## 1. ABSTRACT

**COLLEGE WEBSITE**

* 1. **Brief Overview of Project:**

A project on a college website involves redesigning or updating the existing website to improve functionality, user experience, and visual appeal. This includes research, planning, design, development, content management, UX improvements, responsive design, SEO optimization, security implementation, testing, feedback gathering, and ongoing maintenance. The goal is to create a user-friendly, visually appealing website that meets the needs of students, faculty, staff, and the community.

**1.2 Existing System:**

College websites prioritize user experience by providing easy navigation, clear organization of information, and responsive design. They typically feature a user-friendly layout with intuitive menus and search functionality to help users find information quickly. The websites often include visually appealing design elements, such as high-quality images and videos, to engage users and create a positive impression of the college. They also prioritize accessibility, ensuring that the website is usable for people with disabilities. Interactive features, such as event calendars, news feeds, and social media integration, keep users informed about campus events and news. Many college websites also offer personalized content for students, faculty, and staff, such as portals for accessing grades, course materials, and administrative services.Overall, college websites aim to provide a seamless and engaging user experience that reflects the institution's values and priorities.

**1.3 Proposed System:**

A proposed system for a modern college website would incorporate several key features to enhance user experience and functionality. The frontend would utilize a responsive design framework like Bootstrap or Tailwind CSS to ensure the website is optimized for various devices. Interactive elements and dynamic content would be implemented using JavaScript frameworks like React or Vue.js. For the backend, a server-side language such as Node.js or Python could be used along with a database like MongoDB or MySQL for efficient data storage and retrieval. Content management would be facilitated by a CMS like WordPress or Drupal, allowing for easy management of pages, posts, and multimedia content. Security measures would include HTTPS, secure authentication, and data encryption to protect user information. The website would be optimized for SEO with proper meta tags, structured data, and content optimization. Accessibility features would also be implemented to ensure the website is usable for people with disabilities .Overall, the proposed system would aim to provide a modern, user-friendly experience for visitors to the college website, with a focus on responsiveness, interactivity, and accessibility.

## 2. ANALYSIS

### **2.1 INTRODUCTION**

A college website serves as the digital gateway to an educational institution, offering a window into its academic programs, campus life, and resources. It is a central hub for students, faculty, staff, and the community to access vital information, ranging from admissions and course offerings to campus news and events. College websites often reflect the institution's identity and values, showcasing its achievements, facilities, and vibrant campus culture. With an emphasis on user experience and accessibility, these websites strive to provide a seamless and informative experience for visitors, highlighting the unique aspects of the college and fostering engagement among its stakeholders.

**Top of Form**

### **2.2 STUDENT BIO-DATA**

### Student bio data typically includes essential information about a student, such as their full name, date of birth, gender, contact information (address, phone number, email), educational background (schools attended, academic achievements), extracurricular activities, and any other relevant information. This data is used for various purposes, such as admissions, scholarships, and student records management. It helps institutions maintain accurate records and provide personalized services to students.

### Top of Form

### **2.3 STUDENT ATTENDANCE**

Student attendance refers to the record of a student's presence or absence from classes or other educational activities. It is an important aspect of academic monitoring and performance assessment. Attendance data is typically collected by teachers or school staff and is used to track students' participation, identify trends in attendance patterns, and intervene if there are concerns about a student's attendance. Attendance records may also be used for administrative purposes, such as determining eligibility for exams or maintaining compliance with attendance policies. Modern systems often use digital tools like attendance management software to streamline the process and improve accuracy.

**Top of Form**

**2.3 FACULTY BIO-DATA**

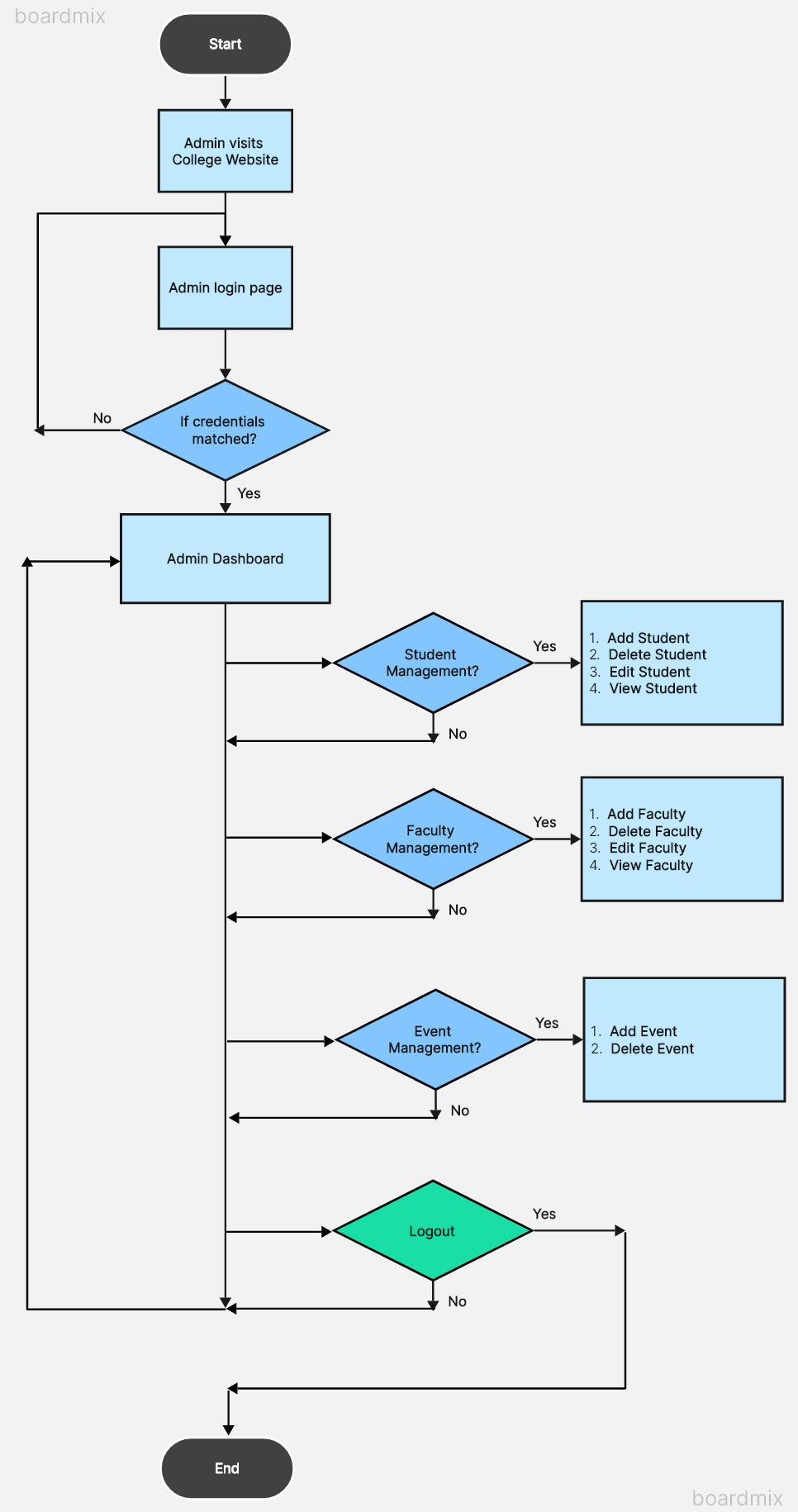
Faculty bio data typically includes essential information about a faculty member, such as their full name, educational qualifications, professional experience, areas of expertise, research interests, publications, and contact information. This data is used by educational institutions to showcase the qualifications and expertise of their faculty members to students, parents, and the wider community. It helps establish credibility and trust in the institution's academic programs and faculty. Additionally, faculty bio data may be used for internal purposes, such as faculty recruitment, performance evaluation, and academic planning.

Top of Form

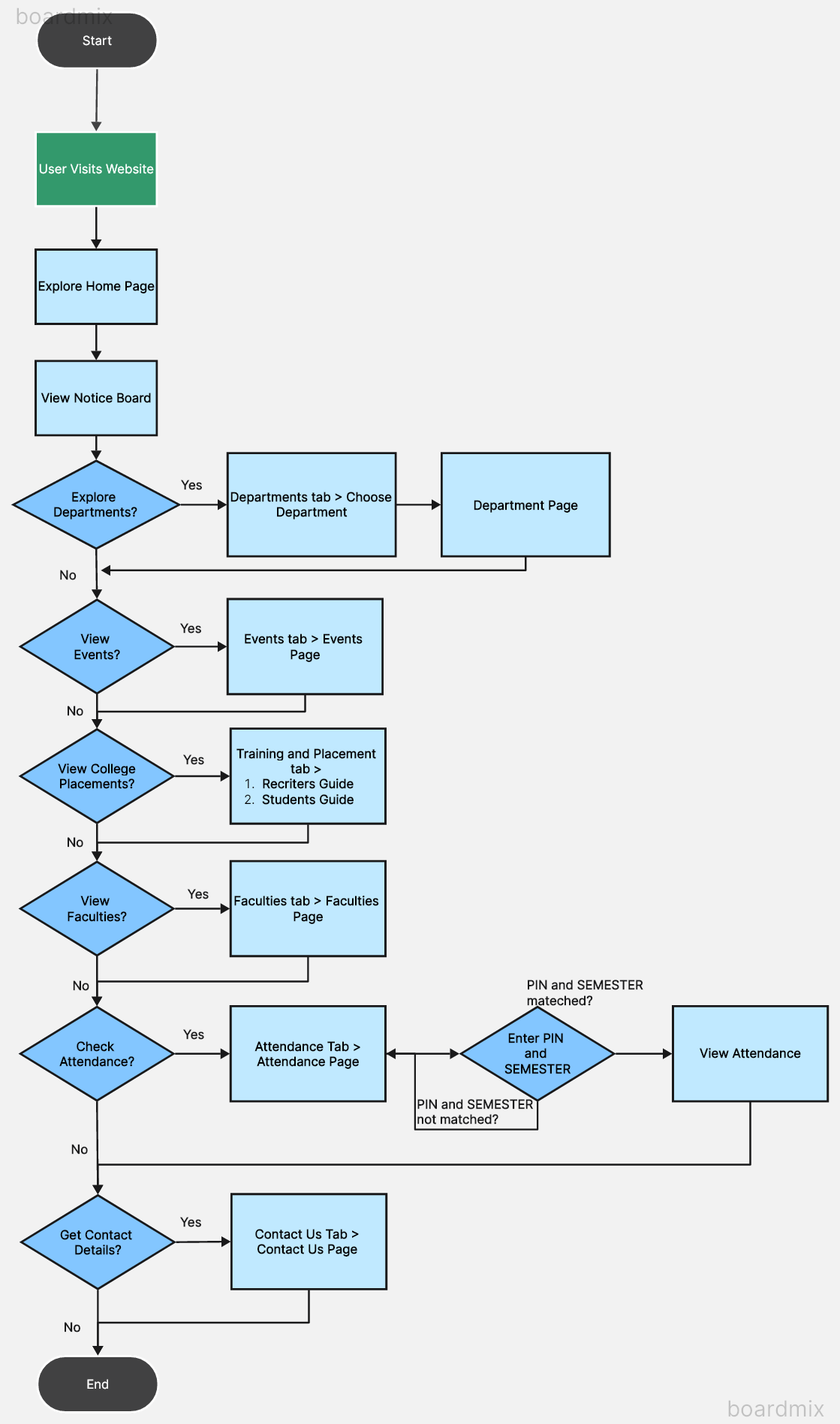
**2.4 EVENTS**

Colleges often host a variety of events, including academic lectures, workshops, cultural festivals, sports tournaments, and social gatherings. These events enrich the student experience, promote learning outside the classroom, and foster a sense of community among students, faculty, and staff. They also serve as opportunities for networking, skill development, and cultural exchange.

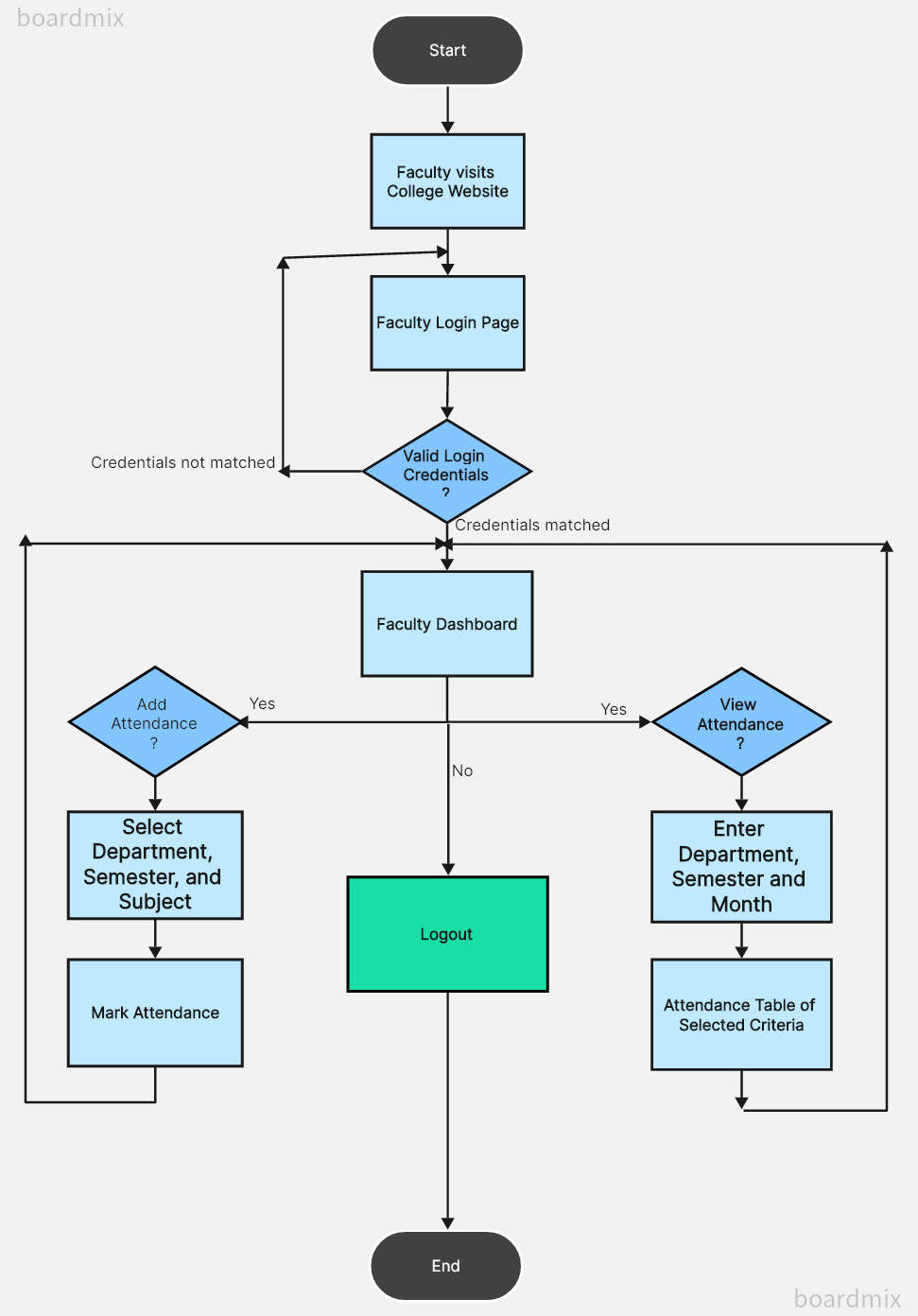
## 3. SYSTEM DESIGN

****

3.1.1 System design diagram for Admin

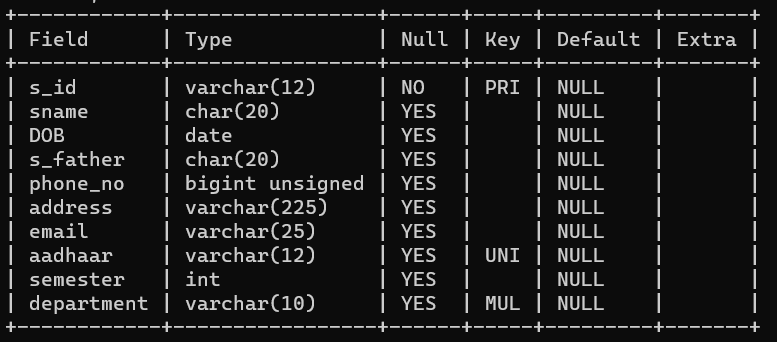
****

3.1.2 System design diagram for User

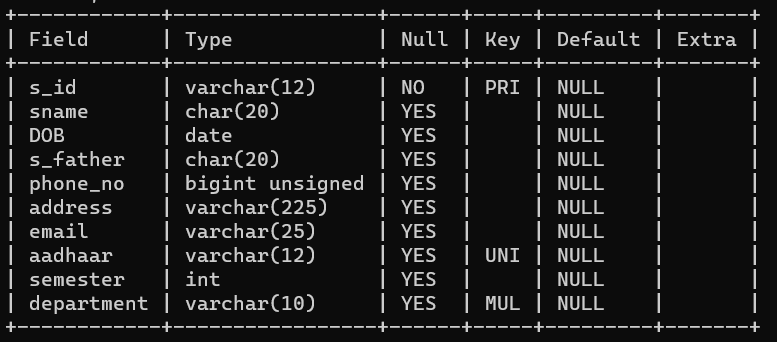
****

3.1.3 System design diagram for Faculty

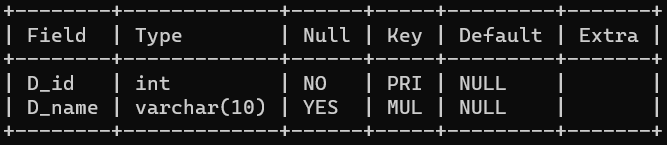
**3.2 Database Table Diagram**

****

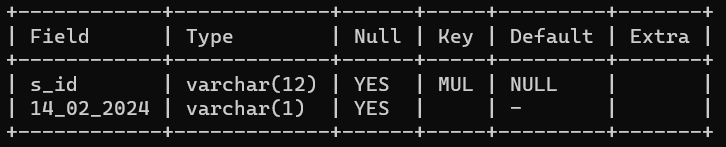
3.2.1 Schema for Student Management



3.2.2 Schema to store the details of Passed Out students

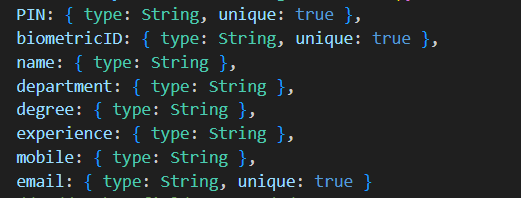


3.2.3 Schema to store Department Details



3.2.4 Schema to store Attendance Status of Students

14\_02\_2024 Field in above schema stores the status of that particular date.



3.2.5 Schema for Faculty Management

## 4. TECHNOLOGIES USED

## **4.1 VS Code**

Visual Studio Code (VS Code) is a popular, open-source code editor developed by Microsoft. It has a wide range of features, including syntax highlighting, debugging, and integrated source control. In 2022, there may have been several updates and new features added to VS Code, including new extensions, improved performance, and bug fixes. Additionally, the VS Code team may continue to improve its support for various programming languages and development environments. Overall, VS Code is a widely-used and respected code editor among developers for its flexibility and ease of use.

**4.2 Json:**

JSON (JavaScript Object Notation) is a lightweight data-interchange format that is easy for humans to read and write, and easy for machines to parse and generate. It is often used for transmitting data between a server and a web application, as well as for storing and exchanging data in a structured format.

Here are some key features of JSON:

1. **Simple Syntax**: JSON has a simple and straightforward syntax consisting of key-value pairs, arrays, and nested objects. It closely resembles the syntax used for defining objects in JavaScript.
2. **Data Types**: JSON supports several data types, including strings, numbers, booleans, arrays, objects, and null values. This flexibility allows you to represent a wide range of data structures and formats.
3. **Human Readable**: JSON is designed to be easy for humans to read and write. Its concise and welldefined syntax makes it an ideal choice for configuration files, APIs, and other scenarios where readability is important.
4. **Language Agnostic**: JSON is language agnostic, meaning it can be used with any programming language that has support for parsing and generating JSON data. This makes it a popular choice for interoperability between different systems and platforms.
5. **Interchangeable**: JSON is widely supported across various programming languages, frameworks, and tools. It is commonly used for exchanging data between web servers and clients, as well as for storing data in databases and files.
6. **Lightweight**: JSON is a lightweight format with minimal overhead, making it efficient for transmitting data over the network and storing large volumes of data.

## **4.3 CSS**

Cascading Style Sheets, fondly referred to as CSS, is a simply designed language intended to simplify the process of making web pages presentable. CSS allows you to apply styles to web pages. More importantly, CSS enables you to do this independent of the HTML that makes up each web page. It describes how a webpage should look: it prescribes colors, fonts, spacing, and much more. In short, you can make your website look however you want. CSS lets developers and designers define how it behaves, including how elements are positioned in the browser.

While html uses tags, CSS uses rulesets. CSS is easy to learn and understand, but it provides powerful control over the presentation of an HTML document.

### **4.4 HTML**

HTML (Hypertext Markup Language) is the standard markup language used to create and design web pages. It consists of a set of elements or tags that structure the content of a webpage. Here are some key features of

HTML:

1. **Structure**: HTML provides a structured way to organize content on a webpage using elements such as headings, paragraphs, lists, tables, and more. These elements define the hierarchy and layout of the page.
2. **Semantics**: HTML offers semantic elements that convey the meaning and purpose of the content. For example, <header>, <footer>, <nav>, <article>, and <section> provide context to the content, making it more accessible to both humans and search engines.
3. **Hyperlinks**: HTML allows you to create hyperlinks using the <a> element, enabling users to navigate between different web pages and resources. Hyperlinks can link to other web pages, files, email addresses, or specific sections within a page (using anchor links).
4. **Images and Multimedia**: HTML supports the inclusion of images, videos, and audio files using the

<img>, <video>, and <audio> elements, respectively. This allows you to enhance the visual and interactive aspects of your web pages.

1. **Forms**: HTML provides form elements (<form>, <input>, <textarea>, <select>, <button>) for capturing user input and interacting with visitors. Forms are used for various purposes, such as collecting user feedback, processing user-generated content, and enabling e-commerce transactions.
2. **Accessibility**: HTML includes features for creating accessible web content, such as alternative text for images (alt attribute), semantic elements for better screen reader support, and form elements with builtin accessibility features.
3. **Cross-Browser Compatibility**: HTML is supported by all modern web browsers, ensuring consistent rendering and functionality across different platforms and devices.
4. **Extensibility**: HTML can be extended and enhanced using other web technologies such as CSS (Cascading Style Sheets) for styling and layout, JavaScript for interactivity and dynamic behavior, and various web APIs for accessing device features and external services.
5. **Separation of Concerns**: HTML promotes a separation of concerns by separating the structure and content of a webpage from its presentation (handled by CSS) and behavior (handled by JavaScript). This makes it easier to maintain and update web pages over time

### **4.5 MYSQL Database**

MySQL is a widely used open-source relational database management system (RDBMS) that provides a powerful and scalable platform for storing, managing, and retrieving data. Here are some key features of MySQL:

1. **Relational Database**: MySQL is a relational database system, which means it organizes data into tables consisting of rows and columns, with relationships defined between them. This relational model allows for efficient querying, indexing, and manipulation of data.
2. **SQL Support**: MySQL supports the Structured Query Language (SQL), which is a standard language for managing relational databases. SQL allows users to perform various operations such as querying data, inserting, updating, deleting records, and defining database structures (tables, indexes, constraints).
3. **High Performance**: MySQL is designed for high performance and scalability, making it suitable for handling large datasets and high traffic websites. It includes features such as indexing, query optimization, caching, and support for multi-threading to maximize performance.
4. **Cross-Platform Compatibility**: MySQL is available for various operating systems, including Linux, Windows, macOS, and Unix-like systems, making it a versatile choice for different environments and platforms.
5. **Client/Server Architecture**: MySQL follows a client/server architecture, where multiple clients can connect to a MySQL server to access and manipulate data stored in databases. This architecture allows for concurrent access and efficient resource utilization.
6. **Data Security**: MySQL provides robust security features to protect sensitive data, including user authentication, access control, encryption, and auditing. It supports SSL/TLS encryption for secure communication between clients and the server.
7. **Scalability and Replication**: MySQL supports horizontal scalability through features such as replication, clustering, and sharding. Replication allows you to create multiple copies of a database for load balancing, fault tolerance, and data redundancy.
8. **High Availability**: MySQL offers features such as master-slave replication, automatic failover, and clustering to ensure high availability and fault tolerance. These features help minimize downtime and ensure continuous access to data even in the event of hardware failures or network issues.
9. **Community and Ecosystem**: MySQL has a large and active community of developers, users, and contributors, providing extensive documentation, tutorials, forums, and third-party tools/plugins. This vibrant ecosystem enhances the usability, reliability, and extensibility of MySQL.
10. **Open Source**: MySQL is an open-source software released under the GNU General Public License (GPL), which means it is free to use, modify, and distribute. Its open-source nature fosters innovation, collaboration, and widespread adoption in various industries and applications.

### **4.6 Node js**

Node.js is an open-source, cross-platform JavaScript runtime environment that executes JavaScript code outside of a web browser. It allows you to build server-side applications with JavaScript. Node.js uses an event-driven, non-blocking I/O model that makes it lightweight and efficient, perfect for data-intensive real-time applications that run across distributed devices.

Key features of Node.js include:

1. **Asynchronous and Event-Driven**: Node.js uses asynchronous programming, which means that multiple operations can be executed concurrently without waiting for the completion of one another. This is particularly useful for I/O-bound tasks, such as reading files or making network requests.
2. **Fast Execution**: Node.js is built on the V8 JavaScript engine, which is developed by Google for the Chrome browser. V8 compiles JavaScript into native machine code, making Node.js applications fast and efficient.
3. **Single-threaded, Non-blocking**: Node.js uses a single-threaded event loop model for handling requests. This means that Node.js can handle a large number of simultaneous connections without the need for threads, which can be resource-intensive.
4. **NPM (Node Package Manager)**: Node.js comes with npm, a package manager that allows developers to easily install and manage third-party packages and libraries. npm hosts the world's largest collection of open-source packages.
5. **Cross-platform**: Node.js is compatible with Windows, macOS, and Linux, making it a versatile choice for developers working on different operating systems.
6. **Scalable**: Node.js applications can be easily scaled horizontally by adding more nodes to distribute the load. Additionally, Node.js supports clustering, which allows you to take advantage of multi-core systems.
7. **Rich Ecosystem**: Node.js has a rich ecosystem of libraries and frameworks that make it easy to build a variety of applications, including web servers, APIs, real-time chat applications, and more.

Overall, Node.js is a powerful tool for building fast, scalable, and efficient server-side applications using JavaScript.

**Top of Form**

### **4.7 Express js**

Express.js, commonly known as Express, is a minimal and flexible Node.js web application framework that provides a robust set of features for web and mobile applications. It is designed for building web applications and APIs and is the de facto standard server framework for Node.js.

Key features of Express.js include:

1. **Routing**: Express provides a simple and expressive way to define routes for handling different HTTP requests (GET, POST, PUT, DELETE, etc.) to specific URL paths.
2. **Middleware**: Express uses middleware functions to handle requests. Middleware functions have access to the request and response objects, and they can perform tasks such as logging, authentication, parsing request bodies, etc.
3. **Template Engines**: Express supports various template engines like Pug, EJS, and Handlebars, which allow you to dynamically generate HTML pages based on data from your application.
4. **Static File Serving**: Express can serve static files, such as images, CSS, and JavaScript files, from a directory on the server, making it easy to build web applications with client-side assets.
5. **Error Handling**: Express provides built-in error handling middleware that you can use to handle errors that occur during the request-response cycle.
6. **Routing Parameters**: Express allows you to define route parameters in your routes, which can be used to capture values from the URL and pass them to your route handler functions.
7. **RESTful Routing**: Express makes it easy to build RESTful APIs by providing features like route patterns, HTTP methods, and middleware to handle requests.
8. **Integration with Other Middleware**: Express can be easily integrated with other middleware and frameworks to add additional functionality to your applications, such as authentication, session management, and database integration.

### **4.8 Bootstrap**

Bootstrap is a free and open-source front-end framework for developing responsive and mobile-first websites and web applications. It was originally created by Twitter developers and is now maintained by the open-source community.

Key features of Bootstrap include:

1. **Responsive Grid System**: Bootstrap provides a responsive grid system based on a 12-column layout, making it easy to create responsive designs that adapt to different screen sizes and devices.
2. **CSS Components**: Bootstrap includes a wide range of CSS components, such as buttons, forms, navigation bars, and typography styles, that you can use to quickly style your website or application.
3. **JavaScript Plugins**: Bootstrap comes with several JavaScript plugins, such as modals, carousels, tooltips, and dropdowns, that enhance the functionality of your website and improve user experience.
4. **Customizable**: Bootstrap is highly customizable, allowing you to modify its variables, mixins, and components to match your design requirements.
5. **Documentation and Community**: Bootstrap has extensive documentation and a large community of developers, which makes it easy to get started and find solutions to common problems.
6. **Browser Compatibility**: Bootstrap is compatible with all modern browsers, including Chrome, Firefox, Safari, and Edge, ensuring consistent appearance and functionality across different browsers.
7. **Integration with Other Tools**: Bootstrap can be easily integrated with other front-end tools and frameworks, such as jQuery and AngularJS, to enhance its functionality and streamline development.

### **4.9 MongoDB**

MongoDB is a popular open-source NoSQL database management system that uses a document-oriented data model. Instead of using tables and rows as in traditional relational databases, MongoDB uses collections and documents.

Key features of MongoDB include:

1. **Document-Oriented**: MongoDB stores data in flexible, JSON-like documents, which makes it easy to store and query data without needing to define a schema upfront.
2. **Scalability**: MongoDB is designed to scale horizontally across multiple servers, making it suitable for handling large volumes of data and high-traffic applications.
3. **High Performance**: MongoDB uses a variety of techniques to provide high performance, including indexing, sharding, and in-memory storage.
4. **Rich Query Language**: MongoDB supports a rich query language that includes support for queries, projections, sorting, and aggregations.
5. **Flexible Schema**: MongoDB's flexible schema allows you to store data of any structure, making it easy to evolve your data model over time.
6. **Ad Hoc Queries**: MongoDB supports ad hoc queries, which means you can query your data without needing to define the query in advance.
7. **Replication and High Availability**: MongoDB supports replica sets, which provide high availability and data redundancy by maintaining multiple copies of data across different servers.
8. **Geospatial Indexing**: MongoDB supports geospatial indexing and queries, making it suitable for location-based applications.
9. **Aggregation Framework**: MongoDB includes a powerful aggregation framework that allows you to perform complex aggregations on your data.
10. **Map-Reduce**: MongoDB supports map-reduce operations for performing large-scale data processing tasks.

### **4.10 React js**

React.js, commonly referred to as React, is an open-source JavaScript library developed by Facebook for building user interfaces (UIs) and single-page applications. React allows developers to create large web applications that can change data without reloading the page. It is often used as the V (View) in the MVC (Model-View-Controller) architecture.

Key features of React include:

1. **Component-Based**: React uses a component-based architecture, where UIs are divided into reusable components. This makes it easier to manage and update complex UIs.
2. **Virtual DOM**: React uses a virtual DOM (Document Object Model), which is a lightweight copy of the actual DOM. React's virtual DOM allows it to efficiently update and render UI components, resulting in better performance.
3. **JSX**: JSX is a syntax extension for JavaScript that allows developers to write HTML-like code within their JavaScript files. JSX makes it easier to create and maintain React components.
4. **One-Way Data Binding**: React implements a unidirectional data flow, where data flows only in one direction, from parent components to child components. This helps to maintain a predictable state of the application.
5. **React Hooks**: React Hooks are functions that allow developers to use state and other React features in functional components. Hooks provide a way to add state and lifecycle methods to functional components, which were previously only available in class components.
6. **Declarative Programming**: React uses a declarative programming paradigm, where developers describe the desired state of the UI, and React takes care of updating the UI to match that state.
7. **Virtual DOM Diffing**: React uses a process called "diffing" to determine the minimum number of DOM manipulations needed to update the UI. This makes React applications more efficient and performant.
8. **React Router**: React Router is a routing library for React applications that allows developers to create single-page applications with dynamic, client-side routing.
9. **React Native**: React Native is a framework for building native mobile applications using React. It allows developers to use the same React components to build both web and mobile applications.

5. CODING

**STUDENT BIO-DATA MANAGEMENT**

**Addstudent.jsx**

function AddStudent() {

const [file, setFile] = useState(null);

const [successMessage, setSuccessMessage] = useState(null);

const onDrop = (e) => {

setFile(e.target.files[0]);

};

const uploadFile = async () => {

if (!file) {

console.error('No file selected.');

return;

}

const formData = new FormData();

formData.append('file', file);

try {

await axios.post('http://localhost:3000/upload', formData, {

headers: { 'Content-Type': 'multipart/form-data' },

});

console.log('File uploaded successfully!');

alert("Student Details Added Successfully");

setSuccessMessage('File uploaded successfully!');

} catch (error) {

console.error('Error uploading file:', error);

alert("Data format violated or Duplicate data entered!!!");

setSuccessMessage(null);

}

};

const downloadExcelTemplate = () => {

const templateUrl = process.env.PUBLIC\_URL + '/details.xlsx';

const link = document.createElement('a');

link.href = templateUrl;

link.download = 'Book1.xlsx';

link.click();

};

const processSixthSemester = async () => {

try {

await axios.post('http://localhost:3000/processSixthSemester');

console.log('Sixth-semester students processed successfully!');

setSuccessMessage('Sixth-semester students processed successfully!');

alert("Student Proccessed Successfully")

} catch (error) {

console.error('Error processing sixth-semester students:', error);

setSuccessMessage(null);

alert("Error in proccessing students")

}

};

return (

<>

{successMessage && (

<div style={{ backgroundColor: 'green', color: 'white', textAlign: 'center', padding: '10px', marginBottom: '10px' }}>

{successMessage}

</div>

)}

<div style={{ backgroundColor: 'white', padding: '20px', marginTop: '20px', height: 'auto', display: 'flex', flexDirection: 'column', justifyContent: 'center', alignItems: 'center' }}>

<h2 style={{ padding: '20px', backgroundColor: 'white' }}>Download The Template</h2>

<button type="button" onClick={downloadExcelTemplate} className='btn btn-primary p-10'>

Download Excel Template

</button>

</div>

<div className="container mt-5">

<div className="card p-4">

<div className="custom-file">

<input type="file" className="custom-file-input" id="customFile" onChange={onDrop} />

<label className="custom-file-label" htmlFor="customFile">

{file ? file.name : 'Choose file'}

</label>

</div>

<button className="btn btn-success mt-3" onClick={uploadFile}>

Upload

</button>

<button className="btn btn-primary mt-3 ml-3" onClick={processSixthSemester}>

Process Sixth Semester

</button>

</div>

</div>

</>

);

}

export default AddStudent;

**ViewStudent.jsx**

import axios from 'axios';

import React, { useEffect, useState } from 'react';

import { Modal, Button } from 'react-bootstrap';

import EditStudent from './EditStudent';

import \* as XLSX from 'xlsx'; // Import xlsx library

const ViewStudent = () => {

const [selectedSemester, setSelectedSemester] = useState('');

const [selectedDepartment, setSelectedDepartment] = useState('');

const [students, setStudents] = useState([]);

const [error, setError] = useState(null);

const [deleteModal, setDeleteModal] = useState({ show: false, studentId: null });

const [editModal, setEditModal] = useState({ show: false, studentId: null, studentDetails: null });

const departments = ['AU', 'CIV', 'cse', 'EEE', 'ECE', 'MEC'];

const handleSemesterChange = (event) => {

const selectedSemester = event.target.value;

setSelectedSemester(selectedSemester);

};

const handleEdit = (studentId, studentDetails) => {

setEditModal({ show: true, studentId, studentDetails });

};

const handleEditModalHide = () => {

setEditModal({ show: false, studentId: null, studentDetails: null });

};

const handleEditSuccess = () => {

handleEditModalHide();

fetchStudentDetails();

};

const handleDepartmentChange = (event) => {

const selectedDepartment = event.target.value;

setSelectedDepartment(selectedDepartment);

};

const fetchStudentDetails = async () => {

try {

const response = await axios.get('http://localhost:3000/getstudents', {

params: {

semester: selectedSemester,

department: selectedDepartment,

},

});

if (response.status === 200) {

setStudents(response.data);

setError(null);

} else {

console.error('Error in fetching details', response.statusText);

setError('Error fetching student details');

}

} catch (error) {

console.error('Error in fetching details', error);

setError('Error fetching student details');

}

};

const handleDelete = async (studentId) => {

try {

const response = await axios.delete(`http://localhost:3000/deletestudent/${studentId}`);

if (response.status === 200) {

setDeleteModal({ show: false, studentId: null });

fetchStudentDetails();

} else {

console.error('Error in deleting student', response.statusText);

}

} catch (error) {

console.error('Error in deleting student', error);

}

};

const handleDownloadExcel = () => {

alert('Downloading')

const worksheet = XLSX.utils.json\_to\_sheet(students);

const workbook = XLSX.utils.book\_new();

XLSX.utils.book\_append\_sheet(workbook, worksheet, 'Students');

XLSX.writeFile(workbook, 'students.xlsx');

alert('Students data downloaded successfully!');

};

useEffect(() => {

if (selectedSemester && selectedDepartment) {

fetchStudentDetails();

}

}, [selectedSemester, selectedDepartment]);

return (

<>

<div className='student-attendance-container'>

<div className='row'>

<div className='col-md-5'>

<label htmlFor='department' className='input-label'>

Select Department:

</label>

<select

className='form-control input-field'

id='department'

value={selectedDepartment}

onChange={handleDepartmentChange}

>

<option value=''>Select</option>

{departments.map((department) => (

<option key={department} value={department}>

{department}

</option>

))}

</select>

</div>

<div className='col-md-5'>

<label htmlFor='semester' className='input-label'>

Enter Batch:

</label>

<input

type='text'

id='semester'

value={selectedSemester}

onChange={handleSemesterChange}

placeholder='example:21'

/>

</div>

<div className='col-md-2'>

<button

type='button'

className='btn btn-primary mt-4'

onClick={fetchStudentDetails}

>

Get Student Details

</button>

</div>

</div>

<div className='mt-4'>

<h3 className='table-header'>Student Details:</h3>

{error ? (

<p className='text-danger'>{error}</p>

) : Array.isArray(students) && students.length > 0 ? (

<div className='table-responsive'>

<table className='faculty-table'>

<thead>

<tr>

<th>ID</th>

<th>Name</th>

<th>Date of Birth</th>

<th>Father's Name</th>

<th>Phone Number</th>

<th>Address</th>

<th>Semester</th>

<th>Department ID</th>

<th>Aadhar</th>

<th>Email</th>

<th colSpan={2}>Action</th>

</tr>

</thead>

<tbody>

{students.map((student) => (

<tr key={student.s\_id}>

<td>{student.s\_id}</td>

<td>{student.sname}</td>

<td>{student.DOB}</td>

<td>{student.s\_father}</td>

<td>{student.phone\_no}</td>

<td>{student.address}</td>

<td>{student.semester}</td>

<td>{student.department}</td>

<td>{student.aadhaar}</td>

<td>{student.email}</td>

<td>

<button

className='dark'

onClick={() => setDeleteModal({ show: true, studentId: student.s\_id })}

>

Delete

</button>

</td>

<td>

<button

className='dark'

onClick={() => handleEdit(student.s\_id, student)}

>

Edit

</button>

</td>

</tr>

))}

</tbody>

</table>

</div>

) : (

<p>No student details available.</p>

)}

</div>

<button

type='button'

className='btn btn-success mt-4'

onClick={handleDownloadExcel}

>

Download students

</button>

</div>

{deleteModal.show && (

<Modal show={true} onHide={() => setDeleteModal({ show: false, studentId: null })}>

<Modal.Header closeButton>

<Modal.Title>Confirm Deletion</Modal.Title>

</Modal.Header>

<Modal.Body>

<p>Are you sure you want to delete this student?</p>

</Modal.Body>

<Modal.Footer>

<Button variant="danger" onClick={() => handleDelete(deleteModal.studentId)}>

Delete

</Button>

<Button variant="secondary" onClick={() => setDeleteModal({ show: false, studentId: null })}>

Close

</Button>

</Modal.Footer>

</Modal>

)}

<h1>{console.log('Edit Modal State:', editModal)}</h1>

{/\* Edit Modal \*/}

<EditStudent

show={editModal.show}

studentId={editModal.studentId}

studentDetails={editModal.studentDetails}

onHide={handleEditModalHide}

onEdit={handleEditSuccess}

/>

</>

);

};

export default ViewStudent;

**UpdateSemester.jsx**

import React, { useState } from 'react';

import axios from 'axios';

const UpdateAllSemestersPage = () => {

const [updateStatus, setUpdateStatus] = useState(null);

const handleUpdateAllSemesters = async () => {

const confirmed = window.confirm('Are you sure you want to update all semesters?');

// If the user confirms, proceed with the update

if (confirmed) {

try {

// Send request to update all semesters except semester 6

const response = await axios.post('http://localhost:3000/api/updateAllSemesters');

// Log the response for confirmation

console.log(response.data);

setUpdateStatus('Successfully updated all semesters.');

} catch (error) {

console.error('Error updating student semesters:', error);

setUpdateStatus('Failed to update semesters. Please try again.');

}

} else {

// If the user cancels, do nothing or show a message

setUpdateStatus('Update canceled.');

}

};

return (

<div style={{ backgroundColor: 'white', padding: '20px', width: '100%', marginTop: '30px' }}>

<div style={{ textAlign: 'center', marginTop: '50px' }}>

<h1 style={{ color: '#333', padding: '20px' }}>Update All Semesters</h1>

<button

style={{

padding: '10px 20px',

fontSize: '16px',

backgroundColor: 'blue',

color: 'white',

border: 'none',

borderRadius: '5px',

cursor: 'pointer',

}}

onClick={handleUpdateAllSemesters}

>

Update All Semesters

</button>

{updateStatus && (

<p

style={{

marginTop: '20px',

fontSize: '18px',

color: updateStatus.includes('Successfully') ? 'green' : 'red',

}}

>

{updateStatus}

</p>

)}

</div>

</div>

);

};

export default UpdateAllSemestersPage;

FACULTY BIODATA MANAGEMENT

**FacultyManagement.jsx**

import React, { useState, useEffect } from 'react';

import \* as xlsx from 'xlsx';

import axios from 'axios';

const FacultyManagement = () => {

const [facultyData, setFacultyData] = useState([]); //for viewing faculty data

const [removePIN, setRemovePIN] = useState(''); //for removing faculty data

const [updateData, setUpdateData] = useState({ // for updating existing faculty data

PIN: '',

biometricID: '',

name: '',

department: '',

degree: '',

experience: '',

mobile: '',

email: '',

});

const [tableVisible, setTableVisible] = useState(false); // State to manage table visibility

const downloadTemplate = async () => {

try {

const response = await axios.get('http://localhost:5001/api/faculty/download-template', {

responseType: 'blob',

});

const url = window.URL.createObjectURL(new Blob([response.data]));

const link = document.createElement('a');

link.href = url;

link.setAttribute('download', 'faculty\_template.xlsx');

document.body.appendChild(link);

link.click();

alert('Template downloaded successfully!');

} catch (error) {

console.error('Error downloading template:', error);

alert('Error downloading template. Please try again.');

}

};

const addFaculty = async () => {

try {

// Implement logic to upload faculty data

const fileInput = document.getElementById('facultyFileInput');

const formData = new FormData();

formData.append('facultyFile', fileInput.files[0]);

await axios.post('http://localhost:5001/api/faculty/add-faculty', formData, {

headers: {

'Content-Type': 'multipart/form-data',

},

});

alert('Faculty data added successfully!');

} catch (error) {

console.error('Error adding faculty:', error);

alert('Error adding faculty. Please try again.');

}

};

const viewFaculty = async () => {

try {

// Call the endpoint to get faculty data

const response = await axios.get('http://localhost:5001/api/faculty/get-faculty');

setFacultyData(response.data);

} catch (error) {

console.error('Error getting faculty data:', error);

alert('Error getting faculty data. Please try again.');

}

};

const toggleTableVisibility = () => {

setTableVisible(!tableVisible);

};

const removeFaculty = async () => {

try {

await axios.delete(`http://localhost:5001/api/faculty/remove-faculty/${removePIN}`);

alert(`Faculty with PIN ${removePIN} removed successfully!`);

setRemovePIN(''); // Clear input after removal

viewFaculty(); // Refresh faculty data after removal

} catch (error) {

console.error('Error removing faculty:', error);

alert('Error removing faculty. Please try again.');

}

};

const editFaculty = (faculty) => {

setUpdateData({

PIN: faculty.PIN,

biometricID: faculty.biometricID,

name: faculty.name,

department: faculty.department,

degree: faculty.degree,

experience: faculty.experience,

mobile: faculty.mobile,

email: faculty.email || '',

});

};

const updateFaculty = async () => {

try {

await axios.put(`http://localhost:5001/api/faculty/update-faculty/${updateData.PIN}`, updateData);

alert('Faculty data updated successfully!');

setUpdateData({

PIN: '',

biometricID: '',

name: '',

department: '',

degree: '',

experience: '',

mobile: '',

email: '',

});

viewFaculty();

} catch (error) {

console.error('Error updating faculty:', error);

alert('Error updating faculty. Please try again.');

}

};

const downloadFacultyData = async () => {

try {

alert('Downloading...');

const filteredFacultyData = facultyData.map(({ \_id,\_\_v, ...rest }) => rest);

const ws = xlsx.utils.json\_to\_sheet(filteredFacultyData);

const wb = xlsx.utils.book\_new();

xlsx.utils.book\_append\_sheet(wb, ws, 'Faculty');

xlsx.writeFile(wb, 'faculty\_data.xlsx');

} catch (error) {

console.error('Error downloading faculty data:', error);

alert('Error downloading faculty data. Please try again.');

}

};

useEffect(() => {

// Auto-load faculty data on component mount

viewFaculty();

}, []);

return (

<div className='faculty-management-container'>

<h1 style={{fontFamily:'times new roman', color:'hsl(205, 85%, 25%)', textAlign:'center'}}>Faculty Management</h1><hr />

<h4 style={{fontFamily:'arial', color:'hsl(205, 85%, 25%)'}}>Download Faculty Data Template</h4>

<button onClick={downloadTemplate} className='faculty-button'>Download Template</button><hr />

<h4 style={{fontFamily:'arial', color:'hsl(205, 85%, 25%)'}}>Add Faculty</h4>

<input

type="file"

id="facultyFileInput"

accept=".xlsx, .xls"

/>

<button onClick={addFaculty} className='faculty-button'>Add Faculty</button> <hr />

<h4 style={{fontFamily:'arial', color:'hsl(205, 85%, 25%)'}}>Remove Faculty</h4>

<label htmlFor="removePIN">Enter PIN to remove faculty:</label>

<input

type="text"

id="removePIN"

value={removePIN}

onChange={(e) => setRemovePIN(e.target.value)}

/> <br /><br />

<button onClick={removeFaculty} className='faculty-button'>Remove Faculty</button> <hr />

<div className='edit-form'>

<h4 style={{fontFamily:'arial', color:'hsl(205, 85%, 25%)'}}>Edit Faculty</h4>

<input

type='text'

placeholder='PIN'

value={updateData.PIN}

onChange={(e) => setUpdateData({ ...updateData, PIN: e.target.value })}

/>

<input

type='text'

placeholder='Biometric ID'

value={updateData.biometricID}

onChange={(e) => setUpdateData({ ...updateData, biometricID: e.target.value })}

/>

<input

type='text'

placeholder='Name'

value={updateData.name}

onChange={(e) => setUpdateData({ ...updateData, name: e.target.value })}

/>

<input

type='text'

placeholder='Department'

value={updateData.department}

onChange={(e) => setUpdateData({ ...updateData, department: e.target.value })}

/>

<input

type='text'

placeholder='Degree'

value={updateData.degree}

onChange={(e) => setUpdateData({ ...updateData, degree: e.target.value })}

/>

<input

type='text'

placeholder='Experience'

value={updateData.experience}

onChange={(e) => setUpdateData({ ...updateData, experience: e.target.value })}

/>

<input

type='text'

placeholder='Mobile'

value={updateData.mobile}

onChange={(e) => setUpdateData({ ...updateData, mobile: e.target.value })}

/>

<input

type='text'

placeholder='Email'

value={updateData.email}

onChange={(e) => setUpdateData({ ...updateData, email: e.target.value })}

/> <br /><br />

<button onClick={updateFaculty} className='faculty-button'>

Update Faculty

</button> <hr />

</div>

<h4 style={{ fontFamily: 'arial', color: 'hsl(205, 85%, 25%)' }}>View Faculty</h4>

<button onClick={toggleTableVisibility} className='faculty-button'>

{tableVisible ? 'Hide Faculty' : 'View Faculty'}

</button> <br /><br />

{tableVisible && ( // Conditionally render the table based on visibility state

<table className='faculty-table'>

<thead>

<tr>

<th>PIN</th>

<th>Biometric ID</th>

<th>Name</th>

<th>Department</th>

<th>Degree</th>

<th>Experience</th>

<th>Mobile</th>

<th>Email</th>

<th>Edit</th>

</tr>

</thead>

<tbody>

{facultyData.map((faculty) => (

<tr key={faculty.\_id}>

<td>{faculty.PIN}</td>

<td>{faculty.biometricID}</td>

<td>{faculty.name}</td>

<td>{faculty.department}</td>

<td>{faculty.degree}</td>

<td>{faculty.experience}</td>

<td>{faculty.mobile}</td>

<td>{faculty.email}</td>

<td>

<button onClick={() => editFaculty(faculty)}>Edit</button>

</td>

</tr>

))}

</tbody>

</table>

)} <hr />

<h4 style={{ fontFamily: 'arial', color: 'hsl(205, 85%, 25%)' }}>Download Faculty Data</h4>

<button onClick={downloadFacultyData} className='faculty-button'>Download Faculty</button>

</div>

);

};

export default FacultyManagement;

**STUDENT ATTENDANCE MANAGEMENT**

**TakeAttendance.jsx**

import React, { useState, useEffect } from 'react';

import axios from 'axios';

const TakeAttendance = () => {

const [departments, setDepartments] = useState([]);

const [semesterOptions] = useState([1, 2, 3, 4, 5, 6]);

const [shiftOptions] = useState([1, 2]);

const [department, setDepartment] = useState('');

const [semester, setSemester] = useState('');

const [shift, setShift] = useState('');

const [subject, setSubject] = useState('');

const [attendanceResults, setAttendanceResults] = useState([]);

const [showDateInputAttendance, setShowDateInputAttendance] = useState(false);

const [attendanceStatusArray, setAttendanceStatusArray] = useState([]);

const [attendanceDate, setAttendanceDate] = useState('');

const [fetchAttempted, setFetchAttempted] = useState(false);

const [selectAll, setSelectAll] = useState(false);

const [showHolidayModal, setShowHolidayModal] = useState(false);

const [holidayType, setHolidayType] = useState('H');

const [holidayDate, setHolidayDate] = useState('');

useEffect(() => {

fetchDepartments();

}, []);

const fetchDepartments = async () => {

try {

const response = await axios.get('http://localhost:3000/api/departments');

setDepartments(response.data);

} catch (error) {

console.error('Error fetching departments:', error);

}

};

const updateStatus = (index) => {

setAttendanceStatusArray((prevArray) => {

const newArray = [...prevArray];

newArray[index] = newArray[index] === 'A' ? 'P' : 'A';

return newArray;

});

};

const toggleSelectAll = () => {

setSelectAll(!selectAll);

setAttendanceStatusArray((prevArray) => prevArray.map(() => (selectAll ? 'A' : 'P')));

};

const submitInput = () => {

if (department && semester && subject && shift) {

setFetchAttempted(true);

axios.post('http://localhost:3000/api/student\_details', { department, semester, subject,shift })

.then(response => {

console.log('Student details:', response.data);

setAttendanceResults(response.data);

setAttendanceStatusArray(new Array(response.data.length).fill('A'));

//setShowDateInput(true);

})

.catch(error => {

console.error('Error fetching student details:', error);

});

} else {

alert('Please fill in all the fields');

}

};

const submitAttendance = () => {

if (attendanceResults.length === 0) {

alert('No students are enrolled in the specified department and semester.');

} else {

axios.post('http://localhost:3000/api/submit\_attendance', {

attendanceStatusArray: attendanceStatusArray,

})

.then(response => {

console.log('Attendance status array sent to the server successfully', response.data);

alert('Attendance submitted successfully.');

})

.catch(error => {

console.error('Error sending attendance status array to the server:', error);

alert('Already submitted or Error submitting attendance.');

});

}

};

const submitAttendanceDate = () => {

if (!attendanceDate) {

alert('Please enter a valid date.');

return;

}

axios.post('http://localhost:3000/api/date\_attendance', {

attendanceDate: attendanceDate,

})

.then(response => {

if (response.data.message === 'Attendance date received successfully.') {

console.log('Date sent to the server successfully', response.data);

alert("Today's attendance added successfully");

} else {

console.error('Error adding date to the server:', response.data.message);

alert("Error submitting today's Attendance. Please try again.");

}

})

.catch(error => {

console.error('Error sending date to the server:', error);

alert("Already Submitted or Error submitting today's Attendance.");

});

setAttendanceDate('');

setShowDateInputAttendance(false);

};

const openHolidayModal = () => {

setShowHolidayModal(true);

};

const closeHolidayModal = () => {

setShowHolidayModal(false);

};

const showDate = () => {

setShowDateInputAttendance(true);

};

const submitHoliday = () => {

// Make a POST request to the server endpoint with holiday data

axios.post('http://localhost:3000/api/holiday\_attendance', {

holidayType: holidayType,

holidayDate: holidayDate,

})

.then(response => {

// Handle the response from the server

console.log('Holiday data sent to the server successfully', response.data);

alert('Holiday/Weekend submitted successfully.');

// Close the modal after successful submission

setShowHolidayModal(false);

})

.catch(error => {

console.error('Error sending holiday data to the server:', error);

alert('Already submitted or Error submitting Holiday/Weekend.');

});

};

return (

<div className='take-attendance'>

<h2 style={{fontFamily:'times new roman', color:'hsl(205, 85%, 25%)', textAlign:'center'}}>Take Attendance</h2><hr />

<label style={{fontFamily:'arial', color:'hsl(205, 85%, 25%)',fontWeight:'bold'}}>

Department:

<select

value={department}

onChange={(e) => setDepartment(e.target.value)}

required

style={{ margin: '20px', fontSize: '18px',padding:'3px' }}

>

<option value="" disabled>Select Department</option>

{departments.map((dept) => (

<option key={dept.D\_id} value={dept.D\_name}>

{dept.D\_name}

</option>

))}

</select>

</label>

<br />

<label style={{fontFamily:'arial', color:'hsl(205, 85%, 25%)',fontWeight:'bold'}}>

Semester:

<select

value={semester}

onChange={(e) => setSemester(e.target.value)}

required

style={{ margin: '20px', fontSize: '18px',padding:'3px' }}

>

<option value="" disabled>Select Semester</option>

{semesterOptions.map((sem) => (

<option key={sem} value={sem}>

{sem}

</option>

))}

</select>

</label>

<br />

<label style={{fontFamily:'arial', color:'hsl(205, 85%, 25%)',fontWeight:'bold'}}>

Shift:

<select

value={shift}

onChange={(e) => setShift(e.target.value)}

required

style={{ margin: '20px', fontSize: '18px',padding:'3px' }}

>

<option value="" disabled>Select Shift</option>

{shiftOptions.map((shift) => (

<option key={shift} value={shift}>

{shift}

</option>

))}

</select>

</label>

<br />

<label style={{fontFamily:'arial', color:'hsl(205, 85%, 25%)',fontWeight:'bold'}}>

Subject:

<input

type="text"

value={subject}

onInput={(e) => {

const inputValue = e.target.value;

if (/^[a-zA-Z0-9\_]+$/.test(inputValue)) {

setSubject(inputValue);

}

}}

pattern="[a-zA-Z0-9\_]+"

title="Only letters, numbers, and underscores are allowed"

required

style={{ margin: '20px', fontSize: '20px' }}

/>

</label>

<br />

<button onClick={submitInput} className='faculty-button' style={{fontSize:'18px', height:'30px'}}>Mark Attendance</button><br /><br />

{/\* "Mark Holiday/Weekend" section \*/}

<button onClick={openHolidayModal} className='faculty-button' style={{fontSize:'18px',height:'30px'}}>Mark Holiday/Weekend</button><br /><br />

{/\* Holiday/Weekend Modal \*/}

{showHolidayModal && (

<div>

<label>

Select Type:

<select

value={holidayType}

onChange={(e) => setHolidayType(e.target.value)}

required

style={{ margin: '20px', fontSize: '20px' }}

>

<option value="H">Holiday</option>

<option value="W">Weekend</option>

</select>

</label>

<br />

<label>

Select Date:&emsp;

<input

type="date"

value={holidayDate}

onChange={(e) => setHolidayDate(e.target.value)}

/>

</label>

<br />

<br />&emsp;

<button onClick={submitHoliday} className='faculty-button'>Submit</button>&emsp;

<button onClick={closeHolidayModal} className='faculty-button'>Cancel</button>

</div>

)}

{fetchAttempted && attendanceResults.length === 0 && department && semester && subject && (

<div>

<p>No students are enrolled in the specified department and semester.</p>

</div>

)}

{attendanceResults.length > 0 && (

<>

<h2>Attendance Table</h2>

<label>

Select All

<input

type="checkbox"

checked={selectAll}

onChange={toggleSelectAll}

/>

</label>

<table border="1" width="60%" style={{ textAlign: 'center' }}>

<thead>

<tr>

<th>Pin</th>

<th>Name</th>

<th>Status</th>

</tr>

</thead>

<tbody>

{attendanceResults.map((student, index) => (

<tr key={index}>

<td>{student.s\_id}</td>

<td>{student.sname}</td>

<td>

<input

type="checkbox"

id={`status${index}`}

checked={attendanceStatusArray[index] === 'P'}

onChange={() => updateStatus(index)}

/>

</td>

</tr>

))}

</tbody>

</table>

<button onClick={submitAttendance} className='faculty-button'>Submit</button>

<div>

<p>Click this button to add Complete today's Attendance</p>

<p>Note:-<span style={{color:'red'}}>select this only after taking all subjects Attendance</span></p>

<button onClick={showDate} className='faculty-button'>Enter Date</button>

</div>

<div id="attendanceResult"></div>

{showDateInputAttendance && (

<>

<label htmlFor="attendanceDate">Enter Attendance Date:</label>

<input

type="date"

id="attendanceDate"

value={attendanceDate}

onChange={(e) => setAttendanceDate(e.target.value)}

/>&emsp;

<button onClick={submitAttendanceDate} className='faculty-button'>Submit Attendance Date</button>

</>

)}

</>

)}

</div>

);

};

export default TakeAttendance;

**ViewAttendance.jsx**

import React, { useState } from 'react';

import axios from 'axios';

function AttendanceComponent() {

const [attendanceData, setAttendanceData] = useState([]);

const [startDate, setStartDate] = useState('');

const [endDate, setEndDate] = useState('');

const [studentId, setStudentId] = useState('');

const [showPinColumn, setShowPinColumn] = useState(false);

const handleFormSubmit = async (e) => {

e.preventDefault();

if (studentId) {

try {

console.log('Sending request with parameters:', { studentId, startDate, endDate });

const response = await axios.get('http://localhost:3000/api/attendance', {

params: {

studentId,

startDate: startDate || null, // Conditionally include startDate

endDate: endDate || null, // Conditionally include endDate

},

});

console.log('Attendance details:', response.data);

if (response.data.length === 0) {

setAttendanceData([]);

setShowPinColumn(false);

alert("No attendance found");

console.log("Error: No attendance status available");

} else {

setAttendanceData(response.data);

setShowPinColumn(true); }

} catch (error) {

console.error('Error fetching attendance details:', error);

if (error.response) {

if (error.response.status === 404) {

setAttendanceData([]);

setShowPinColumn(false);

alert("Invalid PIN. Please enter a correct PIN.");

console.log("Error: Invalid PIN");

} else {

setAttendanceData([]);

setShowPinColumn(false);

alert("Error fetching attendance details kindly check specified DATE or Entered PIN");

console.log("Error: Error fetching attendance details");

}

} else {

setAttendanceData([]);

setShowPinColumn(false);

alert("Network error. Please try again."); }

}

} else {

alert('Please fill in all the fields'); }

};

return (

<div className='view-attendance'>

<div className="inner-view-attendance">

<h2 style={{fontFamily:'times new roman', color:'hsl(205, 85%, 25%)', textAlign:'center'}}>Check Attendance</h2><hr />

<form onSubmit={handleFormSubmit}>

<label>

Student Pin:

</label>

<input

type="text"

style={{marginLeft:'15px'}}

value={studentId}

onChange={(e) => setStudentId(e.target.value)}

pattern="[0-9]{5}-[a-zA-Z]{2}-[0-9]{3}"

title="Please enter a valid student ID (e.g.,21001-CS-033)"

required

/>

<br /><br />

<button type="submit" className='faculty-button'>Submit</button> <br /><br />

<p style={{fontFamily:'times new roman',textDecoration:'underline',fontSize:'18px'}}> Select 'From Date' and 'To Date' to get Attendance of specific Month</p>

<label>

From Date:

</label>

<input style={{marginLeft:'15px'}} type="date" value={startDate} onChange={(e) => setStartDate(e.target.value)} />

<br /><br />

<label>

To Date:

</label>

<input style={{marginLeft:'15px'}} type="date" value={endDate} onChange={(e) => setEndDate(e.target.value)} /><br/>

</form>

{showPinColumn && (

<div>

<br />

<table border="2">

{/\* Display student details \*/}

<thead>

<tr>

<th>Pin</th>

</tr>

</thead>

<tbody>

{attendanceData.map((entry, index) => (

<tr key={index}>

<td>{entry.s\_id}</td>

</tr>

))}

</tbody>

</table>

</div>

)}

{attendanceData.length > 0 && (

<div>

<h2>Attendance</h2>

<table border="2">

<thead>

<tr>

<th>Date</th>

<th>Status</th>

</tr>

</thead>

<tbody>

{Object.keys(attendanceData[0])

.slice(1) // Exclude Pin from the iteration

.map((date, index) => (

<tr key={index}>

<td><b>{date}</b></td>

<td style={{ color: attendanceData[0][date] === 'A' ? 'red' : (attendanceData[0][date] === 'P' ? 'green' : 'black') }}>

<b>{attendanceData[0][date]}</b>

</td>

</tr>

))}

</tbody>

</table>

</div>

)}

</div>

</div>

);

}

export default AttendanceComponent;

**ViewAttendanceServer.js**

const getAttendanceData = (startDate, endDate, studentId, callback) => {

console.log('Executing getAttendanceData function');

console.log('startDate:', startDate);

console.log('endDate:', endDate);

console.log('studentId:', studentId);

const dateColumns = [];

const currentDate = new Date(startDate);

while (currentDate <= new Date(endDate)) {

const day = currentDate.getDate().toString().padStart(2, '0');

const month = (currentDate.getMonth() + 1).toString().padStart(2, '0');

const year = currentDate.getFullYear();

const formattedDate = `${day}\_${month}\_${year}`;

dateColumns.push(`\`${formattedDate}\``);

currentDate.setDate(currentDate.getDate() + 1);

}

const selectColumns = ['s\_id', ...dateColumns].join(', ');

let query;

if(!startDate && !endDate){ query = `SELECT \* FROM attendance WHERE s\_id = ? `; }

else{ query = `SELECT ${selectColumns} FROM attendance WHERE s\_id = ? `; }

pool.query(query, [studentId], (err, results) => {

if (err) {

console.error('Error executing MySQL query:', err);

callback(err, null);

} else {

callback(null, results);

}

});

};

app.get('/api/attendance', (req, res) => {

console.log('Received request with parameters:', req.query);

const { startDate, endDate, studentId } = req.query;

if (!studentId) {

return res.status(400).json({ error: 'Bad Request: Missing required parameters' });

}

getAttendanceData(startDate, endDate, studentId, (err, results) => {

if (err) {

res.status(500).json({ error: 'Internal Server Error' });

} else {

res.json(results);

}

});

});

6. SYSTEM TEST CASES

**White Box Testing**

**Definition:**

White box testing is a testing technique that examines the internal structure and logic of an application's code. Also known as "clear box testing," "glass box testing," or "structural testing," this method involves evaluating the code and its internal pathways, conditions, and data structures. The goal of white box testing is to ensure that all components of the software application function as intended, and it often requires detailed knowledge of the internal workings of the software.

**Key aspects of white box testing include:**

**Code Coverage:** White box testing aims to achieve high code coverage, ensuring that a significant portion of the code is exercised during testing. This helps in identifying areas of the code that are not executed.

**Path Testing:** Test cases are designed to cover various paths through the code, including different branches, loops, and decision points. This ensures that different execution scenarios are tested.

**Statement Coverage:** This involves ensuring that every line of code is executed at least once during testing. It helps in identifying any unreachable or dead code.

**Branch Coverage:** This measures the percentage of branches (decision points) that have been executed. It ensures that both true and false conditions of decision points are tested.

**Loop Testing:** White box testing also involves testing loops to ensure that the code behaves correctly when loops are executed a specific number of times or not at all.

**Data Flow Testing:** This focuses on how data is input, processed, and output within the application. Test cases are designed to follow the flow of data through the code.

**Integration Testing:** White box testing is often part of the integration testing process, where the interactions between different components of the software are examined.

**Unit Testing:** Unit tests, which are focused on individual functions or modules, are a common practice in white box testing.

**Black Box Testing**

**Definition:**

Black box testing is a software testing technique where the internal workings, code structure, and implementation details of a system are not known to the tester. The testing focuses on validating the functionality and behavior of the system by providing inputs and observing outputs, treating the software as a "black box." The objective is to ensure that the software meets its specified requirements and functions correctly from the end-user's perspective.

**Key Points :**

**1. Visibility into Internal Structure:**

* Tester does not have access to the internal code, logic, or implementation details of the software.

**2. Objective:**

* Focuses on validating the functionality and behavior of the system without knowledge of its internal workings.

**3. Testing Technique:**

* Involves providing inputs and observing outputs, treating the software as a "black box."

**4. Testing Levels:**

* Applicable at various testing levels, including unit testing, integration testing, system testing, and acceptance testing.

**White Box Testing :**

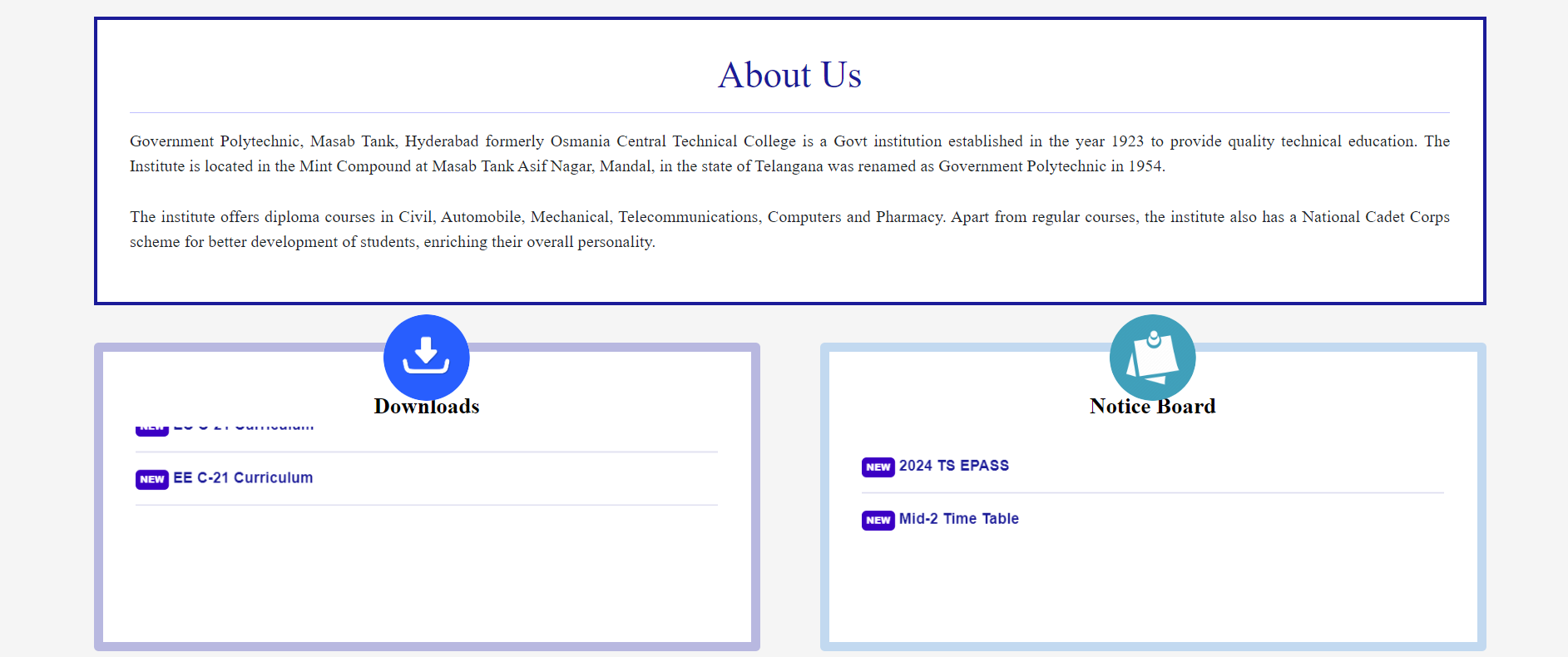
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Test Case Code | Module/Method | Test Scenario | Expected Outcome | Actual Result | Status  (Pass/Fail) |
| 1 | AddStudent | Upload valid Excel file | File uploaded successfully | File uploaded successfully | Pass |
| 2 | AddStudent | Upload without selecting a file | Error: "No file selected" | Error: "No file selected" | Pass |
| 3 | AddStudent | Process Sixth Semester successfully | Success message displayed | Success message displayed | Pass |
| 4 | AddStudent | Upload valid Excel file | Error: "No file selected" | Error: "No file selected" | Pass |
| 5 | ViewStudent | Fetch student details successfully | Student details displayed | Student details displayed | Pass |
| 6 | ViewStudent | Fetch student details with error  Error message displayed | Error message displayed | Error message displayed | Pass |
| 7 | ViewStudent | Click "Delete" for a student | Confirm deletion modal displayed | Confirm deletion modal displayed | Pass |
| 8 | ViewStudent | Click "Edit" for a student | Edit modal displayed with data | Edit modal displayed with data | Pass |
| 9 | EditStudent | Update student details | Details updated successfully | Details updated successfully | Pass |
| 10 | EditStudent | Update with invalid data | Error: "Invalid data" | Error: "Invalid data" | Pass |
| 11 | downloadTemplate | Download the faculty template | The template is downloaded successfully | The template is downloaded successfully | Pass |
| 12 | addFaculty | Add faculty data using a valid Excel file | Faculty data added successfully | Faculty data added successfully | Pass |
| 13 | viewFaculty | View faculty data | Faculty data is displayed | Faculty data is displayed | Pass |
| 14 | toggleTableVisibility | Toggle visibility of the faculty table | The faculty table visibility toggles correctly | The faculty table visibility toggles correctly | Pass |
| 15 | removeFaculty | Remove faculty data using a valid PIN | Faculty with the specified PIN is removed successfully | Faculty with the specified PIN is removed successfully | Pass |
| 16 | editFaculty | Edit faculty data | Faculty data is loaded into the update form | Faculty data is loaded into the update form | Pass |
| 17 | updateFaculty | Update faculty data using the update form | Faculty data is updated successfully | Faculty data is updated successfully | Pass |
| 18 | downloadFacultyData | Download faculty data | Faculty data is downloaded successfully | Faculty data is downloaded successfully | Pass |
| 19 | markAttendance | Add Attendance | Attendance added Successfully | Attendance added Successfully | Pass |
| 20 | viewAttendance | View Attendance | Attendance table displayed successfully | Attendance table displayed successfully | Pass |

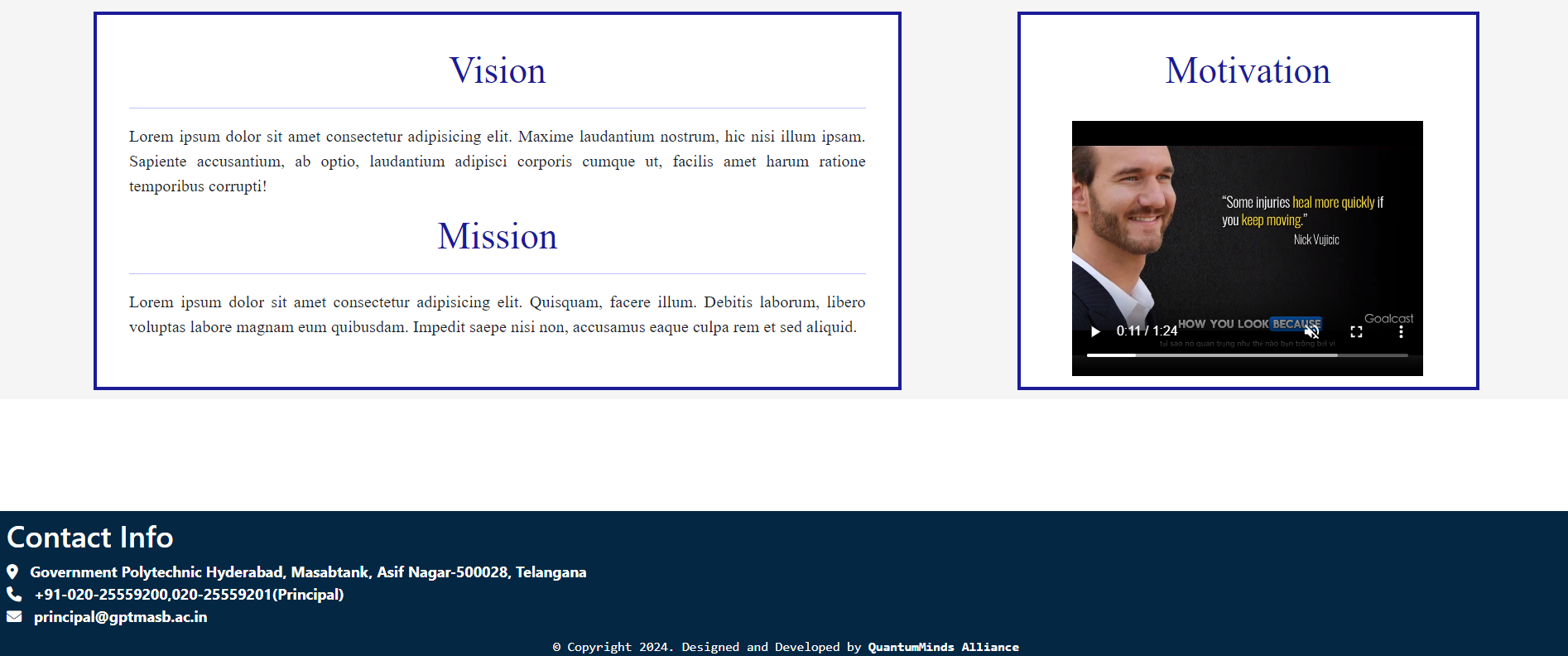
**Black Box Testing :**

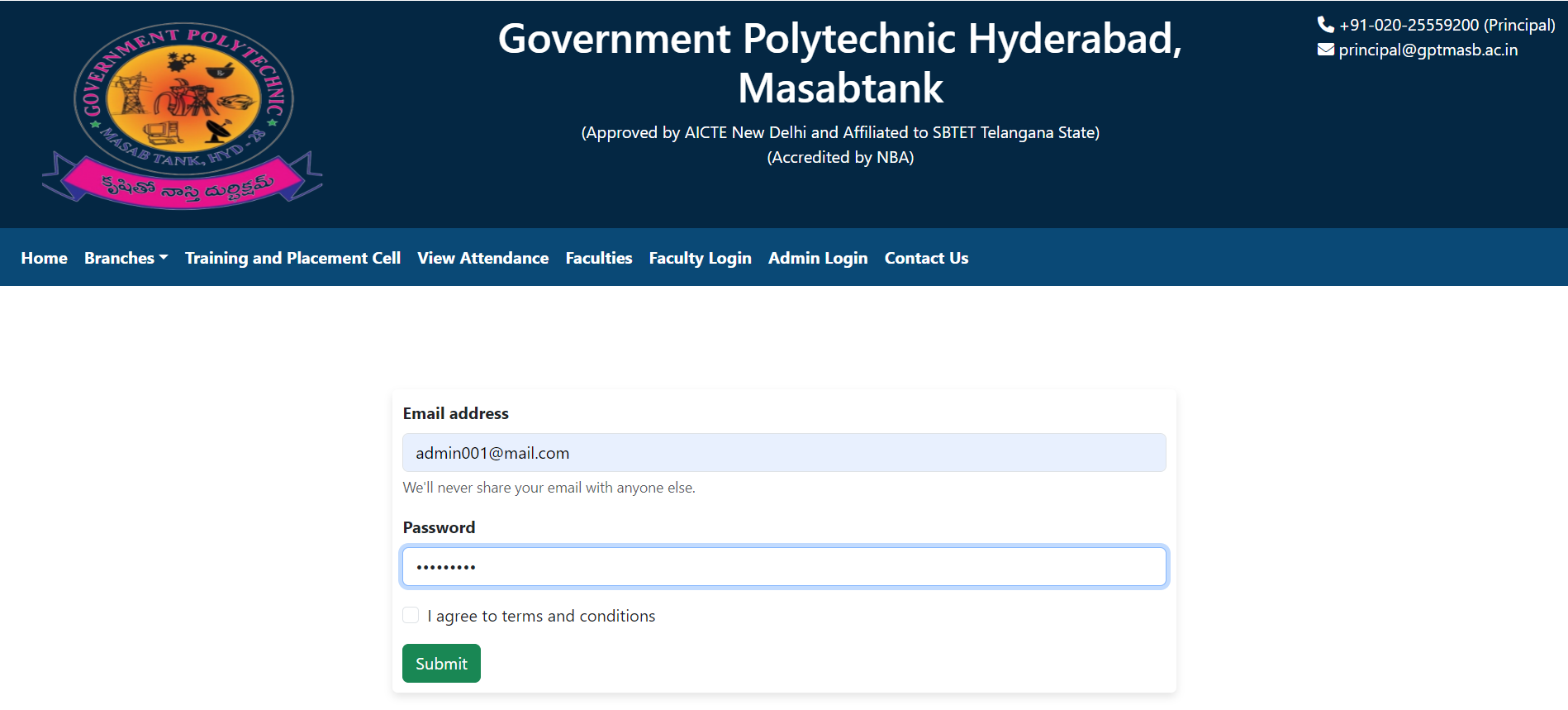
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Test Case Code | Test Case Description | Test Data | Expected Result | Actual Result | Status  (Pass/Fail) |
| 1 | Import student data from a valid Excel sheet And Uploading Into Database | Pin:21001-CS-030  Name: Omkar  Date Of Birth: 2005/08/02  Father Name: Madan Rao  Mobile No:6303989090  Address: XYZ  Email Id: xyz@gmail.com  Aadhar: 565678987890  Semester: 2  Department: cse | Data Successfully Added in Database | Data Successfully Added in Database | Pass |
| 2 | Attempt To Import Redundant data | Pin:21001-CS-030  Name: Omkar  Date Of Birth: 2005/08/02  Father Name: Madan Rao  Mobile No:6303989090  Address: XYZ  Email Id: xyz@gmail.com  Aadhar: 565678987890  Semester: 2  Department: cse | Data Already Available in the Database | Data Already Available in the Database | Pass |
| 3 | Delete a student from the system | Student PIN: 21001-CS-030 | Student Deleted Successfully | Student Deleted Successfully | Pass |
| 4 | Edit student details | Student PIN:21001-CS-030 | Student Details  updated in the Database Successfully | Student Details  updated in the Database Successfully | Pass |
| 5 | Move students to Completed Students table after 6th semester | Students in 6th semester: {StudentID1, StudentID2 | Students Moved Successfully into Complted\_students Table | Students Moved Successfully into Complted\_students Table | Pass |
| 6 | Verify Update Semester Button | Students in semesters 1 to 5: {StudentID1, StudentID2} | Student Semester Updated Successfully | Student Semester Updated Successfully | Pass |
| 7 | Import Faculty data from a valid Excel sheet And Uploading Into Database | Pin:21001-CS-030  Biometric ID: 15058-21086  Mobile No:6303989090  Address: XYZ  Email Id: xyz@gmail.com  Aadhar: 565678987890  Department: cse | Data Successfully Added to the Database | Data Successfully Added In Database | Pass |
| 8 | Delete a Faculty from the system | Faculty PIN: 21001-CS-030 | Faculty Deleted Successfully | Faculty Deleted Successfully | Pass |
| 9 | Edit Faculty details | Pin:21001-CS-030  Biometric ID: 15058-21086  Mobile No:6303989090  Address: XYZ  Email Id: xyz@gmail.com  Aadhar: 565678987890  Department: CS | Faculty Details Edited Successfully | Faculty Details Edited Successfully | Pass |
| 9 | Verify View Functionality | View Faculty button click | Faculty Details Visisble on the Screen with there unique id | Faculty Details Visisble on the Screen with there unique id | Pass |
| 10 | Verify the Add Event Fucntionality | Selected Date: 03/02/2007  Add Event: Freshers & Farewell | Event Added Successfully on Selected Date | Event Added Successfully on Selected Date | Pass |
| 11 | Verify Delete Functionality of the Added Event | Date: 03/02/2007 | Event Deleted  Successfully | Event Deleted  Successfully | Pass |
| 12 | Verify Add Attendance | Department: CSE  Semester: 1  Shift: 2  Subject: C++ | Attendance Added Successfully | Attendance Added Successfully | Pass |
| 13 | Verify View Attendance | Pin: 21001-CS-030 | Attendance of The Entered Pin Is Displayed | Attendance of The Entered Pin Is Displayed | Pass |

## 7. OUTPUT SCREENS

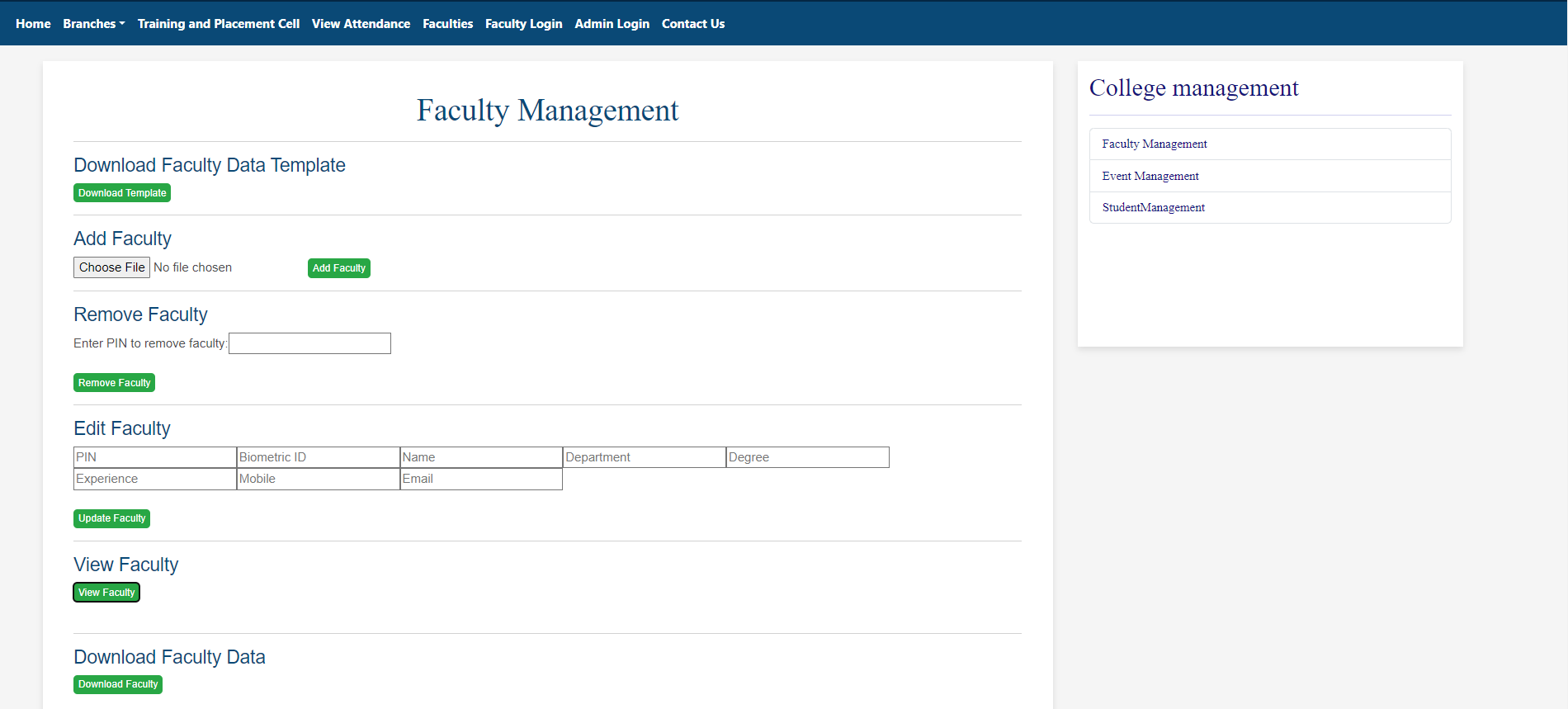




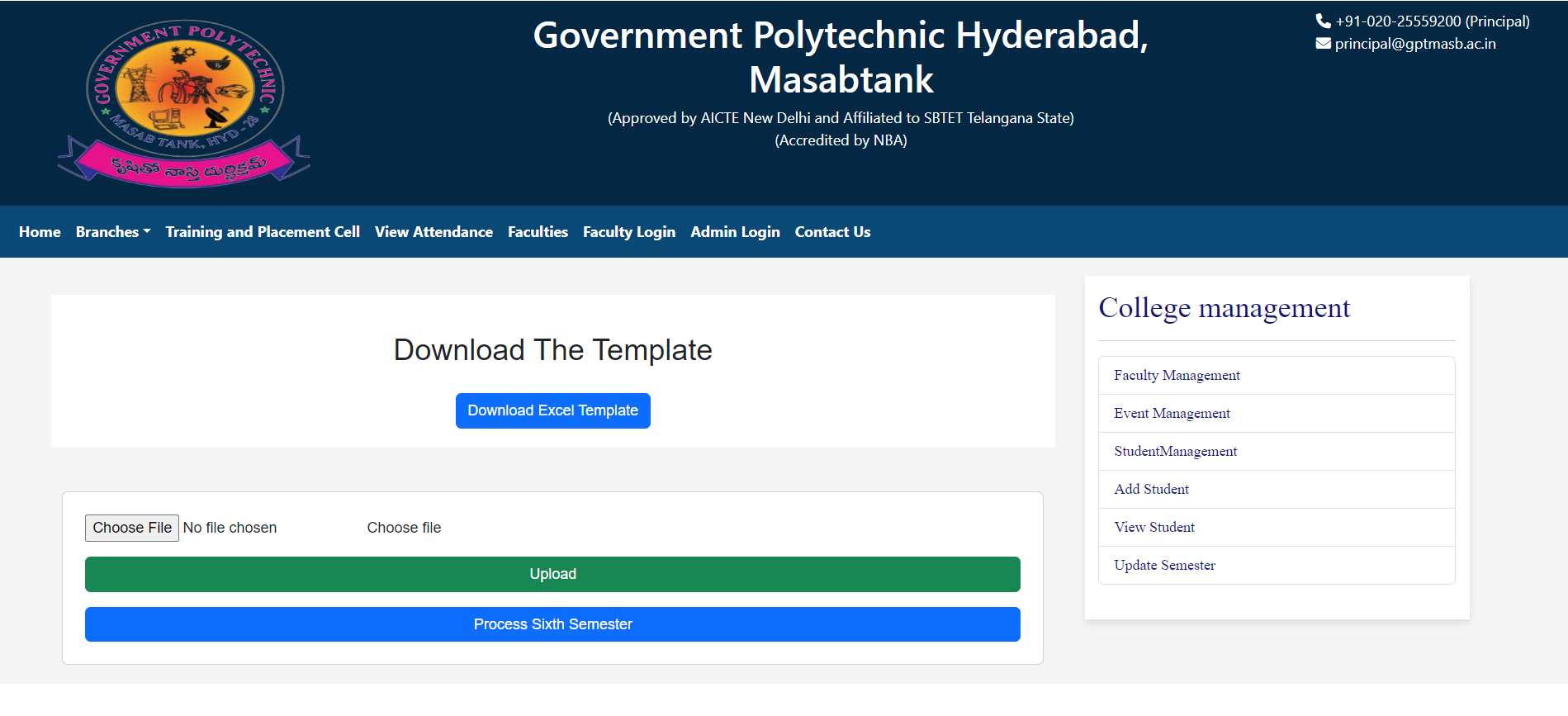




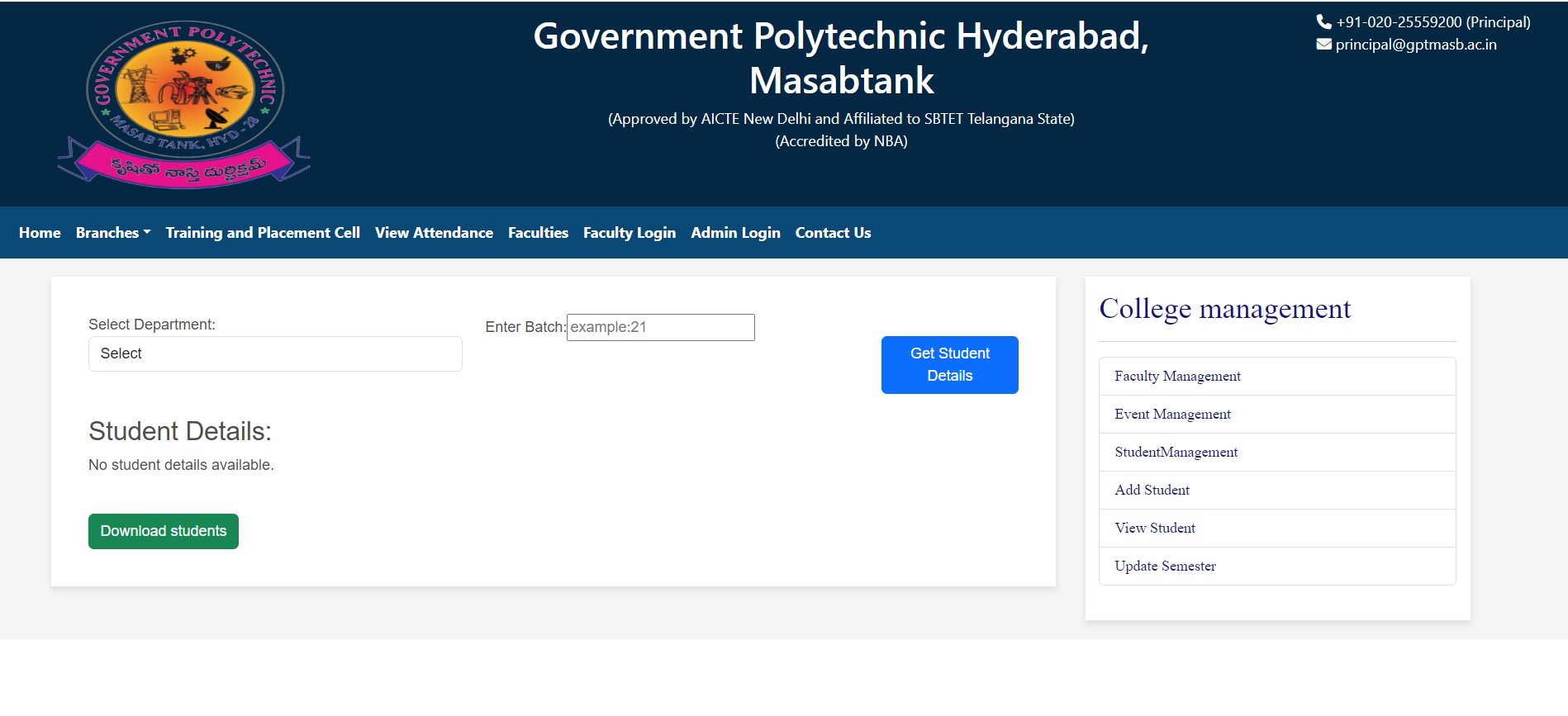
7.1 Admin Authentication



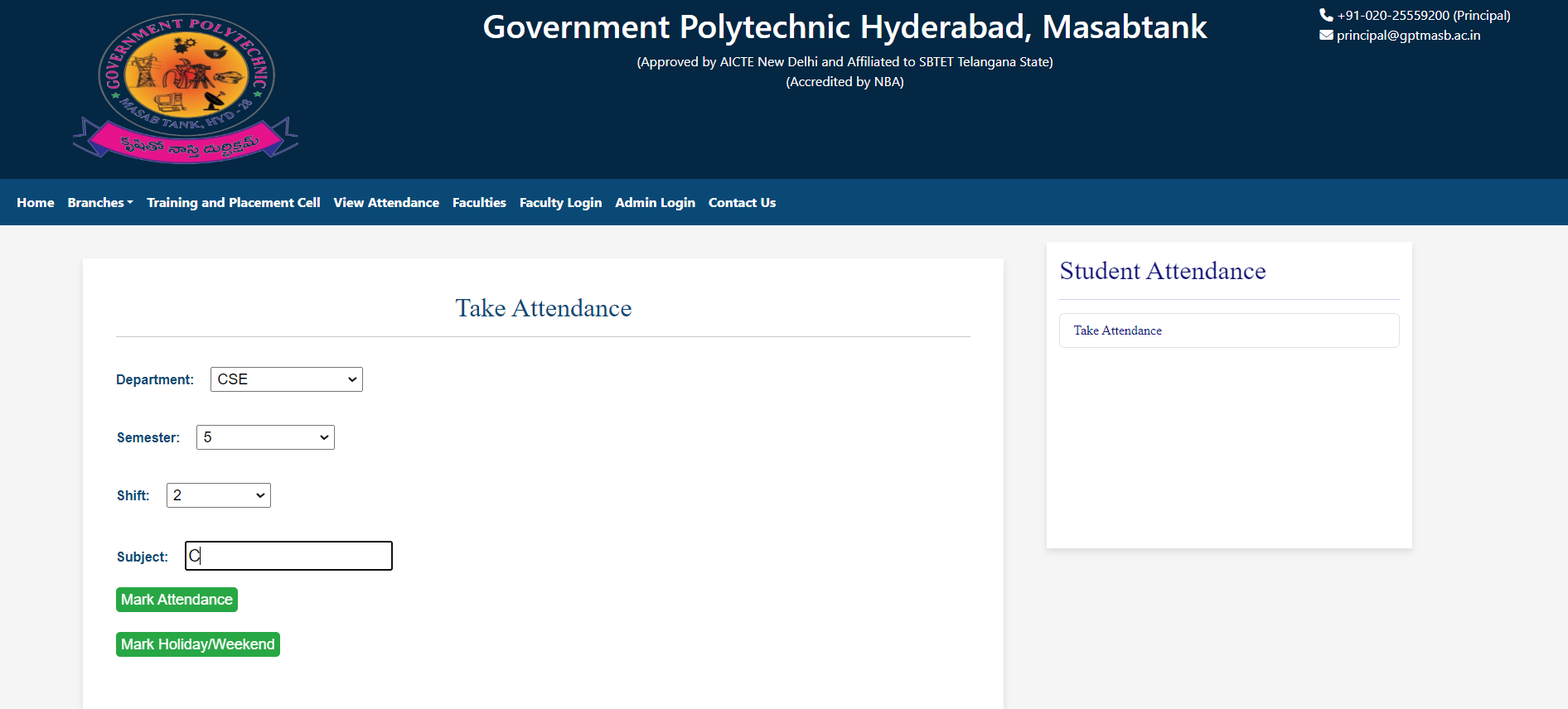
7.2 Faculty Management



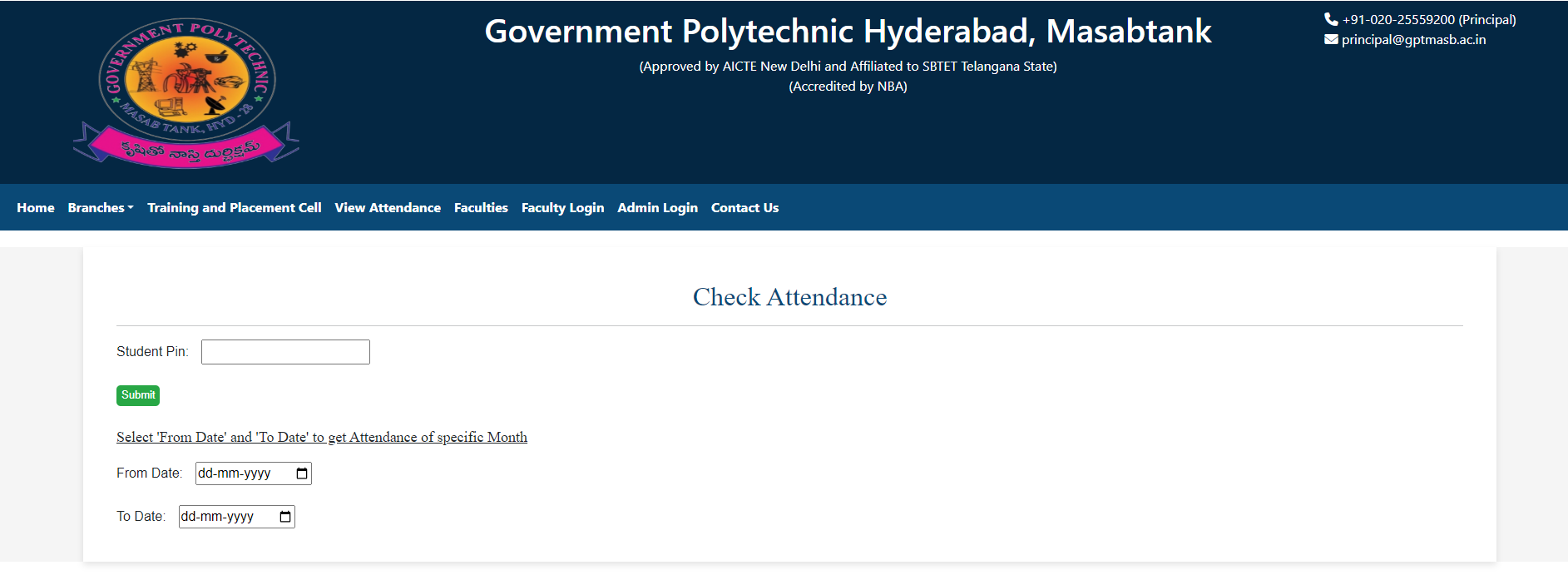
7.3.1 Add Student

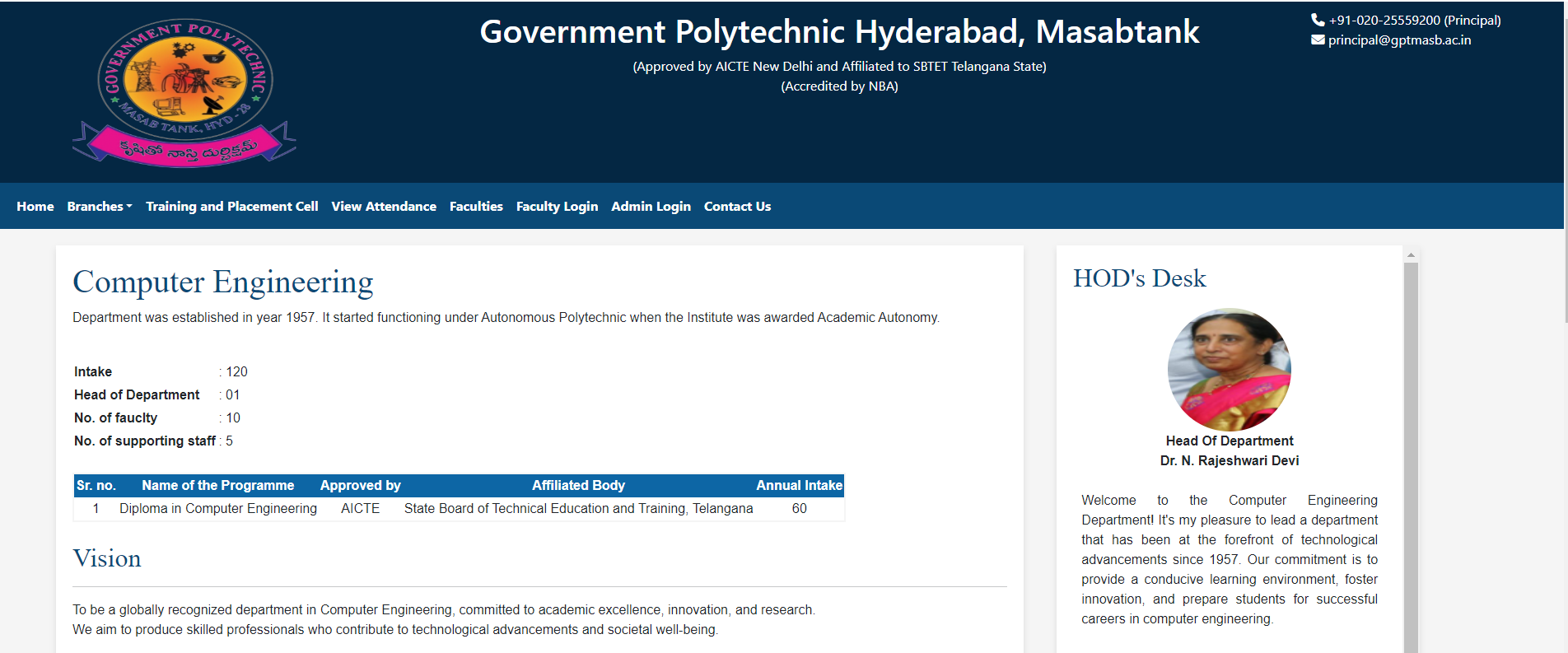
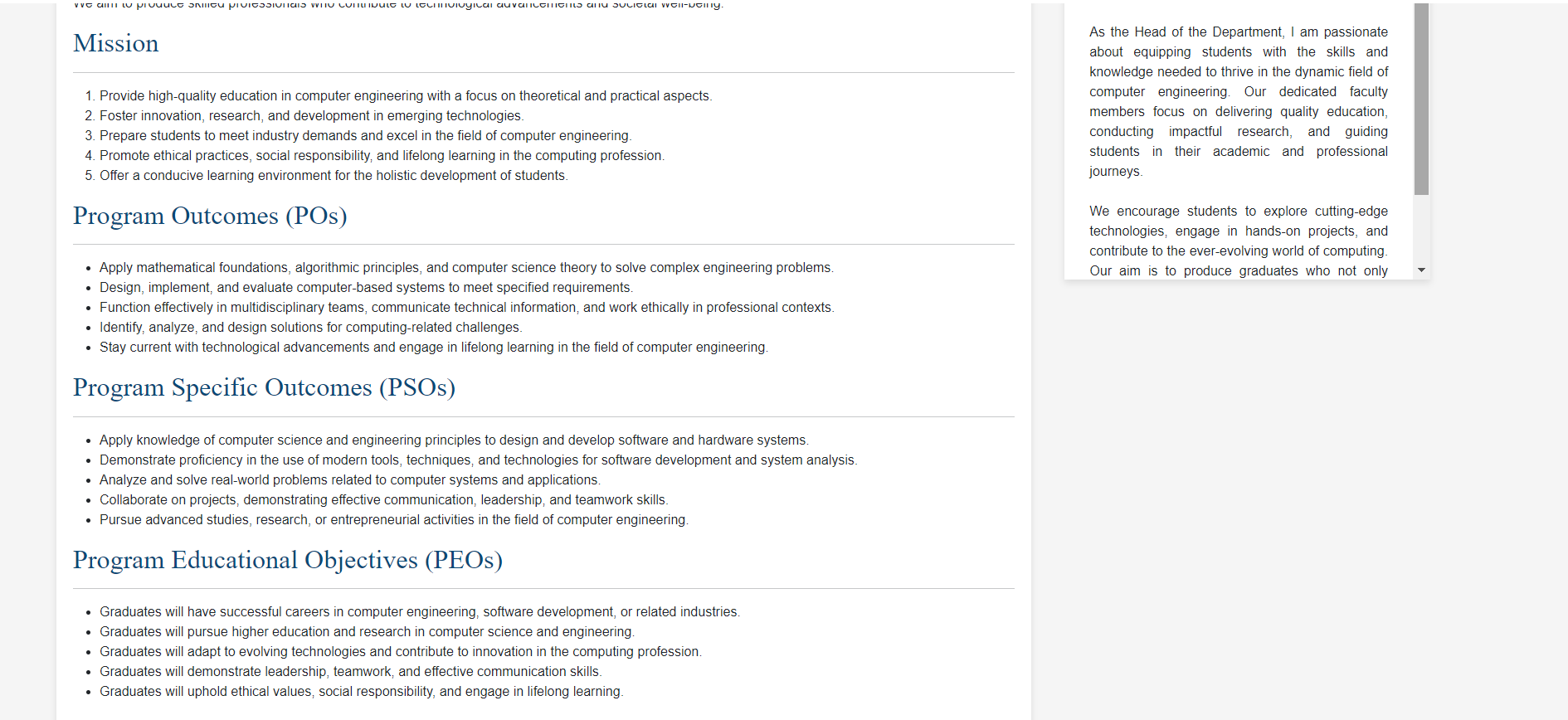


7.3.2 View Student

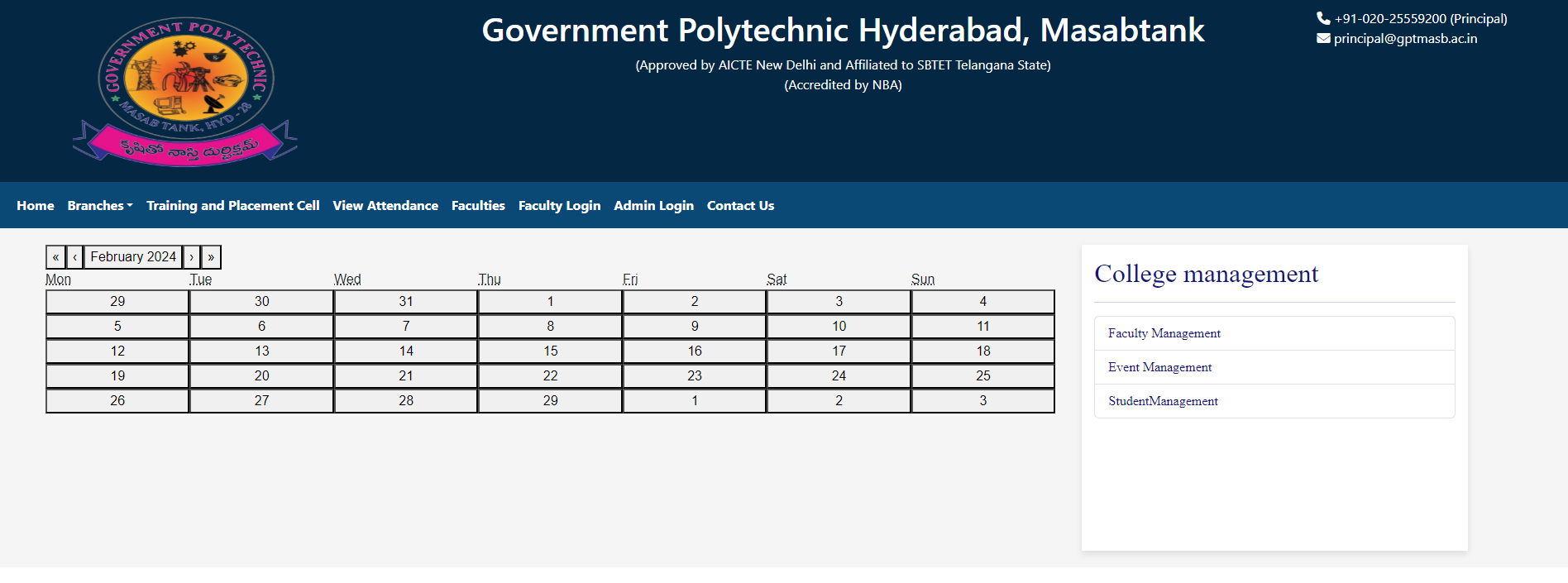


7.4 Attendance



7.5ViewAttendance  

7.6 Computer Engineering Department



7.7 Event Management

## 8. FUTURE SCOPE

The future scope of a college website can be broad and dynamic, evolving with advancements in technology and changes in education trends. Here are several potential areas for growth and development:

**1. Personalized Learning Platforms:**

* Implementing personalized learning experiences for students based on their preferences, progress, and learning styles.
* Incorporating adaptive learning technologies to tailor content and assessments to individual student needs.

**2. Virtual and Augmented Reality Integration:**

* Offering virtual campus tours and immersive experiences for prospective students.
* Utilizing virtual reality (VR) or augmented reality (AR) for interactive and engaging educational content.

**3. Online Courses and MOOCs:**

* Expanding the range of online courses and Massive Open Online Courses (MOOCs) to cater to a global audience.

**4. Student Engagement and Collaboration:**

* Enhancing communication and collaboration tools for students, faculty, and staff.
* Implementing social platforms or forums for students to connect, share resources, and collaborate on projects.

**5. Data Analytics and Learning Analytics:**

* Leveraging data analytics to track student performance, identify trends, and improve educational outcomes.
* Implementing learning analytics to assess the effectiveness of teaching methods and make data-driven decisions.

**6. Mobile Optimization:**

* Ensuring the website is mobile-friendly and responsive to accommodate the increasing use of smartphones and tablets.
* Developing mobile apps for easy access to essential features and information.

**7. E-Portfolios and Career Services:**

* Facilitating the creation of electronic portfolios for students to showcase their achievements and skills.
* Expanding career services through the website, including job postings, internship opportunities, and networking events.

**8. Cybersecurity Measures:**

* Strengthening cybersecurity protocols to protect sensitive student and institutional data.
* Implementing measures to ensure the privacy and security of online learning environments.

**9. Accessibility and Inclusivity:**

* Ensuring the website and online resources are accessible to individuals with disabilities.
* Implementing inclusive design principles to accommodate diverse learning needs.

**10. Blockchain for Credentials:**

* Exploring the use of blockchain technology for secure and transparent verification of academic credentials.

**11. Green Initiatives:**

* Embracing sustainable practices in website development and maintenance to align with environmental concerns.
* Adapting to these trends and embracing technological advancements can help colleges stay relevant, provide quality education, and enhance the overall student experience. It's crucial to regularly assess the needs of the institution and its stakeholders to stay ahead in the rapidly changing educational landscape.

## CONCLUSION

In conclusion, the college website serves as a critical interface between the institution and its stakeholders, playing a pivotal role in shaping the educational experience. As we look to the future, the trajectory of college websites is marked by transformative trends and technological advancements that promise to enhance accessibility, engagement, and the overall quality of education. Key considerations for the future of college websites include personalization, advanced technology integration, global outreach, communication and collaboration tools, data-driven decision-making, mobile optimization, cybersecurity, and sustainability initiatives.

A successful college website of the future will not only cater to the diverse needs of students, faculty, and staff but will also embrace innovation to stay ahead in the ever-evolving landscape of education. It will be a dynamic platform that fosters a sense of community, facilitates learning through cutting-edge technologies, and ensures that the institution remains adaptive and responsive to the changing needs of its users. As we navigate these trends, the college website stands as a gateway to a more inclusive, interactive, and sustainable educational ecosystem.

## REFERENCES

**Reference Books:**

Pro MERN Stack **– by Vasan Subramanian**

MERN Quick Start Guide **– by Eddy Wilson**

Learning MERN Stack **– by Upkar Lidder and Nikhil Sachdeva**

**Online References:**

<https://getbootstrap.com/>

<https://react.dev/learn/tutorial-tic-tac-toe>

<https://expressjs.com/en/guide/database-integration.html>

<https://nodejs.org/en/learn/getting-started/introduction-to-nodejs>

https://mongoosejs.com/docs/

https://stackoverflow.com/