REPUBLIC OF CAMEROON

MINISTRY OF HIGHER EDUCATION

Peace-Work-Fatherland

REPUBLIC DU CAMEROUN

MINISTERE DE L'ENSEIGNEMENT SUPERIEUR

Paix-Travail-Patrie

UNIVERSITY OF BUEA

FACULTY OF ENGINEERING AND TECHNOLOGY,

CEF440:

INTERNET PROGRAMMING AND MOBILE PROGAMING

GROUP 23: TASK 4

Names	Matriculation Number
NDIFON LEMUEL ASHU-MBI	FE21A247
NGWASIRI RYAN TANIFIORM ONGA	FE21A266
NJIDDA SALIFU	FE21A272
ASONGNKWELLE COURAGE AYIM	FE21A142
NKWO BRAINIE NGONDA	FE21A282

Course Instructor:

Dr. Nkemeni Valery

May, 2024

Table of Content

1. Introduction 4

I. System Design Diagrams 1

2. Context Diagram 4

I. Introduction 1

II. Importance of Context Diagram 2

III. Creating Context Diagram for Fingerprint Studennt Attendance App 2

IV. Components of Context Diagram for Fingerprint Studennt Attendance Ap 2

V. Context Diagram Data 4

VI. Context Diagram (Figure 1) 5

3. Use Case Diagram 9

I. Introduction 6

II. Key elements of a use case diagram 6

III. Use case diagram (Figure 2) 7

IV. Actors 8

V. Use Cases 8

VI. Interractions 10

VII. Relationships 10

4. Sequence Diagram 11

I. Introduction 11

II. Components of a Sequence Diagram 11

III. Admin Interaction 13

IV. Admin Interaction Sequence Diagram (Figure 3) 13

V. Student Interaction 14

VI. Student InteractionSequence Diagram (Figure 4) 14

VII. Other Student Interraction 14

VIII. Other Student Interraction Sequence Diagram (Figure 5) 15

5. Class Diagram 19

I. Introduction 16

II. Classes , Attribules and Methods 16

III. Relationships and Cardinality 17

IV. Class Diagram (Figure 6) 18

6. Deployment Diagram.....19

I. Intoduction.....19
II. Key Elements of a Deployment Diagram.....19
III. Purpose of a Deployment Diagram.....19
IV. Deployment Diagram Description of a Fingerprint Student Attendance App.....20
V. Nodes, Artifacts and Communication Paths.....20
VI. Hierarchy of Deployment Diagram 21
VII. Deployment Description.....22
VIII. Deployment Diagram (Figure 7).....23

7. Conclusion.....24

8. Refrences.....25

1. Introduction

System design is a crucial phase in the software development lifecycle, where high-level planning and architectural decisions are made to define the structure and behavior of a system. It involves the detailed specification of the system's components, their interactions, and the technologies used to implement them.

I. Diagrams for System Design

- ✓ Context Diagram
- ✓ Use Case Diagram
- ✓ Sequence Diagram
- ✓ Class Diagram
- ✓ Deployment Diagram

2. Context Diagram

Introduction

A context diagram is a high-level, graphical representation of a system and its interactions with external entities. It is used in system analysis and design to provide a clear, concise overview of the system's scope and boundaries.

Here are the key features and components of a context diagram:

- ✓ **System Boundary:** Represented by a single central process or system bubble. This bubble encapsulates the entire system and is usually labeled with the name of the system.

- ✓ **External Entities:** Represented by squares or rectangles around the system bubble. These entities interact with the system but are not part of it. They could be users, other systems, organizations, or any external factors that send data to or receive data from the system.

- ✓ **Data Flows:** Arrows that show the direction of data or information flow between the external entities and the system. Each arrow is usually labeled to indicate the type of data or the nature of the interaction.

- ✓ **Simple and Clear Representation:** A context diagram does not delve into the internal workings of the system. Instead, it provides a high-level view that highlights how the system fits into its external environment.

I. Importance of Context Diagram

- ✓ **Clarify System Scope:** Helps stakeholders understand what is included in the system and what is not.
- ✓ **Identify External Interactions:** Clearly shows how the system interacts with external entities.
- ✓ **Simplify Communication:** Provides a straightforward way to communicate system boundaries and interfaces to both technical and non-technical stakeholders.

II. Creating Context Diagram of Fingerprint Student Attendance App

To create a context diagram, follow these steps:

- ✓ Identify the main system or process to be represented.
- ✓ Determine all external entities that interact with the system.
- ✓ Define the data flows between the system and these external entities.
- ✓ Draw the system boundary, external entities, and data flows using appropriate symbols (a circle for the system, rectangles for external entities, and arrows for data flows).
- ✓ Label all components clearly.

III. Components of Context Diagram for Fingerprint Student Attendance App

Context diagrams are a foundational tool in systems analysis and are often used as a starting point for more detailed system modeling.
For a fingerprint student attendance app, the main system or process and its interactions with external entities can be identified and represented as follows

a) Main System

- ✓ Fingerprint Student Attendance System

b) External Entities

- ✓ Students
- ✓ Teachers
- ✓ School Administration
- ✓ Parents
- ✓ Database Server

✓

c) Data Flow

- ✓ **From Students to Fingerprint Student Attendance System:**
    - Fingerprint Scans
    - Student ID
- ✓ **From Fingerprint Student Attendance System to Students:**
    - Attendance Confirmation (e.g., a message on a display)
- ✓ **From Teachers to Fingerprint Student Attendance System:**
    - Attendance Overrides (manual adjustments)
    - Attendance Reports Requests
- ✓ **From Fingerprint Student Attendance System to Teachers:**
    - Attendance Reports
    - Notifications of Attendance Issues
- ✓ **From School Administration to Fingerprint Student Attendance System:**
    - Student Enrollment Data
    - Class Schedules
- ✓ **From Fingerprint Student Attendance System to School Administration:**
    - Aggregated Attendance Data
    - Alerts for Truancy or Anomalies
- ✓ **From Fingerprint Student Attendance System to Parents:**
    - Attendance Notifications
    - Alerts for Absences
- ✓ **From Database Server to Fingerprint Student Attendance System:**
    - Stored Attendance Records
    - Student Information
- ✓ **From Fingerprint Student Attendance System to Database Server:**
    - New Attendance Records
    - Updates to Student Information

IV. Context Diagram Data

Here's a textual representation of the context diagram:

Fingerprint Student Attendance System (Central System Bubble)

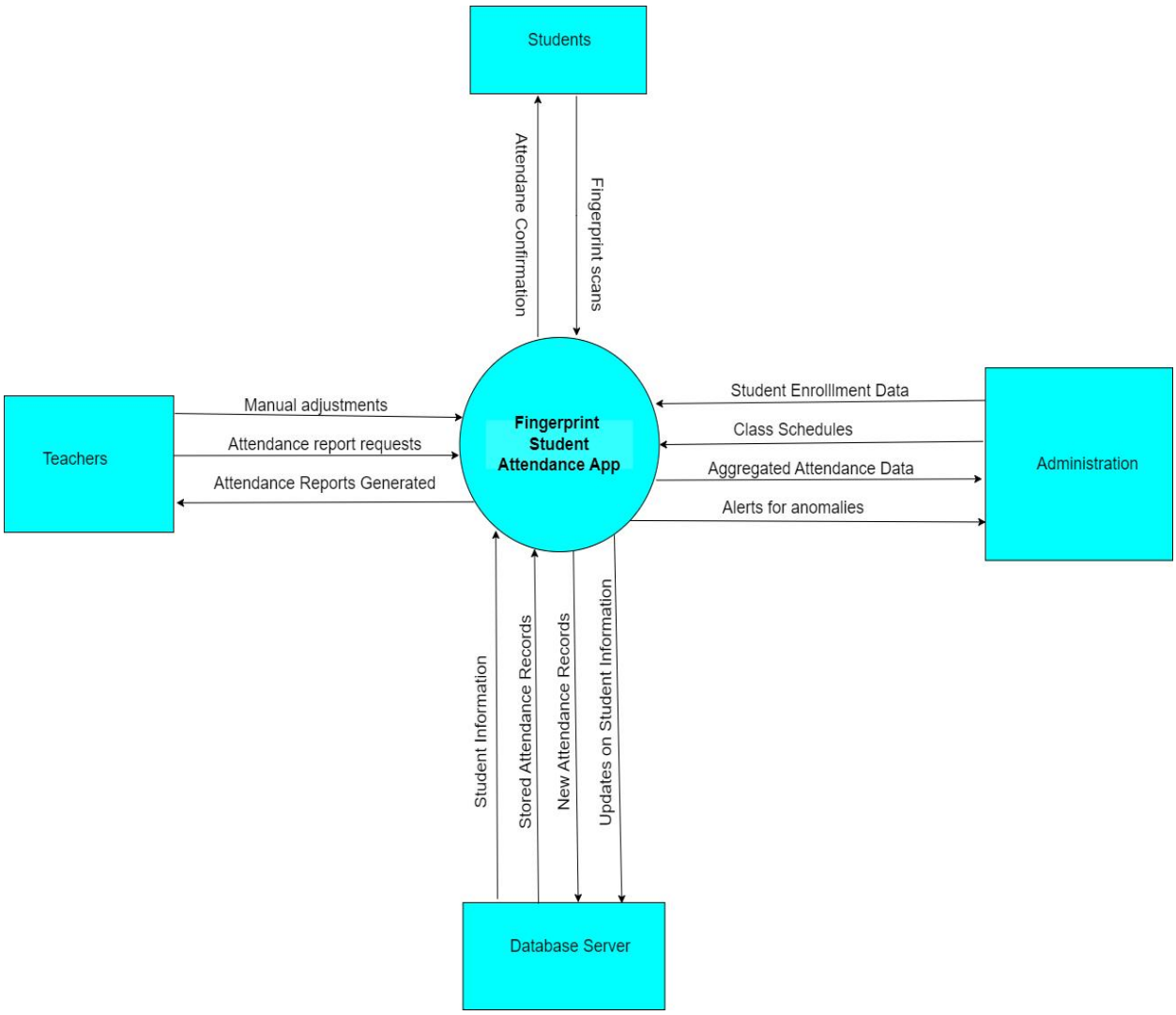
External Entities:

- ✓ Students (providing Fingerprint Scans and Student ID)
- ✓ Teachers (sending Attendance Overrides, requesting Attendance Reports)
- ✓ School Administration (providing Student Enrollment Data, Class Schedules)
- ✓ Parents (receiving Attendance Notifications and Absence Alerts)
- ✓ Database Server (storing/retrieving Attendance Records, Student Information)

Data Flows:

- ✓ From Students to System: Fingerprint Scans, Student ID
- ✓ From System to Students: Attendance Confirmation
- ✓ From Teachers to System: Attendance Overrides, Attendance Reports Requests
- ✓ From System to Teachers: Attendance Reports, Notifications of Attendance Issues
- ✓ From School Administration to System: Student Enrollment Data, Class Schedules
- ✓ From System to School Administration: Aggregated Attendance Data, Truancy/Anomaly Alerts
- ✓ From System to Parents: Attendance Notifications, Absence Alerts
- ✓ From Database Server to System: Stored Attendance Records, Student Information
- ✓ From System to Database Server: New Attendance Records, Student Information Updates
- ✓ From System to Notification Service: Attendance Alerts and Messages

V. Context Diagram (Figure 1)



3. Use Case Diagram

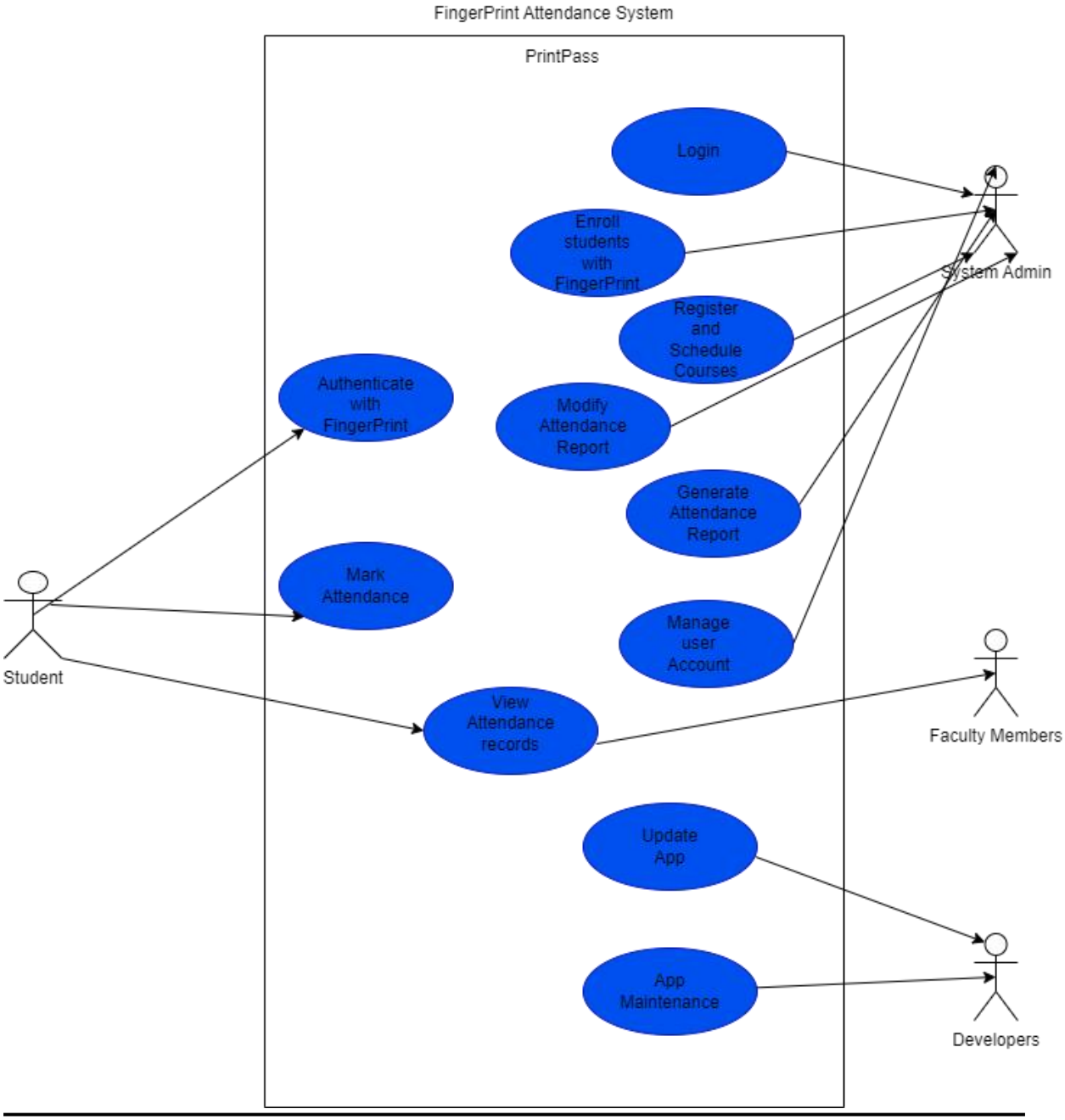
I. Introduction

A use case diagram is a type of UML (Unified Modeling Language) diagram that visually represents the interactions between a system and its users (or "actors"). It is used to model the functionality of a system by identifying the different types of users and the specific ways they will interact with the system.

II. Key Elements of a Use Case

- **Actors:** The different types of users or external entities that will interact with the system. Actors can be human users, other systems, or even hardware devices.
- **Use Cases:** The specific tasks or functions that the system will perform for the actors. Each use case represents a goal or task that the actor wants to achieve using the system.
- **Relationships:** The connections between the actors and the use cases. These relationships can include:
  - **Association:** Indicates that an actor participates in a use case.
  - **Generalization:** Indicates that one actor is a specialized version of another actor.
  - **Inclusion:** Indicates that one use case is included within another use case.
  - **Extension:** Indicates that one use case can extend the behavior of another use case under certain conditions.

III. Use Case Diagram(Figure 2)



IV. Actors

- The actors of our system are:
    - **Students**: The primary users of the mobile app who will use it to mark their attendance through fingerprint recognition.
    - **System Administrators:** Responsible for overseeing the implementation, administration, and maintenance of the app within the educational institution.
    - **Faculty Members:** Will have access to the student attendance database to query attendance records per course or for individual students.
    - **Developers:** Responsible for designing, developing, and maintaining the app, as well as providing updates and scaling the system as needed.

Actors are divided into primary and secondary and for our system the primary and secondary actors are:

- **Primary Actor(s):** The primary actors are the core users of the system who initiate the use cases to achieve their goals. They directly interact with the system and trigger the primary functionality. Primary actors are usually the main stakeholders or end-users of the system, such as customers and employees. In our system we have as primary actor:
    - **Students**
- **Secondary Actor(s):** The secondary actors are supporting players that provide information or services to the primary actors, but do not initiate the use cases themselves. Secondary actors can be systems, devices, or other entities that the primary actors rely on to accomplish their goals but are not the direct users of the system. In our system we have as secondary actors:
    - **System Administrators**
    - **Faculty Members**
    - **Developers**

V. Use Cases

- **Login/Authenticate:** Students must authenticate themselves, typically through fingerprint recognition, to access the app's features while administrators authenticate themselves with the use of a password.
- **Mark Attendance:** Students use the app to mark their attendance by scanning their fingerprint.

- **View Attendance Records:** Faculty members and administrators can view attendance records for individual students or for entire courses.
- **Generate Attendance Reports:** Administrators can generate attendance reports for various purposes, such as monitoring student attendance or generating class attendance records.
- **Manage User Accounts:** Administrators can create, update, and delete user accounts for students.
- **Provide App Updates:** Developers are responsible for providing updates to the app, addressing feedback from users, and scaling the system as needed.

VI. Interactions

- Students interact with the app to mark their attendance and view their own attendance records.
- Faculty members interact with the app to view attendance records for their courses.
- Administrators interact with the app to manage user accounts and oversee the overall performance and maintenance of the system.
- Developers interact with the app to provide updates, address user feedback, and scale the system as required.

VII.Relationships

- The use cases demonstrate the different functionalities and interactions between the various stakeholders and the Fingerprint Student Attendance Mobile App.
- The use cases are interconnected, as some use cases (e.g., Login/Authenticate) are required for accessing other use cases (e.g., Mark Attendance, View Attendance Records).
- The use cases also highlight the different responsibilities and concerns of the stakeholders, such as the students' need for a user-friendly app, the administrators' focus on system management and security, and the developers' role in maintaining and updating the app.

Use case diagrams are typically created during the early stages of the software development process, as they help to identify and document the system's functional requirements. They provide a high-level overview of the system's functionality and can be used to communicate the system's requirements to stakeholders, developers, and other project team members.

The use case diagram provides a comprehensive overview of the key stakeholders and their interactions with the Fingerprint Student Attendance Mobile App, which serves as a foundation for the subsequent design and development phases of the project.

4. Sequence Diagram

I. Introduction

A sequence diagram is a type of interaction diagram that shows how objects interact in a particular scenario of a business process or a system.

In our project , interactions exist for the admin , the student and other intergratable systems with the application

II. Interaction Types

In a sequence diagram, interactions are represented to show how objects or components within a system communicate with each other. Here are the main interaction types typically found in sequence diagrams:

a) Synchronous Message:

- ✓ **Description:** A message where the sender waits for the receiver to process the message and return control.
- ✓ **Notation:** Represented by a solid line with a filled arrowhead pointing from the sender to the receiver.

b) Asynchronous Message:

- ✓ **Description:** A message where the sender sends the message and continues without waiting for the receiver to process it.
- ✓ **Notation:** Represented by a solid line with an open arrowhead pointing from the sender to the receiver.

c) Reply Message:

- ✓ **Description:** A message sent back to the sender to confirm that a synchronous message has been processed.
- ✓ **Notation:** Represented by a dashed line with an open arrowhead pointing back to the original sender.

d) Create Message:

- ✓ **Description:** A message that creates a new instance of an object.

- ✓ **Notation:** Represented by a solid line with a filled arrowhead pointing to the object's lifeline where it starts.

e) Delete Message:

- ✓ **Description:** A message that destroys an object.
- ✓ **Notation:** Represented by a solid line with a filled arrowhead pointing to an 'X' at the end of the object's lifeline.

f) Self-Message (Self-call):

- ✓ **Description:** A message that an object sends to itself, often to invoke its own methods.
- ✓ **Notation:** Represented by a solid line that loops back to the same object's lifeline with a filled arrowhead.

g) Found Message:

- ✓ **Description:** A message that originates from an unknown or external source.
- ✓ **Notation:** Represented by a solid line with a filled arrowhead pointing to the lifeline from the top of the diagram.

h) Lost Message:

- ✓ **Description:** A message sent to an unknown or external receiver.
- ✓ **Notation:** Represented by a solid line with a filled arrowhead pointing away from the lifeline to the bottom of the diagram.

i) Combined Fragments:

- ✓ **Description:** These are structures that represent conditional or iterative behavior within the sequence diagram. They include:
  - **Alternative (alt):** Represents a choice between two or more message sequences.
  - **Option (opt):** Represents a message sequence that occurs only if a certain condition is true.
  - **Loop (loop):** Represents a message sequence that is repeated.
  - **Parallel (par):** Represents parallel message sequences.
  - **Break (break):** Represents an interruption in the message sequence.
- ✓ **Notation:** Enclosed in a rectangular frame with the operator (alt, opt, loop, etc.) in the top-left corner.

j) Gates:

- ✓ **Description:** These represent the entry and exit points of messages when dealing with complex interactions or interactions across different sequence diagrams.
- ✓ **Notation:** Small squares or circles on the lifeline with a label indicating entry or exit.

III. Admin Interactions

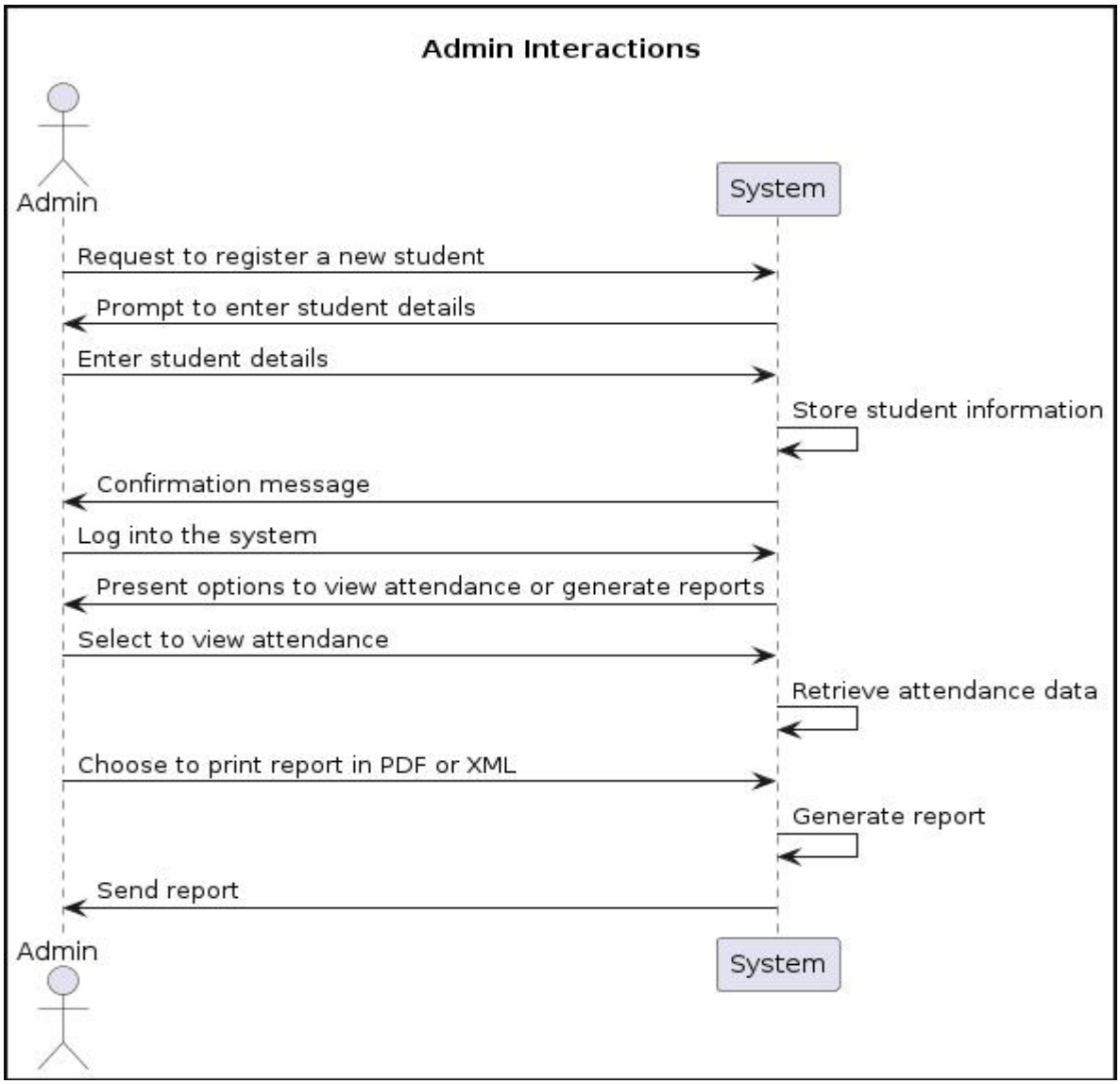
Description

This sequence diagram depicts the interactions between an administrator (Admin) and the School Attendance Tracking System. It illustrates the process of registering a new student and viewing attendance reports.

Explanation

The first interaction sequence demonstrates how an administrator initiates the registration process for a new student. Upon receiving the request, the system prompts the administrator to input the necessary student details, including personal information and fingerprint data. Once entered, the system securely stores the information in the database and provides confirmation to the administrator. Additionally, the sequence illustrates how administrators can log into the system to access attendance data, choose specific reports, and generate them in various formats for further analysis or distribution.

IV. Admin Interactions Sequence Diagram (Figure 3)



V. Student Interactions

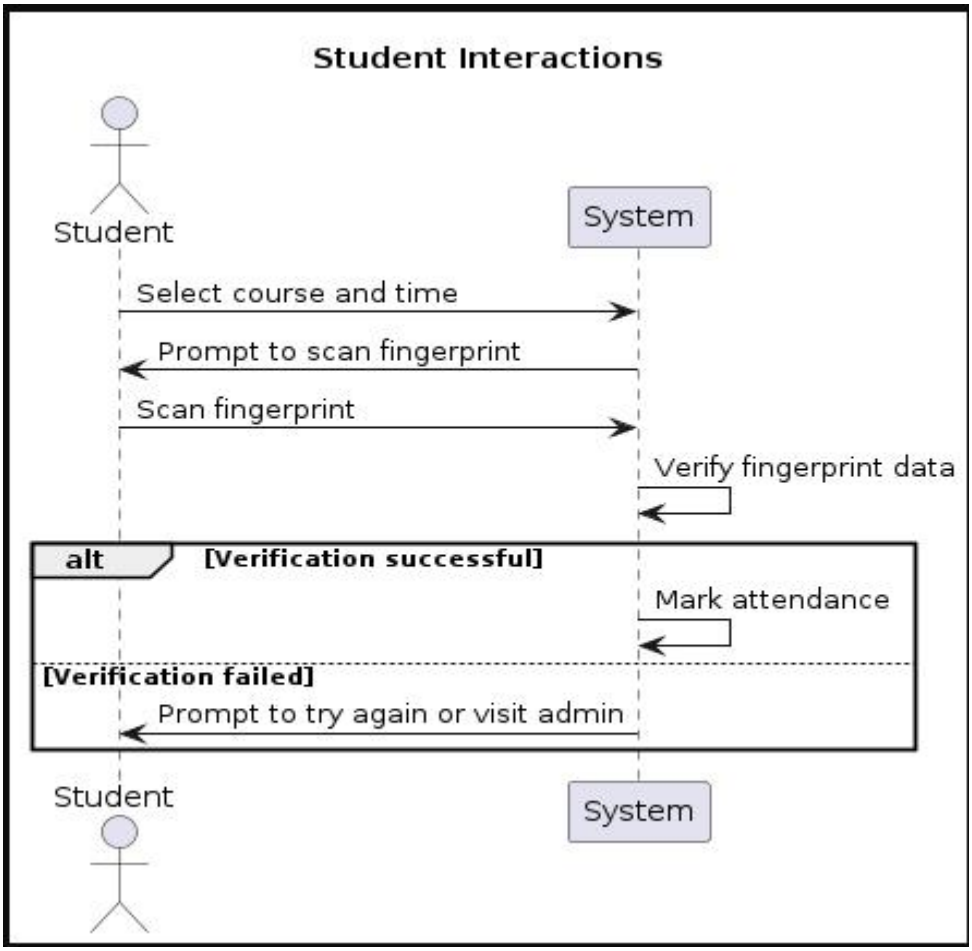
Description

This sequence diagram illustrates the interactions between a student (Student) and the School Attendance Tracking System. It showcases the process of student attendance, including selecting courses, scanning fingerprints, and marking attendance.

Explanation

The second sequence diagram details the workflow of a student interacting with the attendance tracking system. Students begin by selecting their courses and corresponding time slots. Upon arrival, they authenticate their attendance by scanning their fingerprint, which the system verifies against stored data. If the verification is successful, the student's attendance is marked. However, in cases of failed verification, students are prompted to retry or seek assistance from administrators.

VI. Student Interaction Sequence Diagram (Figure 4)



VII.Other System Interactions

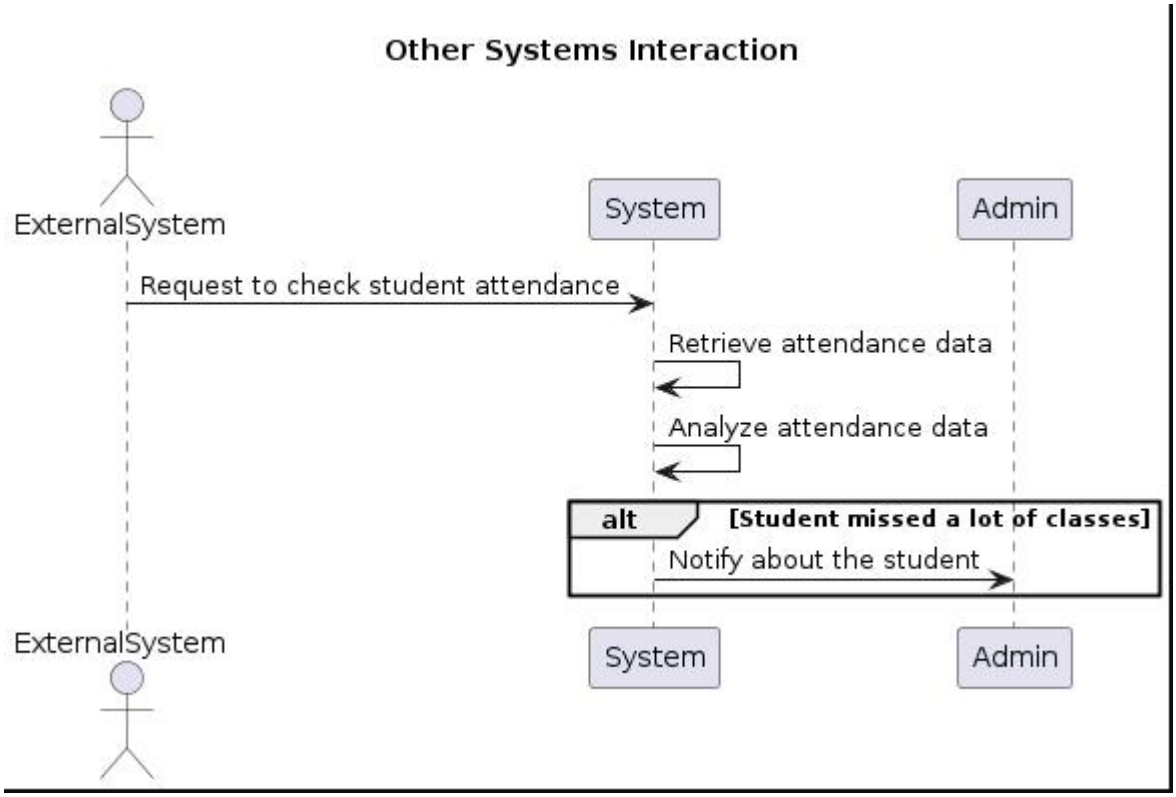
Description

This sequence diagram demonstrates interactions between external systems (ExternalSystem) and the School Attendance Tracking System. It outlines the process of external systems requesting student attendance data and the system's response, including notifications to administrators.

Explanation

The third sequence diagram outlines how external systems interact with the attendance tracking system to access student attendance data for analytical or monitoring purposes. External systems initiate requests for attendance data, which the system handles by retrieving relevant information from the database. After analysis, the system identifies cases where students have missed significant classes and notifies administrators accordingly. This interaction highlights the system's interoperability and its role in facilitating data-driven decision-making processes.

VIII. Other System Interactions Sequence Diagram(Figure 5)



The sequence diagrams presented in this report provide a comprehensive overview of the Fingerprint Studennt Attendance System's functionality and interactions with different actors. These diagrams, coupled with detailed explanations, serve as valuable documentation for understanding the system's behavior and can aid in future development and maintenance efforts.

5. Class Diagram

I. Introduction

A class diagram for a School Attendance Biometric Management System illustrates the system's structure, including the classes, attributes, methods, and relationships between classes.

II. Classes, Attributes and Methods of System

1. **Course**: Represents a course offered in the school.

   It has the following attributes:

   +Id: string
   +Name: string
   +Teacher: string

2. **Students**: Represents students enrolled in the school.

   It has the following attributes an:

   +ld: string
   +CourseDate: date
   +Name: string
   +Fingerprint: template
   -Matricule: string

   It has one method which is MarkAttendance()

3. **Admin**: Represents an administrator who manages the system.

   It has the following attributes:

   -Id: string
   -Name: string
   -Password: string
   -Email: string

   It also has the following methods;

   -ViewAttendance()
   -PrepareAttendanceReport()
   -AddNewStudents()
   -AddNewCourses()
   -ModifyCourses()
   -ManagePeriods()

4. **Period**: Represents a time period within a day (e.g., a class period).

It has the following attributes:

+Id: string
+Courseld: string
+CourseDay: string
+Course Time: time

5. **Attendance**: Represents the attendance record of a student for a specific period of a course.

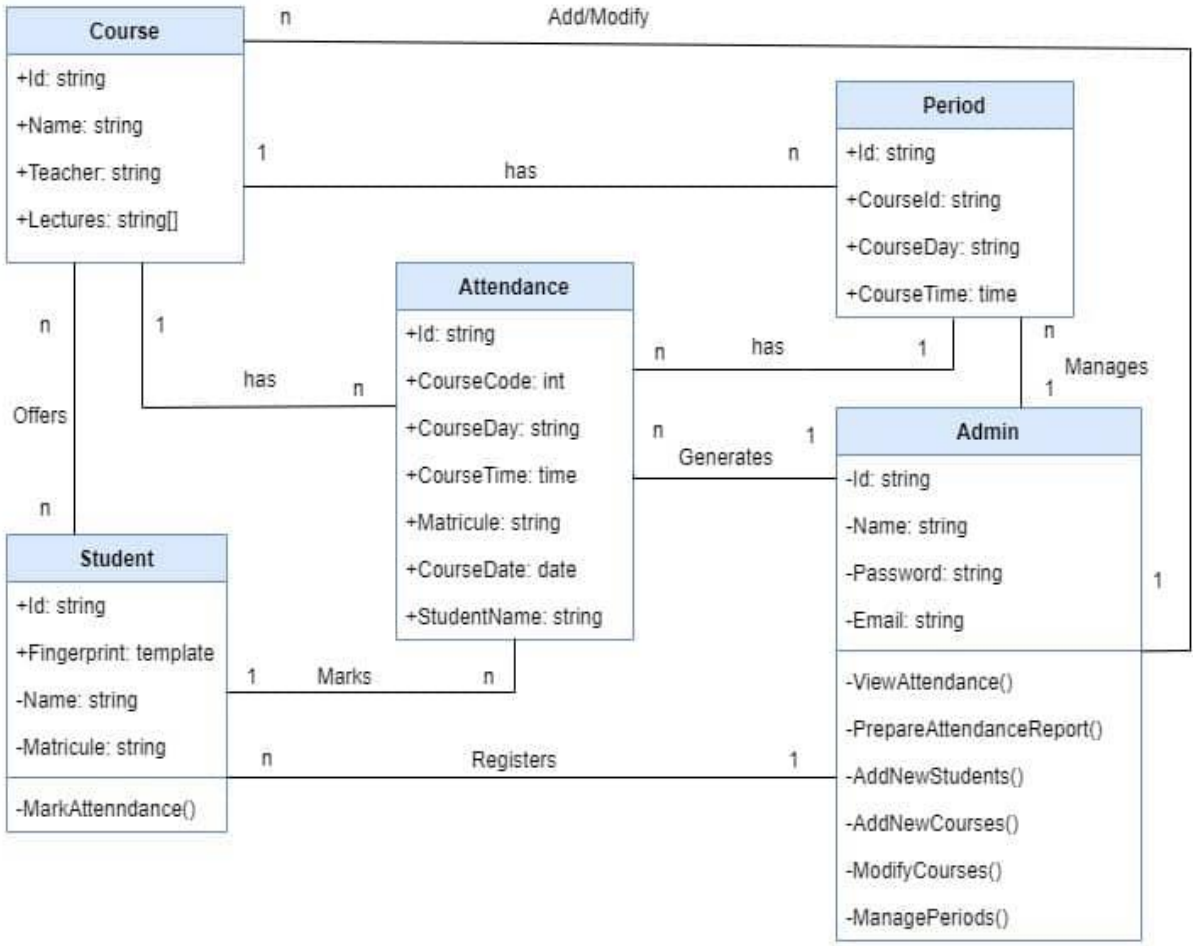
It has the following attributes:

+Id: string
+CourseDay: string
+Course Time: time
+Matricule: string
+Course Date: date
+StudentName: string

III. Relationships and cardinality

- ✓ **Course - Student:** Many-to-Many (A student can enroll in many courses, and a course can have many students)
- ✓ **Course - Attendance:** One-to-Many (A course can have multiple attendance records)
- ✓ **Course-Period:** One-to-Many (A course can have many periods)
- ✓ **Student - Attendance:** One-to-Many (A student can have multiple attendance records)
- ✓ **Period - Attendance:** One-to-Many (A period can have multiple attendance records)
- ✓ **Admin - Student:** One-to-Many (An admin registers many students)
- ✓ **Admin - Period:** One-to-Many (An admin manages many periods)
- ✓ **Admin - Course:** one-to-Many (An admin can add/modify many courses)
- ✓ **Admin - Attendance:** One-to-Many (An admin can generate many attendance records)

IV. Class Diagram (Figure 6)



6. Deployment Diagram

I. Introduction

A deployment diagram is a type of structural diagram in the Unified Modeling Language (UML) that shows the physical arrangement of hardware and software components in a system. It provides a visualization of the deployment of artifacts to nodes and represents the hardware and execution environment in which a system operates.

II. Key Elements of a deployment diagram

✓ **Nodes:**

- ***Devices:*** Physical hardware components (e.g., servers, computers, mobile devices).
- ***Execution Environments:*** Software environments in which application components are deployed (e.g., operating systems, virtual machines, databases).

✓ **Artifacts:**

Represent the concrete elements that are deployed on nodes (e.g., executable files, libraries, databases).

✓ **Communication Paths:**

Represent the relationships and interactions between nodes, showing how they communicate with each other.

✓ **Components:**

Higher-level elements that might be deployed, such as software components or web services.

III. Purpose of a deployment diagram

✓ **Visualization of Physical Structure:** Helps in understanding the physical layout of the system.

✓ **Deployment Planning:** Assists in planning the deployment of a system by mapping out where each software component will reside.

✓ **Performance Analysis**: Helps in analyzing the potential performance bottlenecks by showing communication paths and dependencies.

✓ **System Maintenance:** Aids in system maintenance by providing a clear view of where components are deployed.

IV. Deployment Diagram Description for Fingerprint Student Attendance App

Here is the creation of a deployment diagram for a fingerprint student attendance mobile application. This application allows students to mark their attendance using their fingerprints, which are scanned by their mobile devices. Here's a detailed description:

The deployment diagram showcases the setup of a fingerprint student attendance mobile app.

- ✓ Mobile devices, equipped with fingerprint scanners, are used by students for attendance.
- ✓ The app communicates with an application server for authentication and data processing.
- ✓ An application server, hosted remotely, manages the app's logic and database interactions.
- ✓ It authenticates students based on their fingerprint data.
- ✓ Attendance data is securely transmitted and processed by the application server
.
- ✓ A separate database server stores student records and attendance history.
- ✓ The app communicates with the database server to fetch and store data.
- ✓ Mobile devices connect to the application server via network connectivity.

Together, these components enable efficient attendance tracking using fingerprint technology. From the description here is the data drawn for the deployment diagram

V. Nodes, Artifacts and Communication Paths

a) Nodes:

- ✓ **Mobile Devices (Client):**
  Represents the smartphones or tablets used by students.
- ✓ **Application Server:**
  Hosts the backend logic and database of the attendance system.
- ✓ **Database Server:**
  Stores student information, attendance records, and other relevant data.

b) Artifacts:

- ✓ **Mobile Application:**
  The application installed on mobile devices, facilitating attendance marking and fingerprint scanning.
- ✓ **Server-side Application:**
  The software deployed on the application server, managing authentication, attendance processing, and database interactions.
- ✓ **Database:**

Represents the database schema and data stored on the database server, including student records and attendance logs.

c) Communication Paths:

✓ **Between Mobile Devices and Application Server:**

- Handles requests from the mobile application for authentication and attendance marking
.
- Enables the transmission of fingerprint data from mobile devices to the application server.

✓ **Between Application Server and Database Server:**

- Facilitates communication for retrieving student information and storing attendance data
.
- Manages interactions between the server-side application and the database.

VI. Hierarchy of Deployment Diagram Components

The mobile devices are standalone entities and are not contained within any other component.

1) Fingerprint Scanner (Hardware):

This component is usually integrated into or attached to the mobile devices. Therefore, it resides within the Mobile Devices (Client) component.

2) Mobile Application:

The mobile application software runs on the mobile devices and utilizes the device's hardware, including the fingerprint scanner. Therefore, the Mobile Application resides within the Mobile Devices (Client) component.

3) Application Server:

The application server is a separate entity from the mobile devices. It hosts the backend logic of the attendance system and serves requests from the mobile application. Therefore, the Application Server component is external to the Mobile Devices (Client) component.

4) Server-side Application:

The server-side application logic, responsible for authentication, attendance processing, and database interactions, runs on the Application Server. Therefore, the Server-side Application resides within the Application Server component.

5) Database Server:

The database server stores student information and attendance records. It is separate from the application server but interacts with it to store and retrieve data. Therefore, the Database Server component is external to the Application Server component.

6) Database:

The database schema and data reside within the Database Server component. It stores student records and attendance logs. Therefore, the Database resides within the Database Server component.

VII.Deployment Description

✓ **Mobile Devices:**

- Each student uses their mobile device to access the attendance application.
- The application utilizes the device's fingerprint scanner for authentication.
- Communication with the application server is established through network connectivity (e.g., Wi-Fi, cellular data).

✓ **Fingerprint Scanner:**

- Integrated into the mobile devices, the fingerprint scanner captures and verifies the students' fingerprints.
- The scanned fingerprint data is securely transmitted to the application server for authentication.

✓ **Application Server:**

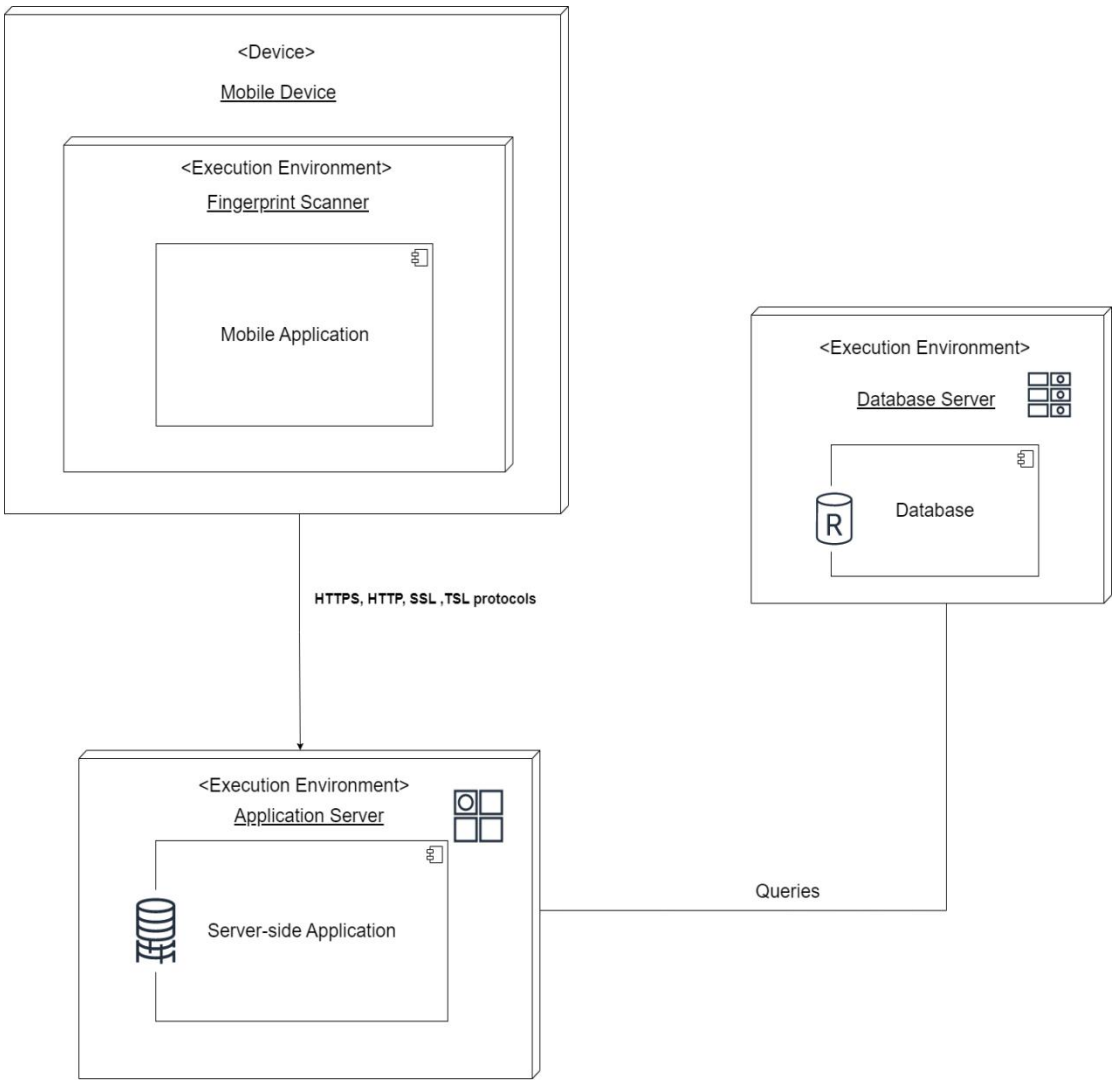
- Hosted on a server infrastructure, the application server manages the core logic of the attendance system.
- It authenticates students based on their fingerprint data and retrieves student information from the database.
- Attendance data is stored and processed on the application server before being saved to the database.

✓ **Database Server:**

- Hosts the database containing student records, including personal information and attendance history.
- The database server handles data storage and retrieval operations requested by the application server.

In summary, the deployment diagram illustrates the physical components and communication pathways involved in the fingerprint student attendance mobile application. It demonstrates how mobile devices interact with the application server and database server to enable attendance marking and data management functionalities.

VIII. Deployment Diagram (Figure 7)



7. Conclusion

The system design of the Student Fingerprint Attendance App represents a significant advancement in modernizing and streamlining the attendance tracking process in educational institutions. This report has detailed the comprehensive design and architecture, highlighting key components such as the fingerprint recognition module, the secure database management system, and the intuitive user interface. By leveraging biometric technology, the app ensures high accuracy and eliminates common issues associated with traditional attendance methods, such as manual errors and proxy attendance.

The proposed system not only enhances operational efficiency but also improves the overall user experience for both students and administrators. Its scalability ensures that it can be effectively implemented in institutions of varying sizes, and its security measures protect sensitive student data, complying with privacy regulations.

In summary, the Student Fingerprint Attendance App offers a robust, reliable, and secure solution for attendance management. The design principles and technologies employed promise to deliver a practical and innovative tool that meets the evolving needs of educational institutions, paving the way for future advancements in academic administration.

8. References

- ✓ Jain, Anil K., Flynn, Patrick, and Ross, Arun A.
  Handbook of Biometrics.
  Springer, 2008.
  ISBN: 978-0387710402.

- ✓ Maltoni, Davide, Maio, Dario, Jain, Anil K., and Prabhakar, Salil.
  Handbook of Fingerprint Recognition.
  Springer, 2009.
  ISBN: 978-1848822535.
  Navabi, Zainalabedin.

- ✓ Biometric Systems: Technology, Design and Performance Evaluation.
  Springer, 2005.
  ISBN: 978-0387202747.
  Elmasri, Ramez, and Navathe, Shamkant B.

- ✓ Fundamentals of Database Systems.
  Pearson, 2015.
  ISBN: 978-0133970777.
  Pressman, Roger S.

- ✓ Software Engineering: A Practitioner's Approach.
  McGraw-Hill Education, 2019.
  ISBN: 978-1260548006.
  Sommerville, Ian.

- ✓ Software Engineering.
  Pearson, 2015.
  ISBN: 978-0133943030.
  Martin, Robert C.

- ✓ Clean Architecture: A Craftsman's Guide to Software Structure and Design.
  Prentice Hall, 2017.
  ISBN: 978-0134494165.
  Bass, Len, Clements, Paul, and Kazman, Rick.

- ✓ Software Architecture in Practice.
  Addison-Wesley Professional, 2012.
  ISBN: 978-0321815736.
  Connolly, Thomas M., and Begg, Carolyn E.

- ✓ Database Systems: A Practical Approach to Design, Implementation, and Management.
  Pearson, 2014.
  ISBN: 978-0132943269.
  Niemeyer, Patrick, and Knudsen, Daniel Leuck.