Terms of Reference
Hiliriset AI Agent Social Commerce dan Ride Fleet

Ver0.2

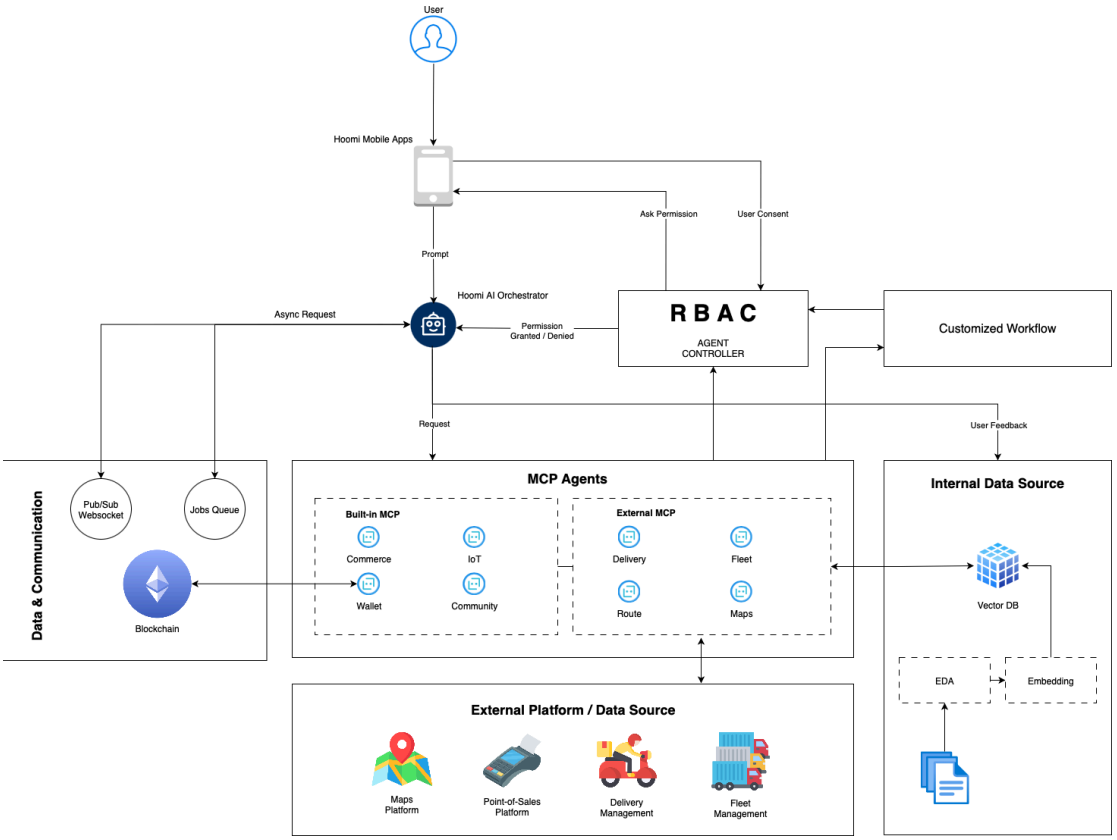


Figure 1. Hoomi AI Agent Orchestrator Infrastructure

A. PERAN DAN TANGGUNG JAWAB MASING-MASING PIHAK

1. Tim Hoomi:
    a. Membangun platform aplikasi integrator dalam bentuk webapp diperuntukan implementasi AI agent dalam bentuk MCP

ataupun API services yang dapat digunakan oleh pihak UI dan juga pihak aplikasi 3rd party.

b. Mengintegrasikan produk akhir dari AI agent kedalam aplikasi Hoomi untuk dipergunakan sebagai *proof-of-concept*.
c. Menyediakan server infrastruktur untuk pengembangan *(development)* yang spesifikasinya disesuaikan berdasarkan teknologi pendukung untuk pengembangan AI agent.
d. Mengembangkan smart contract yang diperuntukan sebagai mediator transaksi antara platform Hoomi, pengguna akhir dan merchant.
e. Menyediakan sumber data yang akan digunakan sebagai data training untuk pengembangan data model yang dapat mengefisiensikan rute pengiriman dan juga rekomendasi produk.

2. Tim Universitas Indonesia:
    a. Pengembangan MCP terkait dengan service pendukung yang digunakan pada *AI Agent Orchestrator Hoomi* untuk melakukan channeling yang disesuaikan dengan *prompt* dari user.
    b. Implementasi *Guardrails* atau pun *MCP human-in-the-loop* untuk melimitasi otoritas agent dalam melakukan aksi yang memerlukan otorisasi pengguna.
    c. Melakukan optimasi kinerja MCP untuk mendapatkan respon yang sesuai dengan prompt dari pengguna akhir (*user).*
    d. Pengembangan modul workflow AI khususnya untuk fitur transportasi dan pengantaran barang / makanan yang dimiliki oleh Hoomi.
    e. Pengembangan fitur pada AI Orchestrator khususnya di bagian *chain-of-thought prompting* dalam bentuk task untuk menyediakan AI agent yang lebih baik dalam menyelesaikan suatu masalah langkah demi langkah.

B. MEKANISME IMPLEMENTASI, VALIDASI TEKNOLOGI, DAN ADOPSI

1. Platform dan Infrastruktur

- **Platform Cloud**: menggunakan AWS sebagai platform cloud untuk hosting infrastruktur. Layanan seperti *AWS ECS (Elastic Container Service)* ataupun *AWS EKS (Elastic Kubernetes Service)* untuk orkestrasi kontainer dan AWS S3

untuk penyimpanan akan digunakan untuk menjaga skalabilitas dan ketersediaan data.

- **Blockchain**: Mengimplementasikan platform *Ethereum L2 BASE* dengan menggunakan node validator dan non-validator untuk memvalidasi transaksi dan mendukung proses smart contract yang terdesentralisasi. Sistem ini memerlukan pengaturan node dan validator:
  - **Validator Nodes**: Memvalidasi transaksi dan menjalankan smart contract.
  - **Non-Validator Nodes**: Menangani aktivitas transaksi namun tidak melakukan validasi.
- **AI** : Untuk orkestrasi agent, *tool calling*, implementasi *human-in-the-loop (HITL)* pada agent, dan penggunaan augmentasi *vector embedding* menggunakan RAG, dengan memberikan rekomendasi produk yang sekiranya diminati oleh pengguna akhir. Menggunakan *AWS Bedrock Agent Core* untuk membuat agent model *tool calling*. Untuk lebih rinci spesifikasi agent yang diperlukan ialah sebagai berikut:
  - Orkestrasi Agent: Agent channeling menggunakan *tool agent* yang dapat diimplementasikan menggunakan library seperti Strands, LangGraph ataupun CrewAI.
  - Database Vector: Menyimpan data vektor untuk embedding menggunakan *pgvector* yang selanjutnya digunakan untuk embedding.
  - Agent: Pengembangan agent dapat menggunakan Amazon Agent Core dengan model agent yang tersedia di Agent Core.
  - *Guardrails HITL*: Implementasi *guardrails* yang dapat memberikan response agent lebih terstruktur dan membatasi akses agent untuk data penting yang memerlukan *approval* kepada pengguna terlebih dahulu.
  - *Adaptive Agent Response*: Menyajikan tampilan respon agent yang user-friendly dan adaptif menyesuaikan tipe data respon untuk dapat digunakan interaksi dengan pengguna.

- Personalisasi: Collaborative filtering dan embedding. Data disimpan di S3/DynamoDB. Menggunakan Amazon Bedrock untuk vector embedding.
- Pengiriman dan Estimasi Harga: Regression, GNNs, RL. Data disimpan pada S3 + Aurora/Redshift.

2. Arsitektur Lapisan Sistem

Lapisan Pengguna (User Layer) pada Aplikasi Mobile Hoomi

Pengguna berinteraksi dengan aplikasi Hoomi yang menyediakan prompt untuk mengakses berbagai layanan, seperti personalisasi produk, prediksi permintaan, dan manajemen inventaris. Aplikasi ini terhubung dengan Hoomi AI Orchestrator yang akan mengatur permintaan dan alur kerja berdasarkan izin yang diberikan oleh pengguna (melalui RBAC atau Role-Based Access Control) dan human controlled restrictive requests melalui *Guardrails HITL*.

Lapisan Permintaan dan Pengolahan (Request and Processing Layer)

- Asynchronous Request: Setiap permintaan yang dikirimkan dari aplikasi mobile pengguna diteruskan ke sistem menggunakan Asynchronous Requests untuk diproses lebih lanjut oleh Hoomi AI Orchestrator.
- AI Orchestrator dan Pengaturan Izin (*Guardrails HITL*): Orkestrator akan menentukan apakah izin pengguna diberikan atau ditolak sebelum memproses permintaan lebih lanjut.

Lapisan Agent MCP (MCP Agents Layer)

- Internal MCP : Layanan yang terintegrasi secara internal untuk mendukung transaksi dan interaksi dalam aplikasi seperti Delivery, IoT, Commerce, Wallet, dan Community.
- External MCP: Agen eksternal yang terhubung dengan platform lain seperti aplikasi produktivitas rekanan. Contohnya untuk aplikasi document processing ataupun task management, dan lain-lain.

Lapisan Data dan Komunikasi (Data and Communication Layer)

- Blockchain: Untuk memastikan keamanan dan transparansi dalam transaksi, blockchain digunakan untuk memvalidasi dan menyimpan data transaksi, serta mendukung komunikasi data antar platform.
- WebSocket, Pub/Sub dan Job Queue: Digunakan untuk komunikasi real-time dan manajemen antrian pekerjaan yang datang dari aplikasi dan agen-agen.

Lapisan Sumber Data (Data Sources Layer)

- Sumber Data Internal: Data transaksi, rute perjalanan, dan pemesanan disimpan dalam Vector DB dan digunakan oleh agen AI untuk analisis dan rekomendasi berbasis data dan juga membantu perencanaan pengiriman, estimasi harga, dan manajemen armada kendaraan.
- Sumber Data Eksternal: Data yang berasal dari rekanan existing untuk pemesanan dan juga *dispatch*. Untuk data ini masih dalam tahap pengumpulan karena kurangnya jumlah data dan komponen yang dibutuhkan tidak tersedia.
- Data ini diintegrasikan dengan MCP Agents yang mengelola *dispatch* dan *merchant*.

3. Konfigurasi Smart Contract dan Pengelolaan API

- smart contracts yang dapat mengelola transaksi otomatis dan memenuhi aturan bisnis (misalnya, pengelolaan inventaris otomatis atau estimasi harga).
- Menggunakan Solidity untuk mengembangkan kontrak pintar pada platform *Ethereum L2 BASE*.

4. Pelatihan dan Pengujian Model AI

Pelatihan Model: Menggunakan *AWS Bedrock* untuk melatih model AI yang dibutuhkan, seperti model prediksi permintaan atau personalisasi produk berdasarkan data historis.

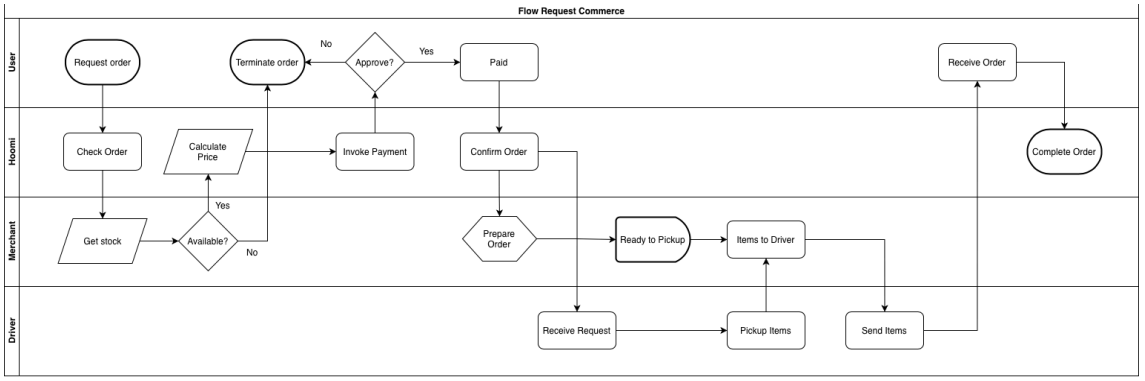
Pengujian:

- Menggunakan *AWS Distributed Load Testing* atau *AWS Load Balancer* untuk menguji sistem dengan beban tinggi.
- Menggunakan *AWS Bedrock Guardrails* untuk pengamanan response dan access control (RBAC) pada AI agent.

- Lakukan pengujian unit dan integrasi pada smart contracts dan algoritma AI dengan platform seperti *Locus*, *Truffle Suite* atau *Corda Test Framework*.

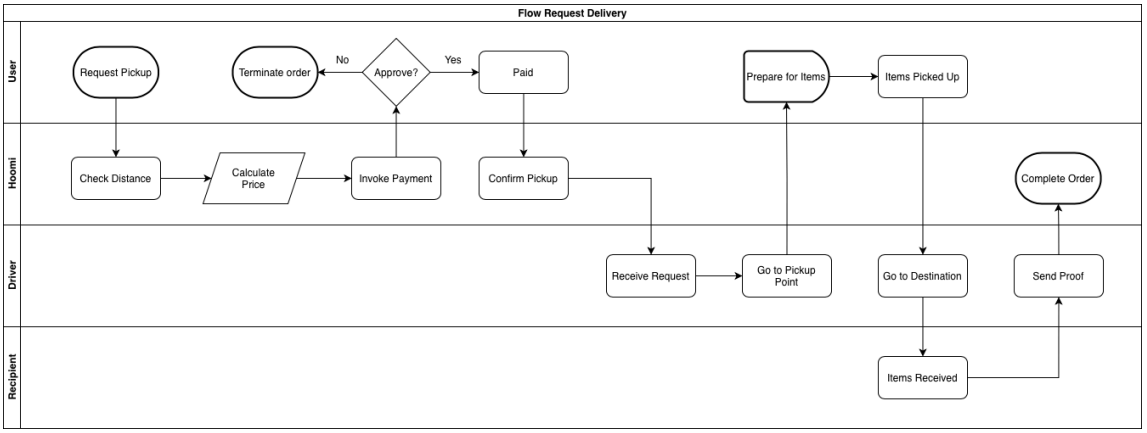
C. SPESIFIKASI ALUR DAN PROSES BISNIS

Untuk projek hiliriset ini, kami memerlukan agent yang digunakan untuk proses jual beli dan juga pengantaran yang menghubungkan tiga pihak. Pihak penjual, pembeli dan juga pengantar barang. Dengan menggunakan satu agent yang bertindak sebagai orkestrator, diharapkan dapat mengefisiensikan proses bisnis terotomasi. Berikut adalah rincian spesifikasi alur dan proses bisnis:

<table>
<tr><td></td><td colspan="2">Jual Beli Barang</td></tr>
<tr><td></td><td colspan="2"> 
</td></tr>
<tr><td></td><td>Nama Proses</td><td>Deskripsi</td></tr>
<tr><td></td><td>Pencarian Barang Jualan</td><td>● Pengguna dapat menggunakan aplikasi Hoomi ataupun melalui AI agent dengan memilih barang yang dicari oleh pengguna.
● Pengguna memilih barang yang ditampilkan oleh aplikasi Hoomi / AI agent.
● Sistem memerlukan melakukan pengecekan stok sebelum mengirimkan respon kepada pengguna.</td></tr>
<tr><td></td><td>Penentuan Pengiriman</td><td>● Pengguna memasukkan alamat ke dalam aplikasi Hoomi yang dapat disimpan pada menu profil yang sudah ditentukan titik GPS.
● Pengguna juga dapat memasukkan alamat langsung ke AI agent untuk disesuaikan titik GPS secara otomatis oleh API Google Geolocation. Ataupun meminta request melalui browser / aplikasi untuk titik GPS.
● Sistem / AI Agent akan menentukan jarak dari titik</td></tr>
</table>

		toko barang dan alamat tujuan untuk memberikan respon harga. ● Pada aplikasi Hoomi, akan ditampilkan rute perjalanan beserta update titik dari driver. Pada AI Agent akan ditampilkan screenshot map dari rute yang akan dilalui dan terdapat tautan untuk membuka map.
	Pembayaran	● Apabila pengguna melakukan pembayaran melalui aplikasi Hoomi, pengguna dapat langsung memilih menggunakan wallet. ● Untuk pembayaran melalui AI agent, agent akan melakukan panggilan melalui *Guardrails MCP Wallet* untuk meminta izin terlebih dahulu. ● Jika sudah melakukan pembayaran, sistem akan mengirimkan konfirmasi ke sistem delivery internal Hoomi dan juga mengirimkan notifikasi kepada penjual. ● Sistem akan otomatis memilih driver terdekat untuk pengantaran.
	Persiapan Barang dan Penjemputan	● Penjual mempersiapkan barang yang akan diserahkan kepada driver. ● Saat barang diterima oleh driver, driver akan melakukan foto dari barang dan juga melakukan konfirmasi. ● Sistem akan memberikan notifikasi kepada pengguna untuk persiapan penerimaan barang.
	Pengantaran Barang dan Penerimaan	● Driver melakukan pengantaran dan akan terupdate oleh sistem secara *real-time* yang dapat dimonitor oleh penerima barang dan pengirim. ● Saat barang sampai dan diterima oleh penerima, driver akan melakukan update pada aplikasi dan sistem mengirimkan update kepada penerima dan pengirim.

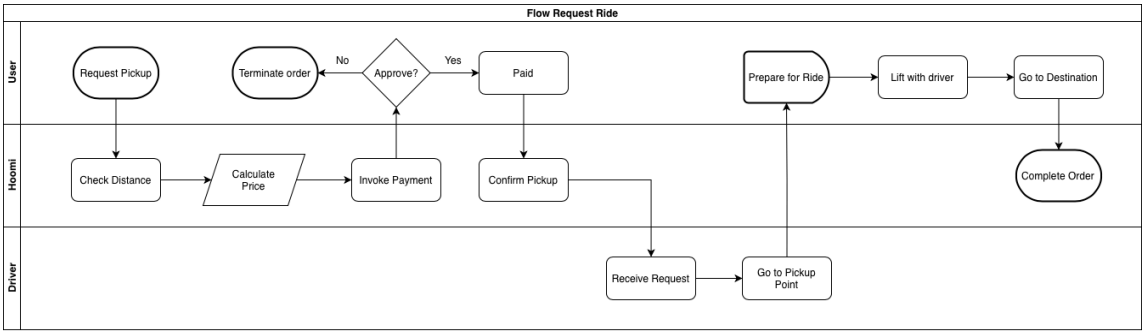
Pengantaran Barang



Nama Proses	Deskripsi
Penentuan Tujuan Pengiriman	● Pengguna memasukkan alamat asal untuk penjemputan. Aplikasi ataupun AI agent dapat meminta lokasi pengguna dengan fitur GPS. ● Pengguna memasukkan alamat tujuan untuk pengantaran. Aplikasi ataupun AI agent dapat meminta lokasi pengguna dengan fitur GPS. ● Sistem / AI Agent akan menentukan jarak dari titik penjemputan dan alamat tujuan untuk memberikan respon harga. ● Pada aplikasi Hoomi, akan ditampilkan rute perjalanan beserta update titik dari driver. Pada AI Agent akan ditampilkan screenshot map dari rute yang akan dilalui dan terdapat tautan untuk membuka map.
Pembayaran	● Apabila pengguna melakukan pembayaran melalui aplikasi Hoomi, pengguna dapat langsung memilih menggunakan wallet. ● Untuk pembayaran melalui AI agent, agent akan melakukan panggilan melalui *Guardrails MCP Wallet* untuk meminta izin terlebih dahulu. ● Jika sudah melakukan pembayaran, sistem akan mengirimkan konfirmasi ke sistem delivery internal Hoomi dan juga mengirimkan notifikasi kepada penjual. ● Sistem akan otomatis memilih driver terdekat untuk pengantaran.
Persiapan Barang dan Penjemputan	● Pengirim mempersiapkan barang yang akan diserahkan kepada driver. ● Saat barang diterima oleh driver, driver akan melakukan foto dari barang dan juga melakukan konfirmasi.

		● Sistem akan memberikan notifikasi kepada penerima untuk persiapan penerimaan barang.
	Pengantaran Barang dan Penerimaan	● Driver melakukan pengantaran dan akan terupdate oleh sistem secara *real-time* yang dapat dimonitor oleh penerima barang dan pengirim. ● Saat barang sampai dan diterima oleh penerima, driver akan melakukan update pada aplikasi dan sistem mengirimkan update kepada penerima dan pengirim.

Antar Jemput Penumpang



	Nama Proses	**Deskripsi**
	Penentuan Tujuan	● Pengguna memasukkan alamat asal untuk penjemputan. Aplikasi ataupun AI agent dapat meminta lokasi pengguna dengan fitur GPS. ● Pengguna memasukkan alamat tujuan untuk pengantaran. Aplikasi ataupun AI agent dapat meminta lokasi pengguna dengan fitur GPS. ● Sistem / AI Agent akan menentukan jarak dari titik penjemputan dan alamat tujuan untuk memberikan respon harga. ● Pada aplikasi Hoomi, akan ditampilkan rute perjalanan beserta update titik dari driver. Pada AI Agent akan ditampilkan screenshot map dari rute yang akan dilalui dan terdapat tautan untuk membuka map.
	Pembayaran	● Apabila pengguna melakukan pembayaran melalui aplikasi Hoomi, pengguna dapat langsung memilih menggunakan wallet. ● Untuk pembayaran melalui AI agent, agent akan melakukan panggilan melalui *Guardrails MCP Wallet* untuk meminta izin terlebih dahulu. ● Jika sudah melakukan pembayaran, sistem akan

		mengirimkan konfirmasi ke sistem kendaraan internal Hoomi dan juga mengirimkan notifikasi kepada pengguna. ● Sistem akan otomatis memilih driver terdekat untuk pengantaran.
	Penjemputan	● Driver melakukan penjemputan ke titik lokasi penjemputan. ● Sistem menginfokan pemesan saat driver sudah mendekati titik penjemputan. ● Pemesan bersama driver ke lokasi tujuan.
	Pengantaran ke Tujuan	● Pengendara dengan driver sampai tujuan. ● Sistem akan terotomatis mendeteksi titik lokasi tujuan melalui smartphone driver. ● Driver mengakhiri pengantaran dengan menyelesaikan pemesanan pada aplikasi Hoomi.

D. INDIKATOR CAPAIAN DAN METODE VALIDASI

1. Indikator Pencapaian

Keberhasilan Integrasi Sistem:

- Aplikasi Hoomi dapat mengakses layanan AI Agent melalui MCP maupun API Services.
- Orkestrator mampu menyalurkan permintaan (prompt) sesuai izin pengguna (RBAC) menggunakan *Guardrails* yang disediakan oleh *Amazon Bedrock*.
- Persentase *success rate* integrasi API minimal 95% pada pengujian end-to-end.

Kinerja Model AI:

- Agentic: Memiliki satu Agent orchestrator yang akan mendistribusikan permintaan / tasks ke masing-masing agent. Agent yang akan disiapkan adalah:
    - Storefront Agent / Customer Service (Orchestrator): dapat meminta akses ke data penting atas izin pengguna dan melakukan *channeling* ke AI agent pembantu.
    - Dispatch Agent: Menghubungkan antara pengguna dengan rekanan *driver* untuk melakukan pekerjaan pengantaran.

    - Merchant Agent: Menghubungkan antara pengguna, *driver*, dan juga rekanan untuk memproses pemesanan.
- Izin Terkontrol: Menggunakan *Human-in-the-loop (HITL)* melalui *Guardrails* yang dibentuk sebagai MCP ataupun *middleware* untuk data sensitif terbatas saat *tools* memerlukan akses. Permintaan yang diwajibkan untuk menggunakan HITL adalah:
    - Akses data pribadi
    - Akses dompet digital (Wallet)
    - Akses lokasi (Geolocation)
    - Akses perangkat IoT
    - Akses pesan (messages)
- Dynamic Response Rendering: Memberikan respon terstandarisasi sesuai dengan tipe data yang didapatkan dari MCP *tools* ataupun respon generatif dari AI agent.
    - Komponen Utama (Built-in):
        - Komponen Fluid (Komponen yang dapat disusun pada *stack view*):
            - Text: Menghasilkan respon textual hasil dari generative respon AI agent dan text hasil data binding.
            - Image: Menampilkan gambar yang dilengkapi dengan tools image preview untuk zoom
            - Video: Menampilkan image preview dari video dan memiliki fitur layaknya video player.
            - Audio: Menampilkan audio player yang dapat menjalankan suara data audio tersebut.
            - Maps: Menampilkan konten map 2 dimensi yang dinamis untuk dapat di render pada *window* percakapan.
        - Komponen Native:
            - Rich Text Viewer: Menampilkan text yang memiliki format tertentu, contohnya seperti source code ataupun text *markdown*. Dilengkapi dengan fitur *copy to clipboard*.
            - File: Menampilkan nama file dengan icon dari format file tertentu saja.
            - Link: Menampilkan link untuk mendownload atau mengarahkan pengguna ke website external yang bersifat referensi.
            - Table: Menampilkan hasil data dalam bentuk tabular dan memiliki fitur *copy table to clipboard* dalam bentuk csv.

                - Blockquote: Menampilkan text penting yang dapat ditampilkan seperti *kutipan*.
        - Komponen Antarmuka:
            - Button: Dapat memanggil fungsi untuk *tool function* apabila agent memberikan opsi pilihan kepada pengguna.
        - Komponen Wadah (Container):
            - Stack View (Horizontal / Vertical): Bentuk tampilan susunan yang dapat dikostumisasi oleh pembuat MCP internal maupun external yang dapat diisi oleh Komponen Fluid. Setiap komponen memerlukan binding ataupun di definisikan oleh pembuat MCP yang digenerate secara otomatis oleh agent.
        - Komponen Struktur:
            - Horizontal Array
            - Vertical Array
- Personalisasi Produk:
    - Akurasi rekomendasi (precision/recall) minimal 70%.
    - Choice Option Rate meningkat dibandingkan baseline daftar produk.
- Pengiriman & Estimasi Harga:
    - Error estimasi waktu pengiriman < 10%.
    - Error estimasi harga < 5%.

Skalabilitas & Reliabilitas Sistem:

- Sistem mampu menangani minimal 1000 concurrent users dengan latency < 500ms.
- Auto-scaling di AWS ECS/EKS berjalan sesuai threshold CPU/memory.
- Tingkat downtime < 0.1% dalam periode uji.

Keamanan & Transparansi:

- Semua transaksi tercatat di blockchain Ethereum L2 BASE dan tervalidasi oleh node.
- Smart contract lolos pengujian unit & integrasi tanpa *critical vulnerability*.
- *Guardrails* terbukti membatasi akses sesuai peran (no privilege escalation).

2. Metode Validasi

Validasi Integrasi:

- Uji coba end-to-end antara aplikasi mobile Hoomi, Orchestrator, dan MCP Agents (internal/eksternal).
- Monitoring melalui CloudWatch, X-Ray, dan log audit.

Validasi Kinerja Model AI:

- Offline testing:
    - Menggunakan dataset historis → evaluasi dengan metrik akurasi (MAPE, RMSE, Precision, Recall, F1-score).
- Online A/B testing:
    - Bandingkan performa rekomendasi/estimasi dengan baseline sistem.
- Load testing:
    - Gunakan AWS Distributed Load Testing atau Locust untuk simulasi ribuan pengguna.

Validasi Skalabilitas:

- Stress test sistem menggunakan simulasi trafik puncak.
- Uji auto-scaling policy di ECS/EKS.
- Failover test dengan mematikan node untuk memastikan redundansi.

Validasi Keamanan:

- Audit *Guardrails* dan RBAC dengan AWS CloudTrail dan AWS CloudWatch untuk Guardrails Event.
- Uji penetrasi pada smart contracts menggunakan Truffle, MythX, atau Slither.
- Pengujian negative testing → mencoba akses data tanpa izin (harus ditolak).

Validasi Blockchain & Smart Contract:

- Unit test dan integrasi kontrak pintar dengan Truffle Suite atau Hardhat.
- Simulasi transaksi di testnet Ethereum L2 BASE sebelum mainnet deployment.
- Audit eksternal (jika memungkinkan) untuk menguji potensi bug/vulnerability.