Student Attendance Management System

(Report)

Python3

11/11/2024

Krishna Paul (12033028)

Shariar Khan (12033913)

Table of Contents

Overview

The **Student Attendance Management System** is designed to handle the registration of students and the recording of their attendance in a classroom setting. This application is developed using Python, with **Tkinter** for the graphical user interface (GUI) and **Pandas** for data management. The primary objective of the system is to provide instructors with an easy-to-use tool for tracking attendance while ensuring the protection of student data through secure access control and role-based authentication.

The system's functionality includes:

- Student registration
- Attendance marking
- Data storage and retrieval
- Secure login for instructors

The system is built with maintainability in mind, following best practices in programming and ensuring the code is easy to update and test.

System Design and Planning (15%)

Conceptual Design

The system supports two types of users:

1. **Students**: Can register themselves in the system using their name.
2. **Instructors**: Can log in using a password, mark attendance for students, and view attendance reports.

The system is designed to be simple yet effective, allowing instructors to easily manage student attendance. The attendance data is securely stored and can be accessed or modified by authorized users only.

Data Model

The application uses **CSV** files to store student and attendance data:

- **students.csv**: Contains student IDs and names.
- **attendance.csv**: Tracks the attendance status (present/absent) for each student, along with the date.

Here’s how the data is structured:

students.csv:

student_id	name
1	John Doe
2	Jane Smith

attendance.csv:

attendance_id	student_id	date	status
1	1	2024-11-01	Present
2	2	2024-11-01	Absent

The system uses **Pandas** to load, store, and manipulate data. If the CSV files do not exist, they are created automatically.

Core Functionality (25%)

Student Registration Mechanism

The **Student Registration** feature allows students to register themselves by entering their name.

A unique student ID is generated for each student.

Here is the relevant code for the student registration process:

```python
def show_registration():
    registration_window = tk.Toplevel()
    registration_window.title("Student Registration")
    registration_window.config(bg="lightblue")
    registration_window.geometry("1400x700")

    tk.Label(registration_window, text="Student Registration", font=("Arial", 16)).pack(pady=20)

    tk.Label(registration_window, text="Enter your name:", font=("Arial", 12)).pack(pady=5)
    entry_name = tk.Entry(registration_window, font=("Arial", 12))
    entry_name.pack(pady=5)
    tk.Label(registration_window, text="Enter your ID:", font=("Arial", 12)).pack(pady=5)
    entry_id = tk.Entry(registration_window, font=("Arial", 12))
    entry_id.pack(pady=5)

    def register_student():
        name = entry_name.get()
        if name:
            students, _ = load_data()
            student_id = len(students) + 1  # Generate unique student ID
            students = students.append({'student_id': student_id, 'name': name}, ignore_index=True)
            save_data(students, None)
            messagebox.showinfo("Success", f"Student {name} registered successfully!")
            registration_window.destroy()
        else:
            messagebox.showerror("Error", "Please enter a valid name.")

    tk.Button(registration_window, text="Register", command=register_student).pack(pady=10)
```

The register_student function appends the new student to the students.csv file.

Attendance Recording System

Instructors can log in and mark attendance for students. They select a date, and then checkboxes are displayed for each registered student. The instructor can mark attendance by selecting the appropriate checkbox.

Here's the code for attendance marking:

```
def instructor_functions():
    global checkboxes, date_label

    instructor_window = tk.Toplevel()
    instructor_window.title("Instructor Functions")
    instructor_window.geometry("1400x700")
    instructor_window.config(bg="lightblue")

    tk.Label(instructor_window, text="Instructor Attendance Functions", font=("Arial", 24, "bold"), bg="lightblue", fg="black").pack(pady=10)

    frame_attendance = tk.Frame(instructor_window, bg="lightblue")
    frame_attendance.pack(pady=20)

    students, _ = load_data()
    checkboxes = []

    tk.Label(frame_attendance, text="Mark Attendance", bg="lightblue").grid(row=0, column=0, pady=10)
    for index, student in students.iterrows():
        var = tk.BooleanVar()
        tk.Checkbutton(frame_attendance, text=student['name'], variable=var, bg="lightblue").grid(row=index + 1, column=0, sticky='w')
        checkboxes.append(var)

    tk.Button(frame_attendance, text="Record Attendance", command=mark_attendance).grid(row=len(students) + 1, columnspan=2, pady=10)
```

The mark_attendance function stores the attendance records in the attendance.csv file:

```python
def mark_attendance():
    students, attendance = load_data()
    date = date_label.cget("text").split(": ")[-1]

    for idx, student in students.iterrows():
        status = "Present" if checkboxes[idx].get() else "Absent"
        new_record = pd.DataFrame([[len(attendance)+1, student['student_id'], date, status]],
                                  columns=['attendance_id', 'student_id', 'date', 'status'])
        attendance = pd.concat([attendance, new_record], ignore_index=True)

    save_data(students, attendance)
    messagebox.showinfo("Success", "Attendance recorded successfully")
```

Data Storage and Retrieval

The system loads data using the load_data() function and saves it with save_data(). This ensures persistence across sessions. The attendance.csv and students.csv files are updated every time the attendance is marked or a student is registered.

User Interface (UI) Development (15%)

The system's interface is developed using **Tkinter**, making it accessible and easy to navigate. The UI features:

- A login screen for both students and instructors
- A student registration page
- An attendance marking page for instructors

For example, the login page is created with the following code:

```python
def initialize_login():
    global login_window, user_type_var, entry_password

    login_window = tk.Tk()
    login_window.geometry("1400x700")
    login_window.title("Login Page")
    login_window.config(bg="lightblue")

    tk.Label(login_window, text="Welcome To Sigma College", font=("Arial", 40, "bold"), bg="lightblue").pack(pady=10)

    user_type_var = tk.StringVar(value="Student")
    tk.Radiobutton(login_window, text="Student", font=("Arial", 16), variable=user_type_var, value="Student", bg="lightblue").pack(pady=5)
    tk.Radiobutton(login_window, text="Instructor", font=("Arial", 16), variable=user_type_var, value="Instructor", bg="lightblue").pack(pady=5)

    tk.Label(login_window, text="Password (Instructors only)", font=("Arial", 16), bg="lightblue").pack(pady=10)
    entry_password = tk.Entry(login_window, show="*")
    entry_password.pack(pady=5)

    tk.Button(login_window, text="Login", command=handle_login).pack(pady=20)

    login_window.mainloop()
```

Security and Privacy (10%)

Data Privacy Measures

Sensitive student data, such as attendance records, are stored securely in the system. Access to this data is restricted to authorized users only. Additionally, all login credentials (both admin and instructor) are validated before access to data is granted.

Role-Based Access Control

Access to features is determined by the role of the user:

- **Students**: Can only register and access their own information.
- **Instructors**: Can mark attendance, view student lists, and generate attendance reports.

Documentation and Testing (20%)

Documentation

The code is well-documented with inline comments explaining the logic behind the different parts of the system. The **user manual** provides clear instructions for how to use the system, including registration, attendance marking, and generating reports.

Unit Testing

The system undergoes unit tests for key functions:

- **Testing Student Registration**: Verifies that student data is properly saved and retrieved.
- **Testing Attendance Marking**: Ensures that attendance is recorded accurately in the CSV file.

Code Style

The code adheres to **PEP 8** style guidelines, ensuring consistency in naming conventions, indentation, and code structure. This facilitates readability and future maintenance.

Requirements

To successfully run the **Student Attendance Management System**, ensure that your development environment meets the following requirements:

Software Requirements

1. **Python**: The system is developed using **Python 3.7+**. You can download the latest version of Python from python.org.
2. **Tkinter**: This is the default GUI library for Python and is required for the user interface. Tkinter comes pre-installed with Python, so you do not need to install it separately in most environments. However, you can verify or install it using:
3. bash

```
sudo apt-get install python3-tk    # For Ubuntu/Linux
brew install python-tk             # For macOS (if not already installed)
```

4. **Pandas**: This library is used for handling and manipulating CSV data. Install it using:

```
bash
pip install pandas
```

5. **Pillow**: The Pillow library is used for image handling, which is required in your project. You can install it using:

```
bash
pip install pillow
```

6. **qrcode**: This library is used to generate QR codes. You can install it via:

bash

pip install qrcode

7. **tkcalendar**: This library is used for the calendar widget to allow instructors to select dates for attendance. Install it using:

bash

pip install tkcalendar

Hardware Requirements

- **Operating System**: The system should run on any major operating system, including:
  - **Windows** (7 or later)
  - **macOS** (10.12 or later)
  - **Linux** (Ubuntu, Debian, Fedora, etc.)
- **Memory**: A minimum of **2 GB RAM** is recommended for running the application smoothly.
- **Storage**: The application stores student and attendance data in CSV files. Ensure you have at least **50 MB of free disk space** to store data and logs, which should be sufficient for small to medium-sized class data.

Running the Project

1. **Clone or Download the Project Files**:

a. If you have a GitHub repository, you can clone it:

```
bash
git clone https://github.com/yourusername/attendance-management.git
```

b. Alternatively, download the ZIP file of the project and extract it to your desired directory.

2. **Navigate to the Project Directory**:

```
bash
cd /path/to/attendance-management
```

3. **Run the Python Application**: To start the application, run the following command:

```
bash
python3 attendance_management.py
```

4. **Optional - Create Virtual Environment**: For managing dependencies in a controlled environment, you can create a virtual environment:

```
bash
python3 -m venv venv
source venv/bin/activate      # On Windows, use venv\Scripts\activate
```

```
pip install -r requirements.txt
```

Contributions

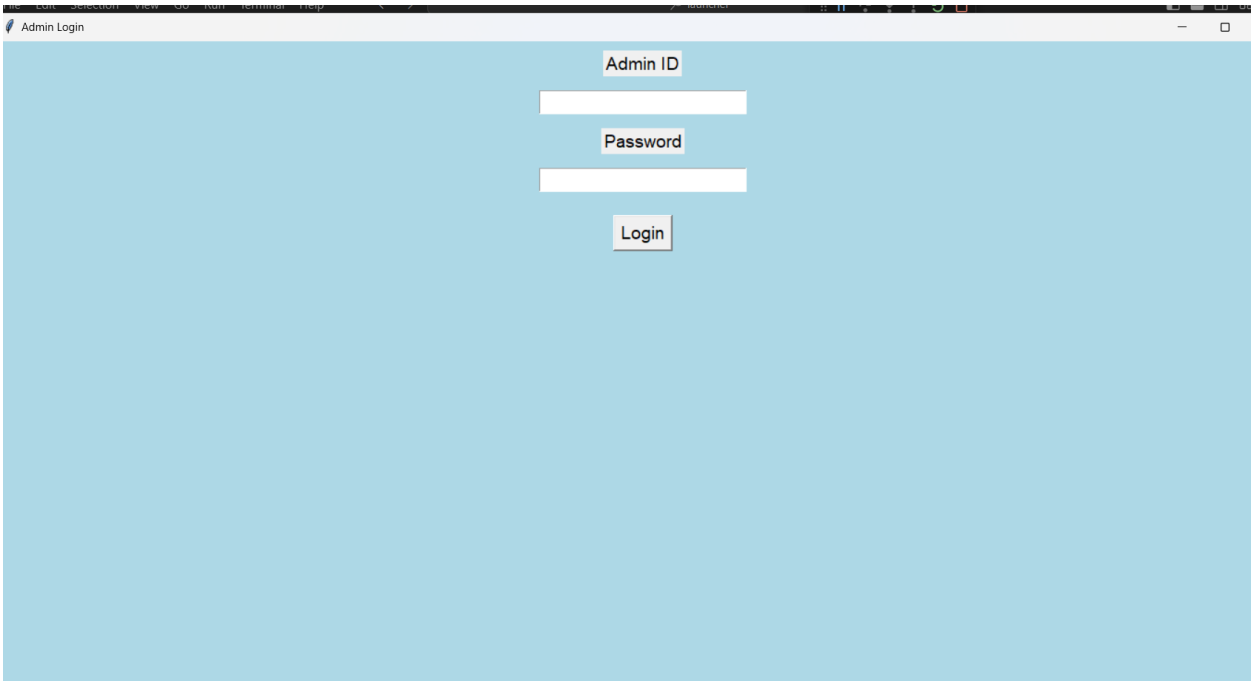
We welcome contributions from developers to improve and extend the system. Fork the repository, make changes, and submit a pull request.

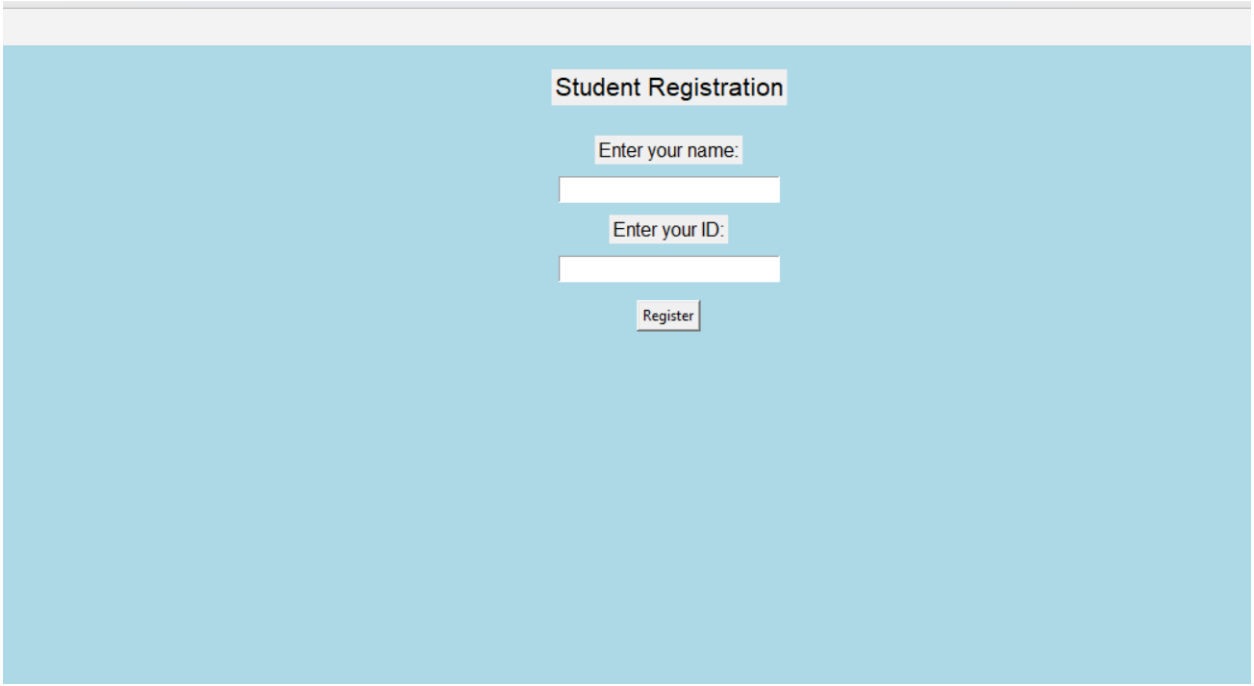
Developers

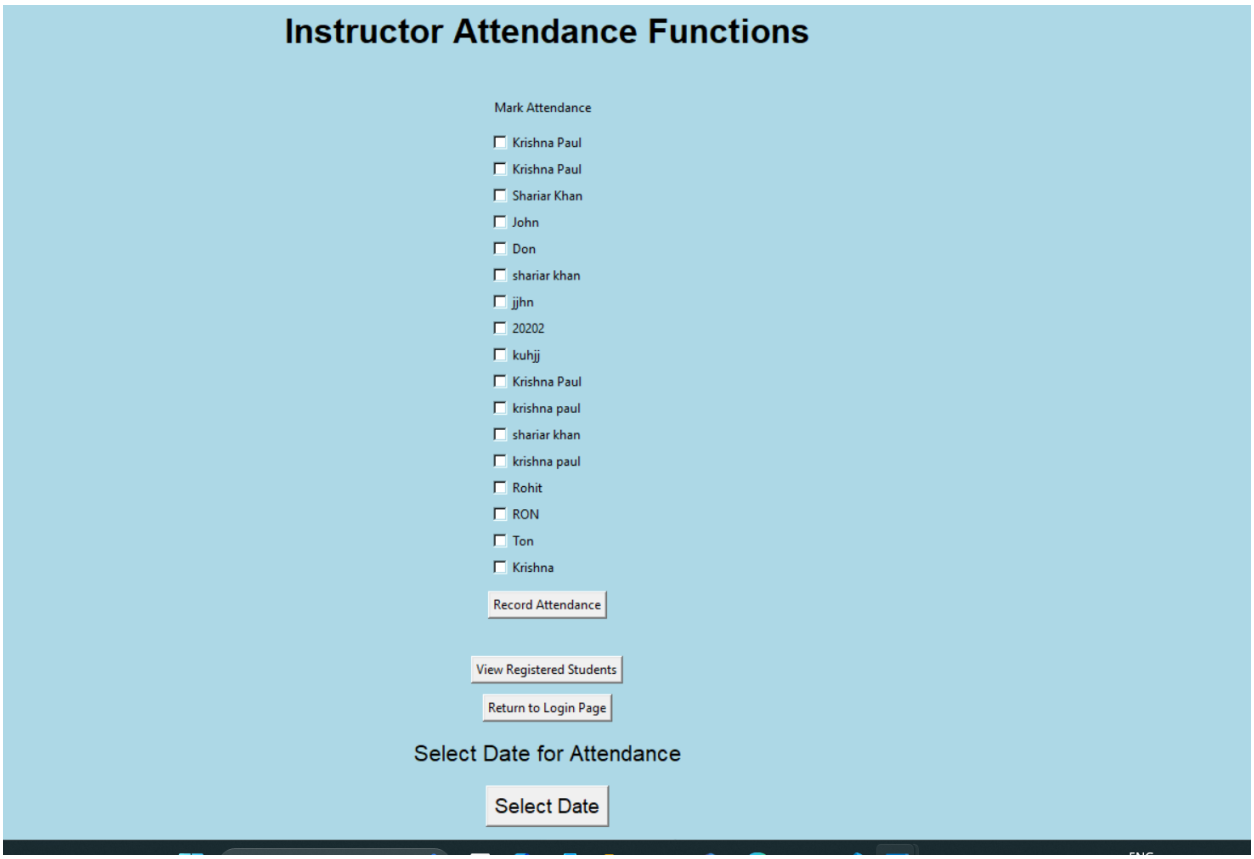
The **Student Attendance Management System** was developed by krishna paul(12033028), shariar khan (12033913)

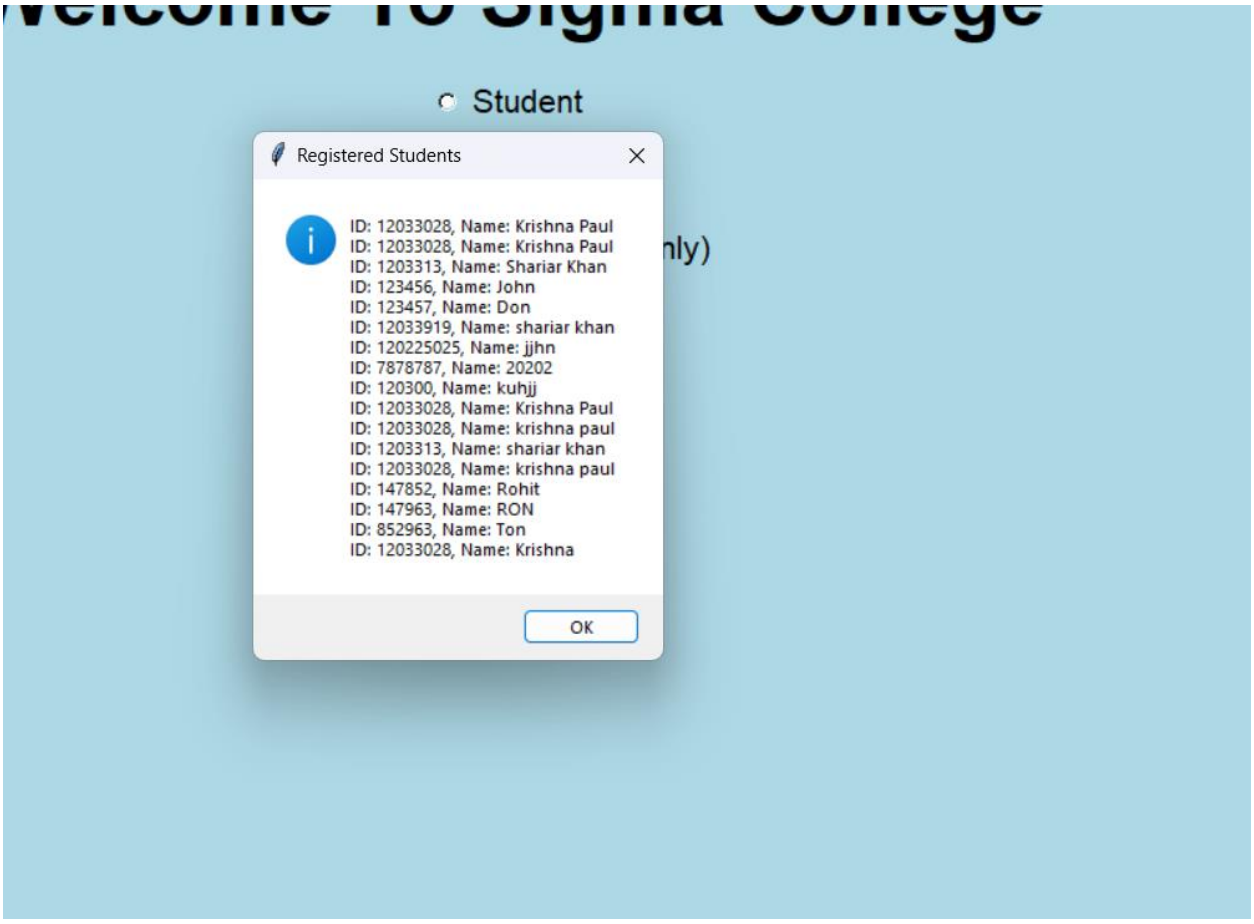
For any inquiries or further information, please contact us via the project repository.

Screenshot


Admin Login
Admin ID
Password
Login


Student Registration
Enter your name:
Enter your ID:
Register


Instructor Attendance Functions
Mark Attendance
Krishna Paul
Krishna Paul
Shariar Khan
John
Don
shariar khan
jjhn
20202
kuhjj
Krishna Paul
krishna paul
shariar khan
krishna paul
Rohit
RON
Ton
Krishna
Record Attendance
View Registered Students
Return to Login Page
Select Date for Attendance
Select Date



GitHub Link:

Student-Attendence-System/Python_assignment final.py at main · Krishna12033/Student-Attendence-System

https://github.com/shariar87/Student-Attendence-Management-system