ITE 1942 – ICT PROJECT

PROJECT REPORT

Level 01

Student attendants Management System

Submitted by:

R.R.G.S. Rajapaksha

E2410978

Bachelor of Information Technology (External Degree)

Faculty of Information Technology

University of Moratuwa

Table of Contents

Figure

Chapter 1: INTRODUCTION

1.1 Introduction

This project report presents the design and development of a **Computer-Based Student Management and Attendance System**, created to streamline the essential administrative tasks of schools and educational institutions. The system focuses on automating student registration, class management, attendance tracking, and reporting, thereby reducing the inefficiencies and errors associated with manual record-keeping.

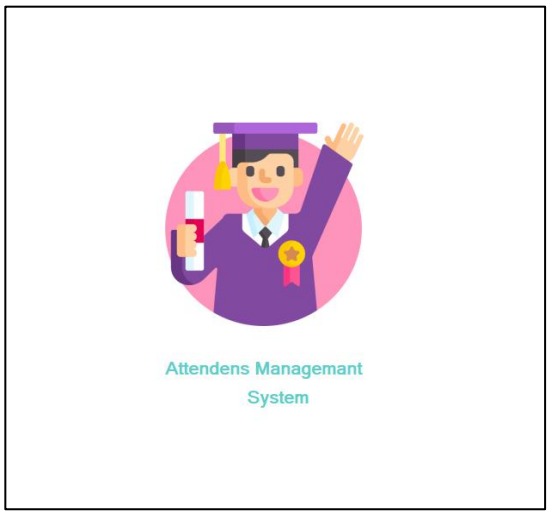
In many rural and semi-urban schools, administrative processes are still handled using paper registers or basic spreadsheets. These methods are time-consuming, error-prone, and lack the ability to provide real-time insights. The proposed system addresses these challenges by offering a **centralized digital platform** that ensures accuracy, improves accessibility, and enhances decision-making for teachers and administrators. By integrating a secure login mechanism, role-based access, and user-friendly interfaces, the system provides a reliable solution that balances simplicity with functionality.

Ultimately, the system aims to improve the efficiency of school operations, reduce the workload of staff, and ensure accurate documentation of student records and attendance. This contributes to better resource management and improved educational outcomes.

1.1.1 Project Name

Student Management and Attendance System

1.1.2 Logo



1.2 Background

Educational institutions, especially in rural areas, face significant challenges in managing student information and attendance. Manual systems often result in:

- **Time inefficiency**: Teachers spend valuable time recording attendance and preparing reports instead of focusing on teaching.
- **Data duplication and errors**: Paper-based records are prone to mistakes, leading to inaccurate reporting.
- **Limited accessibility**: Retrieving past records is cumbersome and often delayed.
- **Security risks**: Physical records can be lost, damaged, or accessed by unauthorized individuals.

With the rapid growth of digital technologies, there is a pressing need for affordable, easy-to-use systems that can automate these processes. Unlike large-scale Learning Management Systems (LMS) that are costly and complex, this project is designed to be **lightweight, cost-effective, and tailored to the needs of schools with limited resources**.

The system leverages **C# (Windows Forms)** for the user interface and **MySQL** as the backend database, ensuring both performance and scalability. By automating attendance, student registration, and reporting, the system bridges the digital divide and empowers schools to operate more efficiently.

1.3 Manual Handling Issues and Proposed Solution

1.3.1 Manual Handling Issues

1. **Inefficiency in Operations**
    - Recording attendance manually consumes significant time.
    - Preparing reports requires repetitive calculations.
2. **Limited Accessibility**
    - Teachers and administrators struggle to retrieve past records quickly.
    - No real-time visibility of student attendance trends.
3. **Error-Prone Processes**
    - Manual entry increases the risk of duplication and inaccuracies.
    - Difficult to track changes or maintain consistent records.
4. **Security Risks**

- Paper registers can be misplaced or accessed by unauthorized persons.
- Sensitive student data lacks proper protection.

1.3.2 Proposed Solution

The **Student Management and Attendance System** addresses these issues by automating key processes:

- **Digital Attendance**: Teachers can mark attendance with a simple interface, and records are stored securely in the database.
- **Automated Reporting**: The system generates class-wise and student-wise attendance reports instantly.
- **Centralized Database**: All student and class information is stored in one place, ensuring consistency and easy retrieval.
- **Role-Based Access**: Admins, teachers, and students have different levels of access, ensuring security.
- **Error Reduction**: Input validation and automated checks minimize human errors.

1.3.3 Project Aim

The aim of this project is to **develop a user-friendly, cost-effective, and reliable Student Management and Attendance System** that automates the core administrative functions of schools. The system is designed to:

- Enhance operational efficiency.
- Reduce manual workload.
- Improve accuracy of student records.
- Provide real-time access to attendance and class data.
- Strengthen data security and integrity.

1.3.3.1 Objectives

To achieve the project aim, the following objectives were defined:

1. **Automate Student Registration**: Enable adding, updating, and deleting student records with validation.
2. **Implement Role-Based Authentication**: Provide secure login for Admins, Teachers, and Students.
3. **Attendance Management**: Allow teachers to mark daily attendance and update records.
4. **Class Management**: Support creation and management of academic classes.
5. **Reporting**: Generate attendance summaries and student-specific reports.

6. **Dashboard Overview**: Provide administrators with real-time statistics on classes, students, and users.
7. **Data Security**: Ensure sensitive information is protected through authentication and controlled access.

1.4 Primary Stakeholders

The primary stakeholders of the system include:

1. **School Administrators**
    - Oversee overall operations and require accurate reports.
2. **Teachers**
    - Responsible for marking attendance and managing student records.
3. **Students**
    - End-users whose records and attendance are maintained.
4. **System Administrators/Developers**

- Maintain the system, ensure security, and implement updates.

1.5 Technology Stack and Development Tools

1. Programming Language – C#:

A powerful and versatile programming language used for developing the Windows desktop application. It supports object-oriented programming, making it suitable for building scalable and maintainable software.

2. Database Management System – MySQL:

An open-source relational database management system used to store and manage the data for the guest house. MySQL is known for its reliability and performance in handling complex queries and large datasets.

3. Development Environment

• Visual Studio: The primary integrated development environment (IDE) used for C# development. It provides powerful debugging tools, code completion, and project management features.

• phpMyAdmin: A web-based tool used for managing and administering the MySQL database. It simplifies tasks such as creating, modifying, and querying databases through an intuitive graphical interface.

4. User Interface Framework• Windows Forms:

A UI framework within the .NET framework used to create rich desktop applications. It allows for the development of interactive and user-friendly interfaces.

Guna UI: A modern UI framework that enhances the visual appeal of Windows Forms applications. Guna UI provides a wide range of customizable controls, animations, and themes, allowing for the creation of a visually engaging and user-friendly interface for the guest house management system.

Version Control

Git: A version control system used to manage the source code and track changes throughout the development process. Git enables collaboration among developers and maintains a history of code changes.

Additional Tools

Canva: Used for designing graphics, flowcharts, and other visual elements for the project documentation.

1.6 Summary

This chapter introduced the **Student Management and Attendance System**, highlighting the background, challenges of manual processes, and the proposed digital solution. The project aims to provide a **lightweight, secure, and efficient platform** for managing students, classes, and attendance. The next chapters will discuss related work, system analysis, design, and implementation in detail.

Chapter 2: RELATED WORK

2.1 Existing Manual Systems

In many schools, particularly in rural and resource-limited areas, student management and attendance tracking are still handled manually. These systems typically rely on paper registers, logbooks, or basic spreadsheets. While simple, such methods introduce several challenges:

1. **Inefficiency in Operations**
    - Teachers spend significant time recording attendance and preparing reports.
    - Administrative staff face delays in compiling student records.
2. **Data Duplication and Errors**
    - Manual entry increases the risk of duplicate or inconsistent records.
    - Errors in attendance or student details can lead to inaccurate reporting.
3. **Limited Accessibility**
    - Retrieving historical data is time-consuming.
    - Records are not centralized, making it difficult to share information across departments.
4. **Security Risks**
    - Paper registers are vulnerable to loss, damage, or unauthorized access.
    - Sensitive student information lacks proper protection.
5. **Error-Prone Processes**

- Updating or modifying records is cumbersome.
- Tracking attendance trends or generating summaries is nearly impossible without manual calculations.

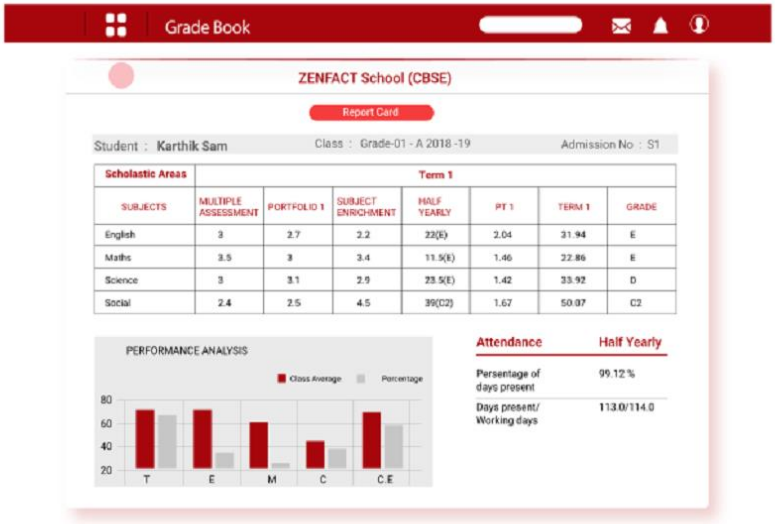
2.2 Existing Digital Solutions

Several digital platforms exist to address the challenges of student management and attendance. However, most are either too complex, too costly, or not tailored to the needs of smaller institutions. Below are some notable systems:

1. Fedena School ERP

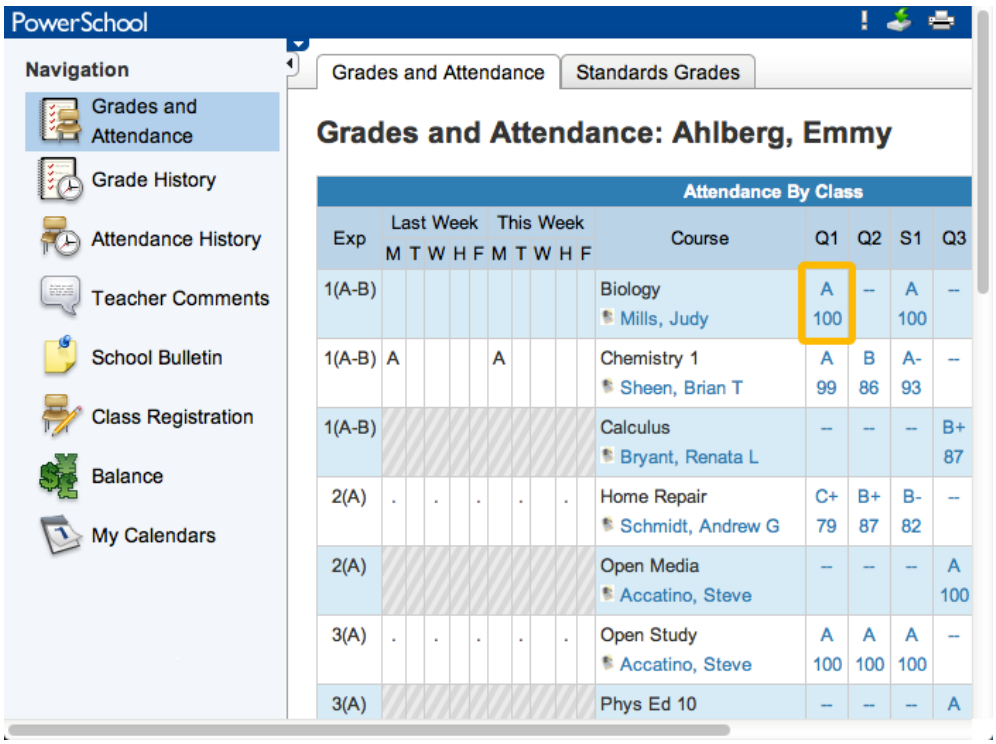
- **Key Features**: Student information management, attendance tracking, examinations, fee management, communication tools. (Foradian Technologies, 2023)

- **Limitations**: Requires technical expertise for setup and maintenance; often too complex for small schools.



2. PowerSchool

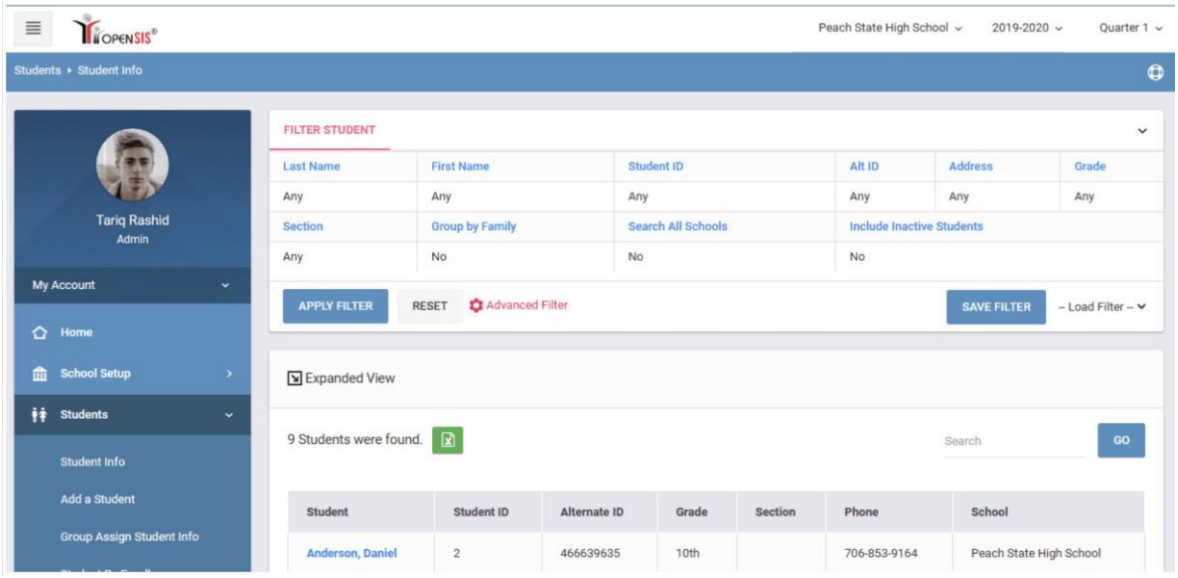
- **Key Features**: Comprehensive student data management, attendance, grading, parent and student portals, advanced analytics. (PowerSchool, 2023)
- **Limitations**: High cost and infrastructure requirements make it unsuitable for rural schools.



3. OpenSIS

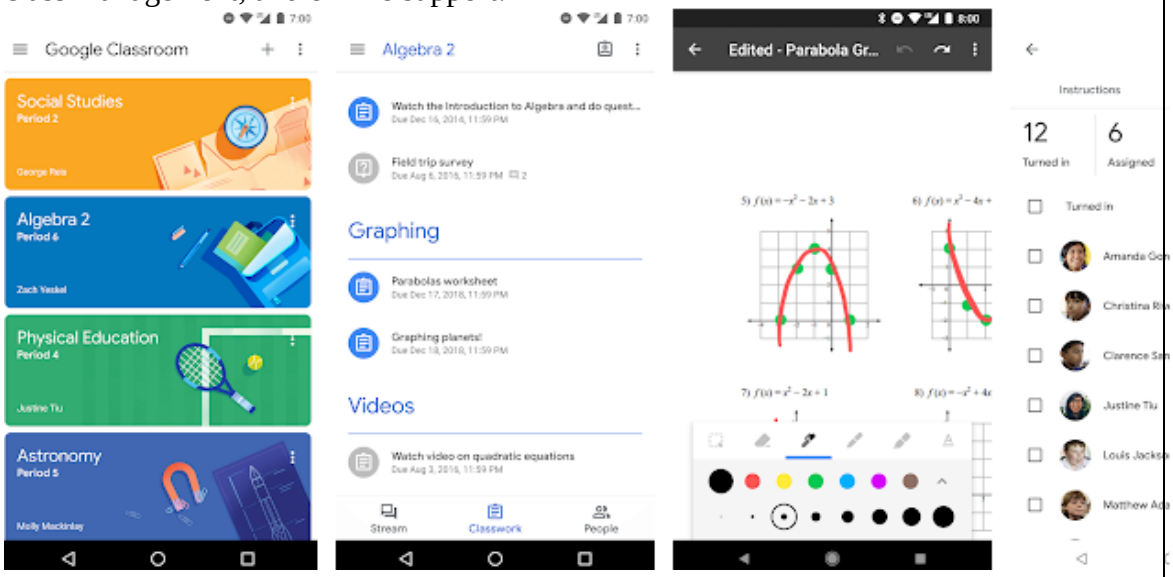
- **Key Features**: Open-source student information system with attendance, scheduling, and reporting. (OS4ED (Open Solutions for Education, Inc.), 2023)

Limitations: Requires IT knowledge to install and maintain; not user-friendly for non-technical staff.



4. Google Classroom

- **Key Features**: Assignment distribution, grading, teacher-student communication, integration with Google Workspace. (Google, 2023)
- **Limitations**: Not a full student management system; lacks attendance tracking, class management, and offline support.



2.3 Identified Gaps

From the review of manual and digital systems, the following gaps are evident:

- **Complexity**: Many existing systems are designed for large institutions and are too complex for small schools.
- **Cost**: Commercial platforms like PowerSchool are expensive and require significant infrastructure.
- **Technical Barriers**: Open-source solutions like OpenSIS demand technical expertise that rural schools often lack.
- **Feature Gaps**: Tools like Google Classroom do not provide core features such as attendance tracking or class management.

2.4 Conclusion

The limitations of manual systems and the shortcomings of existing digital solutions highlight the need for a **lightweight, affordable, and user-friendly Student Management and Attendance System**. The proposed system fills this gap by focusing on essential features — secure login, student registration, class management, attendance tracking, and reporting — while ensuring simplicity and accessibility for schools with limited resources.

Chapter 3: SYSTEM ANALYSIS

3.1 Requirement Gathering Process

To ensure the system meets the needs of all stakeholders, a structured requirement gathering process was followed:

1. **Identify Stakeholders**
    - **Internal Stakeholders**: School administrators, teachers, IT support staff.
    - **External Stakeholders**: Students, parents (indirectly benefiting from accurate records).
2. **Define Objectives**
    - Automate attendance and student management.
    - Provide accurate, real-time reporting.
    - Ensure secure access through role-based authentication.
3. **Techniques Used**

- **Interviews** with teachers and administrators to understand pain points.
- **Observation** of manual attendance processes.
- **Document Analysis** of existing registers and spreadsheets.

3.2 Functional Requirements

The system must support the following core functions:

1. **User Authentication and Role Management**
    - Secure login for Admins, Teachers, and Students.
    - Role-based access to features.
2. **Student Management**
    - Add, update, delete, and search student records.
    - Store details such as name, registration number, class, gender, DOB, CNIC, and contact information.
3. **Class Management**
    - Create and manage academic classes.
    - Assign students to classes.
4. **Attendance Management**
    - Mark daily attendance for each class.
    - Update existing attendance records.
    - Store attendance status (Present/Absent).
5. **Reporting**

- Generate class-wise attendance summaries.
- Produce student-specific attendance history.
- Provide dashboard statistics (total classes, students, users).

3.3 Non-Functional Requirements

1. **Performance**
    - Attendance saving operations should complete within 2–3 seconds.

    - Reports should generate within 5–10 seconds.
2. **Usability**
    - Intuitive interface requiring minimal training.
    - Clear error messages and tooltips for guidance.
3. **Security**
    - Role-based access control.
    - Secure authentication for all users.
4. **Scalability**
    - Support increasing numbers of students, classes, and users.
5. **Reliability**

- Ensure 99% system availability during academic hours.
- Maintain data integrity across all transactions.

3.4 Technical Requirements

- **Hardware:** Standard desktop/laptop with minimum 4GB RAM, dual-core processor.
- **Software:** Windows OS, Visual Studio, MySQL Server, phpMyAdmin.
- **Database:** MySQL relational database with tables for Users, Students, Classes, and Attendance.

3.5 Inputs & Outputs of Each Function

Function	Subfunction	Inputs	Outputs
User Login	Enter Credentials	Username, Password	Credentials captured
	Validate Credentials	Username, Password → Query `User Table`	Valid/Invalid result
	Role Assignment	User_Role from DB	Redirect to Admin Dashboard / Student Dashboard
	Error Handling	Invalid credentials	Error message displayed
Student Managemen	Add Student	Name, RegNo, Class, Gender, DOB, CNIC, Email, Phone, Address	New student record inserted into `Student_Table`
	Update Student	Student_ID, updated details	Record updated in database
	Delete Student	Student_ID	Record removed from database
	Search Student	Search term (Name/RegNo/Class)	Matching records displayed in DataGridView
Class Managemen	Create Class	Class Name, Class ID	New class record inserted into `Class_Table`
	Assign Students	Student_ID, Class_ID	Student assigned to class
	View Classes	Query request	List of classes displayed
Attendance Managemen	Load Students	Class_ID, Date	Student list with default attendance status
	Mark Attendance	Student_ID, Checkbox (Present/Absent)	Temporary attendance status in grid
	Save Attendance	Student_ID, Date, Status	Insert/Update in `Attendance_Table`

	Update Attendance	Existing record (Student_ID, Date)	Attendance status updated
Generate Report	Generate Class Report	Class, Date	Class-wise attendance summary (Total, Present, Absent, Rate)
	Generate Student Report	Student_ID	Student-specific attendance history
	Export/Print Report	Report data	Printable/exportable formatted report
Dashboard Statistics	Count Classes	Query `Class_Tab`	Total number of classes displayed
	Count Students	Query `Student_T`	Total number of students displayed
	Count Users	Query `User_Tabl`	Total number of users displayed

Table 1(Inputs & Outputs of Each Function)

3.6 Summary

This section detailed the **inputs and outputs** of each major function, broken down into **sub functions**. By analyzing the system at this granular level, it becomes clear how each module (Login, Student Management, Class Management, Attendance, Reporting, Dashboard) interacts with the database and the user interface. This structured breakdown ensures that all requirements are traceable to specific system operations, improving both design clarity and implementation accuracy

Chapter 4: SYSTEM DESIGN

4.1 Flowcharts

4.1.1 User Authentication Process

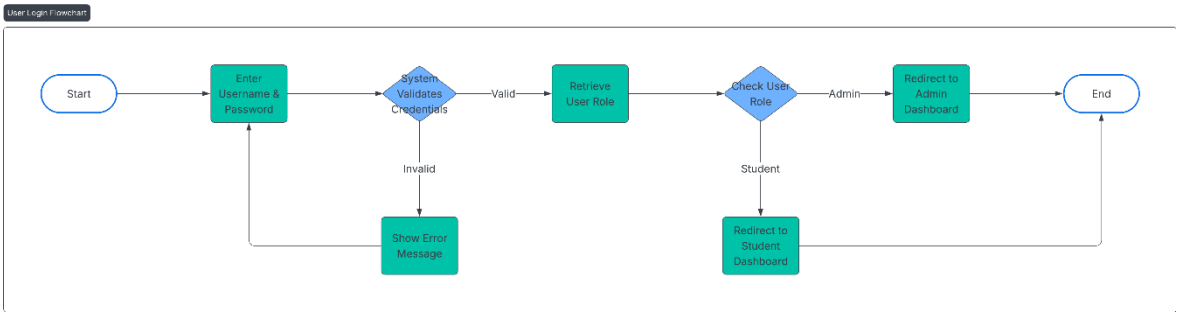


Figure 1(User Authentication Process FlowChart)

Flow:

4.1.2 User Registration Process

Flow:

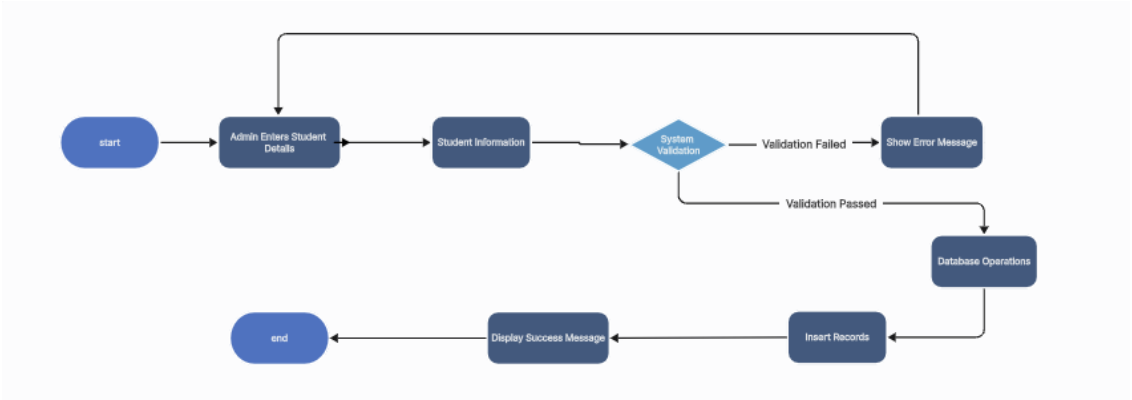


Figure 2User Registration Process

4.1.3 Attendance Marking Process

Flow:

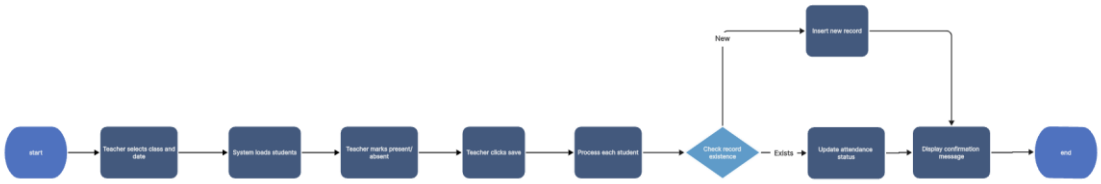


Figure 3Attendance Marking Process

4.1.4 Report Generation Process

Flow:

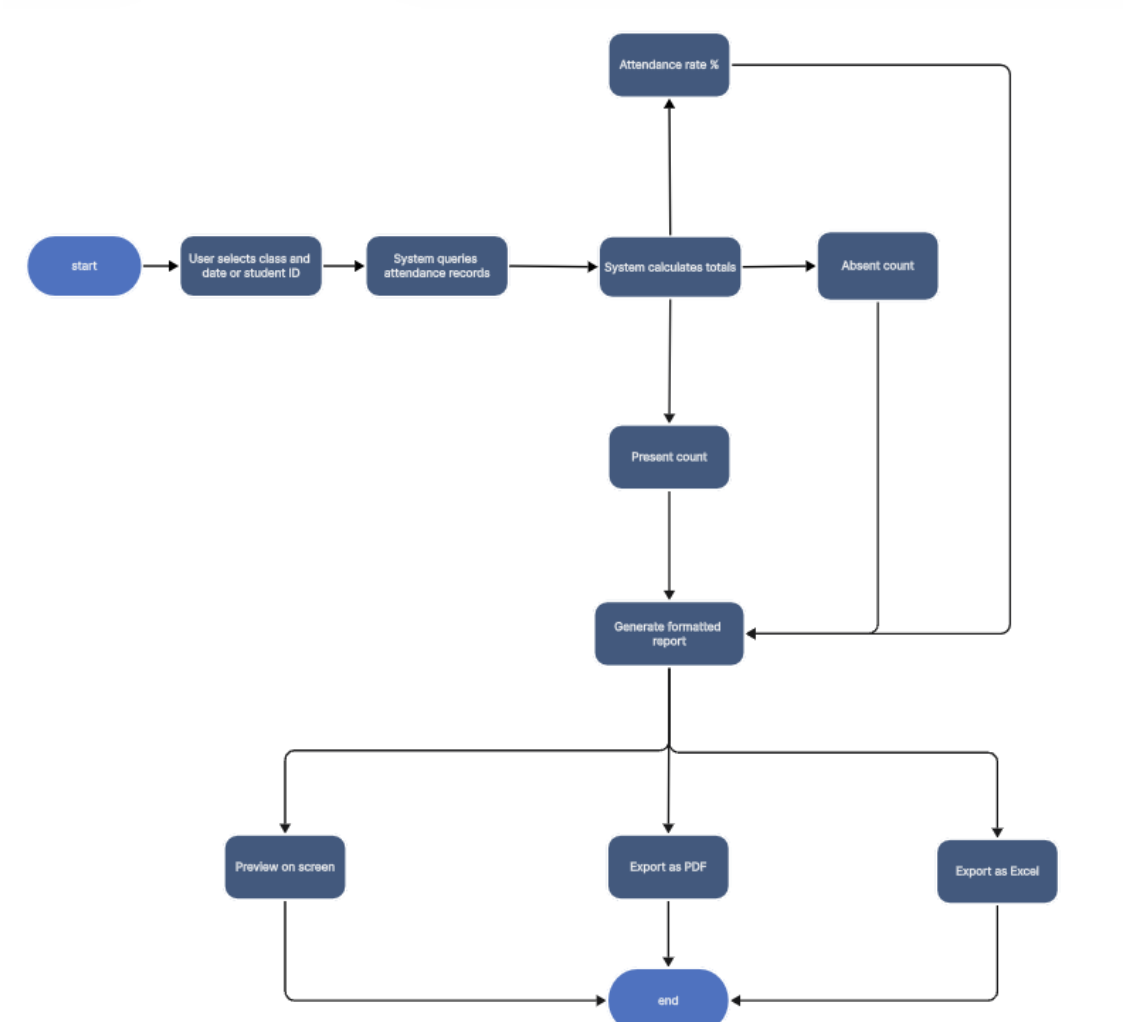


Figure 4Report Generation Process

4.1.5 Dashboard Statistics

Flow:

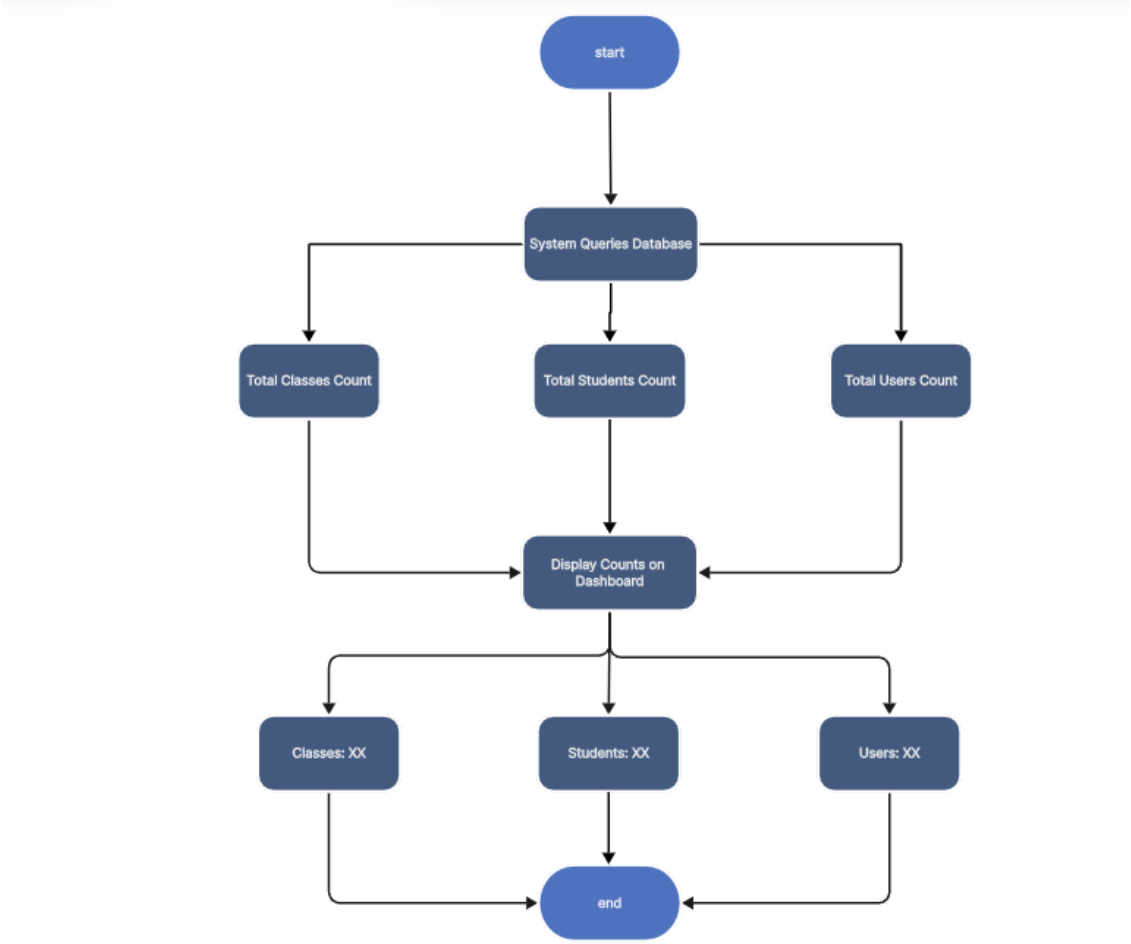


Figure 5Figure 5Dashboard Statistics

4.2 Pseudocode

4.2.1 User Authentication

```
START
INPUT username, password
QUERY User_Table WHERE User_name = username AND User_Pass = password
IF result FOUND THEN
   role = result.User_Role
   IF role == "Admin" THEN
      Redirect to Admin Dashboard
   ELSE IF role == "Student" THEN
      Redirect to Student Dashboard
   END IF
```

```
ELSE
   DISPLAY "Invalid Username or Password"
END IF
END
```

4.2.2 User Registration

```
START
INPUT student details (Name, RegNo, Class, Gender, DOB, CNIC, Email,
Phone, Address)
VALIDATE all fields
IF validation fails THEN
   DISPLAY "Error: Invalid or missing data"
   RETURN
END IF
CHECK for duplicates in Student_Table
IF duplicate found THEN
   DISPLAY "Error: Student already exists"
   RETURN
END IF
INSERT INTO Student_Table and User_Table
DISPLAY "Student registered successfully"
END
```

4.2.3 Attendance Saving

```
START
INPUT classId, date
FOR EACH student IN class
   status = (checkbox checked) ? "Present" : "Absent"
   QUERY attendance_table WHERE Student_ID = student.ID AND
Attendance_Date = date
   IF record exists THEN
      UPDATE attendance_table SET Attendance_Status = status
   ELSE
      INSERT INTO attendance_table (Student_ID, Attendance_Date,
Attendance_Status)
   END IF
END FOR
```

DISPLAY "Attendance saved successfully"
END

4.2.4 Report Generation

START

INPUT classId, date

QUERY attendance_table JOIN student_table WHERE Class_ID = classId AND Attendance_Date = date

CALCULATE total_students, present_count, absent_count

attendance_rate = (present_count / total_students) * 100

DISPLAY report with summary and detailed records

END

4.2.5 Dashboard Statistics

START

QUERY Class_Table COUNT(*) → total_classes

QUERY Student_Table COUNT(*) → total_students

QUERY User_Table COUNT(*) → total_users

DISPLAY totals on dashboard

END

Chapter 5: SYSTEM IMPLEMENTATION

5.1 Login Form

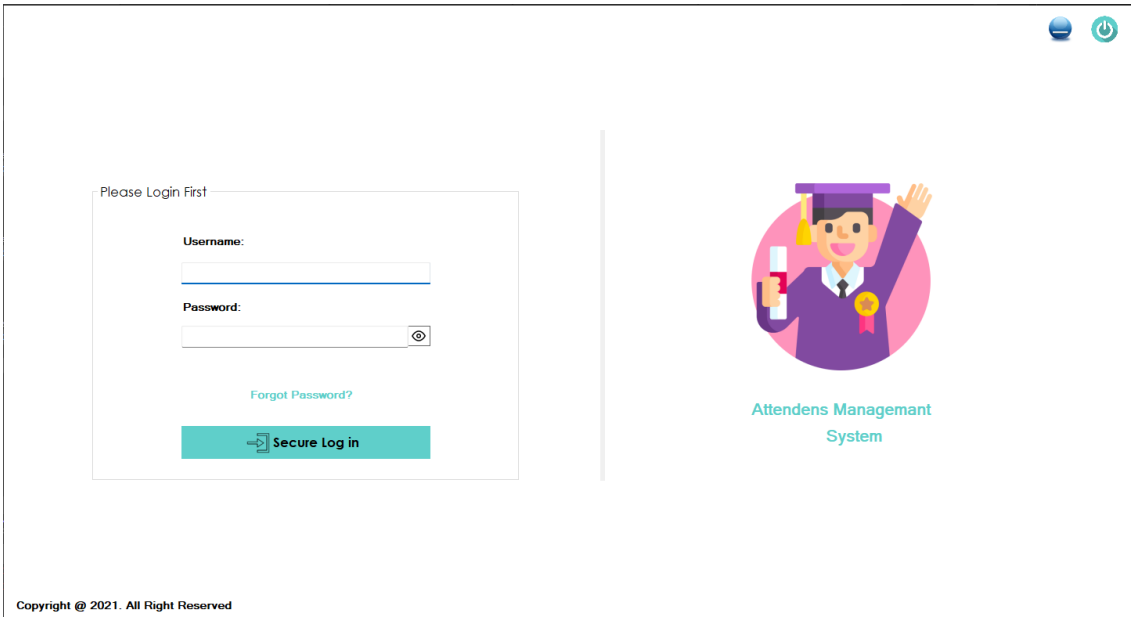


Figure 6 UI Login From

The **Login Form** provides secure access to the system. It validates user credentials against the `User_Table` in the database and redirects users based on their role.

- **Features**:

- Username and password input.
- Password show/hide toggle.
- Role-based redirection (Admin → full access, Student → restricted access).
- Error messages for invalid login.
- Forgot Password link.

```
using MySql.Data.MySqlClient;
using System;
using System.Windows.Forms;

namespace Student_Managemant.PLA.Forms
{
    public partial class FormLogin : Form
    {
        private string connectionString = "Server=localhost;
Database=attendens_managment_system; Uid=root; Pwd=;";

        public FormLogin()
        {
            InitializeComponent();
        }
```

```
        private void buttonLogin_Click(object sender, EventArgs e)
        {
            string username = textBoxUsername.Text.Trim();
            string password = textBoxPassword.Text.Trim();

            if (string.IsNullOrEmpty(username) ||
string.IsNullOrEmpty(password))
            {
                MessageBox.Show("Please enter both username and
password.", "Validation Error", MessageBoxButtons.OK,
MessageBoxIcon.Warning);
                return;
            }

            string query = "SELECT User_Name, User_Role FROM
User_Table WHERE User_Name = @username AND User_Pass = @password";

            try
            {
                using (MySqlConnection connection = new
MySqlConnection(connectionString))
                {
                    connection.Open();
                    using (MySqlCommand command = new
MySqlCommand(query, connection))
                    {
                        command.Parameters.AddWithValue("@username",
username);
                        command.Parameters.AddWithValue("@password",
password);

                        using (MySqlDataReader reader =
command.ExecuteReader())
                        {
                            if (reader.Read())
                            {
                                string role =
reader["User_Role"].ToString();
                                FormMain mainForm = new FormMain();
                                mainForm.Username = username;
                                mainForm.Role = role;
                                this.Hide();
                                mainForm.ShowDialog();
                                this.Close();
                            }
                            else
```

```
                            {
                                MessageBox.Show("Invalid username or
password.", "Login Failed", MessageBoxButtons.OK,
MessageBoxIcon.Error);
                            }
                        }
                    }
                }
            }
            catch (Exception ex)
            {
                MessageBox.Show("Database connection error: " +
ex.Message, "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
            }
        }

        private void buttonExit_Click(object sender, EventArgs e)
        {
            Application.Exit();
        }

        private void checkBoxShowPassword_CheckedChanged(object
sender, EventArgs e)
        {
            textBoxPassword.UseSystemPasswordChar =
!checkBoxShowPassword.Checked;
        }

        private void linkLabelForgetPassword_LinkClicked(object
sender, LinkLabelLinkClickedEventArgs e)
        {
            FormForgetPassword forgetPasswordForm = new
FormForgetPassword();
            forgetPasswordForm.ShowDialog();
        }
    }
```

5.2 Main Dashboard Form

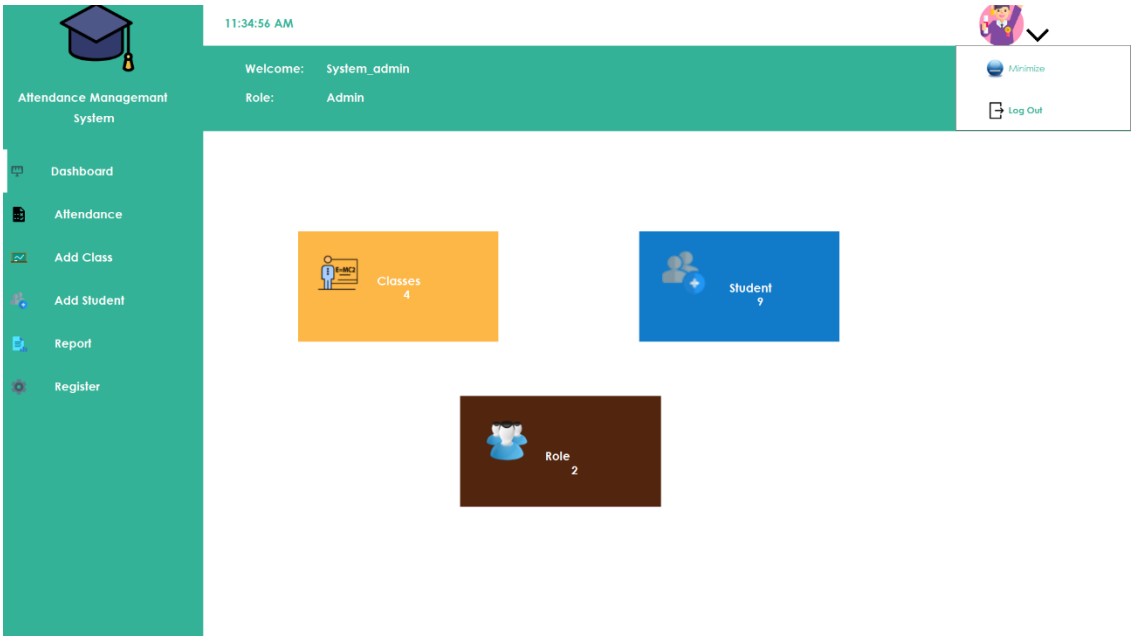


Figure 7Main Dashboard Form

The **Main Form** acts as the central navigation hub after login. It displays the logged-in user's name and role, and provides access to all modules.

- **Features**:

- Side panel navigation (Dashboard, Attendance, Add Class, Add Student, Reports, Register).
- Role-based visibility (Students cannot access Admin-only features).
- Real-time clock display.
- Expand/Collapse panel for additional options.

```
using System;
using System.Data;
using System.Drawing;
using System.Windows.Forms;

namespace Student_Managemant.PLA.Froms
{
    public partial class FormMain : Form
    {
        public string Username, Role;

        public FormMain()
        {
            InitializeComponent();
            timerDateAndTime.Start();
        }
```

```
        private void buttonLogOut_Click(object sender,
EventArgs e)
        {
            DialogResult dialogResult = MessageBox.Show("Are
you sure you want to log out?", "Log Out",
MessageBoxButtons.YesNo, MessageBoxIcon.Question);
            if (dialogResult == DialogResult.Yes)
            {
                timerDateAndTime.Stop();
                Close();
            }
            else
            {
                panelExpand.Hide();
            }
        }

        private void FormMain_Load(object sender, EventArgs e)
        {
            panelExpand.Hide();
            labelUsername.Text = Username;
            labelRole.Text = Role;

            if (Role == "Student")
            {
                buttonDashboard.Hide();
                buttonAddClass.Hide();
                buttonAddStudent.Hide();
                userControlDashbord1.Visible = false;
                buttonRegister.Hide();
            }
        }

        private void buttonMinimize_Click(object sender,
EventArgs e)
        {
            panelExpand.Hide();
            WindowState = FormWindowState.Minimized;
        }

        private void timerDateAndTime_Tick(object sender,
EventArgs e)
        {
            DateTime now = DateTime.Now;
            labelTime.Text = now.ToString("hh:mm:ss tt");
        }

        private void buttonDashboard_Click(object sender,
EventArgs e)
        {
            MoveSidePanel(buttonDashboard);
            userControlAddClass1.Visible = false;
            userControlDashbord1.Count();
            userControlDashbord1.Visible = true;
```

```
            userControlAddStudent1.Visible = false;
            userControl1Register1.Visible = false;
        }

        private void buttonAttendance_Click(object sender,
EventArgs e)
        {
            MoveSidePanel(buttonAttendance);
            userControlAttendance1.Visible = true;
            userControlAddClass1.ClearTxtBox();
            userControlAddClass1.Visible = false;
            userControlDashbord1.Visible = false;
            userControlAddStudent1.Visible = false;
            userControl1Register1.Visible = false;
        }

        private void buttonAddClass_Click(object sender,
EventArgs e)
        {
            MoveSidePanel(buttonAddClass);
            userControlAddClass1.Visible = true;
            userControlDashbord1.Visible = false;
            userControlAttendance1.Visible = false;
            userControlAddStudent1.Visible = false;
            userControl1Register1.Visible = false;
        }

        private void buttonAddStudent_Click(object sender,
EventArgs e)
        {
            MoveSidePanel(buttonAddStudent);
            userControlAddClass1.Visible = false;
            userControlDashbord1.Visible = false;
            userControl1Register1.Visible = false;
            userControlAddStudent1.Visible = true;
            userControlAddStudent1.ClearTxtBox();
            userControlAttendance1.Visible = false;
        }

        private void buttonReport_Click(object sender,
EventArgs e)
        {
            MoveSidePanel(buttonReport);
            userControlAddClass1.Visible = false;
            userControlDashbord1.Visible = false;
            userControlAddStudent1.Visible = false;
            userControl1Register1.Visible = false;
            userControlAttendance1.Visible = false;
        }

        private void buttonRegister_Click(object sender,
EventArgs e)
        {
            MoveSidePanel(buttonRegister);
```

```
            userControl1Register1.Visible = true;
            userControlAddClass1.Visible = false;
            userControlDashbord1.Visible = false;
            userControlAddStudent1.Visible = false;
            userControlAttendance1.Visible = false;
            userControlAddStudent1.ClearTxtBox();
        }

        private void MoveSidePanel(Control button)
        {
            panelSlide.Location = new Point(button.Location.X -
button.Location.X, button.Location.Y - 255);
        }

        private void pictureBoxExpand_Click(object sender,
EventArgs e)
        {
            if (panelExpand.Visible)
                panelExpand.Visible = false;
            else
                panelExpand.Visible = true;
        }


    }
}
```

5.3 Forget Password Form

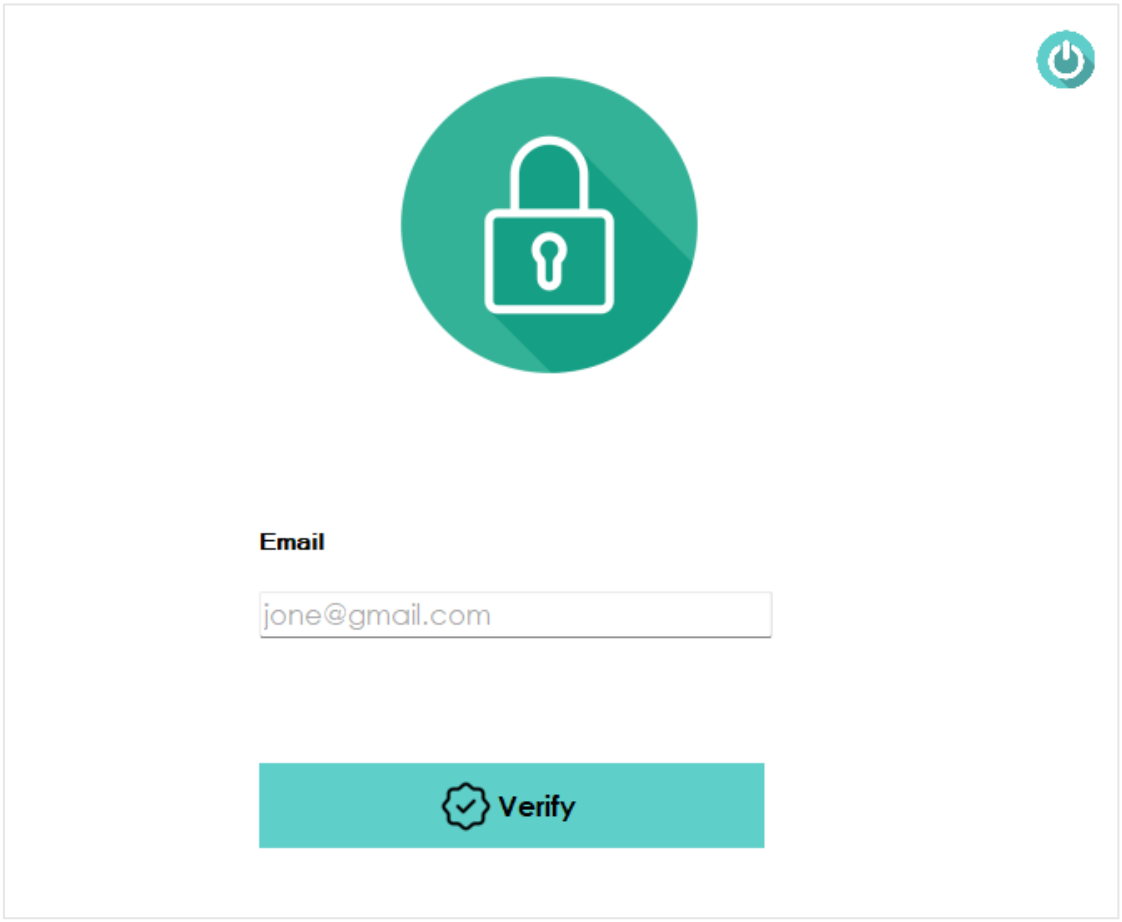


Figure 8Forget Password Form

The **Forget Password Form** allows users to recover their credentials by verifying their registered email.

- **Features**:

- Email validation with error icon.
- Database query to retrieve username and password.
- User-friendly error messages if email not found.

```
using MySql.Data.MySqlClient;
using System;
using System.Net.Mail;
using System.Windows.Forms;

namespace Student_Managemant.PLA.Forms
{
    public partial class FormForgetPassword : Form
    {
```

```
        private string connectionString = "Server=localhost;
Database=attendens_managment_system; Uid=root; Pwd=;";

        public FormForgetPassword()
        {
            InitializeComponent();
        }

        private bool IsValidEmail(string email)
        {
            try
            {
                var addr = new MailAddress(email);
                return addr.Address == email;
            }
            catch
            {
                return false;
            }
        }

        private void buttonSearch_Click(object sender, EventArgs e)
        {
            string email = textBoxEmail.Text.Trim();

            if (string.IsNullOrEmpty(email) || !IsValidEmail(email))
            {
                MessageBox.Show("Please enter a valid email address.",
"Validation Error", MessageBoxButtons.OK, MessageBoxIcon.Warning);
                return;
            }

            string query = "SELECT User_Name, User_Pass FROM
User_Table WHERE User_Email = @Email";

            try
            {
                using (MySqlConnection connection = new
MySqlConnection(connectionString))
                {
                    connection.Open();
                    using (MySqlCommand command = new
MySqlCommand(query, connection))
                    {
                        command.Parameters.AddWithValue("@Email",
email);

                        using (MySqlDataReader reader =
command.ExecuteReader())
                        {
                            if (reader.Read())
```

```
                            {
                                string username =
reader["User_Name"].ToString();
                                string password =
reader["User_Pass"].ToString();

                                MessageBox.Show($"Username:
{username}\nPassword: {password}", "Account Details",
MessageBoxButtons.OK, MessageBoxIcon.Information);
                            }
                            else
                            {
                                MessageBox.Show("No account found with
this email address.", "Not Found", MessageBoxButtons.OK,
MessageBoxIcon.Error);
                            }
                        }
                    }
                }
            }
            catch (Exception ex)
            {
                MessageBox.Show("Database error: " + ex.Message,
"Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
            }
        }

        private void buttonClose_Click(object sender, EventArgs e)
        {
            this.Close();
        }
    }
```

5.4 User Registration Module

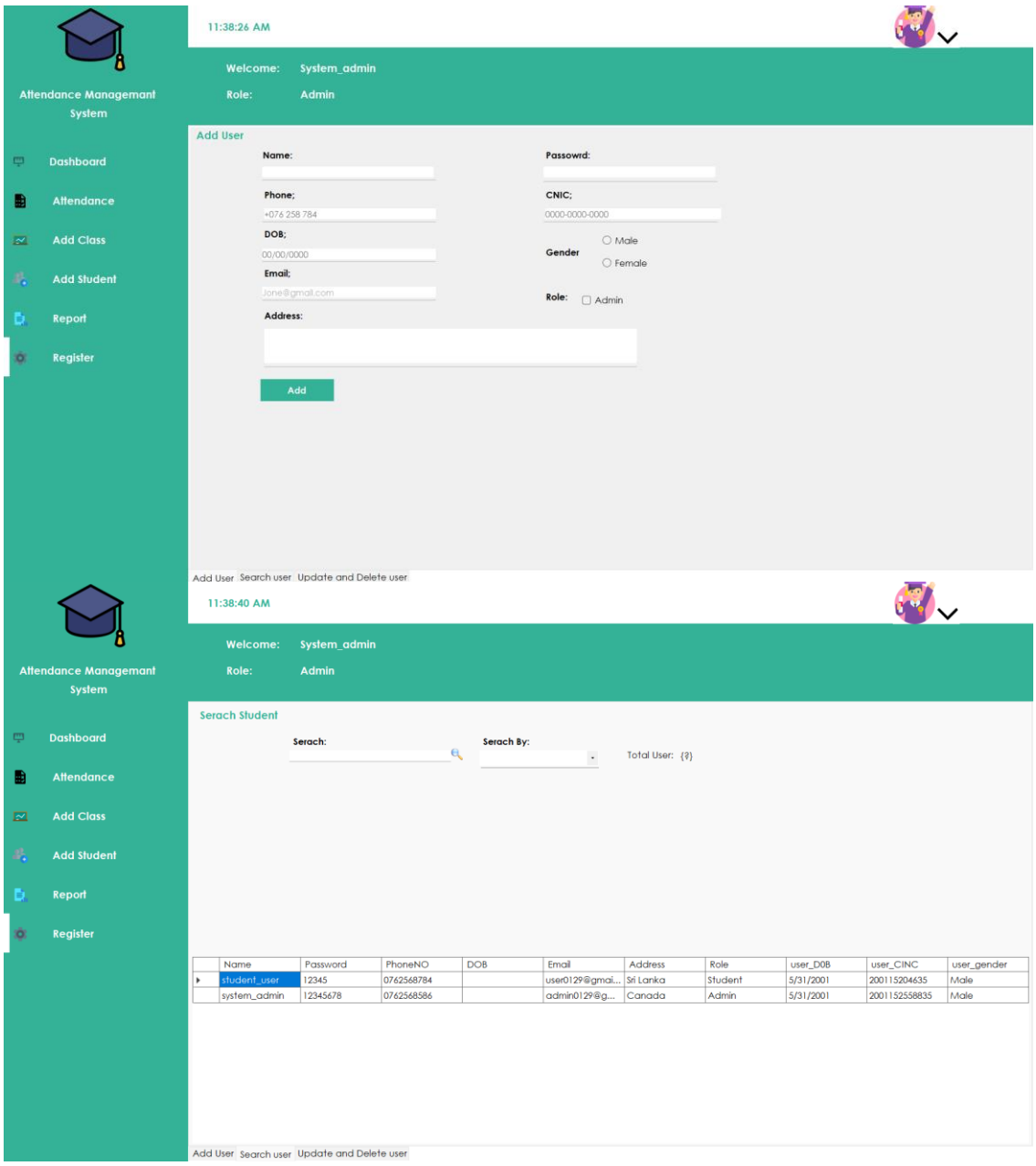


Figure 9User Registration Module

Implemented as `UserControl1Register`, this module allows administrators to manage system users.

- **Features**:

- Add new users with validation (Name, Email, CNIC, Phone, DOB, Gender, Role).
- Update existing user details.
- Delete users with confirmation and dependency checks.
- Search users by Name, Phone, or CNIC.
- Display total number of users.

```
using MySql.Data.MySqlClient;
using System;
using System.Data;
using System.Drawing;
using System.Net.Mail;
using System.Windows.Forms;

namespace Student_Managemant.PLA.userControl
{
    public partial class UserControl1Register : UserControl
    {
        private string connectionString = "Server=localhost;
Database=attendens_managment_system; Uid=root; Pwd=;";
        public string UID = "";
        private string gender = "", Role, ID = "";

        public UserControl1Register()
        {
            InitializeComponent();
        }

        private void panel2_Paint(object sender, PaintEventArgs e) { }

        private void textBox1_TextChanged(object sender, EventArgs e) { }

        private void tabPageUPUser_Click(object sender, EventArgs e) { }

        private void HideEroorPic()
        {
            pictureBoxErrorPho.Visible = false;
            pictureBoxErrorCINC.Visible = false;
            pictureBoxErrorEmail.Visible = false;
            pictureBoxErrorDOB.Visible = false;
        }
```

```
private void HideEroorPic1()
{
    pictureBoxErrorPho1.Visible = false;
    pictureBoxErrorEmail1.Visible = false;
    pictureBoxErrorCNIC1.Visible = false;
    pictureBoxErrorDOB1.Visible = false;
}

public void ClearTextBox()
{
    textBoxName.Clear();
    textBoxPassword.Clear();
    textBoxEmail.Clear();
    maskedTextBoxPho.Text = "000 00 00 000";
    maskedTextBoxPho.ForeColor = Color.DarkGray;
    maskedTextBoxCNIC.Text = "0000-0000-0000";
    maskedTextBoxDOB.Text = "00/00/0000";
    maskedTextBoxDOB.ForeColor = Color.DarkGray;
    radioButtonMale.Checked = false;
    radioButtonFemale.Checked = false;
    textBoxEmail.Text = "Gayan@gmail.com";
    textBoxEmail.ForeColor = Color.DarkGray;
    checkBoxRole.Checked = false;
    textBoxAdd.Clear();
    tabControlRegister.SelectedTab = tabPageAddUser;
}

public void ClearTextBox1()
{
    textBoxName.Clear();
    textBoxEmail1.Clear();
    maskedTextBoxPho1.Text = "000 00 00 000";
    maskedTextBoxPho1.ForeColor = Color.DarkGray;
    maskedTextBoxCNIC1.Text = "0000-0000-0000";
    maskedTextBoxDOB1.Text = "00/00/0000";
    maskedTextBoxDOB1.ForeColor = Color.DarkGray;
    radioButtonMale1.Checked = false;
    radioButtonFemale1.Checked = false;
    textBoxEmail1.Text = "Gayan@gmail.com";
    textBoxEmail1.ForeColor = Color.DarkGray;
    checkBoxRole1.Checked = false;
    textBoxAdd1.Clear();
    ID = "";
}

private void Mask(MaskedTextBox txtBox)
{
    BeginInvoke((MethodInvoker)delegate ()
    {
        txtBox.Select(0, 0);
    });
```

```
}

private bool IsValidEmail(string email)
{
    try
    {
        var addr = new MailAddress(email);
        return addr.Address == email;
    }
    catch
    {
        return false;
    }
}

private bool IsValidCNIC(string cnic)
{
    if (cnic.Length != 15) return false;
    for (int i = 0; i < cnic.Length; i++)
    {
        if (i == 4 || i == 9)
        {
            if (cnic[i] != '-') return false;
        }
        else
        {
            if (!char.IsDigit(cnic[i])) return false;
        }
    }
    return true;
}

private bool IsValidaDate(string date)
{
    DateTime d;
    bool ChValidity;
    try
    {
        ChValidity = DateTime.TryParse(date, out d);
        return ChValidity;
    }
    catch
    {
        return false;
    }
}

private void pictureBoxSUser_MouseHover(object sender, EventArgs e)
{
    toolTip1.SetToolTip(textBoxUser, "Serach");
}
```

```
        private void textBoxEmail_Enter(object sender, EventArgs e)
        {
            if (textBoxEmail.Text == "Gayan@gmail.com")
            {
                if (textBoxEmail.Text == "")
                {
                    textBoxEmail.Text = "";
                    textBoxEmail.ForeColor = Color.Black;
                }
            }
            if (!IsValidEmail(textBoxEmail.Text) || textBoxEmail.Text ==
"Gayan@gmail.com")
                pictureBoxErrorEmail.Visible = true;
            else
                pictureBoxErrorEmail.Visible = false;
        }

        private void textBoxEmail_Leave(object sender, EventArgs e)
        {
            textBoxEmail1.Text = "Gayan@gmail.com";
            textBoxEmail1.ForeColor = Color.DarkGray;
        }

        private void maskedTextBoxPho_Enter(object sender, EventArgs e)
        {
            if (maskedTextBoxPho.Text == "000 00 00 000")
            {
                maskedTextBoxPho.Text = "";
                maskedTextBoxPho.ForeColor = Color.Black;
            }
            if (!maskedTextBoxPho.MaskCompleted)
            {
                pictureBoxErrorPho.ForeColor = Color.Black;
                Mask(maskedTextBoxPho);
            }
            else
            {
                pictureBoxErrorPho.Visible = false;
            }
        }

        private void maskedTextBoxPho_Leave(object sender, EventArgs e)
        {
            if (!maskedTextBoxPho.MaskCompleted || maskedTextBoxPho.Text ==
"+000 00 00 000")
            {
                maskedTextBoxPho.Text = "000 00 00 000";
                maskedTextBoxPho.ForeColor = Color.DarkGray;
                pictureBoxErrorPho.Visible = true;
            }
```

```
    else
    {
        maskedTextBoxPho.ForeColor = Color.Black;
        pictureBoxErrorPho.Visible = false;
    }
}

private void maskedTextBoxCNIC_Enter(object sender, EventArgs e)
{
    if (maskedTextBoxCNIC.Text == "0000-0000-0000")
        maskedTextBoxCNIC.Text = "";

    if (!maskedTextBoxCNIC.MaskCompleted)
    {
        maskedTextBoxCNIC.ForeColor = Color.Black;
        Mask(maskedTextBoxCNIC);
    }
}

private void maskedTextBoxCNIC_Leave(object sender, EventArgs e)
{
    if (!maskedTextBoxCNIC.MaskCompleted)
    {
        maskedTextBoxCNIC.Text = "0000-0000-0000";
        maskedTextBoxCNIC.ForeColor = Color.DarkGray;
        pictureBoxErrorCINC.Visible = true;
        return;
    }

    if (maskedTextBoxCNIC.Text == "0000-0000-0000")
    {
        maskedTextBoxCNIC.ForeColor = Color.DarkGray;
        pictureBoxErrorCINC.Visible = true;
    }
    else
    {
        maskedTextBoxCNIC.ForeColor = Color.Black;
        pictureBoxErrorCINC.Visible = false;
    }
}

private void maskedTextBoxDOB_Enter(object sender, EventArgs e)
{
    if (maskedTextBoxDOB.Text == "00/00/0000")
        maskedTextBoxDOB.Text = "";
    if (!maskedTextBoxDOB.MaskCompleted)
    {
        maskedTextBoxDOB.ForeColor = Color.Black;
        Mask(maskedTextBoxDOB);
    }
}
```

```
        private void maskedTextBoxDOB_Leave(object sender, EventArgs e)
        {
            if (maskedTextBoxDOB.Text.Trim() == "/   /")
            {
                maskedTextBoxDOB.Text = "00/00/0000";
                maskedTextBoxDOB.ForeColor = Color.DarkGray;
            }
            if (!maskedTextBoxDOB.MaskCompleted)
            {
                maskedTextBoxDOB.ForeColor = Color.DarkGray;
            }
            if (!IsValidaDate(maskedTextBoxDOB.Text) ||
maskedTextBoxDOB.Text == "00/00/000" ||
DateTime.Parse(maskedTextBoxDOB.Text) > DateTime.Now)
                pictureBoxErrorDOB.Visible = true;
            else
                pictureBoxErrorDOB.Visible = false;
        }

        private void maskedTextBoxDOB_MouseHover(object sender, EventArgs e)
        {
            toolTip1.SetToolTip(maskedTextBoxDOB, "DD/MM/YYYY");
        }

        private void pictureBoxErrorEmail_MouseHover(object sender,
EventArgs e)
        {
            toolTip1.SetToolTip(pictureBoxErrorEmail, "Invalid Email
Format");
        }

        private void pictureBoxErrorDOB_MouseHover(object sender, EventArgs
e)
        {
            toolTip1.SetToolTip(pictureBoxErrorDOB, "Invalid Date Format");
        }

        private void pictureBoxErrorPho_MouseHover(object sender, EventArgs
e)
        {
            toolTip1.SetToolTip(pictureBoxErrorPho, "Invalid Phone Number");
        }

        private void pictureBoxErrorCINC_MouseHover(object sender, EventArgs
e)
        {
            toolTip1.SetToolTip(pictureBoxErrorCINC, "Invalid CNIC Format");
        }

        private void maskedTextBoxPho1_Enter(object sender, EventArgs e)
```

```
        {
            if (maskedTextBoxPho1.Text == "000 00 00 000")
            {
                maskedTextBoxPho1.Text = "";
                maskedTextBoxPho1.ForeColor = Color.Black;
            }
            if (!maskedTextBoxPho1.MaskCompleted)
            {
                pictureBoxErrorPho1.ForeColor = Color.Black;
                Mask(maskedTextBoxPho1);
            }
            else
            {
                pictureBoxErrorPho1.Visible = false;
            }
        }

        private void maskedTextBoxPho1_Leave(object sender, EventArgs e)
        {
            if (!maskedTextBoxPho1.MaskCompleted || maskedTextBoxPho1.Text
== "+000 00 00 000")
            {
                maskedTextBoxPho1.Text = "000 00 00 000";
                maskedTextBoxPho1.ForeColor = Color.DarkGray;
                pictureBoxErrorPho1.Visible = true;
            }
            else
            {
                maskedTextBoxPho1.ForeColor = Color.Black;
                pictureBoxErrorPho1.Visible = false;
            }
        }

        private void pictureBoxErrorPho1_MouseHover(object sender, EventArgs
e)
        {
            toolTip1.SetToolTip(pictureBoxErrorPho1, "Invalid Phone
Number");
        }

        private void maskedTextBoxCNIC1_Enter(object sender, EventArgs e)
        {
            if (maskedTextBoxCNIC1.Text == "0000-0000-0000")
                maskedTextBoxCNIC1.Text = "";

            if (!maskedTextBoxCNIC1.MaskCompleted)
            {
                maskedTextBoxCNIC1.ForeColor = Color.Black;
                Mask(maskedTextBoxCNIC1);
            }
        }
```

```
        private void maskedTextBoxCNIC1_Leave(object sender, EventArgs e)
        {
            if (!maskedTextBoxCNIC1.MaskCompleted)
            {
                maskedTextBoxCNIC1.Text = "0000-0000-0000";
                maskedTextBoxCNIC1.ForeColor = Color.DarkGray;
            }

            if (maskedTextBoxCNIC1.Text == "0000-0000-0000")
            {
                maskedTextBoxCNIC1.ForeColor = Color.DarkGray;
                pictureBoxErrorCNIC1.Visible = true;
            }
            else
            {
                pictureBoxErrorCNIC1.ForeColor = Color.Black;
                pictureBoxErrorCNIC1.Visible = false;
            }
        }

        private void pictureBoxErrorCNIC1_MouseHover(object sender,
EventArgs e)
        {
            toolTip1.SetToolTip(pictureBoxErrorCNIC1, "Invalid CNIC
Format");
        }

        private void maskedTextBoxDOB1_Enter(object sender, EventArgs e)
        {
            if (maskedTextBoxDOB1.Text == "00/00/0000")
                maskedTextBoxDOB1.Text = "";
            if (!maskedTextBoxDOB1.MaskCompleted)
            {
                maskedTextBoxDOB1.ForeColor = Color.Black;
                Mask(maskedTextBoxDOB1);
            }
        }

        private void maskedTextBoxDOB1_Leave(object sender, EventArgs e)
        {
            if (maskedTextBoxDOB1.Text.Trim() == "/   /")
            {
                maskedTextBoxDOB1.Text = "00/00/0000";
                maskedTextBoxDOB1.ForeColor = Color.DarkGray;
            }
            if (!maskedTextBoxDOB1.MaskCompleted)
            {
                maskedTextBoxDOB1.ForeColor = Color.DarkGray;
            }
```

```
            if (!IsValidaDate(maskedTextBoxDOB1.Text) ||
maskedTextBoxDOB1.Text == "00/00/000" ||
DateTime.Parse(maskedTextBoxDOB1.Text) > DateTime.Now)
                pictureBoxErrorDOB1.Visible = true;
            else
                pictureBoxErrorDOB1.Visible = false;
        }

        private void pictureBoxErrorDOB1_MouseHover(object sender, EventArgs
e)
        {
            toolTip1.SetToolTip(pictureBoxErrorDOB1, "Invalid Date Format");
        }

        private void textBoxEmail1_Enter(object sender, EventArgs e)
        {
            if (textBoxEmail1.Text == "Gayan@gmail.com")
            {
                if (textBoxEmail1.Text == "")
                {
                    textBoxEmail1.Text = "";
                    textBoxEmail1.ForeColor = Color.Black;
                }
            }
            if (!IsValidEmail(textBoxEmail1.Text) || textBoxEmail1.Text ==
"Gayan@gmail.com")
                pictureBoxErrorEmail1.Visible = true;
            else
                pictureBoxErrorEmail1.Visible = false;
        }

        private void textBoxEmail1_Leave(object sender, EventArgs e)
        {
            textBoxEmail1.Text = "Gayan@gmail.com";
            textBoxEmail1.ForeColor = Color.DarkGray;
        }

        private void pictureBoxErrorEmail1_MouseHover(object sender,
EventArgs e)
        {
            toolTip1.SetToolTip(pictureBoxErrorEmail1, "Invalid Email
Format");
        }

        private void textBoxUser_TextChanged(object sender, EventArgs e)
        {
            string searchTerm = textBoxUser.Text.Trim();
            if (string.IsNullOrEmpty(searchTerm))
            {
                return;
            }
```

```
            string searchColumn = "";
            string searchOperator = "LIKE";
            bool useWildcards = true;

            if (comboBoxSearchBy.SelectedIndex == 0)
            {
                searchColumn = "user_name";
            }
            else if (comboBoxSearchBy.SelectedIndex == 1)
            {
                searchColumn = "user_pho";
            }
            else if (comboBoxSearchBy.SelectedIndex == 2)
            {
                searchColumn = "user_CNIC";
                searchOperator = "=";
                useWildcards = false;
            }
            else
            {
                MessageBox.Show("Please select a search category first.",
"Search Error", MessageBoxButtons.OK, MessageBoxIcon.Warning);
                return;
            }

            string query = $"SELECT * FROM user_table WHERE {searchColumn}
{searchOperator} @SearchTerm";

            try
            {
                using (MySqlConnection connection = new
MySqlConnection(connectionString))
                {
                    connection.Open();

                    using (MySqlCommand command = new MySqlCommand(query,
connection))
                    {
                        string finalSearchTerm = useWildcards ?
$"%{searchTerm}%" : searchTerm;
                        command.Parameters.AddWithValue("@SearchTerm",
finalSearchTerm);

                        using (MySqlDataAdapter adapter = new
MySqlDataAdapter(command))
                        {
                            DataTable dataTable = new DataTable();
                            adapter.Fill(dataTable);
                            dataGridViewStudent.DataSource = dataTable;
                        }
```

```
                    }
                }

                string query1 = "SELECT COUNT(*) FROM user_table";

                try
                {
                    using (MySqlConnection connection = new
MySqlConnection(connectionString))
                    {
                        connection.Open();
                        using (MySqlCommand command = new
MySqlCommand(query1, connection))
                        {
                            object result = command.ExecuteScalar();

                            if (result != null)
                            {
                                labelTotalUser.Text = result.ToString();
                            }
                            else
                            {
                                labelTotalUser.Text = "0";
                            }
                        }
                    }
                }
                catch (MySqlException ex)
                {
                    MessageBox.Show("Database Error while loading total
users: " + ex.Message, "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
                    labelTotalUser.Text = "Error";
                }
                catch (Exception ex)
                {
                    MessageBox.Show("An unexpected error occurred while
loading total users: " + ex.Message, "Error", MessageBoxButtons.OK,
MessageBoxIcon.Error);
                    labelTotalUser.Text = "Error";
                }
            }
            catch (MySqlException ex)
            {
                MessageBox.Show("Database Error during search: " +
ex.Message, "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
            }
            catch (Exception ex)
            {
                MessageBox.Show("An unexpected error occurred during search:
" + ex.Message, "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
            }
```

```
        }

        private void tabPageSearchUser_Enter(object sender, EventArgs e)
        {
            textBoxUser.Clear();
            comboBoxSearchBy.SelectedIndex = -1;

            string query = "SELECT * FROM User_Table";

            try
            {
                using (MySqlConnection connection = new
MySqlConnection(connectionString))
                {
                    connection.Open();

                    using (MySqlCommand command = new MySqlCommand(query,
connection))
                    {
                        MySqlDataAdapter adapter = new
MySqlDataAdapter(command);
                        DataTable dataTable = new DataTable();
                        adapter.Fill(dataTable);
                        dataGridViewStudent.DataSource = dataTable;
                    }
                }

                if (dataGridViewStudent.Columns.Count > 0)
                {
                    dataGridViewStudent.Columns[0].Visible = false;
                }

                dataGridViewStudent.Text =
dataGridViewStudent.Rows.Count.ToString();
            }
            catch (MySqlException ex)
            {
                MessageBox.Show("Database Error: Failed to load user list. "
+ ex.Message, "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
            }
            catch (Exception ex)
            {
                MessageBox.Show("An unexpected error occurred: " +
ex.Message, "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
            }
        }

        private void tabPageAddUser_Leave(object sender, EventArgs e)
        {
            HideEroorPic();
            HideEroorPic1();
```

```
            ClearTextBox();
        }

        private void buttonAdd_Click(object sender, EventArgs e)
        {
            HideEroorPic();
            bool validationPassed = true;

            string name = textBoxName.Text.Trim();
            string password = textBoxPassword.Text;
            string address = textBoxAdd.Text.Trim();
            string email = textBoxEmail.Text.Trim();
            string phone = maskedTextBoxPho.Text.Replace(" ", "").Trim();
            string cnic = maskedTextBoxCNIC.Text.Trim();
            string dobText = maskedTextBoxDOB.Text.Trim();

            string role = checkBoxRole.Checked ? "Admin" : "User";

            string gender = "";
            if (radioButtonMale.Checked)
            {
                gender = "Male";
            }
            else if (radioButtonFemale.Checked)
            {
                gender = "Female";
            }

            if (string.IsNullOrWhiteSpace(name))
            {
                MessageBox.Show("Name is required.", "Validation Error",
MessageBoxButtons.OK, MessageBoxIcon.Warning);
                textBoxName.Focus();
                validationPassed = false;
            }
            else if (string.IsNullOrWhiteSpace(password) || password.Length
< 5)
            {
                MessageBox.Show("Password is required and must be at least 6
characters.", "Validation Error", MessageBoxButtons.OK,
MessageBoxIcon.Warning);
                textBoxPassword.Focus();
                validationPassed = false;
            }
            else if (string.IsNullOrWhiteSpace(gender))
            {
                MessageBox.Show("Gender selection is required.", "Validation
Error", MessageBoxButtons.OK, MessageBoxIcon.Warning);
                validationPassed = false;
            }
```

```
            if (validationPassed && (!IsValidEmail(email)))
            {
                pictureBoxErrorEmail.Visible = true;
                MessageBox.Show("Invalid Email format.", "Validation Error",
MessageBoxButtons.OK, MessageBoxIcon.Warning);
                textBoxEmail.Focus();
                validationPassed = false;
            }

            if (validationPassed && !maskedTextBoxCNIC.MaskCompleted)
            {
                pictureBoxErrorCINC.Visible = true;
                MessageBox.Show("CNIC field is incomplete.", "Validation
Error", MessageBoxButtons.OK, MessageBoxIcon.Warning);
                maskedTextBoxCNIC.Focus();
                validationPassed = false;
            }

            if (validationPassed && !maskedTextBoxPho.MaskCompleted)
            {
                pictureBoxErrorPho.Visible = true;
                MessageBox.Show("Phone field is incomplete.", "Validation
Error", MessageBoxButtons.OK, MessageBoxIcon.Warning);
                maskedTextBoxPho.Focus();
                validationPassed = false;
            }

            if (validationPassed)
            {
                DateTime dob;
                if (!IsValidaDate(dobText) || !DateTime.TryParse(dobText,
out dob) || dob > DateTime.Now)
                {
                    pictureBoxErrorDOB.Visible = true;
                    MessageBox.Show("Invalid Date of Birth format or date is
in the future.", "Validation Error", MessageBoxButtons.OK,
MessageBoxIcon.Warning);
                    maskedTextBoxDOB.Focus();
                    validationPassed = false;
                }
            }

            if (validationPassed)
            {
                string query = "INSERT INTO User_Table (User_Name,
User_Pass, User_Email, User_Pho, User_CNIC, User_DOB, User_Gender,
User_Role, User_Add) " +
                               "VALUES (@Name, @Password, @Email, @Phone,
@CNIC, @DOB, @Gender, @Role, @Address)";

                try
```

```
                {
                    using (MySqlConnection connection = new
MySqlConnection(connectionString))
                    {
                        connection.Open();
                        using (MySqlCommand command = new
MySqlCommand(query, connection))
                        {
                            command.Parameters.AddWithValue("@Name", name);
                            command.Parameters.AddWithValue("@Password",
password);
                            command.Parameters.AddWithValue("@Email",
email);
                            command.Parameters.AddWithValue("@Phone",
phone);
                            command.Parameters.AddWithValue("@CNIC", cnic);
                            command.Parameters.AddWithValue("@DOB",
dobText);
                            command.Parameters.AddWithValue("@Gender",
gender);
                            command.Parameters.AddWithValue("@Role", role);
                            command.Parameters.AddWithValue("@Address",
address);

                            int rowsAffected = command.ExecuteNonQuery();

                            if (rowsAffected > 0)
                            {
                                MessageBox.Show("User registered
successfully!", "Success", MessageBoxButtons.OK,
MessageBoxIcon.Information);
                                ClearTextBox();
                            }
                            else
                            {
                                MessageBox.Show("User registration failed.
No rows were inserted.", "Error", MessageBoxButtons.OK,
MessageBoxIcon.Error);
                            }
                        }
                    }
                }
                catch (MySqlException ex)
                {
                    if (ex.Number == 1062)
                    {
                        MessageBox.Show("Registration failed: CNIC or Email
already exists.", "Database Error", MessageBoxButtons.OK,
MessageBoxIcon.Error);
                    }
                    else
```

```
                    {
                        MessageBox.Show("Database Error during registration:
" + ex.Message, "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
                    }
                }
                catch (Exception ex)
                {
                    MessageBox.Show("An unexpected error occurred: " +
ex.Message, "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
                }
            }
        }

        private void tabPageAddUser_Enter(object sender, EventArgs e)
        {
            ClearTextBox1();
        }

        private void tabPageUPUser_Leave(object sender, EventArgs e)
        {
            HideEroorPic1();
            ClearTextBox1();
        }

        private void dataGridViewStudent_CellClick(object sender,
DataGridViewCellEventArgs e)
        {
            if (e.RowIndex >= 0)
            {
                DataGridViewRow row = dataGridViewStudent.Rows[e.RowIndex];
                ID = row.Cells["user_ID"].Value.ToString();
                textBox4.Text = row.Cells["user_Name"].Value.ToString();
                textBoxPass1.Text = row.Cells["user_Pass"].Value.ToString();
                textBoxEmail1.Text =
row.Cells["user_Email"].Value.ToString();
                textBoxEmail1.ForeColor = Color.Black;
                maskedTextBoxPho1.Text =
row.Cells["user_Pho"].Value.ToString();
                maskedTextBoxPho1.ForeColor = Color.Black;
                maskedTextBoxCNIC1.Text =
row.Cells["user_CNIC"].Value.ToString();
                maskedTextBoxCNIC1.ForeColor = Color.Black;
                maskedTextBoxDOB1.Text =
row.Cells["user_DOB"].Value.ToString();
                maskedTextBoxDOB1.ForeColor = Color.Black;
                gender = row.Cells["gender"].Value.ToString();
                if (gender == "Male")
                    radioButtonMale1.Checked = true;
                else
                    radioButtonFemale1.Checked = true;
                Role = row.Cells["user_Role"].Value.ToString();
```

```
                if (Role == "Admin")
                    checkBoxRole1.Checked = true;
                textBoxAdd1.Text = row.Cells["user_Add"].Value.ToString();
            }
        }

        private void buttonUpdate_Click(object sender, EventArgs e)
        {
            if (string.IsNullOrEmpty(ID))
            {
                MessageBox.Show("Please select a user from the Search tab
before attempting to update.", "Selection Required", MessageBoxButtons.OK,
MessageBoxIcon.Warning);
                return;
            }

            HideEroorPic1();
            bool validationPassed = true;

            string name = textBox4.Text.Trim();
            string email = textBoxEmail1.Text.Trim();
            string phone = maskedTextBoxPho1.Text.Replace(" ", "").Trim();
            string cnic = maskedTextBoxCNIC1.Text.Trim();
            string dobText = maskedTextBoxDOB1.Text.Trim();
            string address = textBoxAdd1.Text.Trim();

            string role = checkBoxRole1.Checked ? "Admin" : "User";

            string gender = "";
            if (radioButtonMale1.Checked)
            {
                gender = "Male";
            }
            else if (radioButtonFemale1.Checked)
            {
                gender = "Female";
            }

            if (string.IsNullOrWhiteSpace(name))
            {
                MessageBox.Show("Name is required.", "Validation Error",
MessageBoxButtons.OK, MessageBoxIcon.Warning);
                textBox4.Focus();
                validationPassed = false;
            }
            else if (string.IsNullOrWhiteSpace(gender))
            {
                MessageBox.Show("Gender selection is required.", "Validation
Error", MessageBoxButtons.OK, MessageBoxIcon.Warning);
                validationPassed = false;
            }
```

```
            if (validationPassed && (!IsValidEmail(email) || email ==
"Gayan@gmail.com"))
            {
                pictureBoxErrorEmail1.Visible = true;
                MessageBox.Show("Invalid Email format.", "Validation Error",
MessageBoxButtons.OK, MessageBoxIcon.Warning);
                textBoxEmail1.Focus();
                validationPassed = false;
            }

            if (validationPassed && !maskedTextBoxCNIC1.MaskCompleted)
            {
                pictureBoxErrorCNIC1.Visible = true;
                MessageBox.Show("CNIC field is incomplete.", "Validation
Error", MessageBoxButtons.OK, MessageBoxIcon.Warning);
                maskedTextBoxCNIC1.Focus();
                validationPassed = false;
            }

            if (validationPassed)
            {
                DateTime dob;
                if (!IsValidaDate(dobText) || !DateTime.TryParse(dobText,
out dob) || dob > DateTime.Now)
                {
                    pictureBoxErrorDOB1.Visible = true;
                    MessageBox.Show("Invalid Date of Birth format or date is
in the future.", "Validation Error", MessageBoxButtons.OK,
MessageBoxIcon.Warning);
                    maskedTextBoxDOB1.Focus();
                    validationPassed = false;
                }
            }

            if (validationPassed)
            {
                string query = "UPDATE User_Table SET " +
                               "User_Name = @Name, " +
                               "User_Email = @Email, " +
                               "User_Pho = @Phone, " +
                               "User_CNIC = @CNIC, " +
                               "User_DOB = @DOB, " +
                               "User_Gender = @Gender, " +
                               "User_Role = @Role, " +
                               "User_Add = @Address " +
                               "WHERE User_ID = @ID";

                try
                {
```

```
                    using (MySqlConnection connection = new
MySqlConnection(connectionString))
                    {
                        connection.Open();
                        using (MySqlCommand command = new
MySqlCommand(query, connection))
                        {
                            command.Parameters.AddWithValue("@Name", name);
                            command.Parameters.AddWithValue("@Email",
email);
                            command.Parameters.AddWithValue("@Phone",
phone);
                            command.Parameters.AddWithValue("@CNIC", cnic);
                            command.Parameters.AddWithValue("@DOB",
dobText);
                            command.Parameters.AddWithValue("@Gender",
gender);
                            command.Parameters.AddWithValue("@Role", role);
                            command.Parameters.AddWithValue("@Address",
address);
                            command.Parameters.AddWithValue("@ID", ID);

                            int rowsAffected = command.ExecuteNonQuery();

                            if (rowsAffected > 0)
                            {
                                MessageBox.Show("User details updated
successfully!", "Success", MessageBoxButtons.OK,
MessageBoxIcon.Information);
                                ClearTextBox1();
                            }
                            else
                            {
                                MessageBox.Show("Update failed. User ID may
not exist or data was unchanged.", "Error", MessageBoxButtons.OK,
MessageBoxIcon.Error);
                            }
                        }
                    }
                }
                catch (MySqlException ex)
                {
                    MessageBox.Show("Database Error during update: " +
ex.Message, "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
                }
                catch (Exception ex)
                {
                    MessageBox.Show("An unexpected error occurred: " +
ex.Message, "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
                }
            }
```

```
        }

        private void buttonDelete_Click(object sender, EventArgs e)
        {
            if (string.IsNullOrEmpty(ID))
            {
                MessageBox.Show("Please select a user from the Search tab
before attempting to delete.", "Selection Required", MessageBoxButtons.OK,
MessageBoxIcon.Warning);
                return;
            }

            DialogResult confirm = MessageBox.Show($"Are you sure you want
to permanently delete the user with ID **{ID}**?",
                                                   "Confirm Deletion",
                                                   MessageBoxButtons.YesNo,
                                                   MessageBoxIcon.Question);

            if (confirm == DialogResult.No)
            {
                return;
            }

            string query = "DELETE FROM User_Table WHERE user_ID = @ID";

            try
            {
                using (MySqlConnection connection = new
MySqlConnection(connectionString))
                {
                    connection.Open();
                    using (MySqlCommand command = new MySqlCommand(query,
connection))
                    {
                        command.Parameters.AddWithValue("@ID", ID);

                        int rowsAffected = command.ExecuteNonQuery();

                        if (rowsAffected > 0)
                        {
                            MessageBox.Show("User was successfully
deleted!", "Success", MessageBoxButtons.OK, MessageBoxIcon.Information);
                            ClearTextBox1();
                            ID = "";
                        }
                        else
                        {
                            MessageBox.Show("Deletion failed. User ID was
not found.", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
                        }
                    }
```

```
                }
            }
            catch (MySqlException ex)
            {
                if (ex.Number == 1451)
                {
                    MessageBox.Show("Cannot delete user: Dependent records
(e.g., attendance data) exist. Remove those first.", "Deletion Blocked",
MessageBoxButtons.OK, MessageBoxIcon.Error);
                }
                else
                {
                    MessageBox.Show("Database Error during deletion: " +
ex.Message, "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
                }
            }
            catch (Exception ex)
            {
                MessageBox.Show("An unexpected error occurred: " +
ex.Message, "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
            }
        }
    }
}
```

5.5 Attendance Module

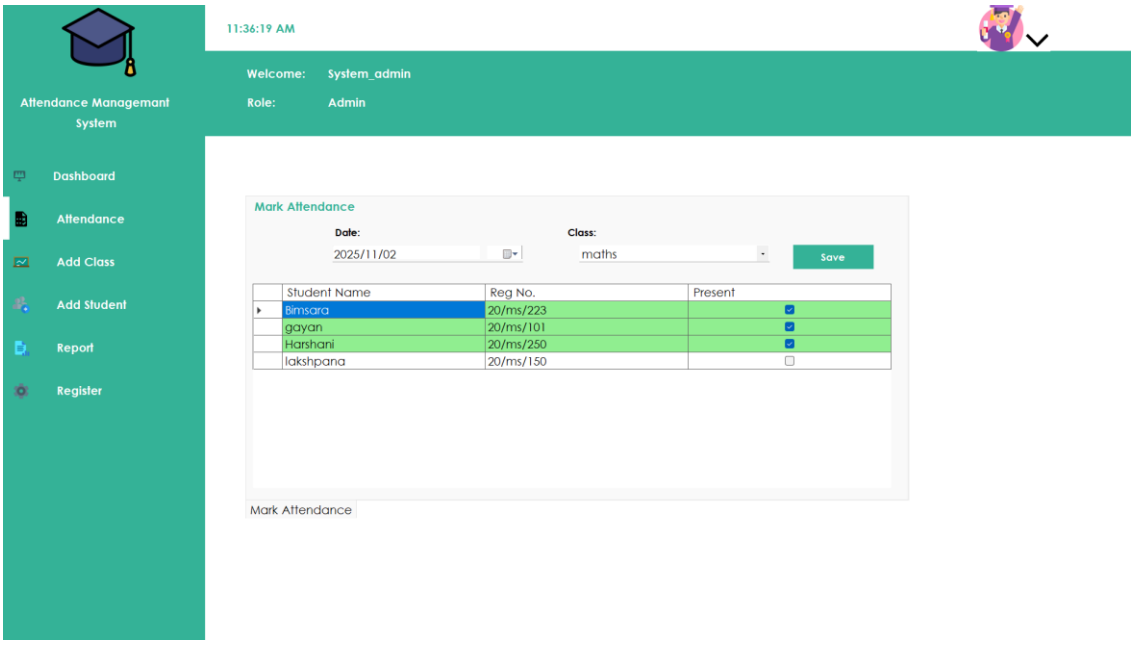


Figure 10 Figure 10Attendance Module

this module manages daily attendance.

- **Features**:

- Load classes dynamically from `Class_Table`.
- Display students of a selected class with checkboxes for Present/Absent.
- Color-coded rows (green for Present, white for Absent).
- Save attendance with transaction support (Insert/Update).
- Reload updated attendance after saving.

```
using System;
using System.Data;
using System.Drawing;
using System.Windows.Forms;
using MySql.Data.MySqlClient;

namespace Student_Managemant.PLA.userControl
{
    public partial class UserControlAttendance : UserControl
    {
        private const string ConnectionString =
"server=localhost;port=3306;database=attendens_managment_syst
em;uid=root;password=;";

        public UserControlAttendance()
        {
            InitializeComponent();

            comboBoxClass.DisplayMember = "Class_Name";
            comboBoxClass.ValueMember = "Class_ID";

            ConfigureDataGridViewColumns();
        }

        private void ConfigureDataGridViewColumns()
        {

dataGridViewMarkAttendance.Columns["Column1"].DataPropertyNam
e = "Student_ID";

dataGridViewMarkAttendance.Columns["Column1"].Visible =
false;
```

```
dataGridViewMarkAttendance.Columns["Column2"].DataPropertyNam
e = "Student_Name";

dataGridViewMarkAttendance.Columns["Column2"].HeaderText =
"Student Name";


dataGridViewMarkAttendance.Columns["Column3"].DataPropertyNam
e = "Student_RegNo";

dataGridViewMarkAttendance.Columns["Column3"].HeaderText =
"Reg No.";


dataGridViewMarkAttendance.Columns["Column4"].DataPropertyNam
e = "Is_Present_Flag";

dataGridViewMarkAttendance.Columns["Column4"].HeaderText =
"Present";


dataGridViewMarkAttendance.Columns["Column5"].DataPropertyNam
e = "Attendance_Status";

dataGridViewMarkAttendance.Columns["Column5"].Visible =
false;
        }

        private void UserControlAttendance_Load(object
sender, EventArgs e)
        {
            LoadClasses();
        }

        private void dateTimePickerDate_ValueChanged(object
sender, EventArgs e)
        {
            if (comboBoxClass.SelectedValue != null &&
comboBoxClass.SelectedValue != DBNull.Value)
            {
```

```
                if
(int.TryParse(comboBoxClass.SelectedValue.ToString(), out int
classId))
                {
                    LoadStudentsForClass(classId,
dateTimePickerDate.Value.ToString("yyyy-MM-dd"));
                }
            }
        }

        private void
comboBoxClass_SelectedIndexChanged(object sender, EventArgs
e)
        {
            if (comboBoxClass.SelectedValue != null &&
comboBoxClass.SelectedValue != DBNull.Value)
            {
                if
(int.TryParse(comboBoxClass.SelectedValue.ToString(), out int
classId))
                {
                    LoadStudentsForClass(classId,
dateTimePickerDate.Value.ToString("yyyy-MM-dd"));
                }
            }
            else
            {
                dataGridViewMarkAttendance.DataSource = null;
            }
        }

        private void
dataGridViewMarkAttendance_CurrentCellDirtyStateChanged(objec
t sender, EventArgs e)
        {
            if
(dataGridViewMarkAttendance.IsCurrentCellDirty)
            {

dataGridViewMarkAttendance.CommitEdit(DataGridViewDataErrorCo
ntexts.Commit);
            }
        }
```

```
        private void
dataGridViewMarkAttendance_CellFormatting(object sender,
DataGridViewCellFormattingEventArgs e)
        {
            if (e.RowIndex >= 0 &&
dataGridViewMarkAttendance.Columns["Column4"] != null &&
e.ColumnIndex ==
dataGridViewMarkAttendance.Columns["Column4"].Index)
            {
                if (e.Value != null && e.Value is bool
isPresent)
                {
                    if (isPresent)
                    {

dataGridViewMarkAttendance.Rows[e.RowIndex].DefaultCellStyle.
BackColor = Color.LightGreen;
                    }
                    else
                    {

dataGridViewMarkAttendance.Rows[e.RowIndex].DefaultCellStyle.
BackColor = Color.White;
                    }
                }
            }
        }

        private void LoadClasses()
        {
            string query = "SELECT Class_ID, Class_Name FROM
class_table ORDER BY Class_Name";
            try
            {
                using (MySqlConnection connection = new
MySqlConnection(ConnectionString))
                {
                    connection.Open();
                    using (MySqlDataAdapter adapter = new
MySqlDataAdapter(query, connection))
                    {
                        DataTable dt = new DataTable();
```

```
                        adapter.Fill(dt);
                        comboBoxClass.DataSource = dt;
                    }
                }
            }
            catch (Exception ex)
            {
                MessageBox.Show("Error loading classes: " +
ex.Message, "Database Error", MessageBoxButtons.OK,
MessageBoxIcon.Error);
            }
        }

        private void LoadStudentsForClass(int classId, string
date)
        {
            string query = @"
                SELECT
                    s.Student_ID,
                    s.Student_Name,
                    s.Student_RegNo,
                    a.Attendance_Status
                FROM
                    student_table s
                LEFT JOIN
                    attendance_table a ON s.Student_ID =
a.Student_ID AND a.Attendance_Date = @date
                WHERE
                    s.Class_ID = @classId
                ORDER BY s.Student_Name";

            try
            {
                DataTable dt = new DataTable();
                using (MySqlConnection connection = new
MySqlConnection(ConnectionString))
                {
                    connection.Open();
                    using (MySqlCommand command = new
MySqlCommand(query, connection))
                    {

command.Parameters.AddWithValue("@classId", classId);
```

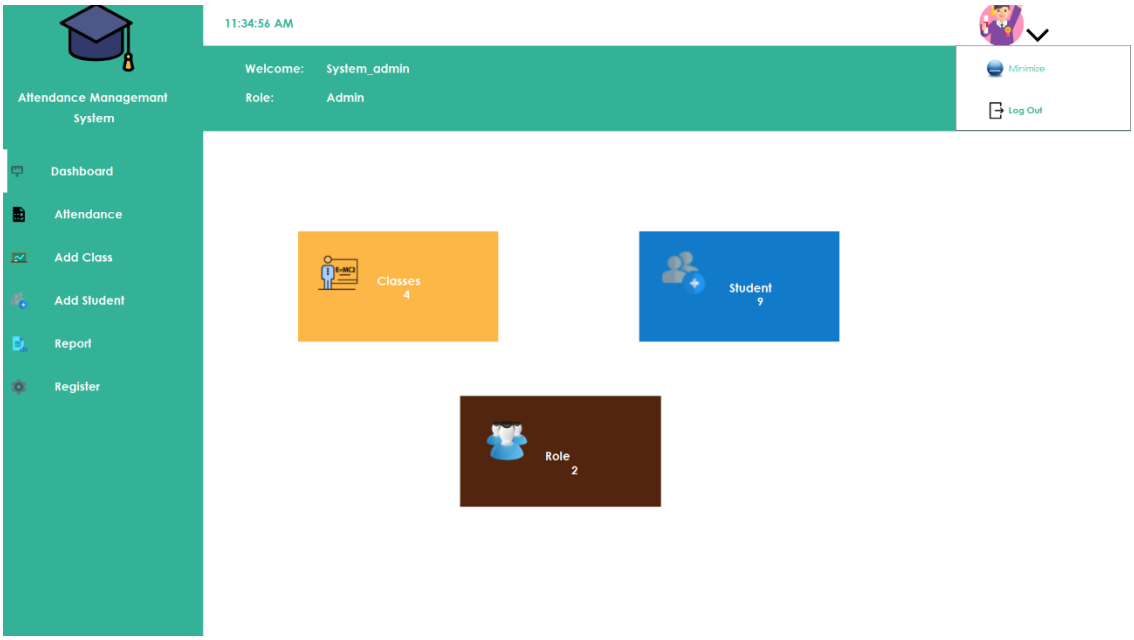
```
command.Parameters.AddWithValue("@date", date);

                        using (MySqlDataAdapter adapter = new
MySqlDataAdapter(command))
                        {
                            adapter.Fill(dt);
                        }
                    }
                }

                dt.Columns.Add("Is_Present_Flag",
typeof(bool));

                foreach (DataRow row in dt.Rows)
                {
                    string status = row["Attendance_Status"]
== DBNull.Value ? "Absent" :
row["Attendance_Status"].ToString();
                    row["Attendance_Status"] = status;
                    row["Is_Present_Flag"] =
status.Equals("Present", StringComparison.OrdinalIgnoreCase);
                }

                dataGridViewMarkAttendance.DataSource = dt;
                ConfigureDataGridViewColumns();
            }
            catch (Exception ex)
            {
                MessageBox.Show("Error loading student
attendance: " + ex.Message, "Database Error",
MessageBoxButtons.OK, MessageBoxIcon.Error);
                dataGridViewMarkAttendance.DataSource = null;
            }
        }

        private void buttonSave_Click(object sender,
EventArgs e)
        {
            if (comboBoxClass.SelectedValue == null ||
comboBoxClass.SelectedValue == DBNull.Value)
            {
```

```
                MessageBox.Show("Please select a class before
saving.", "Warning", MessageBoxButtons.OK,
MessageBoxIcon.Warning);
                return;
            }

            DataTable attendanceTable =
dataGridViewMarkAttendance.DataSource as DataTable;

            if (attendanceTable == null ||
attendanceTable.Rows.Count == 0)
            {
                MessageBox.Show("No students to save
attendance for.", "Warning", MessageBoxButtons.OK,
MessageBoxIcon.Warning);
                return;
            }

            string date =
dateTimePickerDate.Value.ToString("yyyy-MM-dd");
            int recordsSaved = 0;

            try
            {
                using (MySqlConnection connection = new
MySqlConnection(ConnectionString))
                {
                    connection.Open();

                    using (MySqlTransaction transaction =
connection.BeginTransaction())
                    {
                        foreach (DataRow row in
attendanceTable.Rows)
                        {
                            int studentId =
Convert.ToInt32(row["Student_ID"]);
                            bool isPresent =
Convert.ToBoolean(row["Is_Present_Flag"]);
                            string status = isPresent ?
"Present" : "Absent";
```

```
                            string checkQuery = "SELECT
COUNT(*) FROM attendance_table WHERE Student_ID = @studentId
AND Attendance_Date = @date";
                            MySqlCommand checkCommand = new
MySqlCommand(checkQuery, connection, transaction);

checkCommand.Parameters.AddWithValue("@studentId",
studentId);

checkCommand.Parameters.AddWithValue("@date", date);

                            long count =
Convert.ToInt64(checkCommand.ExecuteScalar());

                            string saveQuery;
                            if (count > 0)
                            {
                                saveQuery = "UPDATE
attendance_table SET Attendance_Status = @status WHERE
Student_ID = @studentId AND Attendance_Date = @date";
                            }
                            else
                            {
                                saveQuery = "INSERT INTO
attendance_table (Attendance_Date, Attendance_Status,
Student_ID) VALUES (@date, @status, @studentId)";
                            }

                            MySqlCommand saveCommand = new
MySqlCommand(saveQuery, connection, transaction);

saveCommand.Parameters.AddWithValue("@date", date);

saveCommand.Parameters.AddWithValue("@status", status);

saveCommand.Parameters.AddWithValue("@studentId", studentId);

                            recordsSaved +=
saveCommand.ExecuteNonQuery();
                        }

                        transaction.Commit();
```

```
                        MessageBox.Show($"{recordsSaved}
attendance records saved successfully for {date}.",
"Success", MessageBoxButtons.OK, MessageBoxIcon.Information);
                    }
                }

                if (comboBoxClass.SelectedValue is int
classId)
                {
                    LoadStudentsForClass(classId, date);
                }
            }
            catch (Exception ex)
            {
                MessageBox.Show("Error saving attendance: " +
ex.Message, "Database Error", MessageBoxButtons.OK,
MessageBoxIcon.Error);
            }
        }
    }
}
```

5.6 Dashboard Statistics Module



Implemented as `UserControlDashbord`, this module provides administrators with a quick overview.

- **Features**:

- Displays total number of Classes, Students, and Users.
- Automatically updates when the dashboard loads.

```
using MySql.Data.MySqlClient;
using System;
using System.Data;
using System.Windows.Forms;

namespace Student_Managemant.PLA.userControl
{
    public partial class UserControlDashbord : UserControl
    {
        private string connectionString = "Server=localhost;
Database=attendens_managment_system; Uid=root; Pwd=;";

        public UserControlDashbord()
        {
            InitializeComponent();
        }

        public void Count()
        {
```

```
            string query = "SELECT COUNT(*) FROM Class_Table";
            string query1 = "SELECT COUNT(*) FROM
Student_Table";
            string query3 = "SELECT COUNT(*) FROM User_Table";

            using (MySqlConnection connection = new
MySqlConnection(connectionString))
            {
                connection.Open();
                using (MySqlCommand command = new
MySqlCommand(query, connection))
                {
                    object result = command.ExecuteScalar();
                    labelTotalClasses.Text = result != null ?
result.ToString() : "0";
                }
                using (MySqlCommand comandS = new
MySqlCommand(query1, connection))
                {
                    object resultS = comandS.ExecuteScalar();
                    label2.Text = resultS != null ?
resultS.ToString() : "0";
                }
                using (MySqlCommand comandU = new
MySqlCommand(query3, connection))
                {
                    object resultU = comandU.ExecuteScalar();
                    labelTotalRole.Text = resultU != null ?
resultU.ToString() : "0";
                }
            }
        }

        private void label2_Click(object sender, EventArgs e) {
}

        private void label4_Click(object sender, EventArgs e) {
}

        private void UserControlDashbord_Load(object sender,
EventArgs e)
        {
            Count();
```

7 Bibliography

documentation, M. (2023). *MySQL*. Retrieved from MySQL: https://dev.mysql.com/doc/

Foradian Technologies. (2023). *School Management Software & School Management System – Fedena*. Retrieved from Fedena: https://fedena.com/

Google. (2023). *Classroom Management Tools & Resources – Google Classroom*. Retrieved from Google for Education: https://edu.google.com/intl/en/workspace-for-education/products/classroom/

Microsoft. (2023). *C# documentation*. Retrieved from Microsoft Learn: https://learn.microsoft.com/en-us/dotnet/csharp/

OS4ED (Open Solutions for Education, Inc.). (2023). - *openSIS – Cloud Based Student Information System School Management Software*. Retrieved from openSIS: https://www.opensis.com/

PowerSchool. (2023). *PowerSchool – Education Technology Platform*. Retrieved from PowerSchool: https://www.powerschool.com/