Perancangan dan Implementasi Aplikasi Perdagangan Saham dengan Memanfaatkan gRPC untuk Komunikasi Real-Time

Rahmat Hidayatullah[1], I Made Suartana[2]

[1,2] S1 Teknik Informatika, Universitas Negeri Surabaya
[1]rahmat.21020@mhs.unesa.ac.id
[2]madesuartana@unesa.ac.id

***Abstrak*— Perkembangan teknologi informasi mendorong digitalisasi dalam sektor keuangan, khususnya perdagangan saham yang menuntut akses informasi cepat, akurat, dan real-time. Penelitian ini bertujuan untuk merancang dan mengimplementasikan aplikasi perdagangan saham berbasis Android yang memanfaatkan *framework* gRPC guna menunjang kebutuhan tersebut. Metode yang digunakan meliputi tahapan analisis kebutuhan, perancangan sistem menggunakan arsitektur *client-server* dan *bidirectional streaming*, implementasi dengan Flutter untuk klien dan Golang untuk server, serta pengujian melalui uji serialisasi dan uji beban. Uji serialisasi menunjukkan bahwa penggunaan Protocol Buffers menghasilkan ukuran data lebih kecil (33 byte) dan waktu pemrosesan lebih cepat (421 ns) dibandingkan format JSON (226 byte dan 1.05 µs). Uji beban menggunakan 20 *virtual users* menunjukkan latensi rata-rata sebesar 4 milidetik dengan 100% permintaan berhasil tanpa *error*, menandakan sistem sangat responsif dan andal untuk komunikasi *real-time*. Fitur utama seperti pemantauan harga saham secara langsung dan transaksi jual beli dengan notifikasi status secara otomatis berhasil diimplementasikan dengan baik. Hasil penelitian ini menunjukkan bahwa gRPC dengan *bidirectional streaming* dapat meningkatkan efisiensi dan kecepatan komunikasi data pada aplikasi perdagangan saham secara signifikan**

***Kata Kunci* — gRPC, perdagangan saham, Android, real-time, Flutter, bidirectional streaming**

I. PENDAHULUAN

Untuk memaksimalkan keuntungan, industri keuangan kontemporer sangat bergantung pada pengaturan sumber daya yang tepat. Pasar saham adalah salah satu cara yang sering digunakan oleh individu dan organisasi untuk mencapai tujuan keuangan [1]. Perdagangan saham yang sebelumnya dilakukan secara konvensional kini semakin banyak beralih ke platform digital, sehingga transaksi dapat dilakukan secara daring dan lebih efisien.

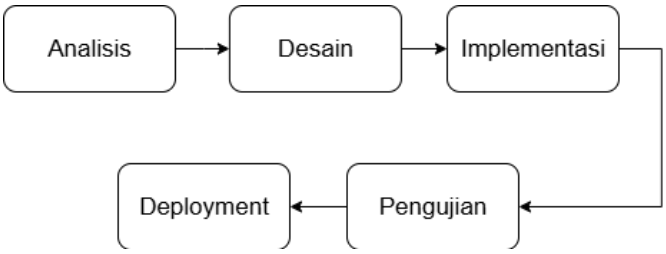
Perdagangan saham secara daring, terutama melalui perangkat mobile telah mengubah gaya investasi industri. Dengan adanya aplikasi perdagangan saham berbasis Android, para investor dapat memantau harga saham, melakukan transaksi jual beli saham dan menerima notifikasi perubahan pasar secara langsung kapan pun dan di mana pun. Hal ini membuat pasar lebih mudah diakses dan membantu investor membuat keputusan lebih cepat saat harga saham berubah-ubah.

Tujuan utama dari investasi di pasar saham adalah memperoleh keuntungan dengan cara membeli saham di harga rendah dan menjualnya di harga tinggi [2]. Untuk mencapai tujuan ini, investor sangat membutuhkan akses ke informasi harga saham yang akurat dan *real-time*. Aplikasi perdagangan saham yang andal dan mampu memberikan informasi pasar secara *real-time* menjadi kebutuhan yang tak terelakkan, mengingat setiap perubahan harga yang terjadi bisa berpengaruh langsung terhadap keuntungan atau kerugian investor.

Dalam beberapa tahun terakhir, perdagangan frekuensi tinggi (*high-frequency trading*) telah muncul sebagai strategi populer di kalangan investor besar. Dengan volume transaksi yang besar dan waktu eksekusi yang sangat cepat, perdagangan frekuensi tinggi bergantung pada kecepatan dan stabilitas infrastruktur teknologi [3]. Dengan kecepatan eksekusi perdagangan yang lebih cepat menjadi milidetik atau bahkan mikrodetik, mereka yang mampu menggunakan teknologi terbaru mendapat keuntungan. Oleh karena itu, kehadiran aplikasi perdagangan saham yang cepat, stabil dan dapat diandalkan menjadi kebutuhan utama bagi para investor, terutama dalam hal memantau pergerakan harga saham secara *real-time*.

Penelitian ini bertujuan untuk mengatasi tantangan yang dihadapi oleh investor dalam perdagangan saham, khususnya terkait volatilitas pasar serta kebutuhan akan kecepatan dan keakuratan informasi, dengan merancang dan mengimplementasikan aplikasi perdagangan saham berbasis Android yang memanfaatkan gRPC sebagai *framework* komunikasi *real-time*. gRPC dirancang sebagai *framework* yang dapat berkomunikasi dengan cepat dan efisien, sehingga dapat menyampaikan informasi harga saham secara *real-time* dengan latensi yang minimal. Aplikasi yang dirancang dalam penelitian ini diharapkan dapat menawarkan pengalaman pengguna yang responsif dengan data yang akurat dengan menerapkan gRPC. Selain itu, penerapan teknologi ini akan memudahkan pengguna dalam memantau harga saham secara tepat waktu dan melakukan transaksi dengan lebih cepat dan efisien, hal ini nantinya akan diukur dari segi beban dan ukuran dari data. Aplikasi ini akan dilengkapi dengan fitur *bidirectional streaming* pada pengambilan data harga saham secara *real-time.*

II. METODE PENELITIAN



Gbr. 1 Metodologi Penelitian

Tujuan dari penelitian ini adalah untuk merancang dan mengimplementasikan aplikasi perdagangan saham yang dapat berkomunikasi secara *real-time* dengan memanfaatkan gRPC. Penggunaan teknologi ini diharapkan dapat membuat aplikasi perdagangan saham berjalan dengan cepat dan secara *real-time*.

Sebagaimana yang terlihat pada Gbr. 1, terdapat beberapa alur rancangan penelitian. Penelitian ini dimulai dengan tahap analisis, dimana penulis menganalisis sistem gRPC dan mengidentifikasi variabel-variabel apa saja yang diperlukan dalam pembuatan sistem perdagangan saham pada platform android. Selanjutnya, pada tahap desain akan dilakukan rancangan desain sistem yang dibuat berdasarkan hasil dari analisis kebutuhan yang telah diidentifikasi. Setelah itu, akan masuk ke tahap implementasi, yang melibatkan penulisan kode sesuai rancangan desain sistem sebelumnya. Setelah implementasi selesai, tahap pengujian dilakukan untuk memastikan apakah aplikasi berjalan sesuai dengan baik dan tidak ada masalah atau *bug* apapun. Jika pengujian menunjukkan hasil yang memuaskan, penelitian berlanjut ke tahap akhir, yaitu *deployment*, di mana aplikasi akan di-*deploy* ke VPS. Tujuannya adalah untuk mengisolasi *node* yang akan berjalan.

A. *Analisis*

Pada tahap ini, analisis sistem gRPC dilakukan untuk mengidentifikasi variabel yang diperlukan untuk pengembangan aplikasi perdagangan saham pada platform Android. Selain itu, analisis kebutuhan juga dilakukan dengan memperhatikan beberapa aspek, yaitu kebutuhan fungsional (aktivitas atau layanan yang disediakan oleh sistem) dan kebutuhan non-fungsional (kondisi operasional). Berikut adalah spesifikasi yang diterapkan dalam aplikasi perdagangan saham pada platform Android:

1) Kebutuhan Fungsional

- Aplikasi harus dapat melakukan *register* dan *login*.
- Pengguna aplikasi harus dapat melakukan jual dan beli saham.
- Pengguna aplikasi harus dapat melihat data saham secara *real-time*, yang berupa pesanan, diagram *candlestick* saham, dan portfolio saham.

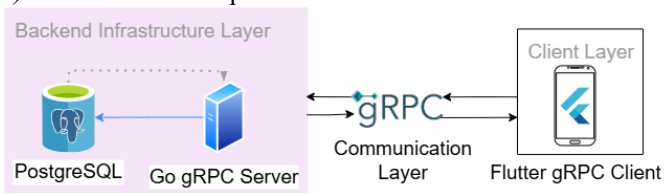
2) Kebutuhan Non-Fungsional

- Aplikasi dapat menangani permintaan RPC dalam skala banyak secara bersamaan.
- Sistem harus dapat menangani komunikasi dengan latensi rendah untuk memastikan responsivitas perdagangan *real-time*.
- Data pasar saham harus dikirimkan secara konsisten dan tanpa kehilangan data pada koneksi *streaming* gRPC.

B. *Desain*

Pada tahap ini, penulis akan membuat rancangan desain yang didasarkan pada kebutuhan yang telah diidentifikasi, dimulai dari bagaimana desain arsitektur gRPC, proses implementasi gRPC, *serialization, deserialization, dan real-time messaging*. Berikut ini adalah detail rancangan desain aplikasi perdagangan saham pada platform Android yang memanfaatkan gRPC untuk komunikasi *real-time*:

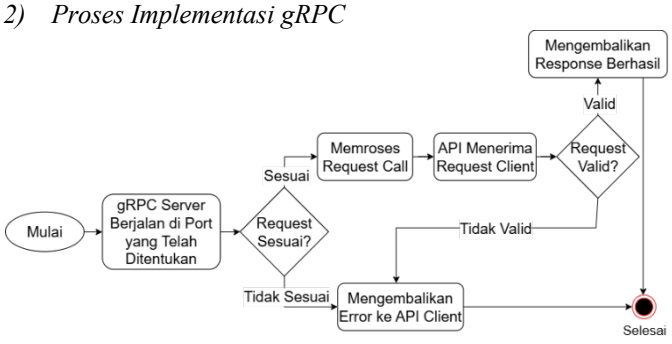
1) Desain Sistem Aplikasi



Gbr. 2 Desain Sistem Aplikasi

Pada Gbr. 2 tersebut, menunjukkan alur komunikasi antara aplikasi perdagangan saham berbasis Android dengan server menggunakan *framework* gRPC untuk komunikasi *real-time*. Android berfungsi sebagai klien dengan mengirimkan permintaan (*request*) ke server melalui *framework* gRPC, misalnya untuk mendapatkan data harga saham terbaru atau melakukan transaksi jual beli saham. Permintaan ini diteruskan oleh API ke *database* yang menyimpan informasi seperti data saham, riwayat transaksi, dan informasi pengguna. Setelah *database* memproses permintaan, server API mengirimkan respons (*response*) kembali ke aplikasi. Selain itu, API juga mendukung fitur *bidirectional streaming*, di mana data harga saham terus diperbarui dan dikirimkan ke aplikasi tanpa perlu membuat koneksi baru. Dengan menggunakan *framework* gRPC, komunikasi antar komponen menjadi efisien, cepat, dan mendukung pembaruan data secara langsung, yang sangat penting dalam konteks perdagangan saham.

2) Proses Implementasi gRPC

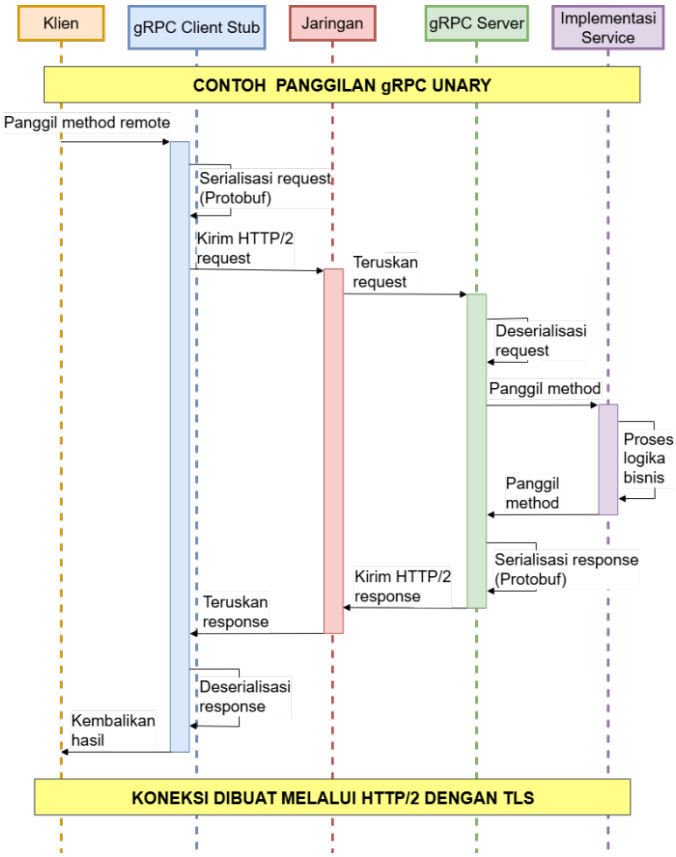


Gbr. 3 Proses Implementasi gRPC

Pada Gbr. 3 ini menunjukkan diagram proses implementasi gRPC, yang menggambarkan bagaimana klien dan server berkomunikasi data selama proses permintaan berlangsung. Server gRPC menerima permintaan dari klien, seperti data harga saham atau transaksi, kemudian memverifikasi format dan isi permintaan. Jika validasi gagal, server mengembalikan

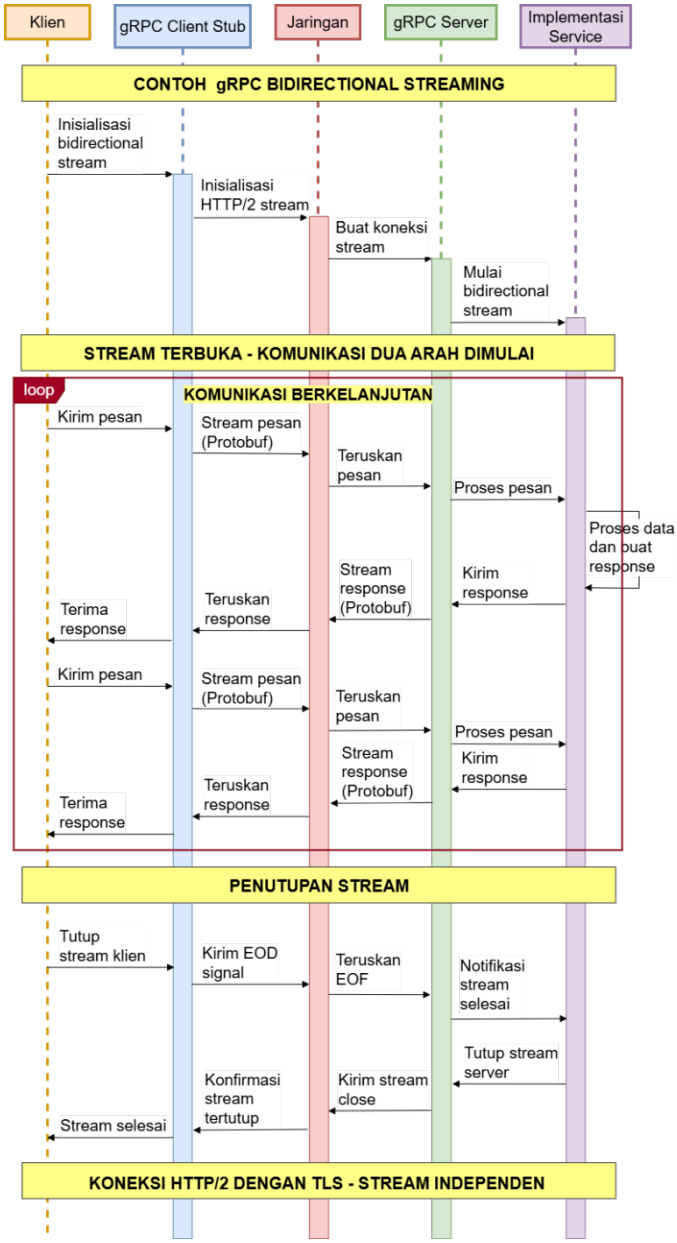
pesan *error*. Jika validasi berhasil, permintaan diproses melalui API *backend* dan dikirim kembali ke klien. Proses ini mendukung komunikasi yang cepat, efisien, dan akurat, khususnya dalam memenuhi kebutuhan aplikasi perdagangan saham secara *real-time*.

3) Alur Komunikasi gRPC



Gbr. 4 Unary

Diagram pada Gbr. 4 tersebut menggambarkan alur komunikasi pada gRPC, yaitu *Unary*. Pada komunikasi *Unary*, klien mengirim satu permintaan kepada server dan menerima satu respons. Proses diawali ketika klien memanggil *method* melalui gRPC *Client Stub*, lalu data diserialisasi menggunakan Protobuf dan dikirim melalui protokol HTTP/2 yang aman dengan TLS. Setelah permintaan diterima oleh server, data dideserialisasi dan diproses oleh layanan yang menjalankan logika bisnis. Respons yang dihasilkan kemudian diserialisasi kembali, dikirim melalui HTTP/2, dan diterima oleh klien setelah dideserialisasi oleh *stub*.



Gbr. 5 Bidirectional Streaming

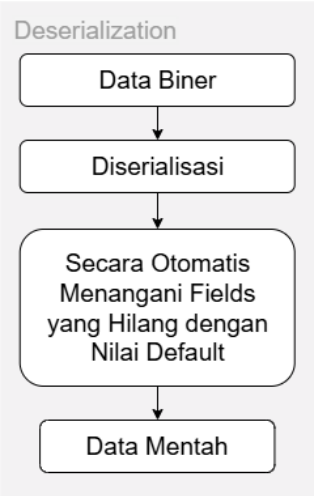
Diagram pada Gbr. 5 tersebut menggambarkan alur komunikasi pada gRPC, yaitu *Bidirectional Streaming*. Pada *Bidirectional Streaming*, komunikasi terjadi dua arah secara simultan melalui koneksi *stream* yang terbuka. Proses dimulai dengan inisialisasi koneksi oleh klien, yang kemudian diteruskan sebagai *stream* HTTP/2 dengan enkripsi TLS. Setelah *stream* aktif, klien dan server dapat saling mengirim pesan secara paralel dan berulang tanpa menunggu respons penuh dari pihak lain. Ketika komunikasi selesai, klien mengirim sinyal EOF yang diteruskan ke server sebagai penanda akhir *stream*. Server kemudian menutup *stream* dan sistem memastikan koneksi ditutup secara aman. Model ini mendukung komunikasi *real-time* yang efisien dan independen.

4) Serialization

Gbr. 6 Serialization

Proses *serialization* dalam implementasi gRPC, seperti yang diilustrasikan pada Gbr. 6, merupakan langkah penting langkah penting dalam mengubah data mentah menjadi format biner yang dapat ditransmisikan melalui jaringan. Proses ini dimulai dengan pengecekan kesesuaian data terhadap skema yang telah ditentukan dalam *Proto Contract* menggunakan Protocol Buffers (.proto file). Setelah validasi, data diserialisasi menjadi format biner yang ringan dan cepat. Dalam sistem perdagangan saham, proses ini mendukung pengiriman data seperti harga saham dan instruksi transaksi secara *real-time* dengan latensi rendah dan efisiensi tinggi. Selain itu, serialisasi memastikan bahwa data tetap aman selama proses pengiriman, sehingga klien dan server dapat memproses data dengan benar.

5) Deserialization



Gbr. 7 Deserialization

Proses *deserialization* dalam implementasi gRPC, seperti yang diilustrasikan pada Gbr.7, merupakan tahap konversi data biner yang diterima melalui jaringan menjadi data mentah yang dapat diolah oleh aplikasi. Proses ini menggunakan skema dari Proto Contract, dan secara otomatis menangani field yang tidak tersedia dengan nilai default sesuai definisi pada file .proto. Hal ini menjaga validitas data meskipun terdapat *field* yang tidak dikirimkan. Dalam implementasinya pada aplikasi perdagangan saham, deserialisasi berperan dalam pemrosesan data seperti harga, volume, dan status transaksi secara cepat dan akurat, sehingga mendukung kelancaran transaksi *real-time*.

6) Real-Time Messaging

Dengan menggunakan *real-time messaging*, pembaruan data harga saham dapat dilakukan secara langsung dan terus-menerus tanpa perlu permintaan ulang dari klien, sehingga pengguna dapat memantau perubahan pasar secara akurat. Dengan memanfaatkan fitur *streaming* pada gRPC, data harga saham dikirimkan dari server ke klien menggunakan protokol HTTP/2, yang mendukung komunikasi dengan latensi rendah melalui kompresi *header* dan *multiplexing*. Format data menggunakan Protobuf juga berperan penting dalam mempercepat transmisi karena menghasilkan ukuran pesan yang ringan. Dengan pendekatan ini, aplikasi dapat memberikan pengalaman *real-time* yang optimal kepada pengguna, khususnya dalam mengelola portofolio saham dan mengambil keputusan investasi secara tepat waktu.

C. Impementasi

Setelah tahap desain sistem selesai, maka proses selanjutnya ialah tahap implementasi. Pada tahap ini, penulis melakukan implementasi dari berbagai fitur dan fungsionalitas yang telah ditentukan sebelumnya dengan menggunakan teknologi dan bahasa pemrograman yang sesuai.

D. Pengujian

Dalam penelitian ini, skenario pengujian dimulai dengan mengumpulkan data untuk permintaan (*request*) pada aplikasi. Selanjutnya, dipilih beberapa indikator yang cukup untuk kebutuhan pengujian.

1) Pengumpulan Data

TABEL I
CONTOH DATASET

Tanggal	Close	High	Low	Open	Volume
01/04/2021	Rp6,254.94	Rp6,273.24	Rp6,149.70	Rp6,222.91	47937000
01/05/2021	Rp6,488.30	Rp6,488.30	Rp6,268.67	Rp6,277.82	104831000
01/06/2021	Rp6,355.60	Rp6,474.57	Rp6,296.12	Rp6,451.69	89753500
01/07/2021	Rp6,373.91	Rp6,451.69	Rp6,323.58	Rp6,405.94	71360000
01/08/2021	Rp6,451.69	Rp6,479.15	Rp6,383.06	Rp6,437.97	75033500

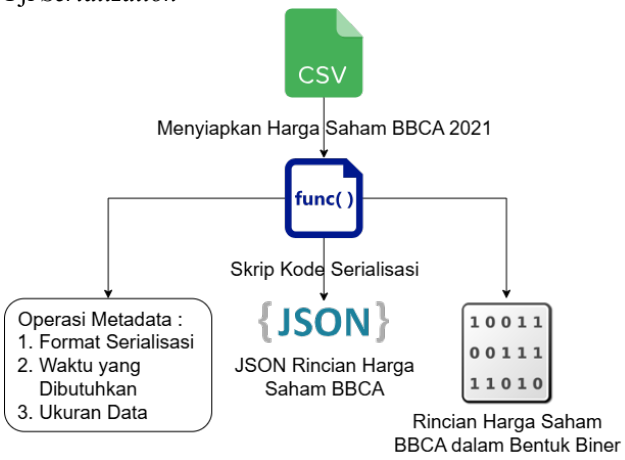
Data pada Tabel I merupakan contoh dataset yang digunakan dalam penelitian ini, yang tersedia di Yahoo Finance, dengan fokus pada tahun 2021. Penggunaan data ini dilakukan secara spesifik hanya untuk menguji kredebillitas gRPC dalam pembaruan stok saham secara *real-time*. Penulis hanya menggunakan data ini untuk menguji kredibilitas gRPC dalam pembaruan stok saham secara *real-time*. Perlu diingat bahwa tujuan utama penelitian hanyalah untuk menguji gRPC sebagai alat komunikasi antar sever dan klien.

2) Pengujian

Pengujian aplikasi perdagangan saham ini dimulai dengan penulis memberikan data harga saham BBCA historis tahun 2021 dalam format CSV ke server *backend*. *Script Custom*

Stock Injector kemudian mengirim data ke Go gRPC server, yang kemudian memproses dan menyiarkan data ke aplikasi klien yang menggunakan *framework* Flutter secara *real-time*. Pengujian ini berfokus pada dua indikator utama, yakni:

- Uji *Serialization*

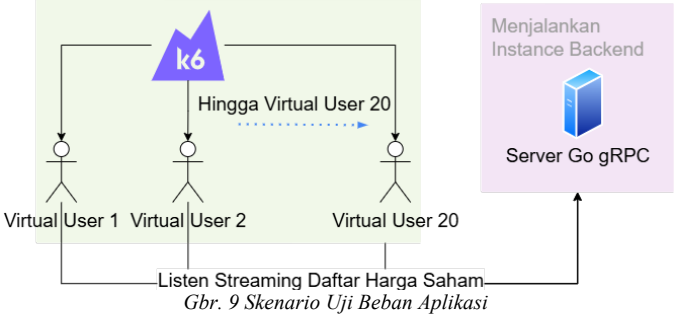


Gbr. 8 Skenario Uji Serialisasi

Pada Gbr. 8, menunjukkan skenario pengujian *serialization*, dengan membandingkan efisiensi ukuran data antara format biner dan JSON dalam proses transmisi. Melalui skrip khusus, data dari file CSV diambil sebanyak 50 entri pertama, lalu diserialisasi ke dalam kedua format tersebut. Pengujian ini menghasilkan informasi berupa waktu serialisasi dan ukuran file akhir dari masing-masing format, guna mengevaluasi efisiensi penggunaan format biner dibandingkan JSON yang umum digunakan.

- Uji Beban

Pengujian ini bertujuan untuk menilai kemampuan aplikasi dalam menangani sejumlah permintaan secara bersamaan. Proses pengujian dilakukan menggunakan alat bantu Grafana K6. Variabel pengujian pada uji beban ini adalah 20 *virtual users,* gRPC *request duration (mean, max, median, min)*, serta total *request* berhasil dan gagal. Pada Gbr. 9 ini merupakan rincian dari skenario uji beban:



Gbr. 9 Skenario Uji Beban Aplikasi

Proses pengujian beban ini dimulai dari k6 membuat virtual user sebanyak 20 *virtual users*. Lalu masing-masing virtual user akan mensimulasikan proses *subscribe* harga saham yang di-*broadcast* dari server yang telah disimulasikan pada uji simulasi harga saham.

3) Analisa Hasil

Untuk mendapatkan hasil latensi rata-rata (*avg*) total, maka diperlukan formula sebagai berikut: [13]

$$\frac{\sum_{i=1}^{n} \times (avg_i \times jumlah\ request_i}{\sum_{1=1}^{n} \times jumlah\ request_i)}$$

TABEL II
KETERANGAN FORMULA

Keterangan	
$\sum_{i=1}^{n}$	Notasi sigma yang menyatakan penjumlahan dari iterasi ke-1 sampai ke-n
n	Jumlah total iterasi pengujian
avg_i	Nilai rata-rata (*average*) latensi dalam milidetik pada iterasi ke-i
$jumlah\ request_i$	Jumlah total permintaan (*requests*) yang berhasil pada iterasi ke-i

Berdasarkan penelitian dalam jurnal-jurnal terkait [14], batas wajar latensi sistem yang masih dapat diterima oleh pengguna adalah maksimal 250 milidetik (ms). Jika latensi sistem melebihi 250 milidetik (ms), keterlambatan mulai terasa dan dapat mengganggu interaksi pengguna. Ini terutama berlaku untuk sistem dengan arsitektur thin *client*. Oleh karena itu, latensi kurang dari 250 ms dianggap baik dan ideal.

E. Deployment

Pada tahap ini, aplikasi akan di-*deploy* ke VPS. Tujuannya adalah untuk mengisolasi node yang akan berjalan.

III. HASIL DAN PEMBAHASAN

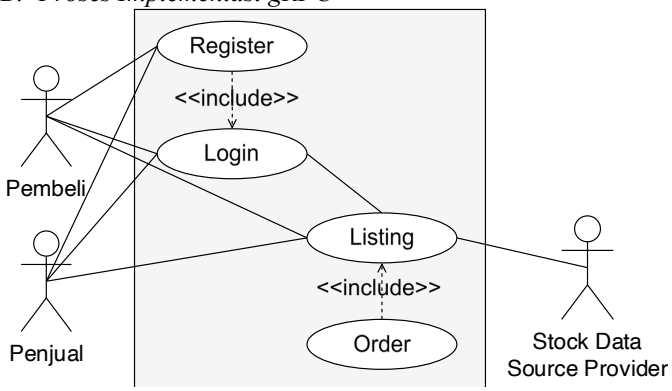
Pada bagian ini, dijelaskan proses perancangan sistem berdasarkan hasil analisis kebutuhan fungsional dan kebutuhan non-fungsional yang telah dijabarkan pada bab sebelumnya. Dari hasil analisis tersebut, dirumuskan rancangan sistem yang akan dikembangkan, berupa Class Diagram, Sequenced Diagram, dan Use Case Diagram. Rancangan ini bertujuan untuk memberikan gambaran awal terhadap arsitektur sistem, alur data, antarmuka pengguna, serta struktur *database*. Desain ini mencakup berbagai aspek teknis yang diperlukan agar sistem dapat berjalan sesuai dengan tujuan dan kebutuhan pengguna yang telah diidentifikasi sebelumnya.

A. Class Diagram

Gbr. 10 Class Diagram

Pada Gbr. 10 menunjukkan diagram kelas (class diagram) yang menggambarkan sistem aplikasi perdagangan saham yang terdiri dari enam class utama, yaitu *User, Stock, Order, Portfolio*, dan *StockPrice*, beserta properti, *method*, dan relasi antar class-nya. Sistem ini dirancang untuk menyediakan kapabilitas bagi pengguna membeli, menjual, dan memantau saham mereka dalam portofolio, serta melihat riwayat harga saham.

B. Proses Implementasi gRPC

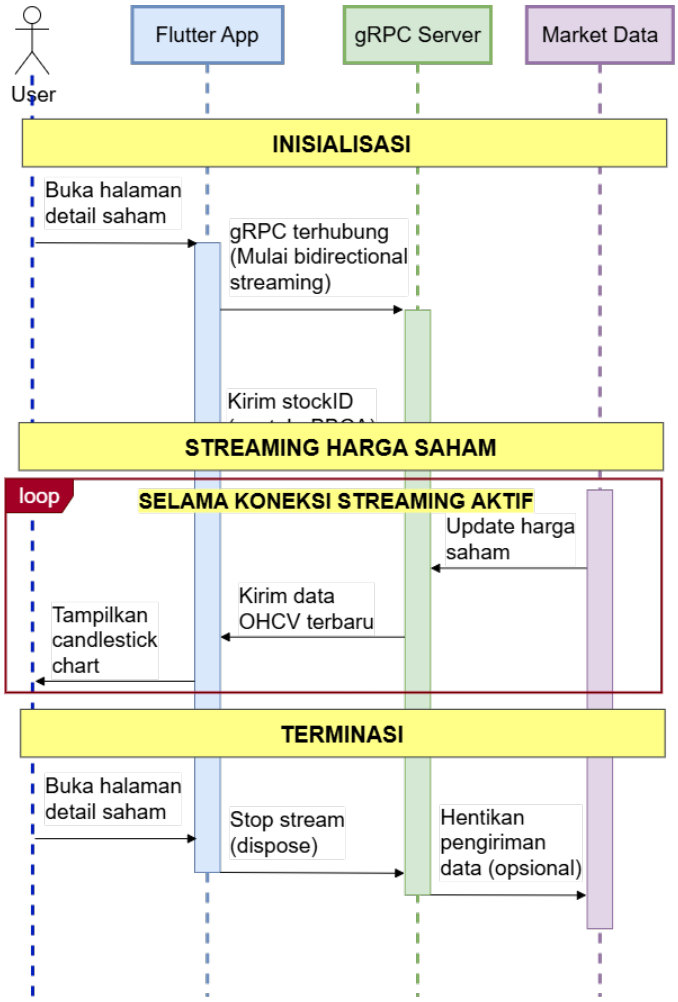


Gbr. 11 Use Case Diagram Sistem Perdagangan Saham

Aplikasi perdagangan saham perlu memusatkan fungsionalitasnya pada kegiatan utama, yaitu transaksi jual dan beli saham sebagai inti dari aktivitas pasar modal. Dalam use case diagram pada Gbr. 11, aktivitas ini direpresentasikan melalui tiga aktor utama, yakni Pembeli, Penjual, dan *Stock Data Source Provider*. Ketiganya terlibat dalam berbagai proses sistem, seperti pembuatan pesanan, pembaruan status transaksi, akses harga saham secara *real-time*, serta penerimaan umpan balik transaksi. Interaksi antar aktor dan sistem difasilitasi melalui komunikasi dua arah (*bidirectional streaming*) menggunakan *framework* gRPC, yang dijelaskan lebih lanjut pada bagian berikutnya.

1) Harga Saham Real-Time

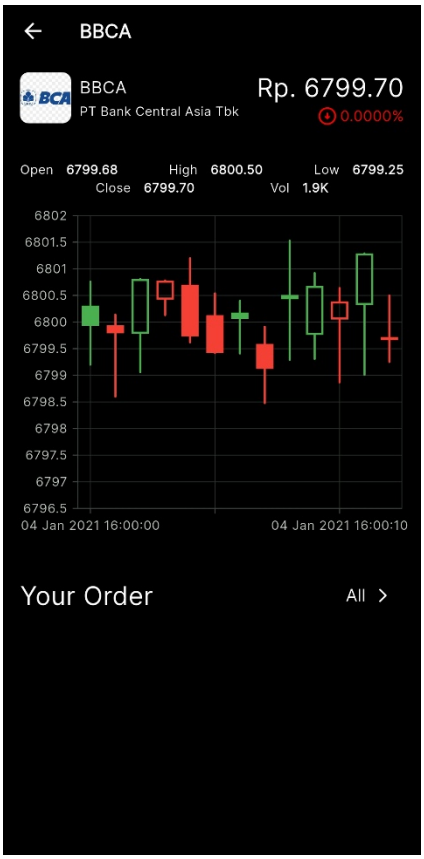
Saat pengguna membuka halaman detail saham, aplikasi langsung membentuk koneksi *streaming* dan mengirimkan *stockID* ke server. Server kemudian menghubungkan *client* ke channel data saham terkait dan mengirimkan data OHLCV secara *real-time*.



Gbr. 12 Sequenced Diagram Alur Real-Time

Gbr. 12 menggambarkan sequence diagram yang menunjukkan alur komunikasi *real-time* dalam sistem. Diagram ini dipilih karena mampu memvisualisasikan urutan interaksi antar komponen secara kronologis, mulai dari koneksi hingga penghentian *streaming*. Proses *real-time* dilakukan melalui komunikasi *bidirectional streaming* menggunakan gRPC antara aplikasi Flutter sebagai klien dan server. Ketika pengguna membuka halaman detail saham, klien membentuk koneksi *streaming* dan mengirimkan *stockID*, seperti “BBCA”, ke server. Server kemudian meneruskan permintaan ini ke penyedia data pasar yang secara berkala memperbarui harga saham dalam format *OHLCV (Open, High, Low, Close, Volume)*. Data ini dikirim secara terus-menerus ke klien dan divisualisasikan dalam bentuk grafik *candlestick*. Pengguna dapat berpindah ke saham lain cukup dengan mengirim *stockID*

baru tanpa memutus koneksi, sehingga lebih efisien. Koneksi akan berhenti otomatis saat pengguna keluar dari halaman. Pendekatan ini mendukung penyajian data harga saham yang akurat, responsif, dan efisien tanpa perlu melakukan *polling* berulang.

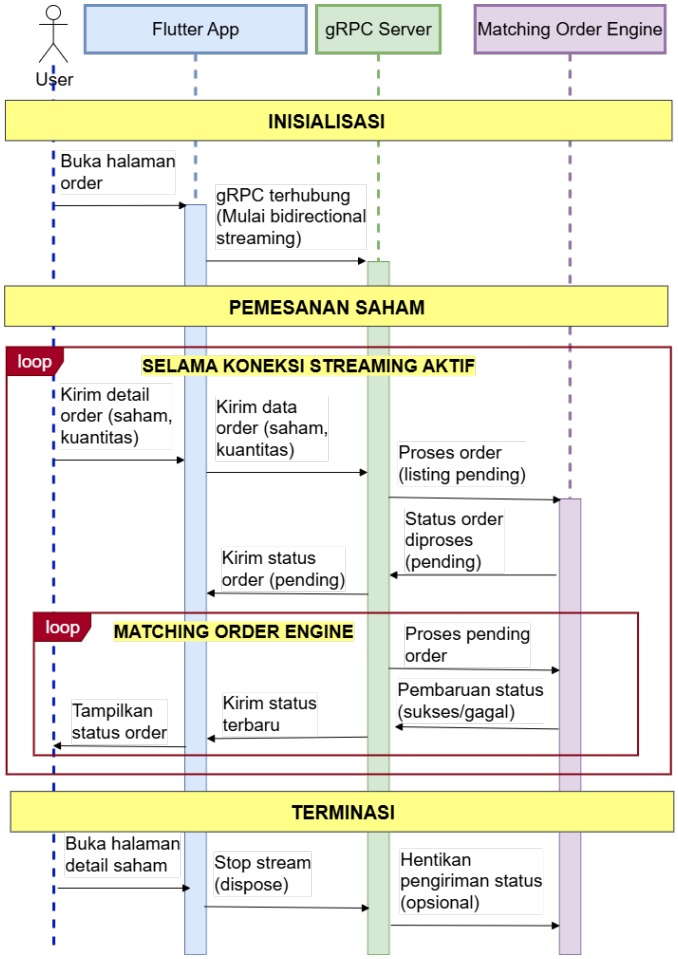


Gbr. 13 User Interface Streaming Harga Saham

Pada Gbr. 13, menunjukkan implementasi sisi klien pada fitur *bidirectional streaming* harga saham BBCA yang divisualisasikan melalui grafik *candlestick*. Grafik ini menyajikan data *open, high, low, close*, dan *volume* dari harga saham yang telah di-*subscribe*. Selama pengguna berada pada halaman detail harga saham, proses *streaming* berlangsung secara kontinu. Sebaliknya, ketika pengguna berpindah dari halaman tersebut, sistem akan secara otomatis menghentikan langganan *streaming* yang sedang aktif.

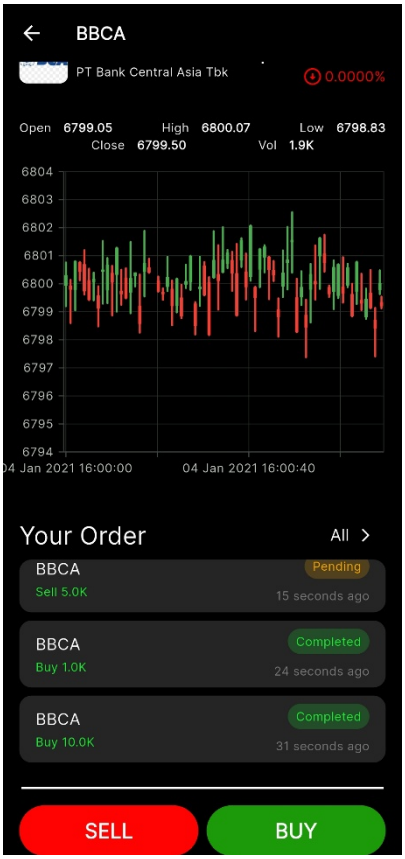
2) Transaksi Jual Beli Harga Saham

Saat pengguna ingin membeli saham, data transaksi dikirim melalui *streaming* yang sama. Server akan menampilkan status awal "*pending*" dan memperbarui status transaksi secara langsung setelah proses *matching*.



Gbr. 14 Sequenced Diagram Alur Implementasi Pembelian Saham Real-Time

Pada Gbr. 14 menunjukkan *sequence diagram* dari implementasi pembelian saham secara *real-time* menggunakan mekanisme gRPC *bidirectional streaming* antara aplikasi Flutter sebagai klien dan server. Ketika pengguna membuka halaman daftar saham, aplikasi memulai koneksi *streaming* ke server. Selama koneksi berlangsung, pengguna dapat mengirim detail pembelian yang kemudian diproses oleh *Matching Order Engine*. Status pesanan akan diperbarui secara *real-time* melalui *stream* yang sama, sehingga pengguna melihat hasil transaksi tanpa perlu memuat ulang halaman. Pendekatan ini mengurangi latensi dan meningkatkan responsivitas aplikasi. Setelah pengguna keluar dari halaman, koneksi *streaming* ditutup secara otomatis untuk menjaga efisiensi komunikasi.



Gbr. 15 Tampilan Proses Jual Beli Saham

Pada Gbr. 15 memperlihatkan antarmuka pengguna untuk proses jual beli saham yang mengintegrasikan fitur pemesanan dengan tampilan harga saham secara *real-time*. Transaksi yang dilakukan pengguna akan langsung dikirim dan ditampilkan dari server. Setelah pesanan diverifikasi oleh server, status transaksi akan otomatis muncul dan diperbarui pada sisi klien melalui sistem *matching order engine*.

C. Implementasi Serialization

Serialization dan *deserialization* diimplementasikan menggunakan Protocol Buffers, sebagaimana dirancang pada Gbr. 6 dan Gbr. 7. Implementasi pada Gbr. 28 dan Gbr. 29 memperlihatkan hasil kompilasi file .proto menjadi *struct* Go yang siap diserialisasi ke format biner dan JSON.

D. Implementasi Real-Time Messaging

Dengan memanfaatkan HTTP/2 dan mekanisme *bidirectional streaming*, server dapat mengirimkan data harga saham secara kontinu kepada klien yang telah melakukan *subscribe*. Gbr. 13 hingga Gbr. 19 menggambarkan penggunaan *channel broadcaster* untuk menyalurkan data secara serentak ke banyak klien, sehingga mengubah komunikasi gRPC yang semula bersifat *unicast* menjadi *multicast* dengan bantuan *map* dan *goroutine*. Setiap klien yang berlangganan *stockID* tertentu akan menerima pembaruan harga secara bersamaan. Selanjutnya, antarmuka pengguna akan menampilkan grafik *candlestick* secara *real-time*, dan koneksi *streaming* akan otomatis ditutup saat pengguna berpindah halaman.

E. Uji Serialization

Pengujian serialisasi ini mencakup dua aspek utama, yaitu ukuran hasil serialisasi dalam satuan byte dan durasi proses dalam satuan nanodetik.

TABEL III
HASIL UJI SERIALISASI

Iterasi	Ukuran Proto	Durasi Proto (ns)	Ukuran JSON	Durasi JSON (ns)
1	33	1272	117	1943
2	33	561	118	1683
3	33	491	118	1442
4	33	441	116	1403
5	33	430	118	1472
6	33	421	117	1533
7	33	421	118	1262
8	33	411	118	1302
9	33	421	118	1262
10	33	421	118	1263

Pada Tabel III menunjukkan hasil uji serialisasi dengan melakukan perbandingan antara metode serialisasi Protocol Buffers (Proto) dan JSON untuk mengukur efisiensi komunikasi data pada aplikasi perdagangan saham. Pengujian melibatkan pengukuran ukuran data hasil serialisasi serta waktu prosesnya dalam beberapa iterasi. Hasilnya menunjukkan bahwa Proto menghasilkan data yang lebih kecil, yaitu 33 byte, dibandingkan JSON yang mencapai 117–118 byte. Dari sisi kecepatan, Proto juga lebih unggul dengan waktu rata-rata 421 nanodetik, sedangkan JSON membutuhkan sekitar 1262 nanodetik. Dengan demikian, Protocol Buffers dinilai lebih efisien dan sesuai untuk aplikasi yang membutuhkan latensi rendah dan performa tinggi.

F. Uji Beban Aplikasi

Pengujian beban aplikasi dilakukan dengan metode *blackbox* pada lingkungan cloud menggunakan server berspesifikasi 2 vCPU AMD EPYC 7002 dan 2 GB RAM. Alat uji yang digunakan adalah Grafana k6, dengan skrip push data harga saham dijalankan pada server identik. Pemilihan metode *blackbox* bertujuan untuk meminimalkan gangguan eksternal sehingga hasil pengujian lebih merepresentasikan performa sistem secara objektif.

TABEL IV
HASIL ITERASI PENGUJIAN BEBAN

Iterasi	Avg (ms)	Max (ms)	Med (ms)	Min (ms)	Total	Success
1	217	967	98	27	9857	9857
2	237	2400	111	27	9701	9701
3	234	3400	115	27	9725	9725
4	247	1000	123	27	9621	9621
5	265	1000	128	27	9488	9488

Pengujian performa sistem dalam menangani komunikasi *real-time* melalui fitur *streaming* harga saham berbasis gRPC

dilakukan sebanyak 5 kali dengan durasi masing-masing 30 menit. Hasil pengujian pada Tabel VI menunjukkan bahwa waktu respons rata-rata (*avg*) konsisten pada angka 4 ms, hal ini mencerminkan stabilitas sistem. Waktu respons maksimum (*max*) tercatat sebesar 57 ms pada iterasi ketiga, namun masih berada dalam batas toleransi untuk kebutuhan *real-time*. Sementara itu, nilai *median (med)* tetap pada 3 ms, hal ini menunjukkan bahwa sebagian besar permintaan diproses dengan cepat. Waktu respons *minimum (min)* mencapai 0.0004 ms, ini mencerminkan efisiensi tinggi dalam proses tertentu. Selain itu, seluruh permintaan berhasil diproses tanpa kesalahan, yang mengindikasikan tingkat reliabilitas sistem yang sangat baik.

Gbr. 16 Grafik Hasil Iterasi Pengujian Beban

Grafik batang pada Gbr. 16 menunjukkan perbandingan waktu respons berdasarkan nilai rata-rata, maksimum, median, dan minimum untuk setiap iterasi selama 30 menit. Batang biru gelap menunjukkan waktu rata-rata, batang oranye menunjukkan waktu maksimum, batang hijau menunjukkan median, dan batang biru muda menunjukkan waktu minimum. Visualisasi ini membuat lebih mudah untuk memahami variasi waktu respons untuk setiap pengukuran.

Berdasarkan penelitian dalam jurnal-jurnal terkait [12], batas wajar latensi sistem yang masih dapat diterima oleh pengguna adalah maksimal 250 milidetik (ms). Dengan demikian, latensi kurang dari 250 ms tergolong baik dan optimal. Berikut ini adalah rincian penghitungan total avg latensi sistem perdagangan saham berdasarkan rumus *Weighted Average* [13]:

$$Latensi\ Rata-Rata\ Total\ (ms) = \frac{\sum_{i=1}^{n} \times (avg_i \times jumlah\ request_i)}{\sum_{1=1}^{n} \times jumlah\ request_i)}$$

$$= \frac{(4 \times 904{,}806) + (4 \times 913{,}410) + (4 \times 914{,}436) + (4 \times 911{,}574) + (4 \times 936{,}342)}{904{,}806 + 913{,}410 + 914{,}436 + 911{,}574 + 936{,}342}$$

$$= \frac{18{,}322{,}272}{4{,}580{,}568} = 4\ ms$$

Hasil pengujian menunjukkan bahwa penerapan gRPC pada sistem perdagangan saham mampu memberikan performa komunikasi *real-time* yang cepat, stabil, dan andal. Sistem juga terbukti mampu menangani permintaan dalam jumlah besar secara konsisten tanpa mengalami *error*, dengan waktu respons yang tetap berada di bawah 250 milidetik. Temuan ini mengindikasikan bahwa arsitektur yang digunakan telah memenuhi kebutuhan sistem perdagangan saham berbasis *real-time*.

G. Komparasi dengan Aplikasi Sejenis

Penelitian ini membandingkan aplikasi perdagangan saham yang menggunakan gRPC dengan aplikasi konvensional berbasis REST API untuk menilai perbedaan dari sisi teknis, performa sistem, dan efisiensi komunikasi data *real-time*. Aplikasi pembanding, yaitu Wealth's Secret dan iInvest yang masih mengandalkan pendekatan tradisional seperti *pull* data melalui REST atau *push* data via *middleware*, yang memiliki keterbatasan dalam hal latensi, efisiensi *bandwidth*, dan komunikasi dua arah [15], [8]. Sebaliknya, aplikasi Zen Stock yang dikembangkan dalam penelitian ini menggunakan gRPC dengan dukungan *bidirectional streaming* melalui protokol HTTP/2 dan format Protocol Buffers. Pendekatan ini menawarkan komunikasi data yang lebih cepat, ringan, dan efisien, serta mampu mendukung transaksi saham secara *real-time secara* lebih optimal.

TABEL V
KOMPARASI DENGAN APLIKASI SEJENIS

Aspek	Wealth Secret	iInvest	Zen Stock
Teknologi	REST API, Yahoo Finance, Firebase	REST API	gRPC, Yahoo Finance
Protokol Komunikasi	HTTP /1.1	HTTP /1.1	HTTP/2
Metode Pengiriman	Pull data (klien mengambil info saham secara *real-time*)	Pull data via REST	*Bidirectional Streaming*
Format Data	JSON	JSON	Protocol Buffers (binary)
Tipe Komunikasi	*One-way*	*One-way*	*Full Duplex* (*client* ⇄ server secara simultan)

Berdasarkan hasil komparasi pada Tabel V, aplikasi Zen Stock yang dikembangkan menggunakan gRPC menunjukkan keunggulan signifikan dibandingkan aplikasi lain yang masih mengandalkan REST API. Zen Stock lebih efisien dalam komunikasi data karena memanfaatkan protokol HTTP/2 dan format Protocol Buffers yang lebih ringan dibandingkan JSON. Selain itu, fitur *bidirectional streaming* mendukung komunikasi dua arah secara simultan antara klien dan server, sehingga dapat meningkatkan responsivitas sistem. Dari sisi kinerja, gRPC mampu menangani banyak koneksi secara bersamaan dengan latensi rendah, sehingga lebih unggul dalam skenario beban tinggi. Arsitektur Zen Stock juga lebih modern dan mudah dipelihara melalui penggunaan file .proto yang mendukung pembuatan kode secara otomatis. Secara keseluruhan, gRPC menjadikan Zen Stock lebih siap untuk diimplementasikan dalam sistem perdagangan saham yang

menuntut kecepatan, efisiensi, dan komunikasi *real-time* antar entitas.

IV. KESIMPULAN

Penelitian ini berhasil merancang dan mengimplementasikan aplikasi perdagangan saham berbasis Android dengan memanfaatkan *framework* gRPC untuk komunikasi *real-time*. Proses pengembangan dilakukan melalui tahapan analisis, perancangan, implementasi, dan pengujian. Aplikasi dibangun dengan arsitektur *client-server* menggunakan Flutter pada sisi klien dan Go pada sisi server, serta mendukung fitur utama seperti registrasi, *login*, transaksi jual beli saham, dan pemantauan harga. Penerapan *bidirectional streaming* ini mendukung komunikasi dua arah antara klien dan server secara simultan tanpa *polling*, sehingga informasi harga saham dapat diterima secara *real-time* dengan lebih efisien.

Hasil pengujian menunjukkan bahwa penggunaan Protocol Buffers lebih unggul dibandingkan JSON dalam hal ukuran data dan kecepatan serialisasi. Selain itu, hasil uji beban dengan 20 *virtual users* menunjukkan performa sistem yang stabil dan responsif dengan latensi rata-rata 4 ms. Secara keseluruhan, sistem yang dikembangkan terbukti memiliki performa tinggi, efisiensi komunikasi yang baik, dan stabil dalam menangani transaksi *real-time*, sesuai dengan tujuan penelitian.

REFERENSI

[1] X. Wu, H. Chen, J. Wang, L. Troiano, V. Loia, dan H. Fujita, "Adaptive stock trading strategies with deep reinforcement learning methods," *Inf Sci (N Y)*, vol. 538, hlm. 142–158, Okt 2020, doi: 10.1016/j.ins.2020.05.066.

[2] J. M. T. Wu, L. Sun, G. Srivastava, dan J. C. W. Lin, "A Novel Synergetic LSTM-GA Stock Trading Suggestion System in Internet of Things," *Mobile Information Systems*, vol. 2021, 2021, doi: 10.1155/2021/6706345.

[3] T. Singh, R. Kalra, S. Mishra, Satakshi, dan M. Kumar, "An efficient real-time stock prediction exploiting incremental learning and deep learning," *Evolving Systems*, vol. 14, no. 6, hlm. 919–937, Des 2023, doi: 10.1007/s12530-022-09481-x.

[4] Abdurohim, "Analisa transaksi perdagangan saham pada pasar sekunder," Jurnal Ekonomi, Bisnis, Manajemen dan Akuntansi, vol. 18, no. 1, may 2021. [Online]. Available: https://jurnal.iain bone.ac.id/index.php/ieb/article/view/3760/1468

[5] "gRPC Technology White Paper," 2020.

[6] G. team, "The Go Programming Language — go.dev," https://go.dev/, [Accessed 20-10-2024].

[7] Group, "PostgreSQL — postgresql.org," https://www.postgresql.org/, [Accessed 20-10-2024]

[8] A. Rahim dan E. Shaubari, "The Design and Development of iInvest Application," *Applied Information Technology And Computer Science*, vol. 4, no. 1, hlm. 598–613, 2023, doi: 10.30880/aitcs.2023.04.01.034.

[9] Y. M. Tripathi dan M. P. Modi, "Real-Time Chat Application."

[10] B. Setiawan, D. P. Nugraha, A. Irawan, R. J. Nathan, dan Z. Zoltan, "User innovativeness and fintech adoption in indonesia," *Journal of Open Innovation: Technology, Market, and Complexity*, vol. 7, no. 3, Sep 2021, doi: 10.3390/joitmc7030188.

[11] L. L. Chong, H. B. Ong, dan S. H. Tan, "Acceptability of mobile stock trading application: A study of young investors in Malaysia," *Technol Soc*, vol. 64, Feb 2021, doi: 10.1016/j.techsoc.2020.101497.

[12] M. Bolanowski *dkk.*, "Eficiency of REST and gRPC realizing communication tasks in microservice-based ecosystems." [Daring]. Tersedia pada: https://orcid.org/0000-0002-

[13] D. S. Voss, "Aggregation," *Encyclopedia of Social Measurement, Three-Volume Set*, vol. 1, hlm. 33–42, Jan 2005, doi: 10.1016/B0-12-369398-5/00044-X.

[14] C. Attig, N. Rauh, T. Franke, dan J. F. Krems, "System latency guidelines then and now – Is zero latency really considered necessary?," dalam *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, Springer Verlag, 2017, hlm. 3–14. doi: 10.1007/978-3-319-58475-1_1.

[15] Mr. D. Tripathi, "WEALTH'S SECRET," *INTERANTIONAL JOURNAL OF SCIENTIFIC RESEARCH IN ENGINEERING AND MANAGEMENT*, vol. 08, no. 05, hlm. 1–5, Jun 2024, doi: 10.55041/IJSREM35276.