Attendance Management System

A

Real Time Research/Societal Project Report

Submitted to

Jawaharlal Nehru Technological University, Hyderabad

In partial fulfilment of the requirements for the

award of the degree of

BACHELOR OF TECHNOLOGY

In

COMPUTER SCIENCE AND ENGINEERING

By

POPURI LAKSHMI SAHITHI

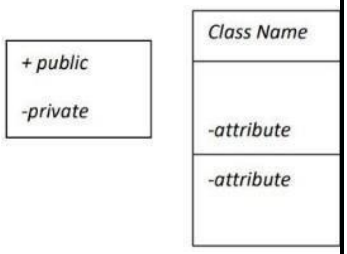
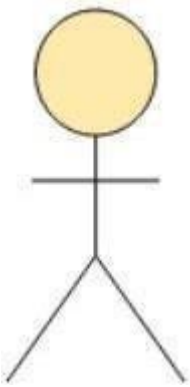
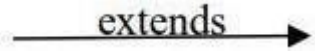
ABSTRACT

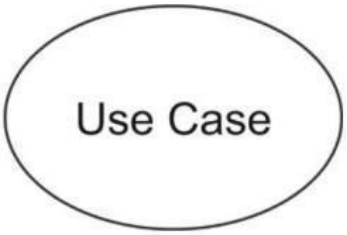
The Attendance Management System (AMS) is a digital solution designed to streamline and automate the process of tracking and managing student. This system aims to address these limitations by introducing advanced features such as real-time attendance tracking, automatic alert generation and notifications. A key feature of this enhanced system is its alert mechanism. When an individual's attendance falls below a predefined threshold, the system automatically triggers alerts to the user. The Attendance Management System enhances overall productivity and accountability, ensuring that attendance data is always up-to-date, accurate, and actionable. By integrating these proactive features, the system promotes a more disciplined environment, reduces administrative overhead, and improves user engagement.

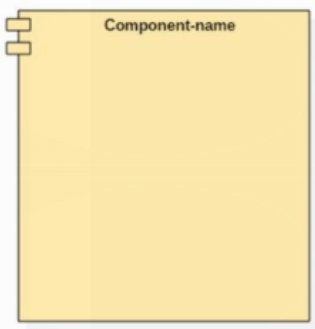
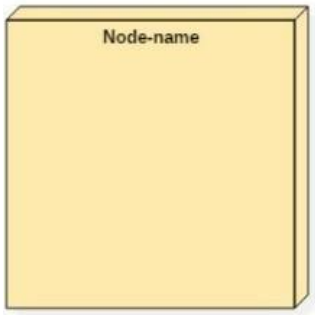
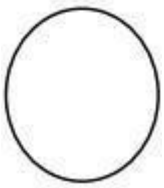
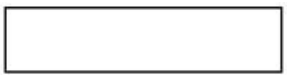
KEYWORDS: *Streamline, Attendance, Digital, Automated, Tracking, Student, Mechanism, Real-Time, Alerts, Management, Data, Accurate, System, Report, Registers, User, Accountable, Features, Notification, Individual.*

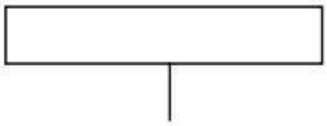
FIG. NO/TAB.NO	LIST OF FIGURES AND TABLES	PAGE NO.

LIST OF SYMBOLS

SNO.	Name of Symbol	Notation	Description
1	CLASS		Represents a collection of similar entities grouped together.
2	ASSOCIATION		Associations represent static relationships between classes. Roles represent the way the two classes see each other.
3	ACTOR		It aggregates several classes into a single class.
4	RELATION (uses)	*Uses*	Used for additional process communication.
5	RELATION (extents)		Extends relationship is used when one use case is similar to another use case.

6	COMMUNICATION		Communication between various use cases.
7	STATE		State of the process
8	INITIAL STATE		Initial state of the object
9	FINAL STATE		Final state of the object
10	CONTROL FLOW		Represents various control flow between the states.
11	DECISION BOX		Represents decision making process from a constraint
12	USE CASE		Interact ion between the system and external environment.

13	COMPONENT		Represents physical modules which is a collection of components.
14	NODE		Represents physical modules which are a collection of components.
15	DATA PROCESS/ STATE		A circle in DFD represents a state or process which has been triggered due to some event or action.
16	EXTERNAL ENTITY		Represents external entities such as keyboard, sensors, etc
17	TRANSITION		Represents communication that occurs between processes.

18	OBJECT LIFELINE		Represents the vertical dimensions that the object communications.
19	MESSAGE	Message	Represents the message exchanged.

CHAPTER 1
INTRODUCTION

1.1 GENERAL

Attendance management is a crucial aspect of academic institutions, directly impacting student performance, discipline, and academic records. Traditional methods of attendance tracking, such as manual registers or paper-based logs, are time-consuming, error-prone, and lack real-time monitoring capabilities.

This project, titled “Online Attendance Management System with Absentee and Low Attendance Notifications,” aims to streamline and digitize the attendance process. It provides a user-friendly online platform where students can mark their attendance digitally. The system automatically tracks absentees, informs their parents through notifications, and alerts the admin when a student’s attendance drops below a predefined threshold (e.g., 75%)

By automating these tasks, the system reduces administrative workload, ensures timely communication, and promotes accountability among students. The solution is scalable, efficient, and can be integrated with other academic systems to enhance institutional transparency and effectiveness.

The Online Attendance Management System is designed to digitize and automate the process of recording and managing student attendance in educational institutions. It eliminates the need for manual roll calls or physical registers by providing a centralized platform where students can mark their attendance online.

The system ensures accuracy, transparency, and real-time monitoring of attendance data. It is built to support daily attendance tracking, generate absentee lists automatically, notify parents about absences, and alert administrators about students falling below a set attendance percentage.

This solution can be used by schools, colleges, and training institutes to save time, reduce human error, and improve the communication loop between faculty, students, and parents. With the ability to scale and extend its features, this system serves as a reliable and efficient tool for managing attendance in a modern, tech-driven academic environment.

1.2 PROBLEM STATEMENT

In many educational institutions, attendance tracking is still conducted manually using physical registers or spreadsheets. This traditional method is time-consuming, inefficient, and prone to human errors such as incorrect entries, misplacement of records, and lack of timely updates.

Moreover, manual systems do not offer real-time absentee tracking or automated notifications to parents and administrators. As a result, important actions—like addressing chronic absenteeism or identifying students at risk—are often delayed or overlooked. There is also a lack of transparency and accountability, as students may falsely claim presence, and faculty may miss recording or analyzing attendance trends. Monitoring low attendance and communicating it to concerned stakeholders is tedious without automation.

Hence, there is a need for a reliable, online attendance management system that automates attendance marking, detects absentees, and sends timely notifications to both parents and administrators—thereby improving efficiency, accountability, and communication.

1.3.EXISTING SYSTEM

Many institutions have started shifting from manual attendance to online attendance systems, which typically involve faculty logging in to a web portal or app to mark attendance. These systems may use basic form submissions, spreadsheets uploaded through a portal, or simple login timestamps.

While these online systems improve over manual methods, they still have several limitations that affect their effectiveness and automation:

Faculty can mark attendance using a web-based portal.

Students may log in to mark themselves present in some self-mark systems.

Basic reporting such as daily or monthly attendance summaries.

Excel/CSV export of attendance data.

1. No Absentee Alerts:
   In many current attendance systems, when a student is marked absent, there is no automatic notification sent to the concerned stakeholders. This means:

- Parents remain unaware of their child's absence unless manually informed by school staff.
- Administrators have no instant insight into daily absentees unless they manually review the records.
- Delayed response to critical or frequent absenteeism can result in poor student performance or hidden issues (like health or behavioral concerns) going unnoticed.

2. Lack of Automation:

Many systems still operate in a semi-manual fashion:

- Teachers or admins have to open forms daily, click checkboxes for each student, and submit attendance manually.
- No integration with timetables, holiday calendars, or recurring absentee alerts.
- No logic to automatically skip Sundays, public holidays, or flag long-term absentees.

3. No Low Attendance Monitoring:

Low attendance is a serious issue in many educational institutions, yet most systems:

- Do not monitor attendance percentage on a regular basis.
- Fail to trigger warnings when a student falls below a set attendance threshold (e.g., 75%).
- Lack tools to highlight students at risk of academic penalties due to poor attendance.

4. Limited Communication Integration:

Most systems are isolated platforms without integration to external communication tools. They lack:

- Built-in SMS functionality to notify parents in real time.
- Email notifications for reports or alerts.
- WhatsApp API integration (which is widely used and cost-effective in India).

- Admin dashboards to monitor communication logs.

5. Lack of Real-time Notifications:

Even if attendance is recorded properly, many systems only generate static reports that must be downloaded or viewed manually.

- No real-time push notifications or alerts for students with repetitive absences.
- No live updates when a student is marked absent.
- No threshold-based triggers (e.g., 3 consecutive days absent → auto-alert).

While current online attendance systems reduce paperwork and improve convenience, they fall short in automating key responsibilities like absentee tracking, parent communication, and low attendance alerts. This creates a gap in communication and enforcement, especially in large institution.

1.3.1 DISADVANTAGES OF EXISTING SYSTEM

Despite the shift toward digital solutions, existing online attendance management systems still suffer from several functional and operational limitations. These drawbacks hinder their effectiveness in creating a fully automated, transparent, and responsive attendance ecosystem.

1**. No Real-time Absentee Alerts**

The system does not instantly inform the admin or parents when a student is absent, resulting in delayed awareness and action.

2**. Manual Data Entry Dependency**

Even in online platforms, attendance marking often requires manual input by faculty or students, which can still lead to errors or misuse (e.g., proxy attendance).

3. **Lack of Parent Involvement**

Most systems do not notify parents about student absences or low attendance, which can lead to uninformed guardians and lack of timely corrective action.

4. **No Automatic Attendance Percentage Calculation**

The system may store attendance data but often lacks automated analytics to calculate percentages and generate alerts for low attendance.

5. **Limited Customization**

Many existing systems are rigid, lacking customizable features like time-based cutoffs, scheduled reports, or integration with academic systems.

6. **No Alert System for Chronic Defaulters**

Students with consistently low attendance are not flagged or tracked automatically, which makes it hard for faculty to take preventive action early.

7. **Security and Data Integrity Issues**

Poorly designed systems may be vulnerable to data manipulation, unauthorized access, or accidental data loss.

These limitations highlight the need for a more advanced, automated system that not only records attendance online but also enhances monitoring, communication, and decision-making through smart notifications and analytics.

1.4 PROPOSED SYSTEM

The proposed system is a fully automated Online Attendance Management System designed to overcome the limitations of both manual and existing online systems. It not only allows students to mark attendance online but also includes real-time absentee tracking, automatic low attendance alerts, and parent/admin notifications, creating a smart and responsive attendance environment.

1. **Online Attendance Marking**

Students can mark their attendance through a secure online portal or interface.

2. **Automatic Absentee Detection**

At the end of each attendance session, the system automatically generates a list of students who failed to mark attendance.

3. **Admin Notifications**

Absentee data is immediately sent to the admin for daily monitoring.

4**. Parent Notification System**

Parents receive SMS or email alerts when their child is absent, ensuring they are informed in real-time.

5. **Low Attendance Monitoring**

The system calculates the attendance percentage for each student.

If the attendance falls below a predefined threshold (e.g., 75%), alerts are sent to both admin and parents.

6. **Centralized Record Management**

All attendance records are securely stored in a database and can be retrieved, analyzed, or exported when needed.

7. **User-Friendly Dashboard**

A simple and intuitive interface for students, teachers, and admins to view attendance status, percentages, and alerts.

The proposed system ensures a transparent, automated, and efficient approach to attendance management by integrating online attendance, data analytics, and communication tools. It enhances administrative control, supports timely interventions, and strengthens the relationship between institutions, students, and parents.

1.4.1 ADVANTAGES OF PROPOSED SYSTEM

The proposed Online Attendance Management System offers several significant advantages over traditional and existing digital methods. It brings automation, accuracy, and communication to the forefront of attendance tracking, ensuring a modern and reliable solution for educational institutions.

1. **Time-Saving and Efficient**

Attendance is marked online, reducing the time spent on daily roll calls.
Automated absentee detection and notifications minimize administrative work.

2. **Real-Time Notifications**

Absentees are identified instantly, and notifications are sent to both admins and parents without delay.

3. **Improved Communication**

Parents stay informed about their child's attendance status regularly, fostering better accountability and involvement.

4. **Automated Low Attendance Alerts**

The system continuously monitors attendance percentages and triggers alerts when they fall below the set threshold (e.g., 75%).

5. **Accurate and Reliable Records**

Digital storage ensures accurate, tamper-proof attendance records that are easy to retrieve, analyze, and export.

6. **Student Accountability**

With parents and admins kept in the loop, students are more likely to attend regularly and take responsibility for their attendance.

7**. Data Analytics and Reporting**

Provides attendance summaries, student-wise reports, and insights that help in academic decision-making.

8. **Scalability and Flexibility**

The system can be scaled for different institutions and upgraded to include new features like biometric login or facial recognition.

9. **Secure and Centralized Data**

All data is stored securely in a centralized database, reducing the risk of loss or manipulation.

These advantages make the proposed system a comprehensive solution for modern attendance management. It not only simplifies attendance tracking but also enhances institutional transparency, student discipline, and parental engagement.

CHAPTER 2
LITERATURE SURVEY

2.1. RFID-Based Attendance System

Authors: ***N. Sandeep, R. K. Sharma***

Journal/Conference: International Journal of Computer Applications, 2015

This system used RFID tags for students to mark attendance by scanning their ID cards. It reduced manual entry but lacked any parental notification or absentee monitoring features. There will be no SMS or email alerts for absentees.

There will be no attendance percentage tracking or automation for low attendance alerts.

It will be fully depended on students physically scanning cards, which can be bypassed by proxy.

2.2. Biometric Fingerprint Attendance System

Authors: ***S. S. Manvi, P. Laxmi, A. Desai***

Journal: Procedia Computer Science, 2016

The authors proposed a fingerprint scanner system that ensured higher accuracy by using unique biometric data to prevent proxy attendance. No communication with parents or admins on absentee status. No real-time alerts or low attendance warnings. Hardware-based model increases cost and complexity.

2.3. SMS-Based Attendance System for Monitoring Students

Authors: ***K. Dinesh Kumar, P. Kavitha***

Journal: International Journal of Engineering Research & Technology (IJERT), 2017

Focused on using GSM modules to send SMS to parents when students were marked absent. Lacked UI for real-time tracking by admins. No smart scheduling or integration with academic calendars. System only sent SMS, no detailed attendance analytics or reports.

2.4. Mobile Application for Attendance Monitoring

Authors: ***A. Sharma, M. Kaur***

Conference: IEEE International Conference on Mobile Computing, 2018

Used a mobile app to mark attendance and allowed limited reporting features. The app could generate basic attendance reports. There are No automatic alert system for low attendance, No parent communication or admin notifications and App was not capable of real-time monitoring of absenteeism.

2.5. Automated Attendance with Real-Time Face Recognition

Authors: ***R. Patel, D. R. Shah***

Journal: International Research Journal of Engineering and Technology (IRJET), 2020

This system used face detection for attendance and stored data in a central database. Some of the limitations are High implementation cost due to camera and software requirements, No integration with SMS or email APIs for absentee alerts, Didn't track low attendance or defaulter detection over time.

2.6. IoT-Based Smart Attendance System with Alerting

Authors: ***T. Ramesh, P. Bhuvaneshwari***

Journal: International Journal of Innovative Research in Computer and Communication Engineering, 2021

Proposed an IoT-enabled attendance system that can trigger alerts. However, it was more theoretical and lacked proper implementation or communication API integration. There will be no practical deployment examples,

no complete cycle of attendance tracking, report generation, and communication, no real-time performance logs or response-based messaging.

CHAPTER 3

TECHNICAL REQUIREMENTS

3.1 GENERAL

These are the requirements for doing the project.
They are:

1. Hardware Requirements
2. Software Requirements

3.2 HARDWARE REQUIREMENTS

The hardware requirements may serve as the basis for a contract for the implementation of the system and should therefore be a complete and consistent specification of the whole system. They are used by software engineers as the starting point for the system design. It should be what the system does and not how it should be implemented.
Minimum hardware requirements are very dependent on the particular software being developed by a given Enthought Python / Canopy / VS Code user. Applications that need to store large arrays/objects in memory will require more RAM, whereas applications that need to perform numerous calculations or tasks more quickly will require a faster processor.

- **Processor : Minimum intel i3**
- **RAM : 4 GB**
- **Hard disk : 500 GB**

3.3 SOFTWARE REQUIREMENTS

The software requirements document is the specification of the system. It should include both a definition and a specification of requirements. It is a set of what the system should do rather than how it should do it. The software requirements provide a basis for creating the software requirements specification. It is useful in adding student, marking the attendance, viewing the attendance and tracking the students throughout the development activity. The functional requirements or the overall description documents include the product perspective and features, operating system and operating environment, graphics requirements, design constraints and user documentation.

The appropriation of requirements and implementation constraints gives the general overview of the project in regards to what the areas of strength and deficit are and how to tackle them.

- **Windows OS 10**
- **Linux OS – Ubuntu 20.04**
- **Python 3.12.5**
- **MySQL (for backend)**
- **Modules Req.- tkinter, smtplib, requests, datetime, pandas, sqlite3**
- **APIs Req.- Fast2SMS, Twilio, SMTP**
- **VS Code, PyCharm**

CHAPTER-4

SYSTEM DESIGN

4.1 GENERAL

The Smart Attendance Management System with Automated Alerts is designed with a modular and layered architecture to streamline the attendance process and ensure instant communication with stakeholders. The system consists of four main layers: the presentation layer, application logic, data management, and communication layer. The presentation layer provides an intuitive graphical interface (built using Python's Tkinter or a web interface using Flask) where teachers or administrators can log in securely and mark student attendance. This interface interacts with the application layer, which contains the core logic to validate attendance, monitor absenteeism, and detect students with low attendance percentages. Attendance data is stored in a centralized database such as SQLite or MySQL, which holds student records, daily logs, and contact details.

Once a student is marked absent, the system automatically checks whether it is a regular school day, and if so, it triggers the alert mechanism. The notification layer is responsible for sending SMS alerts using Fast2SMS API and email notifications using SMTP protocol. Optionally, the system can be integrated with WhatsApp Business API for message delivery via WhatsApp. Additionally, the backend includes modules for generating reports, monitoring weekly/monthly attendance trends, and identifying students who consistently fall below a set attendance threshold. These defaulters can be flagged, and a second round of alerts can be triggered to inform their guardians or school authorities. The system also maintains logs of alerts sent for transparency and follow-up. Overall, the system is designed to be secure, responsive, and scalable, with the goal of reducing manual workload and improving student accountability through real-time communication.

Three key considerations involved in the feasibility analysis are

- ECONOMICAL FEASIBILITY
- TECHNICAL FEASIBILITY
- SOCIAL FEASIBILITY

- **ECONOMICAL FEASIBILITY**

This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited.

The expenditures must be justified. Thus the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

- **TECHNICAL FEASIBILITY**

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

- **SOCIAL FEASIBILITY**

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system.

4.2 SYSTEM ARCHITECTURE

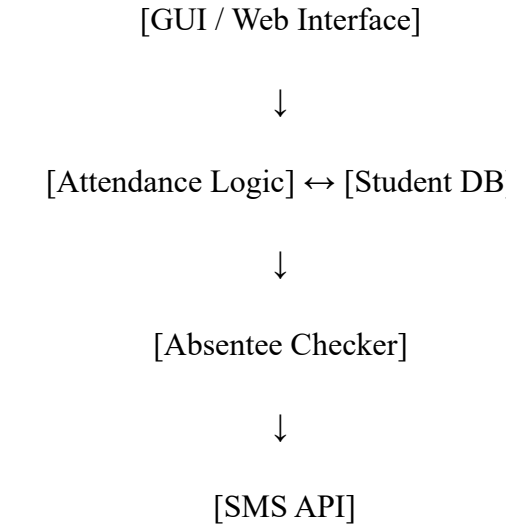


Figure 4.1: Architecture Diagram

4.3 UML DESIGN

Unified Modeling Language (UML) is a general purpose modeling language. The main aim of UML is to define a standard way to visualize the way a system has been designed.
It is quite similar to blueprints used in other fields of engineering.

UML is not a programming language; it is rather a visual language.Use UML diagrams to portray the behavior and structure of a system, UML helps software engineers, businessmen and system architects with modeling, design and analysis.

It’s been managed by OMG ever since. International Organization for Standardization (ISO) published UML as an approved standard in 2005. UML has been revised over the years and is reviewed periodically.

UML combines best techniques from data modeling (entity relationship diagrams), business modeling (work flows), object modeling, and component modeling. It can be used with all processes, throughout the software development life cycle, and across different implementation technologies.

UML has synthesized the notations of the Booch method, the Object-modeling technique (OMT) and Object-oriented software engineering (OOSE) by fusing them into a single, common and widely usable modeling language. UML aims to be a standard modeling language which can model concurrent and distributed systems.

The Unified Modeling Language (UML) is used to specify, visualize, modify, construct and document the artifacts of an object-oriented software intensive system under development. UML offers a standard way to visualize a system's architectural blueprints, including elements such as: ▪ Actors ▪ Business processes ▪ (logical) Components ▪ Activities ▪ Programming Language Statements ▪ Database Schemes ▪
Reusable software components.

- ➢ Complex applications need collaboration and planning from multiple teams and hence require a clear and concise way to communicate amongst them.
- ➢ Businessmen do not understand code. So UML becomes essential to communicate with non-programmer's essential requirements, functionalities and processes of the system.
- ➢ A lot of time is saved down the line when teams are able to visualize processes, user interactions and static structure of the system.
- ➢ UML is linked with object oriented design and analysis. UML makes the use of elements and forms associations between them to form diagrams. Diagrams in UML can be broadly classified as:

The Primary goals in the design of the UML are as follows

- ➢ Provide users a ready-to-use, expressive visual modeling Language so that they can develop and exchange meaningful models.
- ➢ Provide extendibility and specialization mechanisms to extend the core concepts.
- ➢ Be independent of particular programming languages and development processes.
- ➢ Provide a formal basis for understanding the modeling language.
- ➢ Encourage the growth of the OO tools market.
- ➢ Support higher level development concepts such as collaborations, frameworks, patterns and components.
- ➢ Integrate best practices.

4.3.1 USE-CASE DIAGRAM

A Use Case is a kind of behavioral classifier that represents a declaration of an offered behavior.

Each use case specifies some behavior, possibly including variants that the subject can perform in collaboration with one or more actors. Use cases define the offered behavior of the subject without reference to its internal structure.

These behaviors, involving interactions between the actor and the subject, may result in changes to the state of the subject and communications with its environment. A use case can include possible variations of its basic behavior, including exceptional.

The primary components of a use case diagram include:

- **Actor**

An actor is an external entity that interacts with the system. Actors can be people, other systems, or even hardware devices. Actors are represented as stick figures or simple icons. They are placed outside the system boundary, typically on the left or top of the diagram.

- **Use Case**

  A use case represents a specific functionality or action that the system can perform in response to an actor's request. Use cases are represented as ovals within the system boundary.

  The name of the use case is written inside the oval.

- **Association Relationship**

  An association relationship is a line connecting an actor to a use case. It represents the interaction or communication between an actor and a use case.

  The arrowhead indicates the direction of the interaction, typically pointing from the actor to the use case.

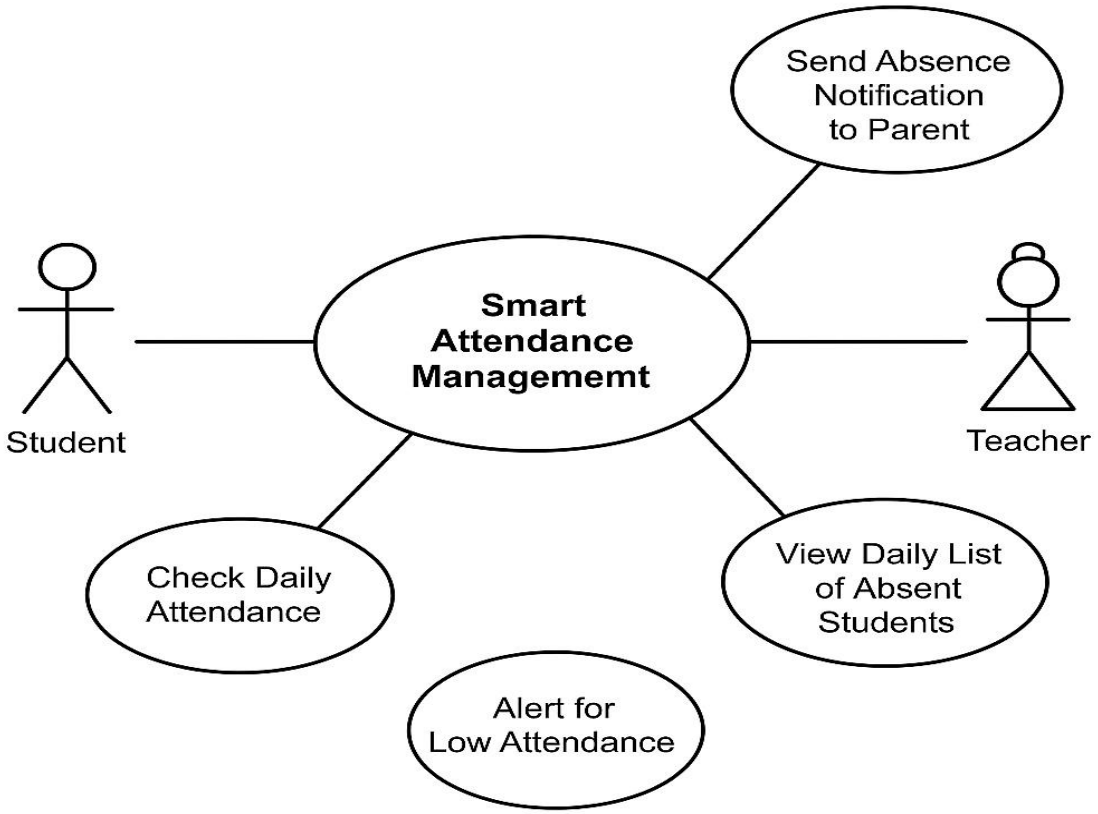


Figure 4.2: Use-Case-Diagram

4.3.2 CLASS DIAGRAM

A class diagram in Unified Modeling Language (UML) is a type of structural diagram that represents the static structure of a system by depicting the classes, their attributes, methods, and the relationships between them. Class diagrams are fundamental in object-oriented design and provide a blueprint for the software's architecture.

Here are the key components and notations used in a class diagram:

- **Class**

A class represents a blueprint for creating objects. It defines the properties (attributes) and behaviours (methods) of objects belonging to that class. Classes are depicted as rectangles with three compartments: the top compartment contains the class name, the middle compartment lists the class attributes, and the bottom compartment lists the class methods.

- **Attributes**

Attributes are the data members or properties of a class, representing the state of objects. Attributes are shown in the middle compartment of the class rectangle and are typically listed as a name followed by a colon and the data type (e.g., name: String).

- **Methods**

Methods represent the operations or behaviours that objects of a class can perform. Methods are listed in the bottom compartment of the class rectangle and include the method name, parameters, and the return type (e.g., calculateCost(parameters):
ReturnType).

- **Visibility Notations**

  Visibility notations indicate the access level of attributes and methods. The common notations are:

  + (public): Accessible from anywhere.

  - (private): Accessible only within the class.

  # (protected): Accessible within the class and its subclasses.

  ~ (package or default): Accessible within the package.

- **Associations**

  Associations represent relationships between classes, showing how they are connected. Associations are typically represented as a solid line connecting two classes. They may have multiplicity notations at both ends to indicate how many objects of each class can participate in the relationship (e.g., 1..*).

  Aggregations and Compositions: Aggregation and composition are special types of associations that represent whole-part relationships. Aggregation is denoted by a hollow diamond at the diamond end, while composition is represented by a filled diamond. Aggregation implies a weaker relationship, where parts can exist independently, while composition implies a stronger relationship, where parts are dependent on the whole.

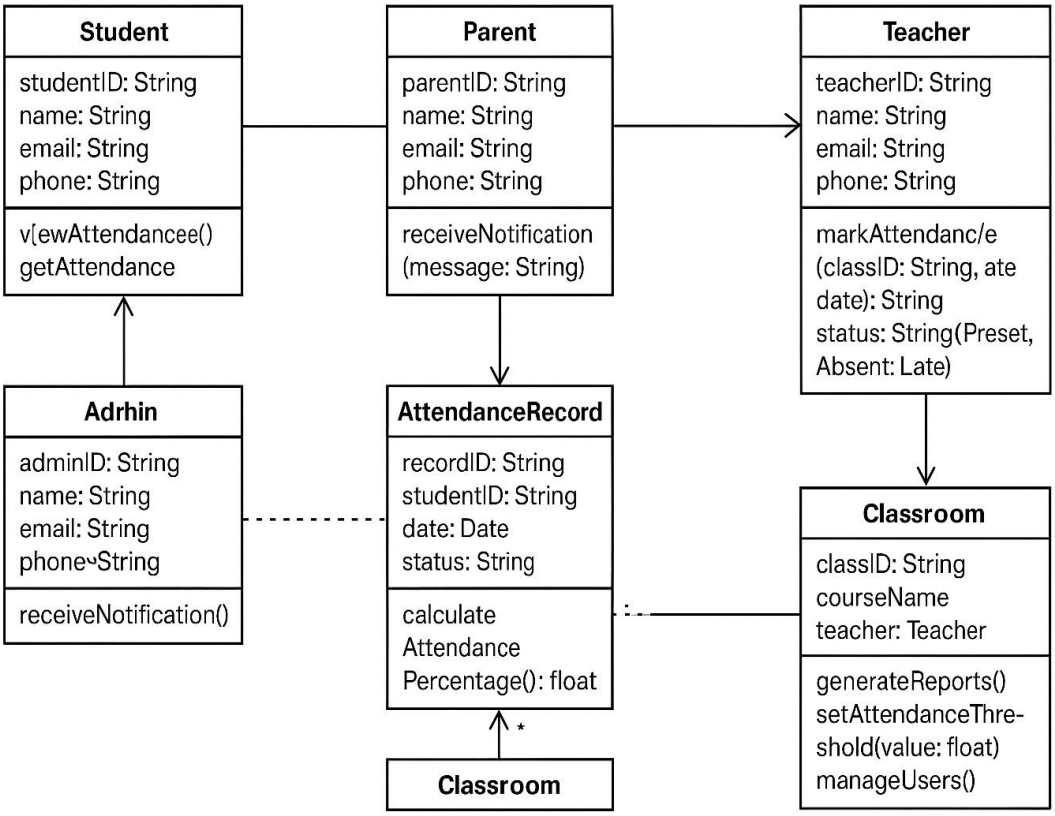


Figure 4.3: Class diagram

4.3.3 ACTIVITY DIAGRAM

An activity diagram portrays the control flow from a start point to a finish point showing the various decision paths that exist while the activity is being executed.

The diagram might start with an initial activity such as "User approaches the door." This activity triggers the system to detect the presence of the user's Bluetooth-enabled device, initiating the authentication process.
Next, the diagram could depict a decision point where the system determines whether the detected device is authorized. If the device is recognized as authorized, the diagram would proceed to the activity "Unlock the door." Conversely, if the device is not authorized, the diagram might show alternative paths such as prompting the user for additional authentication credentials or denying access.

The key components and notations used in an activity diagram:

- **Initial Node**

  An initial node, represented as a solid black circle, indicates the starting point of the activity diagram. It marks where the process or activity begins.

- **Activity/Action**

  An activity or action represents a specific task or operation that takes place within the system or a process. Activities are shown as rectangles with rounded corners. The name of the activity is placed inside the rectangle.

- **Control Flow Arrow**

Control flow arrows, represented as solid arrows, show the flow of control from one activity to another. They indicate the order in which activities are executed.

- **Decision Node**

  A decision node is represented as a diamond shape and is used to model a decision point or branching in the process. It has multiple outgoing control flow arrows, each labeled with a condition or guard, representing the possible paths the process can take based on condition.

- **Merge Node**

  A merge node, also represented as a diamond shape, is used to show the merging of multiple control flows back into a single flow.

- **Fork Node**

  A fork node, represented as a black bar, is used to model the parallel execution of multiple activities or branches. It represents a point where control flow splits into multiple concurrent paths**.**

- **Join Node**

  A join node, represented as a black bar, is used to show the convergence of multiple control flows, indicating that multiple paths are coming together into a single flow.

- **Final Node**

  A final node, represented as a solid circle with a border, indicates the end point of the activity diagram. It marks where the process or activity concludes.

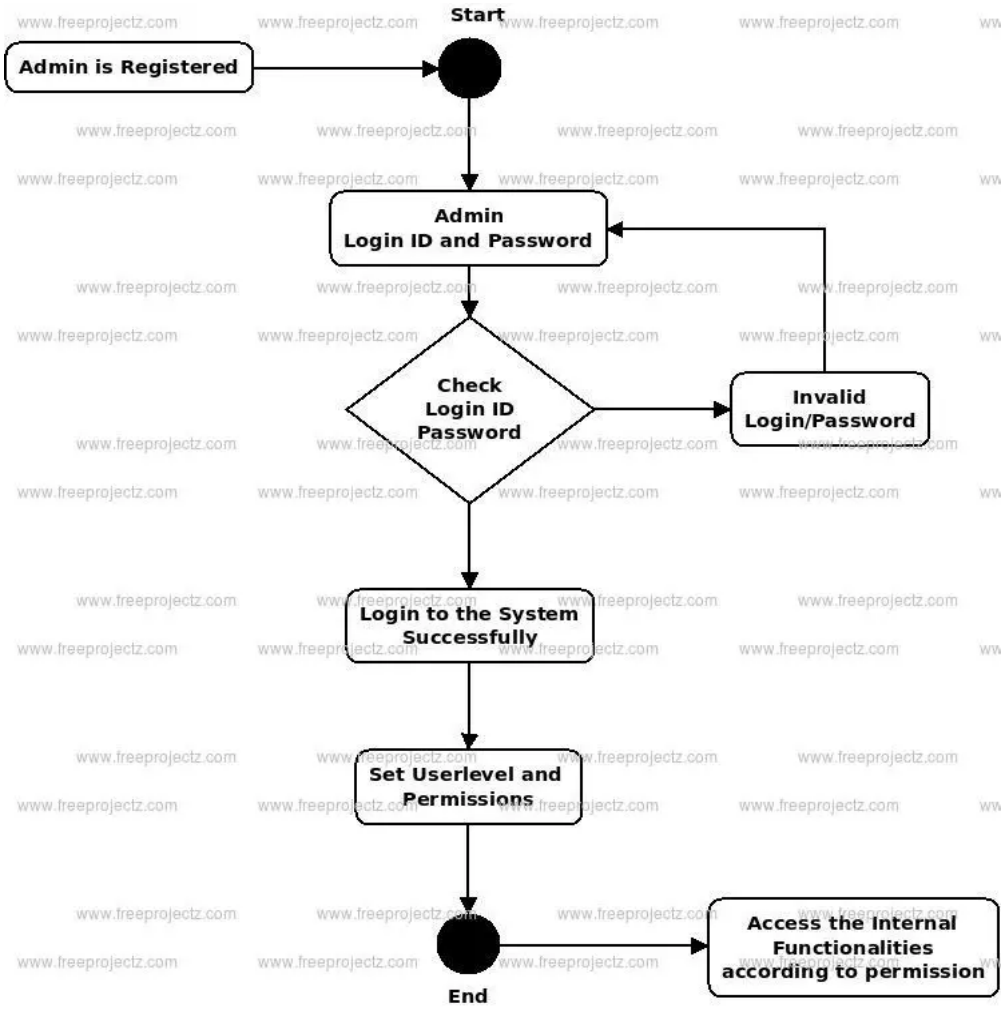


Figure 4.4: Activity diagram

CHAPTER-5
TECHNOLOGY DESCRIPTION

5.1 WHAT IS PYTHON

Python is a high-level, interpreted programming language created by Guido van Rossum and first released in 1991. It is known for its simple and readable syntax, making it an excellent choice for beginners as well as experienced developers. Python is dynamically typed, meaning you don't have to declare variable types, and it runs line by line, which makes debugging easier. It supports multiple programming paradigms, including object-oriented, procedural, and functional programming, allowing flexibility in how code is written and structured.

One of Python's biggest strengths is its extensive standard library and large ecosystem of third-party libraries. These libraries cover a wide range of applications—from web development (with frameworks like Django and Flask) and data analysis (using Pandas and NumPy), to artificial intelligence and machine learning (with tools like TensorFlow and Scikit-learn). Python is also widely used for automation, scripting, creating desktop applications, and even game development. Its cross-platform nature means Python code can run on various operating systems such as Windows, macOS, and Linux without modification.

Because of its versatility and strong community support, Python has become one of the most popular programming languages in the world. Whether you're building a small script to automate tasks or developing complex machine learning models, Python provides the tools and simplicity needed to get the job done efficiently. With its clean syntax and vast capabilities, Python continues to be a go-to language for developers in many fields.

Python's popularity continues to grow because it is not just easy to learn but also highly productive. Educational institutions across the globe use Python as the first language to teach programming. Major companies like Google, Netflix, Instagram, and NASA use Python in real-world projects because of its efficiency and scalability. Moreover, Python integrates well with other languages such as C/C++ and Java, making it a flexible option for software development.

The open-source nature of Python allows anyone to contribute to its improvement, leading to constant updates and a vibrant developer community. Platforms like GitHub, Stack Overflow, and PyPI (Python Package Index) provide rich resources, ready-to-use code, and support for developers at all levels. Whether you're building small hobby projects or large-scale enterprise systems, Python provides the tools, libraries, and frameworks to help you achieve your goals.

In summary, Python stands out as a powerful yet beginner-friendly language with vast application areas and community support. Its readable syntax, dynamic features, and extensive libraries make it suitable for everything from data science and artificial intelligence to web apps and automation. Learning Python opens doors to a wide range of careers and project opportunities in the tech world.

5.2 ADVANTAGES OF JAVA

- **Platform Independence**: Python programs are compiled into bytecode, which can run on any device that has a Python IDLE Shell. This "write once, run anywhere" capability allows Python applications to be highly portable across different platforms without requiring recompilation.

- **Object-Oriented**: Python is based on the object-oriented programming (OOP) paradigm, which promotes modular design, code reuse, and easier maintenance. Objects in Python encapsulate data and behavior, making it easier to model complex systems.

1. Easy to Learn and Use

- Python has a clean and readable syntax, which resembles plain English.
- Ideal for beginners and reduces the learning curve for new programmers.

2. Free and Open Source

- Python is completely free to use, even for commercial purposes.
- The source code is open, so developers can modify or contribute to it.

3. Versatile and Cross-Platform

- Python runs on Windows, macOS, Linux, and more without needing to change the code.
- Suitable for a wide range of applications: web development, data science, AI, game development, automation, etc.

4. Huge Standard Library and Third-Party Modules

- Comes with a rich standard library (math, OS, file handling, etc.).
- Thousands of third-party libraries like NumPy, Pandas, TensorFlow, Flask, etc., are available to simplify tasks.

5. Strong Community Support

- A large, active community means you can easily find help, tutorials, and solutions to problems.
- Platforms like Stack Overflow, GitHub, and Python.org offer great support..

6. Great for Rapid Development

- Python allows quick prototyping and development, making it ideal for startups and projects that need fast results.

- **7. Used in Emerging Technologies**

  - Python is the preferred language in fields like **Artificial Intelligence**, **Machine Learning**, **Data Science**, and **Cybersecurity**.

5.3 LIBRARIES

- **1. Data Analysis & Manipulation**

  Pandas:
  Used for data manipulation and analysis. It provides data structures like DataFrames for handling structured data (like tables). With Pandas, you can clean, filter, and analyse datasets easily.

  NumPy(Numerical-Python):
  Provides support for large multi-dimensional arrays and matrices. It also includes mathematical functions for linear algebra, Fourier transforms, and statistical operations.

- **2. Data Visualization**

  Matplotlib:
  A plotting library used to create static, interactive, and animated visualizations such as bar charts, line graphs, scatter plots, etc.

  Seaborn:
  Built on top of Matplotlib, it offers a higher-level interface for drawing attractive and informative statistical graphics.

- **3. Machine Learning & AI**

Scikit-learn:

A simple and efficient tool for data mining and machine learning. It supports classification, regression, clustering, dimensionality reduction, and model evaluation.

TensorFlow:

An open-source library developed by Google for deep learning and neural networks. It is widely used for AI applications.

Keras:

A high-level neural networks API, built on top of TensorFlow, that allows for easy and fast prototyping of deep learning models.

- **4. Web Development**

Flask:

A lightweight and flexible web framework used to build simple web applications quickly.

Django:

A high-level web framework that follows the "batteries-included" philosophy. It is great for building secure and scalable full-stack web applications.

- **5. GUI Development**

Tkinter:

The standard GUI package for Python. Used for building simple desktop applications with graphical interfaces.

PyQt/PySide:

Used to create advanced desktop applications with features like buttons, menus, and tabs.

- **6. Automation & Scripting**

OS/Shutil:

Used for file and directory management, system commands, etc.

Selenium:

Used for automating web browser actions—commonly used for testing websites and scraping data.

Requests:

A user-friendly HTTP library for sending HTTP requests like GET and POST. Commonly used for APIs and web scraping.

- **7. Game Development**

Pygame:

A set of Python modules designed for writing video games. It provides functionality like graphics, sound, and input handling.

5.4 DISADVANTAGES OF PYTHON

- **1. Slower Execution Speed**

Python is an **interpreted language**, meaning it executes code line by line, which makes it slower compared to compiled languages like C or C++. For tasks requiring high performance (like heavy computation or graphics), this can be a drawback.

- **2. High Memory Usage**

Python uses more memory than other languages, especially when handling large data sets. Its dynamic typing and flexibility come at the cost of increased memory consumption, making it less suitable for mobile applications or devices with limited resources.

- **3. Not Ideal for Mobile App Development**

While Python is strong in desktop and web development, it is **not widely used for mobile app development**. There are tools like Kivy and BeeWare, but they are not as mature or commonly used as Java (for Android) or Swift (for iOS).

◇ 4. Database Access Limitations

Python's database access layers are **not as advanced** or robust as those in languages like Java or .NET. For complex enterprise-level database applications, Python might not be the first choice.

◇ 5. Weak in Multithreading

Python has a **Global Interpreter Lock (GIL)** that restricts execution of multiple threads at the same time. This limits Python's performance in CPU-intensive multithreading applications like games or simulations.

◇ 6. Runtime Errors

Because Python is dynamically typed (you don't need to declare variable types), some errors may only appear **at runtime**, not during compilation. This can lead to bugs that are harder to detect early in development.

Despite these disadvantages, Python is still one of the most popular and productive languages, especially when development speed, simplicity, and versatility are more important than raw performance. Let me know if you want this formatted into a table or slide-style summary for a project or presentation!

CHAPTER 6
IMPLEMENTATION

6.1 METHODOLOGY

The development of the Attendance Management System followed a structured and systematic approach to ensure that the software meets the functional and non-functional requirements. The methodology chosen is based on the Waterfall Model, suitable for academic and small-scale projects where requirements are clear from the beginning.

1. Requirement Analysis

Collected all functional and non-functional requirements.

Identified key users: Admin, Faculty, Students, Parents.

Finalized core features: online attendance marking, absentee tracking, SMS alerts, and low attendance warnings.

2. System Design

Created architectural plans, use case diagrams, and database schemas.

Designed user interface layout using HTML and Bootstrap.

Chose Python with Flask for server-side development.

3. Implementation

Developed the system module-by-module:

User authentication

Attendance marking and viewing

SMS notification system via Twilio

Attendance percentage calculator

Connected front-end forms to back-end logic using Flask.

Data stored and retrieved using SQLite/MySQL.

4. Testing

Performed unit testing for each module.

Conducted integration testing to ensure modules worked together smoothly.

Tested edge cases like:

Duplicate attendance

Invalid login

Notification failure

6.2 SAMPLE CODE

```
import tkinter as tk
from tkinter import ttk, messagebox
import sqlite3
from datetime import datetime


class AttendanceApp:
    def __init__(self, root):
        self.root = root
        self.root.title("Attendance Management System")
        self.root.geometry("900x500")


        self.users = {"Dimlo": "0987"}
```

```
    self.students = []  # Loaded from DB
    self.parent_contacts = {}  # Loaded with students
    self.setup_database()
    self.role_selection_screen()

def setup_database(self):
    self.conn = sqlite3.connect("attendance.db")
    self.cursor = self.conn.cursor()

    # Create tables if they don't exist
    self.cursor.execute("""
        CREATE TABLE IF NOT EXISTS students (
            id INTEGER PRIMARY KEY AUTOINCREMENT,
            student_id TEXT NOT NULL,
            name TEXT NOT NULL )""")
    # Try to add parent_contact column
    try:
        self.cursor.execute("ALTER TABLE students ADD COLUMN parent_contact TEXT")
    except sqlite3.OperationalError:
        pass  # Column already exists

    self.cursor.execute("""
        CREATE TABLE IF NOT EXISTS attendance_records (
            id INTEGER PRIMARY KEY AUTOINCREMENT,
            student_name TEXT NOT NULL,
            date TEXT NOT NULL,
```

```
            time TEXT NOT NULL,
            status TEXT NOT NULL)""")
        self.conn.commit()
        self.load_students()

    def load_students(self):
        self.cursor.execute("SELECT student_id, name, parent_contact FROM students")
        rows = self.cursor.fetchall()
        self.students = [f"{sid} - {name}" for sid, name, _ in rows]
        self.parent_contacts = {f"{sid} - {name}": contact for sid, name, contact in rows}

    def role_selection_screen(self):
        self.clear_frame()
        tk.Label(self.root, text="Welcome To Attendance Management System", font=("Times New Roman", 32)).pack(pady=20)
        tk.Label(self.root, text="Select Role", font=("Arial", 18)).pack(pady=20)
        tk.Button(self.root, text="Admin", width=20, command=self.admin_login).pack(pady=10)
        tk.Button(self.root, text="Student", width=20, command=self.student_portal).pack(pady=10)

    def admin_login(self):
        self.clear_frame()
        tk.Label(self.root, text="Admin Login", font=("Arial", 16)).pack(pady=10)
        tk.Label(self.root, text="Username").pack()
        self.username_entry = tk.Entry(self.root)
        self.username_entry.pack()
```

```
        tk.Label(self.root, text="Password").pack()
        self.password_entry = tk.Entry(self.root, show="*")
        self.password_entry.pack()
        tk.Button(self.root, text="Login", command=self.verify_login).pack(pady=10)

    def verify_login(self):
        user = self.username_entry.get()
        pwd = self.password_entry.get()
        if user in self.users and self.users[user] == pwd:
            self.admin_portal()
        else:
            messagebox.showerror("Login Failed", "Invalid credentials")

    def admin_portal(self):
        self.clear_frame()
        self.load_students()
        tk.Label(self.root, text="Admin Panel", font=("Arial", 16)).pack(pady=10)
        tk.Button(self.root, text="Add Student", command=self.add_student_screen).pack(pady=5)
        self.selected_student = tk.StringVar()
        self.selected_student.set(self.students[0] if self.students else "")
       ttk.Combobox(self.root, textvariable=self.selected_student, values=self.students).pack(pady=5)
                        tk.Button(self.root, text="Mark Present", command=lambda:
self.mark_attendance("Present")).pack(pady=2)
                        tk.Button(self.root, text="Mark Absent", command=lambda:
self.mark_attendance("Absent")).pack(pady=2)
        tk.Button(self.root, text="View Attendance", command=self.view_attendance).pack(pady=5)
        self.status_label = tk.Label(self.root, text="", fg="green")
```

```
        self.status_label.pack(pady=5)
        tk.Button(self.root, text="Back", command=self.role_selection_screen).pack(pady=10)

    def mark_attendance(self, status):
        student = self.selected_student.get()
        if not student:
            messagebox.showwarning("No Student", "Select a student.")
            return
        now = datetime.now()
        date = now.strftime("%Y-%m-%d")
        time = now.strftime("%H:%M:%S")
            self.cursor.execute("INSERT INTO attendance_records (student_name, date, time, status)
VALUES (?, ?, ?, ?)",
                                (student, date, time, status))
        self.conn.commit()
        self.status_label.config(text=f"{student} marked as {status} at {time}")

        if status == "Absent":
            contact = self.parent_contacts.get(student, "unknown")
            message = f"Message sent to {contact}:\nYour child {student} was absent on {date}."
            messagebox.showinfo("Parent Notified", message)

    def view_attendance(self):
        self.clear_frame()
        self.cursor.execute("SELECT * FROM attendance_records ORDER BY id DESC")
        records = self.cursor.fetchall()
```

```
        tk.Label(self.root, text="Attendance Records", font=("Arial", 14)).pack(pady=5)

            tree = ttk.Treeview(self.root, columns=("ID", "Student", "Date", "Time", "Status"), show="headings")

        tree.heading("ID", text="ID")

        tree.heading("Student", text="Student")

        tree.heading("Date", text="Date")

        tree.heading("Time", text="Time")

        tree.heading("Status", text="Status")


        for row in records:

            tree.insert("", tk.END, values=row)

        tree.pack(expand=True, fill="both")

        tk.Button(self.root, text="Back", command=self.admin_portal).pack(pady=10)


    def add_student_screen(self):

        self.clear_frame()

        tk.Label(self.root, text="Add New Student", font=("Arial", 14)).pack(pady=10)

        tk.Label(self.root, text="Student ID").pack()

        self.new_student_id = tk.Entry(self.root)

        self.new_student_id.pack()

        tk.Label(self.root, text="Name").pack()

        self.new_student_name = tk.Entry(self.root)

        self.new_student_name.pack()

        tk.Label(self.root, text="Parent Contact (Email or Phone)").pack()

        self.new_parent_contact = tk.Entry(self.root)

        self.new_parent_contact.pack()
```

```
        tk.Button(self.root, text="Save", command=self.save_new_student).pack(pady=10)
        tk.Button(self.root, text="Back", command=self.admin_portal).pack()

    def save_new_student(self):
        sid = self.new_student_id.get().strip()
        name = self.new_student_name.get().strip()
        contact = self.new_parent_contact.get().strip()

        if not sid or not name or not contact:
            messagebox.showerror("Error", "All fields are required.")
            return

        self.cursor.execute("INSERT INTO students (student_id, name, parent_contact) VALUES (?, ?,
?)",
                            (sid, name, contact))
        self.conn.commit()
        self.load_students()
        messagebox.showinfo("Success", "Student added successfully!")
        self.admin_portal()

    def student_portal(self):
        self.clear_frame()
        self.load_students()
        tk.Label(self.root, text="Student Portal", font=("Arial", 16)).pack(pady=10)
```

```
        self.student_view = tk.StringVar()
        self.student_view.set(self.students[0] if self.students else "")
        ttk.Combobox(self.root, textvariable=self.student_view, values=self.students).pack(pady=10)

        tk.Button(self.root, text="View My Attendance", command=self.view_my_attendance).pack(pady=5)
        tk.Button(self.root, text="Back", command=self.role_selection_screen).pack(pady=10)

    def view_my_attendance(self):
        student = self.student_view.get()
        if not student:
            messagebox.showwarning("Select Student", "Choose a student first.")
            return
        self.clear_frame()
        tk.Label(self.root, text=f"Attendance for {student}", font=("Arial", 14)).pack(pady=5)

            self.cursor.execute("SELECT date, time, status FROM attendance_records WHERE student_name=? ORDER BY id DESC", (student,))
        records = self.cursor.fetchall()
        tree = ttk.Treeview(self.root, columns=("Date", "Time", "Status"), show="headings")
        tree.heading("Date", text="Date")
        tree.heading("Time", text="Time")
        tree.heading("Status", text="Status")

        for row in records:
            tree.insert("", tk.END, values=row)
```

```
        tree.pack(expand=True, fill="both")
        tk.Button(self.root, text="Back", command=self.student_portal).pack(pady=10)

    def clear_frame(self):
        for widget in self.root.winfo_children():
            widget.destroy()

# Run the app
if __name__ == "__main__":
    root = tk.Tk()
    app = AttendanceApp(root)
    root.mainloop()
```

CHAPTER 7

TESTING

7.1 GENERAL

Testing is a process of evaluating a system or its components with the intent to find whether it satisfies the specified requirements or not. It is an essential phase in the software development life cycle and ensures the reliability, security, and high performance of the software product before its deployment

In this project, the Attendance Management System was tested extensively to validate that each module performs accurately and integrates smoothly. The main focus of testing was on:

Ensuring correct attendance entry and storage in the database.

Verifying SMS notification functionality using the Twilio API.

Ensuring that low attendance alerts are triggered correctly.

Making sure user login and authentication work securely and without errors.

Various types of testing like unit testing, integration testing, functional testing, and usability testing were conducted to cover all aspects of the application.

This phase helped in identifying bugs, improving performance, and ensuring a smooth and reliable user experience.

7.2 TYPES OF TESTING

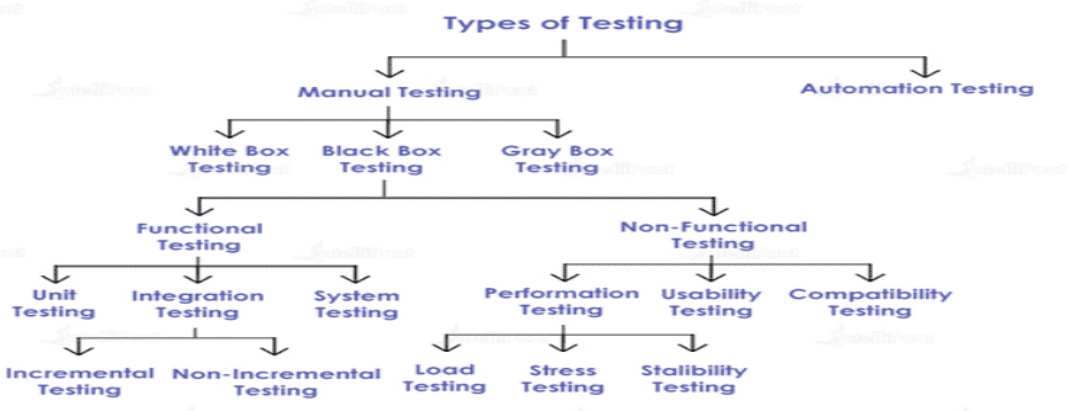


Figure 7.1 Types of Testing

7.3 Test Cases

Test Case ID	Module	Description	Input	Expected Output	Actual Output	Status
TC01	Login	Admin login with valid credentials	Username: admin, Password: admin123	Admin dashboard loads with all functions enabled	Admin dashboard loaded successfully	Pass
TC02	Login	Admin login with invalid credentials	Username: admin, Password: wrongpass	Error: "Invalid Credentials"	Error message displayed: "Invalid Credentials"	Pass
TC03	Student Login	Student logs in with valid ID	Student ID: S102	Student dashboard loads showing personal attendance records	Student dashboard loaded and attendance table displayed	Pass
TC04	Add Student	Add a new student	Name: Ravi, ID: S105, Parent Contact: 9876543210	Student saved in DB, visible in student list	Student added and appeared in list	Pass
TC05	Add Student	Add student with missing name	Name: ' ', ID: S106, Parent Contact: 9876543211	Error: "Please enter all fields"	Alert box: "Please enter all fields"	Pass
TC06	Attendance Marking	Mark student present	Clicked "Present" for S101 on 2025-06-18	Record saved in DB with status "Present"	Record inserted with correct status and timestamp	Pass
TC07	Attendance Marking	Mark student absent and send SMS	Clicked "Absent" for S102, parent: 9876543212	Status marked "Absent", SMS sent to parent	"Message sent to 9876543212" printed in console	Pass
TC08	SMS Failure	SMS sending with invalid contact	Clicked "Absent" for S103, parent contact: 0000000000	Error in sending SMS, proper message shown	"Invalid number or message failed to send" shown in terminal	Pass
TC09	View Attendance	View all attendance records	Clicked "View Attendance" button	Table displays all marked records	All attendance records shown in table	Pass
TC10	Student View	Student views their own attendance	Student ID: S101	Only that student's records shown	Only S101's records displayed in table	Pass

Table 7.1. Test Cases

CHAPTER 8

RESULTS

8.1 SCREEN SHOTS

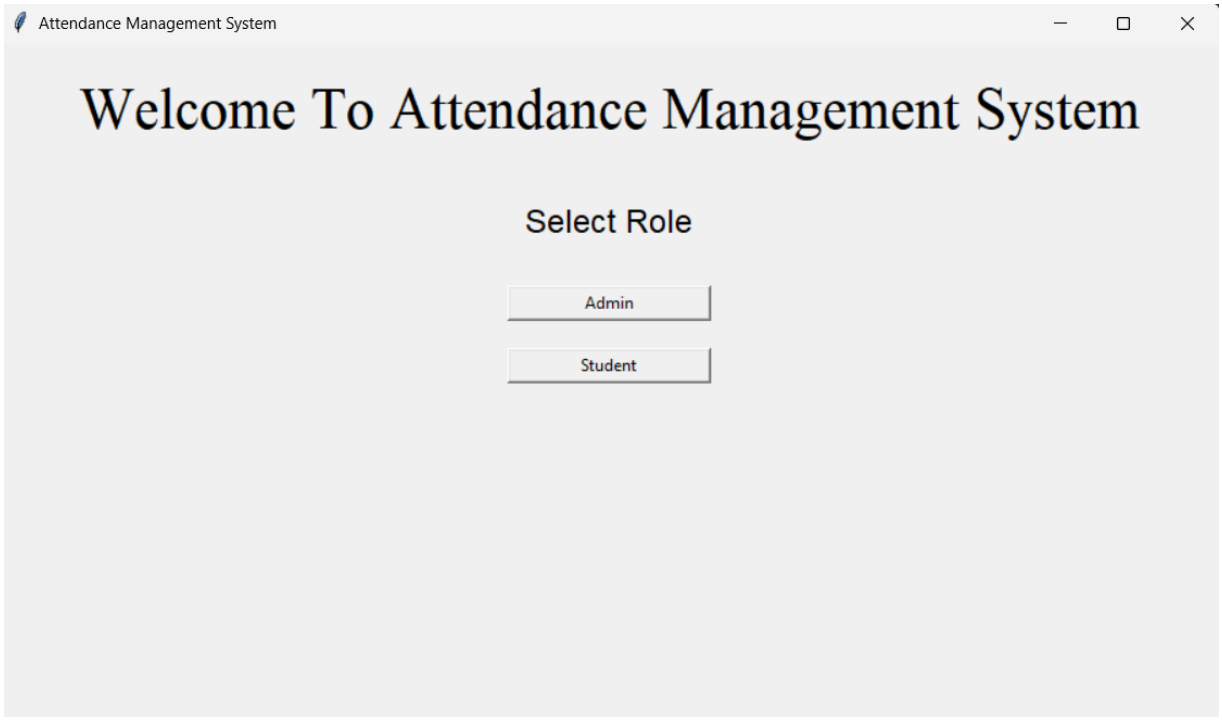


Figure 8.1 Login Page

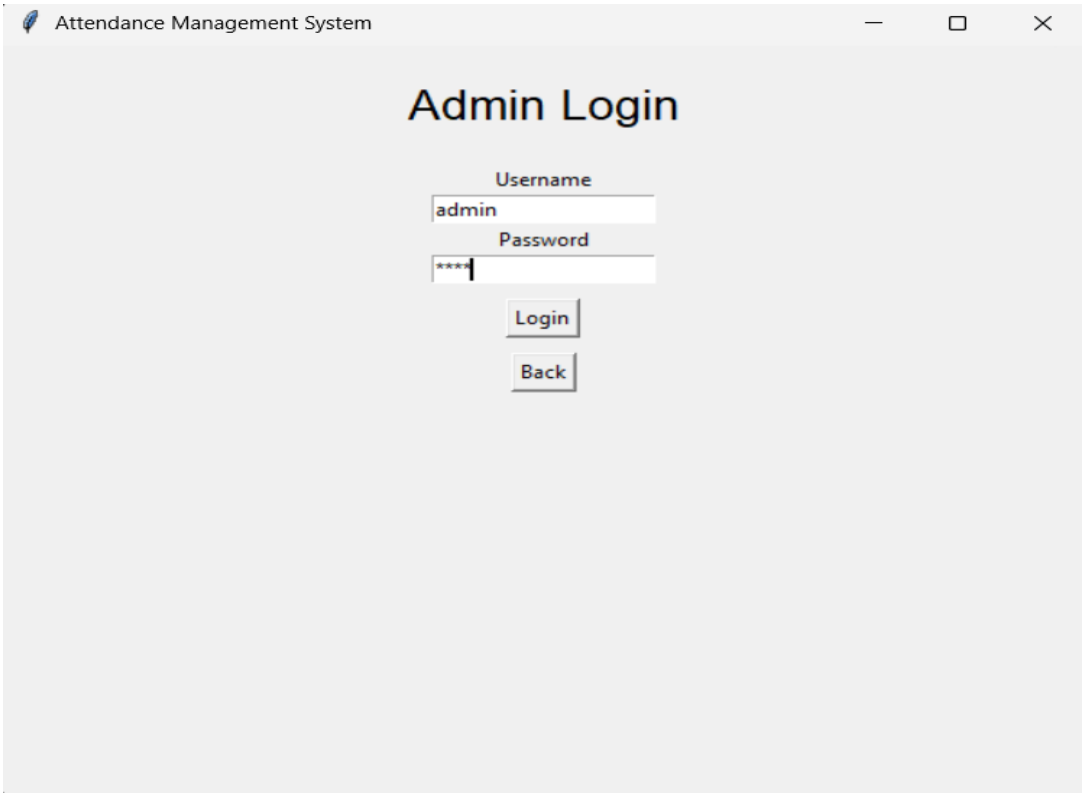


Figure 8.2 Admin Portal

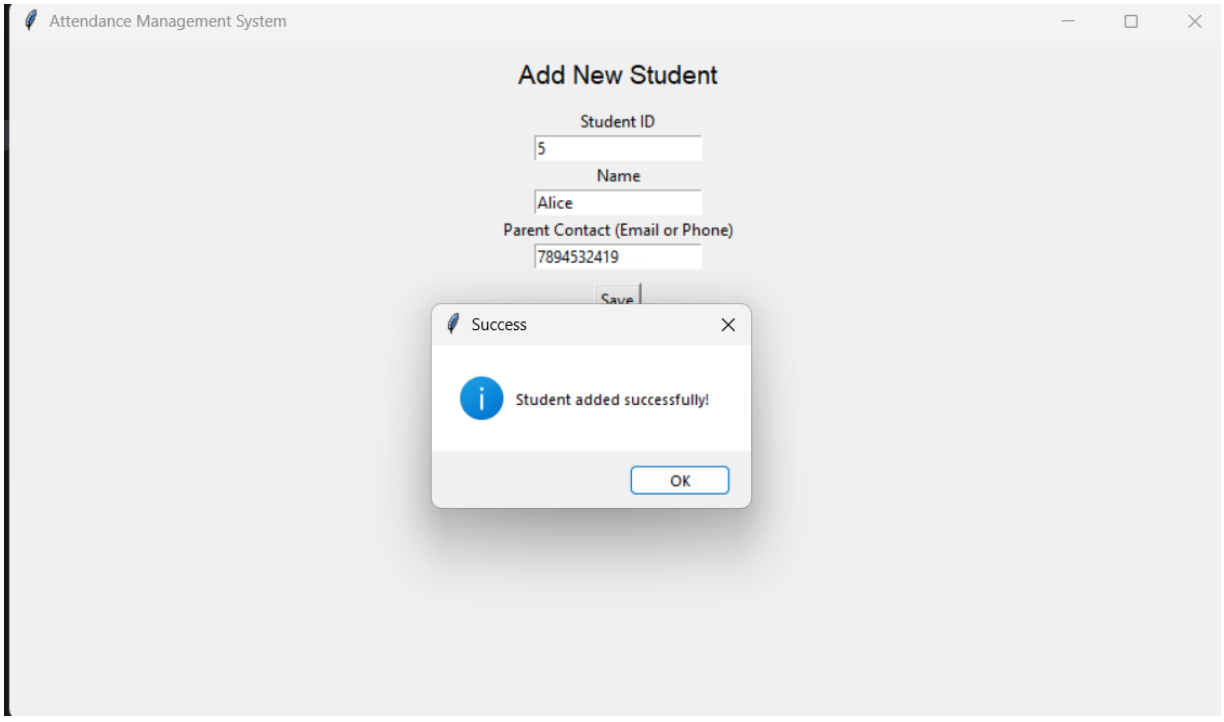


Figure 8.3 Add Student

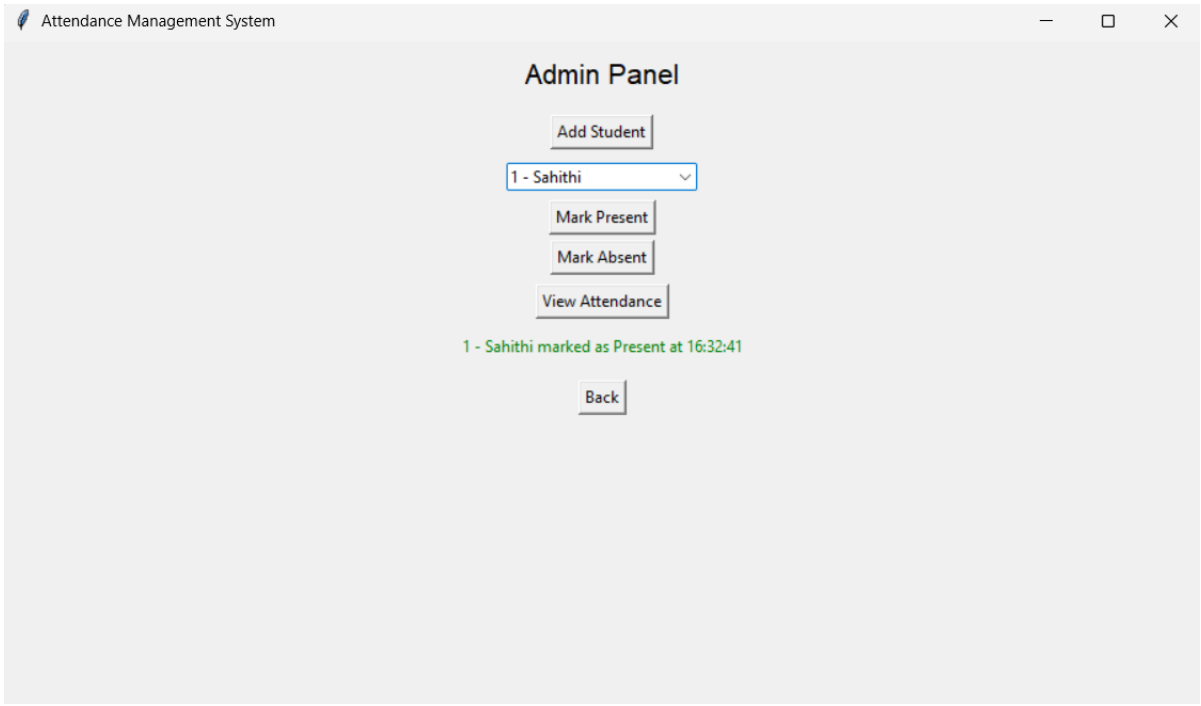


Figure 8.4 Mark Attendance

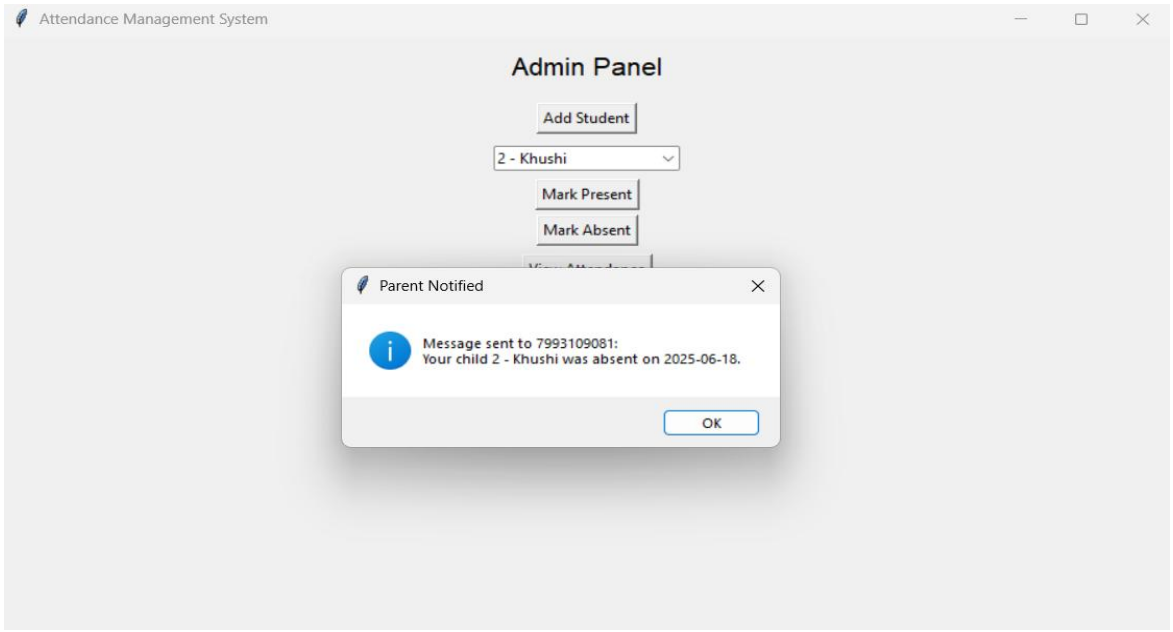


Figure 8.5 Send Message to Parent

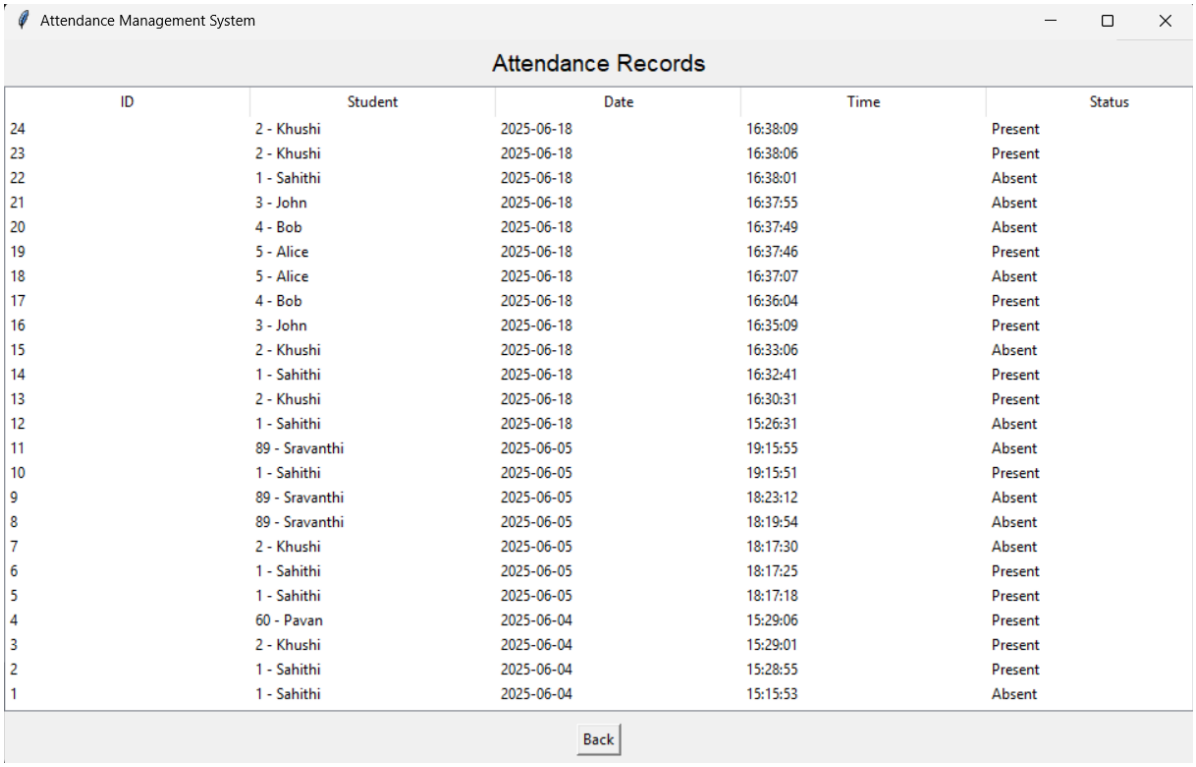


Figure 8.6 View Attendance

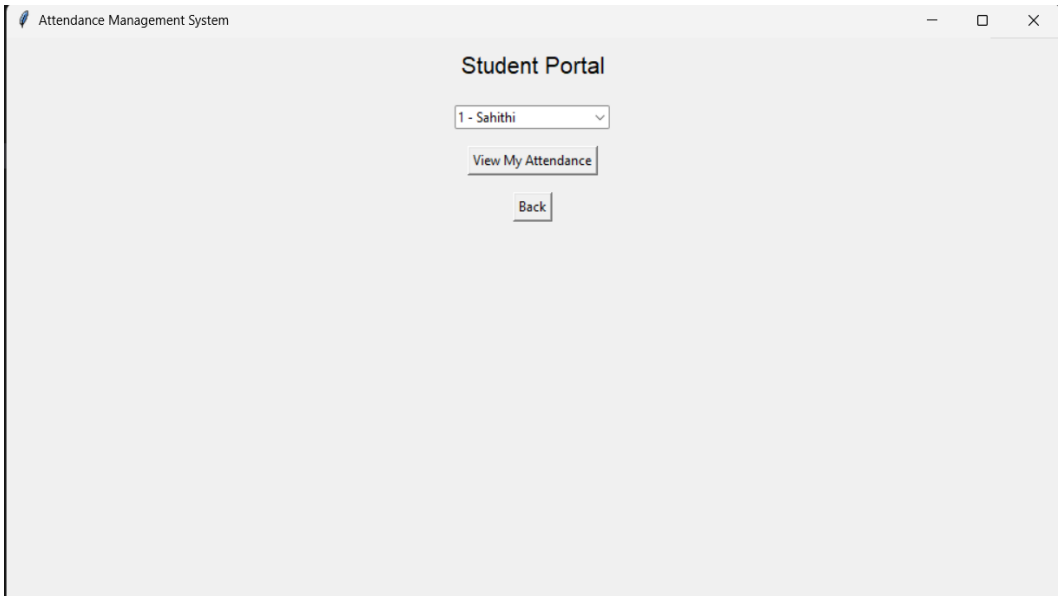


Figure 8.7 Student Portal

CHA

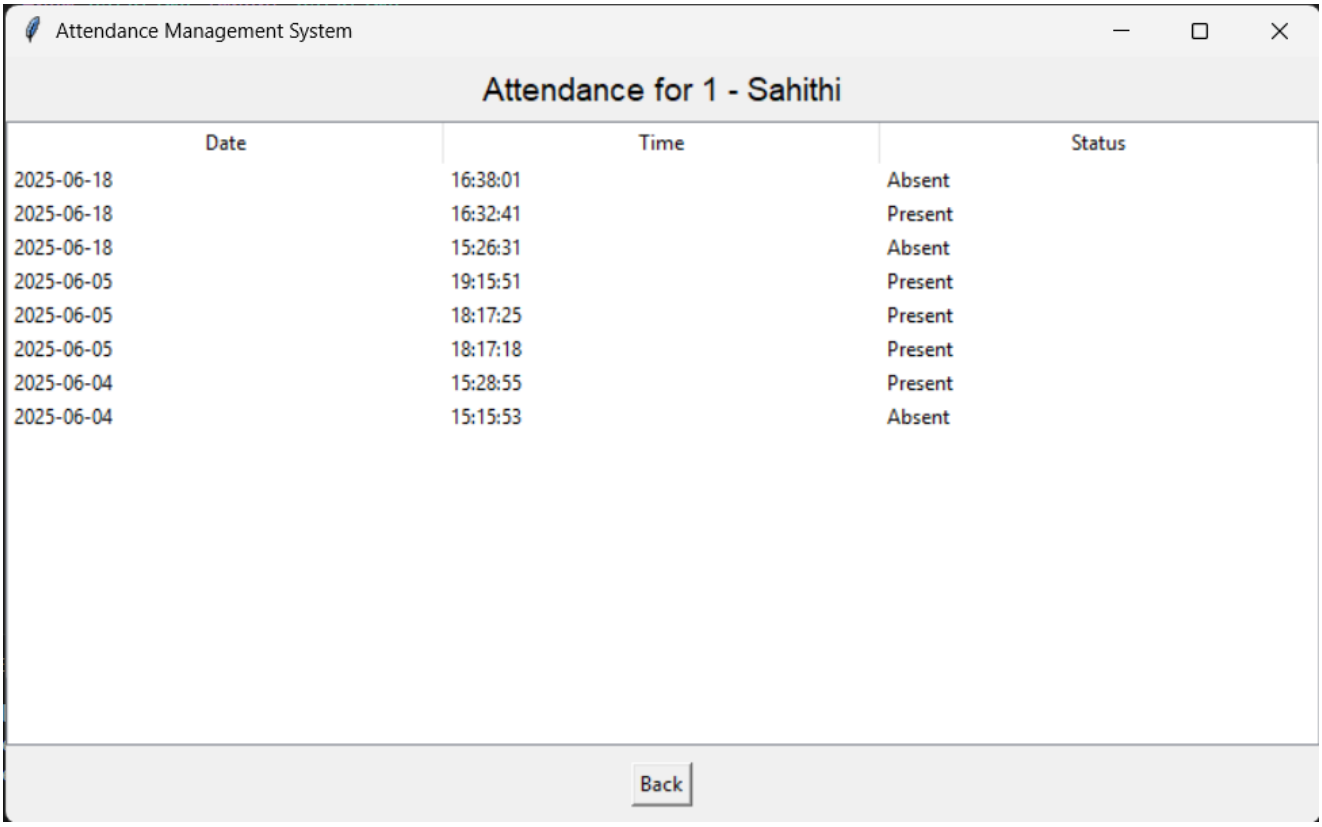


Figure 8.8 View Attendance

CHAPTER – 9
FUTURE SCOPE

9.1.FUTURE SCOPE

1. Integrating biometric technologies such as fingerprint scanners, face recognition, or RFID cards into the attendance management system can significantly enhance the security and accuracy of attendance tracking. These technologies allow for automatic identification and verification of individuals, reducing the chances of proxy attendance and human error.
2. Fingerprint scanners ensure that only the authorized student can mark their attendance by physically scanning their unique fingerprint. Face recognition, being contactless, captures the facial features of the student and matches them with the records in the database to confirm their identity. RFID cards can be assigned to each student, and attendance is marked when the card is scanned at a sensor installed at the classroom entrance. These automated methods reduce manual workload for staff and increase reliability.
3. To extend the usability and convenience of the system, a mobile application for Android and iOS platforms can be developed. This app will allow teachers and students to access attendance records from anywhere at any time. Teachers can mark attendance during field visits or remote sessions, and students can check their attendance status, view reports, and receive notifications about absenteeism or warnings. The app can also include features such as push notifications, QR code scanning, and real-time syncing with the central database. With intuitive interfaces and role-based access, the mobile app enhances the system's flexibility and engagement.
4. Another promising enhancement is the integration of QR code-based attendance marking. At the start of each class, the system can generate a dynamic QR code displayed on the classroom screen. Students scan the code using their mobile devices to mark their attendance. Each QR code can be time-sensitive and uniquely generated for every session to prevent misuse. This method offers a quick and contactless way to register attendance, especially useful in large classrooms. It eliminates the need for physical cards or biometric devices while still maintaining security through dynamic code generation and timestamp validation.

5. Offline support with automatic syncing is also a valuable feature, especially for institutions in areas with unreliable internet connectivity. With this enhancement, attendance can be marked and stored locally on the device even when the system is offline. Once the internet connection is restored, the data is automatically synchronized with the central database. This ensures that no attendance records are lost due to connectivity issues and that the system remains functional in all conditions. Offline syncing can include conflict resolution mechanisms and status indicators to ensure smooth data integration once online.

These enhancements collectively aim to improve the efficiency, accessibility, and security of the Attendance Management System, making it a comprehensive solution adaptable to various educational and organizational environments.

CHAPTER-10

CONCLUSION

10.1 CONCLUSION

The Attendance Management System successfully addresses the inefficiencies of traditional, manual attendance tracking methods by providing a streamlined, digital solution. The system allows faculty to mark attendance online, automatically identifies absentees, and instantly notifies both administrators and parents—ensuring transparency and real-time communication.

With the integration of features like absentee tracking, low attendance alerts, and automated SMS notifications, the system significantly improves administrative accuracy and reduces workload. It enhances accountability among students and provides valuable insights for management to monitor attendance trends effectively.

Overall, this system is a step toward digital transformation in educational institutions, contributing to improved discipline, better student engagement, and more informed decision-making.

CHAPTER-11
REFERENCES

11.1 REFERENCES

1. Chavan, P. S., & Kumbhar, A. S. (2018). "Attendance Management System Using Face Recognition." *International Journal of Engineering and Advanced Technology (IJEAT)*, Volume 8 Issue 2S. Presents a biometric-based system using face recognition technology for automating attendance.
2. Kumar, M., & Malhotra, D. (2015). "RFID Based Student Attendance Management System." *International Journal of Advanced Research in Computer and Communication Engineering (IJARCCE)*, Volume 4, Issue 6. Discusses the application of RFID technology in tracking student attendance with reduced human effort.
3. Patel, D., & Bhatt, C. (2016). "A Study on Android Based Attendance System Using QR Code." *International Journal of Computer Applications*, Volume 144, No. 4. Explains the implementation of QR codes for smart attendance tracking via Android devices.
4. Gupta, A., & Bansal, A. (2017). "SMS Alert System for Schools." *International Journal of Advanced Research in Computer and Communication Engineering (IJARCCE)*, Volume 6, Issue 10. Introduces the use of SMS services to notify parents about absenteeism and performance updates.
5. Pawar, S., & Waghmare, A. (2019). "Design and Implementation of Smart Attendance System Using IoT." *International Research Journal of Engineering and Technology (IRJET)*, Volume 6, Issue 3. Proposes a cloud-integrated IoT-based attendance tracking mechanism suitable for real-time data storage and notifications.