LAPORAN FINAL PROJECT

PEMROGRAMAN JARINGAN (C)

Implementasi Game Checkers Multiplayer Berbasis TCP/IP dengan Arsitektur Client-Server

Anggota Kelompok:

Decya Giovanni	5025221027
Karina Rahmawati	5025221041
Audrey Sasqhia Wijaya	5025221055
Putri Meyliya Rachmawati	5025221062
Lathifah Sahda	5025221159

DEPARTEMEN TEKNIK INFORMATIKA

FAKULTAS TEKNOLOGI DAN INFORMATIKA CERDAS

INSTITUT TEKNOLOGI SEPULUH NOPEMBER

SURABAYA

2025

Deskripsi Game

Game yang dikembangkan adalah Checkers, sebuah game dengan genre strategy turn-based multiplayer. Game ini dimainkan oleh dua pemain dengan tujuan mengalahkan lawan melalui strategi melompati dan mengurangi jumlah bidak lawan. Pada permainan ini, setiap pemain menggerakkan bidaknya secara bergantian karena permainan bersifat turn-based. Pemain dapat melakukan lompatan (jump) untuk mengeliminasi bidak milik lawan, dan pemain yang kehilangan seluruh bidaknya akan dinyatakan kalah. Selain itu, jika sebuah bidak berhasil mencapai ujung papan milik lawan, bidak tersebut akan berubah menjadi king dan mendapatkan kemampuan bergerak lebih fleksibel.

Seluruh permainan disajikan dalam antarmuka grafis yang interaktif dan menarik menggunakan library Pygame. Game Checkers dipilih karena memiliki gameplay yang sederhana namun menantang dari sisi strategi. Selain itu, game ini cocok untuk diimplementasikan dalam sistem jaringan client-server karena pertukaran data seperti status papan dan giliran dapat dilakukan secara efisien menggunakan protokol TCP. Sistem ini menggunakan HTTP server berbasis thread pool yang memungkinkan server memproses banyak request secara paralel tanpa harus membuat thread baru setiap saat. Dengan demikian, sinkronisasi permainan tetap terjaga meskipun kedua client terus melakukan polling dalam interval waktu yang singkat.

Pembagian Tugas

No.	Nama	NRP	Deskripsi
1	Decya Giovanni	5025221027	Implementasi server utama, manajemen thread pool, dan protokol komunikasi TCP/IP
2	Karina Rahmawati	5025221041	Implementasi komunikasi client-server (HTTP request/response, JSON handling)
3	Audrey Sasqhia Wijaya	5025221055	Desain dan implementasi antarmuka client menggunakan Pygame
4	Putri Meyliya Rachmawati	5025221062	Logika permainan (aturan Checkers,

			validasi gerakan, perubahan ke King)
5	Lathifah Sahda	5025221159	Implementasi server utama, dokumentasi proyek, dan pengujian fungsionalitas

Repository GitHub

https://github.com/aikoextended/Progjar-Tugas-EAS

Definisi Protokol Multiplayer

Protokol multiplayer yang digunakan pada game Checkers ini dirancang untuk mendukung komunikasi antara client dan server dengan menggunakan protokol TCP/IP. Server membuka koneksi pada port 8080, menerima permintaan dari client melalui protokol HTTP, dan memprosesnya dengan format pesan bertipe JSON. Seluruh interaksi dilakukan secara stateless dengan polling berkala dari client untuk memastikan sinkronisasi status permainan. Komunikasi antara client dan server menggunakan TCP/IP sebagai dasar koneksi, yang diimplementasikan dalam server_thread_pool_http.py menggunakan socket TCP (SOCK_STREAM). Di atas TCP, protokol HTTP digunakan untuk mengirim dan menerima data antar client dan server.

Format komunikasi antar client dan server menggunakan struktur JSON untuk memastikan interoperabilitas dan kemudahan parsing data. Contoh format pesan:

- Request Join Game (POST /join_game)

  ```
  (body kosong)
  ```

- Response Join Game

  ```
  {
    "player_id": "d7051c5a-0d8c-4db2-b9b6-xxxx",
    "game_id": "1",
    "status": "game_started"
  }
  ```

- Request Make Move (POST /make_move)

  ```
  {
    "game_id": "1",
    "player_id": "d7051c5a-...",
  ```

```
  "from": [2, 3],
  "to": [3, 4]
}
```

- Response Game State (GET /game_state?game_id=1&player_id=...)

```
{
  "game_id": "1",
  "board": [...],
  "current_player": 1,
  "game_state": "playing",
  "your_turn": true,
  ...
}
```

Alur komunikasi antara client dan server dalam game Checkers ini dimulai ketika client pertama kali mengirim permintaan POST /join_game untuk bergabung ke dalam permainan. Server kemudian membalas dengan memberikan player_id dan game_id sebagai identitas unik pemain dan sesi permainan. Setelah itu, client secara berkala melakukan polling ke server menggunakan permintaan GET /check_status?player_id=... untuk mengecek apakah sudah ada lawan bermain. Jika sudah, server akan merespons dengan status game_started dan menyertakan game_id yang digunakan client untuk memulai polling status permainan.

Setelah permainan dimulai, client terus melakukan polling setiap 200 milidetik ke endpoint GET /game_state?game_id=...&player_id=... untuk memperoleh pembaruan tentang papan permainan, giliran pemain, skor, sisa bidak, serta informasi apakah giliran saat ini milik client tersebut. Ketika pemain melakukan gerakan di papan, client mengirimkan permintaan POST /make_move dengan data posisi asal dan tujuan bidak yang digerakkan. Server kemudian memvalidasi gerakan tersebut, memperbarui status permainan jika valid, dan membalas dengan kondisi permainan terbaru.

Jika permainan telah selesai, kedua client masih bisa meminta untuk memulai ulang permainan dengan mengirim POST /restart_game disertai player_id dan game_id. Server akan mencatat permintaan tersebut, dan hanya akan mereset permainan apabila kedua pemain telah menyetujui permintaan restart. Seluruh komunikasi dilakukan melalui protokol HTTP di atas TCP/IP, menggunakan format data JSON, dengan pendekatan polling dari sisi client untuk menjaga sinkronisasi status permainan.

Handling Disconnect, Timeout, dan Restart:

- Disconnect: Server tidak memiliki sistem eksplisit untuk mendeteksi disconnect. Karena komunikasi berbasis HTTP polling, jika client berhenti mengirim request (background_updater), maka dianggap tidak aktif. Tidak ada timeout otomatis.
- Timeout: Tidak ada timeout sisi server untuk mematikan game atau mengganti pemain secara otomatis. Namun, karena arsitektur berbasis polling, client bisa kembali sewaktu-waktu.
- Restart: Sistem mendukung restart game oleh kedua pemain. Jika salah satu pemain mengklik tombol restart (RESTART GAME), client mengirim POST /restart_game. Game hanya di-restart jika kedua pemain menyetujui.

Arsitektur Implementasi

Sistem game Checkers ini menggunakan arsitektur client-server berbasis HTTP. Dalam arsitektur ini, server bertindak sebagai koordinator pusat yang mengatur seluruh jalannya permainan, termasuk manajemen pemain, giliran, validasi gerakan, dan status permainan. Server menyimpan seluruh state permainan dan hanya ia yang memiliki otoritas untuk mengubahnya.

Setiap client berperan sebagai antarmuka pengguna yang menerima masukan dari pemain, seperti memilih dan memindahkan bidak. Saat pemain melakukan tindakan, client mengirimkan data tersebut ke server melalui request HTTP (POST /make_move), dan server akan memproses serta memvalidasi gerakan tersebut. Jika valid, state permainan di-update dan akan tersedia saat client berikutnya memanggil GET /game_state.

Meskipun tidak ada sistem push atau socket real-time, server tetap mem-broadcast perubahan state secara pasif melalui metode polling yang dilakukan client secara berkala (setiap 200 ms). Client akan menerima update terbaru saat memanggil API game_state, yang berisi kondisi terkini permainan.

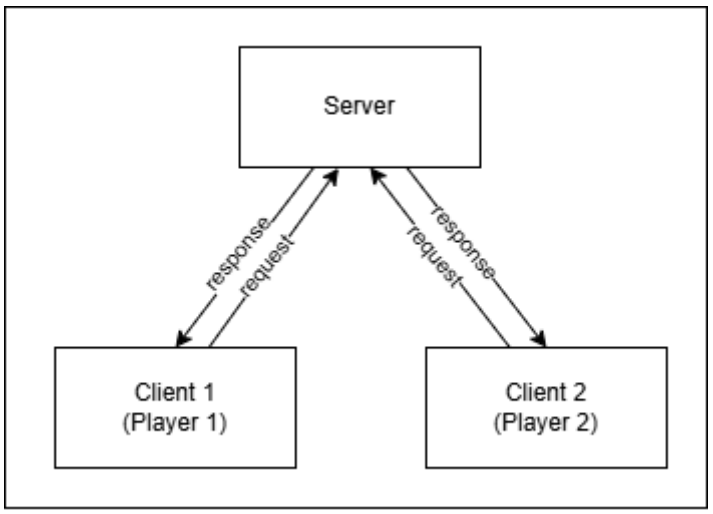
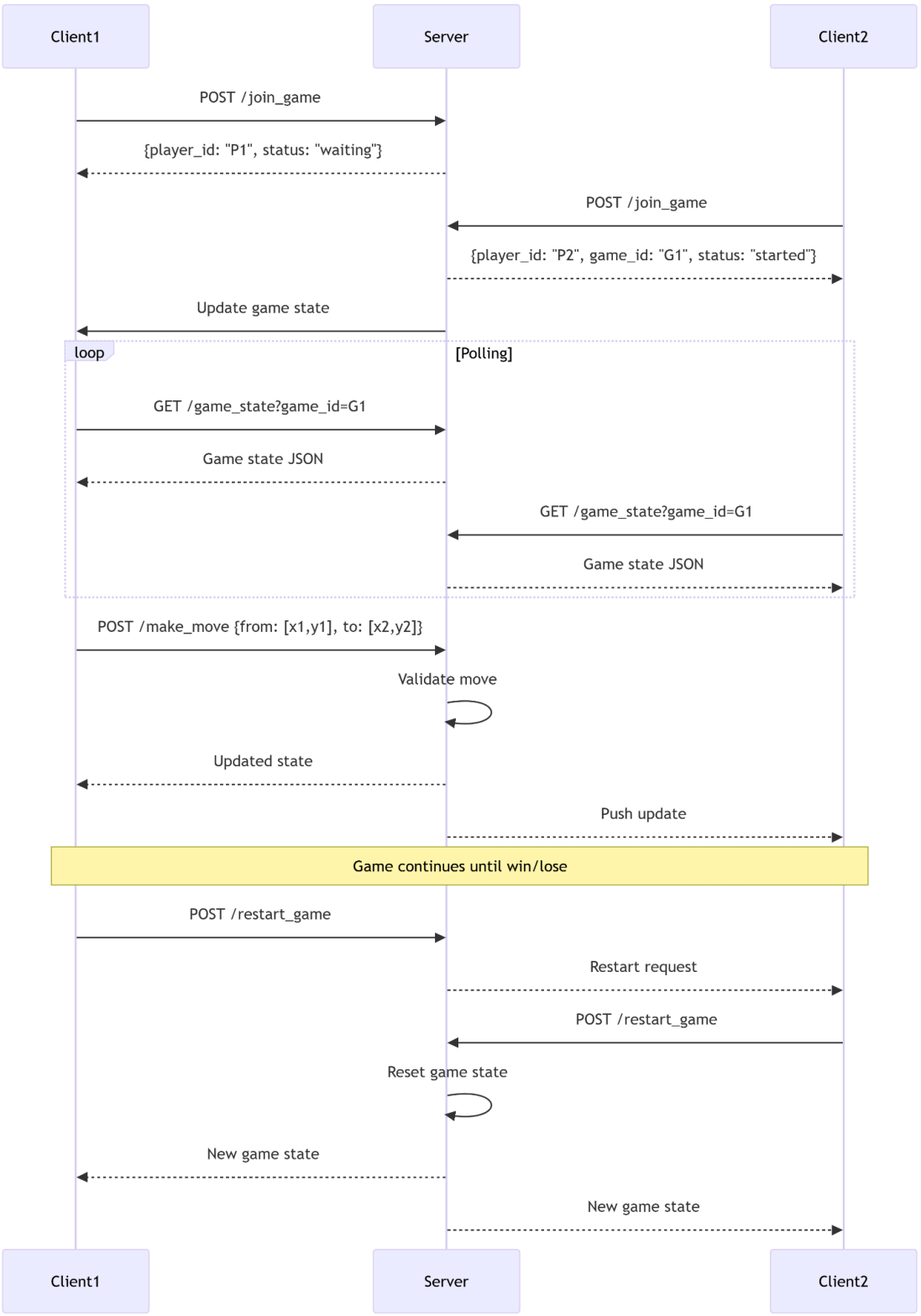


Diagram Arsitektur Client-Server



Sequence Diagram

IP Address dan Port

- IP server: localhost
- IP client 1: localhost
- IP client 2: localhost
- Port TCP/IP: 8080

Instruksi Menjalankan Server dan Client

Untuk menjalankan game ini, ikuti langkah-langkah berikut:

a. Requirements/Pustaka yang Dibutuhkan:
   i. Python 3.x
   ii. pygame (untuk sisi client)
   iii. socket, json, threading, uuid, time, queue, copy, enum (pustaka standar Python)

b. Platform OS yang Digunakan untuk Pengujian:
   i. Windows
   ii. Linux (Ubuntu)

c. Langkah-langkah Menjalankan:
   i. Langkah 1: Jalankan File Server

      Buka terminal atau command prompt dan navigasikan ke direktori tempat server_thread_pool_http.py dan http_server.py berada. Jalankan server dengan perintah:

      ```
      python server_thread_pool_http.py
      ```

   ii. Langkah 2: Jalankan File Client

      Buka dua terminal atau command prompt terpisah (atau pada perangkat yang berbeda) dan navigasikan ke direktori tempat client.py berada. Jalankan masing-masing client dengan perintah:

      ```
      python client.py [server_host] [server_port]
      ```

      Ganti [server_host] dengan alamat IP server (misalnya localhost jika berjalan di mesin yang sama) dan ganti [server_port] dengan 8080.

   iii. Ulangi perintah ini di terminal kedua untuk menjalankan client kedua.

      Pastikan semua berada di jaringan yang sama: Jika Anda menjalankan client di perangkat yang berbeda, pastikan semua perangkat terhubung ke jaringan yang

sama dan dapat saling berkomunikasi. Gunakan alamat IP server yang benar untuk argumen server_host.

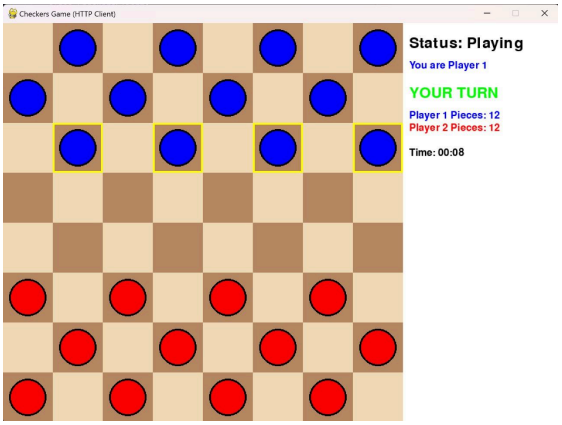
Kesimpulan

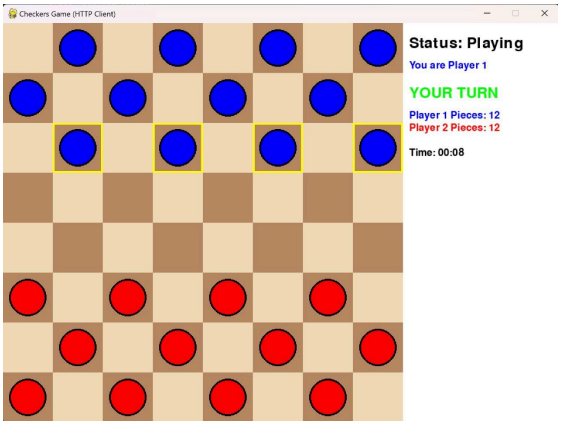
Implementasi game Checkers multiplayer berbasis TCP/IP dengan arsitektur client-server ini berhasil merealisasikan fungsionalitas inti permainan yang memungkinkan dua pemain saling berinteraksi secara real-time melalui jaringan. Protokol HTTP digunakan di atas koneksi TCP/IP dengan format pesan JSON untuk mendukung komunikasi yang terstruktur dan mudah diolah. Server berperan sebagai pusat pengelolaan game state, validasi langkah, serta manajemen giliran, sedangkan client menyajikan antarmuka grafis interaktif menggunakan Pygame untuk menerima input pemain dan menampilkan status permainan secara dinamis.

Untuk menjaga performa sistem dan respons terhadap polling berkala dari client, server diimplementasikan menggunakan HTTP server berbasis thread pool. Mekanisme ini memungkinkan server menangani banyak request secara paralel tanpa membuat thread baru setiap kali permintaan masuk, sehingga sinkronisasi status permainan tetap lancar. Proyek ini menunjukkan keberhasilan integrasi antara logika permainan strategi dan komunikasi jaringan, serta mencerminkan pemahaman teknis tentang implementasi aplikasi multiplayer berbasis client-server secara efisien.

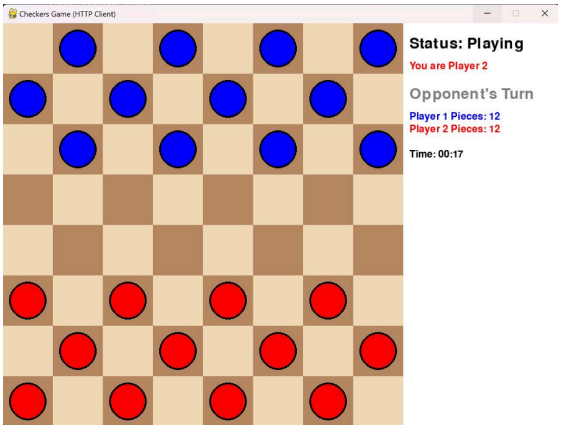
Lampiran

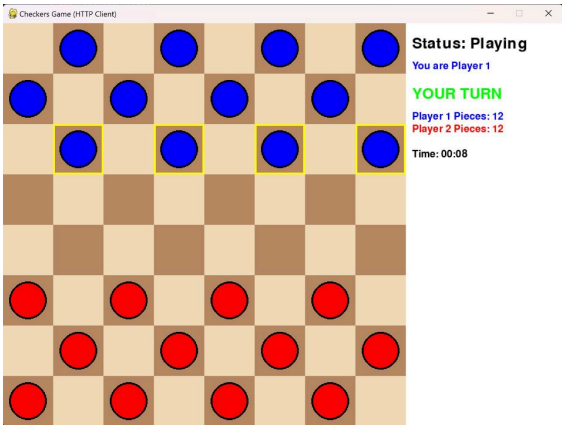
Start

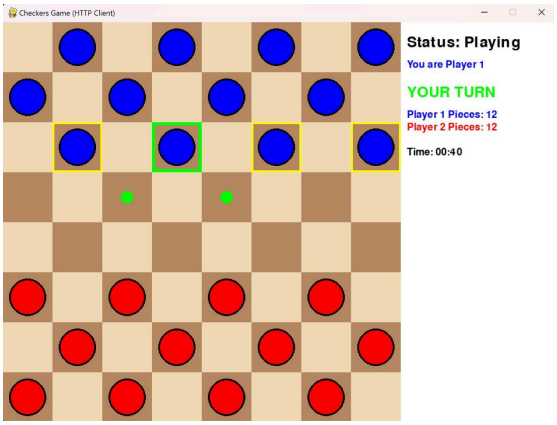


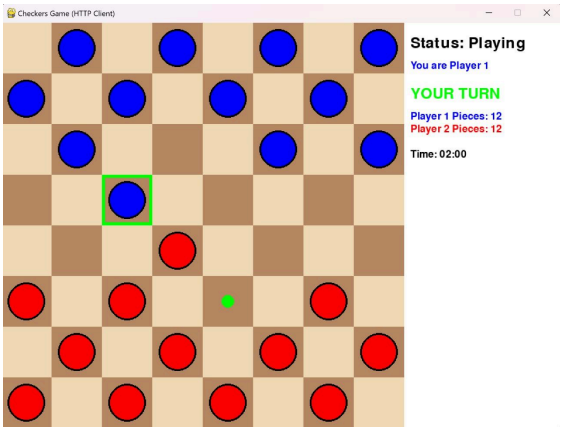


Step

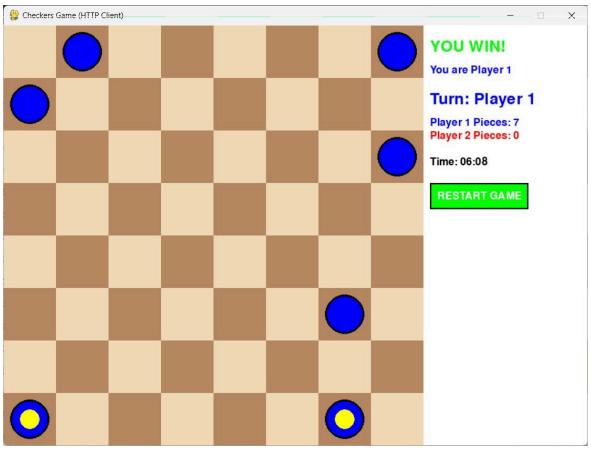


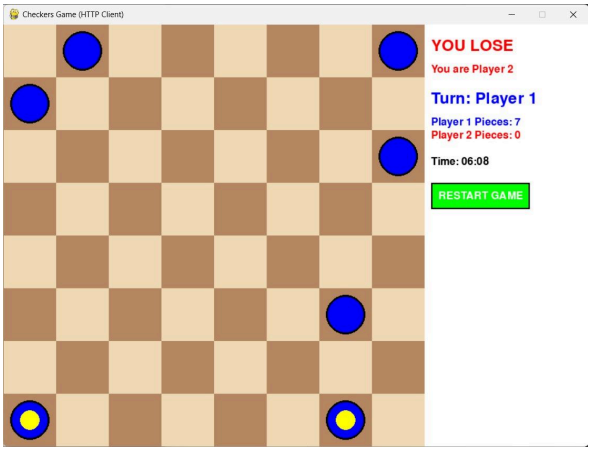






End





Request Restart