Laporan Tugas Besar

Aplikasi Pelaporan Warga

disusun untuk memenuhi tugas
Mata Kuliah IF4031 Arsitektur Aplikasi Terdistribusi

oleh

Muhammad Atpur Rafif	13522086@std.stei.itb.ac.id
Andhika Tantyo Anugrah	13522094@std.stei.itb.ac.id
Justin Lawrance	18222006@std.stei.itb.ac.id

PROGRAM STUDI TEKNIK INFORMATIKA
SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG

2025

Tabel Kontribusi

NIM	Nama	Kontribusi
13522086	Muhammad Atpur Rafif	Kode sumber *proof-of-concept* (PoC), Penjelasan hasil implementasi
13522094	Andhika Tantyo Anugrah	Deskripsi umum sistem, Daftar teknologi + alasan pemilihannya, Pemilihan fungsionalitas dan kualitas yang diimplementasi + alasan, Dokumentasi PoC, bug fix kode PoC
18222006	Justin Lawrance	Diagram arsitektur sistem + penjelasannya, Daftar teknologi + alasan, Asumsi

Deskripsi Umum Sistem

Sistem yang dibangun adalah platform pelaporan warga suatu kota untuk melaporkan permasalahan-permasalahan di lingkungannya. Sistem ini juga akan memudahkan lembaga terkait untuk fokus pada masalah yang dapat mereka tangani dan menyelesaikannya. Sistem ini didesain dengan arsitektur microservices yang berkomunikasi secara sinkron menggunakan API gaya REST dan secara asinkron dengan message queue untuk notifikasi dan pemrosesan analytics.

Arsitektur terdistribusi microservices dipilih untuk mempermudah pemisahan penerima laporan dan pengelola laporan. Hal ini dilakukan karena setiap lembaga pasti memiliki proses bisnis dan kebijakan yang berbeda-beda. Metode komunikasi sinkron dipilih untuk mendukung interaksi real-time serta response time yang memuaskan.

Layanan-layanan ini didesain stateless agar dapat:

- Menangani volume pengguna yang banyak
- Memudahkan ketika di-deploy menjadi beberapa instance layanan dan di-scale secara horizontal (jadi scaleable) menggunakan orchestrator seperti Kubernetes
- Meningkatkan fault tolerance

Karena desain stateless tadi, sistem menjadi lebih mudah untuk dijadikan lebih scalable dengan mendeploy layanan-layanan tersebut menggunakan node orchestrator seperti Kubernetes. Kubernetes digunakan bersamaan dengan ingress (load balancer), horizontal pod autoscaler (HPA), dan metrics server untuk melakukan scaling (up maupun down). Karena horizontal pod autoscaler menggunakan metrics server untuk mengamati load, maka penggunaan resource menjadi lebih optimal karena mengurangi jumlah sumber daya yang digunakan ketika beban tidak besar. Ketika load mengalami lonjakan beban yang tajam dan tidak terduga (load spike), Kubernetes akan menanganinya dengan melakukan scale up pada layanan yang overloaded.

Load balancing sendiri tidak bersifat real-time karena perlu menunggu beban terjadi (diamati pada metrics server selama periode tertentu) dan melakukan scale-up. Oleh karena itu, pasti ada sebagian kecil request yang terpaksa harus dikorbankan (di-drop). Untuk mengatasi hal ini, frontend dari aplikasi dibuat untuk menyimpan semua form (benda apapun yang fillable, selectable, ataupun modifiable) agar user tidak kehilangan semua proses yang telah dilakukannya untuk membuat laporan (UX tidak rusak, user hanya perlu refresh/submit ulang). Untuk mengatasi POST request yang tidak idempotent, sistem dapat request terlebih dahulu nomor laporan agar nantinya hanya perlu dilakukan PUT request yang sifatnya idempotent.

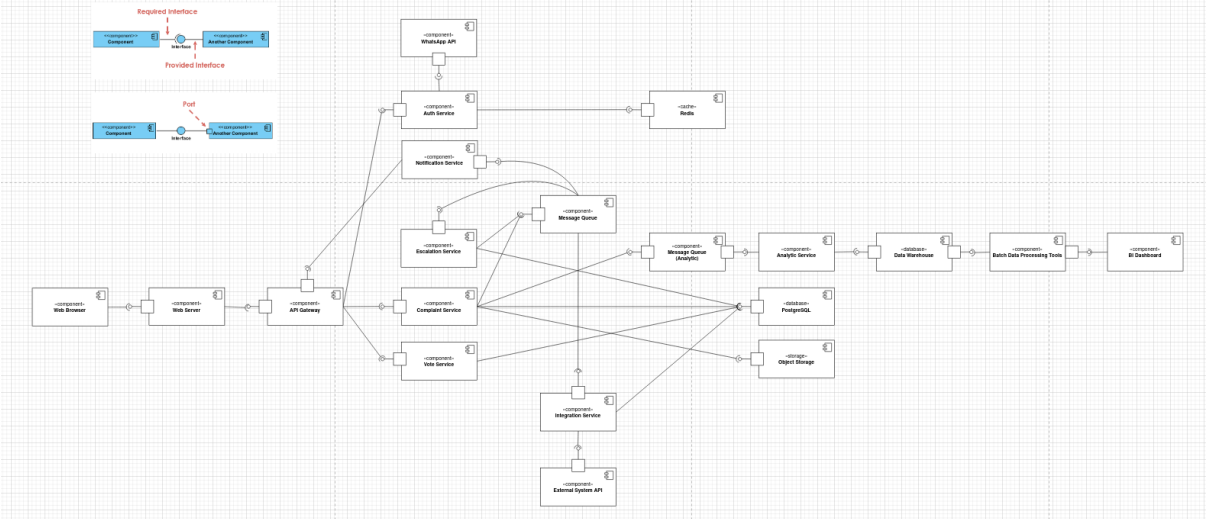
WAF dapat di-deploy di belakang load balancer untuk melakukan inspeksi konten yang masuk ke dalam sistem. Hal ini dimungkinkan karena ingress dapat melakukan early SSL termination, lalu meneruskannya ke WAF sebelum diteruskan ke layanan tujuan. Dengan adanya WAF, diharapkan sistem menjadi lebih aman dari serangan yang mungkin terjadi, seperti `react2shell` yang baru-baru ini terjadi dapat dicegah di sini jika sudah melakukan deployment WAF yang keras.

Observabilitas dicapai dengan menggunakan Prometheus untuk melakukan observasi pada kubernetes cluster, layanan web, firewall log, dan lain-lain. Hasil observasi ini akan dilempar ke pipeline analytics dan disimpan ke data warehouse. Setelah sampai di warehouse, data diproses menggunakan batch data processing tools dan divisualisasikan melalui BI dashboard. Dashboard dapat disesuaikan dengan kebutuhan masing-masing tim untuk melakukan tracing dan investigasi sistem oleh tim keamanan dan infrastruktur.

Secara umum, sistem terdiri dari beberapa layanan berikut:

1. API Gateway untuk mengatur routing request ke layanan-layanan backend
2. Auth Service untuk melakukan autentikasi dan otorisasi pengguna, integrasi dengan WhatsApp API untuk OTP
3. Notification service untuk mengirimkan notifikasi ke notifikasi kepada pelapor ketika terdapat kemajuan pada penyelesaian laporan-laporannya
4. Escalation service untuk eskalasi laporan ke departemen terkait
5. Complaint service untuk mengelola laporan-laporan yang telah warga buat serta melihat status penyelesaiannya
6. Vote service dibuat agar warga dapat melakukan upvote ke laporan-laporan yang ingin didukung
7. Integration service untuk integrasi dengan sistem eksternal
8. External system API untuk integrasi dengan sistem eksternal
9. WhatsApp API untuk mengirimkan OTP
10. Analytic service untuk visualisasi di BI dashboard
11. Batch data processor untuk memproses data

Diagram Arsitektur Sistem + Penjelasannya

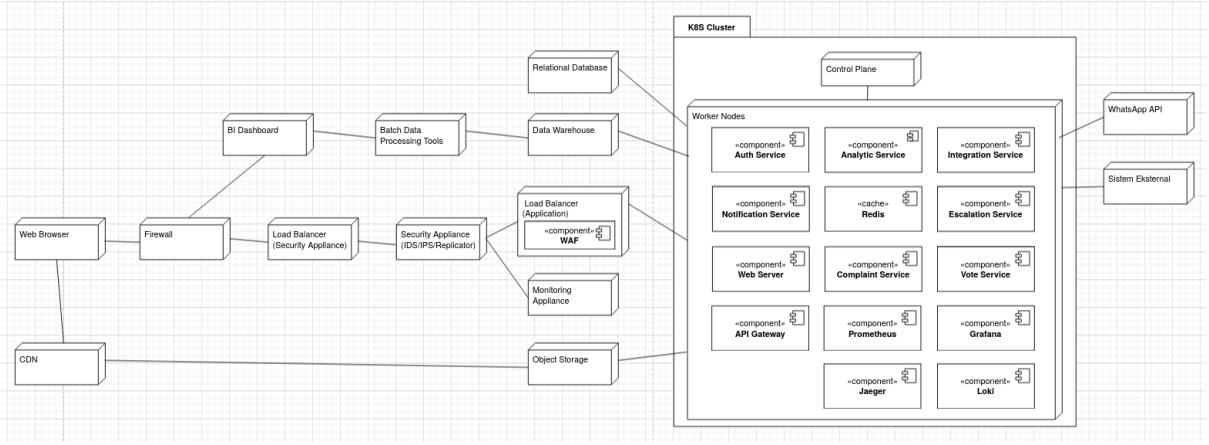


[Link draw.io]

UML Component Diagram menjelaskan interaksi antar komponen pada sistem. Berdasarkan kebutuhan fungsional sistem pada spesifikasi, diagram memiliki korelasi sebagai berikut:

1. Sistem menerapkan RBAC dengan session token dari Auth Service yang disimpan di Redis, serta jenis laporan (public, private, anonim) diatur oleh complaint service.

2. Pelaporan dan pengelolaan laporan dilakukan oleh complaint service dan disimpan di postgreSQL database.
3. Voting dilakukan melalui vote service yang melakukan write ke database
4. Terdapat 2 service yang menggunakan outbox pattern, complaint service menggunakan outbox pattern akan mengirimkan event ke message queue untuk mentrigger beberapa fitur seperti notifikasi, eskalasi pihak yang berwenang, analitik, integrasi sistem eksternal. Escalation service menggunakan outbox pattern untuk mengirimkan event setelah mengeskalasi pihak yang berwenang di DB ke message queue untuk men-trigger notification
5. Ketika ada laporan masuk atau laporan diselesaikan oleh pihak berwenang. Complaint service melakukan CRUD ke database. Lalu mengirimkan event ke message queue yang akan mentrigger notification melalui websocket
6. Ketika dilakukan suatu write oleh complaint service, akan dikirim event ke message queue analytic (mis. kafka) untuk diproses dan ditransfer secara batch oleh analytic service ke data warehouse, selanjutnya dapat digunakan batch processing tools seperti apache flink, atau query langsung ke data warehouse untuk divisualisasikan di BI dashboard
7. Pihak berwenang dapat meneruskan laporan terkait ke sistem eksternal melalui complaint service yang akan mengganti status "*forwarded*" pada laporan di DB dan mengirimkan event ke message queue, selanjutnya integration service akan membaca event, query database, dan meneruskan laporan ke sistem eksternal.
8. Untuk eskalasi pihak berwenang berdasarkan umur laporan, setiap write laporan oleh complain service ke database akan mengirimkan event dengan TTL ke delayed queue, ketika TTL habis, message otomatis dipindahkan ke queue lain dan dikonsumsi oleh Escalation Service yang akan mengecek status complaint, bila tidak solved akan di eskalasi.



Link draw.io

UML Deployment Diagram memvisualisasikan bentuk fisik atau node dari sistem. Berdasarkan kebutuhan non-fungsional sistem pada spesifikasi, diagram memiliki korelasi sebagai berikut:

1. Secara umum untuk mendukung skalabilitas dan latensi rendah, berdasarkan wilayah yang luas dan pengguna yang banyak serta complaint yang kemungkinan besar akan berhubungan dengan region tersebut. dapat dilakukan database sharding

menjadi beberapa lokasi utama (mis. Jawa Barat, Jawa Timur, Kalimantan, Papua). Sistem yang sama juga di-deploy multi-region sesuai pembagian shard.
2. Bila dibutuhkan suatu akses data terpusat (mis. Agregasi data beberapa region) bisa melalui sistem lain melalui integration service.
3. Sistem memvalidasi pengguna menggunakan OTP melalui whatsapp
4. Enkripsi *in-transit* menggunakan TLS dari client ke sistem dan antar komponen dalam sistem. Data pada storage dan database akan dienkripsi *at-rest*
5. Untuk mencegah cascading failure diterapkan circuit breaker pada service yang berhadapan dengan API luar seperti integration service, auth service dan service lain yang berpotensi menyebabkan cascading failure.
6. Sistem bisa menangani spike melalui rate limiting dan menggunakan message queue bila sudah masuk di dalam sistem. Setiap komponen dapat scaling secara horizontal kecuali write instance dari komponen stateful (DB)
7. Untuk memantau lalu lintas sistem dapat digunakan traffic replicator dan monitoring appliance sebelum application load balancer yang menuju sistem.

Daftar Teknologi untuk Setiap Komponen Sistem + Alasan Pemilihan Teknologi

- Framework backend digunakan Express karena flexible, minimal, dan cepat untuk melakukan pengembangan web service dengan framework ini.
- Kafka digunakan sebagai message queue untuk melakukan pengumpulan data analitik secara asinkron dan diproses secara batching.
- RabbitMQ digunakan sebagai message queue untuk melakukan pengumpulan event secara asinkron untuk orkestrasi antar service yang bergantung pada event.
- Database menggunakan Postgresql karena menggunakan prinsip ACID yang andal. DBMS ini juga digunakan karena skema database yang sudah terstruktur dan dapat diprediksi.
- Redis digunakan sebagai caching session token untuk RBAC agar latency bisa ditekan.
- Object storage (seperti MinIO) digunakan untuk menyimpan semua lampiran yang berupa unstructured data seperti foto dari laporan warga.
- Authorization memanfaatkan JWT karena sudah memanfaatkan JSON yang umum digunakan dalam pengembangan web service. Selain itu, JWT juga menghilangkan kebutuhan untuk menyimpan token dalam database, sehingga database tidak dibebankan oleh operasi yang sebenarnya tidak perlu.
- Node orchestrator menggunakan Kubernetes (k8s) dengan memanfaatkan fitur HPA, ingress (nginx), dan metrics server untuk memudahkan pengembang sistem untuk mengorkestrasi deployment layanan. Fitur-fitur k8s seperti HPA, ingress, dan metrics server juga memudahkan sysadmin/SRE untuk melakukan pengawasan metric, autoscaling, dan otopsi ketika ada layanan yang mati.
- Prometheus, loki, dan Grafana digunakan untuk pengumpulan dan visualisasi metrics dan logs sistem dan aplikasi secara real-time
- Jaeger digunakan untuk tracing agar dapat melacak aliran request end-to-end di berbagai layanan.
- Apache flink digunakan untuk batch processing yang menghasilkan visualisasi di BI.

- BI Dashboard untuk menampilkan visualisasi yang bisa digunakan oleh kewenangan tertinggi untuk memantau kinerja bawahan-bawahannya dalam menanggapi masalah-masalah yang dilaporkan oleh warga.
- Encryption at rest (seperti dm-crypt, dll) untuk menjaga keamanan ketika server dimatikan oleh adversary.
- Transport Layer Security 1.3 (TLS 1.3) untuk encryption in-transit data dari client ke server.
- Web application firewall (WAF) untuk proteksi web, gunakan sesuai cloud provider yang digunakan atau deploy sendiri.
- Websocket digunakan untuk real-time notification ke pelapor dan update pada dashboard pihak yang berwenang.

Pemilihan Fungsionalitas dan Kualitas yang Diimplementasi + Alasan

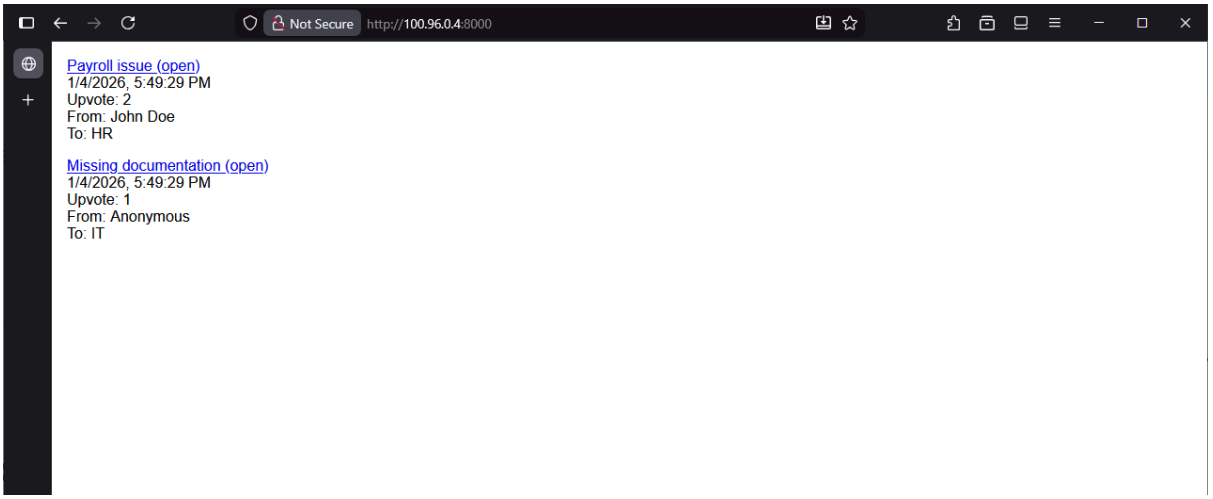
Untuk *Proof-of-Concept* (PoC), dipilih fungsionalitas pelaporan (buat, kelola, dan lihat), login, comment, dan upvote. Fungsionalitas-fungsionalitas ini berhadapan dengan user (warga yang membuat laporan) dan dapat mewakilkan aplikasi akhir (yang sudah jadi). Fungsionalitas-fungsionalitas untuk departemen pemerintah tidak diimplementasikan karena pada akhirnya, pengguna yang dijadikan first-class citizen adalah warga yang ingin melakukan pelaporan. Justifikasinya adalah jika warganya saja tidak mau menggunakan aplikasi ini, pasti pada akhirnya aplikasi ini akan mati.

Selain itu, juga dipilih kualitas keamanan yang dapat terlihat dari login dan role checking setiap kali melakukan invokasi sebuah service. Kualitas reliability juga dipilih yang terlihat ketika 1 service mati, aplikasi keseluruhan tidak mati. Kualitas kinerja yang terlihat dari response time PoC yang cepat dan terakhir, kualitas scalability yang terlihat dari desain service yang stateless. Kualitas-kualitas ini dipilih karena kualitas-kualitas ini dapat diimplementasikan tanpa melakukan deployment seperti aplikasi final.

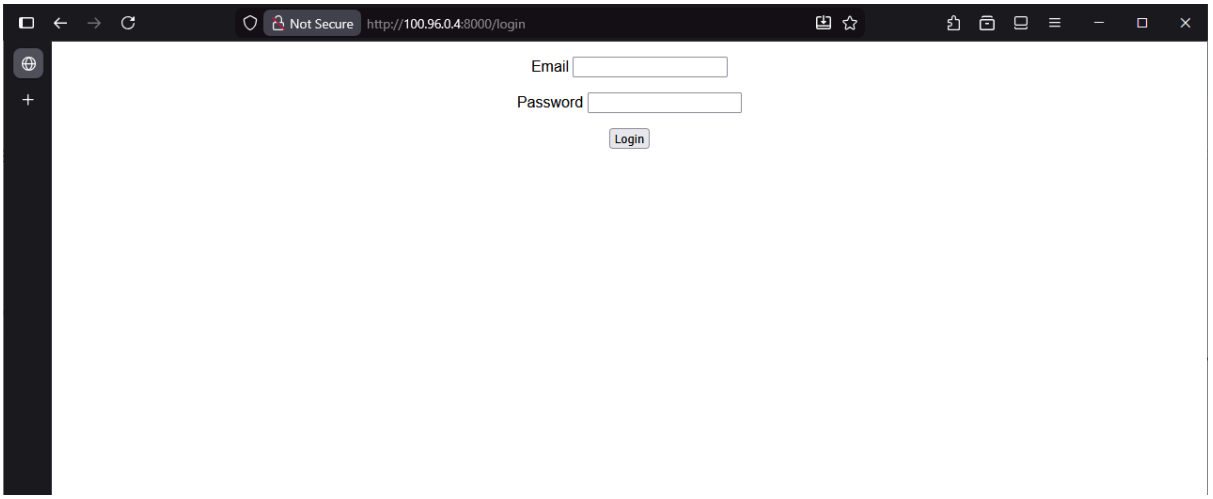
Penjelasan Hasil Implementasi PoC

PoC yang diimplementasikan memiliki dua *service* yang *stateless* beserta salah satu *service persistent* berupa *database*. Dua *service stateless* tersebut adalah *frontend* dan *backend*. Seluruh *service* ini memiliki fungsionalitas seperti yang dijelaskan sebelumnya. Pada bagian *frontend*, digunakan html, css, dan javascript biasa. Selain itu juga digunakan nginx sebagai *reverse proxy*, yang juga meneruskan *request* menuju *backend* pada *endpoint* `/api`. Pada *backend* sendiri digunakan *express* serta *library* untuk melakukan koneksi dengan *database*. Autentikasi dan autorisasi dilakukan menggunakan JWT.

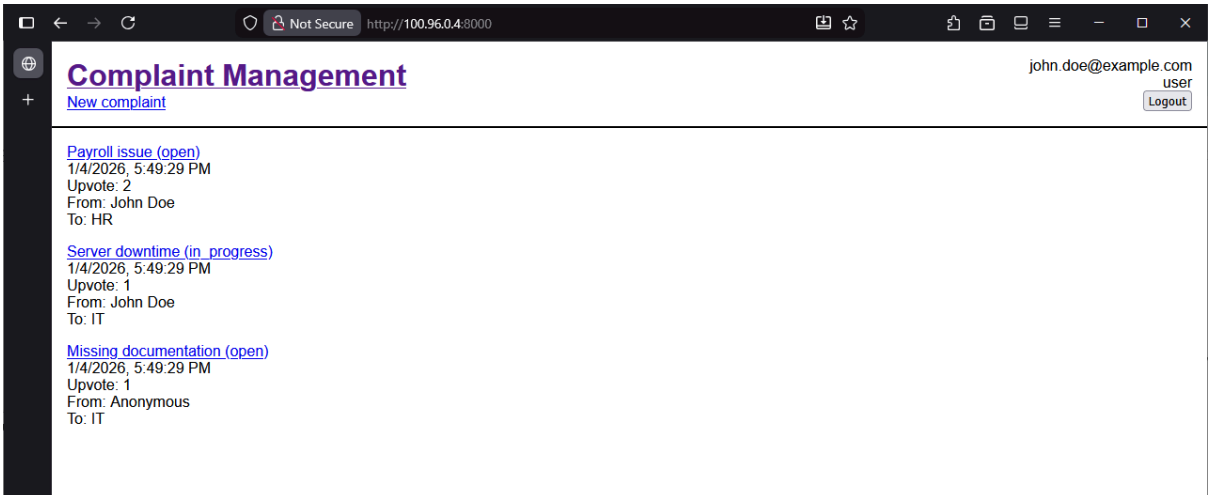
Dokumentasi PoC



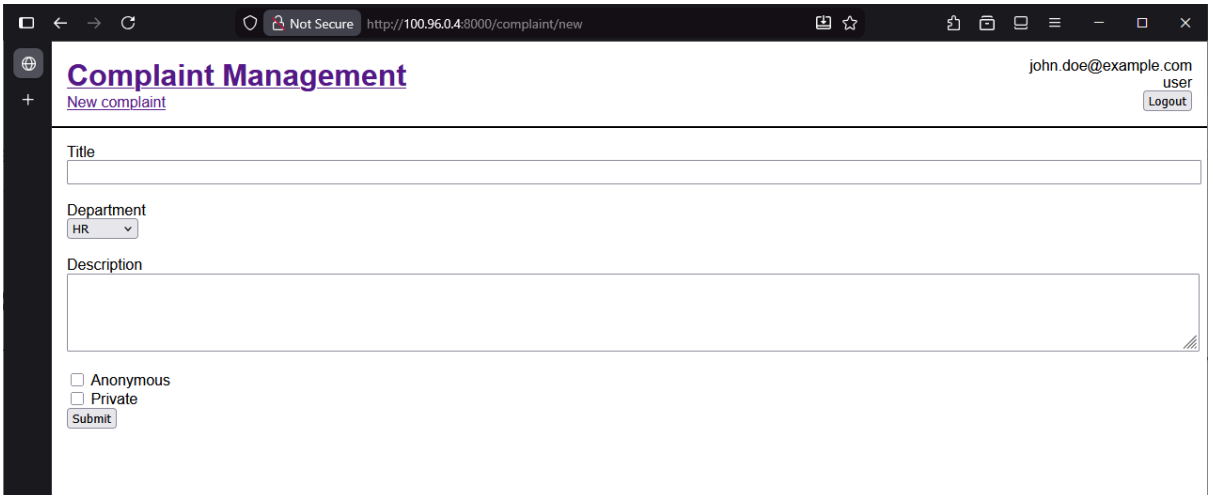
Gambar 1. Halaman List Complaint Public



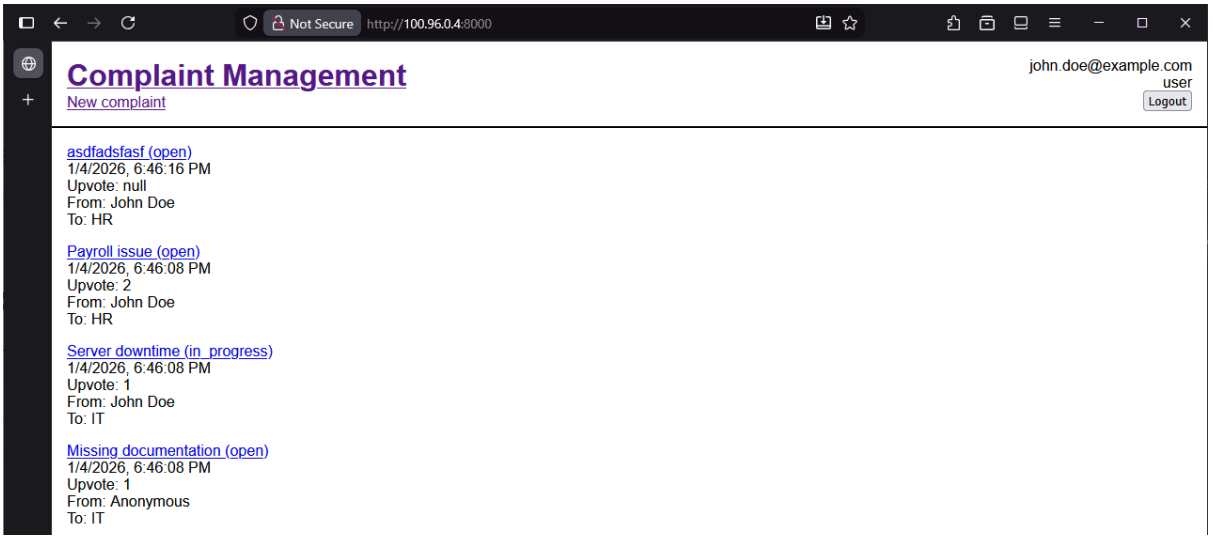
Gambar 2. Halaman Login



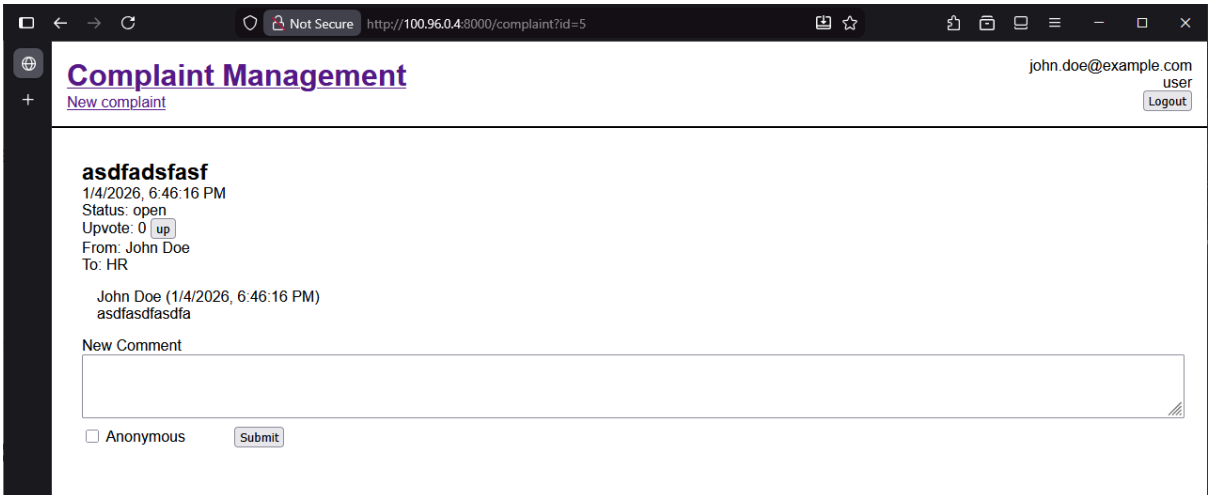
Gambar 3. Halaman List Complaint setelah Login



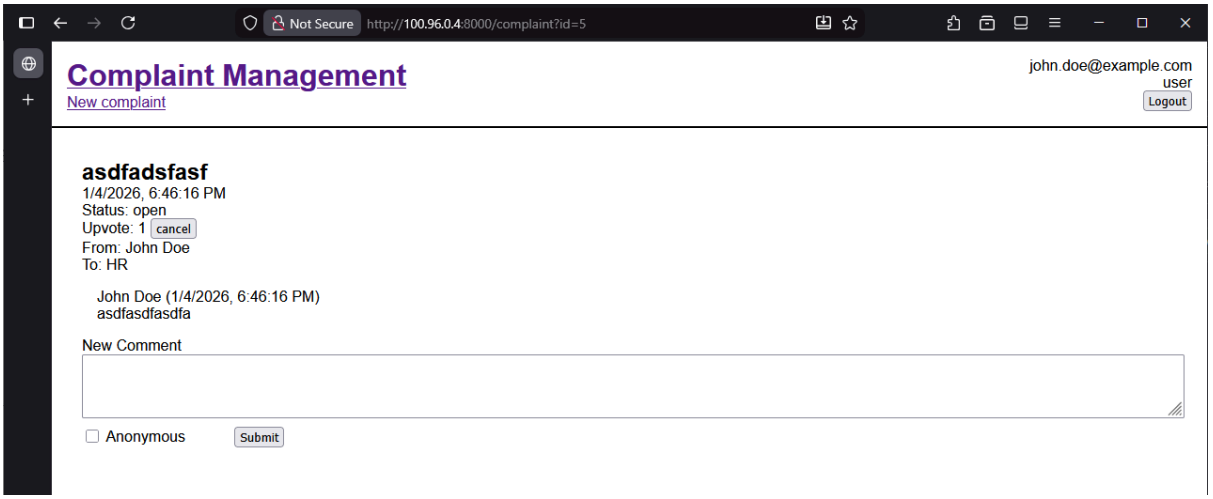
Gambar 4. Halaman Pembuatan Complaint Baru



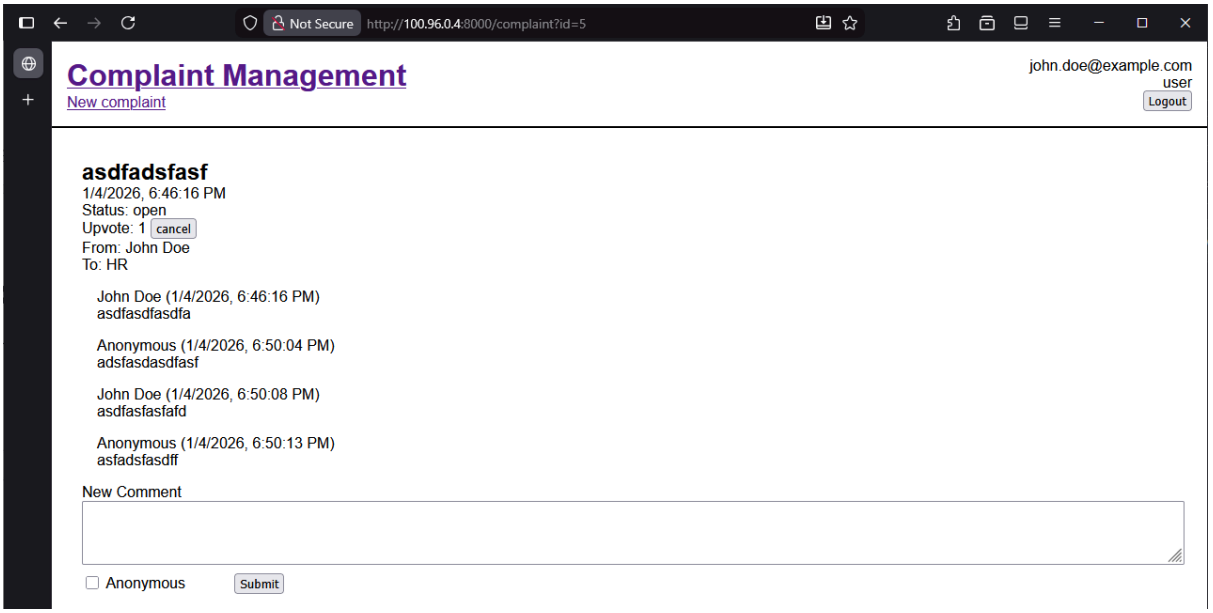
Gambar 5. Halaman List Complaint setelah Membuat Complaint Baru



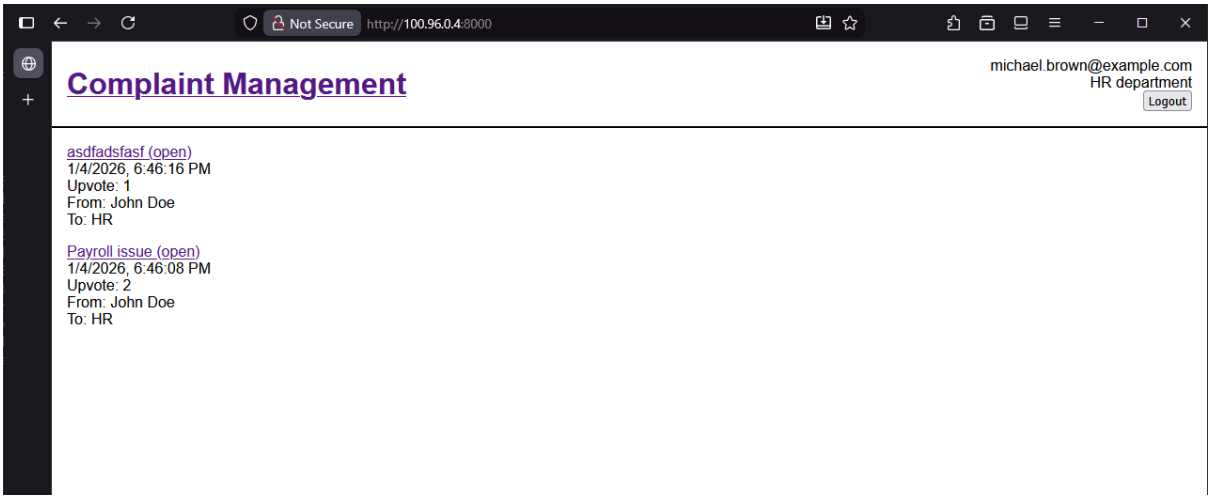
Gambar 6. Detail Complaint yang Telah Dibuat



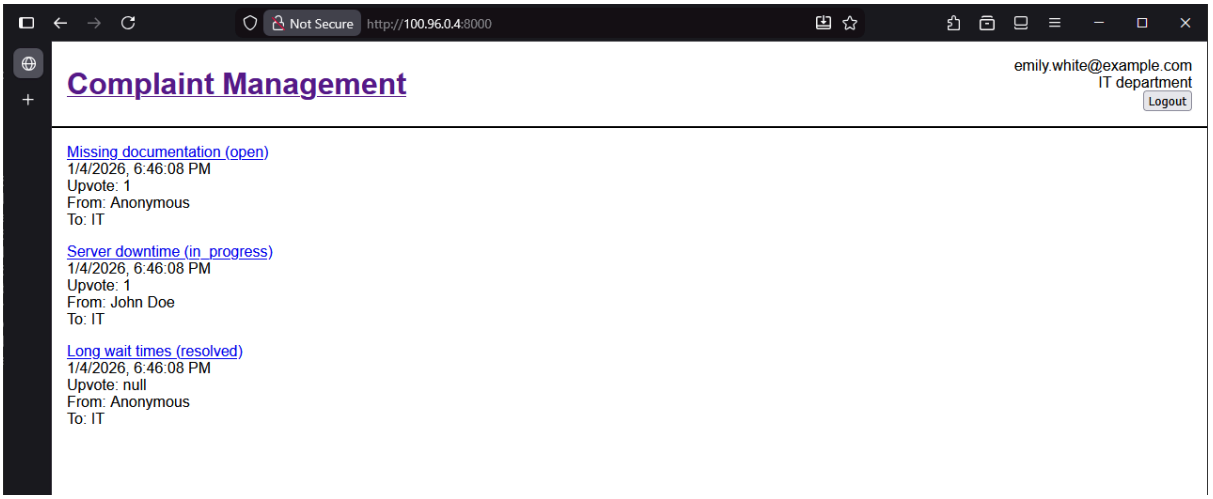
Gambar 7. Detail Complaint yang Telah Di-vote-up



Gambar 8. Detail Complaint yang Telah Diisi Komentar



Gambar 9. List Complaint dari Sisi Departemen HR



Gambar 10. List Complaint dari Sisi Departemen IT

Asumsi

1. Analisis data terkait masalah yang dilaporkan diasumsikan menjadi batch analysis seperti ranking department mana yang memiliki masalah paling banyak pada rentang waktu tertentu.

Tautan Kode Sumber Implementasi Proof-of-Concept (PoC)

https://github.com/atpur-rafif/IF4031.git

Tautan Video Demonstrasi Aplikasi

https://drive.google.com/file/d/1WFPxgzHLqYDTUjTmpvUyZSVfHevYwABv/view?usp=sharing