

ATTENDANCE MANAGEMENT SYSTEM

A Project Report

Submitted in partial fulfilment of the requirements for the award of the Degree of
BACHELOR OF SCIENCE (INFORMATION TECHNOLOGY)

by

Kunal Mahendra Giri

1068679

and

Priya Rajkumar Maurya

1068864

Under the esteemed guidance of

Mr. Farhan Shaikh

&

Mrs. Supritha Bhandary

Lecturers



DEPARTMENT OF INFORMATION TECHNOLOGY

M. L. DAHANUKAR COLLEGE OF COMMERCE

(Affiliated to University of Mumbai)

MUMBAI, 400057

MAHARASHTRA

2023-2024



Parle Tilak Vidyalyaya Association's

M. L. DAHANUKAR COLLEGE OF COMMERCE

(DEGREE COLLEGE) VILE PARLE (E), MUMBAI 400

057 DEPARTMENT OF INFORMATION TECHNOLOGY

CERTIFICATE

This is to certify that the report of project entitled

undertaken at M.L. Dahanukar College of Commerce by

_____ Seat no. _____

and _____ Seat no. _____ in

partial fulfillment of B.Sc.(I.T) Degree (Semester. VI)

Examination has not been submitted for any other examination

and does not form part of any other course undergone by the

candidate. It is further certified that they have completed all

required phases of the project.

Internal Examiner

External Examiner

Coordinator

College Seal

Abstract

The Attendance Management System (AMS) described in the abstract is a solution incorporating website to automate attendance tracking in academic institutions and corporate environments. It ensures accuracy and efficiency by utilizing facial recognition for secure attendance recording. Additionally, a user-friendly mobile app allows students, employees, and faculty members access to view their real-time attendance status. The system also generates detailed reports, enabling institutions and organizations to monitor attendance patterns and trends effectively. By enhancing accountability and reducing administrative workload, AMS proves invaluable in optimizing attendance management processes.

ACKNOWLEDGEMENT

It is indeed with a great pleasure and immense sense of gratitude we acknowledge the help of our principal Prof Dr. Kanchan Fulmali for the facilities provided to accomplish this project. We are extremely thankful to our coordinator Smt. Archana Talekar for her constant support and inspiration in completing this project. The project could not be completed without the support of Mr. Farhan Shaikh and Mrs. Supritha Bhandary. Their support has been throughout the process and helped us to clear up even the smallest doubts related to documentation and helped us in the development of the project by suggesting new features to be added that can improve my project. Finally, we express our sincere thanks to all the IT faculties, non-teaching staff, and all our friends who directly or indirectly helped us in the completion of the project.

DECLARATION

We hereby declare that the project entitled, “**Attendance Management System**” done at “**M. L. DAHANUKAR COLLEGE OF COMMERCE**” has not been in any case duplicated to submit to any other university for the award of any degree. To the best of my knowledge other than us, no one has submitted to any other university.

The project is done in partial fulfillment of the requirements for the award of degree of **BACHELOR OF SCIENCE (INFORMATION TECHNOLOGY)** to be submitted as final semester project as part of our curriculum.

Kunal Mahendra Giri

Priya Rajkumar Maurya

TABLE OF CONTENTS

Chapter 1 Introduction.....	1
1.1 Background	1
1.2 Objectives	2
1.3 Purpose, Scope and Applicability	2
1.3.1 Purpose	2
1.3.2 Scope.....	3
1.3.3 Applicability	3
1.4 Achievements.....	4
1.5 Organization of Report	4
Chapter 2 Survey of Technologies.....	5
2.1 Available Technologies.....	5
2.2 List of Technologies	6
2.3 Comparative Study.....	8
2.4 Selected Technology.....	9
Chapter 3 Requirements and Analysis	10
3.1 Problem Definition.....	10
3.2 Requirement Specification	11
3.3 Planning and Scheduling.....	12
3.4 Software and Hardware Requirements	13
3.5 Preliminary Product Description	14
3.6 Conceptual Model.....	14
Chapter 4 System Design.....	24
4.1 Basic Modules.....	24
4.2 Data Design.....	25
4.2.1 Schema Design	25
4.2.2 Data Integrity and Constraints	25

4.3 Procedural Design.....	27
4.3.1 Logic Diagrams	27
4.3.2 Data Structures.....	28
4.3.3 Algorithm Design	29
4.4 User Interface Design	30
4.5 Security Issues.....	31
4.6 Test Case Design	32
Chapter 5 Implementation Testing	33
5.1 Implementation Approach	33
5.2 Coding Details and Code Efficiency	35
5.3 Testing Approaches	45
5.4 Modification and Implementation.....	46
5.5 Test Case.....	46
Chapter 6 Results and Discussion	50
6.1 Test Case Reports	50
6.2 Screenshots	56
Chapter 7 Conclusion	61
7.1 Conclusion	61
7.2 Limitations of the System.....	61
7.3 Future Scope.....	61
SUMMARY	63
GLOSSARY.....	64
REFERENCE.....	65

List of Tables

Table 4.2.2.1 login_data.....	26
Table 4.2.2.2 details_data.....	26
Table 4.2.2.3 attendance_data.....	26
Table 4.2.2.4 role_data.....	26
Table 4.6.1 Test Case 1	32
Table 4.6.2 Test Case 2	32
Table 5.5.1 Test Case for Admin Access and Operations	47
Table 5.5.2 Test Case for User Access and Operations.....	48
Table 5.5.3 Test Case for Staff Access and Operations.....	48
Table 5.5.4 Test Case for Face Recognition.....	49
Table 5.5.5 Test Case for Authorization Functions	49
Table 6.1.1 Test Case Report for Admin Access and Operations	51
Table 6.1.2 Test Case Report for Admin Access and Operations	52
Table 6.1.3 Test Case Report for Staff Access and Operations.....	53
Table 6.1.4 Test Case Report for Face Recognition.....	54
Table 6.1.5 Test Case Report for Authorization Functions	55

List of Figures

Fig 3.3.1 Gantt Chart Representation	12
Fig 3.3.2 Pert Chart.....	13
Fig 3.6.1 Use Case Diagram	16
Fig 3.6.2.1 Data Flow Diagram (Context Level).....	18
Fig 3.6.2.2 Data Flow Diagram for User (First Level)	18
Fig 3.6.2.3 Data Flow Diagram for Staff (First Level)	19
Fig 3.6.2.4 Data Flow Diagram for Admin (First Level).....	19
Fig 3.6.3.1 Activity Diagram for Staff.....	21
Fig 3.6.3.2 Activity Diagram for Admin.....	21
Fig 3.6.3.3 Activity Diagram for User	22
Fig 3.6.4 Entity Relationship Diagram	23
Fig 4.3.1.1 Logic Diagram.....	27
Fig 4.3.2.1 Data Structures Diagram	28
Fig 4.3.3.1 Algorithm Design	29
Fig 4.4.1 Login Page.....	30
Fig 4.4.2 Home Page.....	30
Fig 4.4.3 Attendance Page	31
Fig 6.2.1 Login Page.....	56
Fig 6.2.2.1 Admin Page.....	56
Fig 6.2.2.2 Admin Page adding Staff.....	57
Fig 6.2.3.1 Staff Page.....	57
Fig 6.2.3.2 Staff Page adding user	58
Fig 6.2.3.3 Staff Page (View Attendance).....	58
Fig 6.2.3.3 Staff Page (Add Attendance)	59
Fig 6.2.4.1 User Page (Dashboard).....	59
Fig 6.2.4.2 User Page (View Attendance).....	60
Fig 6.2.3.1 Face Recognition Page	60

Chapter 1

Introduction

1.1 Background

Face recognition technology for attendance management systems has evolved over time it aims to replace traditional manual attendance recording methods, such as paper-based sheets, with a more efficient and accurate system.

In terms of existing work in the area, facial recognition technology has been extensively researched and developed over the years. Due to the introduction of Eigenfaces, it improved accuracy, but lighting and expression variations remained problematic. Local feature analysis methods, such as LBP, emerged in the late 1990s to address these issues.

Several academic studies, research papers, and commercial systems have focused on facial recognition-based attendance management systems. These systems have demonstrated significant potential in improving attendance accuracy, efficiency, and security. Many existing projects utilize advanced facial recognition algorithms to identify on unique facial features. These algorithms extract facial landmarks, such as the position of eyes, nose, and mouth, and analyse them to create a unique facial template for each individual. This template is then compared with the stored templates in the database to determine the identity of the person.

Overall, facial attendance management systems have laid the foundation for the development of automated and accurate attendance tracking solutions. The research and implementation of facial recognition algorithms, machine learning techniques, and anti-spoofing measures have contributed to creating robust systems that offer improved efficiency and reliability compared to traditional attendance management methods.

1.2 Objectives

- Automate attendance tracking process.
- Improve accuracy and efficiency of attendance management.
- Provide real-time monitoring and reporting capabilities.
- Scalable and flexible to accommodate different organizational needs.
- User-friendly interface for administrators and users.
- Cost-effective and Time-efficient solutions as compared to traditional methods.

1.3 Purpose, Scope and Applicability

1.3.1 Purpose

Why this project is being done?

The purpose of implementing a face recognition attendance management system is to automate and improve the accuracy, security, and efficiency of attendance tracking processes. It aims to enhance user experience, enable real-time monitoring and reporting, integrate with existing systems, and provide a cost-effective solution. The system addresses the need for reliable attendance management, reducing errors, saving time and resources, and enhancing overall operational effectiveness.

How should the project improve the system?

To improve a face recognition attendance management system, the project should prioritize accuracy, robustness, speed, user-friendliness, real-time monitoring, integration with existing systems, scalability, data security, error handling, and continuous improvement. By focusing on these areas, the system can deliver accurate and efficient attendance tracking, seamless integration, enhanced user experience, and strong data protection.

1.3.2 Scope

The attendance management system automates attendance tracking using facial recognition technology. It accurately identifies individuals, enhances security, integrates with existing systems, and offers scalability and reporting capabilities. It streamlines the process, improves efficiency, reduces fraud, and provides valuable insights for decision-making.

Limitations:

The proposed Attendance Management System may have certain limitations, such as:

- **Accuracy:** The system's performance may be affected by factors like lighting conditions, and pose variations, leading to occasional false positive or negative identifications.
- **Hardware Requirements:** The system's accuracy and efficiency may depend on the quality of cameras or webcams used for image capture.
- **Scalability:** The system's performance may be affected when handling a large number of users simultaneously.

Assumption:

- Admin should have access to a PC.
- Users should have a mobile phone.
- Internet should be available.

1.3.3 Applicability

The facial attendance management system benefits the computer world and people by enhancing efficiency, accuracy, security, and data analytics. It automates attendance tracking, saving time and effort. It ensures accurate identification and reduces the risk of errors and fraud attendance. The system integrates with existing systems, and maintaining data integrity. It provides valuable insights for decision-making and offers a user-friendly experience. Overall, it represents a technological advancement in attendance management.

1.4 Achievements

The individuals working on a face recognition attendance management project will gain knowledge in face recognition technology, machine learning, image processing, data handling, software development, user interface design, system integration, privacy and ethics, project management, and problem-solving. These skills are valuable in computer vision, AI, and software development domains.

In this project, goals related to automation, efficiency, accuracy, and security are fully achieved. However, goals of integration, data analytics, and user experience are partially achieved and may vary depending on system implementation and customization.

1.5 Organization of Report

The Face Recognition Attendance Management System automates attendance with accuracy and real-time monitoring and enhancing management.

Chapter 2: An introduction about the language and framework used in the making of the project and comparison of currently present technologies.

Chapter 3: Conceptual models will be displayed about the product.

Chapter 4: Design layout of the project and information about user interface and experience.

Chapter 5: Working of the project.

Chapter 6: Explaining Conclusions of Future work of the project.

Chapter 7: References.

Chapter 2

Survey of Technologies

2.1 Available Technologies

The frequent use of technology has brought about significant changes. We can develop systems using a variety of technologies and software. Web services and different frameworks, etc. Have made everything very simple, from designing to developing and creating components.

The Attendance Management System provides users with a digital approach towards Recording Attendance. It is a website that will provide users to update and fetch attendance records.

You can create apps for Android phones, tablets, Android Wear, Android TV, and Android Auto using Android Studio, which offers a single development environment. You can break your project into functional pieces that you can separately create, test, and debug using structured code modules.

There are many different programming languages, but the most common ones used in web development are HTML and CSS. CSS controls how a website looks and is typically used in collaboration with HTML. CSS describes how elements should be rendered on screen. CSS will be used in this website to design different pages. HTML only organizes site text into blocks.

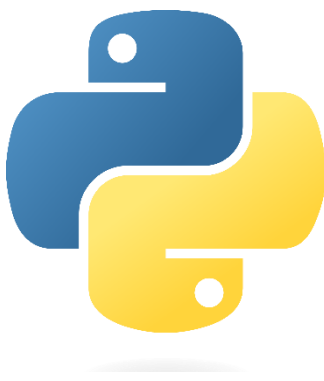
JavaScript is a scripting language used to develop web pages. It allows users to create a dynamic and interactive web page to interact with visitors and execute complex actions. It also enables users to load content into a document without reloading the entire page.

2.2 List of Technologies



1) Java:

- General-purpose, object-oriented language known for platform independence.
- Widely used in enterprise applications, Android app development, and server-side programming.
- Executes code on a Java Virtual Machine (JVM).



2) Python:

- High-level, easy-to-read language with a focus on simplicity and readability.
- Suitable for web development, data analysis, artificial intelligence, and scripting.
- Offers a large standard library and numerous third-party packages.



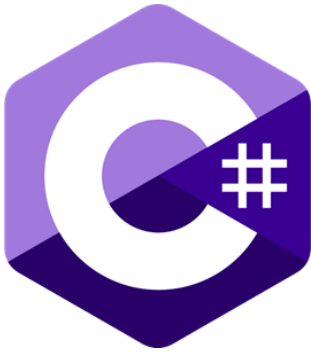
3) C++:

- Versatile, high-performance language with low-level and object-oriented capabilities.
- Commonly used for system-level programming, game development, and performance-critical applications.
- Provides direct memory manipulation and efficient resource management.



4) JavaScript:

- Lightweight, interpreted language used for web development.
- Runs in browsers and enables interactive content on websites.
- With Node.js, JavaScript can be used for server-side development.



5) C# (C-Sharp):

- Developed by Microsoft, C# is an object-oriented language for building Windows applications and games (Unity).
- Part of the .NET framework, suitable for web, desktop, and mobile development.



6) Swift:

- Developed by Apple for iOS, macOS app development.
- Offers safety, speed, and modern syntax, making it easier to write robust and secure code.



7) Kotlin:

- Runs on the Java Virtual Machine (JVM) and is officially supported for Android app development.
- Concise, safe, and interoperable with Java, making it an attractive language for Android development.



8) PHP:

- Primarily used for web development, embedded in HTML to create dynamic websites.
- Provides many frameworks like Laravel and WordPress for building web applications and content management systems.

HTML



9) HTML

- HTML is a markup language for webpages.
- Elements are enclosed in angle brackets (<>) with opening and closing tags.
- It forms the foundation of web development along with CSS and JavaScript.

2.3 Comparative Study

Technology	Features	Advantages	Disadvantages
JAVA	<ul style="list-style-type: none"> • Object Oriented • Platform Independent • Secure 	<ul style="list-style-type: none"> • High security • Speed • Powerful 	<ul style="list-style-type: none"> • Not debugging attractive look • Complex code
Python	<ul style="list-style-type: none"> • Robust Standard Library • Portable • Extensible 	<ul style="list-style-type: none"> • Improved Productivity • Interpreted Language • Dynamically Typed 	<ul style="list-style-type: none"> • Not Memory Efficient • Weak in Mobile Computing • Runtime Errors
C++	<ul style="list-style-type: none"> • Object Oriented Programming • Machine Independent. • Compiler-Based 	<ul style="list-style-type: none"> • Portability • Low-level Manipulation • Memory Management 	<ul style="list-style-type: none"> • Use of Pointers • Security Issue • Absence of Garbage Collector, Built-in Thread
JavaScript	<ul style="list-style-type: none"> • Scripting Language • Interpreter Based • Event Handling. 	<ul style="list-style-type: none"> • Portability • Rich Interface • Versatile 	<ul style="list-style-type: none"> • Debugging facilities need improvement. • Client-side Security.
C#(C-Sharp)	<ul style="list-style-type: none"> • Boolean Condition • Automatic garbage collection • Assembly Versioning 	<ul style="list-style-type: none"> • Portability • Concept of oops • Automatic garbage collection 	<ul style="list-style-type: none"> • Less flexible • Low level concurrency • Poor cross platform UI
Swift	<ul style="list-style-type: none"> • iOS-oriented • Safety • High-performance 	<ul style="list-style-type: none"> • Performance • Interoperability • IOs-Centric 	<ul style="list-style-type: none"> • Learning Curve • Limited Legacy Support • Doesn't have Android Development
Kotlin	<ul style="list-style-type: none"> • Functional and object oriented • Modern Features 	<ul style="list-style-type: none"> • Highly compatible • Simple syntax • Excellence tooling support 	<ul style="list-style-type: none"> • Small Community • Limited resources for learning
PHP	<ul style="list-style-type: none"> • Performance 	<ul style="list-style-type: none"> • Free cost 	<ul style="list-style-type: none"> • Security Issue

	<ul style="list-style-type: none"> • Familiarity • Platform independent 	<ul style="list-style-type: none"> • Extremely Flexible • User Friendly 	<ul style="list-style-type: none"> • Not Secure • Poor error handling method
HTML	<ul style="list-style-type: none"> • User friendly • Semantic structure • Flexible 	<ul style="list-style-type: none"> • Browser friendly • Static language • Free to use 	<ul style="list-style-type: none"> • Limited security • Dependency issue

Table 2.3.1 Comparative Study

2.4 Selected Technology

- Python
- HTML/CSS
- MySQL

Selected Technologies and Why?

- **Python:** Python's simplicity, extensive library support, and strong community make it ideal for face recognition and scanning algorithms. The rich ecosystem, along with OpenCV and ML integration, enables seamless development of sophisticated systems.
- **HTML/CSS:** HTML forms the UI foundation, enabling backend interaction in attendance systems. It ensures cross-device compatible attendance display. Integration with other web tech boosts efficiency. CSS enhances UI by styling, ensuring consistency, responsive layouts, and user appeal in attendance management.
- **MySQL:** MySQL is vital in attendance management systems for its relational database capabilities, data integrity, and efficient data retrieval. Its scalability ensures reliable storage of attendance-related data, maintaining accurate records for a smooth and effective system.

Chapter 3

Requirements and Analysis

3.1 Problem Definition

The conventional methods of manual attendance tracking in various environments, such as educational institutions, workplaces, and events, have proven to be inefficient, error-prone, and lacking real-time insights. These outdated methods involve time-consuming processes that often result in inaccurate records and difficulties in data management. Additionally, traditional attendance systems do not provide a seamless and secure way to verify attendance and monitor it.

- **Inefficiency and Time-Consuming Processes:**
Traditional methods rely on physically checking attendance, which can be slow and inefficient, especially in large groups.
Taking attendance manually requires significant time and effort, leading to delays and wasted productivity.
- **Human Error and Inaccurate Records:**
Manual data entry and recording are prone to human error, leading to incorrect attendance records.
Illegible handwriting, misinterpretation, or oversight can result in inaccurate attendance tracking.
- **Scalability Challenges:**
As the size of the group or organization grows, manual attendance tracking becomes even more cumbersome and error prone.
Scaling up the process requires more resources, exacerbating the inefficiencies.

- **Difficulties in Data Management:**

Storing and managing paper-based attendance records is cumbersome and can lead to data loss or misplacement.

Retrieving historical attendance information becomes challenging, making it harder to analyse trends and patterns.

3.2 Requirement Specification

The proposed Attendance Management System aims to replace traditional manual attendance tracking methods in educational institutions, workplaces, and events. The system will provide a modern, efficient, and reliable solution for tracking attendance, offering real-time insights, reducing errors, and simplifying data management.

- **Efficiency and Real-Time Insights:**

The system should provide real-time attendance tracking and reporting to enable instant insights and timely decision-making.

It should be capable of handling a high volume of attendees efficiently, even in large group settings.

- **Accuracy and Data Integrity:**

The system must minimize human errors through automated data entry mechanisms and validation checks.

Data integrity should be ensured through secure and tamper-proof storage of attendance records.

- **Scalability:**

The system should be scalable, capable of accommodating varying group sizes without compromising performance or accuracy.

It should support easy expansion as the organization or event grows.

- Data Management and Accessibility:

Attendance data should be stored in a centralized, cloud-based database, allowing easy access, retrieval, and management.

User-friendly interfaces should enable authorized personnel to quickly view and manage attendance records.

3.3 Planning and Scheduling

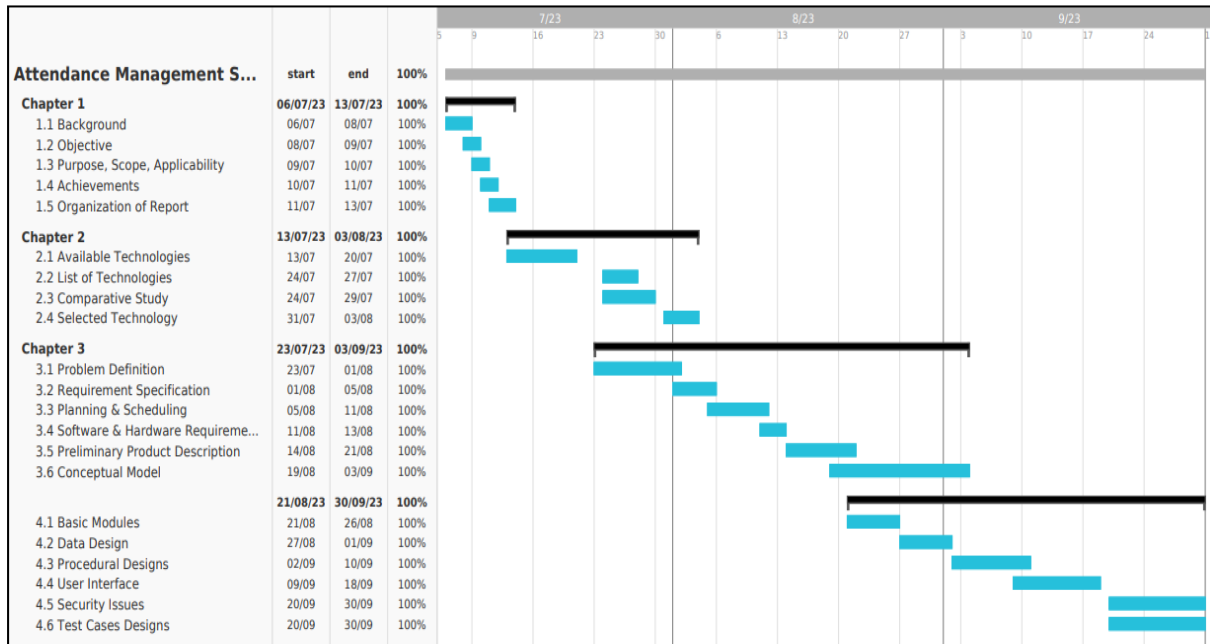


Fig 3.3.1 Gantt Chart Representation

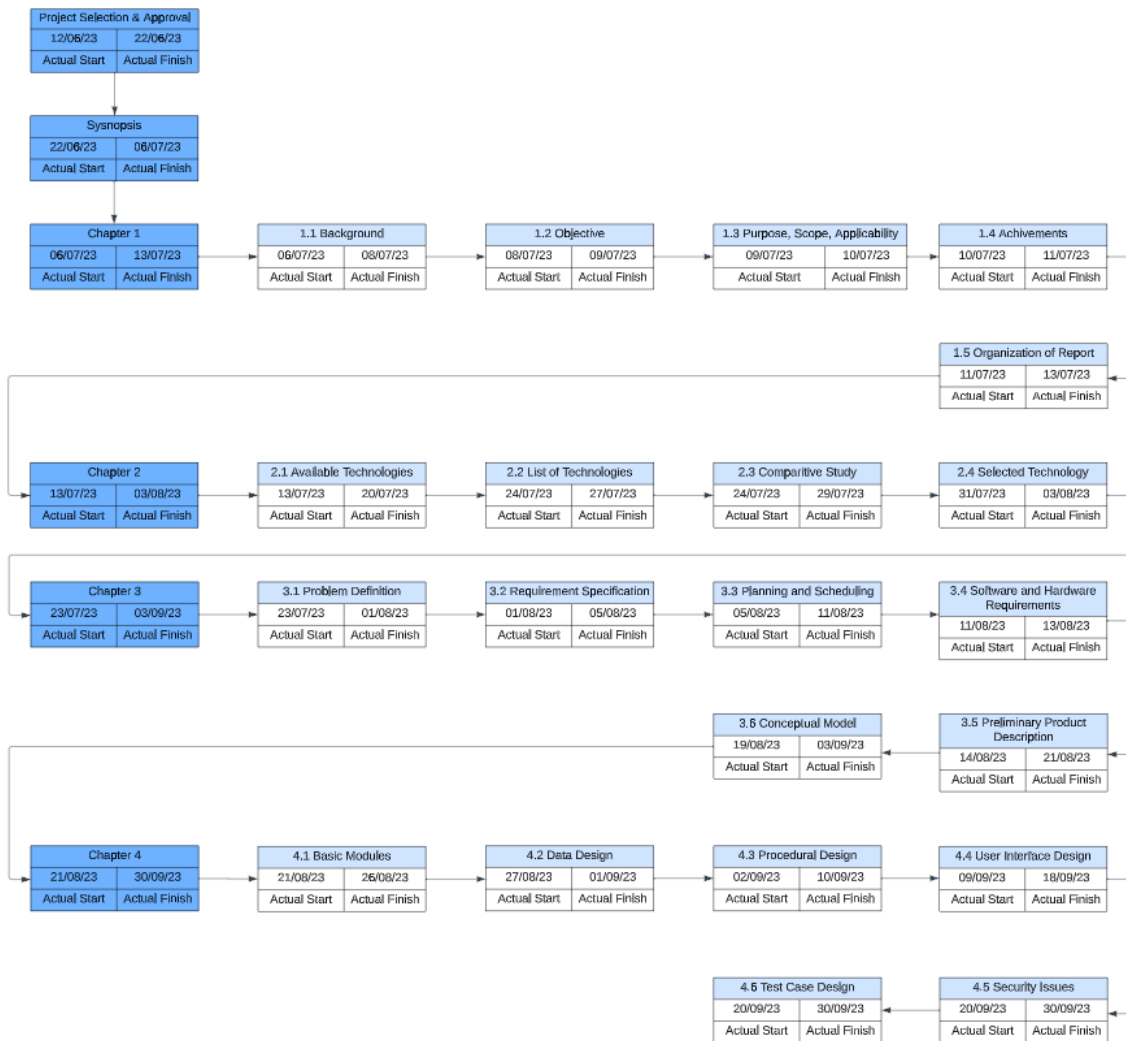


Fig 3.3.2 Pert Chart

3.4 Software and Hardware Requirements

Software Requirement:

- Operating System: Window 10 or +
- Continuous Internet Connectivity

Hardware Requirement:

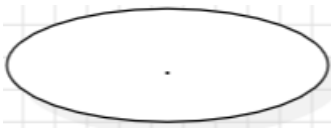


- Personal computer with keyboard and mouse maintained with uninterrupted power supply.
- Webcam.

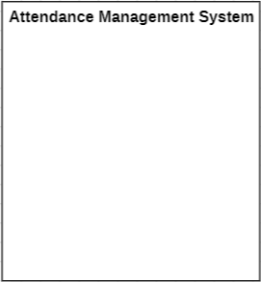
3.5 Preliminary Product Description

The main objective of project is to reduce the effort and time consumed in marking the attendance in offline mode. The problem is effectively reduced by allowing users to mark attendance digitally. This Attendance Management System will also provide easier method to save time and effort. The Attendance Management System is a website that helps the user to mark and check attendance in online mode by saving time and option to manage the Attendance efficiently.

3.6 Conceptual Model

Use Case: A Use Case diagram is a visual representation in the Unified Modelling Language (UML) that illustrates the interactions between various actors (e.g., users, systems, external entities) and a system to achieve specific goals or functionalities. It provides a high-level view of the system's functionality by showing the different ways the system can be used and the actors involved in those interactions. Use Case diagrams are valuable tools for understanding system requirements, communicating system functionalities to stakeholders, and serving as a basis for further development and design.

Name	Symbol	Description
Use Case		Represent the specific functionalities or actions that the system provides to its actors. Use cases describe the interactions between the actors and the system to achieve a particular goal or task.
Actor		Represent the users, systems, or external entities that interact with the system. Actors are typically depicted as stick figures or blocks outside the system boundary.
Direct Association		Indicates that an actor is associated with a use case, representing a relationship between an actor and the functionalities they can access.

System Boundary	 <p>The diagram shows a rectangular box labeled "Attendance Management System" at the top. Inside this box, there are several smaller, empty rectangular boxes arranged in a grid-like pattern, representing use cases within the system boundary.</p>	Represents the boundaries of the system being modelled, separating the system from its external actors.
--------------------	---	---

Actors in the project:

- User
- Admin
- Staff

Use cases:

- Login
- Add Branch
- Add Staff
- Add Attendance
- Add Student
- Modify Branch
- Modify Staff
- Modify Student
- Modify Attendance
- View Attendance
- Change Password
- Logout

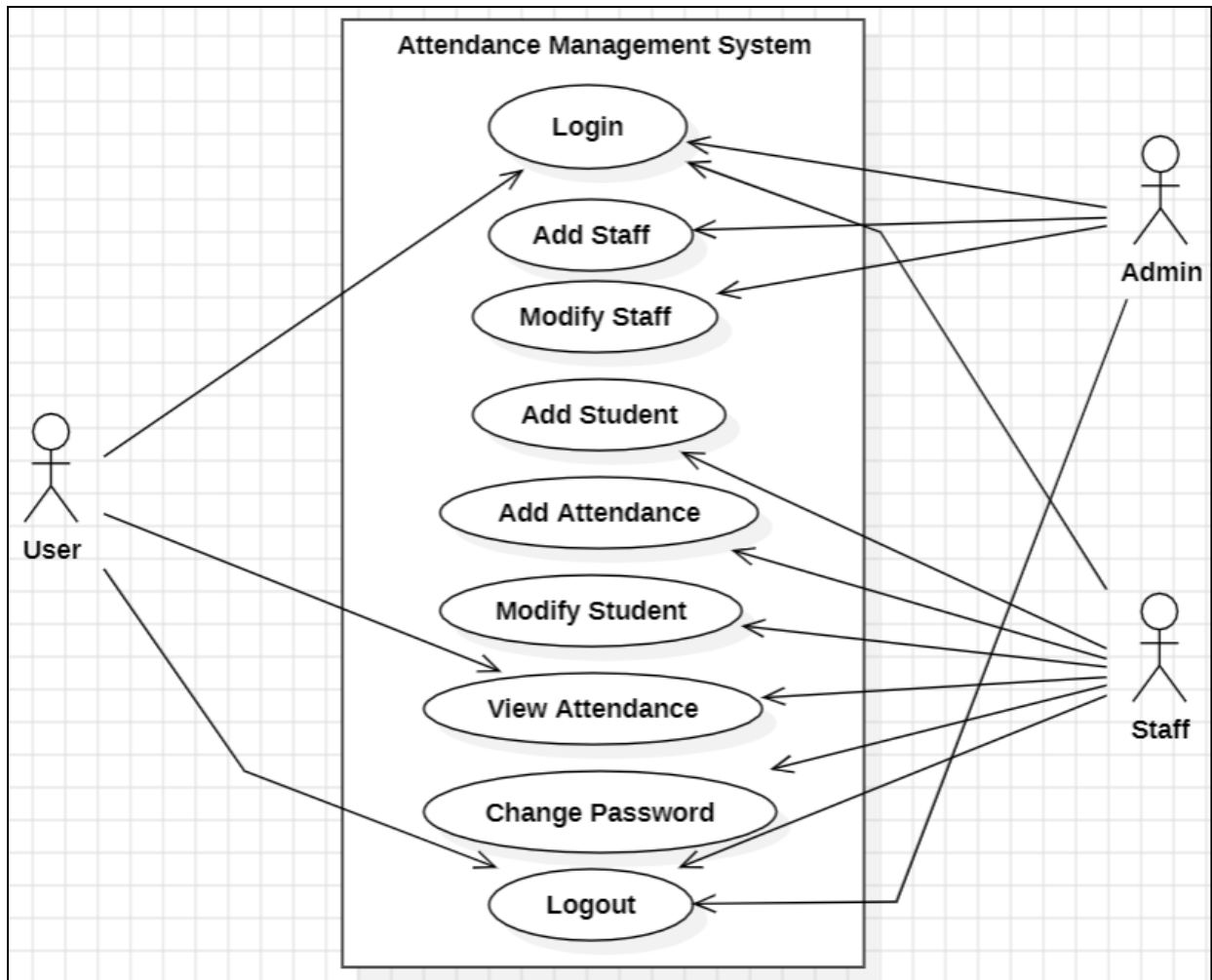






Fig 3.6.1 Use Case Diagram

Data Flow Diagram: A Data Flow Diagram (DFD) is a graphical representation that illustrates how data flows through a system, depicting the processes, data stores, data flow, and external entities involved. It provides a structured way to model the flow of data within a system and its interactions with external entities. DFDs are useful for visualizing the system's data flow, understanding data processing, identifying data stores, and providing a high-level overview of the system's functionality. They are commonly used in system analysis and design to model information systems, aid in requirements analysis, and facilitate effective communication between stakeholders.

Name	Symbol	Description
External Entities		Represent entities outside the system that interact with the system by providing input data or receiving output data. These can be users, other systems, or devices.
Process		Represent the functions or transformations that occur within the system. Each process takes input data, performs a function, and produces output data.
Data Stores		Represent repositories where data is stored within the system. These can be databases, files, or any other storage locations.
Data Flow		Represents the movement of data between processes, data stores, and external entities. It shows how data is input, processed, and output by the system.

Entities Used:

- Admin
- User
- Staff

Processes:

- Login
- Add/Modify Attendance
- View Attendance
- Add/Modify Student
- Logout

Data Stores:

- Login_db
- Data_db

a) Context Level Diagram

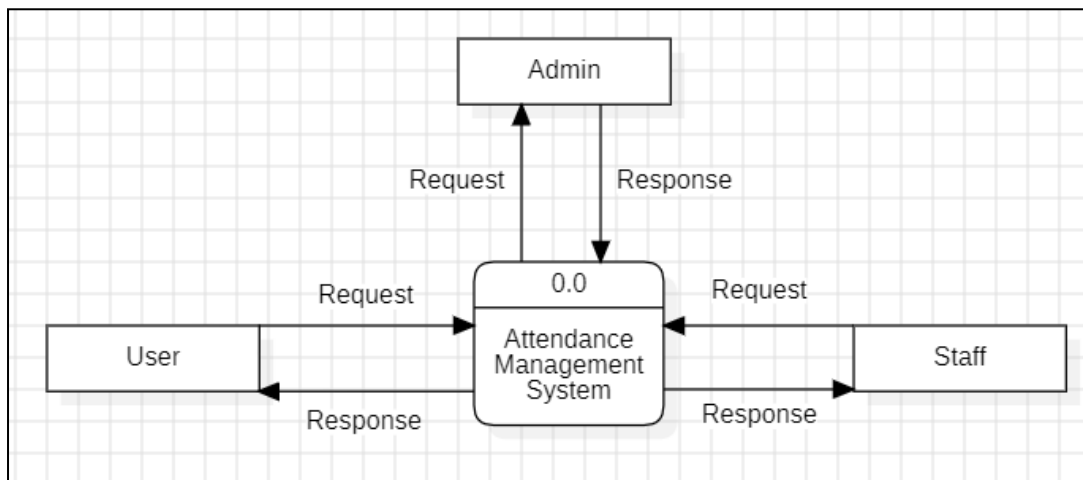


Fig 3.6.2.1 Data Flow Diagram (Context Level)

b) First Level Diagram

i) For User

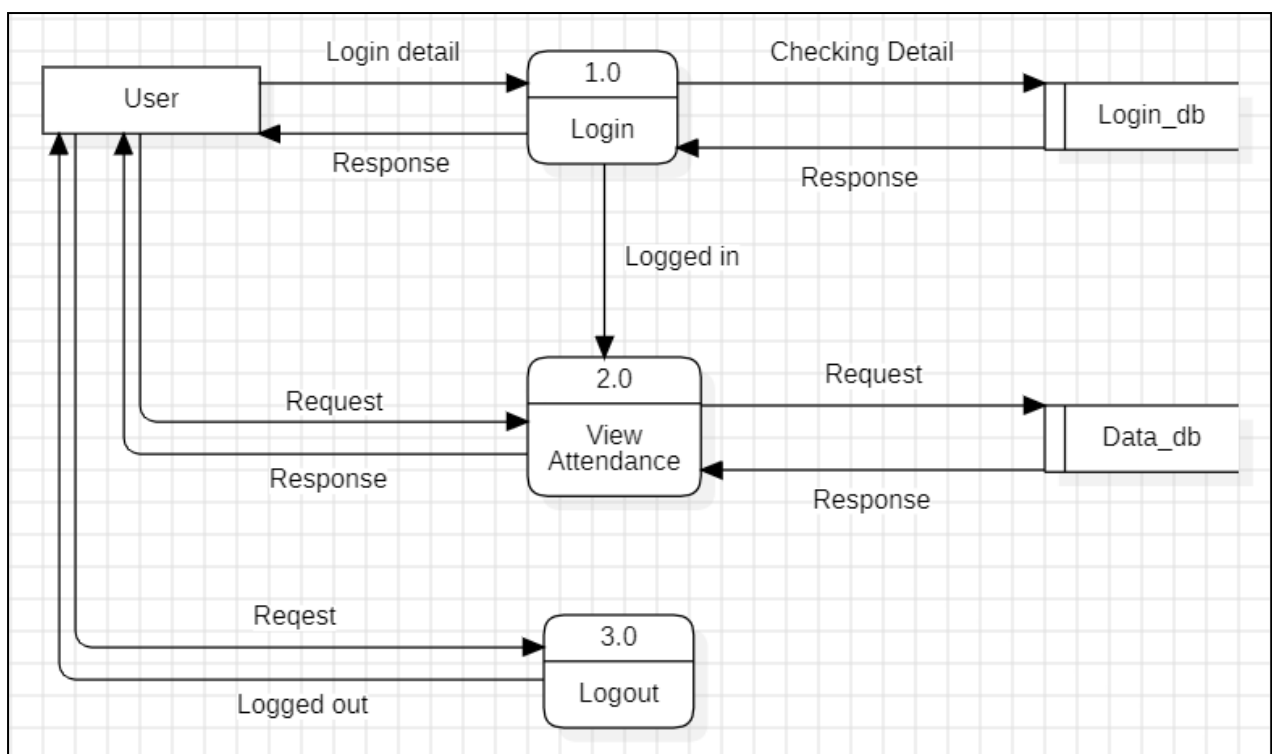


Fig 3.6.2.2 Data Flow Diagram for User (First Level)

ii) For Staff

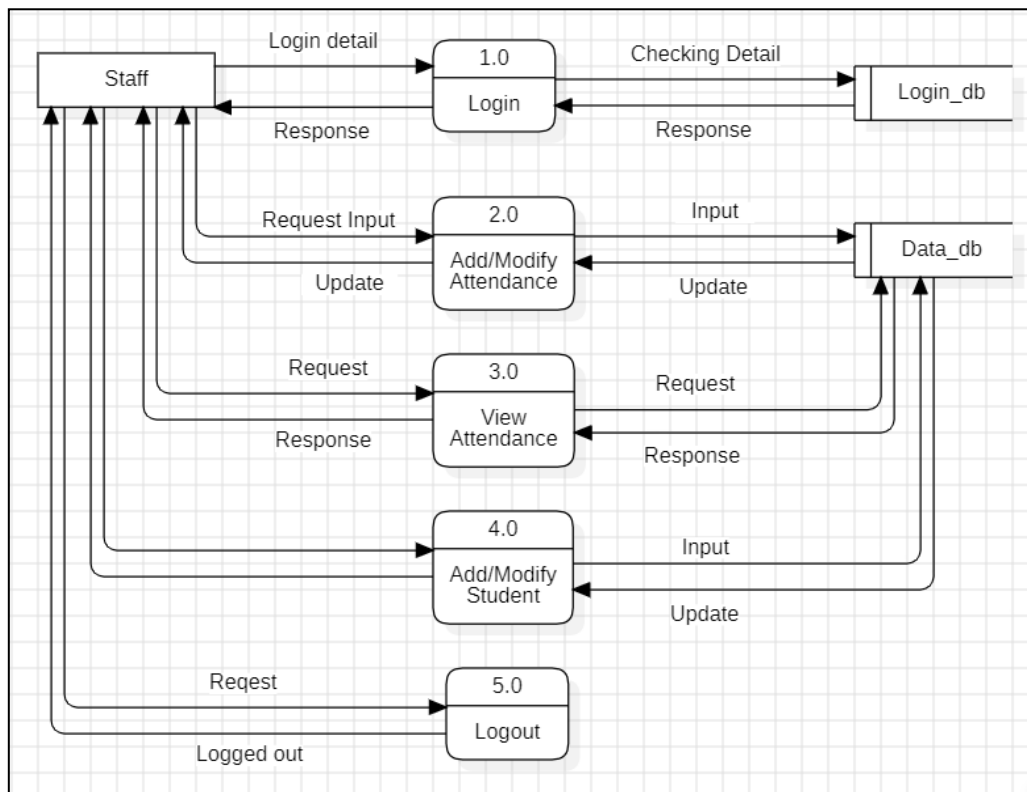


Fig 3.6.2.3 Data Flow Diagram for Staff (First Level)

iii) For Admin

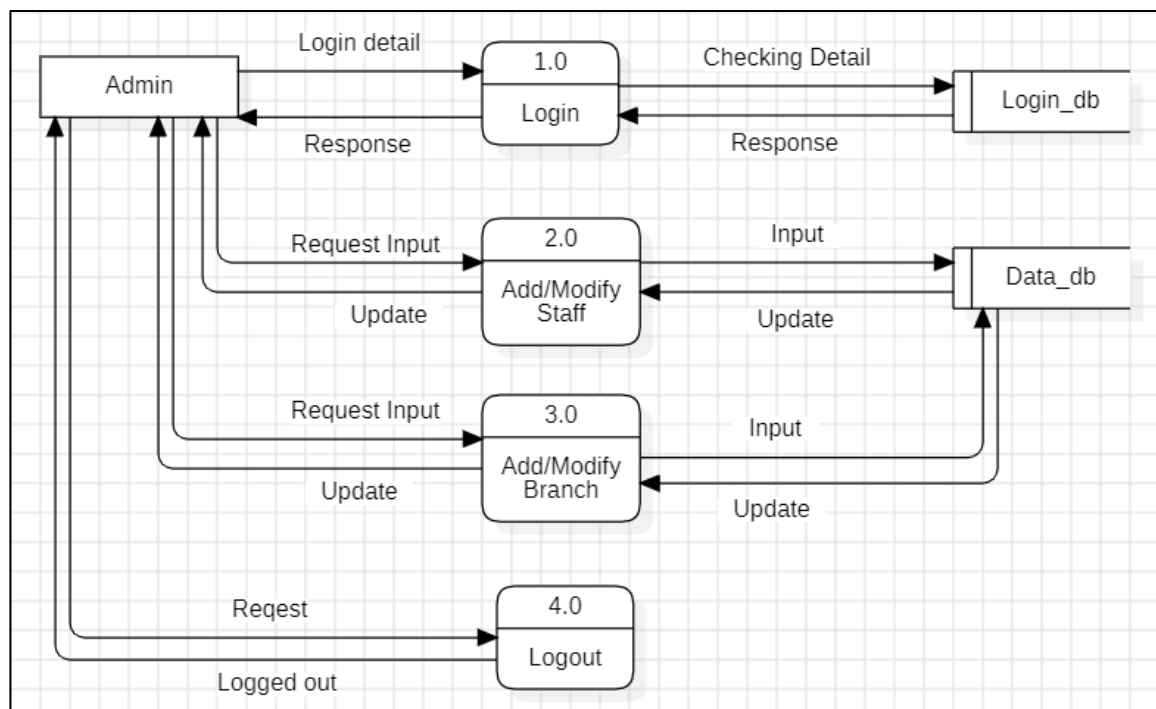




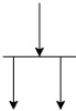
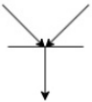



Fig 3.6.2.4 Data Flow Diagram for Admin (First Level)

Activity Diagram:

An activity diagram is a graphical representation that illustrates the flow of activities or actions within a system or process. It is widely utilized in software development to model the sequential steps and actions involved in a business process, a use case, or any other system behaviour. Activity diagrams are powerful tools for visualizing and understanding the workflow and interactions between various components in a system. They provide a clear, structured view of how actions or tasks progress and interact, making it easier for stakeholders to comprehend and analyse complex processes in software development and other domains.

Name	Symbol	Description
Start Node		<ul style="list-style-type: none">Represents the starting point of the activity diagram.
Action		<ul style="list-style-type: none">Represents a specific action or operation performed in the system. Actions can include calculations, data processing, or any other task.
Control Flow		<ul style="list-style-type: none">Represents the flow of control or the sequence in which activities are performed. Arrows connect actions and control flow nodes.
Decision Node		<ul style="list-style-type: none">Used to depict a decision point in the process flow, where different paths or actions may be taken based on conditions.
Fork		<ul style="list-style-type: none">Indicates the start of concurrent activities or parallel processing.
Join		<ul style="list-style-type: none">Indicates the merging of concurrent activities back into a single path.
End Node		<ul style="list-style-type: none">Represents the endpoint of the activity diagram, indicating the completion of the process.

- For Staff

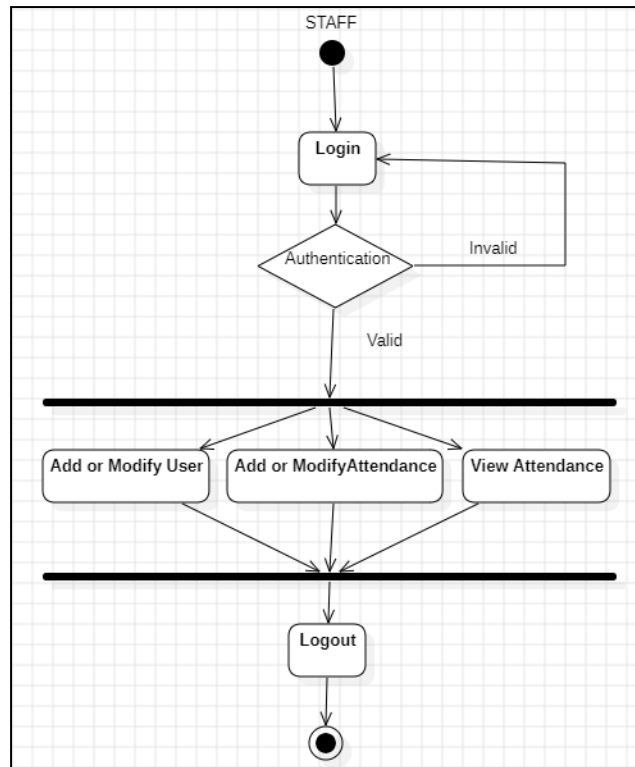


Fig 3.6.3.1 Activity Diagram for Staff

- For Admin

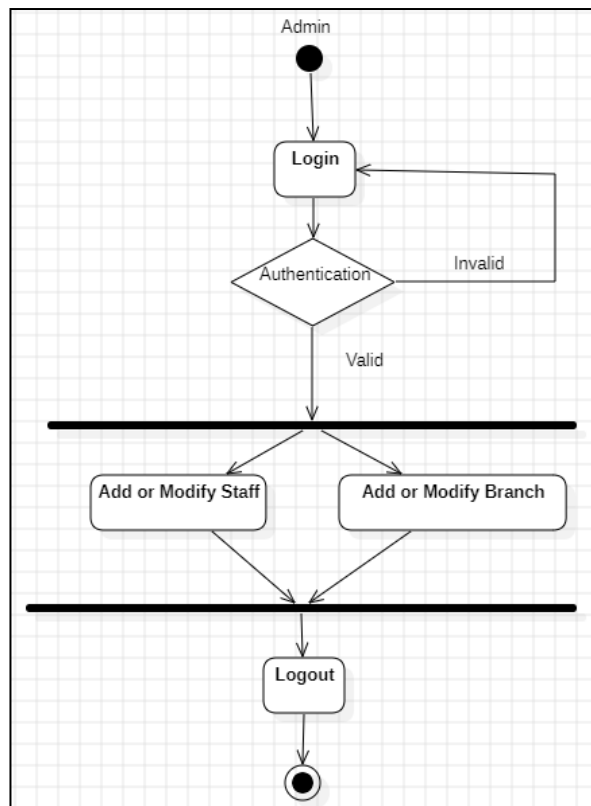


Fig 3.6.3.2 Activity Diagram for Admin

- For User

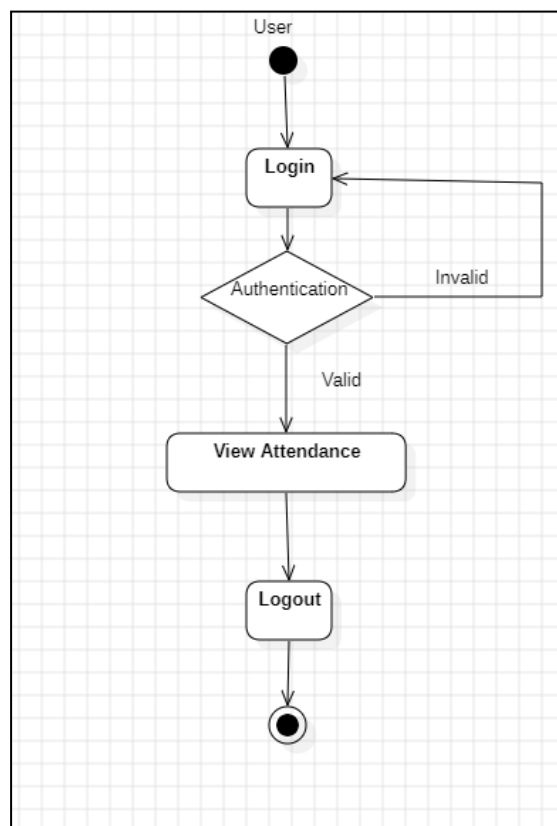
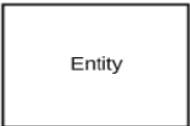


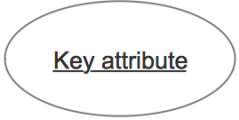


Fig 3.6.3.3 Activity Diagram for User

Entity Relationship Diagram:

An Entity-Relationship Diagram (ERD) is a visual representation used in database design to illustrate the relationships between entities and their attributes in a system. It's a widely used technique to model the logical structure of a database and how data is related within that database.

Name	Symbol	Description
Entity		Represents a real-world object, concept, or thing in the system that stores data. Each entity typically corresponds to a table in a database.
Relationship		Describes how entities are related to each other and can be one-to-one, one-to-many, or many-to-many. Relationships are represented by connecting lines between entities.

Attribute		Describes properties or characteristics of an entity and is depicted within the entity's box. Attributes are the columns in a database table.
Key Attribute		An attribute that uniquely identifies an entity within its entity set. It's typically underlined in the entity.

Types of Multiplicities-

- Many-to-Many (M:M)
- Many-to-One (M:1)
- One-to-Many (1:M)
- One-to-One (1:1)

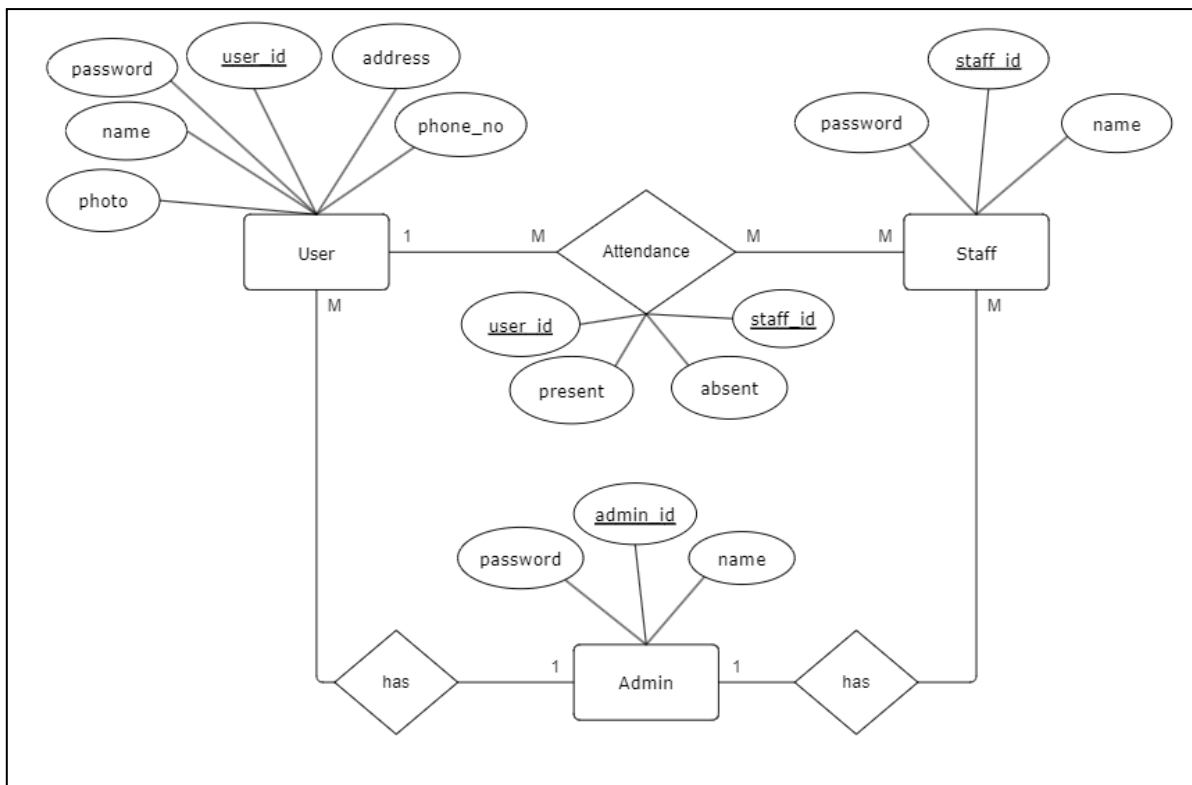


Fig 3.6.4 Entity Relationship Diagram

Chapter 4

System Design

4.1 Basic Modules

In software development or system design, "basic modules" refer to core building blocks or essential components that perform specific functions or encapsulate distinct features. These modules are designed for ease of management, development, and to contribute to the construction of complex systems. They are often reusable, maintainable, and serve as fundamental units within the system.

In this project we have number of modules which are:

- **User Authentication and Access Control Module:** This module allows users to log in securely using their credentials. It also defines roles and permissions to control access levels for different users (e.g., admin, staff, user).
- **Attendance Tracking Module:** This module enables the recording of attendance data for individuals for specific dates and times.
- **Reporting Module:** This module creates monthly based reports of the attendance.
- **Settings and Configuration Module:** This module enables administrators and staff to configure the features and attendance, etc.
- **Dashboard and Visualization Module:** This module displays an easy to use and understandable UI presenting real time information in form of graphs, charts or tables.
- **Notification and Alert Module:** This module notifies the users for updates or any late arrivals or absent markings through email, sms or inbox.
- **Database Module:** This module consists of databases consisting of all the user data, login data and attendance data.

4.2 Data Design

4.2.1 Schema Design

Schema design, in the context of databases, refers to the process of creating the structure and organization of a database to ensure efficient storage, retrieval, and management of data. It involves defining the tables, relationships, constraints, and other elements that constitute the overall architecture of the database.

The goal of schema design is to design a database schema that meets the requirements of the application while optimizing performance, minimizing redundancy, and ensuring data integrity.

A database schema can be divided broadly into two categories:

- **Physical Database Schema:** This schema pertains to the actual storage of data and its form of storage like files, indices, etc. It defines how the data will be stored in a secondary storage.
- **Logical Database Schema:** This schema defines all the logical constraints that need to be applied on the data stored. It defines tables, views, and integrity constraints.

Database Name:

- AMS_db

List of Tables:

- login_data
- details_data
- attendance_data
- role_data

4.2.2 Data Integrity and Constraints

Database Name:

AMS_db

Database Table:

- login_data

Field	Type	Null	Key	Default	Extra
login_id	Varchar(50)	NO		NULL	
password	Varchar(50)	NO		NULL	
role_id	int	YES	FK	NULL	
id	Varchar(50)	YES	FK	NULL	

Table 4.2.2.1 login_data

- details_data

Field	Type	Null	Key	Default	Extra
Id	Varchar(50)	NO	PK	NULL	
Name	Varchar(100)	NO		NULL	
phone_no	Varchar(50)	NO		NULL	
email_id	Varchar(150)	YES		NULL	
Address	Varchar(150)	YES		NULL	
role_id	int	YES	FK	NULL	

Table 4.2.2.2 details_data

- attendance_data

Field	Type	Null	Key	Default	Extra
id	Varchar(50)	NO	FK	NULL	
attendance	Char(1)	NO		NULL	
date	timestamp	NO		NULL	

Table 4.2.2.3 attendance_data

- role_data

Field	Type	Null	Key	Default	Extra
role_id	int	NO	PK	NULL	
Role	Varchar(100)	NO		NULL	

Table 4.2.2.4 role_data

4.3 Procedural Design

4.3.1 Logic Diagrams

A logical diagram visually represents key concepts, processes, or relationships. It can take forms like flowcharts, mind maps, ERDs, Gantt charts, or system architecture diagrams, aiding in explaining research methodology, data flow, and theoretical frameworks succinctly.

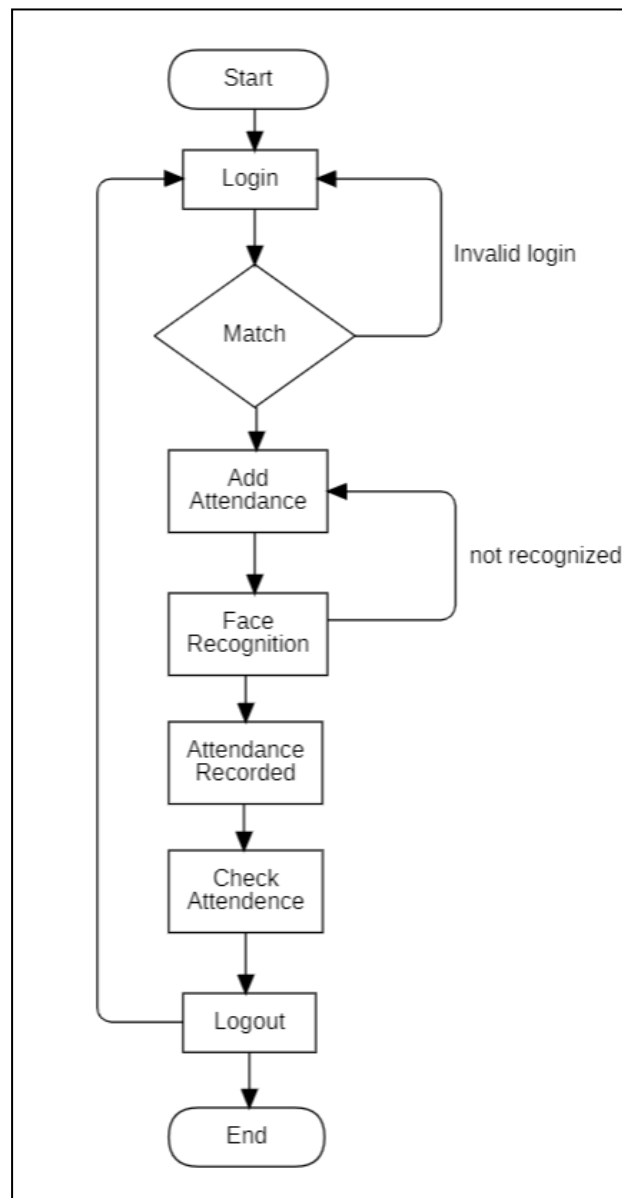


Fig 4.3.1.1 Logic Diagram

4.3.2 Data Structures

Data structures refer to how information is organized, stored, and manipulated for analysis. It involves choosing appropriate data structures based on the type of data and the research objectives. Common data structures in a project dissertation include:

Graphs Data Structure: For representing relationships between data points, often used in social network analysis or any research involving interconnected data.

Arrays Data Structure: Ordered collections of items, suitable for storing data that can be accessed using an index, commonly used in algorithms and computations.

Trees Data Structure: Hierarchical structures useful for organizing data with parent-child relationships frequently used in organizing file systems or representing organizational hierarchies.

We will be using Tree Data Structure for this project:

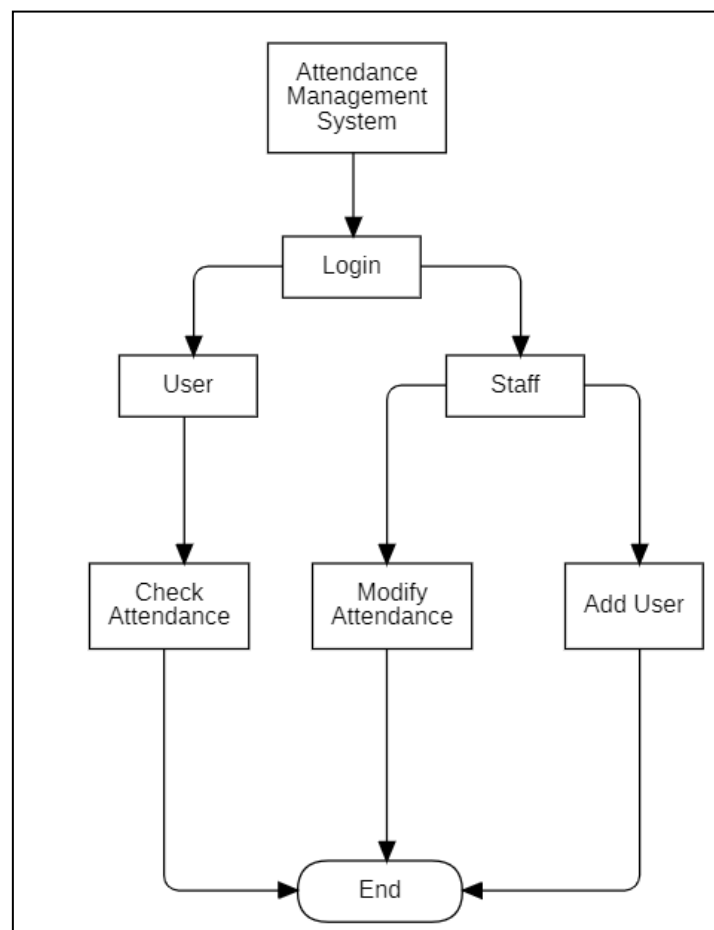


Fig 4.3.2.1 Data Structures Diagram

4.3.3 Algorithm Design

An algorithm is step-by-step analysis of the process, while flowchart explains the steps of a program in a graphical way. Algorithm and flowcharts help to clarify all the steps for the following face recognition process.

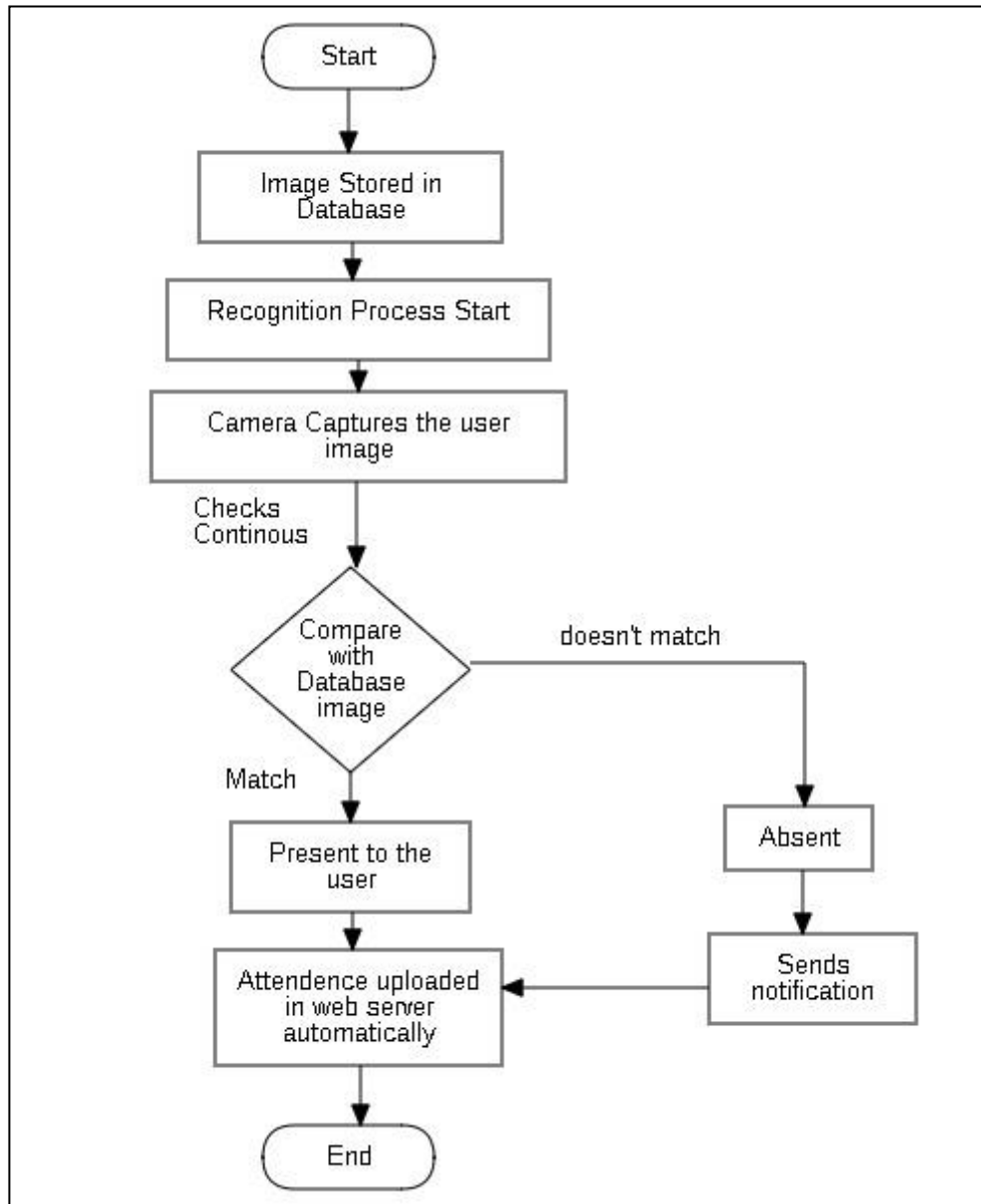


Fig 4.3.3.1 Algorithm Design

4.4 User Interface Design

Login Page:

- Displaying the login page for the website.

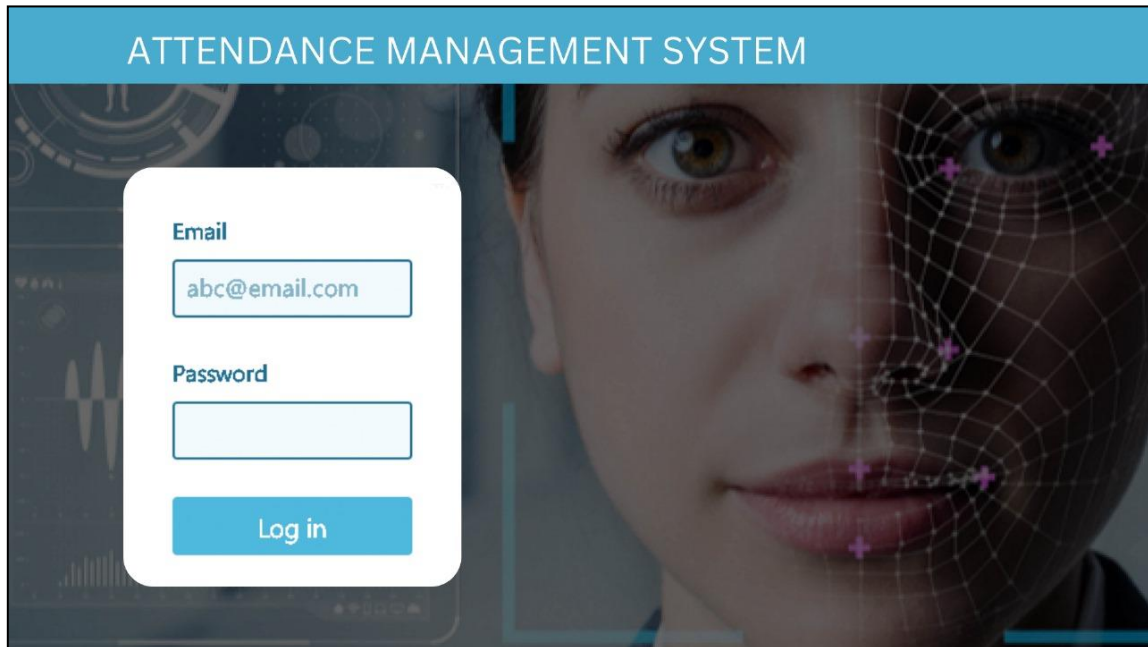


Fig 4.4.1 Login Page

Home Page:

- The Homepage of the website after login into User1's account. Showcasing the dashboard with pie representation of the attendance.



Fig 4.4.2 Home Page

Attendance Page:

- The page staffs will be shown while taking the attendance in face recognition mode.

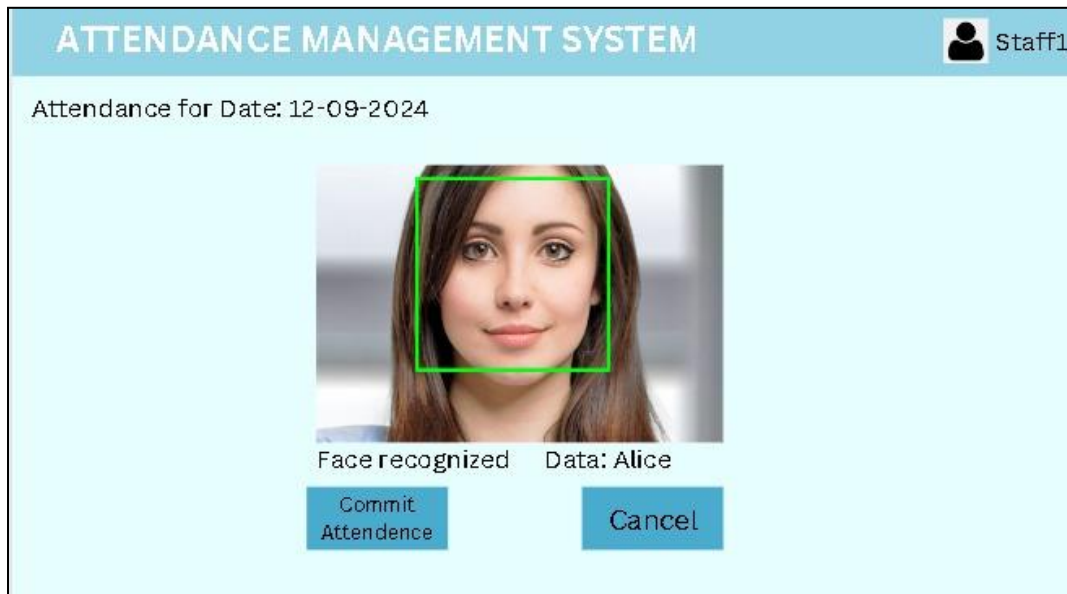


Fig 4.4.3 Attendance Page

4.5 Security Issues

1. Unrestricted Access:

- Issue: Allowing unrestricted access to the attendance management system, including unnecessary privileges.
- Solution: Implement role-based access controls to restrict access based on job roles, granting only the necessary permissions for each user.

2. Data Theft:

- Issue: Unauthorized users stealing attendance data for malicious purposes.
- Solution: Regularly audit and monitor access logs to detect and investigate any suspicious activity.

3. Inadequate User Training:

- Issue: Employees not being aware of potential security risks associated with the attendance management system.
- Solution: Provide regular training to employees on how to use the system securely, recognize potential threats, and report any suspicious activity.

4. Improper Logout and Session Management:

- Issue: Users not logging out properly, leaving their sessions open and vulnerable to misuse.

- Solution: Implement an automatic logout feature after a period of inactivity and educate users to log out properly when they finish using the system.

5. Spoofing or Presentation Attacks:

- Issue: Malicious users may attempt to deceive the face recognition system by presenting a photo, video, or a 3D model of an authorized person's face.
- Solution: Implement anti-spoofing techniques, such as liveness detection, which can differentiate between a real human face and a spoof.

4.6 Test Case Design

TEST CASE ID	LOGIN ID	PASSWORD	EXPECTED RESULT	ACTUAL RESULT	STATUS
Login_01	user1	user123	Login Password Incorrect		
Login_02	user1	user@123	Login successful		
Login_03	user100	user@123	Login Username Incorrect		
Login_04	user10	user@13	Login Credential Incorrect		

Table 4.6.1 Test Case 1

ID	Test Case	Input	Output	Test Result
1	Face not recognised	Improper scanned face	Absent marked attendance	Fail
2	Face recognised	Proper Scan	Present marked attendance	Pass
3	Dirty camera	Face not recognised	Absent marked attendance	Fail

Table 4.6.2 Test Case 2

Chapter 5

Implementation Testing

5.1 Implementation Approach

- The entire system uses the Incremental Development Approach of software development for its development.
- The software is developed in small-small increments on a weekly basis.
- All these increments are combined in the end to form a full-fledged website.
- These increments are carried out until the final software is developed.
- Here the software/system is developed in increments/units, in the end all these units/increments are merged to form the final system/software.

Phases of Incremental Development Approach:

1. Requirement Analysis

- Here we identify all the requirements which are needed and necessary for building the software/system.
- HTML, CSS, JavaScript, Flask, MongoDB Atlas were chosen for building the software.
- The requirements are:- Building an Attendance Management System which will be capable of recording attendance by recognizing face, Which and all pages might be needed, Requirements of the appearance of User Interface.

2. Designing

- Here we analyze the requirements and convert them into a design which acts as a blueprint which is to be followed during development of the software.
- Designing the UI, Identifying the modules, etc is supposed to be done here in this phase.

3. Development and Implementation

- In this phase we develop whatever is made in the design phase and implement it to achieve a working project as per the requirements specified.

- We choose to do our frontend using HTML and CSS, and we choose to manage our backend using python (flask) and JavaScript, and for database we are using MongoDB Atlas Cloud.

4. Testing

- Here we test the system to ensure everything works without any possible errors and bugs.
- Different kinds of test cases and strategies are used to identify the bugs and errors in the system.

This project was completed in 6 increments.

Increment 1:

- In the first increment, face recognition was implemented.
- Here when the user faces the camera it will read the data of the user from the database and match it with the face it is capturing at that moment and mark attendance accordingly.
- After this, the user's name will be displayed in the command prompt.

Increment 2:

- Here we worked on the User Interface of the website.
- Pages related to Login and Admin have been made.

Increment 3:

- The two units of UI and face recognition were merged into one module.
- Also, the backend for all the pages were done in this increment.

Increment 4:

- UI for the Staff and User were made, and sample data has been added in this increment.

Increment 5:

- Backend is completed as per the progress done with the UI.
- Features and functionality are added and tested accordingly.

Increment 6:

- Merging all modules and it into a single software. Also, the software was testing for proper functioning of features and the values it is fetching storing etc.

5.2 Coding Details and Code Efficiency

```
from flask import Flask, request, jsonify, render_template, session, redirect, url_for,
send_from_directory
from flask_pymongo import PyMongo
from flask_cors import CORS
from functools import wraps
import os
import cv2
import dlib
import base64
import numpy as np
from pymongo import MongoClient
from datetime import datetime, timedelta
from bson import ObjectId

app = Flask(__name__)
CORS(app)
app.config['MONGO_URI'] =
'mongodb+srv://kunalgiri:8149509019k@ams.1qyodxz.mongodb.net/data'
app.secret_key = 'thisisasecretkey'
mongo = PyMongo(app)

client = MongoClient('mongodb+srv://kunalgiri:8149509019k@ams.1qyodxz.mongodb.net/')
db = client['attendance']
attendance_records = db['attendance_records']
learn_collection = db['learn_data']
access_granted = False
recognized_persons = set()
```

```

detector = dlib.get_frontal_face_detector()
predictor = dlib.shape_predictor("shape_predictor_68_face_landmarks.dat")
face_recognizer =
dlib.face_recognition_model_v1("dlib_face_recognition_resnet_model_v1.dat")

known_faces = { }

@app.route('/favicon.ico')
def favicon():
    return send_from_directory(os.path.join(app.root_path, 'static'), 'favicon.ico',
                               mimetype='image/vnd.microsoft.icon')

def authorize_admin(func):
    @wraps(func)
    def wrapper(*args, **kwargs):
        if 'username' not in session:
            return jsonify({'message': 'Unauthorized access!'}), 401
        if session.get('role') != 'admin':
            return jsonify({'message': 'Unauthorized access!'}), 403
        return func(*args, **kwargs)
    return wrapper

def authorize_user(func):
    @wraps(func)
    def wrapper(*args, **kwargs):
        if 'username' not in session:
            return jsonify({'message': 'Unauthorized access!'}), 401
        if session.get('role') != 'user':
            return jsonify({'message': 'Unauthorized access!'}), 403
        return func(*args, **kwargs)
    return wrapper

def authorize_staff(func):
    @wraps(func)
    def wrapper(*args, **kwargs):
        if 'username' not in session:
            return jsonify({'message': 'Unauthorized access!'}), 401
        if session.get('role') != 'staff':
            return jsonify({'message': 'Unauthorized access!'}), 403
        return func(*args, **kwargs)
    return wrapper

@app.route('/')
def index():
    return render_template('index.html')

@app.route('/login', methods=['POST'])
def login():

```

```

data = request.get_json()
username = data.get('username')
password = data.get('password')
print(f"Received login request for username: {username}, password: {password}")
user = mongo.db.creds.find_one({'username': username, 'password': password})

if user:
    session['username'] = username
    session['role'] = user['role']
    print(f"User found in the database: {user}")
    return jsonify({'message': 'Successfully logged in!', 'role': user['role']})
else:
    print("Invalid credentials")
    return jsonify({'message': 'Invalid credentials'}), 401

@app.route('/logout', methods=['POST'])
def logout():
    session.pop('username', None)
    session.pop('role', None)
    return redirect(url_for('index'))

@app.route('/admin', methods=['GET', 'POST'])
@authorize_admin
def admin():
    if request.method == 'POST':
        name = request.form.get('name')
        email = request.form.get('email')
        title = request.form.get('title')
        username = request.form.get('username')
        password = request.form.get('password')
        uid = request.form.get('uid')

        mongo.db.staff.insert_one({
            'name': name,
            'email': email,
            'title': title,
            'role': 'staff',
            'uid': uid
        })

        mongo.db.creds.insert_one({
            'username': username,
            'password': password,
            'role': 'staff',
            'uid': uid
        })

    return redirect(url_for('admin'))

```

```

else:
    staff_data = mongo.db.staff.find()
    user_data = mongo.db.user.find()
    return render_template('admin.html', staff_data=staff_data, user_data=user_data)

@app.route('/delete_staff', methods=['POST'])
def delete_staff():
    staff_id = request.form['staff_id']
    staff_uid = request.form['staff_uid']
    mongo.db.staff.delete_one({'_id': ObjectId(staff_id)})
    mongo.db.creds.delete_one({'uid': staff_uid})

    return redirect(url_for('admin'))

@app.route('/user')
@authorize_user
def user():
    username = session.get('username')
    user_data = mongo.db.creds.find_one({'username': username})

    if user_data:
        uid = user_data.get('id')
        user_profile = mongo.db.user.find_one({'uid': uid})

        if user_profile:
            user_name = user_profile.get('name')
            img_folder = user_profile.get('img_folder')
            profile_image_path = url_for('static',
filename=f'images/profile/{img_folder}/profile.jpg')
            attendance_data = fetch_attendance_data_for_user(user_name, 2024, 3)
            present_count = sum(1 for record in attendance_data if record['status'] == 'Present')
            absent_count = sum(1 for record in attendance_data if record['status'] == 'Absent')

            notifications = mongo.db.messages.find({'recipient': 'user', 'sender':
'staff'}).sort([('timestamp', -1)])

            birthdate_data = mongo.db.user.find({}, {'name': 1, 'birthdate': 1})

            birthdate_data_formatted = []
            for user in birthdate_data:
                birthdate_str = user['birthdate']
                birthdate_obj = datetime.strptime(birthdate_str, '%Y-%m-%d')
                user['birthdate'] = datetime.strftime(birthdate_obj, '%d-%m-%Y')
                birthdate_data_formatted.append(user)

            return render_template('user.html',

```

```

        profile_image_path=profile_image_path,
        user_name=user_name,
        attendance_data=attendance_data,
        birthdate_data=birthdate_data_formatted,
        present_count=present_count,
        absent_count=absent_count,
        notifications=notifications)
    else:
        return jsonify({'message': 'User profile not found!'}), 404
    else:
        return jsonify({'message': 'User data not found!'}), 404

def fetch_attendance_data_for_user(username, year, month):
    start_date = datetime(year, month, 1)
    end_date = datetime.now()
    attendance_data_monthly = []
    current_date = start_date
    while current_date < end_date:
        formatted_date = current_date.strftime("%d-%m-%Y")

        attendance_data_daily = list(attendance_records.find({'name': username, 'date':
formatted_date}))

        if attendance_data_daily:
            for record in attendance_data_daily:
                attendance_data_monthly.append({
                    'name': record['name'],
                    'date': formatted_date,
                    'time': record.get('time', '-'),
                    'status': record['status']
                })
        else:
            attendance_data_monthly.append({
                'name': username,
                'date': formatted_date,
                'time': '-',
                'status': 'Absent'
            })
        current_date += timedelta(days=1)

    return attendance_data_monthly

@app.route('/staff', methods=['GET', 'POST'])
@authorize_staff
def staff():
    if request.method == 'POST':
        if 'name' in request.form:
            name = request.form.get('name')

```



```

email = request.form.get('email')
birthdate = request.form.get('birthdate')
username = request.form.get('username')
password = request.form.get('password')
uid = request.form.get('uid')
mongo.db.user.insert_one({
    'name': name,
    'email': email,
    'role': 'user',
    'birthdate': birthdate,
    'uid': uid,
    'img_folder': username
})

mongo.db.creds.insert_one({
    'username': username,
    'password': password,
    'role': 'user',
    'id': uid
})

return redirect(url_for('staff'))

elif 'notification' in request.form:
    notification_message = request.form.get('notification')
    mongo.db.messages.insert_one({
        'recipient': 'user',
        'sender': 'staff',
        'message': notification_message,
        'timestamp': datetime.now()
    })

    return redirect(url_for('staff'))

else:
    attendance_data = fetch_attendance_data(2024, 3)
    unique_names = set(record['name'] for record in attendance_data)
    unique_dates = set(record['date'] for record in attendance_data)
    user_data = mongo.db.user.find()
    return render_template('staff.html', user_data=user_data,
attendance_data=attendance_data, unique_names=list(unique_names),
unique_dates=unique_dates)

def fetch_attendance_data(year, month):
    start_date = datetime(year, month, 1)
    end_date = datetime.now()

```

```

attendance_data_monthly = []

current_date = start_date
while current_date < end_date:
    formatted_date = current_date.strftime("%d-%m-%Y")

    attendance_data_daily = list(attendance_records.find({'date': formatted_date}))

    if attendance_data_daily:
        for record in attendance_data_daily:
            attendance_data_monthly.append({
                'name': record['name'],
                'date': formatted_date,
                'time': record.get('time', '-'),
                'status': record['status']
            })
    else:
        user_data = mongo.db.user.find({}, {'name': 1})
        for user in user_data:
            attendance_data_monthly.append({
                'name': user['name'],
                'date': formatted_date,
                'time': '-',
                'status': 'Absent'
            })

    current_date += timedelta(days=1)

return attendance_data_monthly

@app.route('/update_attendance', methods=['POST'])
def update_attendance():
    if request.method == 'POST':
        record_name = request.form['record_name']
        record_date = request.form['record_date']

        if 'attendance-toggle' in request.form:
            toggle_value = request.form.get('attendance-toggle')
            new_status = 'Present' if toggle_value == 'on' else 'Absent'
            time = datetime.now().strftime("%H:%M:%S")
        else:
            new_status = 'Absent'
            time = '-'

        attendance_records.update_one({'name': record_name, 'date': record_date}, {'$set':
{'status': new_status, 'time': time}})

```

```

        return redirect(url_for('staff'))

@app.route('/delete_user', methods=['POST'])
def delete_user():
    user_id = request.form['user_id']
    user_uid = request.form['user_uid']
    mongo.db.user.delete_one({'_id': ObjectId(user_id)})
    mongo.db.creds.delete_one({'id': user_uid})

    return redirect(url_for('staff'))

@app.route('/toggle_access', methods=['POST'])
def toggle_access():
    global access_granted
    data = request.get_json()
    access_granted = data.get('accessGranted')
    return jsonify({'status': 'success', 'message': 'Access toggled successfully'})

@app.route('/check_access')
def check_access():
    return jsonify({'accessGranted': access_granted})

@app.route('/video')
@authorize_user
def face():
    username = session.get('username')
    user_data = mongo.db.creds.find_one({'username': username})

    if user_data:
        uid = user_data.get('id')
        user_profile = mongo.db.user.find_one({'uid': uid})

        if user_profile:
            user_name = user_profile.get('name')
            img_folder = user_profile.get('img_folder')
            profile_image_path = url_for('static',
filename=f'images/profile/{img_folder}/profile.jpg')
            return render_template('face.html',
profile_image_path=profile_image_path,user_name=user_name)
        else:
            return jsonify({'message': 'User profile not found!'}), 404
    else:
        return jsonify({'message': 'User data not found!'}), 404

@app.route('/capture', methods=['POST'])
def capture():
    try:
        image_data_url = request.json['image']

```

```

_, encoded = image_data_url.split(",", 1)
decoded = base64.b64decode(encoded)
np_arr = np.frombuffer(decoded, dtype=np.uint8)
img = cv2.imdecode(np_arr, cv2.IMREAD_COLOR)

recognized_name = recognize_face(img)
print(recognized_name)

if recognized_name:
    today_date = datetime.now().strftime("%d-%m-%Y")
    existing_record = db.attendance_records.find_one({'name': recognized_name,
'date':today_date, 'status': "Present"})

    if existing_record:
        return jsonify({'status': 'success', 'message': f'Attendance already recorded for
{recognized_name} today', 'redirect_url': '/user'})

        db.attendance_records.update_one({
            'name': recognized_name,
            'date': today_date
        }, {'$set': {
            'username': session.get('username'),
            'status': 'Present',
            'time': datetime.now().strftime("%H:%M:%S")
        }})

        return jsonify({'status': 'success', 'message': f'Attendance recorded for
{recognized_name}', 'redirect_url': '/user'})
    else:
        return jsonify({'status': 'fail', 'message': 'Face not recognized'})

except Exception as e:
    print(e)
    return jsonify({'status': 'error', 'message': 'Error processing the image'})

def learn_faces_from_images(images_folder):
    global known_faces
    known_faces = {}

    for person_name in os.listdir(images_folder):
        person_path = os.path.join(images_folder, person_name)
        if os.path.isdir(person_path):
            person_images = [cv2.imread(os.path.join(person_path, img)) for img in
os.listdir(person_path)]
            person_face_descriptors = []

            for img in person_images:
                gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

```

```

        faces = detector(gray)

        if faces:
            landmarks = predictor(gray, faces[0])
            face_descriptor = face_recognizer.compute_face_descriptor(img, landmarks)
            person_face_descriptors.append(face_descriptor)

        if person_face_descriptors:
            known_faces[person_name] = np.mean(person_face_descriptors, axis=0)

    return known_faces

def recognize_face(img):
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    faces = detector(gray)

    for face in faces:
        landmarks = predictor(gray, face)
        face_descriptor = face_recognizer.compute_face_descriptor(img, landmarks)

        for name, known_descriptor in known_faces.items():
            distance = np.linalg.norm(face_descriptor - known_descriptor)

            if distance < 0.6:
                return name

    return None

def add_initial_attendance_records():
    today_date_str = datetime.now().strftime('%d-%m-%Y')
    user_data = mongo.db.user.find({}, {'name': 1, 'uid': 1})
    creds_data = mongo.db.creds.find({}, {'username': 1, 'id': 1})
    uid_to_username = {cred['id']: cred['username'] for cred in creds_data}
    attendance_records = db.attendance_records

    for user in user_data:
        if user['uid'] in uid_to_username:
            username = uid_to_username[user['uid']]
            name = user['name']
            existing_record = attendance_records.find_one({'name': name, 'date': today_date_str})

            if existing_record:
                continue

            result = attendance_records.insert_one({
                'name': name,
                'username': username,
                'status': 'Absent',

```

```

        'date': today_date_str
    })

if __name__ == '__main__':
    print("Adding initial attendance records...")
    add_initial_attendance_records()
    images_folder = "images"
    learn_faces_from_images(images_folder)
    app.run(debug=True)

```

5.3 Testing Approaches

1) Unit Testing for Backend Functions:

Executed unit tests to validate functions such as ‘login’, ‘logout’, ‘sending notification’, and database operations for adding, updating, and deleting staff and user data and attendance data.

2) Integration Testing for API Endpoints:

Conducted integration tests to validate endpoints such as ‘/login’, ‘/logout’, ‘/add_staff’, and ‘/delete_staff’, ‘/add_user’, and ‘/delete_user’ ensuring proper handling of requests and responses.

3) UI Testing for Frontend Pages:

Performed UI testing to validate user interactions such as toggling the sidebar, adding, and deleting staff/user members, and capturing images for face recognition.

4) Security Testing for Authentication:

Conducted security testing to assess the robustness of authentication functionalities against common vulnerabilities such as SQL injection and cross-site scripting.

5) Cross-browser Testing for Compatibility:

Performed cross-browser testing to validate the compatibility of front-end pages on popular browsers like Brave, Chrome, and Opera.

6) Data Validation Testing:

Implemented data validation tests to verify the correctness of input data formats, including email addresses, staff information, and user credentials. Validated inputs against predefined rules and constraints to ensure data consistency and integrity.

7) Output Verification Testing:

Conducted output verification tests to validate the correctness of data displayed on front-end pages, including staff lists, user information, and face recognition results. Compared the expected output with the actual output to identify any discrepancies or errors in data presentation.

8) Regression Testing for Code Changes:

Implemented regression testing to re-validate existing functionalities and ensure that recent code modifications do not introduce new defects or break existing features.

5.4 Modification and Implementation

- Face recognition logic initially used was modified into newer logic which accepts input for checking data from database instead of from the data embedded into the code itself.
- Constant changes in the UI of the website to make it better and better. Changes in color and themes, styles, components, etc.
- Shifting from MongoDB Shell to MongoDB Atlas for easy data accessibility.
- Testing every single component individually that is implemented on the website, checking the working of it that if it is performing as expected.

5.5 Test Case

1) Admin Access and Operations:

Sr. No.	Test Case Description	Test Values	Expected Results
1	Test Admin Access with Valid Credentials	Valid username, password	Ensure access to the admin route is granted upon successful login.

2	Test Admin Access with Invalid Credentials	Invalid username, password	Verify that access to the admin route is denied with appropriate error message.
3	Test Admin Form Submission (POST)	Form data with valid inputs	Confirm that form submission adds staff data to the database and redirects to the admin route.
4	Test Admin Form Submission with Invalid Inputs	Form data with missing or invalid fields	Validate that the system rejects the submission with appropriate error messages.
5	Test Admin Staff Deletion (POST)	Staff ID to be deleted	Ensure that deleting a staff member removes associated records and updates the view accordingly.

Table 5.5.1 Test Case for Admin Access and Operations

2) User Access and Operations:

Sr. No.	Test Case Description	Test Values	Expected Results
1	Test User Access with Valid Credentials	Valid username, password	Ensure access to the user route is granted upon successful login.
2	Test User Access with Invalid Credentials	Invalid username, password	Verify that access to the user route is denied with appropriate error message.
3	Test User Profile Display	Valid user data	Confirm that the user's profile and attendance data are displayed correctly on the user page.
4	Test User Attendance Data Display	Mock attendance data	Validate that attendance data for the user is fetched and displayed accurately.
5	Test User Logout (POST)	Logout request	Ensure logging out redirects to the index

			page and removes session data.
--	--	--	--------------------------------

Table 5.5.2 Test Case for User Access and Operations

3) Staff Access and Operations:

Sr. No.	Test Case Description	Test Values	Expected Results
1	Test Staff Access with Valid Credentials	Valid username, password	Ensure access to the staff route is granted upon successful login.
2	Test Staff Access with Invalid Credentials	Invalid username, password	Verify that access to the staff route is denied with appropriate error message.
3	Test Staff Form Submission (POST)	Form data with valid inputs	Confirm that form submission adds a new user to the database and redirects to the staff route.
4	Test Staff Form Submission with Invalid Inputs	Form data with missing or invalid fields	Validate that the system rejects the submission with appropriate error messages.
5	Test Staff Attendance Data Display	Mock attendance data	Ensure that attendance data for all users is fetched and displayed correctly on the staff page.

Table 5.5.3 Test Case for Staff Access and Operations

4) Face recognition:

Sr. No.	Test Case Description	Test Values	Expected Results
1	Test Face Recognition with Recognized Face	Valid image data of a recognized face	Confirm that the system records attendance for the recognized user and updates the database accordingly.

2	Test Face Recognition with Unrecognized Face	Valid image data of an unrecognized face	Ensure that the system handles unrecognized faces appropriately with no attendance recorded.
3	Test Face Recognition with Invalid Image Data	Invalid or corrupted image data	Validate that the system gracefully handles errors during image processing with appropriate error messages.

Table 5.5.4 Test Case for Face Recognition

5) Authorization Functions:

Sr. No.	Test Case Description	Test Values	Expected Results
1	Test Authorization for Admin Functions	Mock admin session data	Confirm that only users with admin privileges can access admin-specific routes and functions.
2	Test Authorization for User Functions	Mock user session data	Ensure that only authenticated users can access user-specific routes and functions.
3	Test Authorization for Staff Functions	Mock staff session data	Validate that only staff members can access staff-specific routes and functions.

Table 5.5.5 Test Case for Authorization Functions

Chapter 6

Results and Discussion

6.1 Test Case Reports

1) Admin Access and Operations:

Sr. No.	Test Case Description	Test Values	Expected Results	Actual Results	Test Case Reports
1	Test Admin Access with Valid Credentials	Valid username, password	Ensure access to the admin route is granted upon successful login.	Admin login successful. Access to admin route granted.	All admin access tests passed successfully.
2	Test Admin Access with Invalid Credentials	Invalid username, password	Verify that access to the admin route is denied with appropriate error message.	Admin login failed with invalid credentials. Access denied as expected.	Admin login functionality tested and verified.
3	Test Admin Form Submission (POST)	Form data with valid inputs	Confirm that form submission adds staff data to the database and redirects to the admin route.	Form submitted successfully. Staff data added to the database. Redirected to admin route after submission.	Admin form submission functionality verified.

4	Test Admin Form Submission with Invalid Inputs	Form data with missing or invalid fields	Validate that the system rejects the submission with appropriate error messages.	Form submission failed due to missing/invalid inputs. Error messages displayed as expected.	Admin form validation tested and validated.
5	Test Admin Staff Deletion (POST)	Staff ID to be deleted	Ensure that deleting a staff member removes associated records and updates the view accordingly.	Staff member deleted successfully. Associated records removed. View updated after deletion.	Staff deletion functionality tested and confirmed.

Table 6.1.1 Test Case Report for Admin Access and Operations

2) User Access and Operations:

Sr. No.	Test Case Description	Test Values	Expected Results	Actual Results	Test Case Reports
1	Test User Access with Valid Credentials	Valid username, password	Ensure access to the user route is granted upon successful login.	User login successful. Access to user route granted.	All user access tests passed successfully.
2	Test User Access with Invalid Credentials	Invalid username, password	Verify that access to the user route is denied with appropriate error message.	User login failed with invalid credentials. Access denied as expected.	User login functionality tested and verified.

3	Test User Profile Display	Valid user data	Confirm that the user's profile and attendance data are displayed correctly on the user page.	User profile and attendance data displayed accurately on the user page.	User profile display functionality validated.
4	Test User Attendance Data Display	Mock attendance data	Validate that attendance data for the user is fetched and displayed accurately.	Attendance data fetched and displayed correctly for the user.	User attendance data display functionality verified.
5	Test User Logout (POST)	Logout request	Ensure logging out redirects to the index page and removes session data.	User logout successful. Redirected to index page. Session data cleared.	User logout functionality tested and confirmed.

Table 6.1.2 Test Case Report for Admin Access and Operations

3) Staff Access and Operations:

Sr. No.	Test Case Description	Test Values	Expected Results	Actual Results	Test Case Reports
1	Test Staff Access with Valid Credentials	Valid username, password	Ensure access to the staff route is granted upon successful login.	Staff login successful. Access to staff route granted.	All staff access tests passed successfully.
2	Test Staff Access with Invalid Credentials	Invalid username, password	Verify that access to the staff route is denied with appropriate error message.	Staff login failed with invalid credentials.	Staff login functionality tested and verified.

				Access denied as expected.	
3	Test Staff Form Submission (POST)	Form data with valid inputs	Confirm that form submission adds a new user to the database and redirects to the staff route.	Form submitted successfully. New user added to the database. Redirected to staff route after submission.	Staff form submission functionality validated.
4	Test Staff Form Submission with Invalid Inputs	Form data with missing or invalid fields	Validate that the system rejects the submission with appropriate error messages.	Form submission failed due to missing/invalid inputs. Error messages displayed as expected.	Staff form validation tested and confirmed.
5	Test Staff Attendance Data Display	Mock attendance data	Ensure that attendance data for all users is fetched and displayed correctly on the staff page.	Attendance data fetched and displayed accurately for all users.	Staff attendance data display functionality verified.

Table 6.1.3 Test Case Report for Staff Access and Operations

4) Face Recognition:

Sr. No.	Test Case Description	Test Values	Expected Results	Actual Results	Test Case Reports
1	Test Face Recognition with Recognized Face	Valid image data of a recognized face	Confirm that the system records attendance for the recognized user and updates the database accordingly.	Face recognized. Attendance recorded for the recognized user. Database updated.	Face recognition functionality tested and confirmed.
2	Test Face Recognition with Unrecognized Face	Valid image data of an unrecognized face	Ensure that the system handles unrecognized faces appropriately with no attendance recorded.	Face not recognized. No attendance recorded.	Face recognition with unrecognized face validated.
3	Test Face Recognition with Invalid Image Data	Invalid or corrupted image data	Validate that the system gracefully handles errors during image processing with appropriate error messages.	Error processing image data. Expected behavior for invalid image.	Face recognition error handling tested and verified.

Table 6.1.4 Test Case Report for Face Recognition

5) Authorization Functions:

Sr. No.	Test Case Description	Test Values	Expected Results	Actual Results	Test Case Reports
1	Test Admin Access	Admin credentials	Ensure that only users with admin privileges can access admin-specific routes and functionalities.	Admin credentials authenticated. Admin-only routes accessible as expected.	Admin access functionality validated.
2	Test User Access	User credentials	Validate that regular users can access user-specific routes and functionalities upon successful authentication.	User credentials authenticated. User-specific routes accessible as expected.	User access functionality verified.
3	Test Staff Access	Staff credentials	Confirm that staff members can access staff-specific routes and functionalities after successful authentication.	Staff credentials authenticated. Staff-only routes accessible as expected.	Staff access functionality tested and confirmed.

Table 6.1.5 Test Case Report for Authorization Functions

6.2 Screenshots

- Login Page

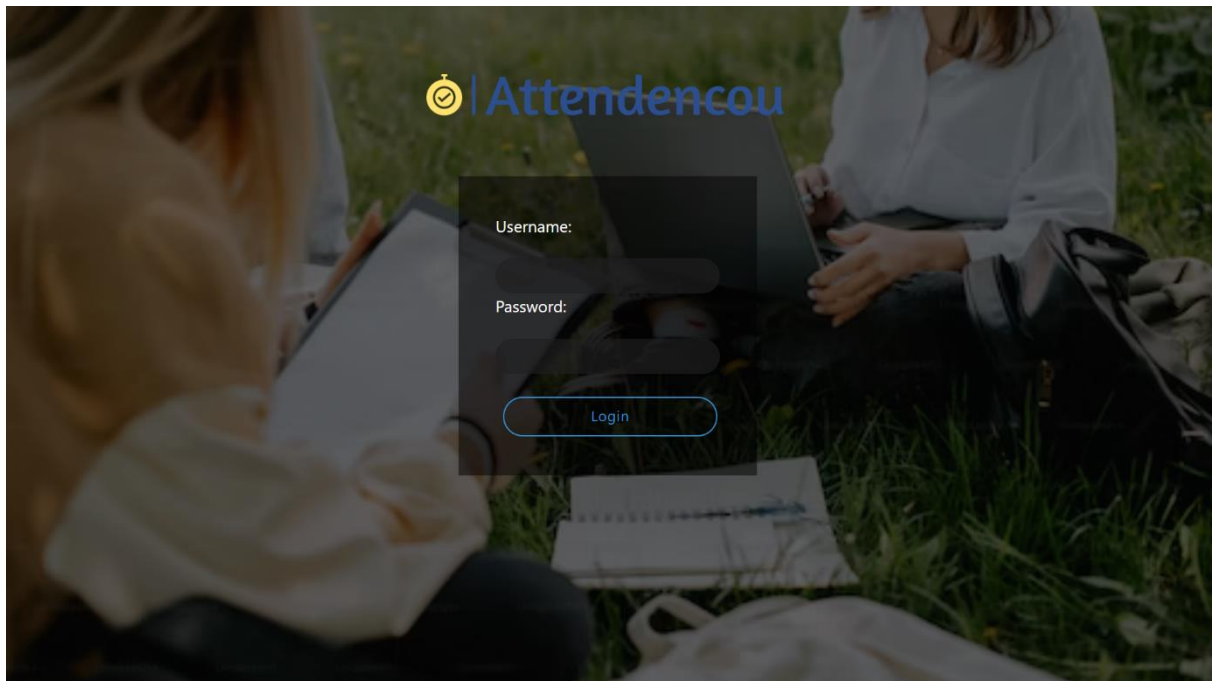


Fig 6.2.1 Login Page

- Admin Page

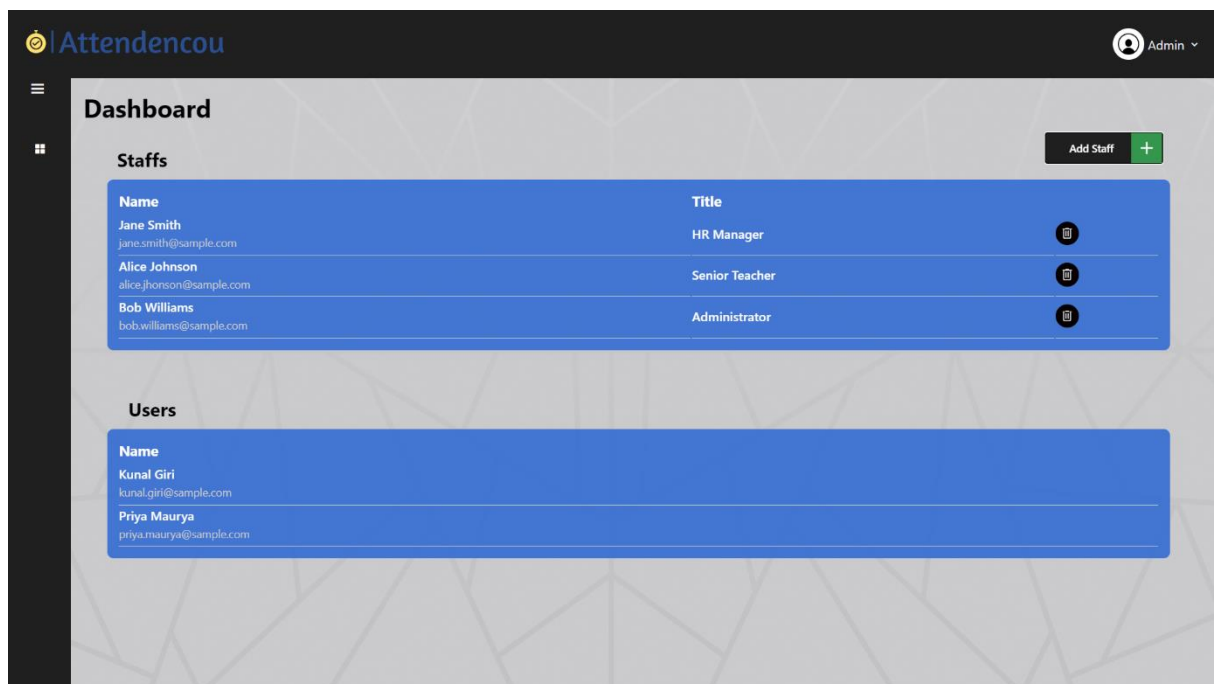


Fig 6.2.2.1 Admin Page

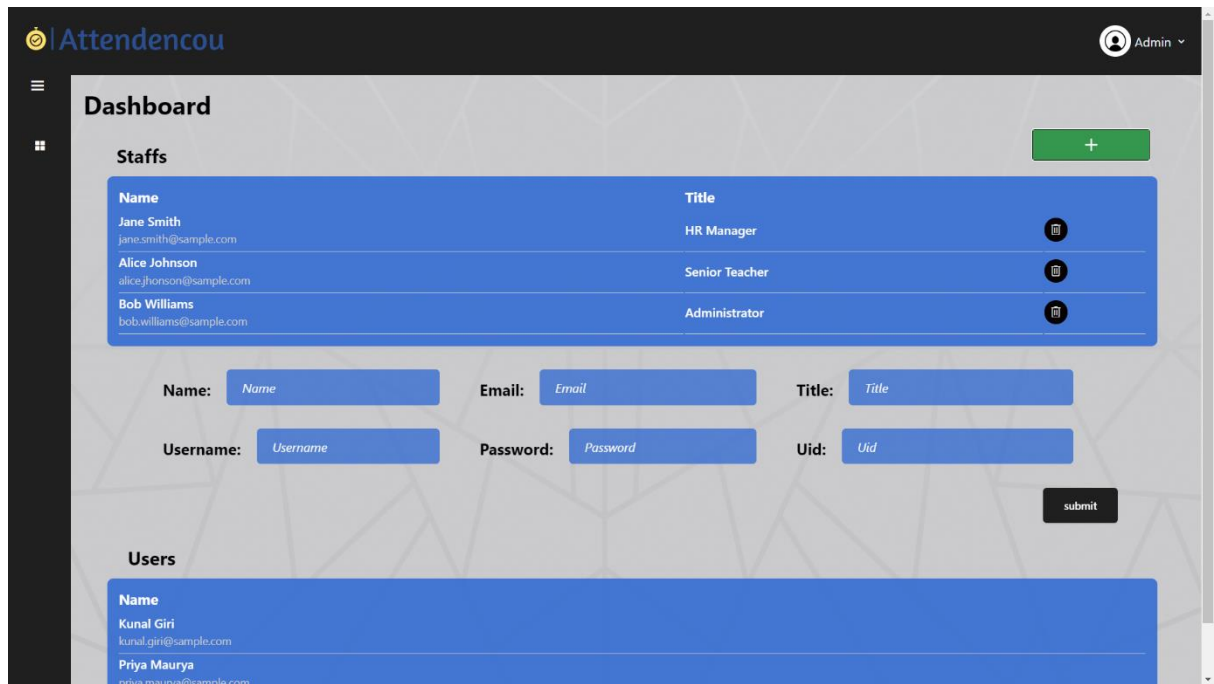


Fig 6.2.2.2 Admin Page adding Staff

- Staff Page

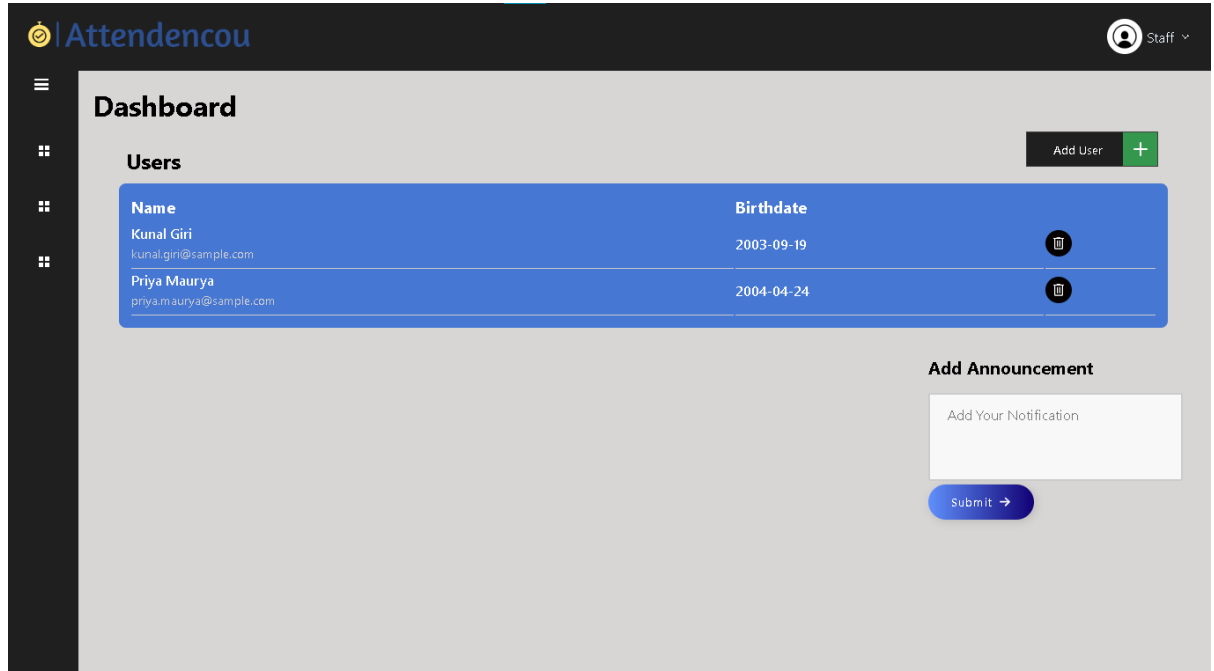


Fig 6.2.3.1 Staff Page

Staff

Dashboard

Users

Add User

Name	Birthdate	
Kunal Giri kunal.giri@sample.com	2003-09-19	
Priya Maurya priya.maurya@sample.com	2004-04-24	

Name:

Email:

Birthdate:

Username:

Password:

Uid:

submit

Add Announcement

Add Your Notification

Submit

Fig 6.2.3.2 Staff Page adding user

Staff

View Attendance

Filter by Status: All

Filter by Name: All

Filter by Date: All

Name	Date	Time	Status
Kunal Giri	01-03-2024	00:51:05	Present
Priya Maurya	01-03-2024	-	Absent
Kunal Giri	02-03-2024	-	Absent
Priya Maurya	02-03-2024	-	Absent
Kunal Giri	03-03-2024	-	Absent
Priya Maurya	03-03-2024	-	Absent
Kunal Giri	04-03-2024	-	Absent
Priya Maurya	04-03-2024	-	Absent
Kunal Giri	05-03-2024	-	Absent
Priya Maurya	05-03-2024	-	Absent
Kunal Giri	06-03-2024	-	Absent
Priya Maurya	06-03-2024	-	Absent
Kunal Giri	07-03-2024	-	Absent
Priya Maurya	07-03-2024	-	Absent
Kunal Giri	08-03-2024	-	Absent
Priya Maurya	08-03-2024	-	Absent
Kunal Giri	09-03-2024	-	Absent
Priya Maurya	09-03-2024	-	Absent
Kunal Giri	10-03-2024	-	Absent
Priya Maurya	10-03-2024	-	Absent
Kunal Giri	11-03-2024	22:46:40	Present

Fig 6.2.3.3 Staff Page (View Attendance)

Name	Date	Status	
Kunal Giri	01-03-2024	Present	✓
Priya Maurya	01-03-2024	Absent	■
Kunal Giri	02-03-2024	Absent	■
Priya Maurya	02-03-2024	Absent	■
Kunal Giri	03-03-2024	Absent	■
Priya Maurya	03-03-2024	Absent	■
Kunal Giri	04-03-2024	Absent	■
Priya Maurya	04-03-2024	Absent	■
Kunal Giri	05-03-2024	Absent	■
Priya Maurya	05-03-2024	Absent	■
Kunal Giri	06-03-2024	Absent	■
Priya Maurya	06-03-2024	Absent	■
Kunal Giri	07-03-2024	Absent	■
Priya Maurya	07-03-2024	Absent	■
Kunal Giri	08-03-2024	Absent	■
Priya Maurya	08-03-2024	Absent	■
Kunal Giri	09-03-2024	Absent	■
Priya Maurya	09-03-2024	Absent	■
Kunal Giri	10-03-2024	Absent	■
Priya Maurya	10-03-2024	Absent	■
Kunal Giri	11-03-2024	Present	✓
Priya Maurya	11-03-2024	Present	✓

Fig 6.2.3.3 Staff Page (Add Attendance)

- User Page

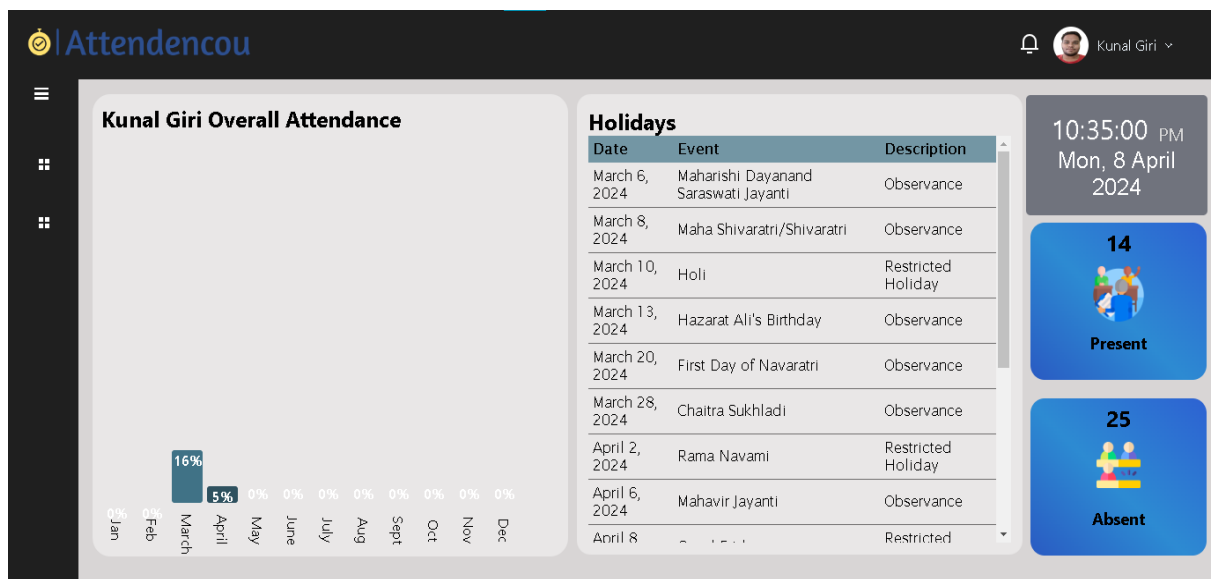


Fig 6.2.4.1 User Page (Dashboard)



Date	Time	Status
01-03-2024	-	Absent
02-03-2024	-	Absent
03-03-2024	-	Absent
04-03-2024	-	Absent
05-03-2024	-	Absent
06-03-2024	-	Absent
07-03-2024	-	Absent
08-03-2024	-	Absent
09-03-2024	-	Absent
10-03-2024	-	Absent
11-03-2024	22:46:40	Present
12-03-2024	-	Absent
13-03-2024	22:51:59	Present
14-03-2024	23:18:05	Present
15-03-2024	23:18:05	Present
16-03-2024	-	Absent
17-03-2024	-	Absent
18-03-2024	-	Absent
19-03-2024	-	Absent
20-03-2024	-	Absent
21-03-2024	-	Absent

Fig 6.2.4.2 User Page (View Attendance)

- Face Recognition Page

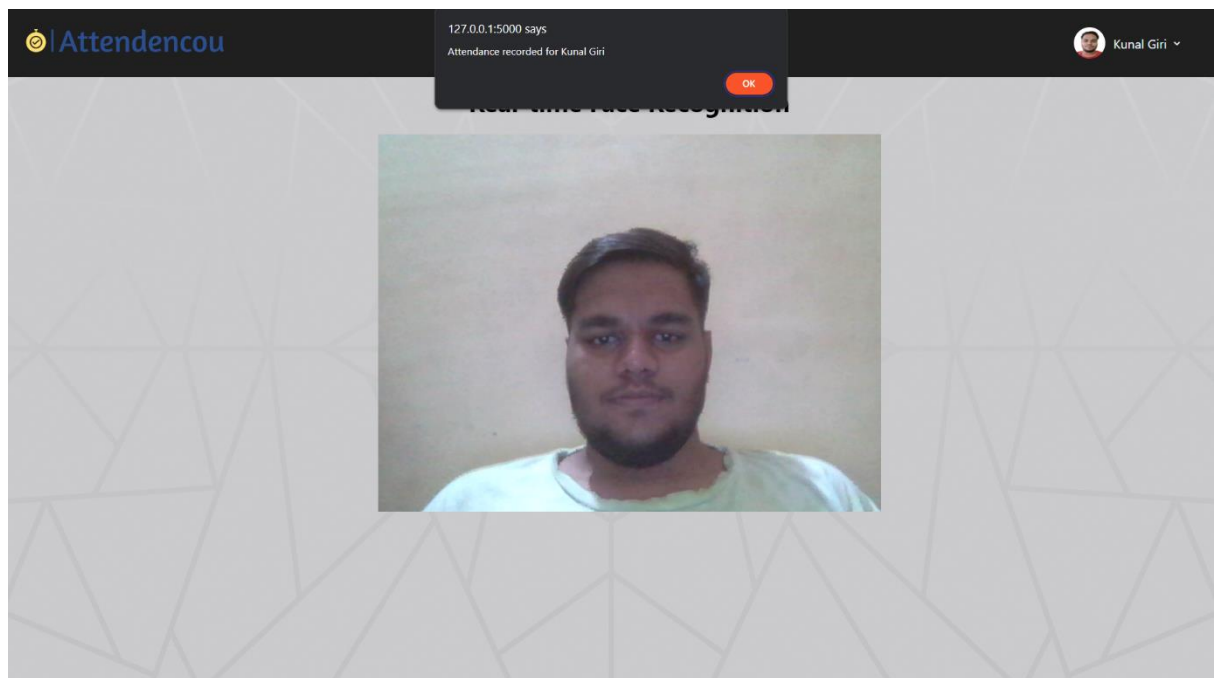


Fig 6.2.3.1 Face Recognition Page

Chapter 7

Conclusion

7.1 Conclusion

The implementation of a Face Recognition-based Attendance Management System marks a significant departure from traditional attendance tracking methods, offering a modern and efficient solution to the challenges of manual record-keeping.

This system leverages real-time face recognition technology to streamline attendance marking, providing immediate and accurate records without the need for manual input. By automating the attendance process, organizations can save valuable time, reduce administrative costs, and enhance overall operational efficiency.

7.2 Limitations of the System

- One limitation of our system is the accuracy of face recognition, which can be influenced by factors like lighting conditions, facial expressions, and image quality. We aim to continuously refine our algorithms to improve reliability and reduce instances of false positives or negatives.
- Currently, our system does not allow staff to upload user's photos directly while adding new users. This is a limitation that we plan to address in the future to make it easier to add new user's images and enhance system flexibility.
- Another limitation is our system's dependency on stable internet connectivity. Weak or unreliable internet connections can impact performance.

7.3 Future Scope

- Emphasizing ongoing refinement through continuous improvement and feedback mechanisms.

- Making the attendance system fully customizable, catering to the needs of organizations at any level.
- Implementation of editing features for users and UI enhancements to improve usability, ensuring a seamless experience.
- Full customization of the website according to customer needs, enhancing flexibility and adaptability.
- Expansion of the attendance management system to incorporate diverse recording options such as QR code scanning, biometric authentication, geolocation tracking, NFC tags, and Iris scan.

SUMMARY

The implementation of an Attendance Management System based on face recognition represents a significant advancement in modernizing attendance tracking processes. By leveraging real-time face recognition technology, this system offers a seamless and efficient method to mark attendance instantly.

Key features of this system include real-time attendance marking, eliminating the need for manual record-keeping, and reducing administrative overhead. Users benefit from improved accuracy, time savings, and reduced costs associated with traditional attendance methods.

Utilizing a technology stack consisting of HTML, CSS, Flask, JavaScript, and MongoDB Atlas, this project integrates web-based front-end interfaces with a robust back-end infrastructure for data management and processing.

During development, challenges such as integrating HTML and Python were overcome by leveraging Flask, a web framework for Python. Additionally, managing attendance data storage and retrieval was addressed through effective database management practices.

In summary, the Face Recognition-based Attendance Management System offers a transformative solution that streamlines attendance processes, enhances efficiency, and provides a foundation for scalable and customizable attendance management across different organizational settings.

GLOSSARY

LBP - Local binary patterns

UI – User Interface

ML – Machine Learning

ERD – Entity Relationship Diagram

SMS – Short Message Service

REFERENCE

<https://chat.openai.com/>

<https://opencv.org/>

<https://github.com/>

<https://flask.palletsprojects.com/en/3.0.x/>

<https://www.mongodb.com/atlas/database>

<https://boxicons.com/>

<https://uiverse.io/>

<https://www.w3schools.com/>

<https://www.geeksforgeeks.org/>

<https://stackoverflow.com/>

<https://www.programiz.com/>