BAB I
PENDAHULUAN

1.1. Latar Belakang

Keterampilan pemrograman telah menjadi salah satu kompetensi inti yang sangat krusial di era digital saat ini, ditandai dengan permintaan sumber daya manusia dengan keterampilan pemrograman yang terus meningkat di berbagai industri [1]. Merespons kebutuhan akan talenta yang tidak hanya sekadar mampu menulis kode tetapi juga memiliki logika penyelesaian masalah yang efisien, metode pengembangan dan evaluasi keahlian ini telah berevolusi melampaui kurikulum akademis tradisional. Dalam konteks inilah kompetisi pemrograman atau Competitive Programming hadir dan berkembang menjadi sebuah "olahraga pikiran" (*mind sport*) yang diakui secara global. Di ajang ini, peserta ditantang untuk menyelesaikan serangkaian masalah algoritmik di bawah batasan waktu dan memori yang ketat, yang kini menjadi tolak ukur krusial bagi pengembangan talenta digital. Daya tarik kompetisi ini semakin menguat seiring dengan pengakuannya sebagai salah satu jalur paling efektif bagi talenta muda untuk mendapatkan eksposur internasional dan peluang karier di perusahaan teknologi global, yang pada gilirannya mendorong pertumbuhan komunitas pemrograman ini secara eksponensial [2].

Validitas dari pertumbuhan partisipasi ini diperkuat oleh data empiris yang mengindikasikan eskalasi global yang signifikan. Pada tingkat pendidikan tinggi, International Collegiate Programming Contest (ICPC) melaporkan partisipasi sebanyak 52.709 mahasiswa yang merepresentasikan 3.233 universitas dari 110 negara pada tahun 2018, merefleksikan skala adopsi yang masif di lingkungan akademik [2]. Tren pertumbuhan serupa teramati pada tingkat pendidikan menengah, di mana International Olympiad in Informatics (IOI) mencatat peningkatan partisipasi dari 278 peserta pada tahun 2000 menjadi 362 peserta yang mewakili 91 negara pada penyelenggaraan terkini [3]. Di ranah platform daring, Codeforces mencatat pertumbuhan metrik aktivitas pengguna berkisar antara 15%

hingga 45%, sementara kompetisi nasional seperti USACO Open menunjukkan peningkatan kuantitas partisipan hingga empat kali lipat dalam satu dekade terakhir [3]. Akumulasi data ini mengonfirmasi bahwa ekosistem kompetisi pemrograman telah bertransformasi menjadi entitas global yang menuntut dukungan infrastruktur teknologi yang scalable dan reliable.

Dalam ekosistem yang semakin masif ini, keberadaan platform yang handal memegang peranan yang sangat vital karena tidak hanya berfungsi sebagai penilai kode atau Online Judge semata melainkan juga bertindak sebagai Contest Management System (CMS) yang kompleks [4]. Berbeda dengan Online Judge konvensional yang umumnya digunakan untuk keperluan latihan mandiri ataupun sebagai repositori masalah, sebuah CMS dirancang secara khusus untuk menangani lingkungan kompetisi formal yang dikarakteristikkan oleh beban kerja komputasi yang intensif akibat lonjakan konkurensi submisi serta tuntutan pemrosesan waktu nyata [4]. Sebuah CMS yang mumpuni harus memiliki kapabilitas untuk menangani berbagai aspek krusial mulai dari pendaftaran peserta dan manajemen soal hingga distribusi masalah serta penilaian otomatis secara real-time yang diiringi dengan penyajian papan peringkat atau scoreboard yang dinamis [4]. Kompleksitas ini menuntut infrastruktur yang tidak hanya kuat namun juga responsif guna menjamin kelancaran jalannya kompetisi bagi seluruh peserta.

Tantangan teknis utama dalam infrastruktur coding contest adalah skalabilitas dan reliabilitas sistem di bawah beban kerja yang ekstrem. Mayoritas platform penilaian kode yang ada saat ini masih mengadopsi arsitektur monolitik, di mana antarmuka pengguna, logika bisnis, dan mekanisme penilaian terintegrasi erat, mengakibatkan kohesi kode yang tinggi serta pemeliharaan yang sulit [5]. Pendekatan monolitik ini menciptakan keterikatan yang kuat antara alur penilaian dan sistem file, yang menghambat kemampuan sistem untuk melakukan scaling secara presisi sesuai beban kerja nyata [6]. Akibatnya, arsitektur ini rentan terhadap *bottleneck* saat terjadi lonjakan submisi, di mana pemrosesan antrean secara FIFO pada sistem tradisional sering kali gagal menangani beban intensif secara responsif [6]. Lebih lanjut, *load testing* membuktikan secara empiris bahwa arsitektur monolitik memiliki keterbatasan signifikan dalam waktu pemrosesan hasil -

dibandingkan dengan arsitektur terdistribusi yang mampu melakukan horizontal scaling, yang terbukti jauh lebih efisien dan stabil dalam menangani konkurensi tinggi [7].

Selain permasalahan skalabilitas, infrastruktur ini juga dihadapkan pada tantangan operasional krusial yang berkaitan dengan keamanan serta keadilan. Sistem diwajibkan untuk mengeksekusi kode sewenang-wenang (*arbitrary code*) dari pengguna yang berpotensi mengandung instruksi berbahaya sehingga dapat mengancam integritas server utama [1]. Lebih jauh lagi, urgensi untuk menjamin konsistensi hasil penilaian menjadi sangat vital dalam sebuah kompetisi karena kode peserta harus dieksekusi dalam lingkungan yang benar-benar terisolasi guna memastikan pengukuran waktu eksekusi yang adil tanpa dipengaruhi oleh fluktuasi beban kerja proses lain pada mesin penilai [8]. Mekanisme isolasi ini tidak hanya menjamin keadilan tetapi juga berfungsi sebagai lapisan pertahanan yang mencegah kode berbahaya merusak sistem *host*, karena kegagalan dalam aspek ini dapat memicu variasi waktu penilaian yang tidak adil seperti *Time Limit Exceeded* yang tidak konsisten serta membuka celah kerentanan keamanan yang serius [8]. Di samping isu keamanan dan isolasi tersebut, arsitektur yang memanfaatkan kontainerisasi juga memunculkan tantangan operasional tambahan berupa beban cold-start, yang menjadi hambatan performa utama (bottleneck) sistem akibat latensi saat harus menginisiasi kontainer penilai baru untuk mengeksekusi sebuah submisi [9].

Sebagai respons terhadap berbagai limitasi tersebut, penelitian ini mengusulkan pembangunan platform coding contest menggunakan arsitektur microservices. Pola ini mendekomposisi fungsionalitas sistem menjadi layanan-layanan kecil yang independen, seperti layanan submisi dan layanan penilaian [10]. Pendekatan ini memungkinkan setiap layanan dikembangkan dan di-*deploy* secara terpisah, sehingga jika terjadi lonjakan beban pada antrean penilaian, kapasitas sumber daya dapat ditingkatkan secara spesifik pada komponen tersebut tanpa mematikan fitur lainnya [6].

Guna mendukung sifat terdistribusi ini, sistem akan menerapkan *Event-Driven Architecture* dengan menggunakan RabbitMQ sebagai *message broker* yang

menghubungkan *Submission Service* dan *Judging Service* secara asinkron. Penggunaan *message broker* ini memfasilitasi pemisahan proses penerimaan dan evaluasi kode secara efektif, sehingga antarmuka pengguna dipastikan tetap responsif bahkan di bawah beban kerja intensif. Selanjutnya, guna mengeliminasi latensi *cold-start*, sistem mempertahankan sebuah *warm worker pool* yang selalu dalam keadaan siap menjalankan evaluasi alih-alih menginisialisasi kontainer baru untuk setiap submisi. Untuk menjamin keamanan dan presisi penilaian, sistem mengimplementasikan mekanisme isolasi berlapis dengan membungkus perangkat lunak Isolate di dalam kontainer Docker. Pemanfaatan Isolate ini memberdayakan Linux kernel *namespaces* dan *cgroups* demi menjamin isolasi sumber daya di tingkat kernel, sehingga menghasilkan pengukuran waktu eksekusi yang deterministik dan konsisten bagi kelima bahasa pemrograman yang didukung (C++, Java, Python, JavaScript, dan Golang). Dari sisi manajemen data, sistem memisahkan beban penyimpanan relasional melalui PostgreSQL dan pengelolaan *object storage* untuk *test case* serta kode sumber menggunakan MinIO. Implementasi sistem ini bertujuan untuk menghadirkan platform kompetisi yang tidak hanya tangguh secara teknis dalam menghadapi lonjakan trafik, tetapi juga mampu menciptakan ekosistem kompetisi yang adil, transparan, dan reliabel, sehingga dapat menyelenggarakan kontes pemrograman yang berkualitas dan profesional.

1.2. Rumusan Masalah

Penyelenggaraan kompetisi pemrograman berskala besar menuntut infrastruktur sistem yang tidak hanya akurat dalam menilai kode, tetapi juga memiliki tingkat ketersediaan dan keadilan yang tinggi di tengah beban kerja yang ekstrem. Namun, penggunaan arsitektur monolitik konvensional pada platform yang ada saat ini sering kali memicu hambatan teknis berupa *blocking process* saat terjadi lonjakan submisi, serta ketidakstabilan performa akibat kurangnya isolasi sumber daya yang berdampak pada inkonsistensi waktu penilaian dan kerentanan keamanan. Berdasarkan permasalahan tersebut, penelitian ini berfokus pada perancangan platform coding contest yang menerapkan arsitektur microservices

dan *message broker* untuk menangani konkurensi tinggi, serta mengimplementasikan Online Judge terdistribusi berbasis kontainerisasi. Pendekatan ini dipilih untuk menjamin skalabilitas, konsistensi eksekusi kode, dan keamanan infrastruktur terukur yang mampu menghadirkan pengalaman kompetisi yang responsif, adil, dan *reliable* bagi seluruh peserta maupun penyelenggara.

1.3. Batasan Penelitian

Penelitian ini difokuskan pada perancangan dan pembangunan backend serta infrastruktur sistem untuk platform coding contest dengan arsitektur microservices, tanpa membahas aspek bisnis atau pemasaran platform. Adapun batasan masalah secara spesifik adalah sebagai berikut:

1) Arsitektur & Komunikasi: Sistem backend dikembangkan menggunakan kerangka kerja Spring Boot dengan implementasi pola Microservices. Komunikasi antar-layanan untuk proses submisi dan penilaian dilakukan secara asinkron menggunakan RabbitMQ sebagai message broker.
2) Mekanisme Penilaian: Eksekusi kode peserta dilakukan dalam lingkungan terisolasi (*sandbox*) menggunakan perangkat lunak Isolate yang dibungkus di dalam kontainer Docker. Hal ini untuk menjamin keamanan sistem dari kode berbahaya serta memastikan determinisme waktu eksekusi.
3) Manajemen Data: Penyimpanan data relasional (pengguna, soal, scoreboard) menggunakan basis data PostgreSQL, sedangkan penyimpanan objek (object storage) untuk berkas test case (input/output) dan kode sumber solusi menggunakan MinIO.
4) Bahasa Pemrograman: Sistem penilaian dibatasi untuk mendukung kompilasi dan eksekusi pada lima bahasa pemrograman utama yang umum digunakan, yaitu: C++, Java, Python, Javascript, dan Golang.
5) Cakupan Fitur: Fitur fungsional sistem dibatasi pada: manajemen kontes dan soal, pendaftaran peserta, mekanisme submisi kode, penilaian otomatis secara *real-time*, mekanisme pengumuman dan klarifikasi, dan tampilan papan peringkat secara *live*.
6) Antarmuka Pengguna: Sisi frontend dikembangkan menggunakan React.js

berbasis web (browser), dan tidak mencakup pengembangan aplikasi mobile (Android/iOS).

7) Pengujian: Pengujian sistem difokuskan pada pengujian fungsional *black-box testing* dan pengujian performa *load testing* untuk mengukur throughput dan latency sistem di bawah beban konkurensi tinggi.

1.4. Pertanyaan Penelitian

1) Bagaimana merancang dan membangun platform coding contest berbasis arsitektur microservices yang mengintegrasikan message broker dengan mekanisme *warm worker pool* dan lingkungan eksekusi terisolasi menggunakan mekanisme *sandboxing* guna mendukung skalabilitas sistem serta interaksi *real-time*?
2) Bagaimana validitas fungsionalitas dan kinerja sistem yang diusulkan jika dibandingkan dengan sistem online judge konvensional pada skenario konkurensi tinggi, ditinjau dari metrik throughput, latensi, dan stabilitas sumber daya?

1.5. Tujuan Penelitian

1) Merancang dan membangun platform *coding contest* berbasis arsitektur *microservices* yang mengintegrasikan komunikasi asinkron melalui RabbitMQ dengan mekanisme *warm worker pool* guna mengeliminasi *cold-start latency* dan lingkungan eksekusi terisolasi guna memisahkan beban kerja submisi dan penilaian secara independen untuk mendukung skalabilitas sistem.
2) Memvalidasi fungsionalitas sistem melalui pengujian fungsional *black box* serta mengevaluasi kinerjanya dibandingkan dengan sistem *online judge* konvensional pada skenario konkurensi tinggi untuk parameter throughput, latensi, dan tingkat kesealahan

1.6. Manfaat Penelitian

1) Manfaat Teoretis

Penelitian ini memberikan kontribusi pada pengembangan ilmu Rekayasa Perangkat Lunak dan Sistem Terdistribusi dengan menyediakan bukti empiris mengenai efektivitas arsitektur *microservices* yang dipadukan dengan mekanisme *warm worker pool* berbasis RabbitMQ dalam memitigasi *bottleneck* pemrosesan dan latensi *cold-start* yang lazim ditemui pada arsitektur monolitik di lingkungan Online Judge. Selain itu, penelitian ini memvalidasi keandalan metode isolasi berbasis kontainer sebagai pendekatan yang menjamin konsisten hasil untuk menjamin akurasi pengukuran waktu eksekusi, sehingga memperkaya literatur akademik mengenai standar keadilan (fairness) dalam perancangan sistem kompetisi pemrograman modern.

2) Manfaat Praktis

Penelitian ini menghadirkan solusi infrastruktur yang andal bagi institusi penyelenggara untuk menggelar kompetisi berskala besar dengan risiko kegagalan sistem yang minimal dan efisiensi sumber daya yang optimal melalui arsitektur *microservices*. Bagi peserta kompetisi, implementasi sistem ini mengeliminasi hambatan teknis berupa latensi submisi dan ketidakpastian penilaian, sehingga menjamin pengalaman kompetisi yang responsif dan adil. Arsitektur yang dihasilkan berfungsi sebagai *blueprint* teknis bagi pengembang perangkat lunak dalam merancang sistem backend yang membutuhkan penanganan konkurensi tinggi melalui integrasi teknologi message broker dan orkestrasi kontainer.