IMPLEMENTASI ALGORITMA *Q-LEARNING* PADA *GAME* EDUKASI BAHASA INGGRIS *LEXICON VAULT*

Arnoldus Julio Pratama, Latius Hermawan
Informatika, Univesitas Katolik Musi Charitas
Jl. Bangau No.60, 9 Ilir, Kec. Ilir Timur III, Palembang, Sumatera Selatan, Indonesia
arnoldusjuliopratama2907@gmail.com

ABSTRAK

Pembelajaran bahasa Inggris melalui media digital semakin populer, namun efektivitasnya sering terhambat oleh sistem permainan yang monoton dan tingkat kesulitan yang statis. Hal ini menyebabkan pemain cepat bosan atau merasa frustrasi karena tingkat kesulitan tidak sesuai dengan kemampuan mereka. Penelitian ini bertujuan merancang dan mengimplementasikan algoritma Q-Learning untuk menciptakan penyesuaian tingkat kesulitan yang dinamis pada game edukasi Bahasa Inggris berbasis RPG 2D bernama "Lexicon Vault" untuk platform Android. Sistem dikembangkan menggunakan Unity Engine dengan metode Agile Scrum. NPC dalam game ini dibekali Q-Learning dengan konfigurasi 6 State, 5 Action, Learning Rate $\alpha = 0{,}1$, dan Discount Factor $\gamma = 0{,}9$. Sistem memberikan Reward +10 untuk jawaban benar dan -5 untuk jawaban salah guna memperbarui Q-Table secara *real-time*. Evaluasi sistem dilakukan melalui analisis konvergensi Q-Table dan Black-Box Testing. Hasil menunjukkan Q-Table berhasil konvergen, memungkinkan sistem menyesuaikan tingkat kesulitan soal secara otomatis berdasarkan performa pemain. Seluruh pengujian dinyatakan lulus, membuktikan penerapan Q-Learning efektif menghasilkan transisi level kesulitan yang dinamis sesuai pola jawaban pemain.

Kata kunci :*Q-Learning*, *Game* Edukasi, *RPG*, *Android*, *Reinforcement Learning*.

1. PENDAHULUAN

Game edukasi telah berkembang menjadi media pembelajaran interaktif yang efektif dalam meningkatkan keterlibatan dan motivasi belajar[1]. Namun, efektivitas media digital berbasis game ini sering kali terkendala oleh rendahnya partisipasi siswa jika game dianggap terlalu monoton dan tidak memberikan tingkat kesulitan yang sesuai bagi pemain. Sebagai contoh, sistem pembelajaran Duolingo yang linier dan repetitif sering kali menimbulkan kejenuhan karena tantangan yang diberikan tidak sebanding dengan peningkatan kemampuan pemain. Model pembelajaran berbasis aplikasi memerlukan tingkat ketekunan (*grit*) yang tinggi untuk mengatasi penurunan minat akibat pola latihan yang monoton[2].

Meskipun efektif bagi pemula dalam membangun dasar kosakata, pola soal yang tidak berubah sering kali menimbulkan kejenuhan bagi pengguna karena tantangan yang diberikan tidak sebanding dengan peningkatan kemampuan mereka[3]. Hal ini menjadi bukti nyata dari keterbatasan sistem tingkat kesulitan statis yang gagal menjaga keseimbangan antara kemampuan pemain dan tantangan permainan. Tantangan teknis utama dalam perancangan game edukasi adalah penyesuaian tingkat kesulitan yang dinamis, yang mampu mengakomodasi berbagai macam kemampuan individual antar pemain secara otomatis [4]. Tingkat kesulitan game yang monoton banyak diterapkan memiliki keterbatasan mendasar: pemain berkemampuan tinggi akan mengalami kebosanan karena kurangnya tantangan, sementara pemain berkemampuan rendah rentan mengalami frustrasi akibat beban kognitif berlebih[5]. Di Indonesia, data menunjukkan bahwa sekitar 78,57% siswa SMP masih memperoleh skor di bawah standar ketuntasan dalam penguasaan kosakata Bahasa Inggris, yang menandakan perlunya media pembelajaran yang lebih adaptif[6]. Penyesuaian tingkat kesulitan yang dinamis hadir sebagai solusi untuk menyesuaikan parameter permainan berdasarkan model pemain secara berkelanjutan[4].

Pada penelitian ini tingkat penyesuaian kesulitan yang dinamis menggunakan Algoritma Reinforcement Learning (RL), khususnya Q-Learning, menawarkan solusi teknis yang menjanjikan karena kemampuannya belajar dari interaksi lingkungan tanpa memerlukan model eksplisit dari kemampuan pemain [7]. Selain itu, kemampuan Reinforcement Learning ini memungkinkan sistem untuk belajar dari interaksi pemain secara berkelanjutan, serta menyesuaikan strategi berdasarkan umpan balik performa. Dengan demikian, media game edukasi dapat menjadi jembatan antara aspek pedagogik dan teknologi pembelajaran cerdas [8]. Algoritma ini mengoptimalkan kebijakan pengambilan keputusan NPC (Non Playable Character) melalui pembaruan iteratif nilai Q. Algoritma ini digunakan untuk mengatasi keterbatasan tersebut dengan mengintegrasikan modul Q-Learning secara langsung (*on-device*) pada Unity Engine untuk menghasilkan mekanisme penyesuaian tingkat kesulitan secara dinamis [9].

2. TINJAUAN PUSTAKA

2.1. *Game* Edukasi

Game Edukasi merupakan *Game* yang dirancang untuk meningkatkan motivasi belajar pemain. Dengan adanya elemen elemen *Game* seperti kompetisi,

hadiah, dan kemajuan level, siswa merasa lebih terdorong untuk berpartisipasi aktif dalam proses belajar. Selain itu juga *Game* edukasi melibatkan tantangan yang harus dipecahkan oleh pemain. Hal ini mendorong siswa untuk berpikir kritis dan secara tidak langsung mengembangkan keterampilan pemecahan masalah [10].

2.2. *Reinforcement Learning*

Reinforcement Learning adalah metode pembelajaran dalam machine learning yang bertujuan untuk memaksimalkan sinyal *Reward* dengan mempelajari hubungan antara kondisi (*State*) dan aksi (*Action*) melalui proses trial and error. Proses trial and error ini melatih sistem untuk memberikan keputusan terbaik. [11]

2.3. *Q-Learning*

Q-Learning merupakan salah satu metode dari *Reinforcement Learning* yang cara kerjanya dengan menentukan keputusan yang paling optimal dari lingkungan yang dinamis. Dengan kata lain metode *Q-Learning* ini bekerja tanpa mengetahui pasti bagaimana keadaan lingkungan yang sebenarnya. [12].

2.4. Penelitian Terdahulu

Beberapa penelitian terdahulu yang serupa tentang *Game* edukasi Bahasa Inggris berbasis *android* menggunakan *MIT APP INVENTOR* yakni penelitian oleh misveria et all [13]. Penerapan *Game* edukasi 3D berbasis *Android* untuk mata pelajaran Bahasa Inggris berhasil dikembangkan dengan metode *Game* Development Life Cycle (GDLC). [14]. Perbedaan penelitian sebelumnya dengan penelitian yang dilakukan penulis adalah berfokus pada penerapan algoritma *Q-Learning* pada *Game* edukasi Bahasa Inggris *Lexicon Vault* untuk menyesuaikan tingkat kesulitan pertanyaan yang diberikan oleh NPC untuk pemain secara dinamis

.

3. METODE PENELITIAN

3.1. Desain dan Jenis Penelitian

Penelitian ini berfokus pada pengembangan aplikasi *Game* edukasi berjudul 'Lexicon Vault'. Dalam proses pengerjaannya, saya menggunakan metode *Agile Scrum* yang dibagi ke dalam tiga tahap utama, yaitu fase *PreGame* untuk tahap awal perancangan, fase *Game* untuk proses *Sprint* selama 5 minggu, dan fase *PostGame* untuk tahap pengujian akhir (Daniel Adi Setya Rahardjo, 2023)

3.2. Fase Pre*Game*

Fase *PreGame* berfokus pada tahap Perencanaan dan analisis awal untuk menyusun kebutuhan fungsional serta non-fungsional sistem. Pada tahap penelitian ini, dilakuka proses merancang arsitektur sistem dan mendefinisikan komponen inti *Q-Learning* yang meliputi *State*, *Action*, dan *Reward*. Selain itu, dibuat pula *Product Backlog*, *storyboard* sebagai panduan dokumentasi desain. Perancangan daftar pertanyaan yang disimpan dalam format file *JSON* dilakukan untuk memastikan struktur penyimpanan data soal dan skor sudah siap sebelum proses penulisan kode dimulai. Seluruh persiapan ini bertujuan agar proses pengembangan memiliki landasan teknis yang kuat dan terorganisir.

3.3. Fase *Game*

Fase *Game* merupakan tahap implementasi utama yang dilakukan melalui siklus *Sprint* dengan durasi 1 minggu di setiap iterasi. Selama iterasi ini,proses penelitian dibangun dengan pembuatan modul peta, kontrol karakter, hingga integrasi lapisan algoritma *Q-Learning* untuk mengatur logika adaptasi tingkat kesulitan soal. Setiap satu siklus *Sprint* selesai, dilakukan pengujian unit secara mandiri untuk memastikan tidak ada masalah baik dari segi fungsional maupun non-fungsional. Apabila ditemukan kesalahan atau *error* di tengah *Sprint*, perbaikan akan dimasukkan kembali ke dalam *backlog* untuk diselesaikan pada iterasi berikutnya. Fokus utama fase ini adalah menghasilkan produk yang fungsional dan responsif terhadap input jawaban pemain secara *Real-time*. Berikut merupakan rincian iterasi dalam 1 minggu.

Tabel 1. Rincian *Sprint* Pengembangan Sistem

Sprint	**Durasi**	**Target Teknis**	**Output**
Sprint 1	1 Minggu	Setup project Unity, implementasi mekanik pergerakan karakter	Scene *Game*play dasar berjalan tanpa error
Sprint 2	1 Minggu	Implementasi sistem bank soal kosakata, logika input jawaban, UI pertanyaan	Modul soal fungsional dengan 3 level kesulitan
Sprint 3	1 Minggu	Implementasi kelas *Q-Learning (Q-Table, update function, State transition),* integrasi ke *GameManager*	Modul AI *Q-Learning* aktif memperbarui *Q-Table*
Sprint 4	1 Minggu	Integrasi UI skor, visualisasi level aktif, feedback jawaban benar/salah, sistem simpan data	Sistem UI lengkap dan responsif
Sprint 5	1 Minggu	Optimasi performa APK, pengujian internal, perbaikan *bug*, build final	File APK stabil siap uji

3.4. Fase *PostGame*

Fase Post*Game* melibatkan tahap evaluasi akhir dan validasi terhadap seluruh fitur yang telah selesai dikembangkan melalui prosedur *Solo Review*. Melakukan peninjauan terhadap setiap fitur yang telah selesai dibangun, serta mengevaluasi kendala selama

pengerjaan untuk dijadikan bahan perbaikan pada tahap selanjutnya. Pengujian kualitas perangkat lunak dijalankan menggunakan metode *Black-Box Testing* untuk verifikasi fungsional eksternal dan *White-Box Testing* untuk memastikan jalur logika internal sesuai rancangan. Selain itu, dilakukan analisis terhadap log konvergensi *Q-Table* untuk mengukur stabilitas serta akurasi adaptasi sistem *AI* dalam sesi simulasi permainan. Hasil dari fase ini menjadi dasar penilaian akhir untuk menentukan apakah aplikasi tersebut sudah sesuai sebagai media pembelajaran bagi Pengguna.

3.5. Perancangan Algoritma *Q-Learning*

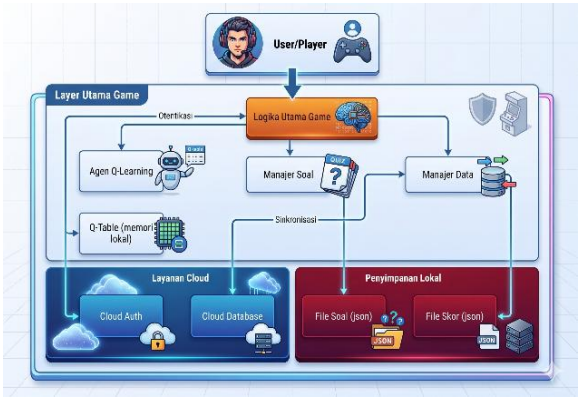
Sistem menggunakan 6 *State* (S0-S5) seperti yang terlihat pada Tabel 2 yang merepresentasikan level soal aktif dan tren jawaban pemain. NPC dapat mengambil 5 *Action* (A0-A4) seperti yang terlihat pada table 3 untuk mempertahankan, menaikkan, atau menurunkan level kesulitan soal. Fungsi *Reward* ditetapkan berdasarkan skor performa pemain, yaitu +10 untuk jawaban benar sebagai penguat positif dan -5 untuk jawaban salah sebagai penalti. Pembaruan nilai *Q* menggunakan parameter alpha = 0,1 dan gamma = 0,9 sesuai dengan persamaan berikut:

$$Q(s,a) = Q(s,a) + \alpha\,[\, r + \gamma.\ max\ Q('s,'a) - Q(s,a)]$$

Penting untuk dicatat bahwa sistem memisahkan variabel *Reward* untuk agen AI dan variabel skor untuk pengguna. Fungsi *Reward* menggunakan skema *Reward* (+10 dan -5) untuk mengoptimalkan stabilitas perhitungan algoritma *Q-Learning* sesuai parameter α dan γ yang ditetapkan. Sementara itu, sistem skor menggunakan skala +10 dan -5 sebagai instrumen evaluasi hasil belajar dan elemen kompetitif bagi pemain, yang secara otomatis terekam ke dalam data manajer.

3.6. Arsitektur Sistem

Arsitektur Sistem *Game Lexicon Vault* ini terdiri dari 3 layer. Terdiri dari layer utama *Game*(Client & Data), Layer Manajer dan Agen, dan Layer layanan Cloud dan Penyimpanan. Berikut merupakan gambar dari Arsitektur sistem *Game* Lexicon Vault:



Gambar 1. Arsitektur Sistem

Layer utama *Game* berfokus pada interaksi langsung antara pemain dan antarmuka aplikasi *Lexicon Vault* di perangkat *Android*. Pemain memberikan input melalui aksi di dalam permainan yang kemudian direspon secara visual oleh sistem melalui *Game Scene*. Seluruh navigasi dan pergerakan karakter dikelola oleh logika utama *Game* untuk memastikan pengalaman bermain yang lancar dan imersif. Komponen ini juga bertanggung jawab menangani tampilan antarmuka (*UI*) serta sinkronisasi gerakan karakter selama sesi permainan berlangsung. Data interaksi dari lapisan ini selanjutnya diteruskan ke bagian pemrosesan pusat untuk menentukan langkah adaptasi kesulitan berikutnya.

Layer manajer dan agen berfungsi sebagai pusat kendali logika yang memproses seluruh data permainan secara *Real-time*. Bagian ini terdiri dari manajer soal, manajer skor, dan manajer jaringan yang bekerja secara terintegrasi untuk mendukung fitur *multiplayer* menggunakan Photon. Implementasi kecerdasan buatan dilakukan oleh agen *Q-Learning* yang bertugas menghitung mekanisme Penyesuaian tingkat kesulitan yang dinamis. Nilai pada *Q-Table* diperbarui secara dinamis menggunakan parameter α dan γ berdasarkan respons jawaban benar atau salah dari pemain. Arsitektur ini memungkinkan sistem untuk menjadi lebih cerdas dan responsif dalam menyesuaikan tingkat kesulitan soal dengan kemampuan individual pengguna.

Layer layanan cloud dan penyimpanan bertanggung jawab atas persistensi data serta integrasi fitur daring secara global. Sistem menggunakan penyimpanan lokal berbasis file *JSON* untuk mencatat bank soal dan skor pemain secara mandiri di perangkat. Otentikasi pengguna dikelola melalui layanan *cloud* untuk memastikan keamanan akses serta sinkronisasi profil masing-masing pemain. Pemanfaatan *Photon* API dan *UGS Leaderboard* memungkinkan adanya interaksi antar-pemain serta sistem peringkat global yang kompetitif. Secara keseluruhan, lapisan ini menjaga integritas data dan stabilitas konektivitas sistem agar hasil belajar dapat terdokumentasi dengan baik.

3.7. Perancangan Dan Implementasi Algortima *Q-Learning*

a. *State*: merupakan salah satu komponen utama yang sangant mempengaruhi dari cara kerja algoritma *Q-Learning*. *State* merepresentasikan kondisi pembelajaran pemain pada suatu waktu t. *State* didefinisikan berdasarkan dua dimensi: level kesulitan soal aktif dan tren performa jawaban terkini (dihitung dari [n] jawaban terakhir).

Tabel 2. *State* Pada *Q-Learning Agent*

***State* ID**	**Level Soal Aktif**	**Tren Pefroma**	**Deskripsi Kondisi**
S0	Mudah	Netral/Awal	Kondisi Awal
S1	Mudah	Positif (Benar ≥ 3)	Pemain Menguasai Level Mudah
S2	Sedang	Netral	Pemain Stabil di Level sedang
S3	Sedang	Positif (Benar ≥ 3)	Pemain Menguasai Level Sedang
S4	Sulit	Netral	Pemain Stabil di Level Sulit
S5	Sedang/Sulit	Negatif (Salah ≥ 3)	Pemain kesulitan perlu menururn kan level

b. *Action*: *Action* merepresentasikan keputusan yang dieksekusi agen terhadap sistem soal sebagai respons atas *State* yang diobservasi. Di dalam *Game* ini *Action* memberikan efek pada sistem soal yaitu menaikkan dan menurunkan level pertanyaan yang diberikan oleh NPC. Level pertanyan di dalam *Game* ini terdiri dari 3 level yaitu level mudah, sedang, dan sulit. *Action* dilakukan setelah memvalidasi ulang jawaban dari pemain. Ketika jawaban yang diberikan pemain benar makan level soal akan ditingkatkan, sebaliknya Ketika pemain menjawab pertanyaan dengan salah maka level diturunkan.

Tabel 3. *Action* Pada *Q-Learning Agent*

***Action* ID**	**Nama *Action***	**Efek Pada Sistem Soal**
A0	Level Awal (Mudah)	Level Soal Tidak Berubah
A1	Naikkan Level (mudah→ sedang)	Soal Berubah ke level Sedang
A2	Naikkan Level (Sedang→Sulit)	Soal berubah ke level Sulit
A3	Turunkan Level (Sulit →Sedang)	Soal berubah ke level Sedang
A4	Turunkan Level (Sedang→Mudah)	Soal berubah ke level Mudah

c. *Reward*: *Reward* dalam *Game* Lexicon Vault didefinisikan sebagai sinyal umpan balik numerik yang diterima oleh agen kecerdasan buatan setelah pemain memberikan jawaban atas soal yang diberikan. Fungsi *Reward* ini ditetapkan secara skema *Reward*, yaitu +10 jika jawaban pemain benar dan -5 jika jawaban salah, guna memberikan sinyal belajar yang jelas bagi algoritma *Q-Learning*. Nilai tersebut digunakan secara internal untuk memperbarui estimasi nilai pada *Q-Table* melalui persamaan Bellman agar agen dapat menyesuaikan tingkat kesulitan soal secara *Real-time*. Skema fungsi *Reward* pada *Game* lexicon vault akan dijelaskan pada table berikut:

Tabel 4. Skema Fungsi *Reward*

Kondisi Interaksi (Jawaban Pemain)	**Nilai *Reward* (Rt)**	**Sinyal Pembelajaran**
Menjawab Dengan Benar	+10	Memberikan insentif positif yang mengindikasikan tingkat kesulitan saat ini sudah tepat atau layak dipertahankan.
Menjawab Dengan Salah	-5	Memberikan penalti yang mengindikasikan bahwa tingkat kesulitan terlalu berat dan perlu disesuaikan.

3.8. Fungsi Pembaruan Table (Persamaan Bellman)

Proses adaptasi tingkat kesulitan soal dalam sistem Lexicon Vault dilakukan melalui pembaruan nilai pada *Q-Table* secara berkala berdasarkan interaksi pemain. Pembaruan ini dihitung menggunakan persamaan Bellman dengan mengintegrasikan variabel *Reward* yang diterima serta estimasi nilai optimal di masa depan. Melalui penerapan parameter *Learning Rate* (α) sebesar 0,1 dan *Discount Factor* (γ) sebesar 0,9, agen dapat mempelajari kebijakan pengambilan keputusan yang paling efektif untuk menyeimbangkan tantangan permainan. Berikut adalah tabel simulasi pembaruan nilai *Q-Table* yang menggambarkan perubahan estimasi nilai untuk setiap pasangan *State-Action* setelah menerima umpan balik dari pemain:

Tabel 5. Table Parameter Persamaan *Bellman*

Notasi	**Deskripsi**	**Nilai yang digunakan**
$Q(s_t,a_t)$	Nilai Q saat ini untuk pasangan *State* s_t dan *Action* a_t	-
α	*Learning Rate* berfungsi untuk mengontrol seberapa besar pembaruan menggeser nilai Q lama	0,1
γ	Discount Factor berfungsi untuk mengontrol bobot *Reward* masa depan terhadap *Reward* saat ini	0,9
r_t	*Reward* dinamis yang diterima setelah agem mengeksekusi actio a_t	+10/-5
maxQ(s',a')	Estimasi returt tertinggi yang dapat dicapai dari *State* berikutnya	-

3.9. Inisialisasi *Q-Table*

Sebelum memulai proses pembelajaran, agen cerdas diinisialisasi dalam kondisi tanpa pengetahuan awal mengenai preferensi atau kemampuan pemain. Seluruh nilai di dalam matriks *Q-Table* ditetapkan pada angka nol, yang merepresentasikan kondisi *tabula rasa* di mana agen akan mulai mengeksplorasi setiap pasangan *State* dan *Action* berdasarkan umpan balik nyata. Struktur awal ini berfungsi sebagai fondasi bagi algoritma untuk melakukan pembaruan nilai secara iteratif menggunakan persamaan Bellman seiring dengan berlangsungnya sesi permainan. Berikut adalah rincian inisialisasi *Q-Table* yang mencakup seluruh kemungkinan interaksi sistem sebelum proses adaptasi tingkat kesulitan dimulai:

Tabel 6 Tabel Inisialisasi *Q-Table*

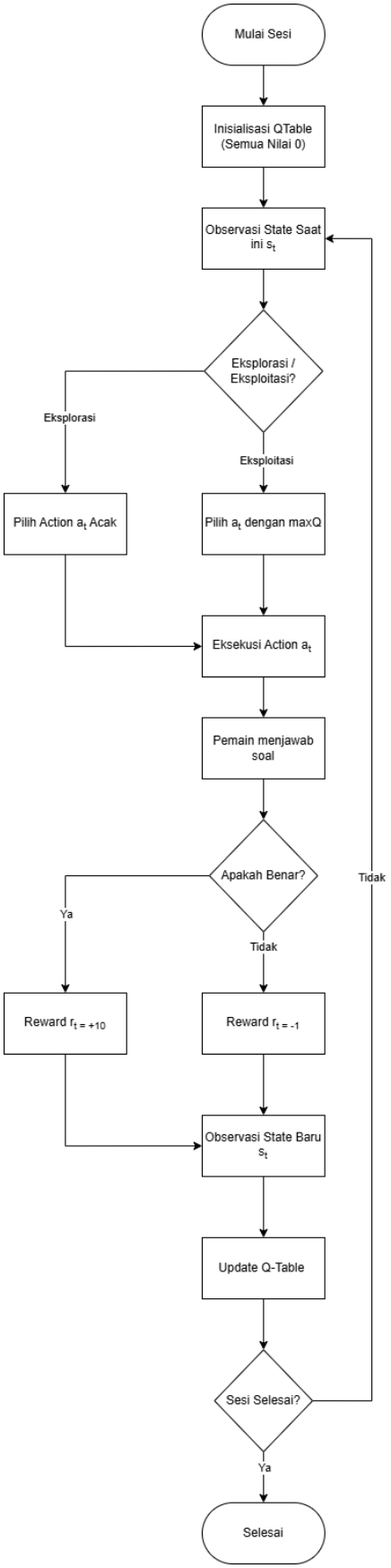
State/Action	**A0**	**A1**	**A2**	**A3**	**A4**
S0	0	0	0	0	0
S1	0	0	0	0	0
S2	0	0	0	0	0
S3	0	0	0	0	0
S4	0	0	0	0	0
S5	0	0	0	0	0

3.10. Alur Kerja Algoritma

Alur kerja algoritma *Q-Learning* pada *Game Lexicon Vault* dirancang untuk berjalan secara siklis guna memastikan adaptasi tingkat kesulitan soal terjadi secara kontinu. Proses dimulai dengan tahap inisialisasi *Q-Table*, di mana seluruh sel dalam matriks *State-Action* ditetapkan pada nilai 0. Setelah sesi dimulai, agen melakukan observasi terhadap *State* saat ini (s_t) sebelum memutuskan untuk melakukan eksplorasi (memilih aksi acak) atau eksploitasi (memilih aksi dengan nilai *Q* tertinggi). Tahapan selanjutnya adalah eksekusi aksi (a_t), yaitu sistem menentukan level soal (Mudah, Sedang, atau Sulit) yang akan ditampilkan kepada pemain. Setelah pemain memberikan jawaban, sistem melakukan validasi:Jika jawaban Benar, agen menerima *Reward* (r_t). Setelah menerima *Reward*, agen akan melakukan observasi *State* baru (s{t+1}) dan melakukan pembaruan nilai pada *Q-Table* menggunakan Persamaan Bellman:

$$Q(s,a) \leftarrow Q(s,a) + \alpha\,[r + \gamma \max Q(s',a') - Q(s,a)]$$

agen secara otomatis mempelajari kebijakan terbaik untuk menyesuaikan tingkat kesulitan soal pada iterasi berikutnya hingga sesi permainan dinyatakan selesai.



Gambar 2. Alur Kerja Algoritma

3.11. Metode Evaluasi Sistem

Evaluasi sistem dalam penelitian ini difokuskan pada validasi teknis algoritma dan stabilitas fungsional perangkat lunak. Metode pengujian yang digunakan adalah *Black-Box* Testing untuk memverifikasi integritas operasional setiap modul sistem, mulai dari manajemen data hingga sinkronisasi layanan *cloud*. Selain itu, dilakukan analisis kuantitatif terhadap performa agen cerdas melalui observasi log konvergensi *Q-Table*. Evaluasi ini bertujuan untuk mengukur tingkat akurasi algoritma dalam merespons *input* pengguna secara *Real-time* serta memastikan bahwa mekanisme Penyesuaian tingkat kesulitan yang dinamis beroperasi sesuai dengan parameter α dan γ yang telah ditentukan tanpa adanya kegagalan logika pada arsitektur sistem.

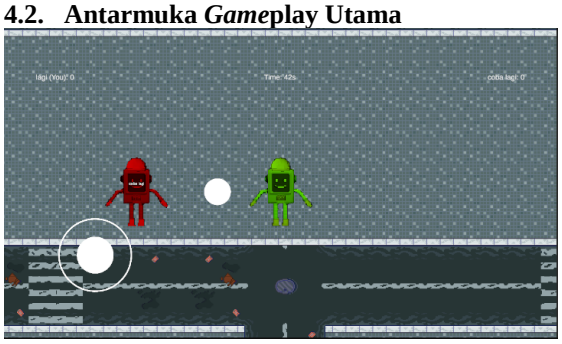
4. HASIL DAN PEMBAHASAN

4.1. Antarmuka Lobby dan Manajemen Room

Gambar 3 Antarmuka Lobby Dan Manajemen Room

Antarmuka Lobby Scene (Gambar 3) berfungsi sebagai gerbang utama bagi pemain untuk menentukan identitas dan mengatur sesi permainan. Pada bagian ini, sistem mengintegrasikan Photon API untuk memfasilitasi pembuatan (Create) dan bergabung (Join) ke dalam room permainan. Secara teknis, lapisan ini bertanggung jawab untuk melakukan inisialisasi awal data pemain dan memastikan konektivitas ke server sebelum agen AI mulai melakukan observasi terhadap *State* awal permainan.

4.2. Antarmuka *Game*play Utama



Gambar 4 Antarmuka *Game*play Utama

Tampilan *Game*play (Gambar 4) menyajikan lingkungan *RPG 2D* dalam perspektif top-down. Di sini, pemain dapat mengendalikan karakter untuk mengeksplorasi peta dan mencari NPC pemberi tantangan. Secara sistem, layer ini menangkap koordinat pergerakan dan memicu peristiwa interaksi yang akan mengirimkan sinyal kepada Manajer Soal. Struktur UI pada tahap ini dibuat minimalis dengan fokus pada kontrol pergerakan untuk menjaga performa frame rate pada perangkat *Android*.

4.3. Antarmuka Panel Pertanyaan

Gambar 5 Antarmuka Panel Pertanyaan

Panel Pertanyaan (Gambar Z) merupakan komponen paling krusial dalam sistem, karena di sinilah siklus utama *Q-Learning* terjadi. Saat panel ini muncul, agen AI telah menentukan *Action* (tingkat kesulitan soal) berdasarkan *State* pemain saat ini. Panel ini menampilkan teks pertanyaan dan pilihan jawaban yang menjadi sumber input bagi algoritma. Setiap kali pemain memilih jawaban, sistem secara otomatis menghitung Rewar dan melakukan pembaruan pada *Q-Table*, yang hasilnya disimpan secara transparan ke dalam berkas log CSV.

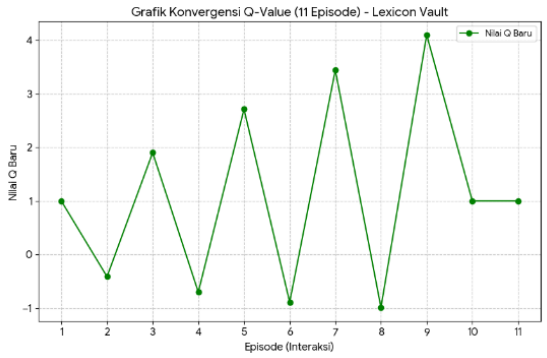
4.4. Analisis Konvergensi *Q-Table*

Data yang disajikan dalam analisis konvergensi ini diperoleh melalui mekanisme pencatatan otomatis (*logging system*) yang diintegrasikan ke dalam Bahasa pemrograman *C#* pada Unity. Setiap kali pemain melakukan interaksi dengan NPC dan memberikan jawaban, sistem secara *Real-time* menangkap dan merekam seluruh parameter teknis algoritma ke dalam sebuah berkas penyimpanan lokal berformat *.csv*. Berkas log ini mencatat variabel-variabel yang meliputi nomor *Episode*, identitas pemain, *State* awal, aksi yang diambil, nilai *Reward*, hingga pembaruan nilai *Q* (*Nilai Q Baru*) yang dihitung berdasarkan persamaan Bellman.

Berdasarkan data tersebut, dilakukan proses ekstraksi untuk menghasilkan grafik konvergensi *Q-Value* guna memberikan gambaran visual mengenai stabilitas pembelajaran agen cerdas selama sesi permainan berlangsung. Grafik ini memetakan fluktuasi nilai *Q* terhadap urutan *Episode* jawaban pemain untuk membuktikan bahwa mekanisme Penyesuaian tingkat kesulitan yang dinamis bekerja sesuai dengan performa nyata pengguna di lapangan. Berikut adalah cuplikan data log berformat *CSV* dan visualisasi grafik konvergensi yang dihasilkan dari sesi pengujian sistem:

	A	B	C	D	E	F	G	H
1	Episode	Pemain	State (Level)	Action (Benar/Salah)	Reward	Nilai_Q_Lama	Nilai_Q_Baru	Next_State (Level_Baru)
2	1	Arnold	Level 1	Benar	10	0	1	Level 2
3	2	Arnold	Level 2	Salah	-5	0	-0.41	Level 1
4	3	Arnold	Level 1	Benar	10	1	1.9	Level 2
5	4	Arnold	Level 2	Salah	-5	-0.41	-0.698	Level 1
6	5	Arnold	Level 1	Benar	10	1.9	2.71	Level 2
7	6	Arnold	Level 2	Salah	-5	-0.698	-0.884	Level 1
8	7	Arnold	Level 1	Benar	10	2.71	3.439	Level 2
9	8	Arnold	Level 2	Salah	-5	-0.884	-0.986	Level 1
10	9	Arnold	Level 1	Benar	10	3.439	4.095	Level 2
11	10	Arnold	Level 2	Benar	10	0	1	Level 3
12	11	Arnold	Level 3	Benar	10	0	1	Level 3

Gambar 6. Data Log Yang Dihasilkan Dari Satu Sesi Permainan



Gambar 7. Data Log yang sudah dikonversi kedalam grafik

4.5. Hasil *Black-Box Testing*

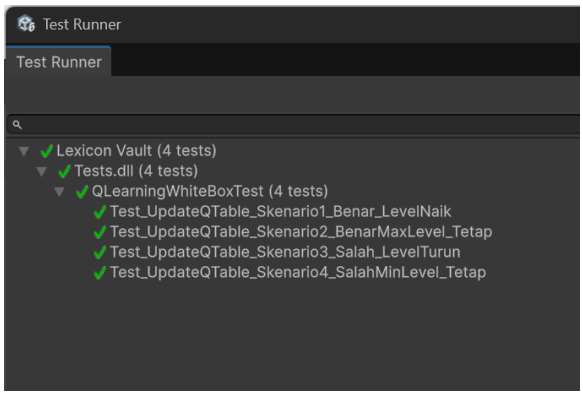
Pengujian fungsionalitas pada sistem *Lexicon Vault* dilakukan menggunakan metode *Black-Box Testing* untuk memastikan seluruh fitur berjalan sesuai dengan spesifikasi kebutuhan yang telah dirancang sebelumnya. Fokus utama pengujian ini meliputi validasi proses otentikasi pengguna, responsivitas elemen antarmuka (UI), serta ketepatan transisi tingkat kesulitan soal yang dikelola secara otomatis oleh agen AI. Berdasarkan hasil eksekusi pada berbagai skenario *input*, seluruh fitur fungsional dinyatakan berhasil atau memenuhi kriteria kelulusan (berhasil) tanpa ditemukan kesalahan logika yang menghambat jalannya permainan. Hal ini membuktikan bahwa sistem mampu merespons setiap tindakan pemain dengan stabil, termasuk dalam hal sinkronisasi data skor global melalui *leaderboard* serta akurasi pencatatan log *Q-Table* ke dalam memori lokal. Secara keseluruhan, hasil pengujian ini menunjukkan bahwa aplikasi telah memenuhi standar kelayakan teknis untuk digunakan sebagai media pembelajaran adaptif yang andal bagi Pemain. Rincian hasil pengujian fungsionalitas sistem secara mendetail disajikan dalam Tabel 7 berikut ini:

Tabel 7. Tabel Hasil Pengujian *Black Box*

No	Komponen Pengujian	Parameter Input	Output Sistem (Expected)	Status
1	*Module Authentication*	Kredensial Pengguna	Akses diberikan & profil dimuat dari Cloud Server.	berhsdil
2	*Logic - Positive*	*Positive Reward (+10)*	Nilai *Q* diperbarui & sistem memicu kenaikan level kesulitan.	berhsdil
3	*Logic - Negative*	*Negative Reward (-5)*	Nilai *Q* diperbarui & sistem memicu penurunan level kesulitan.	berhsdil
4	*Q-Table Convergence*	Interaksi Berulang	Estimasi nilai pada *Q-Table* mencapai titik stabil/konvergen.	berhsdil
5	*Automated Logging*	Sesi Berakhir	Berkas .csv terbentuk otomatis dengan struktur kolom yang valid.	berhsdil
6	*Data Synchronization*	*Score Update*	Data terunggah dan sinkron dengan leaderboard global secara *Real-time*.	berhsdil

4.6. Hasil *Test Runner Unity*

Setelah melakukan *Black-Box Testing* Untuk memvalidasi logika internal algoritma *Q-Learning*, pengujian tidak lagi dilakukan secara manual melainkan secara otomatis menggunakan fitur Test Runner bawaan Unity guna memastikan perhitungan matematis terbebas dari kesalahan (*logic error*). Pengujian difokuskan pada empat skenario krusial Penyesuaian tingkat kesulitan yang dinamis, yaitu evaluasi transisi state saat pemain menjawab benar atau salah, terutama ketika pemain menyentuh kondisi batas atas (level maksimal) dan batas bawah (level minimum). Hasil eksekusi menunjukkan bahwa seluruh simulasi mendapatkan status lulus (*Passed*), di mana agen *AI* secara presisi mampu menaikkan, menurunkan, atau menahan tingkat kesulitan soal sesuai dengan rumusan pada skrip *C#*. Keberhasilan pengujian otomatis ini membuktikan bahwa fungsi matematika dan alur percabangan pada sistem telah bekerja sangat akurat, sehingga *Q-Learning* dipastikan siap beradaptasi dengan pola jawaban pemain secara real-time tanpa memicu kendala sistem.



Gambar 8. Hasil *Test Runner* Pada Unity

5. KESIMPULAN DAN SARAN

untuk mengakomodasi perbedaan kemampuan setiap pemain, sehingga efektif mencegah kebosanan maupun frustrasi. Penggunaan parameter $\alpha = 0{,}1$ dan $\gamma = 0{,}9$ memungkinkan sistem untuk memperbarui *Q-Table* secara akurat dan mencapai kebijakan optimal dalam menaikkan, mempertahankan, atau menurunkan level soal. Keberhasilan ini divalidasi melalui hasil pengujian otomatis pada *Test Runner* yang mengonfirmasi bahwa transisi *state* dan perhitungan rumus berjalan stabil tanpa adanya kesalahan logika (*logic error*). Untuk penelitian selanjutnya, disarankan agar dilakukan pengembangan parameter input seperti memperhitungkan waktu respons pemain, penggunaan algoritma *Deep Q-Network* (DQN) untuk skala *state* yang lebih besar, serta pengujian berskala luas kepada pengguna akhir melalui distribusi rilis publik di aplikasi seperti Google Play Store.

DAFTAR PUSTAKA

[1] Y. Jia and X. Y. Zhou, "q-Learning in Continuous Time," 2023. [Online]. Available: http://jmlr.org/papers/v24/22-0755.html.

[2] E. Sudina and L. Plonsky, "The effects of frequency, duration, and intensity on L2 learning through Duolingo," *J. Second Lang. Stud.*, vol. 7, no. 1, pp. 1–43, 2024, doi: 10.1075/jsls.00021.plo.

[3] Y. Kim, C. Payant, S. Skalicky, and Y. Namkung, "Comparing the effectiveness of Duolingo, Classroom instruction, and Classroom + Duolingo instruction conditions on beginner-level French language development," *Stud. Second Lang. Acquis.*, pp. 1–28, 2026, doi: 10.1017/S0272263126101521.

[4] N. Fisher and A. K. Kulshreshth, "Exploring Dynamic Difficulty Adjustment Methods for Video Games," *Virtual Worlds*, vol. 3, no. 2, pp. 230–255, 2024, doi: 10.3390/virtualworlds3020012.

[5] F. Ignasia and H. Haryanto, "Pengaruh Pembelajaran Adaptif Berbasis Game dengan Personalisasi terhadap Hasil Belajar Peserta Didik dalam Konteks Keragaman Karakteristik," *J. Pendidik. Mat. dan Sains*, vol. 13, no. Special_issue, pp. 273–282, 2025, doi: 10.21831/jpms.v13ispecial_issue.89758.

[6] M. N. Ikhsan, Y. Mardianti Zebua, and F. N. Tarigan, "Analisis Kesulitan Dan Media Pembelajaran Kosakata Bahasa Inggris Bagi Siswa SMP Negeri 2 Gebang," *J. Dunia Pendidik. (Articles Progress)*, vol. 3, 2023.

[7] V. Wijaya, "Implementasi Q-Learning Pada Agen yang Memainkan Permainan SOS," vol. 10, no. 4, pp. 3163–3171, 2025.

[8] M. Neves, M. Vieira, and P. Neto, "A study on a Q-Learning algorithm application to a manufacturing assembly problem," *J. Manuf. Syst.*, vol. 59, pp. 426–440, Apr. 2021, doi: 10.1016/j.jmsy.2021.02.014.

[9] C. H. R. Souza, S. S. Oliveira, L. O. Berretta, and S. T. Carvalho, "Large Language Models and Dynamic Difficulty Adjustment: An Integration Perspective," *Simp. Bras. Jogos e Entretenimento Digit.*, no. SBGames, pp. 31–36, 2024, doi: 10.5753/sbgames_estendido.2024.241217.

[10] S. Indra, E. Sujadi, and H. P. Putra, "Evaluasi Dampak Game Based Learning dalam Era Pendidikan Digital: Systematic Literature ReviewDigital game-based learning: Towards an experiential gaming model," *Al-Riwayah J. kependidikan*, vol. 17, no. April, pp. 1–21, 2025, doi: 10.1016/j.iheduc.2004.12.001.1.

[11] A. Riedmann, P. Schaper, and B. Lugrin, *Reinforcement Learning in Education: A Systematic Literature Review*, vol. 35, no. 5. Springer New York, 2025. doi: 10.1007/s40593-025-00494-6.

[12] Z. Wang *et al.*, *Article in Press Dynamic programming with Q-learning based reinforcement learning optimization for multi IN*. 2026.

[13] M. V. Waru, A. Nuzul, H. Buana, M. I. Dharma, S. Ayu, and U. Lamappapoleonro, "Perancangan Game Edukasi Berbasis Android Untuk Mendukung Pembelajaran Bahasa Inggris Di Sdn 41 Tonrong Pejja," *PGSD Univ. Lamappapoleonro*, vol. 4, no. 1, pp. 28–39, 2025.

[14] M. A. Fitrah and D. Priyanto, "Penerapan Game Edukasi 3D Pada Mata Pelajaran Bahasa Inggris Menggunakan Metode Game Development Life Cycle (GDLC) Berbasis Android," *Corisindo 2025*, no. September 2025, pp. 483–490, 2025, [Online]. Available: https://journal.universitasbumigora.ac.id/corisindo2025/article/view/5544