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 gathers LeetCode contest data and categorizes participants based on their universities. The platform allows users to filter rankings by institution, track their performance relative to peers, and compare standings across universities.

Our motivation stems from the idea that seeing rankings at a university level could enhance the competitive spirit of coding contests. By providing visibility on how students perform within and across institutions, we aim to promote a sense of motivation and engagement in competitive programming.

LeeterBoard is a novel idea, as we are not aware of any existing applications that offer this type of university-based ranking system. It fills a gap by introducing an extra layer of motivation and competition for LeetCode contest participants.

Cole - I have built a full-stack iOS application and worked on different websites, with my main exposure areas being in that of C++, React, and Swift.
Dhruv - I have worked on a few iOS development projects with SwiftUI. Additionally, I also have some exposure with web structure and have some experience with projects involving data plotting and analysis.
Caleb - I have developed scripts for post-processing and data manipulation as well as simple card games. Experience with C, C++, and python.
Brody - I have worked on a VR fishing game, a C++ poker game in terminal, and other minor projects. Experience with C, C++, and Java.

Customer Value

Customer Need

Our primary customers are university students who participate in LeetCode contests and are looking for a structured way to compare their rankings against peers within their institution and across other universities. Currently, LeetCode does not provide a built-in feature for university-specific rankings or peer comparisons, leaving students without an efficient way to benchmark their performance within their academic community.

Proposed Solution

LeeterBoard addresses this gap by providing a comprehensive leaderboard system that categorizes contest participants by university. This feature enables students to assess their performance relative to their peers, fostering a competitive environment that encourages continuous improvement.

Measures of Success

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

- User Adoption: The number of students actively using the platform to track rankings.
- User Friendly: Users are able to navigate through the website from multiple different devices, having seamless experiences.
- Impact on Participation: Increased involvement in LeetCode contests as students strive to improve their rankings.
- User Feedback: Positive responses from students who find the leaderboard beneficial for motivation and performance tracking.

Technology

System

Our software will pull contest data from the LeetCode Leaderboard and it will filter the data by University. We may also add features that allow you to filter it by state/country/etc.

The main components of our system include:

- The data scraper (backend) that fetches the contest data, extracting relevant participant information.
- The database will store and organize the contest results, user profiles, and university mappings.
- UI/Display (frontend) will display the leaderboard, search filters, user profile, and rankings in an interactive UI.
- Deployment and Hosting will include the use of either AWS or Heroku to run the servers.

The LeeterBoard system has a React.js frontend, a python backend, a MongoDB database, and a Python data scraper. The frontend displays rankings and filters data by university, sending requests to the backend, which processes queries and retrieves contest results from the database. The database stores user rankings, contest results, and university mappings. A Python-based data scraper periodically fetches LeetCode contest data by hitting the API workaround, ensuring up-to-date rankings. This setup allows students to track and compare their competitive programming performance efficiently.

A minimal LeeterBoard system would have a static React.js or HTML page displaying university rankings from a manually updated CSV or JSON file. A Python scraper would periodically fetch LeetCode contest data, storing it locally instead of using a database. The frontend would load this data for display without backend processing. Hosting could be done on GitHub Pages or Vercel, ensuring a simple, low-maintenance solution that still provides university-based rankings for users.

Enhancements could include live data updates for real-time rankings, user authentication for personalized tracking, and interactive filtering for easier comparisons. Historical performance tracking with visual trends would help users monitor progress, while notifications could alert them to ranking changes and contests. A mobile-friendly design, customizable leaderboards, and social features like friend leaderboards and community challenges would further boost engagement.

Testing will include unit tests for individual components, integration tests for data flow, and end-to-end tests for user interactions. Load testing will assess performance, and security tests will check for vulnerabilities. User feedback and real-world testing will ensure functionality and usability.

Tools

We will be using Python to query the LeetCode public API and store the retrieved contest data in a MongoDB database. The frontend will be built using React.js, providing an interactive and responsive user

experience for filtering rankings and comparing standings. For deployment and hosting, we will utilize AWS or Heroku, ensuring scalable and reliable infrastructure for both the backend data processing and database management. These will be the tools we leverage throughout development. This technology stack enables LeeterBoard to efficiently track and display university-based rankings.

Team

Skills

Cole has built a full stack iOS App and a couple of front-end web applications, but not a full-stack web application functioning in the way we are aiming for here. Dhruv also has some experience with iOS app development, but is also unfamiliar with development of a full-stack web application. Brody has not worked in iOS or web application development but has had a variety of experiences in unique projects and programs that could be helpful. Caleb has limited experience in iOS and web application development as well but also has valuable experience through diverse projects.

Most of the tools are new to most of the team, such as MongoDB, AWS, and Python. This aligns with our goal, as we are also trying to learn a lot from building this web app.

Roles

The roles of the team members are relatively similar. We will all work together towards the same goal, which is to produce an effective and efficient application. We all plan on communicating well and collaborating with one another throughout the project. We will all be serving as developers to develop the best product we can. We will adjust roles throughout, but I feel that jumping into this with flexibility will prove to serve us well.

Project Management

Schedule

Week of 2/10/2025	Complete project proposal for submission.
Week of 2/17/2025	Start to explore tools that will be used to get more comfortable with them.
Week of 2/24/2025	Begin development of the first iteration of the web-app.
Week of 3/3/2025	Continue the development of the first iteration.
Week of 3/10/2025	Complete first version of web-app.
Week of 3/17/2025	Spring Break.
Week of 3/24/2025	Begin testing and perfecting any flaws that may exist from the first iteration.
Week of 3/31/2025	Continue testing and perfecting.
Week of 4/7/2025	Complete second and finalized version of the web-app, and begin on report.
Week of 4/14/2025	Complete project report and present finished project.

Constraints

In terms of legality, our methods will not be crossing any serious lines there. In terms of ethics, some may view web scraping as unethical, but we are doing nothing malicious with the data. We will simply be

accessing public data and displaying it in a user friendly way. Our web application will not be using user data for purposes other than catering to the user in a beneficial way.

Resources

The LeetCode API itself is not public, but there are many workarounds for that. There are tons of endpoints out there that we could hit to grab this data. With this being said, we will definitely be able to gather what we need for our intended purposes.

Descoping

In the worst case, if the full functionality cannot be implemented, we will still find ways to make our web application beneficial to the average LeetCode user. I don't foresee this being an issue, but if it is, I know that we will still be able to craft something that will be very beneficial, even if it doesn't have the full capability that we hope for. The goal is to make something very useful that, in the worst possible case, will still be somewhat useful, and pretty cool at that.