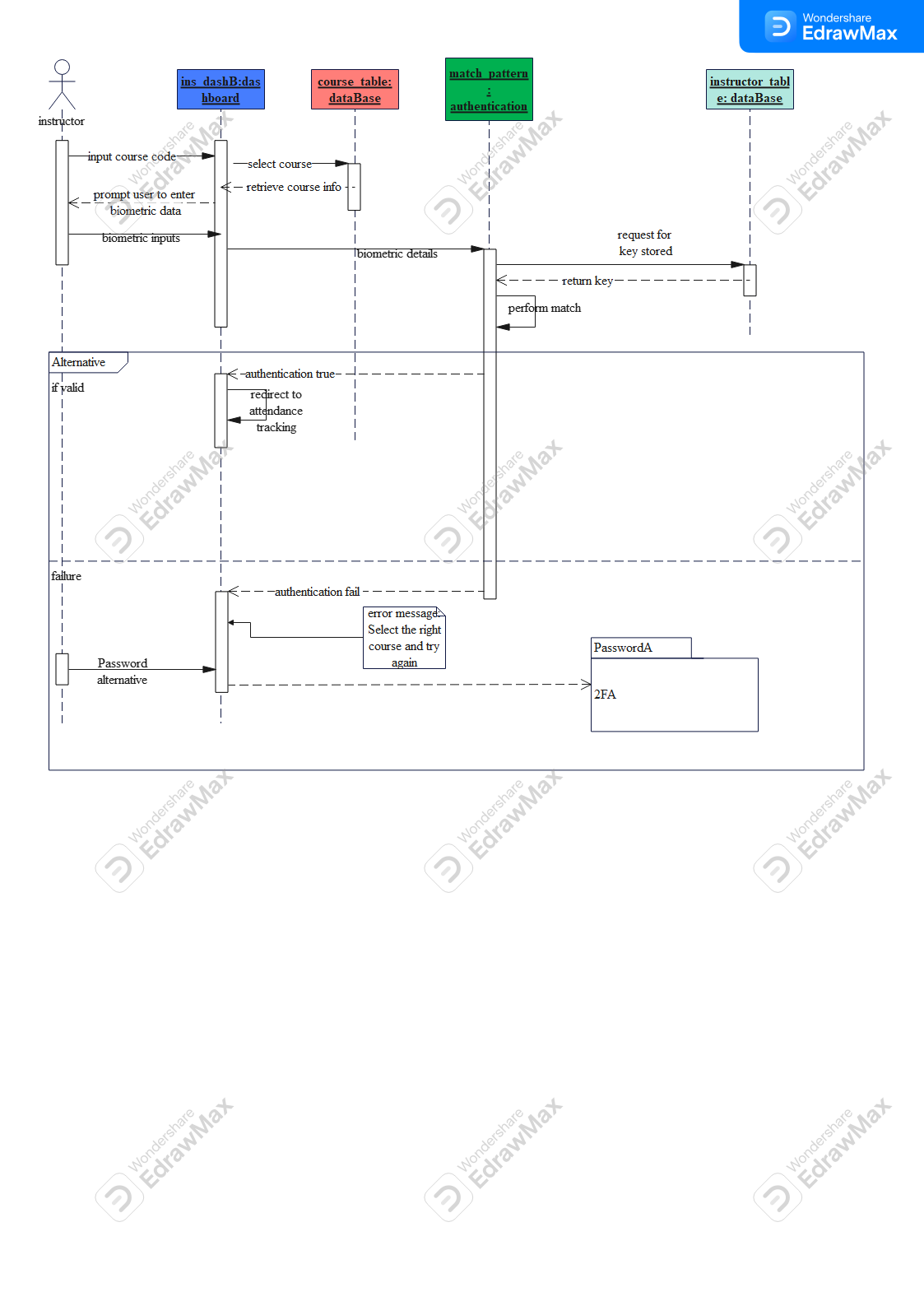
* + **Message:** Password alternative
  + **Description:** If biometric authentication fails, the system prompts the instructor to enter a password as an alternative method.
* **Instructor enters the password**
  + **Message:** Password entered
  + **Response:** Redirect user to 2FA page
  + **Description:** The system stores the entered password performs a matching of the provided password against the stored password during registration, in case of match and redirects the instructor to the 2FA page.
* **2FA sends a message to Instructor**
  + **Message:** 2FA message (auth\_code@example.com)
  + **Description:** The 2FA system sends a two-factor authentication code to the instructor's email.
* **Instructor enters the authentication code**
  + **Message:** Auth code entered
  + **Description:** The instructor enters the authentication code received via email.
* **2FA checks the authentication code**
  + **Message:** Check auth\_code
  + **Description:** The 2FA system checks the entered authentication code.

Alternative 2a

* + **Condition:** Authentication successful
  + **Response:** 2FA successful
  + **Description:** If the authentication code is correct, the instructor is redirected to the attendance tracking module.

Alternative 2b

* + **Condition:** Auth failed
  + **Message:** Wrong auth\_code entered, try again
  + **Description:** The authentication code is incorrect



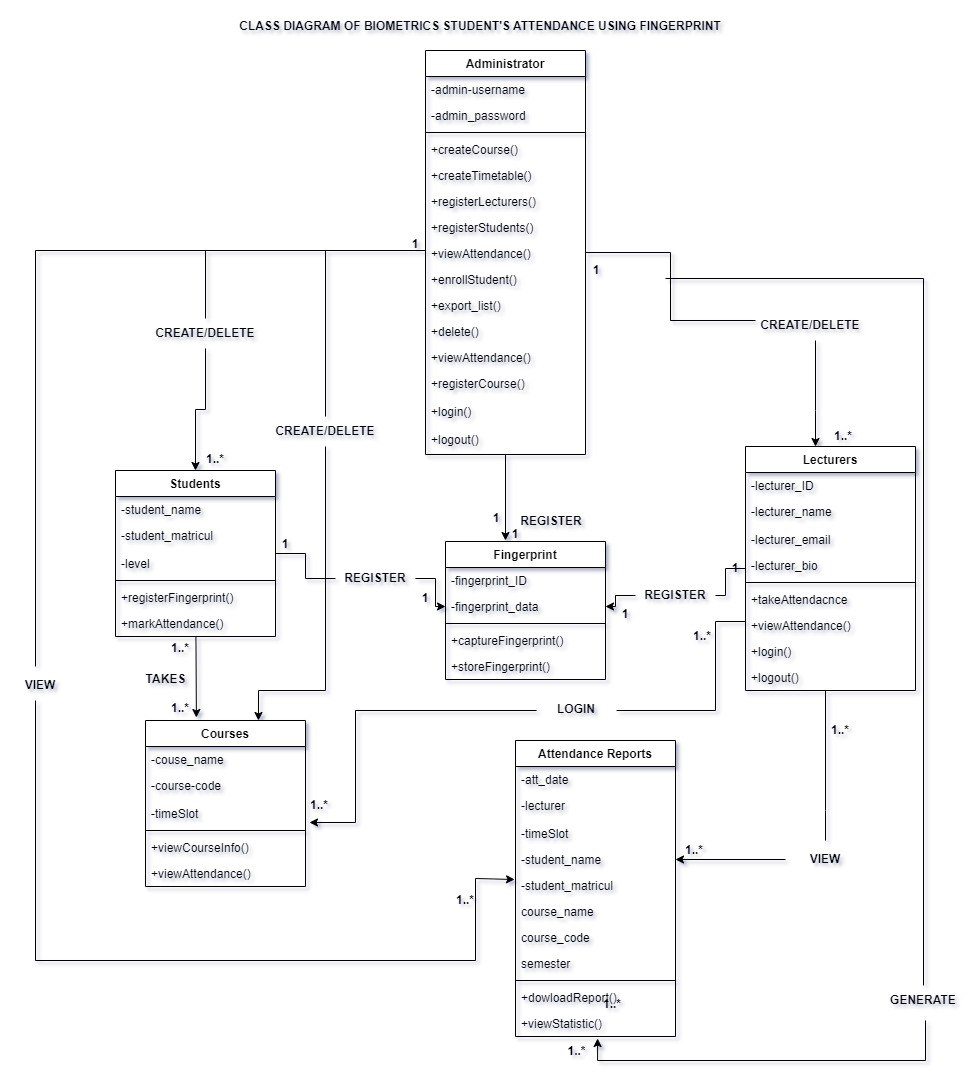
# CLASS DIAGRAM

A class diagram is a type of static structure diagram in Unified Modeling Language (UML) that describes the structure of a system by showing its classes, attributes, methods, and the relationships among objects. It is a blueprint of an application that provides a visual representation of the system's architecture.

# 5.1 Components of a Class Diagram

1. **Classes**: Represented by rectangles divided into three parts: the top part contains the class name, the middle part contains the attributes, and the bottom part contains the methods (operations).
2. **Attributes**: Characteristics or properties of a class.
3. **Methods (Operations)**: Functions or procedures that a class can perform.
4. **Relationships**
   * **Association**: A link between two classes that shows a relationship between them.
   * **Multiplicity**: Indicates the number of instances of one class that can be associated with one instance of another class.
   * **Generalization (Inheritance)**: Indicates that one class is a subclass of another class.
   * **Aggregation/Composition**: Represents a "whole-part" or “part of” relationship between classes.

## 5.2 Biometrics student’s attendance app class diagram



The class diagram of a biometric attendance system using fingerprint outlines the key entities (classes), their attributes, methods, and the relationships between them. Here is an overview of the main components and their interactions.

## 5.2.1 Key Entities (Classes) And Relationships

***Administrator***

The Administrator class is responsible for managing the overall system. They have the capability to create courses and timetables, registering both students and lecturers, viewing attendance, enrolling both students and lecturers, exporting lists, and managing user authentication (login and logout). The administrator's information is stored using a username and password.

**Cardinalities**:

* “1” (Administrator) to “1..\*” (Students)
* “1” (Administrator) to “1..\*” (Lecturers)
* “1” (Administrator) to “1..\*” (Courses)
* “1” (Administrator) to “1..\*” (Fingerprints)
* “1” (Administrator) t0 “1..\*” (Attendance Reports)

**Relationships**:

One Administrator can register one or multiple students, lecturers, fingerprint and courses

***Students***

The student’s class represents the students enrolled in the system. Each student has a name, a matriculation number, and a level. Students can register their fingerprints in the system and mark their attendance using their fingerprints.

**Cardinalities**:

* “1” (Student) to “1” (Fingerprint)
* “1..\*” (Student) to “1..\*” (Courses)

**Relationships**:

One Student is associated with one and only one Fingerprint.

One or many students takes one or many courses.

***Lecturers***

Lecturers are responsible for taking student attendance and can view the attendance report. They also have unique fingerprints linked to their accounts for authentication.

* “1” (Lecturer) to “1” (Fingerprint)
* “1..\*” (Lecturer) to “1..\*” (Courses)

**Relationships**:

One Lecturer is associated with one and only one Fingerprint.

Lecturers can teach one or multiple Courses.

***Courses***

The Courses class contains information about the courses offered in the system. Each course has a name, a course code, and a time slot. This class allows viewing course information and viewing the attendance for each course.

Cardinalities:

* “1..\*” (Course) to “1..\*” (Students)
* “1..\*” (Course) to “1..\*” (Lecturers)
* “1” (Course) to “1..\*” (Attendance Reports)

**Relationships**:

One or Multiple Courses can be taken by one or multiple Students

***Fingerprint***

The Fingerprint class manages the fingerprint data used for biometric authentication. Each fingerprint record includes a fingerprint ID and the fingerprint data itself. This class handles capturing and storing the fingerprints of students and lecturers.

**Cardinalities**:

* “1” (Fingerprint) to “1” (Students)
* “1” (Fingerprint) to “1” (Lecturer)

**Relationships**:

One Fingerprint is linked to one and only one Student or one and only one Lecturer or administrator.

***Attendance Reports***

The Attendance Reports class generates reports on attendance data. Each report includes the attendance date, lecturer, time slot, student name, student matriculation number, and a timestamp, semester, course name and course code. This class allows downloading attendance reports and viewing statistical data about attendance

Cardinalities:

* “1..\*” (Attendance Reports) to “1..” (Administrator)

**Relationships**:

Each Attendance Report is linked to one Course.

One or multiple Attendance report can be generated by one and only one Administrator.

Lecturers can View the attendance report

# DEPLOYMENT DIAGRAM

A deployment diagram is a type of UML (Unified Modeling Language) diagram that shows the physical arrangement of hardware and software in a system. It illustrates how different software components and artifacts are deployed on various nodes. This report outlines the key elements, purpose, and construction of a deployment diagram, with a focus on a practical example of a biometric student attendance mobile application.

## 6.1 Key Elements of a Deployment Diagram

**Nodes**: Represent physical hardware or computational resources where components are deployed. Nodes can be devices, servers, or any hardware capable of processing information.

Example: Tablet Device, Database Server, API Gateway.

**Artifacts**: Represent the physical files or executables that are deployed on the nodes. Artifacts can be software files like executables, libraries, or archives.

Example: AttendanceMarkingApp.apk, APIGatewayService.jar, BiometricAuthService.war.

**Components**: Logical units or modules of software that represent a set of related functionalities. Components are often packaged as artifacts.

Example: Authentication Service, Mobile Application, Course Management System.

**Communication Paths**: Show the interaction and data exchange between nodes. These paths indicate the protocols or methods used for communication.

Example: HTTPS, JDBC, REST API, Firebase SDK.

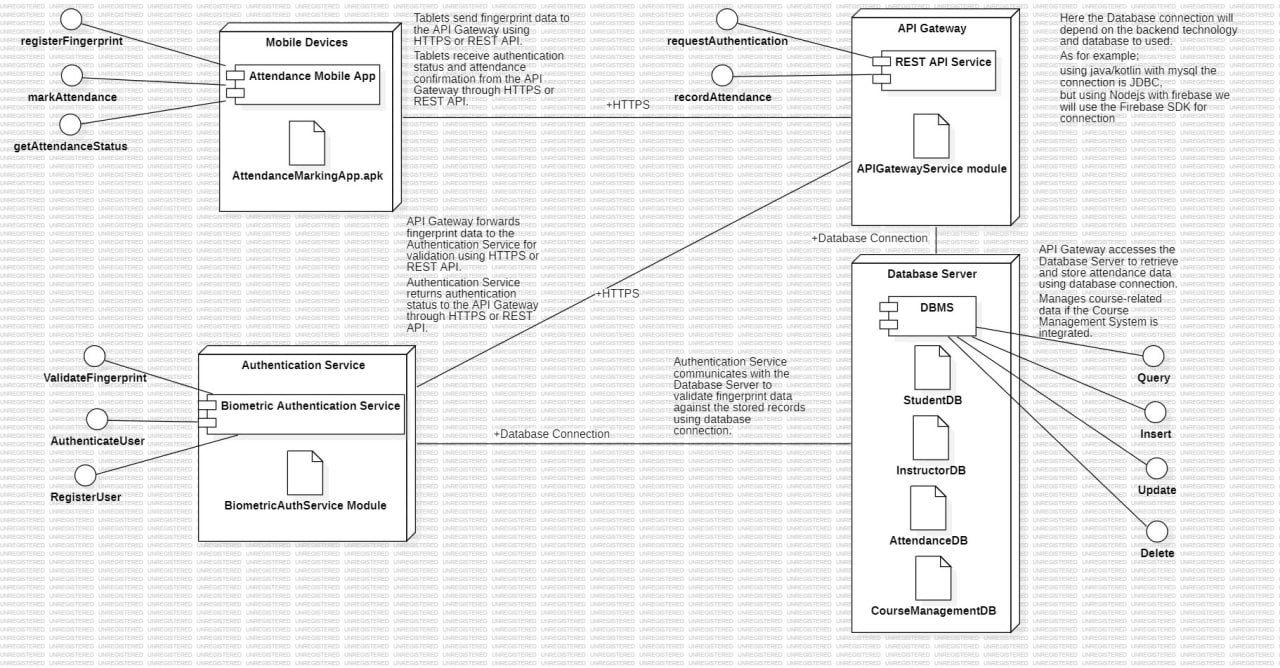
**Interfaces**: Define the interaction points for components. They specify the methods through which components communicate, ensuring standardization and consistency.

Example:

## 6.2 Purpose of Deployment Diagrams

* Visualize System Architecture: Provide a clear picture of the hardware and software environment.
* Identify System Requirements: Help in understanding the physical and infrastructural requirements.
* Facilitate Communication: Aid in communication among stakeholders by providing a common understanding of system deployment.
* Optimize Performance: Assist in identifying potential bottlenecks and optimizing the deployment for better performance and scalability.
* Practical Example: Biometric Student Attendance Mobile Application

## Biometric student’s attendance app Deployment diagram



The biometric student attendance mobile application uses tablet devices as fingerprint readers. The main stakeholders include administrators, students, and instructors. Administrators manage course creation, registration, and attendance records. The system captures student and instructor fingerprints for authentication during class sessions.

This deployment diagram is made up of **five** tools. They include;

* **Node**: A physical or virtual device that hosts components and artifacts in a system
* **Component**: A modular, encapsulated unit of software functionality, often representing a service or a logical function.
* **Artifact**: A deployable software unit, such as a file, library, or executable, used within a node.
* **Interface**: A defined set of operations provided or required by a component, specifying the "what" of interactions without detailing the "how".
* **Communication Path**: The connection through which data is transmitted between nodes, specifying the protocol or method used. This specifies the “how”

For our biometric attendance mobile application, we came out with a number of nodes, where each node contains an artifact and a component and each component has one or more interfaces. Now some nodes are connected to one or more different nodes with the help of communication link.

1. **Mobile Device (Tablet)**: this is a node and has the following;

**Component**: Attendance Mobile App

**Interface**:

**Artifact**: AttendanceMarkingApp.apk

1. **API Gateway**: this is a node and has the following;

**Component**: REST API Service

**Interface**:

**Artifact:** APIGatewayService module

1. **Database Server**: this is a node and has the following;

**Component**: Database Management System (DBMS)

**Interface**:

**Artifact**:

1. **Authentication Service**: this is a node and has the following;

**Component**: Biometric Authentication Service

**Interface**:

**Artifact**: BiometricAuthService Module

***Communication*:**

For communication path here is the breakdown;

**HTTPS**

1. This is between the API Gateway and Mobile Device

* Tablets send fingerprint data to the API Gateway using HTTPS or REST API.
* Tablets receive authentication status and attendance confirmation from the API Gateway through HTTPS or REST API.

1. This is between the API Gateway and the Authentication Service

* API Gateway forwards fingerprint data to the Authentication Service for validation using HTTPS or REST API.
* Authentication Service returns authentication status to the API Gateway through HTTPS or REST API.

**Database Connection**

1. This is between API Gateway and Database Server

* API Gateway accesses the Database Server to retrieve and store attendance data using database connection.
* Manages course-related data if the Course Management System is integrated.

1. This Is between Authentication Service and Database Server

* Authentication Service communicates with the Database Server to validate fingerprint data against the stored records using database connection.

For our mobile application the database connection highly depends on the backend technology and database we will use.

So, we will use **node.js** as backend technology and **firebase** as database due to it cloud storage advantage making us choose a database connection Firebase SDK.

# 7.0 CONCLUSION

With the system modeling and design phase complete, we have a clear and detailed architectural blueprint in place. This strong foundation is critical as we transition to the UI design phase, where we will focus on creating intuitive interfaces that align with our system's capabilities and objectives.