RAJALAKSHMI ENGINEERING COLLEGE

RAJALAKSHMI NAGAR, THANDALAM – 602 105

CS23333 OOPS Using Java

Laboratory Record Note Book

Name : P. Giri Prasad

Year / Branch / Section : II/B.E C.S.E

University Register No. : 2116240701147

College Roll No. : 240701147 .

Semester : III .

Academic Year : 2025-26 .

RAJALAKSHMI ENGINEERING COLLEGE

An Autonomous Institution

BONAFIDE CERTIFICATE

Name: P.Giri Prasad

Academic Year: II **Semester:** III **Branch:** CSE

Register No. 2116240701147

Certified that this is the bonafide record of work done by the above student in the..Laboratory during the academic year 2025- 2026

Signature of Faculty in-charge

Submitted for the Practical Examination held on……………………………

Internal Examiner **External Examiner**

INDEX

EX.NO	DATE	NAME OF THE EXPERIMENT	GITHUB QR
1		I/O, Data Types, Operators	
2		Control Structures	
3		Arrays	
4		Strings	
5		Classes & Objects	
6		Inheritance	
7		Interface	
8		Exceptions	
9		Collections	
10		Collections	
11		Project	
12		Lambda	

Abstract

Our goal is to develop the Attendance Monitoring System and Decentralized Course Selection is a full-featured Java software with a MySQL database as a backend. This increases the efficiency on how schools and colleges run their academic operations. Managing attendance and schedules in the traditional way is generally done by hand, which can lead to mistakes and lack of flexibility. This solution solves these problems by giving students real-time access to their individualized schedules and assigned teachers, with live updates that show any changes in scheduling, replacements, or room assignments. The technology automatically tracks attendance and updates it right away, so students and teachers can see how many people are there at present. The platform has easy-to-use interfaces for entering and updating schedules, keeping track of attendance, and making reports on the administrative side. The solution improves transparency, lowers the amount of labor that needs to be done by hand, and helps data-driven decision-making in schools by combining dynamic scheduling with live attendance monitoring. In the future, this approach could get even better by adding RFID technology to ID cards or mobile devices that can use NFC. Logging attendance is quick and safe with just a tap or scan. This makes the procedure not only faster but also almost flawless. This modern approach will help institutions work smarter, reduce mistakes and focus more on learning.

Keywords: Java, DBMS, Automated attendance tracking, Data-driven decision-making, Educational technology

Contents

List of Figures

List of Tables

Listings

Chapter 1

Introduction

The *Attendance Monitoring System* is a high tech digital tool that solves the problems that come with conventional ways of keeping track of attendance in schools. Traditional ways of keeping track of attendance by hand are generally slow, make mistakes, and aren't very useful. The suggested solution makes this procedure automatic, which makes sure that the data is accurate, efficient, and available in real time.

This system uses JavaFX for the graphical user interface and a database driven architecture to store attendance data safely, reliably, and in a way that can grow. It is a step forward in the digital revolution of the administration of schools. As more schools and universities use smart campus management systems, this one brings together important features like monitoring attendance, managing schedules, and reporting to administrators into one platform.

The system follows traditional software engineering practices, using **object oriented design** and **modular programming** to make it more reliable, scalable, and easy to maintain. Database normalisation also reduces duplication and improves data integrity, which makes sure that information is preserved in the best way possible. Integrating attendance and schedule management without any problems boosts productivity and makes administrative tasks easier. In general, this project helps make the school administration environment more contemporary, automated, and smart.

1.1 Problem Statement

Recording student attendance by hand is old fashioned, slow, and prone to mistakes. Teachers typically have trouble keeping track of a lot of documents, getting old data, and making reports that show how things are going. Also, the fact that attendance and schedule data don't match up makes it hard to keep track of things and leads to inconsistencies. The goal of this project is to fix these problems by creating an automated system that makes it easier to keep track of attendance, makes sure the data is correct, and lets instructors and administrators see the data in real time.

1.2 Scope of Work

This project covers designing and building a completely working **Attendance Monitoring System** with separate parts for **Administrators**, **Teachers**, and **Students**.

- The **Admin module** keeps track of users, topics, and schedules.
- The **Teacher module** makes it easier to take attendance and see the class schedules that have been issued.
- The **Student module** lets students see their attendance records and schedules.

1.3 Aim and Objectives

The goal of this project is to make it easier for schools to keep track of attendance by providing a quick, safe, and easy-to-use automated system. This will make it easier to record, save, and analyse attendance data. The system is meant to replace the old-fashioned way of taking attendance with a digital, automated one. It does this by using a strong database-backed architecture to make sure that data is always accurate and up-to-date. The system was made using JavaFX and has an interactive and efficient user interface that lets administrators, instructors, and students access role-based dashboards. It combines the schedule and attendance modules to improve data accuracy and coordination. It also improves data integrity via normalisation and safe access control methods. Finally, it decreases the amount of work that administrators have to do while increasing the overall efficiency of the operation.

Chapter 2

Literature Review

2.1 Intelligent Attendance and IoT based Systems

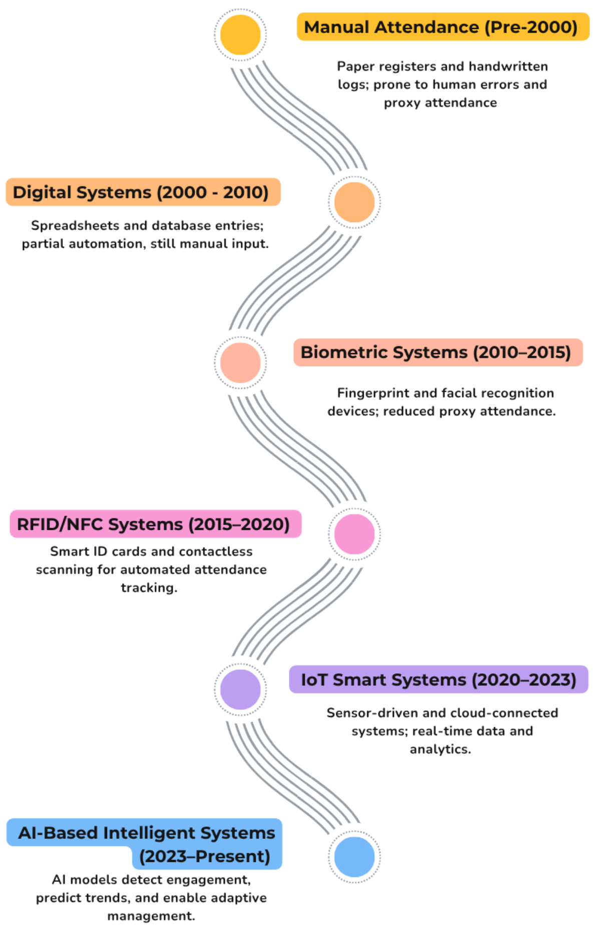

Manual Attendance (Pre-2000)
Paper registers and handwritten logs; prone to human errors and proxy attendance
Digital Systems (2000 - 2010)
Spreadsheets and database entries; partial automation, still manual input.
Biometric Systems (2010–2015)
Fingerprint and facial recognition devices; reduced proxy attendance.
RFID/NFC Systems (2015–2020)
Smart ID cards and contactless scanning for automated attendance tracking.
IoT Smart Systems (2020–2023)
Sensor-driven and cloud-connected systems; real-time data and analytics.
AI-Based Intelligent Systems (2023–Present)
AI models detect engagement, predict trends, and enable adaptive management.

Fig. 2.1 Flowchart Depicting the Development of Attendance Systems

The Internet of Things (IoT) has changed how schools keep track of attendance in a big way. It has made it easier, more accurate, and available in real time. Nguyen et al. [1] introduced an IoT based intelligent attendance system framework that amalgamates sensors and cloud communication to guarantee the smooth synchronization of attendance data. Their implementation showed that it was reliable and could be used in diverse network settings. Chiang et al. [2] created and tested a mobile attendance monitoring software that used Android GPS and NFC modules. It worked well in both indoor and outdoor classrooms.

Wu et al. [3] presented a novel solution using photovoltaic Bluetooth beacon badges that operate without batteries, signifying a sustainable method for extensive attendance control. The report by Anonymous [4] also spoke about how IoT, AI, and cloud technologies can change the way schools work and make them smarter by making both administrative tasks and learning analytics better. Together, these efforts show that IoT based attendance systems provide the basis for future smart and adaptive learning systems.

2.2 AI and Deep Learning Approaches for Student Engagement Detection

Recent research has concentrated on using Artificial Intelligence (AI) and Deep Learning (DL) models to assess and forecast student involvement levels, in addition to automating attendance. Bhardwaj et al. [5] used deep learning to evaluate behavioral patterns in e learning contexts, showing enhanced predictive accuracy compared to conventional machine learning techniques. Mahmood et al. [6] examined emotional and behavioral engagement characteristics using sophisticated neural network models to improve comprehension of student involvement and motivation.

Jiang et al. [7] introduced a multimodal fusion method that integrates visual, auditory, and postural inputs to identify engagement states in authentic classroom settings. Yan et al. [8] built on this notion by combining multimodal deep learning processes to make it possible to monitor engagement in real time. Malekshahi et al. [9] devised a framework for engagement detection using convolutional and recurrent neural networks, while Bell et al. [10] conducted a trend analysis emphasizing the growing significance of AI in educational engagement research.

Bellarhmouch [11, 12] investigated predictive modeling using assessment data, including quizzes and examinations, therefore enhancing the comprehensive knowledge of engagement measurement. These studies together demonstrate that AI driven engagement analysis may enhance IoT based attendance systems, resulting in fully intelligent academic monitoring settings that provide tailored analytics and adaptive learning responses.

Fig. 2.2 IoT-based Attendance System using RFID Sensors

Chapter 3

System Design

3.1 ER Diagram

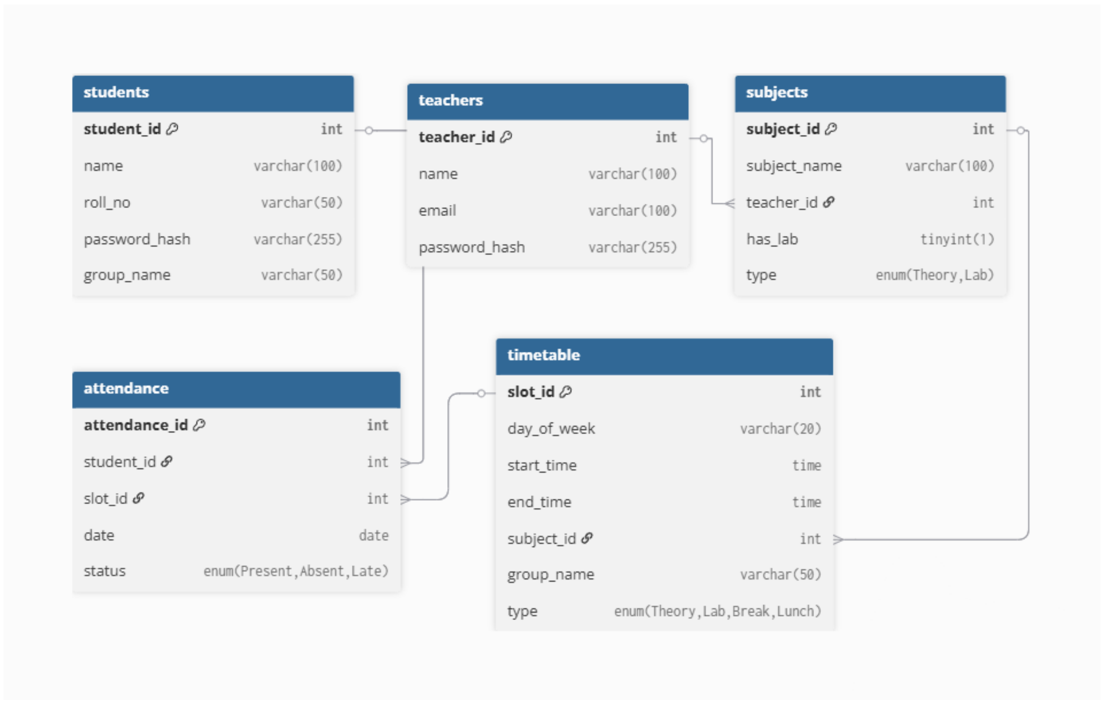


Fig. 3.1 ER Diagram of Proposed Attendance Management System

3.2 Database Schema Design

The database structure is designed to handle student attendance, schedules, topics, and instructor assignments in the best way possible. For institutional use, it makes sure that referential integrity, modularity, and scalability are all in place.

The system has five main tables: students, instructors, topics, the schedule, and attendance. To make joins and searches faster, each table is connected to another by main and foreign keys.

Entity Descriptions

- ✓ **Students:** Stores basic student details such as name, roll number, and group. Each student is identified by a unique `student_id`.
- ✓ **Teachers:** Contains faculty information including name and email address.
- ✓ **Subjects:** Defines academic subjects, their type (Theory/Lab), and assigns each to a specific teacher.
- ✓ **Timetable:** Represents the academic schedule, linking each slot to a subject and group.
- ✓ **Attendance:** Records the attendance status of students for specific timetable slots and dates.

Relational Schema

Table 3.1 Schema of the `students` Table

Field	Type	Null	Key	Description
student_id	INT	NO	PRI	Student specific ID
name	VARCHAR(20)	YES		Full name of the student
roll_no	VARCHAR(10)	YES	UNI	Unique roll number
password_hash	VARCHAR(255)	YES		Encrypted student password
group_name	VARCHAR(20)	YES		Academic group name

Table 3.2 Schema of the `teachers` Table

Field	Type	Null	Key	Description
teacher_id	INT	NO	PRI	Teacher specific ID
name	VARCHAR(20)	YES		Name of the teacher
email	VARCHAR(30)	YES	UNI	Unique email address
password_hash	VARCHAR(255)	YES		Encrypted teacher password

Table 3.3 Schema of the `subjects` Table

Field	Type	Null	Key	Description
subject_id	INT	NO	PRI	Subject specific ID
subject_name	VARCHAR(20)	YES		Name of the subject
teacher_id	INT	YES	MUL	Subject linked teacher
has_lab	TINYINT(1)	YES		Indicates if the subject includes lab sessions
type	ENUM('Theory','Lab')	YES		Specifies the subject type

Table 3.4 Schema of the `timetable` Table

Field	Type	Null	Key	Description
slot_id	INT	NO	PRI	Unique time slot ID
day_of_week	VARCHAR(20)	YES		Day of the week
start_time	TIME	YES		Lecture start time
end_time	TIME	YES		Lecture end time
subject_id	INT	YES	MUL	Linked subject for the slot
group_name	VARCHAR(50)	YES		Associated group name
type	ENUM('Theory','Lab', 'Break','Lunch')	YES		Type of session

Table 3.5 Schema of the `attendance` Table

Field	Type	Null	Key	Description
attendance_id	INT	NO	PRI	Attendance record unique ID
student_id	INT	YES	MUL	Linked student identifier
slot_id	INT	YES	MUL	Linked timetable slot
date	DATE	YES		Date of the lecture
status	ENUM('Present','Absent', 'Late')	YES		Attendance status

The schema normalizes, matches, and scales data for future additions like face recognition based attendance.

Chapter 4

Implementation

4.1 Key Java Implementation Snippets

4.1.1 Login Management

Listing 4.1 Login Handling in MainController

```
@FXML
private void handleLogin() {
 String role = roleChoice.getValue();
 String id = idField.getText();
 String password = passwordField.getText();

 if (role == null || id.isEmpty() || password.isEmpty()) {
 messageLabel.setText("Please␣fill␣all␣fields!");
 return;
 }

 try (Connection conn = DBConnection.getConnection()) {
 String sql = "";
 switch (role) {
 case "Student":
```

```
 sql = "SELECT␣*␣FROM␣students␣WHERE␣roll_no=?␣AND␣password=?"
    ;
 break;
 case "Teacher":
 sql = "SELECT␣*␣FROM␣teachers␣WHERE␣teacher_id=?␣AND␣password
    =?";
 break;
 case "Admin":
 sql = "SELECT␣*␣FROM␣admin␣WHERE␣email=?␣AND␣password=?";
 break;
 }

 PreparedStatement pst = conn.prepareStatement(sql);
 pst.setString(1, id);
 pst.setString(2, password);

 ResultSet rs = pst.executeQuery();
 if (rs.next()) {
 loadDashboard(role, rs);
 } else {
 messageLabel.setText("Invalid␣credentials!");
 }

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

Loading the Dashboard

Listing 4.2 Dashboard Loading

```
private void loadDashboard(String role, ResultSet rs) throws
   SQLException {
 Stage stage = (Stage) idField.getScene().getWindow();
 FXMLLoader loader = null;

 switch (role) {
 case "Student":
 loader = new FXMLLoader(
 getClass().getResource("/gui/StudentDashboard.fxml"));
 stage.setScene(new Scene(loader.load()));
 StudentDashboardController ctrl = loader.getController();
 ctrl.setRollNo(rs.getString("roll_no"));
 break;
 case "Teacher":
 loader = new FXMLLoader(
 getClass().getResource("/gui/TeacherDashboard.fxml"));
 stage.setScene(new Scene(loader.load()));
 TeacherDashboardController tCtrl = loader.getController();
 tCtrl.setTeacherId(rs.getInt("teacher_id"));
 break;
 case "Admin":
 loader = new FXMLLoader(
 getClass().getResource("/gui/AdminDashboard.fxml"));
 stage.setScene(new Scene(loader.load()));

 break;
 }

 stage.setTitle(role + "␣Dashboard");
}
```

4.1.2 Student Dashboard - TimeTable Console

Today’s Timetable Display

Listing 4.3 Show Today’s Timetable

```
private void showTodayTimetable() {
```

```
VBox box = new VBox(10);
box.setPadding(new Insets(10));

TableView<TimetableEntry> table = new TableView<>();
table.setColumnResizePolicy(TableView.
   CONSTRAINED_RESIZE_POLICY);

TableColumn<TimetableEntry, String> colTime = new TableColumn
   <>("Time");
colTime.setCellValueFactory(cell -> cell.getValue().
   timeProperty());

TableColumn<TimetableEntry, String> colSubject = new
   TableColumn<>("Subject");
colSubject.setCellValueFactory(cell -> cell.getValue().
   subjectProperty());

TableColumn<TimetableEntry, String> colType = new TableColumn
   <>("Type");
colType.setCellValueFactory(cell -> cell.getValue().
   typeProperty());

table.getColumns().addAll(colTime, colSubject, colType);

ObservableList<TimetableEntry> entries = FXCollections.
   observableArrayList();

try (Connection conn = DBConnection.getConnection();
PreparedStatement pst = conn.prepareStatement(
"SELECT␣t.start_time,␣t.end_time,␣s.subject_name,␣t.type␣" +
"FROM␣timetable␣t␣LEFT␣JOIN␣subjects␣s␣ON␣t.subject_id␣=␣s.
   subject_id␣" +
"WHERE␣t.group_name␣=␣?␣AND␣t.day_of_week␣=␣?␣ORDER␣BY␣t.
   start_time")) {
```

```
 pst.setString(1, groupName);
 pst.setString(2, LocalDate.now().getDayOfWeek().toString());

 ResultSet rs = pst.executeQuery();
 while (rs.next()) {
 String time = rs.getTime("start_time") + "␣-␣" + rs.getTime("
    end_time");
 String subject = rs.getString("subject_name");
 if (subject == null) subject = rs.getString("type");
 entries.add(new TimetableEntry(time, subject, rs.getString("
    type")));
 }

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

 table.setItems(entries);
 VBox.setVgrow(table, Priority.ALWAYS);

 box.getChildren().addAll(new Label("Today's␣Timetable"),
    table);
 centerPane.getChildren().setAll(box);
}
```

Full Timetable Display

Listing 4.4 Show Full Timetable

```
private void showFullTimetable() {
 VBox box = new VBox(10);
 box.setPadding(new Insets(10));

 TableView<FullTimetableRow> table = new TableView<>();
```

```
table.setColumnResizePolicy(TableView.
   CONSTRAINED_RESIZE_POLICY);

TableColumn<FullTimetableRow, String> colDay = new
   TableColumn<>("Day");
colDay.setCellValueFactory(cell -> cell.getValue().
   dayProperty());
table.getColumns().add(colDay);

for (int i = 0; i < 6; i++) {
final int index = i;
TableColumn<FullTimetableRow, String> col = new TableColumn
   <>("Slot␣" + (i + 1));
col.setCellValueFactory(cell -> cell.getValue().slotProperty(
   index));
table.getColumns().add(col);
}

ObservableList<FullTimetableRow> rows = FXCollections.
   observableArrayList();

try (Connection conn = DBConnection.getConnection();
PreparedStatement pst = conn.prepareStatement(
"SELECT␣t.day_of_week,␣t.start_time,␣t.end_time,␣s.
   subject_name␣" +
"FROM␣timetable␣t␣LEFT␣JOIN␣subjects␣s␣ON␣t.subject_id␣=␣s.
   subject_id␣" +
"WHERE␣t.group_name␣=␣?␣ORDER␣BY␣t.day_of_week,␣t.start_time"
   )) {

pst.setString(1, groupName);
ResultSet rs = pst.executeQuery();

String currentDay = "";
```

```
 FullTimetableRow row = null;
 int slotIndex = 0;

 while (rs.next()) {
 String day = rs.getString("day_of_week");
 String subject = rs.getString("subject_name");

 if (!day.equals(currentDay)) {
 if (row != null) rows.add(row);
 row = new FullTimetableRow(day);
 currentDay = day;
 slotIndex = 0;
 }

 row.setSlot(slotIndex++, subject != null ? subject : "Break")
    ;
 }

 if (row != null) rows.add(row);

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

 table.setItems(rows);
 VBox.setVgrow(table, Priority.ALWAYS);

 box.getChildren().addAll(new Label("Full␣Timetable"), table);
 centerPane.getChildren().setAll(box);
}
```

4.1.3 Student Dashboard - Attendance Console

Attendance Summary

Listing 4.5 Show Attendance Summary

```
private void showAttendanceSummary() {
 VBox box = new VBox(10);
 box.setPadding(new Insets(10));

 TableView<AttendanceEntry> table = new TableView<>();
 table.setColumnResizePolicy(TableView.
    CONSTRAINED_RESIZE_POLICY);

 TableColumn<AttendanceEntry, String> colSubject = new
    TableColumn<>("Subject");
 colSubject.setCellValueFactory(cell -> cell.getValue().
    subjectProperty());

 TableColumn<AttendanceEntry, Integer> colPresent = new
    TableColumn<>("Present");
 colPresent.setCellValueFactory(cell -> cell.getValue().
    presentProperty().asObject());

 TableColumn<AttendanceEntry, Integer> colAbsent = new
    TableColumn<>("Absent");
 colAbsent.setCellValueFactory(cell -> cell.getValue().
    absentProperty().asObject());

 table.getColumns().addAll(colSubject, colPresent, colAbsent);

 ObservableList<AttendanceEntry> entries = FXCollections.
    observableArrayList();

 try (Connection conn = DBConnection.getConnection();
 PreparedStatement pst = conn.prepareStatement(
 "SELECT␣s.subject_name,␣" +
 "SUM(CASE␣WHEN␣a.status='Present'␣THEN␣1␣ELSE␣0␣END)␣AS␣
    present,␣" +
 "SUM(CASE␣WHEN␣a.status='Absent'␣THEN␣1␣ELSE␣0␣END)␣AS␣absent
    ␣" +
 "FROM␣attendance␣a␣" +
 "JOIN␣timetable␣t␣ON␣a.slot_id␣=␣t.slot_id␣" +
 "JOIN␣subjects␣s␣ON␣t.subject_id␣=␣s.subject_id␣" +
 "WHERE␣a.student_id␣=␣?␣" +
 "GROUP␣BY␣s.subject_name")) {

 pst.setInt(1, studentId);
 ResultSet rs = pst.executeQuery();

 while (rs.next()) {
 entries.add(new AttendanceEntry(
 rs.getString("subject_name"),
 rs.getInt("present"),
 rs.getInt("absent")
 ));
 }

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

 table.setItems(entries);
 VBox.setVgrow(table, Priority.ALWAYS);

 box.getChildren().addAll(new Label("Attendance␣Summary"),
    table);
 centerPane.getChildren().setAll(box);
}
```

Logout Module

Listing 4.6 Logout Student Dashboard

```
private void logout() {
 Stage stage = (Stage) btnLogout.getScene().getWindow();
 try {
 FXMLLoader loader = new FXMLLoader(getClass().getResource("/
    gui/Main.fxml"));
 stage.setScene(new Scene(loader.load()));
 stage.setTitle("Attendance␣System");
 } catch (Exception e) {
 e.printStackTrace();
 }
}
```

4.1.4 Teacher Dashboard

Previous Slots

Listing 4.7 Load Previous Slots for Teacher

```
private void loadPreviousSlots() {
 VBox box = new VBox(10);
 box.setPadding(new Insets(10));

 String day = LocalDate.now().getDayOfWeek().toString();
 Time now = Time.valueOf(LocalTime.now());

 String sql = """
␣SELECT␣t.slot_id,␣t.start_time,␣t.end_time,␣t.group_name,␣s.
   subject_name
␣FROM␣timetable␣t
␣JOIN␣subjects␣s␣ON␣t.subject_id␣=␣s.subject_id
␣WHERE␣s.teacher_id␣=␣?␣AND␣t.day_of_week␣=␣?␣AND␣t.end_time␣<
   ␣?
␣ORDER␣BY␣t.end_time␣DESC
␣""";

 try (Connection conn = DBConnection.getConnection();
 PreparedStatement pst = conn.prepareStatement(sql)) {

 pst.setInt(1, teacherId);
 pst.setString(2, day);
 pst.setTime(3, now);
 ResultSet rs = pst.executeQuery();

 while (rs.next()) {
 int slotId = rs.getInt("slot_id");
 String time = rs.getTime("start_time") + "␣-␣" + rs.getTime("
    end_time");
 String group = rs.getString("group_name");
 String subject = rs.getString("subject_name");

 Button slotBtn = new Button(subject + "␣|␣" + group + "␣|␣" +
     time);
 slotBtn.setMaxWidth(Double.MAX_VALUE);
 slotBtn.setOnAction(e -> markAttendanceForSlot(slotId, group)
    );

 box.getChildren().add(slotBtn);
 }

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

 contentPane.getChildren().clear();
 contentPane.getChildren().add(box);
}
```

Attendance Marking

Listing 4.8 Mark Attendance for a Slot

```
private void markAttendanceForSlot(int slotId, String group) {
 VBox box = new VBox(10);
 box.setPadding(new Insets(10));

 Map<Integer, String> attendanceMap = new LinkedHashMap<>();
 List<Integer> studentIds = TeacherAttendanceApp.
    getAllStudentIds(group);
 Map<Integer, String> studentMap = TeacherAttendanceApp.
    getStudentMapForGroup(group);

 for (Integer studentId : studentIds) {
 HBox row = new HBox(10);
 Label lbl = new Label(studentMap.get(studentId));

 Button btnP = new Button("P");
 Button btnA = new Button("A");
 Button btnL = new Button("L");

 btnP.setOnAction(e -> attendanceMap.put(studentId, "Present")
    );
 btnA.setOnAction(e -> attendanceMap.put(studentId, "Absent"))
    ;
 btnL.setOnAction(e -> attendanceMap.put(studentId, "Late"));

 row.getChildren().addAll(lbl, btnP, btnA, btnL);
 box.getChildren().add(row);
 }

 Button saveBtn = new Button("Save␣Attendance");
 saveBtn.setStyle("-fx-background-color:#3b82f6;␣-fx-text-fill
    :white;");
 saveBtn.setOnAction(e -> {
 TeacherAttendanceApp.markAttendance(slotId, attendanceMap);
 loadPreviousSlots();
 });

 box.getChildren().add(saveBtn);
 contentPane.getChildren().clear();
 contentPane.getChildren().add(box);
}
```

Teacher's Timetable

Listing 4.9 Teacher Full Timetable

```
private List<TimetableEntry> getTimetableForTeacher(int
   teacherId) {
 List<TimetableEntry> list = new ArrayList<>();
 String sql = """
␣SELECT␣t.start_time,␣t.end_time,␣s.subject_name,␣t.group_name
   ,␣t.day_of_week
␣FROM␣timetable␣t
␣JOIN␣subjects␣s␣ON␣t.subject_id␣=␣s.subject_id
␣WHERE␣s.teacher_id␣=␣?
␣ORDER␣BY␣t.day_of_week,␣t.start_time
␣""";

 try (Connection conn = DBConnection.getConnection();
 PreparedStatement pst = conn.prepareStatement(sql)) {

 pst.setInt(1, teacherId);
 ResultSet rs = pst.executeQuery();

 while (rs.next()) {
 list.add(new TimetableEntry(
 rs.getTime("start_time"),
 rs.getTime("end_time"),
 rs.getString("subject_name"),
 rs.getString("group_name"),
 rs.getString("day_of_week")
 ));
 }

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

 return list;
}
```

Teacher's Logout GUI

Listing 4.10 Logout Teacher Dashboard

```
private void logout() {
 Stage stage = (Stage) btnLogout.getScene().getWindow();
 try {
 FXMLLoader loader = new FXMLLoader(getClass().getResource("/
    gui/Main.fxml"));
 stage.setScene(new Scene(loader.load()));
 stage.setTitle("Attendance␣System");
 } catch (Exception e) {
 e.printStackTrace();
 }
}
```

4.1.5 Admin Dashboard Controller

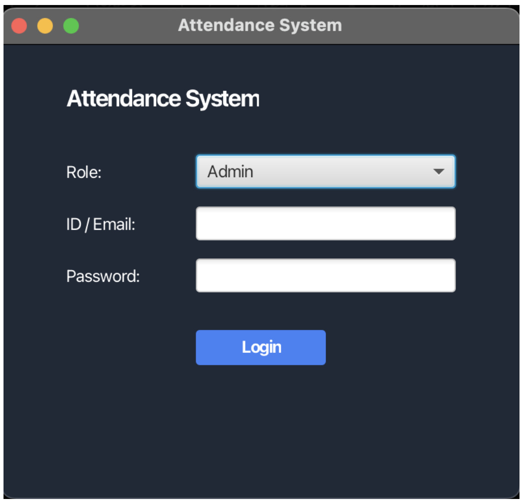
Listing 4.11 AdminDashboardController.java

```
package gui.Controllers;
```

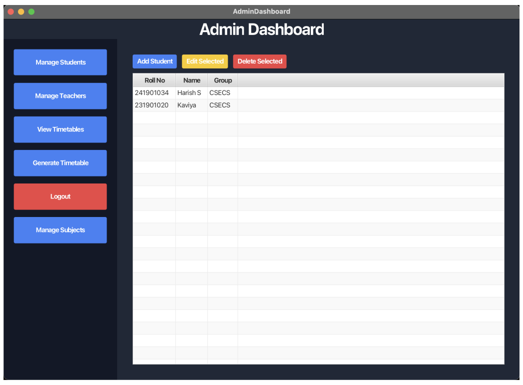
```
import javafx.fxml.FXML;
import javafx.scene.layout.StackPane;
import javafx.scene.layout.VBox;
import javafx.scene.control.Button;
import javafx.scene.layout.Priority;
import javafx.geometry.Insets;
import javafx.stage.Stage;
import javafx.scene.Scene;
import javafx.fxml.FXMLLoader;

public class AdminDashboardController {

 @FXML private StackPane centerPane;
 @FXML private Button btnStudents;
 @FXML private Button btnTeachers;
 @FXML private Button btnTimetable;
 @FXML private Button btnGenerate;
 @FXML private Button btnSubjects;
 @FXML private Button btnLogout;

 @FXML
 private void initialize() {
 btnStudents.setOnAction(e -> loadStudentManagement());
 btnTeachers.setOnAction(e -> loadTeacherManagement());
 btnTimetable.setOnAction(e -> loadTimetableView());
 btnGenerate.setOnAction(e -> generateTimetable());
 btnSubjects.setOnAction(e -> loadSubjectManagement());
 btnLogout.setOnAction(e -> logout());
 }

 private void loadStudentManagement() {
 VBox box = new VBox(10);
 box.setPadding(new Insets(10));
 // Add student management controls here
```

```
centerPane.getChildren().setAll(box);
}

private void loadTeacherManagement() {
VBox box = new VBox(10);
box.setPadding(new Insets(10));
// Add teacher management controls here
centerPane.getChildren().setAll(box);
}

private void loadTimetableView() {
VBox box = new VBox(10);
box.setPadding(new Insets(10));
// Add timetable viewing UI here
centerPane.getChildren().setAll(box);
}

private void generateTimetable() {
VBox box = new VBox(10);
box.setPadding(new Insets(10));
// Generate timetable algorithm or UI
centerPane.getChildren().setAll(box);
}

private void loadSubjectManagement() {
VBox box = new VBox(10);
box.setPadding(new Insets(10));
// Add subject management UI
centerPane.getChildren().setAll(box);
}

private void logout() {
Stage stage = (Stage) btnLogout.getScene().getWindow();
try {
 FXMLLoader loader = new FXMLLoader(
 getClass().getResource("/gui/Main.fxml"));
 stage.setScene(new Scene(loader.load()));
 stage.setTitle("Attendance␣System");
 } catch (Exception e) {
 e.printStackTrace();
 }
 }
}
```

4.2 System Interface and Module Snapshots

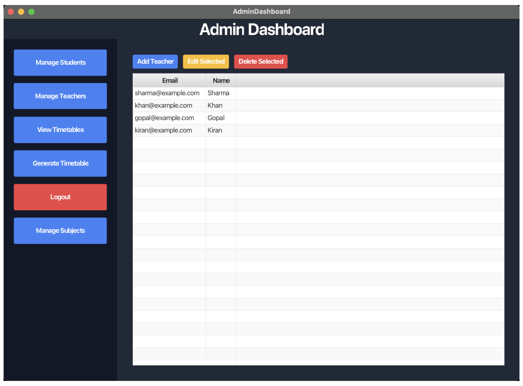


(a) Login Page



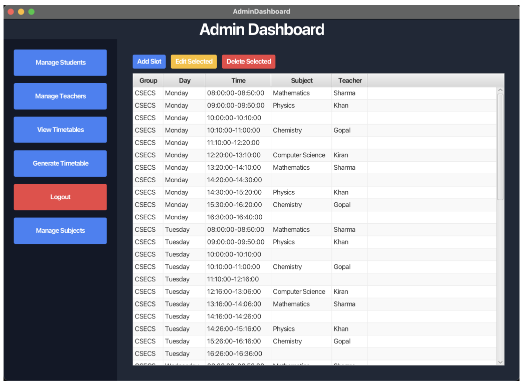
Roll No	Name	Group
241901034	Harish S	CSECS
231901020	Kaviya	CSECS

(b) Admin - Students



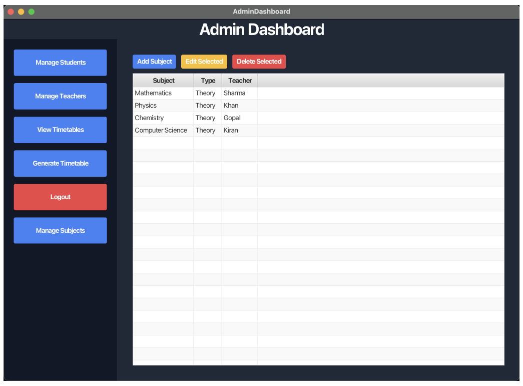
Email	Name
sharma@example.com	Sharma
khan@example.com	Khan
gopal@example.com	Gopal
kiran@example.com	Kiran

(c) Admin - Teachers



Group	Day	Time	Subject	Teacher
CSECS	Monday	08:00:00-08:50:00	Mathematics	Sharma
CSECS	Monday	09:00:00-09:50:00	Physics	Khan
CSECS	Monday	10:00:00-10:10:00		
CSECS	Monday	10:10:00-11:00:00	Chemistry	Gopal
CSECS	Monday	11:10:00-12:20:00		
CSECS	Monday	12:20:00-13:10:00	Computer Science	Kiran
CSECS	Monday	13:20:00-14:10:00	Mathematics	Sharma
CSECS	Monday	14:20:00-14:30:00		
CSECS	Monday	14:30:00-15:20:00	Physics	Khan
CSECS	Monday	15:30:00-16:20:00	Chemistry	Gopal
CSECS	Monday	16:30:00-16:40:00		
CSECS	Tuesday	08:00:00-08:50:00	Mathematics	Sharma
CSECS	Tuesday	09:00:00-09:50:00	Physics	Khan
CSECS	Tuesday	10:00:00-10:10:00		
CSECS	Tuesday	10:10:00-11:00:00	Chemistry	Gopal
CSECS	Tuesday	11:10:00-12:16:00		
CSECS	Tuesday	12:16:00-13:06:00	Computer Science	Kiran
CSECS	Tuesday	13:16:00-14:06:00	Mathematics	Sharma
CSECS	Tuesday	14:16:00-14:26:00		
CSECS	Tuesday	14:26:00-15:16:00	Physics	Khan
CSECS	Tuesday	15:26:00-16:16:00	Chemistry	Gopal
CSECS	Tuesday	16:26:00-16:36:00		

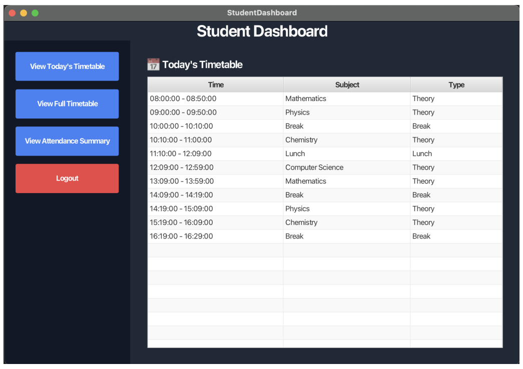
(d) Admin - Timetable



Subject	Type	Teacher
Mathematics	Theory	Sharma
Physics	Theory	Khan
Chemistry	Theory	Gopal
Computer Science	Theory	Kiran

(e) Admin - Subjects

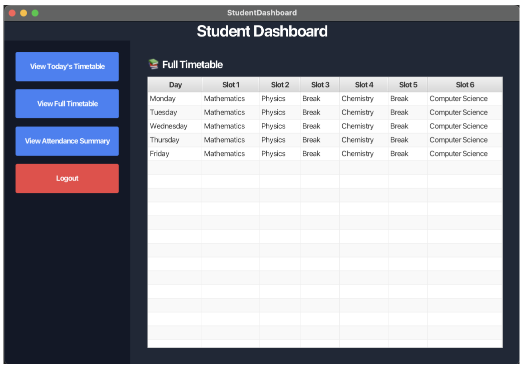
Fig. 4.1 System Interface – Admin Modules



📅 Today's Timetable

Time	Subject	Type
08:00:00 - 08:50:00	Mathematics	Theory
09:00:00 - 09:50:00	Physics	Theory
10:00:00 - 10:10:00	Break	Break
10:10:00 - 11:00:00	Chemistry	Theory
11:10:00 - 12:09:00	Lunch	Lunch
12:09:00 - 12:59:00	Computer Science	Theory
13:09:00 - 13:59:00	Mathematics	Theory
14:09:00 - 14:19:00	Break	Break
14:19:00 - 15:09:00	Physics	Theory
15:19:00 - 16:09:00	Chemistry	Theory
16:19:00 - 16:29:00	Break	Break

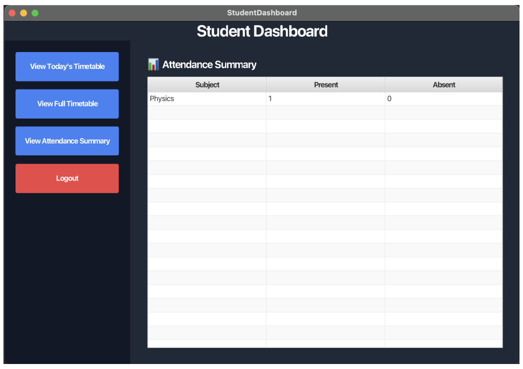
(a) Student Dashboard



📚 Full Timetable

Day	Slot 1	Slot 2	Slot 3	Slot 4	Slot 5	Slot 6
Monday	Mathematics	Physics	Break	Chemistry	Break	Computer Science
Tuesday	Mathematics	Physics	Break	Chemistry	Break	Computer Science
Wednesday	Mathematics	Physics	Break	Chemistry	Break	Computer Science
Thursday	Mathematics	Physics	Break	Chemistry	Break	Computer Science
Friday	Mathematics	Physics	Break	Chemistry	Break	Computer Science

(b) Student Timetable

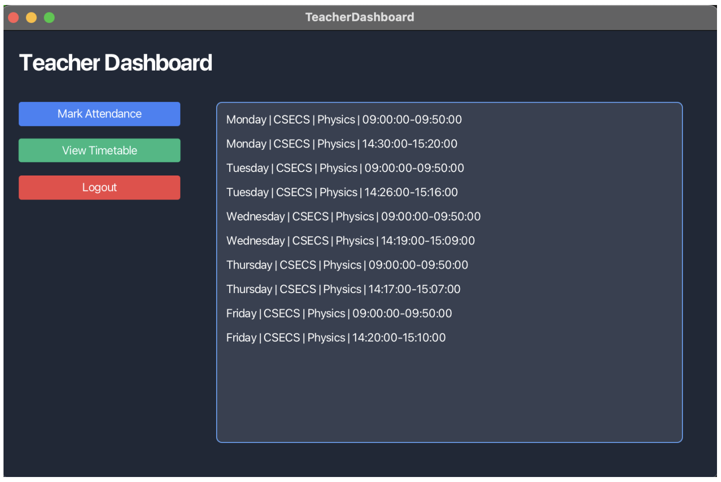


📊 Attendance Summary

Subject	Present	Absent
Physics	1	0

(c) Student Attendance

Fig. 4.2 System Interface – Student Modules



Monday | CSECS | Physics | 09:00:00-09:50:00
Monday | CSECS | Physics | 14:30:00-15:20:00
Tuesday | CSECS | Physics | 09:00:00-09:50:00
Tuesday | CSECS | Physics | 14:26:00-15:16:00
Wednesday | CSECS | Physics | 09:00:00-09:50:00
Wednesday | CSECS | Physics | 14:19:00-15:09:00
Thursday | CSECS | Physics | 09:00:00-09:50:00
Thursday | CSECS | Physics | 14:17:00-15:07:00
Friday | CSECS | Physics | 09:00:00-09:50:00
Friday | CSECS | Physics | 14:20:00-15:10:00

(a) Teacher - Timetable

Fig. 4.3 System Interface – Teacher Module

Chapter 5

Development Lifecycle

5.1 Phased Development Plan

The Smart Attendance Management System was made in a planned and step by step way to make sure that everything was clear, consistent, and moving forward quickly. The method was broken down into a few main steps, which are shown below.

Phase 1: Finding Problems and Analyzing Requirements

This phase looked at the main challenge of taking attendance by hand and the problems with the systems that were already in place. Faculty and students were asked for their needs to figure out what features were most important, such as automatic attendance marking, scheduling based on the schedule, and report generating.

Phase 2: System Design

The system's architecture was planned based on the needs. This entailed making Entity Relationship (ER) diagrams, designing database schemas, and showing how data would move between modules. Modularity and scalability were given special consideration so that IoT and AI based attendance monitoring might be added in the future.

Phase 3: Database and Backend Development

The goal of this phase was to set up the database structure using MySQL and the server side logic with Java. The main functions, such CRUD (Create, Read, Update, Delete), authentication, and CSRF protection, were made and tested to make sure they were safe and worked with data.

Phase 4: User Interface Development

The user interface was made using JavaFX, which made the desktop environment interactive and easy on the eyes. The login screens, dashboards, and attendance panels in the GUI were all made using an easy-to-understand structure to make it easy to go about. To keep everything looking and feeling the same, all modules used the same color schemes, icons, and controls. The interface works for both admins and students, making it easy to control attendance, see schedules, and get performance data. There was a lot of focus on how responsive, easy to use, and error-free the interface with the backend was.

Phase 5: Integration and Testing

All components were put together, and system-level testing was done to make sure that the frontend, backend, and database all worked together. Functional, usability, and security testing made sure that the system worked as it should and that there were no problems with performance or data conflicts.

Phase 6: Testing and Evaluation

The completed solution was put into place and tested as a separate JavaFX program that worked with a MySQL backend. The program was tested based on things including how easy it was to use, how quickly the system responded, and how reliable the data was. Functional testing made sure that things like registering attendance, seeing the schedule, and logging in worked as they should. The system was tested using many student and admin accounts to make sure it was stable when used by more than one person at a time. Feedback from teachers and students helped make small changes that showed the app was generally efficient and strong.

Phase 7: Future Enhancements

In the future, we will include IoT-based attendance (RFID/face recognition), real-time analytics dashboards, and AI-based engagement detection. These changes will make attendance even more automatic and make it easier to keep an eye on students' academic progress.

5.2 System and Technical Requirements

Hardware Requirements

- **Processor:** Intel Core i3 or higher (2.0 GHz or above)
- **RAM:** Minimum 4 GB (8 GB recommended)
- **Storage:** At least 500 MB free disk space
- **Display:** 1366×768 resolution or higher
- **Peripherals:** Keyboard, Mouse, and optional Biometric / RFID module

Software Requirements

- **Operating System:** Windows 10 / 11, Linux (Ubuntu 20.04+) or macOS
- **Programming Language:** Java
- **Database:** MySQL / SQLite for backend storage
- **Development Tools:** Visual Studio Code, XAMPP / WAMP, or Intellij IDEA

Network and Connectivity Requirements

- **Internet Connection:** Required for synchronization and updates
- **Local Server:** Enabled for LAN based attendance access
- **Ports:** HTTP (80), HTTPS (443), or custom API endpoint
- **Security:** SSL/TLS encryption for data transmission

Additional Tools and APIs

- **Authentication APIs:** Firebase / OAuth 2.0 for user login
- **Data Analytics:** Java with Apache Spark or Power BI integration
- **Cloud Storage:** Google Cloud / AWS / Firebase for backup (Future Development)
- **Version Control:** GitHub for source management

Environmental Setup

- All dependencies and configurations are containerized for easy deployment.
- Environment variables are securely managed to ensure modular scalability.
- The system is optimized for cross-platform operation and responsive design.

Chapter 6

Functional Overview

6.1 Functional Modules and Core Features

The **Java Attendance Monitoring System** is designed as a modular, efficient, and scalable platform that automates attendance recording, enhances transparency, and ensures secure data management. The modules are organized into three conceptual layers: **User Interaction Layer**, **Processing Layer**, and **Data Management Layer**.

1. User Interaction Layer

This layer focuses on how various users such as administrators, faculty, and students interact with the system.

- ✓ **Login & Authentication:** Secure sign in with encrypted credentials and role based access.
- ✓ **Dashboard Interface:** Personalized dashboards display attendance records, statistics, and alerts.
- ✓ **User Roles:** Supports Admin, Faculty, and Student views with tailored privileges.

2. Processing Layer

This layer handles all logical operations, data validation, and communication between modules.

- **Attendance Management:** Core module that records, updates, and validates attendance entries.
- **Notifications & Alerts:** Sends automatic messages to students or parents for low attendance.
- **Report & Analytics Engine:** Generates detailed attendance reports and visual graphs.

3. Data Management Layer

This is the backbone of the system ensuring reliable, consistent, and secure storage of information.

- **Database Control:** MySQL based centralized database with efficient indexing.
- **Security & Recovery:** Implements encrypted storage and automatic backup procedures.
- **Data Integrity:** Ensures that records remain consistent during read/write operations.

Table 6.1 Core Features

Feature	**Description**
Real Time Attendance	Attendance data is recorded and updated instantly.
Automated Notifications	Sends real time alerts for low attendance or schedule updates.
Report Generator	Produces analytical and graphical summaries for review.
Role Based Access	Secure, customized access for different user types.
Data Visualization	Displays trends using charts and graphs.
Scalability	System is easily extensible for AI, mobile, or biometric modules.

Chapter 7

Conclusion

The project met its main goal of creating a safe and efficient attendance management system that reduces the amount of work done by hand and makes institutional operations more open. The system makes sure that attendance information is accurate, dependable, and easy to find by combining real time data processing with automatic record keeping. The system is flexible and scalable, so new features may be added without any problems, while performance and data security will stay high.

The system's suitability for implementation shows that it can be used in schools and businesses, laying the groundwork for digital transformation and more efficient administrative processes.

The project has a lot of room for improvement in the future. For example, it could include a mobile app for access on the go, biometric authentication for more accuracy, AI and ML driven attendance prediction, cloud synchronization for distributed environments, and support for multiple campuses for large institutions. Adding advanced analytics will give administrators even more data driven insights, which will help them make better decisions and run their institutions more efficiently.

References

[1] V. D. Nguyen, H. V. Khoa, T. N. Kieu, E. -N. Huh, Internet of Things Based Intelligent Attendance System: Framework, Practice Implementation, and Application, *Electronics* **11**(19) (2022) 3151. DOI: 10.3390/electronics11193151

[2] T.-W. Chiang *et al.*, Development and Evaluation of an Attendance Tracking System Using Android (GPS/NFC), *Journal of Experimental & Theoretical Artificial Intelligence* (2022). DOI: 10.1080/08839514.2022.2083796

[3] C. Wu *et al.*, Attendance Tracking System Using Many Battery Free Photovoltaic Bluetooth Beacon Badges, *ACM Proceedings on Interactive, Mobile, Wearable and Ubiquitous Technologies*, ACM (2024). DOI: 10.1145/3698384.3699613

[4] Anonymous, Transforming Educational Institutions: Harnessing the Power of IoT, Cloud and AI, *Future Internet* **15**(11) (2023) 367. DOI: 10.3390/fi15110367

[5] P. Bhardwaj, P. K. Gupta, H. Panwar, M. Khubeb, A. Bhaik, Application of Deep Learning on Student Engagement in E-Learning Environments, *Computers & Electrical Engineering* (2021). DOI: 10.1016/j.compeleceng.2021.107277

[6] N. Mahmood, S. M. Bhatti, H. Dawood, M. R. Pradhan, H. Ahmad, Measuring Student Engagement through Behavioral and Emotional Features Using Deep Learning Models, *Algorithms* **17**(10) (2024) 458. DOI: 10.3390/a17100458

[7] X. Jiang *et al.*, Deep Learning Based Multimodal Fusion Algorithm for Recognition of Student Engagement State in a Classroom Environment, *Applied Sciences Conference Proceedings*, ACM (2025). DOI: 10.1145/3759179.3759192

[8] L. Yan *et al.*, Student Engagement Assessment Using Multimodal Deep Learning, *PLOS ONE* (2025). DOI: 10.1371/journal.pone.0325377

[9] F. R. T. Ferreira *et al.*, Development of a Framework Using Deep Learning for Detecting Student Engagement, *Multimedia Tools and Applications*, Springer (2025). DOI: 10.1007/s13278-025-01408-z

[10] Y. Bell *et al.*, Artificial Intelligence Supported Student Engagement: Trend Analysis and Methodologies, *European Journal of Education*, Wiley (2025). DOI: 10.1111/ejed.70008

[11] Y. Bellarhmouch, Detecting Student Engagement in an Online Learning Environment, *International Journal of Educational Research and Innovation* **12**(2) (2025) 44. DOI: 10.3390/ijeri12020044

[12] Y. Bellarhmouch, Predictive Modeling of Learner Engagement Using Exams and Quizzes, *International Journal of Educational Research and Innovation* **12**(2) (2025) 44. DOI: 10.3390/ijeri12020044