A

PROJECT REPORT

ON

“MOBILE ATTENDANCE SYSTEM WITH INTEGRATED FINGERPRINT SENSOR”

Submitted in

Partial Fulfilment of requirement for the Award Of

DIPLOMA

In

COMPUTER ENGINEERING

Submitted by

POLABATHINA RAMCHARAN TEJA (21259-CS-024)

Under the esteemed guidance of

Mrs. P. SIREESHA M. Tech.

(Sr. Lecturer)

SAMSKRUTI COLLEGE OF ENGINEERING &TECHNOLOGY

II SHIFT POLYTECHNIC

Kondapur (V), Ghatkesar (M), Medchal (D), Telangana.

Accredited by NAAC, Approved by AICTE, Affiliated to JNTUH/SBTET, Hyderabad.

(2021-2024)

SAMSKRUTI COLLEGE OF ENGINEERING &TECHNOLOGY
II SHIFT POLYTECHNIC
Kondapur (V), Ghatkesar (M), Medchal (D), Telangana.
Accredited by NAAC, Approved by AICTE, Affiliated to JNTUH/SBTET, Hyderabad.

Department of Computer Engineering

CERTIFICATE

This is to certify that the project report entitled **“MOBILE ATTENDANCE SYTEM WITH INTERGRATED FINGERPRINT SENSOR”** Is being submitted by **POLABATHINA RAMCHARAN TEJA (21259-CS-024)** in partial fulfilment of the requirements for the award of DIPLOMA from SBTET Hyderabad in COMPUTER ENGINEERING. This record is a bonafide work carried out by them under my guidance and supervision. The results embodied in this report have not been submitted to any other university for the award of any degree.

INTERNAL GUIDE	**HEAD OF SECTION**
Mrs. P. SIREESHA	**Mr. RAVIKUMAR B. C.**
M. Tech.	**M. Tech.**

PROJECT CO-ORDINATOR	**EXTERNAL EXAMINER**

Place:

Date:

ACKNOWLEDGEMENT

Firstly, I would thank GOD for having helped us through this work. I would like to express our sincere gratitude to our honorable principal **Mr. K.SHIVA KESHAVA REDDY** for granting us permission to do the project in our college and for his encouragement. I profoundly thank **Mr. RAVIKUMAR B. C.,** Head of Section of computer engineering, Samskruti College of Engineering & Technology II Shift Polytechnic for his help and encouragement.

I would like to express our deep sense of gratitude and whole-hearted thanks to our project guide **Mrs. P. SIREESHA**, Sr. Lecturer, Dept. of CME, Samskruti College of Engineering & Technology for giving me the privilege of working under the esteemed guidance in carrying out this project work.

Finally I extend our gratefulness to one and all who are directly or indirectly involved in the successful completion of this project work.

With Sincere regards,

POLABATHINA RAMCHARAN TEJA **(21259-CS-024)**

MOBILE ATTENDANCE SYTEM
WITH
INTEGRATED FINGERPRINT SENSOR

ABSTRACT

This project presents an advanced **Mobile Attendance System** that employs cutting-edge **fingerprint authentication** technology. The system is designed to streamline the process of marking and viewing student attendance through a **mobile application**, ensuring a secure and user-friendly experience. The innovation lies in its minimalistic requirements, eliminating the need for heavy or costly equipment and relying solely on ubiquitous **mobile devices.**The workflow involves two primary stakeholders: **students and lecturers**. Students download a dedicated mobile application, which becomes their portal for attendance-related activities. Simultaneously, lecturers access a specially designed **website** **for administrative tasks**. Upon **login**, the website generates two **QR** codes, visible to students through a projector during class sessions.To mark attendance, students log in to the mobile application using their unique **credentials**, providing a secure link to their identity. They then scan the QR codes displayed on the lecturer's web browser. One QR code facilitates connection to the Wi-Fi network, while the other marks the student's attendance. This two-step authentication process enhances the accuracy and security of attendance data.Students can conveniently check their attendance records through the mobile application, fostering transparency and accountability. Furthermore, lecturers have the **authority to modify** or edit attendance details via the dedicated website, offering adaptability in managing classroom dynamics.In conclusion, the Innovative Mobile Attendance System with Fingerprint Authentication establishes a secure and efficient method for tracking student attendance. By leveraging the ubiquity of mobile devices, this system represents a significant advancement in modern educational technology, offering a seamless and **user-friendly** experience for both students and lecturers.

CHAPTER – 1
INTRODUCTION

The "Mobile Attendance System" represents a pioneering software endeavour aimed at revolutionizing the traditional approach to student attendance management. In response to the evolving landscape of education technology, this project was conceived to provide a fully automated and secure solution for attendance tracking, ensuring both convenience for students and efficient data management for lecturers.

Central to this system is a user-friendly mobile application tailored for students, offering a seamless interaction with the server. Each student is provided with unique login credentials, enhancing authentication and security. Upon logging in, the application's interface dynamically adjusts to the student's details, facilitating a personalized experience. The primary feature allows students to effortlessly mark their attendance by scanning QR codes displayed on the dedicated website.

For lecturers, a comprehensive web portal has been developed to manage attendance data efficiently. Through secure login credentials, lecturers gain access to QR codes, enabling them to mark students as present or absent and modify individual details as needed. The system employs an Apache Tomcat server, utilizing Java servlets and JSP pages for request processing and response generation.

The mobile application, developed using Kotlin and XML in Android Studio, complements the server-side functionality. The server, employing HTML, CSS, JavaScript, XML, PHP, JSP, and Java, leverages a MySQL database connected through JDBC drivers. This integrated approach ensures a robust and scalable solution, enhancing the reliability and effectiveness of attendance management.

In the subsequent sections of this report, we will delve into the intricacies of the system architecture, the development process, and the technologies employed, providing a comprehensive understanding of the "Mobile Attendance System."

CHAPTER – 2
SYSTEM ANALYSIS

1. User Requirements:

Students:

Mobile Application Interface:

- Secure and user-friendly UI for attendance marking.
- Personalization based on individual login credentials.

Authentication:

- Fingerprint authentication within the mobile application.
- Generation of a unique ID during the first login.

Attendance Marking:

- QR code scanning for seamless attendance tracking.

Lecturers:

Web Portal:

- Secure login for access to attendance management features.
- QR code generation for student attendance.

Attendance Administration:

- Marking students present or absent.
- Modification of individual student details.

2. Functional Requirements:

Mobile Application:

- Implementation of secure login mechanisms.
- Dynamic UI adjustments based on student details.
- Integration of fingerprint authentication for secure access.
- QR code scanning functionality for marking attendance.

Web Portal:

- Secure login functionality for lecturers.
- QR code generation for student attendance tracking.
- Features for marking students present or absent.
- Ability to modify individual student details.

3. System Specifications:

Server-Side:

- Utilization of Apache Tomcat server.
- Implementation of Java servlets and JSP pages for handling requests.
- Integration of HTML, CSS, JavaScript, XML, PHP, and Java.
- Connection to a MySQL database using JDBC drivers for data storage.

Mobile Application:

- Developed in Android Studio using Kotlin and XML.
- Integration of secure authentication mechanisms.
- QR code scanning feature for seamless attendance marking.

4. Security Considerations:

Fingerprint Authentication:

- Fingerprint data on the mobile device is not directly accessible by the server.
- Authentication occurs within the mobile application.
- All fingerprints available on the mobile device can authenticate.
- A unique ID generated during the first login is stored on both the mobile app and server for additional security.

5. Usability and User Experience:

● Iterative testing and refinement of the mobile application's UI.

● Feedback mechanisms for continuous improvement.

● Training sessions for both students and lecturers to ensure ease of use.

6. Scalability and Performance:

● Design considerations for scalability to accommodate potential growth in the user base.

● Optimization of database queries and server-side processes for enhanced performance.

7. Documentation and Training:

● Comprehensive documentation for system usage, administration, and troubleshooting.

● Training sessions for both students and lecturers to ensure effective utilization.

This system analysis outlines the detailed requirements, functionalities, and technological needed for the "Mobile Attendance System." It takes into account the unique features such as fingerprint authentication and the generation of unique IDs to ensure secure and efficient attendance tracking.

CHAPTER -3
EXISTING SYTEM

Traditional Student Attendance Management

Disadvantages:

1.Costly Infrastructure:

● Traditional systems involve substantial investment in biometric machines, infrastructure, and maintenance.

● Limited budgets constrain widespread adoption in educational institutions.

2.Limited Accessibility:

● Biometric machines are confined to specific locations, limiting their accessibility and scalability.

● Requires students to physically reach the machine to mark attendance.

3.Time-Consuming Process:

● Manually entering attendance or waiting in line for biometric machines consumes valuable class time.

● Inefficiencies arise when handling large student populations.

4.Privacy Concerns:

● Biometric data storage raises privacy concerns and requires stringent compliance with data protection regulations.

● Potential legal challenges and concerns about unauthorized access.

5.Installation and Downtime:

● Implementing biometric machines involves installation processes and potential downtime.

● Delays in system setup and occasional maintenance interruptions impact regular operations.

Proposed System: Mobile Attendance System

Advantages:

1.Cost-Effective Solution:

● The proposed system leverages existing mobile devices, eliminating the need for costly infrastructure.

● Significantly reduces financial barriers, making it accessible to a broader range of educational institutions.

2.Universal Accessibility:

● The proposed system operates on mobile devices, ensuring universal accessibility for students.

● Overcomes limitations of traditional systems by enabling attendance marking from any location, enhancing ease of use.

3.Efficient and Time-Saving:

● The proposed system streamlines attendance marking, minimizing disruptions to the class schedule.

● Maximizes efficiency by reducing the time students spend on attendance-related activities.

4.Enhanced Privacy Measures:

● Fingerprint data is stored locally on the mobile device, mitigating privacy concerns.

● Adheres to data protection regulations by limiting server access to unique encrypted IDs, ensuring enhanced privacy and security.

5.Swift Implementation:

● The proposed system's reliance on mobile devices allows for swift implementation with minimal setup and downtime.

● Reduces disruptions to academic activities and ensures consistent functionality.

This comparative analysis underscores the advantages of transitioning from the traditional student attendance management system to the proposed "Mobile Attendance System," offering a modern, efficient, and cost-effective solution.

CHAPTER - 4

FEASABILITY STUDY:

1.Financial Feasibility:

Cost-Benefit Analysis:

● As the system primarily relies on existing mobile devices for student attendance, there is minimal financial investment required.

● Cost savings are evident due to the absence of significant equipment or infrastructure costs.

Financial Viability:

● The project is financially viable, aligning with a cost-effective approach that leverages existing resources.

2.Technical Feasibility:

Infrastructure Assessment:

● The system operates seamlessly without the need for additional equipment or infrastructure, relying on the ubiquity of mobile devices.

● The simplicity of operation and compatibility with various mobile devices enhance technical feasibility.

● Availability of fingerprint data on the server is ensured by storing encrypted unique IDs rather than actual fingerprint images.

3.Operational Feasibility:

Impact on Operations:

● The system introduces minimal disruption to existing operations, leveraging the familiarity and accessibility of mobile devices.

● Operational feasibility is high, given the ease of use and adaptability for both students and lecturers.

4.Legal and Regulatory Feasibility:

Compliance Check:

● The system adheres to legal and regulatory requirements by prioritizing data privacy and security.

● The absence of direct access to fingerprint data on the server aligns with privacy regulations.

5.Schedule Feasibility:

Timeline Definition:

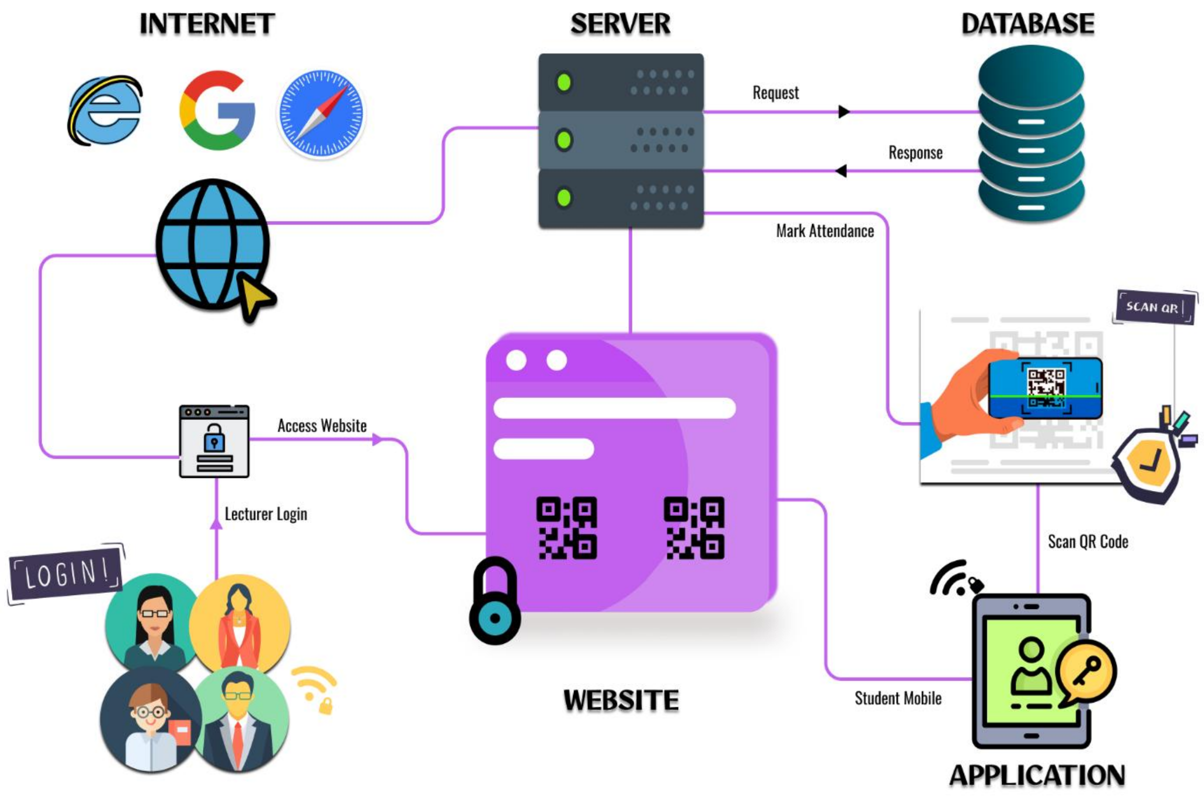
● With a streamlined approach and minimal dependencies, the project timeline is optimized for efficient development, testing, and deployment.

● Schedule feasibility is high, allowing for swift implementation.

This feasibility study affirms the practicality and viability of the "Mobile Attendance System," emphasizing its minimal financial requirements, ease of operation, and adherence to legal and regulatory standards. The system's technical simplicity and reliance on mobile devices ensure widespread accessibility, making it an efficient and cost-effective solution for attendance tracking.

CHAPTER – 5

BLUEPRINT



The above diagram describes the workflow of our project.

First the lecturers will Login into the website which opens the main website with QR codes.

Then the student will be scanning those QR codes with the help of the Mobile Application.

Which is also secured because without correct authentication, it will not allow a student to mark the attendance.

Here every single time we will be accessing the server for the data.

And server has the access to the Database.

The lecturers will also be able to access other webpages related to students' data.

They have the capability to Add new students data, Mark the attendance; present or absent.

CHAPTER – 6

SETUP THE ENVIRONMENT

Install the Eclipse IDE, Android Studio, Xampp Server and MySQL.

Follow their tutorials online on how to download and setup evething.

Eclipse IDE:

while setting up the eclipse choose Java EE and after that create the Dynamic web project. In this project you can add all these required files and run on the Apache tomcat Server.

Also don’t forget to download the Apache tomcat server in the eclipse IDE.

Android Studio:

While using the app you must keep the server OnLive so that the app can send and get request and response respectively.

Then set up the Android Studio by downloading the SDK components and Virtual device or you can use the physical device instead.

Then create a new project and add the required files in that or directly open the app in it.

MySQL:

set the username and password for the SQL. and create the database.

In this you have to create some tables which are necessary for the servlets in the server-side files.

Brief description is given in the Database Setup.(CHAPTER – 7)

Xampp Server:

we use this Xampp server to run the php files by using which are developing a website for the Lecturers. Here we interact with the Database too.

We are using the PHP because it is very easy to use for Dynamic web projects.

CHAPTER – 7

DATABASE SETUP.

For this project we require 4 main tables,

1. Students table
2. student attendance table
3. teachers table
4. passwords reset table

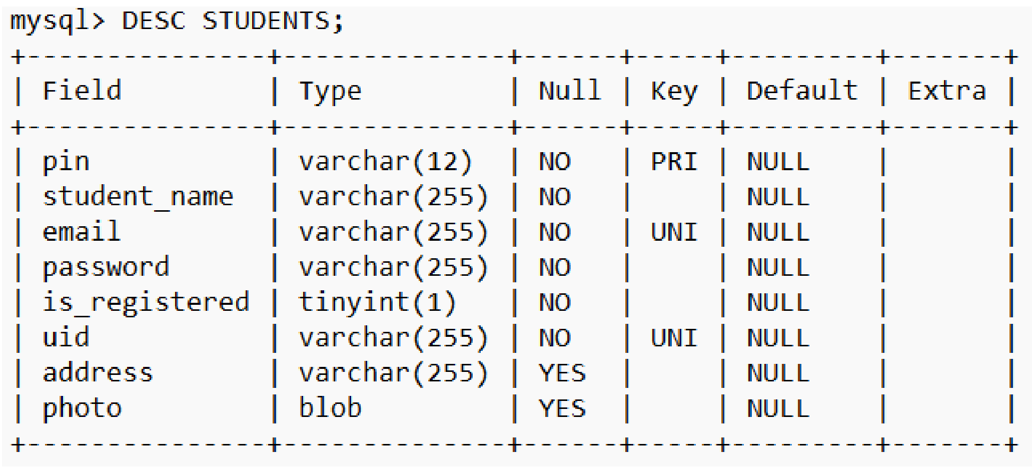
- In students table we will be having details about the student like PIN, NAME,EMAIL,PASSOWRD,ADDRESS,UID,PHOTO etc.,

- In student attendance table we will be referring the PIN in the students table, which will be the foreign key in this table. And has columns like TIME,DATE,PIN,NAME,ID,IS_PRESENT.

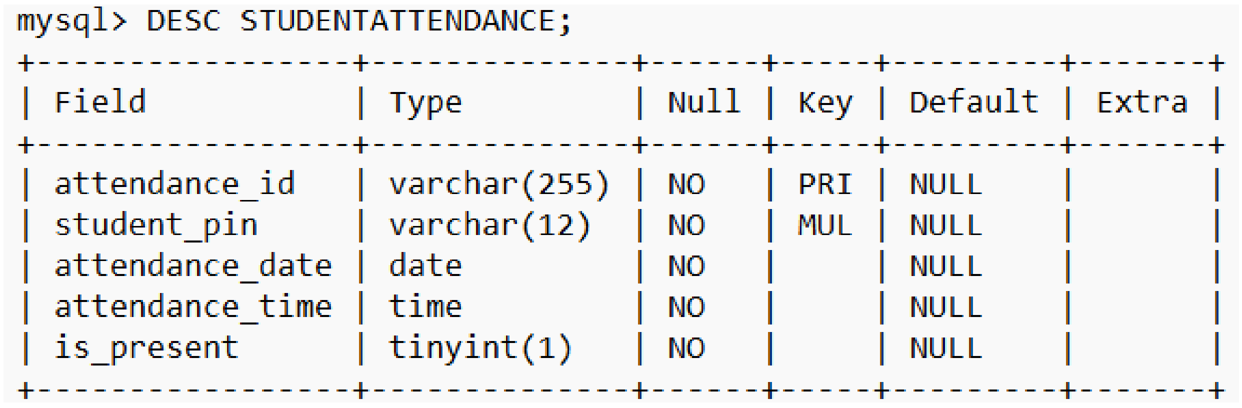
- In teachers table we will be just storing the email and password of the lecturer who will be having access to this all data.

- In password reset table we will be using to store the changed passwords of the students.

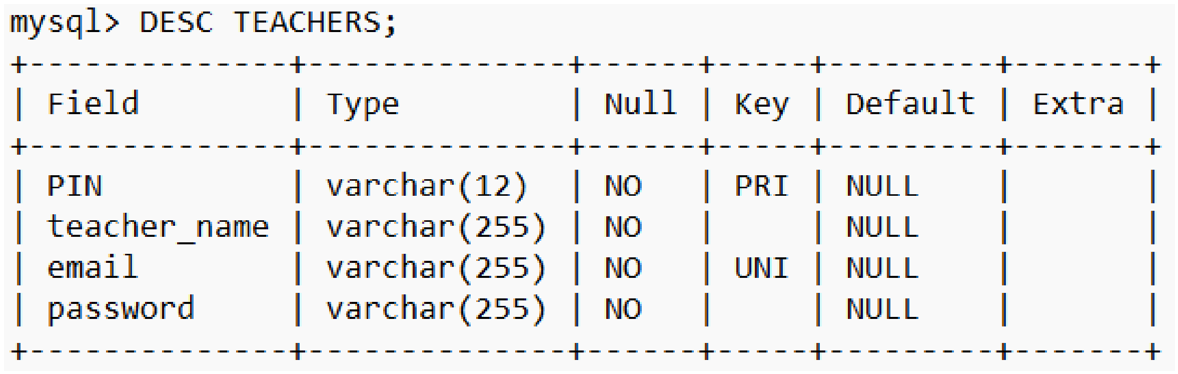
The description of the tables is as follows.



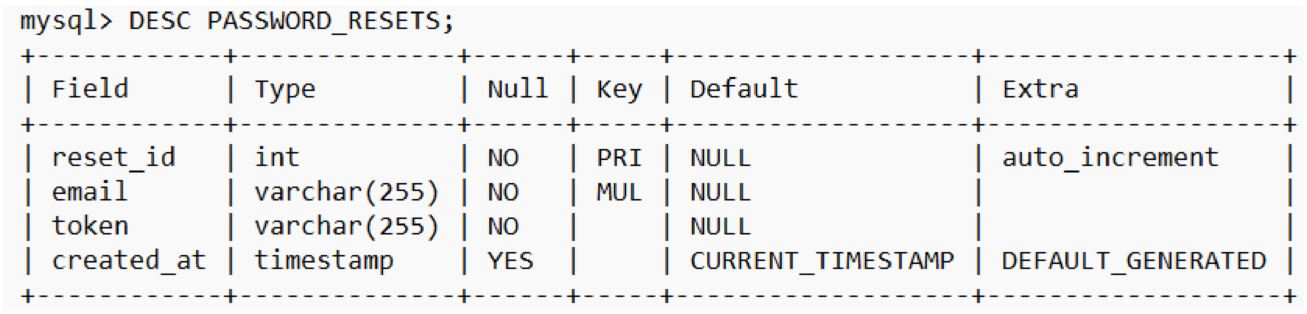
```
mysql> DESC STUDENTS;
+---------------+--------------+------+-----+---------+-------+
| Field         | Type         | Null | Key | Default | Extra |
+---------------+--------------+------+-----+---------+-------+
| pin           | varchar(12)  | NO   | PRI | NULL    |       |
| student_name  | varchar(255) | NO   |     | NULL    |       |
| email         | varchar(255) | NO   | UNI | NULL    |       |
| password      | varchar(255) | NO   |     | NULL    |       |
| is_registered | tinyint(1)   | NO   |     | NULL    |       |
| uid           | varchar(255) | NO   | UNI | NULL    |       |
| address       | varchar(255) | YES  |     | NULL    |       |
| photo         | blob         | YES  |     | NULL    |       |
+---------------+--------------+------+-----+---------+-------+
```



```
mysql> DESC STUDENTATTENDANCE;
+-----------------+--------------+------+-----+---------+-------+
| Field           | Type         | Null | Key | Default | Extra |
+-----------------+--------------+------+-----+---------+-------+
| attendance_id   | varchar(255) | NO   | PRI | NULL    |       |
| student_pin     | varchar(12)  | NO   | MUL | NULL    |       |
| attendance_date | date         | NO   |     | NULL    |       |
| attendance_time | time         | NO   |     | NULL    |       |
| is_present      | tinyint(1)   | NO   |     | NULL    |       |
+-----------------+--------------+------+-----+---------+-------+
```



```
mysql> DESC TEACHERS;
+--------------+--------------+------+-----+---------+-------+
| Field        | Type         | Null | Key | Default | Extra |
+--------------+--------------+------+-----+---------+-------+
| PIN          | varchar(12)  | NO   | PRI | NULL    |       |
| teacher_name | varchar(255) | NO   |     | NULL    |       |
| email        | varchar(255) | NO   | UNI | NULL    |       |
| password     | varchar(255) | NO   |     | NULL    |       |
+--------------+--------------+------+-----+---------+-------+
```



```
mysql> DESC PASSWORD_RESETS;
+------------+--------------+------+-----+-------------------+-------------------+
| Field      | Type         | Null | Key | Default           | Extra             |
+------------+--------------+------+-----+-------------------+-------------------+
| reset_id   | int          | NO   | PRI | NULL              | auto_increment    |
| email      | varchar(255) | NO   | MUL | NULL              |                   |
| token      | varchar(255) | NO   |     | NULL              |                   |
| created_at | timestamp    | YES  |     | CURRENT_TIMESTAMP | DEFAULT_GENERATED |
+------------+--------------+------+-----+-------------------+-------------------+
```

CHAPTER – 8

LANGUAGES USED

Server side:

On the server side we used the **Java servlets** to handle the requests and responses, nothing but back-end. which is very easy and simple to use.

And **Jsp** pages for those servlets.(front-end)

And also some front end development was done using **Html, CSS and Js.**

We used **php** to develop the webpages for the lecturer to Login and perform their actions, and it was integrated within the html.

And we used CSS to provide styles to those pages.

This was all about languages used on the Server side.

Client Side:

Client side is nothing but the app code.

we used **Kotin** as our main language in developing the application.

And used **xml** to develop the User Interface for the application.

Kotlin is the official android studio's language which is very popular and mostly used to develop the android applications.

It is similar to Java but the only difference is it avoids null pointer exceptions.

Php is a server-side scripting language which is easy to learn and use.

and also provides easy access to the database.

CHAPTER – 9

CODING

Here as we know we have to do both server side and client-side setup to implement the project,

SERVER-SIDE CODE:

Student Login Handling servlet:

```
--StudentLoginServlet.java
package ram;

import java.*;
import javax.*;
import org.json.*;

@WebServlet("/studentLogin")
public class StudentLoginServlet extends HttpServlet {
    // JDBC URL, username, and password of MySQL server
    private static final String JDBC_URL = "jdbc:mysql://localhost:3306/voldemort";
    private static final String DB_USER = "root";
    private static final String DB_PASSWORD = "3223";
    private static final int TOTAL_DAYS = 100;  // Change this value as needed
    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws
IOException {
        // Parse student email and password from request parameters
        String email = request.getParameter("email");
        String password = request.getParameter("password");
        String uid = request.getParameter("UID");
        String message = request.getParameter("message");

        // Check the value of the "message" parameter
        if (message != null && message.equals("store")) {
            // If the "message" is "store," store the UID in the table
            boolean stored = storeUID(email, uid);
            if (stored) {
                      String studentPin = authenticateStudent(email, password);

                  if (studentPin != null) {
                      // Retrieve student's attendance data and calculate attendance percentage
                      JSONObject responseData = getStudentAttendanceData(studentPin);

                      // Set the response content type to JSON
                      response.setContentType("application/json");

                      // Write the student details as JSON to the response
                      response.getWriter().write(responseData.toString());
                  } else {
                    // Handle authentication failure
                    response.setStatus(HttpServletResponse.SC_UNAUTHORIZED);
                    response.getWriter().write("Authentication failed");
                }
                System.out.println("UID stored successfully");
            } else {
                     System.out.println("Failed to store UID");

            }
        } else if(message.equals("check")){

            // Authenticate the student and get student PIN
            String studentPin = authenticateStudentwithUid(email, password,uid);

            if (studentPin != null) {
                // Retrieve student's attendance data and calculate attendance percentage
                JSONObject responseData = getStudentAttendanceData(studentPin);

                // Set the response content type to JSON
                response.setContentType("application/json");

                // Write the student details as JSON to the response
                response.getWriter().write(responseData.toString());
            } else {
                // Handle authentication failure
                response.setStatus(HttpServletResponse.SC_UNAUTHORIZED);
                response.getWriter().write("Authentication failed");
            }
        }else {
            response.setContentType("application/json");
            response.getWriter().write("Login to your account big guy");
        }
    }
    private boolean storeUID(String email, String uid) {
        try {
            // Establish a connection to the MySQL database
            Connection connection = DriverManager.getConnection(JDBC_URL, DB_USER,
DB_PASSWORD);

            // Check if the student exists and their email matches
            String emailCheckQuery = "SELECT pin FROM Students WHERE email = ?";
            PreparedStatement emailCheckStatement = connection.prepareStatement(emailCheckQuery);
            emailCheckStatement.setString(1, email);

            ResultSet emailCheckResult = emailCheckStatement.executeQuery();

            if (emailCheckResult.next()) {
                // If the email matches, update the UID
                String updateUIDQuery = “UPDATE Students SET UID = ? WHERE email = ?”;
                PreparedStatement updateUIDStatement =
connection.prepareStatement(updateUIDQuery);
                updateUIDStatement.setString(1, uid);
                updateUIDStatement.setString(2, email);

                int rowsAffected = updateUIDStatement.executeUpdate();
                updateUIDStatement.close();
                connection.close();

                return rowsAffected > 0;
            }

            emailCheckResult.close();
            emailCheckStatement.close();
            connection.close();
        } catch (SQLException e) {
            e.printStackTrace();
        }

        return false;
    }


    private String authenticateStudent(String email, String password) {
        String studentPin = null;

        try {
            // Establish a connection to the MySQL database
            Connection connection = DriverManager.getConnection(JDBC_URL, DB_USER,
DB_PASSWORD);

            // Create a prepared statement to authenticate the student
            String query = “SELECT pin FROM Students WHERE email = ? AND password = ? “;
            PreparedStatement preparedStatement = connection.prepareStatement(query);
            preparedStatement.setString(1, email);
            preparedStatement.setString(2, password);

            ResultSet resultSet = preparedStatement.executeQuery();

            if (resultSet.next()) {
                studentPin = resultSet.getString(“pin”);
            }

            resultSet.close();
            preparedStatement.close();
            connection.close();
        } catch (SQLException e) {
            e.printStackTrace();
        }

        return studentPin;
    }
    private String authenticateStudentwithUid(String email, String password, String uid) {
        String studentPin = null;

        try {
            // Establish a connection to the MySQL database
            Connection connection = DriverManager.getConnection(JDBC_URL, DB_USER,
DB_PASSWORD);

            // Create a prepared statement to authenticate the student
            String query = "SELECT pin FROM Students WHERE email = ? AND password = ? AND
uid = ? ";
            PreparedStatement preparedStatement = connection.prepareStatement(query);
            preparedStatement.setString(1, email);
            preparedStatement.setString(2, password);
            preparedStatement.setString(3, uid);

            ResultSet resultSet = preparedStatement.executeQuery();

            if (resultSet.next()) {
                studentPin = resultSet.getString("pin");
            }

            resultSet.close();
            preparedStatement.close();
            connection.close();
        } catch (SQLException e) {
            e.printStackTrace();
        }

        return studentPin;
    }
    private JSONObject getStudentAttendanceData(String studentPin) {
        JSONObject studentData = new JSONObject();

        int presentDays = 0;
        try {
            // Establish a connection to the MySQL database
            Connection connection = DriverManager.getConnection(JDBC_URL, DB_USER,
DB_PASSWORD);
```

```
        // Retrieve student details from the student’s table
        String studentDetailsQuery = “SELECT * FROM Students WHERE pin = ?”;
        PreparedStatement studentDetailsStatement =
connection.prepareStatement(studentDetailsQuery);
        studentDetailsStatement.setString(1, studentPin);

        ResultSet studentDetailsResultSet = studentDetailsStatement.executeQuery();

        if (studentDetailsResultSet.next()) {
            studentData.put(“student_pin”, studentDetailsResultSet.getString(“pin”));
            studentData.put(“student_name”, studentDetailsResultSet.getString(“student_name”));
            studentData.put(“email”, studentDetailsResultSet.getString(“email”));
            studentData.put(“password”, studentDetailsResultSet.getString(“password”));
        }

        studentDetailsResultSet.close();
        studentDetailsStatement.close();

        // Retrieve student attendance data from the StudentAttendance table
        String attendanceDataQuery = “SELECT attendance_date, attendance_time, is_present
FROM StudentAttendance WHERE student_pin = ?”;
        PreparedStatement attendanceDataStatement =
connection.prepareStatement(attendanceDataQuery);
        attendanceDataStatement.setString(1, studentPin);

        ResultSet attendanceDataResultSet = attendanceDataStatement.executeQuery();

        JSONArray attendanceArray = new JSONArray();

        while (attendanceDataResultSet.next()) {
            JSONObject attendanceObject = new JSONObject();
            attendanceObject.put(“attendance_date”,
attendanceDataResultSet.getString(“attendance_date”));
            attendanceObject.put(“attendance_time”,
attendanceDataResultSet.getString(“attendance_time”));
            attendanceObject.put(“is_present”, attendanceDataResultSet.getBoolean(“is_present”));

            attendanceArray.put(attendanceObject);

            if (attendanceDataResultSet.getBoolean(“is_present”)) {
                presentDays++;
            }
        }

        attendanceDataResultSet.close();
        attendanceDataStatement.close();
        connection.close();
```

```
            studentData.put("attendance_data", attendanceArray);

            // Calculate the attendance percentage
            double attendancePercentage = (double) presentDays / TOTAL_DAYS * 100;
            studentData.put("attendance_percentage", attendancePercentage);

        } catch (SQLException e) {
            e.printStackTrace();
        }

        return studentData;
    }
}
```

Fetching Data into the App from the server servlet:

--DataFetchingServlet.java

```
package ram;
import java.*;
import javax.*;
import org.json.*;

@WebServlet("/DataFetchingServlet")
public class DataFetchingServlet extends HttpServlet {
             // JDBC URL, username, and password of MySQL server
    private static final String JDBC_URL = "jdbc:mysql://localhost:3306/voldemort";
    private static final String DB_USER = "root";
    private static final String DB_PASSWORD = "3223";
    private static final int TOTAL_DAYS = 100;  // Change this value as needed

    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws
IOException {
        // Parse student email and password from request parameters
        String email = request.getParameter("email");
        String password = request.getParameter("password");

        String studentPin = authenticateStudent(email, password);

        if (studentPin != null) {
            // Retrieve student's attendance data and calculate attendance percentage
            JSONObject responseData = getStudentAttendanceData(studentPin);

            // Set the response content type to JSON
            response.setContentType("application/json");

            // Write the student details as JSON to the response
            response.getWriter().write(responseData.toString());
        } else {
            // Handle authentication failure
            response.setStatus(HttpServletResponse.SC_UNAUTHORIZED);
            response.getWriter().write("Authentication failed");
        }
    }

    private String authenticateStudent(String email, String password) {
        String studentPin = null;

        try {
            // Establish a connection to the MySQL database
            Connection connection = DriverManager.getConnection(JDBC_URL, DB_USER,
DB_PASSWORD);

            // Create a prepared statement to authenticate the student
            String query = "SELECT pin FROM Students WHERE email = ? AND password = ? ";
            PreparedStatement preparedStatement = connection.prepareStatement(query);
            preparedStatement.setString(1, email);
            preparedStatement.setString(2, password);

            ResultSet resultSet = preparedStatement.executeQuery();

            if (resultSet.next()) {
                studentPin = resultSet.getString("pin");
            }

            resultSet.close();
            preparedStatement.close();
            connection.close();
        } catch (SQLException e) {
            e.printStackTrace();
        }

        return studentPin;
    }
    private JSONObject getStudentAttendanceData(String studentPin) {
        JSONObject studentData = new JSONObject();

        int presentDays = 0;
        try {
            // Establish a connection to the MySQL database
            Connection connection = DriverManager.getConnection(JDBC_URL, DB_USER,
DB_PASSWORD);

            // Retrieve student details from the Students table
            String studentDetailsQuery = "SELECT * FROM Students WHERE pin = ?";
            PreparedStatement studentDetailsStatement =
connection.prepareStatement(studentDetailsQuery);
            studentDetailsStatement.setString(1, studentPin);

            ResultSet studentDetailsResultSet = studentDetailsStatement.executeQuery();

            if (studentDetailsResultSet.next()) {
                studentData.put("student_pin", studentDetailsResultSet.getString("pin"));
                studentData.put("student_name", studentDetailsResultSet.getString("student_name"));
                studentData.put("email", studentDetailsResultSet.getString("email"));
                studentData.put("password", studentDetailsResultSet.getString("password"));
            }

            studentDetailsResultSet.close();
            studentDetailsStatement.close();

            // Retrieve student attendance data from the StudentAttendance table
            String attendanceDataQuery = "SELECT attendance_date, attendance_time, is_present
FROM StudentAttendance WHERE student_pin = ?";
            PreparedStatement attendanceDataStatement =
connection.prepareStatement(attendanceDataQuery);
            attendanceDataStatement.setString(1, studentPin);

            ResultSet attendanceDataResultSet = attendanceDataStatement.executeQuery();

            JSONArray attendanceArray = new JSONArray();

            while (attendanceDataResultSet.next()) {
                JSONObject attendanceObject = new JSONObject();
                attendanceObject.put("attendance_date",
attendanceDataResultSet.getString("attendance_date"));
                attendanceObject.put("attendance_time",
attendanceDataResultSet.getString("attendance_time"));
                attendanceObject.put("is_present", attendanceDataResultSet.getBoolean("is_present"));

                attendanceArray.put(attendanceObject);

                if (attendanceDataResultSet.getBoolean("is_present")) {
                    presentDays++;
                }
            }

            attendanceDataResultSet.close();
            attendanceDataStatement.close();
            connection.close();

            studentData.put("attendance_data", attendanceArray);
```

```
            // Calculate the attendance percentage
            double attendancePercentage = (double) presentDays / TOTAL_DAYS * 100;
            studentData.put("attendance_percentage", attendancePercentage);

        } catch (SQLException e) {
            e.printStackTrace();
        }

        return studentData;
    }
}
```

Password Reset Handling Servlet:

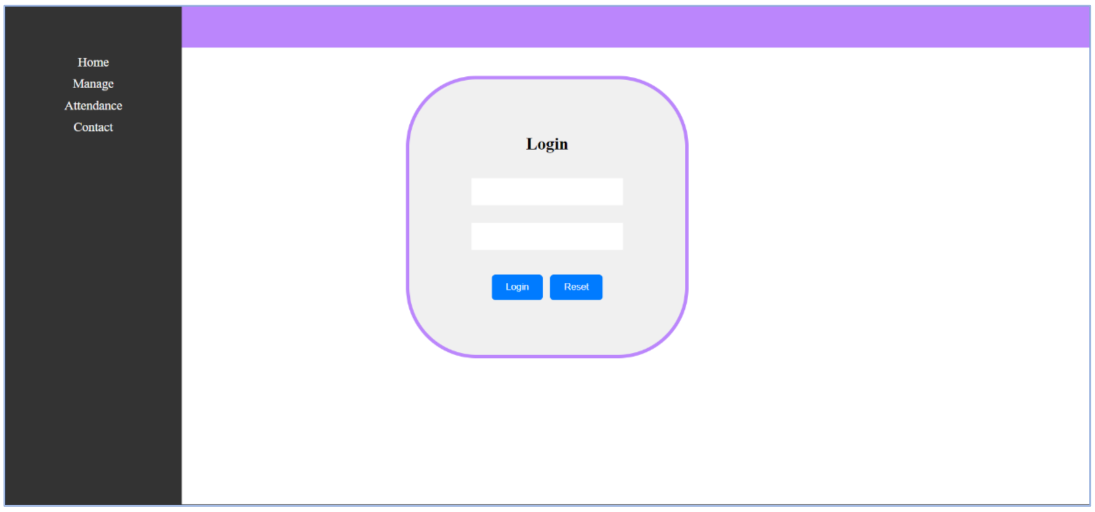
```
--PasswordResetServlet.java
package ram;
import java.*;
import javax.*;
import org.json.*;

@WebServlet("/password-reset")
public class PasswordResetServlet extends HttpServlet {
    private static final String DB_URL = "jdbc:mysql://localhost:3306/voldemort";
    private static final String DB_USER = "root";
    private static final String DB_PASSWORD = "3223";

    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
        String email = request.getParameter("email");
        String resetToken = generateResetToken();

        try {
            Connection connection = DriverManager.getConnection(DB_URL, DB_USER,
DB_PASSWORD);
            String sql = "SELECT email FROM students WHERE email = ?";
            PreparedStatement statement = connection.prepareStatement(sql);
            statement.setString(1, email);
            ResultSet result = statement.executeQuery();

            if (result.next()) {
                // Email exists in the database.
                // Store the reset token in the database for this email.
                String insertSql = "INSERT INTO password_resets (email, token) VALUES (?, ?)";
                PreparedStatement insertStatement = connection.prepareStatement(insertSql);
                insertStatement.setString(1, email);
                insertStatement.setString(2, resetToken);
                insertStatement.executeUpdate();
                // Send a reset email with the token to the user.
                sendResetEmail(email, resetToken);

                response.sendRedirect(“password-reset-sent.jsp”);
            } else {
                // Handle the case where the email doesn’t exist.
                response.sendRedirect(“password-reset-error.jsp”);
            }

            connection.close();
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }

    private String generateResetToken() {
        SecureRandom random = new SecureRandom();
        byte[] bytes = new byte[16];
        random.nextBytes(bytes);
        return Base64.getUrlEncoder().withoutPadding().encodeToString(bytes);
    }

    private void sendResetEmail(String recipientEmail, String resetToken) {
        // Email configuration
        String host = “smtp.gmail.com”; // Your SMTP server host
        String port = “587”; // Port for sending emails
        String username = “resettingemailforstudent@gmail.com”; // Your email address
        String password = “yjux tjsu fiua lxpq”; // Your email password

        // Set email properties
        Properties properties = new Properties();
        properties.put(“mail.smtp.auth”, “true”);
        properties.put(“mail.smtp.starttls.enable”, “true”);
        properties.put(“mail.smtp.host”, host);
        properties.put(“mail.smtp.port”, port);

        // Create a session with the email server
        Session session = Session.getInstance(properties, new javax.mail.Authenticator() {
            protected javax.mail.PasswordAuthentication getPasswordAuthentication() {
                return new javax.mail.PasswordAuthentication(username, password);
            }
        });

        try {
            // Create a message
            MimeMessage message = new MimeMessage(session);
```

```
            message.setFrom(new InternetAddress(username));
            message.addRecipient(Message.RecipientType.TO, new InternetAddress(recipientEmail));
            message.setSubject("Password Reset Request");

            // Email content
            MimeBodyPart textPart = new MimeBodyPart();
            String resetLink = "http://192.168.83.237:8080/project_portable/reset-password.jsp?token="
+ resetToken;
            textPart.setText("To reset your password, click the following link: " + resetLink);

            // Attach the text part to the message
            MimeMultipart multipart = new MimeMultipart();
            multipart.addBodyPart(textPart);
            message.setContent(multipart);

            // Send the email
            Transport.send(message);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

User Interface Code:

Login page:

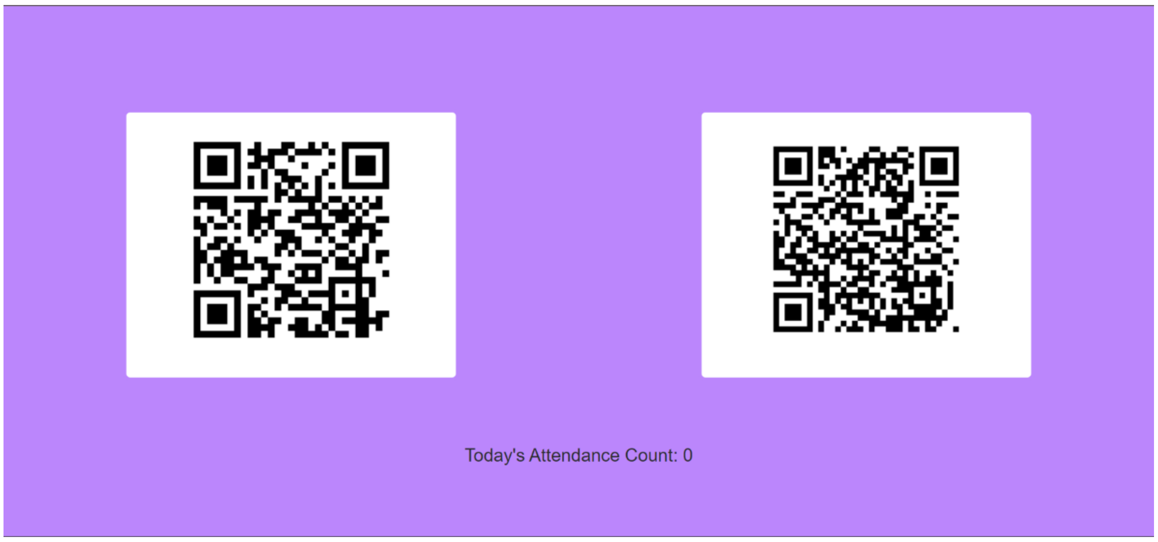
```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
    <link rel="stylesheet" href="stylelogin.css">
    <script src="https://kit.fontawesome.com/afc3acaa52.js" crossorigin="anonymous"></script>
    <script src="indexLogin.js"></script>

</head>
<body>
    <div class="cont" >
    <div class="menu-icon" onclick="toggleNav()">
        <div class="bar"></div>
        <div class="bar"></div>
        <div class="bar"></div>
     </div>

     <nav class="navigation-pane" id="navPane">
        <ul>
            <li><a href="#">Home</a></li>
            <li><a href="http://localhost/Student_Management/index.php">Manage</a></li>
            <li><a
href=”http://localhost/Student_Management/AttendanceIndex.php”>Attendance</a></li>
            <li><a href=”#”>Contact</a></li>
        </ul>
     </nav>
    </div>
    <div class=”content”>
        <div class=”login-text”>
            <h1>Login</h1>
        </div>
        <div class=”boxes”>
        <form action=”Javaservlet” method=”post” class=”form”>
        <div class=”email-box” >
        <input type=”email” name=”mail”>
        </div>
        <div class=”password-box”>
        <input type=”password” name=”password”>
        </div>
        </div>
        <div class=”buttons”>
        <input type=”submit” value=”Login”> <input type=”reset”>
        </div>
        </form>
    </div>
</body>
</html>
```



Scanner UI after Successful LOGIN:

```
<!DOCTYPE html>
<html lang=”en”>
<head>
    <meta charset=”UTF-8”>
    <meta name=”viewport” content=”width=device-width, initial-scale=1.0”>
    <title>Student Attendance Count</title>
    <style>
        body {
            background-color: #BB86FC; /* Light violet color */
            color: #292b2c; /* Dark color for text */
            font-family: ‘Arial’, sans-serif;
            margin: 0;
            padding: 0;
            text-align: center;
        }

        #qr-code-container {
            display: flex;
            justify-content: space-around;
            align-items: center;
            height: 70vh;
            margin-top: 10vh;
        }

        .qr-code {
            background-color: #ffffff; /* White color for QR code container */
            border: 5px solid #BB86FC; /* Light violet border */
            border-radius: 10px;
            padding: 20px;
            margin: 10px;
            width: 70%; /* Set the width to 70% of the container width */
            max-width: 400px; /* Set a maximum width to avoid overflow */
        }

        .qr-code img {
            width: 100%; /* Make the QR code fill 100% of the container */
            max-width: 300px; /* Set a maximum width for the QR code image */
            height: auto; /* Maintain the aspect ratio */
            margin-bottom: 10px; /* Add a margin below the QR code */
        }

        #attendanceCount {
            font-size: 24px;
            margin-top: 20px;
        }
    </style>
</head>
<body onload=”fetchQr(‘wifi’); fetchQr(‘attendance’); fetchAttendanceCount();”>
    <div id=”qr-code-container”>
        <div id=”wifi-qrcode” class=”qr-code”></div>
        <div id=”attendance-qrcode” class=”qr-code”></div>
    </div>

    <div id=”attendanceCount”>
        <!—The attendance count will be displayed here →
    </div>

    <script src=”https://cdn.rawgit.com/davidshimjs/qrcodejs/gh-pages/qrcode.min.js”></script>
    <script>
         // Function to fetch and update attendance count
        function fetchAttendanceCount(callback) {
            var xhr = new XMLHttpRequest();
            xhr.open(“GET”, “AttendanceCounter”, true);
            xhr.onreadystatechange = function () {
```

```
            if (xhr.readyState === 4 && xhr.status === 200) {
                var response = JSON.parse(xhr.responseText);
                if (“presentCount” in response) {
                    var attendanceCount = response.presentCount;
                    document.getElementById(“attendanceCount”).innerText = “Today’s Attendance
Count: “ + attendanceCount;

                    // Invoke the callback with the updated attendance count
                    if (typeof callback === ‘function’) {
                        callback(attendanceCount);
                    }
                } else {
                    document.getElementById(“attendanceCount”).innerText = “No attendance data
found for today.”;
                }
            }
        };
        xhr.send();
    }

    // Function to fetch QR code
    function fetchQr(type) {
        var xhr = new XMLHttpRequest();
        xhr.open(“GET”, “GenerateQRCodeServlet?type=” + type, true);
        xhr.responseType = ‘arraybuffer’;
        xhr.onload = function () {
            var byteArray = new Uint8Array(xhr.response);
            var binaryData = String.fromCharCode.apply(null, byteArray);
            var base64Data = btoa(binaryData);

            var imgElement = document.createElement(“img”);
            imgElement.src = “data:image/png;base64,” + base64Data;
            imgElement.alt = type.toUpperCase() + “ QR code”;

            var containerDiv = document.getElementById(type + “-qrcode”);
            containerDiv.innerHTML = ‘’; // Clear the container before appending the image
            containerDiv.appendChild(imgElement);
            // Dispatch a custom event to signal that a QR code has been scanned
            var event = new Event(‘qrCodeScanned’);
            document.dispatchEvent(event);
        };
        xhr.send();
    }
    // Listen for the custom event
    document.addEventListener(‘qrCodeScanned’, function () {
        // Update the attendance count after fetching the QR code
        fetchAttendanceCount(function (attendanceCount) {
            // Additional logic to handle the updated attendance count if needed
            console.log(“Updated attendance count:”, attendanceCount);
        });
    });
    </script>
</body>
</html>
```



QR code generating Servlet:

```
package ram;
import java.*;
import javax.*;
import org.json.*;
import com.google.*;

@WebServlet("/GenerateQRCodeServlet")
public class GenerateQRCodeServlet extends HttpServlet {
    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws
IOException {
        String type = request.getParameter("type");

        if ("wifi".equals(type)) {
            generateAndSendWifiQRCode("vivo 1901", "1234567890", response);
        } else if ("attendance".equals(type)) {

generateAndSendAttendanceQRCode("http://192.168.0.152:8080/project_portable/markAttendance"
, response);
        }
    }

    private void generateAndSendWifiQRCode(String ssid, String password, HttpServletResponse
response) throws IOException {
        String wifiData = "WIFI:T:WPA;S:" + ssid + ";P:" + password + ";;";
        generateAndSendQRCode(wifiData, response);
    }

    private void generateAndSendAttendanceQRCode(String url, HttpServletResponse response)
throws IOException {
        generateAndSendQRCode(url, response);
    }
    private void generateAndSendQRCode(String data, HttpServletResponse response) throws
IOException {
        try {
            response.setContentType("image/png");
            OutputStream = response.getOutputStream();
            BitMatrix bitMatrix = generateQRCodeMatrix(data);
            MatrixToImageWriter.writeToStream(bitMatrix, "PNG", outputStream);
            outputStream.flush();
            outputStream.close();
        } catch (WriterException e) {
            e.printStackTrace();
        }
    }

    private BitMatrix generateQRCodeMatrix(String data) throws WriterException {
        QRCodeWriter qrCodeWriter = new QRCodeWriter();
        return qrCodeWriter.encode(data, BarcodeFormat.QR_CODE, 200, 200,
                java.util.Collections.singletonMap(EncodeHintType.MARGIN, 2));
    }
}
```

Attendance counting servlet: (Total students present)

```
 package ram;

package ram;
import java.*;
import javax.*;
import org.json.*;

@WebServlet("/AttendanceCounter") // Define your servlet mapping
public class AttendanceCounter extends HttpServlet {
    private static final long serialVersionUID = 1L;

    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws
IOException {
        String url = "jdbc:mysql://localhost:3306/voldemort";
        String username = "root";
        String password = "3223";

        // Create a JSON object to hold the response
        JSONObject jsonResponse = new JSONObject();

        try (Connection connection = DriverManager.getConnection(url, username, password)) {
            // Get today's date in the format used in the database (assuming yyyy-MM-dd)
            LocalDate today = LocalDate.now();
            DateTimeFormatter dateFormatter = DateTimeFormatter.ofPattern("yyyy-MM-dd");
            String todayFormatted = today.format(dateFormatter);

            // Prepare and execute the SQL query
            String sql = "SELECT COUNT(*) AS present_count FROM studentattendance WHERE
attendance_date = ? AND is_present = 1";
```

```
            try (PreparedStatement statement = connection.prepareStatement(sql)) {
                statement.setString(1, todayFormatted);
                try (ResultSet resultSet = statement.executeQuery()) {
                    if (resultSet.next()) {
                        int presentCount = resultSet.getInt("present_count");
                        jsonResponse.put("presentCount", presentCount); // Add attendance count to the
response
                    }
                }
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }

        // Set the response content type to JSON
        response.setContentType("application/json");

        // Write the JSON response to the client
        response.getWriter().write(jsonResponse.toString());
    }
}
```

APP CODE:
(ANDROID STUDIO)

MainActivity.kt

The start of the app is with this activity. In this we have a navigation menu on the side and a scanner on the top and list of other activities in this navigation menu. When you click them the new activities are started using the intent and also they are all interconnected using this intent. Here we send and receive the data using the intent putextra() and getString() methods.

```
package com.example.ex10

import android.*
import 31ebView.*
import com.google.zxing.*
import kotlinx.*
import org.json.*
import java.*

class MainActivity2 : AppCompatActivity() {
    private lateinit var drawerLayout: DrawerLayout
    private lateinit var navigationView: NavigationView
    private lateinit var drawerToggle: ActionBarDrawerToggle
    private lateinit var scanLauncher: ActivityResultLauncher<Intent>
    private lateinit var loginLauncher: ActivityResultLauncher<Intent>
    private lateinit var textView5: TextView
    private lateinit var textView2: TextView
    private var loginData: String? = null
    private var login_url: String? = null
    private var email: String? = null
    private var password: String? = null
    private var loggedIn = false
    private val REQUEST_CODE_SERVER_CONNECTIVITY = 2 // Define a request code for
ServerConnectivityActivity

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)

        drawerLayout = findViewById(R.id.drawer_layout)
        navigationView = findViewById(R.id.nav_view)
        drawerToggle = ActionBarDrawerToggle(this, drawerLayout, R.string.open, R.string.close)
        drawerLayout.addDrawerListener(drawerToggle)
        drawerToggle.syncState()
        supportActionBar?.setDisplayHomeAsUpEnabled(true)
        // Initialize the text views from your navigation drawer
        val headerView = navigationView.getHeaderView(0) // This gets the header view from the
navigation drawer
        textView5 = headerView.findViewById(R.id.textView5)
        textView2 = headerView.findViewById(R.id.textView2)
        val sharedPreferences = getSharedPreferences("MyAppPrefs", Context.MODE_PRIVATE)

        updateMenuVisibility()

        // Initialize variables from SharedPreferences
        initializeFromSharedPreferences()

            navigationView.setNavigationItemSelectedListener { item ->
                when (item.itemId) {
                    R.id.home -> {
                        Toast.makeText(this@MainActivity2, "Home selected", Toast.LENGTH_SHORT)
                            .show()
                    }
                    R.id.menu_login -> {
                            if (!loggedIn) {
                                val loginIntent = Intent(this, LoginStudentActivity::class.java)
                                loginLauncher.launch(loginIntent)
                            } else {
                                // Perform logout logic here
//                            loggedIn = false
                                updateMenuVisibility()
                            }
                        }
                    R.id.menu_check_attendance -> {
                        // Handle Check Attendance selection
                        val viewAttendanceIntent =
                            Intent(this, StudentAttendanceActivity::class.java)
                        viewAttendanceIntent.putExtra("student Data",
sharedPreferences.getString("login_response",null).toString())
                        startActivity(viewAttendanceIntent)
                    }
                    R.id.menu_check_result -> {
                        val viewResultActivityIntent =
                            Intent(this, ResultCheckingActivity::class.java)
                        startActivity(viewResultActivityIntent)
                    }
                    R.id.menu_logout -> {
                        // Handle Logout selection
                        if (loggedIn) {
                            // Clear the data except the unique ID
                            clearUserData()
                            loggedIn = false
//                            updateUI(null.toString())
                            resetUI()
                            invalidateOptionsMenu()
                        }
                    }
                }
                drawerLayout.closeDrawer(GravityCompat.START)
                true
            }
            scanLauncher =
                registerForActivityResult(ActivityResultContracts.StartActivityForResult()) { result ->
                    if (loggedIn) {
                        val resultCode = result.resultCode
                        val data = result.data

                        if (resultCode == RESULT_OK) {
                            val scanResult: IntentResult? =
                                IntentIntegrator.parseActivityResult(resultCode, data)

                            if (scanResult != null) {
                                if (scanResult.contents != null) {
                                    // Now, scanResult.contents contains the scanned URL
                                    val scannedURL = scanResult.contents

                                    if (scannedURL.startsWith("WIFI:")) {
                                        // Handle WiFi connection
                                        handleWifiConnection(scannedURL)
                                    } else {
                                        // Open the ServerConnectivityActivity
                                        val intent = Intent(
                                            this@MainActivity2,
                                            ServerConnectivityActivity::class.java
                                        )
                                        intent.putExtra("scanned_url", scannedURL)
                                        intent.putExtra("login_data", loginData)
                                        intent.putExtra("login_url", login_url)
                                        intent.putExtra("login_status", loggedIn)
                                        startActivityForResult(
                                            intent,
                                            REQUEST_CODE_SERVER_CONNECTIVITY
                                        )
                                    }
                                }
                            }
                        }
                    } else {
                        Toast.makeText( this@MainActivity2,"Login to mark attendance",
                            Toast.LENGTH_SHORT).show()
                    }
                }

            // Setup loginLauncher
            loginLauncher =
                registerForActivityResult(ActivityResultContracts.StartActivityForResult()) { result ->
                    if (result.resultCode == RESULT_OK) {
                        val data = result.data
                        if (data != null && data.hasExtra("Student Data")) {
                            val loginResult = data.getStringExtra("Student Data")
                            loginData = loginResult
                            Log.d("MainActivity2", loginData.toString())
                            login_url = data.getStringExtra("login_url")
                            Log.d("MainActivity2", login_url.toString())

                            updateUI(loginResult.toString())
                        }
                    } else {
                        Toast.makeText(this, "cannot find the code ", Toast.LENGTH_SHORT).show()
                    }
                }
        }
        // Function to handle WiFi connection
        private fun handleWifiConnection(wifiData: String) {
            // Extract WiFi credentials from the scanned data (assuming it's a simple format)
            val wifiDataParts = wifiData.split(';')
            val wifiSSID = wifiDataParts[0].substring(5) // Remove "WIFI:"
            val wifiPassword = wifiDataParts[1]

            // Create and configure a WifiConfiguration
            val wifiConfig = WifiConfiguration()
            wifiConfig.SSID = "\"" + wifiSSID + "\""
            wifiConfig.preSharedKey = "\"" + wifiPassword + "\""

            // Connect to the WiFi network
            val wifiManager = applicationContext.getSystemService(Context.WIFI_SERVICE) as
WifiManager
            val networkId = wifiManager.addNetwork(wifiConfig)
            if (networkId != -1) {
            wifiManager.disconnect()
            wifiManager.enableNetwork(networkId, true)
            wifiManager.reconnect()
```

```
            Toast.makeText(this@MainActivity2, “Connecting to WiFi: $wifiSSID”,
Toast.LENGTH_SHORT).show()
        } else {
            Toast.makeText(this@MainActivity2, “Failed to connect to WiFi”,
Toast.LENGTH_SHORT).show()
        }
    }    private fun initializeFromSharedPreferences() {
        val sharedPreferences = getSharedPreferences(“MyAppPrefs”, Context.MODE_PRIVATE)

        // Retrieve the stored JSON data from SharedPreferences
        val storedJsonData = sharedPreferences.getString(“login_response”, null)
        loggedIn = sharedPreferences.getBoolean(“loggedIn”, false)

        if (storedJsonData != null) {
            try {
                val json = JSONObject(storedJsonData)

                // Initialize your variables with data from the JSON object
                loginData = storedJsonData
                email = json.getString(“email”)
                password = json.getString(“password”)
                // Add other variable initializations as needed

                // Update the UI with the retrieved data
                Log.e(“data in shared preferences”,loginData.toString())
                updateUI(loginData.toString())
            } catch (e: JSONException) {
                e.printStackTrace()
            }
        }
    }

    override fun onActivityResult(requestCode: Int, resultCode: Int, data: Intent?) {
        super.onActivityResult(requestCode, resultCode, data)

        if (requestCode == REQUEST_CODE_SERVER_CONNECTIVITY && resultCode ==
RESULT_OK) {
            val updatedData = data?.getStringExtra(“studentData”)
            if (updatedData != null) {
//                SharedData.studentData = updatedData // Update the global variable
               loginData = updatedData
                updateUI(updatedData)

                Log.d(“MainActivity2”, “Updated SharedData.studentData: $updatedData”)
            }
        }
    }
        // Override onCreateOptionsMenu to inflate the menu
    override fun onCreateOptionsMenu(menu: Menu): Boolean {
        menuInflater.inflate(R.menu.menu_scanner, menu)
        updateMenuVisibility()
        return true
    }

    override fun onPrepareOptionsMenu(menu: Menu): Boolean {
        updateMenuVisibility()
        return true
    }

    // Override onOptionsItemSelected to handle menu item clicks
    override fun onOptionsItemSelected(item: MenuItem): Boolean {
        if (drawerToggle.onOptionsItemSelected(item)) {
            return true
        } else {
            when (item.itemId) {
                R.id.action_scan -> {
                    // Handle the scanner action
                    if (loggedIn) {
                        val integrator = IntentIntegrator(this)
                        integrator.setDesiredBarcodeFormats(IntentIntegrator.QR_CODE)
                        integrator.setPrompt("Scan a QR Code")
                        integrator.setCameraId(0)

                        val scanIntent = integrator.createScanIntent()
                        scanLauncher.launch(scanIntent)

                        return true
                    }else{
                        Toast.makeText(this@MainActivity2,"Login to mark
attendance",Toast.LENGTH_SHORT).show()

                    }
                }
                else -> if (drawerLayout.isDrawerOpen(GravityCompat.START)) {
                    drawerLayout.closeDrawer(GravityCompat.START)
                }
            }
        }
        return (drawerLayout.isDrawerOpen(GravityCompat.START))
    }

    private fun updateMenuVisibility() {
        val menu = navigationView.menu
        menu.findItem(R.id.menu_login).isVisible = !loggedIn
        menu.findItem(R.id.menu_check_attendance).isVisible = loggedIn
        menu.findItem(R.id.menu_check_result).isVisible = loggedIn
        menu.findItem(R.id.menu_logout).isVisible = loggedIn
    }
    private fun clearUserData() {
        // Clear or set to null/false the data you want to remove
        loginData = null
        email = null
        password = null
        loggedIn = false
        // Other variables as needed

        // Clear shared data or shared preferences
        SharedData.studentData = null // Assuming you have a shared data object
        val sharedPreferences = getSharedPreferences("MyAppPrefs", Context.MODE_PRIVATE)
        val editor = sharedPreferences.edit()
        editor.putBoolean("loggedIn",false)
        editor.remove("login_response") // Remove user-related data
        // Add more key-value pairs to remove other data if needed
        editor.apply()
    }

    private fun resetUI() {
        val textView5 = findViewById<TextView>(R.id.textView5)
        val textView2 = findViewById<TextView>(R.id.textView2)
//        SharedData.studentData = null
        // Clear the text and hide the UI elements
        textView5.text = "Login"
        textView2.text = ""
        textView5.visibility = View.VISIBLE
        textView2.visibility = View.INVISIBLE
    }

    private fun updateUI(result: String) {


        if (textView5 != null && textView2 != null) {
            if (result != null) {
                try {
                    // Parse the JSON response
                    val json = JSONObject(result)

                    // Get the student name and email from the JSON
                    val studentName = json.getString("student_name")
                    email = json.getString("email")
                    password = json.getString("password")
                    loginData = result

                    // Update the UI elements
                    runOnUiThread {
                        textView5.text = studentName
                        textView2.text = email
                        textView5.visibility = View.VISIBLE
                        textView2.visibility = View.VISIBLE
                        loggedIn = true

                        // Display a Toast on the main thread
                        Toast.makeText(applicationContext, "Data retrieved from SharedPreferences",
Toast.LENGTH_SHORT).show()
```

```
                }
            } catch (e: JSONException) {
                e.printStackTrace()
            }
        } else {
            // If the login was not successful, hide the TextViews
            resetUI()
        }
        updateMenuVisibility()
    }
}

}
```

Login student Activity:

In this we will be sending the request to the server to get the data with correct email and password. nothing but the servlet we already setup.
the servlet sends a json response which will be sent back to the MainActivity2.kt using the intent and stored in the Global variable and also the shared preference because when the student closes the app and opens it should not be lost and again ask to login.

LoginStudentActivity.kt

```
package com.example.ex10

import android.*
import 38ebView.*
import kotlinx.*
import org.json.*
import java.*

class LoginStudentActivity : AppCompatActivity() {
    private var emailEditText: EditText? = null
    private var passwordEditText: EditText? = null
    private var signInButton: Button? = null
    private var message: String? = null

    @SuppressLint("MissingPermission")
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.loginpagestudent)
        val fabBack = findViewById<FloatingActionButton>(R.id.floatingActionButton2)
        emailEditText = findViewById(R.id.editTextTextEmailAddress2)
        passwordEditText = findViewById(R.id.editTextTextPassword2)
        signInButton = findViewById(R.id.button2)
        val forgotPasswordTextView = findViewById<TextView>(R.id.textView4)
        forgotPasswordTextView.setOnClickListener {
            val intent = Intent(this, ForgotPasswordActivity::class.java)
            startActivity(intent)
        }
        fabBack.setOnClickListener {
            // Handle the click event here
            finish() // Finish the current activity to navigate back to MainActivity2
        }

        signInButton?.setOnClickListener(View.OnClickListener {
            val email = emailEditText?.text.toString()
            val password = passwordEditText?.text.toString()

            val servletUrl = “http://192.168.0.152:8080/project_portable/studentLogin”

            // Retrieve the unique identifier from SharedPreferences
            val sharedPreferences = getSharedPreferences(“MyAppPrefs”, Context.MODE_PRIVATE)
            val storedUniqueIdentifier = sharedPreferences.getString(“uniqueIdentifierKey”, null)

            if (storedUniqueIdentifier == null) {
                // If the unique identifier is not available, generate and store it
                val uniqueIdentifier = generateUniqueIdentifier(email) // You need to implement this
function
                storeUniqueIdentifier(uniqueIdentifier)
                message = “store”
                Log.d(“uid store”,storedUniqueIdentifier.toString())
                Log.d(“uid store”,uniqueIdentifier)
                GlobalScope.launch(Dispatchers.Main) {
                    val result = makeHttpRequest(servletUrl, email, password,
uniqueIdentifier,message.toString())
                    Log.d(“HTTP Response”, result)
                    handleLoginResult(result, servletUrl )
                }

            }else{
                message = “check”
                Log.d(“uid  check”,storedUniqueIdentifier)
                GlobalScope.launch(Dispatchers.Main) {
                    val result = makeHttpRequest(servletUrl, email, password,
storedUniqueIdentifier.toString(),message.toString())
                    Log.d(“HTTP Response”, result)
                    handleLoginResult(result, servletUrl )
                }
            }

        })
    }
    private fun storeUniqueIdentifier(uniqueIdentifier: String) {
        // Store the hashed identifier in your backend database, associating it with the user’s account
        // You need to implement this part for your specific database system

        // Store the hashed identifier in SharedPreferences
        val editor = getSharedPreferences(“MyAppPrefs”, Context.MODE_PRIVATE).edit()
        editor.putString(“uniqueIdentifierKey”, uniqueIdentifier)
        editor.apply()
    }

    private fun hashString(input: String): String {
        try {
            val digest = MessageDigest.getInstance(“SHA-256”)
            val hash = digest.digest(input.toByteArray())
            val hexString = StringBuilder()

            for (byte in hash) {
                val hex = Integer.toHexString(0Xff and byte.toInt())
                if (hex.length == 1) {
                    hexString.append(‘0’)
                }
                hexString.append(hex)
            }

            return hexString.toString()
        } catch (e: NoSuchAlgorithmException) {
            e.printStackTrace()
            return “”
        }
    }


    private fun generateUniqueIdentifier(email: String): String {
        // Generate a unique identifier based on the email or other data
        val hashedIdentifier = hashString(email)

        // Store the hashed identifier in SharedPreferences
        val editor = getSharedPreferences(“MyAppPrefs”, Context.MODE_PRIVATE).edit()
        editor.putString(“uniqueIdentifierKey”, hashedIdentifier)
        editor.apply()

        // Store the identifier in your backend database, associating it with the user’s account
        // You need to implement this part for your specific database system

        return hashedIdentifier
    }

    private suspend fun makeHttpRequest(url: String, email: String, password: String, uid: String ,
message: String): String {

        return withContext(Dispatchers.IO) {
            try {
                val urlWithParams =
“$url?email=$email&password=$password&UID=$uid&message=$message”
                val connection = URL(urlWithParams).openConnection() as HttpURLConnection

                connection.requestMethod = “GET”
                connection.connectTimeout = 10000
                connection.readTimeout = 10000
                connection.connect()

                if (connection.responseCode == HttpURLConnection.HTTP_OK) {
                    val reader = BufferedReader(InputStreamReader(connection.inputStream))
                    val result = StringBuilder()
                    var line: String?

                    while (reader.readLine().also { line = it } != null) {
                        result.append(line)
                    }
                    reader.close()
                    connection.disconnect()

                    return@withContext result.toString()
                } else {
                    return@withContext “Connection failed.”
                }
            } catch (e: Exception) {
                e.printStackTrace()
                return@withContext “Exception error.”
            }
        }
    }

    private fun handleLoginResult(result: String, servletUrl: String) {
        if (result != null) {
            if (result == “Connection failed.”) {
                Toast.makeText(this, “Login Failed”, Toast.LENGTH_SHORT).show()
            } else if (result == “Exception error.”) {
                Toast.makeText(this, result, Toast.LENGTH_SHORT).show()
            } else {
                // Store the response in SharedPreferences
                val sharedPreferences = getSharedPreferences(“MyAppPrefs”, Context.MODE_PRIVATE)
                val editor = sharedPreferences.edit()
                editor.putString(“login_response”, result)
                editor.putBoolean(“loggedIn”, true)
                editor.apply()

                // Set the result data in the intent
                val intent = Intent()
                intent.putExtra(“login_state”, true)
                intent.putExtra(“Student Data”, result)
                SharedData.studentData = result
                intent.putExtra(“login_url”, servletUrl)
                Toast.makeText(this@LoginStudentActivity, “Login Successful”,
Toast.LENGTH_SHORT).show()
               // val code = 1
                setResult(RESULT_OK, intent)
                finish()
            }
        } else {
          // Handle other cases like incorrect credentials, parsing errors, connection failures, etc.
          Toast.makeText(this@LoginStudentActivity, “result is null”,
Toast.LENGTH_SHORT).show()
        }
    }
}
```

Password resetting activity:
ForgotPasswordActivity.kt

```
package com.example.ex10

import android.*
import 42ebView.*
import kotlinx.*
import org.json.*
import java.*

@Suppress(“DEPRECATION”)
class ForgotPasswordActivity : ComponentActivity() {
    private val delayMillis: Long = 25000 // Adjust this value as needed (5 seconds in this example)

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.forgot_password_activity)

        val webView = findViewById<WebView>(R.id.webView)

        // Enable JavaScript if your web page requires it
        webView.settings.javaScriptEnabled = true

        // Load your password reset page URL
        webView.loadUrl(“http://192.168.0.152:8080/project_portable/password-reset-request.jsp”)

        // Set a WebViewClient to handle page navigation within the WebView
        webView.webViewClient = object : WebViewClient() {
            override fun shouldOverrideUrlLoading(view: WebView?, url: String?): Boolean {
                // Check if the URL matches the deep link
                    if (url != null) {
                        view?.loadUrl(url)
                        return true
                    }else {
                        return false
                    }
            }
        }
        // Post a delayed task to finish the activity after a certain delay
        Handler().postDelayed({
            finish()
        }, delayMillis)
    }
}
```

DataFetchingActivity.kt

Here we will be fetching the updated data after marking the attendance. and update our StudentAttendanceActivity.

```
package com.example.ex10

import android.*
import 43ebView.*
import kotlinx.*
import org.json.*
import java.*


class DataFetchingActivity : ComponentActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        val loginUrl =intent.getStringExtra("login_url")
        val email = intent.getStringExtra("email")
        val password =   intent.getStringExtra("password")
        val sharedPreferences = getSharedPreferences("MyAppPrefs", Context.MODE_PRIVATE)

        Log.e("data Fetching activity",loginUrl+email+password)
        // ...
        GlobalScope.launch(Dispatchers.IO) {
            val data = makeHttpRequest(loginUrl.toString(), email.toString(), password.toString())
            Log.e("data Fetching activity", data)
            val editor = sharedPreferences.edit()
            editor.putString("login_response", data)
            editor.apply()
            runOnUiThread {
                // Return the data to ServerConnectivityActivity
                val intent = Intent()
                intent.putExtra("studentData", data)
                setResult(RESULT_OK, intent)
                runOnUiThread{              Toast.makeText(applicationContext, SharedData.studentData,
Toast.LENGTH_SHORT).show()
                }
                finish()

            }

        }

    }
    private suspend fun makeHttpRequest(url: String, email: String, password: String): String {
        return withContext(Dispatchers.IO) {
            try {
                val urlWithParams = "$url?email=$email&password=$password"
                val connection = URL(urlWithParams).openConnection() as HttpURLConnection
                connection.requestMethod = “GET”
                connection.connectTimeout = 10000
                connection.readTimeout = 10000
                connection.connect()

                if (connection.responseCode == HttpURLConnection.HTTP_OK) {
                    val reader = BufferedReader(InputStreamReader(connection.inputStream))
                    val result = StringBuilder()
                    var line: String?

                    while (reader.readLine().also { line = it } != null) {
                        result.append(line)
                    }
                    reader.close()
                    connection.disconnect()
                   SharedData.studentData = result.toString()

                    return@withContext result.toString()
                } else {
                    return@withContext “Connection failed.”
                }
            } catch (e: Exception) {
                e.printStackTrace()
                return@withContext “Exception error.”
            }
        }
    }
}
```

ServerConnectivityActivity.kt

This activity is used to mark the attendance. After scanning the Qr code this activity will be started if it is an url or if the scanned data belongs to wifi then that operation will be performed.
But if it is a url then this activity will be started by passing that url to this activity with the help of intent and first the user is authenticated i.e., verify the student with the help of the fingerprint then go to the server request later.
If the student doesn’t verify then server request is not sent else sent.
This is how this activity functions.

```
package com.example.ex10

import android.*
import 44ebView.*
import kotlinx.*
import org.json.*
import java.*

@Suppress(“DEPRECATION”)
class ServerConnectivityActivity : FragmentActivity() {
    private var result: String? = null
    private var REQUEST_CODE_DATA_FETCH = 1
    private lateinit var biometricPrompt: BiometricPrompt

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)

        // Step 1: Get the server URL and login status from the intent
        val markUrl = intent?.getStringExtra("scanned_url")
        val loginUrl = "http://192.168.0.152:8080/project_portable/DataFetchingServlet"
        val loginStatus = intent.getBooleanExtra("login_status", false)
        val loginData = intent.getStringExtra("login_data")
        val jsonObject = JSONObject(loginData)
        val JloginData = jsonObject.getString("student_pin")
        val email = jsonObject.getString("email")
        val password = jsonObject.getString("password")
//        val LoginUrl = intent.getStringExtra("login_url")


        if (markUrl != null && loginStatus) {
            val executor = ContextCompat.getMainExecutor(this)
             biometricPrompt = BiometricPrompt(this, executor, object :
BiometricPrompt.AuthenticationCallback() {
                override fun onAuthenticationSucceeded(result: BiometricPrompt.AuthenticationResult) {
                    super.onAuthenticationSucceeded(result)
                    // Fingerprint authentication succeeded, continue with your code
                    continueWithHttpRequest(JloginData,markUrl)
                    val intent = Intent(this@ServerConnectivityActivity,DataFetchingActivity::class.java)
                    intent.putExtra("login_url",loginUrl)
                    intent.putExtra("email",email)
                    intent.putExtra("password",password)
                    startActivityForResult(intent,REQUEST_CODE_DATA_FETCH)

                }



                override fun onAuthenticationFailed() {
                    super.onAuthenticationFailed()
                    // Fingerprint authentication failed
                    Toast.makeText(applicationContext, "Fingerprint authentication failed",
Toast.LENGTH_SHORT).show()
                }
                 override fun onAuthenticationError(errorCode: Int, errString: CharSequence) {
                     super.onAuthenticationError(errorCode, errString)

                     // Handle errors, including user cancellation
                     if (errorCode == BiometricPrompt.ERROR_NEGATIVE_BUTTON &&
errString.toString()
                             .contains("Cancel")
                     ) {
```

```
Toast.makeText(this@ServerConnectivityActivity,”canelled”,Toast.LENGTH_SHORT).show()
                } else if (errorCode != BiometricPrompt.ERROR_USER_CANCELED) {
                    // Handle other errors
                    Toast.makeText(this@ServerConnectivityActivity,” button not
clicked”,Toast.LENGTH_SHORT).show()
                }
                finish()
            }
        })

        val promptInfo = BiometricPrompt.PromptInfo.Builder()
            .setTitle(“Authenticate”)
            .setSubtitle(“Place your finger on the sensor to authenticate”)
            .setNegativeButtonText(“Cancel”)
            .build()

        biometricPrompt.authenticate(promptInfo)
    }
}

private fun continueWithHttpRequest( loginData : String,lgnUrl : String) {
    // The user has successfully authenticated with their fingerprint.
    // You can proceed with making the HTTP request and other operations here.

    // Replace this with your actual HTTP request logic
    val urlWithPin = “$lgnUrl?student_pin=$loginData”
    Log.d(“url with pin is “, urlWithPin)
    GlobalScope.launch(Dispatchers.IO) {
        try {
            result = makeHttpRequestToMark(urlWithPin)

            if (result != null) {
                val jsonResponse = JSONObject(result.toString())

                val status = jsonResponse.getString(“status”)
                val message = jsonResponse.getString(“message”)

                when (status) {
                    “success” -> {
                        if (message == “Attendance marked successfully.”) {
                            runOnUiThread {
                                // Handle success
                                Toast.makeText(
                                    applicationContext,
                                    “Attendance Marked”,
                                    Toast.LENGTH_SHORT
                                ).show()
                            }
                            // Continue with your code here
                        } else {
                            runOnUiThread {
                                // Handle the case where attendance is already marked
                                Toast.makeText(
                                    applicationContext,
                                    “Attendance is already marked.”,
                                    Toast.LENGTH_SHORT
                                ).show()
                            }
                            // Continue with your code here
                        }
                    }
                    // Handle other status cases if needed
                }
            }
        } catch (e: Exception) {
            e.printStackTrace()
            Log.e(“Exception in HTTP request”, “Error: “ + e.message)
        }
    }
}


// Handle the result from DataFetchingActivity
override fun onActivityResult(requestCode: Int, resultCode: Int, data: Intent?) {
    super.onActivityResult(requestCode, resultCode, data)

    if (requestCode == REQUEST_CODE_DATA_FETCH && resultCode == RESULT_OK) {
        val updatedData = data?.getStringExtra(“studentData”)
        if (updatedData != null) {
            SharedData.studentData = updatedData
            Log.d(“ServerConnectivityActivity”, “Updated SharedData.studentData: $updatedData”)

            // Now, send the updated data back to MainActivity2
            val intent = Intent()
            intent.putExtra(“studentData”, updatedData)
            setResult(RESULT_OK, intent)
            finish()
        }
    }
}

private suspend fun makeHttpRequestToMark(url: String): String? {
    return withContext(Dispatchers.IO) {
        try {
            val connection = URL(url).openConnection() as HttpURLConnection
            Log.d(“inside to mark method “, “URL: $url”)

            connection.requestMethod = “GET”
            connection.connectTimeout = 10000
            connection.readTimeout = 10000
```

```
                connection.connect()

                if (connection.responseCode == HttpURLConnection.HTTP_OK) {
                    val reader = BufferedReader(InputStreamReader(connection.inputStream))
                    val result = StringBuilder()
                    var line: String?

                    while (reader.readLine().also { line = it } != null) {
                        result.append(line)
                    }
                    reader.close()
                    connection.disconnect()

                    return@withContext result.toString()
                } else {
                    return@withContext null
                }
            } catch (e: Exception) {
                e.printStackTrace()
                Log.e("mark attendance exception", "Exception error: " + e.message)
                return@withContext "Exception error."
            }
        }
    }
}
```

StudentAttendanceActivity.kt
This activity helps students to let them check their attendance. It uses a ListView and using ArrayAdapter we set the data in it.

```
package com.example.ex10

import android.*
import 48ebView.*
import kotlinx.*
import org.json.*
import java.*

class StudentAttendanceActivity : AppCompatActivity() {
    private lateinit var textViewStudentName: TextView
    private lateinit var listViewAttendanceData: ListView
    private lateinit var textViewAttendancePercentage: TextView


    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_student_attendance)
        Log.d("StudentAttendanceActivity", "onCreate method is called")


        // Get references to UI elements
        textViewStudentName = findViewById(R.id.textViewStudentName)
        listViewAttendanceData = findViewById(R.id.listViewAttendanceData)
        textViewAttendancePercentage = findViewById(R.id.textViewAttendancePercentage)
        var data = intent.getStringExtra("student Data")
        if(data!= null){
            if (SharedData.studentData!=null){
                data = SharedData.studentData
            }
        }


        Log.d("StudentAttendanceActivity",data.toString())
        updateUIWithAttendanceData(data.toString()
    }

    private fun updateUIWithAttendanceData(attendanceData: String) {
        try {
            if (attendanceData != null) {
                val json = JSONObject(attendanceData)
                val studentName = json.getString("student_name")
                val attendanceArray = json.getJSONArray("attendance_data")
                val attendancePercentage = json.getInt("attendance_percentage") // New line

                // Update student name and attendance percentage
                textViewStudentName.text = studentName
                textViewAttendancePercentage.text = "Attendance Percentage: $attendancePercentage%"
// New line

                // Extract date and time from attendance data
                val attendanceList = ArrayList<String>()
                for (I in 0 until attendanceArray.length()) {
                    val attendanceObject = attendanceArray.getJSONObject(i)
                    val isPresent = attendanceObject.getBoolean("is_present")
                    val date = attendanceObject.getString("attendance_date")
                    val time = attendanceObject.getString("attendance_time")

                    val status = if (isPresent) "Present" else "Absent"
                    val attendanceInfo = "Date: $date, Time: $time, Status: $status"
                    attendanceList.add(attendanceInfo)
                }

                // Create an ArrayAdapter to populate the ListView
                val adapter =
                    ArrayAdapter(this, android.R.layout.simple_list_item_1, attendanceList)
                listViewAttendanceData.adapter = adapter
            } else {
                Toast.makeText(this, "StudentAttendanceActivity", Toast.LENGTH_SHORT).show()
            }
        } catch (e: JSONException) {
            e.printStackTrace()
        }
    }


}
```

WebViewActivity.kt

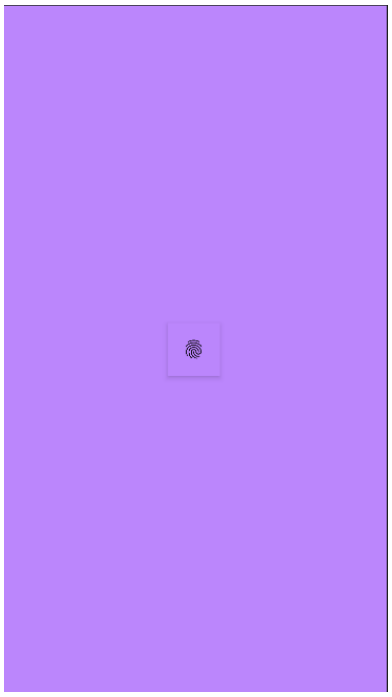
In this activity we will be using the webview layout to load the sbtet's official website in our app. Nothing but this layout supports rendering of the webpages. So instead of creating another table for the marks we are directly using the webview layout to render the Sbtet's website.

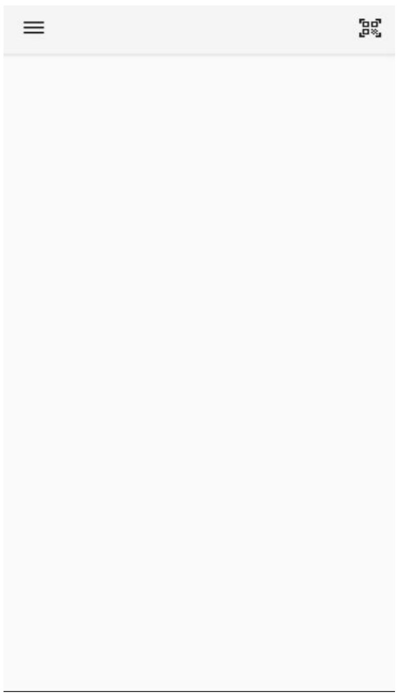
```
package com.example.ex10

import android.*
import 50ebView.*

class WebViewActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_webview)

        val webView = findViewById<WebView>(R.id.webView)

        // Enable JavaScript if your web page requires it
        webView.settings.javaScriptEnabled = true

        // Get the deep link URL from the intent
        val deepLink = intent.data

        if (deepLink != null) {
            // Handle the deep link URL
            webView.loadUrl(deepLink.toString())
        }

        // Set a WebViewClient to handle page navigation within the WebView
        webView.webViewClient = object : WebViewClient() {
            override fun shouldOverrideUrlLoading(view: WebView?, url: String?): Boolean {
                // Load the URL within the WebView
                if (url != null) {
                    view?.loadUrl(url)
                    return true
                }
                return false
            }
        }
    }
}
```

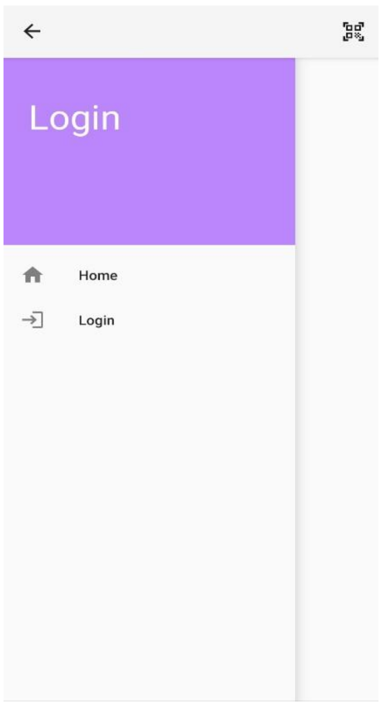
For full code visit GitHub.

https://github.com/print-ramcharan/mobile-attendance-sytem-with-integrated-fingerprint-sensor.

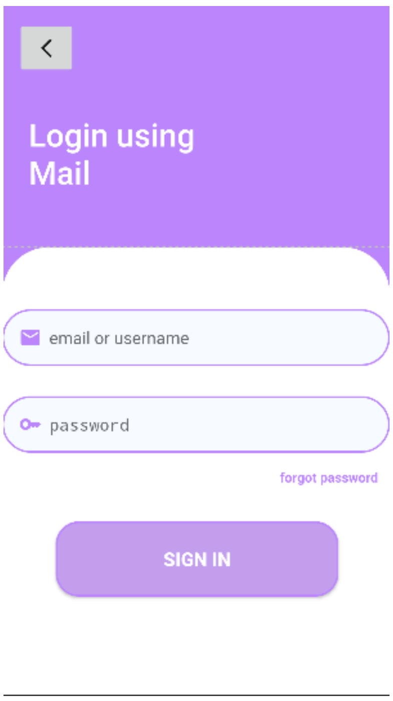
CHAPTER – 10
APP UI


StartingActivity

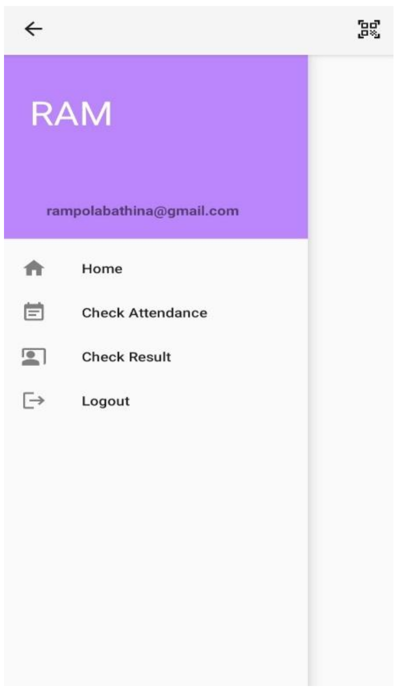

MainActivity



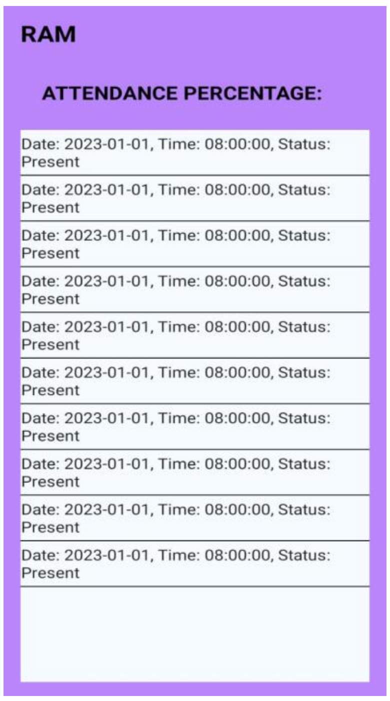
Menu



LoginActivity



UpdatedMenu


RAM

ATTENDANCE PERCENTAGE:

```
Date: 2023-01-01, Time: 08:00:00, Status:
Present
Date: 2023-01-01, Time: 08:00:00, Status:
Present
Date: 2023-01-01, Time: 08:00:00, Status:
Present
Date: 2023-01-01, Time: 08:00:00, Status:
Present
Date: 2023-01-01, Time: 08:00:00, Status:
Present
Date: 2023-01-01, Time: 08:00:00, Status:
Present
Date: 2023-01-01, Time: 08:00:00, Status:
Present
Date: 2023-01-01, Time: 08:00:00, Status:
Present
Date: 2023-01-01, Time: 08:00:00, Status:
Present
Date: 2023-01-01, Time: 08:00:00, Status:
Present
```

AttendanceChecking

CHAPTER – 11

FUTURE REFERENCES

Integration with IoT Devices:
Explore incorporating Internet of Things (IoT) devices to enhance the connectivity and functionality of the mobile attendance system, providing real-time data and analytics.

Machine Learning for Predictive Analysis:
Implement machine learning algorithms to analyses attendance patterns, enabling the system to predict future attendance trends and optimize resource allocation.

Blockchain for Enhanced Security:
Investigate the use of blockchain technology to enhance the security and integrity of attendance records, ensuring immutability and transparency.

Mobile App Enhancements:
Continuously improve the mobile application interface, adding features such as notifications, user-friendly dashboards, and support for various platforms to enhance user experience.

Biometric Recognition Advancements:
Stay updated with the latest advancements in biometric recognition technologies, such as facial recognition and voice authentication, to improve accuracy and user convenience.

Cloud-Based Attendance Management:
Explore migrating the attendance system to cloud infrastructure for scalability, accessibility, and seamless integration with other educational or organizational systems.

Data Analytics for Performance Insights:
Utilize data analytics tools to gain insights into attendance data, helping administrators make informed decisions to improve overall system performance.

Integration with Educational Software:
Collaborate with educational software providers to integrate the attendance system with existing educational platforms, creating a comprehensive solution for academic institutions.

Accessibility and Inclusivity:
Focus on making the system more accessible and inclusive, ensuring compatibility with assistive technologies and accommodating a diverse range of users.

Cybersecurity Measures:
Stay vigilant on emerging cybersecurity threats and regularly update security protocols to safeguard sensitive attendance data from potential breaches.

CONCLUSION

The implementation of a mobile attendance system with a fingerprint scanner stands as a milestone in modern attendance management. This project, with its emphasis on biometric authentication, establishes a robust and tamper-proof method for recording attendance. The educational sector benefits from increased efficiency and reliability, setting a precedent for technological advancements in administrative processes. The adaptability of this system indicates its scalability for diverse organizational needs. As biometric technology evolves, this project lays a foundation for further innovations in secure and seamless attendance tracking.