

A FIELD PROJECT REPORT

on

**“Empowering Digital Communication”**

**Submitted**

**By**

221FA04229  
G. Anitha Sai

A. Sai Sruthi

221FA04630

221FA04701

Sk. Shaafiya

Shatakshi

221FA04232

*Under the guidance of*

***Mr. Kiran Kumar Kaveti***

*Associate Professor*



**SCHOOL OF COMPUTING & INFORMATICS**

**DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING**

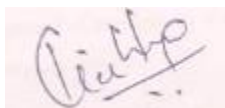
**VIGNAN'S FOUNDATION FOR SCIENCE, TECHNOLOGY AND RESEARCH Deemed  
to be UNIVERSITY**

**Vadlamudi, Guntur.**

**ANDHRA PRADESH, INDIA, PIN-522213.**

### **CERTIFICATE**

This is to certify that the Field Project entitled “**Empowering Digital Communication**” that is being submitted by 221FA04229 (G.Anitha Sai), 221FA04232 (A.Sai Sruthi), 221FA04630 (Sk. Shaafiya) and 221FA04701 (Shatakshi) for partial fulfilment of Field Project is a bonafide work carried out under the supervision of Ms. Dr. N. Sameera., Assistant Professor, Department of CSE.



Guide name& Signature

Designation



Dr. S. V. Phani Kumar

HOD,CSE



## DECLARATION

We hereby declare that the Field Project entitled “**Empowering Digital Communication**” that is being submitted by 221FA04229 (G.Anitha Sai), 221FA04232 (A.Sai Sruthi), 221FA04630 (Sk. Shaafiya) and 221FA04701 (Shatakshi) in partial fulfilment of Field Project course work. This is our original work, and this project has not formed the basis for the award of any degree. We have worked under the supervision of Ms. Dr. N. Sameera., Assistant Professor, Department of CSE.

By

221FA04229 (G.Anitha Sai)

221FA04232 (A.Sai Sruthi)

221FA04630 (Sk. Shaafiya)

221FA04701 (Shatakshi)

# TABLE OF CONTENTS

1. INTRODUCTION.....	13
1.1 Overview of the Digital Notice Board .....	13
1.2 Importance of Digital Communication in Universities .....	14
1.2 Objectives of the Project.....	14
1.3 Scope and Limitations.....	14
2. SOFTWARE REQUIREMENTS SPECIFICATION .....	17
2.1 Requirement Analysis.....	17
2.2 Problem Statement .....	17
2.3 Functional Requirements .....	17
2.4 Software Requirement Specification.....	18
2.5 Software Requirements.....	20
2.6 Hardware Requirements .....	21
2.7 Functional Requirements (Modules).....	21
2.8 Non-Functional Requirements.....	23
2.9 External Interface Requirements.....	23
2.10 Feasibility study.....	24
3. SYSTEM DESIGN AND ANALYSIS.....	27
3.1 Introduction .....	27
3.2 System Overview .....	28
3.3 System Architecture .....	29
3.4 Data Flow and Database Design.....	34
4. USER INTERFACE AND DESIGN.....	39
4.1 User Interface Design .....	39
4.2 Design of Key Features and Modules.....	40
5. IMPLEMENTATION .....	46
5.1 Sample Code and Scripts.....	46
5.2 Screen Captures and Demonstrations.....	55
6. TESTING .....	62
6.1 Software Testing.....	62
6.2 Black box Testing.....	62
6.3 White box Testing .....	62
6.4 Performance Testing.....	62
6.5 Load Testing.....	62

6.6 Manual Testing.....	62
7. RESULTS AND CHALLENGES .....	66
7.1 Results .....	66
7.2 Challenges .....	66
8. CONCLUSION .....	68
8.1 Conclusions .....	68
8.2 Scope for future work.....	68
8.3 Limitations .....	68
8.4 BIBLIOGRAPHY .....	69

## LIST OF FIGURES

Figure 3-1	A System Deployment Model Of Digital Notice Boad .....	28
Figure 3-2	Data Flow Diagram of Digital Notice Board.....	30
Figure 3-3	Database for Digital Notice Board.....	32
Figure 4-1	Use Case Diagram for system.....	38
Figure 4-2	Sequence Diagram for Admin module.....	39
Figure 4-3	Sequence Diagram for User module .....	40
Figure 4-4	Activity Diagram for DNB .....	40
Figure 4-5	Class Diagram for Offline DNB .....	41
Figure 4-6	Deployment Diagram of the System.....	42
Figure 4-7	ER Diagram .....	43
Figure 5-1	Home Screen Code.....	54
Figure 5-2	Admin code.....	55
Figure 5-3	Login code .....	55
Figure 5-4	Home screen .....	56
Figure 5-5	Login code .....	56
Figure 5-6	Register code.....	56
Figure 5-7	Admin Dashboard .....	57
Figure 5-8	Student Dashboard .....	57
Figure 5-9	Customer dash board Activity.....	58
Figure 5-10	Placing the order Activity .....	58
Figure 5-11	Menu activity .....	59
Figure 5-12	Menu item activity.....	59
Figure 6-1	Test Case for Inserting new Record.....	62
Figure 6-2	Test Case for Updating order status .....	63
Figure 6-3	Coverage report for Junit and Jmockit Test Cases.....	63

***CHAPTER - 1***  
***INTRODUCTION***

# 1. INTRODUCTION

## 1.1 Overview of the Digital Notice Board

A Digital Notice Board (DNB) is an advanced system designed to replace traditional paper-based notice boards with electronic displays. These boards use digital screens (such as LED or LCD) to display important information in real-time, often integrating various multimedia elements such as text, images, videos, and animations. Digital notice boards have become a vital tool in modern institutions, especially in universities, as they offer an efficient and dynamic way to disseminate information.

The concept of the digital notice board has evolved with advancements in technology. In its basic form, the system allows administrators to upload announcements, events, and news onto the platform, which are then displayed on screens located throughout the campus. This system can be managed remotely, offering flexibility in controlling what is shown and when. Digital notice boards can be connected to centralized servers or cloud-based systems, allowing updates to be made from anywhere, at any time.

In a university setting, Digital Notice Boards can display various types of information, such as class schedules, exam dates, faculty announcements, student achievements, campus events, emergency alerts, and more. The system can also accommodate scrolling text, video clips, and live feeds, enhancing engagement and ensuring that critical information reaches students and staff promptly.

These boards not only provide real-time updates but also enable better content management and reduce the environmental impact of paper waste. They are scalable and customizable, making them an ideal solution for institutions of varying sizes and needs.

## 1.2 Importance of Digital Communication in Universities

In today's fast-paced academic environment, effective communication plays a crucial role in the smooth functioning of universities. Digital communication, particularly through tools like digital notice boards, has become a vital component in enhancing the flow of information across campuses.

The traditional methods of communication, such as bulletin boards or printed announcements, are becoming obsolete due to their limitations in accessibility, engagement, and speed of updating. Digital notice boards provide a more efficient and reliable method for disseminating information in real-time.

The integration of digital communication offers numerous benefits to universities:

- **Real-Time Information Sharing:** Unlike paper-based systems, digital notice boards allow information to be updated instantly. Whether it's a change in class timings, emergency notices, or last-minute event updates, these systems ensure that the information is delivered immediately to all viewers.
- **Enhanced Engagement:** The interactive nature of digital platforms captures the attention of students and staff more effectively than static paper notices. The inclusion of multimedia content, such as videos, images, and animations, can make



the message more engaging and easier to understand, particularly for large or diverse student bodies.

- **Increased Accessibility:** Information on digital notice boards can be accessed from various locations on campus. Whether in lecture halls, libraries, cafeterias, or corridors, these boards

***CHAPTER - 2***

***SOFTWARE REQUIREMENT***

***SPECIFICATION***

## **2. SOFTWARE REQUIREMENTS SPECIFICATION**

### **2.1 Requirement Analysis**

Requirement analysis is the process of gathering, understanding, and documenting the expectations and needs of the stakeholders involved in the project. For the Digital Notice Board system, this phase involves the following key steps:

**Identifying Stakeholders:** The primary stakeholders are the university administrators, IT department, faculty members, students, and other staff who will interact with the system.

**Understanding the Problem:** Universities currently rely on physical notice boards to communicate important information, which has limitations in terms of accessibility, time-efficiency, and environmental sustainability. The digital notice board aims to replace this outdated method with a more effective, real-time solution.

**Gathering Functional Requirements:** The system should support easy content management, real-time updates, multimedia support, user access control, and integration with existing hardware.

**Understanding Constraints:** The digital notice board system should work within the university's network infrastructure, considering potential hardware limitations and internet connectivity. The design should be scalable to accommodate the number of screens in various locations.

**Defining Success Criteria:** Successful implementation will be measured by the ability of the system to deliver accurate, real-time updates across campus, ease of use for administrators, and minimal downtime.

### **2.2 Problem Statement**

Universities face challenges in communicating important announcements to their students and staff effectively through traditional paper-based notice boards. These methods are inefficient and often lead to outdated or inaccessible information. Physical notice boards are static, require regular maintenance, and fail to engage the campus community in a modern way. The lack of real-time updates and interactive content delivery creates a communication gap. This project aims to develop a Digital Notice Board system that can update and display notices in real-time across the campus, providing an engaging, cost-

effective, and efficient communication platform.

### 2.3 Functional Requirements

The functional requirements define the specific behaviors and features that the Digital Notice Board system must have. These include:

**Content Management:** Admin users should be able to upload and manage various types of content (text, images, videos) through a web interface.

**Real-Time Updates:** Any changes made by the administrator should be reflected instantly on all digital notice boards connected to the system.

**Multimedia Support:** The system must support displaying text, images, video files, and animations.

**User Roles and Access Control:** The system should support different user roles such as Admin, Moderator, and Viewer. Admins will have full access to manage content, while viewers can only view notices.

**Scheduling Notices:** Admins should be able to schedule when notices appear on the board and for how long.

**Display Layouts:** The content displayed on the boards should be configurable, allowing the system to accommodate different layouts for different types of notices (e.g., urgent vs. regular).

**Error Logging:** The system should maintain logs of all errors and system activities for troubleshooting and monitoring.

**User Interface:** A simple, easy-to-navigate interface that allows administrators to upload, edit, and delete content with minimal training.

### 2.4 Software Requirement Specification

The Software Requirement Specification (SRS) provides a comprehensive description of the system's requirements. It includes:

**System Overview:** The Digital Notice Board will be a web-based content management system with the ability to push content to digital screens in real time. The system will be designed to handle both small and large-scale campuses.

**System Functionality:** The system will allow administrators to upload and manage notices, view and schedule content, monitor the status of the boards, and ensure that content is displayed correctly.

**Technological Stack:** The system will use:

**Frontend:** React for the web-based interface.

**Backend:** Node.js with Express for server-side logic and content management.

**Database:** MongoDB to store content and schedules.

**Display Technology:** Integration with digital displays that support HTML/CSS-based content.

**Network:** Local campus network for content delivery.

**Performance Criteria:** The system must be able to handle a large volume of notices and be capable of real-time updates to multiple screens simultaneously.

## 2.5 Software Requirements

This section details the software tools and technologies required to build the Digital Notice Board system:

**Operating System:** Linux or Windows Server for hosting the web application and managing content delivery.

**Backend:**

- Node.js (for backend development).
- Express.js (for routing and managing server-side logic).

**Frontend:**

- React.js (for developing the user interface and content management system).
- HTML5, CSS3, JavaScript for structuring and styling the web interface.

Database:

- MongoDB (for storing notices, schedules, user roles, and logs).

Content Display: Integration with digital screens that support web-based content (HTML/CSS).

Development Tools:

- Git for version control.
- Visual Studio Code or similar IDE for development.
- Nginx or Apache for serving the application.

## **2.6 Hardware Requirements**

The hardware required for the implementation of the Digital Notice Board system includes:

1. **Digital Screens:** High-definition digital displays (LED or LCD) that will be installed at various locations across the university.
2. **Media Player Hardware:** Each screen will be connected to a media player (a small computer or device) capable of displaying content delivered from the server.
3. **Server:** A server to host the content management system, which should have sufficient processing power and storage to handle real-time updates for multiple boards.
4. **Network Infrastructure:** Reliable network connectivity, both wired and wireless, to ensure uninterrupted communication between the server and display units.
5. **Backup Power:** UPS (Uninterrupted Power Supply) systems to ensure the digital notice boards continue operating during power outages.

## **2.7 Functional Requirements (Modules)**

The Digital Notice Board system can be broken down into several functional modules:

1. **Content Management Module:** Allows administrators to add, edit, delete, and schedule notices.

2. Display Control Module: Manages the scheduling and real-time updates of content displayed on digital screens.
3. User Management Module: Handles user roles (Admin, Moderator, Viewer) and access control.
4. Monitoring and Analytics Module: Tracks the status of digital boards and provides logs for errors, activity, and content update history.
5. Notification Module: Sends alerts for urgent messages that need to be displayed immediately.
6. Integration Module: Handles integration with digital displays, ensuring the correct content is delivered and displayed.

## **2.8 Non-Functional Requirements**

Non-functional requirements define the system's operational attributes and performance standards:

1. Reliability: The system must be highly reliable, with minimal downtime.
2. Scalability: The system should be able to scale to accommodate additional digital screens as the university grows.
3. Performance: The system should provide real-time content updates without delays.
4. Security: The system should ensure data security, particularly in terms of user authentication and content access control.
5. Usability: The interface should be intuitive and easy to use for administrators, with minimal training required.
6. Compatibility: The system must be compatible with common operating systems and display hardware.
7. Maintainability: The system should be easy to maintain and update, with clear documentation.

## **2.9 External Interface Requirements**

The system needs to interface with external systems and components:

1. **Digital Screens:** The system will interface with digital displays using standard web technologies (HTML/CSS) or through specialized API integration for certain types of hardware.
2. **University Database:** The system may need to integrate with the university's existing database for user authentication and content management.
3. **Network Infrastructure:** The system requires robust network connectivity to ensure real-time content updates across all screens.

## **2.10 Feasibility Study**

The feasibility study evaluates the practicality of implementing the Digital Notice Board system, focusing on:

1. **Technical Feasibility:** The system is technically feasible, as the required technologies (React, Node.js, MongoDB) are proven, widely used, and support scalability.
2. **Economic Feasibility:** While the initial setup cost for hardware (digital screens, media players, server infrastructure) might be high, the long-term savings from reduced printing and maintenance of physical notice boards make the system economically viable.
3. **Operational Feasibility:** The system will be easy to operate for university administrators, with minimal training required. The user-friendly interface and reliable backend ensure smooth day-to-day operations.



***CHAPTER - 3***  
***ANALYSIS & DESIGN***

## 3. ANALYSIS & DESIGN

### 3.1 Introduction

The **Digital Notice Board** is an advanced information dissemination system designed to replace traditional paper-based notice boards in universities. It provides an efficient, eco-friendly, and centralized way to display announcements, academic schedules, event updates, and urgent notifications. The system enables real-time content updates, reducing dependency on physical notices while ensuring that all students and staff receive important information on time.

This system is particularly beneficial for universities that need to manage multiple notices daily and ensure their accessibility across multiple locations. By utilizing a cloud-based or local server architecture, the system ensures scalability and ease of maintenance while providing a user-friendly interface for administrators and viewers.

The project aims to enhance communication efficiency by integrating digital displays and an easy-to-use content management system (CMS) that allows authorized users to update and schedule notices dynamically.

### 3.2 System Overview

The **Digital Notice Board System** is a web-based application designed to manage and display university notices electronically. The system includes three primary components:

#### 1. Administrator Panel:

- A web-based interface for authorized users (e.g., university administrators) to create, edit, schedule, and remove notices.
- Role-based access control to restrict permissions based on user roles.

#### 2. Database Management System:

- A central database that stores notices, user details, scheduling information, and system logs.
- Efficient indexing to ensure quick retrieval and updates of notices.

### 3. Display Units:

- Digital screens installed at key locations across the university.
- Automatically fetches and updates content from the server at defined intervals.
- Supports multimedia formats, including text, images, and videos.

The system provides **real-time synchronization** of notices across multiple locations, ensuring that the latest information is always available. Additionally, it includes **content scheduling**, allowing administrators to set expiration times for notices, thereby automating content updates without manual intervention.

## 3.3 System Architecture

The **architecture of the Digital Notice Board System** follows a client-server model with modular components to ensure scalability and flexibility.

### Key Components of the Architecture:

#### 1. Frontend (Client-Side)

- Developed using **React.js** for a dynamic and responsive UI.
- Provides an intuitive dashboard for administrators to manage notices.
- Uses **WebSockets or API polling** for real-time updates on display units.

#### 2. Backend (Server-Side)

- Developed using **Node.js with Express.js** to handle business logic.
- Manages user authentication, content processing, and database operations.
- Uses **RESTful APIs** to interact with the frontend and display units.

#### 3. Database (Data Storage)

- Uses **MongoDB** for storing notice data, user credentials, and system logs.
- Ensures efficient indexing and retrieval for seamless content updates.

#### 4. Digital Display System

- Digital screens (TVs, monitors) connected via **Raspberry Pi or mini PCs**.
- Fetches new content from the backend using APIs or WebSockets.
- Displays notices in a structured format with support for multimedia elements.

## 5. Authentication and User Management

- Uses **JWT-based authentication** to secure access.
- Implements role-based access control (RBAC) to manage different types of users (Admins, Moderators, Viewers).

### System Deployment Model

The system can be deployed in two primary ways:

- **On-Premise Deployment:** The university hosts the server and database on its local network.
- **Cloud Deployment:** Uses cloud services (e.g., AWS, Firebase) for scalable content management and delivery.

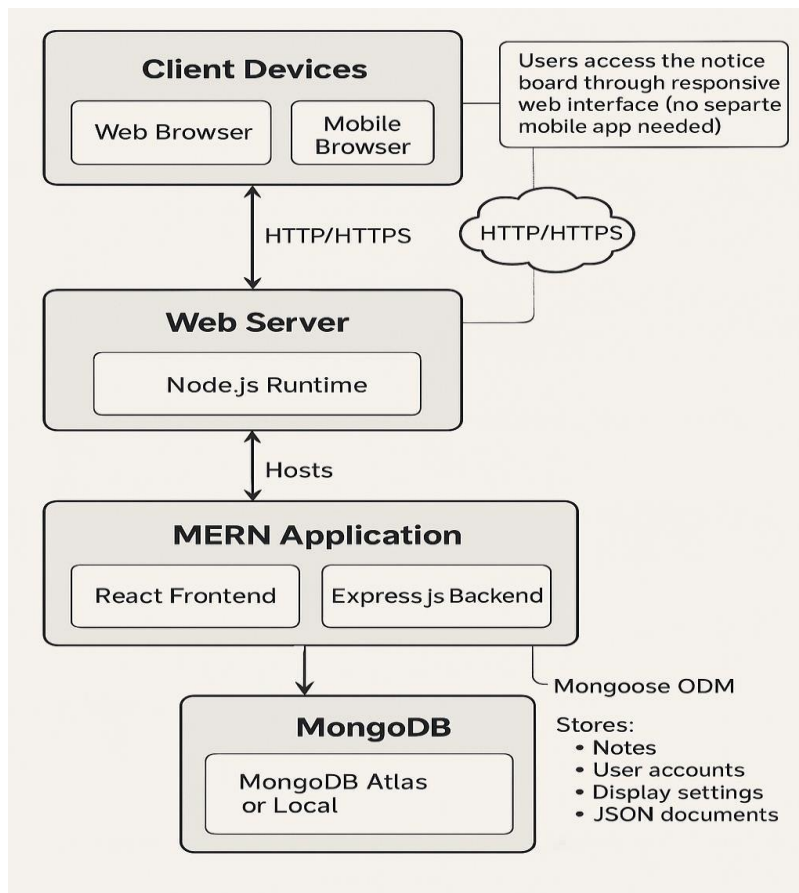


Fig 3.1 : System Deployment Model Of Digital Notice Board

### 3.4 Data Flow and Database Design

#### 3.4.1 Data Flow Diagram (DFD)

The **Data Flow Diagram (DFD)** illustrates how data moves within the system. Below is a high-level breakdown of the data flow:

1. **Admin Logs In** → Authentication request is sent to the server → The server verifies the credentials using the database.
2. **Admin Adds or Updates Notices** → Request is sent to the server → The server updates the database → New content is sent to digital displays.
3. **Digital Display Requests Data** → API call is made to fetch the latest notices → Notices are displayed dynamically.

#### DFD Level 0 (Context Diagram)

- Users (Admins, Moderators) interact with the **Web Application**.
- The web app communicates with the **Backend API**.
- The backend retrieves and updates data in the **MongoDB Database**.
- Digital screens request content from the **Backend API** to display notices.

#### DFD Level 1 (Detailed Flow)

- **Process 1: User Authentication**
  - The administrator logs in using their credentials.
  - The server validates the credentials and provides access.
- **Process 2: Notice Management**
  - The admin submits a new notice via the web interface.
  - The backend stores it in the database.
  - The updated content is pushed to digital screens.
- **Process 3: Content Display on Digital Screens**
  - The screens fetch the latest content periodically via API calls or WebSockets.
  - The content is formatted and displayed dynamically.

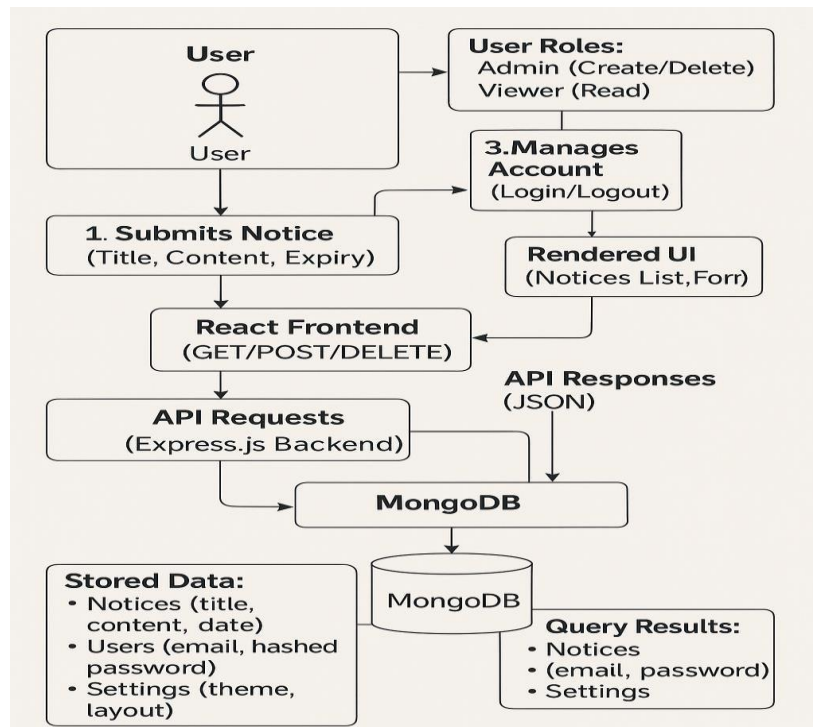


Fig 3.2 : Data Flow Diagram of Digital Notice Board

### 3.4.2 Database Design

The **database schema** is designed to efficiently store and retrieve notice-related information.

#### Tables (Collections in MongoDB)

##### 1. Users Collection (user\_management)

- user\_id (String, Primary Key)
- username (String, Unique)
- password\_hash (String)
- role (Enum: Admin, Moderator, Viewer)

##### 2. Notices Collection (notice\_board)

- notice\_id (String, Primary Key)
- title (String)
- content (Text)
- image\_url (String, Optional)

- video\_url (String, Optional)
- posted\_by (String, Foreign Key from Users)
- created\_at (Timestamp)
- expiry\_date (Timestamp)

### 3. Display Screens Collection (display\_units)

- screen\_id (String, Primary Key)
- location (String)
- status (Online/Offline)
- last\_updated (Timestamp)

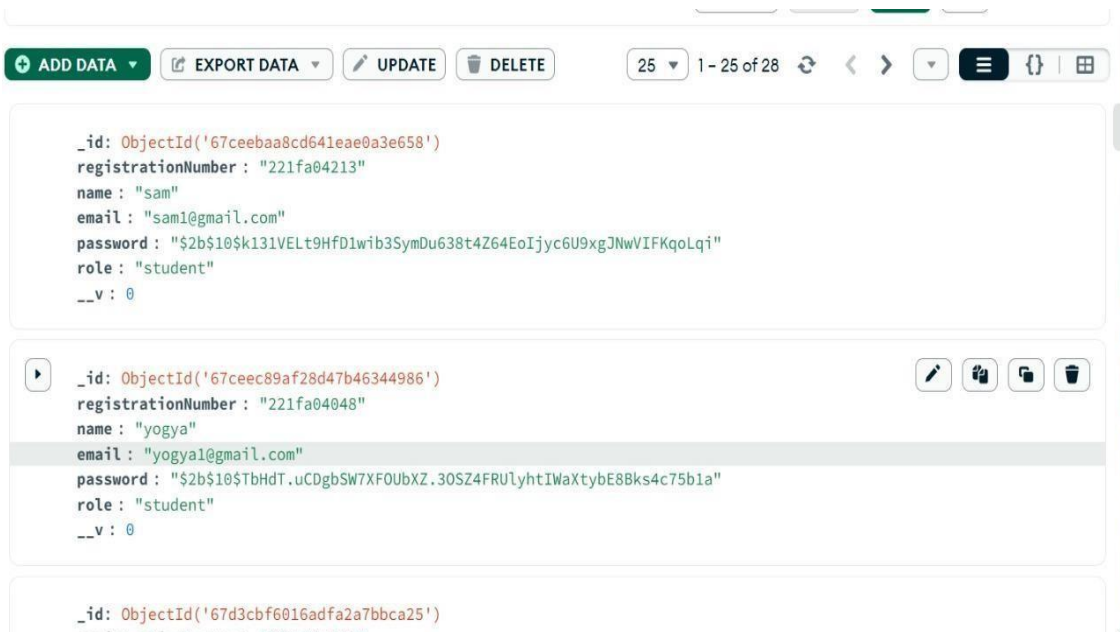


Fig 3.3 :Database for Digital Notice Board

***CHAPTER - 4***  
***MODELING***



## 4. MODELING

### 4.1 Design

Requirements gathering followed by careful analysis leads to a systematic Object Oriented Design (OOAD). Various activities have been identified and are represented using Unified Modeling Language (UML) diagrams. UML is used to specify, visualize, modify, construct and document the artifacts of an object-oriented software-intensive system under development.

#### 4.1.1. Use Case Diagram

In the Unified Modeling Language (UML), the use case diagram is a type of behavioral diagram defined by and created from a use-case analysis. It represents a graphical over view of the functionality of the system in terms of actors, which are persons, organizations or external system that plays a role in one or more interaction with the system. These are drawn as stick figures. The goals of these actors are represented as use cases, which describe a sequence of actions that provide something of measurable value to an actor and any dependencies between those use cases.

In this application there is only actor – soldier and below is the use case diagram of this application.

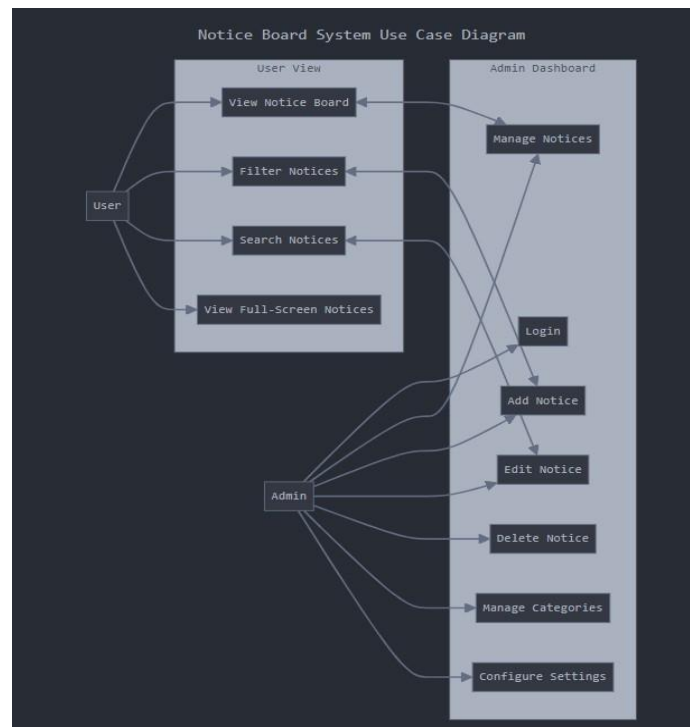


Figure 4-1 Use Case Diagram for System

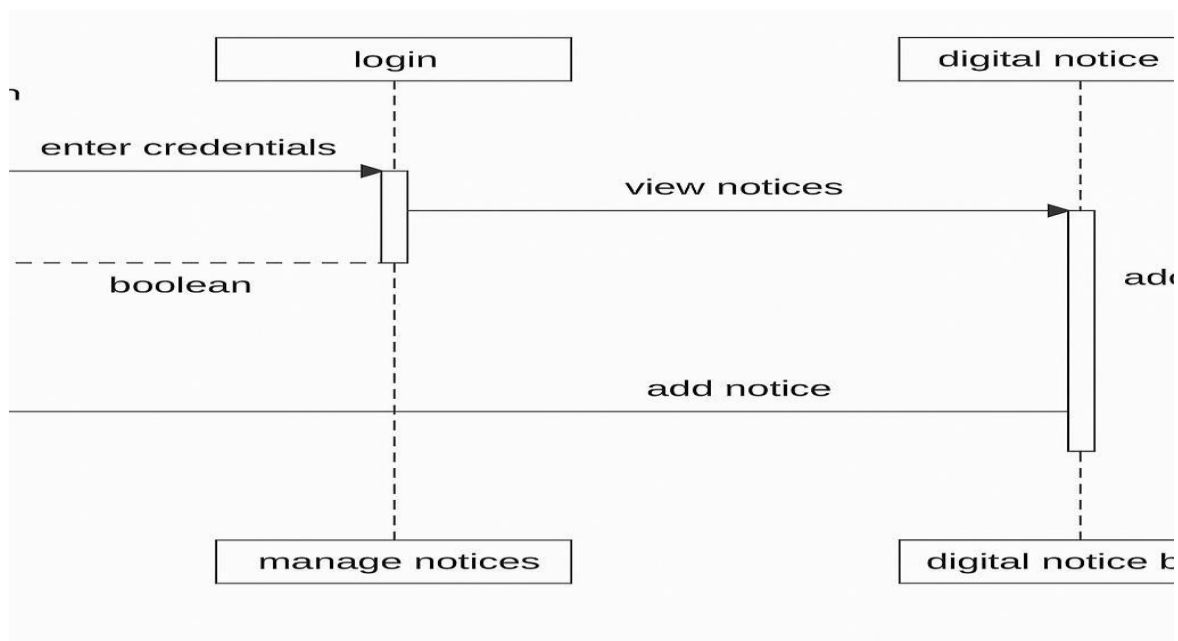
### 4.1.2 Sequence Diagram

UML sequence diagrams are used to show how objects interact in a given situation. An important characteristic of a sequence diagram is that time passes from top to bottom: the interaction starts near the top of the diagram and ends at the bottom (i.e. Lower equals later).

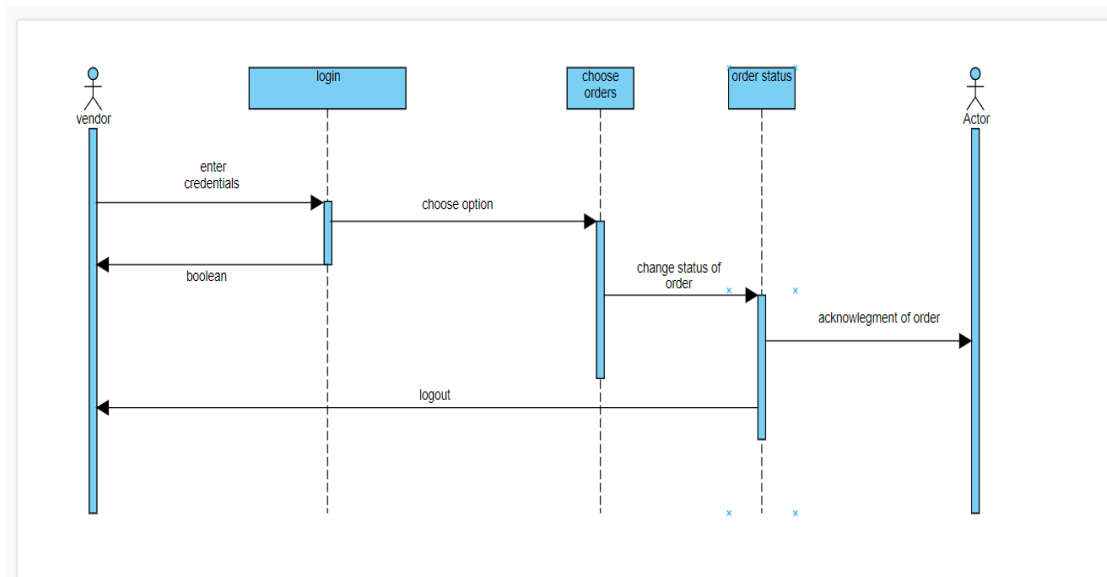
A popular use for them is to document the dynamics in an object-oriented system. For each key, collaboration diagrams are created that show how objects interact in various representative scenarios for that collaboration.

Sequence diagram is the most common kind of interaction diagram, which focuses on the message interchange between a numbers of lifelines.

The following nodes and edges are typically drawn in a UML sequence diagram: lifeline, execution specification, message, combined fragment, interaction use, state invariant, continuation, destruction occurrence.



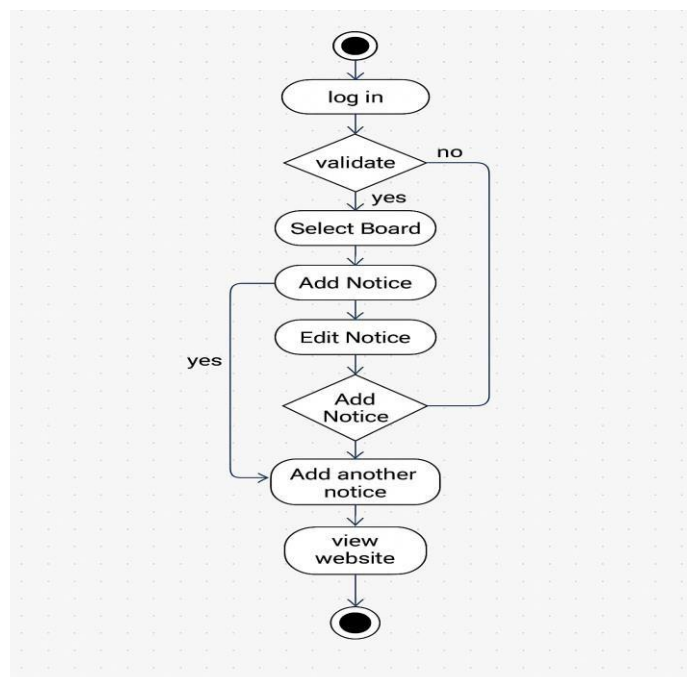
**Figure 4-2 Sequence Diagram for Digital Notice Board**



**Figure 4-3 Sequence Diagram for Vendors Module**

### 4.1.3 Activity Diagram

Activity diagram is another important diagram in UML to describe dynamic aspects of the system. Activity diagram is basically a flow chart to represent the flow from one activity to another activity. The activity can be described as an operation of the system. So the control flow is drawn from one operation to another. This flow can be sequential, branched or concurrent. Activity diagrams deals with all type of flow control by using different elements like fork, join etc. Activity is a particular operation of the system.



**Figure 4-4 Activity Diagram for DNB**

#### 4.1.4 Class Diagram

In software engineering, a class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among the classes.

The class diagram is the main building block of object-oriented Modelling. It is used both for general conceptual modelling of the application, and for detailed modelling translating the models into programming code. Class diagrams can also be used for data modelling. The classes in a class diagram represent both the main objects, interactions in the application and the classes to be programmed.

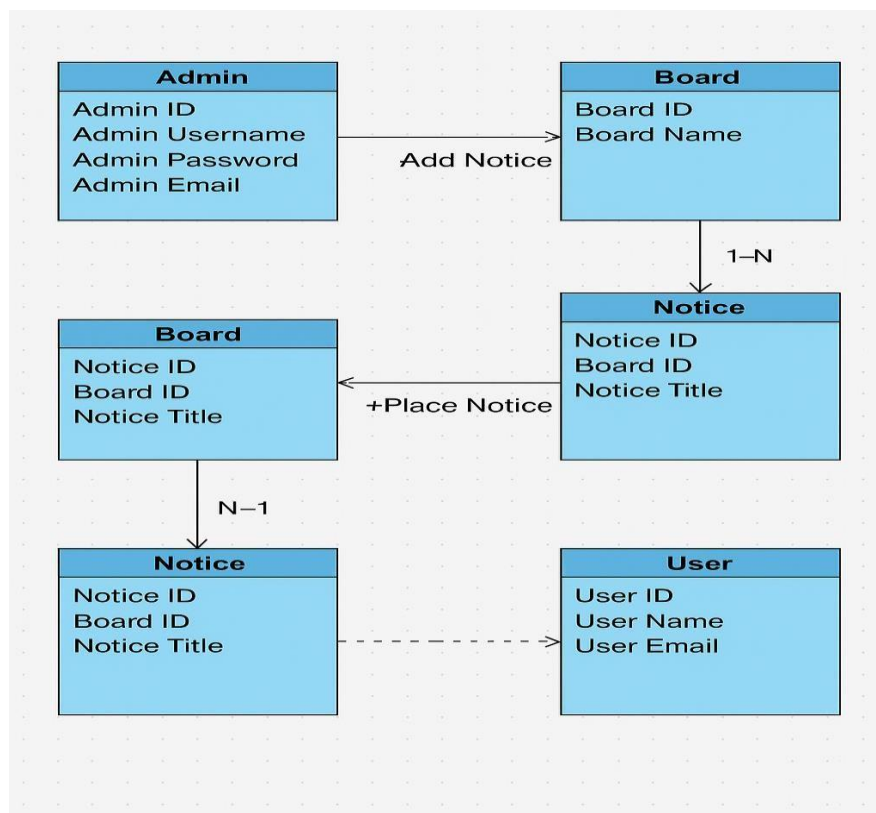
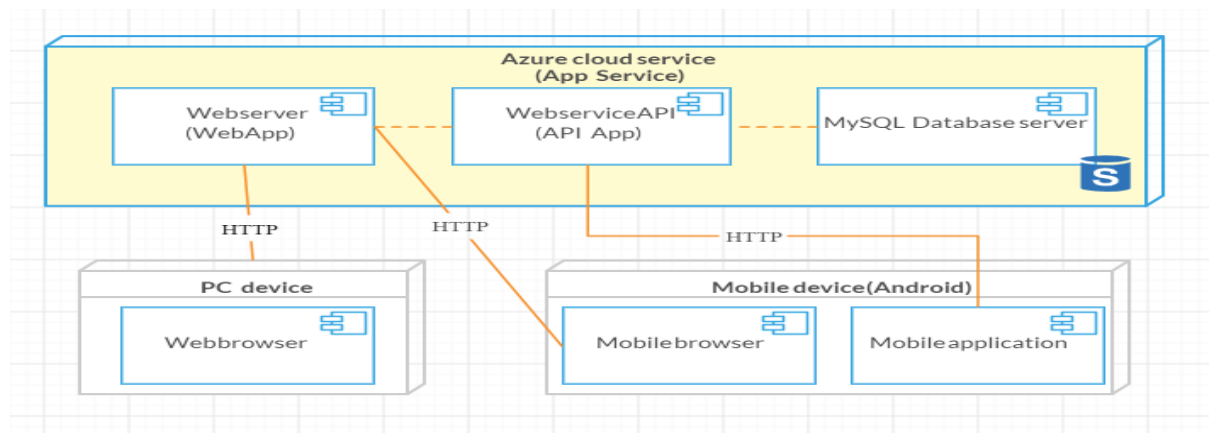


Figure 4-5 Class Diagram for DNB

#### 4.1.5 Deployment Diagram

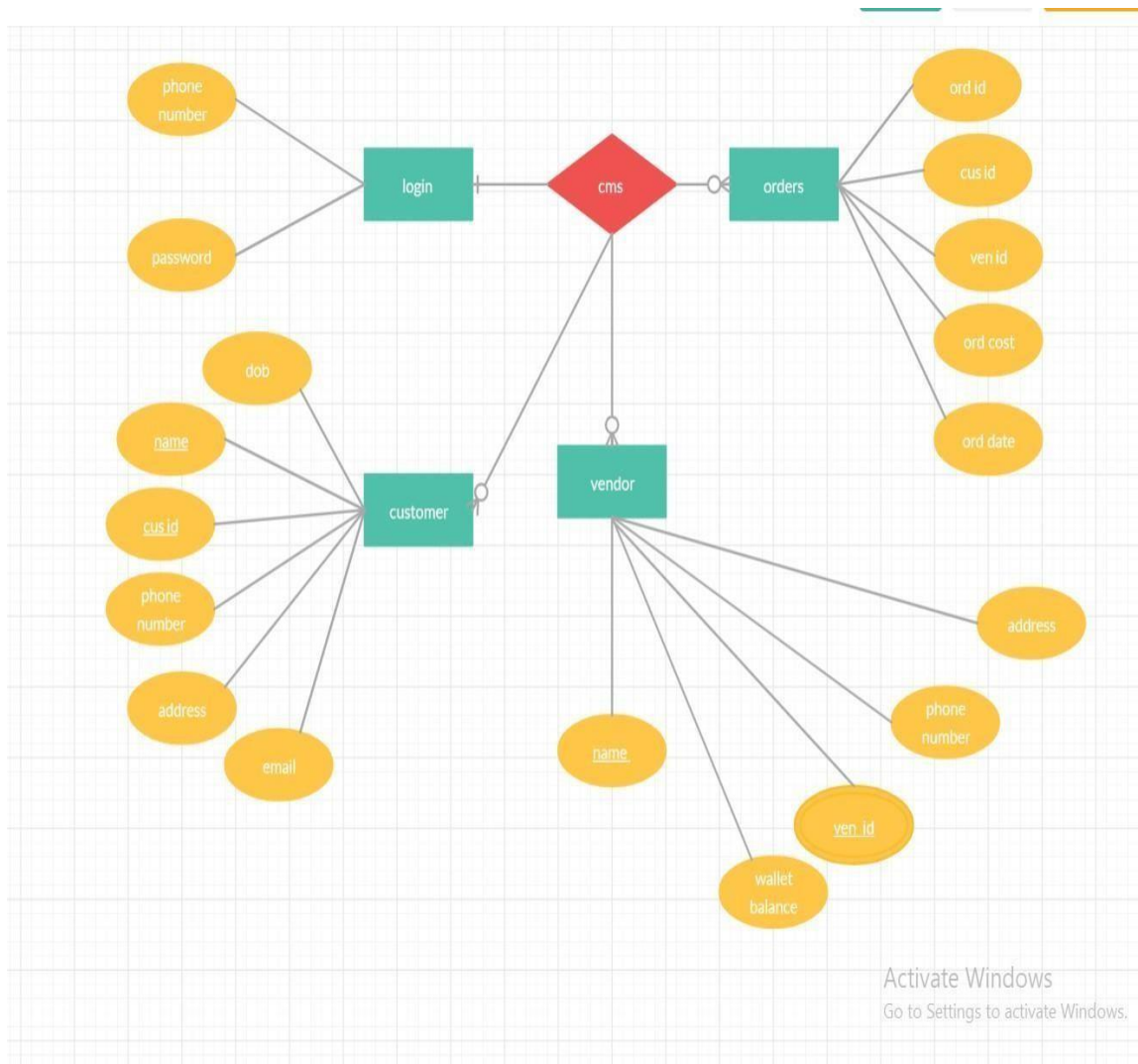
Deployment diagram shows execution architecture of systems that represent the assignment (deployment) of software artifacts to deployment targets (usually nodes). Nodes represent either hardware devices or software execution environments. They could be connected through communication paths to create network systems of arbitrary complexity. Artifacts represent concrete elements in the physical world that are the result of a development process and are deployed on nodes.



**Figure 4-6 Deployment Diagram of the system**

#### 4.1.6 ER Diagram

An ER model is an abstract way to describe a database. Describing a database usually starts with a relational database, which stores data in tables. Some of the data in these tables point to data in other tables - for instance, your entry in the database could point to several entries for each of the phone numbers that are yours. The ER model would say that you are an entity, and each phone number is an entity, and the relationship between you and the phone numbers is 'has a phone number'. Diagrams created to design these entities and relationships are called entity–relationship diagrams or ER diagrams.



**Figure 4-7 ER Diagram**

***CHAPTER - 5***  
***IMPLEMENTATION***

## 5 IMPLEMENTATION

### 5.1 Sample Code

#### 5.1.1 Code for Home Page

```
import React, { useState, useEffect } from "react";
import { Link } from "react-router-dom";
import axios from "axios"; // Import axios for API calls
import "../Homepage.css";
import { FaFacebookF, FaInstagram, FaTwitter, FaLinkedinIn, FaYoutube } from "react-icons/fa"; // Import social media icons

const Homepage = () => {
  const [darkMode, setDarkMode] = useState(false);
  const [isDropdownOpen, setIsDropdownOpen] = useState(false);
  const [userName, setUserName] = useState(null);
  const [notices, setNotices] = useState([]); // State to store notices
  const [searchTerm, setSearchTerm] = useState(""); // State for search term
  const [selectedDate, setSelectedDate] = useState(""); // State for selected date
  const [selectedCategory, setSelectedCategory] = useState(""); // State for selected category

  // Fetch notices from the backend when the component mounts
  useEffect(() => {
    const savedUser = JSON.parse(localStorage.getItem("user"));
    setUserName(savedUser ? savedUser.name : null);
    const savedDarkMode = localStorage.getItem("darkMode") === "true";
    setDarkMode(savedDarkMode);
    document.body.classList.toggle("dark-mode", savedDarkMode);

    fetchNotices(); // Fetch notices when the component mounts
  }, []);

  // Function to fetch notices from the backend
```

#### 5.1.2 Code for Admin Page

```
import React, { useState, useEffect } from "react";
import axios from "axios";
import { Link } from "react-router-dom";
import "../AdminPage.css";

const AdminPage = () => {
  const [notices, setNotices] = useState([]);
  const [title, setTitle] = useState("");
  const [description, setDescription] = useState("");
  const [category, setCategory] = useState("general");
  const [date, setDate] = useState("");
  const [file, setFile] = useState(null);
  const [searchTerm, setSearchTerm] = useState("");
  const [darkMode, setDarkMode] = useState(false);
  const [sorting, setSorting] = useState("date");
  const [filterCategory, setFilterCategory] = useState("all");
  const [editingIndex, setEditingIndex] = useState(null);
  const [userName, setUserName] = useState(null);
  const [isDropdownOpen, setIsDropdownOpen] = useState(false);

  useEffect(() => {
    fetchNotices();
    const savedUser = JSON.parse(localStorage.getItem("user"));
    setUserName(savedUser ? savedUser.name : null);
    const savedDarkMode = localStorage.getItem("darkMode") === "true";
    setDarkMode(savedDarkMode);
```



### 5.1.3 Code for Login Page

```
const Login = ({ setUserRole }) => {
  const [registrationNumber, setRegistrationNumber] = useState("");
  const [email, setEmail] = useState("");
  const [password, setPassword] = useState("");
  const [role, setRole] = useState("student");
  const [error, setError] = useState("");
  const navigate = useNavigate();

  const handleSubmit = async (e) => {
    e.preventDefault();

    try {
      const payload = { password, role };
      if (email) payload.email = email;
      if (registrationNumber) payload.registrationNumber = registrationNumber;

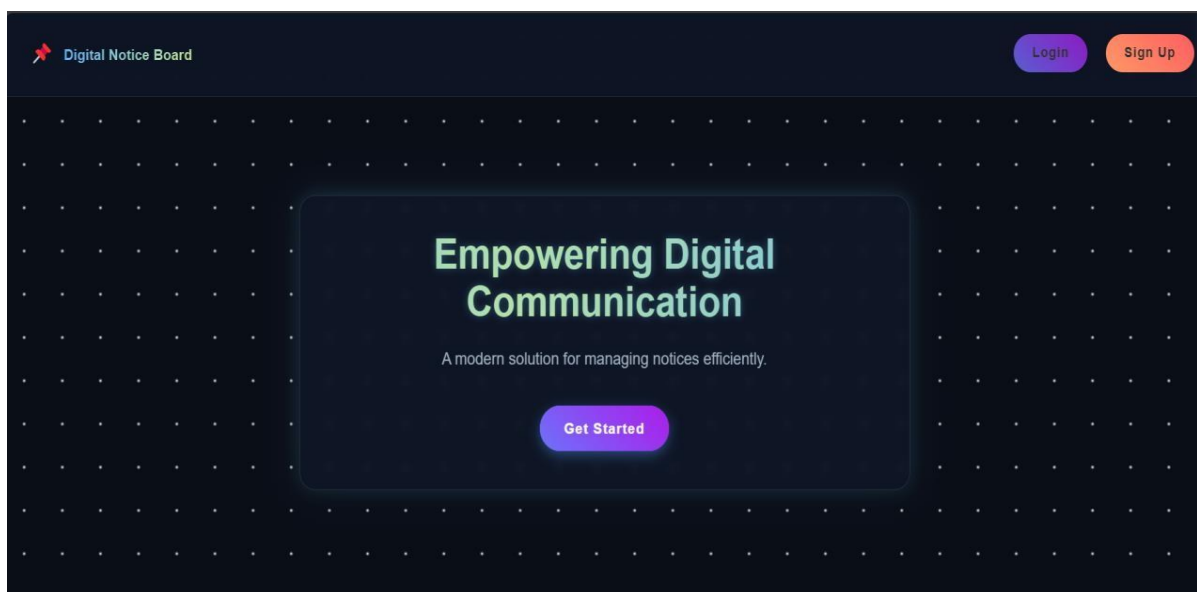
      const res = await axios.post("http://localhost:5000/login", payload);

      localStorage.setItem("token", res.data.token);
      localStorage.setItem("user", JSON.stringify(res.data.user));
      setUserRole(res.data.user.role); // Update role in state

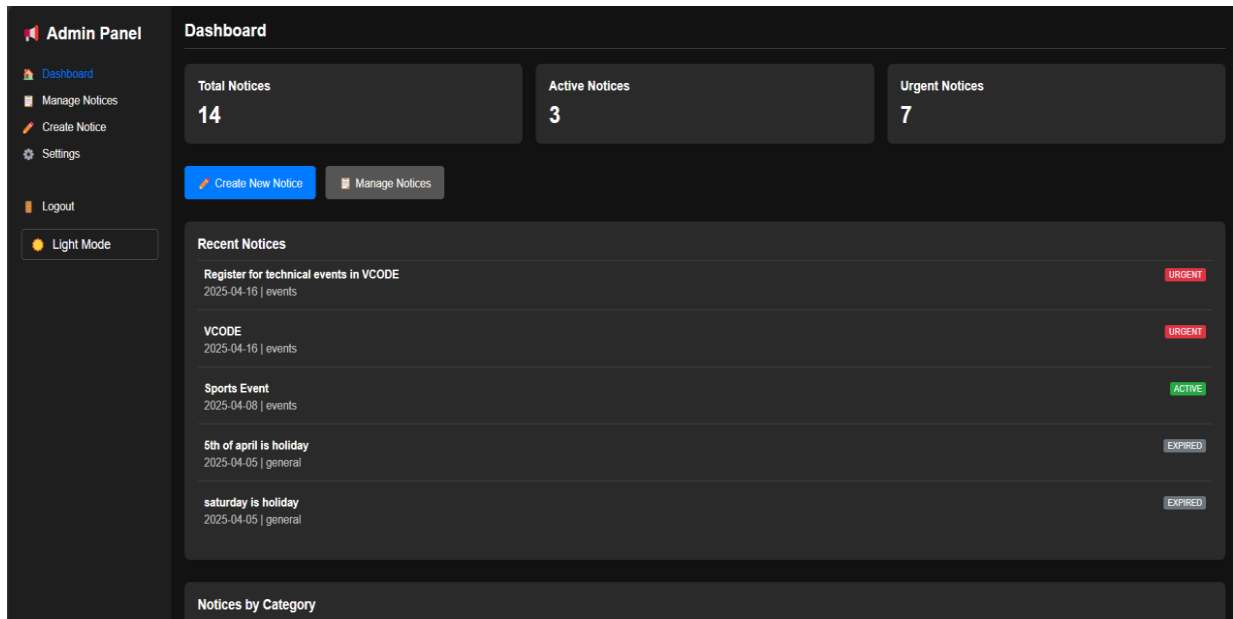
      if (res.data.user.role === "admin") {
        navigate("/admin");
      }
    } catch (error) {
      setError(error.message);
    }
  }
}
```

## 5.2 Screen Captures

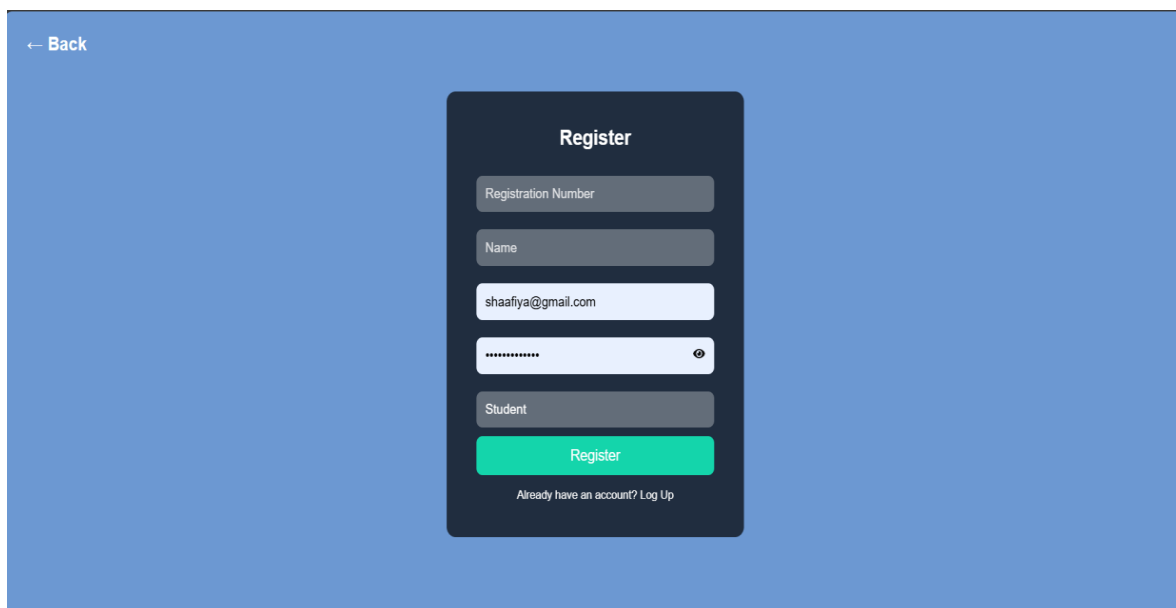
### 5.2.1 home screen



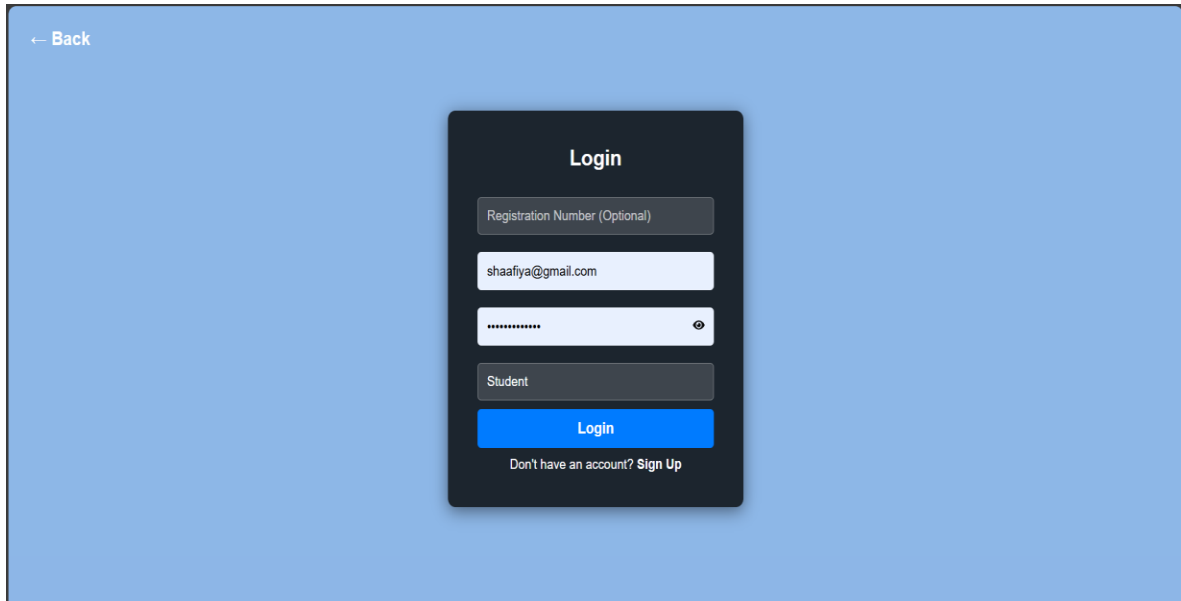
## 5.2.2 Admin page



## 5.2.3 Registration Page

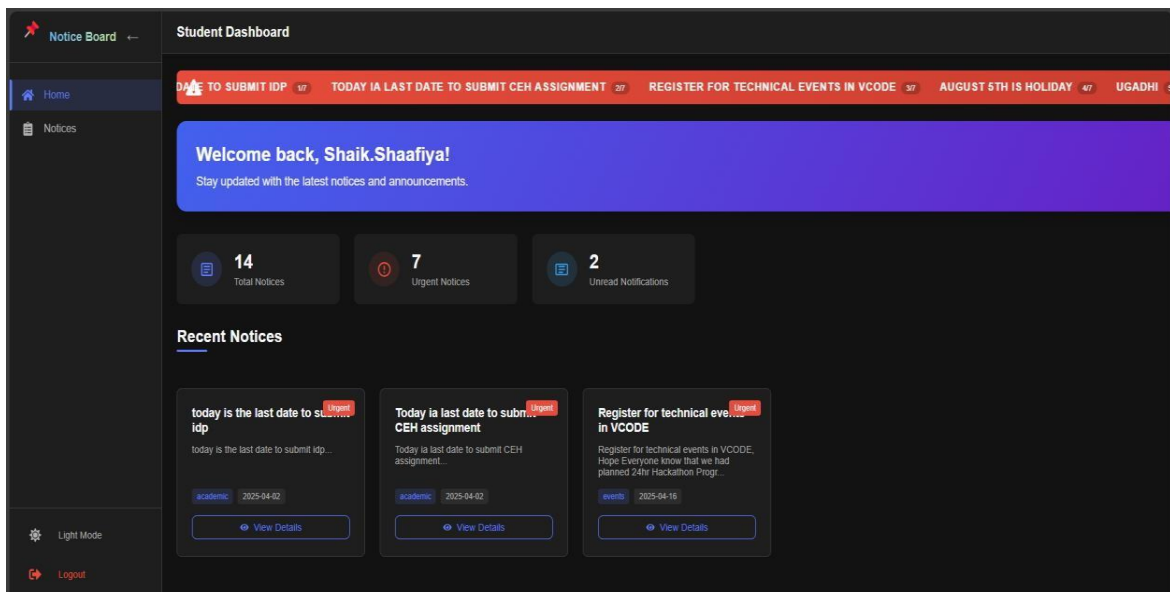


## 5.2.4 Login Page



The screenshot shows a login page with a light blue background. In the top-left corner, there is a back arrow and the text "Back". Centered on the page is a dark grey login card. The card has the title "Login" at the top. Below the title are four input fields: "Registration Number (Optional)", an email field containing "shaafiya@gmail.com", a password field with masked characters and an eye icon, and a role dropdown menu set to "Student". A blue "Login" button is positioned below the inputs. At the bottom of the card, there is a link that says "Don't have an account? Sign Up".

## 5.2.5 Student Dashboard



The screenshot displays a student dashboard with a dark theme. On the left is a sidebar with a "Notice Board" header and a list of items: "Home" (with a house icon) and "Notices" (with a document icon). The main content area has a "Student Dashboard" header. Below this is a red banner with urgent notices: "DATE TO SUBMIT IDP 187", "TODAY IS LAST DATE TO SUBMIT CEH ASSIGNMENT 287", "REGISTER FOR TECHNICAL EVENTS IN VCODE 387", "AUGUST 5TH IS HOLIDAY 487", and "UGADHI 5". A purple welcome banner follows, stating "Welcome back, Shaik.Shaafiya!" and "Stay updated with the latest notices and announcements." Below the welcome banner are three summary cards: "14 Total Notices", "7 Urgent Notices", and "2 Unread Notifications". The "Recent Notices" section contains three cards, each with an "Urgent" tag: "today is the last date to submit idp" (academic, 2025-04-02), "Today is last date to submit CEH assignment" (academic, 2025-04-02), and "Register for technical events in VCODE" (events, 2025-04-16). Each card includes a "View Details" button. At the bottom left of the sidebar, there are "Light Mode" and "Logout" options.

***CHAPTER - 6***  
***TESTING***

## 6 TESTING

### 6.1 Software Testing and Debugging

Software testing is an essential phase in the development of the Digital Notice Board to ensure that it meets all functional and performance requirements. The objective of testing is to ensure that the system operates as expected and is free from bugs. Debugging is performed to identify and fix any errors or issues in the codebase, ensuring that the system is stable and reliable. The testing process involved several methods such as unit testing, integration testing, and system testing.

**Unit Testing:** Each individual module or function of the system was tested in isolation to ensure that it performs its intended task correctly.

**Integration Testing:** After unit testing, integration testing was conducted to ensure that different modules work well together as part of a larger system.

**System Testing:** Finally, system testing was performed to check the entire application as a whole, ensuring that all components function together seamlessly.

### 6.2 Black Box Testing

Black box testing is a method where the functionality of the system is tested without knowing the internal code structure. This approach focuses purely on the inputs and outputs of the system. For the Digital Notice Board, black box testing ensured that features such as notice creation, user authentication, and role-based access control functioned correctly, and that all intended features delivered the expected results. Test cases were designed based on the requirements and specifications of the system, ensuring that user interactions with the system met expectations.

Example Tests:

Verifying if the system correctly displays notices.

Ensuring user authentication works with the correct credentials and roles.

Checking that the system accurately filters notices based on user roles (faculty, students, admin).

### 6.3 White Box Testing

White box testing involves examining the internal workings and code structure of the system. This testing method focuses on the logic, flow, and performance of the code, identifying hidden issues that may not be caught through black box testing. White box testing for the Digital Notice Board was conducted on critical parts of the codebase, including database queries, business logic, and overall system structure.

**Optimizing Database Queries:** White box testing helped improve database queries to ensure quick data retrieval, especially when multiple notices are being displayed at once.

**Code Efficiency:** The internal logic was tested to eliminate unnecessary steps or redundant code, improving the overall performance and maintainability of the system.

**Error Detection:** Potential bugs and logical flaws were identified and fixed, ensuring that the system operates as

intended.

## **6.4 Performance Testing**

Performance testing evaluates how the system performs under various conditions, including how it behaves under heavy usage. This type of testing focuses on the speed, response time, and stability of the Digital Notice Board system. Key metrics like system latency, load times, and response time were measured under varying user loads.

**Response Time:** The system's response time was measured when retrieving and displaying notices to ensure that users experience minimal delays.

**System Latency:** System latency was tested to verify that the application responds swiftly to user actions, such as posting or viewing notices.

**Server Load Capacity:** The server's capacity to handle a large number of concurrent users was tested, ensuring that the system remains responsive during peak times.

## **6.5 Load Testing**

Load testing helps determine how the system performs under high demand. In the case of the Digital Notice Board, load testing was performed to evaluate how well the system manages multiple users interacting with the platform at the same time. This is crucial in an academic environment, where many users may need to access the system simultaneously, such as during exam announcements or important university updates.

**Testing Multiple Users:** Simulated multiple users interacting with the platform, posting notices, and viewing announcements.

**Handling Large Volumes of Data:** Ensured that the system can handle a large number of notices without performance degradation.

**Stress Test:** The system was deliberately subjected to a high load to identify breaking points and understand how the system handles extreme situations.

## **6.6 User Acceptance Testing (UAT)**

User Acceptance Testing (UAT) is performed to validate the system from an end-user perspective. This phase involves real users — students, faculty, and administrators — interacting with the system to ensure it meets their expectations and needs. Feedback collected during UAT helps ensure that the system is intuitive, user-friendly, and provides the necessary features to users.

**Ease of Use:** End users were asked to perform common tasks like posting and viewing notices to determine the system's usability.

**Feature Validation:** UAT was used to confirm that the features such as notice categorization, user roles, and announcements were functioning as intended.

**Feedback Collection:** Feedback was gathered from users about any issues they encountered or features they found difficult to use. This feedback is critical for making final adjustments before the system is deployed.

By implementing these testing methodologies, the Digital Notice Board was refined to provide a seamless and

efficient user experience, ensuring its reliability and effectiveness in the academic environment.

***CHAPTER - 7***

***RESULTS & CHALLENGES***



## **7 RESULTS AND CHALLENGES**

### **7.1 Results**

In this website user can know the canteen near by location and he can choose the canteen he wants. Then he can place order. Then the vendor gets the notification that the order has been placed and he can deliver them to the customer.

In this project, we are going to create a website where there are 2 kinds of users namely vendors and customers.

#### **1. Owner:**

This user can sign in to our website and can enter his canteen details and then he can enter what food items he has in his canteen and the number of each item present in the canteen. He can also update the number of items in the canteen. The vendor can also accept or reject the order of the customer based on the stock availability

#### **2. Customer:**

This user can sign in and then log in to the website and so that he could see the canteen near to him and can place an order of items he wants.

### **7.2 Challenges**

- Understanding the client requirements was one of the crucial tasks of the whole project.
- Graphic User Interface (GUI) design was a difficult task as there are many types of Android devices with varying screen size and resolutions unlike iPhone.
- Implementing synchronization with server on Android was a challenging task.
- Learning different technologies and frameworks with little guidance.

***CHAPTER - 8***

***CONCLUSIONS & FUTURE WORK***

# CONCLUSION

## 8.1 Conclusions

The Digital Notice Board project successfully addresses the need for an efficient and centralized communication system in universities. By replacing traditional paper-based notice boards with a digital solution, this system enhances accessibility, reduces manual effort, and ensures real-time updates. The project leverages the MERN (MongoDB, Express, React, Node.js) stack to provide a scalable, user-friendly, and interactive platform.

The system's ability to facilitate instant notifications, categorize announcements, and provide role-based access control ensures that information is disseminated accurately and securely. The implementation of user authentication further strengthens the reliability of the platform, preventing unauthorized modifications. Moreover, the intuitive interface simplifies content management, allowing administrators, faculty, and students to seamlessly access relevant announcements.

Throughout the development process, various challenges were encountered, including optimizing database queries for performance, ensuring smooth integration between the frontend and backend, and implementing a responsive design for multiple device compatibility. Overcoming these challenges has resulted in a robust and efficient system that meets the core requirements of digital communication in an academic environment.

## 8.2 Scope for Future Enhancements

While the Digital Notice Board has been designed to meet the essential communication needs of universities, several potential enhancements can be explored to improve its functionality further:

**Mobile Application Support:** Developing a dedicated mobile app for Android and iOS platforms to enhance accessibility and provide push notifications.

**Multimedia Support:** Incorporating support for images, videos, and documents in announcements to make communication more interactive and engaging.

**AI-Based Smart Notifications:** Implementing AI algorithms to prioritize and suggest relevant notices based on user preferences and activity history.

**Multi-Language Support:** Enabling multilingual capabilities to cater to a diverse user base and improve accessibility.

**Integration with Other University Systems:** Connecting the notice board with Learning Management Systems (LMS), event management platforms, and student portals for a seamless experience.

**Automated Expiry for Notices:** Introducing a feature that automatically removes expired notices to keep the board updated and clutter-free.

**Analytics and Insights:** Providing administrators with detailed reports and insights into notice engagement, allowing for better communication strategies.

### **8.3 Limitations and Improvements**

Despite its advantages, the Digital Notice Board system has a few limitations that can be addressed in future iterations:

**Dependency on Internet Connectivity:** Since the system is web-based, users require an active internet connection to access notices. Implementing offline functionality could mitigate this limitation.

**Initial Setup and Training:** Some users may require guidance to navigate the platform efficiently. Providing a comprehensive onboarding guide or tutorial videos can help ease adoption.

**Security Concerns:** Although authentication and role-based access control are implemented, continuous security updates and vulnerability assessments are essential to prevent unauthorized access and data breaches.

**Scalability Challenges:** As the number of users and notices increases, optimizing database performance and load balancing will be crucial to maintaining system efficiency.

## BIBLIOGRAPHY

[1] Oracle corporation and its affiliates “MySQL open source database”

with no author

[online] available at: <https://www.mysql.com/>

[2] Sonoo Jaiswal “An Overview on Java”

[online] available at: <https://www.javatpoint.com/>

[3] JMockit “An automated testing toolkit for java” with no author

[online] available at: <http://jmockit.github.io/tutorial/Introduction.html>

[4] M. Vaqqas, September 23, 2014 “RESTful web Services: A Tutorial”

[online] available at <http://www.drdoobbs.com/web-development/tutorial>

[5] The MIT License “Angular one framework” with no author

[online] available at: <https://angularjs.org/>

[6] DaolrevoLtd. Asim “Jasmine and Karma”

[online] available at: <https://codecraft.tv/courses/angular/unit-testing/jasmine-and-karma/>

[7] Software Freedom Conservancy “git local branching on the cheap” with no author

[online] available at: <https://git-scm.com/>