**INSTITUTE OF ENGINEERING AND MANAGEMENT**

**SALT LAKE SECTOR V, KOLKATA**



# 

***LAB PROJECT REPORT OF DBMS***

**GUIDED BY: PROF. DEEP SHUBRA GUHA ROY**

**TITLE: Api Score Calculation System**

**Submitted By:**

Sanjana Kumari (12022002003170)

Deblina Maji (12022002016024)**DECLARATION:**

We, Sanjana Kumari and Deblina Maji, hereby declare that the project work titled API Calculator submitted to [Institute of Engineering and Management, Kolkata] is a record of original work carried out by us under the esteemed guidance and supervision of Prof. Deepshubra Guha Roy. This project has been undertaken as part of our academic curriculum and serves as an essential component in fulfilling the requirements of our course. We confirm that this project work is genuine, and no part of it has been submitted to any other institution or university for the award of any other degree, diploma, or certification. The data, findings, and analyses presented here are a product of our individual efforts and reflect our dedication to achieving a high standard of scholarly work in alignment with the objectives of the API Calculator project.

**BONAFIDE CERTIFICATE:**

This is to certify that the project report titled **API Calculator** is the bonafide work of **Sanjana Kumari** and **Deblina Maji**, who successfully carried out the project work under my supervision and guidance. This project represents their dedicated effort and understanding of full-stack application development, involving the integration of a **MySQL database** with **Spring Boot** for backend development and **React** for the frontend interface. Throughout the course of this project, they demonstrated a strong commitment to learning and implementing various technical skills necessary to develop a robust, user-friendly application that meets the standards and requirements of a real-world, data-driven solution. The **API Calculator** project aims to provide efficient and automated score calculation functionality, showcasing their understanding of database management, server-side logic, and user interface design. The project submission date is \_\_\_\_\_\_\_\_\_\_\_.

**ABSTRACT:**

The **API Calculator** is a comprehensive full-stack web application developed to streamline and automate the calculation of the **Internal Quality Assurance Cell (IQAC) Score** for faculty members. Built using **Spring Boot** for the backend and **React** for the frontend, this application offers a user-friendly interface that allows faculty members or administrators to input personal information and academic achievements systematically. The application evaluates this data to generate a personalized IQAC score, providing an objective measure of each faculty member’s contributions and performance.

In addition to data entry, the API Calculator features automated score calculation logic, which processes achievements in areas such as teaching, research, professional development, and community engagement. Once the scores are calculated, the application generates a **PDF report** that provides a detailed summary, including individual component scores and an overall IQAC score.

This project demonstrates a seamless integration of **RESTful APIs** for efficient data exchange, **database management** using MySQL for storing and retrieving information, and a cohesive **UI** experience through the React frontend. It reflects an understanding of robust backend logic, secure data handling, and a user-focused interface design. The API Calculator ultimately serves as a powerful tool to enhance efficiency in faculty performance evaluation, offering an effective and streamlined approach to IQAC scoring in academic institutions.

**ACKNOWLEDGMENT:**

We would like to express our heartfelt gratitude to **Prof. DeepShubra Guha Roy** for their invaluable guidance, continuous support, and encouragement throughout every stage of this project. Prof. Guha Roy’s expertise and constructive feedback have been instrumental in shaping the development of the **API Calculator**. Their insights into software development, full-stack application design, and effective problem-solving have profoundly enriched our understanding and capabilities, allowing us to address challenges and improve our project at every turn.

We are also deeply appreciative of the support provided by our peers and the department. Their encouragement and the resources made available to us have been crucial in overcoming obstacles and pushing us to reach our full potential. Additionally, we would like to acknowledge the collaborative spirit fostered within our department, which provided a conducive environment for learning and innovation, motivating us to approach this project with diligence and enthusiasm. This project would not have been possible without the combined efforts and encouragement of all those who have supported us, and we are grateful for the opportunity to learn and grow through this experience.

**INDEX:**

|  |  |  |
| --- | --- | --- |
| **SL NO.** | **TITLE** | **PAGE NO** |
| **1** | **Introduction** | **7-9** |
| **2** | **Literature Review** | **10-11** |
| **3** | **System Description** | **12-16** |
| **4** | **Theoretical Analysis** | **16-34** |
| **5** | **Methodology** | **35-36** |
| **6** | **Result and Discussion** | **36-42** |
| **7** | **Conclusion** | **43** |
| **8** | **References** | **43** |

**1.INTRODUCTION:**

**1.1 Background**

The API Calculator project was conceptualized as a digital solution to streamline and automate the calculation of the Internal Quality Assurance Cell (IQAC) Score for faculty members. IQAC scores are essential in higher education, as they provide a standardized evaluation of faculty contributions in areas such as teaching, research, and academic activities. Traditionally, calculating these scores has been a manual, time-consuming process, prone to error and inconsistency. By automating this process, the API Calculator aims to improve accuracy, save time, and provide a transparent scoring system that can benefit both faculty members and administration. This project is not only a technical endeavor but also an initiative to support institutions in maintaining and enhancing educational standards.

**1.2 Purpose and Scope**

The primary purpose of the API Calculator project is to simplify and optimize the process of tracking faculty achievements and to develop an efficient, automated system to calculate IQAC scores based on various performance metrics. These scores encompass areas such as academic output, professional development, and contributions to the community. This tool is designed to be adaptable to various institutional needs, providing flexibility in the configuration of scoring criteria while ensuring consistency in evaluation. The scope of this project includes developing a full-stack application that integrates both frontend and backend components, connecting a MySQL database for data storage, and generating downloadable reports for administrative review. This project is envisioned as a valuable asset for academic institutions, allowing them to efficiently manage faculty evaluations and enhance their internal quality assurance processes.

**1.3 Objectives**

The project objectives are outlined as follows:

* User-Friendly Web Interface: Design and develop an intuitive web application that enables faculty members or administrators to easily input faculty details, personal information, and achievements. The interface will guide users through each step, ensuring all necessary data points are captured accurately. This web interface is built with React and styled with Material-UI to enhance the user experience.
* IQAC Scoring Algorithm: Develop a robust scoring algorithm that calculates the IQAC score based on a faculty member's achievements in various categories, including teaching performance, research output, extension activities, and professional development efforts. This algorithm will be implemented on the backend using Spring Boot, ensuring reliable and consistent scoring based on predefined criteria. The algorithm’s flexibility allows administrators to adjust weights or add new achievement categories as needed.
* Automated Report Generation and Download: Enable automated PDF report generation upon submission of the faculty data and calculated IQAC score. Each report will provide a detailed breakdown of the score by category, along with a summary of the faculty member’s overall performance. This feature uses libraries such as jsPDF for PDF creation, allowing users to download the report directly through the application. The report will be accessible for both administrative review and faculty self-assessment, supporting transparency and motivation for continued professional growth.

**1.4 Project Overview**

The API Calculator is a full-stack web application that integrates frontend and backend technologies to provide a seamless user experience. The frontend, developed in React, enables users to input data through structured forms, making the process of data entry intuitive and efficient. For backend processing, Spring Boot serves as the foundation for the application’s business logic and database interactions. Using RESTful APIs, the frontend and backend communicate securely, facilitating real-time data processing and feedback.

The database layer is handled by MySQL, where faculty details and achievement data are stored and retrieved as needed. This relational database structure supports organized data management, allowing efficient querying and updating of faculty information. To make the application production-ready, deployment practices and optimizations are implemented, ensuring a smooth user experience across different platforms and environments.

Overall, the API Calculator project embodies a complete solution that automates faculty evaluation processes, reduces administrative burden, and supports academic institutions in maintaining high standards. The integration of a scoring algorithm, report generation, and a user-friendly interface makes it a valuable tool for effective faculty management and institutional growth.

Expertise and constructive feedback have been instrumental in shaping the development of the API Calculator. Their insights into software development, full-stack application design, and effective problem-solving have profoundly enriched our understanding and capabilities, allowing us to address challenges and improve our project at every turn.

We are also deeply appreciative of the support provided by our peers and the department. Their encouragement and the resources made available to us have been crucial in overcoming obstacles and pushing us to reach our full potential. Additionally, we would like to acknowledge the collaborative spirit fostered within our department, which provided a conducive environment for learning and innovation, motivating us to approach this project with diligence and enthusiasm. This project would not have been possible without the combined efforts and encouragement of all those who have supported us, and we are grateful for the opportunity to learn and grow through this experience

**2.LITERATURE REVIEW:**

**2.1. Introduction**

Evaluating faculty performance is an essential aspect of academic institutions' quality assurance and accountability efforts. The **Internal Quality Assurance Cell (IQAC)** scoring system, widely used in educational institutions, assesses faculty contributions across teaching, research, and community involvement. However, traditional methods for calculating IQAC scores often involve manual data entry, making the process labor-intensive and prone to errors. With the advancement of **full-stack web development** and **RESTful API** standards, it is now possible to streamline this process through digital applications, enabling real-time data processing and improved accuracy.

**2.2. Existing Systems**

Current systems for faculty performance evaluation range from basic spreadsheet-based methods to partially automated software solutions. Traditional manual approaches rely on filling forms or spreadsheets and then calculating scores manually. Some institutions employ simple, standalone applications for specific tasks like score calculation, but these systems often lack an integrated structure, making it challenging to gather, store, and analyze data efficiently. These applications also frequently suffer from limited scalability and are not designed to integrate with other administrative software, which can hinder data sharing and overall efficiency.

In the realm of web development, **full-stack applications** using **Spring Boot** for backend services and **React** for frontend interfaces have become popular for handling complex, multi-user applications. Spring Boot and **Java Persistence API (JPA)** facilitate effective database interactions, while RESTful APIs enable seamless communication between the frontend and backend, allowing data to be processed and presented to users in real-time.

**2.3. Problems with Existing Systems**

Manual or semi-automated approaches to IQAC score calculation face several limitations:

* **Inconsistent Data Handling**: Manual data entry is prone to errors and inconsistencies, which can impact the accuracy of IQAC scores.
* **Time-Consuming Process**: Manually collecting, validating, and calculating IQAC scores is labor-intensive and can lead to delays in report generation and performance evaluation.
* **Limited Integration**: Existing systems often lack the ability to connect with other databases or applications, making it difficult to track long-term faculty performance and generate comprehensive reports.
* **Lack of Transparency**: Manual or disjointed systems can obscure scoring criteria and methods, making it harder for faculty to understand and trust their evaluations.

**2.4. Solution: Full-Stack API Calculator with RESTful API Integration**

The **API Calculator** project addresses the limitations of existing systems by providing a streamlined, fully integrated solution for calculating IQAC scores. By leveraging a full-stack architecture with **Spring Boot** and **React**, this project offers a user-friendly interface for data entry, along with a robust backend for real-time score calculation and data storage. Key elements of the solution include:

* **RESTful API Communication**: Using RESTful APIs, the React frontend communicates seamlessly with the Spring Boot backend, ensuring that faculty data is efficiently processed and stored in a **MySQL** database.
* **Automated Scoring Algorithm**: The system calculates IQAC scores based on predefined criteria, reducing the possibility of errors and ensuring consistent and fair evaluations. The score calculation algorithm is customizable, allowing institutions to adjust criteria and weightings according to their unique requirements.
* **PDF Report Generation**: A PDF report feature enables automatic generation of feedback for each faculty member, providing a transparent breakdown of scores in various categories such as teaching, research, and professional development.
* **Scalability and Flexibility**: Built with modern technologies, the system is scalable and adaptable, capable of integrating with additional modules or expanding functionality as institutional needs evolve.

**3.SYSTEM DESCRIPTION:**

**3.1. Introduction**

The API Calculator is a web-based application designed to streamline and automate the process of calculating the Internal Quality Assurance Cell (IQAC) Score for faculty members in academic institutions. By digitizing the IQAC scoring process, this system ensures accuracy, transparency, and ease of use for both faculty members and administrators. The system consists of a React frontend for user interaction and a Spring Boot backend that handles business logic and communicates with a MySQL or PostgreSQL database. Users can enter their personal information and achievements, and the system then calculates and generates a downloadable PDF report with the IQAC score.

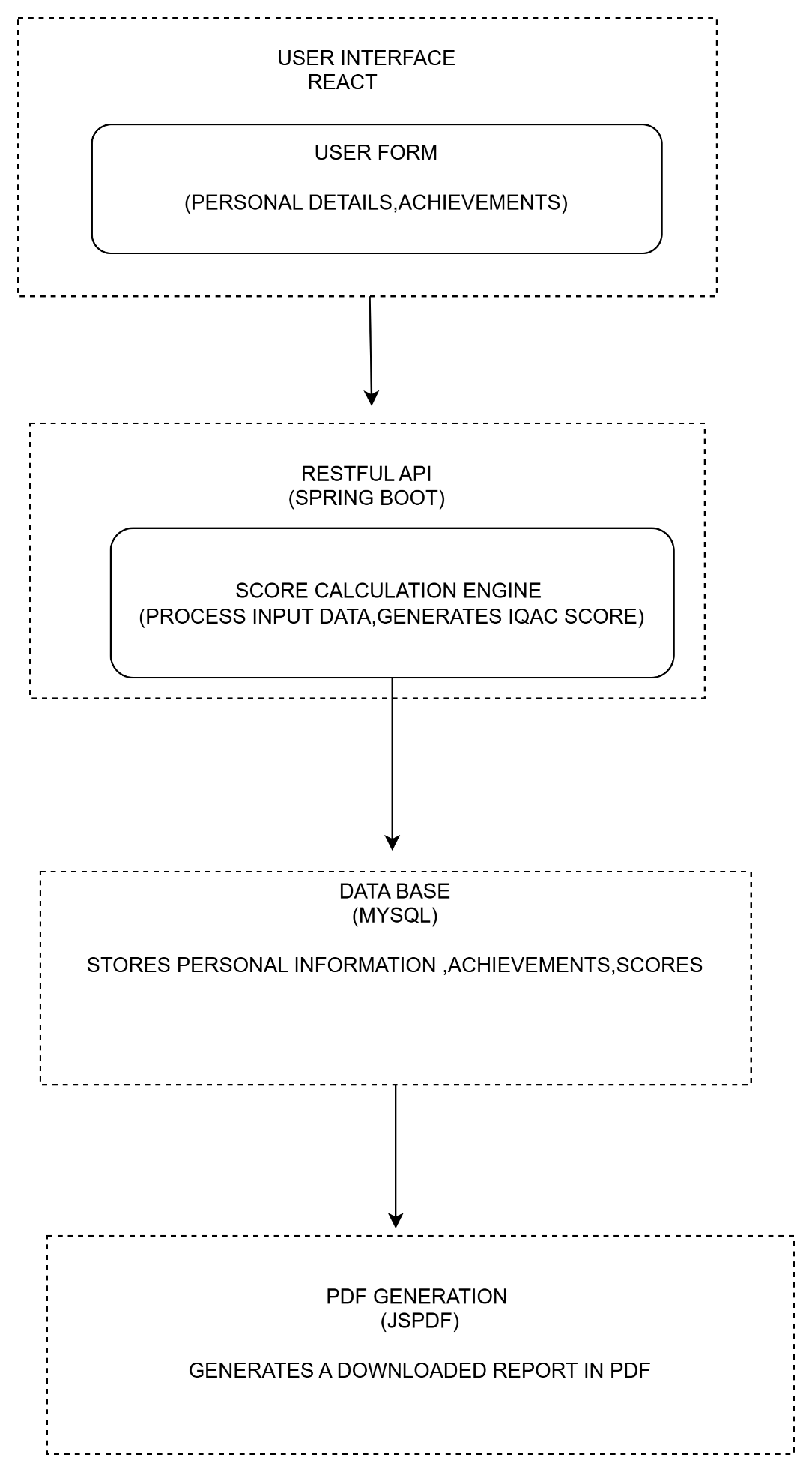
**3.2. SYSTEM ARCHITECTURE:**

Certainly! Here’s a guide on how to create architecture and flow diagrams for a project like the API Calculator. I’ll outline the steps for each and provide examples that you could use as templates for creating actual diagrams.

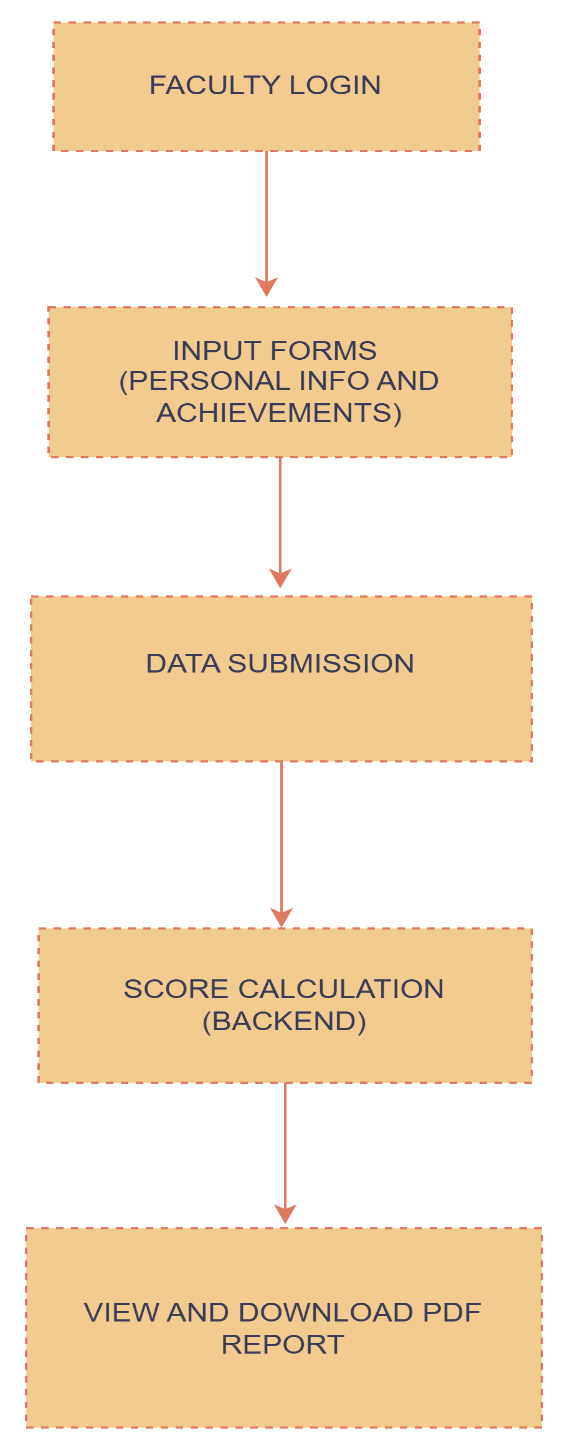
**3.2.1. System Architecture Diagram**

The System Architecture Diagram shows the components of the API Calculator system and how they interact at a high level. Here’s how to structure it:

Key Components:

* Frontend (React): User interface where faculty members and admins interact with the system.
* Backend (Spring Boot): Manages the business logic, handles API requests, and processes data.
* Database (MySQL): Stores faculty data, achievements, and calculated scores.
* PDF Generation Module: Creates downloadable reports in PDF format.
* 

**USER FLOW DIAGRAM:**

****

EXPLANATION:

 Login/Register

* Faculty members log into the application or register as new users.

 Input Forms

* After login, users fill out forms for personal details and achievements.

 Data Submission

* The data entered in the forms is submitted to the backend for processing.

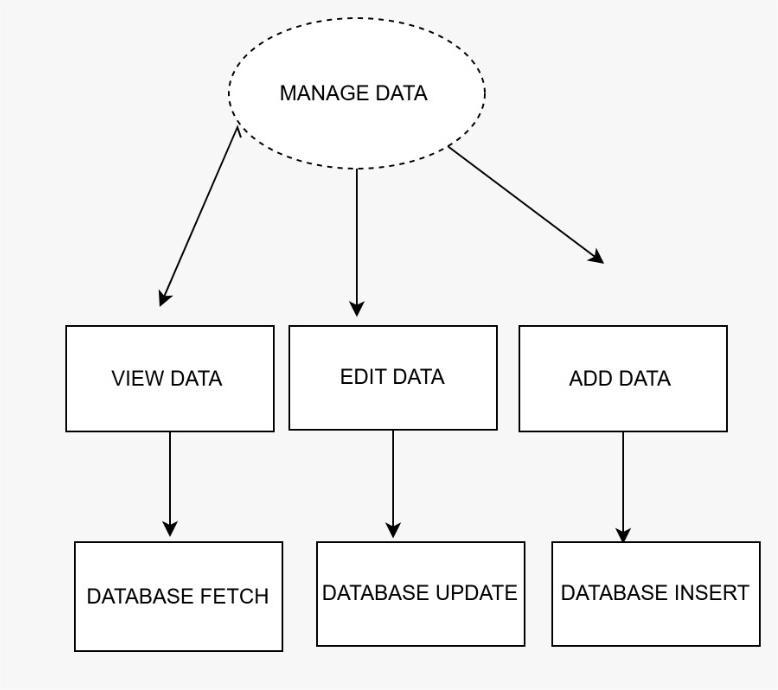
 Score Calculation

* The backend processes the data and calculates the IQAC score.

 View & Download PDF

* The user can view their calculated IQAC score and download a PDF report.

**DATA OPERATION:**

****

This image illustrates a data management flow that is relevant in designing an API calculator. In this context, "Manage Data" serves as the central controller for handling data-related operations. The three main functions are:

1. View Data: Fetches data from the database to display previous calculations or data inputs, corresponding to Database Fetch.
2. Edit Data: Updates existing entries, such as modifying saved calculation parameters or results, mapped to Database Update.
3. Add Data: Adds new calculation data or parameters to the database, corresponding to Database Insert.

An API calculator would use these functions to interact with the database for storing, retrieving, and updating calculation data, enhancing functionality and data persistence**.**

**4.THEORETICAL ANALYSIS:**

**4.1** INTRODUCTION TO TOOLS USED IN PROJECTS:

**Introduction to React**

React is a widely used JavaScript library for building user interfaces, especially for single-page applications where data changes dynamically. It was developed by Facebook and allows developers to create large web applications that can update and render efficiently in response to data changes. React uses a component-based architecture, where the user interface is split into reusable components.

FOR EXAMPLE:

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

import axiosInstance from '../config/axiosInstance';

import { useNavigate } from 'react-router-dom';

import {

Grid,

Card,

CardContent,

Typography,

Button,

TextField,

Container,

Box,

Dialog,

DialogActions,

DialogContent,

DialogTitle,

} from '@mui/material';

const Faculty = () => {

const [facultyList, setFacultyList] = useState([]);

const [searchTerm, setSearchTerm] = useState(''); // State for search term

const [isAddingFaculty, setIsAddingFaculty] = useState(false);

const [newFaculty, setNewFaculty] = useState({

name: '',

email: '',

department: '',

address: '',

contactNumber: '',

designation: '',

joiningDate: '',

});

const [openDeleteModal, setOpenDeleteModal] = useState(false);

const [facultyIdToDelete, setFacultyIdToDelete] = useState(null);

const navigate = useNavigate();

useEffect(() => {

fetchFaculty();

}, []);

const fetchFaculty = async () => {

try {

const response = await axiosInstance.get('/api/faculties');

setFacultyList(response.data);

} catch (error) {

console.error(error);

}

};

const handleDeleteClick = (id) => {

setFacultyIdToDelete(id);

setOpenDeleteModal(true);

};

const handleDelete = async () => {

try {

await axiosInstance.delete(`/api/faculties/${facultyIdToDelete}`);

setOpenDeleteModal(false);

fetchFaculty();

} catch (error) {

console.error(error);

}

};

const handleCloseDeleteModal = () => {

setOpenDeleteModal(false);

};

const handleViewDetails = (id) => {

navigate(`/faculties/${id}`);

};

const handleIQACView = (id) => {

navigate(`/faculties/iqac/${id}`);

};

const handleInputChange = (e) => {

const { name, value } = e.target;

setNewFaculty((prevState) => ({

...prevState,

[name]: value,

}));

};

const handleAddFaculty = async (e) => {

e.preventDefault();

try {

await axiosInstance.post('/api/faculties', newFaculty);

setIsAddingFaculty(false);

fetchFaculty();

setNewFaculty({

name: '',

email: '',

department: '',

address: '',

contactNumber: '',

designation: '',

joiningDate: '',

});

} catch (error) {

console.error(error);

}

};

// Handle search input change

const handleSearchChange = (e) => {

setSearchTerm(e.target.value);

};

// Handle resetting the list by clearing search term

const handleShowAllFaculty = () => {

setSearchTerm(''); // Clear the search term to show full list

};

// Filter faculty list based on search term

const filteredFacultyList = searchTerm

? facultyList.filter((faculty) =>

faculty.name.toLowerCase().includes(searchTerm.toLowerCase())

)

: facultyList; // Show full list if searchTerm is empty

return (

<Container>

<Box mt={4} textAlign="center" pb={3}>

<Typography variant="h3" color="primary" sx={{ fontWeight: 'bold', fontFamily: 'Poppins, sans-serif' }}>

Welcome to the API Calculator System

</Typography>

</Box>

<Box mt={4}>

<Typography variant="h4" gutterBottom align="center">

{isAddingFaculty ? 'Add Faculty' : 'Faculty List'}

</Typography>

<Box mb={4} display="flex" justifyContent="center" gap={2}>

<Button

variant="contained"

color="primary"

onClick={() => { setIsAddingFaculty(false); handleShowAllFaculty(); }}

sx={{ fontFamily: 'Roboto, sans-serif' }}

>

Show Faculty List

</Button>

<Button

variant="outlined"

color="secondary"

onClick={() => setIsAddingFaculty(true)}

sx={{ fontFamily: 'Roboto, sans-serif' }}

>

Add Faculty

</Button>

</Box>

{/\* Render filtered faculty list \*/}

{!isAddingFaculty ? (

<>

<Box mt={4} mb={4} display="flex" justifyContent="center" gap={2}>

<TextField

label="Search Faculty by Name"

variant="outlined"

value={searchTerm}

onChange={handleSearchChange}

fullWidth

sx={{ maxWidth: 300 }}

/>

</Box>

{filteredFacultyList.length === 0 ? (

<Typography variant="h6" color="error" align="center">

No matching faculty found.

</Typography>

) : (

<Grid container spacing={3}>

{filteredFacultyList.map((faculty) => (

<Grid item xs={12} sm={6} md={4} key={faculty.id}>

<Card sx={{ boxShadow: 3 }}>

<CardContent>

<Typography variant="h6" sx={{ fontWeight: 'bold', fontFamily: 'Poppins, sans-serif' }}>

{faculty.name}

</Typography>

<Typography color="textSecondary">

<strong>Email:</strong> {faculty.email}

</Typography>

<Typography color="textSecondary">

<strong>Department:</strong> {faculty.department}

</Typography>

<Typography color="textSecondary">

<strong>Contact:</strong> {faculty.contactNumber}

</Typography>

<Typography color="textSecondary">

<strong>Designation:</strong> {faculty.designation}

</Typography>

<Typography color="textSecondary">

<strong>Joining Date:</strong> {faculty.joiningDate}

</Typography>

<Box mt={2} display="flex" gap={2}>

<Button

variant="contained"

color="info"

onClick={() => handleViewDetails(faculty.id)}

>

Details

</Button>

<Button

variant="outlined"

color="primary"

onClick={() => handleIQACView(faculty.id)}

>

Get IQAC

</Button>

<Button

variant="outlined"

color="error"

onClick={() => handleDeleteClick(faculty.id)}

>

Delete

</Button>

</Box>

</CardContent>

</Card>

</Grid>

))}

</Grid>

)}

</>

) : (

<form onSubmit={handleAddFaculty}>

<Box mb={3}>

<TextField

label="Name"

variant="outlined"

fullWidth

margin="normal"

name="name"

value={newFaculty.name}

onChange={handleInputChange}

required

/>

<TextField

label="Email"

variant="outlined"

fullWidth

margin="normal"

name="email"

value={newFaculty.email}

onChange={handleInputChange}

required

/>

<TextField

label="Department"

variant="outlined"

fullWidth

margin="normal"

name="department"

value={newFaculty.department}

onChange={handleInputChange}

required

/>

<TextField

label="Address"

variant="outlined"

fullWidth

margin="normal"

name="address"

value={newFaculty.address}

onChange={handleInputChange}

required

/>

<TextField

label="Contact Number"

variant="outlined"

fullWidth

margin="normal"

name="contactNumber"

value={newFaculty.contactNumber}

onChange={handleInputChange}

required

/>

<TextField

label="Designation"

variant="outlined"

fullWidth

margin="normal"

name="designation"

value={newFaculty.designation}

onChange={handleInputChange}

required

/>

<TextField

label="Joining Date"

type="date"

variant="outlined"

fullWidth

margin="normal"

name="joiningDate"

value={newFaculty.joiningDate}

onChange={handleInputChange}

required

InputLabelProps={{

shrink: true,

}}

/>

</Box>

<Button type="submit" variant="contained" color="success" sx={{ fontFamily: 'Roboto, sans-serif' }}>

Add Faculty

</Button>

</form>

)}

</Box>

{/\* Delete Confirmation Modal \*/}

<Dialog open={openDeleteModal} onClose={handleCloseDeleteModal}>

<DialogTitle>Are you sure you want to delete this faculty?</DialogTitle>

<DialogContent>

<Typography>Once deleted, this action cannot be undone.</Typography>

</DialogContent>

<DialogActions>

<Button onClick={handleCloseDeleteModal} color="primary">

No

</Button>

<Button onClick={handleDelete} color="error">

Yes, Delete

</Button>

</DialogActions>

</Dialog>

</Container>

);

};

export default Faculty;

**ABOUT SPRING BOOT:**

Spring Boot is an open-source Java-based framework used to create stand-alone, production-grade Spring-based applications. It simplifies the configuration and setup of Spring applications by providing default configurations and making the development process more efficient. It’s particularly known for its ease of integration with databases and REST APIs.

EXAMPLE:

<?xml version="1.0" encoding="UTF-8"?>

<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<parent>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-parent</artifactId>

<version>3.3.4</version>

<relativePath/> <!-- lookup parent from repository -->

</parent>

<groupId>com.iqac</groupId>

<artifactId>iqac-calculator</artifactId>

<version>0.0.1-SNAPSHOT</version>

<name>iqac-calculator</name>

<description>api score calculator project</description>

<url/>

<licenses>

<license/>

</licenses>

<developers>

<developer/>

</developers>

<scm>

<connection/>

<developerConnection/>

<tag/>

<url/>

</scm>

<properties>

<java.version>17</java.version>

</properties>

<dependencies>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-data-jpa</artifactId>

</dependency>

<!-- <dependency>-->

<!-- <groupId>org.springframework.boot</groupId>-->

<!-- <artifactId>spring-boot-starter-security</artifactId>-->

<!-- </dependency>-->

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-web</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-devtools</artifactId>

<scope>runtime</scope>

<optional>true</optional>

</dependency>

<dependency>

<groupId>com.mysql</groupId>

<artifactId>mysql-connector-j</artifactId>

<scope>runtime</scope>

</dependency>

<dependency>

<groupId>org.projectlombok</groupId>

<artifactId>lombok</artifactId>

<optional>true</optional>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-test</artifactId>

<scope>test</scope>

</dependency>

<!-- <dependency>-->

<!-- <groupId>org.springframework.boot</groupId>-->

<!-- <artifactId>spring-boot-starter-security</artifactId>-->

<!-- </dependency>-->

<!-- &lt;!&ndash; BCrypt Password Encoder &ndash;&gt;-->

<!-- <dependency>-->

<!-- <groupId>org.springframework.security</groupId>-->

<!-- <artifactId>spring-security-crypto</artifactId>-->

<!-- </dependency>-->

<!-- <dependency>-->

<!-- <groupId>io.jsonwebtoken</groupId>-->

<!-- <artifactId>jjwt</artifactId>-->

<!-- <version>0.9.1</version>-->

<!-- </dependency>-->

<dependency>

<groupId>org.apache.pdfbox</groupId>

<artifactId>pdfbox</artifactId>

<version>2.0.29</version> <!-- Or latest version available -->

</dependency>

<dependency>

<groupId>org.springdoc</groupId>

<artifactId>springdoc-openapi-starter-webmvc-ui</artifactId>

<version>2.6.0</version>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-maven-plugin</artifactId>

<configuration>

<excludes>

<exclude>

<groupId>org.projectlombok</groupId>

<artifactId>lombok</artifactId>

</exclude>

</excludes>

</configuration>

</plugin>

</plugins>

</build>

</project>

**5.METHODOLOGY:**

1. Prerequisites

* Installation of Node.js, Java JDK, Maven, and MySQL.

2. System Setup

* Clone the repository, update the database credentials, and set up the backend and frontend using Spring Boot and React.

3. Workflow

* Step 1: Input faculty data through the frontend.
* Step 2: Submit data to the backend, where it is stored and processed.
* Step 3: Calculate the IQAC score based on the entered achievements.
* Step 4: Generate a PDF report with the IQAC score and feedback summary.
* Step 5: Download the report for record-keeping.

4. API Communication

* Frontend (React): Uses Axios to send HTTP requests to the backend.
* Backend (Spring Boot): Receives, processes, and stores data, then calculates scores.

**6.RESULT AND DISCUSSION:**

**6.1Code:**

Css part:

.App {

text-align: center;

}

body {

margin: 0;

font-family: 'Lora', serif;

-webkit-font-smoothing: antialiased;

-moz-osx-font-smoothing: grayscale;

background-color: black;

}

.App-logo {

height: 40vmin;

pointer-events: none;

}

@media (prefers-reduced-motion: no-preference) {

.App-logo {

animation: App-logo-spin infinite 20s linear;

}

}

.App-header {

background-color: #282c34;

min-height: 100vh;

display: flex;

flex-direction: column;

align-items: center;

justify-content: center;

font-size: calc(10px + 2vmin);

color: white;

}

.App-link {

color: #61dafb;

}

@keyframes App-logo-spin {

from {

transform: rotate(0deg);

}

to {

transform: rotate(360deg);

}

}

**Faculty .js**

import React, { useState } from 'react';

import axios from 'axios';

const FacultyForm = ({ facultyToEdit, onFormSubmit }) => {

const [faculty, setFaculty] = useState(facultyToEdit || { name: '', email: '' });

const handleChange = (e) => {

setFaculty({ ...faculty, [e.target.name]: e.target.value });

};

const handleSubmit = async (e) => {

e.preventDefault();

if (faculty.id) {

await axios.put(`/api/faculty/${faculty.id}`, faculty);

} else {

await axios.post('/api/faculty', faculty);

}

onFormSubmit();

};

return (

<div className="container mt-4">

<h2>{faculty.id ? 'Edit Faculty' : 'Add Faculty'}</h2>

<form onSubmit={handleSubmit}>

<div className="mb-3">

<label className="form-label">Name</label>

<input

type="text"

name="name"

className="form-control"

value={faculty.name}

onChange={handleChange}

required

/>

</div>

<div className="mb-3">

<label className="form-label">Email</label>

<input

type="email"

name="email"

className="form-control"

value={faculty.email}

onChange={handleChange}

required

/>

</div>

<button type="submit" className="btn btn-success">Save</button>

</form>

</div>

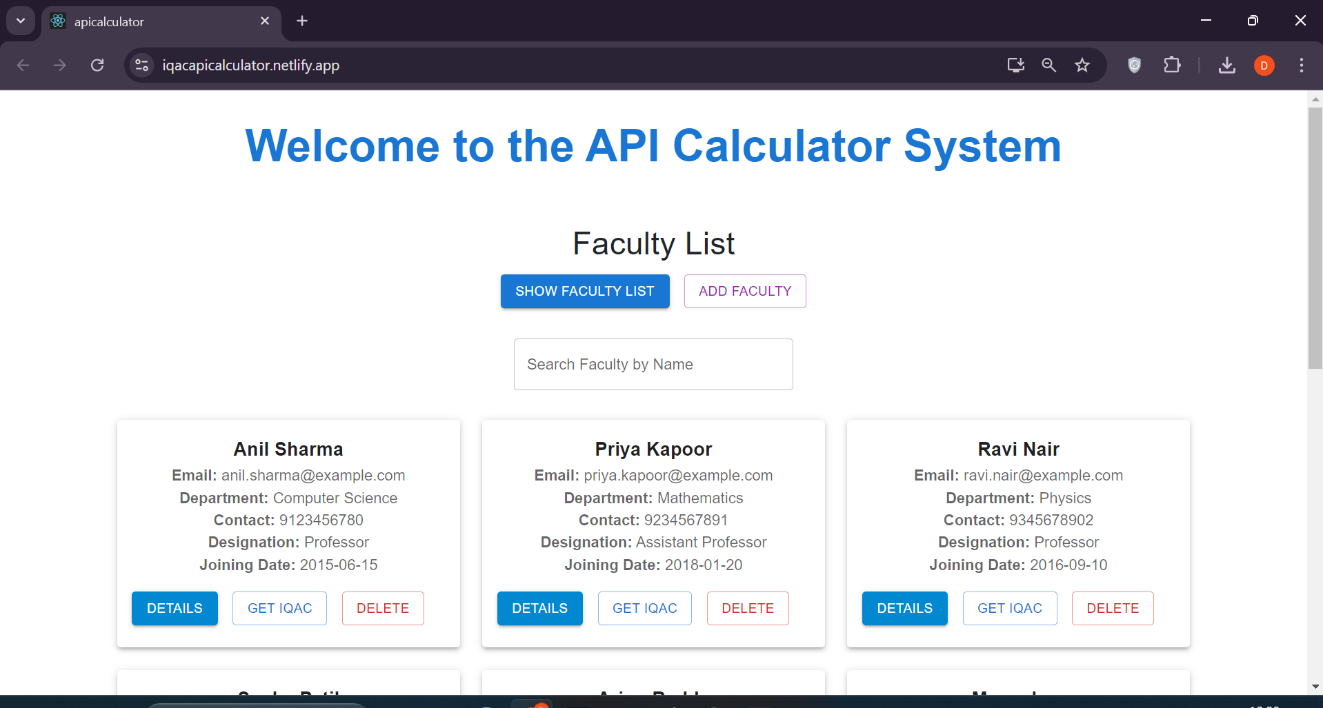
);

};

export default FacultyForm;

**6.2 Desription of findings:**

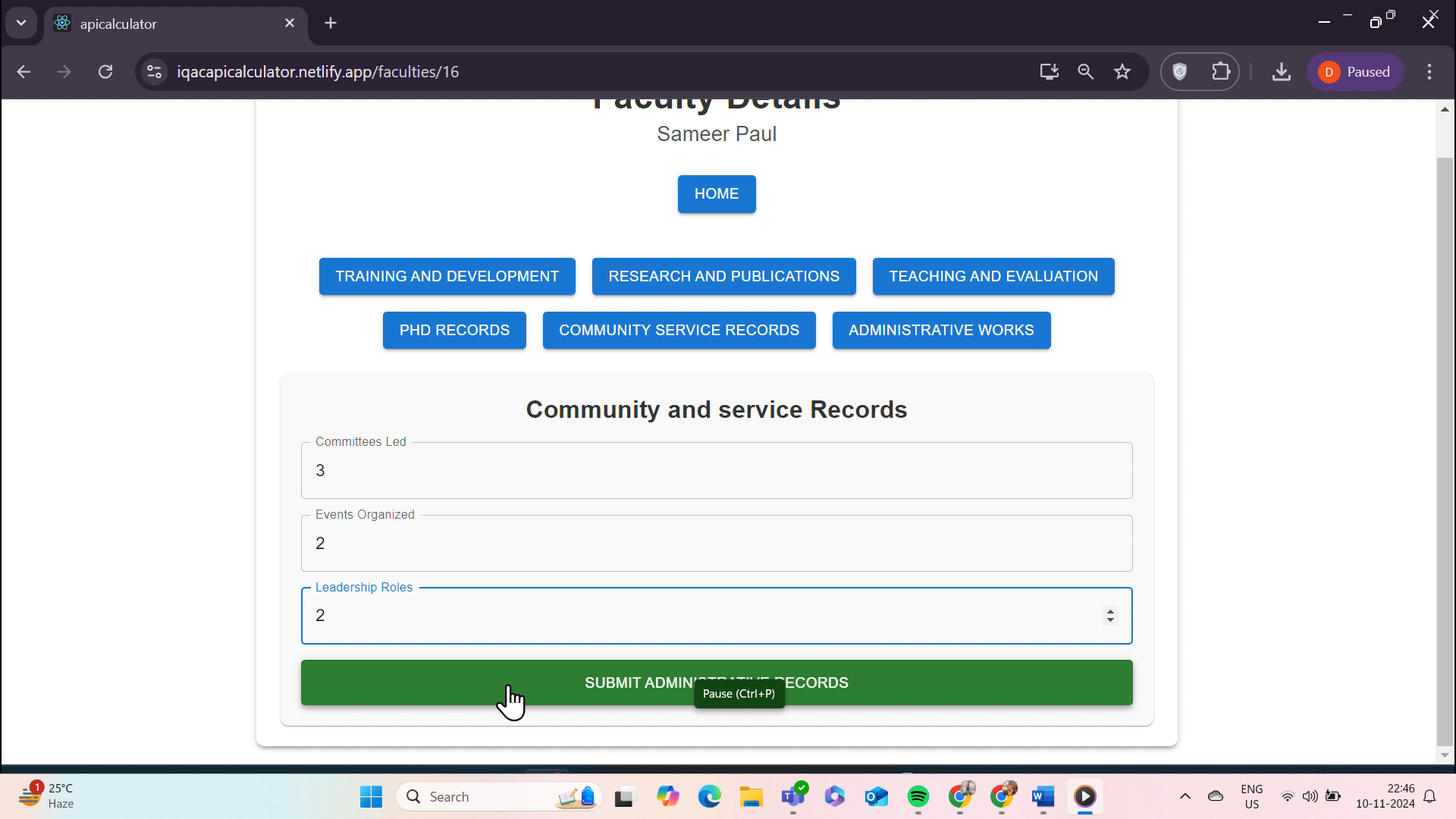
**Login page:**



1. Title and Buttons: At the top, a welcome message is followed by buttons for "Show Faculty List" and "Add Faculty."
2. Search Bar: A search box labeled "Search Faculty by Name" allows users to find specific faculty members quickly.
3. Faculty Cards: Each faculty member has a card displaying their name, email, department, contact, designation, and joining date.
4. Actions on Each Card:
   * Details: View more information.
   * Get IQAC: Calculate and show the IQAC score.
   * Delete: Remove the faculty record from the system.

The clean layout and interactive buttons, created using React, make the system easy to navigate and use.

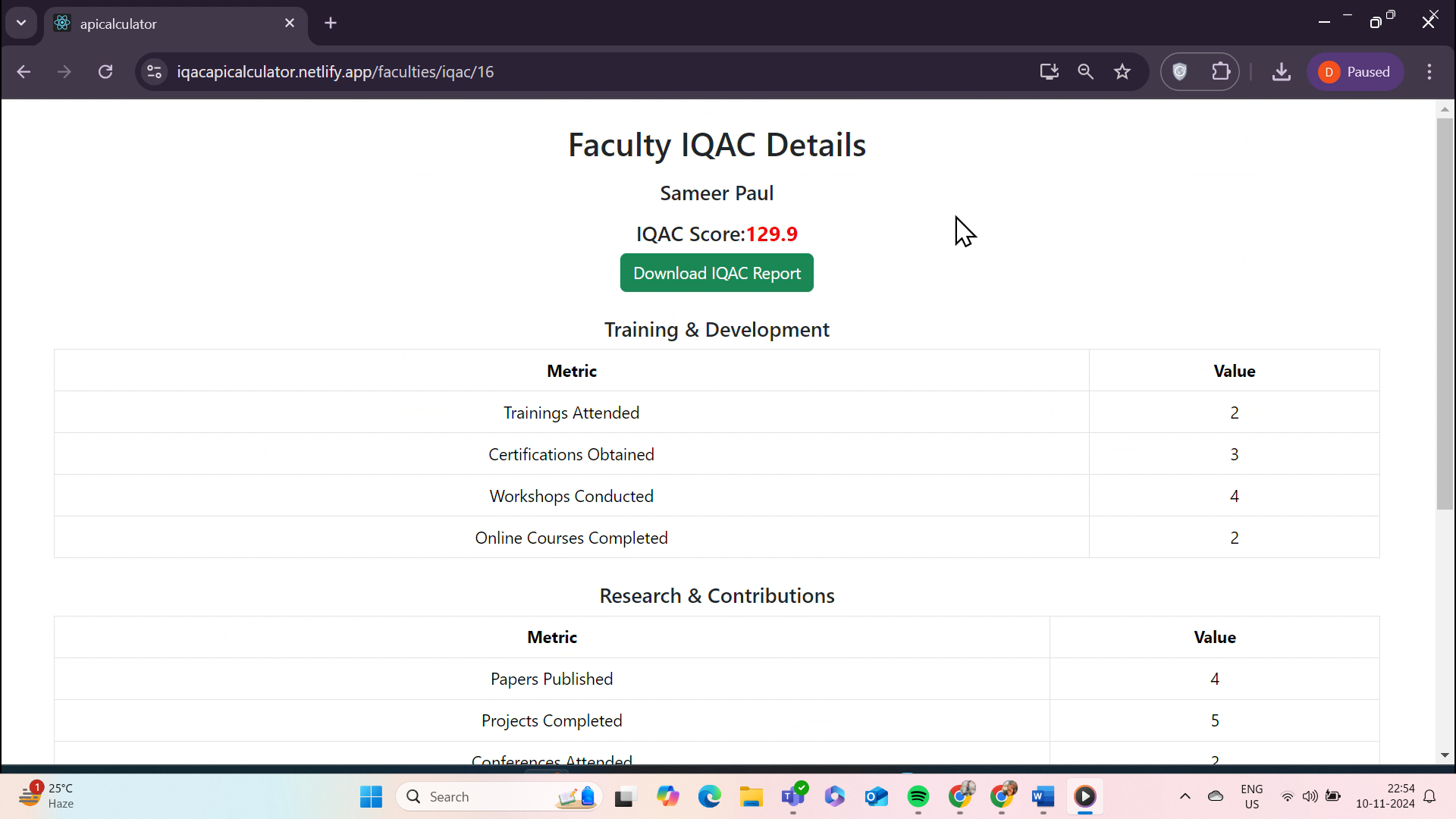
Detail filling page:



In this page of the **API Calculator System**, users can enter details in various categories, such as **Community and Service Records** (e.g., *Committees Led*, *Events Organized*, *Leadership Roles*). After inputting data, the user clicks **Submit Administrative Records**, which sends the data to the backend.

The backend, using **Spring Boot**, processes this information and stores it in a MySQL database. This data is then used for calculations like the IQAC score, which can later be retrieved, viewed, or downloaded by the user for further reference. This streamlined process enables efficient management and tracking of faculty contributions.

**Output generates page:**

****

This page displays the **Faculty IQAC Details**, showing the calculated **IQAC Score** based on data entered in various categories, such as *Training & Development* and *Research & Contributions*. Each category lists metrics (e.g., *Trainings Attended*, *Certifications Obtained*) along with their values.

The **backend** calculates the IQAC score by applying specific weights to each metric and summing the results. This score reflects the faculty member's overall contributions and is shown at the top of the page. Users can also download the report in PDF format by clicking **Download IQAC Report**. This report summarizes the faculty member’s activities and the calculated IQAC score for easy reference.

**Results:**

* The application successfully collects and processes faculty details, calculates the IQAC score based on the defined criteria, and generates a report in PDF format.
* The user-friendly interface provides easy input and management of faculty data.
* The backend's efficient scoring system allows accurate evaluation of faculty achievements

**Discussion:**

 User Experience: The integration of Material-UI in the frontend enables a seamless and professional user experience, making the forms easy to navigate and submit.

 Backend Performance: The Spring Boot backend efficiently handles data processing and calculations, but future improvements could include optimizing the database for faster retrieval times.

 Limitations: The scoring system could benefit from more customizable settings, allowing for finer adjustments to scoring criteria to better suit different departments.

 Future Enhancements: Introducing an option for faculty to view their historical scores.

**7.CONCLUSION:**

In this lab project, we developed an IQAC score calculator for faculty, incorporating data entry, score computation, and report generation functionalities. The system provides a structured interface for recording faculty achievements across multiple categories, such as *Training & Development*, *Research & Contributions*, and *Community Service*. Using Spring Boot for backend development and MySQL for data storage, we calculated IQAC scores based on weighted metrics and presented them through an intuitive user interface. This project demonstrates the practical application of web development and database integration in building a performance evaluation tool, contributing to efficient and organized faculty assessment.

**8.REFERENCES:**

 Spring Boot Documentation – Spring Boot Framework Documentation. Available at: <https://docs.spring.io/spring-boot/>

 MySQL Documentation - MySQL Database Management System. Available at: <https://dev.mysql.com/doc/>.

 Bootstrap Documentation - CSS Framework for Frontend Development. Available at: <https://getbootstrap.com/docs/>.

 Material UI Documentation - Material UI In React Documentation. Available at: <https://mui.com/material-ui/>

 IQAC Guidelines - Guidelines for Internal Quality Assurance Cell (IQAC) by NAAC. Available at: <https://naac.gov.in>.

 W3Schools - Web Development Tutorials. Available at: <https://www.w3schools.com>.