LAPORAN TUBES PEMROGRAMAN

Aplikasi Game Pokemon Airstrike

Oleh:
Kelompok 2:

HAIDAR RAFLI (18123002)
KESHA MUFRIH RAMADHAN (18123003)
ZHAFAR UMAR (18123004)
FADHIL AGLY HAKIM (18123005)

TEKNIK TELEKOMUNIKASI
SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG

2024

Daftar Isi

I. PENDAHULUAN

A. Latar Belakang

Industri game terus berkembang dengan kehadiran berbagai jenis permainan yang memanfaatkan teknologi interaktif dan memberikan pengalaman bermain yang menarik. Salah satu genre yang banyak diminati adalah game aksi dan petualangan yang menggabungkan berbagai elemen fiksi populer, seperti karakter-karakter ikonik dan alur cerita yang seru. Game seperti ini tidak hanya menarik bagi pemain yang mencari hiburan, tetapi juga bagi mereka yang ingin merasakan tantangan dan kompetisi dalam berbagai bentuk gameplay.

Proyek ini bertujuan untuk mengembangkan game berbasis web dengan konsep Airstrike, yang memadukan elemen permainan pertempuran udara dengan karakter-karakter dari dunia Pokemon. Dalam game ini, pemain terlibat dalam pertempuran udara melawan musuh dan dapat menggunakan power-up untuk meningkatkan kemampuan bertarung.

Pokemon, sebagai franchise global yang terkenal, dikenal dengan berbagai karakter unik dan konsep pertarungan antar makhluk yang saling berkompetisi. Di sisi lain, Airstrike merupakan genre game klasik yang berfokus pada pertempuran udara di mana pemain harus menghancurkan musuh sambil menghindari tembakan musuh dan mengumpulkan power-up untuk meningkatkan kemampuan bertarung.

Dalam game Pokemon Airstrike, pemain hanya akan menggunakan satu jenis power-up, yang memberikan keuntungan strategis saat bertempur. Game ini dikembangkan menggunakan teknologi web dengan backend berbasis Flask dan SQLAlchemy untuk manajemen database, serta WebSocket untuk komunikasi real-time antar pemain. Antarmuka pengguna dibangun menggunakan HTML, CSS, dan JavaScript untuk memastikan tampilan yang responsif dan interaktif.

Proyek ini tidak hanya bertujuan untuk menghasilkan permainan yang menghibur, tetapi juga untuk memperdalam pemahaman tentang pengembangan aplikasi berbasis web yang melibatkan integrasi antara berbagai teknologi, termasuk backend, frontend, dan komunikasi real-time. Game ini dirancang agar dapat diakses dengan mudah melalui browser, menawarkan pengalaman bermain yang menyenangkan, dan dapat dimainkan dengan kontrol yang sederhana namun menantang.

B. Tujuan

Tujuan dari proyek Pokemon Airstrike ini adalah untuk mengembangkan sebuah game berbasis web yang menggabungkan elemen-elemen klasik dari genre Airstrike dengan karakter-karakter populer dari dunia Pokemon. Proyek ini bertujuan untuk mencapai beberapa hal, antara lain:

1. Membuat Game Interaktif dan Menarik.
   Menghasilkan sebuah game yang menyenangkan dan menantang bagi pemain, dengan gameplay yang seru dan power-up yang meningkatkan pengalaman bermain.

2. Menerapkan Teknologi Web
   Menggunakan teknologi web modern seperti Flask untuk backend, SQLAlchemy untuk manajemen database, dan WebSocket untuk komunikasi real-time antara pemain. Hal ini

bertujuan untuk memperdalam pemahaman dan pengalaman dalam pengembangan aplikasi web.

3. Membangun Sistem Leaderboard
   Menyediakan sistem leaderboard yang dapat menampilkan peringkat pemain berdasarkan skor tertinggi yang dicapai dalam permainan, dengan penyimpanan data di database MySQL.

4. Memperkenalkan Penggunaan Power-up dalam Gameplay
   Mengimplementasikan satu jenis power-up dalam game yang memberikan keuntungan strategis bagi pemain saat bertempur melawan musuh.

5. Menawarkan Pengalaman Pengguna yang Responsif
   Membangun antarmuka pengguna yang responsif dan interaktif menggunakan HTML, CSS, dan JavaScript, agar dapat dimainkan dengan nyaman di berbagai perangkat, terutama di layar desktop.

6. Meningkatkan Kemampuan dalam Pengembangan Aplikasi Web
   Proyek ini bertujuan sebagai sarana pembelajaran untuk memperdalam pengetahuan dalam pengembangan aplikasi berbasis web, serta integrasi antara backend, frontend, dan komunikasi real-time menggunakan SocketIO.

Dengan tercapainya tujuan-tujuan tersebut, diharapkan game Pokemon Airstrike dapat memberikan pengalaman bermain yang menarik sekaligus menjadi contoh penerapan teknologi dalam pengembangan aplikasi berbasis web.

C. Ruang Lingkup

Proyek ini berfokus pada pengembangan sebuah game berbasis web yang menggabungkan elemen-elemen dari genre Airstrike dengan karakter-karakter dari dunia Pokemon. Game ini dirancang untuk dimainkan secara individual dengan fokus pada pertempuran udara dan pengumpulan power-up untuk meningkatkan kemampuan bertarung. Adapun ruang lingkup proyek ini meliputi beberapa hal sebagai berikut:

1. Pengembangan Game Web
   Game ini akan diakses melalui browser dan dirancang dengan tampilan antarmuka yang responsif, sehingga dapat dimainkan dengan nyaman di perangkat desktop.

2. Fitur Gameplay
   Pemain akan mengendalikan karakter dalam pertempuran udara melawan musuh. Dalam hal ini, hanya ada satu jenis power-up yang dapat digunakan untuk meningkatkan kemampuan pemain selama bertempur.

3. Pengelolaan Skor dan Leaderboard
   Sistem leaderboard akan menampilkan 10 pemain teratas berdasarkan skor tertinggi yang diperoleh selama bermain. Data pemain dan skor akan disimpan dalam database yang dikelola oleh SQLAlchemy dan dapat diakses dengan mudah oleh pemain.

4. Implementasi Backend dan Frontend
   Backend game ini dibangun menggunakan Flask, sementara frontend menggunakan HTML, CSS, dan JavaScript. Game ini juga mengintegrasikan penggunaan WebSocket untuk komunikasi real-time antar pemain.

5. Batasan Pengembangan
   Proyek ini tidak akan mencakup pengembangan fitur multiplayer atau pengenalan berbagai power-up lebih dari satu jenis. Fokus utamanya adalah menciptakan gameplay yang stabil dengan sistem leaderboard berbasis skor.

6. Pengujian dan Perbaikan

Pengujian akan dilakukan untuk memastikan fungsionalitas game, terutama dalam hal interaksi pengguna, pengelolaan database, dan tampilan antarmuka. Perbaikan dan penyempurnaan akan dilakukan berdasarkan hasil pengujian dan umpan balik yang diterima.

Dengan ruang lingkup yang telah ditentukan, proyek ini bertujuan untuk memberikan pemahaman yang lebih mendalam mengenai pengembangan game berbasis web, penggunaan teknologi backend, dan penerapan sistem database dalam konteks permainan.

D. Metodologi

Dalam pengembangan game Pokemon Airstrike, metodologi yang digunakan mengacu pada pendekatan Iteratif dan Incremental yang memungkinkan pengembangan bertahap dengan fokus pada evaluasi dan perbaikan berkelanjutan. Proyek ini mengikuti beberapa tahap utama sebagai berikut:

1. Perencanaan dan Analisis Kebutuhan
   Pada tahap awal, dilakukan identifikasi kebutuhan proyek, termasuk fitur-fitur utama yang ingin diterapkan dalam game, seperti gameplay pertempuran udara, penggunaan power-up, dan sistem leaderboard. Analisis kebutuhan juga meliputi pemilihan teknologi yang tepat untuk pengembangan, yaitu Flask untuk backend, SQLAlchemy untuk database, dan WebSocket untuk komunikasi real-time.

2. Desain dan Arsitektur Sistem
   Setelah kebutuhan dikumpulkan, dilakukan perancangan arsitektur sistem, baik untuk backend maupun frontend. Desain antarmuka pengguna (UI) difokuskan pada kenyamanan dan responsivitas, memastikan pengalaman bermain yang optimal di berbagai perangkat. Selain itu, desain database dibuat untuk menyimpan data pengguna dan skor dengan sistem yang efisien.

3. Pengembangan dan Implementasi
   Tahap ini mencakup pengembangan fitur-fitur utama game, yaitu:
   a. Backend: Pembuatan API untuk pengelolaan data pemain dan skor, serta penghubungan ke database menggunakan SQLAlchemy.
   b. Frontend: Pengembangan tampilan antarmuka menggunakan HTML, CSS, dan JavaScript, termasuk implementasi fitur-fitur seperti tampilan permainan, tombol power-up, dan sistem leaderboard.
   c. Fitur Komunikasi Real-time: Penggunaan WebSocket untuk memastikan interaksi antar pemain berjalan lancar dan tanpa jeda.
4. Pengujian (Testing)
   Setelah pengembangan fitur selesai, dilakukan pengujian untuk memastikan bahwa semua sistem berjalan dengan baik. Pengujian ini meliputi:
   a. Pengujian Fungsionalitas: Memastikan bahwa game berjalan sesuai dengan aturan dan fitur yang diinginkan, seperti pertempuran udara, penggunaan power-up, dan pencatatan skor.
   b. Pengujian Performa: Mengukur kinerja game, baik dari sisi frontend maupun backend, untuk memastikan game dapat dimainkan tanpa hambatan.
   c. Pengujian Pengalaman Pengguna: Memastikan tampilan antarmuka mudah digunakan dan responsif pada perangkat desktop.
5. Evaluasi dan Perbaikan
   Berdasarkan hasil pengujian, dilakukan evaluasi terhadap kinerja game dan pengalaman pengguna. Setiap masalah atau bug yang ditemukan akan diperbaiki pada tahap ini, dan pengembangan fitur tambahan akan dilakukan jika diperlukan.

6. Penyelesaian dan Penyerahan
   Setelah seluruh pengujian dan perbaikan selesai, proyek akan disiapkan untuk diserahkan, dengan dokumentasi yang lengkap mengenai cara kerja game, struktur database, serta penjelasan teknis tentang setiap bagian dari sistem yang telah dikembangkan.

Metodologi yang digunakan dalam proyek ini memungkinkan pengembangan yang fleksibel, memastikan bahwa setiap tahap dapat dievaluasi dan disesuaikan untuk mencapai hasil yang optimal sesuai dengan tujuan yang telah ditetapkan.

II. TINJAUAN PUSTAKA

A. Teknologi Yang Digunakan

Dalam pengembangan game Pokemon Airstrike, berbagai teknologi digunakan untuk membangun backend, frontend, dan sistem komunikasi real-time. Setiap teknologi dipilih untuk mendukung kebutuhan proyek dan memastikan game dapat berjalan dengan lancar dan memberikan pengalaman pengguna yang baik. Berikut adalah teknologi-teknologi utama yang digunakan dalam pengembangan proyek ini:

1. Flask (Backend)

Flask adalah framework Python yang ringan dan fleksibel, digunakan untuk membangun backend aplikasi. Flask dipilih karena kesederhanaannya dalam membangun aplikasi web yang cepat dan efisien, serta kemudahan dalam mengintegrasikan berbagai komponen seperti database, API, dan WebSocket. Flask juga memungkinkan pengembangan dengan pendekatan yang modular, memudahkan pengelolaan berbagai fitur dalam aplikasi.

2. SQLAlchemy (Database)

SQLAlchemy adalah toolkit dan Object-Relational Mapping (ORM) untuk Python yang digunakan untuk mengelola database dalam aplikasi. SQLAlchemy memungkinkan pengelolaan database dengan cara yang lebih terstruktur dan mudah dipahami, serta menyediakan fitur untuk melakukan query dan manipulasi data dalam database dengan lebih efisien. Dalam proyek ini, SQLAlchemy digunakan untuk menangani penyimpanan dan pengambilan data pengguna, skor, serta leaderboard.

3. MySQL (Database Management System)

MySQL adalah sistem manajemen database relasional yang digunakan untuk menyimpan data dalam proyek ini. MySQL dipilih karena kemampuannya untuk menangani sejumlah besar data dengan performa yang baik, serta kompatibilitasnya dengan SQLAlchemy. MySQL digunakan untuk menyimpan data pemain, skor, dan informasi lainnya yang terkait dengan gameplay, termasuk leaderboard.

4. HTML, CSS, dan JavaScript (Frontend)

Frontend aplikasi dibangun menggunakan teknologi web standar, yaitu HTML, CSS, dan JavaScript.

a. HTML digunakan untuk struktur dan elemen-elemen dasar halaman game.
b. CSS digunakan untuk desain dan tampilan antarmuka pengguna, memastikan game memiliki tampilan yang responsif dan menarik di layar desktop.
c. JavaScript digunakan untuk menangani interaktivitas dalam game, termasuk logika permainan seperti pergerakan karakter, interaksi dengan musuh, dan penggunaan power-up.
d.

5. WebSocket dan SocketIO (Komunikasi Real-Time)

WebSocket adalah protokol komunikasi yang memungkinkan pertukaran data secara real-time antara server dan klien. SocketIO, yang dibangun di atas WebSocket, digunakan untuk menyediakan komunikasi dua arah secara real-time antara frontend dan backend game. Teknologi ini memungkinkan pemain untuk berinteraksi dengan game secara langsung, baik dalam pengiriman skor atau pengelolaan status permainan.

6. Git dan GitHub (Version Control)

Untuk pengelolaan kode sumber dan kolaborasi dalam tim, Git digunakan sebagai sistem version control. Kode proyek disimpan di GitHub untuk memudahkan kolaborasi dan melacak

perubahan yang dilakukan selama pengembangan. GitHub juga digunakan untuk mempublikasikan dokumentasi proyek dan berbagi dengan pihak lain yang terlibat.

7. Visual Studio Code (IDE)

Visual Studio Code (VS Code) digunakan sebagai Integrated Development Environment (IDE) utama untuk menulis kode. VS Code dipilih karena kemampuannya dalam menangani berbagai bahasa pemrograman, kemudahan dalam penggunaan, dan dukungan untuk berbagai plugin yang mendukung pengembangan web, seperti ekstensi untuk Python, HTML, CSS, dan JavaScript.

B. Teknologi Yang Digunakan

1. Konsep Game Airstrike

Game Airstrike adalah genre permainan yang berfokus pada aksi pertempuran udara di mana pemain mengendalikan pesawat atau kendaraan terbang untuk melawan musuh yang datang dari berbagai arah. Biasanya, dalam game Airstrike, pemain dihadapkan pada tantangan untuk bertahan hidup, menghindari tembakan musuh, serta menghancurkan target atau musuh yang bergerak. Elemen utama dalam game ini mencakup:

a. Pesawat atau Kendaraan Terbang, pemain mengendalikan pesawat yang dapat bergerak dalam dua dimensi, biasanya dari kiri ke kanan atau sebaliknya, dengan tujuan menghindari serangan musuh sambil menyerang balik.
b. Musuh dan Obstacle, pemain dihadapkan dengan berbagai jenis musuh yang menyerang dengan tembakan atau dengan menghindari halangan yang ada di layar.
c. Power-up, pemain dapat mengumpulkan power-up yang meningkatkan kemampuan mereka, seperti peluru tambahan, kecepatan, atau perlindungan dari serangan musuh.
d. Skor dan Leaderboard, skor dihitung berdasarkan jumlah musuh yang dihancurkan atau waktu bertahan hidup, dan biasanya diperlombakan melalui leaderboard.

2. Konsep Game Pokemon

Pokemon adalah waralaba multimedia yang menggabungkan elemen pertarungan antara berbagai makhluk imajiner yang dikenal sebagai "Pokemon". Setiap Pokemon memiliki kemampuan unik yang dapat digunakan dalam pertempuran. Konsep utama dalam game Pokemon adalah menangkap, melatih, dan bertarung dengan Pokemon, serta mengumpulkan berbagai karakter dengan kekuatan dan elemen yang berbeda. Konsep ini meliputi:

a. Pokemon sebagai Karakter, pemain mengendalikan Pokemon yang memiliki kekuatan dan kemampuan spesial, seperti serangan berbasis elemen (api, air, listrik, dll.).
b. Pertarungan dan Strategi, pemain berkompetisi dalam pertempuran, baik melawan musuh NPC maupun pemain lain. Setiap Pokemon memiliki atribut dan kemampuan yang berbeda, memberikan kesempatan untuk memilih strategi yang sesuai dengan lawan yang dihadapi.
c. Pengembangan dan Pembaruan, pokemon dapat berkembang melalui pertempuran, meningkatkan level dan kemampuannya.
d. Koleksi, pemain bertujuan untuk menangkap berbagai jenis Pokemon yang berbeda, yang masing-masing memiliki kekuatan dan kelemahan tertentu.

3. Gabungan Konsep Game Airstrike dan Pokemon

Dalam Pokemon Airstrike, konsep dari kedua genre digabungkan untuk menciptakan sebuah permainan aksi pertempuran udara dengan elemen-elemen dari dunia Pokemon. Beberapa fitur kunci dari penggabungan konsep ini antara lain:

a. Karakter Pokemon, pemain mengendalikan karakter berbentuk Pokemon yang dapat terbang di udara, bertarung dengan musuh yang datang dari berbagai arah.
b. Power-up dan Kemampuan Khusus, power-up yang tersedia dalam game ini terinspirasi dari kemampuan spesial Pokemon, seperti serangan api, listrik, atau efek bertahan hidup lainnya.
c. Aksi Pertempuran Udara, seperti dalam game Airstrike klasik, pemain harus menghindari serangan musuh dan menghancurkan musuh dengan tembakan, tetapi dengan kekuatan dan karakteristik unik yang terinspirasi dari Pokemon.

d. Sistem Skor dan Leaderboard, pemain bersaing untuk mencetak skor tertinggi dalam pertempuran udara, yang didasarkan pada jumlah musuh yang dihancurkan atau waktu bertahan hidup dalam pertempuran.

Melalui gabungan kedua konsep ini, Pokemon Airstrike memberikan pengalaman yang menarik bagi pemain yang suka dengan tantangan aksi pertempuran udara, sambil menikmati elemen-elemen taktis dan kemampuan unik yang berasal dari dunia Pokemon.

III. DESKRIPSI SISTEM

A. Deskripsi Umum

Pokemon Airstrike adalah permainan aksi berbasis dua dimensi yang menggabungkan elemen pertempuran udara dengan karakter-karakter dari dunia Pokemon. Pemain mengendalikan karakter yang berbentuk Pokemon yang dapat terbang dan bertarung melawan musuh dalam bentuk awan yang menghalangi perjalanan mereka. Game ini menantang pemain untuk bertahan hidup selama mungkin sambil menghindari rintangan dan meningkatkan skor.

1. Tujuan Permainan
   Tujuan utama dari permainan ini adalah untuk bertahan hidup dan mengalahkan musuh yang muncul, serta meningkatkan skor sebanyak mungkin. Pemain akan mendapatkan poin berdasarkan waktu bertahan hidup dan seberapa banyak rintangan yang berhasil dihindari. Skor yang tinggi akan disimpan dalam leaderboard, dan pemain dapat berkompetisi untuk mencatatkan nama mereka di peringkat teratas.

2. Kontrol Pemain
   Pemain mengendalikan karakter Pokemon yang dapat bergerak secara horizontal di sepanjang layar untuk menghindari awan yang muncul sebagai obstacle. Pemain dapat menggunakan tombol atau kontrol layar sentuh untuk bergerak kiri atau kanan, serta mengaktifkan power-up untuk meningkatkan kecepatan. Kontrol permainan ini sederhana dan mudah diakses oleh berbagai kalangan pemain.

3. Musuh dan Obstacle
   Dalam game ini, musuh hanya berupa awan, yang berfungsi sebagai rintangan yang harus dihindari oleh pemain. Awan ini dapat muncul dengan pola acak dan bergerak secara vertikal dari atas layar, memaksa pemain untuk bergerak cepat agar tidak terhantam. Awan ini tidak menyerang, namun keberadaannya menghalangi jalur terbang pemain.

4. Power-up
   Sepanjang permainan, pemain dapat mengumpulkan power-up yang meningkatkan kecepatan karakter. Power-up ini memberikan pemain kemampuan untuk bergerak lebih cepat, membantu mereka menghindari rintangan dan bertahan lebih lama dalam permainan. Pemain harus memanfaatkan power-up ini dengan bijak untuk memperpanjang durasi permainan mereka.

5. Tingkat Kesulitan
   Permainan ini memiliki tingkat kesulitan yang meningkat seiring berjalannya waktu. Awan sebagai rintangan akan muncul lebih sering dan bergerak lebih cepat, memaksa pemain untuk terus meningkatkan keterampilan mereka dalam menghindar dan bertahan hidup. Meskipun hanya ada satu jenis musuh dan obstacle, meningkatnya kecepatan rintangan memberikan tantangan yang lebih besar bagi pemain.

6. Tampilan dan Visual
   Visual dalam Pokemon Airstrike dirancang dengan gaya yang sederhana namun menarik. Latar belakang permainan berganti-ganti untuk memberikan kesan dinamis, sementara karakter Pokemon yang dimainkan memiliki animasi pergerakan yang halus. Awan sebagai obstacle terlihat jelas dan mudah dikenali, sementara efek visual dari pergerakan dan power-up dirancang untuk memberikan pengalaman bermain yang menyenangkan.

7. Leaderboard
   Setiap permainan yang dimainkan akan mencatat skor akhir pemain yang berhasil bertahan hidup. Skor tersebut akan dimasukkan ke dalam leaderboard, yang menyimpan sepuluh pemain teratas. Pemain dapat melihat ranking mereka dan berusaha untuk mengalahkan skor tertinggi yang ada, menciptakan elemen kompetitif dalam game ini.

B. Fitur Utama

1. Gameplay
   Gameplay dalam Pokemon Airstrike berfokus pada aksi pertempuran udara yang sederhana namun menantang. Pemain mengendalikan karakter Pokemon yang bergerak secara horizontal untuk menghindari awan yang muncul sebagai obstacle. Tujuan utama pemain adalah untuk bertahan hidup selama mungkin sambil menghindari rintangan, meningkatkan kecepatan, dan mendapatkan skor sebanyak mungkin. Game ini mengedepankan respons cepat, ketepatan dalam menghindar, dan penggunaan power-up yang bijaksana untuk bertahan lebih lama.

2. Pengaturan Karakter dan Musuh
   a. Karakter, pemain mengendalikan karakter Pokemon yang dapat bergerak secara horizontal di layar. Karakter ini dapat bergerak kiri dan kanan untuk menghindari musuh dan rintangan. Setiap karakter Pokemon memiliki kontrol yang sederhana dan intuitif, membuatnya mudah dimainkan oleh pemain dari berbagai kalangan.
   b. Musuh, musuh dalam game ini berupa awan yang muncul sebagai obstacle yang harus dihindari dan Pokemon ‘Ghast” sebagai musuh yang akan menembakkan tembakan yang dapat mengurangi lawan pemain Awan ini bergerak dari atas layar dan memaksa pemain untuk bergerak cepat agar tidak terhantam. Tidak ada musuh yang menyerang, namun adanya awan sebagai penghalang membuat pemain harus terus bergerak untuk bertahan.

3. Power-up dan Items
   Dalam permainan ini, terdapat power-up yang memberikan keuntungan bagi pemain:
   a. Kecepatan, power-up yang paling sering muncul adalah power-up yang meningkatkan kecepatan karakter. Dengan power-up ini, pemain bisa bergerak lebih cepat, memungkinkan mereka untuk menghindari rintangan (awan) yang muncul lebih cepat.

4. Level dan Skor
   a. Level meskipun game ini tidak memiliki banyak level tradisional seperti game lainnya, kesulitan permainan akan meningkat seiring berjalannya waktu. Awan akan muncul lebih sering dan bergerak lebih cepat, memberikan tantangan yang lebih besar kepada pemain. Pemain harus menyesuaikan strategi mereka untuk bertahan lebih lama.
   b. Skor, skor dihitung berdasarkan durasi bertahan hidup pemain dan jumlah rintangan (awan) yang berhasil dihindari. Pemain akan mendapatkan skor tambahan jika mereka berhasil bertahan lebih lama atau mengumpulkan power-up kecepatan. Skor ini akan terus bertambah selama pemain tidak terkena musuh atau rintangan.

5. Halaman Leaderboard
   Halaman Leaderboard akan menampilkan sepuluh pemain teratas berdasarkan skor tertinggi yang tercatat dalam permainan. Pemain dapat melihat ranking mereka setelah

permainan berakhir, memberikan elemen kompetitif dalam game ini. Halaman ini juga memungkinkan pemain untuk berusaha mengalahkan skor tertinggi yang ada, memotivasi mereka untuk terus mencoba dan meningkatkan performa mereka dalam permainan.

C. Alur Game

Alur permainan dalam Pokemon Airstrike sangat sederhana dan langsung, dengan fokus pada aksi pertempuran udara yang menantang. Berikut adalah langkah-langkah alur permainan secara rinci:

1. Tampilan Utama (Homepage)
   Pemain memulai permainan dari halaman utama yang menampilkan tombol "Start" untuk memulai permainan. Pada halaman ini, pemain akan diminta untuk memasukkan username mereka. Username ini akan digunakan untuk mencatat skor akhir mereka dalam leaderboard. Setelah memasukkan nama, pemain dapat menekan tombol "Play" untuk memulai permainan.

2. Memulai Permainan
   Setelah tombol "Play" ditekan, pemain akan diarahkan ke halaman gameplay, dimana karakter Pokemon yang mereka pilih akan muncul di bagian bawah layar. Pemain dapat mulai menggerakkan karakter Pokemon tersebut secara horizontal menggunakan kontrol yang telah disediakan. Tujuan utama adalah untuk menghindari awan yang muncul di layar.

3. Menghindari Awan
   Setiap awan yang muncul di layar bergerak dari atas ke bawah dan harus dihindari oleh pemain. Pemain dapat bergerak kiri atau kanan untuk menghindari tabrakan dengan awan. Kecepatan awan ini meningkat seiring berjalannya waktu, menambah tantangan bagi pemain. Pemain juga dapat mengumpulkan power-up yang muncul di sepanjang permainan, yang memberikan keuntungan berupa peningkatan kecepatan.

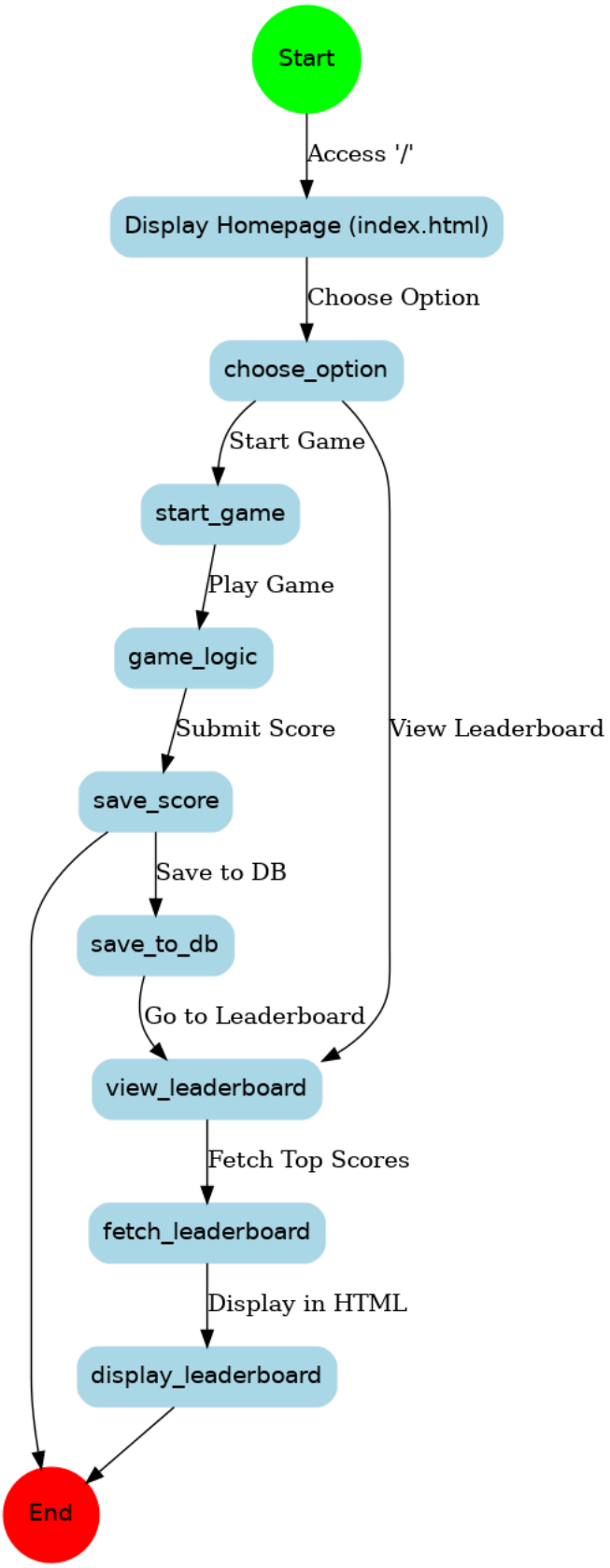
4. Mengumpulkan Power-up
   Power-up yang paling sering muncul adalah kecepatan. Power-up ini memungkinkan karakter untuk bergerak lebih cepat, memberikan keunggulan bagi pemain dalam menghindari awan yang datang lebih cepat. Pemain harus berusaha untuk mengumpulkan power-up ini sambil terus menghindari rintangan.

5. Game Over
   Permainan berakhir ketika karakter pemain terkena awan atau keluar dari area permainan yang telah ditentukan. Setelah game over, sebuah layar akan muncul menampilkan skor akhir pemain dan opsi untuk kembali ke halaman utama atau melihat leaderboard.

6. Leaderboard
   Jika pemain memilih untuk melihat leaderboard, mereka akan diarahkan ke halaman yang menampilkan sepuluh pemain teratas berdasarkan skor tertinggi yang tercatat dalam permainan. Pemain dapat melihat peringkat mereka dan mencoba untuk mencapai skor tertinggi pada permainan berikutnya.

7. Kembali ke Halaman Utama
   Setelah melihat leaderboard atau skor mereka, pemain dapat kembali ke halaman utama untuk memainkan permainan lagi, mengubah username, atau menutup game.

Berdasarkan alur game yang telah disebutkan berikut merupakan gambaran diagram alur mengenai game ini.



IV. Arsitektur dan Teknologi

A. Desain Arsitektur Sistem

Desain arsitektur sistem Pokemon Airstrike mengacu pada pembagian tugas dan komponen dalam aplikasi yang saling berinteraksi untuk menyajikan pengalaman permainan yang mulus. Arsitektur ini terdiri dari beberapa komponen utama yang mencakup frontend, backend, dan database. Berikut adalah gambaran umum tentang arsitektur sistem:

1. Frontend

   Frontend adalah bagian yang berinteraksi langsung dengan pengguna (pemain). Pada Pokemon Airstrike, frontend dibangun menggunakan HTML, CSS, dan JavaScript untuk membuat antarmuka pengguna yang responsif dan interaktif.

   a. HTML digunakan untuk struktur halaman permainan, seperti halaman utama, gameplay, dan leaderboard.
   b. CSS bertanggung jawab untuk desain tampilan dan penataan elemen-elemen di halaman, memastikan tampilan yang bersih dan mudah digunakan oleh pemain.
   c. JavaScript digunakan untuk logika interaktif, seperti kontrol pergerakan karakter Pokemon, animasi, pengumpulan power-up, dan menghitung skor pemain.

2. Backend

   Backend bertanggung jawab untuk menangani logika permainan, pengelolaan data, serta komunikasi antara server dan frontend. Flask, sebuah framework Python, digunakan untuk membangun backend permainan. Komponen backend ini mencakup:

   a. Routing: Backend menggunakan route untuk menangani permintaan dari pemain, seperti memulai permainan, menyimpan skor, dan mengambil leaderboard.
   b. Gameplay Logic: Backend juga menangani logika permainan seperti meningkatkan kecepatan musuh, menghitung skor pemain, dan mengatur waktu permainan.
   c. API untuk Leaderboard: Backend menyediakan API untuk menyimpan dan mengambil data leaderboard, memungkinkan pemain untuk melihat peringkat mereka berdasarkan skor tertinggi.

3. Database

   Database berfungsi untuk menyimpan data yang dibutuhkan dalam game, seperti data pengguna dan skor. MySQL digunakan sebagai sistem manajemen basis data dalam game ini. Database memiliki beberapa tabel utama, seperti:

   a. Tabel Pengguna: Menyimpan data pengguna, termasuk username yang dimasukkan pada halaman utama.
   b. Tabel Skor: Menyimpan skor yang tercatat selama permainan, yang nantinya akan diambil untuk ditampilkan di leaderboard.

4. Komunikasi Antara Komponen
   a. Frontend ke Backend. Ketika permainan dimulai atau skor tercatat, frontend akan mengirimkan data ke backend melalui HTTP request. Data seperti username pemain dan skor dikirim menggunakan form submission atau URL

parameters. Backend kemudian akan memproses data tersebut dan menyimpannya di database atau mengirimkan respons kembali ke frontend.

b. Backend ke Database. Backend mengelola komunikasi dengan database menggunakan SQLAlchemy (karena kamu menggunakan MySQL untuk penyimpanan leaderboard). Setiap kali ada skor baru yang ingin disimpan atau data leaderboard yang perlu diambil, backend akan melakukan query ke database untuk mendapatkan atau memperbarui informasi tersebut.

c. Frontend Menampilkan Data. Setelah backend memproses permintaan dan data berhasil disimpan atau diperbarui, respons dikirimkan kembali ke frontend. Di halaman leaderboard, misalnya, skor yang terbaru akan ditampilkan menggunakan template HTML yang diproses dengan Flask templating.

5. Server

   Seluruh aplikasi berjalan di server yang menjalankan Flask. Server ini akan menangani permintaan dari pemain yang ingin memulai permainan, mengirimkan data skor, dan mengakses leaderboard.

6. Flow Diagram Sistem
   a. Halaman Utama. Pemain memasukkan username dan menekan tombol "Play". Permainan dimulai, dan data pengguna dikirim ke backend.
   b. Gameplay. Pemain bergerak, menghindari awan, dan mengumpulkan power-up. Skor dihitung di frontend dan dikirim ke backend saat permainan selesai.
   c. Leaderboard. Backend menyimpan dan mengambil data skor dari database. Skor ditampilkan di leaderboard di frontend.

7. Keamanan dan Validasi

   Pada sisi backend, terdapat mekanisme validasi untuk memastikan bahwa data yang dikirim dari frontend sesuai dengan format yang benar, seperti username yang tidak kosong atau skor yang valid. Keamanan lebih lanjut dapat ditambahkan dengan pengamanan akses API yang lebih ketat jika diperlukan.

Arsitektur ini memastikan bahwa sistem permainan dapat berfungsi dengan lancar, dengan masing-masing komponen memiliki tugas yang jelas dan dapat saling berinteraksi dengan baik untuk menciptakan pengalaman bermain yang optimal.

B. Teknologi yang Digunakan

1. Backend (Flask)

   Backend dalam proyek ini dibangun menggunakan Flask, sebuah framework web Python yang ringan dan fleksibel. Flask mempermudah pembuatan aplikasi dengan menyediakan berbagai fitur seperti routing dan pengelolaan template. Routing: Flask memungkinkan pembuatan rute untuk menangani permintaan HTTP dari frontend, seperti ketika pengguna memulai permainan atau mengirimkan data skor. Setiap rute mengarah pada fungsi Python yang memproses permintaan dan mengirimkan respons. Template Rendering: Flask menggunakan Jinja2 untuk rendering template HTML, yang memungkinkan pemisahan antara logika aplikasi dan tampilan. Flask dipilih karena kemudahan penggunaannya dan kemampuan untuk memberikan kontrol penuh atas aplikasi, cocok untuk proyek game berbasis web ini.

2. Frontend (HTML, CSS, JavaScript)

Untuk antarmuka pengguna (UI), proyek ini menggunakan HTML, CSS, dan JavaScript. HTML digunakan untuk membangun struktur halaman-halaman web dalam game, termasuk halaman utama, gameplay, dan leaderboard. CSS digunakan untuk memperindah tampilan halaman, seperti mengatur tata letak, warna, dan animasi. JavaScript menangani logika permainan, seperti pergerakan karakter, penghitungan skor, dan pengaturan elemen-elemen interaktif di halaman. JavaScript juga digunakan untuk mengirimkan data dari frontend ke backend setelah permainan selesai, termasuk nama pengguna dan skor akhir yang dicapai.

3. Database (MySQL)

   Database menggunakan MySQL, sistem manajemen basis data relasional yang menyimpan data penting game, seperti username dan skor. Tabel Pengguna menyimpan informasi nama pemain yang dimasukkan pada awal permainan, sementara Tabel Skor menyimpan skor yang dicapai pemain. SQLAlchemy, sebuah ORM, digunakan untuk mempermudah interaksi antara aplikasi Flask dan database MySQL. MySQL dipilih karena kemampuannya menangani data dalam jumlah besar dan skalabilitasnya yang baik, memungkinkan game untuk menyimpan dan mengelola data pemain dengan efisien.

4. WebSocket dan API

   Meskipun aplikasi ini tidak menggunakan WebSocket untuk komunikasi real-time dalam gameplay, API berbasis HTTP digunakan untuk interaksi antara frontend dan backend. API ini memungkinkan frontend untuk mengirimkan data ke backend, seperti saat permainan dimulai (mengirimkan username) atau saat permainan selesai (menyimpan dan mengambil data skor). Jika diperlukan, WebSocket dapat digunakan di masa depan untuk menciptakan komunikasi dua arah secara langsung antara frontend dan backend, terutama untuk fitur yang membutuhkan pembaruan real-time yang cepat.

V. IMPLEMENTASI

A. Penjelasan Kode

1. Struktur Direktori

   Struktur direktori dalam proyek ini disusun untuk memudahkan pengelolaan berbagai komponen aplikasi. Berikut adalah gambaran umum struktur direktori:

```
/pokemon_airstrike
   / __pycache__
   /instance
     -game_db.sqlite
   /queries
     -game_db.sql
   /static
      /assets
        /images
          -background.gif
          -enemy_projectile.png
          -enemy.png
          -home-bg.gif
          -Leaderboard-Logo,png
          -obstacle.gif
          -play.png
          -player_projectile.gif
```

-player.gif
-powerup_speed.png
-title.png

/css <- File CSS untuk tampilan frontend.
-game-styles.css
-leaderboard-styles.css
-styles.css
/js <- File JavaScript yang mengatur logika permainan.
-game.js
-leaderboard.js
-script.js
/sounds
-click.wav
-fire.wav
-gameplay.mp3
-intro.mp3
-notice.wav
/templates
- home.html <- Halaman utama untuk memulai permainan.
- game.html <- Halaman utama permainan.
- leaderboard.html <- Halaman untuk menampilkan leaderboard.
- login.html
/app
- app.py <- File utama yang menangani routing dan logika backend.
- database.py <- File untuk pengelolaan interaksi dengan database.
/requirements.txt <- Daftar dependensi yang diperlukan untuk menjalankan aplikasi.

2. Deskripsi Kode Utama

Kode utama aplikasi terdapat dalam file app.py, yang menangani pengaturan rute dan logika utama aplikasi. Berikut adalah penjelasan tentang bagian-bagian utama kode dalam aplikasi:

a. Rute /: Menangani halaman utama aplikasi, yang memungkinkan pengguna untuk memasukkan username dan memulai permainan. Halaman ini menggunakan render_template untuk menampilkan halaman HTML utama (index.html).

b. Rute /start_game: Rute ini menangani permintaan untuk memulai permainan setelah pengguna mengirimkan username. Username diterima dalam format JSON, dan jika valid, permainan akan dimulai. Pada bagian ini, data JSON diambil dan username akan disimpan untuk sesi permainan.

c. Rute /game: Menangani halaman permainan itu sendiri, di mana permainan berlangsung. Rute ini menerima permintaan POST yang berisi data seperti username dan skor pemain. Skor pemain disimpan ke dalam database menggunakan fungsi save_score_to_database. Selain itu, rute ini juga menangani permintaan GET untuk menampilkan halaman game jika pengguna belum memulai permainan.

d. Rute /leaderboard: Rute ini digunakan untuk menampilkan leaderboard dengan 10 pemain teratas berdasarkan skor tertinggi. Rute ini mengambil data dari database dan mengirimkan data leaderboard ke template HTML (leaderboard.html) menggunakan render_template. Fungsi get_leaderboard digunakan untuk mengambil data dan mengurutkannya berdasarkan skor tertinggi.

e. Rute /save_score: Digunakan untuk menyimpan skor pemain setelah permainan selesai. Data username dan skor dikirimkan dalam format JSON, dan aplikasi akan menyimpan skor tersebut ke database.

f. Rute /gameover: Setelah permainan berakhir, skor pemain disimpan ke database. Jika proses penyimpanan berhasil, aplikasi akan merespons dengan informasi bahwa skor telah disimpan dan memberi tahu frontend untuk mengarahkan pengguna ke halaman leaderboard.

Backend berkomunikasi dengan frontend melalui JSON untuk mentransfer data dinamis (seperti skor), dan render_template digunakan untuk merender tampilan HTML yang menyertakan data yang dibutuhkan. Data seperti leaderboard diambil dengan meng-query database menggunakan SQLAlchemy dan disertakan di dalam template HTML yang sesuai.

3. Penanganan Database dan Keyboard

Aplikasi menggunakan MySQL sebagai database untuk menyimpan informasi pengguna dan skor. Berikut adalah penjelasan singkat mengenai bagaimana aplikasi menangani database:

a. Model SQLAlchemy
Menggunakan SQLAlchemy untuk mengelola interaksi antara Python dan MySQL. Model User menyimpan informasi pengguna seperti nama, sementara model Score menyimpan skor akhir.
b. Fungsi Penyimpanan Skor
Ketika permainan berakhir, skor pemain disimpan dalam tabel Score. Fungsi ini memeriksa apakah sudah ada lebih dari 10 pemain di leaderboard. Jika sudah ada, fungsi akan mengurutkan skor dan memastikan hanya 10 pemain dengan skor tertinggi yang ditampilkan.
c. Fungsi Pengambilan Leaderboard
Mengambil data leaderboard dari database dan menampilkannya pada halaman leaderboard. Data diurutkan berdasarkan skor tertinggi, menampilkan nama pengguna dan skor mereka.

Proses penyimpanan dan pengambilan data dari database ini menggunakan SQLAlchemy ORM, yang menyederhanakan proses query dan manipulasi data dalam MySQL.

B. Pengujian Sistem

Pengujian sistem dilakukan untuk memastikan bahwa aplikasi berjalan sesuai dengan yang diharapkan dan semua fitur berfungsi dengan baik. Pengujian dibagi menjadi dua kategori utama: pengujian fungsionalitas dan pengujian kinerja.

1. Pengujian Fungsionalitas

Pengujian fungsionalitas dilakukan untuk memastikan bahwa setiap fitur dalam aplikasi berfungsi sebagaimana mestinya. Berikut adalah beberapa pengujian yang dilakukan:

a. Halaman Utama (/): Pengujian dilakukan untuk memastikan bahwa halaman utama dapat diakses dengan benar dan menampilkan form untuk memasukkan username. Setelah memasukkan username, pengguna diarahkan dengan benar ke halaman permainan.

b. Memulai Permainan (/start_game): Pengujian dilakukan dengan mengirimkan data JSON yang berisi username dan memastikan aplikasi menerima data dengan benar dan menampilkan pesan yang sesuai. Pengguna dapat memulai permainan setelah data valid diterima.

c. Permainan (/game): Pengujian dilakukan untuk memastikan bahwa halaman permainan dapat memproses dan menyimpan skor pemain dengan benar melalui permintaan POST. Pengujian dilakukan dengan mengirimkan data skor dan username dan memverifikasi bahwa data tersebut tersimpan dengan benar di database.

d. Leaderboard (/leaderboard): Pengujian dilakukan untuk memverifikasi bahwa leaderboard mengambil data dengan benar dari database dan menampilkan 10 pemain teratas dengan skor yang sesuai. Pengujian memastikan leaderboard terupdate sesuai dengan data yang ada di database.

e. Simpan Skor (/save_score): Pengujian dilakukan untuk memastikan skor pemain disimpan dengan benar setelah permainan selesai, dan pengguna menerima respons yang sesuai jika data berhasil disimpan.

f. Game Over (/gameover): Pengujian dilakukan untuk memverifikasi bahwa ketika permainan selesai, skor disimpan dengan benar dan pengguna diarahkan ke halaman leaderboard setelahnya.

2. Pengujian Kinerja

Pengujian kinerja dilakukan untuk memastikan bahwa aplikasi dapat menangani berbagai macam skenario pengguna tanpa penurunan performa yang signifikan. Beberapa aspek yang diuji meliputi:

a. Waktu Respons Server: Pengujian dilakukan untuk memverifikasi bahwa server merespons permintaan pengguna dengan cepat, terutama untuk permintaan seperti pengambilan leaderboard dan penyimpanan skor. Waktu respons yang lebih lambat dari 2 detik akan dianggap sebagai masalah.

b. Kapasitas Database: Pengujian dilakukan untuk memverifikasi bahwa database dapat menangani penyimpanan data skor pemain dalam jumlah besar, tanpa terjadinya lag atau kegagalan penyimpanan data. Dalam hal ini, sistem diuji dengan memasukkan lebih dari 10 entri dalam leaderboard dan memverifikasi bahwa leaderboard tetap terurut dengan benar.

c. Komunikasi WebSocket: Pengujian dilakukan untuk memverifikasi bahwa komunikasi WebSocket dapat berjalan dengan lancar selama permainan, memastikan bahwa data antara client dan server tetap sinkron tanpa masalah latensi yang signifikan.

d. Stabilitas Aplikasi: Pengujian dilakukan dengan menjalankan aplikasi dalam waktu lama dan dengan banyak pengguna untuk memeriksa apakah aplikasi tetap stabil dan tidak mengalami crash atau kelebihan beban.

Dengan struktur ini, Anda dapat menjelaskan dengan jelas pengujian yang telah dilakukan pada aplikasi Anda dan memberikan gambaran yang jelas mengenai bagaimana sistem diuji untuk memastikan fungsionalitas dan kinerja yang optimal.

VI. Hasil dan Pembahasan

A. Penjelasan Kode

B. Analisis Hasil Pengujian

C. Tantangan yang Dihadapi

D. Perbaikan yang Dilakukan

VII. Kesimpulan

A. Ringkasan Hasil

Proyek game Pokemon Airstrike telah berhasil dikembangkan dengan menggunakan teknologi Flask, SQLAlchemy, dan SocketIO. Game ini memungkinkan pemain untuk memulai permainan dengan memasukkan username dan mencatat skor mereka dalam leaderboard. Fitur utama yang ada dalam game mencakup kontrol karakter, pengaturan skor, leaderboard, dan pengaturan tampilan game. Skor pemain disimpan di dalam database MySQL (atau SQLite, sesuai pengaturan), dan leaderboard selalu menampilkan 10 pemain teratas.

Secara keseluruhan, game ini berjalan dengan baik di platform web dan dapat menangani pengambilan dan penyimpanan data secara efisien. Interaksi real-time dengan menggunakan SocketIO berfungsi dengan lancar, dan sistem penyimpanan skor bekerja sesuai yang diharapkan.

B. Saran untuk Pengembangan Lebih Lanjut

Meskipun game Pokemon Airstrike telah berhasil diselesaikan, ada beberapa area yang dapat diperbaiki atau ditingkatkan untuk pengembangan lebih lanjut. Beberapa saran yang dapat dipertimbangkan adalah:

a. Peningkatan Visual

Visual game saat ini masih cukup sederhana. Mengembangkan dan memperhalus elemen visual, seperti karakter dan latar belakang, dapat meningkatkan daya tarik visual game.
Menggunakan visual assets yang lebih menarik dan berkualitas tinggi akan memberikan pengalaman bermain yang lebih memuaskan.

b. Fitur Pause

Penambahan fitur pause akan memungkinkan pemain untuk menghentikan permainan sementara tanpa kehilangan progres. Fitur ini sangat berguna jika pemain ingin istirahat atau melanjutkan permainan di waktu yang berbeda.

c. Validasi Username

Saat ini, tidak ada aturan yang mencegah pemain memasukkan username yang sama. Untuk meningkatkan pengalaman pengguna, disarankan untuk menambahkan validasi yang memastikan bahwa setiap username yang dimasukkan unik dan tidak ada duplikasi dalam leaderboard.

d. Penambahan Suara (Sound)

Meskipun suara tidak diterapkan pada versi ini, penambahan efek suara akan meningkatkan atmosfer permainan, seperti suara saat game over atau saat membuka leaderboard. Efek suara dapat memberikan umpan balik yang lebih nyata kepada pemain dan membuat pengalaman lebih imersif.

e. Scroll pada Leaderboard

Saat ini, leaderboard menampilkan hanya 10 pemain teratas. Untuk pemain yang ingin melihat lebih banyak, menambahkan fitur scroll di leaderboard akan memungkinkan mereka untuk menggulir dan melihat lebih banyak data pemain lainnya.

f. Peningkatan Kinerja dan Skalabilitas

Jika jumlah pemain meningkat, pengujian lebih lanjut terhadap kinerja dan skalabilitas aplikasi sangat penting. Penambahan fitur seperti caching untuk leaderboard atau pemisahan database ke beberapa tabel dapat memperbaiki performa.

Dengan perbaikan-perbaikan ini, game Pokemon Airstrike dapat menjadi lebih menarik, fungsional, dan dapat memberikan pengalaman bermain yang lebih menyenangkan bagi pengguna. Pengembangan lebih lanjut akan memperkuat kualitas game ini dan meningkatkan kepuasan pemain.

VIII. Daftar Pustaka

[1] WebSockets Standard, "WebSockets specification," websockets.spec.whatwg.org, Nov. 30, 2024. [Online]. Available: https://websockets.spec.whatwg.org.

[2] L. Kine, "Intro to APIs: History of APIs," 2019.

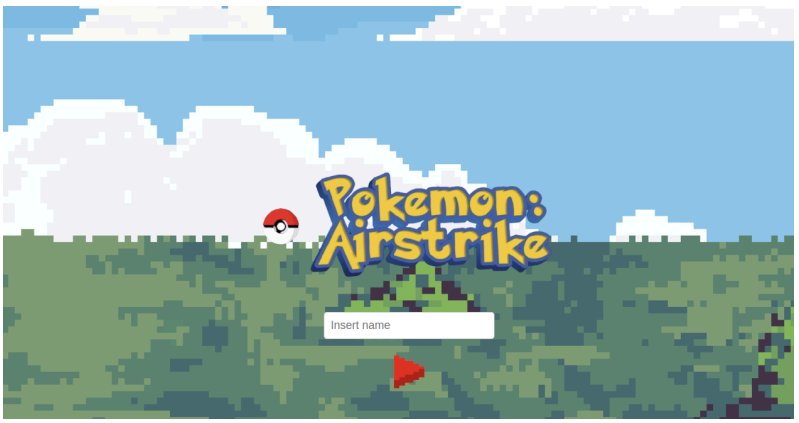
[3] B. Jin, S. Sahni, and A. Shevat, Designing Web APIs. O'Reilly Media, 2018.

IX. Lampiran

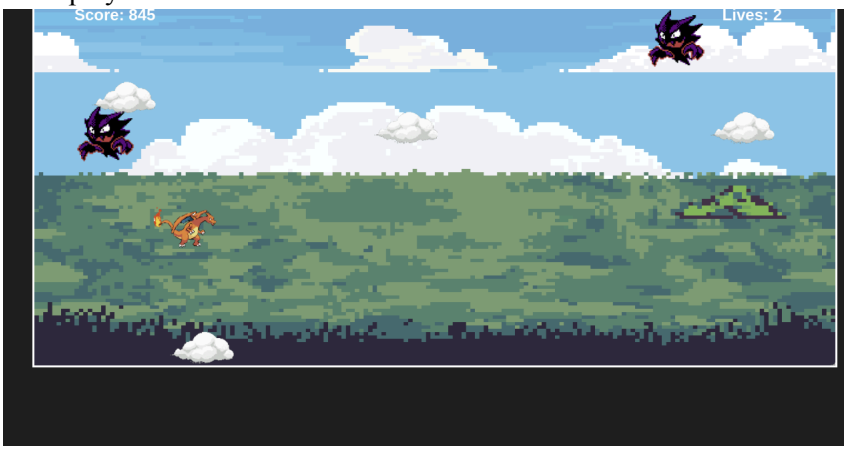
A. Gambar

1. Homepage

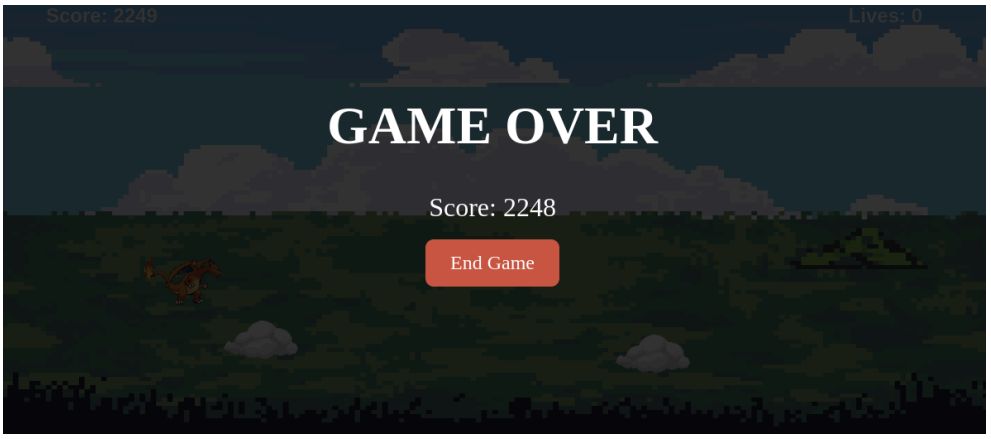
2. Insert Username



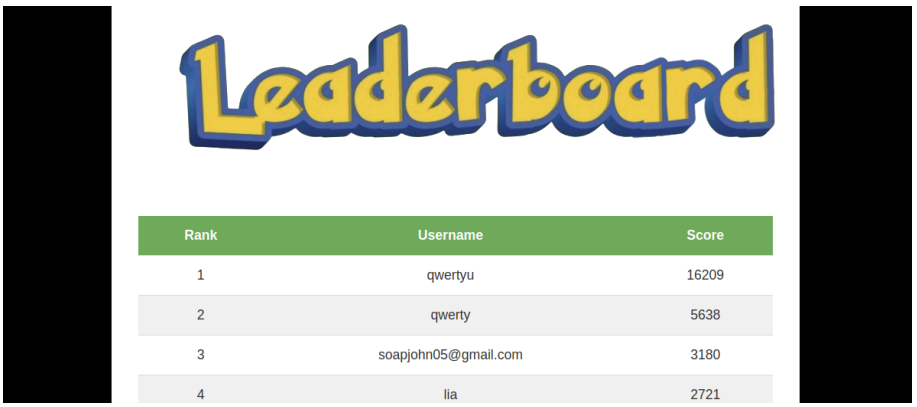
3. Gameplay



4. Gameover



5. Leaderboard



B. Kode

1. app.py

```
from flask import Flask, render_template, request, jsonify, url_for
from flask_socketio import SocketIO, emit
from flask_sqlalchemy import SQLAlchemy

# Inisialisasi Flask dan SocketIO
app = Flask(__name__, static_folder = 'static')
socketio = SocketIO(app, cors_allowed_origins="*")  # Mengizinkan semua origin

# Konfigurasi database URL untuk MySQL
app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///game_db.sqlite'
app.config['SQLALCHEMY_TRACK_MODIFICATIONS'] = False  # Menonaktifkan fitur pengawasan
perubahan objek

# Inisialisasi SQLAlchemy
db = SQLAlchemy(app)

# Definisikan model untuk leaderboard
class Leaderboard(db.Model):
    __tablename__ = 'leaderboard'

    id = db.Column(db.Integer, primary_key=True)
    username = db.Column(db.String(80), unique=True, nullable=False)
    score = db.Column(db.Integer, nullable=False)

    def __init__(self, username, score):
        self.username = username
        self.score = score

    def __repr__(self):
        return f'<Leaderboard {self.username}, {self.score}>'
```

```python
# Fungsi untuk menyimpan skor ke database
def save_score_to_database(username, score):
    """
    Menyimpan skor dan username ke dalam leaderboard.
    """
    try:
        # Log data yang diterima
        print(f"Saving score: username={username}, score={score}")  # Log input

        # Cek apakah username sudah ada di database
        existing_entry = Leaderboard.query.filter_by(username=username).first()

        if existing_entry:
            # Jika sudah ada, perbarui skor
            existing_entry.score = score
            print(f"Updated score for {username}: {score}")  # Log pembaruan
        else:
            # Jika tidak ada, buat entri baru
            new_entry = Leaderboard(username=username, score=score)
            db.session.add(new_entry)
            print(f"New entry added: {username} with score {score}")  # Log entri baru

        # Commit perubahan ke database
        db.session.commit()
        print(f"Score for {username} saved successfully: {score}")  # Log sukses

    except Exception as e:
        db.session.rollback()  # Rollback jika terjadi error
        print(f"Error saving score for {username}: {e}")  # Log error

# Fungsi untuk menyimpan leaderboard ke database
def save_to_leaderboard(leaderboard_data):
    """
    Menyimpan leaderboard data (hanya 10 pemain teratas) ke dalam database.
    """
    # Urutkan leaderboard berdasarkan score (dari yang terbesar)
    leaderboard_data = sorted(leaderboard_data, key=lambda x: x['score'], reverse=True)

    # Ambil 10 pemain teratas
    leaderboard_data = leaderboard_data[:10]

    # Hapus semua data leaderboard lama
    Leaderboard.query.delete()

    # Sisipkan data leaderboard yang baru
    for entry in leaderboard_data:
        leaderboard_entry = Leaderboard(username=entry['username'], score=entry['score'])
        db.session.add(leaderboard_entry)

    # Commit transaksi ke database
    try:
        db.session.commit()
        print("Leaderboard successfully updated.")  # Debugging message
    except Exception as e:
        db.session.rollback()  # Rollback jika terjadi error
        print(f"Error updating leaderboard: {e}")


# Fungsi untuk mengambil leaderboard dari database
def get_leaderboard():
```

```
    """
    Mengambil leaderboard (hanya 10 pemain teratas) dari database.
    """
    try:
        leaderboard = Leaderboard.query.order_by(Leaderboard.score.desc()).limit(10).all()
        print(f"Leaderboard fetched successfully, {len(leaderboard)} entries retrieved.")

        # Ubah hasil query menjadi list of dicts
        leaderboard_data = [{'username': entry.username, 'score': entry.score} for entry in
leaderboard]
        return leaderboard_data
    except Exception as e:
        print(f"Error fetching leaderboard: {e}")
        return []


# Route untuk halaman utama
@app.route('/')
def index():
    """
    Menampilkan halaman utama dengan tombol Start dan leaderboard.
    """
    return render_template('index.html')


# Route untuk memulai game
@app.route('/start_game', methods=['POST'])
def start_game():
    """
    Memproses data username yang dikirimkan dari halaman utama dan memulai game.
    """
    try:
        data = request.get_json()
    except Exception as e:
        return jsonify({'success': False, 'message': f"Invalid JSON format: {str(e)}"}), 400

    username = data.get('username')

    if not username:
        return jsonify({'success': False, 'message': 'Username is required!'}), 400

    # Simpan username ke sesi atau log untuk tujuan tertentu
    print(f"Game started for username: {username}")

    return jsonify({'success': True, 'message': f"Welcome, {username}!"})


# Route untuk halaman game
@app.route('/game', methods=['POST', 'GET'])
def game():
    if request.method == 'POST':
        try:
            data = request.get_json()  # Mencoba mengambil data JSON
            username = data.get('username')
            score = data.get('score')

            # Log untuk debugging
            print(f"Received data: username={username}, score={score}")  # Cek data yang
diterima
```

```
            # Simpan skor ke database
            if username and score is not None:
                save_score_to_database(username, score)  # Simpan skor ke database

                return jsonify({'success': True, 'message': 'Score saved successfully!'})

            return jsonify({'success': False, 'message': 'Invalid data'})

        except Exception as e:
            print(f"Error processing data: {e}")
            return jsonify({'success': False, 'message': 'Error processing data'}), 500

    return render_template('game.html')


# Route untuk menampilkan leaderboard
@app.route('/leaderboard', methods=['GET'])
def get_leaderboard_route():
    """
    Mengambil leaderboard dari database dan menampilkannya.
    """
    leaderboard_data = get_leaderboard()  # Ambil data leaderboard dari database

    # Menggunakan render_template untuk mengirimkan data leaderboard ke template HTML
    return render_template('leaderboard.html', leaderboard=leaderboard_data)


# Route untuk menyimpan leaderboard setelah game selesai
@app.route('/save_score', methods=['POST'])
def save_score():
    """
    Menyimpan skor dan username ke dalam leaderboard setelah game selesai.
    """
    data = request.get_json()
    username = data.get('username')
    score = data.get('score')

    # Debug log untuk cek apakah data valid
    print(f"Received save_score data: username={username}, score={score}")

    if username and score:
        # Simpan skor ke database
        save_score_to_database(username, score)

        return jsonify({'success': True, 'message': 'Score saved successfully!'})

    return jsonify({'success': False, 'message': 'Username or score is missing'})


# Route untuk menyimpan leaderboard secara keseluruhan
@app.route('/gameover', methods=['POST'])
def gameover():
    """
    Menyimpan skor dan username ke dalam leaderboard setelah game selesai.
    """
    try:
        # Ambil data username dan score dari JSON body
        data = request.get_json()
        username = data.get('username')
        score = data.get('score')
```

```
        if not username or score is None:
            return jsonify({'success': False, 'message': 'Username or score missing'}), 400

        # Simpan skor ke dalam database
        save_score_to_database(username, score)

        # Setelah data berhasil disimpan, redirect ke leaderboard
        return jsonify({'success': True, 'message': 'Score saved successfully', 'redirect':
'/leaderboard'})

    except Exception as e:
        print(f"Error in /gameover route: {e}")
        return jsonify({'success': False, 'message': 'Error saving score'}), 500


# Jalankan aplikasi
if __name__ == '__main__':
    socketio.run(app, debug=True)
```

2. *index.html*

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Pokemon Airstrike</title>
    <link rel="stylesheet" href="{{ url_for('static', filename='css/styles.css')
}}">
    <script src="{{ url_for('static', filename='js/script.js') }}"
defer></script>
</head>
<body>
    <!-- Background Image -->
    <div class="background-image"></div>

    <div id="home-container">
        <!-- Title GIF -->
        <div class="title-container">
            <img src="{{ url_for('static', filename='assets/images/title.png')
}}" alt="Pokemon Airstrike" class="game-title">
        </div>
    <!-- Modal Notification -->
    <div id="modal" class="modal">
        <div class="modal-content">
            <span class="close-btn">&times;</span>
            <p>Please insert your username!</p>
        </div>
    </div>

        <!-- Game Button -->
```

```
        <div id="game-button-container">
            <img id="game-button" src="{{ url_for('static',
filename='assets/images/play.png') }}" alt="Game Button" onclick="startGame()">
        </div>

        <!-- Play Form -->
        <div id="play-container" style="display: none;">
            <input type="text" id="username" placeholder="Insert name">
            <img id="play-button" src="{{ url_for('static',
filename='assets/images/play.png') }}" alt="Play Button"
onclick="submitUsername()">
        </div>
    </div>

    <!-- Suara Intro (looping otomatis) -->
    <audio id="intro-sound" src="{{ url_for('static',
filename='sounds/intro.mp3') }}" loop ></audio>
    <script>
        document.addEventListener('DOMContentLoaded', () => {
            const introSound = document.getElementById('intro-sound');

            // Putar suara intro setelah tombol game diklik
            const gameButton = document.getElementById('game-button');
            gameButton.addEventListener('click', () => {
                introSound.play().catch((error) => {
                    console.log('Error playing intro sound:', error);
                });
            });
        });
    </script>
    <!-- Suara Klik -->
    <audio id="click-sound" src="{{ url_for('static',
filename='sounds/click.wav') }}"preload="auto"></audio>

    <!-- Suara Notifikasi (untuk username kosong) -->
    <audio id="notice-sound" src="{{ url_for('static',
filename='sounds/notice.wav') }}"preload="auto"></audio>

    <script>
        function startGame() {
            // Main game button klik, putar suara klik
            document.getElementById('click-sound').play();

            document.getElementById('game-button-container').style.display =
'none';
            document.getElementById('play-container').style.display = 'block';
        }

        function submitUsername() {
            const username = document.getElementById('username').value;
            if (username) {
```

```
                // Putar suara klik saat tombol play diklik
                document.getElementById('click-sound').play();

                // Redirect ke gameplay dengan username
                window.location.href = `/game?username=${username}`;
            } else {
                // Putar suara notifikasi jika username kosong
                document.getElementById('notice-sound').play();
                // Menampilkan modal jika username kosong
                document.getElementById('username-modal').style.display =
'block';
            }
        }
        function closeModal() {
            document.getElementById('username-modal').style.display = 'none';
        }
    </script>
</body>
</html>
```

3. *styles.css*

```
/* Global Styles */
body {
    margin: 0;
    padding: 0;
    height: 100vh;
    display: flex;
    flex-direction: column; /* Tata elemen secara vertikal */
    align-items: center;
    justify-content: center;
    background: url('/static/assets/images/home-bg.gif') no-repeat center center
fixed;
    background-size: cover; /* Sesuaikan ukuran background */
    font-family: Arial, sans-serif;
    color: white;
    text-align: center;
}

/* Modal Styles */
.modal {
    display: none; /* Hidden by default */
    position: fixed; /* Fixed position to stay in place */
    top: 50%; /* Center the modal */
    left: 50%;
    transform: translate(-50%, -50%); /* Adjust position by half of its width
and height */
    background-color: rgba(0, 0, 0, 0.8); /* Dark background with transparency
*/
    color: white;
```

```
    padding: 20px;
    border-radius: 10px;
    z-index: 1000; /* Make sure modal is on top of other content */
    text-align: center;
    width: 300px; /* Tentukan lebar modal */
    min-height: 150px; /* Tentukan ketinggian minimal */
}

/* Modal Content */
.modal-content {
    background-color: #333;
    border-radius: 10px;
    padding: 30px;
    font-family: 'Courier New', Courier, monospace; /* Retro font */
    font-size: 18px;
    text-transform: uppercase;
}

/* Modal Message */
.modal-message {
    font-size: 18px;
    color: black;
    margin-bottom: 15px;
    font-family: 'Press Start 2P', cursive;
}

/* Modal Button */
.modal-btn {
    padding: 10px 20px;
    font-size: 16px;
    cursor: pointer;
    background-color: #28a745;
    color: white;
    border: none;
    border-radius: 5px;
    font-family: 'Press Start 2P', cursive; /* Retro font */
    transition: background-color 0.3s ease;
}

.modal-btn:hover {
    background-color: #218838;
}

/* Close Button */
.close-btn {
    font-size: 24px;
    font-weight: bold;
    color: #f44336; /* Red close button */
    cursor: pointer;
    position: absolute;
    top: 10px;
```

```
    right: 10px;
}

.close-btn:hover {
    color: white;
    background-color: #f44336;
    border-radius: 50%;
}

/* Title Container */
.title-container {
    display: flex; /* Gunakan Flexbox untuk memusatkan */
    flex-direction: column; /* Atur tata letak elemen secara vertikal */
    justify-content: center; /* Pusatkan secara vertikal */
    align-items: center; /* Pusatkan secara horizontal */
    margin-bottom: 20px; /* Tambahkan jarak dari elemen berikutnya */
    margin-top: 50px; /* Sesuaikan jarak dari atas halaman */
}

/* Title PNG */
.game-title {
    width: 70%; /* Perbesar ukuran sesuai kebutuhan */
    max-width: 800px; /* Tetapkan batas maksimum lebar */
    height: auto; /* Sesuaikan tinggi secara proporsional */
}

/* Game Button */
#game-button-container {
    margin: 20px 0;
}

#game-button {
    width: 50px;
    cursor: pointer;
    transition: transform 0.2s ease; /* Tambahkan efek hover */
}

#game-button:hover {
    transform: scale(1.1); /* Perbesar saat hover */
}

/* Fade Effect */
#game-button-container, #play-container, #home-container {
    transition: opacity 0.5s ease-in-out;
}

/* Center Play Form */
#play-container {
    flex-direction: column; /* Vertikal */
    align-items: center; /* Centered horizontally */
    justify-content: center; /* Centered vertically */
```

```
    margin-top: 20px;
}


#username {
    font-size: 18px;
    padding: 10px;
    margin-bottom: 15px; /* Tambahkan jarak dari tombol */
    border-radius: 5px;
    border: 1px solid #ccc;
    width: 250px;
}

#play-button {
    width: 50px; /* Perkecil ukuran tombol */
    height: auto; /* Sesuaikan tinggi proporsional */
    cursor: pointer;
    transition: transform 0.2s ease; /* Tambahkan efek hover */
    margin-top: 10px; /* Tambahkan jarak antara tombol dan form */
}

#play-button:hover {
    transform: scale(1.1); /* Perbesar sedikit saat hover */
}
```

4. *script.js*

```
document.addEventListener('DOMContentLoaded', () => {
    const gameButton = document.getElementById('game-button');
    const playContainer = document.getElementById('play-container');
    const gameButtonContainer =
document.getElementById('game-button-container');
    const playButton = document.getElementById('play-button');
    const usernameInput = document.getElementById('username');
    const homeContainer = document.getElementById('home-container');
    const introSound = document.getElementById('intro-sound');

    // Play intro sound
    introSound.play().catch((error) => {
        console.log('Error playing intro sound:', error);
    });

    // Modal setup
    const modal = document.getElementById('modal');
    const closeBtn = document.querySelector('.close-btn');

    // Game button click event
    gameButton.addEventListener('click', () => {
        console.log('Game button clicked');
        const clickSound = document.getElementById('click-sound');
```

```
        clickSound.play().then(() => {
            console.log('Click sound played');
        }).catch((error) => {
            console.log('Error playing click sound:', error);
        });

        // Hide game button and show the play form with fade effect
        gameButtonContainer.style.opacity = '0';
        setTimeout(() => {
            gameButtonContainer.style.display = 'none';
            playContainer.style.display = 'flex';
            playContainer.style.opacity = '0';
            setTimeout(() => {
                playContainer.style.opacity = '1';
            }, 100);
        }, 300);
    });

    playButton.addEventListener('click', () => {
        const username = usernameInput.value.trim();
        console.log('Play button clicked');

        if (!username) {
            console.log('Username empty, showing modal');
            // Show modal notification
            modal.style.display = 'block';

            // Play notice sound
            const noticeSound = document.getElementById('notice-sound');
            noticeSound.play().then(() => {
                console.log('Notice sound played');
            }).catch((error) => {
                console.log('Error playing notice sound:', error);
            });
            return;
        }

        // Add fade-out effect before navigating to gameplay
        homeContainer.style.opacity = '0';
        setTimeout(() => {
            window.location.href =
`/game?username=${encodeURIComponent(username)}`;
        }, 500);
    });

    // Close modal on click
    closeBtn.addEventListener('click', () => {
        modal.style.display = 'none';
    });

    // Close modal if clicked outside of it
```

```
    window.addEventListener('click', (event) => {
        if (event.target === modal) {
            modal.style.display = 'none';
        }
    });
});
```

5. *game.html*

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Pokemon Airstrike - Gameplay</title>
    <link rel="stylesheet" href="{{ url_for('static',
filename='css/game-styles.css') }}">
</head>
<body>
    <!-- Game Container -->
    <div id="game-container">
        <!-- Background -->
        <div id="game-background"></div>

        <!-- Gameplay Canvas -->
        <canvas id="gameCanvas" width="1200" height="800"></canvas>

        <!-- UI Elements -->
        <div id="ui">
            <input type="hidden" id="username" value="{{
request.args.get('username') }}">
            <span id="score">Score: 0</span>
            <span id="lives">Lives: 3</span>
        </div>
        <div id="game-over-screen" style="display: none; position: absolute;
top: 0; left: 0; width: 100%; height: 100%; background: rgba(0, 0, 0, 0.8);
text-align: center; z-index: 1000;">
            <h1 style="color: white; font-size: 4rem; margin-top: 10%;">GAME
OVER</h1>
            <p id="final-score" style="color: white; font-size: 2rem; margin:
20px 0;">Score: 0</p>
            <button id="end-game-btn" style="padding: 15px 30px; font-size:
1.5rem; background: #e74c3c; color: white; border: none; border-radius: 10px;
cursor: pointer;">
                End Game
            </button>
        </div>

        <!-- End Game Button -->
```

```
        <button id="end-game" onclick="saveScore()">End Game</button>
    </div>

    <!-- Game Script -->
    <script src="{{ url_for('static', filename='js/game.js') }}"></script>
</body>
</html>
```

6. *game-styles.css*

```
/* Global Styles */
body {
    margin: 0;
    padding: 0;
    font-family: Arial, sans-serif;
    background: #222; /* Fallback background for gameplay */
    color: white;
    overflow: hidden;
}

/* Tampilan kontainer game */
.container {
    display: flex;
    justify-content: center;
    align-items: center;
    width: 100%;
    height: 100vh;
}

/* Game Container */
#game-container {
    position: relative;
    width: 1200px; /* Disesuaikan dengan ukuran canvas */
    height: 550px; /* Disesuaikan dengan ukuran canvas */
    margin: 50px auto;
    border: 3px solid white;
    overflow: hidden; /* Prevent elements from overflowing */
}

/* Background */
#game-background {
    position: absolute;
    top: 0;
    left: 0;
    width: 100%;
    height: 100%;
    background: url('/static/assets/images/background.gif') no-repeat center
center;
    background-size: auto;
    z-index: -1; /* Ensure background is at the back */
```

```
    overflow: visible;
}

/* Player */
#player {
    position: absolute;
    width: 100px; /* Lebih besar */
    height: 100px; /* Lebih besar */
    background: url('/static/assets/images/player.gif') no-repeat center center;
    background-size: auto;
    z-index: 2; /* Player above background */
}

/* Player Projectile */
.player-projectile {
    position: absolute;
    width: 20px; /* Lebih besar */
    height: 20px; /* Lebih besar */
    background: url('/static/assets/images/player_projectile.gif') no-repeat
center center;
    background-size: contain;
    z-index: 3; /* Above player */
}

/* Enemy */
.enemy {
    position: absolute;
    width: 75px; /* Lebih besar */
    height: 75px; /* Lebih besar */
    background: url('/static/assets/images/enemy.gif') no-repeat center center;
    background-size: auto;
    z-index: 2; /* Same layer as player */
}

/* Enemy Projectile */
.enemy-projectile {
    position: absolute;
    width: 25px; /* Lebih besar */
    height: 25px; /* Lebih besar */
    background: url('/static/assets/images/enemy_projectile.png') no-repeat
center center;
    background-size: contain;
    z-index: 3; /* Above enemy */
}

/* Obstacle */
.obstacle {
    position: absolute;
    width: 75px; /* Lebih besar */
    height: 75px; /* Lebih besar */
    background: url('/static/assets/images/obstacle.gif') no-repeat center
```

```
center;
    background-size: auto;
    z-index: 2; /* Same layer as player */
}

/* Power-Up */
.power-up {
    position: absolute;
    width: 50px; /* Lebih besar */
    height: 50px; /* Lebih besar */
    background: url('/static/assets/images/powerup_speed.png') no-repeat center
center;
    background-size: contain;
    z-index: 2; /* Same layer as player */
    animation: float 2s infinite ease-in-out;
}

/* Animation for Floating Power-Up */
@keyframes float {
    0%, 100% {
        transform: translateY(-10px);
    }
    50% {
        transform: translateY(10px);
    }
}

/* UI Elements */
#ui {
    position: absolute;
    top: 10px;
    left: 50%;
    transform: translateX(-50%);
    z-index: 10; /* Above all game elements */
    font-size: 24px; /* Lebih besar */
    font-weight: bold;
    display: flex;
    justify-content: space-between;
    width: 90%;
    color: white; /* Warna teks putih */
}

#ui span {
    margin-right: 20px;
}



#game-over-screen h1 {
    font-family: 'Press Start 2P', cursive;
    font-size: 4rem; /* Sesuaikan ukuran font */
```

```
    color: white;
}

#game-over-screen #final-score {
    font-family: 'Press Start 2P', cursive;
    font-size: 2rem;
    color: white;
}

#game-over-screen #end-game-btn {
    font-family: 'Press Start 2P', cursive;
    font-size: 1.5rem;
    background: #e74c3c; /* Warna tombol */
    color: white;
    padding: 15px 30px;
    border: none;
    border-radius: 10px;
    cursor: pointer;
    transition: transform 0.2s ease; /* Tambahkan animasi */
}

#game-over-screen #end-game-btn:hover {
    transform: scale(1.1); /* Perbesar tombol saat hover */
}
```

7. *game.js*

```
// Select canvas and context
const canvas = document.getElementById('gameCanvas');
const ctx = canvas.getContext('2d');
canvas.width = 1200; // Enlarged canvas width
canvas.height = 550; // Enlarged canvas height

// Load assets
const backgroundImg = new Image();
backgroundImg.src = '/static/assets/images/background.gif';
const playerImg = new Image();
playerImg.src = '/static/assets/images/player.gif';
const playerProjectileImg = new Image();
playerProjectileImg.src = '/static/assets/images/player_projectile.gif';
const enemyImg = new Image();
enemyImg.src = '/static/assets/images/enemy.gif';
const enemyProjectileImg = new Image();
enemyProjectileImg.src = '/static/assets/images/enemy_projectile.png';
const obstacleImg = new Image();
obstacleImg.src = '/static/assets/images/obstacle.gif';
const powerupSpeedImg = new Image();
powerupSpeedImg.src = '/static/assets/images/powerup_speed.png';
const gameplaySound = new Audio('/static/assets/sounds/gameplay.mp3');
const shootingSound = new Audio('/static/sounds/fire.wav');
```

```
// Game state variables
let score = 0;
let lives = 3;
let isGameOver = false;
let timeElapsed = 0;
let soundPlayed = false;  // Variabel untuk menandakan apakah suara sudah
diputar

let backgroundMusic = new Audio('/static/sounds/gameplay.mp3');
backgroundMusic.loop = true
// Fungsi untuk memutar suara gameplay
function playGameplaySound() {
    const gameplaySound = new Audio('/static/sounds/gameplay.mp3');
    gameplaySound.loop = true;  // Suara diputar berulang
    gameplaySound.play().catch(error => console.error("Error memutar suara:",
error)); // Tangani error
}

// Fungsi untuk memeriksa dan memutar suara ketika skor lebih dari 100
function checkAndPlaySound() {
    if (score > 100 && !soundPlayed) {
        playGameplaySound();  // Panggil fungsi untuk memutar suara
        soundPlayed = true;   // Tandai bahwa suara sudah diputar
    }
}

// Fungsi untuk memutar suara tembakan
function playShootingSound() {
    shootingSound.play();
}

// Player
const player = {
    x: canvas.width / 2 - 50,
    y: canvas.height - 150,
    width: 100, // Larger player
    height: 100,
    speed: 7,
};

// Arrays for game objects
const bullets = [];
const enemies = [];
const enemyProjectiles = [];
const obstacles = [];
const powerups = [];

// Key events
const keys = {};
document.addEventListener('keydown', (e) => {
    keys[e.key] = true;
```

```
});
document.addEventListener('keyup', (e) => {
    keys[e.key] = false;
    if (e.key === ' ') {
        shootBullet();
    }
});

// Function to draw background
function drawBackground() {
    ctx.drawImage(backgroundImg, 0, 0, canvas.width, canvas.height);
}

function drawPlayer() {
    ctx.drawImage(playerImg, player.x, player.y, player.width, player.height);

    // Hitbox untuk player (80% dari ukuran)
    const playerHitbox = {
        x: player.x + player.width * 0.1, // Margin 10% kiri
        y: player.y + player.height * 0.1, // Margin 10% atas
        width: player.width * 0.8, // Lebar 80%
        height: player.height * 0.8, // Tinggi 80%
    };

    // Debugging: Visualisasi hitbox player
    //ctx.strokeStyle = 'blue';
    //ctx.strokeRect(playerHitbox.x, playerHitbox.y, playerHitbox.width,
playerHitbox.height);
}

// Function to draw bullets
function drawBullets() {
    bullets.forEach((bullet, bulletIndex) => {
        bullet.x += 5; // Speed up projectile
        ctx.drawImage(playerProjectileImg, bullet.x, bullet.y, 40, 40); //
Larger projectile

        // Check collision with enemies
        enemies.forEach((enemy, enemyIndex) => {
            if (
                bullet.x < enemy.x + enemy.width &&
                bullet.x + 40 > enemy.x &&
                bullet.y < enemy.y + enemy.height &&
                bullet.y + 40 > enemy.y
            ) {
                // Remove the enemy and the bullet
                bullets.splice(bulletIndex, 1);
                enemies.splice(enemyIndex, 1);

                // Hitbox untuk player projectile
    const bulletHitbox = {
```

```
        x: bullet.x + 5, // Sedikit margin untuk hitbox
        y: bullet.y + 5,
        width: 10,
        height: 10,
    };

    // Debugging: Visualisasi hitbox player projectile
    ctx.strokeStyle = 'green';
    ctx.strokeRect(bulletHitbox.x, bulletHitbox.y, bulletHitbox.width,
bulletHitbox.height);
                // Increase the score
                score += 1000;
            }
        });

        // Remove bullet if out of screen
        if (bullet.x > canvas.width) bullets.splice(bulletIndex, 1);
    });
}

// Function to shoot bullet
function shootBullet() {
    bullets.push({ x: player.x + player.width, y: player.y + player.height / 2 -
10 });
    playShootingSound()
}


// Function to draw enemies
function drawEnemies() {
    enemies.forEach((enemy, index) => {
        enemy.x -= 3; // Slower movement
        enemy.y += enemy.movement;
        if (enemy.y <= 0 || enemy.y >= canvas.height - enemy.height) {
            enemy.movement *= -1; // Reverse direction if hitting boundary
        }

        ctx.drawImage(enemyImg, enemy.x, enemy.y, enemy.width, enemy.height);

        // Enemy shoots projectiles
        if (Math.random() < 0.01) {
            enemyProjectiles.push({
                x: enemy.x,
                y: enemy.y + enemy.height / 2,
                width: 10,
                height: 10,
            });
        }
    });
}
```

```
// Function to draw enemy projectiles
function drawEnemyProjectiles() {
    enemyProjectiles.forEach((proj, index) => {
        proj.x -= 5;
        ctx.drawImage(enemyProjectileImg, proj.x, proj.y, 40, 40); // Enlarged
enemy projectile

        // Hitbox untuk enemy projectile
        const projHitbox = {
            x: proj.x + 10,  // Adjusted hitbox to match the larger projectile
size
            y: proj.y + 10,
            width: 20,  // Adjusted width for the larger projectile
            height: 20, // Adjusted height for the larger projectile
        };

        // Debugging: Visualisasi hitbox enemy projectile
        ctx.strokeStyle = 'yellow';
        ctx.strokeRect(projHitbox.x, projHitbox.y, projHitbox.width,
projHitbox.height);

        // Collision dengan player
        if (
            projHitbox.x < player.x + player.width &&
            projHitbox.x + projHitbox.width > player.x &&
            projHitbox.y < player.y + player.height &&
            projHitbox.y + projHitbox.height > player.y
        ) {
            lives--;
            enemyProjectiles.splice(index, 1);
        }

        // Hapus projectile jika keluar dari layar
        if (proj.x < 0) enemyProjectiles.splice(index, 1);
    });
}
function drawObstacles() {
    obstacles.forEach((obs, index) => {
        obs.x -= 3;
        ctx.drawImage(obstacleImg, obs.x, obs.y, obs.width, obs.height);

        // Hitbox untuk obstacle
        const hitbox = {
            x: obs.x + obs.width * 0.1,
            y: obs.y + obs.height * 0.1,
            width: obs.width * 0.8,
            height: obs.height * 0.8,
        };

        // Debugging: Visualisasi hitbox obstacle
        //ctx.strokeStyle = 'red';
```

```
        //ctx.strokeRect(hitbox.x, hitbox.y, hitbox.width, hitbox.height);

        // Collision dengan player
        if (
            hitbox.x < player.x + player.width &&
            hitbox.x + hitbox.width > player.x &&
            hitbox.y < player.y + player.height &&
            hitbox.y + hitbox.height > player.y
        ) {
            lives--;
            obstacles.splice(index, 1); // Hapus obstacle setelah tabrakan
        }
    });
}

// Function to draw power-ups
function drawPowerups() {
    powerups.forEach((powerup, index) => {
        powerup.x -= 3;
        ctx.drawImage(powerupSpeedImg, powerup.x, powerup.y, 30, 30); //
Slightly larger powerup

        // Collision with player
        if (
            powerup.x < player.x + player.width &&
            powerup.x + 30 > player.x &&
            powerup.y < player.y + player.height &&
            powerup.y + 30 > player.y
        ) {
            powerups.splice(index, 1);
            activateSpeedBoost();
        }
    });
}

// Activate speed boost for player
function activateSpeedBoost() {
    player.speedBoost = true;
    player.speed = 14; // Double speed
    setTimeout(() => {
        player.speedBoost = false;
        player.speed = 7; // Reset speed after 5 seconds
    }, 5000);
}

// Function to update score over time (like the Chrome Dinosaur game)
function updateScoreOverTime() {
    timeElapsed++;
    score += 1;  // Meningkatkan skor setiap detik
    document.getElementById('score').textContent = `Score: ${score}`;
    checkAndPlaySound()
```

```
}

function getUsernameFromInput() {
    return document.getElementById('username').value;  // Ambil username dari
input hidden
}


function getScoreFromGame() {
    // Ambil skor dari variabel skor game
    return score;
}

/// Update logika updateGame untuk memanggil gameOver ketika lives habis
function updateGame() {
    // Player movement
    if (keys['ArrowLeft'] && player.x > 0) player.x -= player.speed;
    if (keys['ArrowRight'] && player.x < canvas.width - player.width) player.x
+= player.speed;
    if (keys['ArrowUp'] && player.y > 0) player.y -= player.speed;
    if (keys['ArrowDown'] && player.y < canvas.height - player.height) player.y
+= player.speed;

    // Spawn enemies randomly
    if (Math.random() < 0.001) {
        enemies.push({
            x: canvas.width,
            y: Math.random() * (canvas.height - 100),
            width: 100,
            height: 100,
            movement: Math.random() < 0.5 ? -1 : 1,
        });
    }

    // Spawn obstacles randomly
    if (Math.random() < 0.005) {
        obstacles.push({
            x: canvas.width,
            y: Math.random() * (canvas.height - 50),
            width: 100,
            height: 100,
        });
    }

    // Spawn power-ups randomly
    if (Math.random() < 0.002) {
        powerups.push({
            x: canvas.width,
            y: Math.random() * (canvas.height - 30),
        });
    }
```

```
    // End game jika lives habis
    if (lives <= 0 && !isGameOver) {
        gameOver();
        backgroundMusic.pause();
    }

    // Update score over time
    updateScoreOverTime();
}

// Fungsi untuk menampilkan Game Over Screen
function showGameOverScreen() {
    // Tampilkan elemen Game Over
    const gameOverScreen = document.getElementById("game-over-screen");
    const finalScore = document.getElementById("final-score");

    // Update skor akhir di layar game over
    finalScore.textContent = `Score: ${score}`;

    // Tampilkan layar Game Over
    gameOverScreen.style.display = "block";

    // Nonaktifkan game loop
    isGameOver = true;
}

// Modifikasi fungsi gameOver untuk menggunakan Game Over Screen
function gameOver() {
    isGameOver =  true
    backgroundMusic.pause()
    backgroundMusic.currentTime = 0
    // Tampilkan layar Game Over
    showGameOverScreen();

    // Kirimkan skor ke server
    saveScore();
}

function saveScore() {
    const username = getUsernameFromInput();  // Ambil username dari input atau
session
    const score = getScoreFromGame();  // Ambil skor dari game

    fetch('/gameover', {
        method: 'POST',
        headers: {
            'Content-Type': 'application/json'
        },
        body: JSON.stringify({ username: username, score: score })
    })
```

```
    .then(response => response.json())
    .then(data => {
        if (!data.success) {
            // Jika ada error dalam menyimpan skor, tampilkan pesan kesalahan
            alert('Error saving score to the database');
        }
    })
    .catch(error => {
        console.error('Error during fetch:', error);
        alert('Network error while saving score');
    });
}



// Menambahkan event listener pada tombol "End Game"
document.getElementById("end-game-btn").addEventListener("click", function() {
    // Pastikan game sudah berakhir sebelum menyimpan skor
    if (isGameOver) {
        saveScore();
        window.location.href = "/leaderboard";  // Arahkan ke halaman
leaderboard setelah game selesai
    }
});



/// Game loop tetap dijalankan kecuali game sudah berakhir
function gameLoop() {
    if (isGameOver) {
        return; // Jangan lanjutkan game loop jika game over
    }
    // Cek dan mainkan musik jika perlu
    if (!backgroundMusic.paused && !isGameOver) {
        checkAndPlaySound(); // Jika game belum selesai, teruskan cek suara
    }

    // Clear canvas
    ctx.clearRect(0, 0, canvas.width, canvas.height);

    // Draw and update game objects
    drawBackground();
    drawPlayer();
    drawBullets();
    drawEnemies();
    drawEnemyProjectiles();
    drawObstacles();
    drawPowerups();
    updateGame();

    // Display score and lives
```

```
    document.getElementById('score').textContent = `Score: ${score}`;
    document.getElementById('lives').textContent = `Lives: ${lives}`;

    // Loop the game
    requestAnimationFrame(gameLoop);
}

// Start game loop
gameLoop();
```

8. *leaderboard.html*

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Leaderboard</title>
    <link rel="stylesheet" href="{{ url_for('static',
filename='css/leaderboard-styles.css') }}">
</head>
<body>
    <!-- Background Image -->
    <div class="background-image"></div>
    <div id="leaderboard-container">
        <!-- Title GIF -->
        <div class="leaderboard-title-container">
            <img src="{{ url_for('static',
filename='assets/images/Leaderboard-Logo.png') }}" alt="Leaderboard"
class="Leaderboard">
        </div>

    <table>
        <thead>
            <tr>
                <th>Rank</th>
                <th>Username</th>
                <th>Score</th>
            </tr>
        </thead>
        <tbody>
            {% if leaderboard %}
                {% for player in leaderboard %}
                <tr>
                    <td>{{ loop.index }}</td>
                    <td>{{ player.username }}</td>
                    <td>{{ player.score }}</td>
                </tr>
                {% endfor %}
            {% else %}
```

```
                <tr><td colspan="3">No leaderboard data available.</td></tr>
            {% endif %}
        </tbody>
    </table>
    <a href="/" class="button">Back to Homepage</a>
</body>
</html>
```

9. *leaderboard-styles.css*

```
/* Global Styles */
* {
    margin: 0;
    padding: 0;
    box-sizing: border-box;
    font-family: 'Arial', sans-serif;
}

/* Background */
body {
    background-color: #000000;
    color: #333;
    overflow: hidden;
}

/* Leaderboard Container */
#leaderboard-container {
    position: relative; /* Menambahkan posisi relatif */
    max-width: 900px;
    margin: 0 auto;
    padding: 35px;
    background-color: #fff;
    box-shadow: 0 5px 15px rgba(0, 0, 0, 0.1);
    border-radius: 8px;
    text-align: center;
    z-index : 0;
    overflow: hidden;
}

/* Leaderboard Background */
#leaderboard-background {
    position: fixed; /* Gunakan fixed agar background tetap di belakang saat
scroll */
    top: 0;
    left: 0;
    width: 100vw; /* Gunakan viewport width */
    height: 100vh; /* Gunakan viewport height */
    background: url('/static/assets/images/home-bg.gif') no-repeat center
center;
    background-size: cover; /* Agar background menyesuaikan ukuran halaman */
```

```
    z-index: -1; /* Pastikan background di belakang konten */
}

/* Header */
h1 {
    font-size: 2.5rem;
    margin-bottom: 20px;
    color: #333;
    font-weight: bold;
}

/* Table Styles */
table {
    width: 100%;
    border-collapse: collapse;
    margin: 20px 0;
}

th, td {
    padding: 15px;
    text-align: center;
    font-size: 1.1rem;
    border-bottom: 1px solid #ddd;
}

th {
    background-color: #4CAF50;
    color: white;
}

tr:nth-child(even) {
    background-color: #f2f2f2;
}

tr:hover {
    background-color: #f1f1f1;
}

/* Button */
.button {
    background-color: #4CAF50;
    color: white;
    padding: 10px 20px;
    text-align: center;
    text-decoration: none;
    border-radius: 5px;
    margin-top: 20px;
    display: inline-block;
    font-size: 1rem;
}
```

```
.button:hover {
    background-color: #45a049;
    cursor: pointer;
}

/* Responsive Design */
@media (max-width: 768px) {
    table {
        font-size: 0.9rem;
    }

    th, td {
        padding: 12px;
    }

    h1 {
        font-size: 2rem;
    }

    .button {
        font-size: 0.9rem;
    }
}
```

10. leaderboard.js

```
// leaderboard.js
const leaderboardSound = new Audio('/static/sounds/leaderboard.wav');
const clickSound = new Audio('/static/sounds/click.wav');

function loadLeaderboard() {
    fetch('/leaderboard', {
        method: 'GET',
    })
    .then(response => {
        if (!response.ok) {
            throw new Error('Network response was not ok');
        }
        return response.json(); // Mengubah respons ke format JSON
    })
    .then(data => {
        // Cek jika data yang diterima kosong
        if (data.length === 0) {
            document.getElementById('leaderboard-table').innerHTML = "<tr><td
colspan='2'>No leaderboard data available.</td></tr>";
            return; // Tidak melanjutkan jika tidak ada data
        }

        // Menampilkan data leaderboard dalam tabel
```

```
        const leaderboardTable =
document.getElementById('leaderboard-table').getElementsByTagName('tbody')[0];
        leaderboardTable.innerHTML = ''; // Clear tabel terlebih dahulu

        // Menambahkan data leaderboard ke dalam tabel
        data.forEach(entry => {
            const row = document.createElement('tr');
            const usernameCell = document.createElement('td');
            usernameCell.textContent = entry.username;
            const scoreCell = document.createElement('td');
            scoreCell.textContent = entry.score;
            row.appendChild(usernameCell);
            row.appendChild(scoreCell);
            leaderboardTable.appendChild(row);
        });
        // Memainkan suara leaderboard.wav setelah data berhasil dimuat
        const leaderboardSound = new Audio('/static/sounds/leaderboard.wav');
        leaderboardSound.play().catch(error => {
            console.error("Error playing leaderboard sound:", error);
        });
    })
    .catch(error => {
        console.error('Error loading leaderboard:', error);
        document.getElementById('leaderboard-table').innerHTML = "<tr><td
colspan='2'>Error loading leaderboard.</td></tr>";
    });

    // Menambahkan event listener untuk tombol "Back to Homepage"
    const backToHomepageButton = document.querySelector('a.button');
    if (backToHomepageButton) {
        backToHomepageButton.addEventListener('click', function() {
            // Memainkan suara click.wav saat tombol diklik
            const clickSound = new Audio('/static/sounds/click.wav');
            clickSound.play().catch(error => {
                console.error("Error playing click sound:", error);
            });
        });
    }
}

// Panggil fungsi loadLeaderboard saat halaman selesai dimuat
window.onload = function() {
    loadLeaderboard();
};
```