LeeterBoard

Full-Stack University Contest Leaderboard

Team Name: Runtime Terror | GitHub Team: LeeterBoard
Members: Cole Price, Dhruv Patel, Caleb Damron, Brody Curry

Introduction

Overview

LeeterBoard is a full-stack web application that collects LeetCode contest data and organizes participants based on their universities. The platform allows users to filter rankings by school, view standings within their institution, and compare themselves to students at other universities. Our goal was to create a tool that encourages friendly competition, helps students track their progress, and makes contests more engaging.

We conceived the idea upon realizing that LeetCode lacks features for comparing performance among classmates or friends. Its global rankings can feel impersonal and less relatable. By introducing school-based filters and a streamlined leaderboard system, we aimed to provide students with a clearer sense of their standing within their academic communities.

Initially, our scope was straightforward—display a list of usernames, scores, and ranks filtered by university. However, as development progressed, we identified opportunities to enhance the user experience. We incorporated line graphs to depict score history over time, implemented badge and tier systems to recognize high performers, and ensured the site was responsive across various screen sizes.

Our focus encompassed both functionality and presentation. Subtle enhancements, such as hover effects and mobile-friendly layouts, contributed to a polished and complete user interface. Each team member played a vital role, contributing to backend scraping, database design, frontend styling, and performance optimization.

This project enabled us to apply the principles of planning, teamwork, and software development acquired throughout the course. Ultimately, we delivered a functional product that addresses a genuine need, and we take pride in the outcome.

Customer Value (Changes from Proposal: No Changes).

Customer Need

Our primary customers are university students who participate in LeetCode contests and desire a means to compare their rankings with peers, both within their own institution and across others. Currently, LeetCode does not offer such a feature, making it challenging for users to gauge their performance relative to fellow students.

Proposed Solution

LeeterBoard bridges this gap by providing a clean, user-friendly leaderboard that organizes contest results by university. This feature motivates students to engage more actively in contests and offers a more relevant and engaging way to track their progress.

Measures of Success

The success of LeeterBoard will be evaluated based on key metrics, including:

- User Adoption: The number of students utilizing the platform to check rankings.
- User Friendly: A simple, intuitive UI that functions seamlessly across devices.
- Impact on Participation – Increased involvement in contests driven by the motivation of university rankings.

- User Feedback: Positive responses from students who find the tool beneficial for maintaining competitiveness and improving performance.

Technology

Architecture

LeeterBoard is structured as a full-stack web platform. The backend is developed using Python with the Flask framework, managing routing, API endpoints, and server-side logic. Data is stored in a MongoDB database, which maintains user profiles, contest results, and university mappings.

A custom Python script periodically fetches contest data from LeetCode. This scraper extracts relevant information and updates the MongoDB database to ensure the leaderboard remains current. It employs a workaround to collect contest data not officially exposed by LeetCode's API.

The frontend is constructed using HTML, and CSS, with Jinja2 templates facilitating server-side rendering. Chart.js is integrated to generate dynamic graphs, allowing users to visualize performance trends over time. The user interface is designed to be responsive, ensuring compatibility across various devices and screen sizes, such as computers, tablets, and smartphones.

Additional background scripts handle recurring tasks such as weekly ranking refreshes, initial data loading for new users, and processing user-university mappings. These scripts ensure data consistency and reduce backend load during frequent access.

Tools

- Languages & Frameworks: Python, Flask, HTML, and CSS
- Database: MongoDB
- Graphing Library: Chart.js
- Data Scraper: Custom Python script using API workaround
- Version Control & Collaboration: GitHub

Team

Roles

The project ran pretty smoothly overall. Everyone pitched in early on to get the initial site layout and routing working. Once that was done, we focused on implementing features in phases — starting with the leaderboard and university filters, then adding things like graphs, badges, and extra styling for mobile support. We met regularly (usually weekly) to review progress, assign new tasks, and troubleshoot bugs. Each team member owned parts of the UI or backend logic and contributed changes via GitHub.

Project Management

Schedule

The project advanced efficiently, with prompt development of the site's core functionalities. Team members focused on various UI elements to ensure a comprehensive and user-friendly design, incorporating quality-of-life features to enhance accessibility and usability.

All project aspects have been completed, with refinements to existing systems enhancing smoothness, user satisfaction, and professionalism. Interactive elements feature subtle hover effects, and the university-specific user pages have been optimized for clarity.

Reflection

This project was successful from start to completion. Our planning was effective, and development progressed steadily. While some team members initially faced challenges with local development setups, we worked together to resolve them quickly and maintained consistent communication throughout the project.

As development continued, we found that working with Flask and Chart.js gave us the flexibility we needed to build out key features like graphing performance data and rendering dynamic content. Although we encountered minor hurdles, we adapted well to new technologies, while leveraging the tools we were already familiar with.

Our GitHub issue tracking system and regular team check-ins were essential to staying aligned and meeting deadlines. These practices kept us organized and helped ensure steady progress across all phases of development.

Ultimately, we delivered a comprehensive web application that works across devices, features a polished and responsive user interface, and fulfills a clear purpose. We are proud of what we built and believe it offers real value to university-level LeetCode participants.