IMPLEMENTASI MODE *MULTIPLAYER REAL-TIME* PADA *GAME* ANDROID SIMULASI PASAR SAHAM "STOCK RISING" MENGGUNAKAN *FRAMEWORK* PHOTON PUN 2

PROYEK AKHIR

Disusun oleh:

Christoffel Aristo Marbun

3312311092

Disusun untuk memenuhi salah satu syarat untuk memperoleh gelar

Ahli Madya (A.Md.) Teknik Informatika

PROGRAM STUDI TEKNIK INFORMATIKA

JURUSAN TEKNIK INFORMATIKA

POLITEKNIK NEGERI BATAM

2026

KATA PENGANTAR

Puji syukur ke hadirat Tuhan Yang Maha Esa atas segala rahmat dan karunia-Nya, sehingga saya dapat menyelesaikan penyusunan Laporan Tugas Akhir yang berjudul "Implementasi Mode *Multiplayer Real-Time* Pada *Game* Android Simulasi Pasar Saham "Stock Rising" Menggunakan *Framework* Photon PUN 2". Laporan ini disusun sebagai salah satu syarat untuk memperoleh gelar Ahli Madya (A.Md.) pada Program Studi Diploma III Teknik Informatika, Politeknik Negeri Batam. Penyelesaian Tugas Akhir ini tentu tidak lepas dari doa, bimbingan, serta dukungan dari berbagai pihak. Oleh karena itu, pada kesempatan ini, saya ingin menyampaikan rasa terima kasih yang sebesar-besarnya kepada:

1. Kedua Orang Tua dan Saudara/i Tercinta, yang tidak pernah putus memberikan doa, semangat, kasih sayang, serta dukungan moral maupun material sejak awal perkuliahan hingga penyelesaian penyusunan Tugas Akhir ini.
2. Ibu Ir. Liony Lumombo, S.ST., M.IDes, IPM., selaku Dosen Pembimbing, yang selalu meluangkan waktunya untuk memberikan arahan, ilmu, dan masukan yang sangat berharga selama proses pengerjaan proyek dan penulisan laporan ini.
3. Ibu Yeni Rokhayati, S.Si., M.Sc., selaku Ketua Program Studi Teknik Informatika, Politeknik Negeri Batam.
4. Bapak Nanda Putra Perkasa S.Tr.Kom., selaku Manajer Proyek (Manpro) pada tahap pengembangan dan persiapan lomba, yang telah meluangkan banyak waktu untuk melakukan *meeting*, memberikan ide-ide penyempurnaan, serta mengawal proses pengembangan *game* "Stock Rising" dengan penuh dedikasi.
5. Seluruh bapak/ibu dosen Politeknik Negeri Batam, khususnya di Program Studi Teknik Informatika, yang telah memberikan banyak ilmu dan pengalaman berharga selama masa perkuliahan.
6. Seluruh teman-teman dan pihak lain yang tidak dapat saya sebutkan satu per satu, yang telah memberikan bantuan dan motivasi dalam menyelesaikan proyek ini.

Saya menyadari bahwa laporan Tugas Akhir ini masih jauh dari kata sempurna. Oleh karena itu, saya sangat terbuka terhadap segala kritik dan saran yang membangun demi perbaikan di masa mendatang. Akhir kata saya berharap semoga laporan ini dapat memberikan manfaat, baik bagi pengembangan ilmu pengetahuan maupun bagi para pembaca.

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

ABSTRAK

Pengembangan literasi keuangan sangat penting di era digital saat ini, di mana peluang investasi semakin mudah diakses melalui perangkat *mobile*. Aplikasi *game* edukasi "Stock Rising" hadir untuk memberikan pengalaman belajar investasi yang menyenangkan. Namun, dinamika pasar saham di dunia nyata sangat dipengaruhi oleh interaksi banyak pihak secara bersamaan, sehingga simulasi yang efektif memerlukan lingkungan *multiplayer* waktu nyata (*real-time*). Penelitian ini berfokus pada implementasi mode *multiplayer real-time* pada *game* "Stock Rising" berbasis Android menggunakan Unity Engine dan *framework* Photon Unity Networking (PUN) 2. Tantangan utama dalam implementasi ini adalah pengelolaan dan sinkronisasi status permainan (*Game State*) antarpemain agar tidak terjadi desinkronisasi data. Penggunaan PUN 2 memungkinkan penyederhanaan infrastruktur jaringan tingkat rendah. Implementasi mencakup pembuatan sistem *matchmaking*, ruang tunggu (*Lobby* dan *Waiting Room*), serta pengelolaan *Remote Procedure Call* (RPC) menggunakan skrip C# untuk mendistribusikan data secara presisi pada lima fase permainan: *Bidding, Action, Selling, Rumour,* dan *Resolution*. Proses implementasi menerapkan metode *Agile* yang berfokus pada kecepatan dan iterasi berkelanjutan. Berdasarkan hasil pengujian *User Acceptance Testing* (UAT), implementasi arsitektur jaringan ini berfungsi dengan baik, mampu memfasilitasi interaksi kompetitif yang lancar antarpemain, dan secara efektif mendukung tujuan edukasi dari simulasi investasi ini.

Kata Kunci: Simulasi Investasi, *Multiplayer Real-Time*, Photon Unity Networking (PUN) 2, Sinkronisasi *State*, Android, C#.

BAB I PENDAHULUAN

1.1 Latar Belakang

Di tengah pesatnya perkembangan teknologi digital yang mempermudah akses terhadap instrumen investasi, tingkat literasi keuangan masyarakat Indonesia justru masih tergolong rendah. Laporan Otoritas Jasa Keuangan (OJK) tahun (2022) mencatat bahwa tingkat literasi keuangan nasional baru mencapai 17,99%. Kurangnya pengetahuan mengenai mekanisme, risiko, dan strategi ini menjadi penghalang utama bagi calon investor pemula untuk mengambil keputusan yang tepat. Sebagai solusi yang relevan dengan generasi muda, pemanfaatan teknologi berupa *game* edukasi digital terbukti mampu menyajikan materi pembelajaran secara interaktif, menyenangkan, dan memfasilitasi pengalaman belajar langsung tanpa risiko finansial. Berdasarkan kebutuhan tersebut, dikembangkanlah aplikasi simulasi investasi berbasis *game* Android bernama "Stock Rising".

Namun, pengalaman simulasi pasar saham tidak akan representatif jika hanya dilakukan secara tunggal (*Singleplayer*). Di dunia nyata, dinamika investasi sangat dipengaruhi oleh interaksi, sentimen, dan keputusan banyak investor secara bersamaan. Oleh karena itu, implementasi mode *multiplayer* menjadi sangat esensial dalam *game* simulasi finansial. Fitur *multiplayer* dirancang untuk menghadirkan suasana kompetitif, di mana keputusan setiap pemain dapat memengaruhi jalannya permainan. Hal ini sejalan dengan penelitian yang membuktikan bahwa sistem *multiplayer* memungkinkan adanya kolaborasi dan interaksi sosial antarpemain, yang secara signifikan dapat meningkatkan keterlibatan dan memperkuat proses pembelajaran di dalam lingkungan virtual (Abimanyu & Rizqi, 2024). Oleh karena itu, untuk mewujudkan interaksi dan kolaborasi yang lancar tersebut, diperlukan perancangan sistem jaringan yang mampu menangani sinkronisasi data secara presisi.

Implementasi mode *multiplayer real-time* pada platform Android menggunakan Unity memiliki tantangan teknis yang tinggi. Tantangan utama terletak pada pengelolaan sinkronisasi status permainan (*Game State*) secara *real-time* seperti penentuan giliran, sinkronisasi nilai aset, hingga efek pembukaan kartu *rumour* agar tidak terjadi ketidaksinkronan (*Desync*) data antarpemain. Untuk mengatasi kompleksitas infrastruktur jaringan tingkat rendah tersebut, diperlukan *framework* jaringan yang andal.

Photon Unity Networking (PUN) 2 dipilih sebagai solusi *backend* karena mampu menyediakan layanan *matchmaking* (Pencarian Lawan), pembuatan *room* (Ruang Permainan),

dan sinkronisasi data antarklien secara efektif berbasis *cloud* (To, 2024). Dengan penerapan PUN 2, siklus permainan "Stock Rising" yang terdiri dari fase *Bidding, Action, Selling, Rumour*, dan *Resolution* dapat tereksekusi secara sinkron bagi seluruh pemain. Berdasarkan latar belakang dan tantangan teknis tersebut, penelitian ini difokuskan pada aspek perancangan serta implementasi sistem jaringan permainan, yang dituangkan dalam Tugas Akhir berjudul "Implementasi Mode *Multiplayer Real-Time* pada *Game* Android Simulasi Pasar Saham "Stock Rising" Menggunakan *Framework* PUN 2".

Berdasarkan uraian-uraian di atas, dapat dirangkum bahwa rendahnya literasi keuangan masyarakat, khususnya terkait investasi, perlu diatasi dengan pendekatan edukatif yang inovatif melalui *game* "Stock Rising". Mengingat dinamika pasar saham sangat bergantung pada interaksi banyak pihak secara simultan, mode *multiplayer real-time* berarsitektur *client*-server menggunakan *framework* PUN 2 menjadi solusi teknis yang paling tepat. Implementasi sistem jaringan dan sinkronisasi status permainan yang solid ini diharapkan mampu menghasilkan simulasi pasar saham yang interaktif, stabil, dan memberikan pengalaman edukasi investasi yang optimal bagi para pemainnya.

1.2 Rumusan Masalah

Berdasarkan uraian pada latar belakang di atas, maka perumusan masalah dalam penelitian ini difokuskan pada implementasi arsitektur jaringan permainan, yaitu:

a. Bagaimana merancang dan mengimplementasikan sistem *matchmaking* dan pembuatan ruang bermain (*Room*) menggunakan *framework* Photon Unity Networking (PUN) 2 pada aplikasi *game* "Stock Rising"?
b. Bagaimana membangun dan mengintegrasikan antarmuka jaringan, khususnya halaman *Lobby* dan *Waiting Room*, yang dapat merespons status koneksi pemain secara *real-time*?
c. Bagaimana mengelola sinkronisasi status permainan (*Game State*) dan mendistribusikan data antarpemain di setiap fase permainan (*Bidding, Action, Selling, Rumour*, dan *Resolution*) agar tidak terjadi ketidaksinkronan (*Desync*) data?

1.3 Tujuan

Sejalan dengan rumusan masalah yang telah ditetapkan, tujuan dari penelitian dan implementasi ini adalah:

a. Mengimplementasikan *framework* Photon Unity Networking (PUN) 2 untuk membangun sistem *matchmaking* dan fitur pembuatan ruang bermain (*room*) pada aplikasi *game* edukasi "Stock Rising".
b. Merancang dan mengimplementasikan antarmuka jaringan (*User Interface*), khususnya untuk halaman *Lobby* dan *Waiting Room*, yang mampu merespons dan menampilkan status koneksi pemain secara *real-time*.
c. Membangun sistem sinkronisasi status permainan (*Game State*) berbasis *Remote Procedure Call* (RPC) menggunakan C# *scripting* untuk memastikan keakuratan distribusi data antarpemain pada setiap fase permainan (*Bidding, Action, Selling, Rumour*, dan *Resolution*) guna mencegah terjadinya desinkronisasi data.

1.4 Batasan Masalah

Agar penelitian dan implementasi ini lebih terfokus serta sesuai dengan ruang lingkup pengerjaan, maka ditetapkan beberapa batasan masalah sebagai berikut:

a. Implementasi sistem *multiplayer* dilakukan menggunakan *game engine* Unity dengan bahasa pemrograman C#, dan ditargetkan untuk berjalan pada platform Android.
b. Sistem jaringan *multiplayer* dan *matchmaking* dibangun secara eksklusif menggunakan *framework* Photon Unity Networking (PUN) 2.
c. Ruang lingkup perancangan antarmuka (UI) jaringan dan logika pemrograman dibatasi pada fitur pembuatan ruang (*Create Room*), bergabung ke ruang (*Join Room*), halaman *Lobby*, dan halaman *Waiting Room*.
d. Sinkronisasi status permainan (*Game State*) dan pengelolaan *Remote Procedure Call* (RPC) dibatasi pada alur *gameplay multiplayer* yang mencakup lima fase utama permainan, yaitu fase *Bidding, Action, Selling, Rumour*, dan *Resolution*.
e. Penelitian ini tidak membahas pembuatan aset visual 2D/3D, rancang bangun mode *singleplayer*, serta integrasi *database* (seperti Firebase) untuk sistem autentikasi (*Login*/*Logout*), *friendlist*, dan *shop* dikarenakan hal tersebut merupakan ruang lingkup pengerjaan anggota tim pengembang lainnya.
f. Seluruh simulasi transaksi dan pembagian dividen dalam jaringan permainan ini murni untuk tujuan edukasi dan tidak menggunakan uang asli.

1.5 Manfaat

Penelitian ini diharapkan dapat memberikan manfaat, baik secara teoritis maupun praktis, antara lain:

1.5.1 Bagi Pengembang dan Akademisi

Menjadi referensi teknis dan memperkaya kajian literatur mengenai tata cara implementasi arsitektur jaringan, pengelolaan *state machine*, dan penggunaan *Remote Procedure Call* (RPC) menggunakan *framework* PUN 2 pada implementasi *game* edukasi berbasis fase (*Turn-Based*).

1.5.2 Bagi Pengguna (Pemain)

Menghadirkan lingkungan simulasi investasi yang kompetitif, interaktif, tersinkronisasi dengan baik secara *real-time*, sehingga memberikan pengalaman belajar literasi keuangan yang lebih mendalam, menyenangkan, dan relevan dengan dinamika pasar nyata.

1.5.3 Bagi Peneliti

Mengasah dan membuktikan kemampuan analitis serta keterampilan teknis dalam bidang *Network Programming*, khususnya dalam memecahkan masalah desinkronisasi data pada *gameplay* yang kompleks.

1.5.4 Kontribusi terhadap *Sustainable Development Goals* (SDGs)

Selain manfaat teknis di atas, implementasi *game* edukasi ini turut berkontribusi langsung pada pencapaian Tujuan Pembangunan Berkelanjutan (SDGs), khususnya Tujuan 4: Pendidikan Bermutu (*Quality Education*). Dengan menyediakan simulasi investasi pasar saham yang aman dan tanpa risiko finansial nyata, proyek ini mendukung perluasan akses edukasi literasi keuangan yang inklusif dan merata bagi generasi muda untuk menghadapi tantangan ekonomi digital.

1.6 Kolaborasi dan Sinergi dalam Pelaksanaan Proyek

Proyek pengembangan *game* “Stock Rising” ini merupakan hasil kerja kolaboratif dalam sebuah tim pengembang. Pembagian peran dan tanggung jawab ditetapkan secara spesifik berbasis fitur dan modul teknis. Saya bertanggung jawab secara menyeluruh pada perancangan dan implementasi arsitektur jaringan *multiplayer*, sedangkan integrasi antarmuka dan basis data dikerjakan oleh anggota tim lainnya. Rincian pembagian peran disajikan pada Tabel 1.1 berikut:

Tabel 1. 1 Pembagian Peran dan Tanggung Jawab Tim

Nama Mahasiswa	Fitur/Modul yang Dikerjakan	Ruang Lingkup Tanggung Jawab
Christoffel Aristo Marbun	Modul Jaringan dan *Multiplayer* (PUN 2)	• Mengintegrasikan proyek aplikasi ke dalam infrastruktur Photon Unity Networking (PUN) 2. • Merancang dan mengimplementasikan logika *gameplay* untuk mode *multiplayer*. • Membangun fitur *Create Room, Room List Lobby,* dan *Join Lobby* untuk koneksi antarpemain. • Mengelola dan menyusun *scene gameplay multiplayer*.
Dewangga Fadillah Yusuf	Modul Basis Data (Firebase) dan Sistem Akun	• Mengintegrasikan aplikasi dengan Firebase untuk manajemen basis data *cloud*. • Membangun sistem autentikasi (*Login Guest*, *Logout*, dan notifikasi validasi *password*). • Mengembangkan fitur interaksi sosial, meliputi profil pengguna dan sistem pertemanan. • Mengimplementasikan fitur *Shop* untuk transaksi *Avatar* dan *Border*.
Jamal Nur Agung	Modul Antarmuka, Mode *Singleplayer*, dan Aset 3D	• Merancangan dan membuat aset 3D serta tampilan antarmuka pengguna (FrontEnd). • Merumuskan dan mengimplementasikan *gameflow* serta aturan permainan (*gamerule*). • Merancang logika *gameplay* dan menyusun *scene* khusus untuk mode *Singleplayer*.

BAB II TINJAUAN PUSTAKA

2.1 Tinjauan Solusi dan Sistem Terkait

Tinjauan terhadap beberapa solusi dan sistem terdahulu dilakukan untuk memberikan gambaran mengenai pendekatan teknis yang telah ada, teknologi yang digunakan, serta keterbatasan yang menjadi acuan dalam pengembangan arsitektur jaringan "Stock Rising". Uraian ini tidak dimaksudkan untuk mencari kebaruan (*Novelty*) penelitian ilmiah, melainkan sebagai dasar pendukung perancangan solusi teknis yang aplikatif pada jenjang Diploma Tiga.

Penelitian kedua oleh Coprich (2022) yang mengkaji perbandingan arsitektur jaringan *client-server* dan *peer-to-peer* dalam *game multiplayer*. Penelitian ini menyimpulkan bahwa arsitektur *client-server* memiliki keunggulan dalam hal keamanan dan manajemen data terpusat. Meskipun secara teori sangat komprehensif, penelitian ini bersifat konseptual dan tidak diimplementasikan secara spesifik pada pengembangan *game* simulasi pasar saham *mobile*.

Penelitian ketiga oleh Sarwodi, dkk. (2020) menerapkan sistem *multiplayer* menggunakan Photon Unity Networking (PUN) 2 pada *game* bergenre *Tower Defense*. Penelitian ini membuktikan bahwa *framework* PUN sangat andal dalam menangani latensi dan sinkronisasi kemunculan objek antarpemain secara *real-time*. Akan tetapi, sinkronisasi yang dilakukan pada penelitian tersebut lebih berfokus pada pergerakan fisik objek (*Transform Synchronization*), dan belum menyentuh sinkronisasi data *game state* kompleks yang berbasis giliran (*Turn-Based*) dan kalkulasi ekonomi/finansial seperti pada simulasi saham.

Berdasarkan tinjauan pustaka tersebut, Tugas Akhir ini hadir untuk mengisi celah (*gap*) yang ada dengan mengimplementasikan jaringan *multiplayer real-time* berarsitektur *client-server* menggunakan *framework* PUN 2 secara spesifik pada *game* simulasi pasar saham "Stock Rising". Pembaruan dari proyek ini terletak pada penggunaan komunikasi RPC untuk menyinkronisasi *game state* yang kompleks, seperti penentuan *Master Client*, alur giliran fase permainan, dan kalkulasi distribusi dividen kepada seluruh pemain dalam satu ruangan virtual.

Berikut adalah perbandingan antara penelitian-penelitian sebelumnya dengan penelitian yang saat ini dapat dilihat pada tabel 2.1.

Tabel 2. 1 Tabel Studi Literatur

Peneliti/ Tahun	Judul/Topik Penelitian	Metode/ Teknologi	Kelebihan	Kekurangan
Nauval dkk. (2021)	Rancang Bangun *Game* Edukasi Berbasis Android Menggunakan Unity Engine.	Unity Engine, Android.	Tampilan menarik, efektif sebagai media edukasi.	Masih bersifat *singleplayer*, tidak ada interaksi antarpemain secara *real-time.*
Coprich, (2022)	*Client-Server and Peer-to-Peer Architectures in Multiplayer.*	Komparasi Arsitektur *Client-Server* & *Peer-to-Peer.*	Analisis arsitektur jaringan yang mendalam dan aman.	Bersifat konseptual, tidak diterapkan pada pembuatan *game* edukasi.
Sarwodi dkk. (2020)	Penerapan *Multiplayer* Pada Gim *Tower Defense* Menggunakan Photon Unity.	Photon Unity Networking, Unity Engine.	Andal dalam menyinkronisasi kemunculan dan pergerakan objek secara *real-time*.	Fokus pada pergerakan fisik, belum mengelola sinkronisasi data *state* berbasis giliran dan kalkulasi ekonomi antarpemain.
Penelitian ini (2026)	Implementasi Mode *Multiplayer Real-Time* pada *Game* Android Simulasi Pasar Saham "Stock Rising" Menggunakan Framework Photon PUN 2.	PUN 2, *Client-Server*, RPC.	Sinkronisasi data *game state* (dividen, giliran, fase) yang kompleks secara presisi dan *real-time.*	Masih bergantung pada batasan CCU (Concurrent Users) gratis dari server Photon Cloud.

2.2 Landasan Konsep dan Teknologi

Pada bagian ini, akan diuraikan berbagai konsep dasar dan landasan teori yang menjadi acuan dalam pelaksanaan penelitian dan pengembangan Tugas Akhir. Pembahasan difokuskan pada teori-teori teknis yang berkaitan langsung dengan topik utama, meliputi konsep dasar *game* edukasi dan literasi keuangan, prinsip pengembangan *game mobile* menggunakan Unity Engine, arsitektur jaringan pada *game multiplayer real-time* beserta mekanisme sinkronisasi *state*, serta tinjauan mendalam mengenai *framework* Photon Unity Networking (PUN) 2 dan implementasi *Remote Procedure Call* (RPC) dalam pertukaran data antarpemain.

2.2.1 *Game* Edukasi dan Literasi Keuangan

Dalam era digital saat ini, pendekatan pembelajaran mengalami transformasi yang signifikan, salah satunya melalui pemanfaatan media interaktif seperti *game* edukasi digital. *Game* edukasi merupakan perangkat lunak yang dirancang secara khusus untuk menyampaikan materi pembelajaran melalui struktur interaktif, tantangan terukur, dan lingkungan simulasi yang menyenangkan. Menurut Pitriani, dkk. (2024), penggunaan *game* sebagai media pembelajaran digital terbukti mampu memberikan pengalaman belajar yang aktif, meningkatkan motivasi, serta menciptakan lingkungan pembelajaran yang menyenangkan bagi penggunanya.

Dalam konteks literasi keuangan, penerapan *game* edukasi memberikan ruang bagi prinsip *experiential learning*, di mana pengguna memperoleh pengetahuan melalui pengalaman langsung dalam lingkungan yang terkontrol. Literasi keuangan merupakan tingkat pengetahuan dan pemahaman yang membentuk keterampilan serta keyakinan individu untuk mengelola keuangan pribadi dan mengambil keputusan finansial yang tepat (Rosa & Listiadi, 2020). Individu dengan literasi keuangan yang baik cenderung membuat keputusan investasi dan pengelolaan keuangan yang lebih rasional serta menghindari risiko tinggi tanpa pertimbangan yang matang (Mahendra, 2022). Pemahaman yang matang ini sangat krusial ketika dihadapkan pada instrumen keuangan yang kompleks dan fluktuatif, seperti investasi di pasar saham.

Oleh karena itu, simulasi interaktif berbasis gamifikasi menjadi alat bantu yang sangat relevan untuk mengilustrasikan topik abstrak seperti dinamika pasar saham, fluktuasi harga, dan strategi portofolio tanpa membebankan risiko finansial nyata kepada penggunanya. Simulasi ini menuntut pemain untuk tidak hanya memahami teori investasi,

tetapi juga mengasah kemampuan berpikir strategis dan analitis dalam merespons perubahan kondisi simulasi.

2.2.2 Konsep Dasar Pasar Saham

Pasar saham merupakan suatu sistem atau fasilitas yang mempertemukan pihak pembeli dan penjual instrumen keuangan khususnya surat bukti kepemilikan modal suatu perusahaan (saham). Secara umum, mekanisme kerja pasar saham digerakkan oleh hukum permintaan dan penawaran (*Supply and Demand*). Menurut Utami dkk. (2022), ketika suatu saham perusahaan banyak diminati oleh investor, maka harganya akan cenderung naik, begitu pula sebaliknya. Keuntungan yang dapat diperoleh investor di pasar saham umumnya terbagi menjadi dua, yaitu *capital gain* (keuntungan dari selisih harga dan harga jual saham) serta dividen (pembagian sebagian keuntungan perusahaan kepada pemegang saham). Fluktuasi harga di pasar saham bersifat sangat dinamis karena dipengaruhi oleh berbagai faktor, mulai dari kondisi ekonomi makro, laporan keuangan perusahaan, hingga *rumour* dan sentimen pasar yang beredar. Oleh karena itu, simulasi pergerakan harga, pembagian dividen, dan efek *rumour* pasar ini menjadi mekanik inti yang direpresentasikan secara virtual di dalam *game* "Stock Rising" untuk memberikan gambar investasi yang realistis.

2.2.3 *Mobile Game Development* dan Unity Engine

Gambar 2. 1 Logo Unity Engine

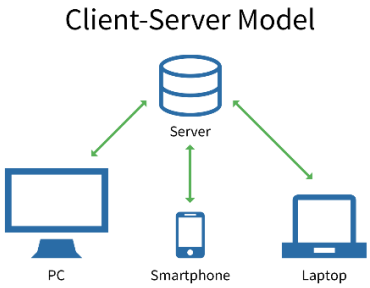
Perkembangan perangkat *smartphone*, khususnya yang berbasis sistem operasi Android, telah menjadikan *mobile gaming* sebagai platform hiburan dan edukasi yang paling mudah diakses oleh masyarakat luas. Aksesibilitas ini menjadikannya media yang ideal untuk mendistribusikan aplikasi edukasi berskala luas seperti simulasi pasar saham. Namun, pengembangan aplikasi *game* yang kompleks dengan kebutuhan interaksi grafis dan logika *real-time* membutuhkan lingkungan pengembangan yang tangguh dan efisien.

Untuk memfasilitasi kebutuhan tersebut, proyek simulasi investasi "Stock Rising" dikembangkan menggunakan Unity Engine. Unity adalah *software* mesin permainan lintas platform (*Cross-Platform*) komprehensif yang memadukan lingkungan desain visual

dengan kerangka kerja pemrograman tingkat lanjut. Menurut Nauval dkk. (2021), Unity sangat efektif digunakan dalam perancangan *game* edukasi berbasis *Android* karena menyediakan berbagai fitur terintegrasi yang sangat memudahkan pengembang dalam menyusun mekanik permainan. Keunggulan utama Unity terletak pada arsitektur berbasis komponen (*Component-Based Architecture*), di mana setiap objek dalam permainan (*GameObject*) dapat diberikan perilaku spesifik melalui penambahan komponen skrip.

Dalam pengembangan logika dan mekanik permainannya, Unity menggunakan bahasa pemrograman berorientasi objek C# (*C-Sharp*). Penggunaan C# memungkinkan pengembang untuk merancang sistem *state machine* yang rumit, mengelola siklus hidup permainan (*game loop*), dan memanipulasi elemen antarmuka (UI) secara dinamis. Selain itu, ekosistem Unity sangat mendukung integrasi pustaka pihak ketiga (*Third-Party Plugins/Frameworks*), menjadikannya lingkungan yang sangat ideal untuk menanamkan infrastruktur jaringan *multiplayer* tingkat lanjut, yang akan dibahas lebih dalam pada arsitektur *client-server* permainan ini.

2.2.4 Konsep *Multiplayer Real-Time* dan Sinkronisasi *State*



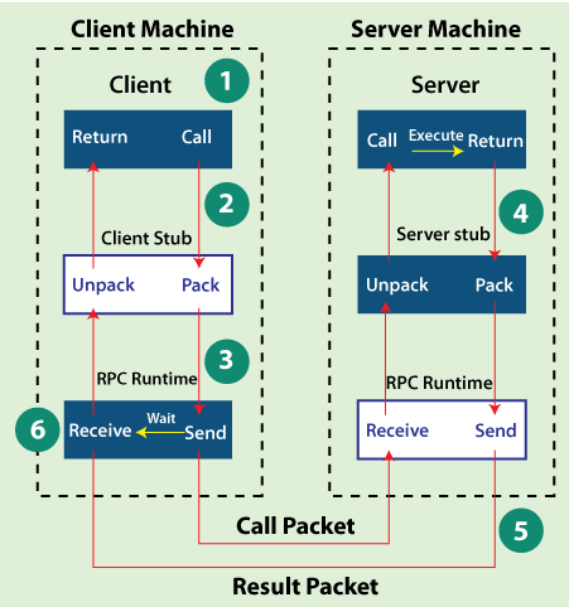
Gambar 2. 2 Arsitektur Model *Client-Server*

Dalam pengembangan *game*, mode *multiplayer* memungkinkan beberapa pemain untuk berinteraksi di dalam satu lingkungan virtual yang sama. Pada mode *multiplayer real-time*, interaksi ini terjadi seketika tanpa adanya jeda waktu yang signifikan, menuntut pertukaran data yang konstan antarpemain. Arsitektur jaringan yang umum digunakan untuk memfasilitasi hal ini adalah model *client-server*. Dalam model ini, satu entitas bertindak sebagai pusat otoritas (server) yang memvalidasi logika permainan, sementara perangkat pemain (*Client*) bertugas mengirimkan input dan menerima hasil visualisasi dari server (Coprich, 2022). Penerapan model terpusat ini sangat penting untuk menjaga integritas data permainan dan mencegah manipulasi dari sisi perangkat pemain.

Salah satu tantangan paling kritis dalam *game multiplayer* adalah menjaga konsistensi *game state* atau status permainan. *Game state* mencakup seluruh variabel yang mendefinisikan kondisi permainan pada satu waktu tertentu, seperti giliran pemain, jumlah aset yang dimiliki, status fase yang sedang berjalan, dan posisi objek. Jika perangkat pemain gagal menerima pembaruan data secara akurat atau tepat waktu, akan terjadi ketidaksinkronan (*Desync*). Fenomena *desync* menyebabkan pemain melihat versi realitas permainan yang berbeda-beda, yang pada akhirnya merusak integritas dan pengalaman bermain.

Untuk mencegah *desync*, diperlukan mekanisme sinkronisasi *state* yang andal. Pengembang harus merancang struktur data dan membatasi otoritas eksekusi kode. Biasanya, logika utama (*Core Game Loop*) dan transisi antarfase hanya boleh dieksekusi oleh entitas yang memiliki otoritas (seperti *Master Client*). Setelah *Master Client* memproses suatu aksi atau memutuskan transisi fase permainan, perubahan status tersebut kemudian didistribusikan secara serentak ke seluruh *client* lainnya di dalam ruang permainan (*Room*) melalui metode pemanggilan prosedur jarak jauh atau *Remote Procedure Call* (RPC). Dengan demikian, setiap *client* dapat memperbarui tampilan antarmuka (UI) dan objek lokal mereka berdasarkan instruksi yang tervalidasi.

2.2.5 Photon Unity Networking (PUN) 2 dan *Remote Procedure Call* (RPC)



Gambar 2. 3 Mekanisme Komunikasi *Remote Procedure Call* (RPC)

Photon Unity Networking (PUN) 2 adalah sebuah *framework* jaringan pihak ketiga yang terintegrasi secara langsung dengan Unity Engine untuk memfasilitasi pengembangan *game multiplayer* berbasis *cloud*. PUN 2 menyederhanakan kompleksitas pemrograman jaringan tingkat rendah dengan menyediakan infrastruktur *backend server* (server Photon) yang menangani proses autentikasi, *matchmaking* (pencarian lawan), dan pengelolaan ruang virtual (*room*). Dalam arsitektur PUN 2, pemain yang saling terhubung akan dikelompokkan ke dalam *room* yang sama. Secara *default*, pemain pertama yang membuat atau memasuki *room* akan ditunjuk sebagai *Master Client*. *Master Client* bertindak sebagai pemegang otoritas sementara (*Host*) yang bertugas mengeksekusi logika utama permainan dan menjaga agar sinkronisasi antarpemain tetap terkendali (To, 2024). Pengendalian otoritas oleh satu entitas ini menjadi kunci utama untuk mencegah konflik instruksi yang rentan terjadi pada *game multiplayer*.

Untuk mendistribusikan data dan perintah secara presisi antarklien didalam sebuah *room*, PUN 2 mengandalkan fitur *Remote Procedure Call* (RPC). RPC adalah metode komunikasi jaringan yang memungkinkan pemanggilan sebuah fungsi (metode) pada suatu perangkat komputasi untuk dieksekusi secara serentak di perangkat *client* lainnya yang terhubung dalam jaringan yang sama.

Dalam implementasi *game* berbasis fase (*turn-based*), penggunaan RPC sangatlah vital. Alih-alih menyinkronkan setiap pergerakan secara terus menerus yang dapat membebani *bandwidth* jaringan (*Continuous Sync*), pengembang dapat menggunakan RPC untuk memicu *event* atau perubahan status (*state*) pada momen-momen tertentu saja. Contohnya, saat *Master Client* memutuskan untuk melakukan transisi dari fase *Bidding* ke fase *Action*, atau saat sistem membagikan dividen. *Master Client* cukup memanggil metode RPC terkait, dan secara otomatis PUN 2 akan mengirimkan instruksi tersebut agar fungsi yang sama dijalankan pada antarmuka (UI) dan objek lokal masing-masing pemain. Pendekatan ini memastikan pengalaman bermain tetap tersinkronisasi dan terhindar dari *desync* secara efisien.

2.3 Metode Pengembangan Produk

Dalam pelaksanaan Tugas Akhir ini, metode pengembangan difokuskan secara spesifik pada implementasi sistem jaringan *multiplayer*, bukan pada pembuatan antarmuka (UI) atau elemen *game* secara keseluruhan. Metode yang digunakan adalah metode *Prototyping*. Metode (Pressman & Maxim, 2020) ini dipilih karena memungkinkan pengembang untuk membuat

dan menguji model dasar interaksi jaringan (*client-server*) sebelum mengintegrasikannya ke dalam alur *gameplay* utama *game* "Stock Rising". Tahapan metode *Prototyping* yang diterapkan pada implementasi jaringan PUN 2 ini meliputi:

2.3.1 Pengumpulan Kebutuhan (*Requirements Gathering*)

Teori: Tahap ini bertujuan untuk mengidentifikasi dan mengumpulkan seluruh kebutuhan teknis dari sistem yang akan dibangun berdasarkan referensi atau kondisi yang ada.

Penerapan: Pada tahap ini, dilakukan analisis kebutuhan batasan *Concurrent User* (CCU) pada *Photon Cloud*, penentuan spesifikasi *Remote Procedure Call* (RPC) yang dibutuhkan, serta identifikasi variabel data *game state* (seperti dividen dan urutan giliran) yang harus disinkronisasi antarpemain.

2.3.2 Perancangan Cepat (*Quick Design)*

Teori: Tahap ini berfokus pada penyajian desain singkat yang mencakup aspek-aspek logika dan arsitektur dari sistem sebelum proses *coding*.

Penerapan: Mengacu pada kebutuhan jaringan, dilakukan pembuatan desain arsitektur jaringan dengan model *client-server*, pembuatan *flowchart* alur *matchmaking* pemain, dan penyusunan diagram *state machine* untuk sinkronisasi fase permainan.

2.3.3 Membangun Prototipe (*Building Prototype*)

Teori: Tahap ini merupakan proses pembuatan versi awal (*prototype*) dari sistem berdasarkan desain arsitektur yang telah dirancang.

Penerapan: Mengimplementasikan skrip C# menggunakan pustaka PUN 2 ke dalam Unity Engine, Prototipe awal ini berfokus pada keberhasilan koneksi ke Photon Server, proses membuat/bergabung ke dalam Room, dan pengujian pertukaran data sederhana antar dua *client* (perangkat).

2.3.4 Evaluasi Prototipe (*Customer/User Evaluation*)

Teori: Prototipe yang telah dibangun dievaluasi dan diuji kinerjanya untuk menemukan celah kelemahan atau *error* sebelum dikembangkan lebih lanjut.

Penerapan: Menguji skrip jaringan prototipe menggunakan dua simulator perangkat untuk memastikan latensi jaringan stabil, data RPC terkirim dan diterima dengan benar, serta memvalidasi logika penentuan *Master Client* berjalan tanpa hambatan.

2.3.5 Penyempurnaan Prototipe (*Refining Prototype/Integration*)

Teori: Prototipe disempurnakan dan diintegrasikan berdasarkan hasil evaluasi hingga sistem jaringan siap digunakan secara utuh.

Penerapan: Mengintegrasikan skrip jaringan PUN 2 yang telah stabil ke dalam proyek utama *game* “Stock Rising”. Proses ini mencakup penyambungan logika *multiplayer* dengan antarmuka (UI) permainan, dan diakhiri dengan proses *build* proyek ke dalam format aplikasi Android (.apk) untuk pengujian akhir.

BAB III ANALISIS DAN PERANCANGAN

Tahap ini dilakukan untuk membedah tantangan teknis dalam mengimplementasikan mode *multiplayer real-time* pada *game* simulasi pasar saham Stock Rising. Fokus utama analisis adalah kebutuhan akan sinkronisasi data yang presisi pada setiap perpindahan fase permainan (*Bidding, Action, Selling, Rumour,* dan *Resolution*) guna mencegah terjadinya diskrepansi nilai aset antarpemain. Penggunaan *framework* Photon Unity Networking (PUN) 2 dipilih sebagai solusi untuk menangani komunikasi data antarklien dengan arsitektur *Client-Hosted Listen Server*.

3.1 Analisis Kebutuhan

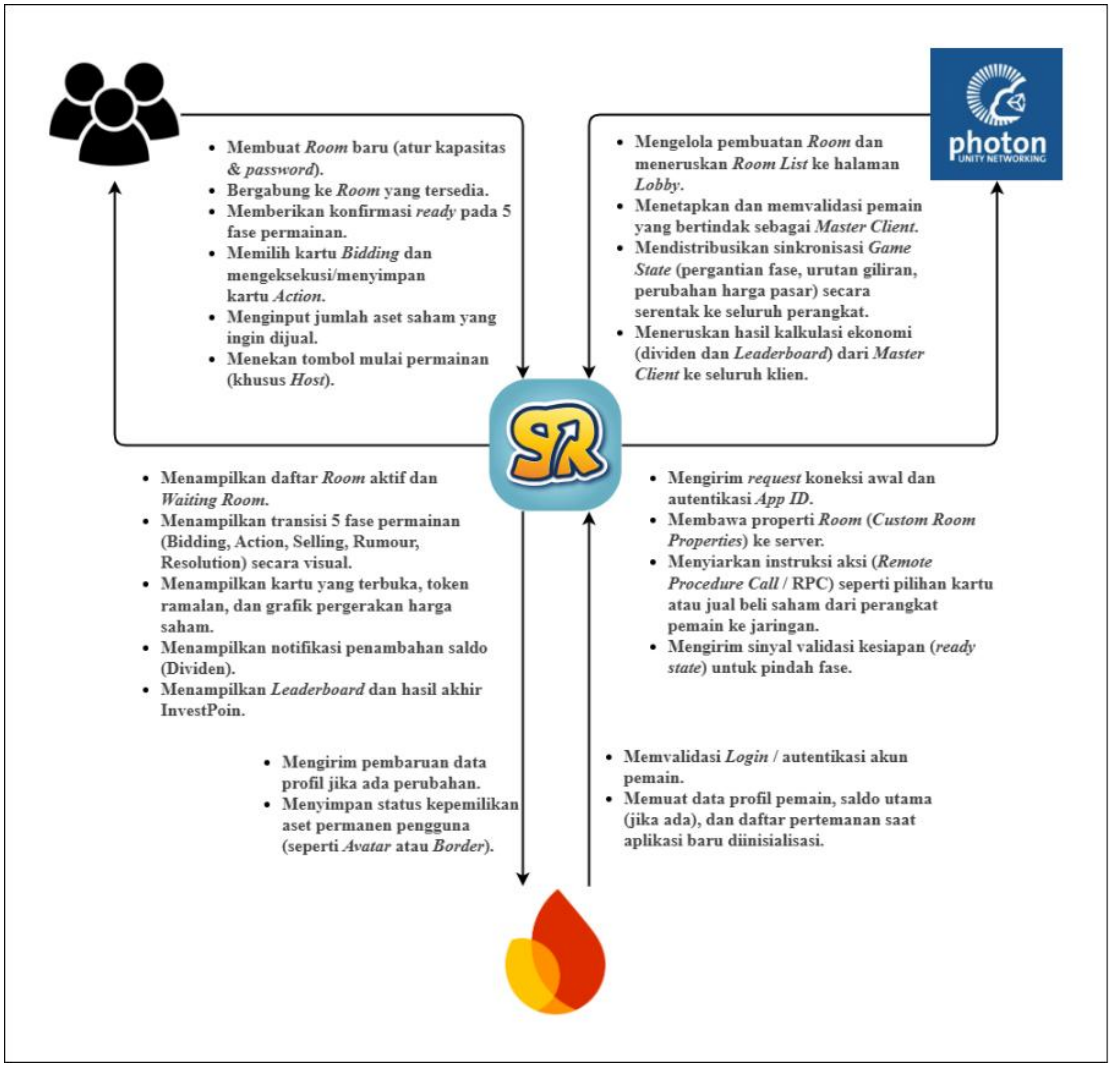
Tahap analisis kebutuhan dilakukan untuk mengidentifikasi permasalahan pada kondisi sistem yang ada saat ini (*baseline*), serta merumuskan tujuan dan spesifikasi dari penerapan metode jaringan yang akan diimplementasikan pada *game* "Stock Rising".

3.1.1 Proses Bisnis Berjalan

Pada kondisi *baseline*, *game* simulasi pasar saham "Stock Rising" dikembangkan dengan batasan mode *singleplayer*. Pemain hanya berinteraksi dengan sistem kecerdasan buatan (AI) lokal atau data statis tanpa adanya interaksi langsung dengan pemain manusia lainnya. Keterbatasan ini mengurangi tingkat kompetisi dan realisme dalam sebuah simulasi pasar saham, di mana pergerakan harga dan strategi investasi seharusnya dipengaruhi oleh aksi jual-beli dari banyak pihak secara bersamaan. Selain itu, belum ada infrastruktur jaringan yang mampu menangani sinkronisasi data antarperangkat secara *real-time*, sehingga simulasi ekonomi tidak dapat berjalan secara dinamis antarpengguna.

3.1.2 Gambaran Umum Sistem

Tujuan dari penerapan arsitektur jaringan pada Proyek Akhir ini adalah untuk mentransformasi kondisi *baseline* menjadi sistem *multiplayer real-time*. Penerapan metode jaringan menggunakan *framework* Photon Unity Networking (PUN) 2 diusulkan untuk menangani koneksi *client-server*, manajemen ruang bermain (*Matchmaking*), dan sinkronisasi status permainan (*Game State*). Sistem jaringan ini bertujuan untuk memastikan seluruh pemain di dalam satu ruang virtual menerima pembaruan data yang seragam tanda adanya *desync*, khususnya pada fase krusial seperti kalkulasi dividen dan distribusi efek kartu tanpa melibatkan mekanik keberuntungan seperti lempar dadu.



Gambar 3. 1 Gambaran Umum Aplikasi Stock Rising

3.1.3 Spesifikasi Perangkat Keras dan Perangkat Lunak

Untuk memastikan sistem beroperasi sesuai rancangan, ditetapkan sebagai berikut:

a. Perangkat Lunak: Unity Engine versi 2022.3.44f1, SDK Photon PUN 2, dan OS Android minimal versi 8.0 (Oreo).
b. Perangkat Keras: Laptop pengembangan dengan RAM minimal 8 GB dan perangkat uji berupa *smartphone* Android.

3.2 Perancangan

Pada bagian ini, diuraikan rancangan teknis dari implementasi mode *multiplayer* pada *game* "Stock Rising" berdasarkan analisis kebutuhan yang telah dijabarkan sebelumnya.

3.2.1 Kebutuhan Fungsional

Kebutuhan Fungsional mendefinisikan fitur-fitur jaringan yang harus disediakan oleh sistem:

Tabel 3. 1 Kebutuhan Fungsional

Kode FR	Deskripsi Kebutuhan Fungsional (*Functional Requirements*)
FR-01	Sistem harus mampu menghubungkan perangkat pemain ke *Master Server* Photon secara otomatis saat aplikasi dimulai.
FR-02	Sistem harus menyediakan fitur pembuatan ruang bermain (*Create Room*) dengan pengaturan kapasitas pemain dan kata sandi.
FR-03	Sistem harus mampu menampilkan daftar ruang yang tersedia (*Room List*) dan memfasilitasi pemain untuk bergabung (*Join Room*).
FR-04	Sistem harus menyinkronkan status kesiapan dan daftar pemain di dalam *Waiting Room*.
FR-05	Sistem harus menjamin sinkronisasi perpindahan 5 fase permainan secara serentak ke seluruh klien melalui perintah *Master Client*.

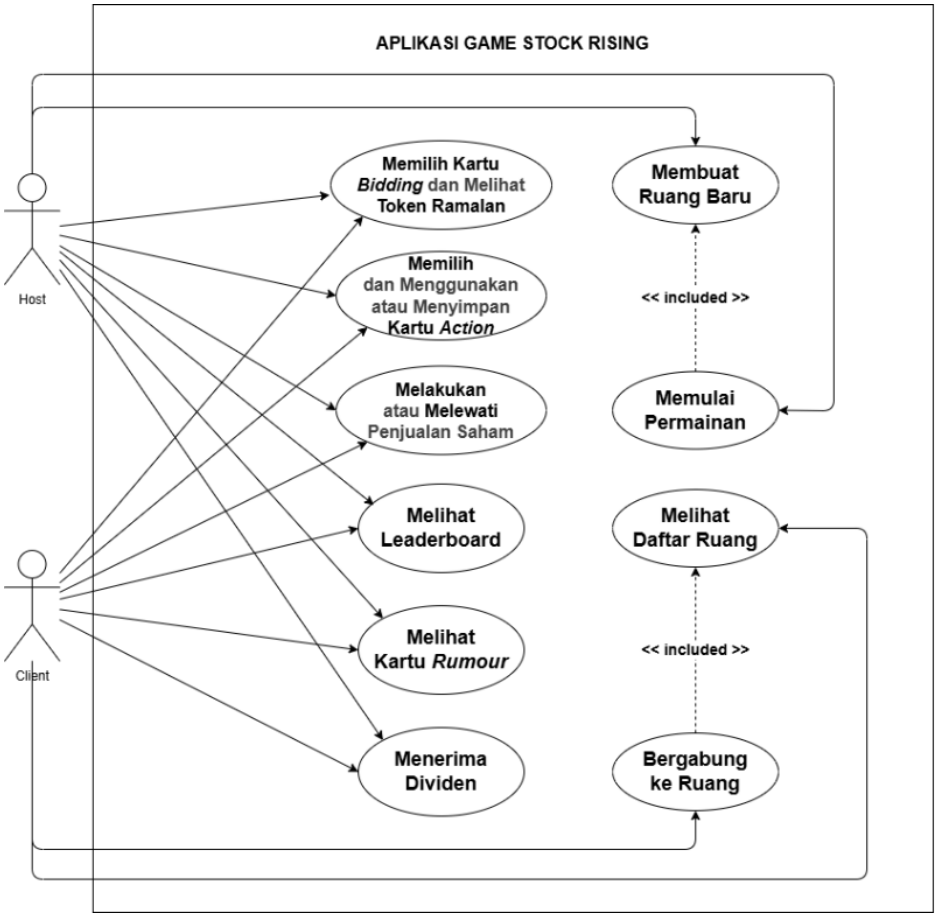
3.2.2 Kebutuhan Non-Fungsional

Kebutuhan non-fungsional mencakup batasan operasional, teknis, serta aspek legal dan etis:

Tabel 3. 2 Kebutuhan Non-Fungsional

Kode NFR	Kategori	Deskripsi Kebutuhan Non-Fungsional (*Non-Functional Requirements*)
NFR-01	*Responsivity*	Responsivitas (*Latency*): Sistem jaringan harus mampu menyinkronkan perpindahan fase antarpemain dengan latensi minimal untuk menjaga kelancaran alur strategi *turn-based*.
NFR-02	*Stability*	Stabilitas Sinkronisasi: Mekanisme RPC harus memastikan seluruh klien menerima pembaruan *game state* yang identik di setiap fase guna mencegah terjadinya *desync*.
NFR-03	*Connectivity*	Kapasitas Koneksi: Infrastruktur jaringan harus mampu menangani hingga 4 pemain aktif dalam satu *room* secara simultan tanpa penurunan performa sinkronisasi.

3.2.3 Diagram *Use Case*



Gambar 3. 2 *Use Case* Stock Rising

Diagram ini menggambarkan interaksi terdapat fungsi *Connect*, *Join Lobby*, dan *Phase Sync*, namun hanya *host* yang memiliki otoritas untuk *Create Room* dan memulai permainan (*Start Game*).

3.2.4 Skenario *Use Case*

Tabel berikut merinci salah satu skenario krusial, yaitu pembuatan ruang bermain:

Tabel 3. 3 Skenario *Use Case* Membuat Ruang Baru

Identifikasi	
Nomor	UC-01
Nama *Use Case*	Membuat Ruang Baru
Deskripsi	*Host* melakukan inisialisasi pembuatan ruang bermain baru di server Photon.

Kondisi Awal	Pemain sudah terkoneksi dengan *Master Server* Photon dan berada di antarmuka *Lobby*.
Kondisi Akhir	Ruang bermain berhasil dibuat, pemain ditetapkan sebagai *Master Client*, dan layar berpindah ke *Waiting Room*.
Skenario Utama	a. *Host* mengonfigurasi kapasitas pemain dan kata sandi ruangan. b. *Host* menekan tombol "*Create Room*". c. Sistem memanggil fungsi `PhotonNetwork.CreateRoom()` dan menyematkan pengaturan ke dalam *Custom Room Properties*. d. Server Photon membuat ruang virtual secara otomatis menetapkan *host* tersebut sebagai *Master Client*. e. Layar *host* berpindah ke antarmuka *Waiting Room*.
Skenario Alternatif	Jika koneksi ke server terputus, sistem menampilkan pesan *error* dan *Host* tetap berada di antarmuka *Lobby*.

Tabel 3. 4 Skenario *Use Case* Melihat Daftar Ruang

Identifikasi	
Nomor	UC-02
Nama *Use Case*	Melihat Daftar Ruang
Deskripsi	Pemain melihat daftar ruang bermain yang tersedia di *Room List*.
Kondisi Awal	Pemain sudah terkoneksi dengan *Master Server* Photon dan masuk ke antarmuka *Lobby*.
Kondisi Akhir	Pemain dapat melihat daftar ruang yang aktif dan memilih ruangan untuk bergabung.
Skenario Utama	a. Pemain memasuki menu *Lobby*. b. Sistem menerima pembaruan dari server melalui *callback* jaringan (seperti `OnRoomListUpdate()`). c. Sistem menampilkan daftar ruangan di layar pemain.
Skenario Alternatif	Jika ada ruang yang aktif atau dibuat oleh pemain lain, sistem menampilkan pesan informasi bahwa tidak ada ruangan yang tersedia.

Tabel 3. 5 Skenario *Use Case* Bergabung ke Ruang

Identifikasi	
Nomor	UC-03
Nama *Use Case*	Bergabung ke Ruang
Deskripsi	*Client* memasuki ruangan bermain yang sebelumnya telah dibuat oleh *Host*.
Kondisi Awal	*Client* berada di antarmuka *Lobby* dan ruang yang dituju tersedia di jaringan.
Kondisi Akhir	*Client* berhasil masuk ke dalam ruang bermain dan bergabung bersama *Host* di *Waiting Room*.
Skenario Utama	a. *Client* memilih ruangan dari daftar dan memasukkan kata sandi. b. *Client* menekan tombol "*Join Room*". c. Sistem memvalidasi jumlah pemain yang bergabung dan pencocokan kata sandi yang telah di input oleh *client* dengan data di *Custom Room Properties* server. d. Sistem mengirim request `PhotonNetwork.JoinRoom()` . e. Layar *client* berpindah ke antarmuka *Waiting Room* dan sistem memperbarui daftar pemain yang hadir secara *real-time*.
Skenario Alternatif	Jika kata sandi salah atau kapasitas pemain di dalam ruang sudah penuh, sistem menolak koneksi dan menampilkan notifikasi gagal.

Tabel 3. 6 Skenario *Use Case* Memulai Permainan

Identifikasi	
Nomor	UC-04
Nama *Use Case*	Memulai Permainan
Deskripsi	*Host* memberikan instruksi kepada sistem untuk memulai permainan bagi seluruh pemain di ruangan.
Kondisi Awal	Seluruh pemain telah berkumpul di *Waiting Room* dan kapasitas pemain telah terpenuhi.
Kondisi Akhir	Antarmuka seluruh pemain berpindah secara serentak ke arena permainan (*Gameplay Scene*).

Skenario Utama	a. *Host* menekan tombol "*Play*". b. Sistem memvalidasi otoritas pemain sebagai *Master Client*. c. *Master Client* mengeksekusi fungsi `PhotonNetwork.LoadLevel()` untuk mentransisikan *scene*. d. Server menginstruksikan seluruh c*lient* di dalam ruangan tersebut untuk membuat *scene* permainan secara serentak.
Skenario Alternatif	Sistem mengabaikan instruksi jika pemain menekan tombol "*Play*" namun tidak bertindak sebagai *Master Client*.

Tabel 3. 7 Skenario *Use Case* Memilih Kartu *Bidding* dan Melihat Token Ramalan

Identifikasi	
Nomor	UC-05
Nama *Use Case*	Memilih Kartu *Bidding* dan Melihat Token Ramalan.
Deskripsi	Pemain melakukan aksi pemilihan kartu strategi *Bidding* dan sistem menyinkronkan data tersebut ke seluruh pemain.
Kondisi Awal	Permainan telah dimulai dan berada pada fase *Bidding*.
Kondisi Akhir	Kartu *Bidding* yang dipilih tersimpan, token ramalan terbuka, dan antarmuka pemain tersinkronisasi.
Skenario Utama	a. Semua pemain memilih kartu *Bidding* di layar masing-masing b. Sistem lokal mengirimkan data pilihan tersebut melalui *Remote Procedure Call* (RPC) ke *Master Client* (`RpcTarget.MasterClient`). c. *Master Client* mengirimkan RPC balasan ke seluruh pemain (`RpcTarget.All`) untuk mengeksekusi hasil *Bidding* dan membuka token ramalan. d. Layar antarmuka seluruh pemain memperbarui status kartu dan token ramalan secara serentak.
Skenario Alternatif	Jika koneksi *Client* terputus saat fase berlangsung, Master *Client* akan mendeteksi hilangnya koneksi dan melanjutkan fase tanpa input dari *Client* tersebut.

Tabel 3. 8 Skenario *Use Case* Memilih dan Menggunakan atau Menyimpan Kartu *Action*

Identifikasi	
Nomor	UC-06
Nama *Use Case*	Memilih dan Menggunakan atau Menyimpan Kartu *Action*.
Deskripsi	Pemain memutuskan untuk mengaktifkan efek kartu *Action* atau menyimpannya, lalu sistem mendistribusikannya ke jaringan.
Kondisi Awal	Sistem telah mentransisikan status permainan ke fase *Action*.
Kondisi Akhir	Efek dari kartu *Action* teraplikasi pada pemain terkait dan data status tersinkronisasi di seluruh perangkat.
Skenario Utama	a. Pemain memilih untuk menggunakan atau menyimpan kartu *Action* pada gilirannya. b. Sistem klien mengirimkan sinyal keputusan tersebut ke *Master Client* melalui RPC. c. *Master Client* memproses efek kartu (misalnya pengurangan saldo lawan atau penambahan aset). d. *Master Client* menyiarkan hasil kalkulasi ke seluruh *Client* melalui RPC. e. Sistem lokal di masing-masing perangkat memperbarui data nilai aset dan status inventaris pemain terkait secara serentak.
Skenario Alternatif	Jika pemain melewati batas waktu atau *disconnect*, sistem *Master Client* secara otomatis memproses langkah *default* (misalnya menyimpan kartu) untuk pemain tersebut agar alur permainan tetap berjalan.

Tabel 3. 9 Skenario *Use Case* Melakukan atau Melewati Penjualan Saham

Identifikasi	
Nomor	UC-07
Nama *Use Case*	Melakukan atau Melewati Penjualan Saham
Deskripsi	Pemain menentukan jumlah saham yang ingin dijual untuk mencairkan modal sebelum rumor terbuka.
Kondisi Awal	Sistem telah mentransisikan status permainan ke fase *Selling*

Kondisi Akhir	Total Saldo uang virtual dan jumlah saham pemain diperbarui secara global.
Skenario Utama	a. Pemain memasukkan jumlah saham yang ingin dijual atau memilih melewati giliran. b. Pemain menekan tombol konfirmasi jual. c. Sistem klien mengirimkan parameter jumlah jual ke *Master Client* via RPC. d. *Master Client* memvalidasi kepemilikan saham dan menambahkan saldo berdasarkan harga pasar saat ini. e. *Master Client* memanggil RPC untuk menyinkronkan saldo terbaru dan jumlah kepemilikan saham ke layar seluruh pemain.
Skenario Alternatif	Jika pemain mencoba menginput jumlah saham melebihi yang dimilikinya, sistem *Master Client* menolak aksi tersebut dan tidak memproses perubahan.

Tabel 3. 10 Skenario *Use Case* Melihat Kartu *Rumour*

Identifikasi	
Nomor	UC-08
Nama *Use Case*	Melihat Kartu *Rumour*
Deskripsi	Sistem membuka kartu *Rumour* secara global dan pemain melihat dampaknya pada pergerakan harga pasar.
Kondisi Awal	Sistem telah menyelesaikan fase *Selling* dan mentransisikan permainan ke fase *Rumour*.
Kondisi Akhir	Harga saham di seluruh sektor diperbarui sesuai efek kartu *Rumour* yang terbuka di semua perangkat pemain.
Skenario Utama	a. *Master Client* mengeksekusi logika pembukaan kartu *Rumour* teratas dari tumpukan (*deck*). b. *Master Client* mengalkulasi dampak kartu tersebut terhadap harga masing-masing sektor investasi. c. *Master Client* mengirimkan RPC yang berisi ID kartu *Rumour* dan nilai harga saham terbaru ke seluruh klien.

	d. Sistem lokal pada perangkat pemain menampilkan animasi kartu *Rumour* yang terbuka. e. Harga pasar di layar pemain berubah secara serentak mengikut data tervalidasi dari *Master Client*.
Skenario Alternatif	Jika tumpukan kartu *Rumour* habis, *Master Client* akan menginstruksikan pengecekan ulang (*Reshuffle*) kartu sisa melalui RPC sebelum membuka kartu baru.

Tabel 3. 11 Skenario *Use Case* Menerima Dividen

Identifikasi	
Nomor	UC-09
Nama *Use Case*	Menerima Dividen
Deskripsi	Pemain menerima pembagian keuntungan (dividen) berdasarkan jumlah saham yang dipegang dan performa sektor.
Kondisi Awal	Permainan memasuki fase *Resolution* (akhir semester).
Kondisi Akhir	Saldo setiap pemain bertambah sesuai kalkulasi dividen dan tersinkronisasi di semua layar.
Skenario Utama	a. *Master Client* menghitung nilai dividen masing-masing pemain berdasarkan kepemilikan aset dan performa sektor (token ramalan). b. *Master Client* menambahkan nilai dividen ke saldo InvestPoin masing-masing pemain di *server state*. c. *Master Client* menyiarkan perintah RPC pembagian dividen keseluruhan perangkat. d. Perangkat pemain menerima data RPC dan memicu pembaruan pada UI lokal. e. Pemain melihat penambahan saldo di antarmuka masing-masing secara bersamaan.
Skenario Alternatif	Jika performa perusahaan sangat buruk berdasarkan indikator token ramalan, *Master Client* akan menghitung nilai dividen nol (0), dan pemain tidak menerima penambahan saldo.

Tabel 3. 12 Skenario *Use Case* Melihat *Leaderboard* dan Hasil Akhir Permainan

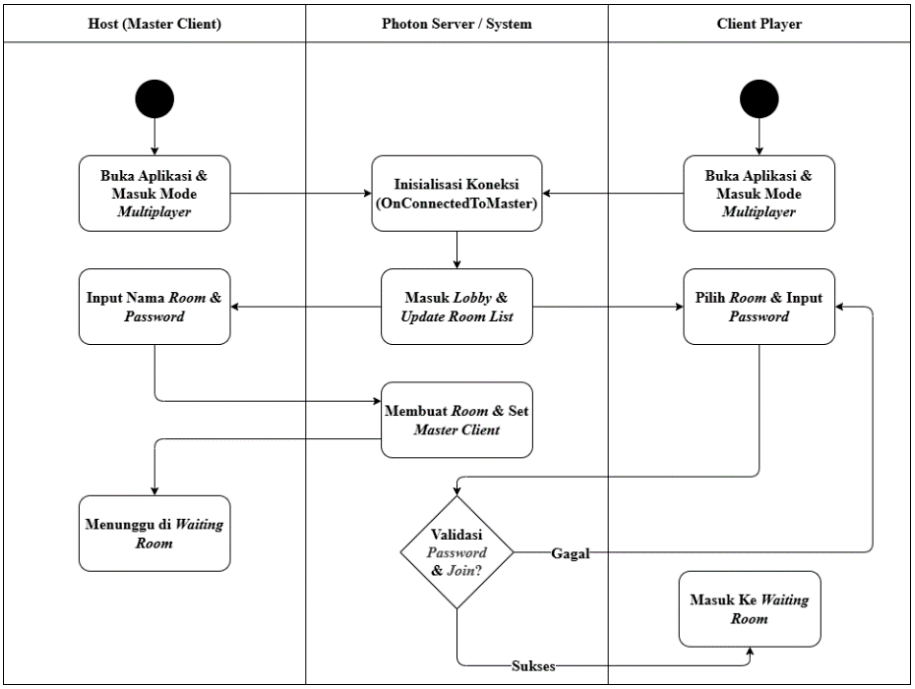
Identifikasi	
Nomor	UC-10
Nama *Use Case*	Melihat *Leaderboard* dan Hasil Akhir Permainan
Deskripsi	Pemain melihat antarmuka peringkat pemenang (*Leaderboard*) berdasarkan total aset yang dimiliki seluruh pemain di akhir permainan yang telah dihitung dan disinkronkan oleh sistem.
Kondisi Awal	Fase *Resolution* pada putaran terakhir (Semester 4) telah selesai dieksekusi.
Kondisi Akhir	Layar seluruh pemain menampilkan antarmuka *Leaderboard* dengan urutan peringkat yang sama dan tersinkronisasi.
Skenario Utama	a. *Master Client* memvalidasi bahwa permainan telah mencapai batas akhir putaran (selesai Semester 4). b. *Master Client* mengalkulasi total kejayaan tiap pemain, yang merupakan akumulasi dari: modal awal, penerimaan dividen, hasil penjualan saham, serta konversi nilai kartu saham yang masih disimpan (*keep*) berdasaran harga pasar terakhir di fase *Rumour*. c. *Master Client* mengurutkan data pemain dari total aset tertinggi hingga terendah untuk menentukan peringkat. d. *Master Client* mengirimkan paket data peringkat dan total aset tersebut ke seluruh client melalui *Remote Procedure Call* (RPC) (`RpcTarget.All`). e. Sistem lokal di masing-masing perangkat menerima RPC, memuat antarmuka *Leaderboard*, dan menampilkan hasil akhir secara serentak.
Skenario Alternatif	Jika terdapat dua atau lebih pemain dengan total aset yang sama (seri/*draw*), *Master Client* akan mengevaluasi pemenang berdasarkan aturan *game* (misalnya, pemain dengan urutan giliran terakhir akan ditempatkan di peringkat lebih tinggi), lalu menyinkronkan hasil tersebut ke seluruh perangkat.

3.2.5 Diagram Aktivitas (*Activity Diagram*)

Penyusunan diagram aktivitas pada sistem *multiplayer* "Stock Rising" dibagi menjadi tiga bagian utama, untuk mempermudah pemahaman alur kerja jaringan, mulai dari proses inisialisasi hingga penentuan pemenang.

A. Diagram Aktivitas Inisialisasi *Matchmaking*

Diagram ini menggambarkan alur saat pemain mulai menjalankan aplikasi hingga berkumpul di dalam satu ruang bermain (*Waiting Room*).



Gambar 3. 3 Diagram Aktivitas *Initiating* & *Matchmaking*

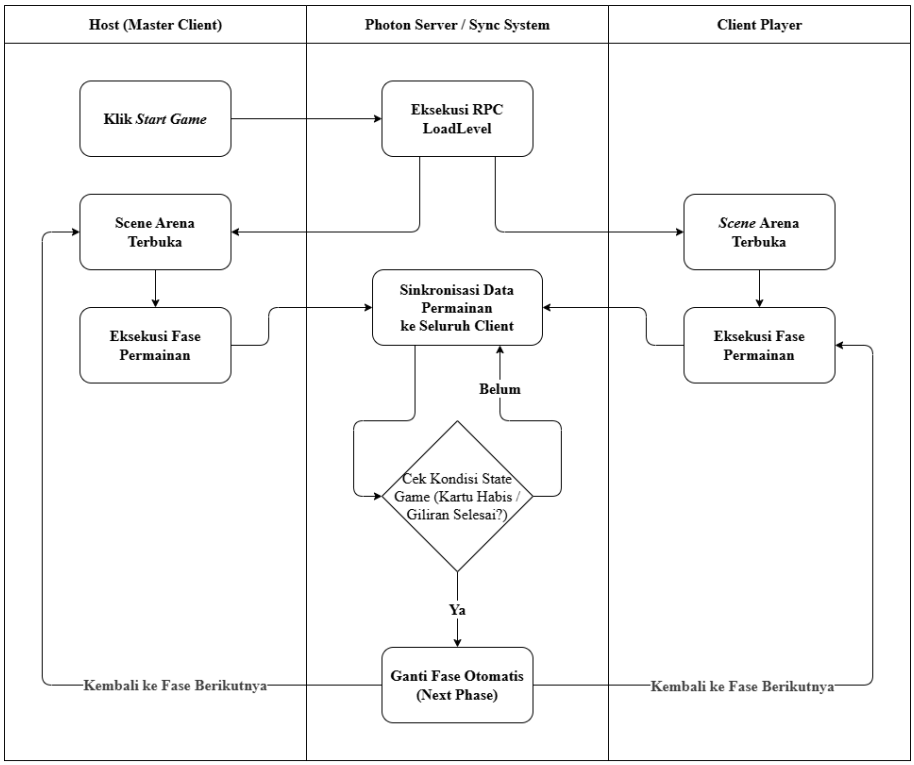
Berdasarkan Gambar 3.2, tahapan inisialisasi dan *matchmaking* adalah sebagai berikut:

a. Alur dimulai saat *Host* dan *Client* membuka aplikasi dan masuk ke menu *multiplayer*.
b. Sistem secara otomatis melakukan inisialisasi koneksi menggunakan *callback* `OnConnectedToMaster()` untuk menghubungkan perangkat dengan *Master Server* Photon.
c. Setelah terhubung, sistem mengarahkan pemain masuk ke dalam *Lobby* dan memperbarui daftar ruangan (*Room List*) yang tersedia secara *real-time*.

d. *Host* melakukan aksi pembuatan ruangan dengan memasukkan nama *room* dan *password*. Sistem kemudian mengeksekusi fungsi pembuatan ruangan dan menetapkan perangkat tersebut sebagai *Master Client*.
e. Di sisi lain, *client* memilih ruangan yang tersedia dan memasukkan *password*.
f. Sistem melakukan validasi kecocokan kata sandi. Jika sukses, *client* akan bergabung ke dalam ruangan. Jika gagal, sistem akan mengembalikan *client* ke menu pemilihan *room*.
g. Proses berakhir ketika seluruh pemain telah masuk dan berkumpul di antarmuka *Waiting Room*.

B. Diagram Aktivitas Sinkronisasi *Gameplay*

Diagram ini menjelaskan mekanisme fase permainan yang dikelola secara sinkron oleh sistem jaringan.



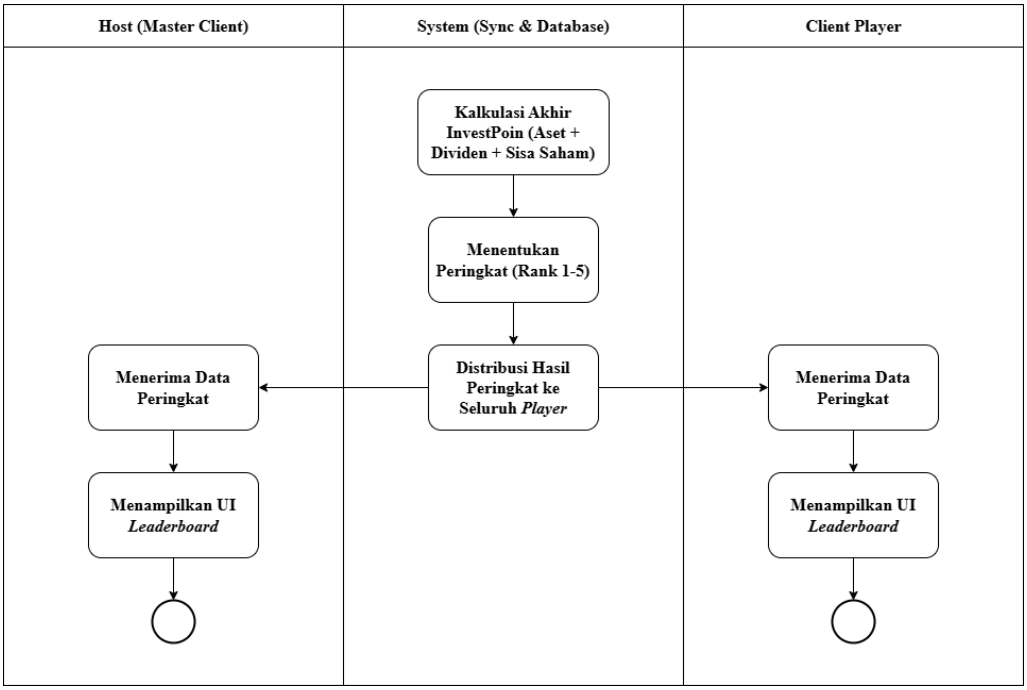
Gambar 3. 4 Diagram Aktivitas *Gameplay Synchronization*

Berdasarkan Gambar 3.3, alur sinkronisasi *gameplay* Stock Rising berjalan sebagai berikut:

a. Permainan dimulai saat *host* menekan tombol *Start Game*, yang memicu sistem untuk mengeksekusi `PhotonNetwork.LoadLevel()` agar seluruh perangkat memuat *scene* Arena secara bersamaan.
b. Seluruh pemain kemudian masuk ke dalam siklus fase permainan (*Bidding, Action, Selling, Rumour,* dan *Resolution).*
c. Setiap data aktivitas yang dilakukan oleh pemain (baik *host* maupun *client*) disinkronkan secara *real-time* ke seluruh perangkat melalui jaringan Photon.
d. Sistem secara kolektif melakukan sinkronisasi dan mengecek apakah seluruh pemain di dalam ruangan tersebut sudah berstatus *ready*.
e. Apabila syarat kesiapan seluruh pemain terpenuhi, sistem secara otomatis mengeksekusi perintah transisi fase (*Next Phase*) melalui mekanisme RPC.
f. Alur sistem kemudian berputar kembali untuk menjalankan fase berikutnya. Siklus ini terus berulang secara otomatis oleh sistem hingga permainan menyelesaikan seluruh fase pada Semester 4.

C. Diagram Aktivitas *Leaderboard* dan Hasil Akhir

Diagram ini menjelaskan proses penutupan permainan dan perhitungan nilai akhir untuk menentukan peringkat pemenang.



Gambar 3. 5 Diagram Aktivitas *Leaderboard*

Berdasarkan Gambar 3.4, alur penentuan hasil akhir permainan adalah sebagai berikut:

a. Setelah fase terakhir pada Semester 4 selesai, sistem melakukan kalkulasi nilai InvestPoin untuk setiap pemain.
b. Perhitungan InvestPoin dilakukan secara otomatis oleh sistem dengan mengakumulasikan sisa modal, total penerimaan dividen, hasil penjualan saham serta nilai kartu saham yang masih disimpan (*keep*) berdasarkan harga pasar terakhir.
c. Berdasarkan hasil kalkulasi tersebut, sistem menentukan peringkat pemain dari urutan 1 sampai 5.
d. Sistem mendistribusikan data peringkat tersebut ke seluruh perangkat yang terhubung melalui instruksi RPC.
e. Perangkat *host* dan *client* menerima data tersebut dan menampilkan antarmuka *Leaderboard* di layar masing-masing secara serempak.
f. Alur berakhir saat seluruh pemain telah menerima informasi hasil akhir permainan, menandakan siklus mode *multiplayer* telah selesai sepenuhnya.

3.2.6 Perancangan Basis Data

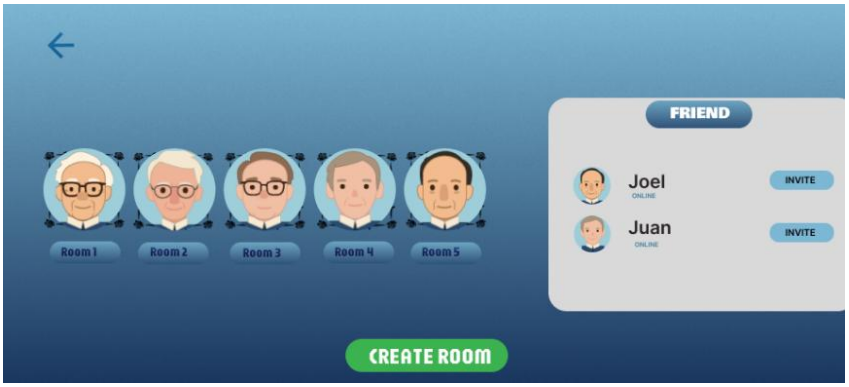
Penyimpanan data profil dan aset permanen pemain dilakukan menggunakan Firebase Realtime Database. Meskipun fokus utama penelitian ini adalah pada sinkronisasi jaringan Photon, basis data ini berperan penting dalam menyediakan data awal pemain sebelum memasuki sesi *multiplayer*.

3.2.7 Perancangan Antarmuka (*Mockup*)

Perancangan antarmuka (*Mockup*) bertujuan untuk memberikan gambaran visual serta tata letak elemen antarmuka pengguna (*User Interface*) pada aplikasi “Stock Rising”. Desain ini berfungsi sebagai acuan dalam tahap pengembangan dan implementasi mode *multiplayer* untuk memastikan fungsionalitas fitur-fitur jaringan dapat diakses dengan mudah oleh pengguna.

A. Antarmuka *Lobby*

Antarmuka ini merupakan titik awal bagi pemain untuk masuk ke dalam ekosistem *multiplayer*. Komponen utama pada halaman ini difokuskan pada manajemen ruangan (*Room Management*).



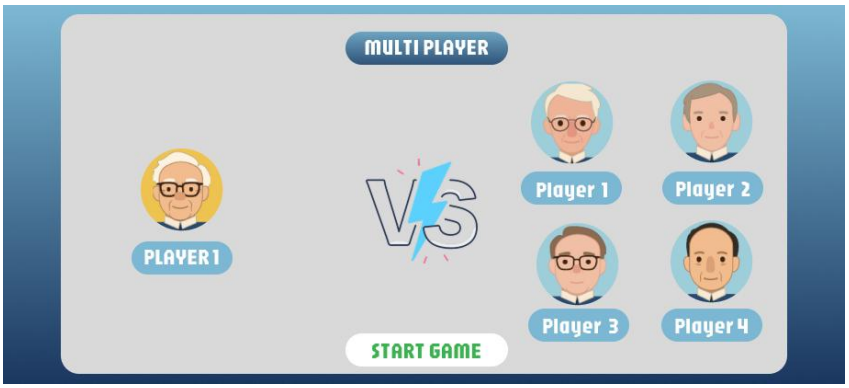
Gambar 3. 6 *Mockup* Antarmuka *Lobby*

Penjelasan komponen antarmuka *Lobby* adalah sebagai berikut:

a. Panel *Room List*: Menampilkan daftar seluruh ruangan yang sedang aktif di server Photon secara dinamis.
b. Tombol *Create Room*: Digunakan oleh *host* untuk membuka *pop-up* pembuatan ruangan baru.
c. Input *Field Password*: Kolom keamanan untuk memastikan hanya pemain yang memiliki kode akses yang dapat bergabung ke ruangan tertentu.

B. Antarmuka *Waiting Room*

Halaman ini berfungsi sebagai ruang tunggu sebelum permainan dimulai untuk memastikan seluruh pemain terkoneksi dengan sinkron.



Gambar 3. 7 *Mockup* Antarmuka *Waiting Room*

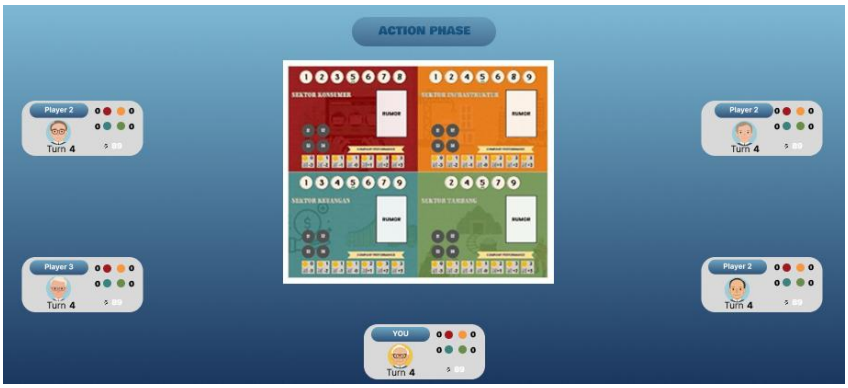
Penjelasan komponen antarmuka *Waiting Room* adalah sebagai berikut:

a. Daftar pemain (*Player List*): Menampilkan nama-nama pemain yang sudah berhasil masuk ke dalam ruangan secara *real-time*.

b. Indikator Status: Menunjukkan informasi identitas pemain, termasuk penanda khusus untuk pemain yang bertindak sebagai *Master Client*.
c. Tombol *Start Game*: Tombol khusus yang hanya muncul pada layar *host* untuk memicu sistem agar memulai permainan bagi seluruh pemain.

C. Antarmuka Arena *Gameplay*

Antarmuka utama saat simulasi pasar saham berlangsung, yang menekankan pada sinkronisasi status fase permainan antar-perangkat.



Gambar 3. 8 *Mockup* Antarmuka Arena *Gameplay*

Penjelasan komponen antarmuka Arena *Gameplay* adalah sebagai berikut:

a. Indikator Fase: Menampilkan informasi fase yang sedang berjalan (*Bidding*, *Action*, *Selling*, *Rumour*, *Resolution*) yang tersinkronisasi untuk seluruh pemain. Sistem secara otomatis memicu pergantian fase saat kondisi permainan terpenuhi (misalnya saat seluruh kartu pada fase tersebut telah habis terambil).
b. Area Interaksi Pemain: Panel utama tempat pemain mengeksekusi strategi, seperti mengambil kartu *Bidding* atau kartu efek, sesuai dengan urutan atau aturan fase yang sedang aktif.
c. Panel Status Sinkronisasi: Area informasi yang memvisualisasikan giliran pemain atau memunculkan status tunggu secara *real-time* ketika sistem sedang menyinkronkan data aksi dari perangkat lain di dalam jaringan.

D. Antarmuka *Leaderboard*

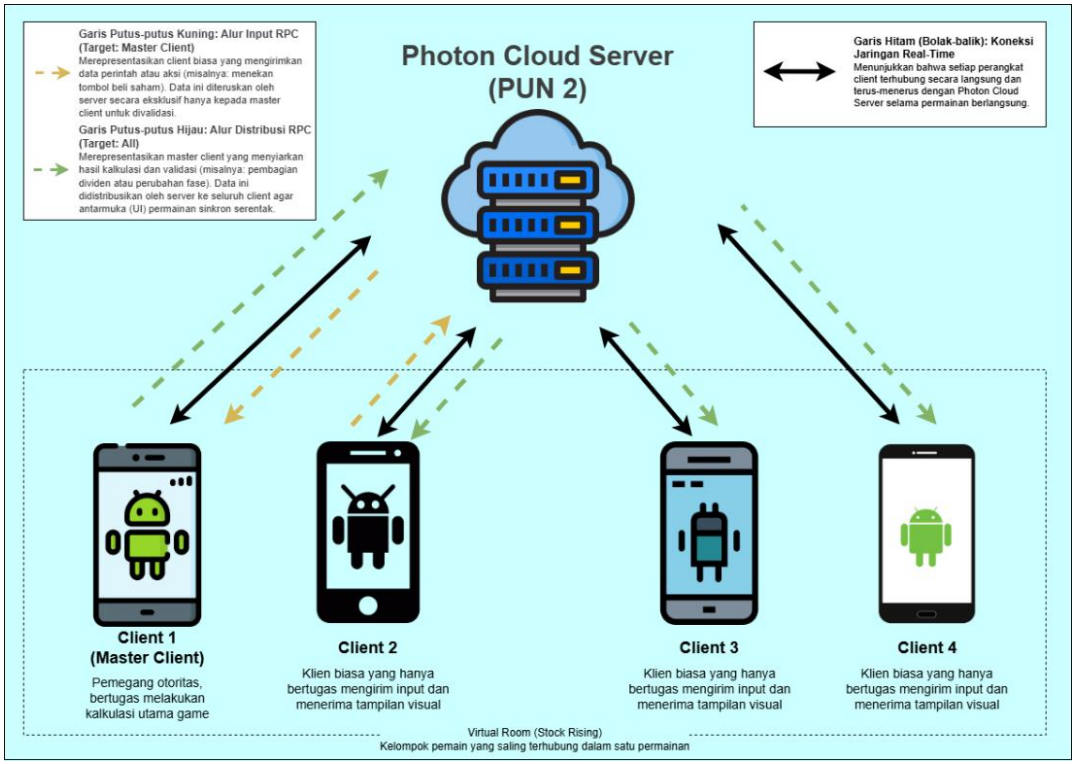
Gambar 3. 9 *Mockup* Antarmuka *Leaderboard*

Tampilan hasil akhir permainan yang menyajikan data peringkat berdasarkan akumulasi aset seluruh pemain.

Penjelasan komponen antarmuka *Leaderboard* adalah sebagai berikut:

a. Tabel Peringkat: Daftar yang disusun secara berurutan berdasarkan nilai InvestPoin tertinggi hingga terendah.
b. Kolom Data Aset: Menampilkan detail akumulasi modal dan keuntungan yang diperoleh setiap pemain selama siklus permainan berlangsung.
c. Tombol *Back to Lobby*: Fungsi untuk memutus koneksi dari ruangan dan kembali ke menu utama secara serentak.

3.2.8 Arsitektur Jaringan yang Diusulkan (Topologi)



Gambar 3. 10 Arsitektur Jaringan Multiplayer pada game "Stock Rising"

Implementasi mode *multiplayer* pada *game* "Stock Rising" dirancang menggunakan arsitektur jaringan *client-server* yang difasilitasi oleh infrastruktur *cloud* dari *Photon Unity Networking* (PUN) 2. Dalam topologi ini, perangkat seluler masing-masing pemain bertindak sebagai *client* yang menjalankan *game engine* Unity, sementara server Photon bertindak sebagai fasilitator (*Relay Server*) yang meneruskan pesan, sinkronisasi variabel, panggilan metode jarak jauh secara *real-time*. Proses komunikasi jaringan dirancang dengan alur sebagai berikut:

A. Koneksi Awal

Saat pemain membuka *game*, sistem akan secara otomatis mencoba menghubungkan perangkat *client* ke *Master* Server Photon menggunakan konfigurasi jaringan yang telah ditetapkan.

B. Manajemen Ruang (*Lobby* dan *Room*)

Setelah terhubung, *client* akan masuk ke dalam *Lobby* utama. Di sini, server akan mencatat dan memperbarui daftar ruang bermain (*Room*) yang tersedia. Pemain dapat membuat ruang baru atau bergabung ke ruang yang sudah ada.

C. Penentuan Otoritas (*Master Client*)

PUN 2 tidak menggunakan *dedicated server* untuk memproses logika permainan secara mandiri. Sebagai gantinya, sistem akan secara otomatis menunjuk pemain pertama yang membuat atau memasuki *room* sebagai *Master Client.*

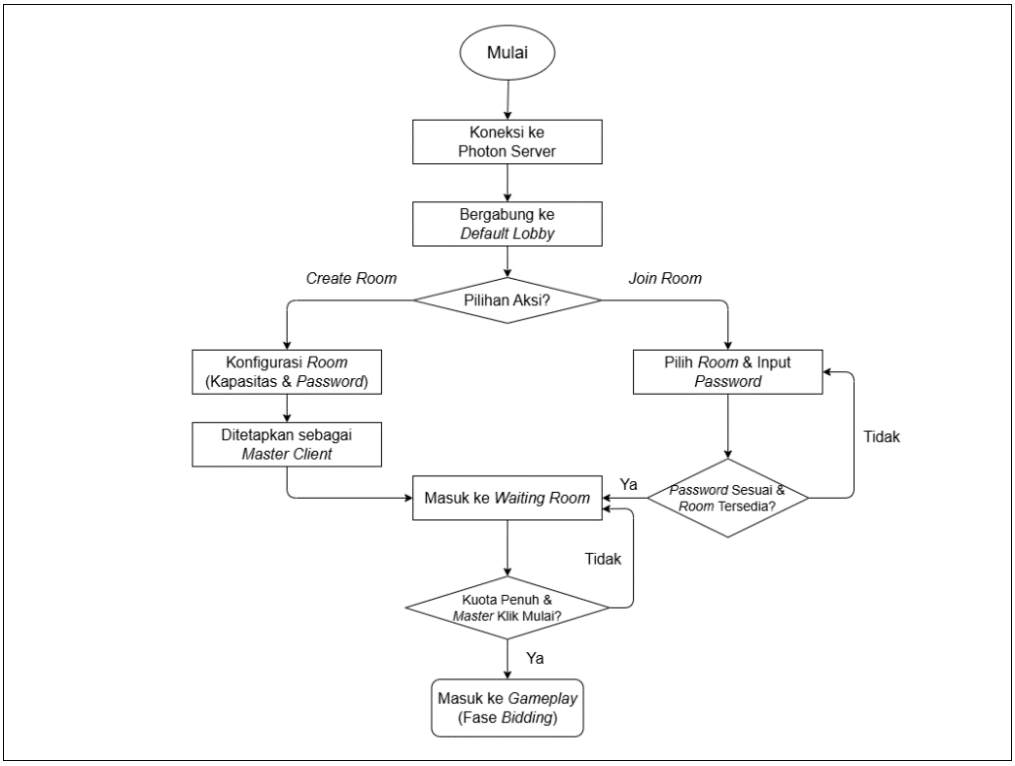
D. Distribusi Data

Selama permainan berlangsung, *Master Client* memegang otoritas penuh untuk mengeksekusi logika utama (*Core Game Loop*), seperti menghitung nilai saham, memvalidasi giliran, dan memicu perpindahan fase permainan. Hasil dari kalkulasi tersebut kemudian didistribusikan oleh *Master Client* ke seluruh *client* lainnya melalui server menggunakan *Remote Procedure Call* (RPC), sehingga antarmuka visual (UI) di setiap perangkat menampilkan kondisi permainan yang seragam dan terhindar dari *desync*.

3.2.9 Desain Konfigurasi Layanan atau Metode Jaringan

Konfigurasi layanan jaringan pada sistem ini berfokus pada tiga aspek konfigurasi utama, yaitu alur pencarian ruang bermain, pengendalian status permainan (*State Machine*), dan manajemen pengiriman pesan antarperangkat.

A. Perancangan Alur *Matchmaking*



Gambar 3. 11 *Flowchart* Alur *Matchmaking* Pemain

Proses *matchmaking* merupakan gerbang utama yang memfasilitasi pertemuan antarklien sebelum simulasi investasi dimulai. Perancangan alur ini dibagi menjadi tiga tahapan utama: inisialisasi koneksi, pengelolaan ruang (*Room*), dan sinkronisasi ruang tunggu (*Waiting Room*).

a. Tahap Inisialisasi Koneksi

Pada tahap awal, saat aplikasi pertama kali dijalankan, sistem secara otomatis memanggil fungsi koneksi dari *framework* PUN 2. Perangkat *client* melakukan autentikasi dengan server Photon menggunakan pengaturan dasar aplikasi. Setelah koneksi ke *Master Server* berhasil terbentuk, sistem secara otomatis mengarahkan *client* untuk bergabung ke dalam *Lobby* utama (*Default Lobby*). *Lobby* ini berfungsi sebagai ruang perantara di mana *client* dapat menerima pembaruan data mengenai daftar *room* yang sedang aktif sebelum akhirnya memuat antarmuka menu utama.

b. Tahap Pembuatan dan Pemilihan Ruang

Setelah berada di menu utama, pemain dihadapkan pada antarmuka *Lobby*. Pada tahap ini, pemain memiliki dua pilihan: membuat ruang baru (*Create Room*) atau bergabung dengan ruang yang sudah ada (*Join Room*).

c. Pembuatan Ruang

Saat pemain membuat ruang baru, sistem mengonfigurasi pengaturan ruang (*RoomOptions*), yang mencakup batas maksimal pemain dan kata sandi (*Password*) menggunakan fitur *Custom Room Properties*. Pemain yang membuat ruang ini akan secara otomatis dimasukkan ke dalam ruang tersebut dan ditetapkan sebagai *Master Client* .

d. Bergabung ke Ruang

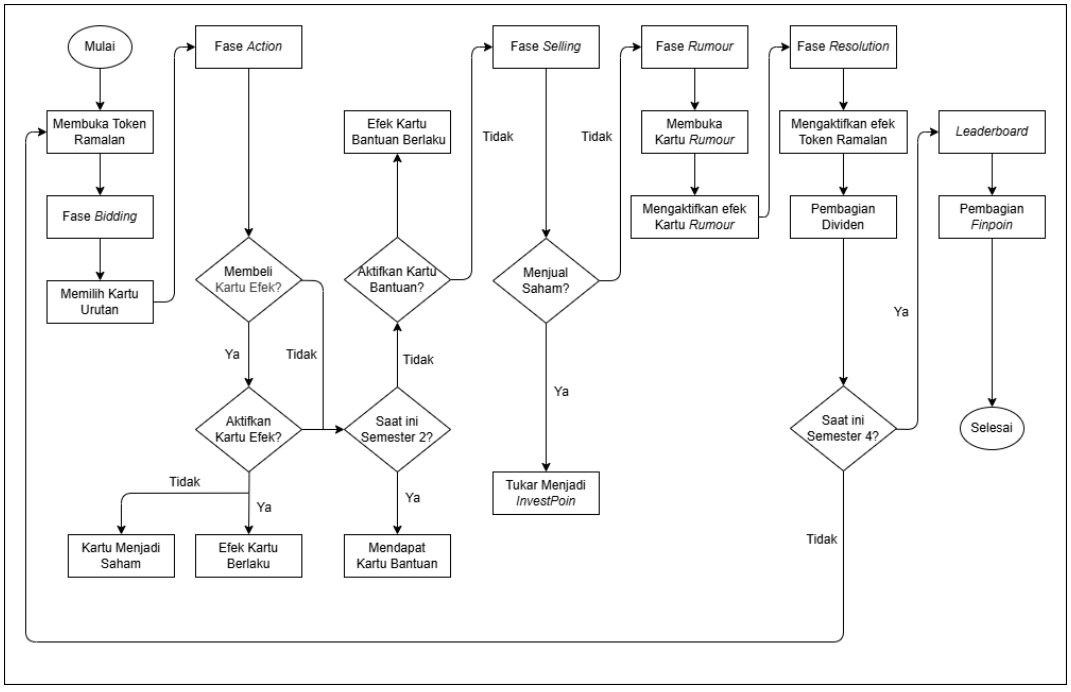
Client lain yang ingin bergabung dapat melihat daftar ruang yang tersedia. Jika ruang tersebut dilindungi oleh kata sandi, sistem akan melakukan validasi pencocokan kata sandi lokal dengan data yang tersimpan di *Custom Room Properties* milik ruang tersebut di server. Jika validasi berhasil dan kapasitas ruang belum penuh, *client* akan diizinkan masuk.

e. Tahap Ruang Tunggu (*Waiting Room*)

Setelah berhasil memasuki *room*, *client* akan dialihkan ke antarmuka ruang tunggu. Pada fase ini, sistem melakukan sinkronisasi visual secara terus-menerus terhadap daftar pemain yang hadir (*Player List*). Untuk memastikan permainan dimulai dengan kondisi yang terstruktur, hak akses tombol "Mulai Permainan" dibatasi secara ketat. Sistem akan melakukan pengecekan otoritas secara *real-time*; tombol tersebut hanya dapat ditekan oleh entitas yang berstatus sebagai *Master Client*, dan hanya jika jumlah pemain di dalam ruang telah mencapai batas maksimal (PlayerCount == MaxPlayers).

Selain itu, sistem pada ruang tunggu ini juga telah dirancang untuk menangani skenario perpindahan otoritas (*Host Migration*). Apabila *Master Client* saat ini terputus koneksinya atau keluar dari ruang, server Photon akan secara otomatis menunjuk *client* lain yang berada di urutan berikutnya sebagai *Master Client* baru, dan sistem antarmuka akan langsung memperbarui hak akses tombol mulai untuk pemain tersebut.

B. Perancangan *State Machine* Permainan



Gambar 3. 12 *Flowchart State Machine* Fase Permainan

Inti dari logika permainan simulasi "Stock Rising" dikendalikan oleh sebuah sistem *state machine* terpusat yang diatur melalui skrip pengelola permainan utama. Karena alur permainan simulasi ini berbasis fase (*Turn-Based*), menjaga

keseragaman status fase pada seluruh perangkat pemain adalah prioritas utama. Perancangan *state machine* ini mengatur siklus hidup dari satu putaran (*Round*) permainan yang terdiri dari lima fase utama yang berjalan secara sekuensial. Adapun rancangan dari kelima fase permainan tersebut adalah sebagai berikut:

a. Fase *Bidding*

Merupakan fase awal di mana para pemain diberikan kesempatan untuk menganalisis pasar dan melakukan pembelian aset saham menggunakan saldo yang mereka miliki.

b. Fase *Action*

Pada fase ini, pemain dapat menggunakan kartu aksi (*Action Card*) yang mereka miliki untuk memengaruhi kondisi permainan, seperti memberikan keuntungan bagi diri sendiri atau memberikan efek strategis terhadap pemain lawan.

c. Fase *Selling*

Setelah serangkaian aksi dilakukan, pemain diberikan jendela waktu untuk menjual aset saham yang mereka miliki ke pasar guna mencairkan modal (*Capital Gain*) sebelum *rumour* pasar dibuka.

d. Fase *Rumour*

Merupakan fase kritis di mana kartu *rumour* dibuka dan diumumkan kepada seluruh pemain. Efek dari kartu *rumour* ini akan memengaruhi fluktuasi harga saham di pasar secara dinamis.

e. Fase *Resolution*

Merupakan fase penutup putaran. Pada fase ini, sistem akan melakukan kalkulasi pembagian dividen bagi pemain yang menahan saham, memperbarui InvestPoin di papan skor (*Leaderboard*) sementara, dan mengevaluasi kondisi akhir permainan (*End-Game Condition*). Jika permainan belum berakhir, sistem akan mengembalikan status permainan ke fase *Bidding* untuk putaran selanjutnya.

Untuk mencegah terjadinya *desync* atau lompatan fase yang tidak terkendali, perancangan perpindahan antarfase (*Phase Transition*) dikunci secara ketat di bawah otoritas *Master Client*. Setiap kali seorang *client* menyelesaikan aktivitasnya di suatu

fase (misalnya menekan tombol “Selesai” pada fase *Bidding*), perangkat *client* tersebut tidak akan berpindah fase secara mandiri. Alih-alih demikian, *client* akan mengirimkan sinyal konfirmasi (*ready state*) kepada server.

Master Client bertugas untuk mengumpulkan dan memantau status kesiapan dari seluruh *client* yang berada di dalam *room*. Apabila kondisi validasi terpenuhi (jumlah pemain yang siap telah sama dengan total pemain aktif), *Master Client* dapat mengeksekusi logika perpindahan fase ke *state* berikutnya. Keputusan ini kemudian didistribusikan secara instan ke seluruh *client* melalui metode *Remote Procedure Call* (RPC), sehingga seluruh layar pemain akan berganti fase secara serentak.

C. Perancangan Sinkronisasi *Remote Procedure Call* (RPC)

a. RPC dari *Client* ke *Master Client*

Perangkat *client* biasa tidak diizinkan untuk mengubah status permainan secara global. Ketika seorang pemain melakukan aksi, seperti mengonfirmasi pembelian saham di fase *Bidding* atau menekan tombol “Selesai”, *client* tersebut akan mengirimkan RPC yang secara spesifik ditargetkan hanya kepada *Master Client* (RpcTarget.MasterClient). Data yang dikirim berupa parameter input pemain (seperti ID aset dan jumlah saldo). Hal ini menjadikan *Master Client* sebagai satu-satunya pusat validasi data, guna mencegah kecurangan (*cheat*) atau manipulasi data dari sisi *client*.

b. RPC dari *Master Client* ke Seluruh *Client*

Setelah *Master Client* menerima, memvalidasi, dan mengalkulasi dampak dari input tersebut terhadap *game state* keseluruhan, *Master Client* akan menyiarkan hasil akhirnya ke seluruh pemain di dalam *room* (RpcTarget.All atau RpcTarget.Others).

Sebagai contoh perancangan pada fase *Resolution* (pembagian dividen): *Master Client* akan menghitung berapa banyak dividen yang berhak diterima oleh masing-masing pemain berdasarkan kepemilikan saham mereka. Setelah kalkulasi selesai, *Master Client* akan memanggil fungsi RPC pembagian dividen. Fungsi ini akan tereksekusi secara serentak di perangkat semua pemain, yang menginstruksikan sistem lokal masing-masing *client* untuk memperbarui variabel

saldo (*balance*) dan menampilkan animasi atau notifikasi perubahan saldo di layar antarmuka mereka.

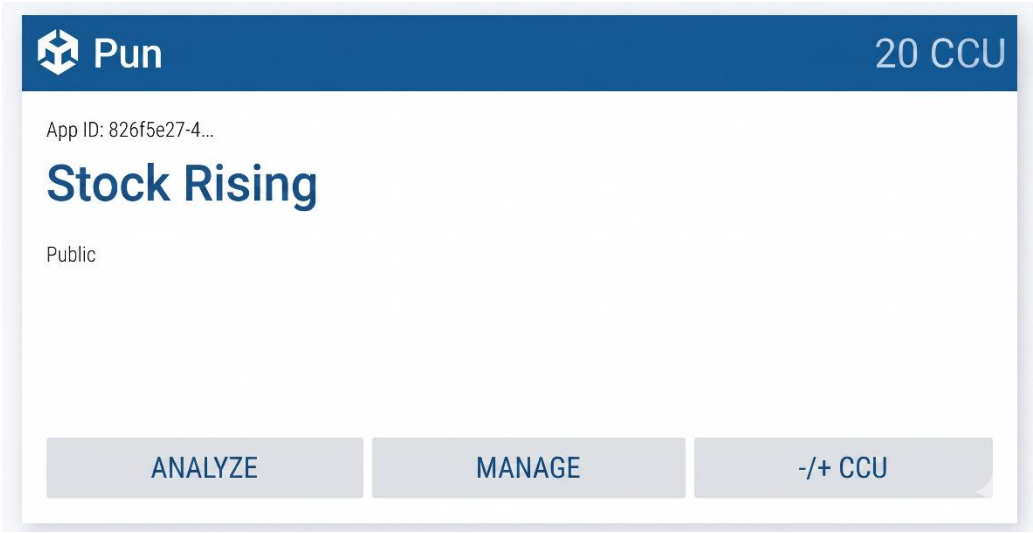
Melalui perancangan desentralisasi tugas ini, penggunaan *bandwidth* internet menjadi sangat hemat, karena paket data hanya dikirimkan ketika terjadi perubahan kondisi (*State Transition*) atau saat kartu aksi/*rumour* dimainkan.

BAB IV IMPLEMENTASI DAN PENGUJIAN

4.1 Implementasi

4.1.1 Implementasi Lingkungan Jaringan / Server

Sebelum fungsi jaringan dapat berjalan pada aplikasi klien, lingkungan server harus disiapkan melalui *dashboard* Photon Engine. Aplikasi "Stock Rising" didaftarkan untuk mendapatkan *App Id* khusus yang menjadi jembatan komunikasi antara aplikasi *game* dengan server Photon Cloud. Pada konfigurasi tersebut, region server diatur secara otomatis atau difokuskan pada area Asia (Asia/Singapura) guna meminimalisir *latency* (keterlambatan pengiriman data) saat pemain berada di wilayah Indonesia.



Gambar 4. 1 Konfigurasi Aplikasi pada *Dashboard* Photon Cloud

4.1.2 Implementasi Modul Jaringan

Modul jaringan diimplementasikan menggunakan bahasa pemrograman C# yang terintegrasi dengan pustaka bawaan dari PUN 2. Terdapat beberapa skrip utama yang menangani siklus hidup jaringan (*Network Lifecycle*), antara lain:

A. Implementasi Koneksi ke Server

```
public class ConnectToServer : MonoBehaviourPunCallbacks
{
    // Start is called before the first frame update
    void Start()
    {
```

```
        PhotonNetwork.ConnectUsingSettings(); // Connect to Photon Server
using settings from the PhotonServerSettings file
    }

    public override void OnConnectedToMaster()
    {
        PhotonNetwork.JoinLobby(); // Join the default Lobby
    }
```

Proses penyambungan aplikasi klien ke server Photon ditangani oleh skrip ConnectToServer.cs. Pada saat *game* pertama kali dimuat, fungsi `PhotonNetwork.ConnectUsingSettings()` akan dipanggil. Sistem akan mengautentikasi konfigurasi *App ID* dan mencoba terhubung ke *Master Server*. Apabila berhasil terhubung, *callback* `OnConnectedToMaster()` akan memicu perintah `PhotonNetwork.JoinLobby()` agar pemain siap mencari atau membuat ruangan.

B. Implementasi Sistem *Matchmaking* (Pembuatan dan Pencarian Ruangan)

```
        RoomOptions options = new RoomOptions();
        options.MaxPlayers = (byte)selectedMaxPlayers;

        // Simpan password di custom properties
        options.CustomRoomProperties = new
ExitGames.Client.Photon.Hashtable()
        {
            { "pwd", password },
            { "maxP", selectedMaxPlayers },
            { "started", false } // ⬅ Tambahkan ini
        };
        options.CustomRoomPropertiesForLobby = new string[] { "pwd",
"started", "maxP" };

        PhotonNetwork.NickName = username; // Simpan username sementara
ke Photon
        PhotonNetwork.CreateRoom(roomName, options);
    }
```

Proses pembentukan sesi permainan ditangani oleh skrip CreateAndJoin.cs. Saat pengguna membuat ruangan, sistem menerapkan `RoomOptions` untuk mengatur batas

maksimal pemain (MaxPlayers) dan atribut ruangan (seperti *password* dan status *started*). Data ini dikirimkan ke server melalui fungsi `PhotonNetwork.CreateRoom()`.

```
public override void OnPlayerEnteredRoom(Player newPlayer)
    {
        UpdatePlayerList();
    }

    public override void OnPlayerLeftRoom(Player otherPlayer)
    {
        UpdatePlayerList();
    }

    // --- TAMBAHAN: PENTING UNTUK HOST MIGRATION ---
    // Dipanggil saat MasterClient lama keluar dan MasterClient baru
terpilih
    public override void OnMasterClientSwitched(Player newMasterClient)
    {
        // Update state tombol untuk semua orang, terutama untuk host
yang baru
        UpdatePlayButtonState();
    }
```

Sedangkan untuk ruang tunggu (*Lobby*), pengelolanya berada di skrip WaitingRoom.cs. Skrip ini menggunakan callback `OnPlayerEnteredRoom()` dan `OnPlayerLeftRoom()` untuk menyinkronkan daftar pemain (Player List) secara real-time. Terdapat pula fungsi `OnMasterClientSwitched()` yang bertugas memastikan perpindahan otoritas secara otomatis jika *host* (Pembuat Ruangan) terputus dari jaringan.

C. Implementasi Sinkronisasi Permainan (*Game State*)

```
[RequireComponent(typeof(PhotonView))]
public class MultiplayerManager : MonoBehaviourPunCallbacks
{
    public static MultiplayerManager Instance;
    private TicketManagerMultiplayer ticketManager;
    private PhotonView gameStatusView;
void Awake()
    {
        if (GameStatusUI.Instance != null)
```

```
        {
            gameStatusView = GameStatusUI.Instance.photonView;
        }

        if (Instance != null) { Destroy(gameObject); } else { Instance =
this; }
    private IEnumerator TransitionToActionPhase()
    {
        yield return new WaitForSeconds(0.5f);

        // 1. Kirim perintah ke semua pemain untuk memulai transisi
        photonView.RPC("Rpc_StartFadeTransition", RpcTarget.All,
TransitionType.Action);

        // 2. Tunggu transisi selesai (sesuaikan durasi: 0.5s fadein +
1.5s hold + 0.5s fadeout = 2.5s)
        yield return new WaitForSeconds(2.0f);
PlayerProfileMultiplayer profileToPlace = null;
            foreach (var profile in allPlayerProfiles)
            {
                if (profile.photonView.Owner == player)
                {
                    profileToPlace = profile;
                    break;
                }
            }
public void ShowLeaderboard()
    {
        // Kirim RPC ke semua pemain untuk menampilkan leaderboard
        photonView.RPC("Rpc_ShowLeaderboard", RpcTarget.All);
    }
            else if (rank == 4) finPoinToAward = 35;
            else if (rank == 5) finPoinToAward = 30;

            if (finPoinToAward > 0)
            {
                // Kirim RPC dengan nilai FinPoin yang akan ditambahkan
                photonView.RPC("Rpc_UpdateFinPoin", targetPlayer,
finPoinToAward);
            }
```

Sinkronisasi alur permainan dikendalikan secara terpusat oleh skrip MultiplayerManager.cs. Skrip ini memanfaatkan komponen PhotonView untuk memastikan bahwa segala perubahan status, perpindahan fase, maupun penambahan nilai InvestPoin ditampilkan secara serentak ke seluruh klien dalam satu ruangan yang sama, sehingga mencegah terjadinya ketidaksesuaian data (*Desync*) antarpemain.

4.2 Implementasi Antarmuka (*Interface*)

Bagian ini menguraikan hasil implementasi rancangan antarmuka ke dalam *game engine* Unity. Fokus utama pada antarmuka ini adalah integrasi visual dengan skrip jaringan Photon PUN 2 untuk memastikan setiap interaksi pemain dapat disinkronkan secara *real-time* ke seluruh perangkat yang terhubung.

4.2.1 Implementasi Antarmuka *Lobby*

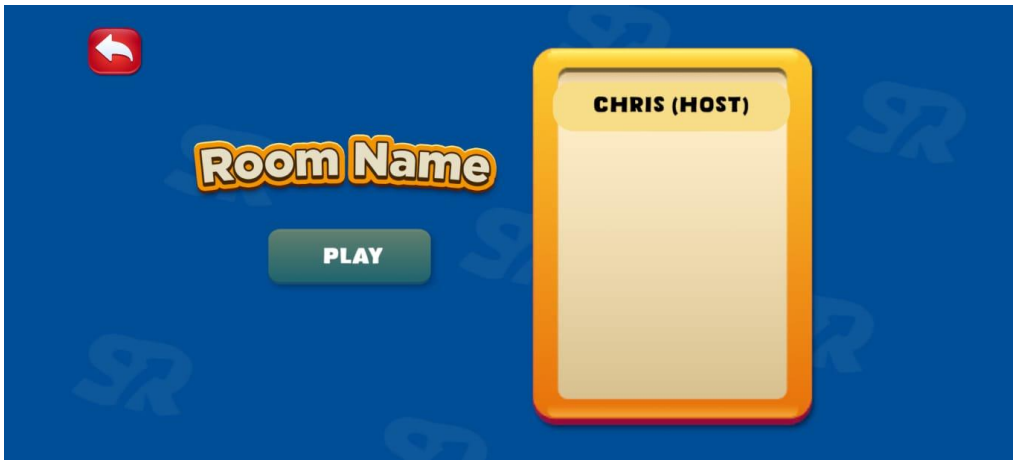
Halaman *Lobby* diimplementasikan sebagai gerbang utama pemain sebelum memasuki ruang permainan. Pada tahap ini, perangkat telah berhasil memanggil `PhotonNetwork.ConnectUsingSettings()` dan terhubung dengan *Master Server*.

Gambar 4. 2 *Screenshot* Antarmuka *Lobby*

Pada antarmuka ini, sistem memanfaatkan *callback* `OnRoomListUpdate()` untuk menampilkan daftar ruangan (*Room List*) yang tersedia secara dinamis. Pemain dapat membuat ruangan baru yang secara otomatis memicu fungsi `PhotonNetwork.CreateRoom()`, atau bergabung ke ruangan yang sudah ada menggunakan `PhotonNetwork.JoinRoom()`. Terdapat juga Validasi kata sandi untuk memastikan keamanan akses ruangan.

4.2.2 Implementasi Antarmuka *Waiting Room*

Waiting Room diimplementasikan untuk mengakomodasikan sinkronisasi pemain sebelum simulasi pasar saham dimulai.

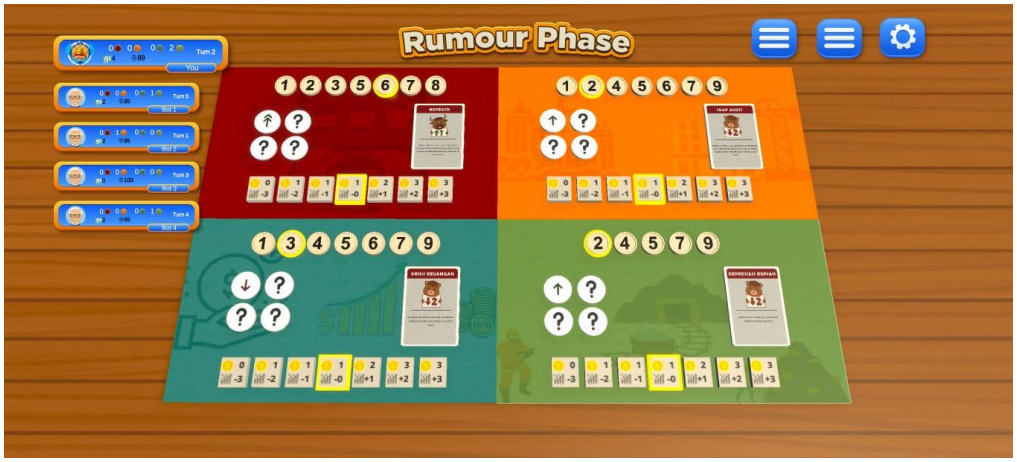


Gambar 4. 3 *Screenshot* Antarmuka *Waiting Room*

Sistem membaca data dari `PhotonNetwork.CurrentRoom.PlayerCount` untuk menampilkan daftar pemain yang berada di dalam ruangan. Daftar ini akan diperbarui secara *real-time* melalui *callback* `OnPlayerEnteredRoom()` dan `OnPlayerLeftRoom()`. Pada antarmuka ini, sistem juga mengecek status `(PhotonNetwork.IsMasterClient)` untuk memunculkan tombol *Start Game* yang hanya berhak diakses oleh *host* ketika jumlah pemain sudah memenuhi syarat batas minimal.

4.2.3 Implementasi Antarmuka Arena *Gameplay*.

Antarmuka Arena *Gameplay* merupakan visualisasi utama dari simulasi pasar saham yang sinkron antar-perangkat. Implementasi antarmuka ini sangat bergantung pada lalu lintas pertukaran data *Remote Procedure Call* (RPC).



Gambar 4. 4 *Screenshot* Antarmuka *Gameplay*

Sesuai dengan mekanik permainan, indikator fase (*Bidding*, *Action*, *Selling*, *Rumour*, dan *Resolution*) pada layar diperbarui bukan berdasarkan input manual untuk berpindah fase, melainkan dikendalikan secara otomatis oleh sistem jaringan. Ketika sistem mendeteksi bahwa kondisi permainan pada suatu fase telah terpenuhi, skrip akan mengeksekusi RPC untuk mengubah status fase dan langsung memperbarui tampilan antarmuka di seluruh *client* secara serempak tanpa jeda yang signifikan.

4.2.4 Implementasi Antarmuka *Leaderboard*

Halaman *Leaderboard* diimplementasikan sebagai layar penutup setelah siklus permainan selama 4 semester selesai dieksekusi.

Gambar 4. 5 *Screenshot* Antarmuka *Leaderboard*

Pada layar ini, antarmuka tidak melakukan perhitungan secara mandiri. *Master Client* bertugas menghitung akumulasi InvestPoin dari setiap pemain berdasarkan sisa

modal, dividen, penjualan, dan aset saham yang disimpan (*keep*). Hasil kalkulasi tersebut kemudian didistribusikan ke seluruh *client* melalui jaringan. Antarmuka *Leaderboard* kemudian menerima *array* atau struktur data tersebut dan melakukan instansiasi visual (*UI instantiation*) untuk menampilkan tabel peringkat secara berurutan dari peringkat pertama hingga kelima.

4.3 Pengujian

Pengujian pada tahap ini difokuskan pada verifikasi teknis untuk memastikan bahwa arsitektur jaringan *client-server* menggunakan *framework* PUN 2 berjalan sesuai dengan rancangan. Pengujian fungsional (Black Box Testing) dilakukan dengan menghubungkan beberapa perangkat android dalam satu jaringan internet untuk mengevaluasi kelancaran pertukaran data, manajemen ruang bermain (*Matchmaking*), sinkronisasi status permainan (*Game State*) antarklien tanpa adanya ketidaksinkronan (*Desync*).

4.3.1 Pengujian Manajemen Ruang dan Otoritas (*Matchmaking* & *Room*)

Pengujian ini mengadaptasi kebutuhan fungsional pembuatan *Lobby*, bergabung *Lobby*, dan memulai permainan guna memverifikasi kinerja Photon Server dalam mengelola ruang virtual dan validasi *Master Client*.

Tabel 4. 1 Pengujian Fungsional - *Matchmaking* & Manajemen Ruang

Test Case ID	Skenario Pengujian (Aksi)	State Akhir/Expected Output (Fokus Jaringan)	Actual Output	Status
TC-MP-01	Pemain membuat *Lobby* dengan batas 4 pemain dan mengisi *password*.	Photon Server membuat ruang, menyimpan *password* di *Custom Room Properties*, dan menetapkan pemain sebagai *Master Client*.	Ruang berhasil dibuat, pemain menjadi *Master Client.*	☑
TC-MP-02	Klien mencoba bergabung ke *Lobby* dengan memasukkan *password* yang benar.	Klien tervalidasi oleh server dan berhasil masuk ke *Waiting Room* yang sama dengan *Master Client*.	Klien berhasil masuk ke *Waiting Room*.	☑

TC-MP-03	Klien mencoba bergabung ke *Lobby* dengan memasukkan *password* yang salah.	Server menolak akses klien, koneksi dibatalkan, klien tetap di halaman *Lobby* utama.	Klien gagal masuk, notifikasi *password* salah muncul.	☑
TC-MP-04	Klien (bukan *Master Client*) menekan tombol *Play* di *Waiting Room*.	Sistem menolak RPC memulai *game*, karena otoritas memuat *scene* hanya dimiliki oleh *Master Client*	Permainan tidak dimulai, tombol tidak berfungsi untuk klien.	☑
TC-MP-05	*Master Client* menekan tombol *Play* saat kapasitas *Lobby* sudah sesuai.	*Master Client* memanggil `PhotonNetwork.LoadLevel()` , seluruh klien berpindah *scene* secara serentak.	Permainan *multiplayer* berhasil dimulai untuk semua pemain.	☑

4.3.2 Pengujian Sinkronisasi Status Permainan (*Game State* & RPC)

Pengujian ini bertujuan untuk memverifikasi keberhasilan komunikasi *Remote Procedure Call* (RPC) dalam menyinkronkan data antarperangkat pada fase-fase kritis permainan.

Tabel 4. 2 Pengujian Fungsional - Sinkronisasi Fase dan Data Ekonomi

Test Case ID	Skenario Pengujian (Aksi)	*State* Akhir/*Expected Output* (Fokus Jaringan)	*Actual Output*	Status
TC-MP-06	Seluruh pemain memilih kartu giliran secara serentak pada fase *Bidding*.	RPC mengirim data pilihan kartu ke *Master Client*. Sistem mencegah benturan (*collision*) data untuk kartu yang sama.	Urutan giliran tersinkronisasi akurat tanpa ada pemain yang mendapat giliran ganda.	☑
TC-MP-07	*Master Client* memicu transisi	*Master Client* mengirimkan `photonView.RPC()` untuk transisi fase. Seluruh klien	Transisi fase serentak di semua	☑

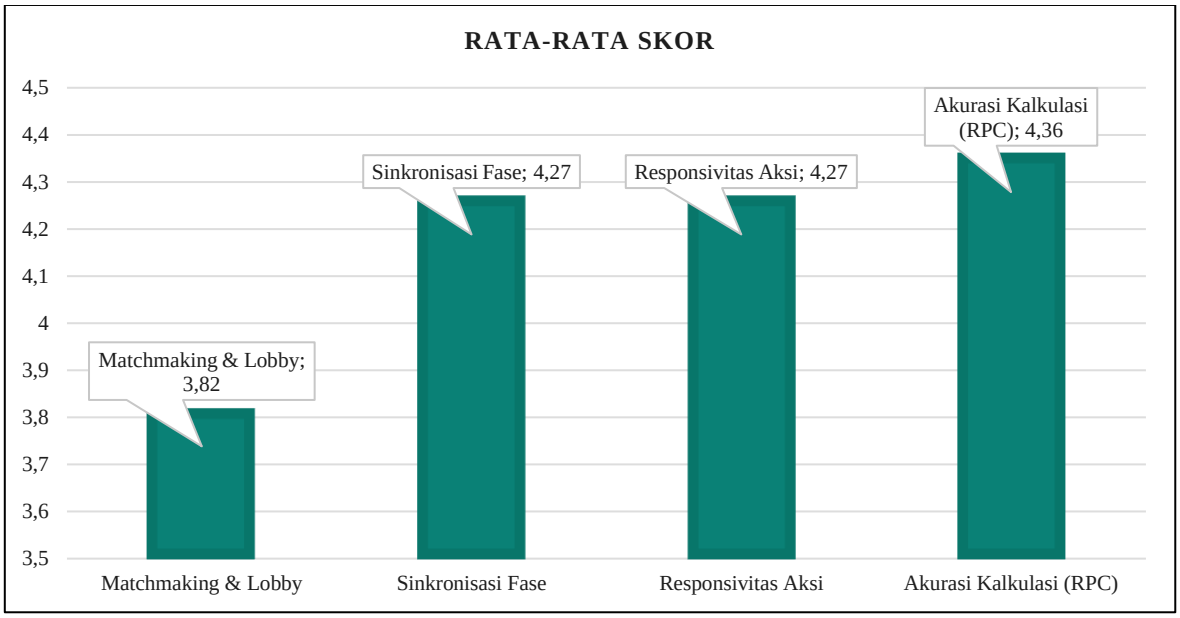
	dari fase *Action* ke fase *Selling*.	berpindah *state* dalam waktu bersamaan.	perangkat tanpa *desync*.	
TC-MP-08	Sistem membuka kartu *Rumour* yang mempengaruhi nilai saham di pasar.	RPC Mendistribusikan ID efek *rumour*, nilai aset di UI seluruh klien berubah secara *real-time*.	Grafik dan harga saham berubah sinkron di semua perangkat.	☑
TC-MP-09	*Master Client* menghitung dividen untuk seluruh pemain di fase *Resolution*.	Data dividen dihitung terpusat, lalu RPC menginstruksikan seluruh perangkat untuk *update* saldo InvestPoin.	Antarmuka InvestPoin di setiap klien bertambah secara persis sesuai instruksi *Master Client*.	☑

4.3.3 Hasil Pengujian User Acceptance Testing (UAT)

Pengujian *User Acceptance Testing* (UAT) dilakukan untuk mengevaluasi stabilitas dan kinerja arsitektur jaringan *multiplayer* secara fungsional berdasarkan pengalaman langsung dari pengguna. Pengujian ini melibatkan 11 responden eksternal yang melakukan simulasi permainan "Stock Rising" secara serentak. Pengumpulan data evaluasi dilakukan menggunakan instrumen kuesioner dengan skala penilaian 1 hingga 5. Secara keseluruhan, implementasi *framework* Photon Unity Networking (PUN) 2 menunjukkan kinerja yang sangat memuaskan dalam memfasilitasi arsitektur *client-server*. Hasil rata-rata penilaian metrik pengujian disajikan sebagai berikut:

a. Proses *Matchmaking* & *Lobby*: 3.82 / 5.00
b. Sinkronisasi Fase Permainan (*Game State*): 4.27 / 5.00
c. Responsivitas Aksi & Transaksi: 4.27 / 5.00
d. Akurasi Kalkulasi Data (RPC): 4.36 / 5.00

Gambar 4. 6 Grafik Rata-Rata Hasil Pengujian UAT



Selain penilaian kuantitatif, pengujian ini juga mengumpulkan umpan balik kualitatif yang mengonfirmasi beberapa batasan masalah dan evaluasi sistem, antara lain:

A. Validasi Batasan Kapasitas Jaringan

Terdapat responden yang melaporkan kegagalan dalam membuat ruang bermain (*Room*) baru saat pengujian berlangsung serentak. Temuan ini secara empiris memvalidasi batasan maksimal 20 *Concurrent Users* (CCU) yang diterapkan pada infrastruktur jaringan gratis dari server Photon Cloud.

B. Evaluasi Transisi Otoritas (*Host Migration*)

Hasil pengujian menemukan adanya kendala teknis saat pembuat ruangan (*Master Client*) keluar dari permainan. Meskipun sistem menerapkan fungsi `OnMasterClientSwitched()` untuk memindahkan otoritas jaringan, transisi status permainan (*Game State*) ke klien yang baru belum berjalan secara optimal, yang mengakibatkan berhentinya alur permainan. Temuan ini menjadi evaluasi batas kemampuan sistem sinkronisasi saat ini.

C. Latensi Komunikasi Jaringan

Terdapat laporan mengenai fluktuasi kecepatan respons (*lag*) pada antarmuka permainan. Hal ini merupakan variabel eksternal yang wajar terjadi pada arsitektur *client-server*, di mana kelancaran pengiriman paket data *Remote Procedure Call* (RPC) sangat bergantung pada kualitas dan stabilitas penyedia layanan internet di masing-masing perangkat klien.

D. Kebutuhan Panduan Pengguna (*Onboarding*)

Terdapat masukan kualitatif yang menyarankan penambahan *scene tutorial*. Meskipun aspek ini berada di luar ruang lingkup implementasi jaringan *backend*, temuan ini mengindikasikan bahwa simulasi investasi dengan mekanik yang kompleks memerlukan panduan interaktif sebelum pemain masuk ke dalam sesi *multiplayer* yang kompetitif.

BAB V KESIMPULAN DAN SARAN

5.1 Kesimpulan

Berdasarkan seluruh proses perancangan, implementasi, hingga pengujian fungsional pada mode *multiplayer real-time game* "Stock Rising", dapat diambil kesimpulan sebagai berikut:

a. Implementasi arsitektur jaringan *client-server* menggunakan *framework* Photon Unity Networking (PUN) 2 berhasil menyediakan sistem *matchmaking* dan manajemen ruang (*room*) yang stabil. Fitur pembuatan ruang (*Create Room*) dengan proteksi kata sandi (*password*) serta pencarian ruang (*Join Room*) berfungsi akurat melalui pemanfaatan *Custom Room Properties*.
b. Antarmuka jaringan pada halaman *Lobby* dan *Waiting Room* telah berhasil diintegrasikan dan mampu merespons status koneksi pemain secara *real-time*. Sistem juga terbukti andal dalam menangani skenario *Host Migration* melalui fungsi `OnMasterClientSwitched()` , sehingga permainan tetap dapat berlanjut meskipun *Master Client* awal terputus dari jaringan.
c. Penggunaan komunikasi *Remote Procedure Call* (RPC) dalam skrip C# terbukti efektif dalam menyinkronkan status permainan (*Game State*) antarpemain pada kelima fase utama (*Bidding, Action, Selling, Rumour*, dan *Resolution*). Mekanisme validasi terpusat pada *Master Client* berhasil mendistribusikan data kalkulasi ekonomi dan transisi fase secara presisi, sehingga permainan terhindar dari kendala ketidaksinkronan data (*desync*).

5.2 Saran

Meskipun sistem *multiplayer* pada *game* "Stock Rising" telah berfungsi sesuai rancangan, terdapat beberapa aspek yang dapat dikembangkan lebih lanjut di masa mendatang, antara lain:

a. Optimasi Infrastruktur Jaringan: Melakukan migrasi atau peningkatan layanan server untuk mengatasi batasan *Concurrent User* (CCU) pada versi gratis Photon Cloud agar aplikasi dapat menampung jumlah pemain yang lebih besar secara bersamaan.
b. Pengembangan Lintas Platform: Mengembangkan dan menguji aplikasi untuk sistem operasi iOS dan *desktop* agar simulasi investasi ini dapat menjangkau basis pengguna yang lebih luas.

c. Integrasi Data *Real-Time*: Menambahkan fitur integrasi data pasar saham dunia nyata melalui API (*Application Programming Interface*) untuk memberikan pengalaman simulasi yang lebih edukatif, dinamis, dan relevan dengan kondisi pasar aktual.
d. Peningkatan Fitur Sosial: Menambahkan fitur komunikasi antar pemain di dalam *Waiting Room* atau saat permainan berlangsung (seperti fitur *chat* atau *emotes*) untuk meningkatkan aspek interaksi sosial dan pengalaman bermain yang lebih interaktif.

DAFTAR PUSTAKA

Abimanyu, A., & Rizqi, M. (2024). Penerapan Teknologi Multiplayer pada Sistem Pembelajaran Berbasis Virtual Reality. *Jurnal Ilmu Komputer dan Bisnis*, *15*(2), 235–246. https://doi.org/10.47927/jikb.v15i2.830

Coprich, Z. (2022). Client-Server and Peer-to-Peer Architectures in Multiplayer Games. *JSU Student Symposium 2022*. Diambil dari https://digitalcommons.jsu.edu/ce_jsustudentsymp_2022/52

Mahendra, K. C. (2022). Pengaruh literasi keuangan terhadap perilaku belanja dan perilaku investasi. *Fakultas Bisnis dan Ekonomika, Universitas Islam Indonesia*.

Nauval, M., Ikhwan, R., & Syahru, R. (2021). Rancang Bangun Game Edukasi Berbasis Android Sebagai Media Pembelajaran Budaya Indonesia Menggunakan Unity Engine. *Coding : Jurnal Komputer dan Aplikasi*, *09*(3), 491–500.

Otoritas Jasa Keuangan (OJK). (2022). *Penguatan Sektor Jasa Keuangan Untuk Mendukung Percepatan Pemulihan Ekonomi Nasional dan Pertumbuhan Ekonomi Baru*. *Laporan Tahunan Annual Report 2022*. Diambil dari www.ojk.go.id

Pitriani, N. W., Dantes, N., & Sariyasa. (2024). Game Based Learning Berorientasi Kahoot! Meningkatkan Motivasi Belajar Siswa Kelas V Sekolah Dasar. *Didaktika: Jurnal Kependidikan*, *13*(1), 643–650. Diambil dari https://jurnaldidaktika.org/643

Pressman, R. S., & Maxim, B. R. (2020). Software engineering: A practitioner's approach. McGraw-Hill Education. *Data Science for Software Engineers*, *9*, 629–638.

Rosa, I., & Listiadi, A. (2020). Effects of financial literacy, financial education on family, peers, and self control on personal financial management. *Jurnal Manajemen*, *12*(2), 244–252.

Sarwodi, S. S., Sukmo Wardhono, W., & Akbar, M. A. (2020). Penerapan Multiplayer Pada Gim Tower Defense Menggunakan Photon Unity. *Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer*, *4*(9), 3235–3245. Diambil dari http://j-ptiik.ub.ac.id

To, S. (2024). PUN's Structure. *Photon Unity Networking (PUN) Article*, 1–7.

Utami, F., Sari, T. N., Selvi, Pratiwi, L. N., Paramita, R. A. S., Rasjid, H., … Hamin, D. I. (2022). *Pengantar Pasar Modal*. Penerbit Tahta Media.

DAFTAR LAMPIRAN

Link *file* skrip proyek Stock Rising mode *Multiplayer*:

https://polibatam.id/StockRisingMultiplayerScript

Link *file* keseluruhan proyek Stock Rising:

https://polibatam.id/StockRisingGithub

Link video presentasi Stock Rising:

https://polibatam.id/StockRisingPresentationVideo

Link video demo Stock Rising:

https://polibatam.id/StockRisingDemoVideo

Link video testing Stock Rising:

https://polibatam.id/StockRisingTestingVideo

Link file presentasi Stock Rising:

https://polibatam.id/StockRisingPresentationFile

Link poster Stock Rising:

https://polibatam.id/StockRisingPosterFile

Link aplikasi Stock Rising:

https://polibatam.id/StockRisingApp