RAJEEV GANDHI MEMORIAL COLLEGE OF ENGINEERING &TECHNOLOGY (AUTONOMOUS)

CSE (DATA SCIENCE)

Accredited by NAAC of UGC, New Delhi with 'A+' Grade

Accredited by NBA, Approved by AICTE, Affiliated to
J.N.T. University Anantapur, Nandyal-518501

Team members: CSE TEAM 10

1. Poreddy Lakshmi Vardhan Reddy
2. Thalapaneni Harshitha
3. Doma Bharathi
4. Deepika Reddy

RGM COLLEGE OF ENGINEERING AND TECHNOLOGY (AUTONOMOUS) NANDYAL

RGM CODEQUEST 24 - 18th, 19th October 2024
Hackathon Problem Statement

Statement:

Best-Performing Student Recognition System

Description:

Design and Develop a Best-Performing Student Recognition System for a college to identify and appreciate the top 3 students from each admitted batch. The system should consider various factors, including but not limited to academic performance, consistency over semesters, excellence in core engineering courses, participation in local and national-level hackathons, paper presentations, and contributions such as assisting course teachers. The platform must feature a user-friendly interface and a well-structured database to manage student records, achievements, and rankings. Additionally, it should implement machine learning techniques to weigh these factors dynamically and rank students based on their overall contributions and performance. The system will aim to provide fair and data-driven results to recognize the best-performing students each year. The challenge lies in ensuring transparency, accuracy, and scalability as the number of students and criteria grows over time.

Development Tools:

If you plan to develop the **Best-Performing Student Recognition System** using **HTML, CSS, and JavaScript**, here's a list of development tools and libraries specifically tailored for building such a system with these technologies.

1. Frontend Development Tools

Since you'll be working primarily with **HTML, CSS, and JavaScript**, it's essential to use frameworks and libraries that enhance functionality, ease of development, and scalability.

HTML/CSS/JavaScript Libraries

- **HTML5**: For structuring the content of your web application.
- **CSS3**: To style your application and make it visually appealing.
- **JavaScript (ES6+)**: For implementing dynamic behaviors, such as handling user input and rendering dynamic content (e.g., student rankings).

CSS Frameworks

- **Bootstrap**: A responsive framework that provides ready-to-use components (e.g., buttons, forms, grids).

2. Backend Integration (if needed)

If you only want to use **HTML, CSS, and JavaScript** on the client-side, you can connect to a backend via **AJAX** or **Fetch API** to retrieve or submit data. You can use:.

- **Django or Flask (Python backend)** if you want to use Python for backend logic.

  You can make **AJAX** calls to interact with your database to dynamically retrieve and send student data, achievements, and rankings.

3. Machine Learning (via APIs)

Since machine learning can be challenging to implement directly with JavaScript, you can:

1. Develop the **machine learning logic** using **Python** and **Pandas** (or another ML framework).
2. **Expose the ML model via an API**: Use a Python backend (Django/Flask) to serve the ML logic as an API.
3. Use **JavaScript (Axios or Fetch API)** to interact with the backend, sending student data and receiving ranking results.

4. Database Integration

You will need a way to store student records, achievements, and ranking factors. A few options include:

- **dbSQLite3**: For relational database storage. You can interact with these databases via backend APIs and send/receive data using JavaScript.
-

5. Development Tools

- **Visual Studio Code**: A powerful and popular code editor for writing HTML, CSS, and JavaScript code.
- **Live Server**: A VS Code extension to run your HTML/CSS/JS code in real time in the browser.
- **Prettier**: A code formatter for consistent and clean HTML/CSS/JS code.
- **Git** + **GitHub**: For version control and collaboration.

6. Deployment Options

You can deploy your system to a cloud service:

Interacting with the App

- **Add Students**: Use the form on the homepage to add students manually.
- **Rank Students**: The app will fetch data from the database, normalize it, and rank students based on various factors (e.g., CGPA, hackathon participation, paper presentations)

- **Extend Functionality**: You can add more features, such as different ranking methods, admin roles, or additional APIs to fetch specific student data.

- **HTML Template** (`templates/index.html`): Provides a simple web interface for adding students and ranking them.
- **CSS and JavaScript** (`static/`): CSS for styling and JavaScript for fetching and displaying data from the backend (Flask).

Running the Application

- **Start the Flask Application**: Run app.py, which starts the Flask development server.
- **Access the Web App**: Open http://127.0.0.1:5000/ in a browser to interact with the system.