1. INTRODUCTION

The Attendance Management System is a web-based application designed to enable users to view and manage the attendance of the students. Unlike traditional systems that require manual data entry and record-keeping, the proposed system automates these processes by leveraging a database management system that maintains relationships between tables and handles data efficiently.

This project responds to the growing demand for a streamlined and user-friendly platform tailored to meet modernise college attendance systems. It explores how technology can be integrated with basis to enhance customer experience, improve accessibility, and optimize operational efficiency.

The system is developed using html for the frontend, Express.js for the backend, and MongoDB as the database. It is structured into three main modules:

- ❖ Staff
- ❖ Admin
- ❖ Student

2. SYSTEM STUDY

2.1 EXISTING SYSTEM

Traditional attendance management methods have several limitations, such as manual record-keeping, risk of data loss, time-consuming processes, the need for physical presence, and overall inconvenience. Additional limitations include difficulty in generating accurate reports and challenges in maintaining long-term data records.

Disadvantages of the existing system include:

- Limited control for other users
- Insufficient student descriptions or missing key details

2.2 PROPOSED SYSTEM

The proposed system is a web-based solution that removes the need for manual data entry by automating key processes. With the help of a database management system, it maintains proper relationships between tables and handles operations like inquiries and product management more efficiently.

Advantages of the proposed system:

- Unified Platform: Manage student, staff, and course attendance records in one system.
- Real-Time Updates: Instantly record and update attendance data without delays.
- Robust Data Management: MongoDB database with efficient CRUD operations.
- User-Friendly Design: Clean, responsive UI for smooth navigation across devices.
- Role-Based Access: Separate modules and permissions for admin, staff, and students.
- Quick Retrieval: Built-in search to quickly find attendance records or student details.
- Data Security: Secure authentication and authorization to protect sensitive information.

2.3 Problem Definition and Project Description

The **College Attendance Management System** is designed to allow students, staff, and administrators to manage and monitor attendance records through an online platform. The system organizes data under various categories such as departments, courses, and class sections, enabling users to access and manage information conveniently based on their roles and requirements.

This platform is user-friendly, secure, and reliable. It incorporates essential features for students, staff, and administrators, ensuring efficient system operation and data protection. The system provides a secure administrative panel where only the admin has access to manage user accounts, assign classes, and generate reports.

The software is divided into several modules to ensure smooth and organized functioning.
These modules include:

Admin Module

- View all attendance records and reports
- Add, edit, or remove student, staff, and course details
- Assign classes and subjects to staff members
- Update and manage attendance statuses
- View feedback or requests from students and staff

Staff Module

- Browse Student Attendance
- Manage and update Student Atendance
- Browse Student Details
- Browse class details

Student Module

- View Student Attendance

3. SYSTEM ANALYSIS

3.1 REQUIREMENTS SPECIFICATION

Hardware Requirements

Processor	:	Intel Core i3 or higher
RAM	:	Minimum 4 GB (8 GB recommended)
Hard Disk	:	Minimum 40 GB free space
Keyboard	:	Standard 104-keys Keyboard
Mouse	:	Optical Mouse

Software Requirements

Operating System	:	Windows 10 / 11 (64-bit) or equivalent
Local Server	:	Node.js with Express framework
Database	:	MongoDB (NoSQL database)
Frontend	:	(HTML, CSS, JavaScript)
Backend	:	Node.js with Express
Database Connectivity	:	Mongoose
Browser	:	Google Chrome / Firefox
Other Tools	:	Visual Studio Code, Postman.

3.2 FEASIBILITY STUDY

A feasibility analysis is conducted to determine the viability of either enhancing the existing manual furniture store operations or developing a completely new automated system. It helps provide an overview of the current challenges and gives a rough estimate of whether a practical solution exists.

There are three major aspects covered in the feasibility study during the preliminary investigation:

- Operational Feasibility
- Technical Feasibility
- Economic Feasibility

TECHNICAL FEASIBILITY

This involves evaluating whether the required technology for the system exists, how complex it is to build, and whether the development team has the necessary experience. The assessment is based on an outline of system requirements in terms of inputs, processes, outputs, fields, modules, and procedures. This evaluation considers data flow, frequency of updates, and expected trends to judge whether the system will perform efficiently.

For the proposed College Attendance Management System, the technical feasibility covers both the hardware and software environment. It ensures that the technology used is current and compatible with industry standards. The system will be developed using technologies like HTML for frontend, Node.js for backend, and MongoDB for database, which are modern, scalable, and widely adopted.

The system can run efficiently on basic configurations. A minimum of Intel i3 processor, 4 GB RAM, and 40 GB hard disk space is required for development and deployment. Software requirements include a 64-bit Operating System, a local development server (like Node.js), and MongoDB database.

Thus, the proposed system is technically feasible, given that it relies on proven and wellsupported technologies.

OPERATIONAL FEASIBILITY

Operational feasibility determines how effectively the proposed system addresses the current problems and meets the functional requirements identified during analysis. It also evaluates the extent to which the system will be accepted and adopted by the organization.

In this case, the College Attendance Management System aims to simplify product management, streamline customer orders, and enable online access to furniture listings, eliminating the constraints of in-store shopping. It also allows admin-level operations such as user management, inventory updates.

The system is designed with a user-friendly interface, making it easy to learn and operate for both store employees and customers. Minimal training will be required, especially for users familiar with basic internet and e-commerce usage. Hence, the system is operationally feasible.

ECONOMIC FEASIBILITY

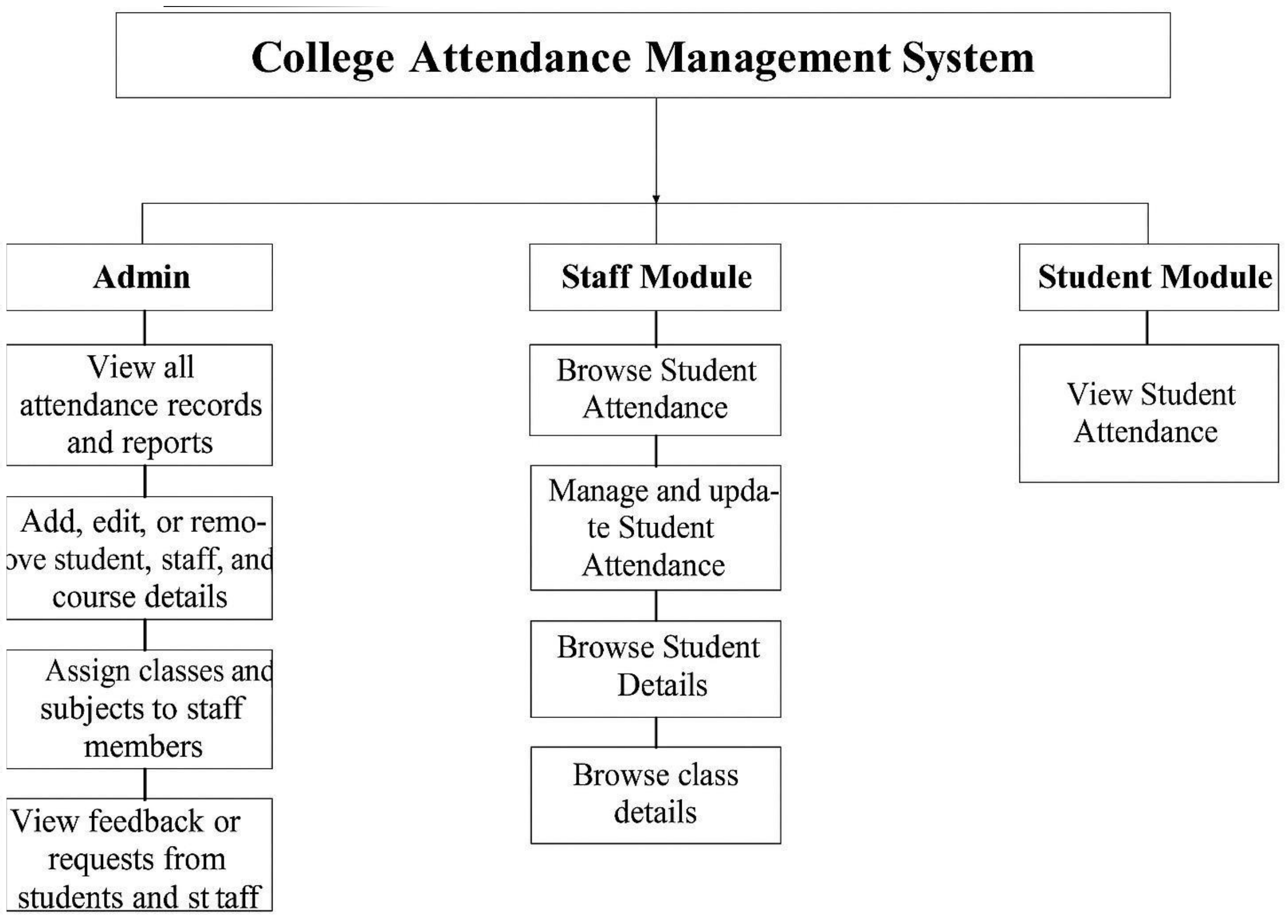
Economic feasibility is evaluated through a cost/benefit analysis to determine whether the expected benefits outweigh the development and implementation costs.

The proposed system is a long-term investment that will reduce manual efforts, minimize errors, improve customer satisfaction, and expand business reach through online visibility. Maintenance and updates will be minimal after initial deployment.

Given the expected gains in efficiency, customer engagement, and brand value, the system is considered economically feasible for development and deployment.

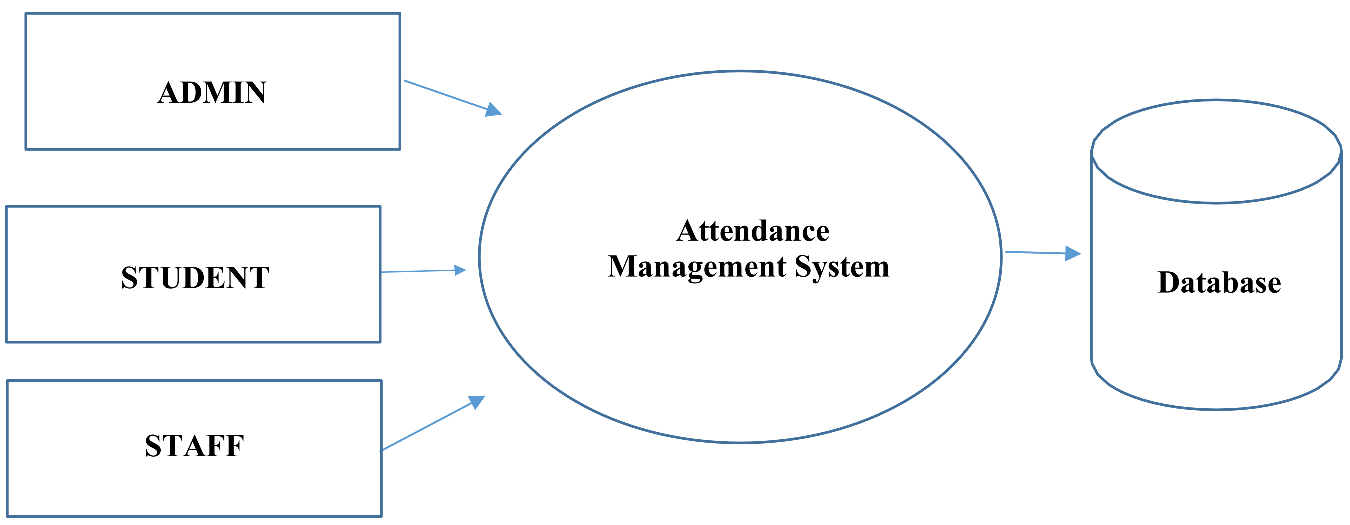
4. SYSTEM DESIGN

4.1 ARCHITECTURE DESIGN

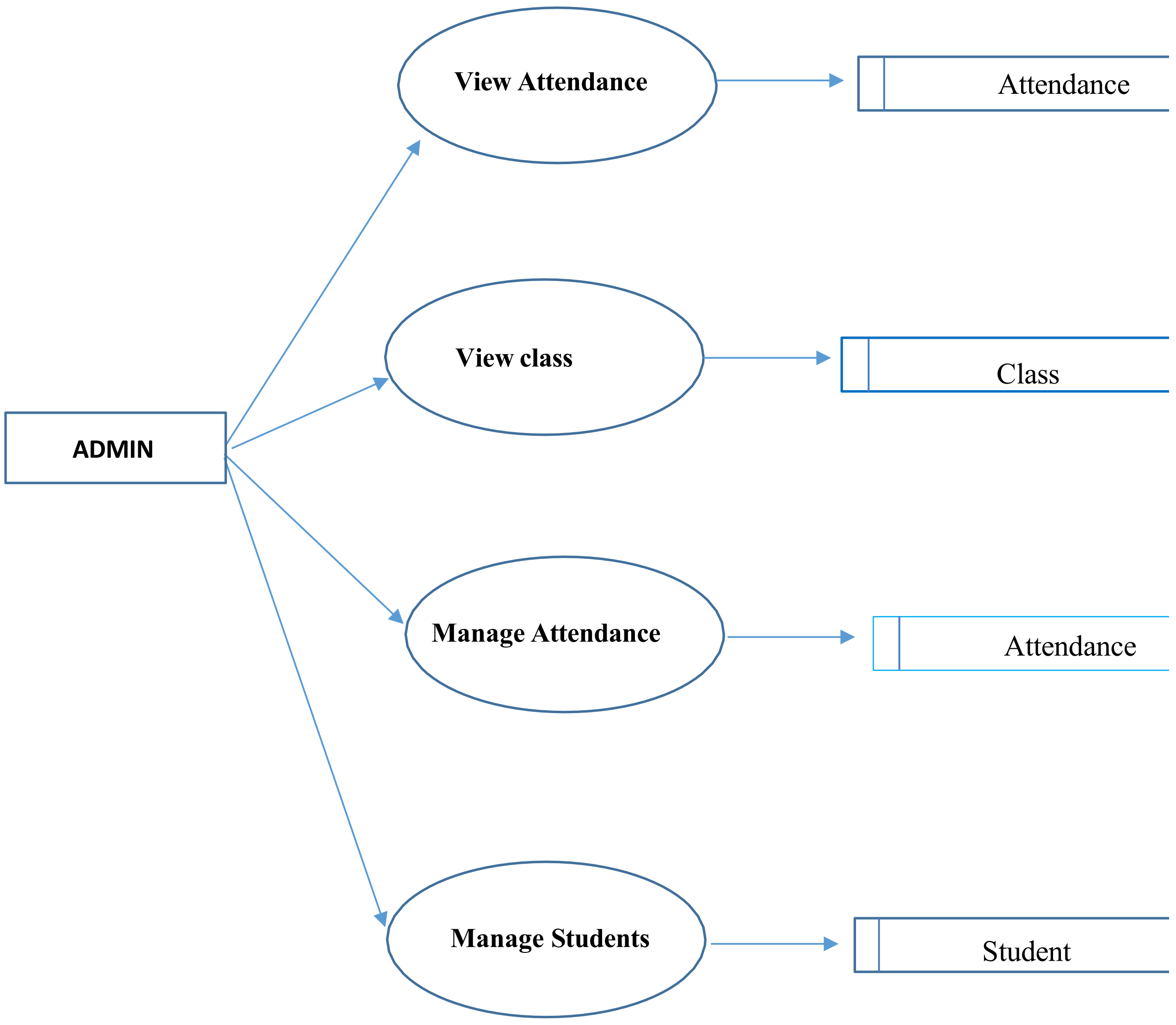

College Attendance Management System
Admin
View all attendance records and reports
Add, edit, or remove student, staff, and course details
Assign classes and subjects to staff members
View feedback or requests from students and st taff
Staff Module
Browse Student Attendance
Manage and update Student Attendance
Browse Student Details
Browse class details
Student Module
View Student Attendance

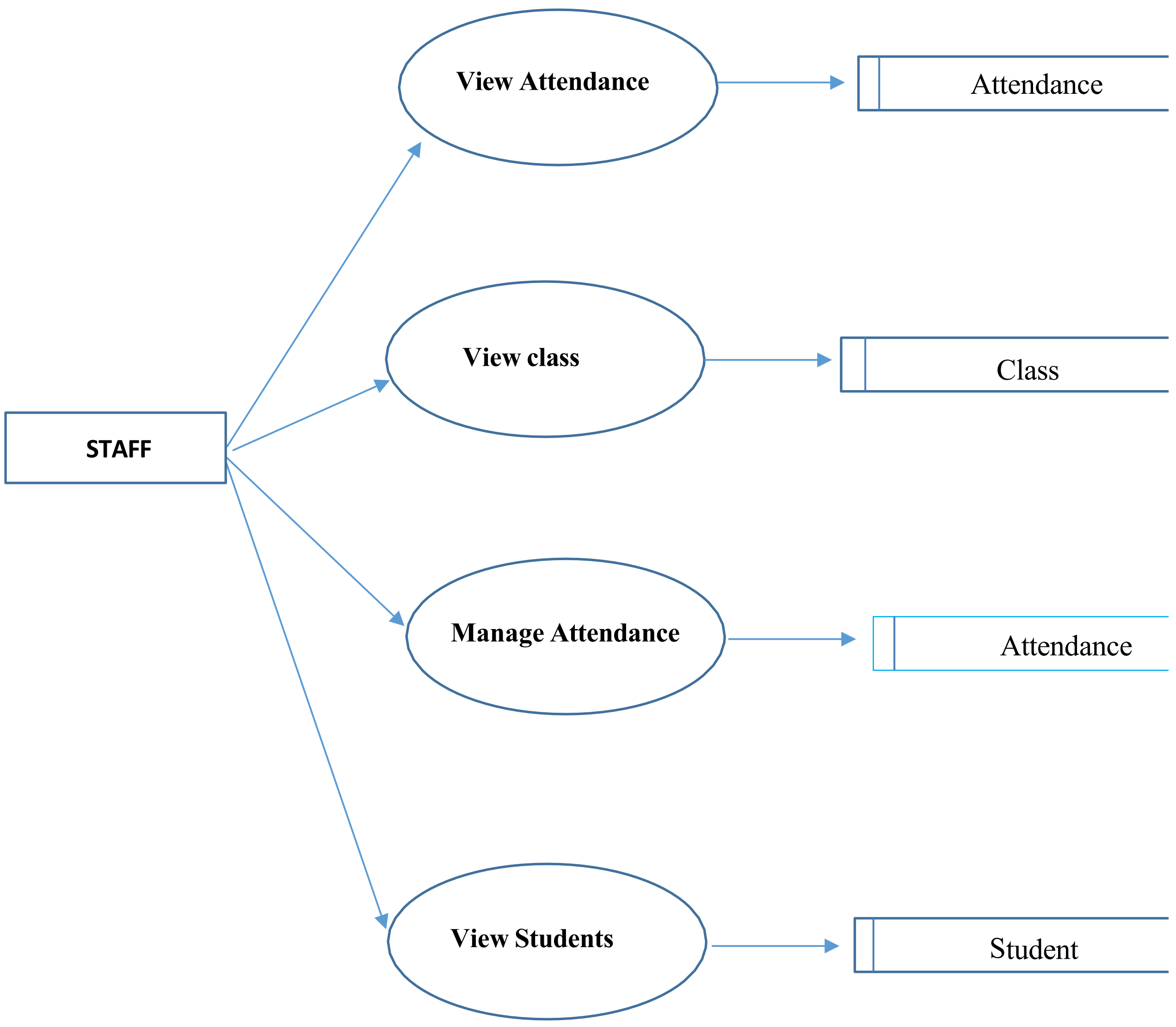
4.2 DATAFLOW DIAGRAM DFD

LEVEL 0:



DFD LEVEL 1:


View Attendance
Attendance
View class
Class
ADMIN
Manage Attendance
Attendance
Manage Students
Student


View Attendance
Attendance
View class
Class
STAFF
Manage Attendance
Attendance
View Students
Student

4.3 DATA DICTIONARY

Collection Name: Admin

Purpose: To store login details of admin

S.No.	Field Name	Data Type	Constraint	Description
1	_id	ObjectId	Primary Key	Unique identifier
2	Admin Name	String	Unique	Admin’s name
3	Email	String	Not null	Admin’s Contact Email
4	Password	String	Not null	Password

Collection Name: Student

Purpose: To Store Student details

S.No.	Field Name	Data Type	Constraint	Description
1	_id	ObjectId	Primary Key	Unique identifier
2	Name	String	Not Null	Student’s Name
3	AdmissionNumber	String	Unique	Student Unique id
4	RollNo	String	Unique	Student id
5	Password	String	Not Null	Student Login password
6	Phone	String	Not Null	Student contact number

7	Address	Object	Not Null	Student address containing street, city, state, and pincode
8	Class	String	Not Null	Student Class

Collection Name: classTeacher

Purpose: Used to store staff details

S.No.	Field Name	Data type	Constraint	Description
1	_id	ObjectId	Primary Key	Unique Identifier
2	Staff id	String	Unique	Staff unique id
3	Name	String	Not Null	Staff Name
4	Password	String	Not Null	Staff Login Password
5	Email	String	Not Null	Staff Contact Email
6	Phone	String	Not Null	Staff Contact Number
7	Address	Object	Not Null	Staff Home Address

Collection Name: attendance

Purpose: To Store attendance details

S.No.	Field Name	Data type	Constraint	Description
1	_id	ObjectId	Primary Key	Unique identifier

2	AdmissionNo	String	Unique	Student unique id
3	RollNo	String	Unique	Student id
4	Class	String	Not Null	Student’s class id
5	Staffid	String	Unique	Staff’s unique id
6	DayValue	String	Not Null	Attendance Date
7	HourValue	String	Not Null	Attendance Hour
8	Student Name	String	Not Null	Student’s name

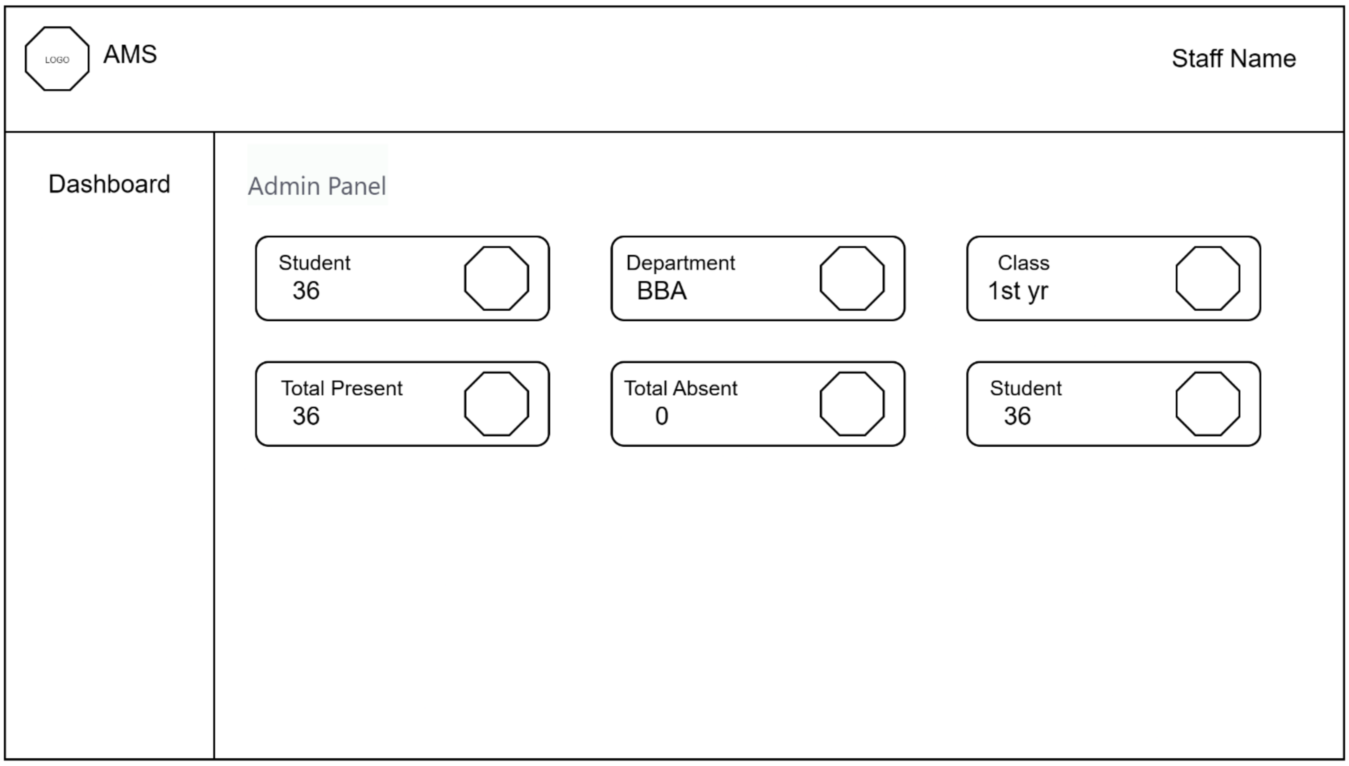
Collection Name: class

Purpose: To Store class details

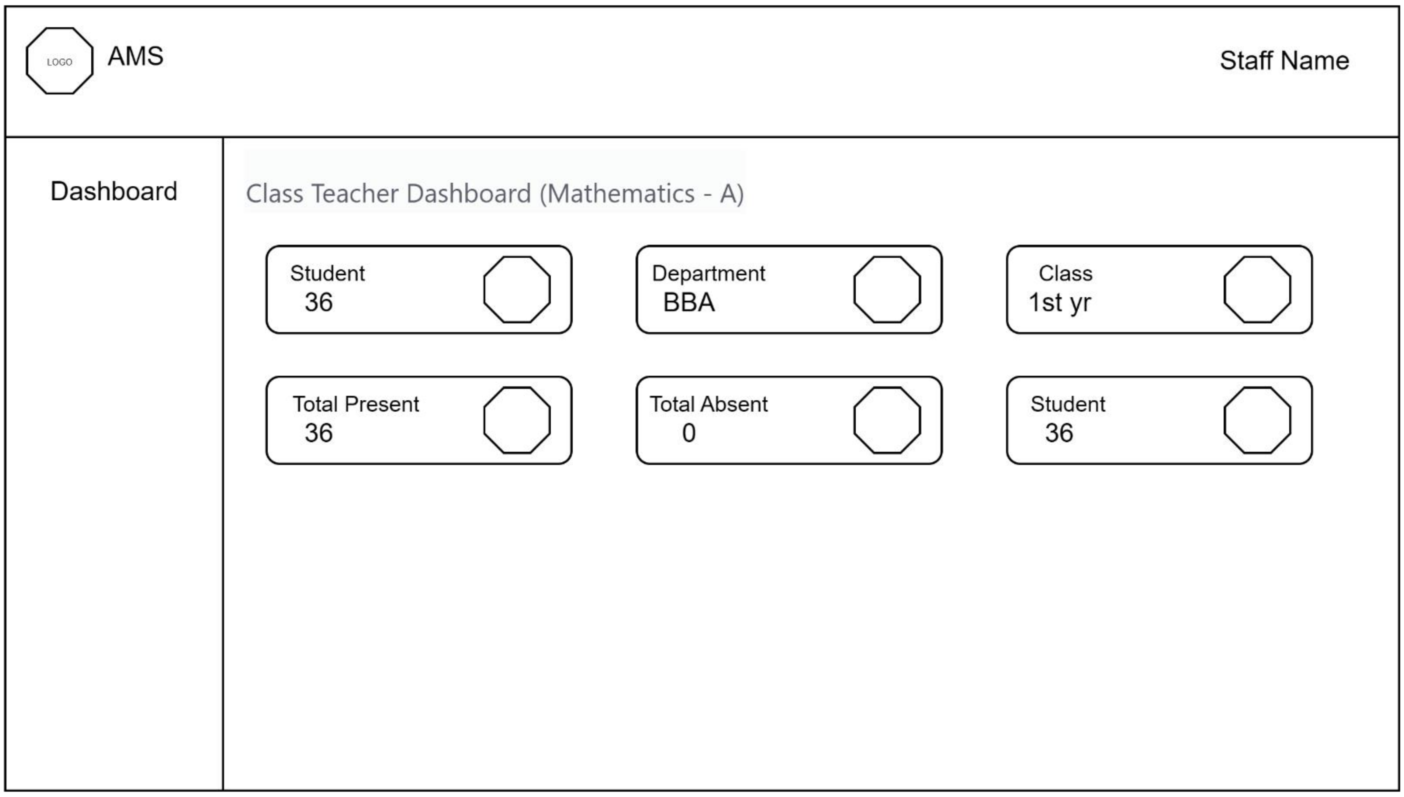
S.No.	Field Name	Data type	Constraint	Description
1	_id	ObjectId	Primary Key	Unique identifier
2	className	String	Unique	Existing Class Name

4.4 USER INTERFACE DESIGN

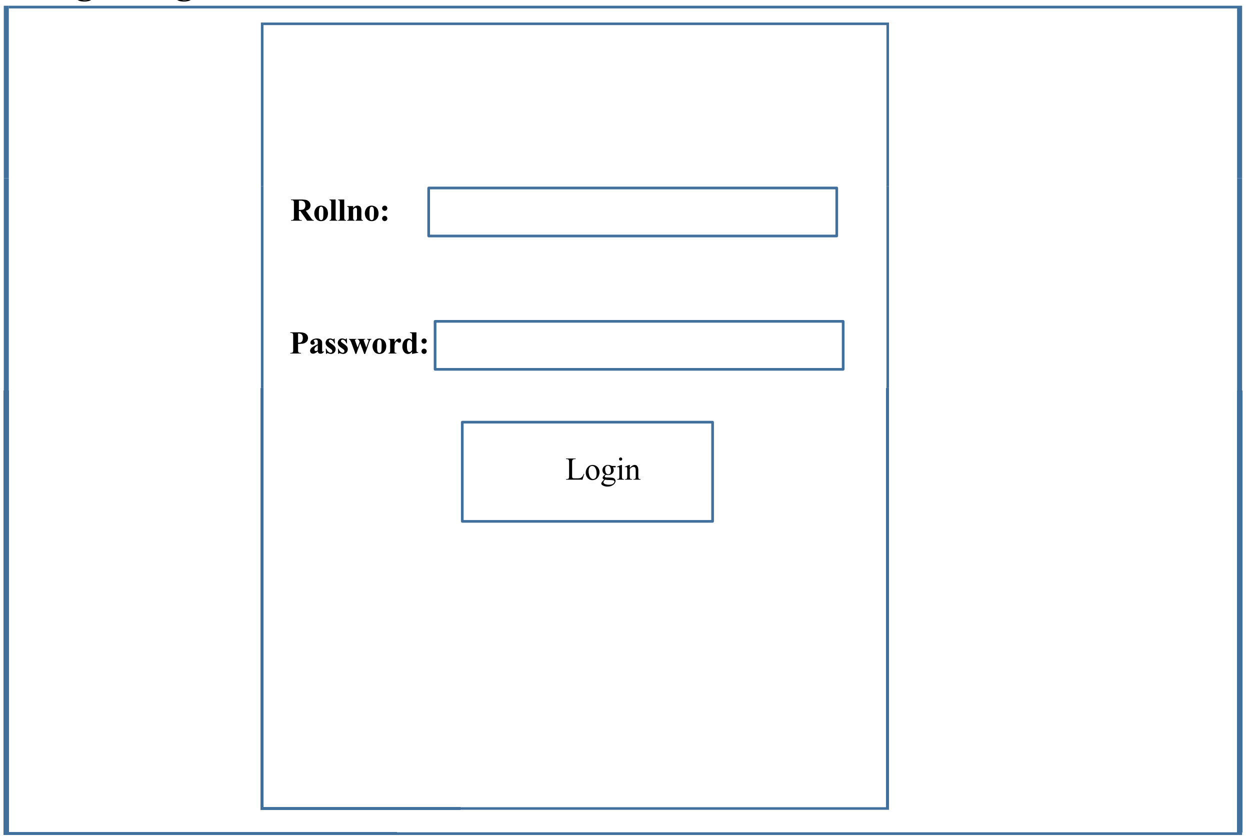
ADMIN DASHBOARD



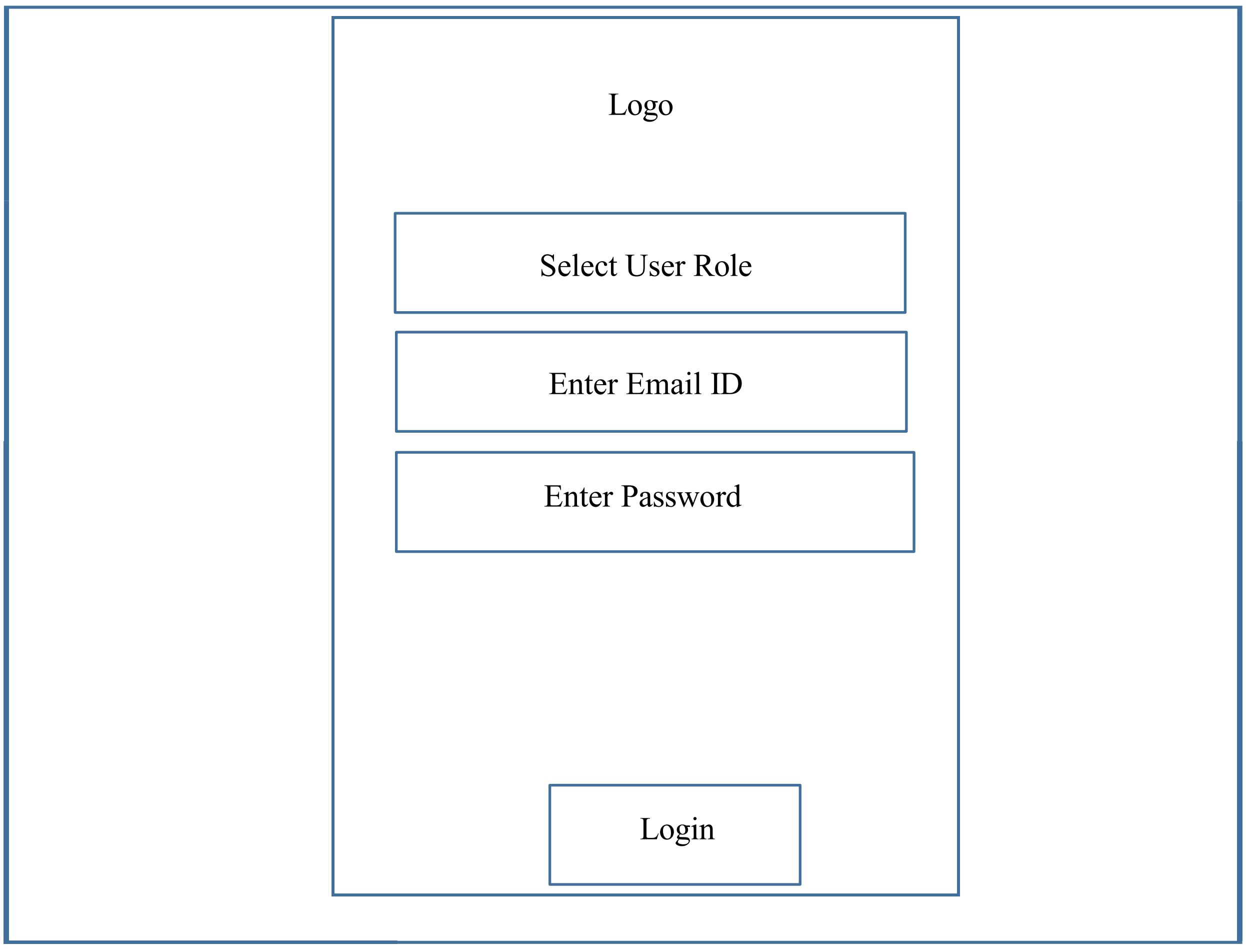
STAFF DASHBOARD



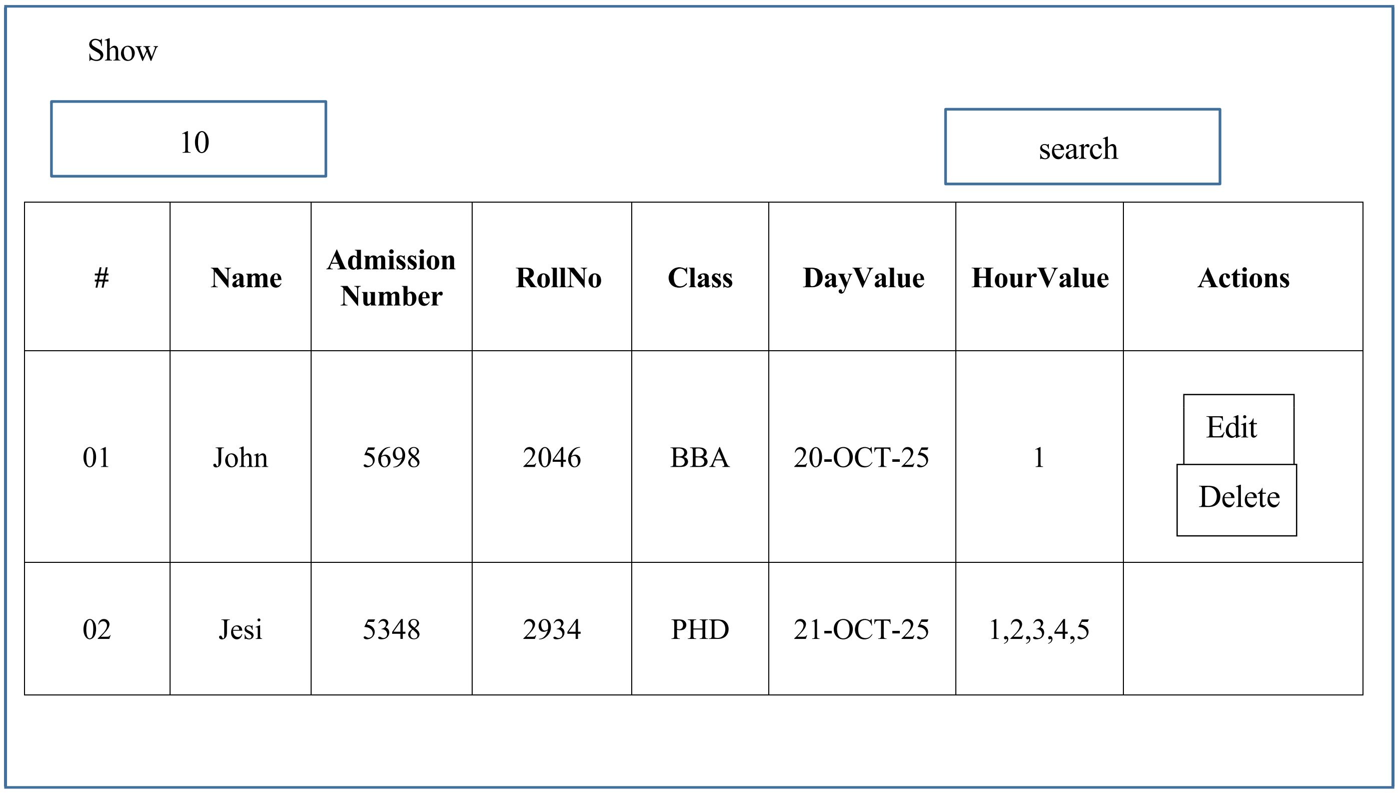
Student Login Page



Staff & Admin Login Page



View Class Attendance Page



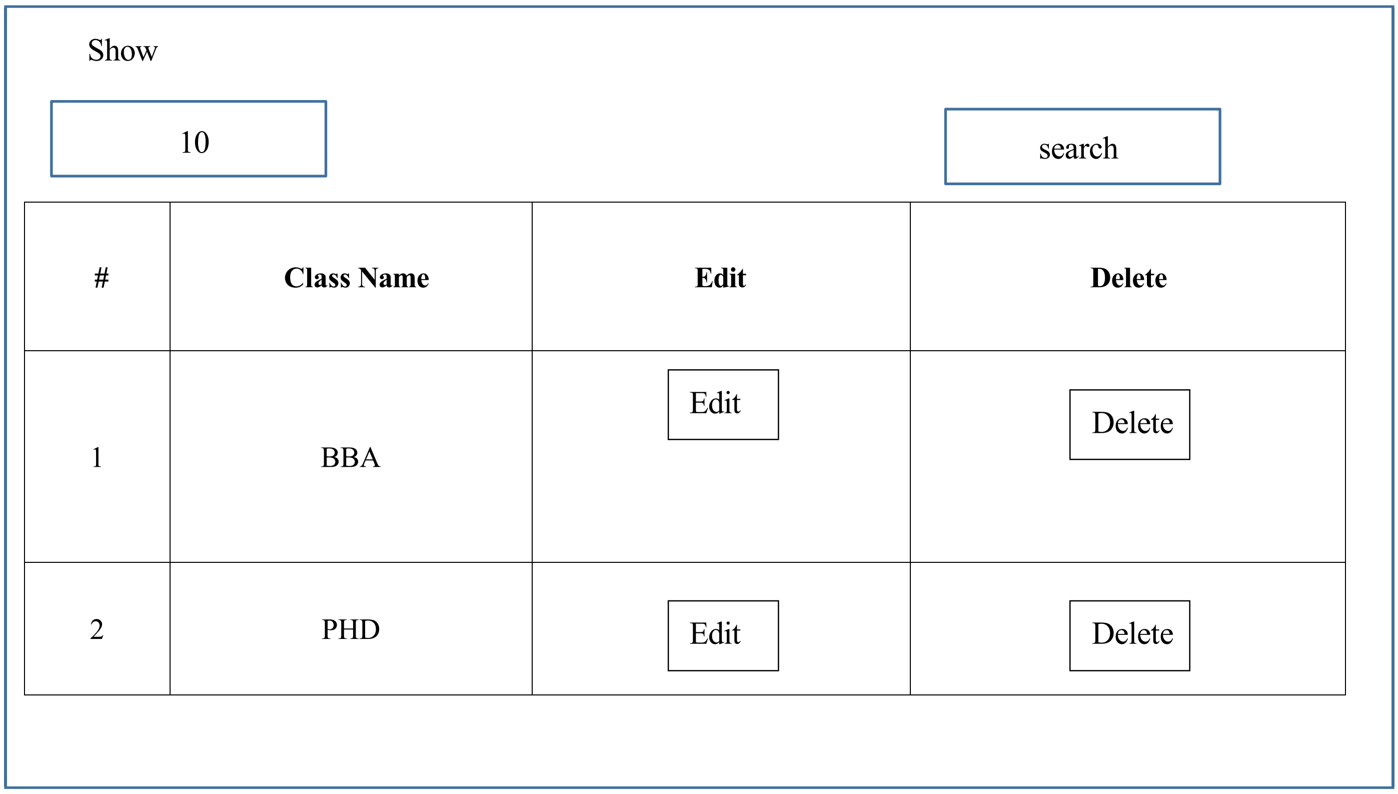
Show

10

search

#	Name	Admission Number	RollNo	Class	DayValue	HourValue	Actions
01	John	5698	2046	BBA	20-OCT-25	1	Edit Delete
02	Jesi	5348	2934	PHD	21-OCT-25	1,2,3,4,5	

All Class Page



Show

10

search

#	Class Name	Edit	Delete
1	BBA	Edit	Delete
2	PHD	Edit	Delete

4.1 NORMALIZATION

Definition

Normalization is the process of organizing data in a database to minimize redundancy and avoid anomalies during **insertion**, **update**, or **deletion** operations. In the context of our **Attendance Management System**, normalization ensures that data like student details, staff details, and class information are stored in separate, well-structured collections and linked using relationships (foreign keys in MongoDB).

By applying normalization, we divide large, unorganized collections into smaller, related collections, such as users, items, orders, and reviews. This avoids data duplication and makes updates more efficient.

FIRST NORMAL FORM (1NF)

For a table (or MongoDB collection) to be in 1NF:

1. **Atomic values only** – Each field must hold a single value (e.g., in users' collection, the phoneNumber field should store only one number, not multiple).
2. **Same data type in a column** – All values in a field should belong to the same type (e.g., rollno in items should always be a Number).
3. **Unique field names** – Each attribute should have a unique name.
4. **Order of data does not matter** – The sequence of records should not affect the data's meaning.

Example in project:

Instead of storing "day": "hour1, hour2" in one field, we create a separate variants array or a new separate collection.

SECOND NORMAL FORM (2NF)

Conditions:

1. Must be in **1NF**.
2. No **Partial Dependency** – Non-key attributes should depend on the whole primary key, not part of it.

Example in project:

In the student collection, if the primary key is (Admission Number, RollNo), then attributes like studentName should not depend only on RollNo— instead, we link to the items collection to fetch product details.

THIRD NORMAL FORM (3NF)

Conditions:

1. Must be in **2NF**.
2. No **Transitive Dependency** – Non-key attributes should not depend on other non-key attributes.

Example in project:

In the student collection, city should not determine postalCode. Instead, we keep city and postal code independent or move location details to addresses subdocument.

Boyce Codd Normal Form (BCNF)

Conditions:

1. Must be in **3NF**.
2. For every functional dependency (X → Y), **X** should be a superkey.

Example in project:

If in the student collection, Admission Number determines both RollNo and Student id, then Admission No should be the superkey to avoid anomalies.