

Full Stack Development -1 (BISFS515)

**MINI PROJECT REPORT**

**On**

**“Campus Portal”**

## Submitted By:

|  |  |
| --- | --- |
| **Nitin Sastry** | **4VV23IS106** |
| **K M Chandan Kumar** | **4VV23IS084** |
| **HV Naga Prahas** | **4VV23IS069** |

**Submitted To:**

**DR. R KASTURI RANGAN,**

**Assistant Professor,**

**Department of Information Science and Engineering, VVCE**

# INDEX

|  |  |  |
| --- | --- | --- |
| 1 | Introduction | 2 |
| 2 | Problem Statement | 2 |
| 3 | System Design (UML Diagrams) | 3 |
| 4 | Proposed Solution | 4 |
| 5 | Implementation | 4 |
| 6 | Conclusion | 7 |
| 7 | References | 8 |
| 8 | Certificates | 8 |
| 9 | GitHub link | 10 |

**INTRODUCTION: -**

The Campus Portal Management System is a web-based application designed to digitalize and streamline the core administrative processes within an educational institution. Built using modern web technologies, it enables administrators to manage admission details, placement records, and cultural event information through a unified digital platform. Students and staff can access updated information in real time, eliminating manual documentation and reducing dependency on disconnected systems. This ensures faster, cleaner, and more organized academic data management across the campus.

The system incorporates a backend service developed using Node.js and Express, which handles authentication, admission data processing, placement record management, and secure operations for all institutional datasets. MongoDB is used as the primary database for storing seat allocations, placement statistics, event schedules, and administrator credentials. The frontend is developed using Angular, providing a responsive, intuitive, and easily navigable interface for campus administrators.

The Campus Information Management System increases operational efficiency, minimizes paperwork, and offers a smooth experience for administrators responsible for managing academic and event-related information.  
The development workflow supports quick deployment, real-time updates, and streamlined handling of admission data, placement records, and cultural event details.

# PROBLEM STATEMENT: -

Traditional academic management processes in many educational institutions still depend on manual registers, Excel sheets, and physical documentation. This method leads to several challenges:

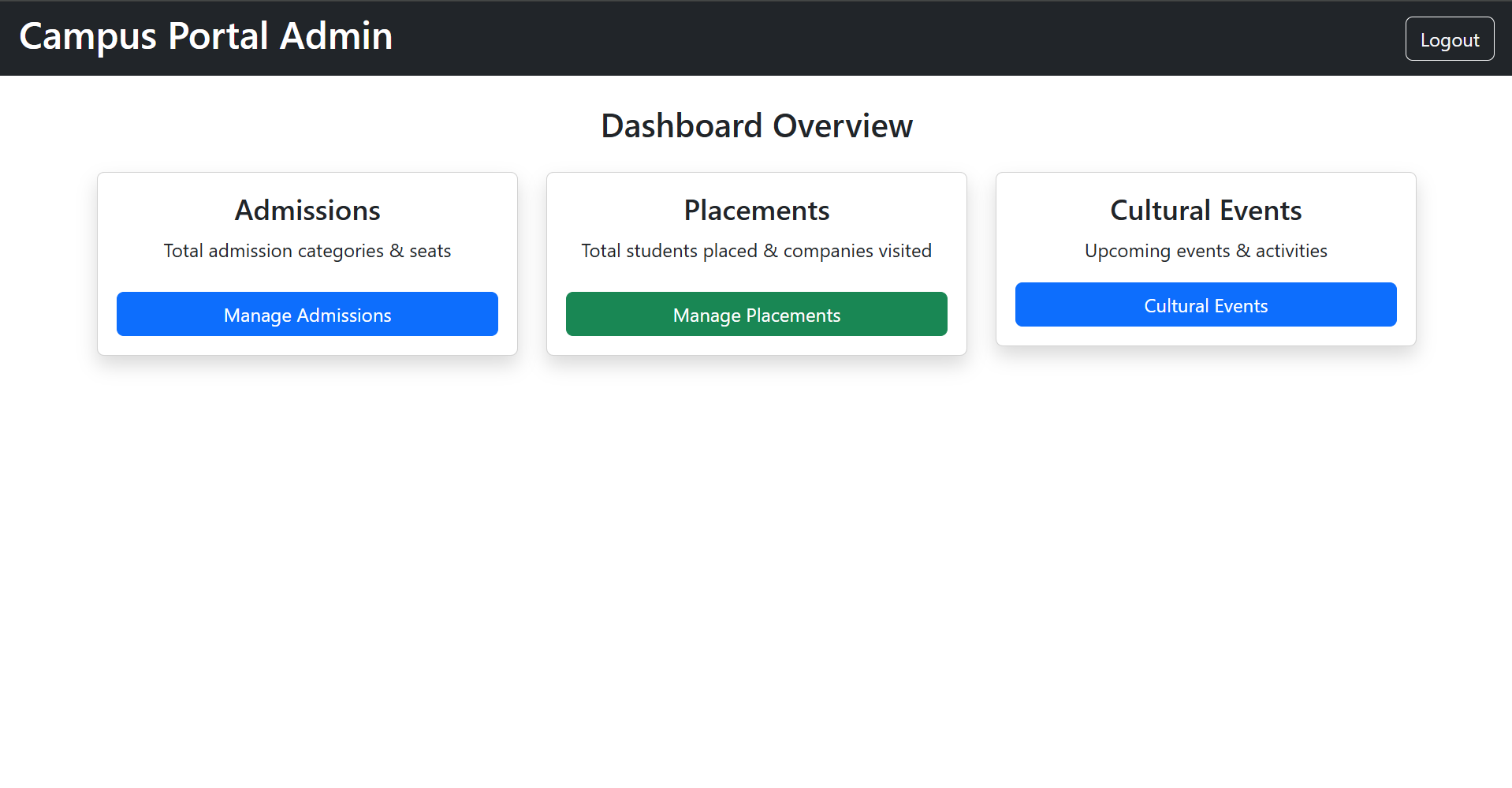
* Delayed access to admission, placement, or event-related information
* Manual effort required to update seat availability, placement statistics, or event schedules
* Difficulty organizing and maintaining large volumes of academic and administrative records
* No centralized or remote access for administrators to view, verify, or update campus data
* Challenges in ensuring accuracy, transparency, and security of institutional information

To overcome these limitations, there is a need for a digital platform where administrators can seamlessly publish, update, and manage admission details, placement records, and event information. Users must be able to access real-time updates from any device. The system should include secure authentication, structured data storage, and an intuitive interface that supports efficient campus management.

**Project Goal:**

The goal of the **Campus Portal Management System** is to digitalize and streamline major academic and administrative processes within an educational institution by providing a unified, centralized platform that:

* Automates the management of admission seat allocation, placement data tracking, and cultural event scheduling.
* Ensures real-time transparency and accessibility for administrators, faculty coordinators, and institutional management.
* Offers a user-friendly, responsive interface that reduces manual workload and supports efficient decision-making through accurate, up-to-date information.

****

**Design System And Analysis:**

5.1 System Architecture Overview

The **Campus Portal Management System** is designed using a modular and service-oriented architectural model built on the **MERN stack (MongoDB, Express.js, React/Angular, Node.js)**. This architecture ensures high scalability, reliability, and maintainability while supporting multiple institutional workflows such as admissions, placements, and cultural event management within a unified platform.

The system architecture is divided into **three primary layers**, each responsible for specific operations within the application.

**1. Client Layer**

The **Client Layer** consists of the frontend interface developed using **Angular or React**, providing a fast, responsive, and component-based user experience. The UI is styled using **Tailwind CSS or Bootstrap**, ensuring consistent page layouts and modern, intuitive interactions. All user requests—including seat updates, placement entries, and event scheduling—originate from this layer and are securely forwarded to the backend API.

**2. Application Layer**

The **Application Layer** contains the core logic of the system and operates through multiple backend services:

**A. Node.js/Express Server (Port 4000/5000)**

This service handles all essential backend operations, including:

* Management of **RESTful APIs** for admissions, placements, and event modules
* Execution of business logic and data validations
* Secure communication with the MongoDB database
* Handling authentication workflows (login, session validation)

By separating concerns, the Node.js server enables modular, clean, and scalable backend development.

**B. Microservices for Analytics (Optional Future Enhancement)**

The architecture supports the addition of independent microservices for analytical features, such as:

* Generating **seat distribution reports**
* Computing **placement statistics and trends**
* Analyzing **event participation and performance**

This optional module ensures the system can evolve into a fully data-driven administrative tool without disrupting current operations

**3. Data Layer**

The **Data Layer** uses **MongoDB Atlas** as the centralized database for securely storing institutional information. The system maintains structured collections for:

* **Admission seat details**
* **Placement records**
* **Cultural event schedules and statuses**
* **Administrator credentials and authentication tokens**

MongoDB provides flexibility through its document-oriented structure and ensures fast read/write operations, even when handling large datasets from multiple modules.

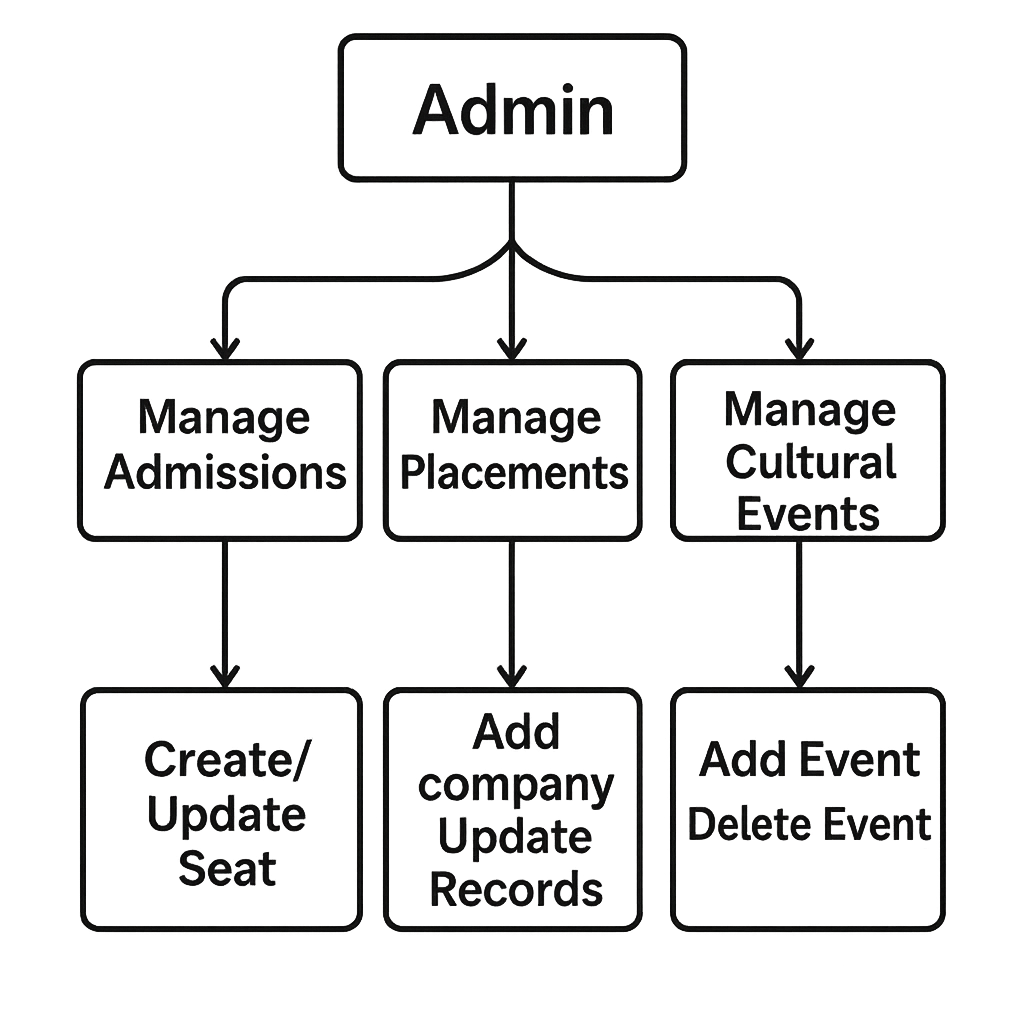
Security is strengthened using **JWT-based role authentication**, which ensures that only authorized users (administrators) can access or modify sensitive institutional data. This approach preserves data integrity and protects confidential information.

**UML Design (Conceptual Mapping):**

**A. Use Case Diagram**

The Use Case Diagram shows what the Admin can do in the Campus Portal System.  
Since only the Admin uses the portal, the diagram includes:

* Manage Admissions – update seats for KCET/COMEDK/Management
* Manage Placements – add companies, update records
* Manage Cultural Events – add or delete events

****

**B. Sequence Diagram**

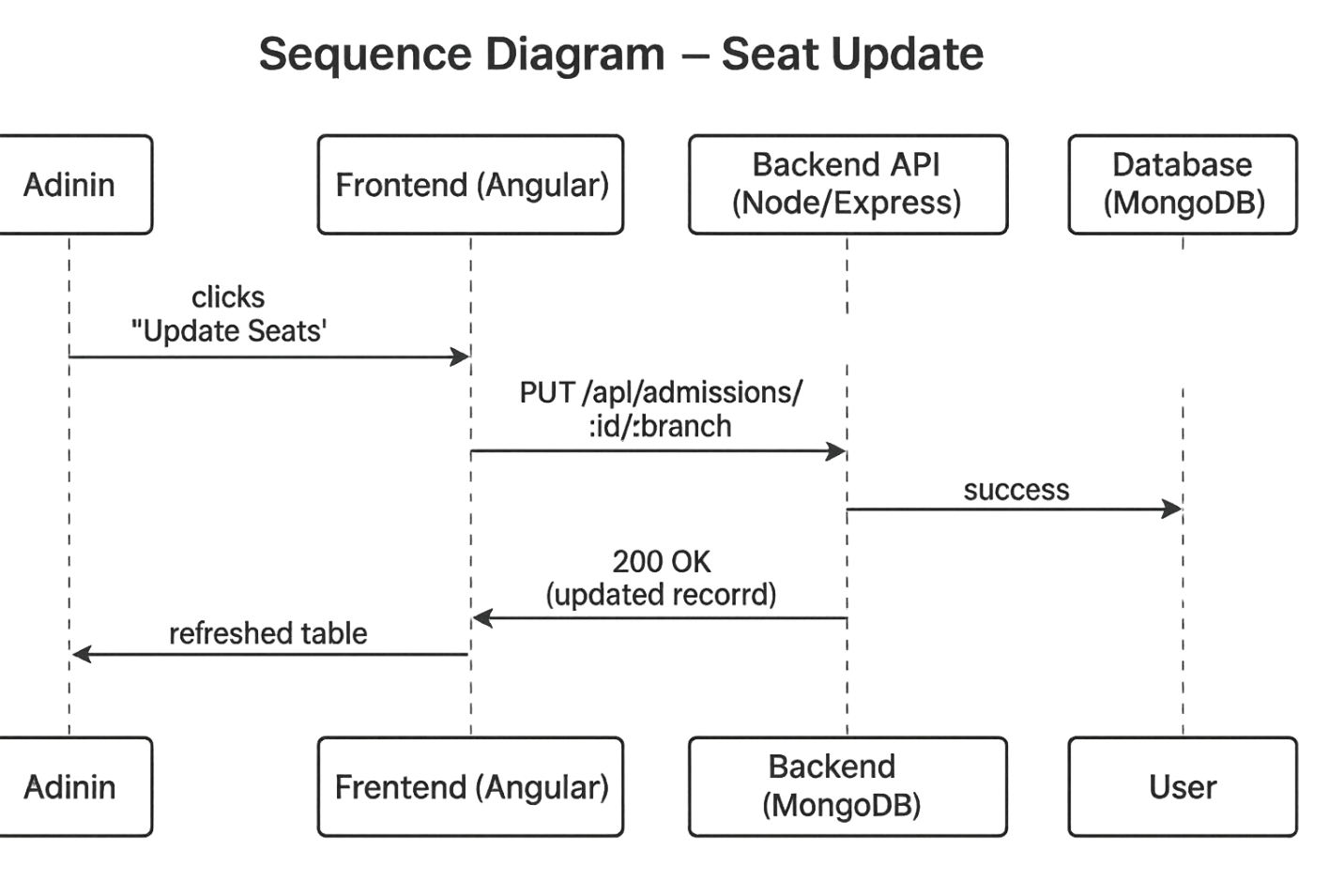
The Sequence Diagram you uploaded explains the **Seat Update Workflow**.

This is how the update process works:

1. **Admin → Frontend (Angular)**  
   Admin clicks the **"Update Seats"** button for a branch/category.
2. **Frontend → Backend API (Node/Express)**  
   A request is sent:  
   PUT /api/admissions/:id/:branch
3. **Backend → Database (MongoDB)**  
   The backend updates the specific seat values in the database.
4. **Database → Backend**  
   MongoDB returns a **success** response.
5. **Backend → Frontend**  
   Backend responds with **200 OK** and updated data.
6. **Frontend → Admin**  
   The table is refreshed showing the updated seat values.

**Purpose of the Sequence Diagram**

* Shows **step-by-step interaction** between system components
* Helps understand the **flow of API calls**
* Useful for debugging and verification during implementation

****



# C. Deployment Diagram

# The deployment architecture of the Campus Portal Management System illustrates how the application’s components are physically organized across different nodes. The system is deployed using a three-tier structure that separates the user interface, application logic, and data storage for improved performance and maintainability.

# The deployment environment consists of the following components:

# Frontend (Angular/React Client – Port 4200) The admin accesses the portal through a web browser, where the Angular/React interface is hosted. This component handles all user interactions, displays admission/placement/event data, and sends requests to the backend API.

# Backend Server (Node.js/Express – Port 4000/5000) The backend server hosts all REST APIs required for managing admissions, placements, and cultural events. It performs authentication, business logic processing, validation, and communicates directly with the database. This server acts as the central processing unit of the system.

# Database Server (MongoDB Atlas / Local MongoDB) MongoDB stores all institutional data, including seat allocations, placement records, event details, and administrator credentials. It ensures high-speed data retrieval and secure, structured storage using collections defined in the system.

# 

# PROPOSED SOLUTION: –

# The proposed solution is to develop a digital Campus Portal Management System using the MERN stack (MongoDB, Express.js, React/Angular, Node.js) that replaces the traditional manual management of admissions, placements, and cultural events.

# The system enables administrators to securely log in and efficiently manage institutional data. Admins can update admission seat allocations, add or modify placement details, schedule cultural events, and remove outdated information—all through a unified digital dashboard. Users (administrators) gain instant access to real-time academic information without relying on physical files or separate systems.

# The backend provides secure REST APIs for authentication, data processing, and module-wise operations, while the frontend delivers a clean and responsive interface for seamless navigation. All admission details, placement records, and event schedules are stored in MongoDB to ensure fast retrieval, structured storage, and high data accuracy. Any update made by the admin is instantly reflected in the system for consistent and real-time information management.

# This solution enhances administrative efficiency, eliminates paperwork, minimizes errors, and provides a centralized platform that allows institutions to manage campus activities effortlessly anytime and from anywhere.

# IMPLEMENTATION: –

# The backend of the Campus Portal Management System was developed using Node.js and Express, where APIs were created to manage admissions, placements, and cultural event data. MongoDB was used as the database to store seat allocation details, placement records, event schedules, and admin login credentials.

# The frontend was built using Angular/React, along with Tailwind CSS/Bootstrap to provide a clean, responsive, and user-friendly interface for managing campus information. Secure administrator login was implemented using JWT authentication, ensuring that only authorized users can access or modify data.

# All updates—such as modifying seat counts, adding placement entries, or scheduling events—are instantly reflected in the system through seamless API communication between the frontend and backend. The portal allows administrators to add, update, or delete data across all modules, while ensuring real-time visibility of the latest campus information.

# admission.html:-

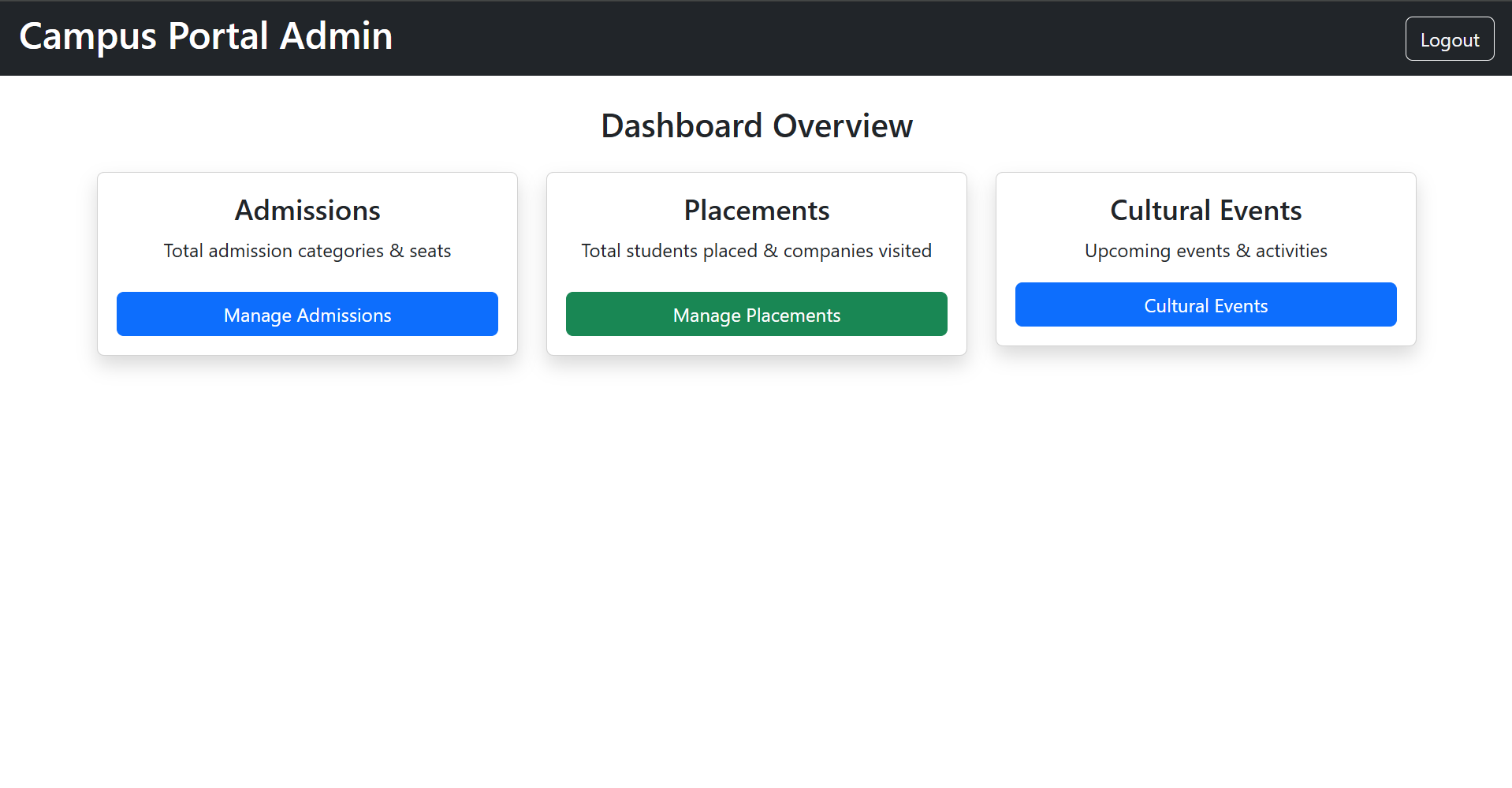
# 

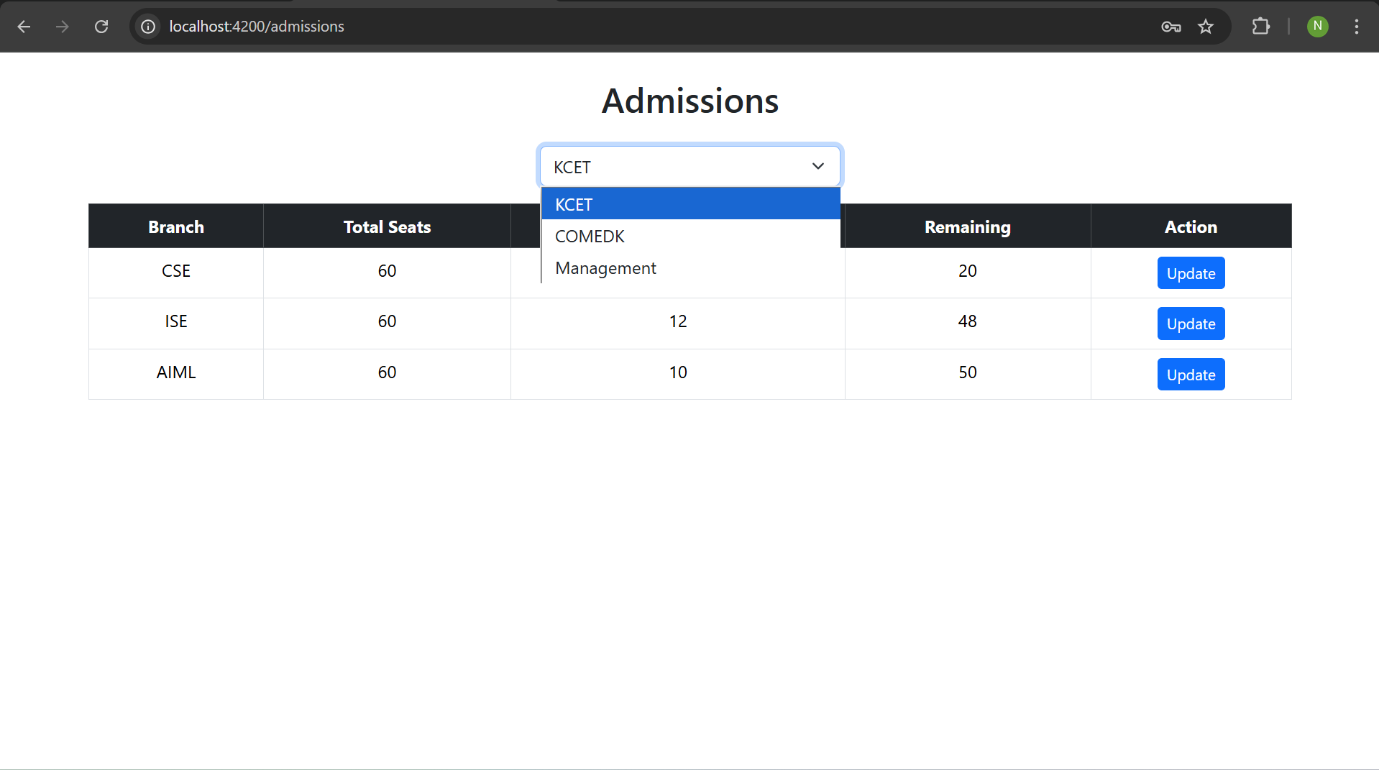
# controller.js:-

# 

# OUTPUT: -



****



# 

# 

# CONCLUSION: -

The Campus Portal Management System marks a major advancement in institutional administration by replacing traditional, manual processes with a secure, efficient, and digitally integrated platform. By leveraging the MERN stack, the system effectively fulfills its primary objectives:

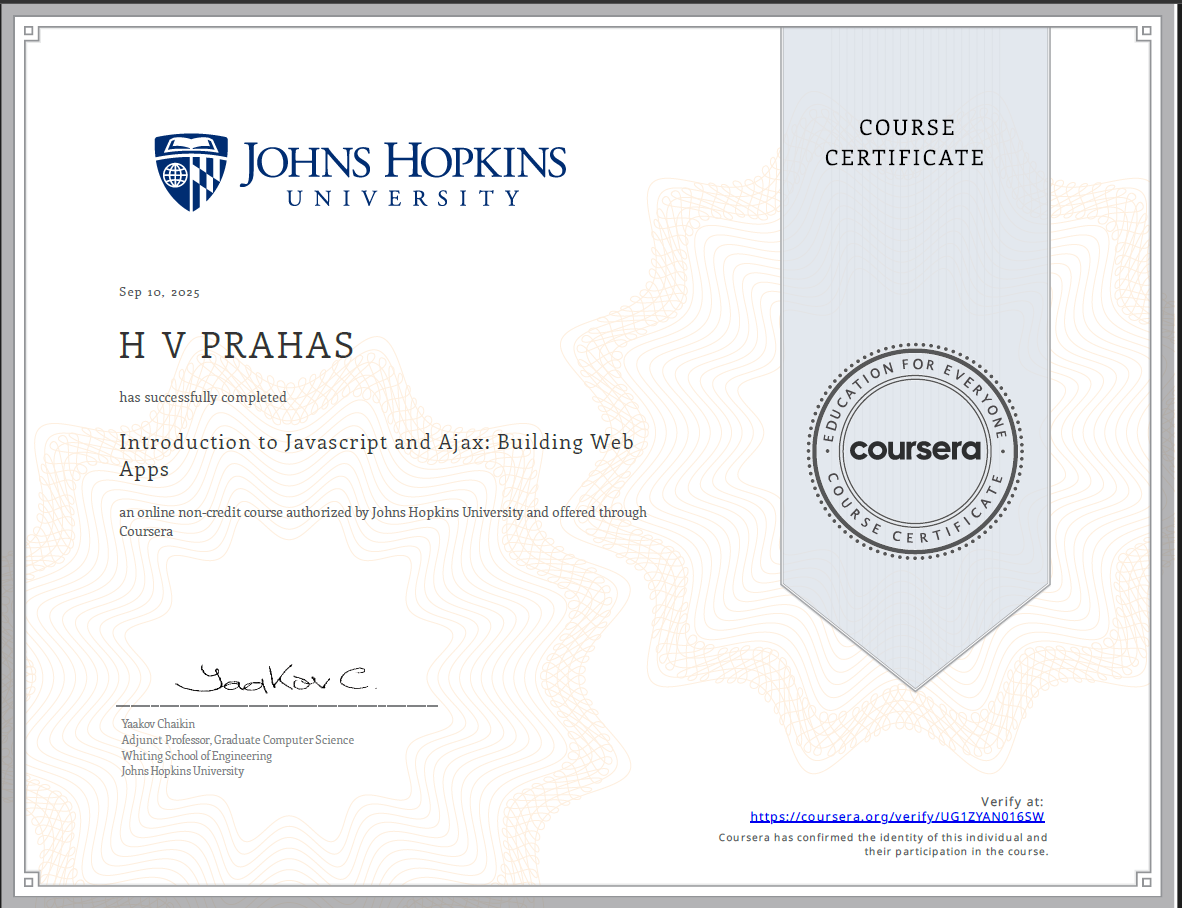
* **Efficiency and Streamlined Management:**  
  It removes the dependency on paper-based records and fragmented tools, enabling administrators to manage admissions, placements, and cultural events through a centralized, automated workflow.
* **Real-Time Accessibility and Transparency:**  
  The platform ensures that academic information such as seat allocations, placement statistics, and event schedules is available instantly and accurately, improving decision-making and fostering smooth coordination within the institution.
* **Technical Reliability and Scalability:**  
  The modular, modern architecture ensures the system is easy to maintain, secure, and capable of supporting future enhancements, including analytics dashboards, additional role-based logins, and extended academic modules.

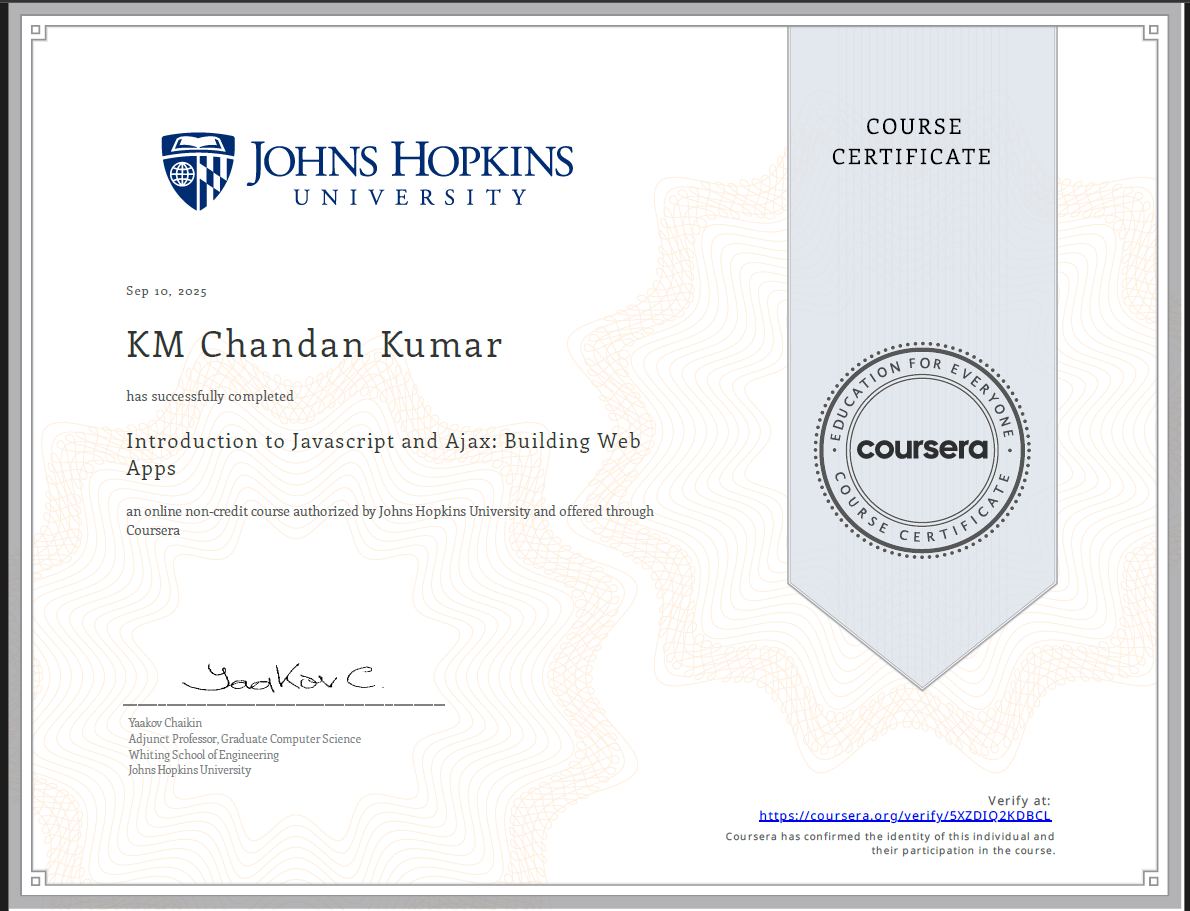
Overall, the Campus Portal Management System demonstrates how technology can transform campus operations, promoting transparency, reducing manual workload, and providing a unified digital environment for efficient institutional managemen

**REFERENCES: –**

1. MongoDB Documentation – <https://www.mongodb.com/>
2. Express.js Documentation – <https://expressjs.com/>
3. Angular / React Documentation – <https://angular.io/> / <https://react.dev/>
4. Node.js Documentation – <https://nodejs.org/>
5. Research Articles on Campus Information Systems and Digital Academic Management Platforms
6. Official tutorials and guides for MERN Stack Development

# CERTIFICATES: -

****





GITHUB LINK: -

[https://github.com/NitinSastry /Campus-portal.git](https://github.com/NitinSastry%20/Campus-portal.git-)