

Implementasi Algoritma Elo Rating System untuk Penyesuaian Tingkat Kesulitan Dinamis pada Game Edukasi Logika Pemrograman Python Berbasis Android

Yono Tuah Simamora^{*1}, Zainal Azis²

Universitas Muhammadiyah Sumatera Utara^{1,2}

e-mail: 123ynpardamean321simamora@gmail.com^{*1}, zainalazis@umsu.ac.id²

ABSTRAK

Pembelajaran logika pemrograman Python bagi pemula memerlukan media yang mampu menyesuaikan tantangan dengan kemampuan pengguna. Penelitian ini bertujuan mengembangkan PyQuest, game edukasi Android yang menerapkan Elo Rating System sebagai mekanisme Dynamic Difficulty Adjustment untuk memilih soal secara adaptif. Pengembangan menggunakan Multimedia Development Life Cycle yang meliputi pengonsepan, perancangan, pengumpulan materi, pembuatan, pengujian, dan distribusi. Aplikasi dibangun dengan Flutter dan Dart, menggunakan SQLite untuk penyimpanan lokal serta Supabase untuk sinkronisasi profil dan papan peringkat. Bank soal terdiri atas 280 butir pada materi variabel, operator, percabangan, dan perulangan. Interaksi pemain dan soal dimodelkan sebagai pertandingan zero-sum, kemudian rating diperbarui menggunakan expected score dan K-Factor dinamis. Soal berikutnya dipilih dengan Nearest Neighbor Search berdasarkan selisih rating terkecil. Pengujian Black-Box terhadap 15 skenario menghasilkan tingkat validitas 100%. Evaluasi System Usability Scale terhadap 20 mahasiswa memperoleh skor 81,3 yang termasuk kategori excellent. Hasil menunjukkan bahwa mekanisme Elo dapat diimplementasikan secara fungsional sebagai penggerak adaptasi kesulitan real-time, sedangkan antarmuka PyQuest memiliki tingkat kegunaan yang tinggi bagi pengguna pemula. Temuan ini mendukung penggunaan model rating ringan untuk personalisasi latihan pemrograman pada perangkat bergerak tanpa komputasi kompleks.

Kata kunci : Android; Elo Rating; Game Edukasi; Kesulitan Dinamis; Python.

ABSTRACT

Learning introductory Python programming logic requires media that can align task difficulty with individual ability. This study aims to develop PyQuest, an Android educational game that applies the Elo Rating System as a Dynamic Difficulty Adjustment mechanism for adaptive question selection. Development followed the Multimedia Development Life Cycle, comprising concept, design, material collecting, assembly, testing, and distribution. The application was built with Flutter and Dart, using SQLite for local storage and Supabase for profile synchronization and a global leaderboard. The question bank contains 280 items covering variables, operators, branching, and loops. Player-question interactions are modeled as zero-sum matches, and ratings are updated using expected scores and a dynamic K-Factor. The next question is selected through Nearest Neighbor Search based on the smallest rating difference. Black-Box testing across 15 scenarios produced a 100% valid result. System Usability Scale evaluation involving 20 students yielded a score of 81.3, categorized as excellent. The findings indicate that Elo can functionally drive real-time difficulty adaptation, while the PyQuest interface provides high usability for novice users. This approach offers lightweight personalization for mobile programming practice.

Keywords : Adaptive Learning; Android; Educational Game; Elo Rating; Python.

PENDAHULUAN

Perkembangan teknologi informasi memperluas pemanfaatan perangkat bergerak berbasis Android sebagai media belajar mandiri dan interaktif. Pada rumpun ilmu komputer, kemampuan memahami logika pemrograman menjadi fondasi bagi mahasiswa untuk membangun cara berpikir algoritmik, menyusun langkah penyelesaian masalah, serta menerjemahkan solusi ke dalam kode. Python banyak digunakan sebagai bahasa pengantar bagi pemula karena struktur sintaksnya relatif ringkas dan mudah dibaca. Meskipun

demikian, materi seperti variabel, operator, percabangan, dan perulangan tetap menuntut pemahaman abstraksi, penelusuran alur eksekusi, dan kemampuan memprediksi keluaran program. Ketika penyajian materi hanya berlangsung secara satu arah atau melalui latihan yang seragam, mahasiswa dengan kemampuan awal yang berbeda akan menerima tingkat tantangan yang sama sehingga pengalaman belajar tidak selalu sesuai dengan kebutuhan masing-masing.

Pemanfaatan *game* edukasi menawarkan cara untuk mengubah latihan menjadi pengalaman yang lebih aktif melalui tujuan, aturan, tantangan, interaksi, dan umpan balik. Konsep *serious game* menempatkan tujuan pendidikan sebagai fungsi utama permainan, sedangkan unsur hiburan digunakan untuk mempertahankan keterlibatan pengguna (Abt, 1970). Dalam kerangka *digital game-based learning*, mekanik permainan dapat dipadukan dengan konten pembelajaran sehingga pengguna memperoleh ruang untuk mencoba, gagal, menerima umpan balik, dan memperbaiki strategi secara berulang (Prensky, 2001). Bukti yang lebih mutakhir di Indonesia juga menunjukkan bahwa *game* kuis berbasis perangkat bergerak dapat menjadi alternatif media yang interaktif dan menyenangkan untuk pembelajaran materi komputasi (Suryadila et al., 2025). Pengembangan *game* Flutter berbasis Android bahkan telah digunakan pada pembelajaran aritmatika dengan hasil penerimaan pengguna yang tinggi, sehingga ekosistem lintas platform tersebut relevan untuk pengembangan media pendidikan bergerak (Lionza et al., 2026).

Masalah utama pada banyak *game* edukasi bukan hanya bagaimana membuat antarmuka menarik, tetapi bagaimana mempertahankan kesesuaian antara kesulitan dan kemampuan pengguna. Plass dan Pawar (2020) menempatkan adaptivitas sebagai mekanisme penting dalam lingkungan belajar berbantuan teknologi karena sistem perlu merespons karakteristik, performa, atau kebutuhan pembelajar. Secara psikologis, kebutuhan tersebut beririsan dengan teori *flow* yang menjelaskan bahwa keterlibatan optimal muncul ketika tingkat tantangan seimbang dengan keterampilan individu (Csikszentmihalyi, 1990). Tantangan yang jauh melampaui kemampuan dapat memunculkan kecemasan dan frustrasi, sedangkan tugas yang terlalu mudah dapat menimbulkan kebosanan. Dengan demikian, penyajian urutan soal yang statis berpotensi mengabaikan variasi kurva belajar antarindividu.

Dynamic Difficulty Adjustment (DDA) dikembangkan untuk mengatasi ketidaksesuaian tersebut dengan mengubah tingkat tantangan berdasarkan performa pemain selama permainan berlangsung. Pada *serious game*, penyesuaian kesulitan dapat digunakan untuk menjaga keseimbangan aspek belajar dan kesenangan. Nugraha et al. (2022) menerapkan DDA dan *procedural content generation* pada *serious game* untuk keterampilan membaca dan menunjukkan bahwa dinamika kesulitan dapat dipadukan dengan tujuan belajar. Penelitian Soedargo dan Junaedi (2022) menegaskan bahwa perbedaan tingkat keterampilan dan kurva belajar membuat kesulitan statis rentan menghasilkan permainan yang tidak seimbang, sehingga DDA berbasis logika *fuzzy* digunakan untuk menyesuaikan konten permainan. Pada ranah *mobile game* rehabilitasi, Krisdiawan et al. (2023) juga menggunakan DDA untuk merespons perbedaan kebutuhan dan kecepatan belajar pengguna. Temuan Sany dan Angkasa (2025) memperluas arah tersebut melalui DDA berbasis *reinforcement learning* pada *game* edukasi Android yang menyesuaikan parameter permainan secara *real-time*. Kajian sistematis terbaru menunjukkan bahwa DDA semakin banyak ditempatkan sebagai komponen penting untuk mendukung efektivitas dan pengalaman pada *serious game* (Víteková et al., 2026).

Walaupun DDA dapat dibangun dengan logika *fuzzy*, pohon keputusan, atau model pembelajaran mesin, perangkat bergerak membutuhkan mekanisme yang cukup ringan untuk dihitung berulang kali selama sesi. Fisher dan Kulshreshth (2025) menjelaskan bahwa DDA modern dapat memanfaatkan performa pemain sebagai dasar perubahan kesulitan secara langsung, tetapi pemilihan teknik harus mempertimbangkan karakter permainan dan biaya komputasi. Untuk kebutuhan tersebut, Elo Rating System menarik karena mengubah hasil interaksi menjadi pembaruan nilai kemampuan secara probabilistik. Elo pada awalnya digunakan untuk mengukur kekuatan relatif pemain kompetitif, tetapi prinsip peringkatnya dapat diterapkan pada objek. Damiri dan Rosyid (2022), misalnya, menggunakan Elo-rating untuk memeringkat item berdasarkan hasil perbandingan antarobjek. Logika tersebut membuka kemungkinan memperlakukan butir soal sebagai entitas ber-rating yang dapat naik atau turun sesuai pola jawaban pengguna.

Dalam pembelajaran pemrograman, Vesin et al. (2022) menunjukkan bahwa modifikasi Elo dapat digunakan untuk memperkirakan kemahiran peserta didik dan kesulitan konten, kemudian memasang keduanya untuk menghasilkan rekomendasi yang lebih personal. Kinerja sistem Elo sangat dipengaruhi oleh

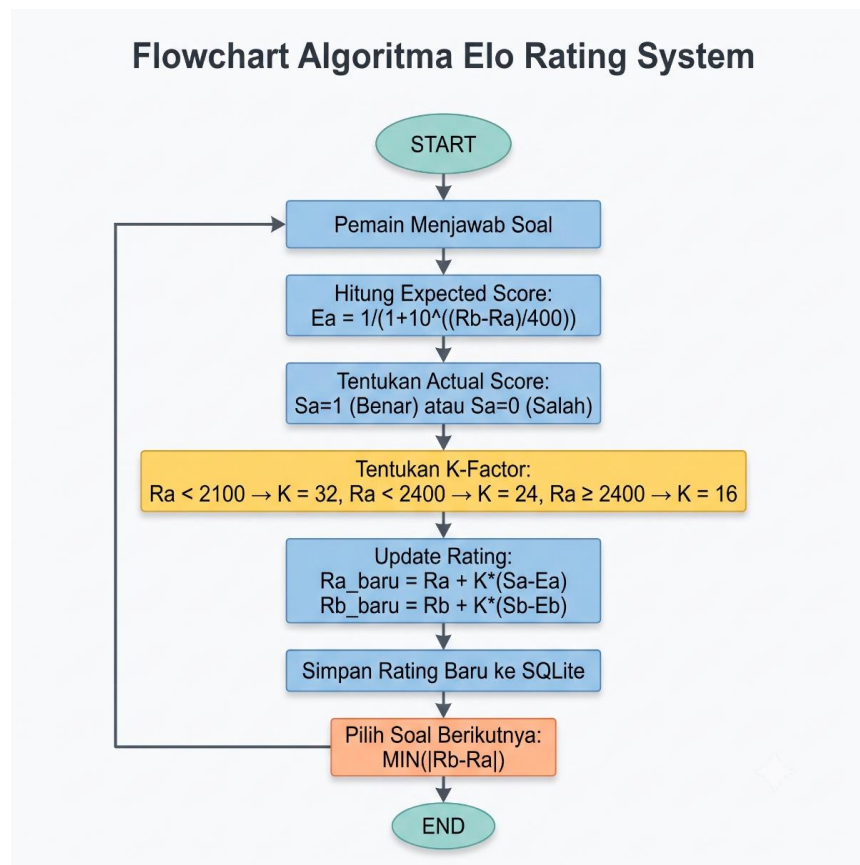
nilai K yang mengendalikan besar perubahan rating. Nilai K yang terlalu besar menghasilkan pembaruan cepat tetapi lebih volatil, sedangkan nilai yang terlalu kecil lebih stabil tetapi lambat mengikuti perubahan kemampuan. Vermeiren et al. (2026) menunjukkan bahwa penyesuaian K secara dinamis dapat membantu menyeimbangkan stabilitas dan fleksibilitas dalam lingkungan belajar adaptif. Temuan tersebut mendukung penggunaan K-Factor dinamis pada konteks latihan pemrograman yang kemampuan penggunaannya dapat berubah setelah beberapa respons.

Kontribusi penelitian ditempatkan pada tiga aspek. Pertama, Elo tidak digunakan hanya sebagai skor pemain, tetapi juga sebagai estimator dinamis untuk butir soal sehingga dua sisi interaksi diperbarui secara bersamaan. Kedua, pemilihan soal tidak bergantung pada kelas kesulitan diskrit seperti mudah, sedang, dan sulit, melainkan pada jarak rating yang berubah setelah setiap respons. Ketiga, mekanisme tersebut diintegrasikan ke aplikasi Android yang dapat bekerja dengan penyimpanan lokal dan sinkronisasi *cloud*. Kombinasi ini membedakan PyQuest dari pendekatan DDA yang mengubah parameter permainan umum, karena objek adaptasinya secara langsung adalah konten latihan pemrograman. Dengan demikian, perubahan rating tidak hanya berfungsi sebagai elemen gamifikasi, tetapi menjadi variabel operasional untuk menentukan soal berikutnya. Desain ini juga memberikan jejak numerik yang dapat ditelusuri karena setiap kenaikan atau penurunan rating berasal dari nilai ekspektasi, hasil aktual, dan K-Factor yang eksplisit.

Berdasarkan kajian tersebut, celah yang dijadikan fokus penelitian adalah masih terbatasnya implementasi Elo Rating System secara *real-time* sebagai mesin DDA pada *game* edukasi logika pemrograman Python berbasis Android dengan model pemain tunggal. Penelitian ini mengadaptasi hubungan pemain dan soal sebagai interaksi *zero-sum*: jawaban benar diperlakukan sebagai kemenangan pemain yang menaikkan rating pemain dan menurunkan rating soal, sedangkan jawaban salah menghasilkan pembaruan sebaliknya. Rating terbaru kemudian dipakai untuk mencari soal dengan perbedaan rating paling kecil melalui *Nearest Neighbor Search*. Pendekatan ini dirancang agar proses adaptasi tetap ringan, transparan, dan dapat dieksekusi langsung pada perangkat Android. Tujuan penelitian adalah merancang dan mengimplementasikan mekanisme tersebut dalam aplikasi PyQuest, memvalidasi fungsi inti melalui *Black-Box Testing*, serta mengevaluasi kegunaan antarmuka dengan System Usability Scale (SUS). Penelitian berfokus pada validitas implementasi dan kegunaan sistem, bukan pada pengujian efektivitas pedagogis dengan desain eksperimen terkontrol.

METODE PENELITIAN

Penelitian menggunakan pendekatan pengembangan perangkat lunak dengan Multimedia Development Life Cycle (MDLC) versi Luther yang dimodifikasi oleh Sutopo. MDLC dipilih karena aplikasi menggabungkan konten pembelajaran, elemen multimedia, mekanik permainan, dan algoritma adaptif. Tahapan yang digunakan terdiri atas *concept*, *design*, *material collecting*, *assembly*, *testing*, dan *distribution*. Penggunaan enam tahap tersebut sejalan dengan pengembangan *game* edukasi berbasis multimedia yang dilakukan Borman dan Purwanto (2019), yaitu dimulai dari penetapan konsep, dilanjutkan perancangan dan pengumpulan bahan, kemudian pembuatan, pengujian, serta distribusi. Pada tahap *concept*, ditetapkan bahwa produk berupa *game* kuis logika Python untuk mahasiswa atau pelajar pemula. Tujuan utama aplikasi adalah memberikan latihan materi dasar Python dengan tingkat kesulitan yang dapat menyesuaikan kemampuan pemain melalui Elo Rating System dan DDA.



Gambar 1. Alur Algoritma Elo Rating System pada Aplikasi PyQuest
 Sumber: Hasil penelitian, 2026

Tahap *design* mencakup perancangan alur sistem, basis data, dan antarmuka. Alur logika memetakan proses pemuatan rating pemain, pemilihan soal, validasi jawaban, perhitungan nilai ekspektasi, pembaruan rating, penyimpanan data, dan pemilihan soal berikutnya. Basis data dirancang dengan entitas pengguna, soal, rating, dan riwayat permainan. Pemain baru diberi rating awal 1.400. Setiap soal memiliki rating kesulitan awal yang kemudian berubah secara dinamis sesuai interaksi pengguna. Pada tahap *material collecting*, disusun bank 280 soal yang dibagi ke empat materi utama, yaitu variabel dan tipe data, operator, percabangan, serta perulangan. Aset visual, ikon, elemen suara, dan bahan ringkas materi juga dikumpulkan untuk mendukung pengalaman belajar.

Tahap *assembly* dilaksanakan menggunakan Flutter dengan bahasa Dart. SQLite digunakan untuk persistensi data lokal sehingga mode latihan tetap dapat berjalan tanpa koneksi internet, sedangkan Supabase digunakan untuk sinkronisasi profil serta papan peringkat global. Arsitektur aplikasi dibagi menjadi lapisan antarmuka, logika bisnis, dan data. Logika Elo ditempatkan pada layanan terpisah agar perhitungan dapat dipanggil setiap kali pemain menjawab soal tanpa membuat objek layanan baru. Aplikasi ditargetkan berjalan pada Android minimum API Level 26 atau Android 8.0 Oreo. Setelah fungsi inti stabil, aplikasi dikemas menjadi Android Package Kit (APK) pada tahap *distribution*.

Perhitungan Elo dimulai dari nilai ekspektasi pemain terhadap soal. Jika rating pemain dinyatakan sebagai Ra dan rating soal sebagai Rb , probabilitas pemain menjawab benar dihitung dengan

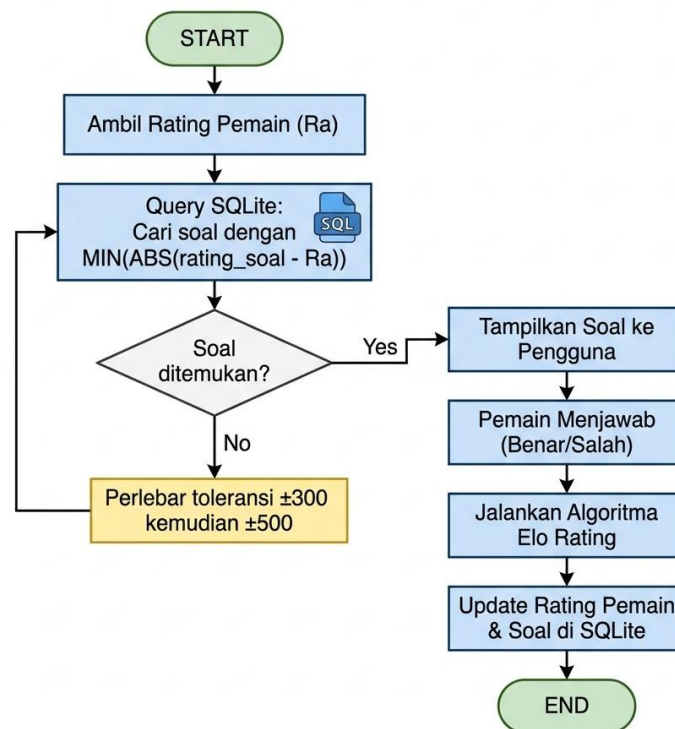
$$Ea = \frac{1}{1 + 10^{\frac{Rb - Ra}{400}}}$$

Nilai ekspektasi soal adalah $Eb = 1 - Ea$. Skor aktual pemain Sa bernilai 1 untuk jawaban benar dan 0 untuk jawaban salah, sedangkan skor aktual soal Sb merupakan kebalikannya. Pembaruan rating dihitung dengan $Ra' = Ra + K(Sa - Ea)$ dan $Rb' = Rb + K(Sb - Eb)$. K-Factor dibuat bertingkat, yaitu $K = 32$ untuk rating pemain di bawah 2.100, $K = 24$ untuk rating 2.100 hingga di bawah 2.400, dan $K = 16$ untuk rating minimal 2.400. Pembagian tersebut dimaksudkan agar rating pengguna pada tingkat awal dapat bergerak lebih responsif, sedangkan rating pengguna yang tinggi lebih stabil. Secara konseptual, pilihan K

dinamis ini mengikuti kebutuhan menyeimbangkan kecepatan adaptasi dan stabilitas rating sebagaimana dibahas Vermeiren et al. (2026).

Mekanisme pemilihan soal menggunakan *Nearest Neighbor Search* berdasarkan selisih absolut antara rating pemain dan rating soal. Soal yang telah muncul pada sesi berjalan dikeluarkan dari kandidat. Kandidat kemudian diurutkan berdasarkan $|R_b - R_a|$, dan sistem memilih secara acak satu soal dari tiga kandidat terdekat untuk mengurangi pola yang monoton. Pada lapisan SQLite, pencarian dilakukan dengan fungsi ABS() dan pengurutan berdasarkan selisih rating. Jika tidak ditemukan soal pada rentang terdekat, toleransi pencarian diperlebar secara bertahap. Setelah jawaban diberikan, rating pemain dan rating soal diperbarui, disimpan, lalu proses pencarian dijalankan kembali. Alur perhitungan Elo disajikan pada Gambar 1, sedangkan mekanisme DDA dan pencarian soal adaptif ditunjukkan pada Gambar 2.

Flowchart Mekanisme DDA PyQuest



Gambar 2. Mekanisme Dynamic Difficulty Adjustment pada PyQuest

Sumber: Hasil penelitian, 2026

Pengujian dilakukan dalam dua bagian. Pertama, *Black-Box Testing* digunakan untuk memeriksa fungsi dari sudut pandang pengguna tanpa menilai struktur kode internal. Lima belas skenario mencakup autentikasi, mode tamu, akses materi, pemilihan kuis adaptif, respons jawaban benar dan salah, kalkulasi Elo, pemilihan soal DDA, persistensi data, statistik, papan peringkat, hasil sesi, serta sinkronisasi *cloud*. Skenario dinilai valid apabila keluaran aktual sesuai dengan hasil yang diharapkan. Kedua, evaluasi kegunaan menggunakan System Usability Scale yang dikembangkan Brooke (1996). Instrumen SUS terdiri atas 10 pernyataan dengan skala Likert 1 sampai 5. Untuk butir ganjil, kontribusi dihitung dari skor dikurangi 1, sedangkan butir genap dihitung dari 5 dikurangi skor. Jumlah kontribusi kemudian dikalikan 2,5 sehingga menghasilkan skor 0 sampai 100. Interpretasi tingkat kegunaan merujuk pada kajian Bangor et al. (2008).

Evaluasi SUS melibatkan 20 mahasiswa semester 1 dan 2 Program Studi Teknologi Informasi Universitas Muhammadiyah Sumatera Utara yang dipilih secara *purposive sampling* dan belum pernah menggunakan PyQuest sebelumnya. Setiap responden diminta mengeksplorasi aplikasi secara mandiri selama 15 menit, menyelesaikan sedikitnya satu sesi kuis penuh, kemudian mengisi sepuluh pernyataan SUS. Analisis bersifat deskriptif dengan menghitung rata-rata setiap pernyataan, kontribusi SUS, jumlah kontribusi, dan skor

akhir. Selain itu, masukan terbuka mengenai fitur yang diinginkan dicatat sebagai bahan pengembangan. Desain penelitian tidak menggunakan kelompok kontrol, *pre-test*, atau *post-test*, sehingga hasil tidak digunakan untuk menyimpulkan peningkatan hasil belajar Python secara kausal.

Selain validasi fungsi dan usability, data implementasi dianalisis melalui dua skenario numerik. Skenario pertama membandingkan hasil perhitungan manual dengan keluaran formula Elo untuk memastikan tidak terjadi kesalahan pada implementasi matematis. Skenario kedua menelusuri enam soal berurutan untuk melihat hubungan antara rating pemain, rating soal terpilih, hasil jawaban, dan rating baru. Analisis tersebut tidak diperlakukan sebagai uji inferensial, melainkan sebagai verifikasi perilaku algoritma. Kriteria yang diharapkan adalah perubahan rating mengikuti arah benar-salah dan soal yang dipilih tetap berada sedekat mungkin dengan rating pemain. Dengan cara ini, evaluasi sistem mencakup tiga lapisan, yaitu kebenaran komputasi, validitas fungsi aplikasi, dan persepsi kegunaan pengguna.

HASIL DAN PEMBAHASAN

Implementasi Elo Rating System dan DDA

Implementasi ini ditempatkan pada kelas EloService. Fungsi pertama menghitung *expected score*, fungsi kedua menentukan K-Factor dinamis, dan fungsi ketiga memperbarui rating pemain serta rating soal setelah setiap jawaban. Selain rating numerik, aplikasi mengelompokkan capaian ke dalam lima tier, yaitu Intern untuk rating di bawah 1.200, Junior Dev pada 1.200 sampai 1.399, Mid-Level Dev pada 1.400 sampai 1.599, Senior Dev pada 1.600 sampai 1.799, dan Tech Lead pada rating minimal 1.800. Tier berfungsi sebagai elemen gamifikasi dan tidak mengubah formula dasar Elo.

Akurasi implementasi diverifikasi melalui perhitungan manual dengan pemain ber-rating 1.400 yang menjawab benar soal ber-rating 1.500. Karena rating pemain berada di bawah 2.100, K-Factor bernilai 32. Tabel 1 memperlihatkan bahwa nilai ekspektasi pemain sebesar 0,3600 menghasilkan kenaikan rating 20,48 poin, sedangkan rating soal berkurang dalam besaran yang sama. Hasil ini menunjukkan sifat *zero-sum* pada pembaruan pasangan pemain-soal dan menjadi dasar untuk memastikan rumus dalam Dart sesuai dengan perhitungan matematis.

Setelah rating diperbarui, mekanisme DDA memilih soal berikutnya berdasarkan selisih rating. Simulasi enam soal pertama pada Tabel 2 menunjukkan bahwa rating soal terpilih selalu berada dekat dengan rating pemain. Ketika pemain menjawab benar, rating pemain bergerak naik sehingga sistem cenderung memilih soal yang lebih tinggi. Ketika pemain menjawab salah, rating turun dan soal berikutnya bergerak ke tingkat yang lebih mudah. Pola tersebut merupakan fungsi adaptif utama PyQuest dan tidak bergantung pada urutan level yang telah ditentukan sebelumnya.

Tabel 1. Verifikasi Perhitungan Rating Elo pada Pemain Menang terhadap Soal Lebih Sulit

No.	Parameter	Perhitungan/Rumus	Hasil
1	K-Factor	Rating pemain < 2.100	K = 32
2	Expected Score pemain (Ea)	$1 / (1 + 10^{((1500-1400)/400)})$	0,3600
3	Expected Score soal (Eb)	1 - 0,3600	0,6400
4	Rating baru pemain	$1400 + 32 \times (1 - 0,3600)$	1.420,48
5	Rating baru soal	$1500 + 32 \times (0 - 0,6400)$	1.479,52
6	Delta perubahan	Rating baru - rating awal	+20,48 pemain; -20,48 soal

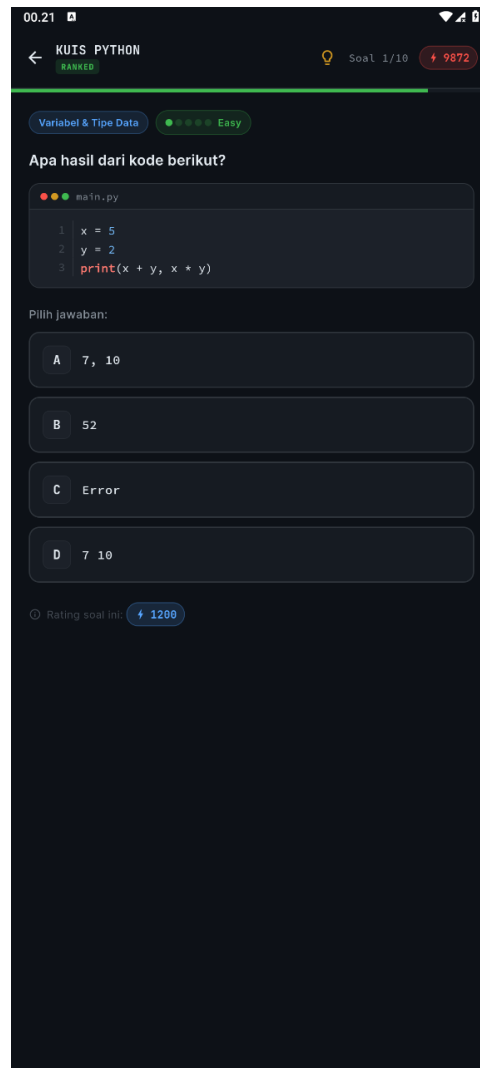
Sumber: Hasil penelitian, 2026

Tabel 2. Simulasi DDA Selama Enam Soal Pertama

Soal	Rating Pemain	Soal Terpilih (Rating)	Kategori	Jawaban	Rating Baru
1	1400,00	#33 (1412)	If-Else	Benar	1416,55
2	1416,55	#71 (1425)	Operator	Salah	1400,94
3	1400,94	#28 (1408)	Variabel	Benar	1417,27
4	1417,27	#95 (1440)	If-Else	Benar	1434,31
5	1434,31	#112 (1460)	Looping	Salah	1419,49
6	1419,49	#84 (1445)	Operator	Benar	1436,67

Sumber: Hasil penelitian, 2026

Implementasi Antarmuka dan Fitur PyQuest



Gambar 3. Halaman Gameplay dengan Soal Kode Python dan Syntax Highlighting
 Sumber: Hasil penelitian, 2026

PyQuest dibangun dengan tema gelap yang terinspirasi antarmuka *code editor* agar lingkungan latihan konsisten dengan konteks pemrograman. Halaman utama menyediakan akses ke materi, kuis adaptif, latihan bebas, profil, papan peringkat, dan pengaturan. Halaman materi menampilkan ringkasan empat topik Python, sedangkan halaman kuis menampilkan kategori, indikator kesulitan, nomor soal, potongan kode dengan *syntax highlighting*, pilihan jawaban, dan informasi rating soal. Contoh halaman *gameplay* disajikan pada Gambar 3. Pada mode *ranked*, jawaban memicu perubahan Elo dan soal berikutnya dipilih oleh DDA. Pada latihan bebas, perubahan Elo dinonaktifkan agar pengguna dapat berlatih tanpa risiko penurunan rating.

Aplikasi juga menyediakan variasi mode permainan untuk mempertahankan keterlibatan. Beberapa mode tantangan dibuka berdasarkan capaian rating sehingga berfungsi sebagai target jangka panjang. Profil menyajikan rating aktif, tier, jumlah jawaban benar dan salah, *win rate*, rating per kategori, serta riwayat perkembangan Elo. Papan peringkat memanfaatkan Supabase dan mengurutkan pengguna berdasarkan rating secara menurun. Arsitektur lokal dan *cloud* memungkinkan mode tamu tetap berjalan dengan SQLite, sementara akun terdaftar dapat menyinkronkan progres ketika koneksi tersedia.

Hasil Pengujian Fungsional

Pengujian fungsional dijalankan langsung pada perangkat Android nyata. Seluruh skenario difokuskan pada keterhubungan fungsi antarmuka, basis data, algoritma Elo, dan seleksi DDA. Ringkasan hasil pengujian ditampilkan hanya dalam bentuk tabel pada Tabel 3.

Tabel 3. Hasil Black-Box Testing pada 15 Skenario Uji

No.	Skenario Uji	Input/Aksi	Hasil yang Diharapkan	Status
1	Splash Screen	Buka aplikasi	Animasi muncul dan aplikasi tidak crash	Valid
2	Registrasi akun baru	Masukkan nama dan e-mail baru	Akun tersimpan di Supabase dan masuk Home	Valid
3	Login akun lama	Masukkan e-mail terdaftar	Data lama dimuat dan masuk Home Screen	Valid
4	Mode tamu offline	Pilih lanjut sebagai tamu	Dapat bermain dan data tersimpan lokal di SQLite	Valid
5	Akses halaman materi	Tekan Materi Python	Empat tab tampil dan kode Python di-highlight	Valid
6	Mulai kuis adaptif	Tekan Mulai Kuis dan pilih kategori	Soal dengan rating terdekat rating pemain dipilih	Valid
7	Menjawab benar	Klik opsi jawaban benar	Panel hijau tampil, Elo naik, dan soal baru dimuat	Valid
8	Menjawab salah	Klik opsi jawaban salah	Animasi shake, panel merah, dan Elo turun	Valid
9	Kalkulasi rating Elo	Jawab pada berbagai kondisi	$Ra = Ra + K(S-E)$, sesuai verifikasi manual	Valid
10	Seleksi soal DDA	Bermain 10 soal berturut-turut	Rating soal selalu mendekati rating pemain terkini	Valid
11	Persistensi data	Tutup paksa lalu buka ulang	Rating dan riwayat tetap tersimpan	Valid
12	Profil dan statistik	Buka halaman Profil	Win rate, rating per kategori, dan grafik Elo akurat	Valid
13	Leaderboard	Buka leaderboard saat online	Daftar pemain terurut berdasarkan Elo dari Supabase	Valid
14	Hasil sesi	Akhiri sesi bermain	Ringkasan sesi dan perubahan Elo benar	Valid
15	Sinkronisasi cloud	Bermain offline lalu sambungkan internet	Data lokal tersinkron ke Supabase otomatis	Valid

Sumber: Hasil penelitian, 2026

Hasil Pengujian Kegunaan

Pengujian kegunaan dilakukan setelah responden menggunakan aplikasi sesuai prosedur. Rata-rata skor setiap item, kontribusi SUS, jumlah kontribusi, dan skor akhir dirangkum pada Tabel 4. Skor akhir 81,3 berada di atas nilai acuan 68 yang umum digunakan sebagai titik pembandingan pada interpretasi SUS dan dalam pengelompokan penelitian ini termasuk kategori *Excellent* dengan Grade A. Hasil ini digunakan sebagai indikator kegunaan antarmuka, bukan sebagai ukuran langsung efektivitas pembelajaran.

Tabel 4. Rekapitulasi Hasil System Usability Scale pada 20 Responden

Pernyataan	Jenis	Rata-Rata Skor	Kontribusi SUS
P1	Positif (ganjil)	4,2	3,2
P2	Negatif (genap)	1,9	3,1
P3	Positif (ganjil)	4,4	3,4
P4	Negatif (genap)	1,8	3,2
P5	Positif (ganjil)	4,4	3,4
P6	Negatif (genap)	1,8	3,2
P7	Positif (ganjil)	4,6	3,6
P8	Negatif (genap)	1,7	3,3
P9	Positif (ganjil)	4,2	3,2
P10	Negatif (genap)	2,1	2,9
Jumlah kontribusi			32,5
Skor SUS akhir (x2,5)			81,3

Sumber: Hasil penelitian, 2026

Pembahasan

1. Adaptasi Elo sebagai Mesin Penyesuaian Kesulitan

Hasil implementasi memperlihatkan bahwa Elo Rating System dapat dipindahkan dari konteks kompetisi pemain ke konteks pemain melawan butir soal. Adaptasi tersebut dimungkinkan karena Elo pada dasarnya memerlukan dua entitas ber-rating, peluang ekspektasi, hasil aktual, dan aturan pembaruan. Damiri dan Rosyid (2022) telah menunjukkan bahwa entitas nonmanusia dapat diperingkatkan melalui hasil perbandingan, sedangkan Vesin et al. (2022) menunjukkan bahwa rating siswa dan tingkat kesulitan konten dapat dipasangkan untuk menghasilkan rekomendasi adaptif pada pembelajaran pemrograman. PyQuest menggabungkan dua gagasan tersebut dengan memperlakukan soal sebagai lawan yang tingkat kesulitannya dikoreksi oleh respons pengguna.

Karakter *self-correcting* muncul ketika rating soal tidak dianggap sebagai label kesulitan yang permanen. Soal yang berulang kali dapat dijawab oleh pengguna dengan rating lebih rendah akan kehilangan poin sehingga secara bertahap bergeser ke rentang yang lebih mudah. Sebaliknya, soal yang sering dijawab salah memperoleh rating lebih tinggi. Mekanisme tersebut mengurangi ketergantungan pada penilaian awal pakar, karena rating awal hanya berfungsi sebagai titik mulai. Namun, kualitas adaptasi tetap bergantung pada jumlah dan keragaman interaksi. Pada fase awal ketika data masih sedikit, kesalahan rating awal masih dapat memengaruhi pilihan soal. Oleh karena itu, K-Factor dinamis menjadi penting untuk mempercepat konvergensi pengguna baru tanpa membuat rating pemain mahir terlalu volatil. Prinsip ini konsisten dengan temuan Vermeiren et al. (2026) mengenai perlunya keseimbangan antara stabilitas dan fleksibilitas dalam pembaruan Elo pada lingkungan belajar adaptif.

2. Keseimbangan Tantangan, DDA, dan Flow

Simulasi DDA menunjukkan selisih rating pemain dan soal berada dalam rentang yang relatif dekat pada enam interaksi awal. Secara desain, kondisi tersebut ditujukan untuk mempertahankan keseimbangan antara tantangan dan kemampuan sebagaimana dijelaskan oleh teori *flow* (Csikszentmihalyi, 1990). Jika pengguna menjawab benar secara berurutan, rating meningkat dan kandidat soal berikutnya menjadi lebih menantang. Jika pengguna menjawab salah, rating menurun sehingga sistem menyediakan jalur pemulihan melalui soal yang lebih sesuai. Pendekatan ini berbeda dari kurva statis karena perpindahan kesulitan ditentukan oleh data performa, bukan oleh nomor level.

Arah hasil tersebut sejalan dengan penelitian DDA pada *game* edukasi. Nugraha et al. (2022) menunjukkan pentingnya dinamika kesulitan untuk menyeimbangkan aspek belajar dan kesenangan. Soedargo dan Junaedi (2022) memperlihatkan bahwa penyesuaian berbasis performa diperlukan karena keterampilan pemain tidak homogen. Krisdiawan et al. (2023) menggunakan DDA pada *mobile game* untuk mengakomodasi kebutuhan pengguna dengan kecepatan belajar berbeda, sedangkan Sany dan Angkasa (2025) melaporkan bahwa DDA berbasis pembelajaran penguatan dapat mempertahankan pengalaman pada zona *flow*. PyQuest menggunakan model yang lebih sederhana secara komputasi, yaitu selisih rating dan pembaruan Elo, sehingga lebih sesuai untuk pemilihan soal pilihan ganda yang harus dilakukan berulang kali pada perangkat bergerak.

Dalam pengujian, 18 dari 20 responden menyelesaikan sesi secara sukarela tanpa melaporkan frustrasi atau kebosanan berlebihan. Data tersebut dapat dipandang sebagai indikasi observasional bahwa mekanisme tidak menciptakan hambatan besar selama sesi, tetapi belum cukup untuk membuktikan tercapainya *flow state* secara psikometrik. Penelitian tidak menggunakan instrumen *flow* tervalidasi, sehingga klaim utama tetap dibatasi pada fungsi adaptasi sistem. Pembatasan ini penting agar hasil tidak ditafsirkan melebihi desain penelitian yang hanya mengevaluasi fungsionalitas dan kegunaan.

3. Kegunaan Antarmuka dan Relevansi Mobile Game Edukasi

Skor SUS 81,3 menunjukkan bahwa antarmuka PyQuest dapat dipelajari dan digunakan dengan relatif mudah oleh responden pemula. Nilai tertinggi terdapat pada pernyataan bahwa kebanyakan orang akan cepat mempelajari penggunaan aplikasi, sedangkan skor rendah pada pernyataan negatif tentang kerumitan mendukung interpretasi bahwa struktur navigasi cukup jelas. Sebagai pembanding, Christina dan Wibowo (2023) melaporkan skor SUS 78,41 pada media pembelajaran berbasis gamifikasi, sedangkan Ibrahim et al.

(2023) memperoleh skor SUS 77,53 pada evaluasi aplikasi OVO. Perbandingan lintas domain tidak digunakan untuk menentukan superioritas, tetapi menunjukkan bahwa nilai PyQuest berada pada rentang kegunaan yang kuat dalam konteks metode SUS.

Penggunaan 20 responden juga memiliki preseden dalam evaluasi aplikasi Android. Anra et al. (2024) melibatkan 20 pengguna untuk mengevaluasi dan merancang ulang UI/UX iKalbar menggunakan SUS bersama teknik lain, serta melaporkan peningkatan kepuasan ke Grade A setelah perancangan ulang. Temuan tersebut memperkuat kegunaan SUS sebagai instrumen ringkas untuk melihat persepsi pengguna terhadap antarmuka, meskipun SUS tidak menggantikan pengukuran efektivitas belajar. Dengan demikian, nilai 81,3 pada PyQuest lebih tepat dibaca sebagai bukti bahwa pengguna dapat berinteraksi dengan aplikasi tanpa hambatan usability yang dominan.

Dari perspektif media belajar bergerak, hasil ini juga relevan dengan temuan bahwa *game* kuis dapat menjadi alternatif interaktif dibanding pembelajaran konvensional yang monoton (Suryadila et al., 2025). Pengembangan Flutter pada penelitian Lionza et al. (2026) menunjukkan bahwa kerangka kerja tersebut dapat digunakan untuk *game* edukasi Android dengan penerimaan pengguna yang baik. Pada PyQuest, Flutter mendukung konsistensi komponen antarmuka, sementara SQLite dan Supabase memberi fleksibilitas antara penggunaan luring dan sinkronisasi daring. Kombinasi ini mendukung tujuan sistem yang tidak hanya adaptif dalam kesulitan, tetapi juga praktis digunakan pada kondisi konektivitas yang beragam.

4. Keterbatasan dan Implikasi Pengembangan

Penelitian memiliki beberapa keterbatasan. Pertama, bank soal masih berfokus pada empat topik dasar dan seluruh soal berbentuk pilihan ganda atau potongan kode sederhana. Pengukuran kemampuan pemrograman karena itu belum mencakup aktivitas menulis, menjalankan, dan memperbaiki kode secara langsung. Kedua, skor aktual Elo hanya membedakan jawaban benar dan salah sehingga waktu respons belum masuk ke dalam pembaruan. Padahal, kecepatan dapat menjadi sinyal tambahan untuk membedakan jawaban benar yang mantap dengan jawaban benar yang diperoleh setelah waktu sangat lama. Ketiga, pengujian SUS menggunakan sampel purposif 20 mahasiswa dari satu program studi sehingga generalisasi ke populasi lain harus dilakukan secara hati-hati.

Keempat, penelitian tidak membandingkan PyQuest dengan pembelajaran konvensional atau versi permainan tanpa DDA. Karena itu, hasil 100% pada *Black-Box Testing* hanya menunjukkan kesesuaian fungsi dengan spesifikasi, sedangkan skor SUS menunjukkan persepsi kegunaan. Keduanya tidak dapat dipakai untuk menyatakan bahwa PyQuest meningkatkan hasil belajar. Penelitian lanjutan perlu menggunakan desain kuasi eksperimen dengan kelompok pembandingan, *pre-test* dan *post-test*, serta instrumen keterlibatan atau *flow* yang tervalidasi. Kelima, mekanisme DDA dapat dikembangkan dengan memasukkan waktu pengerjaan, tingkat keyakinan, histori per topik, atau parameter multidimensi agar estimasi kemampuan lebih kaya. Kajian Fisher dan Kulshreshth (2025) serta Vitéková et al. (2026) menunjukkan bahwa DDA dapat memanfaatkan berbagai sinyal performa, sehingga pengembangan selanjutnya dapat mempertahankan kesederhanaan Elo sambil menambahkan informasi yang benar-benar relevan dengan proses belajar.

SIMPULAN

PyQuest berhasil dikembangkan sebagai *game* edukasi Android untuk latihan logika pemrograman Python dengan Elo Rating System sebagai penggerak Dynamic Difficulty Adjustment. Interaksi pemain dan soal dimodelkan secara *zero-sum*, lalu kemampuan pemain dan kesulitan soal diperbarui menggunakan *expected score*, skor aktual, dan K-Factor dinamis. Rating terbaru digunakan oleh *Nearest Neighbor Search* untuk memilih soal yang paling dekat dengan kemampuan pemain dari bank 280 soal pada empat kategori. Verifikasi manual menunjukkan implementasi perhitungan Dart konsisten dengan rumus matematis, sedangkan simulasi sesi menunjukkan soal terpilih mengikuti perubahan rating pemain. *Black-Box Testing* pada 15 skenario menghasilkan 100% status valid, dan pengujian SUS terhadap 20 mahasiswa menghasilkan skor 81,3 yang termasuk kategori *Excellent* dalam pengelompokan penelitian. Hasil tersebut menunjukkan bahwa Elo dapat diterapkan secara fungsional sebagai mekanisme adaptasi kesulitan yang ringan pada aplikasi Android dan bahwa antarmuka PyQuest memiliki tingkat kegunaan yang tinggi. Penelitian ini belum menguji

peningkatan hasil belajar secara kausal, sehingga evaluasi pedagogis dengan kelompok kontrol, pengukuran *pre-test* dan *post-test*, serta instrumen *flow* tervalidasi diperlukan pada penelitian berikutnya.

DAFTAR PUSTAKA

- Abt, C. C. (1970). *Serious Games*. Viking Press.
- Anra, H., Sari, K. S., & Perwitasari, A. (2024). Evaluasi dan Perancangan Ulang UI/UX Aplikasi iKalbar Menggunakan Metode Double Diamond. *JEPIN (Jurnal Edukasi Dan Penelitian Informatika)*, 10(2), 215–225. <https://doi.org/10.26418/jp.v10i2.73697>
- Bangor, A., Kortum, P. T., & Miller, J. T. (2008). An Empirical Evaluation of the System Usability Scale. *International Journal of Human-Computer Interaction*, 24(6), 574–594. <https://doi.org/10.1080/10447310802205776>
- Borman, R. I., & Purwanto, Y. (2019). Impelementasi Multimedia Development Life Cycle pada Pengembangan Game Edukasi Pengenalan Bahaya Sampah pada Anak. *JEPIN (Jurnal Edukasi Dan Penelitian Informatika)*, 5(2), 119–124. <https://doi.org/10.26418/jp.v5i2.25997>
- Brooke, J. (1996). SUS: A Quick and Dirty Usability Scale. In P. W. Jordan, B. Thomas, B. A. Weerdmeester, & I. L. McClelland (Eds.), *Usability Evaluation in Industry* (pp. 189–194). Taylor & Francis.
- Christina, E., & Wibowo, M. (2023). Media Pembelajaran Perilaku Hidup Bersih dan Sehat Menggunakan Metode Gamifikasi Berbasis Website. *Edumatic: Jurnal Pendidikan Informatika*, 7(1), 131–140. <https://doi.org/10.29408/edumatic.v7i1.17463>
- Csikszentmihalyi, M. (1990). *Flow: The Psychology of Optimal Experience*. Harper & Row.
- Damiri, W. G., & Rosyid, H. (2022). Implementasi Algoritma Elo-Rating untuk Memprediksi Perilaku E-Commerce Item (Studi Kasus pada Olahan Desa Giri Kecamatan Kebomas Kabupaten Gresik). *Indexia: Informatics and Computational Intelligent Journal*, 4(1), 82–91. <https://doi.org/10.30587/indexia.v4i1.3777>
- Fisher, N., & Kulshreshth, A. K. (2025). Dynamic Difficulty Adjustment in Games: Concepts, Techniques, and Applications. In T. Guzsvinecz (Ed.), *From Pixels to Play: The Art and Science of Video Games*. IntechOpen. <https://doi.org/10.5772/intechopen.1011703>
- Ibrahim, A., Alexander, O., Tania, K. D., Putra, P., & Meiriza, A. (2023). Assessing User Experience and Usability in the OVO Application: Utilizing the User Experience Questionnaire and System Usability Scale for Evaluation. *Jurnal RESTI (Rekayasa Sistem Dan Teknologi Informasi)*, 7(4), 953–963. <https://doi.org/10.29207/resti.v7i4.5137>
- Krisdiawan, R. A., Sugiharto, T., & Nugraha, N. (2023). Terapi Rehabilitasi Dyslexia Berbasis Mobile Game dengan Dynamic Difficulty Adjustment (DDA) Menggunakan Algoritma Fuzzy Sugeno. *Jurnal Teknologi Informatika Dan Komputer*, 9(2), 975–991. <https://doi.org/10.37012/jtik.v9i2.1837>
- Lionza, R., Andriyanto, S., & Farma, T. A. (2026). Implementasi Game Edukasi Flutter “Bubble Math” dalam Meningkatkan Pemahaman Operasi Aritmatika Siswa Kelas V. *KHARISMA Tech*, 21(1), 140–153. <https://doi.org/10.55645/kharismatech.v21i1.643>
- Nugraha, M. R., Rosmansyah, Y., Mulyani, E. D. S., Mufizar, T., Anwar, D. S., & Hidayat, C. R. (2022). A Serious Game for Improving Reading Skill of Preschoolers. *Elinvo (Electronics, Informatics, and Vocational Education)*, 7(1), 84–91. <https://doi.org/10.21831/elinvo.v7i1.47912>
- Plass, J. L., & Pawar, S. (2020). Toward a Taxonomy of Adaptivity for Learning. *Journal of Research on Technology in Education*, 52(3), 275–300. <https://doi.org/10.1080/15391523.2020.1719943>
- Prensky, M. (2001). *Digital Game-Based Learning*. McGraw-Hill.
- Sany, D. S., & Angkasa, M. M. (2025). Dynamic Difficulty Adjustment Using Reinforcement Learning for Adaptive Gameplay Experience in Resik. *Media Jurnal Informatika*, 17(2), 425–435. <https://doi.org/10.35194/mji.v17i2.6106>
- Soedargo, D. S. O., & Junaedi, H. (2022). Dynamic Difficulty Adjustment Berbasis Logika Fuzzy untuk Procedural Content Generation pada Permainan Roguelike. *Teknika*, 11(2), 98–105. <https://doi.org/10.34148/teknika.v11i2.468>
- Suryadila, L., Ismanto, E., Novalia, M., & Syahfutra, W. (2025). Syshunt: Game Quiz Mobile untuk Pengenalan Perangkat Keras Komputer Menggunakan Successive Approximation Model. *Edumatic*:

- Jurnal Pendidikan Informatika*, 9(1), 314–323. <https://doi.org/10.29408/edumatic.v9i1.29773>
- Vermeiren, H., Hofman, A. D., Bolsinova, M., van der Maas, H. L. J., & Van Den Noortgate, W. (2026). Balancing Stability and Flexibility: Investigating a Dynamic K Value Approach for the Elo Rating System in Adaptive Learning Environments. *User Modeling and User-Adapted Interaction*, 36(1). <https://doi.org/10.1007/s11257-025-09439-z>
- Vesin, B., Mangaroska, K., Akhuseyinoglu, K., & Giannakos, M. (2022). Adaptive Assessment and Content Recommendation in Online Programming Courses: On the Use of Elo-Rating. *ACM Transactions on Computing Education*, 22(3), 1–27. <https://doi.org/10.1145/3511886>
- Víteková, L., Eichhorn, C., Pirker, J., & Plecher, D. A. (2026). Dynamic Difficulty Adjustment in Serious Games: A Literature Review. *Information*, 17(1). <https://doi.org/10.3390/info17010096>