202210370311395
Haidar Yusuf
Prodi Informatika

BAB I

PENDAHULUAN

Bab ini menyajikan pengantar komprehensif mengenai penelitian yang dilakukan. Pembahasan diawali dengan pemaparan **latar belakang** yang menguraikan urgensi dan tantangan dalam mengintegrasikan *Large Language Models* (LLM) dengan aplikasi interaktif di Unity. Selanjutnya, dirumuskan serangkaian **rumusan masalah** yang spesifik untuk dijawab melalui penelitian ini. Bab ini juga menetapkan **tujuan penelitian** yang ingin dicapai, mendefinisikan **batasan penelitian** untuk menjaga fokus dan kelayakan studi, serta menyajikan **jadwal pengerjaan** sebagai kerangka waktu pelaksanaan proyek.

1.1 Latar Belakang

Integrasi *Large Language Models* (LLM) seperti model sampri-custom:latest (berbasis Llama 3.2)[1] ke dalam platform interaktif seperti Unity[2] menjanjikan pengalaman pengguna yang kaya dan dinamis. Namun, pengembangan aplikasi *real-time* di Unity menghadapi tantangan signifikan terkait arsitektur *single main thread*-nya. Berdasarkan literatur arsitektur *game engine*, sistem ini beroperasi di dalam sebuah siklus komputasi berulang atau *Game Loop* yang mengeksekusi pembaruan logika dan *rendering* secara berkesinambungan. Gregory (2018) menjelaskan bahwa untuk mempertahankan standar 60 *Frames Per Second* (FPS), seluruh beban komputasi pada *main thread* mutlak harus tuntas dalam ambang batas waktu 16,6 milidetik. Jika operasi I/O jaringan dilakukan secara sinkron atau *blocking*, *thread* utama akan tertahan sepenuhnya demi menunggu respons dari *server* jarak jauh, yang mana kecepatan I/O tersebut jauh lebih lambat dibandingkan siklus CPU[26]. Penahanan *main thread* ini mengakibatkan sistem gagal mengirimkan instruksi grafis (*draw calls*) ke GPU tepat pada waktunya, sehingga memicu lonjakan *frame time* yang secara visual

bermanifestasi sebagai *stuttering* (patah-patah) atau *freeze* (layar membeku). Kondisi tersebut merusak pengalaman pengguna secara keseluruhan, sehingga bagi aplikasi yang bergantung pada komunikasi jaringan *real-time* seperti *chatbot* LLM ini, manajemen komunikasi asinkron yang efektif menjadi sebuah keharusan.

Penelitian ini mengusulkan penerapan *Reactive Pattern* sebagai solusi arsitektural inti untuk mengatasi tantangan komunikasi asinkron di Unity. *Reactive Pattern*, dalam konteks ini, merujuk pada prinsip-prinsip desain perangkat lunak yang memungkinkan sistem untuk merespons *event* secara efisien dan *non-blocking*, menjaga aplikasi tetap responsif meskipun terjadi komunikasi jaringan yang intensif. Pendekatan ini sejalan dengan konsep *event-driven architecture* (EDA) modern[9, 10], yang krusial untuk sistem terdistribusi dan interaktif.

Protokol WebSocket dipilih sebagai media komunikasi utama dengan *backend* LLM karena karakteristik teknis dan ilmiahnya yang mendukung komunikasi *full-duplex* berlatensi rendah melalui koneksi TCP persisten, serta efisien untuk *streaming* data berkelanjutan[5, 20, 7, 8, 11]. Fitur-fitur ini secara inheren lebih cocok untuk interaksi dinamis dengan LLM dibandingkan dengan sifat *request-response* pada protokol HTTP tradisional, bahkan pada versi modernnya seperti HTTP/2 dan HTTP/3[3, 4]. Aspek keamanan WebSocket juga telah dikaji dan menjadi pertimbangan[6]. Ogundeyi dan Yinka-Banjo (2019) juga menekankan bahwa pendekatan HTTP tradisional memiliki keterbatasan fundamental dalam hal *overhead header* yang berlebihan untuk aplikasi *real-time*, sehingga penggunaan teknologi WebSocket menjadi solusi vital untuk mereduksi latensi jaringan secara signifikan melalui koneksi soket tunggal[22]. Fokus utama penelitian ini adalah bagaimana *Reactive Pattern* memungkinkan pemanfaatan optimal dari karakteristik fundamental WebSocket tersebut dalam batasan *main thread* Unity.

Dengan demikian, penelitian ini bertujuan untuk merancang, mengimplementasikan, dan menganalisis efektivitas *Reactive Pattern* dalam mengelola komunikasi WebSocket antara aplikasi Unity dan LLM Llama 3.2. Kontribusi penelitian ini terletak pada penyediaan solusi arsitektural konkret dan

analisis mendalam terhadap implementasi pola tersebut untuk masalah umum dalam pengembangan *game* dan aplikasi interaktif.

1.2 Rumusan Masalah

Berdasarkan latar belakang yang telah diuraikan, penelitian ini bertujuan untuk menjawab pertanyaan-pertanyaan berikut:

a. Bagaimana *Reactive Pattern* dapat diimplementasikan secara efektif dalam lingkungan Unity menggunakan C# untuk mengelola komunikasi WebSocket asinkron dengan *backend* LLM (model sampri-custom:latest berbasis Llama 3.2 via Ollama), guna menjaga responsivitas aplikasi dan memastikan interaksi *real-time* yang optimal?

b. Bagaimana efektivitas implementasi Reactive Pattern tersebut dalam menangani event WebSocket, memproses pesan secara asinkron, dan melakukan sinkronisasi kembali ke main thread Unity untuk mengorkestrasi output audio (TTS) dan visual (animasi karakter)?

c. Mengapa karakteristik arsitektural WebSocket (seperti komunikasi *full-duplex* dan koneksi persisten) secara teoritis dan praktis lebih mendukung implementasi *Reactive Pattern* untuk komunikasi *real-time* dengan LLM di Unity dibandingkan dengan alternatif seperti HTTP *streaming*?

1.3 Tujuan Penelitian

Penelitian ini memiliki beberapa tujuan utama:

a. Merancang dan mengimplementasikan secara detail arsitektur berbasis *Reactive Pattern* untuk manajemen komunikasi WebSocket asinkron pada klien Unity, memastikan interaksi *non-blocking* dengan *backend* LLM sampri-custom:latest (berbasis Llama 3.2).

b. Menganalisis secara mendalam efektivitas implementasi Reactive Pattern dalam skrip C# di Unity, khususnya dalam hal penanganan event WebSocket, pemrosesan pesan untuk layanan TTS, sinkronisasi output audio dengan sistem animasi secara aman di main thread, serta dampaknya terhadap kualitas desain perangkat lunak klien.

c. Menyajikan justifikasi ilmiah dan teknis atas pemilihan protokol WebSocket sebagai pendukung arsitektur reaktif yang diusulkan, berdasarkan analisis karakteristik fundamentalnya untuk komunikasi *real-time* dan *streaming* dengan LLM.

1.4 Batasan Penelitian

Untuk menjaga fokus dan kelayakan penelitian, ruang lingkup dibatasi pada beberapa aspek.

a. Model LLM yang digunakan adalah sampri-custom:latest (hasil *fine-tuning* berbasis Llama 3.2), diakses secara lokal melalui *framework* Ollama, dan versi spesifik keduanya dicatat.
b. Fokus utama adalah pada desain, implementasi, dan analisis *Reactive Pattern* dengan WebSocket.
c. Implementasi *Reactive Pattern* difokuskan pada sisi klien (Unity C#) untuk menangani komunikasi WebSocket secara non-blocking, dengan menggunakan pustaka NativeWebSocket untuk Unity.
d. *Server proxy* WebSocket ke Ollama dikembangkan menggunakan Python 3.10 dengan pustaka websockets dan aiohttp.
e. Evaluasi performa kuantitatif dibatasi pada metrik latensi dan *throughput*.
f. Platform target utama untuk aplikasi Unity adalah PC berbasis Windows, menggunakan Unity versi 6000.0.45f1.

1.5 Jadwal Pengerjaan

Tabel 1.5 Jadwal Pengerjaan

No.	Kegiatan	Bulan 1	Bulan 2	Bulan 3
1	Studi Literatur dan Penyusunan Proposal	X		
2	Perancangan Arsitektur Sistem	X		
3	Implementasi Klien Unity (WebSocket & HTTP)	X		
4	Implementasi *Server* Proxy WebSocket (Python)	X		
5	Integrasi dengan Ollama dan Model LLM	X		
6	Persiapan Lingkungan dan Skenario Validasi Fungsional		X	
7	Validasi Fungsional & Analisis Kuantitatif Pola		X	
8	Analisis Arsitektural & Pembahasan		X	
9	Penyusunan Laporan Tugas Akhir			X
10	Bimbingan dan Revisi			X
11	Seminar dan Sidang Tugas Akhir			X