ContestX

Group 24

Group Members:

Hariom Joshi (2023CA40)

Shruti Sahu (2023CA93)

Vedprakash Maliarya (2023CA110)

1. Introduction

Purpose

ContestX is an online competitive programming platform designed to provide a fair and efficient environment for coding contests. The system focuses on two key goals:

1. **Prevent Cheating:** While completely eliminating cheating may not be possible, ContestX takes steps to make it more challenging. These include limiting tab switches (only three per session), disabling code pasting, and allowing users to save only their manually copied code snippets.
2. **Reduce Submission Queue Times:** By leveraging Apache Kafka as the messaging backbone, submissions are processed asynchronously, preventing long queue delays even during high-traffic periods.

Scope

ContestX enables user registration, participation in coding contests, real-time submission processing, and profile management. The platform is built using Next.js with TypeScript for the web application and integrates Apache Kafka for handling high-throughput events, particularly to reduce submission delays.

Overview

- **Tech Stack:** Next.js with TypeScript for frontend and backend, Apache Kafka for event streaming.
- **Key Features:**
    - Fast and efficient submission processing with Kafka for asynchronous event handling.
    - Anti-cheating mechanisms such as tab-switch restrictions, paste prevention, and controlled saved code snippets.
    - User profiles showing ratings and problem-solving history.

2. System Description

2.1 Main Features

User Management:

- User registration, login, and profile management.
- Profiles display user ratings and problem-solving records.

Contest Management:

- Organizers can create contests, set schedules, and upload problems.
- Participants join contests and submit their solutions.

Submission Processing:

- Submissions are sent to a Kafka topic, allowing asynchronous evaluation and reducing long queue delays.
- This will also involve challenges in setting the leaderboard (basically all the problems that comes with multi threading)

Anti-Cheating Measures:

- **Tab Switch Limit:** Only three tab switches are allowed during a contest session.
- **Paste Disabled:** The code editor blocks code pasting to reduce the risk of copying from external sources.
- **User Snippets:** Only manually copied snippets are saved for user reference.

Real-Time Leaderboard:

- The leaderboard updates dynamically based on evaluated submissions.

2.2 Problem Solved by Kafka

Kafka decouples submission handling from code evaluation, enabling fast and parallel processing. This design minimizes wait times during high-traffic periods, ensuring a seamless contest experience.

3. Functional Requirements

User Registration & Profiles:

- Users register using a unique email or username and log in to access their profiles.
- Profiles display user ratings and a list of solved problems.

Contest Creation & Participation:

- Organizers create contests with schedules and rules.
- Participants join contests and submit their solutions via the web interface.

Submission Handling:

- Submissions trigger events published to a Kafka topic (e.g., `submission_events`).
- An evaluation service consumes these events to process submissions in real-time.

Anti-Cheating Enforcement:

- The code editor monitors tab switches and enforces a maximum of three per contest session.
- Code paste functionality is disabled.
- Only manually copied user snippets are permitted for saving.

Real-Time Leaderboard:

- The leaderboard updates dynamically as submissions are evaluated and results are processed.

4. Non-Functional Requirements

Performance:

- Submissions and leaderboard updates should be processed with minimal latency (ideally within one second).

Scalability:

- The system must support high numbers of concurrent users and submissions. Kafka's distributed architecture enables horizontal scaling and high throughput.

Security:

- User data and contest activities are protected through secure authentication and encrypted communications (e.g., HTTPS, TLS for Kafka).

Maintainability:

- Next.js with TypeScript promotes a modular and maintainable codebase, simplifying updates and scalability.

5. System Architecture

Frontend:

- Built using Next.js with TypeScript for a responsive and user-friendly interface for contest participation and profile management.

Backend Services:

- Microservices handle user authentication, contest data, and submission evaluation.
- The submission service publishes events to Kafka, which are consumed asynchronously by an evaluation service.

Messaging Backbone:

- Apache Kafka processes submission events, minimizing delays during peak times and facilitating smooth communication between services.

Database:

- Postgres with Prisma ORM

6. Deployment & Maintenance

Deployment:

- The platform will be deployed on a scalable cloud environment with container orchestration for microservices and a managed Kafka cluster for messaging.

Conclusion

ContestX is a scalable and efficient contest platform designed to minimize submission queue times using Apache Kafka and to eradicate (or at least delay) cheating through simple but effective client-side restrictions. With a modern web application built using Next.js and TypeScript and real-time event processing powered by Kafka, ContestX delivers a smooth and fair competitive programming experience.