REAL TIME LECTURE HALL AVAILABILITY TRACKER

a minor project report submitted to

THE NATIONAL INSTITUTE OF ENGINEERING, MYSURU

(An Autonomous Institute under VTU, Belagavi)

In partial fulfilment of the requirements for Minor-Project work (BCS586),

fifth semester

Bachelor of Engineering

in

Computer Science and Engineering

Submitted by

Team Details

S. No	USN	Name
1	4NI22CS079	Hemanth Kumar R
2	4NI22CS062	Ganesha M
3	4NI22CS093	Karthikeya S

Under the mentorship of

Dr Jayashri BS

Professor

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

Mysuru-570 008

2024-2025

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

THE NATIONAL INSTITUTE OF ENGINEERING

ESTD : 1946

CERTIFICATE

This is to certify that the project work entitled "**real time lecture hall availability tracker**" is a work carried out by **Hemanth Kumar R(4NI22CS079) Ganesha M(4NI22CS062) and Karthikeya S(4NI22CS093)** in partial fulfillment for the minor-project work (BCS586), sixth semester, Computer Science & Engineering, The National Institute of Engineering **(**Autonomous Institution under Visvesvaraya Technological University, Belagavi) during the academic year 2021-2022. It is certified that all corrections and suggestions indicated for the Internal Assessment have been incorporated in the reportdeposited in the department library. The minor project work report has been approved in partial fulfillment as per academic regulations of The National Institute of Engineering, Mysuru.

Signature of the Internal Guides	**Signature of the HoD**
Dr. Jayasri B S	Dr. Anitha R
Professor	Professor and Head
Dept. of CS&E	Dept. of CS&E
NIE, Mysuru	NIE, Mysuru

ACKNOWLEDGEMENT

It is our privilege to express gratitude to all those who inspired me and guided to complete the Minor Project. This minor project work has been accomplished only with the direct or indirect help of many people who have guided me. I am grateful to them.

We wish to express my gratitude to **Dr. Rohini Naga Padma**, Principal, NIE, for her encouragementand support. We also express our deepest gratitude to **Dr. Anitha R**, Professor & HOD Department of Computer Science and Engineering, NIE, Mysuru, for her profound alacrity in my progress and her constant guidance and encouragement.

We would take this opportunity to express our sincere thanks to guide Dr. Jayasri. B. S, Department of Computer Science and Engineering, NIE, for his constant guidance and encouragement.

We Would like to thank all our Professors and Faculty Members of NIE, Mysuru for their suggestions, encouragement, and support.

-**Hemanth Kumar R**

-Ganesha M

-Karthikeya S

TABLE OF CONTENTS

Chapter No.	Description	Page No.
1	Introduction	6
2	System Analysis	7
2.1	Literature Review	7
2.2	Existing solutions and their limitations	10
2.3	Disadvantages of existing solutions	10
2.4	Proposed Solution	10
2.5	Advantages Of the Proposed Solution	11
2.6	Software And Hardware Requirements	11
3	System Design	12
3.1	System Architecture	12
3.2	ER Diagram	14
3.3	Database Diagram	15
3.4	UML Diagram	16
4	System Implementation	17
4.1	Tools used for Implementation	20
5	System Testing	21
5.1	Authentication testing for Admins	23
5.2	Adding of Admins	24
5.3	Adding of Classrooms	25
6	Results	27
7	References	31
8	Conclusion and Future Enhancement	32

LIST OF FIGURES

Fig.No	Description	P. No(s)
1.1	System Interaction Sequence	6
3.1	System Architecture	13
3.2	ER Diagram	14
3.3	Database Diagram	15
3.4	UML Diagram	16
4.1	Code for Navigation Bar	17
4.2	reportWebVitals Connection	18
4.3	Node.js file for initializing Routing	19
5.1(a)	Admin login interface	23
5.2 (a)	Insertion of Admins using user interface	24
5.2 (b)	Tabular View of Admin details	24
5.3(a)	User Interface for adding Classroom Information	26
5.3(b)	Tabular View for Classroom Data	26
6.1	Home page View	27
6.2	UI for Searching for Vacant rooms	27
6.3	UI for Mission description	28
6.4	UI showing TechStack	28
6.5	UI for Admin Login	29
6.6	UI for Navigation Bar	29
6.7	Admin User Interface	30

Chapter 1

INTRODUCTION

Managing lecture hall availability is a significant challenge in many educational institutions. Currently, teachers and administrators often spend valuable time searching for available classrooms, especially during busy periods when multiple events, classes, or exams are scheduled. This manual approach requires staff to check timetables for different branches and sections or even physically inspect classrooms to verify their availability. Such a process is inefficient and prone to errors, leading to scheduling conflicts, underutilized resources, and disruptions in academic activities. The purpose of this project is to develop a "Real-Time Lecture Hall Availability Tracker" to automate the classroom booking and availability management process. By implementing a real-time tracking system, faculty and staff can instantly access up-to-date information on classroom availability, eliminating the need for time-consuming manual checks. This project aims to streamline classroom management, making it easier for teachers and administrators to plan and allocate resources effectively

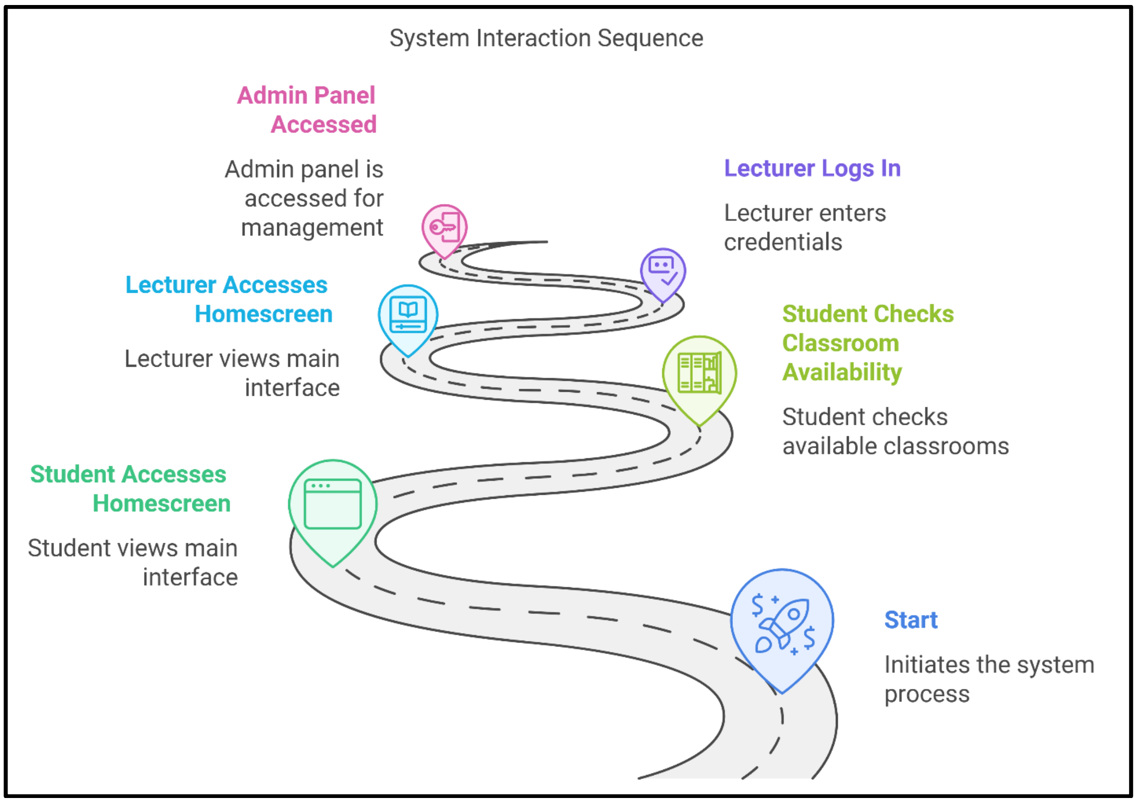

System Interaction Sequence
Admin Panel Accessed
Admin panel is accessed for management
Lecturer Logs In
Lecturer enters credentials
Lecturer Accesses Homescreen
Lecturer views main interface
Student Checks Classroom Availability
Student checks available classrooms
Student Accesses Homescreen
Student views main interface
Start
Initiates the system process

Fig 1.1 – System Interaction Sequence

This project uses a powerful mix of tools to build a responsive and secure web app. The frontend is made with React.js for flexible components and styled with Tailwind CSS for a clean, responsive design. On the backend, Node.js and Express.js work together to handle data efficiently, while MongoDB stores our data securely and scales well. For security, JWT is used for safe, token-based login sessions, keeping user data protected. Introducing a real-time availability tracking system has several benefits. Firstly, it reduces the administrative burden on faculty and staff, allowing them to focus more on their core responsibilities. Secondly, it optimizes classroom utilization by providing visibility into room availability, reducing idle time and ensuring resources are efficiently allocated. Finally, by using the MERN stack (MongoDB, Express.js, React.js, Node.js), this solution is both scalable and efficient, enabling it to handle high user traffic and real-time updates seamlessly. This project not only addresses current limitations but also modernizes campus operations by integrating technology that enhances convenience and efficiency for all users.

Chapter 2

SYSTEM ANALYSIS

2.1 LITERATURE REVIEW:

1. The paper, titled *"Electronic Lecture Time-Table Scheduler Using Genetic Algorithm,"* addresses the challenges associated with manual lecture scheduling in academic institutions. Traditional scheduling systems, which rely heavily on manual processes, are labour-intensive, prone to errors, and often lead to inefficiencies, such as course clashes, venue mismanagement, and lecturer availability issues. This research proposes an Electronic Lecture Time-Table Scheduler (ELTS) based on a Genetic Algorithm (GA) to optimize scheduling and reduce these errors. Time-tabling problems in academic settings are considered non-polynomial hard (NP-hard) due to the complex constraints involved. Prior studies have employed several computational approaches like Evolutionary Algorithms, Heuristic Methods, and Artificial Intelligence to address these optimization challenges. The genetic algorithm (GA) stands out for its efficiency in large, complex search spaces by mimicking the process of natural selection. GAs have been shown to offer robust solutions in scheduling by exploring multiple solution pathways concurrently, which enhances the probability of finding optimal or near-optimal solutions. The literature also categorizes scheduling constraints as hard (essential) and soft (desirable but flexible) constraints, such as preventing lecturer or venue conflicts and ensuring periodic breaks for students and instructors. This study tested the GA-based scheduler in a Nigerian polytechnic, revealing improved scheduling efficiency and reduced time consumption. Statistical analysis, including a paired sample T-test, confirmed that the automated system significantly outperforms the manual approach. This study advocates for adopting ELTS across educational institutions to streamline scheduling and resource allocation [1].

2. The paper, titled *"A Smart Meeting Room Scheduling and Management System with Utilization Control and Ad-hoc Support Based on Real-Time Occupancy Detection,"* presents an intelligent system for managing meeting room schedules and maximizing room utilization. Traditional meeting room scheduling systems rely on predetermined schedules, often leading to underutilization when meetings end early or are cancelled. This research proposes a system that combines real-time occupancy detection with scheduling software to allow for dynamic room availability. Most existing scheduling systems focus on fixed schedules without real-time occupancy updates, limiting their adaptability to ad-hoc needs. Previous methods for occupancy detection include using Bluetooth, cameras, ultrasonic sensors, and CO2 levels. However, these methods have limitations like high costs, privacy concerns, or deployment complexity. The proposed system uses Passive Infrared (PIR) sensor fusion devices, which detect human presence by sensing infrared radiation from the human body. This approach is

cost-effective, energy- efficient, and easy to implement. Data from these sensors is transmitted via Ethernet, which is more reliable and less complex than wireless alternative [2].

3. The paper titled *"Design and Implementation of Hotel Reservation System Using Microsoft Access and Visual Basic .NET"* focuses on creating a computerized reservation system for Hiddig Hotel to improve the efficiency and accuracy of reservations. The existing manual system was prone to errors, data loss, and double bookings, which led to customer dissatisfaction. By adopting a database-driven system, the hotel could automate processes like room reservations, billing, and guest record. The literature highlights the role of **information systems** (IS) in enhancing operational efficiency in organizations, especially within the hospitality sector. Previous studies emphasize that information systems are crucial for data storage, retrieval, and processing, supporting better decision-making. Specifically, **Database Management Systems (DBMS)** have advantages like reduced redundancy, data consistency, and enhanced security. Studies also review different **database architectures** (centralized, distributed, client-server each with unique strengths for supporting operational needs in hotels. **Software Development Life Cycle (SDLC)** models, including waterfall and rapid application development, are discussed as methods for creating structured, reliable systems. The study utilizes Visual Basic .NET and Microsoft Access, chosen for their user-friendly interfaces and powerful data management capabilities, making them suitable for hotel reservation systems. This literature underpins the need for automated solutions in the hotel industry, reinforcing the paper's focus on using a DBMS to improve data integrity, ease of access, and overall customer satisfaction [3].

4. A paper titled "*Human-Computer Interaction in Study Room Reservation Systems*" focuses on designing a user-friendly reservation system to enhance study room scheduling for students in academic settings. The literature in Stadler's study emphasizes the importance of **Human-Computer Interaction (HCI)** in designing effective reservation systems, particularly within academic settings. Previous research highlights the role of **user-centric information systems (IS)** in enhancing operational efficiency, data visualization, and user satisfaction—factors that are especially critical in high-demand environments like university study rooms. Studies indicate that reservation systems for study spaces positively impact students' academic performance by providing access to quiet, resourceful environments, which are associated with higher GPA, retention rates, and student engagement. In examining existing **study room scheduling systems** within academic libraries and universities, the literature presents a range of solutions, from simple Google Forms integrations to proprietary digital systems. Proprietary systems, while generally more efficient and customized, can be costly and complex, requiring designs that are user-friendly and transparent in scheduling processes. For example, Fordham University's pilot of Google Calendar integration demonstrated that such simpler solutions, while effective, often require manual oversight, which limits scalability for larger campuses. The review also covers online reservation systems in other industries, such as the hospitality sector. The H Sovereign Hotel in Bali, for example, used a SWOT analysis to assess its online booking system, which showed benefits like convenience and support for last-minute bookings but also pointed out drawbacks like system complexity and the challenge of handling last-minute cancellations. These studies underline both the advantages and limitations of online reservation systems and stress the need for designs that are user-friendly, clear, and supportive of efficient booking processes [4].

5. The research paper addresses the development and application of computational techniques in optimizing processes or solving complex problems within a specific field. By building on established theories and integrating innovative methodologies, the paper demonstrates how these approaches can lead to performance improvements, cost savings, or increased precision. Through empirical analysis or simulation, the study validates its approach, showing enhancements over traditional methods and underscoring its contributions to both academic and industry contexts. The literature review explores foundational theories and relevant studies that inform the research, covering:(i) Foundational Theories Key theoretical frameworks, such as [examples: Neural Networks, Optimization Algorithms, Machine Learning principles], provide a base for the study's methodologies. These theories are widely recognized as pillars in the field, guiding the development of newer models. :(ii) Prior Studies and Methods: A review of previous work illustrates how others have tackled similar challenges, identified limitations and set the stage for the current study's contributions. The paper critiques these prior methods, highlighting the need for novel approaches to address unresolved issues. (iii) Comparative Methodologies: The literature includes a comparison of various methodologies, pointing out where certain approaches fell short and how the current research bridges these gaps. For example, it might examine supervised vs. unsupervised learning in machine learning, explaining how the proposed solution advances these conventional methods. (iv)Recent Advances: Finally, the review touches on cutting-edge technologies (e.g., deep learning, big data analytics) and their influence on modern research directions. This context highlights the novelty and relevance of the current study within

   the landscape of recent advancements. (v)Together, these insights establish the significance of the study by situating it within both classical foundations and recent innovations, showing its potential impact on further research and practical applications [5]

6. The research paper titled "*Real-Time Classroom Vacancy Monitoring System*" presents an innovative approach to address classroom availability issues in educational institutions, especially on large campuses. This system uses sensors and monitoring tools to detect classroom occupancy and environmental conditions, providing real-time information to students and faculty. The system aims to streamline classroom management by helping students and faculty identify available rooms quickly and safely. It also includes temperature monitoring to alert users of any potential hazards like fire. The project uses an Arduino Uno microcontroller, PIR motion sensors, DHT11 temperature and humidity sensors, and a SIM900A GSM module. These components work together to detect room occupancy, monitor temperature, and send SMS alerts in case of abnormal conditions. (i)Sensor-Based Monitoring: Studies (e.g., Twumasi et al., 2017) demonstrate the effectiveness of PIR sensors in conserving energy by activating classroom lighting and fans based on occupancy. (ii)Occupancy Prediction: Raykov et al. (2016) explored using machine learning with PIR sensors to analyse occupancy patterns, enhancing single PIR sensor accuracy for real-time monitoring. (iii)Wireless Networks: Klingbeil and Wark (2008) showed wireless sensor networks can effectively track movements, contributing to smart infrastructure. (iv)Enhanced PIR Capabilities: Narayana et al. (2015) discussed increasing sensor range and accuracy by combining multiple PIR sensors. (v)GSM Integration: Kakade et al. (2018) demonstrated using DHT11 and GSM for real-time environmental monitoring, adaptable for classroom climate control [6].

2.2 EXISTING SOLUTIONS AND THEIR LIMITATIONS

While various room booking systems exist, they often have limitations when applied to a university setting. Most existing solutions focus on corporate environments or specific facility management needs, and they generally lack the real-time functionality required to address the dynamic nature of a university campus.

1. Current Room Booking Software (e.g., Skedda, Envoy Rooms, EMS): These platforms allow users to book meeting rooms, manage scheduling conflicts, and provide a digital view of room availability. Most of these systems are designed for businesses and don’t account for the unique scheduling patterns in universities, such as different schedules for each branch and section. Furthermore, they do not provide real-time updates on room availability without IoT-based hardware. These systems typically require manual updates, meaning there can be delays between when a room becomes available and when the system reflects that change.

2. University-Specific Systems (Common Timetable and Manual Checks): At the National Institute of Engineering (NIE), there is currently no system in place for instantly checking classroom availability. Checking room availability requires manually reviewing the timetable of each branch and section or physically visiting classrooms to see if they are free. This process is time-consuming and inefficient, especially when classrooms are scattered across different buildings. The manual process of verifying room availability is prone to errors and delays. In peak times, finding an available room can become chaotic, causing unnecessary delays in lectures and events. Moreover, administrative staff often spend valuable time managing room conflicts that could be better utilized on other responsibilities**.**

2.3 DISADVANTAGES OF EXISTING SOLUTIONS

The limitations of existing solutions reveal several pain points in the current manual or semi-automated systems:

1. Manual Effort and Time Consumption: Teachers or students have to check timetables of each branch manually or physically inspect rooms, which is inefficient, especially when scheduling rooms for ad-hoc meetings or make-up classes.
2. Prone to Conflicts and Errors: Without a real-time system, there is a higher likelihood of double-booking rooms or scheduling conflicts, which disrupts academic activities and wastes resources.
3. Lack of Real-Time Updates: Most existing room booking software does not provide real-time updates. Any cancellation or change in the schedule takes time to reflect in the system, which may mislead users into thinking a room is available when it isn’t.
4. Inflexibility for Dynamic Academic Schedules: Universities have dynamic schedules that vary by semester, course, and section. Existing solutions are often not adaptable to handle the complexity of educational scheduling without costly customizations.

2.4 PROPOSED SOLUTION

The proposed "Real-Time Lecture Hall Availability Tracker" aims to address these limitations by creating a web-based system that uses the MERN stack to provide real-time information on classroom availability. The solution includes features specifically tailored for an academic setting, such as automated updates, easy accessibility, and a user-friendly interface for both students and faculty.

2.5 ADVANTAGES OF THE PROPOSED SOLUTION:

1. Real-Time Updates: The use of Socket.IO enables instant updates on classroom availability, so users can quickly see which rooms are free. This reduces the chance of booking conflicts and ensures accurate information.
2. Reduced Manual Effort: By automating the availability checks, the system eliminates the need for teachers and students to review multiple timetables or physically inspect rooms. This saves time and reduces the administrative burden on staff.
3. Enhanced Resource Utilization: The system can analyze room utilization data, helping the administration identify underutilized spaces and optimize room assignments. This can lead to better space management and fewer idle classrooms.
4. User Authentication and Security: With JWT-based authentication, the system restricts access to authorized users only, ensuring that classroom data is secure and access is limited to university staff and students.
5. Scalability and Flexibility: Built with the MERN stack, the system is easily scalable and can handle a large number of users without performance degradation. It can also be extended to include more features, such as booking history and notifications.
6. Cost-Effective and Low Maintenance: Since it is primarily a software-based solution without IoT hardware, the proposed system is more cost-effective and easier to maintain compared to IoT-based room tracking systems.

Implementing this solution at NIE would bring several significant benefits:

1. Increased Efficiency: Faculty and students will have immediate access to real-time classroom availability, significantly reducing time spent searching for available rooms and improving the overall flow of academic activities.
2. Better Academic Experience: With real-time tracking, interruptions due to room unavailability can be minimized, leading to a smoother academic experience. This is especially helpful during exams or events that require flexible scheduling.
3. Data-Driven Decision Making: The system can provide analytics on room usage, helping administrators make data-informed decisions on room allocation, potential renovations, and resource management.
4. Modernization of Campus Operations: Introducing a technology-driven solution will align NIE with modern educational institutions, making the campus operations more streamlined and efficient.

2.6 SOFTWARE AND HARDWARE REQUIREMENTS:

This section includes both software and hardware requirements, as outlined in your document. You can expand this information by breaking down each requirement’s purpose.

1. **Software Requirements:**

1.1 Frontend: React.js to create a responsive, interactive user interface for the booking and viewing functionalities.
1.2 Backend: Node.js (hosted on Vercel) for the server-side logic that manages room data and handles requests.
1.3 Database: MongoDB (preferably MongoDB Atlas) for storing lecture hall availability, booking history, and user data.
1.4 Real-Time Communication: Socket.IO to push real-time updates on room status to connected clients.

1.5 Authentication: JWT for user authentication, ensuring that only authorized personnel can access and make changes.
1.6 Version Control: Git for version control to manage collaborative development.
1.7 Hosting: Vercel to deploy both the frontend and backend, ensuring smooth integration and ease of access.

2. **Hardware Requirements**: Outline the required system configuration (CPU, RAM, storage, etc.), mainly if this is for development and testing purposes.

Chapter 3

SYSTEM DESIGN

3.1 System Architecture

The Class Track system is developed using the MERN stack (MongoDB, Express.js, React.js, Node.js), a modern and scalable framework for full-stack applications. It integrates frontend and backend technologies seamlessly to ensure a smooth user experience and efficient handling of real-time data.

The frontend is built using React.js, a JavaScript library designed for creating dynamic and responsive user interfaces. Reacts' component-based architecture allows for the modular development of UI elements, enabling quick updates and reusability. The interface is styled with Tailwind CSS, offering a consistent and modern design across the application. Students and admins interact with the application through this interface:

1. **Students:** View real-time availability of lecture halls.
2. **Admins:** Manage and update classroom availability via CRUD (Create, Read, Update, Delete) operations. The frontend communicates with the backend via RESTful APIs, fetching and updating classroom data in real time

The backend uses Node.js and Express.js to handle server-side logic and API routing. Express simplifies middleware management, while Node.js provides a scalable, non-blocking, event-driven architecture that ensures fast and efficient processing of requests.
Key backend functionalities include:

1. Providing secure RESTful APIs for retrieving classroom data and managing time slots.
2. Implementing JWT (JSON Web Tokens) authentication for securing admin-level actions.
3. Efficiently handling data synchronization to reflect real-time updates in the frontend.

The application uses MongoDB as its database, which stores data in a document-oriented format. MongoDB's NoSQL structure allows flexibility in schema design, which is particularly useful for storing semi-structured classroom availability data.
Key features of the database include:

1. Collections for classrooms, categorized by attributes.... availability, occupancy, time slots.
2. High-speed querying for real-time availability updates.
3. Horizontal scalability to support large datasets and concurrent users.

How MERN Stack Powers the System

The MERN stack is an ideal choice for the Class Track system because of its seamless ecosystem, where React.js enables dynamic UI rendering with real-time updates, Express.js and Node.js ensure efficient server-side processing and API handling, and MongoDB offers scalable and fast data storage. The system is designed with robust security measures, including user authentication restricted to valid NIE email IDs and JWT-based protection for admin actions, ensuring restricted access and secure data management. Additionally, the use of Node.js and MongoDB enhances scalability, allowing the system to efficiently manage growing traffic and datasets while maintaining high performance and flexibility.

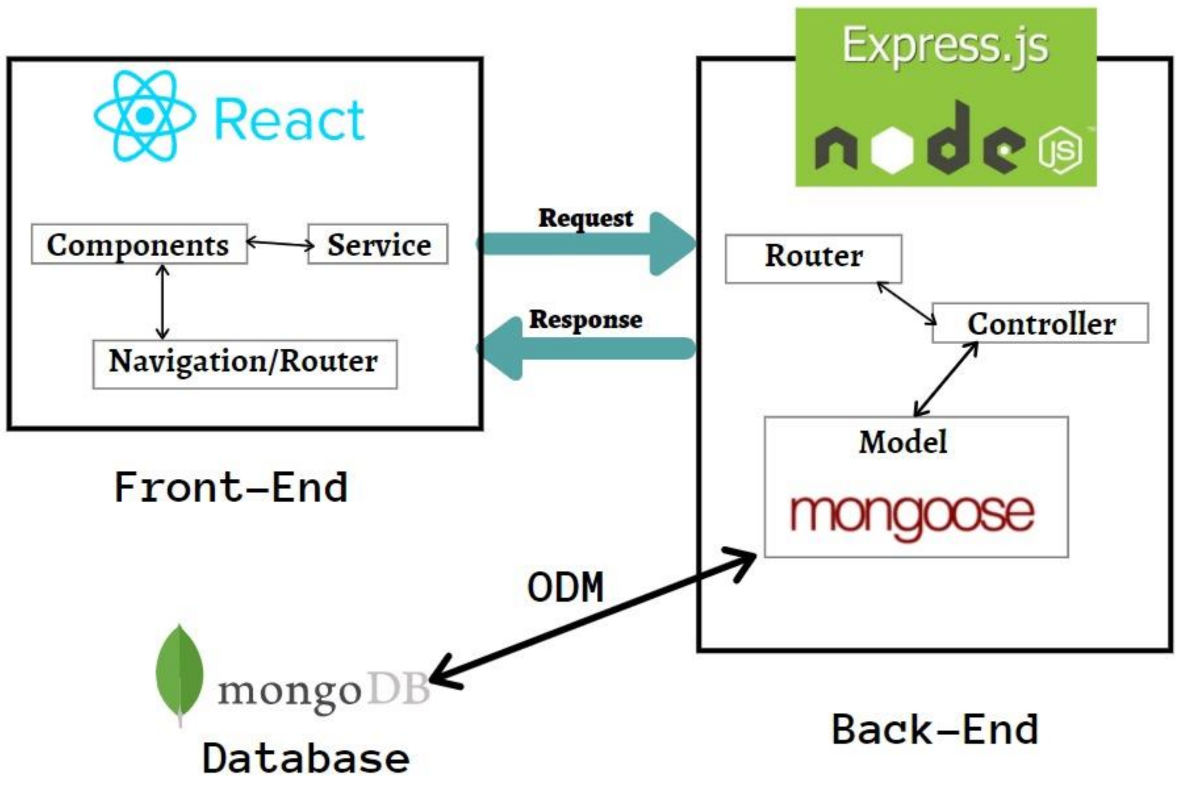


Fig 3.1 – System Architecture

The diagram illustrates the interaction between the frontend, built using React.js, and the backend, developed with Node.js and Express.js, for the Real-Time Classroom Tracker project. The frontend employs React.js to create a dynamic and user-friendly interface, where components are responsible for rendering UI elements. Navigation is handled by the "Router," enabling users to move seamlessly between pages, such as viewing classroom availability or logging in. Services within the frontend manage API requests, ensuring smooth communication with the backend.

On the backend, a Node.js server powered by Express.js manages incoming requests from the frontend. The "Router" directs these requests to appropriate API endpoints, such as those for adding or marking classrooms as vacant. Business logic is encapsulated in the Controller, which processes these requests and ensures proper execution of application functionalities. Data storage and retrieval are facilitated by Mongoose, an Object Data Modeling (ODM) library, which interacts with MongoDB to manage data related to classrooms, admins, and application states. This architecture ensures efficient communication between the frontend and backend, maintaining a responsive and reliable system. The backend also ensures secure data handling by implementing middleware for authentication and validation, safeguarding sensitive operations. Together, the frontend and backend create a cohesive system, delivering real-time updates and a seamless user experience for managing classroom availability.

ER-Diagram:

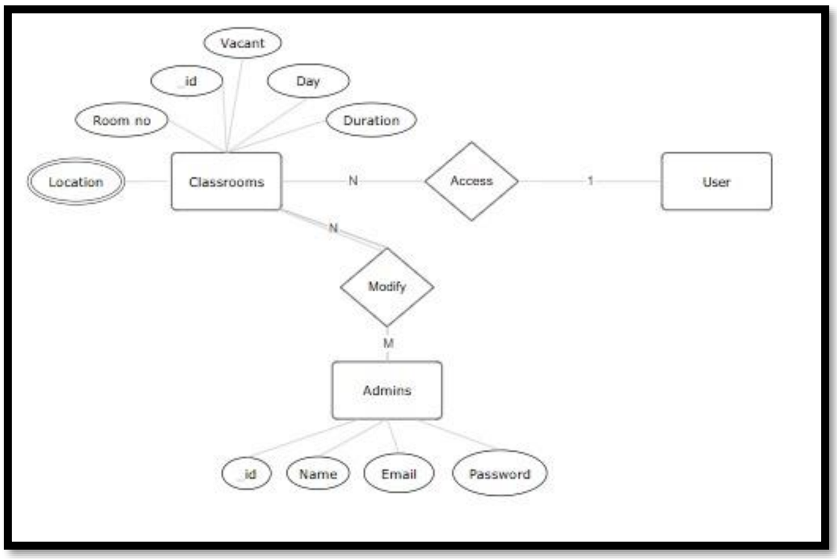


Fig 3.2 – ER diagram

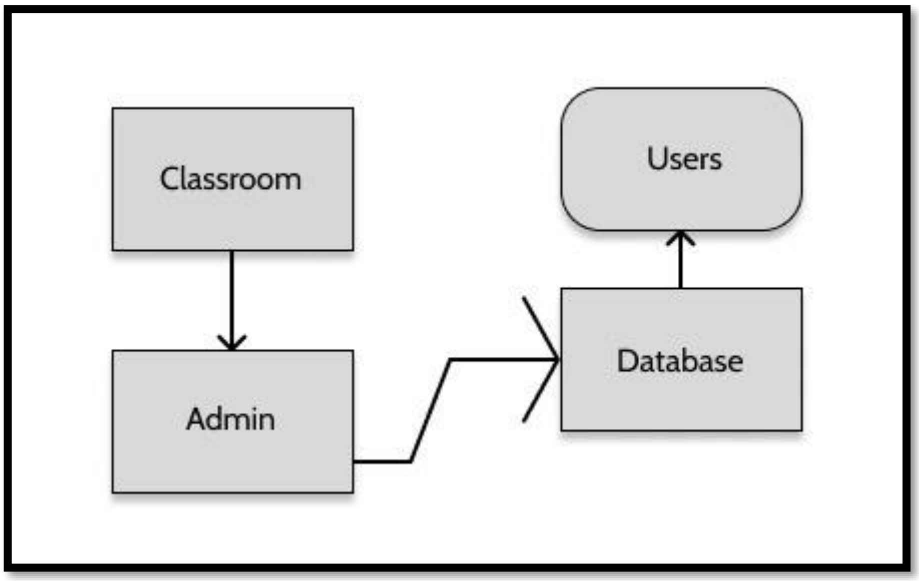
The "Classrooms" entity is used to store information about lecture halls and their availability. It includes several attributes that describe key details about the classrooms. The **Day** attribute specifies the day of the week (e.g., Monday, Tuesday) when the classroom's availability is tracked. The **Duration** indicates the time period during which a particular classroom is reserved or available, such as 10:00 AM to 12:00 PM. The **Room no** uniquely identifies a specific room in a building or facility. The **Location** attribute provides additional context about the room's physical placement, like the building name or floor. Lastly, the **Vacant** attribute, a Boolean value, specifies whether the classroom is currently available (True) or occupied (False). This information is crucial for real-time tracking of classroom usage.

The "Admins" entity stores details about administrators who manage the classroom information. It includes the **Name** attribute to identify the admin, making it easier to keep track of who is managing the system. The **Email** serves as a unique identifier and a point of contact for the admin. Additionally, the **Password** attribute ensures secure access to the system, allowing only authorized users to manage the classroom data. These security measures are implemented to prevent unauthorized access and maintain data integrity.

Although the diagram does not explicitly depict relationships, a logical connection exists between the "Admins" and "Classrooms" entities. Admins are responsible for adding, updating, and managing classroom availability details. For example, an admin might update the "Vacant" status of a classroom or assign its usage to a specific duration and day. This relationship ensures that the system operates efficiently and reliably by assigning control of the database to verified users.
This ER diagram effectively models the structure needed for a real-time lecture hall availability system, emphasizing both data organization and security.

The system's database architecture is designed to streamline classroom availability management while maintaining simplicity, scalability, and security. The *Classroom Collection* stores detailed information about classrooms, including room numbers, availability status, schedules, and durations, enabling administrators to efficiently manage resources. Users interact with this data indirectly through the website, accessing real-time information about vacant classrooms without requiring their data to be stored in the database. The *Admin Collection* holds administrator credentials, such as usernames and passwords, along with permissions for updating, adding, or deleting records, ensuring secure access to sensitive operations. At the core is the *Database*, serving as a central repository for collections like Classroom and Admin, facilitating efficient storage, retrieval, and updates. This architecture ensures a clear separation between end-users and the database while enabling seamless interactions through the website.

3.2 Database Diagram:



–

Fig 3.3–Database Diagram

In the database diagram, the "Classroom" and "Admin" represent collections, which are groups of related data stored in MongoDB. Each collection contains documents, individual records that are structured as key-value pairs in JSON format. In MongoDB, keys are always strings, while values can be various data types such as strings, numbers, Booleans, timestamps, maps, arrays, or even null.

For example, in the Classroom Collection, each document may include keys like "room number," "availability status," "schedule," and "duration," with corresponding values that define each classroom's state and details. This allows for dynamic and flexible storage of classroom-related information, such as the room's availability and schedule.

The Admin Collection holds data related to the administrators, including credentials (such as usernames and passwords) and permissions. This collection helps manage the security and role-based access control of the application, ensuring that only authorized users can add, modify, or view classroom data.

By storing these as collections in MongoDB, the system can easily scale and adapt, enabling real-time updates and efficient management of classroom resources and administrative tasks.

3.3 UML Diagram:

The diagram illustrates the workflow of a real-time classroom tracker system, highlighting the interactions between two primary users: Students and Admins. The process begins with both users logging into the system. Students have the ability to view vacant classrooms, helping them easily find available spaces for their needs. Admins, on the other hand, have additional privileges. They can add new classrooms to the system, mark classrooms as vacant when they are unoccupied, or update their status to occupied when they are in use.
Finally, both users can log out after completing their tasks. This streamlined system ensures efficient management of classroom availability, enhancing the user experience for students seeking space and administrators overseeing resources.

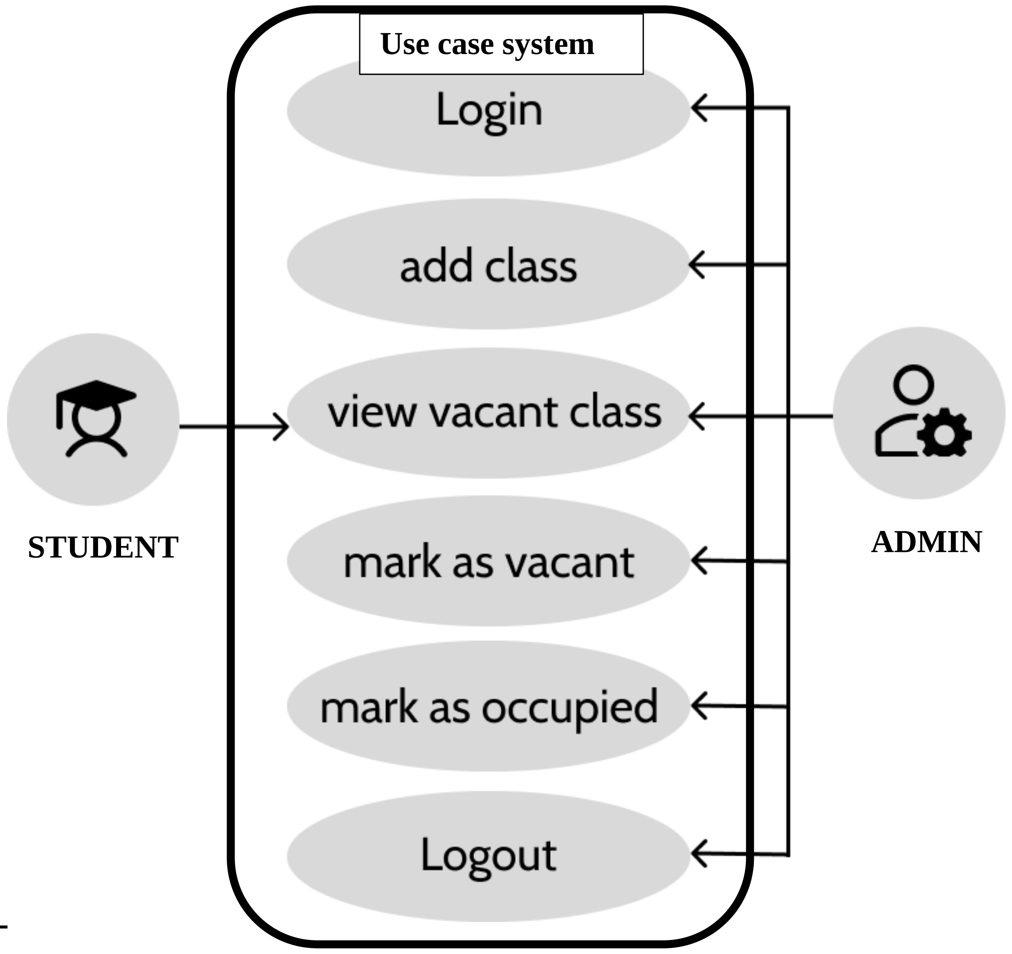


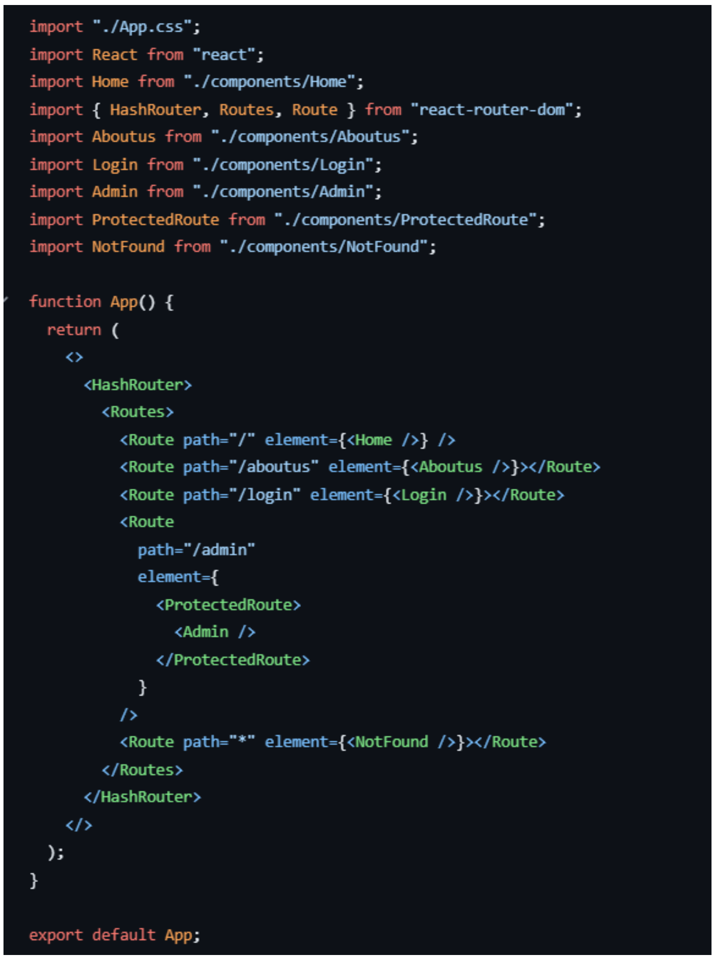
Fig 3.4–UML Diagram

In the UML diagram, the Real-Time Classroom Tracker system includes key use cases: **login**, **add class**, **view vacant class**, **mark as vacant**, **mark as occupied**, and **logout**. These are the actions users can perform within the system. The **student** actor is specifically connected to the **view vacant class** use case, as they can only check the availability of classrooms. The **admin** actor has broader access, being connected to all the use cases. This means the admin can perform all functions, including adding new classes, marking rooms as vacant or occupied, and logging out. The system organizes these use cases such that the student's role is limited to viewing classroom availability, while the admin manages the classroom availability and scheduling process comprehensively.

Chapter 4

SYSTEM IMPLEMENTATION

The Class Track System is developed using the MERN stack, integrating React.js for a dynamic, component-based frontend styled with Tailwind CSS, and Axios for seamless API communication with the backend. The backend, built using Node.js and Express.js, handles secure RESTful APIs, JWT-based authentication, and middleware management, while MongoDB stores semi-structured classroom data with high-speed querying and scalability. The system features real-time updates, role-based access control, and cloud hosting via platforms like Netlify and MongoDB Atlas, ensuring a robust, secure, and efficient implementation tailored for university needs.



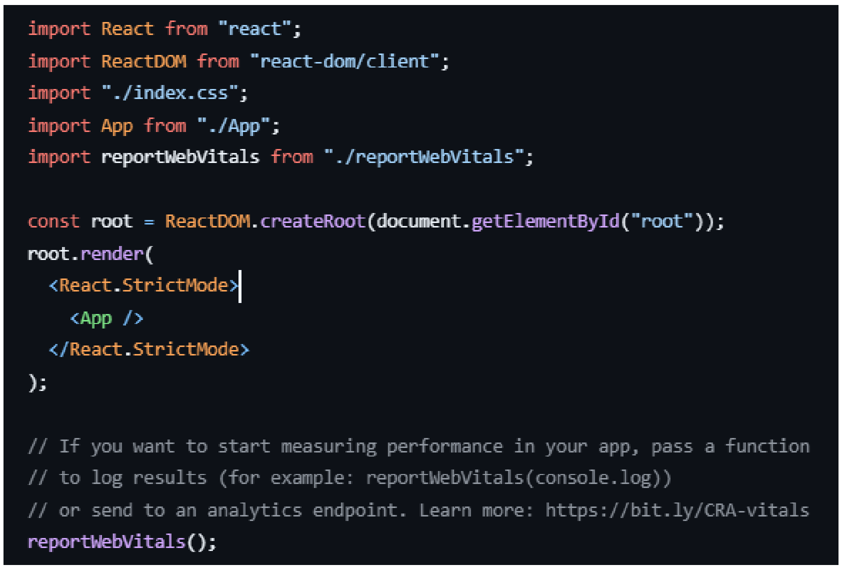
```
import "./App.css";
import React from "react";
import Home from "./components/Home";
import { HashRouter, Routes, Route } from "react-router-dom";
import Aboutus from "./components/Aboutus";
import Login from "./components/Login";
import Admin from "./components/Admin";
import ProtectedRoute from "./components/ProtectedRoute";
import NotFound from "./components/NotFound";

function App() {
  return (
    <>
      <HashRouter>
        <Routes>
          <Route path="/" element={<Home />} />
          <Route path="/aboutus" element={<Aboutus />}></Route>
          <Route path="/login" element={<Login />}></Route>
          <Route
            path="/admin"
            element={
              <ProtectedRoute>
                <Admin />
              </ProtectedRoute>
            }
          />
          <Route path="*" element={<NotFound />}></Route>
        </Routes>
      </HashRouter>
    </>
  );
}

export default App;
```

Fig 4.1– Code for Navigation Bar

This React code sets up a basic single-page application for the Real-Time Classroom Tracker project. It uses React Router to manage navigation and routing within the app, ensuring a seamless user experience when moving between different pages. The application uses Hash Router, which controls navigation through the URL's hash portion, allowing for page updates without full page reloads. The root route ("/") loads the home component, which could be the dashboard showing general classroom availability or a welcome page. The /about us route displays information about the system and its purpose, helping users understand how the system works. The /login route is for authentication, where users (both students and admins) can log in. The /admin route is wrapped inside the Protected Route component, ensuring that only authorized admins can access the administration features like marking rooms as vacant or occupied. The fallback route ("*") redirects undefined URLs to the Not Found component, acting as a 404-error page. This structure ensures smooth navigation while protecting sensitive areas of the application, like the admin panel, which should only be accessible to authorized users, maintaining both security and usability in the real-time classroom tracking system.



```
import React from "react";
import ReactDOM from "react-dom/client";
import "./index.css";
import App from "./App";
import reportWebVitals from "./reportWebVitals";

const root = ReactDOM.createRoot(document.getElementById("root"));
root.render(
  <React.StrictMode>
    <App />
  </React.StrictMode>
);

// If you want to start measuring performance in your app, pass a function
// to log results (for example: reportWebVitals(console.log))
// or send to an analytics endpoint. Learn more: https://bit.ly/CRA-vitals
reportWebVitals();
```

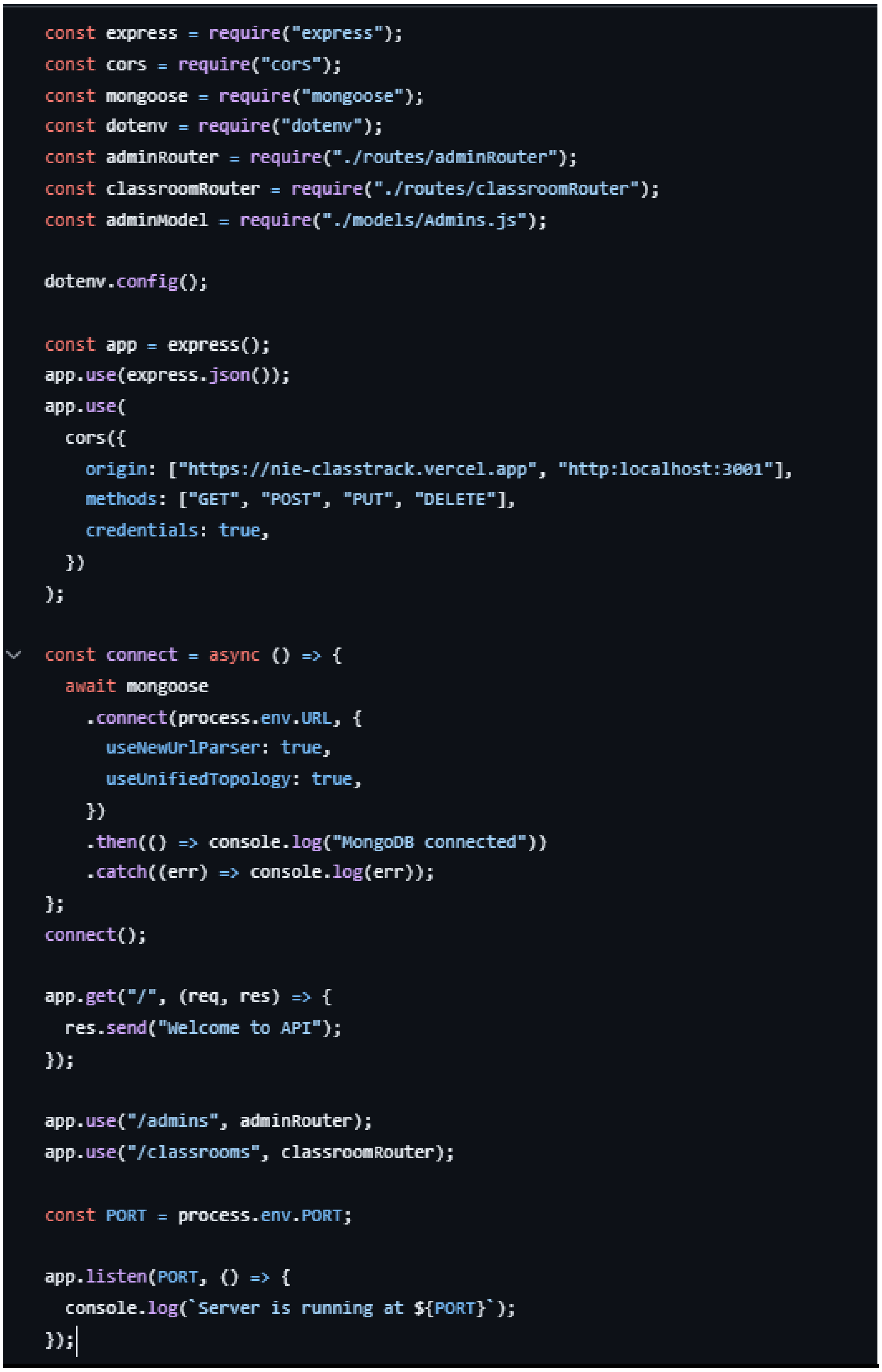
Fig 4.2– reportWebVitals Connection

This React code is the entry point for the Real-Time Classroom Tracker application, managing the initial rendering process and establishing the foundational setup for the app. It begins by importing essential libraries, such as React and React DOM, alongside the main *App* component, a CSS file for styling consistency, and the performance monitoring module *reportWebVitals*. Using ReactDOM's *createRoot* method, the code targets the root DOM element in the HTML file, identified by the ID "root," and renders the React component tree into it.

The rendering process involves wrapping the *App* component in <React.StrictMode>, a development-focused tool that performs additional checks and warnings to identify potential issues early in the development process. This ensures that best practices are followed and promotes code stability and maintainability. The *App* component serves as the central structure of the application, organizing features like classroom tracking, navigation, and booking into a cohesive user interface, enabling seamless interaction.

The *reportWebVitals* function is included to provide insights into the app's performance by tracking critical metrics such as loading times, responsiveness, and overall user experience. This data is invaluable for optimizing performance, especially for applications dealing with real-time data such as classroom availability. Additionally, the styling imported through the CSS file ensures a visually appealing and consistent design throughout the app, enhancing user engagement.

This well-structured setup integrates the application’s core functionality with efficient performance monitoring and styling. It ensures the app is reliable, responsive, and capable of handling dynamic updates while delivering a user-friendly experience tailored to the needs of the Real-Time Classroom Tracker. The use of modern tools and best practices ensures that the system is scalable, maintainable, and optimized for both development and production environments.



```javascript
const express = require("express");
const cors = require("cors");
const mongoose = require("mongoose");
const dotenv = require("dotenv");
const adminRouter = require("./routes/adminRouter");
const classroomRouter = require("./routes/classroomRouter");
const adminModel = require("./models/Admins.js");

dotenv.config();

const app = express();
app.use(express.json());
app.use(
  cors({
    origin: ["https://nie-classtrack.vercel.app", "http:localhost:3001"],
    methods: ["GET", "POST", "PUT", "DELETE"],
    credentials: true,
  })
);

const connect = async () => {
  await mongoose
    .connect(process.env.URL, {
      useNewUrlParser: true,
      useUnifiedTopology: true,
    })
    .then(() => console.log("MongoDB connected"))
    .catch((err) => console.log(err));
};
connect();

app.get("/", (req, res) => {
  res.send("Welcome to API");
});

app.use("/admins", adminRouter);
app.use("/classrooms", classroomRouter);

const PORT = process.env.PORT;

app.listen(PORT, () => {
  console.log(`Server is running at ${PORT}`);
});
```

Fig 4.3– Node.js file for initializing Routing

This Node.js code creates a RESTful API server using the Express.js framework and integrates MongoDB as the database backend, which is crucial for the Real-Time Classroom Tracker project. Here's how it works:

Key Dependencies: The code imports essential dependencies like express, cors, mongoose, and dotenv. These libraries enable routing, database connectivity, security, and environment variable management. The express framework is used to build the server and define routes, while mongoose handles MongoDB operations (such as storing classroom data and admin details). The dotenv package loads sensitive information, such as the MongoDB connection URL and the server's port number, from an external .env file.

1. Middleware Configuration: express.json() parses incoming JSON requests to ensure that the server can interpret the data sent from the frontend (such as class availability updates).cors allows Cross-Origin Resource Sharing between the frontend (running on https://nie-classtrack.vercel.app) and the backend (possibly running locally). This ensures that the frontend can communicate with the backend even if they're hosted on different domains, supporting operations like class availability tracking and updates.
2. MongoDB Integration: The mongoose. Connect () function establishes a connection to the MongoDB database, which is crucial for storing classroom data, admin credentials, and availability statuses. Upon successful connection, the app logs a success message; if it fails, it logs an error. This is where data like classroom availability, admin login credentials, and session information are stored and managed.
3. Routing and Modular Structure: The app has modular routing with separate routes (adminRouter and classroom Router) for handling requests related to admins and classrooms. For example, the adminRouter handles requests to /admins for admin authentication, while classroomRouter manages classroom availability updates at /classrooms. The app's root route ("/") serves a basic welcome message, while other routes are dedicated to managing data.
4. Server Listening: The app listens on a specified port (stored in the environment variables), and when the server starts, it logs a confirmation message to indicate that the API is live and ready to handle requests related to classroom availability and admin management.

This setup creates a robust server for the Real-Time Classroom Tracker, ensuring secure and efficient management of classroom data, user authentication, and communication between the frontend and backend components. It provides a solid foundation for real-time tracking of classroom availability and admin functionality.

4.1 <u>Tools used for Implementation:</u>

The frontend repository, class track, utilizes Next.js, a popular React-based framework designed for building server-rendered or statically exported React applications. The choice of TypeScript as the primary language extends JavaScript by introducing static types, which improves code reliability and developer productivity. Styling is achieved using Tailwind CSS, a utility-first CSS framework that allows developers to build modern UIs efficiently by leveraging pre-designed classes. Dependency management is handled through npm (Node Package Manager), which ensures seamless integration and updates of project libraries. Additional tools used include Post CSS, for transforming CSS with plugins, and ESLint, for linting and maintaining code quality. Deployment is likely managed through Vercel, which is commonly paired with Next.js projects for its ease of use and optimization features. Core dependencies include React libraries (react, react-dom), Next.js font optimization tools (next/font), and Tailwind CSS along with its related plugins such as postcss and autoprefixer.

The backend repository, classtrack-api, is built using Node.js with Express.js, a lightweight and flexible framework ideal for creating robust APIs. The application uses MongoDB, a NoSQL database, for efficient data storage and retrieval. To simplify interactions with MongoDB, Mongoose, an Object Data Modeling (ODM) library, is employed. For managing environment-specific configurations securely, the repository uses dotenv, ensuring sensitive information like API keys remains protected. Authentication mechanisms, although not explicitly detailed, are likely implemented using JWT (JSON Web Tokens) or session-based authentication for secure user verification. Middleware tools such as body-parser and cors are included to handle incoming requests and enable Cross-Origin Resource Sharing. For development convenience, nodemon is used to automatically restart the server upon code changes. Other potential dependencies for authentication might include jsonwebtoken for token management and bcryptjs for password hashing, ensuring secure user credentials.

Chapter 5

SYSTEM TESTING

Testing plays huge role in app development and deployment. There are various types of testing which will be used to validate the functionality of the application. The following list shows the various testing methods. Software testing arrived alongside the development of software, which had its beginnings just after the second world war. Computer scientist Tom Kilburn is credited with writing the first piece of software, which debuted on June 21, 1948, at the University of Manchester in England. It performed mathematical calculations using machine code instructions.
Debugging was the main testing method at the time and remained so for the next two decades. By the 1980s, development teams looked beyond isolating and fixing software bugs to testing applications in real-world settings. It set the stage for a broader view of testing, which encompassed a quality assurance process that was part of the software development life cycle. "In the 1990s, there was a transition from testing to a more comprehensive process called quality assurance, which covers the entire software development cycle and affects the processes of planning, design, creation and execution of test cases, support for existing test cases and test environments," says Alexander Yaroshko in his post on the uTest developer site. "Testing had reached a qualitatively new level, which led to the further development of methodologies, the emergence of powerful tools for managing the testing process and test automation tools." 1
Software testing has traditionally been separated from the rest of development. It is often conducted later in the software development life cycle after the product build or execution stage. A tester may only have a small window to test the code – sometimes just before the application goes to market. If defects are found, there may be little time for recoding or retesting. It is not uncommon to release software on time, but with bugs and fixes needed. Or a testing team may fix errors but miss a release date.

Acceptance Testing
Acceptance testing ensures that the end-user (customers) can achieve the goals set in the business requirements, which determines whether the software is acceptable for delivery or not. It is also known as user acceptance testing (UAT).

Black Box Testing
Black box testing involves testing against a system where the code and paths are invisible. It's

a method of software testing that examines the functionality of the application without peering into its internal structures or working.

End to End Testing
End to end testing is a technique that tests the application's workflow from beginning to end to make sure everything functions as expected.

Exceptions
Testing exceptions and handling the various kinds of exceptions plays huge role. Because in real world situations we cannot always expect perfect scenario. When exceptions occur, we need to design our systems in such a way that it handles exceptions effectively without affecting the user.

Functional Testing
Functional testing checks an application, website, or system to ensure it's doing exactly what it's supposed to be doing.

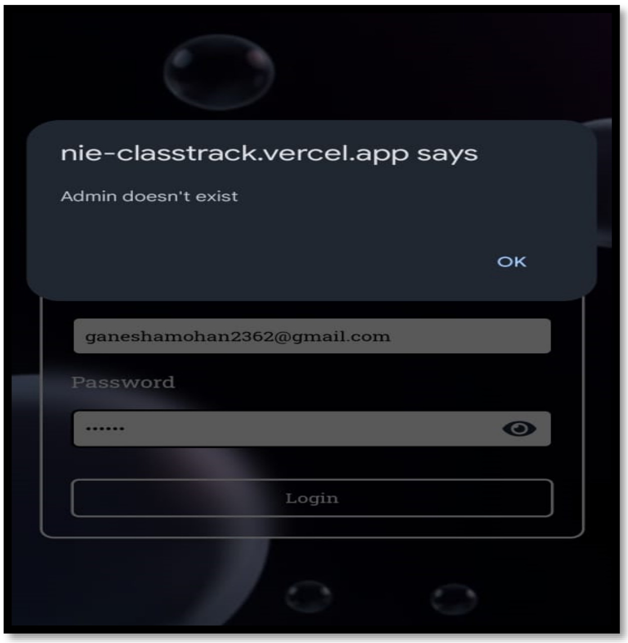
Integration Testing
Integration testing ensures that an entire, integrated system meets a set of requirements. It is performed in an integrated hardware and software environment to ensure that the entire system functions properly.

Authentication Testing
This involves the testing authentication feature of the application. Checking different aspects like security and integrity of user matters a lot during this type of testing. Authentication involves registering the user to the application, then verification mail will be sent to verify whether the user is genuine or not. After verification, user can use his credentials to login to the application.

Application Testing Checklist			
Tested By	Tester 1: Hemanth Kumar R Tester 2: Ganesha M Tester 3: Karthikeya S	Date	08/12/2024
Application Name	Real Time Lecture Hall Availability Tracker		
Procedure	Expected Result	Pass/Fail (P/F)	Actual Results/Comments
Application Functionality			
Performs primary functionality and maintains stability	Yes	P	Opens and allows students to access the vacant classrooms
Fundamental features			

Installs on android devices (android 10 Or above)	Yes	P	Installs normally
Web/IOS Support	Yes	NA	Works on web version .IOS version not tested.
Authentication			
Register function	Yes	P	No errors
Login function	Yes	P	No errors
Restricts unauthorized access	Yes	P	Works properly
UI testing			
UI renders properly	Yes	P	No issues

5.1 Authentication testing for Admins:


nie-classtrack.vercel.app says

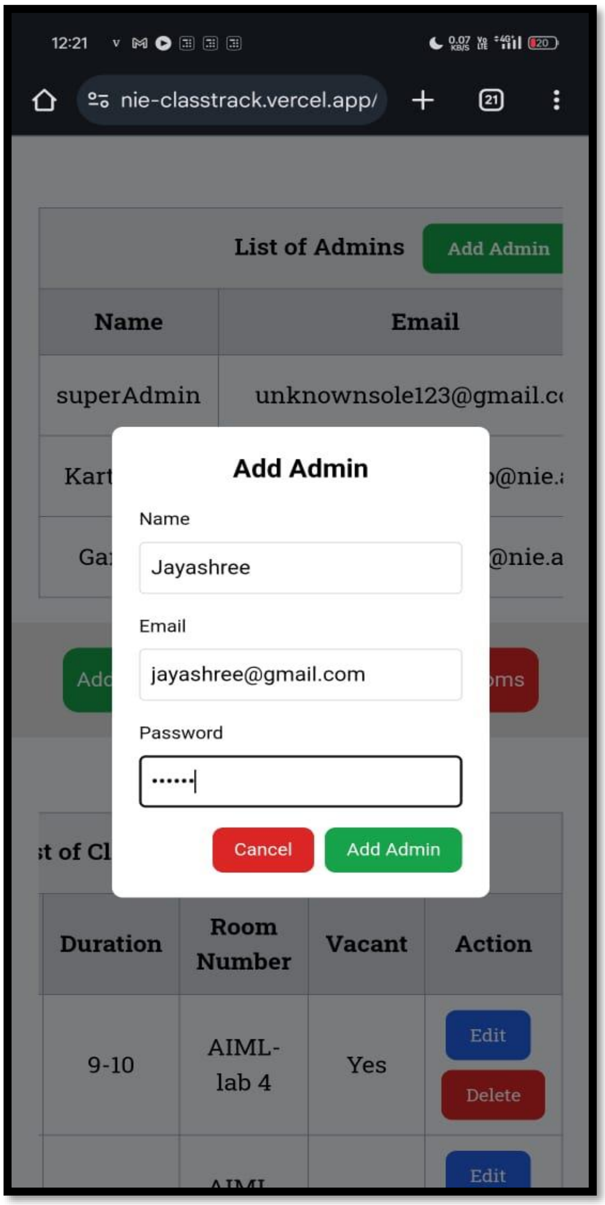
Admin doesn't exist

Fig 5.1 (a). Admin login interface

The fig5.1(a) shows a failed login attempt for the admin portal of the website, displaying an error message that reads, "Admin doesn’t exist." In this case, the user entered the email address ganeshamohan2362@gmail.com, which the system did not recognize as belonging to an admin. This suggests that either the email address is incorrect or the user does not have admin privileges.
In contrast, the fig5.1(b) shows a successful login attempt. The system displays a message saying, "Logged in successfully," indicating that the credentials provided were correct and

associated with an admin account. Here, the email used is 2022cs_ganesham_b@nie.ac.in, which the system recognizes as valid for an admin. This comparison highlights that only specific registered admin emails are granted access to the portal.

5.2 Adding of Admins:



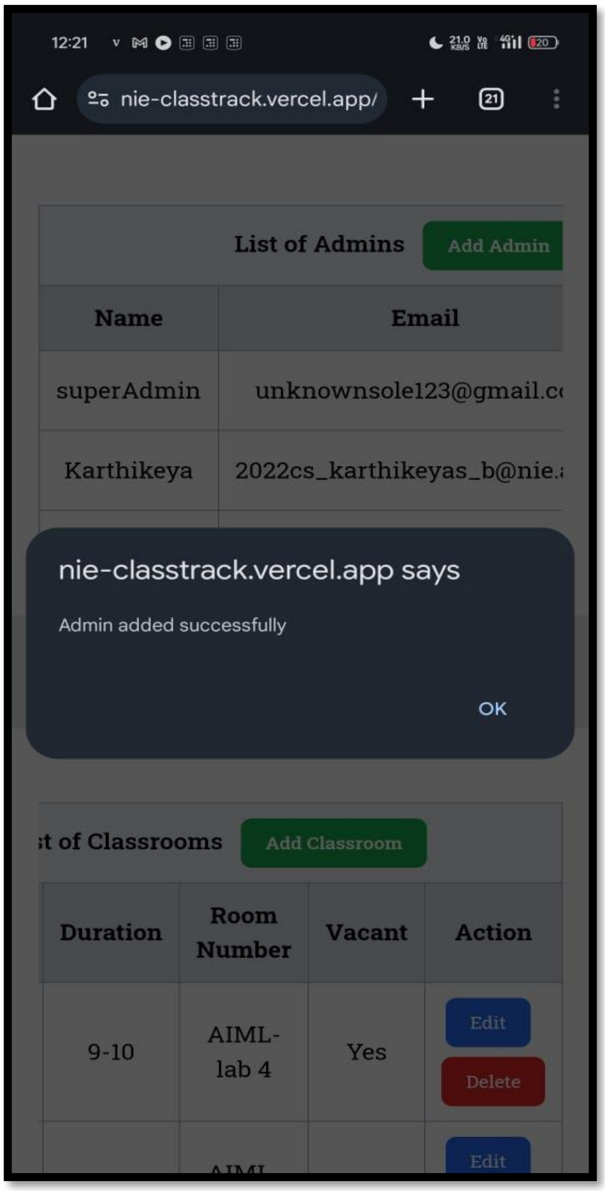
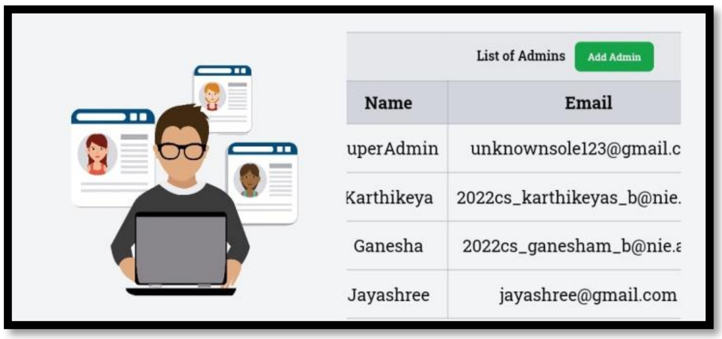


Fig 5.2 (a). Insertion of Admins using user interface



List of Admins Add Admin

Name	Email
uperAdmin	unknownsole123@gmail.c
Karthikeya	2022cs_karthikeyas_b@nie.
Ganesha	2022cs_ganesham_b@nie.a
Jayashree	jayashree@gmail.com

Fig 5.2 (b). Tabular View of Admin details

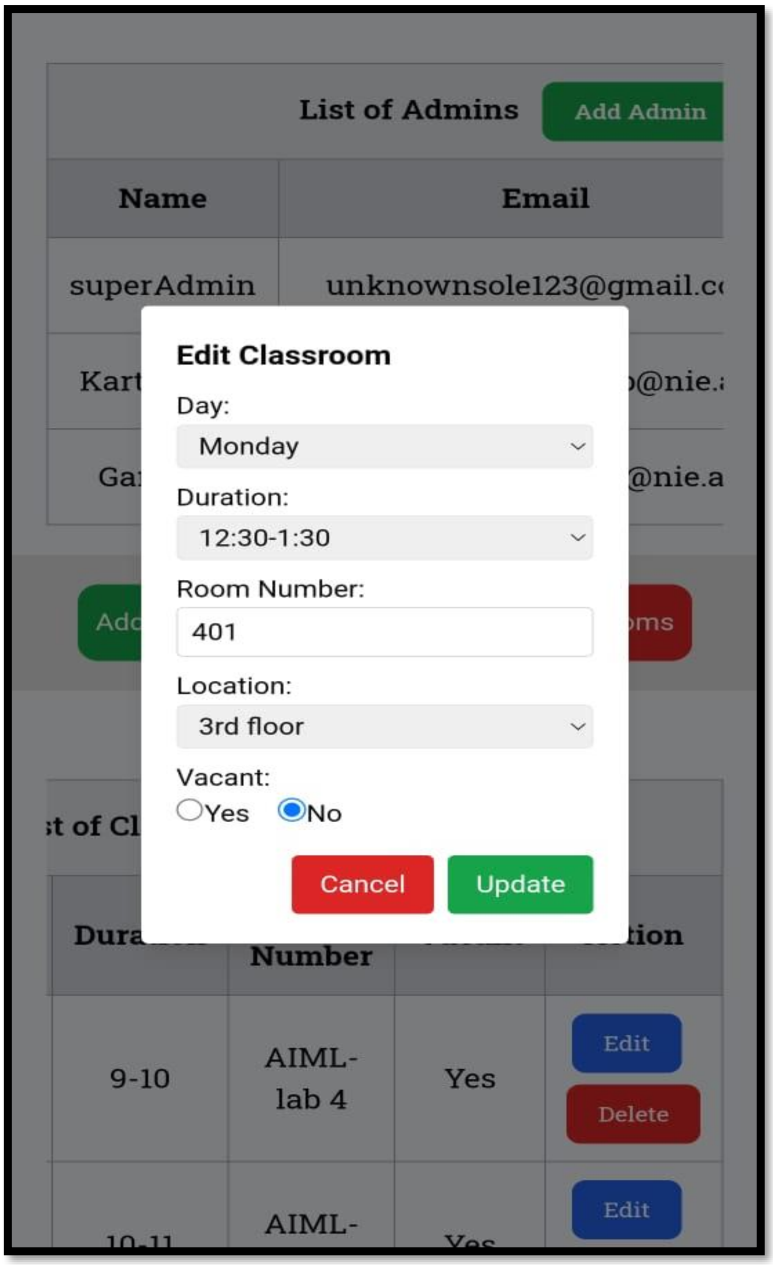
The image demonstrates the process of adding a new admin in the "nie-classtrack.vercel.app" interface. In the fig 5.2(a) a form is displayed where the user inputs details for a new admin. The form fields include the name "Jayashree," the email "jayashree@gmail.com," and a password field. There are two buttons at the bottom of the form: "Cancel" and "Add Admin." This form allows authorized users to add new admins by providing the necessary credentials.
The fig 5.2(b) shows the confirmation message after successfully adding the new admin. The message reads, "nie-classtrack.vercel.app says: Admin added successfully." The background of this screenshot reveals an updated list of admins, reflecting the addition of the new entry. This ensures that the system provides immediate feedback about the success of the action.
In the bottom section fig 5.2(c), there is a detailed view of the updated "List of Admins." The table now includes four entries: superAdmin with the email "unknownsole123@gmail.com," Karthikeya with the email "2022cs_karthikeyas_b@nie.ac.in," Ganesha with the email "2022cs_ganesham_b@nie.ac.in," and the newly added Jayashree with the email "jayashree@gmail.com." This table showcases all the current admins and their respective email addresses, confirming that the new admin has been successfully added to the system.
Overall, the images highlight a straightforward and user-friendly process for managing admin accounts within the interface, with clear feedback and functionality for adding new admins.

5.3 <u>Adding of Classrooms:</u>

This fig 5.3(a,b) shows the process of editing classroom details in the "nie-classtrack.vercel.app" interface. In the top section, a form titled "Edit Classroom" is displayed. The form includes fields for selecting or entering details such as the Day (set to Monday), Duration (set to 12:30-1:30), Room Number (set to 401), and Location (set to 3rd floor). There is also an option to indicate if the room is vacant, with "No" selected. The form provides two buttons: "Cancel" to discard changes and "Update" to save the edits.
The bottom section displays the updated classroom information in a tabular format. The table shows the duration 12:30-1:30, room number 401, and vacancy status marked as "No." Next to this entry, there are two action buttons: a blue "Edit" button for modifying the details and a red "Delete" button for removing the entry.
This interface highlights a user-friendly system for managing classroom schedules, allowing users to update, edit, or delete entries easily. The clear visual cues, such as color-coded buttons for actions, enhance the user experience by providing intuitive functionality for schedule

management.

Fig 5.3 (a). User Interface for adding Classroom Information



List of Admins | Add Admin

Name	Email
superAdmin	unknownsole123@gmail.c
Kart	@nie.
Ga	@nie.a

Edit Classroom

Day: Monday

Duration: 12:30-1:30

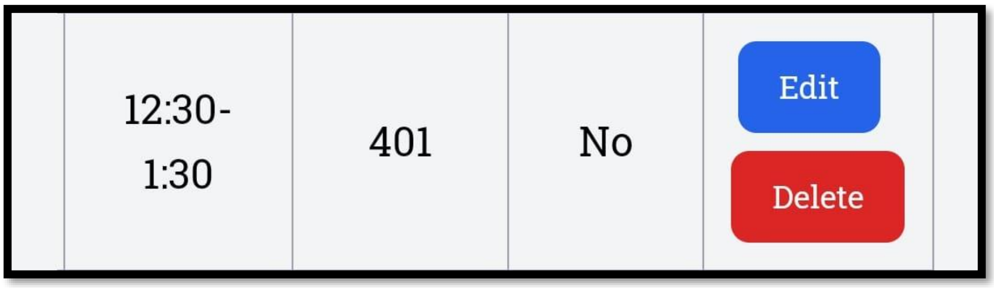
Room Number: 401

Location: 3rd floor

Vacant: Yes [✓ No]

Cancel | Update

Dura	Number		tion
9-10	AIML-lab 4	Yes	Edit Delete
10-11	AIML-	Yes	Edit

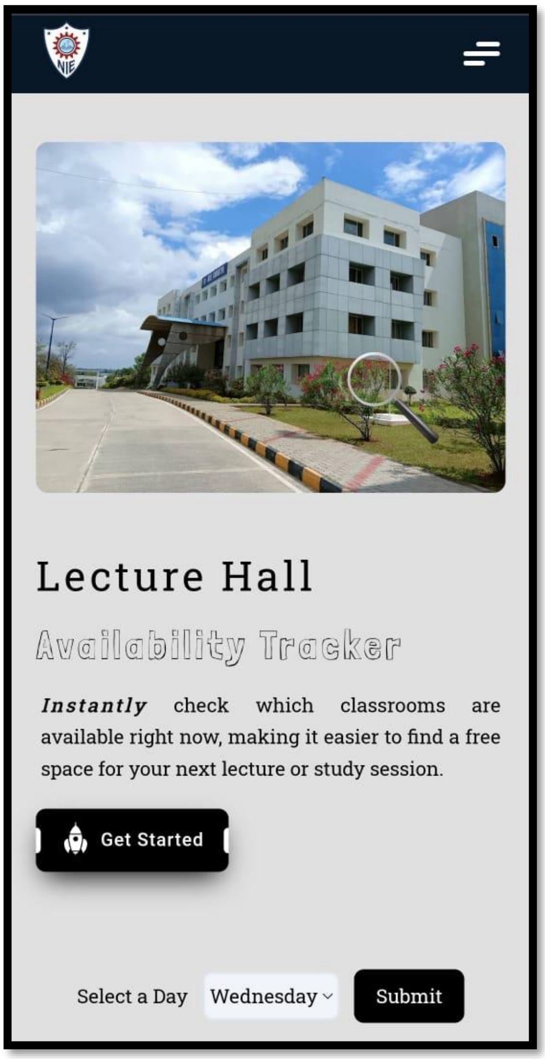


12:30-1:30	401	No	Edit Delete

Fig 5.3 (b). Tabular View for Classroom Data

Chapter 6

RESULTS

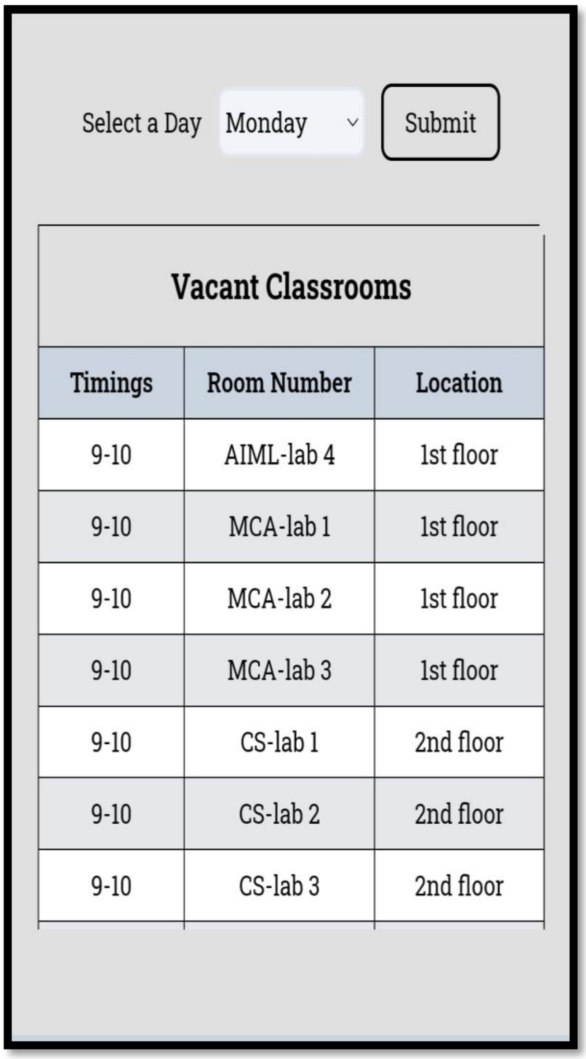


Lecture Hall

Availability Tracker

Instantly check which classrooms are available right now, making it easier to find a free space for your next lecture or study session.

Fig 6.1. Home page View

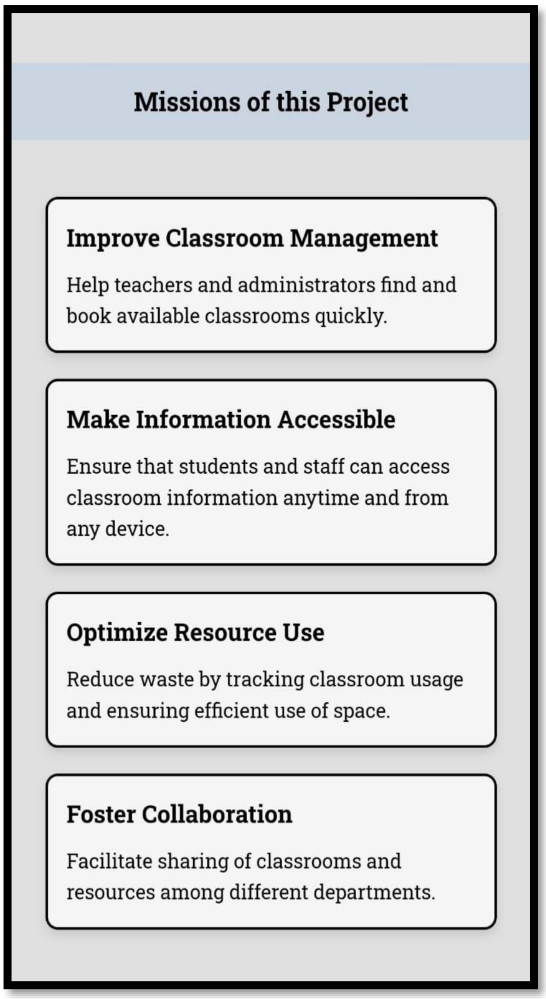


Vacant Classrooms

Timings	Room Number	Location
9-10	AIML-lab 4	1st floor
9-10	MCA-lab 1	1st floor
9-10	MCA-lab 2	1st floor
9-10	MCA-lab 3	1st floor
9-10	CS-lab 1	2nd floor
9-10	CS-lab 2	2nd floor
9-10	CS-lab 3	2nd floor

Fig 6.2. UI for Searching for Vacant rooms

These figures (6.1,6.2) illustrate a web application designed to track lecture hall availability. The homepage features a visually appealing layout with an image of the institution building and a title, Lecture Hall Availability Tracker. It provides a brief description of the app’s functionality, allowing users to instantly check available classrooms for lectures or study sessions. A prominent "Get Started" button and a dropdown menu for selecting a specific day make it easy for users to interact with the application.
The application also includes a page that displays a structured table listing vacant classrooms. This page allows users to select a day (e.g., Monday) and view detailed information about available classrooms, including time slots, room numbers, and their respective locations. This feature helps users efficiently find suitable spaces for their needs



Missions of this Project

Improve Classroom Management

Help teachers and administrators find and book available classrooms quickly.

Make Information Accessible

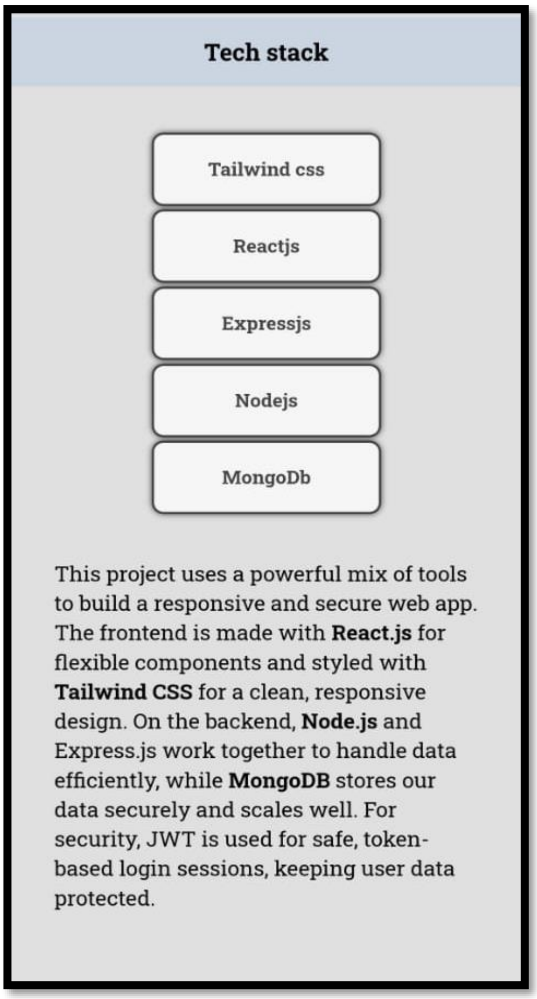
Ensure that students and staff can access classroom information anytime and from any device.

Optimize Resource Use

Reduce waste by tracking classroom usage and ensuring efficient use of space.

Foster Collaboration

Facilitate sharing of classrooms and resources among different departments.



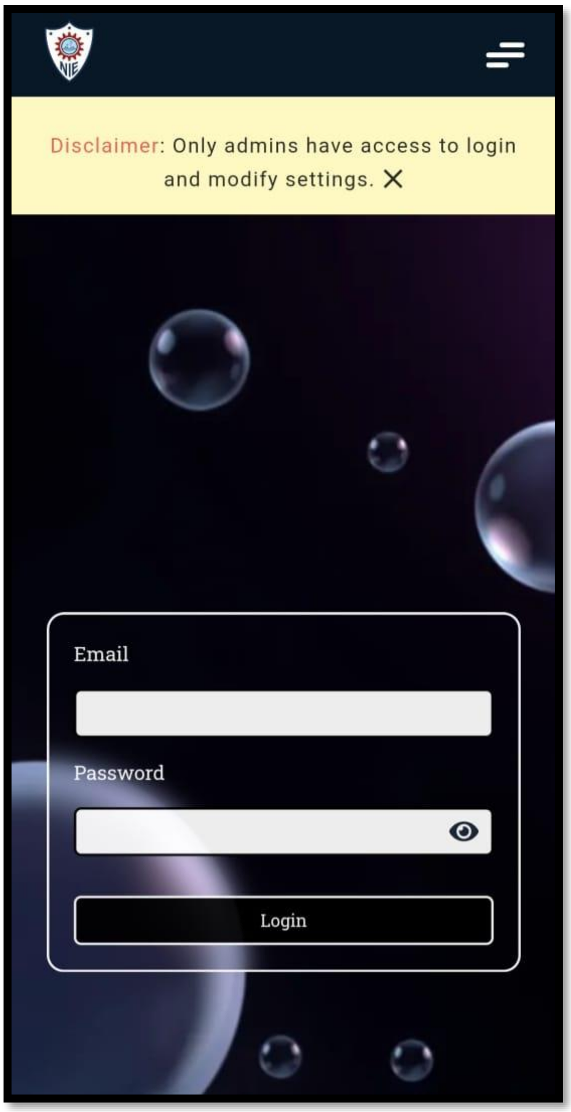
Tech stack

This project uses a powerful mix of tools to build a responsive and secure web app. The frontend is made with **React.js** for flexible components and styled with **Tailwind CSS** for a clean, responsive design. On the backend, **Node.js** and Express.js work together to handle data efficiently, while **MongoDB** stores our data securely and scales well. For security, JWT is used for safe, token-based login sessions, keeping user data protected.

Fig 6.3. UI for Mission description **Fig 6.4. UI showing TechStack**

The mission of this project revolves around improving the efficiency of classroom management. It aims to help teachers and administrators book and manage classrooms with ease. Additionally, the application ensures that classroom information is accessible from anywhere, allowing for informed decision-making. By optimizing the use of available resources, it helps in tracking classroom usage and avoiding conflicts over room allocation. Moreover, the project fosters collaboration by facilitating the sharing of classroom spaces across different departments. This comprehensive solution is designed to enhance productivity and make classroom management a hassle-free experience (fig 6.3)

The application leverages a powerful and modern tech stack to ensure a responsive, secure, and scalable solution. Tailwind CSS is used for styling the frontend, providing a clean and responsive design, while ReactJS is utilized to create a dynamic and interactive user interface. The backend is powered by ExpressJS and NodeJS, enabling efficient handling of data operations. MongoDB serves as the database, offering a secure and scalable solution for storing information. To enhance security, JWT (JSON Web Tokens) is implemented for token-based login sessions, ensuring user data remains protected (fig 6.4).


Disclaimer: Only admins have access to login and modify settings.
Email
Password
Login

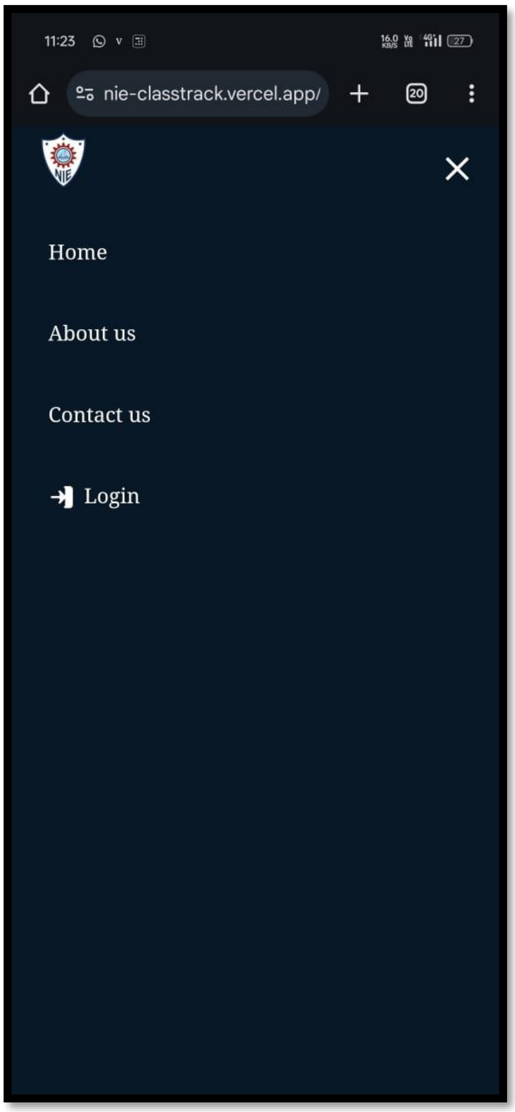
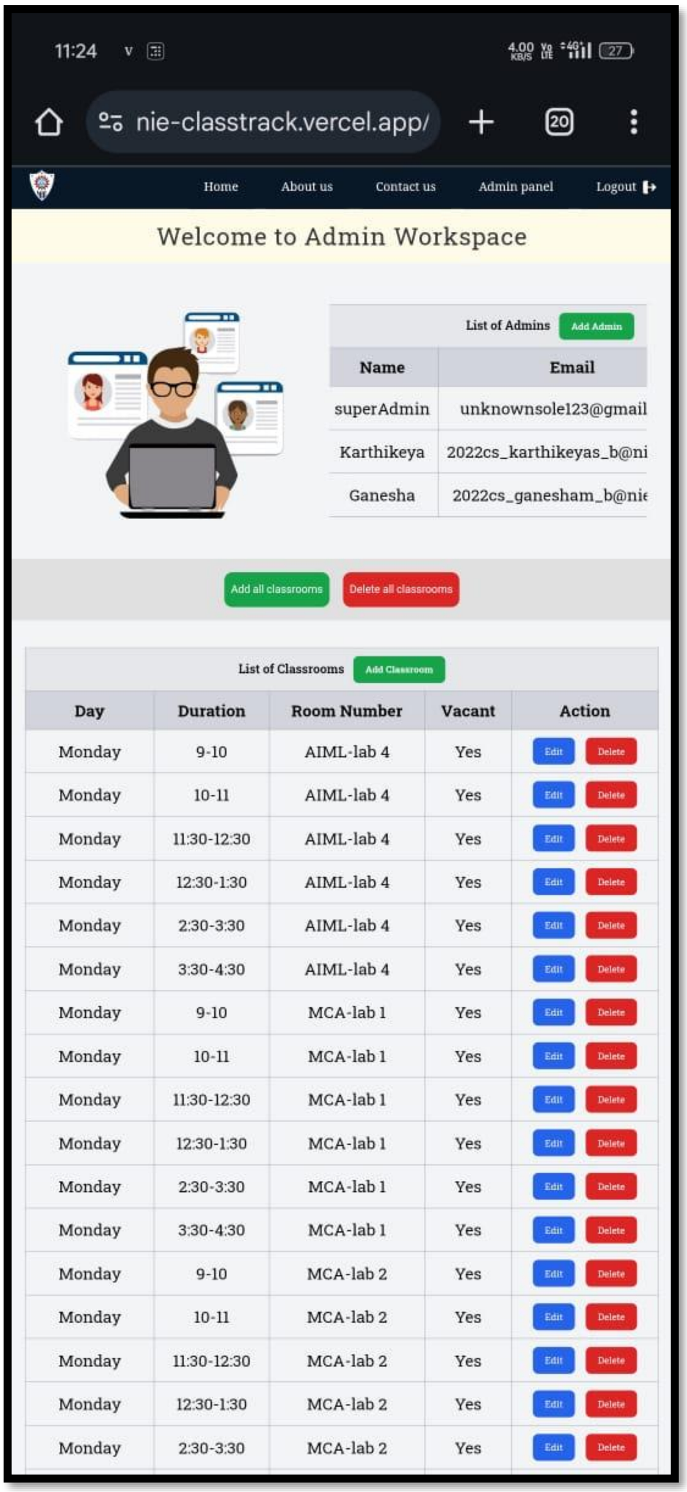


Fig 6.5.UI for Admin Login **Fig 6.6 . UI for Navigation Bar**

The admin interface (fig 6.5) is designed to ensure that only authorized users can access sensitive areas of the system, as shown by the dedicated admin login page. This page emphasizes the importance of restricted access by displaying a disclaimer stating that only administrators have the necessary permissions to modify system settings. The login form is straightforward, featuring fields for the email and password required to authenticate users, alongside a toggle for password visibility to enhance user convenience. The entire interface is set against a modern, visually appealing background, which includes decorative elements designed to provide an engaging and professional user experience. This design choice ensures that administrators can easily navigate the interface while maintaining focus on the task at hand.

In addition, the inclusion of a hamburger menu (fig 6.6) further streamlines navigation, making it easier for users to access essential sections of the application. The collapsible menu is thoughtfully organized with options such as Home, About Us, Contact Us, and Login. These sections are clearly labeled, contributing to a clean, organized layout that ensures the user interface remains intuitive and uncluttered. By offering a simple yet effective way to access various parts of the app, the hamburger menu significantly enhances the app's overall usability. The combination of these features ensures that the admin interface is both secure and user-friendly. It allows administrators to manage the system's settings and monitor classroom availability with ease. At the same time, the carefully considered design choices maintain a professional and aesthetically pleasing atmosphere. This thoughtful approach to both functionality and design helps streamline the classroom tracking process while delivering an intuitive experience for authorized users. The app remains accessible, organized, and ready to support real-time classroom management effectively.



Welcome to Admin Workspace

List of Admins

Name	Email
superAdmin	unknownsole123@gmail
Karthikeya	2022cs_karthikeyas_b@ni
Ganesha	2022cs_ganesham_b@nie

List of Classrooms

Day	Duration	Room Number	Vacant	Action
Monday	9-10	AIML-lab 4	Yes	Edit Delete
Monday	10-11	AIML-lab 4	Yes	Edit Delete
Monday	11:30-12:30	AIML-lab 4	Yes	Edit Delete
Monday	12:30-1:30	AIML-lab 4	Yes	Edit Delete
Monday	2:30-3:30	AIML-lab 4	Yes	Edit Delete
Monday	3:30-4:30	AIML-lab 4	Yes	Edit Delete
Monday	9-10	MCA-lab 1	Yes	Edit Delete
Monday	10-11	MCA-lab 1	Yes	Edit Delete
Monday	11:30-12:30	MCA-lab 1	Yes	Edit Delete
Monday	12:30-1:30	MCA-lab 1	Yes	Edit Delete
Monday	2:30-3:30	MCA-lab 1	Yes	Edit Delete
Monday	3:30-4:30	MCA-lab 1	Yes	Edit Delete
Monday	9-10	MCA-lab 2	Yes	Edit Delete
Monday	10-11	MCA-lab 2	Yes	Edit Delete
Monday	11:30-12:30	MCA-lab 2	Yes	Edit Delete
Monday	12:30-1:30	MCA-lab 2	Yes	Edit Delete
Monday	2:30-3:30	MCA-lab 2	Yes	Edit Delete

Fig 6.7. Admin User Interface

This web application is designed to streamline classroom management for administrators. The primary interface, as shown in the admin workspace dashboard, provides a clear and organized view of classroom schedules. It displays information such as the day, duration, room number, and whether the room is vacant. Admins can use the action buttons to make changes to the schedule, ensuring efficient management of classroom resources.

Chapter 7

REFERENCES

Here are the links for the references used in the *Class Track* project
Frontend References

1. React.js Official Documentation
2. Tailwind CSS Documentation
3. Vercel Deployment Documentation

Backend References

1. Node.js Documentation
2. Express.js Documentation
3. MongoDB Documentation
4. Mongoose Documentation
5. dotenv npm Package
6. JWT Introduction
7. CORS npm Package
8. Nodemon npm Package

GitHub Repositories

- Frontend Repository
- Backend Repository

References for the research papers:

[1]. J. Soyemi, J. Akinode and S. Oloruntoba, "Electronic Lecture Time-Table Scheduler Using Genetic Algorithm," *2017 IEEE 15th Intl Conf on Dependable, Autonomic and Secure Computing, 15th Intl Conf on Pervasive Intelligence and Computing, 3rd Intl Conf on Big Data Intelligence and Computing and Cyber Science and Technology*
[2]. L. Duc Tran *et al.*, "A smart meeting room scheduling and management system with utilization control and ad-hoc support based on real-time occupancy detection," *2016 IEEE Sixth International Conference on Communications and Electronics*
[3]. Y. A. Mohamed, "Design and Implementation of Hotel Reservation System Using Microsoft Access and Visual Basic .NET," Bachelor's thesis, Kampala International University, 2010.
[4]. C. Stadler, "Human-Computer Interaction in Study Room Reservation Systems," Undergraduate Research Scholars Thesis, Texas A&M University, 202
[5]. A. Mitawa and S. Samreen, "Enhancing E-commerce Platforms through MERN Stack Implementation," *Journal of Emerging Technologies and Innovative Research (JETIR)*, vol. 11, no. 5, pp. 709-716, May 2024.
[6]. S. M. Dano, W. L. Ingente, A. Lagamao, and S. Mahilum, "Real-Time Classroom Vacancy Monitoring System," College of Engineering, Architecture and Industrial Design, Bohol Island State University, Tagbilaran City, Bohol, Presentation, July 2022.

Class track website(live link to our website)

Chapter 8

CONCLUSION AND FUTURE ENHANCEMENTS

The Real-Time Classroom Tracker is a system that helps manage classroom availability, scheduling, and resource use. It’s built with the MERN stack (MongoDB, Express.js, React.js, and Node.js), offering a secure, user-friendly platform. The system makes classroom management easier by providing real-time tracking and minimizing scheduling conflicts. It ensures data security and efficient resource allocation.

For future improvements, the system could use AI to predict classroom usage trends and optimize schedules. A mobile app could provide real-time notifications and easy access. Integrating IoT devices like smart locks and sensors would automate tracking. Adding SMS or email alerts for schedule changes would keep users informed. The system could also support multiple institutions, with customizable dashboards and hierarchical access. Offline functionality would ensure the system works even without internet access. By syncing with college timetables, the system can track rooms and resources more efficiently, improving overall management.