PERANCANGAN SISTEM OTOMATISASI EVALUASI *SOURCE CODE* MENGGUNAKAN *DEEPSEEK* DAN *LANGCHAIN* BERBASIS *WEBSITE* PADA MATA KULIAH PEMROGRAMAN

TUGAS AKHIR

VINCENT DWI HARTANTO

312110017

PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS TEKNOLOGI DAN DESAIN
UNIVERSITAS MA CHUNG
MALANG

2025

LEMBAR PENGESAHAN TUGAS AKHIR

PERANCANGAN SISTEM OTOMATISASI EVALUASI *SOURCE CODE* MENGGUNAKAN *DEEPSEEK* DAN *LANGCHAIN* BERBASIS *WEBSITE* PADA MATA KULIAH PEMROGRAMAN

Oleh:

VINCENT DWI HARTANTO

NIM. 312110017

dari:

PROGRAM STUDI TEKNIK INFORMATIKA

FAKULTAS TEKNOLOGI DAN DESAIN

UNIVERSITAS MA CHUNG

Telah dinyatakan lulus dalam melaksanakan Tugas Akhir sebagai syarat kelulusan dan berhak mendapatkan gelar Sarjana S.Kom.

Dosen Pembimbing 1,	Dosen Pembimbing 2,
Windra Swastika, S.Kom., MT., Ph.D.	**Paulus Lucky Tirma Irawan, S.Kom., MT**
NIP. 20070039	**NIP. 20100006**

Dekan Fakultas Sains dan Teknologi

Prof. Dr.Eng. Romy Budhi, ST., MT., M.Pd.

NIP. 20070035

PERNYATAAN KEASLIAN TUGAS AKHIR

Dengan ini saya menyatakan bahwa isi sebagian maupun keseluruhan dari Tugas Akhir saya dengan judul "Perancangan Sistem Otomatisasi Evaluasi *Source Code* Menggunakan *Deepseek* dan *Langchain* Berbasis *Website* pada Mata Kuliah Pemrograman" adalah benar benar hasil karya intelektual mandiri tanpa menggunakan bahan-bahan yang tidak diizinkan dan bukan merupakan karya pihak lain yang saya akui sebagai karya sendiri.

Semua referensi yang dikutip maupun dirujuk telah ditulis secara lengkap pada daftar pustaka. Apabila ternyata pernyataan ini tidak benar, saya bersedia menerima sanksi sesuai peraturan yang berlaku.

Malang, 31 Juli 2025

Vincent Dwi Hartanto

PERANCANGAN SISTEM OTOMATISASI EVALUASI *SOURCE CODE* MENGGUNAKAN *DEEPSEEK* DAN *LANGCHAIN* BERBASIS *WEBSITE* PADA MATA KULIAH PEMROGRAMAN

Vincent Dwi Hartanto[1], Windra Swastika[2], Paulus Lucky Tirma Irawan[3]
Teknik Informatika, Universitas Ma Chung

Abstrak

Proses evaluasi kode program dalam mata kuliah pemrograman sering kali memerlukan waktu yang lama dan hasil evaluasi terkadang bersifat subjektif, terutama jika dilakukan secara manual oleh pengajar. Untuk mengatasi permasalahan tersebut, penelitian ini bertujuan untuk merancang dan mengimplementasikan sebuah sistem evaluasi kode otomatis berbasis *website* menggunakan *Artificial Intelligence* (*AI*). Sistem ini dikembangkan menggunakan *framework Next.js* untuk antarmuka pengguna dan diintegrasikan dengan *Large Language Model* (*LLM*) *DeepSeek-R1* melalui *LangChain* sebagai perantara komunikasi. Melalui integrasi ini, sistem mampu mengevaluasi jawaban dalam bentuk *source code* dari mahasiswa secara otomatis dan memberikan umpan balik dalam waktu singkat. Hasil evaluasi ditampilkan secara *realtime* baik kepada mahasiswa maupun mentor. Hasil pengujian menunjukkan rata-rata *response time* sebesar 5,7 detik, *throughput* sebesar 10,86 *request per second*, skor performa *load time* sebesar 78, serta tingkat kepuasan pengguna berdasarkan *System Usability Scale* (SUS) mencapai skor 77. Sistem terbukti berjalan dengan baik dan memberikan kontribusi dalam mempercepat serta mempermudah proses evaluasi pembelajaran pemrograman.

Kata kunci: evaluasi kode*, Artificial Intelligence* (AI), *Large Language Model* (LLM*)*, *Next.js*, *DeepSeek-R1*, *LangChain*

DESIGN OF A WEB-BASED AUTOMATED CODE EVALUATION SYSTEM USING DEEPSEEK AND LANGCHAIN FOR PROGRAMMING COURSES

Vincent Dwi Hartanto[1], Windra Swastika[2], Paulus Lucky Tirma Irawan[3]
Informatics Engineering, Universitas Ma Chung

Abstract

The evaluation of programming code in computer science courses often requires a significant amount of time and can be subjective, particularly when conducted manually by instructors. To address this issue, this study aims to design and implement an automated code evaluation system based on a web platform utilizing Artificial Intelligence (AI). The system is developed using the Next.js framework for the user interface and integrated with the Large Language Model (LLM) DeepSeek-R1 via LangChain as the communication intermediary. Through this integration, the system is capable of automatically evaluating student-submitted source code and providing prompt feedback. The evaluation results are displayed in real-time to both students and mentors. The results showed an average response time of 5.7 seconds, a throughput of 10.86 requests per second, a load time performance score of 78, and a System Usability Scale (SUS) score of 77. The system has proven to function effectively and contributes to accelerating and simplifying the evaluation process in programming education.

Keywords: *code evaluation, Artificial Intelligence* (*AI*), *Large Language Model* (*LLM)*, *Next.js*, *DeepSeek-R1*, *LangChain*

Kata Pengantar

Puji syukur dipanjatkan kehadirat Tuhan Yang Maha Esa karena atas rahmat dan restu-Nya sehingga tugas akhir dengan judul "Perancangan Sistem Otomatisasi Evaluasi *Source Code* Menggunakan *Deepseek* dan *Langchain* Berbasis *Website* Pada Mata Kuliah Pemrograman" ini dapat diselesaikan dengan baik. Laporan ini disusun untuk menjelaskan hasil pengerjaan Tugas Akhir yang telah selesai dilaksanakan. Tugas Akhir merupakan salah satu mata kuliah wajib bagi mahasiswa dari Prodi Teknik Informatika Universitas Ma Chung Malang sebagai salah satu prasyarat kelulusan.

Pada kesempatan ini, penulis menyampaikan ucapan terima kasih sebesar-besarnya kepada seluruh pihak yang telah memberikan bantuan dan dukungan kepada penulis dalam proses pengerjaan Tugas Akhir hingga selesai. Ucapan terima kasih disampaikan kepada seluruh pihak terkait yang telah membantu, mendukung, dan membimbing kegiatan Tugas Akhir hingga selesai, di antaranya :

1. Bapak Dr. Eng. Romy Budhi, ST, M.T. selaku Dekan Fakultas Teknologi dan Desain Universitas Ma Chung,
2. Bapak Hendry Setiawan, ST, M.Kom, selaku Kepala Program Studi Teknik Informatika,
3. Bapak Windra Swastika, S.Kom., MT., Ph.D. selaku Dosen Pembimbing 1 Tugas Akhir,
4. Bapak Paulus Lucky Tirma Irawan, S.Kom., MT. selaku Dosen Pembimbing 2 Tugas Akhir,
5. Keluarga dan teman-teman terkasih yang telah memberikan dukungan dan semangat selama pengerjaan Tugas Akhir ini,

Laporan ini disusun berdasarkan hasil penelitian di PT. XYZ selama enam bulan dengan judul "Perancangan Sistem Otomatisasi Evaluasi *Source Code* Menggunakan *Deepseek* dan *Langchain* Berbasis *Website* Pada Mata Kuliah Pemrograman".

Malang, 31 Juli 2025

Vincent Dwi Hartanto

DAFTAR ISI

LEMBAR PENGESAHAN TUGAS AKHIR i

PERNYATAAN KEASLIAN TUGAS AKHIR ii

Abstrak iii

Abstract iv

Kata Pengantar v

DAFTAR ISI vi

DAFTAR GAMBAR xi

BAB I Pendahuluan 1

1.1 Latar Belakang 1

1.2 Identifikasi Masalah 3

1.3 Batasan Masalah 3

1.4 Rumusan Masalah 3

1.5 Tujuan 4

1.6 Manfaat 4

1.7 Luaran 4

1.8 Sistematika Penulisan 4

BAB II Tinjauan Pustaka 6

2.1 Algoritma 6

2.2 Artificial Intelligence (AI) 6

2.4 Large Language Model (LLM) 7

2.4.1 Suhu LLM (Temperature) 8

2.4.2 Token LLM 8

2.5 Lang Chain 9

2.6 DeepSeek 10

2.6.1 API DeepSeek 10

2.6.2 Model 11
2.7 HTML 11
2.8 CSS 12
2.9 Typescript 13
2.10 Next.js 13
2.11 Bootstrap 14
2.12 Prisma ORM 14
2.13 PostgreSQL 15
2.14 Ollama 15
2.15 Apache JMeter 15
2.16 Wireshark 16
2.17 Google Lighthouse 16
2.18 Gatling 17
2.19 Penelitian Terdahulu 17
BAB III Analisis dan Perancangan Sistem 20
3.1 Alur Penelitian 20
3.2 Analisis Kebutuhan Sistem 22
3.2.1 Kebutuhan Fungsional 22
3.2.2 Kebutuhan Non-Fungsional 23
3.3 Perancangan Sistem 23
3.3.1 Activity Diagram 24
3.3.1.1 Proses Mahasiswa (*Student*) *Register* akun dan *Login*: 25
3.3.1.2 Proses Mentor (*Educator*) Register akun dan login: 25
3.3.1.3 Mentor Membuat Ujian (Exam) dan Soal (Quiz) 25
3.3.1.4 Mahasiswa Mengerjakan Ujian dan Soal 25
3.3.1.5 Proses Evaluasi Oleh AI 26

3.3.1.6 Sistem Menampilkan Hasil Feedback oleh AI 26
3.3.1.7 Mentor Memberikan Feedback 26
3.3.2 Entity Relationship Diagram (ERD) 26
3.3.3 Rancangan Antarmuka Pengguna (UI) 28
3.3.3.1 Halaman Registrasi Pengguna (Mentor dan Mahasiswa) 28
3.3.3.2 Halaman Login Pengguna (Mentor dan Mahasiswa) 29
3.3.3.3 Halaman Dashboard Mahasiswa 29
3.3.3.4 Halaman Detail Ujian Mahasiswa 30
3.3.3.5 Halaman Dashboard Mentor 31
3.3.3.6 Halaman Detail Ujian Mentor 32
3.3.3.7 Halaman Detail “Mahasiswa” Mentor 32
3.3.3.8 Halaman Tambah Ujian dan Tugas Mentor 33
3.4 Implementasi Sistem 34
3.4.1 Implementasi Website 34
3.4.2 Implementasi Basis Data 35
3.4.3 Implementasi Kecerdasan Buatan (AI) 36
3.5 Pengujian 36
3.5.1 Pengujian oleh Mahasiswa Mata Kuliah Pemrograman 37
3.5.2 Pengujian oleh Mentor (Dosen Mata Kuliah Pemrograman) 37
3.6 Evaluasi dan Dokumentasi 38
3.6.1 Evaluasi Fungsionalitas Sistem 38
3.6.2 Evaluasi Kinerja Sistem 38
3.6.3 Dokumentasi 40
BAB IV Hasil dan Pembahasan 42
4.1 Hasil Implementasi Sistem 42
4.1.1 Implementasi Antarmuka Pengguna (UI) 43

4.1.1.1 Halaman Registrasi Pengguna (Mentor dan Mahasiswa) 43
4.1.1.2 Halaman *Login* Pengguna (Mentor dan Mahasiswa) 44
4.1.1.3 Halaman *Dashboard* Mahasiswa 45
4.1.1.4 Halaman Detail Ujian Mahasiswa 45
4.1.1.5 Halaman *Dashboard* Mentor 47
4.1.1.6 Halaman Detail Ujian Mentor 48
4.1.1.7 Halaman Detail "Mahasiswa" Mentor 49
4.1.1.8 Halaman Tambah Ujian dan Tugas Mentor 52
4.1.2 Implementasi Basis Data 53
4.1.2.1 Teknologi dan Alat yang Digunakan 53
4.1.2.2 Struktur dan Relasi Basis Data 54
4.1.2.3 Alur Penyimpanan dan Pengambilan Data 58
4.1.3 Implementasi Kecerdasan Buatan (AI) 59
4.1.3.1 Mahasiswa Submit Jawaban 60
4.1.3.2 API Route Terima Data 60
4.1.3.3 Parse Soal 61
4.1.3.4 Pemilihan Model AI 61
4.1.3.5 Penyusunan *Prompt* Evaluasi 62
4.1.3.6 Mengirim ke AI (Mendapatkan Feedback dan Score) 62
4.1.3.7 Mengirim Feedback ke AI (mendapatkan isCorrect) 66
4.1.3.8 Kembali ke API Route (feedback, score, isCorrect) 67
4.1.3.9 Simpan ke Database 67
4.1.3.10 Menampilkan ke Antarmuka (UI) 67
4.2 Rangkaian Proses Sistem Secara Keseluruhan 72
4.2.1 Mahasiswa Membuat Akun 72
4.2.2 Mentor Membuat Akun 73

4.2.3 Mentor Membuat Ujian dan Soal 74
4.2.4 Mahasiswa Mengerjakan Ujian dan Soal 76
4.2.5 Proses Evaluasi oleh AI 77
4.2.6 Mentor Memberikan *Feedback* 78
4.3 Evaluasi Sistem melalui Pengujian dan Umpan Balik 79
4.3.1 Pengujian Fungsionalitas Sistem dengan Mahasiswa 80
4.3.1.1 Sampel Evaluasi AI terhadap Jawaban Benar 81
4.3.1.2 Sampel Evaluasi AI terhadap Jawaban Salah 83
4.3.1.3 Hasil Kuisioner *System Usability Scale* (*SUS*) 85
4.3.1.4 Hasil Kritik dan Saran (*feedback*) Setiap Mahasiswa 86
4.3.2 Evaluasi hasil AI oleh Mentor 89
4.3.3 Hasil Pengujian Performa Sistem 90
4.3.3.1 Hasil Uji *Response Time* 91
4.3.3.2 Hasil Uji *Throughtput* 93
4.3.3.3 Hasil Uji *Load Time* 94
4.3.3.4 Hasil Uji Skalabilitas 96
4.4 Pengujian Penggunaan Token dan Estimasi Biaya Evaluasi AI 98
BAB V Simpulan dan Saran 102
5.1 Kesimpulan 102
5.2 Saran 102
DAFTAR PUSTAKA 104
LAMPIRAN A: HASIL *FEEDBACK* AI 109
LAMPIRAN B: HASIL SURVEI SUS 123

DAFTAR GAMBAR

Gambar 3. 1 Flowchart Alur Penelitian 20

Gambar 3. 2 *Activity Diagram* 24

Gambar 3. 3 *Entity Relationship Diagram* 27

Gambar 3. 4 *Wireframe* halaman registrasi 29

Gambar 3. 5 *Wireframe* halaman *login* 29

Gambar 3. 6 *Wireframe* halaman *dashboard* mahasiswa 30

Gambar 3. 7 *Wireframe* halaman detail ujian mahasiswa 31

Gambar 3. 8 *Wireframe* halaman *dashboard* mentor 31

Gambar 3. 9 *Wireframe* halaman detail ujian mentor 32

Gambar 3. 10 *Wireframe* halaman detail "mahasiswa" mentor 33

Gambar 3. 11 Wireframe halaman tambah ujian dan tugas mentor 34

Gambar 4. 1 Implementasi Halaman Register 44

Gambar 4. 2 Implementasi Halaman Login 44

Gambar 4. 3 Implementasi Halaman *Dashboard* Mahasiswa 45

Gambar 4. 4 Implementasi Halaman Detail Ujian Mahasiswa 47

Gambar 4. 5 Implementasi Halaman *Dashboard* Mentor 48

Gambar 4. 6 Implementasi Halaman Detail Ujian Mentor 49

Gambar 4. 7 Implementasi Halaman Detail Mahasiswa Mentor 51

Gambar 4. 8 Implementasi Halaman Tambah Ujian dan Mentor 53

Gambar 4. 9 Alur Penyimpanan dan Pengambilan Data 58

Gambar 4. 10 Diagram Alur Implementasi Kecerdasan Buatan 60

Gambar 4. 11 Hasil Evaluasi oleh AI Jawaban Benar 68

Gambar 4. 12 Hasil Evaluasi oleh AI Jawaban salah 69

Gambar 4. 13 Tampilan Mentor Hasil Evaluasi oleh AI Jawaban Benar 70

Gambar 4. 14 Tampilan Mentor Hasil Evaluasi oleh AI Jawaban Salah 71

Gambar 4. 15 Alur Mahasiswa Membuat Akun 72

Gambar 4. 16 Alur Mentor Membuat Akun 73

Gambar 4. 17 Alur Mentor Membuat Ujian dan Soal 74

Gambar 4. 18 Alur Mahasiswa Mengerjakan Ujian dan Soal 76

Gambar 4. 19 Alur Proses Evaluasi AI 77

Gambar 4. 20 Alur Mentor Memberikan *Feedback* 78

Gambar 4. 21 Soal Evaluasi Hasil AI Jawaban Benar 81

Gambar 4. 22 Rubrik Penilaian Evaluasi Hasil AI Jawaban Benar 82

Gambar 4. 23 Jawaban Mahasiswa dalam Pengujian Evaluasi oleh AI 82

Gambar 4. 24 Hasil Evaluasi oleh AI Jawaban Benar 83

Gambar 4. 25 Soal Hasil Evaluasi AI Jawaban Salah 83

Gambar 4. 26 Rubrik Penilaian Evaluasi Hasil AI Jawaban Salah **Kesalahan!**

Bookmark tidak ditentukan.

Gambar 4. 27 Jawaban Mahasiswa dalam Pengujian Evaluasi oleh AI 84

Gambar 4. 28 Hasil Evaluasi oleh AI Jawaban Salah 85

Gambar 4. 29 Hasil Uji *Response Time* 92

Gambar 4. 30 Hasil Uji *Throughput* 94

Gambar 4. 31 Hasil Uji *Load Time* 1 95

Gambar 4. 32 Hasil Uji *Load Time* 2 95

Gambar 4. 33 Hasil Uji Skalabilitas 97

Gambar 4. 34 Jawaban Pengujian Token 100

Gambar 4. 35 Hasil Feedback AI Pengujian Token 100

Gambar 4. 36 Hasil Pengujian Penggunaan Token dan Estimasi Biaya 101

BAB I
Pendahuluan

1.1 Latar Belakang

Mata kuliah Algoritma merupakan salah satu fondasi utama dalam kurikulum program studi Informatika. Algoritma adalah urutan terbatas dari operasi-operasi terdefinisi dengan baik (Samsudin, Indrawan, & Mulyati, 2020). Mata kuliah ini sangat penting karena membekali mahasiswa dengan kemampuan berpikir logis, menyusun algoritma secara sistematis, serta mengimplementasikannya ke dalam bahasa pemrograman. Penguasaan materi algoritma menjadi dasar bagi mahasiswa untuk memahami konsep-konsep lanjutan dalam pemrograman, pengembangan perangkat lunak, hingga kecerdasan buatan di masa mendatang. Pemrograman komputer merupakan *skill* yang harus dikuasai oleh mahasiswa ilmu komputer (Harimurti, Qoirah, Nh, & Asmunin, 2018). Oleh karena itu, proses pembelajaran dan evaluasi pada mata kuliah ini harus dilaksanakan secara optimal untuk memastikan mahasiswa benar-benar memahami materi yang diajarkan.

Namun, dalam praktiknya, proses evaluasi dan penilaian hasil kode program mahasiswa masih menghadapi berbagai permasalahan. Memberikan umpan balik pada tugas pemrograman merupakan tugas yang sangat membosankan, rentan terhadap kesalahan, dan memakan banyak waktu bagi seorang pengajar, bahkan dalam lingkungan kelas yang standar (Gulwani, Radicek, & Zuleger, 2014). Selain itu, penilaian manual sering kali bersifat subjektif dan kurang konsisten, tergantung pada perspektif masing-masing instruktur. Tidak jarang pula mahasiswa hanya menerima nilai akhir tanpa mendapatkan *feedback* yang jelas dan terperinci mengenai kekurangan atau kesalahan dalam kode program mereka. Keadaan ini disebabkan oleh umpan balik yang diberikan tidak efektif sehingga tidak bermanfaat bagi mahasiswa dan tidak dapat meningkatkan performanya (Riezky, Andrianty, & Suji, 2020). Hal ini menghambat proses pembelajaran karena mahasiswa tidak mengetahui bagian mana yang perlu diperbaiki atau ditingkatkan.

Seiring perkembangan teknologi, *Artificial Intelligence* (AI) atau kecerdasan buatan telah menghadirkan solusi inovatif di berbagai bidang, termasuk dalam dunia pendidikan. Kecerdasan buatan adalah bidang studi yang berhubungan

dengan penangkapan, pemodelan, dan penyimpanan kecerdasan manusia ke dalam sebuah sistem teknologi informasi sehingga sistem tersebut dapat digunakan sebagai proses pengambilan keputusan yang dilakukan oleh manusia (Dewi, 2020). Penggunaan teknologi AI dalam pembelajaran dapat dijadikan sebagai solusi komputasi yang tangguh, teknologi bertenaga AI dapat digunakan untuk membantu proses belajar mengajar dan evaluasi (Zulkarnain & Yunus, 2023). Salah satu perkembangan signifikan dalam AI adalah kemunculan *Large Language Model* (LLM), yang mampu memahami dan menghasilkan teks dalam bahasa alami maupun bahasa pemrograman. Model seperti *DeepSeek-R1* menjadi contoh LLM yang dirancang khusus untuk memahami, mengevaluasi, dan memberikan umpan balik terhadap kode program secara otomatis.

Dengan memanfaatkan kemampuan AI dalam analisis data dan pemodelan, banyak pendekatan baru telah diajukan untuk mengukur kinerja siswa dengan lebih akurat dan efisien (Cahyanto & Sonjaya, 2024). LLM seperti *DeepSeek-R1* dapat dianalisis melalui integrasi dengan *LangChain*, sebuah *framework* yang memungkinkan pengembangan aplikasi berbasis AI secara modular dan fleksibel. Teknologi ini memberikan solusi atas permasalahan evaluasi kode program mahasiswa, dengan menyediakan penilaian yang akurat serta *feedback* yang lebih mendalam dan konstruktif. AI dapat digunakan untuk menilai sejumlah besar ujian dan pada saat yang sama mengidentifikasi pola kinerja siswa dengan data mining untuk memberi informasi kepada guru dalam memberikan umpan balik, seperti memberikan detail umpan balik lebih lanjut di area konten yang tidak dikuasai dengan baik oleh kelompok siswa (Wongvorachan, Lai, Bulut, Tsai, & Chen, 2022). Dengan demikian, mahasiswa dapat memahami kesalahan mereka secara langsung dan memperbaiki kode programnya dengan lebih cepat.

Oleh karena itu, dikembangkan sebuah platform evaluasi kode program berbasis *website* yang dibangun menggunakan *framework Next.js*. Platform ini mengintegrasikan *LangChain* untuk menghubungkan antarmuka pengguna dengan LLM *DeepSeek-R1*, sehingga proses evaluasi kode program dapat berjalan secara otomatis dan interaktif. Dengan adanya sistem ini, diharapkan pengajar dapat lebih mudah dalam melakukan penilaian, sementara mahasiswa memperoleh feedback

yang bermanfaat untuk meningkatkan kemampuan pemrogramannya secara efektif dan efisien.

1.2 Identifikasi Masalah

Berdasarkan latar belakang tersebut, dapat diidentifikasi beberapa permasalahan utama, yaitu lamanya waktu yang dibutuhkan dalam proses evaluasi kode secara manual, adanya potensi subjektivitas dalam penilaian, serta minimnya sistem evaluasi otomatis yang mampu menganalisis kode program secara menyeluruh, baik dari sisi sintaks, logika, maupun efisiensi.

1.3 Batasan Masalah

a) Sistem evaluasi kode yang dikembangkan dapat mendukung bahasa pemrograman apapun.
b) Model *Large Language Model* (LLM) yang digunakan dalam sistem ini adalah *DeepSeek-R1*.
c) *Framework LangChain* digunakan sebagai penghubung antara aplikasi *website* dengan LLM *DeepSeek-R1*.
d) Aplikasi yang dikembangkan berbasis website, menggunakan *framework Next.js*, akan dihosting di *Vercel*, serta menggunakan *Supabase* sebagai *database online*.
e) Proses evaluasi kode fokus pada analisis sintaksis, logika program, serta deteksi dan perbaikan kesalahan (*debugging*).
f) Sistem hanya fokus pada integrasi *Large Language Model DeepSeek-R1* melalui *LangChain* untuk proses evaluasi kode

1.4 Rumusan Masalah

a) Bagaimana cara mempercepat proses evaluasi kode dalam pembelajaran pemrograman agar tidak memerlukan waktu lama seperti evaluasi manual?
b) Bagaimana merancang sistem evaluasi yang dapat mengurangi atau menghilangkan potensi subjektivitas dalam penilaian kode program?
c) Bagaimana memanfaatkan teknologi AI, khususnya LLM *DeepSeek-R1*, untuk mendukung proses evaluasi kode secara *real-time*?

1.5 Tujuan

a) Mempercepat proses evaluasi pembelajaran pemrograman bagi pengajar dan peserta didik.
b) Merancang sistem evaluasi untuk mengurangi potensi subjektivitas dalam penilaian kode program
c) Mengembangkan sistem evaluasi kode otomatis berbasis web yang mengintegrasikan teknologi *deep learning* dan *LLM* seperti *DeepSeek-R1* untuk memberikan *feedback* secara *real-time* kepada peserta didik.

1.6 Manfaat

a) Bagi instruktur atau pengajar, membantu pengajar dalam proses evaluasi kode program peserta didik secara lebih cepat, objektif, dan efisien tanpa harus melakukan penilaian manual.
b) Bagi peserta didik**,** memberikan *feedback* langsung dan konstruktif terhadap kode yang dikirimkan, sehingga peserta didik dapat lebih cepat memahami kesalahan dan memperbaiki kemampuan pemrograman mereka.
c) Bagi Universitas Ma Chung, khususnya Program Studi Teknik Informatika, manfaat yang didapatkan adalah dapat mempersiapkan lulusan yang kompeten dan siap kerja dengan memberikan bekal kepada mahasiswa melalui proses pembelajaran selama Tugas Akhir.

1.7 Luaran

a) Sebuah *website* aplikasi evaluasi kode program secara otomatis, yang terintegrasi dengan LLM *DeepSeek-R1* menggunakan *framework LangChain*.
b) Artikel atau publikasi ilmiah pada seminar atau jurnal, yang membahas penerapan LLM *DeepSeek-R1* dan *LangChain* dalam evaluasi kode secara otomatis.

1.8 Sistematika Penulisan

Penulisan laporan tugas akhir ini ditulis dengan sistematika penulisan sebagai berikut:

1. Bab I: Pendahuluan

   Membahas tentang latar belakang dibentuknya sistem evaluasi kode

otomatis berbasis *website* yang terintegrasi dengan *LLM Deepseek-RI*. Bab ini juga memuat tentang batasan masalah, rumusan masalah, tujuan dan luaran dari sistem yang dikembangkan.

2. Bab II: Tinjauan Pustaka

   Membahas tentang teori-teori dan teknologi relevan yang digunakan dalam pengembangan sistem, seperti algoritma, kecerdasan buatan (AI), *Large Language Model (LLM)*, *LangChain*, *DeepSeek-R1* dan sistem pembuatan *website*. Selain itu disediakan studi pustaka yang berkaitan dengan penelitian terdahulu.

3. Bab III: Analisis dan Perancangan Sistem

   Membahas tentang hasil analisis kebutuhan sistem, baik dari fungsional maupun non fungsional, serta perancangan sistem dari segi antarmuka, diagram aktivitas, sistem basis data, dan kecerdasan buatan (AI). Bab ini juga menjelaskan tentang metode pengujian yang digunakan.

4. Bab IV: Hasil dan Pembahasan

   Membahas tentang hasil implementasi sistem yang telah dirancang sebelumnya pada Bab III, mulai dari segi antarmuka, basis data, hingga proses evaluasi kode dengan AI. Bab ini juga membahas tentang hasil pengujian yang telah dilakukan.

5. Bab V: Penutup

   Membahas tentang kesimpulan dari perancangan sistem hingga hasil implementasi serta saran pengembangan untuk penelitian selanjutnya.

BAB II
Tinjauan Pustaka

2.1 Algoritma

Algoritma adalah urutan terbatas dari operasi-operasi terdefinisi dengan baik (Samsudin, Indrawan, & Mulyati, 2020). Dalam konteks ilmu komputer, algoritma berfungsi sebagai dasar dalam menyusun instruksi-instruksi logis yang dapat dijalankan oleh komputer untuk memproses data atau melakukan perhitungan. Setiap algoritma harus memiliki karakteristik tertentu seperti *input*, *output*, kejelasan langkah, efektivitas, dan berhenti setelah sejumlah langkah tertentu.

Pemahaman dan penguasaan algoritma menjadi pondasi utama dalam proses pengembangan perangkat lunak, karena struktur logika dari suatu program ditentukan oleh algoritma yang digunakan. Dalam dunia pendidikan, khususnya pada mata kuliah Pemrograman, mahasiswa dibekali kemampuan berpikir sistematis dan logis, yang merupakan keterampilan penting dalam menyelesaikan persoalan komputasi secara efisien.

2.2 Artificial Intelligence (AI)

Kecerdasan buatan adalah bidang studi yang berhubungan dengan penangkapan, pemodelan, dan penyimpanan kecerdasan manusia ke dalam sebuah sistem teknologi informasi sehingga sistem tersebut dapat digunakan sebagai proses pengambilan keputusan yang dilakukan oleh manusia (Dewi, 2020). AI bertujuan untuk membuat sistem komputer mampu meniru kemampuan berpikir manusia, seperti memahami, belajar, merencanakan, dan memecahkan masalah.

Dalam praktiknya, AI dikembangkan untuk menyelesaikan tugas-tugas yang membutuhkan kecerdasan manusia, seperti pengenalan suara, pengolahan bahasa alami, pengambilan keputusan, serta pengenalan pola. Teknologi ini memungkinkan komputer untuk belajar dari data, menyesuaikan diri dengan masukan baru, dan menjalankan tugas-tugas tertentu secara otomatis tanpa pemrograman ulang secara eksplisit. Dengan kemampuan tersebut, AI telah menjadi teknologi yang banyak diadopsi di berbagai sektor, termasuk industri, kesehatan, pendidikan, dan bisnis. Dalam konteks sistem berbasis teknologi informasi, penerapan AI memungkinkan peningkatan efisiensi, keakuratan, serta

konsistensi dalam menjalankan fungsi-fungsi yang sebelumnya hanya dapat dilakukan oleh manusia.

2.3 Generative Artificial Intelligence (GAI)

GAI (Generative Artificial Intelligence) mengacu pada kategori luas dari kecerdasan buatan yang berfokus pada menyintesis dan/atau menghasilkan konten atau data yang sering kali sulit dibedakan dari konten yang dibuat oleh manusia (Gupta, Ding, Guan, & Ding, 2024). Generative AI menggunakan algoritma dan model pembelajaran mesin untuk menghasilkan konten yang mirip dengan karya manusia, seperti teks, gambar, dan suara. Beberapa contoh aplikasi berbasis Generative Artificial Intelligence yaitu Duolingo Stories, aplikasi pembelajaran bahasa untuk menghasilkan cerita interaktif dalam bahasa target, Adobe Sensei, menghasilkan konten kreatif, ChatGPT dan lain-lain (Windiart, Bahri, & Prabowo, 2023).

2.3.1 Halusinasi Generative Artificial Intelligence (GAI)

Halusinasi GAI merupakan kondisi ketika model memberikan hasil yang salah atau tidak relevan. Halusinasi ini dibedakan menjadi 2 bagian besar yaitu *intrinsic vs extrinsic* dan *factual vs semantic*. Halusinasi *intrinsic* terjadi ketika model memberikan hasil yang tidak konsisten, sedangkan *extrinsic* terjadi ketika model memberikan hasil yang tidak sesuai dengan fakta. Halusinasi *factual* ini berhubungan dengan hasil model yang menyangkal dengan fakta yang terverfikasi sedangkan *semantic* terjadi ketika model memberikan hasil yang terdengar meyakinkan namun tidak masuk logika (Ahadian & Guan, 2025).

2.4 Large Language Model (LLM)

Large Language Model (LLM) adalah model bahasa berskala besar yang dilatih pada dataset besar dengan pemahaman dan pengetahuan bahasa yang luas tanpa harus mengatur data untuk memproses dan memahami bahasa alami tertentu (Aprilia, Harahap, Yanto, & Yusra, 2024). LLM dilatih menggunakan kumpulan data teks dalam jumlah besar yang mencakup berbagai domain, gaya bahasa, dan struktur kalimat. Pelatihan ini memungkinkan model mengenali pola-pola linguistik dan semantik yang kompleks, sehingga mampu menghasilkan respons yang relevan dan kontekstual terhadap berbagai input teks.

Secara teknis, LLM menggunakan jaringan saraf *transformer* yang memungkinkan pemrosesan kata dalam konteks kalimat secara efisien. Hal ini menjadikan LLM unggul dalam berbagai tugas pemrosesan bahasa alami (*Natural Language Processing*/NLP), seperti penerjemahan, rangkuman, pengenalan entitas, hingga penyusunan teks yang koheren. Selain itu, karena kemampuannya untuk belajar dari data yang sangat besar, LLM juga mampu beradaptasi terhadap beragam gaya penulisan dan konteks, menjadikannya salah satu teknologi utama dalam perkembangan kecerdasan buatan berbasis bahasa.

2.4.1 Suhu LLM (Temperature)

Temperature mengontrol tingkat ketidakpastian atau keacakan dalam proses generasi, sehingga menghasilkan keluaran yang lebih beragam dengan menyeimbangkan probabilitas dari kata-kata kandidat (Peeperkorn, Kouwenhoven, Brown, & Jordanous, 2024). Untuk tugas-tugas kreatif, temperature sering dianggap sebagai parameter yang memungkinkan perilaku kreatif dalam model bahasa (Manjavacas, Karsdorp, Burtenshaw, & Kestemont).

Parameter ini umumnya memiliki rentang nilai antara 0 hingga 1, di mana nilai rendah (misalnya 0.1–0.3) menghasilkan keluaran yang lebih deterministik dan terkontrol, sedangkan nilai tinggi (mendekati 1) cenderung menghasilkan teks yang lebih variatif namun juga lebih tidak terduga. Nilai default temperature adalah 1.0, namun nilai tersebut dapat disesuaikan dengan kebutuhan seperti coding/math dengan nilai 0, data cleaning/data analysis dengan nilai 1.0, general conversation dengan nilai 1.3, translation dengan nilai 1.3 dan creative writing/poetry dengan nilai 1.5 (DeepSeek, 2025).

2.4.2 Token LLM

Token adalah *sub-string* yang digunakan model untuk memecah seluruh teks *input*, serta unit-unit yang kemudian digabungkan kembali untuk menghasilkan teks *output* (Zimmerman et al., 2025). Setiap model LLM memiliki batas maksimum jumlah token yang dapat diproses dalam satu permintaan, baik untuk input maupun output, yang dikenal sebagai *context length limit*. Salah satu contoh *LLM DeepSeek* memiliki batas maksimum

jumlah token (*context length limit*) yang berbeda tergantung pada model yang digunakan. Model *deepseek-chat* memiliki batas hingga 8.000 token, sedangkan model *deepseek-reasoner* mendukung hingga 64.000 token, memungkinkan pemrosesan konteks yang jauh lebih panjang (DeepSeek, 2025).

Batas jumlah token yang dapat diproses oleh suatu model LLM sangat memengaruhi kualitas dan akurasi hasil yang dihasilkan. Adanya penurunan performa yang signifikan ketika input semakin panjang, bahkan sebelum mencapai batas maksimal panjang input yang didukung oleh model (Levy, Jacoby, & Goldber, 2024).

2.4.3 Prompt LLM

Prompt adalah *input* untuk model *generative AI* yang digunakan untuk menghasilkan *output*. Prompt biasanya berisi tulisan, gambar, suara ataupun media lainnya (Schulhoff et al, 2024). Teknik dalam menyusun prompt LLM memiliki banyak kombinasi sesuai dengan kebutuhan. Sebagian besar kombinasi teknik *prompting* tidak memberikan peningkatan (atau penurunan) yang signifikan secara statistik terhadap kebenaran jawaban, kualitas, atau kemiripan kode. Namun, memberikan informasi tipe untuk fungsi yang akan dibuat baik secara eksplisit melalui tanda tangan fungsi, maupun secara implisit lewat contoh *few-shot* memiliki dampak positif yang paling jelas, terutama terhadap kebenaran jawaban (Khojah , Neto, Mohamad, & Leitner, 2025).

2.5 Lang Chain

Langchain adalah sebuah *open-source framework* yang dapat memungkinkan pembuatan aplikasi LLM dengan data custom dari dokumen pribadi, skalabilitas tinggi, dan mempercepat proses pengembangan (Muhajir, Prastiti, & Koeshardianto, 2025). LangChain dirancang untuk mempermudah integrasi antara *Large Language Model* (LLM) dengan sumber data eksternal, sehingga memungkinkan LLM bekerja dengan konteks yang lebih spesifik dan relevan. *Framework* ini mendukung berbagai komponen penting seperti *document loaders*, *retrievers*, *chains*, hingga *agents*, yang memungkinkan pengguna membangun alur pemrosesan bahasa alami yang kompleks namun modular.

Dengan pendekatan yang fleksibel, *LangChain* memungkinkan pengembang menggabungkan berbagai sumber data, seperti dokumen teks, basis data, hingga API eksternal, lalu mengelolanya dalam bentuk *pipeline* logis untuk digunakan oleh model bahasa. Hal ini menjadikan *LangChain* sangat cocok digunakan dalam pengembangan aplikasi yang membutuhkan pemrosesan bahasa alami tingkat lanjut, seperti sistem tanya-jawab, asisten virtual, hingga pencarian berbasis konteks.

2.6 DeepSeek

DeepSeek adalah perusahaan pengembang kecerdasan buatan (AI) yang berbasis di *Hangzhou*, Tiongkok, perusahaan ini didirikan oleh Liang Wenfeng, lulusan Universitas *Zhejiang*, pada bulan Mei 2023 (Kerner, 2025). Fokus utama *DeepSeek* adalah menciptakan model AI yang mampu memahami dan menghasilkan teks secara kontekstual serta relevan, baik dalam bahasa Mandarin maupun bahasa lainnya.

Model yang dikembangkan oleh *DeepSeek* dirancang untuk mendukung berbagai aplikasi AI modern, seperti asisten virtual, sistem tanya-jawab, penulisan otomatis, dan analisis teks. Keunggulan *DeepSeek* terletak pada penggunaan arsitektur canggih dan data pelatihan dalam skala besar, yang memungkinkan model buatannya memiliki kemampuan memahami konteks dan struktur bahasa yang kompleks.

2.6.1 API DeepSeek

Application Programming Interface (API) merupakan teknologi antarmuka yang dibangun oleh pengembang sistem supaya sebagian atau keseluruhan fungsi sistem dapat diakses secara bersamaan dengan baik. Sementara *Representational State Transfer* (*REST*) merupakan salah satu gaya arsitektur dari pengembangan API yang menggunakan *Hypertext Transfer Protocol* (HTTP) dalam melakukan komunikasi data (Triawan & Siboro, 2021).

DeepSeek menyediakan layanan model kecerdasan buatan yang dapat diakses melalui Application Programming Interface (API), yang memungkinkan integrasi langsung ke dalam sistem pihak ketiga. Komunikasi

dengan layanan ini dilakukan melalui *base URL https://api.deepseek.com*, dengan otorisasi berbasis *API key* yang dapat diperoleh melalui proses pendaftaran resmi. Untuk pemanggilan model, *DeepSeek* menyediakan dua model utama yang tersedia melalui API yaitu model *deepseek-chat* yang merepresentasikan versi *DeepSeek-V3*, dan model *deepseek-reasoner* yang menunjuk pada versi *DeepSeek-R1* (DeepSeek, 2025).

2.6.2 Model

Dari kedua model yang disediakan *DeepSeek*, yaitu *deepseek-chat* (*DeepSeek-V3*) dan *deepseek-reasoner* (*DeepSeek-R1*), performa dalam konteks pemrograman menunjukkan hasil yang mencolok. Berdasarkan evaluasi dari *LiveCodeBench*, model *deepseek-reasoner* mencapai skor 73.3 (DeepSeek, 2025), jauh mengungguli *deepseek-chat* yang hanya memperoleh skor 49.2 (DeepSeek, 2025). Hal ini menunjukkan bahwa *deepseek-reasoner* jauh lebih unggul untuk tugas-tugas yang berkaitan dengan pemahaman, penulisan, dan evaluasi kode program dibandingkan *deepseek-chat* .

2.7 HTML

HTML (*HyperText Markup Language*) merupakan bahasa *markup* yang digunakan untuk membangun struktur dasar sebuah halaman web (Summit, 2023). HTML memberikan struktur dasar untuk halaman web dengan menggunakan elemen-elemen yang dikenal sebagai *tag*. HTML bekerja sama dengan CSS untuk menangani presentasi visual dan *JavaScript* untuk menangani interaktivitas, sehingga membuat halaman web menjadi lebih dinamis dan menarik. Berkas HTML memiliki ekstensi .html dan dapat diakses di seluruh web *browser* modern. Berkas tersebut akan ditampilkan sesuai dengan struktur dan gaya yang ditentukan di dalamnya. Berkas HTML juga dapat diakses melalui dua cara, yaitu melalui komputer lokal dan melalui URL. Jika melalui komputer lokal, berkas HTML cukup dibuka seperti biasa. Sedangkan jika melalui URL, berkas HTML harus diunggah terlebih dahulu ke server web sehingga menghasilkan domain yang dapat diakses secara *online*.

Untuk mengetikkan skrip HTML, dapat digunakan *text editor* seperti *Notepad* sebagai bentuk paling sederhana, atau *text editor* khusus yang dapat mengenali

setiap unsur skrip HTML dan menampilkannya dengan warna berbeda agar lebih mudah dibaca, seperti *Notepad*++, *Sublime Text*, dan masih banyak lagi aplikasi lain yang sejenis (Permatasari & Suhendi, 2020). Skrip HTML memberikan struktur dasar untuk halaman web dengan membagi dokumen menjadi dua bagian utama: <head>, yang berisi metadata dan informasi penting seperti judul dan pengaturan karakter; serta <body>, yang berisi konten yang terlihat oleh pengguna seperti teks, gambar, dan tautan.

2.8 CSS

CSS adalah singkatan dari *Cascading Style Sheets*, yaitu dokumen web yang berfungsi mengatur elemen HTML dengan berbagai properti yang tersedia sehingga dapat tampil dengan berbagai gaya yang diinginkan (Permatasari & Suhendi, 2020). Dengan CSS, pengembang dapat mengontrol berbagai aspek visual seperti warna, font, margin, dan tata letak elemen di halaman. CSS juga mendukung desain responsif, memungkinkan halaman web menyesuaikan tampilan secara otomatis sesuai dengan ukuran layar perangkat, seperti smartphone, tablet, dan desktop.

Fungsi penting CSS meliputi pemisahan konten dari presentasi, yang mempermudah pemeliharaan dan pembaruan desain, serta kemampuan untuk menerapkan gaya konsisten di seluruh situs web dengan menggunakan kelas dan ID. CSS dapat diterapkan ke halaman web melalui tiga metode utama: internal, *inline*, dan eksternal. Internal CSS diletakkan dalam elemen <style> di bagian <head> dari dokumen HTML, memungkinkan gaya diterapkan hanya pada halaman tersebut. Inline CSS diterapkan langsung pada elemen HTML menggunakan atribut style, ideal untuk gaya spesifik yang hanya berlaku untuk satu elemen. Eksternal CSS, yang diletakkan dalam berkas terpisah dengan ekstensi .css dan dihubungkan melalui elemen <link>, memungkinkan penerapan gaya secara konsisten di seluruh situs web, memudahkan pemeliharaan dan pembaruan. Masing-masing metode memiliki fungsinya tersendiri dan sering digunakan bersama untuk efisiensi dan keteraturan dalam desain web.

2.9 Typescript

TypeScript adalah sebuah bahasa pemrograman yang berbasis pada *JavaScript*, tetapi menambahkan fitur-fitur seperti pengetikan statis, kelas, antarmuka, dan modul (Nugroho, et al., 2023). Dalam pengembangan aplikasi web modern, termasuk yang menggunakan *framework* seperti *Next.js*, *TypeScript* menjadi pilihan populer karena memberikan kontrol lebih terhadap struktur kode, serta memudahkan proses *debugging* dan pengembangan skala besar. Penggunaan *TypeScript* dalam proyek berbasis *Next.js* membantu membangun antarmuka pengguna yang stabil dan mudah dipelihara. Dengan fitur pengetikan statis, kesalahan dapat dideteksi lebih awal saat pengembangan, tanpa perlu menjalankan aplikasi.

TypeScript juga mendukung integrasi yang rapi dengan pustaka pihak ketiga, termasuk *LangChain* yang menghubungkan antarmuka dengan model AI seperti *DeepSeek*. *TypeScript* memperkuat arsitektur sisi klien dan mempermudah kolaborasi serta pemeliharaan aplikasi. Dengan demikian, *TypeScript* berkontribusi pada pengembangan sistem yang andal dan efisien untuk kebutuhan akademik.

2.10 Next.js

Next.js adalah *framework react* yang bertujuan untuk membantu para pengembang membuat aplikasi *online* modern yang lebih efisien, serbaguna, dan optimal (Maharani, 2025) *Framework* ini menawarkan fitur-fitur unggulan seperti *rendering* sisi server, *static site generation*, serta *routing* otomatis, yang menjadikannya pilihan ideal dalam membangun aplikasi web berskala kecil hingga besar. *Next.js* juga memungkinkan integrasi API langsung melalui API *routes*, sehingga pengembang dapat membuat *backend* sederhana dalam satu proyek tanpa perlu memisahkan *frontend* dan *backend* secara eksplisit.

Kelebihan utama dari *Next.js* antara lain adalah kecepatan dalam pemuatan halaman berkat mekanisme *pre-rendering*, *SEO-friendly* karena mampu menghasilkan halaman HTML yang siap dibaca oleh mesin pencari, serta pengelolaan rute yang lebih mudah tanpa konfigurasi tambahan. Selain itu, *Next.js* juga memiliki dukungan terhadap *TypeScript* secara bawaan, sehingga pengembangan aplikasi menjadi lebih aman dan efisien. Kombinasi dari fitur-fitur

tersebut membuat *Next.js* sangat cocok untuk digunakan dalam proyek sistem berbasis *website* yang membutuhkan kestabilan, efisiensi, dan kemudahan dalam pemeliharaan.

2.11 Bootstrap

Bootstrap adalah *front-end framework* yang bagus dan luar biasa yang mengedepankan tampilan untuk mobile device (*Handphone*, *smartphone* dan lain lain.) guna mempercepat dan mempermudah pengembangan *website* (Suprayogi & Rahmanesa, 2019). *Bootstrap* merupakan *framework* berbasis HTML, CSS, dan *Javascript* yang menyediakan kumpulan alat dan komponen siap pakai untuk desain responsif dan pengembangan antarmuka pengguna yang konsisten.

Framework ini menyediakan berbagai elemen desain siap pakai seperti tombol, formulir, dan sistem *grid* untuk pembuatan layout yang menyesuaikan dengan berbagai ukuran layar. Kelebihan menggunakan *Bootstrap* termasuk kemudahan dalam penerapan desain responsif, konsistensi gaya, dan penghematan waktu pengembangan dengan mengurangi kebutuhan menulis kode dari awal. Untuk mengakses *Bootstrap*, dapat menambahkan file CSS dan *Javascript Bootstrap* ke proyek web melalui tautan *CDN* atau dengan mengunduh berkas dari situs resmi *Bootstrap* dan mengintegrasikannya ke dalam berkas HTML.

2.12 Prisma ORM

Prisma ORM adalah alat yang dapat mengelola interaksi antara aplikasi dan pengguna menggunakan objek dan kode program tanpa perlu menulis kueri dalam pemrosesan data (Khadija, et al., 2024). *Prisma* menerapkan pendekatan *Object-Relational Mapping* (ORM) modern yang terintegrasi dengan baik dalam ekosistem *TypeScript* dan *JavaScript*.

Salah satu keunggulan utama Prisma adalah fitur *auto-completion* serta validasi tipe data yang kuat, yang membantu pengembang meminimalkan kesalahan selama proses pengembangan. *Prisma* juga mendukung berbagai jenis database seperti *PostgreSQL, MySQL, SQLite,* dan *MongoDB*, menjadikannya fleksibel untuk digunakan dalam berbagai kebutuhan pengembangan aplikasi modern.

2.13 PostgreSQL

PostgreSQL merupakan *database* yang dikembangkan oleh *University of California di Berkeley Computer Science Department* (Prasetyo & Susetyo, 2022). Selain itu, *PostgreSQL* juga memungkinkan pengguna untuk membuat fungsi dan prosedur tersimpan menggunakan berbagai bahasa pemrograman, termasuk *PL/pgSQL*, *Python*, dan lainnya.

Salah satu keunggulan utama *PostgreSQL* adalah kemampuannya dalam menangani beban kerja besar dan kompleks, sehingga banyak digunakan dalam aplikasi berskala *enterprise* maupun proyek riset ilmiah. Dengan sifatnya yang fleksibel dan dapat diperluas (*extensible*), *PostgreSQL* dapat diintegrasikan dengan berbagai teknologi modern, serta dioptimalkan untuk berbagai kebutuhan spesifik pengguna. Kombinasi fitur-fitur ini menjadikan *PostgreSQL* sebagai salah satu pilihan utama dalam pengembangan aplikasi berbasis data saat ini.

2.14 Ollama

Ollama adalah platform yang menyediakan solusi untuk membangun dan mengelola model bahasa yang dapat disesuaikan untuk aplikasi tertentu (Jimmy, 2024). *Ollama* dirancang untuk menjalankan model-model bahasa besar (LLM) secara lokal, tanpa perlu mengandalkan layanan *cloud* eksternal. Hal ini memberikan keuntungan signifikan dalam hal privasi data, latensi yang lebih rendah, serta fleksibilitas dalam integrasi dengan berbagai sistem.

Kemampuan *Ollama* dalam mengelola model secara efisien menjadikannya solusi ideal bagi pengembang yang ingin membangun aplikasi berbasis LLM dengan kontrol penuh atas lingkungan pengoperasian. Dengan pendekatan berbasis *command-line* dan integrasi yang mudah ke berbagai bahasa pemrograman, *Ollama* mempercepat proses pengembangan serta mempermudah eksperimen dengan model-model bahasa yang berbeda.

2.15 Apache JMeter

Apache JMeter adalah alat pengujian beban yang serbaguna dan sumber terbuka yang digunakan untuk melakukan pengujian kinerja pada aplikasi *web* (Indrianto, 2023). Alat ini dapat mensimulasikan berbagai jenis permintaan pengguna dan mengukur respons dari server atau aplikasi yang diuji. JMeter

digunakan untuk pengujian skala besar, dengan kemampuan untuk mensimulasikan ribuan hingga jutaan pengguna yang mengakses aplikasi secara bersamaan.

JMeter juga menyediakan fitur visualisasi hasil pengujian dalam bentuk grafik dan laporan yang memudahkan analisis. Alat ini kompatibel dengan berbagai platform seperti Windows, Linux, dan macOS, serta dapat digunakan untuk menguji aplikasi web baik yang berjalan di server lokal maupun cloud.

2.16 Wireshark

Wireshark, sebuah perangkat lunak analisis jaringan yang terkenal, berfungsi sebagai detektif digital di dunia maya (Soepono, 2023). Alat ini memungkinkan pengguna untuk menangkap dan memeriksa paket data yang dikirim melalui jaringan komputer. *Wireshark* memberikan tampilan yang rinci dari paket data, yang memungkinkan pengguna untuk memverifikasi dan menganalisis komunikasi yang terjadi di jaringan.

Penggunaan *Wireshark* meliputi berbagai tugas, seperti mengukur latensi jaringan, memeriksa penggunaan bandwidth, dan menganalisis masalah konektivitas. Dengan antarmuka grafis yang intuitif, pengguna dapat menyaring dan menampilkan paket data sesuai dengan kebutuhan analisis mereka. *Wireshark* sangat berguna dalam memecahkan masalah jaringan yang kompleks dan memastikan kinerja jaringan yang optimal, serta mengidentifikasi area yang membutuhkan perbaikan atau optimasi lebih lanjut.

2.17 Google Lighthouse

Google Lighthouse adalah alat *open-source* yang digunakan untuk mengukur kinerja dan kualitas halaman *web*. Pengujian kinerja situs *web* menggunakan alat seperti *Google Lighthouse* adalah tugas yang umum dalam praktik perangkat lunak dan penelitian (Hericko, Sumak, & Brdnik, 2022). Alat ini memberikan analisis mendalam tentang berbagai aspek situs web, termasuk kinerja, aksesibilitas, SEO, dan praktik terbaik. Dengan menggunakan *Lighthouse*, pengembang dapat mengidentifikasi area yang perlu ditingkatkan untuk memastikan pengalaman pengguna yang lebih baik, seperti waktu muat halaman yang cepat.

Selain itu, *Google Lighthouse* memberikan skor yang memudahkan pengembang untuk memantau progres perbaikan yang dilakukan pada situs. Skor tersebut didasarkan pada berbagai metrik teknis dan pengalaman pengguna yang dapat digunakan sebagai referensi untuk meningkatkan performa situs secara keseluruhan.

2.18 Gatling

Gatling (Gatling Project, Stress Tool, t.t.) dirancang untuk digunakan dalam pengujian beban (*load testing*), menganalisis, dan mengukur kinerja berbagai layanan, dengan fokus utama pada aplikasi web (Paz & Bernardino, 2018). Dengan menggunakan bahasa pemrograman Scala, Gatling mampu membuat simulasi pengguna dalam jumlah besar yang mengakses sistem secara bersamaan. Hal ini memungkinkan pengembang untuk mengetahui batas kapasitas sistem dan titik kritis performa sebelum aplikasi dirilis secara luas.

Selain itu, Gatling dilengkapi dengan antarmuka yang intuitif dan laporan hasil pengujian yang detail, seperti waktu respons, tingkat kesuksesan permintaan, serta throughput. Keunggulan Gatling terletak pada kemampuannya menghasilkan beban yang konsisten dan akurat dengan penggunaan sumber daya yang rendah, sehingga sangat cocok digunakan dalam pengujian performa aplikasi dalam konteks penelitian maupun industri.

2.19 Penelitian Terdahulu

Salah satu penelitian yang membahas tentang penerapan DeepSeek pada tugas-tugas berbasis pemrograman adalah "*DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning*", yang ditulis oleh *DeepSeek-AI et al* (2025*)*. Penelitian ini menunjukkan bahwa *DeepSeek-R1* memiliki kemampuan penalaran yang sangat baik dalam berbagai tugas, termasuk pemrograman. Dalam riset tersebut, *DeepSeek-R1* mencatatkan skor 97,3% pada *MATH-500* dan rating Elo 2.029 di *Codeforces*, yang mengindikasikan bahwa model ini dapat mengalahkan 96,3% peserta manusia dalam kompetisi pemrograman. Hasil ini mengonfirmasi bahwa *DeepSeek-R1* sangat efektif dalam menyelesaikan tugas-tugas *coding* yang kompleks, dan relevansinya dalam pengembangan sistem evaluasi otomatis berbasis AI di bidang pendidikan pemrograman.

Adapun penelitian yang dilakukan oleh Gao Tianchen, Jin Jiashun, Ke Zhang Tracy dan Moryoussef Gabriel (2025) berjudul "*A Comparison of DeepSeek and Other LLMs*" menyatakan bahwa model *DeepSeek* menunjukkan performa yang kompetitif dibandingkan berbagai *Large Language Model* (*LLM*) lainnya. Dalam studi yang membandingkan akurasi prediksi pada tugas klasifikasi authorship dan sitasi, *DeepSeek* mampu mengungguli model seperti Gemini, GPT, dan *LLaMA* di sebagian besar pengujian. Meskipun performanya masih berada di bawah *Claude* dari sisi akurasi, *DeepSeek* tetap menawarkan keunggulan signifikan dari segi efisiensi biaya, di mana penggunaan *DeepSeek* jauh lebih ekonomis dibanding *Claude* yang biayanya berkali-kali lipat lebih tinggi. *Claude* menelan biaya sebesar $12.30, *LLaMA* $1.20, sedangkan *DeepSeek*, *Gemini*, dan *GPT* masing-masing tidak melebihi $0.30.

Selain itu, menurut penelitian yang dilakukan oleh Manik Md Motaleb Hossen (2025) berjudul *"ChatGPT vs. DeepSeek: A Comparative Study on AI-Based Code Generation"*, *DeepSeek* terbukti mengungguli *ChatGPT* dalam menyelesaikan soal-soal kontes pemrograman. Dalam pengujian yang dilakukan, *DeepSeek* berhasil menghasilkan kode yang akurat sejak percobaan pertama, sedangkan *ChatGPT* mengalami kesulitan dalam menyelesaikan banyak soal meskipun telah dilakukan beberapa percobaan. Secara kuantitatif, tingkat *correctness* (kebenaran solusi) *DeepSeek* mencapai 0.5454, jauh di atas *ChatGPT* yang hanya memperoleh skor 0.1875. Dari segi *efficiency* (rasio antara ketepatan dan waktu eksekusi), *DeepSeek* mencatat skor 0.62, sedangkan *ChatGPT* hanya 0.51. Dalam hal *readability* (tingkat keterbacaan kode berdasarkan jumlah error), *DeepSeek* mencatat skor 0.097, sedikit lebih baik dibandingkan *ChatGPT* yang memperoleh 0.127. Temuan ini menunjukkan bahwa *DeepSeek* tidak hanya lebih akurat, tetapi juga lebih efisien dan menghasilkan kode yang lebih bersih untuk tugas-tugas pemrograman kompleks.

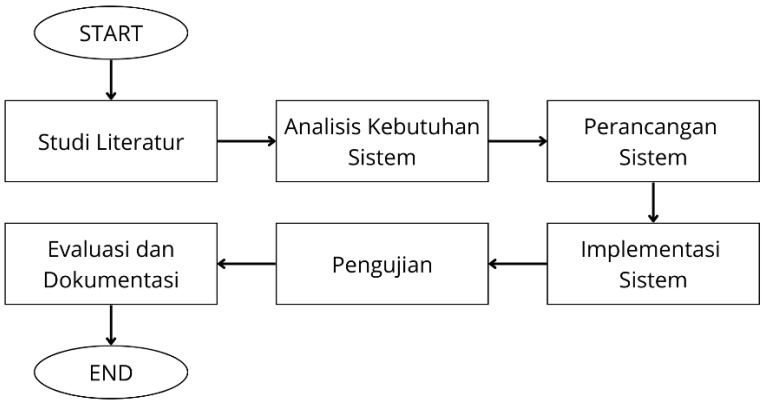
Adapun penelitian lain yang dilakukan oleh Subron Hadid, Ulfa Ramadhani, Silvia Dian, dan Andi Gusmaulia Eka Putri (2024) berjudul *"Analisis Dampak Penggunaan Chatbot AI dalam Pembelajaran di Kalangan Mahasiswa PGSD Universitas Jambi"* membahas penggunaan teknologi *chatbot* berbasis kecerdasan buatan (AI) dalam mendukung proses pembelajaran. Penelitian ini

menunjukkan bahwa penggunaan *chatbot* AI mampu meningkatkan efisiensi pembelajaran sebanyak 80% mahasiswa yang melaporkan bahwa *chatbot* membantu mereka mengakses informasi dengan lebih cepat dan efisien dibandingkan metode tradisional. Selain itu, 75% mahasiswa merasa bahwa *chatbot* AI memberikan dukungan akademik yang lebih personal dan responsif terhadap pertanyaan mereka. Terakhir, 70% mahasiswa melaporkan adanya peningkatan motivasi belajar berkat interaksi yang dinamis dengan *chatbot*. Temuan ini menjadi salah satu dasar penting dalam merancang sistem berbasis AI yang mampu membantu proses pembelajaran dan evaluasi tugas secara otomatis.

BAB III
Analisis dan Perancangan Sistem

3.1 Alur Penelitian

Penelitian ini menggunakan metode rekayasa perangkat lunak yang bertujuan untuk merancang dan membangun sebuah sistem berbasis *website* untuk otomatisasi evaluasi *source code* menggunakan kecerdasan buatan (AI). Penelitian dilakukan dengan pendekatan terstruktur dan melalui beberapa tahapan yang saling berkaitan satu sama lain. Adapun alur penelitian ini dapat dijelaskan melalui gambar dan tahapan-tahapan berikut:



Gambar 3. 1 Flowchart Alur Penelitian

1. Studi Literatur

   Pada tahap ini dilakukan pencarian dan kajian terhadap referensi yang relevan, baik dari jurnal, artikel ilmiah, maupun dokumentasi resmi teknologi yang digunakan seperti *Next.js*, *Bootstrap*, *Prisma ORM*, *PostgreSQL*, *LangChain*, *Large Language Model* (LLM) *DeepSeek* dan *Ollama*. Kajian ini bertujuan untuk memahami kebutuhan pengguna dan referensi dalam perancangan sistem otomatisasi evaluasi *source code*.

2. Analisis Kebutuhan Sistem

   Dilakukan identifikasi terhadap kebutuhan fungsional dan non-fungsional sistem. Kebutuhan ini meliputi fitur-fitur utama yang dibutuhkan oleh pengguna, seperti proses *login* untuk autentikasi, fitur unggah tugas oleh mentor untuk dikerjakan mahasiswa, fitur unggah berkas jawaban tugas

oleh mahasiswa dalam bentuk *source code*, proses evaluasi otomatis oleh AI, tampilan hasil evaluasi, serta kemampuan mentor untuk memberikan feedback manual. Sistem ini dirancang untuk memastikan bahwa hanya pengguna yang memiliki akun terdaftar yang dapat mengakses fitur sesuai perannya masing-masing.

3. Perancangan Sistem

   Merancang struktur dan arsitektur sistem yang akan dibangun. Pada tahap ini dilakukan perancangan activity diagram, perancangan basis data (ERD), serta rancangan antarmuka pengguna (*UI*).

4. Implementasi Sistem

   Tahap ini merupakan proses pembangunan sistem berdasarkan perancangan sebelumnya. Sistem dibangun menggunakan *Next.js* sebagai *framework* untuk *frontend* dan *backend*, *Bootstrap* sebagai *framework frontend* untuk membuat antarmuka pengguna responsif, *PostgreSQL* sebagai basis data, dan *Prisma* sebagai *ORM* untuk mengelola interaksi dengan *database*. Untuk proses evaluasi *source code*, sistem terintegrasi dengan *LLM DeepSeek* R1 melalui *Ollama* dan *LangChain* yang melakukan analisis otomatis terhadap tugas yang diunggah dan memberikan *feedback* serta keputusan final (benar/salah).

5. Pengujian

   Pengujian dilakukan dengan memilih 5-10 mahasiswa mata kuliah Pemrograman secara acak untuk mengisi kuisioner mengenai kelayakan sistem, kemudahan penggunaan, dan evaluasi tugas oleh AI. Hasil evaluasi AI juga akan dinilai oleh mentor (dosen mata kuliah Pemrograman) untuk memastikan kesesuaian *feedback* dengan standar akademik yang berlaku.

6. Evaluasi dan Dokumentasi

   Setelah sistem berjalan, dilakukan evaluasi terhadap performa dan fungsionalitas sistem. Selain itu, sistem didokumentasikan baik dalam bentuk laporan maupun dokumentasi teknis untuk mendukung pengembangan lebih lanjut.

3.2 Analisis Kebutuhan Sistem

Analisis kebutuhan sistem dilakukan untuk mengetahui dan merinci fitur-fitur apa saja yang dibutuhkan dalam pembangunan sistem otomatisasi evaluasi *source code* berbasis *website*. Kebutuhan sistem dibagi menjadi dua jenis, yaitu kebutuhan fungsional dan kebutuhan non-fungsional.

3.2.1 Kebutuhan Fungsional

Kebutuhan fungsional merupakan kebutuhan yang berhubungan langsung dengan fitur dan proses yang dapat dilakukan oleh pengguna. Adapun kebutuhan fungsional sistem ini adalah sebagai berikut:

1. Autentikasi Pengguna

   Pengguna (mahasiswa dan mentor) harus melakukan proses login untuk dapat mengakses sistem sesuai peran masing-masing.

2. Unggah Ujian dan Tugas oleh Mentor

   Mentor dapat mengunggah ujian yang berisi tugas-tugas yang dapat dikerjakan oleh mahasiswa.

3. Unggah Jawaban Tugas Mahasiswa

   Mahasiswa dapat mengunggah berkas *source code* sebagai bentuk pengumpulan tugas.

4. Evaluasi Otomatis oleh AI

   Sistem akan secara otomatis mengirimkan berkas source code yang diunggah ke model AI (LLM *DeepSeek-R1* via *Ollama* dan *LangChain*) untuk dianalisis.

5. Pemberian Feedback Otomatis

   Hasil analisis AI akan berupa feedback dan keputusan final apakah tugas tersebut benar atau salah yang ditampilkan kepada mahasiswa.

6. Akses Mentor ke Tugas Mahasiswa

   Mentor dapat melihat tugas yang dikumpulkan oleh mahasiswa dan hasil evaluasi yang diberikan AI.

7. Feedback Tambahan oleh Mentor (Opsional)

   Mentor memiliki opsi untuk memberikan feedback manual tambahan kepada mahasiswa jika diperlukan.

8. Penyimpanan Data ke Basis Data

   Semua data seperti informasi pengguna, tugas yang diunggah, hasil evaluasi, dan feedback akan disimpan dalam database *PostgreSQL* melalui *Prisma ORM*.

3.2.2 Kebutuhan Non-Fungsional

Kebutuhan non-fungsional adalah aspek-aspek teknis dan kualitas sistem yang harus dipenuhi agar sistem berjalan dengan baik. Adapun kebutuhan non-fungsional dalam sistem ini adalah:

1. Responsif dan Aksesibilitas

   Sistem dapat diakses melalui berbagai perangkat dengan tampilan yang responsif.

2. Performa Sistem

   Proses evaluasi tugas oleh AI harus dilakukan dalam waktu yang wajar (tidak terlalu lama).

3. Ketersediaan Sistem

   Sistem diupayakan untuk dapat diakses kapan saja (24/7), terutama saat masa pengumpulan tugas.

4. Kemudahan Penggunaan (*User Friendly*)

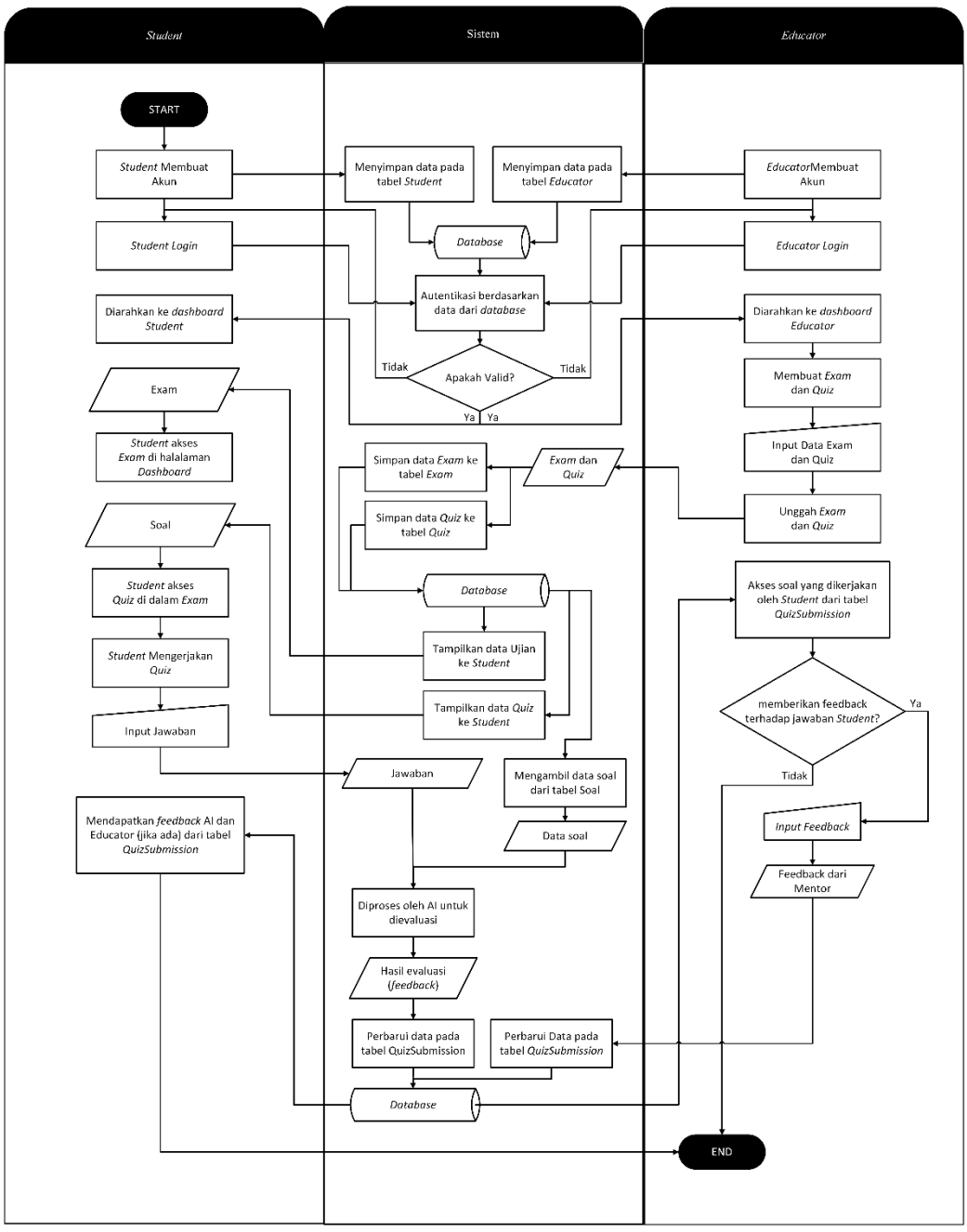
   Antarmuka sistem harus mudah dipahami dan digunakan oleh mahasiswa maupun mentor.

3.3 Perancangan Sistem

Perancangan sistem adalah tahap penting dalam pengembangan sistem berbasis *website* ini. Pada tahap ini, dilakukan perancangan yang mencakup *activity diagram*, perancangan basis data (*Entity Relationship Diagram*/ERD), serta rancangan antarmuka pengguna (UI). Semua perancangan ini bertujuan untuk memberikan gambaran yang jelas mengenai cara kerja sistem yang akan dibangun, serta hubungan antar komponen sistem.

3.3.1 Activity Diagram

Diagram ini menjelaskan secara visual proses dari sistem, mulai dari unggah tugas oleh mentor, pengumpulan tugas oleh mahasiswa, hingga proses evaluasi otomatis yang dilakukan oleh AI. Selain itu, diagram ini juga menggambarkan proses pemberian *feedback*, baik otomatis oleh AI maupun tambahan dari mentor, serta penyimpanan data ke dalam basis data secara berkala. Adapun gambar *activity diagram* sebagai berikut:



Gambar 3. 2 *Activity Diagram*

3.3.1.1 Proses Mahasiswa (*Student*) *Register* akun dan *Login*:

Proses dari *activity diagram* ini diawali dengan mahasiswa melakukan pendaftaran akun. Setelah proses registrasi selesai, mahasiswa diarahkan ke halaman login untuk memasukan data autentikasi, seperti *email*, *password* dan memili peran sebagai *Student*. Setelah itu sistem akan melakukan verifikasi terhadap data yang dimasukkan, jika data sesuai maka mahasiswa akan diarahkan ke halaman *dashboard Student*, namun jika tidak sesuai maka mahasiswa akan diarahakan kembali ke halaman *login.*

3.3.1.2 Proses Mentor (*Educator*) Register akun dan login:

Proses selanjutnya mentor melakukan pendaftaran akun dan dilanjutkan dengan memasukan data autentikasi layaknya mahasiswa di halaman yang sama, namun yang berbeda hanyalah peran yang dipilih sebagai *Educator*. Setelah itu sistem akan melakukan verifikasi terhadap data yang dimasukkan, jika data sesuai maka mahasiswa akan diarahkan ke halaman *dashboard Mentor*, namun jika tidak sesuai maka mentor akan diarahakan kembali ke halaman *login.*

3.3.1.3 Mentor Membuat Ujian (Exam) dan Soal (Quiz)

Setelah Mentor diarahkan ke halaman *dashboard* Mentor, langkah selanjutnya adalah Mentor membuat ujian dan soal-soal yang ada di dalam ujian tersebut. Setelah itu sistem akan menyimpan data ujian ke dalam tabel ujian (*Exam*) dan data soal ke dalam tabel soal (*Quiz*). Setelah data ujian dan soal tersimpan, sistem akan menampilkan data-data tersebut ke *dashboard* Mahasiswa.

3.3.1.4 Mahasiswa Mengerjakan Ujian dan Soal

Mahasiswa mulai mengerjakan soal dengan mengakses ujian yang telah dibuat oleh mentor pada halaman *dashboard* sebelumnya. Mahasiswa dapat menjawab soal dengan cara mengunggah berkas source code dan data akan disimpan oleh sistem pada tabel *QuizSubmission*. Setelah itu sistem akan mengirim data jawaban yang

dikirim oleh Mahasiswa untuk dievaluasi oleh AI pada proses selanjutnya.

3.3.1.5 Proses Evaluasi Oleh AI

Data jawaban yang telah dikirim, selanjutnya diproses oleh AI untuk menghasilkan *feedback* dan keputusan final apakah jawaban tersebut benar atau salah. Pengolahan dilakukan dengan cara mengolah jawaban berdasarkan prompt yang telah ditetapkan oleh sistem. Selain itu, proses ini juga dibantu oleh *LangChain* sebagai perantara antara sistem dan model *LLM DeepSeek-R1*. *LangChain* berperan dalam menyusun alur interaksi, mengirim *prompt* secara terstruktur, serta menerima respons dari model AI untuk kemudian diolah menjadi *feedback* dan penilaian yang relevan terhadap jawaban mahasiswa.

Setelah proses olah jawaban selesai, AI akan menghasilkan *feedback* dan keputusan benar atau salah yang nantinya akan disimpan oleh sistem untuk memperbarui data pada tabel *QuizSubmission*.

3.3.1.6 Sistem Menampilkan Hasil Feedback oleh AI

Setelah sistem mendapatkan hasil evaluasi dari AI, sistem akan memperbarui tampilan pada halaman soal baik dari Mahasiswa maupun Mentor. *Feedback* ini ditampilkan secara *real time* agar mahasiswa dapat langsung mengetahui hasil evaluasi, sementara itu mentor dapat mengakses jawaban mahasiswa dan hasil evaluasi oleh AI.

3.3.1.7 Mentor Memberikan Feedback

Proses ini bersifat opsional, jika mentor ingin memberikan *feedback*, maka sistem akan memperbarui data pada tabel *QuizSubmisison* dan menampilkan hasil *feedback* dari Mentor ke halaman soal yang diakses oleh Mahasiswa. Namun, jika mentor tidak memberikan *feedback* maka proses keseluruhan akan selesai.

3.3.2 Entity Relationship Diagram (ERD)

Diagram ini menggambarkan hubungan antar entitas utama dalam sistem, yaitu pengguna (mentor dan mahasiswa), ujian, dan tugas. Entitas

pengguna dapat berupa mentor atau mahasiswa, di mana mentor memberikan ujian berisi tugas yang kemudian dikerjakan oleh mahasiswa. Relasi antar entitas ini mendukung fungsionalitas sistem dalam mengelola data mentor, mahasiswa, ujian dan tugas secara efisien. Adapun gambar *entity relationshop diagram* sebagai berikut:

Gambar 3. 3 *Entity Relationship Diagram*

Sesuai dengan gambar *ERD* pada gambar 3.3 sistem ini terdiri dari enam entitas utama yaitu *Educator*, *Student*, *Exam*, *Quiz*, *ExamSubmission* dan *QuizSubmission*. Relasi setiap entitas ini menggambarkan fitur utama yang tersedia di dalam sistem.

Entitas *Educator* memiliki relasi dengan *Exam* yang berarti setiap Educator dapat membuat banyak *Exam* yang terdiri dari beberapa *Quiz* dan masing-masing *Quiz* memiliki beberapa atribut seperti instruksi soal, berkas *PDF,* rubrik penilaian dan batas percobaan untuk pengumpulan jawaban.

Entitas *QuizSubmission* juga dilengkapi dengan atribut *score*, *feedback* yang berfungsi untuk menyimpan umpan balik dari *Educator*, *aiNote* yang berfungsi menyimpan hasil evaluasi dari AI dan *isCorrect* untuk menentukan apakah jawaban salah atau benar. Sementara entitas *ExamSubmission* berfungsi untuk menyimpan skor akhir yang diakumulasi dari rata-rata semua nilai *Quiz* di dalamnya.

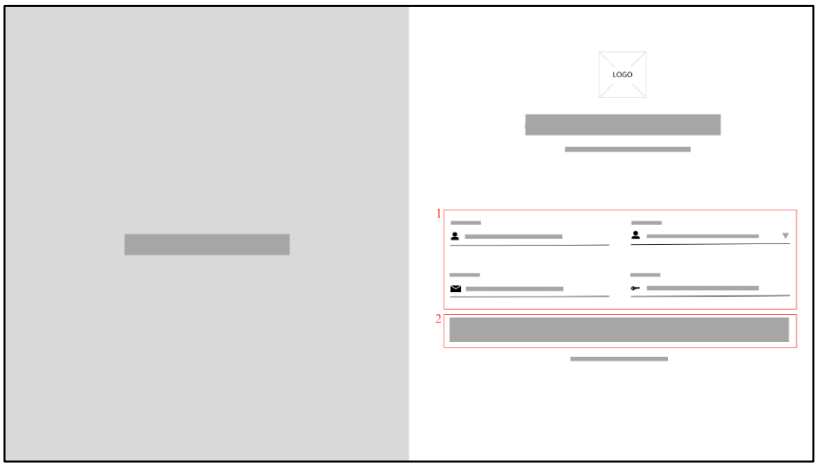
Entitas *Student* memiliki relasi dengan *ExamSubmission* dan *QuizSubmission* yang berarti setiap Student dapat mengakses dan mengerjakan *exam* dan *quiz* yang telah dibuat oleh *Educator* sebelumnya.

3.3.3 Rancangan Antarmuka Pengguna (UI)

Rancangan antarmuka pengguna disajikan dalam bentuk *wireframe* untuk menggambarkan sketsa awal dari tampilan sistem. *Wireframe* ini bertujuan untuk menunjukkan struktur halaman, posisi elemen-elemen penting, serta alur navigasi utama antar halaman. Desain difokuskan pada kesederhanaan dan kemudahan penggunaan, sehingga dapat memberikan pengalaman pengguna yang optimal sebelum masuk ke tahap implementasi desain visual secara menyeluruh. Adapun gambar *wireframe* antarmuka pengguna (UI) sebagai berikut:

3.3.3.1 Halaman Registrasi Pengguna (Mentor dan Mahasiswa)

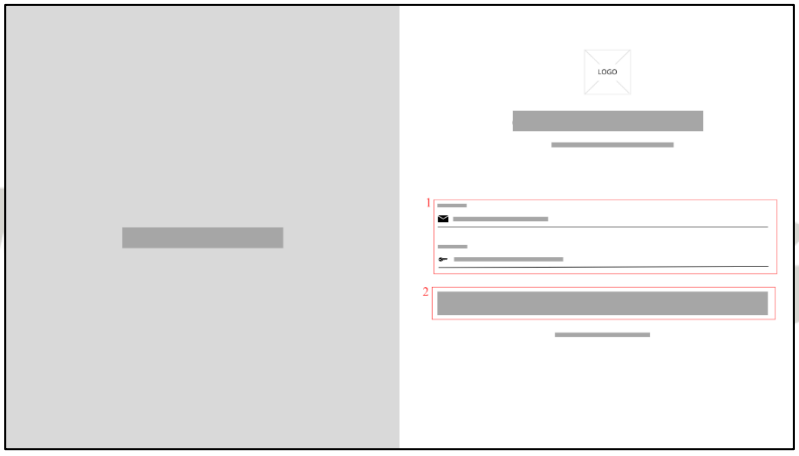
Halaman ini digunakan untuk proses pendaftaran pengguna baru ke dalam sistem. Pengguna diminta untuk mengisi beberapa informasi seperti nama lengkap, email, password, dan memilih peran (mentor atau mahasiswa) melalui *dropdown* atau opsi pilihan (kotak merah nomor 1). Setelah semua data diisi dengan benar, pengguna dapat menekan tombol registrasi (kotak merah nomor 2) untuk menyelesaikan proses pendaftaran dan menyimpan data ke dalam basis data. Berikut gambar *wireframe* untuk halaman registrasi pengguna:



Gambar 3. 4 *Wireframe* halaman registrasi

3.3.3.2 Halaman Login Pengguna (Mentor dan Mahasiswa)

Halaman ini digunakan oleh pengguna untuk masuk ke dalam sistem dengan memasukkan *email* dan *password* (kotak merah nomor 1) yang telah didaftarkan serta menekan tombol *login* (kotak merah nomor 2). Setelah berhasil *login*, pengguna akan diarahkan ke halaman *dashboard* sesuai dengan peran masing-masing. Berikut gambar *wireframe* untuk halaman *login* pengguna:

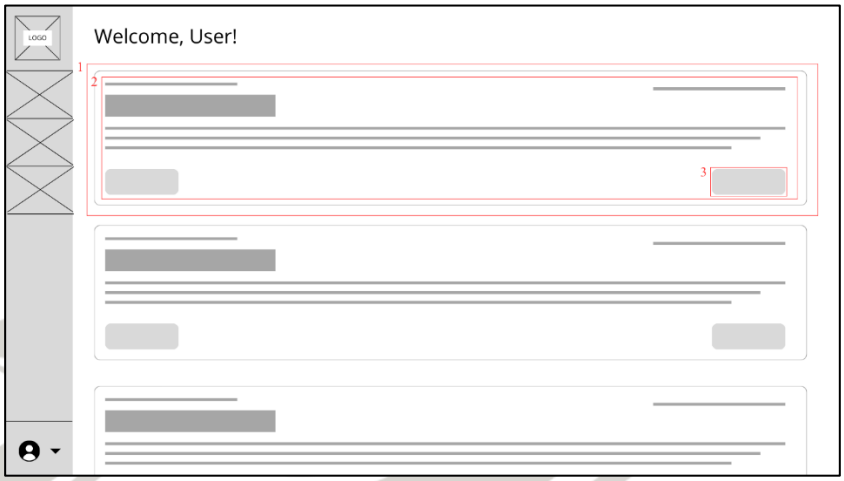


Gambar 3. 5 *Wireframe* halaman *login*

3.3.3.3 Halaman Dashboard Mahasiswa

Halaman ini ditampilkan setelah mahasiswa berhasil *login* ke dalam sistem. Pada halaman ini, mahasiswa dapat melihat daftar ujian yang telah dibuat oleh para mentor. Setiap ujian ditampilkan dalam bentuk kartu (kotak merah nomor 1) atau daftar yang mencakup

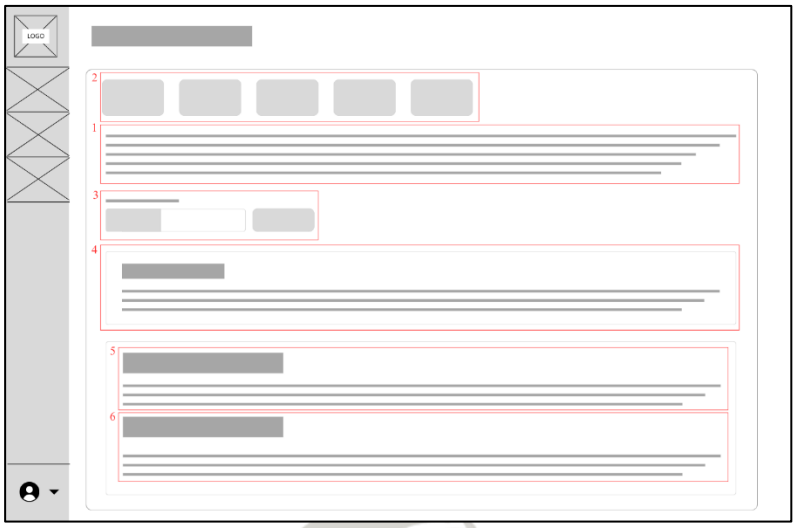
informasi seperti nama mata kuliah, nama mentor, tenggat waktu, judul ujian, deskripsi ujian, serta jumlah soal yang tersedia dalam ujian tersebut (kotak merah nomor 2). Di setiap bagian ujian juga tersedia tombol "*Start*" (kotak merah nomor 3) yang mengarahkan mahasiswa untuk memulai pengerjaan ujian secara langsung. Berikut gambar wireframe untuk dashboard mahasiswa:



Gambar 3. 6 *Wireframe* halaman *dashboard* mahasiswa

3.3.3.4 Halaman Detail Ujian Mahasiswa

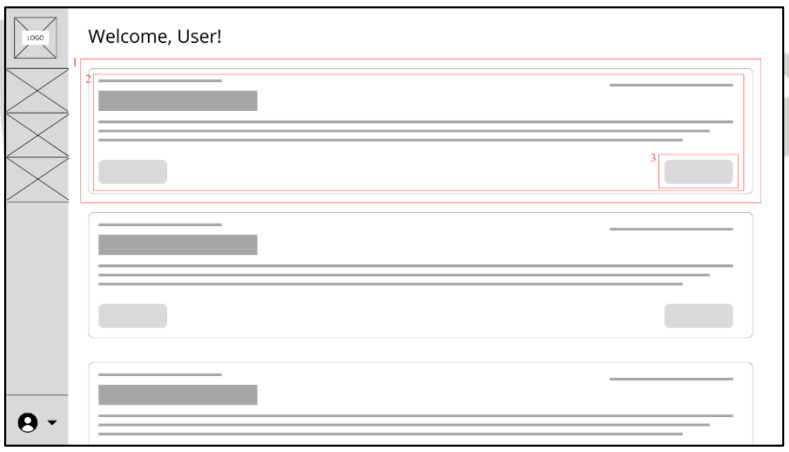
Halaman ini diakses oleh mahasiswa ketika menekan tombol "*Start*" pada salah satu ujian yang tersedia di *dashboard*. Pada halaman ini, mahasiswa dapat melihat instruksi pengerjaan (kotak merah nomor 1) serta mengakses tab nomor soal (kotak merah nomor 2) untuk berpindah antar soal ujian. Setiap soal disertai dengan kolom unggah berkas jawaban (kotak merah nomor 3). Setelah mahasiswa mengunggah file jawaban, sistem akan menampilkan *preview* isi kode dari file yang diunggah (korak merah nomor 4). Jika jawaban telah dievaluasi oleh sistem AI, maka hasil evaluasi berupa *feedback* dari AI akan ditampilkan di bawah *preview* kode (kotak merah nomor 5). Selain itu, jika mentor menambahkan catatan atau penilaian manual, maka *feedback* dari mentor juga akan ditampilkan (kotak merah nomor 6) pada bagian yang sama. Berikut gambar wireframe halaman detail ujian mahasiswa:



Gambar 3. 7 *Wireframe* halaman detail ujian mahasiswa

3.3.3.5 Halaman Dashboard Mentor

Halaman ini diakses setelah mentor berhasil login ke dalam sistem. Pada halaman ini, mentor dapat melihat seluruh ujian yang telah mereka buat. Setiap ujian ditampilkan dalam bentuk kartu (kotak merah nomor 1) atau daftar yang memuat informasi yang sama seperti pada dashboard mahasiswa (kotak merah nomor 2). Di setiap kartu ujian tersedia tombol "Detail" (kotak merah nomor 3) yang dapat ditekan untuk melihat halaman detail ujian. Berikut gambar *wireframe* halaman *dashboard* mentor:



Gambar 3. 8 *Wireframe* halaman *dashboard* mentor

3.3.3.6 Halaman Detail Ujian Mentor

Halaman ini diakses ketika mentor menekan tombol "Detail" pada salah satu ujian di halaman *dashboard*. Di halaman ini, mentor dapat melihat daftar seluruh mahasiswa yang tergabung dalam ujian tersebut. Daftar mahasiswa ditampilkan dalam bentuk kartu (kotak merah nomor 1) yang memuat informasi nama mahasiswa dan status pengerjaan (kotak merah nomor 2) beserta tombol “Detail” (kotak merah nomor 3) untuk melihat detail soal yang dikerjakan oleh mahasiswa. Status pengerjaan menunjukkan jumlah tugas yang telah dikerjakan dari total keseluruhan tugas. Berikut gambar *wireframe* halaman detail ujian mentor:

Gambar 3. 9 *Wireframe* halaman detail ujian mentor

3.3.3.7 Halaman Detail “Mahasiswa” Mentor

Halaman ini diakses ketika mentor menekan tombol "Detail" pada salah satu mahasiswa di halaman detail ujian. Pada halaman ini, ditampilkan nama mahasiswa (kotak merah nomor 1) serta daftar seluruh tugas yang sedang diproses oleh mahasiswa tersebut. Setiap tugas disajikan dalam bentuk kartu (kotak merah nomor 2) yang memuat informasi seperti status pengerjaan (telah dikerjakan atau belum) (kotak merah nomor 3), nomor tugas (kotak merah nomor 4), instruksi tugas (kotak merah nomor 5), dan *preview code* dari berkas jawaban yang telah diunggah oleh mahasiswa (kotak merah nomor 6). Selain itu, ditampilkan pula *feedback* dari AI (kotak merah nomor 7) serta tersedia form *input* untuk mentor memberikan *feedback* tambahan (kotak merah

nomor 8), disertai dengan tombol "*Submit*" (kotak merah nomor 9) untuk menyimpan masukan tersebut. Berikut gambar wireframe halaman detail “mahasiswa” mentor:

Gambar 3. 10 *Wireframe* halaman detail “mahasiswa” mentor

3.3.3.8 Halaman Tambah Ujian dan Tugas Mentor

Halaman ini memungkinkan mentor untuk membuat ujian baru dengan mengisi judul (kotak merah nomor 1), nama mata kuliah (kotak merah nomor 2), deskripsi (kotak merah nomor 3), dan tenggat waktu (kotak merah nomor 4). Mentor dapat menambahkan beberapa instruksi tugas melalui kotak dinamis (kotak merah nomor 5), masing-masing dilengkapi tombol "*Delete*" (kotak merah nomor 6) dan "Tambah Soal" (kotak merah nomor 7). Di akhir terdapat tombol "*Cancel*" dan "*Publish*" (kotak merah nomor 8) untuk membatalkan atau menyimpan ujian. Berikut gambar *wireframe* halaman tambah ujian dan tugas mentor:

Gambar 3. 11 Wireframe halaman tambah ujian dan tugas mentor

3.4 Implementasi Sistem

Pada tahap ini, sistem akan dibangun berdasarkan hasil perancangan sebelumnya. Rencana implementasi dibagi menjadi tiga bagian utama, yaitu implementasi *website*, implementasi basis data, dan implementasi kecerdasan buatan (AI) sebagai komponen evaluasi otomatis. Ketiga bagian ini akan saling terintegrasi untuk membentuk sistem yang utuh dan berfungsi sesuai dengan tujuan yang telah ditetapkan. Pengembangan dan perancangan sistem ini hanya terbatas pada pengembangan awal saja sehingga memfokuskan fitur-fitur penting yang menunjang evaluasi kode dengan AI.

3.4.1 Implementasi Website

Pengembangan website direncanakan menggunakan *Next.js* sebagai framework utama untuk menangani sisi frontend dan backend. *Next.js* dipilih karena mendukung pengembangan berbasis *TypeScript*, yang juga kompatibel dengan integrasi *LangChain* untuk pengolahan kecerdasan buatan. Dengan menggunakan *Next.js*, pengembangan dapat dilakukan secara efisien dan memudahkan pembuatan aplikasi yang dapat diakses di berbagai *platform*.

Desain antarmuka pengguna akan dikembangkan menggunakan *Bootstrap*, yang memungkinkan tampilan sistem menjadi responsif dan ramah pengguna. Fitur-fitur utama seperti *login*, daftar ujian, unggah file tugas, dan

tampilan hasil evaluasi akan diimplementasikan secara bertahap. Setiap bagian akan disesuaikan dengan kebutuhan pengguna, baik mentor maupun mahasiswa, dengan fokus pada kemudahan akses dan interaksi yang lancar.

Selain itu, *Next.js* juga menyediakan solusi yang optimal untuk pengelolaan data yang real-time, seperti tampilan daftar ujian dan *feedback* otomatis. Sistem ini akan memungkinkan mahasiswa mengerjakan ujian dan mengunggah tugas mereka, sementara mentor dapat mengelola ujian dan memberikan *feedback*. Dengan menggunakan Next.js, seluruh proses pengembangan *website* akan terintegrasi dengan baik, memberikan pengalaman pengguna yang intuitif dan responsif.

3.4.2 Implementasi Basis Data

Implementasi basis data pada sistem ini menggunakan *PostgreSQL* sebagai sistem manajemen basis data (DBMS) dan *Prisma ORM* untuk memudahkan interaksi antara aplikasi dan database. *PostgreSQL* dipilih karena kemampuannya dalam menangani data dalam jumlah besar dengan keandalan tinggi serta dukungannya terhadap transaksi yang kompleks, yang sangat penting dalam aplikasi ini yang memerlukan integritas data yang kuat.

Prisma ORM akan digunakan untuk menangani skema *database*, *query*, dan migrasi data dengan cara yang efisien dan deklaratif. Dengan Prisma, pengelolaan basis data menjadi lebih sederhana, karena dapat menggunakan *TypeScript* untuk menulis *query* secara terstruktur dan terhindar dari *SQL* yang raw. *Prisma* juga menyediakan API yang mendukung operasi basis data dengan tipe data yang aman, mengurangi kemungkinan kesalahan dalam *query* dan meningkatkan produktivitas pengembangan.

Struktur basis data dirancang untuk mendukung fitur utama sistem, seperti penyimpanan data pengguna (mentor dan mahasiswa), ujian, dan tugas. Relasi antar tabel dikelola dengan menggunakan relasi *one-to-many* dan *many-to-many*, yang disesuaikan dengan kebutuhan sistem. Selain itu, *Prisma* menyediakan fitur migrasi untuk mempermudah pembaruan struktur *database* selama pengembangan, tanpa kehilangan data yang ada.

3.4.3 Implementasi Kecerdasan Buatan (AI)

Untuk implementasi kecerdasan buatan (AI), sistem ini memilih DeepSeek R1, sebuah model Generative AI (GAI) yang dirancang untuk menghasilkan teks berbasis instruksi (prompt). DeepSeek R1 dipilih karena kompatibilitasnya dengan Ollama, serta kemampuannya dalam memahami dan memproses data secara efisien. Sebagai model generatif, DeepSeek mampu menganalisis dan membentuk respons naratif, yang sangat berguna dalam mengevaluasi jawaban mahasiswa secara kontekstual dan mendalam.

Dengan menggunakan Ollama, sistem dapat menjalankan DeepSeek R1 secara lokal tanpa biaya tambahan, yang mengurangi ketergantungan pada layanan cloud berbayar. Selain itu, Ollama memungkinkan akses lebih fleksibel dan efisien terhadap model AI besar seperti DeepSeek R1, tanpa harus khawatir tentang masalah latensi atau pembatasan penggunaan yang umum ditemukan pada layanan cloud. Penggunaan Ollama juga mengurangi beban biaya operasional dan memastikan kontrol penuh atas proses pengolahan data di sisi lokal.

Untuk menghubungkan sistem dengan model AI secara efisien, digunakan LangChain. Dengan LangChain, proses komunikasi antara sistem dan model AI menjadi lebih terstruktur, memungkinkan evaluasi otomatis yang lebih akurat dan responsif terhadap tugas yang diberikan. LangChain juga memudahkan untuk mengelola dan menghubungkan berbagai sumber daya eksternal yang diperlukan dalam analisis AI, seperti penyimpanan data dan pengolahan teks.

3.5 Pengujian

Sistem akan diuji menggunakan dua metode pengujian yang berbeda untuk memastikan fungsionalitas dan efektivitas fitur yang telah diimplementasikan. Metode pertama akan melibatkan pengujian oleh mahasiswa untuk menilai kemudahan penggunaan dan kelayakan sistem. Metode kedua akan melibatkan pengujian hasil evaluasi AI oleh mentor untuk memastikan akurasi dan kualitas feedback yang diberikan.

3.5.1 Pengujian oleh Mahasiswa Mata Kuliah Pemrograman

Pada tahap ini, 5-10 mahasiswa yang dipilih secara acak dari mata kuliah Pemrograman akan diuji coba menggunakan sistem. Setiap mahasiswa akan mengerjakan 20 tugas dengan topik yang berbeda selama periode 1 bulan. Setelah periode pengerjaan tugas selesai, mahasiswa akan diberikan kuisioner untuk menilai kelayakan dan kemudahan penggunaan sistem. Kuisioner tersebut menggunakan skala penilaian dari 1 hingga 5, di mana 1 berarti "tidak setuju" dan 5 berarti "sangat setuju". Pertanyaan dalam kuisioner akan mencakup berbagai aspek, seperti kemudahan penggunaan antarmuka, pemahaman terhadap proses evaluasi tugas oleh AI, kelayakan hasil yang diberikan, serta pengalaman pengguna secara keseluruhan. Selain itu, kuisioner juga akan mencakup bagian untuk *feedback* langsung, di mana mahasiswa dapat memberikan komentar, kendala yang dihadapi, saran perbaikan, atau aspek yang mereka anggap sudah memuaskan dalam sistem. Hasil dari kuisioner dan *feedback* ini akan digunakan untuk mengevaluasi pengalaman pengguna dan memberikan masukan untuk perbaikan sistem.

Metode evaluasi ini mencakup penggunaan *System Usability Scale* (SUS) untuk mengukur tingkat kemudahan dan kenyamanan penggunaan sistem secara keseluruhan, serta *feedback* langsung dari pengguna yang dapat memberikan wawasan lebih mendalam mengenai kelebihan dan kekurangan sistem.

3.5.2 Pengujian oleh Mentor (Dosen Mata Kuliah Pemrograman)

Pada tahap ini, pengujian terhadap hasil evaluasi yang dihasilkan oleh sistem AI dilakukan oleh mentor, yaitu dosen mata kuliah Pemrograman. Tujuan dari pengujian ini adalah untuk menilai kelayakan dan kualitas *feedback* yang diberikan oleh AI, serta memastikan bahwa *feedback* tersebut sesuai dengan standar akademik yang berlaku dalam konteks pembelajaran pemrograman. Standar akademik yang dimaksud mencakup beberapa kriteria, seperti ketepatan logika penjelasan, kesesuaian dengan materi ajar, kejelasan informasi yang disampaikan, serta nilai edukatif dari umpan balik yang diberikan. Dosen akan menilai apakah penilaian dan arahan yang diberikan oleh sistem AI benar secara akademik, relevan dengan soal yang dikerjakan

mahasiswa, serta mampu membantu mahasiswa memahami dan memperbaiki kesalahan mereka.

Selain menilai isi feedback, dosen juga akan mengevaluasi apakah sistem AI dapat mempercepat proses evaluasi tanpa mengurangi kualitas penilaian yang sesuai dengan standar akademik. Hal ini penting untuk memastikan bahwa kehadiran AI benar-benar memberikan dampak positif terhadap efisiensi proses pengajaran. Hasil dari pengujian ini akan menjadi masukan penting dalam penyempurnaan sistem, agar dapat memberikan umpan balik yang lebih tepat sasaran, akurat, dan mendukung proses pembelajaran secara efektif.

3.6 Evaluasi dan Dokumentasi

Pada tahap evaluasi dan dokumentasi, sistem yang telah dikembangkan akan dievaluasi untuk memastikan bahwa seluruh fitur dan fungsionalitasnya berjalan sesuai dengan ekspektasi dan kebutuhan pengguna. Evaluasi ini mencakup dua aspek utama, yaitu evaluasi fungsionalitas sistem dan evaluasi kinerja sistem.

3.6.1 Evaluasi Fungsionalitas Sistem

Evaluasi fungsionalitas bertujuan untuk memastikan bahwa seluruh fitur yang diimplementasikan, seperti *login*, unggah ujian dan tugas, proses evaluasi otomatis oleh AI, serta tampilan hasil evaluasi berfungsi dengan baik dan sesuai dengan tujuan awal sistem. Pengujian akan dilakukan dengan mengacu pada umpan balik yang diberikan oleh mahasiswa dan mentor, serta memastikan bahwa hasil evaluasi dan *feedback* dari AI sudah sesuai dengan standar akademik.

3.6.2 Evaluasi Kinerja Sistem

Evaluasi kinerja dilakukan untuk menilai seberapa efisien sistem dalam menjalankan tugas-tugasnya, terutama dalam hal kecepatan pengolahan data, waktu pemuatan halaman, dan kemampuan untuk menangani beban pengguna secara bersamaan. Pengujian ini bertujuan untuk mengetahui apakah sistem mampu bekerja dengan baik dalam skala yang lebih besar dan dalam kondisi yang berbeda. Beberapa aspek yang akan diuji adalah sebagai berikut:

1. *Response Time*

   Untuk mengukur waktu respons sistem, digunakan alat *Apache JMeter*. Alat ini memungkinkan simulasi banyak pengguna secara bersamaan untuk mengukur seberapa cepat sistem memberikan respons terhadap permintaan pengguna. Dengan mensimulasikan beban pengguna yang berbeda, *JMeter* memberikan data tentang waktu yang dibutuhkan sistem untuk merespons berbagai jenis interaksi pengguna, yang sangat penting untuk menilai seberapa efisien sistem dalam memberikan feedback kepada pengguna.

2. *Throughput*

   Untuk mengukur *throughput*, yaitu jumlah transaksi atau permintaan yang dapat diproses oleh server per detik, digunakan *Apache JMeter*. Alat ini dapat menguji kapasitas server dalam menangani volume trafik dalam jangka waktu tertentu. JMeter mengukur seberapa banyak permintaan yang bisa diproses dalam satu detik, sehingga memberikan gambaran kapasitas server dalam mengelola beban trafik yang tinggi.

3. *Latency*

   Wireshark digunakan untuk mengukur *latency*, yaitu waktu delay antara pengiriman dan penerimaan data antara server dan klien. *Wireshark* menganalisis paket data yang dikirimkan antara keduanya dan memberikan informasi yang detail mengenai waktu yang dibutuhkan untuk data berpindah dari server ke klien. Pengukuran ini penting untuk mengetahui apakah ada penundaan dalam komunikasi yang dapat memengaruhi pengalaman pengguna.

4. *Load Time*

   Untuk mengukur *Load Time*, atau waktu yang dibutuhkan untuk memuat halaman atau fitur tertentu pada aplikasi, digunakan *Google Lighthouse*. Alat ini memberikan metrik yang komprehensif tentang waktu yang dibutuhkan halaman untuk dimuat sepenuhnya di *browser*, termasuk kecepatan render halaman dan waktu tunggu sebelum tampilan halaman muncul. *Google Lighthouse* memberikan analisis

tentang elemen-elemen yang dapat ditingkatkan untuk mempercepat pemuatan halaman.

5. *CPU and Memory Usage*

   Untuk memantau *CPU* dan *Memory Usage*, alat yang digunakan adalah *New Relic*. *New Relic* memungkinkan pemantauan *real-time* penggunaan *CPU* dan memori server selama aplikasi dijalankan dalam kondisi beban tinggi. Alat ini memberikan data terkait bagaimana penggunaan sumber daya server, seperti CPU dan memori, dapat mempengaruhi kinerja aplikasi, serta membantu mengidentifikasi jika ada masalah terkait konsumsi sumber daya.

6. Skalabilitas

   Pengujian Skalabilitas dilakukan menggunakan *Gatling*, sebuah alat yang dapat mensimulasikan peningkatan jumlah pengguna secara bertahap untuk mengukur bagaimana aplikasi menangani lonjakan beban. *Gatling* mengukur batas kinerja aplikasi sebelum terjadi penurunan performa, memberikan gambaran sejauh mana aplikasi dapat tetap memberikan performa yang baik saat jumlah pengguna meningkat.

7. *Bandwidth Usage*

   Untuk memantau *Bandwidth Usage*, digunakan alat *Wireshark*. *Wireshark* menganalisis data yang dikirimkan antara server dan klien, mengidentifikasi penggunaan bandwidth yang tinggi atau anomali dalam komunikasi data. Alat ini sangat berguna untuk memeriksa seberapa efisien aplikasi dalam menggunakan bandwidth dan mengidentifikasi potensi masalah yang dapat menyebabkan pemakaian *bandwidth* berlebihan.

3.6.3 Dokumentasi

Dokumentasi mencakup penyusunan laporan yang menggambarkan cara kerja sistem, pemrograman yang digunakan, serta keputusan teknis yang diambil selama pengembangan sistem. Selain itu, dokumentasi juga mencakup instruksi penggunaan sistem untuk mahasiswa dan mentor. Dokumentasi ini akan mendukung pemeliharaan dan pengembangan sistem di masa depan, serta

memastikan bahwa sistem dapat digunakan dengan mudah oleh semua pengguna yang terlibat.

BAB IV
Hasil dan Pembahasan

4.1 Hasil Implementasi Sistem

Sistem yang telah dikembangkan merupakan *platform* berbasis *website* yang memungkinkan proses evaluasi otomatis terhadap *source code* mahasiswa pada mata kuliah Pemrograman. Sistem ini dirancang untuk mempermudah proses penilaian oleh dosen dan memberikan *feedback* bagi mahasiswa menggunakan *Large Language Model* (LLM) *DeepSeek-R1*, yang diintegrasikan melalui *LangChain*.

Website ini dibangun menggunakan *framework Next.js* dengan dukungan *TypeScript* dan *Bootstrap* untuk tampilan antarmuka. Selain itu, sistem ini menggunakan *PostgreSQL* sebagai basis data, dengan *Prisma ORM* sebagai alat pengelola dan penghubung antara *website* dan basis data. Autentikasi pengguna dilakukan secara terpisah untuk dua peran, yaitu mahasiswa dan mentor.

Beberapa penyesuaian dilakukan selama proses implementasi, antara lain: metode pengumpulan jawaban mahasiswa yang semula menggunakan unggah berkas, diubah menjadi editor kode langsung di dalam sistem untuk memudahkan validasi dan evaluasi secara *real-time*. Selain itu, penggunaan *Ollama* hanya diterapkan pada tahap pengujian lokal, sementara sistem *online* menggunakan *API* resmi *DeepSeek* untuk mendukung proses evaluasi kode. Perubahan ini dilakukan untuk meningkatkan efisiensi sistem, kemudahan penggunaan, dan stabilitas layanan saat digunakan secara daring.

Di tahap implementasi ini, terdapat juga penyesuaian pada struktur dan penamaan basis data untuk menyesuaikan kebutuhan aktual sistem, serta modifikasi pada *activity* diagram agar mencerminkan alur kerja sistem versi final secara akurat. Perubahan-perubahan ini dilakukan untuk meningkatkan efisiensi, kemudahan penggunaan, dan kestabilan sistem.

Untuk kebutuhan *hosting*, sistem ini awalnya direncanakan untuk dijalankan di *Vercel*, namun pada tahap implementasi dipindahkan ke *Railway* karena *Railway* menyediakan dukungan langsung untuk layanan *PostgreSQL* yang digunakan oleh

sistem ini. *Railway* memungkinkan integrasi yang lebih mudah antara server *backend Next.js* dan basis data *PostgreSQL*, sehingga mempercepat proses *deployment* dan pengelolaan sistem secara keseluruhan. Untuk keperluan akses publik, sistem ini juga telah dihubungkan dengan domain yang dibeli melalui layanan *Hostinger* dengan nama *evaluasikode.site*, agar lebih mudah diakses oleh pengguna.

Sementara itu, berkas soal dalam format *PDF* yang diunggah oleh mentor disimpan secara *online* menggunakan *Supabase Storage*. Tautan berkas dari *Supabase* digunakan sistem untuk mengambil isi soal dan ditampilkan atau diproses lebih lanjut dalam evaluasi oleh AI.

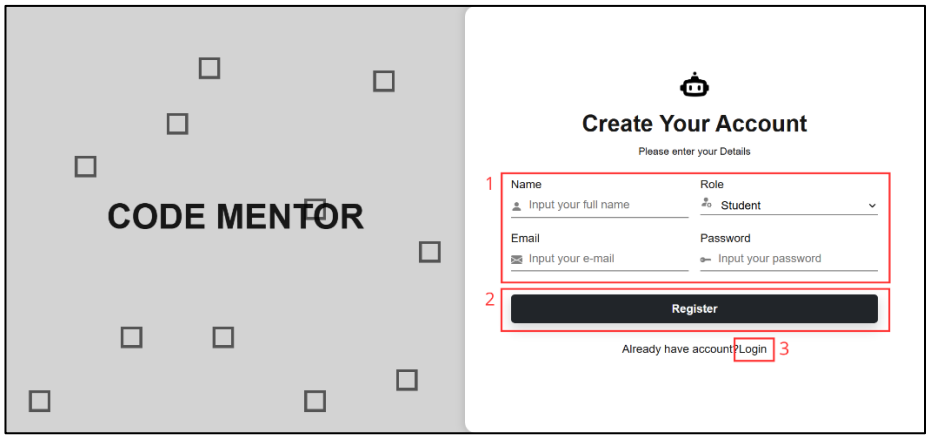
4.1.1 Implementasi Antarmuka Pengguna (UI)

Tampilan antarmuka pengguna telah dibuat sesuai dengan rancangan *wireframe* yang telah dijelaskan pada Bab III. Desain antarmuka pengguna (UI) dibuat sederhana, responsif, dan mudah digunakan baik oleh mahasiswa maupun mentor.

Meskipun secara umum mengikuti perancangan awal, terdapat beberapa penyesuaian minor selama proses implementasi untuk menyesuaikan kebutuhan aktual sistem serta meningkatkan pengalaman pengguna. Perubahan tersebut tidak mengubah alur fungsional utama, tetapi lebih bersifat penyempurnaan dari sisi tata letak, penempatan tombol, dan penambahan beberapa elemen visual pendukung.

4.1.1.1 Halaman Registrasi Pengguna (Mentor dan Mahasiswa)

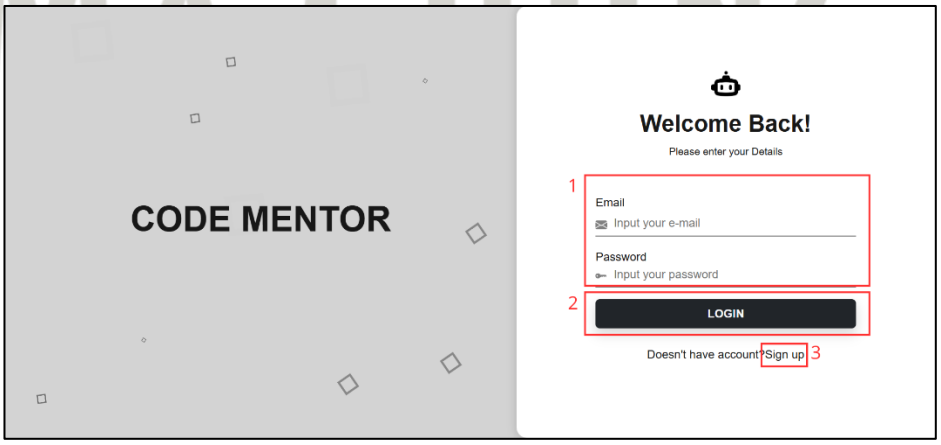
Halaman registrasi telah diimplementasikan sesuai dengan *wireframe* yang telah dirancang sebelumnya pada Bab III. Pengguna baru dapat melakukan pendaftaran dengan mengisi nama lengkap, *email*, *password*, serta memilih peran sebagai *student* atau *educator* (kotak merah nomor 1). Setelah semua data diisi, pengguna dapat menekan tombol Register (kotak merah nomor 2) untuk menyimpan data ke dalam basis data. Selain itu, ditambahkan fitur navigasi menuju halaman *login* (kotak merah nomor 3) bagi pengguna yang telah memiliki akun.



Gambar 4. 1 Implementasi Halaman Register

4.1.1.2 Halaman *Login* Pengguna (Mentor dan Mahasiswa)

Halaman login telah berhasil diimplementasikan sesuai dengan rancangan yang telah dijelaskan pada Bab III. Pada halaman ini, pengguna dapat masuk ke dalam sistem dengan menggunakan *email* dan *password* yang telah didaftarkan sebelumnya (kotak merah nomor 1). Setelah semua data terisi, pengguna dapat menekan tombol login (kotak merah nomor 2) untuk masuk ke dalam system. Tampilan akhir dari halaman login juga dilengkapi dengan tautan untuk berpindah ke halaman registrasi (kotak merah nomor 3) bagi pengguna yang belum memiliki akun. Tidak terdapat perubahan signifikan dari perancangan awal, dan seluruh komponen fungsional telah berjalan dengan baik saat sistem diuji. Berikut adalah tampilan akhir dari halaman login:

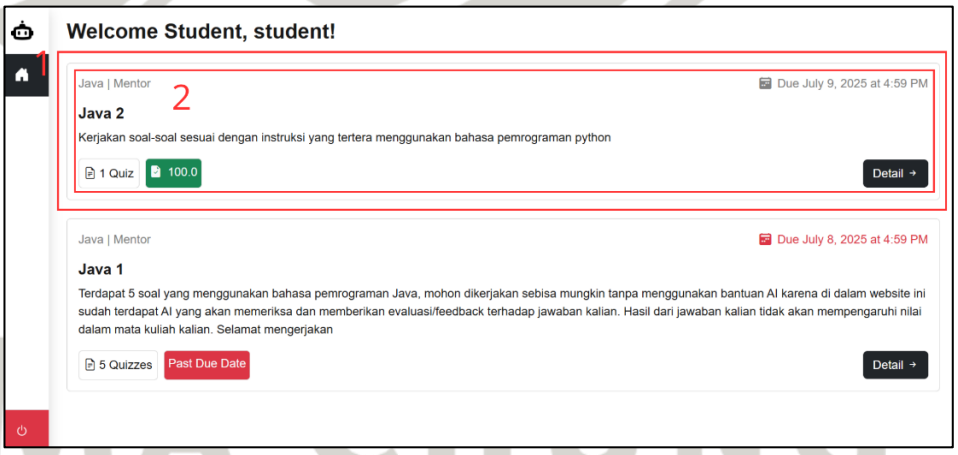


Gambar 4. 2 Implementasi Halaman Login

4.1.1.3 Halaman *Dashboard* Mahasiswa

Halaman dashboard mahasiswa telah berhasil diimplementasikan sesuai dengan perancangan sebelumnya. Setelah berhasil *login*, mahasiswa akan diarahkan ke halaman ini untuk melihat daftar ujian yang tersedia. Setiap ujian ditampilkan dalam bentuk kartu (kotak merah nomor 1) yang memuat informasi seperti nama mata kuliah, nama mentor, tenggat waktu, judul dan deskripsi ujian, serta jumlah soal yang tersedia.

Pada hasil implementasi ini, sistem juga menampilkan status ketepatan waktu pengumpulan serta tombol aksi yang menyesuaikan dengan kondisi pengerjaan. Jika ujian belum dikerjakan atau belum dinilai seluruhnya, maka tombol *Start* akan muncul. Sementara itu, jika semua soal telah diselesaikan dan dievaluasi, maka tombol akan berubah menjadi *Detail* (kotak merah nomor 2) untuk menampilkan hasil ujian. Berikut gambar implementasi untuk *dashboard* mahasiswa:



Gambar 4. 3 Implementasi Halaman *Dashboard* Mahasiswa

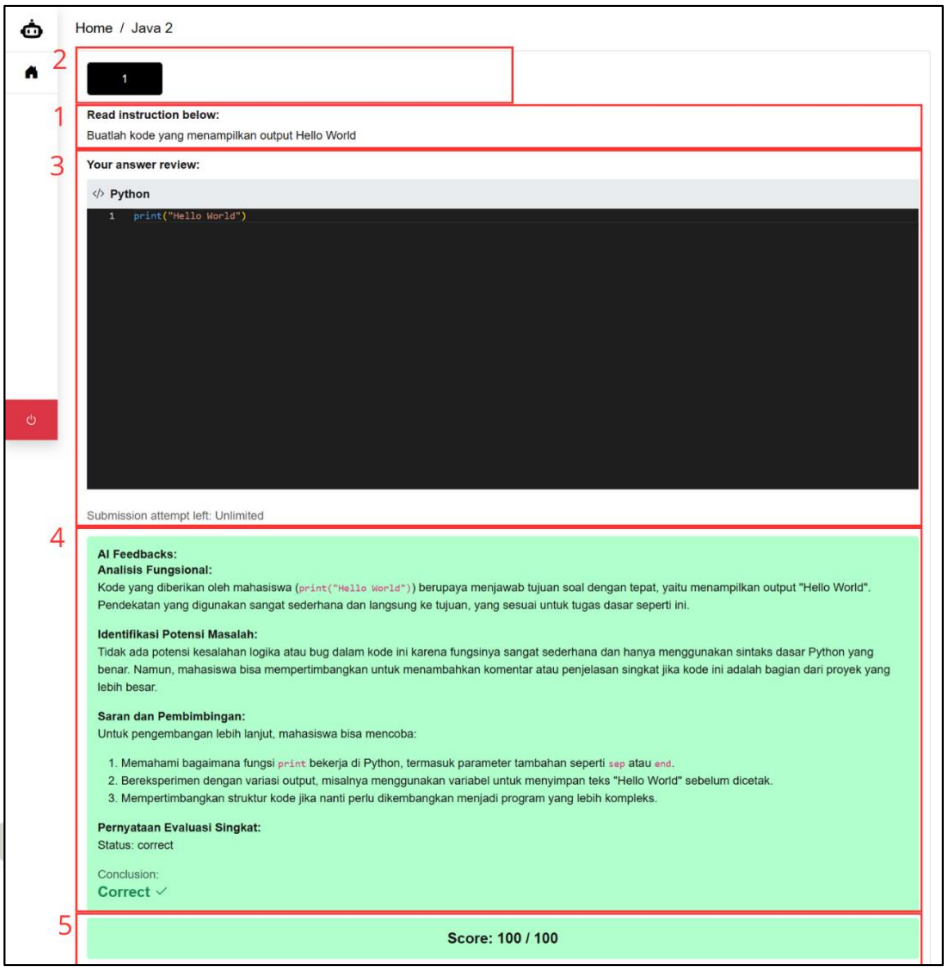
4.1.1.4 Halaman Detail Ujian Mahasiswa

Halaman detail ujian mahasiswa telah berhasil diimplementasikan dan dapat diakses oleh mahasiswa setelah menekan tombol *Start* pada salah satu ujian di *dashboard*. Pada halaman ini, instruksi pengerjaan ditampilkan dalam bentuk teks langsung maupun berkas *PDF* yang diunggah oleh mentor (kotak merah nomor 1), serta disediakan navigasi berupa tab soal (kotak merah nomor 2) untuk memudahkan mahasiswa dalam berpindah antar soal.

Terdapat beberapa perubahan dari rancangan awal yang dilakukan pada tahap implementasi. Salah satunya adalah sistem instruksi yang sebelumnya hanya berupa teks, kini ditingkatkan dengan dukungan unggah file *PDF* agar mentor dapat memberikan penjelasan yang lebih lengkap dan fleksibel. Selain itu, cara mahasiswa dalam menjawab soal juga mengalami perubahan; pada rancangan awal, jawaban dikumpulkan melalui unggahan berkas, namun pada versi implementasi, mahasiswa cukup menuliskan kode secara langsung di dalam editor kode berbasis web (kotak merah nomor 3), yang lebih praktis dan mendukung evaluasi otomatis.

Setelah jawaban dikirimkan, sistem akan memprosesnya melalui model AI untuk dievaluasi secara otomatis. Hasil evaluasi dari AI ditampilkan sebagai feedback di bawah editor, dan jika mentor memberikan catatan tambahan secara manual, maka feedback dari mentor juga akan ditampilkan pada bagian yang sama (kotak merah nomor 4).

Status kebenaran jawaban ditandai dengan warna: hijau untuk jawaban yang benar dan merah untuk jawaban yang salah. Selain itu, sistem juga menampilkan skor akhir untuk setiap soal (kotak merah nomor 5), yang dihitung berdasarkan rubrik penilaian yang telah ditentukan oleh mentor sebelumnya. Skor ini membantu mahasiswa memahami kualitas jawabannya secara kuantitatif.



Gambar 4. 4 Implementasi Halaman Detail Ujian Mahasiswa

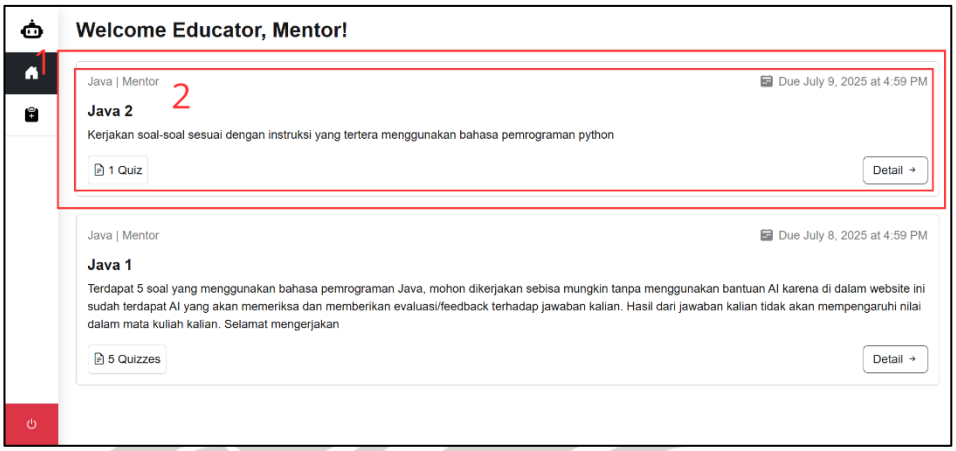
4.1.1.5 Halaman *Dashboard* Mentor

Halaman dashboard mentor telah berhasil diimplementasikan sesuai dengan desain pada tahap perancangan. Setelah berhasil *login*, mentor diarahkan ke halaman ini untuk mengelola ujian yang telah mereka buat.

Setiap ujian ditampilkan dalam bentuk kartu (kotak merah nomor 2) yang memuat informasi seperti judul, mata kuliah, jumlah soal, tenggat waktu, dan status pengerjaan oleh mahasiswa (kotak merah nomor 2). Tampilan informasi disusun secara ringkas namun informatif, menyerupai tampilan pada dashboard mahasiswa.

Selain itu, setiap kartu ujian dilengkapi dengan tombol Detail yang berfungsi untuk membuka halaman hasil pengerjaan mahasiswa

secara lebih mendalam. Seluruh elemen visual dan fungsi utama pada halaman ini telah berfungsi sebagaimana mestinya saat diuji. Berikut adalah tampilan implementasi halaman dashboard mentor:



Gambar 4. 5 Implementasi Halaman *Dashboard* Mentor

4.1.1.6 Halaman Detail Ujian Mentor

Halaman ini telah berhasil diimplementasikan untuk memungkinkan mentor melihat daftar mahasiswa yang tergabung dalam ujian. Daftar tersebut ditampilkan dalam bentuk kartu yang berisi informasi nama mahasiswa, status pengerjaan soal, serta tombol Detail untuk melihat hasil pengerjaan mahasiswa secara individual (kotak merah nomor 1).

Pada tahap implementasi, terdapat beberapa penambahan informasi yang sebelumnya tidak ada dalam rancangan awal. Selain status jumlah soal yang telah dikerjakan, sistem kini juga menampilkan nilai akhir ujian, tanggal pengumpulan, serta keterangan apakah jawaban dikumpulkan tepat waktu atau melewati tenggat waktu (kotak merah nomor 1). Penambahan ini bertujuan untuk memberikan gambaran yang lebih lengkap bagi mentor dalam menilai dan memantau progres mahasiswa.

Selain itu, sistem juga telah dilengkapi dengan fitur pencarian yang memungkinkan mentor untuk dengan mudah menemukan mahasiswa berdasarkan nama (kotak merah nomor 2). Fitur ini sangat membantu

terutama saat jumlah peserta ujian cukup banyak. Berikut adalah tampilan implementasi halaman detail ujian mentor:

Gambar 4. 6 Implementasi Halaman Detail Ujian Mentor

4.1.1.7 Halaman Detail "Mahasiswa" Mentor

Halaman ini telah berhasil diimplementasikan untuk memungkinkan mentor melihat detail pengerjaan soal dari masing-masing mahasiswa dalam satu ujian. Halaman ini dapat diakses melalui tombol Detail yang terdapat pada daftar mahasiswa di halaman sebelumnya. Pada halaman ini, ditampilkan nama mahasiswa (kotak merah nomor 1), serta daftar seluruh soal yang telah dikerjakan. Setiap soal memuat informasi seperti nomor soal (kotak merah nomor 2), instruksi pengerjaan, status pengerjaan, dan preview jawaban dalam bentuk kode yang telah dikirimkan oleh mahasiswa (kotak merah nomor 3).

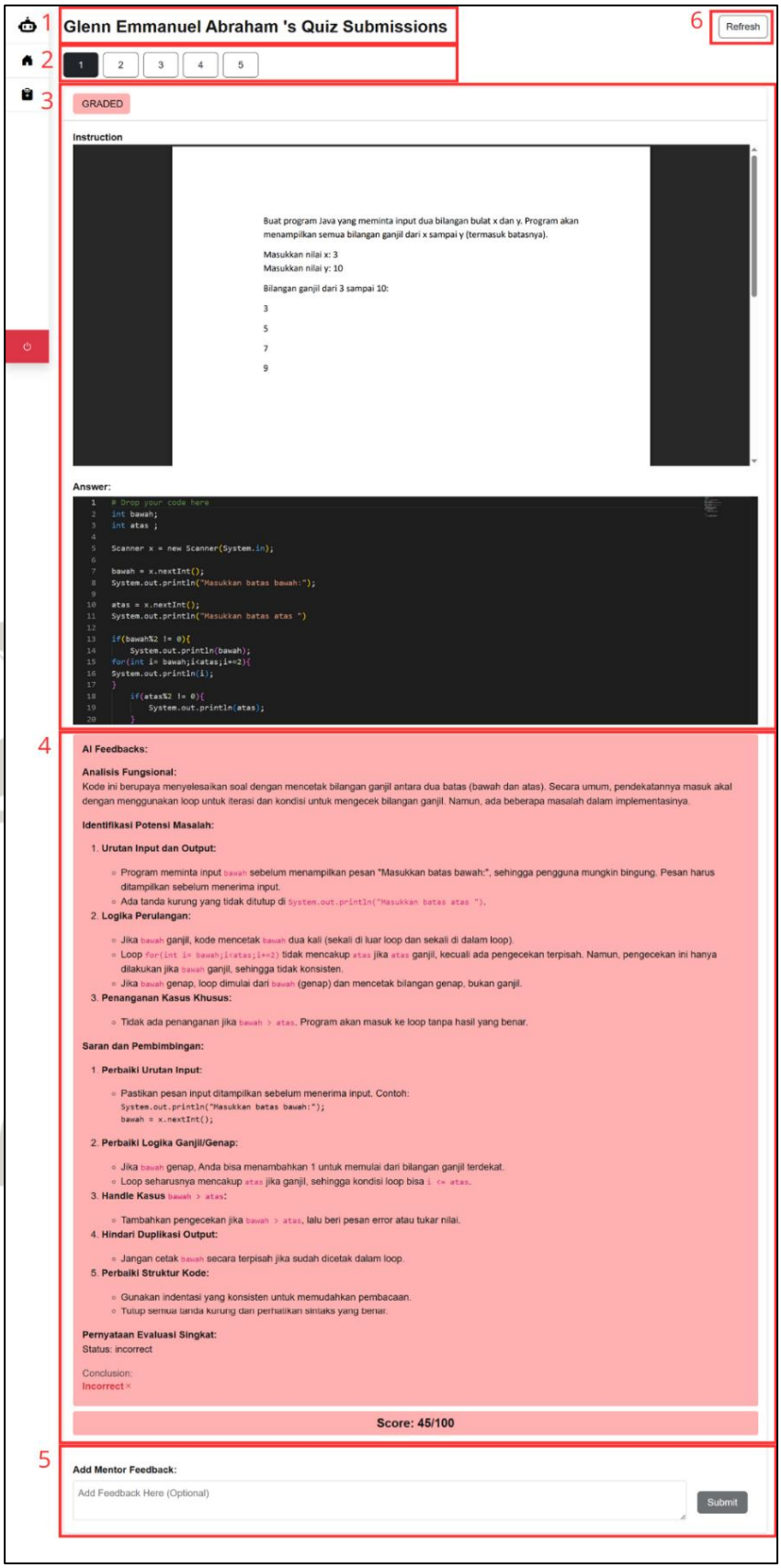
Terdapat beberapa perubahan dari rancangan awal yang diterapkan pada tahap implementasi. Pertama, soal tidak lagi ditampilkan dalam bentuk kartu vertikal seperti pada desain awal, melainkan disusun dalam bentuk navigasi berbasis tab, serupa dengan tampilan pada halaman pengerjaan ujian oleh mahasiswa. Perubahan ini dilakukan untuk menjaga

konsistensi antarmuka dan memberikan pengalaman penggunaan yang lebih efisien, terutama ketika jumlah soal cukup banyak.

Selain itu, perubahan juga dilakukan pada sistem instruksi dan cara mahasiswa menjawab. Instruksi soal kini dapat ditampilkan dalam bentuk teks maupun berkas PDF, sedangkan jawaban mahasiswa yang sebelumnya dirancang dikumpulkan melalui unggahan berkas kode, kini ditulis langsung melalui editor kode berbasis web, dan hasilnya ditampilkan dalam bentuk preview.

Sistem juga menampilkan feedback hasil evaluasi otomatis dari AI dan nilai kuantitatif yang ditentukan oleh AI berdasarkan rubrik yang ditetapkan oleh mentor (kotak merah nomor 4), serta menyediakan form input bagi mentor untuk memberikan catatan tambahan secara manual yang dapat disimpan menggunakan tombol Submit (kotak merah nomor 5).

Sebagai tambahan, telah ditambahkan tombol *Refresh* (kotak merah nomor 6) yang memungkinkan mentor memperbarui data status pengerjaan soal mahasiswa secara real-time tanpa perlu memuat ulang seluruh halaman. Fitur ini berguna untuk memantau perkembangan jawaban mahasiswa secara dinamis saat proses ujian berlangsung. Berikut adalah tampilan implementasi halaman detail “mahasiswa” mentor:



Gambar 4. 7 Implementasi Halaman Detail Mahasiswa Mentor

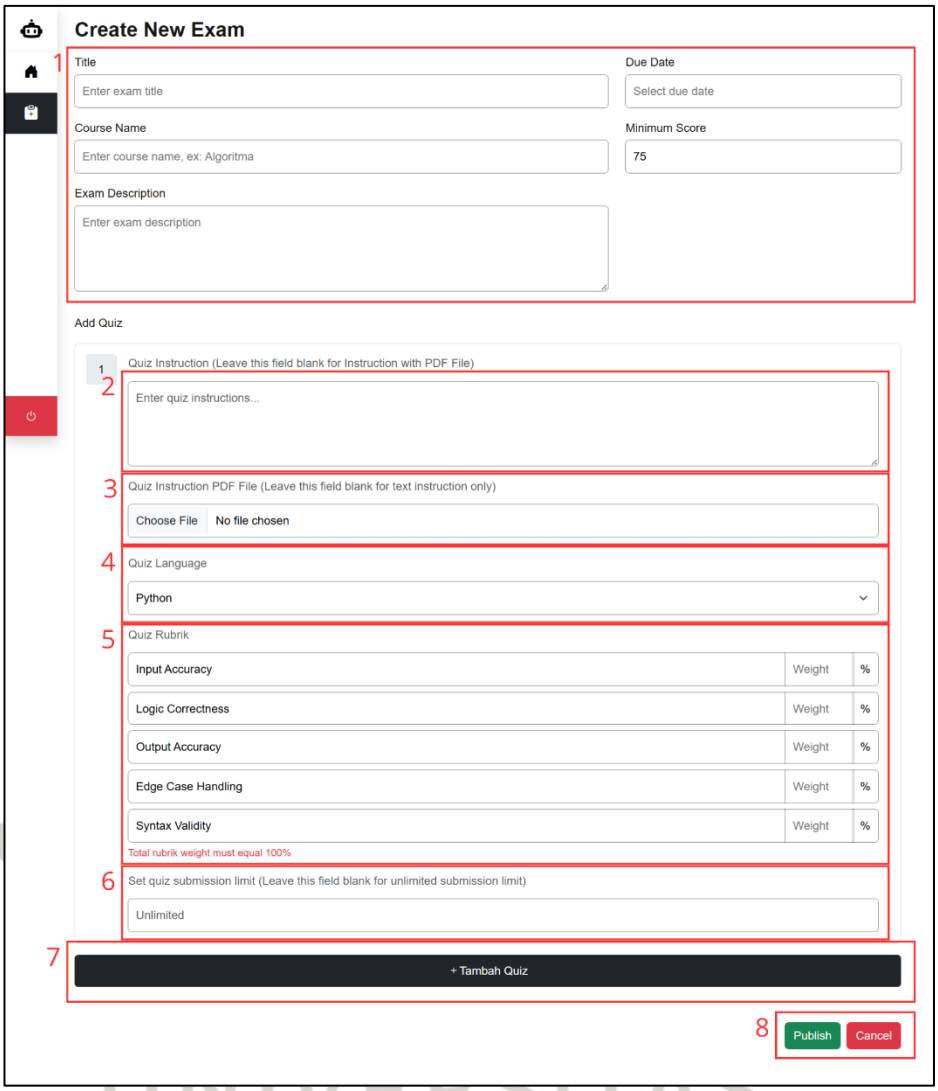
4.1.1.8 Halaman Tambah Ujian dan Tugas Mentor

Halaman tambah ujian dan tugas telah berhasil diimplementasikan dan dapat diakses oleh mentor untuk membuat ujian baru. Pada halaman ini, mentor dapat mengisi informasi utama seperti judul ujian, nama mata kuliah, deskripsi ujian, tenggat waktu dalam bentuk kalender dan nilai minimum ujian (kotak merah nomor 1).

Terdapat beberapa perubahan signifikan dari rancangan awal yang diterapkan pada tahap implementasi. Pada versi final, setiap soal (*quiz*) yang ditambahkan oleh mentor kini dilengkapi dengan beberapa form tambahan yang bersifat opsional dan fleksibel. Untuk setiap *quiz*, sistem menyediakan form untuk mengisi instruksi dalam bentuk teks (kotak merah nomor 2), serta form untuk mengunggah berkas *PDF* sebagai instruksi tambahan (kotak merah nomor 3). Kedua *form* ini dapat diisi secara bersamaan atau salah satu saja, sesuai kebutuhan.

Selain itu, ditambahkan juga *form* untuk menentukan bahasa pemrograman yang akan digunakan dalam pengerjaan soal, guna memberikan konteks yang tepat bagi sistem evaluasi otomatis. Setiap *quiz* juga dilengkapi dengan *form* isian rubrik penilaian (kotak merah nomor 4), yang berfungsi sebagai dasar bagi AI dalam mengevaluasi jawaban mahasiswa dan memberikan nilai kuantitatif, serta form untuk menentukan batas maksimal jumlah pengumpulan jawaban (kotak merah nomor 5), untuk pengumpulan tak terbatas, form ini tidak perlu diisi.

Setiap soal tetap dikelola menggunakan komponen kotak dinamis yang dapat dihapus melalui tombol Delete (kotak merah nomor 6), atau ditambah melalui tombol Tambah Soal (kotak merah nomor 7). Setelah semua data diisi, mentor dapat menekan tombol *Publish* atau *Cancel* (kotak merah nomor 8) untuk menyimpan atau membatalkan pembuatan ujian. Berikut adalah tampilan implementasi halaman tambah ujian dan tugas mentor:



Gambar 4. 8 Implementasi Halaman Tambah Ujian dan Mentor

4.1.2 Implementasi Basis Data

Basis data merupakan komponen penting dalam sistem evaluasi otomatis ini karena menjadi tempat penyimpanan seluruh data pengguna, ujian, soal, serta jawaban yang dievaluasi oleh AI. Implementasi basis data dilakukan dengan menyesuaikan kebutuhan sistem yang sebenarnya, sehingga terdapat sejumlah perubahan dari rancangan awal. Penjabaran implementasi basis data disampaikan dalam beberapa bagian berikut:

4.1.2.1 Teknologi dan Alat yang Digunakan

Dalam pengembangan sistem ini, digunakan *PostgreSQL* sebagai sistem manajemen basis data relasional. *PostgreSQL* dipilih karena bersifat *open-source*, stabil, dan mendukung relasi antar tabel secara

kompleks. Selain itu, *PostgreSQL* memiliki performa tinggi serta fleksibilitas yang baik untuk mendukung pengembangan sistem lebih lanjut.

Sebagai penghubung antara aplikasi berbasis *Next.js* dengan basis data, digunakan *Prisma ORM* (*Object-Relational Mapping*). *Prisma* merupakan *ORM* berbasis *TypeScript* yang memungkinkan pengembang mendefinisikan struktur database melalui berkas *scheme.prisma* dan melakukan interaksi data menggunakan sintaks *JavaScript*/*TypeScript* yang lebih aman dan efisien.

Dalam implementasi sistem ini, fungsionalitas *Prisma* difokuskan untuk operasi *Create* dan *Read* (membuat dan menampilkan data), khususnya untuk menyimpan data pendaftaran pengguna, data ujian, soal, serta hasil evaluasi kode. Sementara itu, operasi *Update* dan *Delete* tidak tersedia secara langsung dalam antarmuka sistem, melainkan dilakukan secara langsung melalui pengelolaan basis data di sisi *backend* untuk menjaga stabilitas dan fokus utama sistem pada pengembangan fitur evaluasi otomatis berbasis AI.

Prisma juga mendukung fitur validasi skema, pencatatan waktu otomatis (*createdAt*, *updatedAt*), serta kemampuan integrasi relasi yang mempermudah pengambilan data bersarang seperti satu ujian dengan seluruh soal di dalamnya.

Selain itu, karena sistem mendukung unggah instruksi soal dalam bentuk berkas *PDF*, digunakan *Supabase Storage* sebagai media penyimpanan *cloud* untuk menyimpan file-file tersebut secara *online*. *URL* file yang tersimpan akan disimpan dalam basis data untuk kemudian ditampilkan kepada pengguna melalui antarmuka sistem.

4.1.2.2 Struktur dan Relasi Basis Data

Struktur database yang diimplementasikan disesuaikan dengan kebutuhan sistem dalam mendukung proses evaluasi otomatis oleh AI.

Terdapat enam tabel utama yang merepresentasikan entitas penting dalam sistem, yaitu:

- ***Student***

  Menyimpan data mahasiswa yang menggunakan sistem, termasuk informasi sebagai berikut

  1. *name*: nama lengkap mahasiswa
  2. *email*: alamat email yang digunakan untuk *login*
  3. *password*: kata sandi yang telah dienkripsi
  4. *createdAt*: waktu saat akun dibuat
  5. *updatedAt*: waktu kapan terakhir akun diperbarui

- ***Educator***

  Menyimpan data mentor atau dosen yang bertugas membuat ujian dan soal, termasuk informasi seperti sebagai berikut:

  1. *name*: nama lengkap mentor
  2. *email*: alamat email yang digunakan untuk *login*
  3. *password*: kata sandi yang telah dienkripsi
  4. *createdAt*: waktu saat akun dibuat
  5. *updatedAt*: waktu kapan terakhir akun diperbarui

- ***Exam***

  Menyimpan informasi ujian yang dibuat oleh *Educator*, termasuk informasi sebagai berikut

  1. *title*: judul ujian
  2. *description*: deskripsi ujian
  3. *courseName*: nama mata kuliah ujian
  4. *startDate*: tanggal ujian dimulai
  5. *endDate*: tenggat waktu ujian berakhir
  6. *minScore*: skor minimum yang harus dicapai
  7. *createdAt*: waktu ujian dibuat
  8. *updatedAt*: waktu kapan terakhir ujain diperbarui
  9. *creatorId*: ID dari *Educator* yang membuat ujian untuk relasi

- ***Quiz***

  Menyimpan data setiap soal individual dalam satu *exam*, termasuk informasi sebagai berikut

  1. *instruction*: instruksi soal dalam bentuk teks
  2. *fileName*: menyimpan nama berkas soal yang diunggah oleh mentor
  3. *filePath*: menyimpan jalur berkas soal yang diunggah oleh mentor
  4. *fileUrl*: menyimpan tautan (URL) dari berkas instruksi soal yang telah diunggah oleh mentor ke *Supabase Storage*
  5. *rubrik*: kriteria penilaian setiap soal yang digunakan oleh AI untuk pemberian nilai kuantitatif
  6. *language*: bahasa pemrograman yang akan digunakan untuk mengerjakan soal
  7. *submission_limit*: batas jumlah percobaan *submit* jawaban
  8. *createdAt*: waktu soal dibuat
  9. *updatedAt*: waktu kapan terakhir soal dirubah

- ***QuizSubmission***

  Merupakan tabel pivot dari *quiz* untuk menyimpan informasi yang berbeda setiap mahasiswa, berisi informasi sebagai berikut:

  1. *answer*: jawaban dari setiap mahasiswa dalam bentuk *source code*
  2. *score*: skor hasil evaluasi dari AI berdasarkan rubrik
  3. *feedback*: umpan balik yang diberikan oleh mentor
  4. *isCorrect*: status benar atau salah dari AI
  5. *aiNote*: umpan balik yang diberikan oleh AI
  6. *status*: merepresentasikan tahap pengerjaan soal oleh mahasiswa. Nilai *OPEN* menunjukkan soal belum dikerjakan, *GRADING* menandakan bahwa jawaban sedang dievaluasi oleh AI, dan *GRADED* berarti proses evaluasi oleh AI telah selesai dilakukan.

7. *submission_count*: jumlah percobaan *submit* yang telah dilakukan
8. *createdAt*: waktu kapan mahasiswa *submit* jawaban pertama kali
9. *updatedAt*: waktu kapan mahasiswa *submit* ulang jawaban

- ***ExamSubmission***

  Merupakan table pivot dari *exam* untuk menyimpan informasi yang berbeda setiap mahasiswa, terdapat informasi sebagai berikut

  1. *score*: nilai akhir ujian yang diakumulasi dari rata-rata seluruh nilai soal di dalam ujian
  2. *status*: merepresentasikan seluruh tahap pengerjaan soal-soal yang ada di dalam ujian. Nilai *OPEN* jika masih ada salah satu soal yang belum dikerjakan. Nilai *GRADING* jika semua soal sudah masuk tahap penilaian. Nilai *GRADED* jika semua soal telah dievaluasi oleh AI
  3. *createdAt*: waktu kapan submisi ujian dibuat
  4. *updatedAt*: waktu kapan submisi ujian dirubah

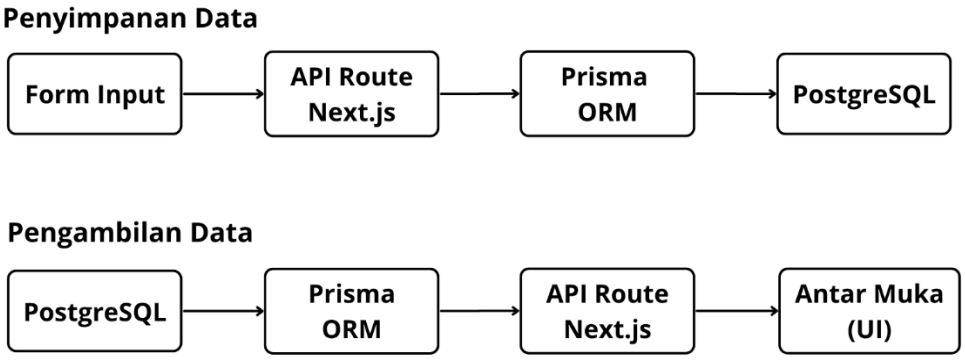
Relasi antar tabel dirancang dengan prinsip *one-to-one dan one-to-many*, dengan hubungan sebagai berikut:

- Satu *Educator* dapat membuat banyak *Exam* (*one-to-many*).
- Satu *Exam* dapat memiliki banyak *Quiz* (*one-to-many*).
- Satu *Exam* dapat memiliki banyak *ExamSubmission* dari *Student* (*one-to-many*).
- Satu *Student* dapat memiliki banyak *ExamSubmission* dan *QuizSubmission* (*one-to-many*).
- Setiap *ExamSubmission* hanya mewakili satu *Student* untuk satu *Exam* (*one-to-one*).
- Satu *Quiz* dapat memiliki banyak *QuizSubmission* dari *Student* yang berbeda (*one-to-many*).
- Setiap *QuizSubmission* hanya mewakili satu *Student* untuk satu soal (*one-to-one*).

Struktur ini memungkinkan sistem untuk secara fleksibel mencatat dan mengelola proses evaluasi mulai dari pembuatan soal hingga hasil akhir penilaian yang diberikan kepada mahasiswa. Setiap relasi antar tabel diimplementasikan secara eksplisit dalam berkas *schema.prisma*, dan disesuaikan dengan kebutuhan fitur yang dikembangkan.

4.1.2.3 Alur Penyimpanan dan Pengambilan Data

Proses penyimpanan dan pengambilan data dalam sistem ini dirancang agar terstruktur dan efisien, menggunakan pendekatan berbasis *API* yang dibangun melalui *framework Next.js*, serta pengelolaan data melalui *Prisma ORM*. Alur umum penyimpanan dan pengambilan data dalam sistem dapat digambarkan sebagai berikut:



Gambar 4. 9 Alur Penyimpanan dan Pengambilan Data

Ketika pengguna mengisi data melalui antarmuka (seperti saat registrasi, membuat ujian, atau mengirim jawaban), data tersebut dikirim ke *backend* melalui *route API Next.js*. Selanjutnya, data diproses oleh *handler* di sisi server dan disimpan ke dalam basis data *PostgreSQL* menggunakan *Prisma ORM*. *Prisma* secara otomatis memvalidasi skema data dan menyimpan informasi ke tabel yang relevan berdasarkan model yang telah ditentukan di dalam berkas schema.prisma.

Begitu pula saat sistem perlu menampilkan data kepada pengguna (misalnya daftar ujian, soal, atau hasil evaluasi), *backend* akan mengambil data dari *PostgreSQL* melalui *Prisma* menggunakan metode seperti *findUnique*, *findMany*, atau *query* relasional yang menggabungkan

beberapa tabel. Prisma mendukung relasi antar entitas, sehingga memungkinkan pengambilan data bersarang (*nested*), seperti satu ujian beserta daftar soal (*Quiz*) atau semua jawaban mahasiswa dalam suatu ujian.

Dengan alur ini, seluruh proses *input* dan *output* data dapat dilakukan secara *real-time* dan aman. Sistem juga memungkinkan pembacaan data secara relasional, contohnya menampilkan seluruh *Exam* yang dibuat oleh satu *Educator*, atau menampilkan *QuizSubmission* milik mahasiswa tertentu beserta informasi evaluasinya.

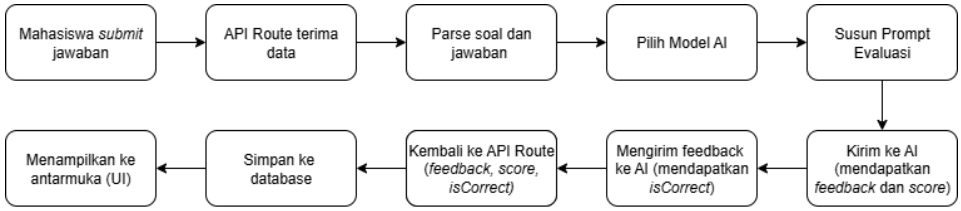
4.1.3 Implementasi Kecerdasan Buatan (AI)

Sistem ini dirancang untuk memberikan evaluasi otomatis terhadap jawaban mahasiswa dalam bentuk *source code*. Sistem ini menggunakan teknologi *Large Language Model* (*LLM*) *Deepseek-R1*, yang terhubung melalui *framework LangChain* sebagai jembatan antara sistem dan AI. Proses evaluasi dilakukan secara otomatis setelah mahasiswa memberikan jawaban melalui antarmuka (UI) sistem. Jawaban tersebut bersama dengan soal, rubrik penilaian dan bahasa pemrograman dari setiap quiz dikirim ke server (API *Route*) dan diolah menjadi *prompt* khusus yang telah ditetapkan sebelumnya, lalu dieksekusi oleh model AI yang telah dipilih (dalam hal ini menggunakan *deepseek-chat*).

Terdapat *prompt* yang akan digunakan untuk menghasilkan *feedback* dan skor kuantitatif terhadap jawaban mahasiswa. Hasil dari proses ini kemudian kembalikan ke sistem, kemudian disimpan ke basis data, dan ditampilkan ke mahasiswa serta mentor melalui antarmuka pengguna dalam bentuk:

- *feedback*: umpan balik dari AI
- *score*: nilai yang ditentukan oleh AI berdasarkan rubrik penilaian

Alur lengkap dari proses pengiriman jawaban oleh mahasiswa, evaluasi otomatis oleh AI, hingga mendapatkan hasil dari AI ini dapat dilihat pada diagram berikut:



Gambar 4. 10 Diagram Alur Implementasi Kecerdasan Buatan

4.1.3.1 Mahasiswa Submit Jawaban

Tahap ini merupakan langkah awal sebelum AI menerima data-data yang dibutuhkan (answer, *instruction*, *rubrik* dan *language*). Pada tahap ini, mahasiswa mengerjakan soal yang tersedia dan mengirimkan jawaban dalam bentuk *source code* pada *teks editor* berbasis *web* yang telah disediakan pada setiap soal.

Saat mahasiswa menekan tombol *submit*, sistem akan mengirim beberapa data yang dibutuhkan oleh AI yaitu:

- *answer*: jawaban *source code* dari mahasiswa
- *quiz_id*: id dari *quiz*
- *exam_id*: id dari *exam*
- *student_id*: ide dari *student*

Seluruh data tersebut dikirim ke backend melalui *API route Nex.js* dan akan diteruskan ke tahap selanjutnya untuk diproses oleh sistem.

4.1.3.2 API Route Terima Data

Pada tahap ini, sistem menerima *payload* yang berisi data-data yang dikirim dari antarmuka sebelumnya (*answer*, *quiz_id, exam_id* dan *student_id*). Data ini belum mencakup informasi soal secara lengkap, sehingga sistem perlu mengambil detail soal berdasarkan *quiz_id* melalui proses *query* ke basis data. Informasi yang diambil mencakup instruksi soal dalam bentuk teks (*instruction*), tautan ke berkas PDF jika tersedia (*fileUrl*) dari *Supabase Storage*, rubrik penilaian (*rubrik*), dan bahasa pemrograman (*language*).

Setelah data soal berhasil diperoleh, sistem akan memeriksa sumber instruksi soal. Jika instruksi hanya berupa teks, maka dapat langsung digunakan. Namun, jika tersedia dalam bentuk berkas *PDF*, sistem akan meneruskan data tersebut ke tahap selanjutnya untuk diproses menggunakan *parser PDF*. Dengan demikian, data instruksi akan dipersiapkan secara lengkap sebelum digunakan dalam penyusunan *prompt* evaluasi oleh AI pada tahapan berikutnya.

4.1.3.3 Parse Soal

Pada tahap ini, sistem memproses instruksi soal yang tersedia dalam bentuk berkas *PDF* untuk dikonversi menjadi teks dengan mengambil tautan dari *fileUrl* yang berada di basis data. Proses ini diperlukan agar isi soal dapat dipahami oleh model AI yang hanya menerima input dalam format teks. Sistem menggunakan library bernama *pdf-parse* untuk mengekstrak isi dari berkas *PDF*.

Jika instruksi soal juga tersedia dalam bentuk teks biasa (instruction), maka sistem akan menggabungkan hasil ekstraksi *PDF* dengan teks tersebut menjadi satu konten soal utuh (*question*). Dengan demikian, baik soal dalam bentuk file PDF maupun teks akan diproses secara seragam sebelum dikirimkan ke tahap pemilihan model AI. Langkah ini memastikan bahwa model AI menerima konteks soal yang lengkap dan relevan.

4.1.3.4 Pemilihan Model AI

Pada tahap ini sistem melanjutkan ke tahap pemilihan model AI untuk mengevaluasi jawaban yang diberikan. Pada implementasi ini, sistem menggunakan model dari provider *DeepSeek-R1*, dengan nama model *deepseek-chat*. Model ini dipanggil melalui koneksi API resmi yang disediakan oleh *DeepSeek*, dengan menyertakan *API key* pribadi dan *baseUrl* yang resmi dari dokumentasi *API DeepSeek* yang telah diatur pada konfigurasi lingkungan (*.env*).

Untuk menjaga konsistensi dan ketepatan jawaban dalam konteks akademik, model dikonfigurasi dengan *temperature* sebesar 0.0, sehingga

respons yang dihasilkan bersifat konsisten dan tidak terlalu acak. Model inilah yang akan menerima *prompt* evaluasi dan mengembalikan feedback, skor, serta status kebenaran dari jawaban mahasiswa.

4.1.3.5 Penyusunan *Prompt* Evaluasi

Setelah model AI dipilih, sistem menyusun *prompt* yang akan dikirim ke model untuk mengevaluasi jawaban mahasiswa. *Prompt* disusun menggunakan teknik *few-shot* karena teknik ini memberikan hasil yang lebih bagus dalam konteks pemrograman dan ditulis dalam format *Markdown*, sebuah format teks ringan yang umum digunakan untuk penulisan terstruktur seperti penebalan teks (****bold****), blok kode (```), *heading* (###), dan lain-lain. Penggunaan *Markdown* bertujuan agar model AI dapat mengenali struktur teks secara lebih baik, memisahkan instruksi, soal, kode, dan bagian penting lainnya secara eksplisit. Dengan struktur yang jelas, model lebih mudah memberikan jawaban yang relevan dan terarah sesuai harapan sistem.

Prompt ini digunakan untuk meminta model AI bertindak sebagai mentor pemrograman yang akan memberikan penilaian terhadap kode mahasiswa. Instruksi disusun dalam Bahasa Indonesia dan mencakup elemen-elemen penting seperti *language, question, rubrik* dan *code*. Model diminta untuk melakukan analisis menyeluruh, mulai dari tinjauan fungsional, identifikasi kesalahan, pemberian saran perbaikan, hingga menyatakan apakah kode benar atau salah dalam satu kalimat. Di akhir *prompt*, sistem juga menginstruksikan model untuk memberikan skor numerik (contoh Skor: 85) yang akan digunakan sebagai penilaian kuantitatif oleh sistem. Berikut adalah *prompt* yang digunakan oleh sistem:

```
`Anda adalah mentor evaluasi kode untuk tugas pemrograman dalam
 Bahasa Indonesia.

Tugas Anda adalah menentukan apakah kode yang diajukan mahasiswa
sudah benar atau salah berdasarkan instruksi soal yang diberikan.

Gunakan rubrik dan contoh di bawah ini sebagai referensi
penilaian:

### RUBRIK PENILAIAN:
```

{rubrik}

CONTOH PENILAIAN 1

QUESTION:

Buat fungsi is_even(n) yang mengembalikan True jika n genap.

STUDENT CODE:

\`\`\`python

def is_even(n):

    return n % 2 == 0

\`\`\`

FEEDBACK:

1. Fungsi menggunakan pendekatan modulus yang tepat.

2. Tidak ditemukan bug atau kesalahan logika.

3. Disarankan menambahkan komentar atau validasi tipe data untuk produksi.

Penilaian:

\`\`\`

Input Accuracy (20/20): Parameter n sesuai dan tidak ada input yang salah ditangani.

Logic Correctness (20/20): Logika modulus tepat untuk menentukan genap.

Output Accuracy (20/20): Output sesuai dengan soal untuk berbagai input.

Edge Case Handling (20/20): Input seperti 0 atau angka besar tetap berfungsi.

Syntax Validity (20/20): Tidak ada kesalahan sintaks. Kode dapat dijalankan.

\`\`\`

Score: 100

CONTOH PENILAIAN 2

QUESTION:

Buat fungsi factorial(n) untuk menghitung faktorial bilangan n.

STUDENT CODE:

\`\`\`python

def factorial(n):

```
            result = 1
            for i in range(n):
                result *= i
            return result
        \`\`\`
```

FEEDBACK:

1. Perulangan mulai dari 0 menyebabkan hasil faktorial menjadi nol untuk n > 0.

2. Seharusnya gunakan range(1, n+1).

3. Perbaiki logika perulangan agar sesuai dengan definisi faktorial.

Penilaian:

\`\`\`

Input Accuracy (20/20) : Parameter n diterima dengan benar.

Logic Correctness (10/20) : Kesalahan utama pada logika perulangan dari 0.

Output Accuracy (10/20) : Output salah karena result selalu 0 untuk n > 0.

Edge Case Handling (10/20) : Kasus n = 0 atau n < 0 tidak ditangani dengan benar.

Syntax Validity (20/20) : Tidak ada kesalahan sintaks. Kode valid secara struktur.

\`\`\`

Score: 70

SEKARANG, EVALUASI BERIKUT INI:

QUESTION:

{question}

SUBMITTED STUDENT'S CODE:

\`\`\`

{code}

```
\`\`\`

Tanggung jawab Anda:

1. **Analisis Fungsional:** Tinjau apakah kode tersebut berupaya menjawab seluruh tujuan dari soal, dan apakah pendekatannya secara umum masuk akal.

2. **Identifikasi Potensi Masalah:** Temukan potensi kesalahan logika, bug, atau kekurangan, **tanpa memberikan solusi kode secara langsung**.

3. **Saran dan Pembimbingan:** Berikan saran berbentuk bimbingan atau arahan, seperti mentor manusia: dorong mahasiswa untuk berpikir ulang tentang logika mereka, pertimbangkan struktur alternatif atau cek bagian tertentu dari kodenya.

4. **Penilaian Berdasarkan Rubrik:** berikan penjabaran nilai untuk setiap aspek rubrik tersebut tanpa persentase, dalam bentuk list bernomor.

Jelaskan alasan setiap skor dalam format:

**[Nama Aspek Rubrik] ([skor]/[maksimum])** : [penjelasan singkat]

### Output tambahan (untuk sistem):

- Di baris terakhir, berikan skor numerik berdasarkan rubrik di atas dengan format: \`Score: 85\` (tanpa penjelasan).

- Skor ini akan digunakan sistem untuk penilaian dan tidak ditampilkan ke mahasiswa.

### Catatan penting:

- Hindari memberikan solusi langsung atau kode jawaban yang benar.

- Tulis dalam Bahasa Indonesia yang sopan, edukatif, dan mudah dipahami.

`
```

4.1.3.6 Mengirim ke AI (Mendapatkan Feedback dan Score)

Pada tahap ini, sistem mulai menjalankan proses evaluasi otomatis dengan mengirimkan data-data yang dibutuhkan oleh AI. Data yang dibutuhkan meliputi *answer (code), question, rubrik, language* dan *model.* Semua data ini disusun ke dalam satu *prompt* evaluasi utama menggunakan bantuan *library PromptTemplate* dan *RunnableSequence* dari *LangChain*.

LangChain memfasilitasi pembuatan *pipeline* evaluasi dengan cara menggabungkan *template prompt,* model AI, dan *parser* dalam sebuah hasil dalam sebuah *chain*. Pada implementasi ini, digunakan *RunnableSequence* yang berisi tiga tahap utama:

1. *PromptTemplate.fromTemplate*(...) untuk menyusun isi *prompt* secara dinamis.
2. *Model AI (ChatOllama* atau *ChatDeepSeek)* yang akan menerima dan memproses *prompt*.
3. *StringOutputParser()* yang akan mengambil hasil keluaran (dalam hal ini *feedback*) dari model dalam bentuk string murni.

Setelah *prompt* dieksekusi dan AI memberikan hasil respons berupa teks panjang berisi evaluasi (*feedback*), sistem akan melakukan proses pemangkasan dari hasil tersebut. Hal ini dilakukan untuk membuang tag <think></think> yang terkadang muncul sebagai *inner thought* model sehingga mendapatkan *feedback* yang murni. Namun di dalam *feedback* tersebut masih terkandung kata *Score:* [angka], maka sistem melakukan pemangkasan Kembali untuk mengambil [angka] tersebut yang akan digunakan untuk mendapatkan nilai kuantitatif dari AI. Maka dalam hal ini sistem telah mendapatkan data yang dibutuhkan yaitu *feedback* dari AI dan *score* kuantitatif.

4.1.3.7 Mengirim Feedback ke AI (mendapatkan isCorrect)

Setelah mendapatkan *feedback* dan *score* dari hasil evaluasi awal, sistem akan melakukan satu kali permintaan tambahan ke model AI untuk menentukan status kebenaran jawaban mahasiswa. Pada tahap ini, sistem mengirimkan *feedback* hasil evaluasi sebelumnya ke dalam AI dalam bentuk *prompt* khusus yang berfungsi untuk menyimpulkan apakah *source code* dari mahasiswa benar atau salah berdasarkan *feedback* yang didapatkan sebelumnya.

Prompt yang digunakan bertujuan untuk membuat model bertindak sebagai *automated code judge* yang hanya memberikan keluaran dalam bentuk bilangan bulat 1 atau 0. Angka 1 menandakan

bahwa jawaban mahasiswa dinilai benar (*correct*), sedangkan angka 0 berarti salah (*incorrect*). *Prompt* ini telah dirancang sedemikian rupa agar model hanya fokus pada isi *feedback* sebelumnya tanpa memberikan penjelasan tambahan. Dengan adanya tahap ini, sistem dapat menyimpan status kebenaran jawaban mahasiswa dalam bentuk *Boolean* (*true*, jika *isCorrect* = 1 dan *false*, jika *isCorrect* = 0).

4.1.3.8 Kembali ke API Route (feedback, score, isCorrect)

Setelah proses evaluasi selesai dan sistem memperoleh hasil berupa *feedback*, *score* dan *isCorrect* dari AI, seluruh data tersebut akan dikembalikan ke API *route*. Pada tahap ini, API *route* bertugas untuk memproses hasil evaluasi dan menyimpannya ke dalam basis data pada tahap selanjutnya, agar dapat ditampilkan kembali pada antarmuka mahasiswa dan mentor sesuai kebutuhan sistem.

4.1.3.9 Simpan ke Database

Setelah *API route* menerima hasil evaluasi dari AI, sistem akan melanjutkan dengan menyimpan dan memperbarui data ke dalam tabel *quizSubmission*. Beberapa atribut yang diperbarui meliputi:

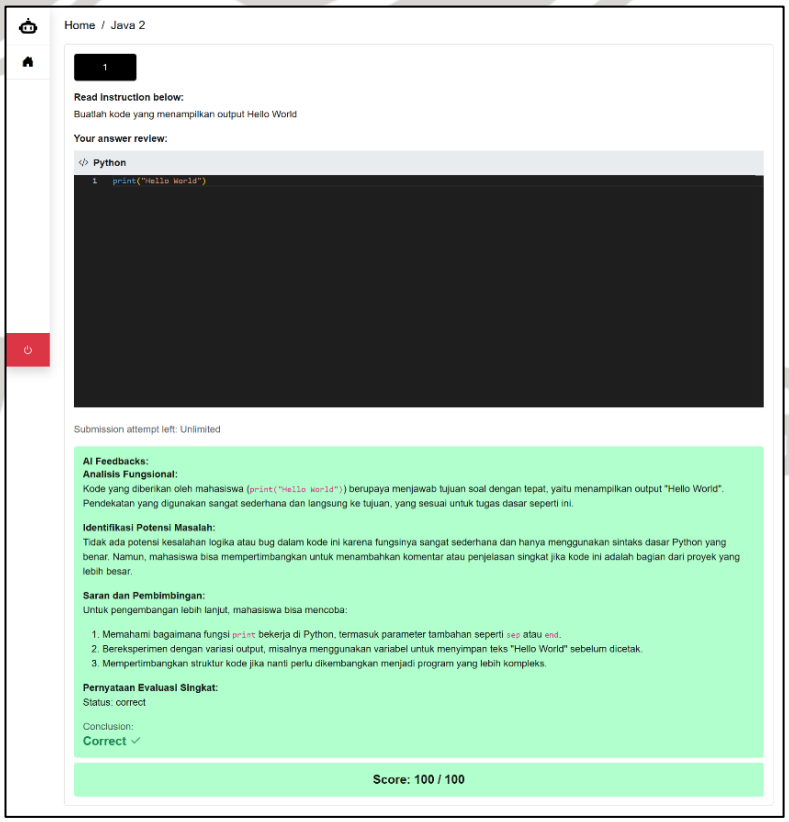
1. *status* yang diubah menjadi *GRADED* sebagai penanda bahwa soal telah selesai dievaluasi
2. *aiNote* yang menyimpan *feedback* dari AI dalam bentuk teks
3. *isCorrect* yang berisi nilai *boolean* hasil penilaian akhir (true untuk 1, false untuk 0)
4. *score* sebagai nilai kuantitatif hasil evaluasi
5. serta *updatedAt* yang secara otomatis diisi ulang sebagai penanda waktu terbaru mahasiswa mengumpulkan atau mengerjakan soal.

4.1.3.10 Menampilkan ke Antarmuka (UI)

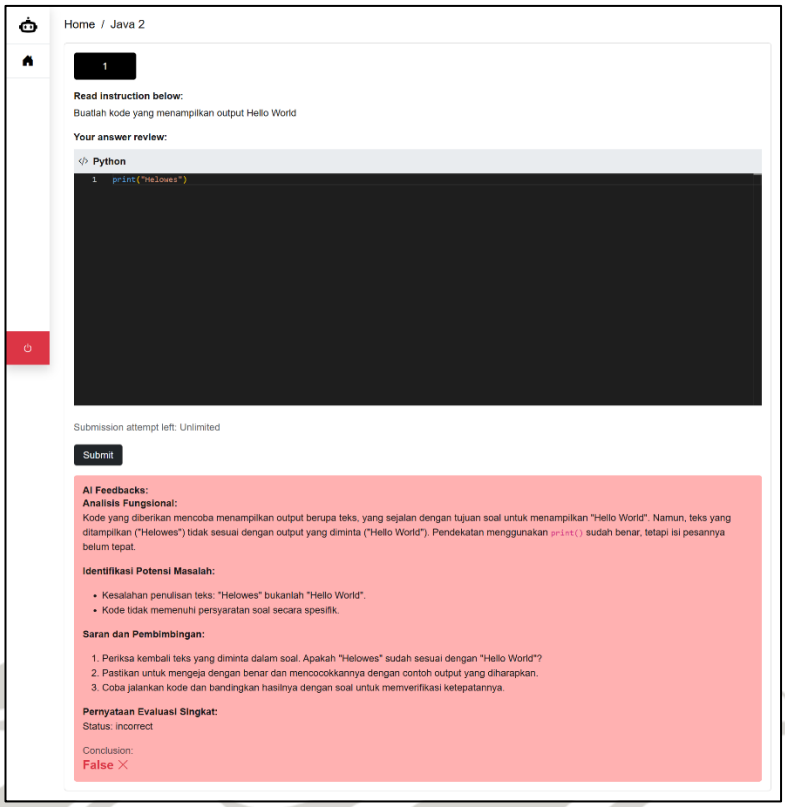
Setelah proses evaluasi selesai dan hasil disimpan ke basis data, data tersebut akan dikirim kembali ke antarmuka pengguna untuk ditampilkan secara langsung. Tampilan ini diperbarui secara otomatis setelah proses penilaian oleh AI selesai dilakukan di *backend*. Dengan integrasi ini, mahasiswa tidak hanya menerima nilai, tetapi juga

mendapatkan umpan balik yang bersifat edukatif sebagai bahan refleksi dan peningkatan kemampuan. Terdapat dua jenis tampilan antarmuka yang menerima hasil evaluasi ini, yaitu antarmuka mahasiswa dan antarmuka mentor.

Pada antarmuka mahasiswa, sistem menampilkan hasil evaluasi AI berupa *feedback* naratif yang berisi ulasan, analisis, dan saran perbaikan terhadap *source code* yang dikumpulkan. Selain itu, sistem juga menampilkan skor kuantitatif serta indikator visual untuk memperjelas status jawaban: warna hijau menunjukkan jawaban benar (*isCorrect = true*), sedangkan merah menandakan jawaban salah (*isCorrect = false*). Tampilan ini dirancang agar mahasiswa dapat memahami hasil penilaian secara cepat dan intuitif, serta terdorong untuk belajar dari kesalahan yang terjadi. Berikut adalah tampilan antarmuka mahasiswa untuk jawaban benar dan salah:

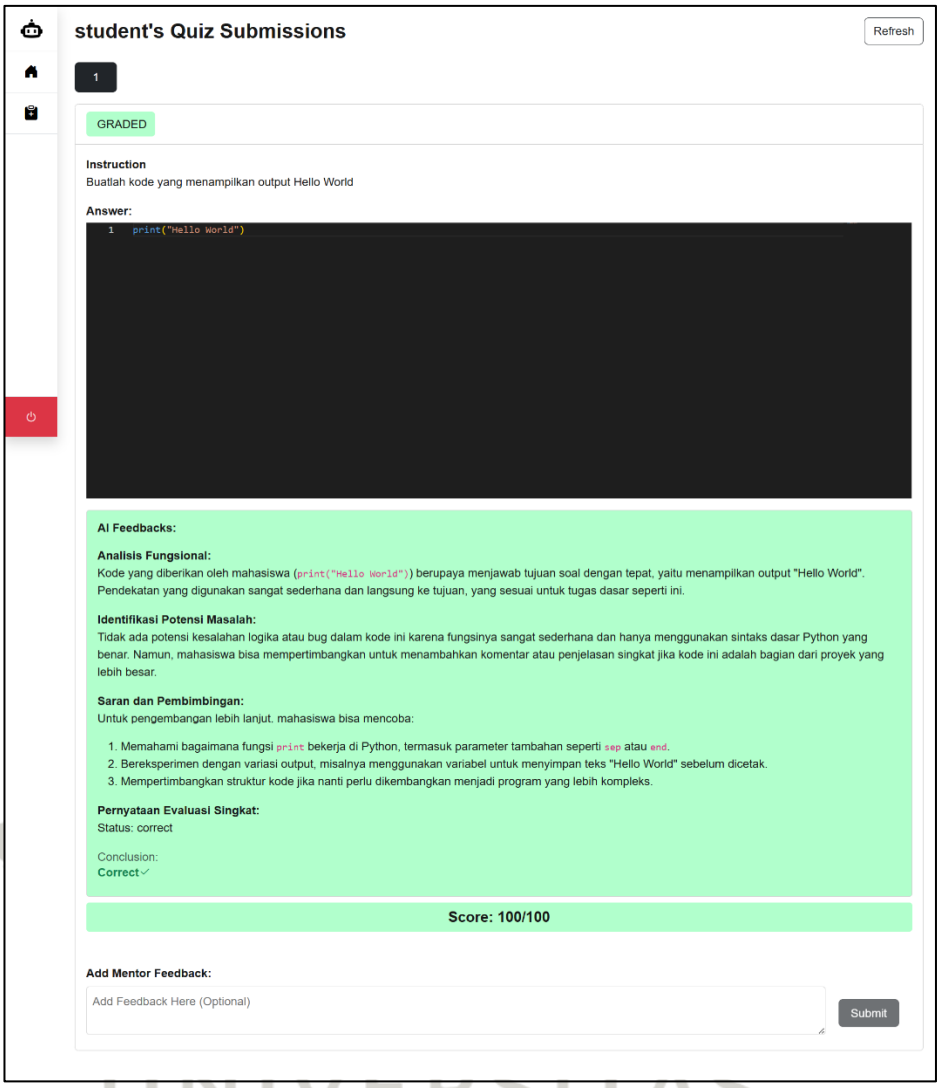


Gambar 4. 11 Hasil Evaluasi oleh AI Jawaban Benar

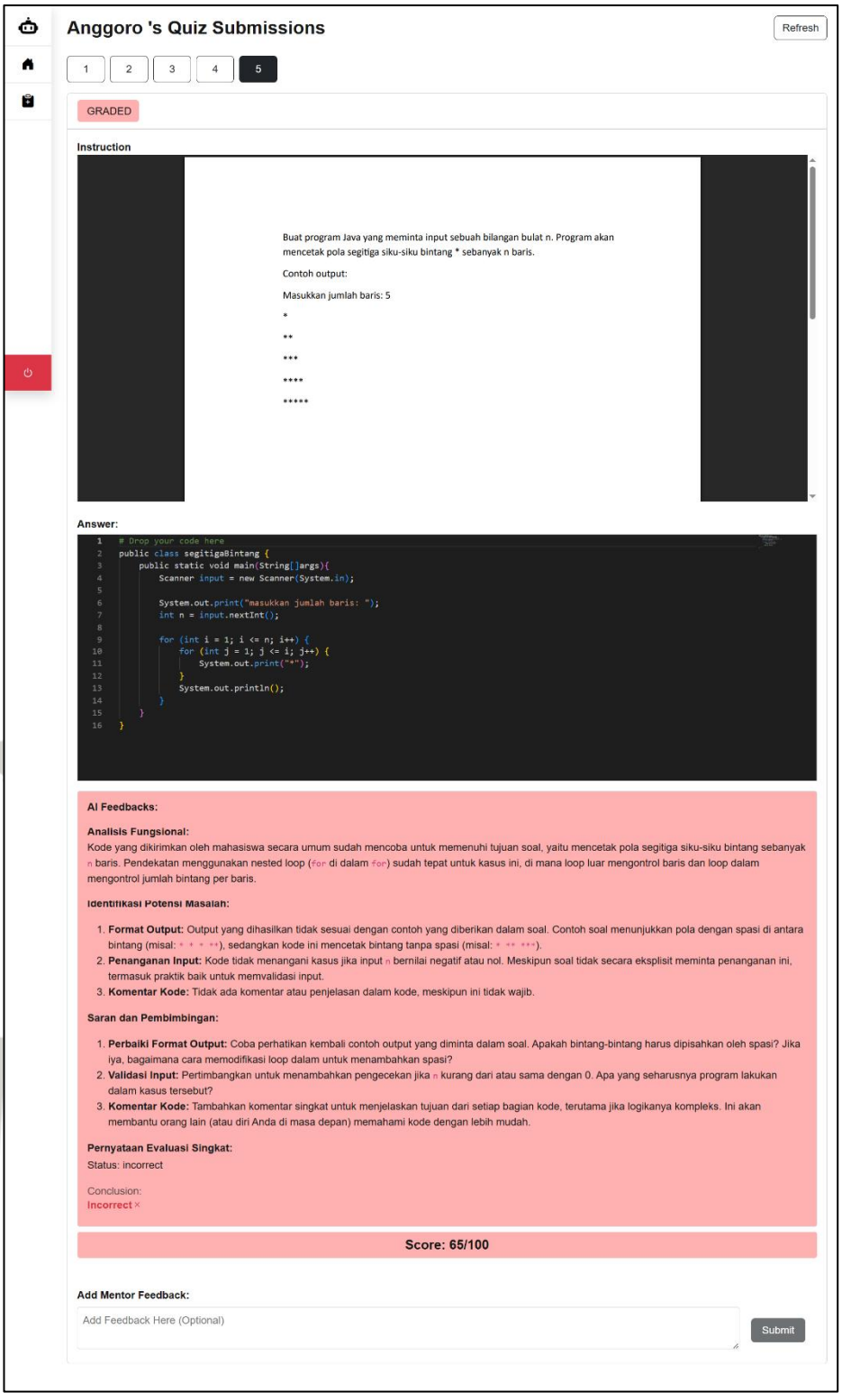


Gambar 4. 12 Hasil Evaluasi oleh AI Jawaban salah

Sementara itu, pada antarmuka mentor, informasi yang ditampilkan mencakup semua elemen yang sama, seperti *feedback* dari AI, skor, dan status benar/salah, namun dilengkapi dengan fitur tambahan untuk memberikan *feedback* manual. Dengan pembaruan ini, baik mahasiswa maupun mentor dapat dengan mudah melihat hasil evaluasi secara menyeluruh dan terperinci langsung dari tampilan web yang telah disediakan. Berikut tampilan antarmuka dari mentor untuk jawaban mahasiswa yang benar dan salah:



Gambar 4. 13 Tampilan Mentor Hasil Evaluasi oleh AI Jawaban Benar



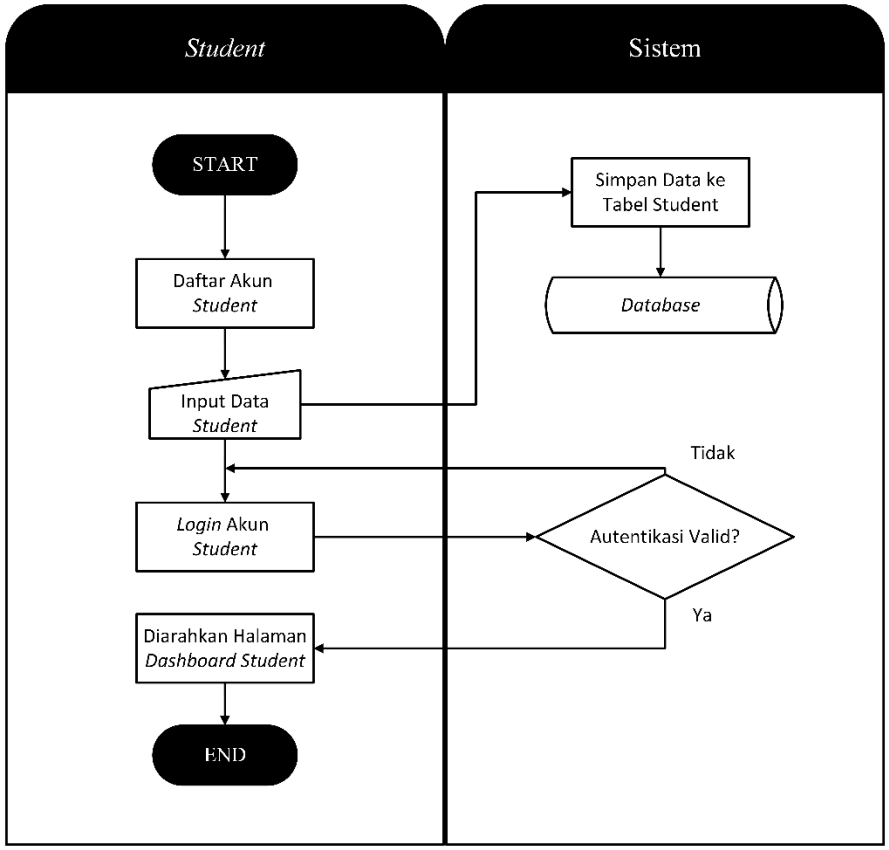
Gambar 4. 14 Tampilan Mentor Hasil Evaluasi oleh AI Jawaban Salah

4.2 Rangkaian Proses Sistem Secara Keseluruhan

Pada bagian ini, akan dijelaskan mengenai alur proses kerja sistem secara menyeluruh dari awal hingga akhir, dimulai dari pendaftaran mahasiswa (*student*) dan mentor (*educator*), hingga proses evaluasi jawaban otomatis oleh AI. Setelah berhasil membuat akun, mentor (*educator*) dapat membuat ujian dan soal, sedangkan mahasiswa (*student*) dapat melihat dan mengerjakan ujian serta soal-soal di dalamnya.

Setiap jawaban mahasiswa (*student*) akan dikirim dan dievaluasi oleh AI menggunakan rubrik penilaian yang telah ditentukan oleh mentor (*educator*). Hasil evaluasi berupa feedback, skor, dan status benar/salah akan disimpan ke dalam database dan ditampilkan ke antarmuka pengguna. Penjelasan lengkap mengenai setiap langkah dalam proses ini akan dijabarkan dalam subbab-subbab berikut.

4.2.1 Mahasiswa Membuat Akun

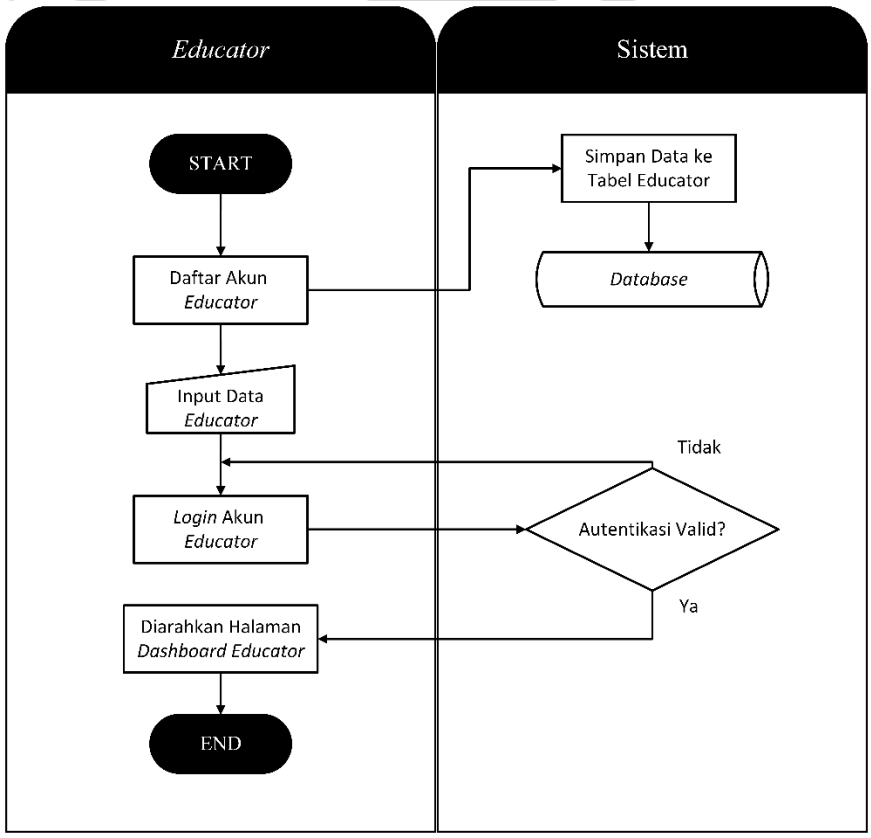


Gambar 4. 15 Alur Mahasiswa Membuat Akun

Pada tahap awal penggunaan sistem, mahasiswa perlu mendaftarkan akun terlebih dahulu. Proses pendaftaran dilakukan melalui halaman registrasi, di mana mahasiswa diminta mengisi nama, *email*, *password* dan peran sebagai *student*. Setelah formulir registrasi dikirimkan, sistem akan memproses data tersebut dan menyimpannya ke dalam basis data pada tabel khusus untuk entitas *Student*. Password yang disimpan telah melalui proses enkripsi menggunakan algoritma hashing agar data tetap aman.

Setelah berhasil membuat akun, mahasiswa dapat langsung melakukan *login* ke sistem. Proses login dilakukan dengan memasukkan email dan password yang telah terdaftar. Sistem akan mencocokkan data login dengan data yang ada di *database*. Jika kredensial tidak valid, sistem akan menolak akses dan mengarahkan kembali ke halaman *login* dengan notifikasi kesalahan. Namun jika data sesuai, mahasiswa akan diarahkan ke halaman *dashboard* utama sesuai peran *student*.

4.2.2 Mentor Membuat Akun

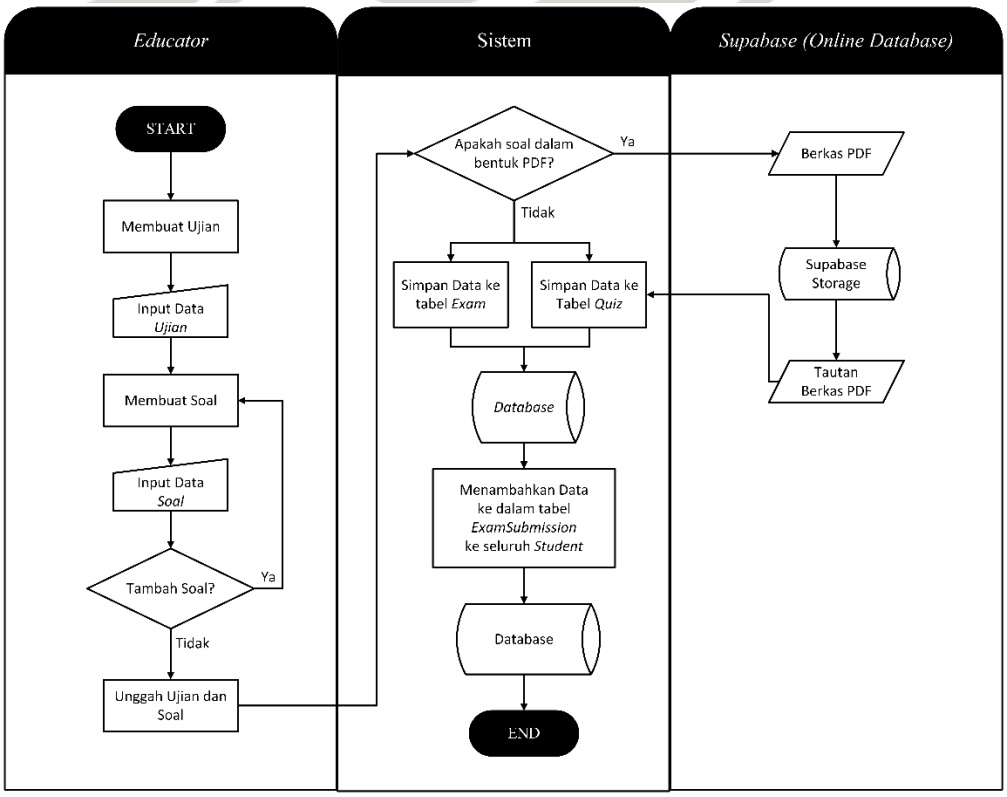


Gambar 4. 16 Alur Mentor Membuat Akun

Pada tahap ini, seluruh proses hampir menyerupai alur yang dilakukan oleh mahasiswa saat membuat akun. Perbedaan terletak pada pemilihan peran yang di mana mentor wajib memilih peran sebagai *educator*. Sistem akan menyimpan data yang dikirimkan dari formulir berupa nama, *email, password* dan peran sebagai *Educator* ke dalam *database Educator*. Sama seperti mahasiswa, password akan diproses menggunakan teknik hashing sebelum disimpan.

Setelah proses registrasi selesai, mentor dapat masuk ke sistem melalui halaman *login*. Sistem akan melakukan proses autentikasi dengan mencocokan email dan password yang tersimpan di *database* sebelumnya. Jika data tidak sesuai, sistem akan menolak dan mengarahkan pengguna kembali ke halaman login. Namun, jika data sesuai maka mentor akan diarahkan ke halaman *dashboard educator*.

4.2.3 Mentor Membuat Ujian dan Soal



Gambar 4. 17 Alur Mentor Membuat Ujian dan Soal

Setelah berhasil *login* ke sistem, mentor dapat membuat ujian baru melalui fitur yang tersedia di halaman *dashboard*. Proses dimulai dengan mengisi informasi dasar terkait ujian, yaitu judul ujian (*title*), nama mata kuliah (*courseName*), deskripsi ujian (*description*), tenggat waktu pengumpulan (*endDate*), dan nilai minimum kelulusan (*minScore*). Semua data tersebut kemudian akan digunakan untuk membentuk satu entitas *Exam* yang nantinya akan dikaitkan dengan beberapa soal (*Quiz*).

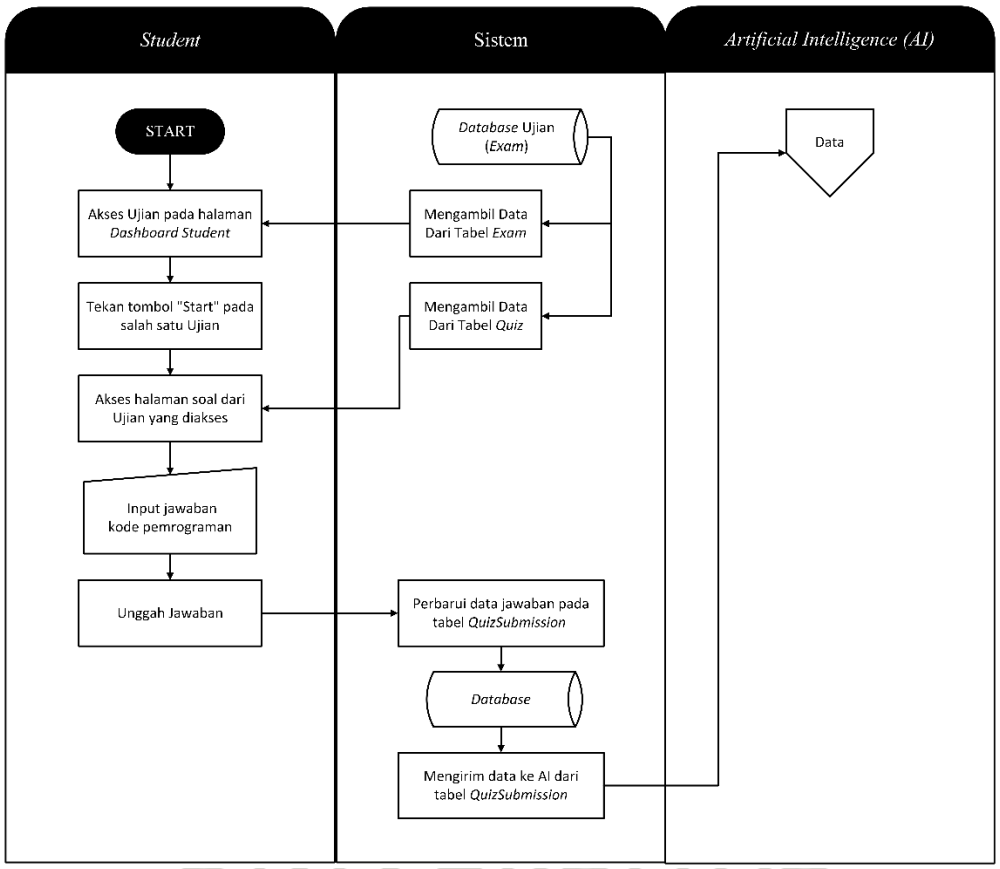
Selanjutnya, mentor dapat menambahkan satu atau lebih soal ke dalam ujian tersebut. Untuk setiap soal, mentor diberikan dua opsi dalam menyusun instruksi: pertama dalam bentuk teks langsung (*instruction*), dan kedua dalam bentuk unggahan berkas *PDF* (*fileUrl*). Keduanya bersifat opsional dan dapat digunakan secara bersamaan maupun salah satu saja. Selain itu, mentor juga harus memilih bahasa pemrograman yang akan digunakan oleh mahasiswa dalam menyelesaikan soal tersebut, seperti *Python*, *Java*, *PHP*, *C*, atau *JavaScript* (*language*). Di samping itu, tersedia juga kolom untuk mengisi rubrik penilaian (*rubrik*) yang akan dijadikan acuan oleh sistem AI dalam mengevaluasi jawaban. Mentor juga dapat menentukan batas maksimum jumlah pengumpulan jawaban (*submissionLimit*), atau membiarkannya kosong untuk memberikan akses tanpa batas.

Setelah satu soal selesai diisi, sistem akan memberikan pilihan kepada mentor untuk menambahkan soal baru atau menyelesaikan proses. Jika mentor memilih untuk menambah soal, maka proses pengisian data soal akan diulang hingga seluruh soal yang diinginkan telah dibuat. Setelah seluruh soal selesai diinput, sistem akan melakukan pengecekan: jika terdapat soal dalam bentuk *PDF*, maka berkas *PDF* tersebut akan diunggah ke *Supabase Storage* secara otomatis, dan tautan berkas (*fileUrl*) yang dihasilkan akan disimpan ke dalam tabel *Quiz*.

Setelah semua data lengkap dan valid, sistem akan menyimpan informasi ujian ke dalam tabel *Exam*, dan menyimpan seluruh soal ke dalam tabel *Quiz*. Untuk menghubungkan mahasiswa dengan ujian yang baru dibuat, sistem juga akan secara otomatis menghasilkan entri *ExamSubmission* untuk

setiap mahasiswa yang terdaftar di sistem, sehingga proses monitoring dan evaluasi dapat dilakukan secara individual. Proses pembuatan ujian pun dinyatakan selesai.

4.2.4 Mahasiswa Mengerjakan Ujian dan Soal



Gambar 4. 18 Alur Mahasiswa Mengerjakan Ujian dan Soal

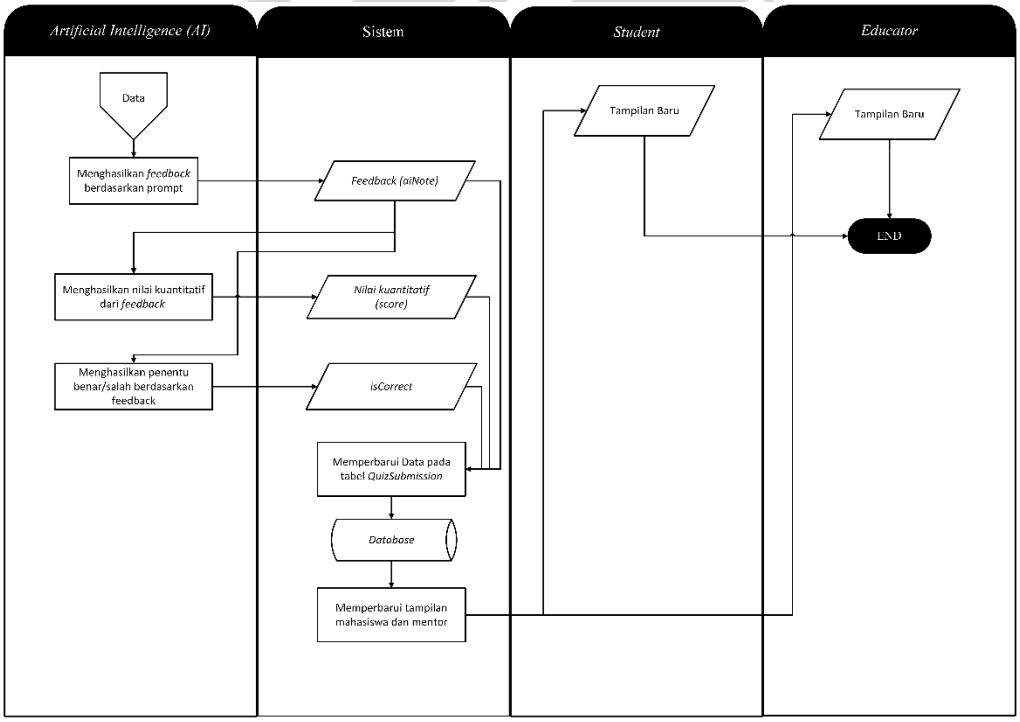
Setelah berhasil masuk ke dalam sistem, mahasiswa akan diarahkan menuju halaman dashboard yang menampilkan daftar ujian (*exam*) yang telah dibuat oleh mentor. Data-data ujian tersebut diambil secara langsung dari basis data *Exam* dan ditampilkan dalam bentuk kartu atau daftar, yang mencakup informasi penting seperti judul ujian (*title*), nama mata kuliah (*courseName*), tenggat waktu (*endDate*), nama mentor (*creatorName*), deskripsi ujian (*description*) dan jumlah soal dalam ujian (*quizzes*s).

Mahasiswa dapat memulai ujian dengan menekan tombol "*Start*", yang kemudian akan mengarahkan mereka ke halaman soal. Di halaman ini, seluruh soal yang tergabung dalam ujian ditampilkan satu per satu berdasarkan data dari

tabel *Quiz*. Mahasiswa dapat membaca instruksi soal, lalu menuliskan jawaban berupa kode pemrograman (*source code*) secara langsung melalui *text editor* yang telah disediakan oleh sistem.

Setelah selesai mengerjakan satu soal, mahasiswa dapat mengunggah jawaban dengan menekan tombol *Submit*. Pada tahap ini, sistem akan menyimpan jawaban ke dalam tabel *QuizSubmission*. Selanjutnya, sistem akan mengirimkan data soal yang bersangkutan ke modul AI untuk dievaluasi. Data yang dikirim meliputi instruksi soal (*instruction*), rubrik penilaian (*rubrik*), bahasa pemrograman (*language*), dan jawaban mahasiswa (*answer*).

4.2.5 Proses Evaluasi oleh AI

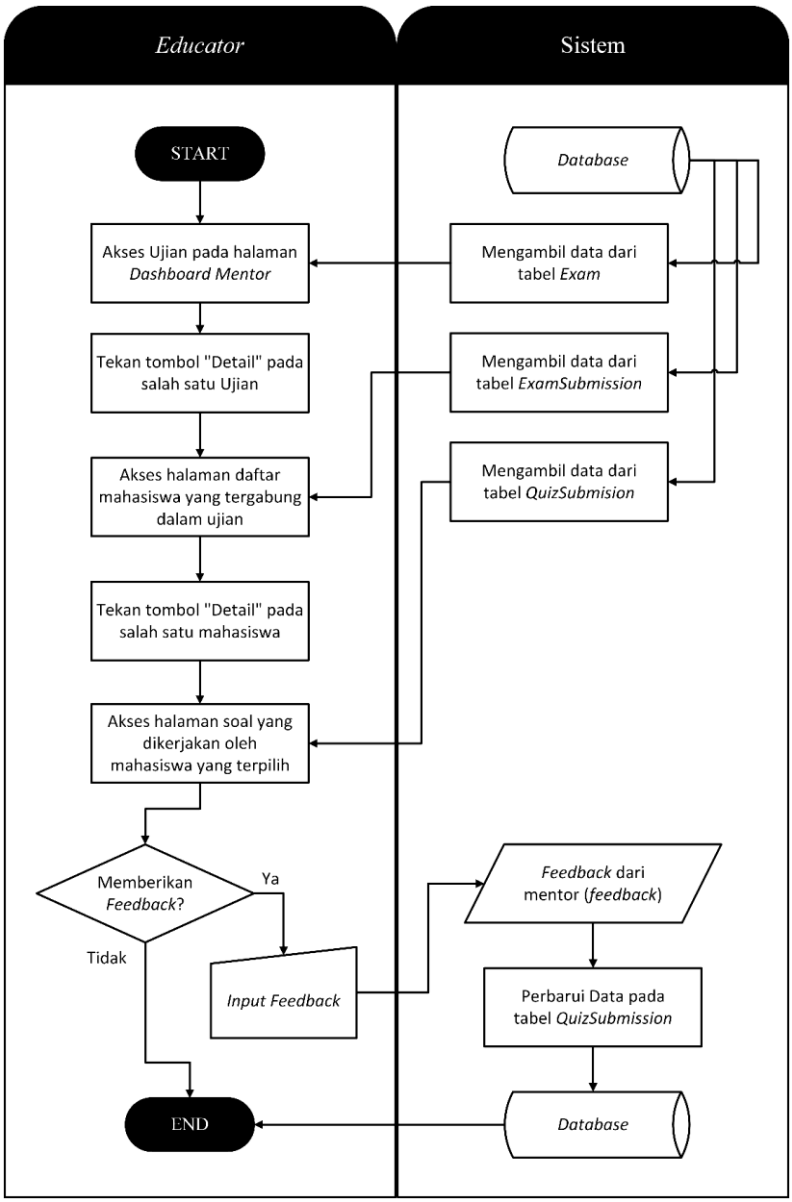


Gambar 4. 19 Alur Proses Evaluasi AI

Setelah sistem mengirim data berupa *instruction*, *answer*, *rubrik*, dan *language* ke model AI, model akan memproses data tersebut menggunakan *prompt* evaluasi yang telah disusun sebelumnya. Hasil dari proses ini adalah *feedback* yang menjelaskan tentang penilaian dari jawaban mahasiswa, serta skor numerik (*score*) yang menunjukkan seberapa baik jawaban memenuhi kriteria dalam rubrik.

Selanjutnya, sistem mengirimkan *feedback* tersebut ke model AI sekali lagi menggunakan prompt berbeda untuk menentukan apakah jawaban benar atau salah secara fungsional (*isCorrect*). Nilai *feedback*, *score*, dan *isCorrect* kemudian disimpan ke dalam tabel *QuizSubmission*. Terakhir, sistem memperbarui tampilan antarmuka mahasiswa dan mentor agar hasil evaluasi dapat langsung dilihat oleh keduanya.

4.2.6 Mentor Memberikan *Feedback*



Gambar 4. 20 Alur Mentor Memberikan *Feedback*

Setelah AI menyelesaikan proses evaluasi otomatis terhadap jawaban mahasiswa, mentor memiliki opsi untuk meninjau dan memberikan umpan balik tambahan secara manual. Proses ini dimulai saat mentor mengakses halaman *dashboard* yang menampilkan daftar ujian yang telah mereka buat. Mentor kemudian memilih salah satu ujian dengan menekan tombol *"*Detail*"* untuk melihat daftar mahasiswa yang tergabung dalam ujian tersebut.

Dari daftar mahasiswa tersebut, mentor dapat menekan tombol *"*Detail*"* pada salah satu mahasiswa untuk melihat hasil pengerjaan setiap soal. Di halaman ini, mentor dapat membaca *feedback* dari AI (*aiNote*) serta melihat jawaban yang telah diberikan oleh mahasiswa. Jika mentor ingin menambahkan penilaian atau masukan secara manual, tersedia *form input* di bawah *feedback* AI. Jika mentor mengisi form tersebut dan menekan tombol *Submit*, maka sistem akan menyimpan masukan tersebut ke dalam kolom *feedback* pada tabel *QuizSubmission*. Jika mentor memilih untuk tidak memberikan *feedback* tambahan, maka proses akan langsung berakhir tanpa perubahan data.

4.3 Evaluasi Sistem melalui Pengujian dan Umpan Balik

Pada tahap ini dilakukan serangkaian pengujian untuk mengevaluasi kualitas, keandalan, dan efektivitas sistem yang telah dikembangkan. Pengujian dibagi menjadi tiga bagian utama, yaitu:

1. Pengujian fungsionalitas sistem dengan mahasiswa
2. Evaluasi hasil AI oleh mentor
3. Pengujian performa sistem.

Pertama, pengujian dilakukan untuk memastikan bahwa seluruh fitur utama sistem, seperti proses pendaftaran dan *login*, pembuatan ujian dan soal oleh mentor, serta pengerjaan soal dan pengumpulan jawaban oleh mahasiswa, dapat berjalan dengan baik sesuai dengan alur penggunaan yang telah dirancang. Pengujian ini juga melibatkan mahasiswa secara langsung untuk mengerjakan soal melalui sistem. Setelah itu, mahasiswa diminta memberikan umpan balik terhadap pengalaman penggunaan sistem melalui kuesioner *System Usability Scale* (*SUS*), guna mengetahui tingkat kenyamanan, kemudahan, dan kepuasan dalam menggunakan sistem.

Selanjutnya, hasil evaluasi otomatis yang diberikan oleh AI terhadap jawaban mahasiswa turut ditinjau kembali oleh dosen atau mentor mata kuliah. Tujuannya adalah untuk menilai sejauh mana akurasi dan kualitas feedback yang dihasilkan AI dapat mencerminkan standar akademik dalam penilaian kode pemrograman.

Terakhir, pengujian performa sistem dilakukan menggunakan berbagai alat bantu seperti *Apache JMeter*, *Wireshark*, *Google Lighthouse*, dan *New Relic*. Pengujian ini mencakup metrik penting seperti *response time*, *latency*, *throughput*, penggunaan *CPU* dan memori, hingga kemampuan sistem dalam menangani banyak pengguna secara bersamaan. Hasil evaluasi performa ini digunakan untuk memastikan sistem tetap stabil, efisien, dan responsif meskipun digunakan dalam skala yang lebih besar di masa depan.

4.3.1 Pengujian Fungsionalitas Sistem dengan Mahasiswa

Terdapat perubahan dalam rencana awal pengujian sistem dengan mahasiswa. Awalnya, pengujian dirancang agar mahasiswa mengerjakan sebanyak 20 soal. Namun, demi efisiensi waktu dan efektivitas proses evaluasi, jumlah soal disederhanakan menjadi 5 soal promrograman yang dianggap cukup merepresentasikan fungsionalitas sistem. Pengujian ini dilakukan terhadap 5 mahasiswa acak yang tergabung dalam mata kuliah Pemrograman. Sebelum memulai ujian, seluruh mahasiswa diminta untuk membuat akun terlebih dahulu pada sistem.

Setelah seluruh akun mahasiswa berhasil dibuat, mentor membuat satu ujian berisi 5 soal pemrograman yang seluruhnya menggunakan bahasa pemrograman Java. Ujian dilakukan secara *online* dan diselesaikan dalam satu hari. Mahasiswa mengerjakan soal dengan menuliskan kode program pada editor yang disediakan di dalam sistem, kemudian mengirimkan jawaban untuk dievaluasi secara otomatis oleh AI. Setiap mahasiswa hanya diberikan satu kesempatan pengumpulan untuk setiap soal, sehingga jawaban yang dikirimkan merupakan hasil akhir dari pengerjaan mereka. Untuk menjaga keaslian dan objektivitas hasil evaluasi, mahasiswa diminta untuk mengerjakan soal tanpa menggunakan bantuan apapun, termasuk bantuan dari AI eksternal, mesin pencari seperti *Google*, maupun sumber lain di luar kemampuan pribadi

masing-masing. Hal ini dilakukan agar hasil penilaian benar-benar mencerminkan pemahaman individu terhadap materi yang diuji.

Secara keseluruhan terdapat 25 *submissions* (5 mahasiswa × 5 soal). Untuk keperluan analisis lebih lanjut, penulis menampilkan dua sampel representatif dari jawaban mahasiswa yang masing-masing mewakili jawaban yang dinilai benar dan salah oleh AI. Sampel didapatkan dengan mengakses halaman detail "mahasiswa" mentor.

Setelah mengerjakan seluruh soal, mahasiswa diminta untuk mengisi kuisioner *System Usability Scale* (*SUS*) sebagai bentuk evaluasi terhadap pengalaman penggunaan sistem. Selain itu, mahasiswa juga memberikan kritik dan saran yang mencakup antarmuka sistem dan hasil evaluasi AI. Berdasarkan hasil pengujian ini, fungsionalitas sistem terbukti berjalan dengan baik: mahasiswa dapat *login*, mengakses ujian, menyelesaikan soal, mengirimkan jawaban, dan menerima hasil evaluasi secara langsung dalam bentuk *feedback* naratif, skor kuantitatif, serta indikator benar/salah dari AI.

4.3.1.1 Sampel Evaluasi AI terhadap Jawaban Benar

Penulis membuat soal pemrograman sederhana tentang mengkonversi angka berupa suhu *Celcius* ke dalam *Farenheit* dan *Kelvin* dengan instruksi soal sebagai berikut:

Buat program Java yang meminta input suhu dalam derajat Celcius, lalu mengkonversi dan menampilkan ke dalam Fahrenheit dan Kelvin.

Contoh Output:

Masukkan suhu dalam Celcius: 25
Fahrenheit: 77.0
Kelvin: 298.15

Gambar 4. 21 Soal Evaluasi Hasil AI Jawaban Benar

Ditetapkan juga rubrik penilaian untuk soal ini sebagai dasar percobaan bagi AI dalam memberikan penilaian kuantitatif. Rubrik ini tidak disusun berdasarkan standar kurikulum tertentu, melainkan hanya

sebagai contoh untuk keperluan pengujian sistem. Di implementasi nyata, rubrik ini sepenuhnya dapat disesuaikan oleh masing-masing mentor sesuai dengan kebutuhan dan standar penilaian yang berlaku. Berikut isi rubrik yang digunakan pada soal ini:

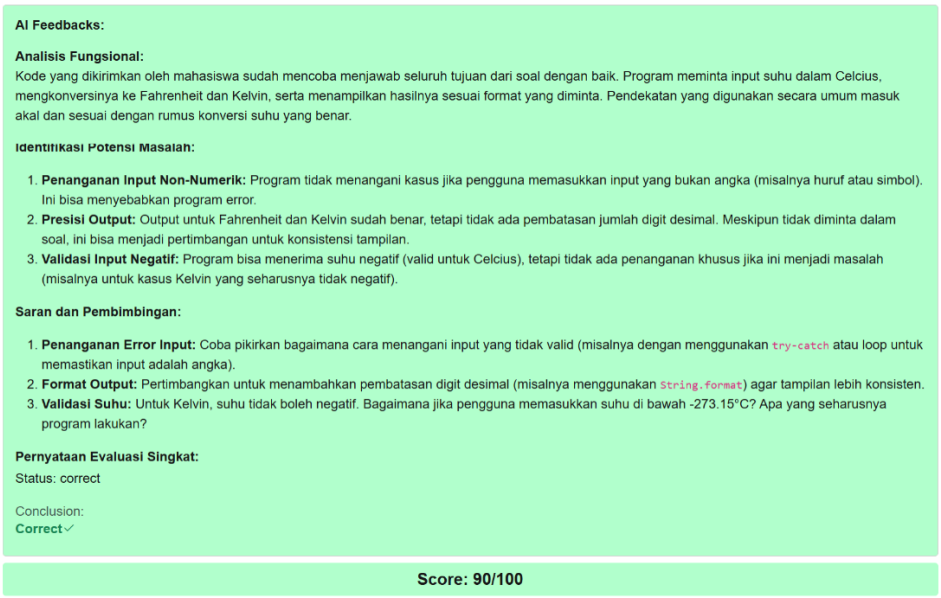
*Input Accuracy:*20
*Logic Correctness:*20
*Output Accuracy:*20
*Edge Case Handling:*20
*Syntax Validity:*20

Selanjutnya, berikut merupakan jawaban kode program yang dituliskan oleh mahasiswa untuk menyelesaikan soal di atas. Mahasiswa diminta menuliskan program dalam bahasa pemrograman Java sesuai dengan instruksi yang telah diberikan, tanpa bantuan eksternal apa pun. Berikut jawaban kode pemrograman yang dikirimkan oleh mahasiswa untuk soal ini:

```java
import java.util.Scanner;

class KonversiSuhu {
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);

        System.out.print("Masukkan suhu dalam Celcius: ");
        double celcius = input.nextDouble();

        double fahrenheit = (celcius * 9 / 5) + 32;
        double kelvin = celcius + 273.15;

        System.out.println("Fahrenheit: " + fahrenheit);
        System.out.println("Kelvin: " + kelvin);
    }
}
```

Gambar 4. 22 Jawaban Mahasiswa dalam Pengujian Evaluasi oleh AI

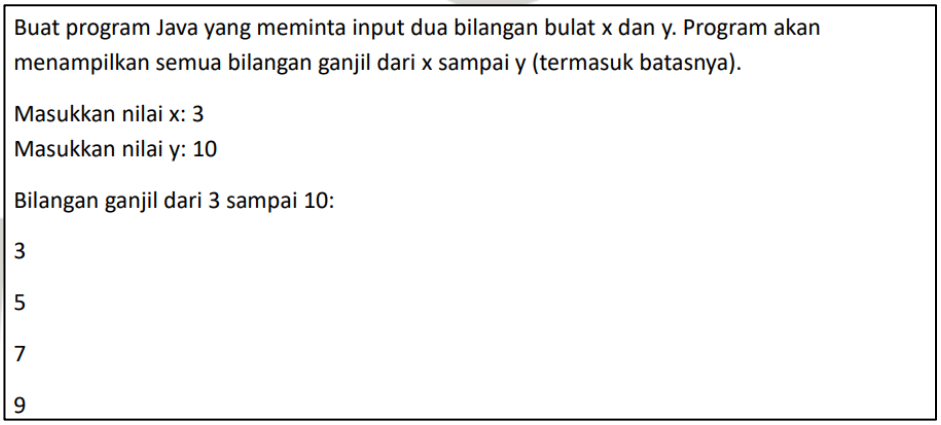
Jawaban tersebut kemudian dievaluasi secara otomatis oleh AI. Hasil evaluasi dari AI menunjukkan bahwa jawaban mahasiswa dinilai benar (*isCorrect*: *true*) dan mendapatkan skor kuantitatif sebesar 90/100. Nilai ini mencerminkan bahwa mayoritas kriteria dalam rubrik telah terpenuhi dengan baik. Berikut adalah *feedback* yang diberikan oleh AI terhadap jawaban tersebut:



Gambar 4. 23 Hasil Evaluasi oleh AI Jawaban Benar

4.3.1.2 Sampel Evaluasi AI terhadap Jawaban Salah

Penulis membuat soal pemrograman sederhana tentang menampilkan bilangan semua bilangan ganjil dari *x* sampai *y* dengan instruksi soal sebagai berikut:



Gambar 4. 24 Soal Hasil Evaluasi AI Jawaban Salah

Ditetapkan juga rubrik penilaian yang sama seperti pada hasil evaluasi sebelumnya untuk menjaga konsistensi penilaian antar jawaban. Sehingga perbandingan antara jawaban benar dan salah dapat dianalisis secara objektif. Berikut isi dari rubrik yang telah ditetapkan:

*Input Accuracy:*20

*Logic Correctness:*20

*Output Accuracy:*20

*Edge Case Handling:*20

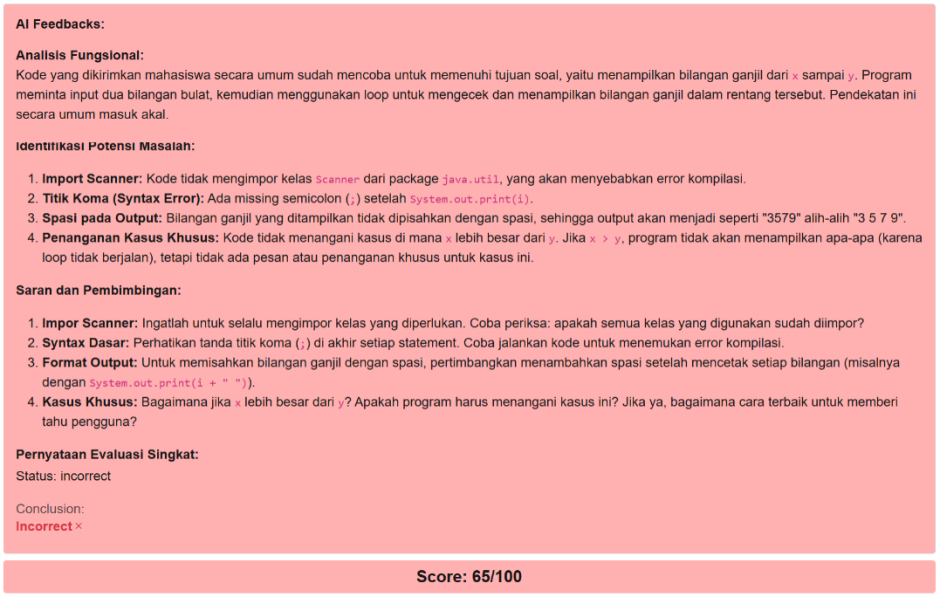
*Syntax Validity:*20

Selanjutnya, berikut merupakan jawaban kode program yang dituliskan oleh mahasiswa untuk menyelesaikan soal di atas. Mahasiswa diminta menuliskan program dalam bahasa pemrograman Java sesuai dengan instruksi yang telah diberikan. Berikut jawaban kode pemrograman yang dikirimkan oleh mahasiswa untuk soal ini:

```
# Drop your code here
public class ganjil{
    public static void main(String[]args){
        Scanner input = new Scanner (System.in);

        System.out.print("masukkan nilai x: ");
        int x = input.nextInt();

        System.out.print("masukkan nilai y: ");
        int y = input.nextInt();

        System.out.println("Bilangan ganjil dari " + x + " sampai " + y + ":");

        for(int i = x; i <= y; i++){
            if(i % 2 == 1){
                System.out.print(i)
            }
        }
    }
}
```

Gambar 4. 25 Jawaban Mahasiswa dalam Pengujian Evaluasi oleh AI

Jawaban tersebut kemudian dievaluasi secara otomatis oleh AI. Hasil evaluasi dari AI menunjukkan bahwa jawaban mahasiswa dinilai salah (*isCorrect*: *false*) dan mendapatkan skor kuantitatif sebesar 65/100. Nilai ini mencerminkan bahwa mayoritas kriteria dalam rubrik telah terpenuhi dengan baik. Berikut adalah *feedback* yang diberikan oleh AI terhadap jawaban tersebut:



Gambar 4. 26 Hasil Evaluasi oleh AI Jawaban Salah

4.3.1.3 Hasil Kuisioner *System Usability Scale* (*SUS*)

Untuk mengevaluasi tingkat kegunaan sistem dari sudut pandang pengguna, dilakukan pengujian menggunakan metode *System Usability Scale* (SUS). Proses pengumpulan data dilakukan melalui sebuah *Google Form* yang dibuat khusus untuk keperluan ini. Formulir tersebut berisi 10 pertanyaan yang sesuai dengan standar format kuisioner SUS.

Setelah seluruh mahasiswa responden mengisi kuisioner, penilaian dilakukan dengan mengikuti aturan perhitungan SUS. Setiap pertanyaan dinilai dalam skala 1 hingga 5. Untuk soal ganjil (nomor 1, 3, 5, dst), skor dikurangi 1. Sedangkan untuk soal genap (nomor 2, 4, 6, dst), skor dihitung sebagai 5 dikurangi skor. Semua skor hasil penyesuaian kemudian dijumlahkan dan dikalikan 2.5 untuk mendapatkan total skor SUS dalam rentang 0–100.

Berikut adalah hasil rekapitulasi penilaian dari lima mahasiswa yang mengikuti pengujian. Data kuisioner disusun dalam bentuk tabel spreadsheet, dengan nilai rata-rata SUS yang diperoleh ditampilkan pada bagian akhir tabel sebagai gambaran umum tingkat usability dari sistem ini.

Tabel 4.1 Hasil Skor Asli *System Usabillity Scale (SUS)*

No	Reponden	Skor Asli									
		Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10
1	Responden 1	4	1	5	1	5	2	5	1	5	1
2	Responden 2	5	5	5	2	5	1	5	1	1	4
3	Responden 3	4	1	4	1	5	1	4	2	5	1
4	Responden 4	3	2	4	1	5	2	5	2	5	4
5	Responden 5	4	4	3	2	4	4	4	3	2	4

Tabel 4.2 Hasil Skor Perhitungan *System Usabillity Scale (SUS)*

Skor Hasil Hitung										Jumlah	Nilai
Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10		(Jumlah x 2.5)
3	4	4	4	4	3	4	4	4	4	38	95
4	0	4	3	4	4	4	4	0	1	28	70
3	4	3	4	4	4	3	3	4	4	36	90
2	3	3	4	4	3	4	3	4	1	31	78
3	1	2	3	3	1	3	2	1	1	20	50
Skor Rata-rata (Hasil Akhir)											77

Dari hasil pengisian kuisioner oleh lima mahasiswa, didapatkan nilai akhir dengan rata-rata skor SUS sebesar 77. Pada tabel penilaian, setiap kolom seperti Q1, Q2, Q3, dan seterusnya mewakili nomor soal dari 1 hingga 10 dalam format kuisioner SUS. Skor pada masing-masing pertanyaan dihitung sesuai aturan konversi standar SUS, kemudian dijumlahkan dan dikalikan 2.5 untuk mendapatkan skor akhir masing-masing responden.

Nilai rata-rata sebesar 77 ini menunjukkan bahwa sistem telah berada pada kategori usability yang baik, mengingat skor SUS di atas 68 umumnya dianggap sebagai nilai di atas rata-rata secara global. Hasil ini menjadi indikator awal bahwa sistem cukup mudah digunakan, dipahami, dan memberikan pengalaman pengguna yang positif dari sudut pandang mahasiswa.

4.3.1.4 Hasil Kritik dan Saran (*feedback*) Setiap Mahasiswa

Pada kuisioner yang sama dengan *System Usability Scale* (*SUS*), disediakan satu kolom isian terbuka di bagian akhir formulir yang memungkinkan mahasiswa memberikan kritik dan saran secara bebas

terhadap sistem. Tujuan dari isian ini adalah untuk memperoleh umpan balik kualitatif terkait pengalaman penggunaan sistem secara keseluruhan, baik dari sisi tampilan, fungsionalitas, maupun kualitas feedback yang diberikan oleh AI.

Dari total lima responden mahasiswa, diperoleh beberapa *feedback* berbeda-beda yang mencerminkan pengalaman masing-masing selama menggunakan sistem. Seluruh feedback tersebut kemudian dikategorikan menjadi dua bagian, yaitu:

- *Feedback* Positif: berisi apresiasi atau keunggulan sistem dari sudut pandang mahasiswa.
- *Feedback* Negatif: berisi saran perbaikan, keluhan, atau kendala yang ditemui selama penggunaan sistem.

Klasifikasi ini dilakukan untuk mempermudah proses evaluasi dan perbaikan sistem di masa mendatang. Seluruh masukan dari responden telah dirangkum dan dianalisis untuk diambil kesimpulan umum, mengingat sebagian besar *feedback* dituliskan dalam bahasa yang tidak formal dan menggunakan gaya tutur sehari-hari. Oleh karena itu, isi *feedback* tidak dicantumkan secara mentah dalam laporan ini, melainkan telah disesuaikan secara redaksional agar tetap menjaga konteks namun sesuai dengan kaidah penulisan ilmiah**.** Berikut adalah ringkasan dari kritik dan saran mahasiswa berdasarkan kategori tersebut:

1. *Feedback* Positif:

- Untuk pemberitahuan oleh AI sudah bagus, lengkap, dan jelas. Penjelasannya dapat membantu pengguna dalam mengerjakannya dengan lebih baik, tidak hanya membetulkan program yang salah tapi juga hal kecil yang dapat memaksimalkan program dengan jauh lebih baik
- Penjelasan dari AI sudah sesuai dan membantu mahasiswa dalam memahami serta memperbaiki kesalahan. Sistem penilaian juga

cukup adil, karena tidak langsung memberikan nilai 0 apabila terdapat kesalahan, melainkan tetap memberikan poin untuk bagian yang benar.

- Secara keseluruhan, pemberian umpan balik oleh AI sudah cukup baik dan menggunakan bahasa yang mudah dipahami.

2. *Feedback* Negatif:

- Tampilan sistem sebaiknya diperbarui agar lebih menarik dan tidak membosankan. Selain itu, disarankan agar fitur-fitur dalam sistem diperbanyak dan antarmuka dirapikan agar lebih nyaman dilihat.
- Pada editor untuk bahasa Java, saat ini tidak tersedia fitur bantuan seperti *auto-complete*, seperti mengetik *sout* secara otomatis menjadi *System.out.println()*;. Disarankan agar fitur tersebut ditambahkan untuk mempermudah pengguna dalam menulis kode.
- Disarankan agar sebelum mengerjakan soal disediakan panduan atau instruksi lebih lanjut mengenai cara pengisian dan pengerjaan soal.
- Halaman lembar soal terlihat terlalu panjang dan menyisakan ruang kosong. Disarankan agar panjang halaman dapat disesuaikan dengan panjang isi soal agar tampilannya lebih proporsional. Selain itu, proses pemberian *feedback* dari AI sebaiknya dapat dipercepat agar pengguna tidak perlu menunggu terlalu lama.
- Disarankan agar tampilan *website* dibuat lebih menarik dan berwarna untuk meningkatkan kenyamanan visual pengguna. Selain itu, ukuran huruf pada bagian soal dinilai terlalu kecil sehingga sebaiknya diperbesar agar lebih mudah dibaca.

Berdasarkan hasil klasifikasi dan analisis terhadap kritik dan saran mahasiswa, dapat disimpulkan bahwa secara umum sistem evaluasi AI telah berfungsi dengan baik, terutama dalam hal kualitas *feedback*. Penjelasan yang diberikan AI dinilai jelas, mudah dipahami, dan membantu mahasiswa dalam memperbaiki kesalahan pemrograman.

Selain itu, sistem penilaian yang tidak langsung memberikan skor 0 atas kesalahan juga diapresiasi karena tetap memberikan nilai untuk bagian kode yang benar.

Namun, dari sisi tampilan dan fungsionalitas sistem, masih ditemukan beberapa kekurangan yang perlu diperbaiki. Mahasiswa menyarankan agar tampilan antarmuka diperbarui menjadi lebih menarik dan responsif, fitur seperti *auto-complete* pada editor kode ditambahkan, serta proses evaluasi AI dipercepat agar pengguna tidak perlu menunggu terlalu lama. Selain itu, masukan juga diberikan terkait tata letak halaman soal, ukuran huruf, dan perlunya panduan yang lebih jelas sebelum mengerjakan soal. Masukan ini diharapkan dapat menjadi acuan dalam pengembangan sistem ke tahap yang lebih baik.

4.3.2 Evaluasi hasil AI oleh Mentor

Evaluasi terhadap hasil penilaian otomatis oleh AI tidak hanya dilakukan dari sisi pengguna mahasiswa, namun juga dari sudut pandang akademisi, dalam hal ini adalah dosen atau mentor mata kuliah Pemrograman. Evaluasi ini bertujuan untuk menilai apakah *feedback* yang dihasilkan oleh sistem AI sudah mencerminkan standar penilaian akademik yang sebenarnya, serta apakah bahasa, struktur, dan bobot evaluasi sudah layak digunakan dalam konteks pendidikan formal.

Proses evaluasi dilakukan dengan meminta mentor untuk membaca dan menelaah hasil evaluasi yang telah diberikan AI terhadap jawaban mahasiswa, baik yang dinyatakan benar maupun salah. Mentor kemudian diminta memberikan umpan balik secara naratif mengenai kualitas, ketepatan, dan kegunaan dari *feedback* tersebut, termasuk apakah skor yang diberikan sudah sesuai dengan isi jawaban mahasiswa berdasarkan rubrik.

Berikut adalah hasil evaluasi dan komentar dari mentor terhadap kinerja sistem evaluasi otomatis berbasis AI yang telah diterapkan:

1. **Ketepatan logika penjelasan:**

   Secara umum logika koding dijelaskan dengan cukup baik meski untuk beberapa kasus dirasa masih dapat diperjelas lagi. Logika penjelasan dapat dilengkapi dengan ilustrasi sederhana sehingga lebih *on point* dan mudah dipahami pengguna.

2. **Kesesuaian dengan materi ajar:**

   Untuk materi umum, *platform* yang dibangun sudah dapat mengakomodasi kebutuhan pengajar. Sementara untuk materi yang bersifat khusus atau kustomisasi, penjelasan yang diberikan masih perlu di lengkapi dengan penjelasan lebih lanjut.

3. **Kejelasan informasi yang disampaikan:**

   Informasi *feedback* yang diberikan sudah cukup jelas dan sesuai dengan prompting yang diberikan; pemberian ilustrasi sederhana dinilai akan dapat membantu pengguna untuk memahami konteks penjelasan dengan lebih baik.

4. **Nilai edukatif dari *feedback* yang diberikan:**

   Penjelasan yang diberikan dinilai sudah memenuhi komponen edukatif yang memungkinkan mahasiswa untuk terus berproses secara mandiri berdasarkan *feedback* yang diberikan.

5. **Apakah AI ini dapat mempercepat proses evaluasi dalam mata kuliah pemrograman:**

   Ya, penggunaan AI dirasa sangat membantu dalam proses evaluasi; yang perlu dipastikan adalah rubrik penilaian yang harus dibuat dengan lebih detil sehingga hasil penilaian yang diberikan bisa more *explainable*, alih-alih memberikan nilai tanpa dasar yang jelas. Penggunaan matriks penilaian yang baik akan mengurangi potensi penilaian subjektif pengajar terhadap hasil kerja mahasiswa.

4.3.3 Hasil Pengujian Performa Sistem

Pengujian performa dilakukan untuk mengevaluasi sejauh mana sistem dapat bekerja secara optimal dalam berbagai kondisi beban. Terdapat tujuh aspek utama yang diuji dalam proses ini, yaitu *response time*, *throughput*,

latency, *load time*, penggunaan CPU dan memori, skalabilitas, serta penggunaan *bandwidth*.

Pada tahap awal, response time dan throughput direncanakan untuk diuji menggunakan *Apache JMeter*. Namun, karena adanya kendala teknis terutama pada pengelolaan autentikasi token (*session-token*) saat melakukan permintaan *POST* terhadap endpoint yang memerlukan otorisasi, maka dilakukan perubahan pendekatan. Sebagai gantinya, digunakan *Grafana* k6, sebuah alat pengujian performa berbasis CLI yang lebih fleksibel dan mudah dalam hal konfigurasi autentikasi. Dengan Grafana k6, pengujian response time dan *throughput* dapat dilakukan secara lancar dan memberikan hasil yang konsisten sesuai skenario yang disimulasikan.

Sementara itu, pengujian load time dilakukan dengan menggunakan Google Lighthouse. Melalui laporan yang dihasilkan oleh Lighthouse, dapat diketahui skor performa keseluruhan dari halaman, serta rekomendasi teknis untuk peningkatan kecepatan dan efisiensi tampilan. Dengan demikian, pengujian ini membantu memastikan bahwa pengalaman pengguna tetap optimal dari sisi kecepatan akses antarmuka sistem.

Dengan pengujian dari berbagai sudut ini, sistem dapat dinilai secara menyeluruh baik dari sisi kecepatan, efisiensi, maupun ketahanannya terhadap beban yang meningkat. Hasil lengkap dari setiap pengujian dibahas pada bagian selanjutnya.

4.3.3.1 Hasil Uji *Response Time*

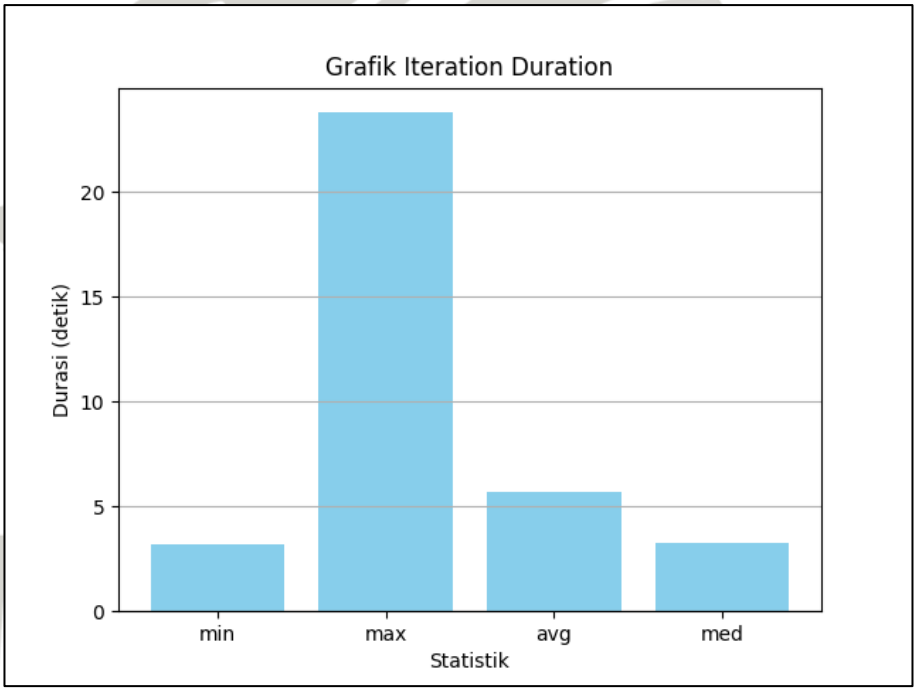
Untuk menguji response time sistem evaluasi otomatis berbasis AI, digunakan alat bantu *K6*, yaitu sebuah tool open-source yang digunakan untuk melakukan uji beban (*load testing*) pada aplikasi berbasis web. Pengujian dilakukan dengan cara mensimulasikan interaksi pengguna, yaitu ketika seorang user mengirimkan jawaban kuis, kemudian sistem akan memproses dan memberikan hasil evaluasi AI.

Pengujian dilakukan menggunakan 20 *virtual user* (VUs) dalam durasi waktu 120 detik. *Script* pengujian dibuat dalam berkas *JavaScript*

yang mengatur proses submit jawaban dan *polling* berkala untuk mendapatkan status penilaian dari sistem. Jawaban yang dikirimkan dalam pengujian ini adalah kode sederhana `print("Hello World")`.

Hasil pengujian yang dihasilkan oleh *Grafana K6* berupa data dalam format *JSON*, yang kemudian diolah menggunakan *Python* dengan bantuan library *matplotlib* untuk menggambarkan metrik penting seperti waktu proses tercepat (*minimum*), waktu rata-rata (*average*), dan waktu terlama (*maximum*) dalam bentuk grafik. Visualisasi ini mempermudah analisis performa sistem dalam memproses permintaan secara real-time.

Hasil dari pengujian ditampilkan dalam bentuk grafik berikut:



Gambar 4. 27 Hasil Uji *Response Time*

Berdasarkan data yang diperoleh dari grafik di atas, dapat disimpulkan bahwa:

- Waktu tercepat dari proses submit hingga mendapatkan hasil evaluasi adalah sekitar 3 detik (*min* = 3168 *ms*),
- Waktu terlama mencapai lebih dari 23 detik (*max* = 23779 *ms*),

- Rata-rata waktu proses (*average*) berada di angka sekitar 5,7 detik (*avg* ≈ 5707 *ms*)
- Median waktu proses, yaitu waktu yang paling merepresentasikan kondisi umum adalah sekitar 3,2 detik (*med* ≈ 3280 *ms*).

Hal ini menunjukkan bahwa sebagian besar proses evaluasi berlangsung cukup cepat, namun terdapat beberapa kasus yang memerlukan waktu lebih lama, kemungkinan disebabkan oleh beban sistem, kompleksitas pemrosesan AI, atau antrian proses evaluasi.

Hasil response time ini dapat bervariasi tergantung pada soal yang diberikan dan jawaban yang dikirimkan. Dalam pengujian ini, sistem hanya memproses jawaban sederhana berupa satu baris *output* `print("Hello World")`. Pada kasus nyata, di mana jawaban bisa lebih kompleks dan evaluasi AI membutuhkan waktu pemrosesan lebih dalam, waktu respons bisa lebih tinggi.

4.3.3.2 Hasil Uji *Throughtput*

Untuk mengukur kemampuan sistem dalam menangani jumlah permintaan secara bersamaan, dilakukan pengujian *throughput* menggunakan alat *K6* dengan skenario yang sama seperti sebelumnya, yaitu 20 *virtual user* (*VUs*) dalam durasi 120 detik. *Throughput* dihitung berdasarkan jumlah permintaan *HTTP* (*HTTP requests*) yang berhasil dikirim dan diterima oleh sistem setiap detik selama pengujian berlangsung. Berdasarkan data hasil pengujian yang terekam dalam berkas *JSON*, diperoleh hasil sebagai berikut:

```
"http_reqs": {
            "count": 1336,
            "rate": 10.865678451266767
        }
```

Gambar 4. 28 Hasil Uji *Throughput*

Dari data tersebut dapat diketahui bahwa sistem mampu melayani sebanyak 1336 request selama 120 detik, dengan rata-rata *throughput* sebesar 10,86 *requests per second* (req/s). Nilai ini menunjukkan bahwa server cukup stabil dalam menangani banyak permintaan secara paralel tanpa mengalami kegagalan atau keterlambatan yang signifikan.

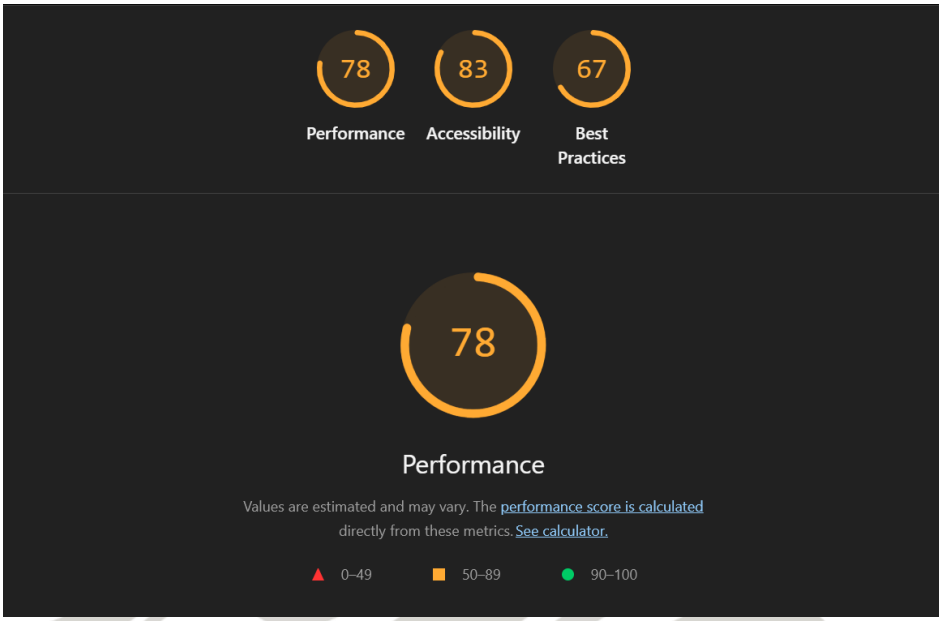
Dapat disimpulkan bahwa sistem evaluasi otomatis berbasis AI memiliki kapasitas penanganan beban yang cukup baik dalam skenario pengujian ini. Namun, performa throughput ini masih bisa meningkat atau menurun tergantung kompleksitas kode yang dikirim user dan beban server saat proses evaluasi berlangsung.

4.3.3.3 Hasil Uji *Load Time*

Pengujian *load time* dilakukan untuk mengukur seberapa cepat halaman web dapat dimuat dan menampilkan konten utamanya secara utuh kepada pengguna. Dalam pengujian ini, digunakan alat bantu *Google Lighthouse*, yaitu sebuah *tool open-source* dari *Google* yang dapat menganalisis performa halaman web secara menyeluruh, termasuk kecepatan loading, efisiensi rendering, dan optimasi sumber daya.

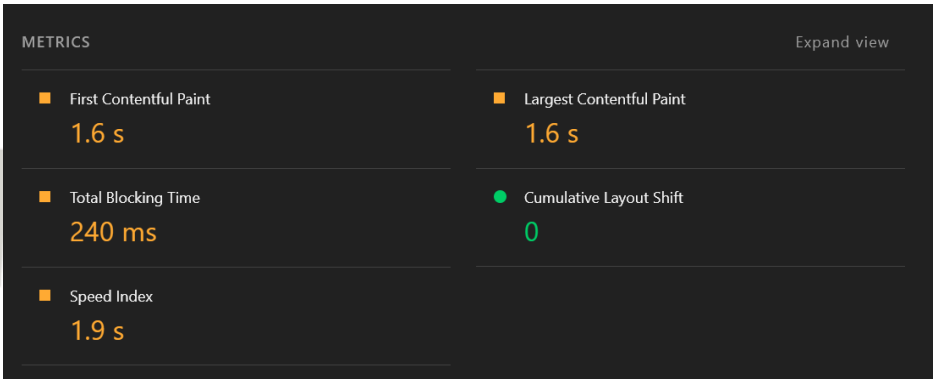
Pengujian dilakukan pada halaman ketika mahasiswa mengakses soal-soal yang terdapat di dalam ujiannya, yaitu saat mahasiswa membuka tampilan daftar kuis pada sebuah ujian. Berikut hasil pengujian

ditunjukkan pada gambar berikut:



Gambar 4. 29 Hasil Uji *Load Time* 1

Berdasarkan hasil tersebut, dapat disimpulkan bahwa performa halaman berada pada skor 78 (kategori sedang), yang menunjukkan bahwa halaman sudah cukup cepat namun masih dapat dioptimalkan lebih lanjut.



Gambar 4. 30 Hasil Uji *Load Time* 2

Beberapa metrik penting yang menjadi acuan antara lain:

- *First Contentful Paint (FCP)*: 1.6 detik
  Menunjukkan waktu pertama kali konten (teks/gambar) muncul di layar.

- *Largest Contentful Paint (LCP)*: 1.6 detik
  Waktu yang dibutuhkan hingga elemen terbesar pada halaman tampil sempurna.
- *Speed Index*: 1.9 detik
  Metrik visual yang mengukur kecepatan konten terlihat saat halaman dimuat.
- *Total Blocking Time (TBT)*: 240 milidetik
  Durasi waktu saat *thread* utama terblokir dan halaman belum bisa merespons interaksi.
- *Cumulative Layout Shift (CLS)*: 0
  Artinya tidak ada pergeseran tata letak yang mengganggu saat halaman dimuat.

Dengan nilai-nilai tersebut, dapat dikatakan bahwa sistem memiliki waktu loading yang cukup responsif, dan tidak menunjukkan pergeseran tata letak yang mengganggu. Namun demikian, *Total Blocking Time* masih bisa dikurangi untuk mencapai performa optimal, terutama pada perangkat dengan spesifikasi rendah atau koneksi lambat.

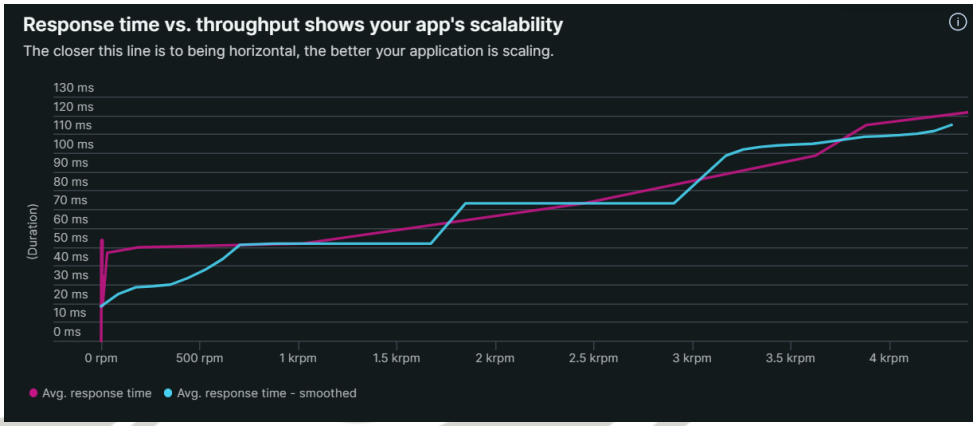
4.3.3.4 Hasil Uji Skalabilitas

Pengujian skalabilitas dilakukan menggunakan alat bantu *k6* untuk mensimulasikan jumlah *virtual users* (VU) yang meningkat secara bertahap. Setiap *virtual user* melakukan pengumpulan jawaban dan menunggu hasil evaluasi dari AI.

Untuk memantau performa sistem selama pengujian berlangsung, digunakan *New Relic* yang telah diintegrasikan langsung ke dalam kode *backend* sistem. Data metrik divisualisasikan dalam bentuk grafik interaktif pada *dashboard New Relic*, sehingga memudahkan analisis terhadap titik-titik lonjakan beban, stabilitas performa, dan batas maksimum skalabilitas sistem.

Pengujian dilakukan dengan pendekatan *ramping load*, di mana jumlah *virtual users* (VU) ditingkatkan secara bertahap untuk mensimulasikan peningkatan beban sistem. Pada menit pertama, sistem

menerima beban awal sebanyak 10 pengguna virtual. Kemudian, pada menit kedua, jumlah pengguna virtual dinaikkan menjadi 50, lalu ditingkatkan lagi menjadi 100 pada menit ketiga. Pada fase beban puncak, yaitu menit keempat hingga kelima, sebanyak 150 pengguna virtual aktif secara bersamaan, memberikan tekanan maksimal terhadap sistem. Seluruh aktivitas pengujian ini dimonitor menggunakan *New Relic* dan ditampilkan grafik sebagai berikut:



Gambar 4. 31 Hasil Uji Skalabilitas

Grafik menunjukkan pola sebagai berikut:

- Garis ungu mewakili response time yang mengalami lonjakan pada saat jumlah pengguna naik, lalu mendatar ketika beban stabil.
- Garis biru menunjukkan *request per second* yang juga naik seiring peningkatan beban.

Grafik pada dashboard *New Relic* menunjukkan pola lonjakan (*spike*) setiap kali jumlah *virtual users* (VU) dinaikkan, misalnya dari 50 ke 100 pengguna. Setiap lonjakan ini diikuti oleh garis horizontal yang stabil, menandakan bahwa setelah menerima beban tambahan, sistem berhasil menyesuaikan diri dan kembali ke performa normal tanpa gejala *overload* seperti *crash* atau peningkatan waktu respons ekstrem.

Polanya yang konsisten ini menunjukkan bahwa sistem memiliki kemampuan skalabilitas yang baik. Artinya, sistem tetap stabil dan responsif meskipun mengalami peningkatan beban secara bertahap dalam

waktu singkat. Hal ini menjadi indikator positif bahwa arsitektur *backend*, proses evaluasi AI, dan infrastruktur server bekerja secara efisien di bawah tekanan tinggi.

4.4 Pengujian Penggunaan Token dan Estimasi Biaya Evaluasi AI

Pengujian ini dilakukan dengan tujuan untuk melacak penggunaan token yang dipakai saat mahasiswa mengirim jawaban hingga AI memberikan *feedback.* Menggunakan library *CallbackManager()* dari *langchain* yang berfungsi untuk mencatat, memantau, *streaming* dan tugas-tugas lainnya termasuk penggunaan token. Dalam implementasinya, *CallbackManager* dikonfigurasi untuk menangani event *handleLLMEnd*, yaitu saat AI selesai melakukan proses inferensi. Di dalam handler tersebut, sistem akan mengekstrak metadata dari *output* model, khususnya bagian yang mencatat jumlah token *input* (*promptTokens*), token *output* (*completionTokens*), dan total token yang digunakan (*totalTokens*). Token input yang diambil merupakan token jenis *cache miss* karena pada sistem ini tidak menggunakan sistem *caching* untuk pengiriman prompt ke AI sehingga token yang dihasilkan merupakan jenis *cache miss*.

Hasil kumpulan token yang didapatkan berupa *JSON*, hasil tersebut kemudian dikalkulasi untuk mendapatkan biaya dalam mata uang *dollar*. Berdasarkan dokumentasi resmi dari *DeepSeek*, biaya penggunaan token untuk model *DeepSeek-R1* (*deepseek-reasoner*) per tanggal 24 Juli 2025 adalah sebagai berikut:

a. Biaya token input (prompt) *cache miss*: $0.00000055 per token
b. Biaya token output (completion): $0.00000219 per token

Biaya tersebut dihitung berdasarkan harga standar (*standard price*) yang berlaku pada jam UTC 00.30 – 06.30, sesuai dengan dokumentasi resmi dari DeepSeek.

Seluruh hasil data token dan biaya yang telah dikalkulasi ditampilkan ke halaman Detail “Mahasiswa” Mentor. Sistem diuji menggunakan soal pemrograman yang kompleks yang mengharuskan implementasi konsep *Object-Oriented Programming* (OOP) seperti *encapsulation*, *inheritance*, dan *polymorphism*. Jawaban yang digunakan dalam pengujian mencapai hampir 200 baris kode Java. Pengujian ini bertujuan untuk melihat berapa banyak token yang

digunakan oleh sistem saat memproses input dengan jawaban yang panjang, serta untuk mengamati estimasi biaya evaluasi yang ditimbulkan dari pemrosesan tersebut.

Pemilihan soal yang kompleks juga bertujuan untuk mensimulasikan skenario penggunaan sistem dalam konteks ujian akhir mahasiswa, di mana jawaban yang diberikan bisa sangat panjang dan melibatkan implementasi penuh konsep OOP. Melalui pengujian ini, dapat diprediksi estimasi jumlah token yang dibutuhkan apabila sistem digunakan dalam skala nyata untuk evaluasi ujian final. Selain itu, pengujian ini juga sekaligus memastikan bahwa sistem tetap dapat berjalan tanpa melewati batas maksimal token yang ditetapkan oleh model DeepSeek-R1, yaitu 64.000 token dalam satu permintaan. Berikut rincian pengujian yang dilakukan:

a. Soal:

Buatlah sebuah program manajemen perpustakaan sederhana menggunakan bahasa Java. Sistem ini harus mampu melakukan hal-hal berikut:

- Menyimpan data buku (judul, penulis, tahun terbit, jumlah stok).
- Menyimpan data anggota (nama, ID, daftar buku yang dipinjam).
- Memungkinkan anggota untuk meminjam dan mengembalikan buku.
- Menampilkan daftar buku yang tersedia dan yang sedang dipinjam.
- Menyimpan dan membaca data buku serta anggota ke/dari file .txt.

Gunakan konsep Object-Oriented Programming secara penuh (Encapsulation, Inheritance, Polymorphism), serta manfaatkan Collection seperti ArrayList dan HashMap.

b. Rubrik:

*Input Accuracy:*20

*Logic Correctness:*20

*Output Accuracy:*20

*Edge Case Handling:*20

*Syntax Validity:*20

c. Jawaban:

```java
import java.io.*;
import java.util.*;

class Book {
    private String title;
    private String author;
    private int year;
    private int stock;

    public Book(String title, String author, int year, int stock) {
        this.title = title;
        this.author = author;
        this.year = year;
        this.stock = stock;
    }

    public String getTitle() { return title; }
    public int getStock() { return stock; }

    public void borrow() {
```

Gambar 4. 32 Jawaban Pengujian Token

d. *Feedback* dari AI:

AI Feedbacks:

Analisis Fungsional

Program ini mencoba memenuhi sebagian besar persyaratan tugas. Data buku disimpan dalam file, ada operasi peminjaman/pengembalian, dan penampilan daftar buku. Namun, ada beberapa tujuan yang belum terpenuhi:

- Data anggota tidak disimpan/dibaca dari file (hanya dummy data).
- Tidak ada fitur menampilkan daftar buku yang sedang dipinjam secara keseluruhan (hanya per anggota).
- Konsep OOP seperti inheritance dan polymorphism belum diterapkan.

Secara umum, pendekatan untuk buku dan operasi dasar sudah logis, tetapi struktur untuk manajemen anggota dan relasi peminjaman perlu diperkuat.

Identifikasi Potensi Masalah

1. **Manajemen Data Anggota**:
   - Data anggota tidak disimpan ke file, sehingga hilang saat program ditutup.
   - Tidak ada mekanisme pendaftaran anggota baru (hanya dummy data).
2. **Logika Peminjaman & Pengembalian**:
   - Tidak ada validasi apakah buku benar-benar dipinjam oleh anggota saat pengembalian (bisa mengembalikan buku yang tidak dipinjam).
   - Pencarian buku hanya berdasarkan judul (bisa ambigu jika ada judul sama).
   - Tidak ada pengecekan batas maksimal peminjaman per anggota.
3. **Konsep OOP**:
   - Tidak ada penerapan inheritance/polymorphism (misal: class dasar `Person` untuk `Member` atau `Librarian`).
   - Class `Member` dan `Book` tidak memiliki relasi yang jelas dengan sistem perpustakaan.
4. **Edge Cases**:
   - Jika file `books.txt` tidak ada, program tidak membuat file baru (hanya pesan error).
   - Input non-angka di menu utama menyebabkan exception.
   - Stok buku bisa bertambah tanpa batas saat pengembalian (tidak ada stok maksimal).
5. **Konsistensi Data**:
   - Objek `Book` yang di-load dari file berbeda dengan yang ada di memori, menyebabkan gagal pengembalian (karena perbandingan objek).

Saran dan Pembimbingan

1. **Perbaiki Manajemen Data**:
   - Pertimbangkan untuk menyimpan data anggota ke file (format CSV/JSON).
   - Tambahkan menu untuk pendaftaran anggota baru.
2. **Tingkatkan Logika Operasi**:
   - Di `returnBook()`, pastikan anggota memang meminjam buku tersebut sebelum mengembalikan.
   - Gunakan ID buku atau kombinasi judul-penulis untuk menghindari ambiguasi.
   - Batasi jumlah buku yang bisa dipinjam per anggota.
3. **Terapkan OOP Secara Menyeluruh**:
   - Coba gunakan inheritance (misal: class `Person` sebagai parent `Member`).
   - Manfaatkan interface untuk operasi file (save/load) agar lebih modular.
4. **Handle Edge Cases**:
   - Gunakan `try-catch` untuk input menu agar tidak crash saat input non-angka.
   - Inisialisasi file buku otomatis jika tidak ditemukan.
   - Tambahkan batas maksimal stok buku.
5. **Perbaiki Relasi Objek**:
   - Override method `equals()` di class `Book` untuk membandingkan judul/penulis.
   - Simpan ID buku yang dipinjam (bukan objek) untuk menghindari inkonsistensi.

Penilaian Berdasarkan Rubrik

1. **Input Accuracy (15/20)** : Penanganan input dasar sudah ada, tetapi tidak ada validasi input non-angka di menu dan tidak ada penanganan data anggota baru.
2. **Logic Correctness (14/20)** : Logika peminjaman/pengembalian dasar berfungsi, tetapi ada bug saat pengembalian buku yang tidak dipinjam dan tidak ada batasan stok.
3. **Output Accuracy (16/20)** : Output daftar buku dan buku yang dipinjam sudah akurat, tetapi tidak ada fitur tampilkan semua buku yang dipinjam secara global.
4. **Edge Case Handling (12/20)** : Beberapa edge case seperti file tidak ditemukan atau input invalid tidak ditangani dengan baik.
5. **Syntax Validity (20/20)** : Sintaksis kode valid dan tidak ada error kompilasi.

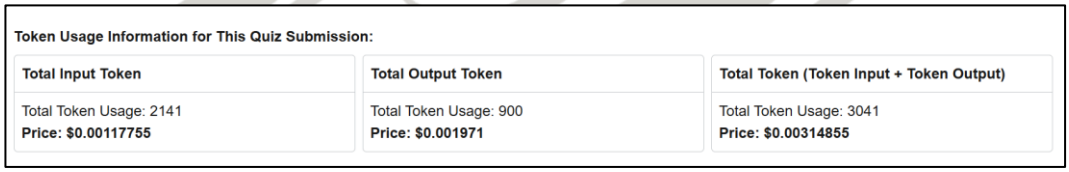
Score: 77 / 100

Gambar 4. 33 Hasil Feedback AI Pengujian Token

Dari hasil pengujian tersebut, sistem mengirimkan kurang lebih 852 kata sebagai token input, yang terdiri dari informasi soal, jawaban, rubrik penilaian, serta jenis bahasa pemrograman yang digunakan. Seluruh elemen ini diinjeksi ke dalam prompt dan dikirimkan ke AI untuk proses evaluasi. Adapun feedback yang dihasilkan oleh AI mencakup sekitar 493 kata sebagai token output. Berdasarkan hasil perhitungan token yang dilacak menggunakan *CallbackManager*, sistem mencatat:

- Token input: 2.141 token
- Token output: 900 token
- Total token: 3.041 token

Maka estimasi biaya yang dihasilkan dari satu kali evaluasi kompleks ini adalah $0.00314855 atau senilai Rp51,29 (lima puluh satu rupiah), dengan rincian sebagai berikut:



Gambar 4. 34 Hasil Pengujian Penggunaan Token dan Estimasi Biaya

Disimpulkan bahwa meskipun sistem diuji menggunakan soal yang kompleks dan jawaban yang panjang, penggunaan token tetap berada dalam batas aman dan jauh di bawah batas maksimum token yang didukung oleh model DeepSeek-R1. Namun demikian, perlu dipertimbangkan penambahan fitur pembatasan panjang jawaban secara sistematis guna mencegah potensi kegagalan evaluasi apabila suatu saat jawaban mahasiswa melebihi batas token maksimal.

BAB V
Simpulan dan Saran

5.1 Kesimpulan

Berdasarkan penelitian yang telah dilakukan, dapat disimpulkan bahwa sistem yang dikembangkan berhasil mencapai tujuan yang telah direncanakan sebelumnya. Sistem ini berhasil dibangun dan terintegrasi dengan *LLM DeepSeek-R1* menggunakan *framework Langchain* dan *Next.js*. Sistem mampu memberikan umpan balik secara *real time* kepada mahasiswa, dan hasil evaluasi dari AI telah diuji oleh mentor serta mendapatkan respon positif dari mahasiswa maupun mentor. Selain itu, sistem juga terbukti mampu mempercepat proses evaluasi dalam mata kuliah pemrograman berdasarkan *feedback* yang disampaikan oleh mentor, sehingga memberikan kemudahan bagi mentor dan mahasiswa.

Sistem juga terbukti dapat berjalan dengan baik berdasarkan hasil pengujian performa sistem. Pengujian menunjukkan bahwa hasil dari *response time* masih dalam batas waktu yang wajar dan stabil, serta *throughput* dan skalabilitas sistem mampu menangani sejumlah besar pengguna secara bersamaan tanpa mengalami penurunan performa yang signifikan. Hasil pengujian *load time* sistem juga berada dalam kategori sedang. Hasil ini menunjukkan bahwa sistem tidak hanya berhasil secara fungsional, namun layak digunakan dalam sisi teknis juga.

5.2 Saran

Berdasarkan hasil implementasi dan pengujian sistem, terdapat beberapa saran pengembangan yang dapat dilakukan untuk penelitian selanjutnya dalam upaya meningkatkan fungsionalitas dan kenyamanan pengguna, antara lain:

- **Peningkatan Desain Antarmuka Pengguna (UI):**
  Tampilan *website* sebaiknya ditingkatkan agar lebih menarik secara visual, namun tetap memperhatikan aspek performa dari segi *front-end*. Saran ini berdasarkan *feedback* dari beberapa mahasiswa yang memberikan masukan tentang desain antarmuka yang kurang menarik.
- **Pemindahan Rubrik Penilaian dari Level *Quiz* ke Level *Exam:***
  Rubrik penilaian yang sebelumnya berada di dalam tabel *Quiz* dapat dipindahkan ke tabel *Exam* agar mentor tidak perlu menambahkan rubrik yang

sama pada setiap soal secara berulang. Hal ini dapat mempermudah mentor dalam menyusun *Exam* dan *Quiz* dengan mencantumkan rubrik satu kali saja.

- **Pembuatan Sistem *Cache* untuk *Prompt* AI**

  Tujuan dari dibuatnya sistem *cache* adalah untuk menghemat biaya yang digunakan untuk input token ke AI. Dari data yang ditunjukan di dokumentasi resmi deepseek, jika token input menggunakan sistem cache hit maka biaya yang dihemat hingga 90%.

- **Pembuatan Sistem Kelas untuk Mahasiswa dan Mentor**

  Kelas difungsikan agar ketika mentor membuat ujian, sistem tidak otomatis mendistribusikan ke seluruh mahasiswa yang terdaftar, dengan adanya kelas, proses pembuatan ujian, soal dan pengerjaan dapat teroganisir.

- **Pembuatan Fitur Pemantauan Aktivitas Mahasiswa**

  Fitur ini memungkinkan pemantauan status pengerjaan ujian secara *real time*, seperti melihat mahasiswa yang hanya melihat soal, sedang aktif mengerjakan, atau sudah menyelesaikan ujian.

- **Pembuatan Fitur Limitasi Input Jawaban**

  Fitur ini membatasi mahasiswa agar tidak dapat menuliskan jawaban yang kira-kira dapat melebihi batas penggunaan maksimum token dari DeepSeek-R1 yaitu 64.000 token.

- **Optimisasi Prompt**

  Optimisasi ini bertujuan agar *output* yang dihasilkan oleh AI tidak menggunakan terlalu banyak token, sehingga mengurangi biaya pengeluaran, selain itu ditujukan untuk menjaga kualitas dan konsistensi *output* agar tidak halusinasi.

DAFTAR PUSTAKA

Aprilia, R., Harahap, N. S., Yanto, F., & Yusra. (2024). SISTEM TANYA JAWAB ILMU KEISLAMAN DENGAN MODEL LARGE . *JIRE (Jurnal Informatika & Rekayasa Elektronika)* , 80-87.

Cahyanto, I., & Sonjaya, N. S. (2024). MEMANFAATKAN KECERDASAN BUATAN UNTUK MENINGKATKAN PROSES EVALUASI PEMBELAJARAN DI SEKOLAH MENENGAH: SUATU TINJAUAN TERHADAP POTENSI DAN TANTANGANNYA. *Edum Journal*, 110-121.

DeepSeek. (2025). *DeepSeek-R1-0528 Release*. Diambil kembali dari DeepSeek API Docs: https://api-docs.deepseek.com/news/news250528

DeepSeek. (2025). *DeepSeek-V3-0324 Release*. Diambil kembali dari DeepSeek API Docs: https://api-docs.deepseek.com/news/news250325

DeepSeek. (2025). *Models & Pricing*. Diambil kembali dari DeepSeek API Docs: https://api-docs.deepseek.com/quick_start/pricing

DeepSeek. (2025). *The Temperature Parameter.* Diambil kembali dari DeepSeek API Docs: https://api-docs.deepseek.com/quick_start/parameter_settings

DeepSeek. (2025). *Your First API Call*. Diambil kembali dari DeepSeek API Docs: https://api-docs.deepseek.com/

DeepSeek-AI. (2025). DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning. *arXiv preprint arXiv:2501.12948v1*, 3-22.

Dewi, A. O. (2020). Kecerdasan Buatan sebagai Konsep Baru pada Perpustakaan. *ANUVA*, 453-461.

Gao, T., Jin, J., Ke, Z. T., & Moryoussef, G. (2025). A Comparison of DeepSeek and Other LLMs. *arXiv:2502.03688v1*, 1-21.

Gulwani, S., Radicek, I., & Zuleger, F. (2014). Feedback Generation for Performance Problems in Introductory Programming Assignments. *arXiv preprint arXiv:1403.4064v2*, 1-12.

Gupta, P., Ding, B., Guan, C., & Ding, D. (2024). Generative AI: A systematic review using topic modelling techniques . *Data and Information Management* , 1-66.

Hadid, S., Ramadhani, U., Suari, S. D., & Putri, A. G. (2024). Analisis Dampak Penggunaan Chatbot AIDalam Pembelajaran Di Kalangan Mahasiswa PGSD Universitas Jambi. *Jurnal Pendidikan Terapan*, 160-166.

Harimurti, R., Qoirah, A., Nh, A. I., & Asmunin. (2018). Pengembangan Fitur Penilaian dan Perangkingan pada Automatic Programming Assessment Tools. *Journal Information Engineering and Educational Technology*, 96-100.

Hericko, T., Sumak, B., & Brdnik, S. (2022). Towards representative web performance measurements with Google Lighthouse. *ResearchGate*, 1-4.

Indrianto. (2023). PERFORMANCE TESTING ON WEB INFORMATION SYSTEM USING APACHE JMETER AND BLAZEMETER. *Jurnal Ilmiah Ilmu Terapan*, 138-149.

Jimmy. (2024). Pengembangan Chatbot Berbasis Generative AI Untuk Memudahkan Mahasiswa Memahami Pedoman Penyusunan Dan Penulisan Skripsi Di Universitas IBBI. *Jurnal Ilmiah Sains dan Teknologi*, 19-32.

Kerner, S. M. (2025, Februari 6). *DeepSeek explained: Everything you need to know*. Diambil kembali dari TechTarget: https://www.techtarget.com/whatis/feature/DeepSeek-explained-Everything-you-need-to-know

Khadija, M. A., Paradhita, A. N., Purbayu, A., Bawono, S. A., Aziz, A., Haryati, S., . . . [. (2024). Backend Programming Techniques in the Development of Resource Manager Features in VRMS Systems Based on ORM Prisma. *Jurnal Pendidikan Teknologi Informasi*, 830-843.

Levy, M., Jacoby, A., & Goldber, Y. (2024). Same Task, More Tokens: the Impact of Input Length on the Reasoning Performance of Large Language Models. *arXiv:2402.14848v2*, 1-15.

Maharani, P. (2025). Pengembangan Website PT. Rantangin Digital Indonesia Menggunakan Framework Next Js dan Tailwind CSS. *Publikasi Teknik Informatika dan Jaringan* , 129-137.

Manik, M. M. (2025). ChatGPT vs. DeepSeek: A Comparative Study on AI-Based Code Generation . *arXiv:2502.18467*, 1-5.

Manjavacas, E., Karsdorp, F., Burtenshaw, B., & Kestemont, M. (t.thn.). Synthetic literature: Writing science fiction in a co-creative process. *Computational Linguistics & Psycholinguistics Research Center*, 29-37.

Muhajir, M. D., Prastiti, N., & Koeshardianto, M. (2025). IMPLEMENTASI CHATBOT MENGGUNAKAN FRAMEWORK LANGCHAIN BERBASIS LLM GPT (STUDI KASUS : PANDUAN AKADEMIK UNIVERSITAS TRUNOJOYO). *JATI (Jurnal Mahasiswa Teknik Informatika)*, 2151-2158.

Nugroho, D. S., Putra, A. A., Luqmanulhakim, H., Pratama, N. F., Saputra, B. A., & Alit, R. (2023). Inovasi Pembaruan Desain Website Penyederhana Link Unesa . *Jurnal Ilmu Teknik* , 35-40.

Paz, S., & Bernardino, J. (2018). Comparative Analysis of Web Platform Assessment Tools. *ResearchGate*, 116-125.

Peeperkorn, M., Kouwenhoven, T., Brown, D., & Jordanous, A. (2024). Is Temperature the Creativity Parameter of Large Language Models? *arXiv:2405.00492v1* , 1-10.

Permatasari, A., & Suhendi. (2020). Rancang Bangun Sistem Informasi Pengelolaan Talent Film berbasis Aplikasi Web. *Jurnal Informatika Terpadu*, 29-37.

Prasetyo, D. A., & Susetyo, Y. A. (2022). Implementasi Information Schema Database Pada Postgre SQL Untuk Pembuatan Tabel Informasi Dengan Menggunakan Python Di PT XYZ. *Jurnal Teknik Informatika dan Sistem Informasi* , 1961-1972.

Riezky, A. K., Andrianty, S. N., & Suji, C. G. (2020). GAMBARAN PROSES PEMBERIAN UMPAN BALIK PADA PROGRAM STUDI PENDIDIKAN DOKTER FAKULTAS KEDOKTERAN UNIVERSITAS ABULYATAMA. *Jurnal Ilmu Kedokteran Dan Kesehatan*, 574-578.

Samsudin, Indrawan, & Mulyati, S. (2020). Perancangan Sistem Informasi Pembelajaran Algoritma dan Pemrograman Berbasis Web pada Program Studi Teknik Informatika STMIK ERESHA. *Informatika Universitas Pamulang*, 521-528.

Soepono, R. (2023). Wireshark: An Effective Tool for Network Analysis. *ResearchGate*, 1-12.

Summit, S. (2023). Memahami Dasar-Dasar HTML dan CSS: Fondasi. *Teknologiterkini.org*, 1-19.

Suprayogi, B., & Rahmanesa, A. (2019). Penerapan Framework Bootstrap dalam Sistem Informasi Pendidikan SMA Negeri 1 Pacet Cianjur Jawa Barat. *Jurnal Teknologi Informasi dan Komunikasi*, 119-127.

Triawan, A., & Siboro, A. R. (2021). Penerapan Application Programming Interface (API) Pada Push Notification Untuk Informasi Monitoring Stok Barang Minim. *Jurnal Ilmiah Teknologi Informasi dan Sains*, 107-114.

Windiart, I. S., Bahri, S., & Prabowo, A. (2023). Melangkah Majudengan Teknologi Generative AI:Peningkatan Kompetensi Kepala Sekolah SMP di Kota Palangkaraya. *Jurnal Pengabdian Kepada Masyarakat*, 46-52.

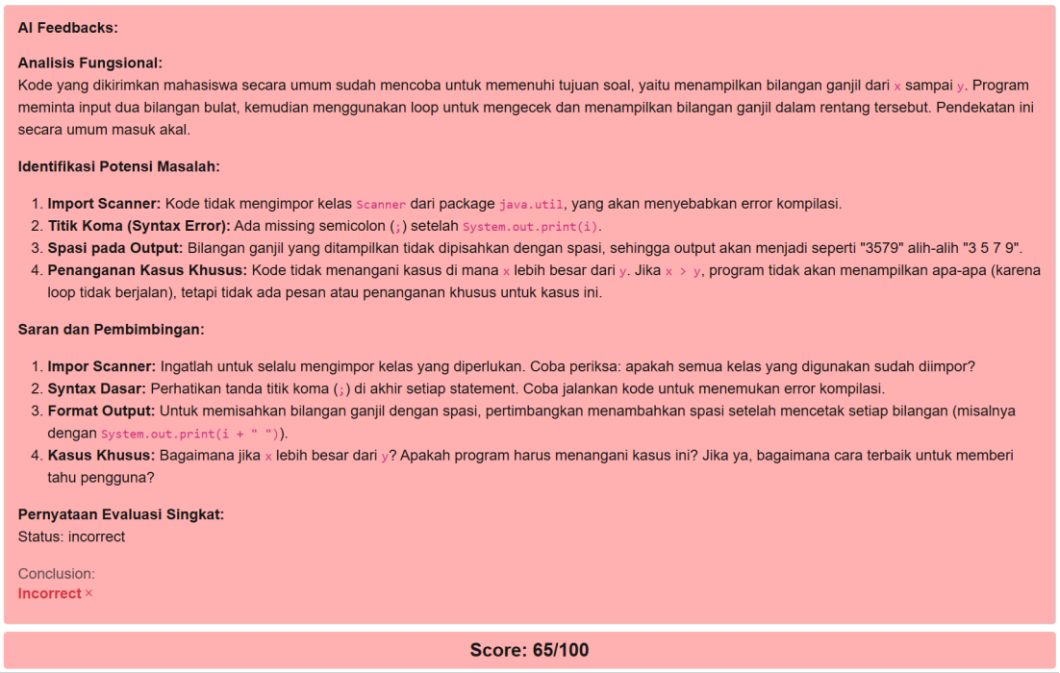
Wongvorachan, T., Lai, K. W., Bulut, O., Tsai, Y.-S., & Chen, G. (2022). Artificial Intelligence: Transforming the Future of Feedback in Education. *Journal of Applied Testing Technology*, 95-116.

Zimmerman, J. W., Hudon, D., & Cramer, K. (2025). Tokens, the oft-overlooked appetizer: Large language models, the distributional hypothesis, and meaning. *arXiv:2412.10924v4* , 1-55.

Zulkarnain, N. S., & Yunus, M. M. (2023). Primary Teachers' Perspectives on Using Artificial Intelligence Technology in English as a Second Language

Teaching and Learning: A Systematic Review. *nternational Journal of Academic Research in Progressive Education and Development*, 861-875.

LAMPIRAN A: HASIL *FEEDBACK* AI



AI Feedbacks:

Analisis Fungsional:
Kode yang dikirimkan mahasiswa secara umum sudah mencoba untuk memenuhi tujuan soal, yaitu menampilkan bilangan ganjil dari `x` sampai `y`. Program meminta input dua bilangan bulat, kemudian menggunakan loop untuk mengecek dan menampilkan bilangan ganjil dalam rentang tersebut. Pendekatan ini secara umum masuk akal.

Identifikasi Potensi Masalah:

1. **Import Scanner:** Kode tidak mengimpor kelas `Scanner` dari package `java.util`, yang akan menyebabkan error kompilasi.
2. **Titik Koma (Syntax Error):** Ada missing semicolon (`;`) setelah `System.out.print(i)`.
3. **Spasi pada Output:** Bilangan ganjil yang ditampilkan tidak dipisahkan dengan spasi, sehingga output akan menjadi seperti "3579" alih-alih "3 5 7 9".
4. **Penanganan Kasus Khusus:** Kode tidak menangani kasus di mana `x` lebih besar dari `y`. Jika `x > y`, program tidak akan menampilkan apa-apa (karena loop tidak berjalan), tetapi tidak ada pesan atau penanganan khusus untuk kasus ini.

Saran dan Pembimbingan:

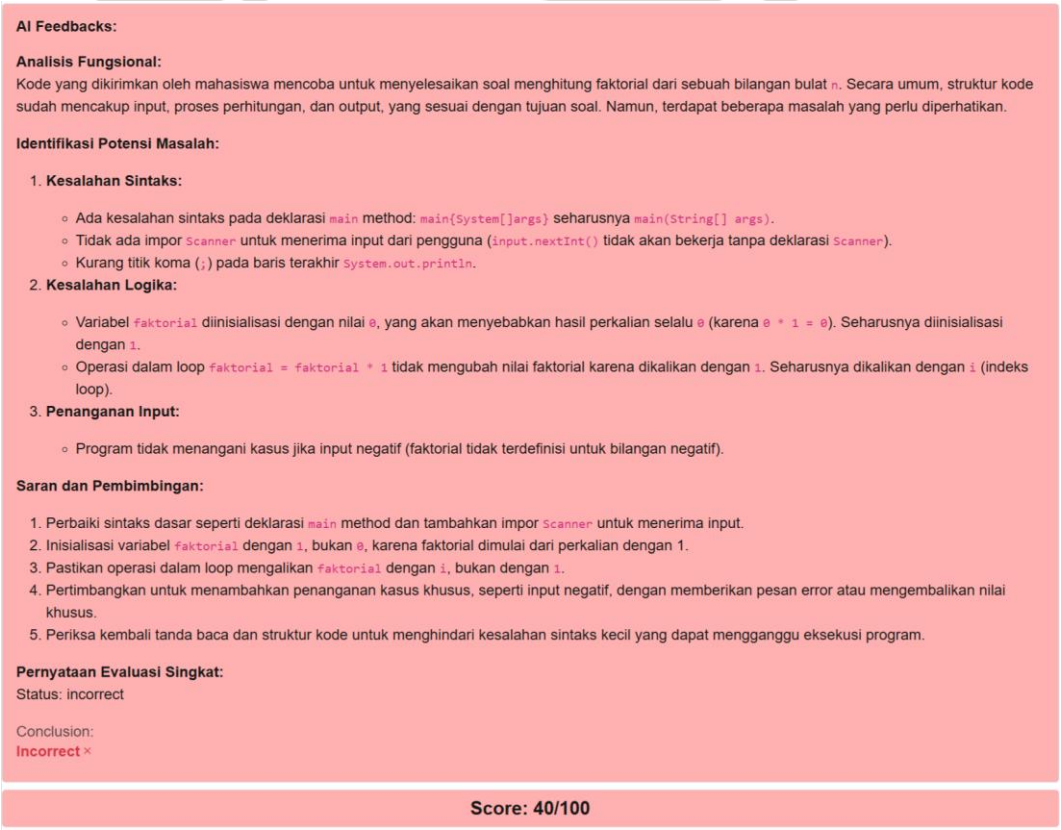
1. **Impor Scanner:** Ingatlah untuk selalu mengimpor kelas yang diperlukan. Coba periksa: apakah semua kelas yang digunakan sudah diimpor?
2. **Syntax Dasar:** Perhatikan tanda titik koma (`;`) di akhir setiap statement. Coba jalankan kode untuk menemukan error kompilasi.
3. **Format Output:** Untuk memisahkan bilangan ganjil dengan spasi, pertimbangkan menambahkan spasi setelah mencetak setiap bilangan (misalnya dengan `System.out.print(i + " ")`).
4. **Kasus Khusus:** Bagaimana jika `x` lebih besar dari `y`? Apakah program harus menangani kasus ini? Jika ya, bagaimana cara terbaik untuk memberi tahu pengguna?

Pernyataan Evaluasi Singkat:
Status: incorrect

Conclusion:
Incorrect ×

Score: 65/100

Gambar A. 1 Hasil *Feedback* AI Responden 1 Nomor 1



AI Feedbacks:

Analisis Fungsional:
Kode yang dikirimkan oleh mahasiswa mencoba untuk menyelesaikan soal menghitung faktorial dari sebuah bilangan bulat `n`. Secara umum, struktur kode sudah mencakup input, proses perhitungan, dan output, yang sesuai dengan tujuan soal. Namun, terdapat beberapa masalah yang perlu diperhatikan.

Identifikasi Potensi Masalah:

1. **Kesalahan Sintaks:**
   - Ada kesalahan sintaks pada deklarasi `main` method: `main{System[]args}` seharusnya `main(String[] args)`.
   - Tidak ada impor `Scanner` untuk menerima input dari pengguna (`input.nextInt()` tidak akan bekerja tanpa deklarasi `Scanner`).
   - Kurang titik koma (`;`) pada baris terakhir `System.out.println`.
2. **Kesalahan Logika:**
   - Variabel `faktorial` diinisialisasi dengan nilai `0`, yang akan menyebabkan hasil perkalian selalu `0` (karena `0 * 1 = 0`). Seharusnya diinisialisasi dengan `1`.
   - Operasi dalam loop `faktorial = faktorial * i` tidak mengubah nilai faktorial karena dikalikan dengan `1`. Seharusnya dikalikan dengan `i` (indeks loop).
3. **Penanganan Input:**
   - Program tidak menangani kasus jika input negatif (faktorial tidak terdefinisi untuk bilangan negatif).

Saran dan Pembimbingan:

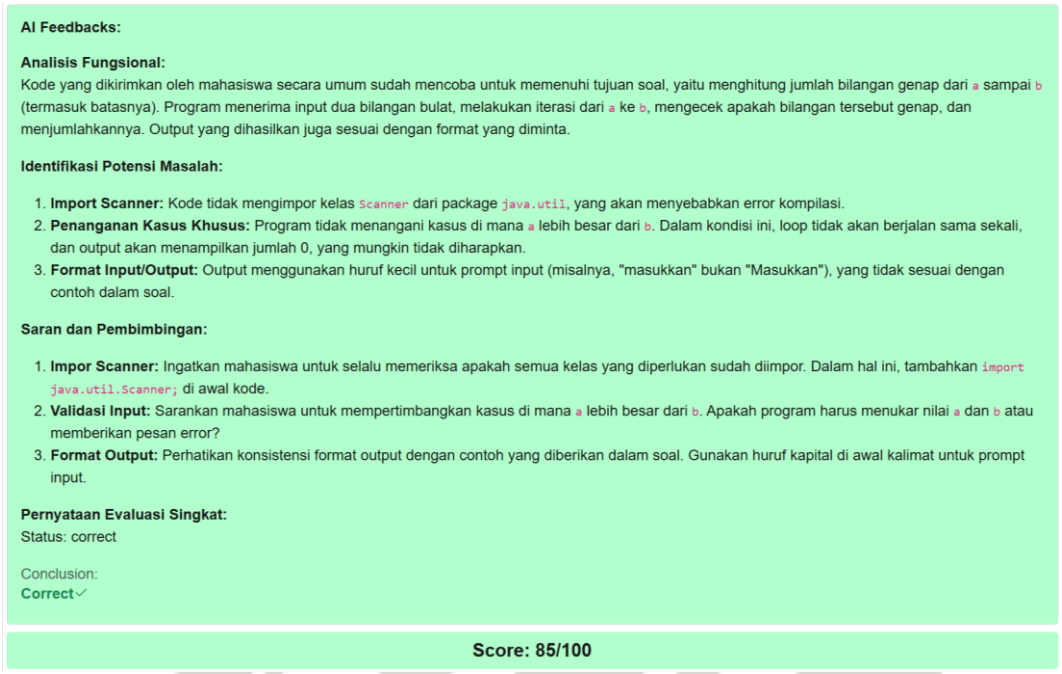
1. Perbaiki sintaks dasar seperti deklarasi `main` method dan tambahkan impor `Scanner` untuk menerima input.
2. Inisialisasi variabel `faktorial` dengan `1`, bukan `0`, karena faktorial dimulai dari perkalian dengan 1.
3. Pastikan operasi dalam loop mengalikan `faktorial` dengan `i`, bukan dengan `1`.
4. Pertimbangkan untuk menambahkan penanganan kasus khusus, seperti input negatif, dengan memberikan pesan error atau mengembalikan nilai khusus.
5. Periksa kembali tanda baca dan struktur kode untuk menghindari kesalahan sintaks kecil yang dapat mengganggu eksekusi program.

Pernyataan Evaluasi Singkat:
Status: incorrect

Conclusion:
Incorrect ×

Score: 40/100

Gambar A. 2 Hasil *Feedback* AI Responden 1 Nomor 2



AI Feedbacks:

Analisis Fungsional:
Kode yang dikirimkan oleh mahasiswa secara umum sudah mencoba untuk memenuhi tujuan soal, yaitu menghitung jumlah bilangan genap dari `a` sampai `b` (termasuk batasnya). Program menerima input dua bilangan bulat, melakukan iterasi dari `a` ke `b`, mengecek apakah bilangan tersebut genap, dan menjumlahkannya. Output yang dihasilkan juga sesuai dengan format yang diminta.

Identifikasi Potensi Masalah:

1. **Import Scanner:** Kode tidak mengimpor kelas `Scanner` dari package `java.util`, yang akan menyebabkan error kompilasi.
2. **Penanganan Kasus Khusus:** Program tidak menangani kasus di mana `a` lebih besar dari `b`. Dalam kondisi ini, loop tidak akan berjalan sama sekali, dan output akan menampilkan jumlah 0, yang mungkin tidak diharapkan.
3. **Format Input/Output:** Output menggunakan huruf kecil untuk prompt input (misalnya, "masukkan" bukan "Masukkan"), yang tidak sesuai dengan contoh dalam soal.

Saran dan Pembimbingan:

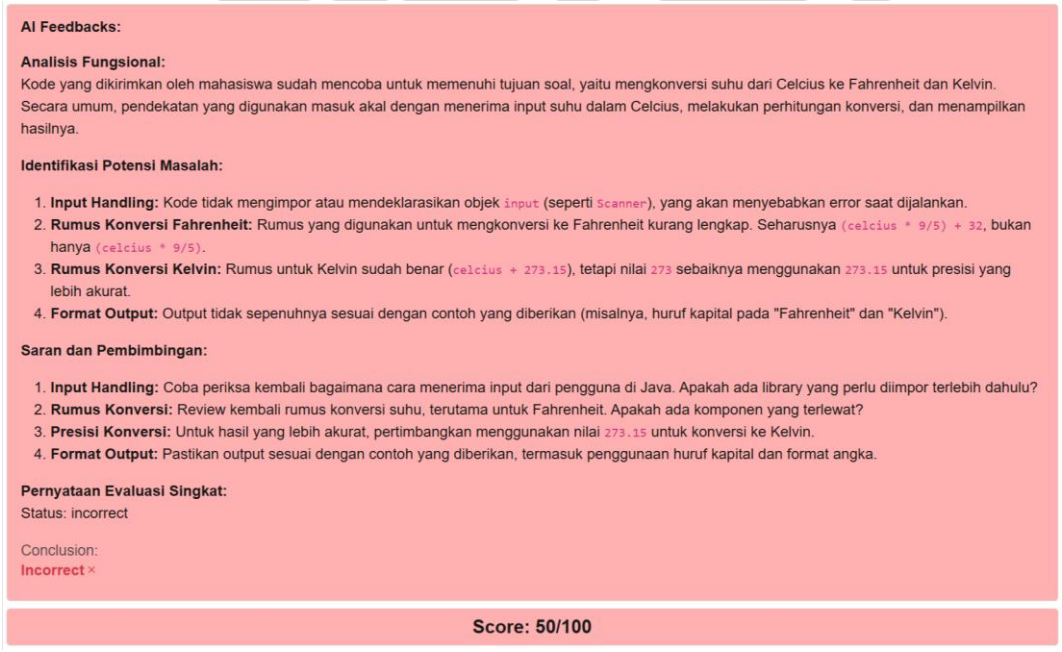
1. **Impor Scanner:** Ingatkan mahasiswa untuk selalu memeriksa apakah semua kelas yang diperlukan sudah diimpor. Dalam hal ini, tambahkan `import java.util.Scanner;` di awal kode.
2. **Validasi Input:** Sarankan mahasiswa untuk mempertimbangkan kasus di mana `a` lebih besar dari `b`. Apakah program harus menukar nilai `a` dan `b` atau memberikan pesan error?
3. **Format Output:** Perhatikan konsistensi format output dengan contoh yang diberikan dalam soal. Gunakan huruf kapital di awal kalimat untuk prompt input.

Pernyataan Evaluasi Singkat:
Status: correct

Conclusion:
Correct ✓

Score: 85/100

Gambar A. 3 Hasil *Feedback* AI Responden 1 Nomor 3



AI Feedbacks:

Analisis Fungsional:
Kode yang dikirimkan oleh mahasiswa sudah mencoba untuk memenuhi tujuan soal, yaitu mengkonversi suhu dari Celcius ke Fahrenheit dan Kelvin. Secara umum, pendekatan yang digunakan masuk akal dengan menerima input suhu dalam Celcius, melakukan perhitungan konversi, dan menampilkan hasilnya.

Identifikasi Potensi Masalah:

1. **Input Handling:** Kode tidak mengimpor atau mendeklarasikan objek `input` (seperti `Scanner`), yang akan menyebabkan error saat dijalankan.
2. **Rumus Konversi Fahrenheit:** Rumus yang digunakan untuk mengkonversi ke Fahrenheit kurang lengkap. Seharusnya `(celcius * 9/5) + 32`, bukan hanya `(celcius * 9/5)`.
3. **Rumus Konversi Kelvin:** Rumus untuk Kelvin sudah benar (`celcius + 273.15`), tetapi nilai `273` sebaiknya menggunakan `273.15` untuk presisi yang lebih akurat.
4. **Format Output:** Output tidak sepenuhnya sesuai dengan contoh yang diberikan (misalnya, huruf kapital pada "Fahrenheit" dan "Kelvin").

Saran dan Pembimbingan:

1. **Input Handling:** Coba periksa kembali bagaimana cara menerima input dari pengguna di Java. Apakah ada library yang perlu diimpor terlebih dahulu?
2. **Rumus Konversi:** Review kembali rumus konversi suhu, terutama untuk Fahrenheit. Apakah ada komponen yang terlewat?
3. **Presisi Konversi:** Untuk hasil yang lebih akurat, pertimbangkan menggunakan nilai `273.15` untuk konversi ke Kelvin.
4. **Format Output:** Pastikan output sesuai dengan contoh yang diberikan, termasuk penggunaan huruf kapital dan format angka.

Pernyataan Evaluasi Singkat:
Status: incorrect

Conclusion:
Incorrect ×

Score: 50/100

Gambar A. 4 Hasil *Feedback* AI Responden 1 Nomor 4

AI Feedbacks:

Analisis Fungsional:
Kode yang dikirimkan oleh mahasiswa secara umum sudah mencoba untuk memenuhi tujuan soal, yaitu mencetak pola segitiga siku-siku bintang sebanyak `n` baris. Pendekatan menggunakan nested loop (`for` di dalam `for`) sudah tepat untuk kasus ini, di mana loop luar mengontrol baris dan loop dalam mengontrol jumlah bintang per baris.

Identifikasi Potensi Masalah:

1. **Format Output:** Output yang dihasilkan tidak sesuai dengan contoh yang diberikan dalam soal. Contoh soal menunjukkan pola dengan spasi di antara bintang (misal: `* * * **`), sedangkan kode ini mencetak bintang tanpa spasi (misal: `* ** ***`).
2. **Penanganan Input:** Kode tidak menangani kasus jika input `n` bernilai negatif atau nol. Meskipun soal tidak secara eksplisit meminta penanganan ini, termasuk praktik baik untuk memvalidasi input.
3. **Komentar Kode:** Tidak ada komentar atau penjelasan dalam kode, meskipun ini tidak wajib.

Saran dan Pembimbingan:

1. **Perbaiki Format Output:** Coba perhatikan kembali contoh output yang diminta dalam soal. Apakah bintang-bintang harus dipisahkan oleh spasi? Jika iya, bagaimana cara memodifikasi loop dalam untuk menambahkan spasi?
2. **Validasi Input:** Pertimbangkan untuk menambahkan pengecekan jika `n` kurang dari atau sama dengan 0. Apa yang seharusnya program lakukan dalam kasus tersebut?
3. **Komentar Kode:** Tambahkan komentar singkat untuk menjelaskan tujuan dari setiap bagian kode, terutama jika logikanya kompleks. Ini akan membantu orang lain (atau diri Anda di masa depan) memahami kode dengan lebih mudah.

Pernyataan Evaluasi Singkat:
Status: incorrect

Conclusion:
Incorrect ×

Score: 65/100

Gambar A. 5 Hasil *Feedback* AI Responden 1 Nomor 5

AI Feedbacks:

Analisis Fungsional:
Kode yang dikirimkan mahasiswa secara umum sudah mencoba untuk memenuhi tujuan soal, yaitu menampilkan bilangan ganjil dari `x` sampai `y`. Program meminta input dua bilangan bulat, kemudian menggunakan loop untuk memeriksa dan menampilkan bilangan ganjil dalam rentang tersebut. Pendekatan ini secara logika sudah benar.

Identifikasi Potensi Masalah:

1. **Format Output:**
   - Ada kesalahan ketik pada output: `"sampai"` seharusnya dipisahkan dengan spasi sebelum dan sesudahnya (`" sampai "`).
   - Setiap bilangan ganjil yang ditampilkan tidak dipisahkan dengan spasi, sehingga output akan menjadi seperti `3579` alih-alih `3 5 7 9`.
2. **Penanganan Kasus Khusus:**
   - Program tidak menangani kasus di mana `x` lebih besar dari `y`. Jika `x > y`, loop tidak akan berjalan sama sekali, tetapi tidak ada pesan yang memberi tahu pengguna tentang hal ini.
3. **Komentar atau Penjelasan Kode:**
   - Tidak ada komentar yang menjelaskan logika atau bagian penting dari kode.
4. **Kerapian Kode:**
   - Ada titik koma (`;`) yang hilang setelah `System.out.println`.
   - Nama kelas `angkaganjil` sebaiknya menggunakan huruf kapital di awal sesuai konvensi Java (`AngkaGanjil`).

Saran dan Pembimbingan:

1. **Perbaiki Format Output:**
   - Tambahkan spasi setelah setiap bilangan ganjil yang dicetak, misalnya dengan `System.out.print(i + " ");`.
   - Perbaiki string output agar sesuai dengan contoh soal, misalnya: `"Bilangan ganjil dari " + x + " sampai " + y + ":"`.
2. **Tambahkan Penanganan Kasus Khusus:**
   - Pertimbangkan untuk menambahkan pengecekan jika `x > y`, lalu beri pesan seperti `"Nilai x harus lebih kecil atau sama dengan y"`.
3. **Tambahkan Komentar (Opsional):**
   - Beri komentar singkat untuk menjelaskan tujuan loop atau kondisi `if`, misalnya: `// Cek apakah bilangan ganjil.`
4. **Perbaiki Struktur Kode:**
   - Pastikan semua pernyataan diakhiri dengan titik koma.
   - Gunakan konvensi penamaan yang benar untuk kelas (PascalCase).

Pernyataan Evaluasi Singkat:
Status: incorrect

Conclusion:
Incorrect ×

Score: 65/100

Gambar A. 6 Hasil *Feedback* AI Responden 2 Nomor 1

AI Feedbacks:

Analisis Fungsional:
Kode yang dikirimkan oleh mahasiswa secara umum sudah mencoba untuk memenuhi tujuan soal, yaitu menghitung faktorial dari sebuah bilangan bulat `n`. Program menggunakan loop `for` untuk mengalikan bilangan dari 1 hingga `n`, yang merupakan pendekatan yang benar untuk menghitung faktorial. Output juga mencoba menampilkan hasil perhitungan sesuai format yang diminta.

Identifikasi Potensi Masalah:

1. **Format Output:** Ada sedikit ketidaksesuaian dalam format output. Soal meminta format seperti "Faktorial dari 5 adalah 120", tetapi kode menghasilkan "jumlah faktorial dari 5 adalah 120". Ini tidak sesuai dengan contoh yang diberikan.
2. **Penanganan Kasus Khusus:** Kode tidak menangani kasus khusus seperti input negatif. Faktorial tidak terdefinisi untuk bilangan negatif, tetapi program akan tetap menghitung dan menghasilkan nilai 1 (karena loop tidak berjalan jika `n <= 0`), yang bisa menyesatkan.
3. **Tipe Data:** Meskipun menggunakan `long` untuk menyimpan hasil faktorial (yang baik untuk menghindari overflow), perlu dipertimbangkan apakah `long` cukup untuk nilai `n` yang sangat besar. Namun, ini bukan masalah utama untuk kasus sederhana.

Saran dan Pembimbingan:

1. **Perbaiki Format Output:** Sarankan mahasiswa untuk mencocokkan format output dengan contoh yang diberikan dalam soal. Perhatikan penggunaan kata "Faktorial" bukan "jumlah faktorial".
2. **Tambahkan Validasi Input:** Arahkan mahasiswa untuk memikirkan penanganan input negatif. Misalnya, dengan menambahkan pesan error jika input tidak valid.
3. **Komentar Kode:** Meskipun tidak wajib, tambahkan komentar sederhana untuk menjelaskan logika perhitungan faktorial agar kode lebih mudah dipahami.

Pernyataan Evaluasi Singkat:
Status: correct

Conclusion:
Correct✓

Score: 85/100

Gambar A. 7 Hasil *Feedback* AI Responden 2 Nomor 2

AI Feedbacks:

Analisis Fungsional:
Kode yang dikirimkan oleh mahasiswa sudah mencoba untuk memenuhi tujuan soal dengan baik. Program meminta input dua bilangan bulat `a` dan `b`, kemudian menghitung jumlah semua bilangan genap dalam rentang tersebut (termasuk batasnya). Pendekatan yang digunakan, yaitu menggunakan loop dan kondisi untuk mengecek bilangan genap, secara umum sudah masuk akal dan sesuai dengan instruksi soal.

Identifikasi Potensi Masalah:

1. **Penanganan Kasus Khusus:** Kode tidak mempertimbangkan kasus di mana `a` lebih besar dari `b`. Jika `a > b`, loop tidak akan berjalan sama sekali, dan output akan menampilkan jumlah 0, yang mungkin tidak diharapkan. Soal tidak secara eksplisit menyebutkan batasan ini, tetapi penanganan kasus seperti ini bisa menjadi nilai tambah.
2. **Input Negatif:** Program bisa menerima input negatif, tetapi tidak ada pengecekan apakah input negatif diperbolehkan atau tidak. Ini mungkin atau mungkin tidak menjadi masalah tergantung pada konteks soal.
3. **Komentar:** Kode tidak memiliki komentar atau penjelasan, meskipun ini tidak wajib, tetapi bisa membantu untuk memahami logika lebih cepat.

Saran dan Pembimbingan:

1. **Kasus `a > b`:** Coba pikirkan apa yang seharusnya terjadi jika `a` lebih besar dari `b`. Apakah program harus menukar nilai `a` dan `b`, atau memberikan pesan error? Diskusikan dengan instruktur atau teman untuk memastikan kebutuhan soal.
2. **Input Validasi:** Pertimbangkan untuk menambahkan validasi input, misalnya memastikan `a` dan `b` adalah bilangan bulat dan tidak menyebabkan masalah dalam perhitungan.
3. **Komentar:** Mulailah kebiasaan untuk menambahkan komentar singkat yang menjelaskan tujuan dari blok kode utama, seperti inisialisasi variabel atau loop. Ini akan membantu orang lain (atau dirimu sendiri di masa depan) memahami kode lebih cepat.

Pernyataan Evaluasi Singkat:
Status: correct

Conclusion:
Correct✓

Score: 85/100

Gambar A. 8 Hasil *Feedback* AI Responden 2 Nomor 3

AI Feedbacks:

Analisis Fungsional:
Kode yang dikirimkan oleh mahasiswa sudah mencoba menjawab tujuan soal dengan baik. Program meminta input suhu dalam Celcius, melakukan validasi input, mengkonversi ke Fahrenheit dan Kelvin, serta menampilkan hasilnya. Pendekatan yang digunakan secara umum sudah masuk akal dan sesuai dengan instruksi soal.

Identifikasi Potensi Masalah:

1. **Validasi Input:** Program sudah mengecek apakah suhu lebih rendah dari -273.15°C, yang merupakan batas bawah suhu valid (nol absolut). Namun, tidak ada penanganan jika input bukan angka (misalnya, huruf atau simbol). Ini bisa menyebabkan program error jika input tidak sesuai tipe data `double`.
2. **Format Output:** Output yang ditampilkan sudah benar, tetapi tidak persis seperti contoh dalam soal (contoh soal menampilkan semua hasil dalam satu baris, sedangkan kode mahasiswa menampilkan dalam dua baris terpisah).
3. **Komentar Kode:** Ada komentar yang menjelaskan bagian-bagian penting, tetapi bisa lebih lengkap untuk menjelaskan rumus konversi atau alasan validasi suhu.

Saran dan Pembimbingan:

1. **Penanganan Input Tidak Valid:** Coba pikirkan bagaimana cara menangani input yang bukan angka (misalnya menggunakan `try-catch` atau metode lain untuk memastikan input selalu valid).
2. **Format Output:** Perhatikan kembali format output yang diminta soal. Apakah hasil harus ditampilkan dalam satu baris atau boleh terpisah? Sesuaikan dengan contoh yang diberikan.
3. **Komentar Tambahan:** Tambahkan komentar untuk menjelaskan rumus konversi atau mengapa -273.15°C dipilih sebagai batas validasi. Ini akan membuat kode lebih mudah dipahami oleh orang lain.
4. **Struktur Kode:** Kode sudah rapi dan terstruktur, tetapi pertimbangkan untuk memisahkan logika konversi dan validasi ke dalam metode terpisah agar lebih modular.

Pernyataan Evaluasi Singkat:
Status: correct

Conclusion:
Correct ✓

Score: 85/100

Gambar A. 9 Hasil *Feedback* AI Responden 2 Nomor 4

AI Feedbacks:

Analisis Fungsional:
Kode yang dikirimkan oleh mahasiswa sudah mencoba untuk memenuhi tujuan soal, yaitu mencetak pola segitiga siku-siku bintang (*) sebanyak `n` baris. Program meminta input dari pengguna dan menggunakan perulangan bersarang untuk mencetak pola yang diminta. Secara umum, pendekatan yang digunakan sudah masuk akal.

Identifikasi Potensi Masalah:

1. **Format Output:** Output yang dihasilkan tidak sesuai dengan contoh yang diberikan dalam soal. Contoh soal menunjukkan pola dengan spasi antara bintang (misalnya: `* * * ** * *** * **** *`), sedangkan kode ini mencetak bintang tanpa spasi (misalnya: `* ** *** **** *****`).
2. **Penanganan Input Negatif:** Program tidak menangani kasus jika pengguna memasukkan bilangan negatif atau nol. Ini bisa menyebabkan output yang tidak diharapkan atau tidak bermakna.

Saran dan Pembimbingan:

1. **Perbaiki Format Output:** Coba perhatikan kembali contoh output yang diminta dalam soal. Apakah bintang-bintang tersebut dipisahkan oleh spasi? Jika iya, bagaimana cara memodifikasi perulangan untuk menambahkan spasi di antara bintang?
2. **Tambahkan Validasi Input:** Pertimbangkan untuk menambahkan pengecekan jika `n` kurang dari atau sama dengan 0. Bagaimana seharusnya program menanggapi input yang tidak valid? Apakah perlu memberi pesan error atau meminta input ulang?
3. **Komentar Kode:** Meskipun tidak wajib, tambahkan komentar untuk menjelaskan logika perulangan agar kode lebih mudah dipahami oleh orang lain.

Pernyataan Evaluasi Singkat:
Status: incorrect

Conclusion:
Incorrect ×

Score: 65/100

Gambar A. 10 Hasil *Feedback* AI Responden 2 Nomor 5

AI Feedbacks:

Analisis Fungsional:
Kode ini berupaya menyelesaikan soal dengan mencetak bilangan ganjil antara dua batas (bawah dan atas). Secara umum, pendekatannya masuk akal dengan menggunakan loop untuk iterasi dan kondisi untuk mengecek bilangan ganjil. Namun, ada beberapa masalah dalam implementasinya.

Identifikasi Potensi Masalah:

1. **Urutan Input dan Output:**
   - Program meminta input `bawah` sebelum menampilkan pesan "Masukkan batas bawah:", sehingga pengguna mungkin bingung. Pesan harus ditampilkan sebelum menerima input.
   - Ada tanda kurung yang tidak ditutup di `System.out.println("Masukkan batas atas ");`.
2. **Logika Perulangan:**
   - Jika `bawah` ganjil, kode mencetak `bawah` dua kali (sekali di luar loop dan sekali di dalam loop).
   - Loop `for(int i= bawah;i<atas;i+=2)` tidak mencakup `atas` jika `atas` ganjil, kecuali ada pengecekan terpisah. Namun, pengecekan ini hanya dilakukan jika `bawah` ganjil, sehingga tidak konsisten.
   - Jika `bawah` genap, loop dimulai dari `bawah` (genap) dan mencetak bilangan genap, bukan ganjil.
3. **Penanganan Kasus Khusus:**
   - Tidak ada penanganan jika `bawah > atas`. Program akan masuk ke loop tanpa hasil yang benar.

Saran dan Pembimbingan:

1. **Perbaiki Urutan Input:**
   - Pastikan pesan input ditampilkan sebelum menerima input. Contoh:
     ```
     System.out.println("Masukkan batas bawah:");
     bawah = x.nextInt();
     ```
2. **Perbaiki Logika Ganjil/Genap:**
   - Jika `bawah` genap, Anda bisa menambahkan 1 untuk memulai dari bilangan ganjil terdekat.
   - Loop seharusnya mencakup `atas` jika ganjil, sehingga kondisi loop bisa `i <= atas`.
3. **Handle Kasus `bawah > atas`:**
   - Tambahkan pengecekan jika `bawah > atas`, lalu beri pesan error atau tukar nilai.
4. **Hindari Duplikasi Output:**
   - Jangan cetak `bawah` secara terpisah jika sudah dicetak dalam loop.
5. **Perbaiki Struktur Kode:**
   - Gunakan indentasi yang konsisten untuk memudahkan pembacaan.
   - Tutup semua tanda kurung dan perhatikan sintaks yang benar.

Pernyataan Evaluasi Singkat:
Status: incorrect

Conclusion:
Incorrect ×

Score: 45/100

Gambar A. 11 Hasil *Feedback* AI Responden 3 Nomor 1

AI Feedbacks:

Analisis Fungsional:
Kode yang dikirimkan oleh mahasiswa mencoba untuk menghitung faktorial dari sebuah bilangan menggunakan rekursi. Secara umum, pendekatan ini masuk akal karena rekursi adalah salah satu cara yang umum digunakan untuk menghitung faktorial. Namun, kode ini tidak sepenuhnya memenuhi soal karena tidak meminta input dari pengguna seperti yang diminta dalam instruksi.

Identifikasi Potensi Masalah:

1. **Ketepatan Input:** Program tidak meminta input dari pengguna (`n`), melainkan langsung menghitung faktorial dari angka 6 yang sudah di-hardcode. Ini tidak sesuai dengan instruksi soal.
2. **Penanganan Kasus Khusus:** Kode tidak menangani kasus jika input yang dimasukkan pengguna adalah bilangan negatif. Faktorial tidak terdefinisi untuk bilangan negatif, sehingga perlu ada pengecekan tambahan.
3. **Kerapian dan Struktur Kode:**
   - Penulisan sintaks Java tidak konsisten (misalnya: `Public` seharusnya `public`, `Void` seharusnya `void`).
   - Kurangnya komentar atau penjelasan tentang alur program.
4. **Kesesuaian Output:** Output yang dihasilkan tidak sesuai dengan format yang diminta dalam soal (contoh: "Masukkan bilangan: 5 Faktorial dari 5 adalah 120").

Saran dan Pembimbingan:

1. **Input Pengguna:** Coba tambahkan kode untuk meminta input dari pengguna menggunakan `Scanner` atau cara lain yang sesuai. Pastikan program membaca nilai `n` dari input pengguna, bukan angka yang sudah ditentukan.
2. **Penanganan Kasus Negatif:** Pertimbangkan untuk menambahkan pengecekan apakah input yang dimasukkan adalah bilangan non-negatif. Jika tidak, berikan pesan error yang jelas.
3. **Perbaikan Sintaks:** Perhatikan penulisan huruf besar/kecil (case sensitivity) dalam Java, seperti `public`, `static`, dan `void`.
4. **Format Output:** Sesuaikan output dengan format yang diminta soal, termasuk teks permintaan input dan hasil perhitungan.

Pernyataan Evaluasi Singkat:
Status: incorrect

Conclusion:
Incorrect ×

Score: 50/100

Gambar A. 12 Hasil *Feedback* AI Responden 3 Nomor 2

AI Feedbacks:

Analisis Fungsional:
Kode yang diberikan oleh mahasiswa secara umum mencoba untuk memenuhi tujuan soal, yaitu menghitung jumlah bilangan genap dari `a` sampai `b`. Pendekatan menggunakan loop dan kondisi modulus (`%2==0`) sudah tepat untuk mengecek bilangan genap. Namun, ada beberapa masalah yang perlu diperhatikan.

Identifikasi Potensi Masalah:

1. **Kesalahan Penulisan Package:** Terdapat typo pada `java.utii.Scanner` seharusnya `java.util.Scanner`.
2. **Variabel yang Tidak Konsisten:** Variabel `a` dan `b` digunakan dalam loop (`for(int i=a;i<=b;i++)`), tetapi input disimpan di `bawah` dan `atas`. Ini akan menyebabkan error karena variabel `a` dan `b` tidak dideklarasikan.
3. **Format Input dan Output:** Soal meminta input dengan pesan seperti "Masukkan nilai a:", tetapi kode hanya menggunakan `nextInt()` tanpa petunjuk untuk pengguna. Output juga hanya menampilkan `sum` tanpa penjelasan sesuai contoh.
4. **Penanganan Kasus Khusus:** Kode tidak mempertimbangkan kasus di mana `a > b`. Jika `a` lebih besar dari `b`, loop tidak akan berjalan, tetapi tidak ada pesan atau penanganan khusus untuk kasus ini.

Saran dan Pembimbingan:

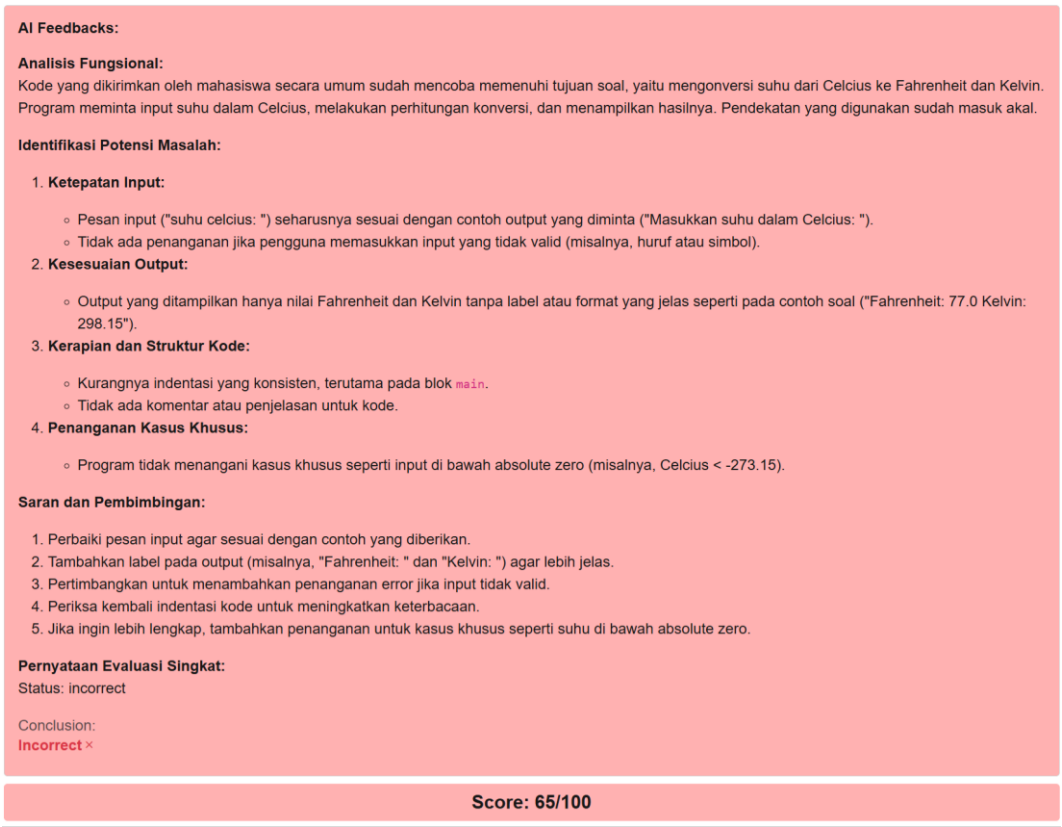
1. Perbaiki typo pada package dan pastikan semua variabel yang digunakan sudah dideklarasikan dengan benar.
2. Tambahkan pesan untuk memandu pengguna saat input, misalnya dengan `System.out.print("Masukkan nilai a: ")` sebelum `nextInt()`.
3. Output sebaiknya sesuai dengan contoh, yaitu mencantumkan penjelasan seperti "Jumlah bilangan genap dari X sampai Y adalah Z".
4. Pertimbangkan untuk menambahkan pengecekan jika `a > b`, misalnya dengan menukar nilai `a` dan `b` atau memberi pesan error.
5. Periksa kembali struktur kode, pastikan class dan method `main` didefinisikan dengan benar (seharusnya ada `public class NamaClass` sebelum `main`).

Pernyataan Evaluasi Singkat:
Status: incorrect

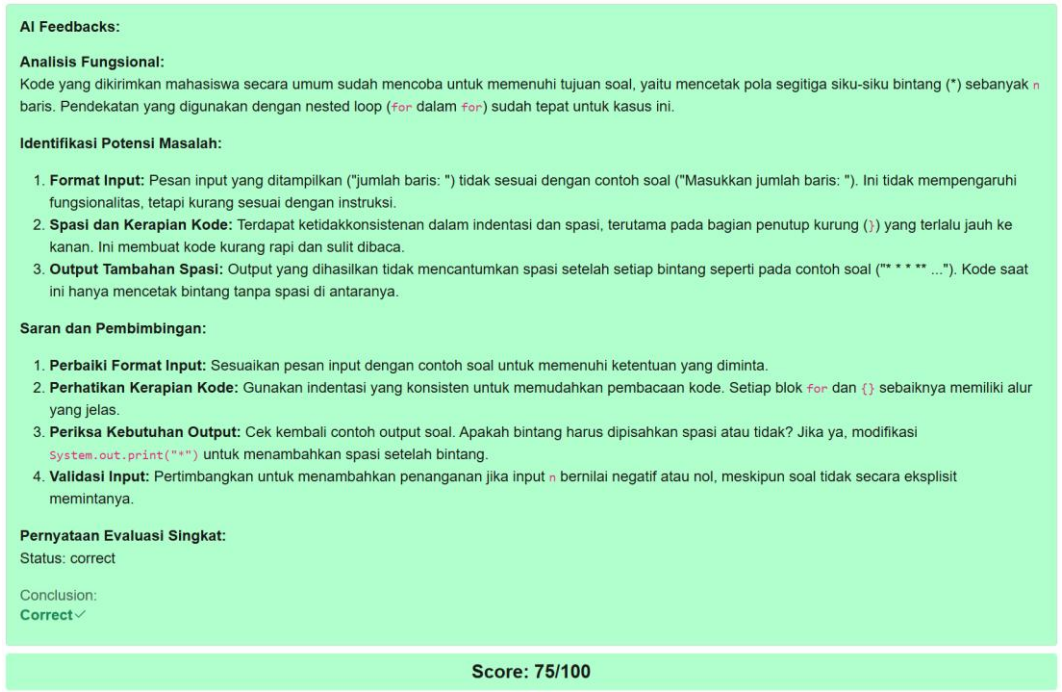
Conclusion:
Incorrect ×

Score: 50/100

Gambar A. 13 Hasil *Feedback* AI Responden 3 Nomor 3



Gambar A. 14 Hasil *Feedback* AI Responden 3 Nomor 4



Gambar A. 15 Hasil *Feedback* AI Responden 3 Nomor 5

AI Feedbacks:

Analisis Fungsional:
Kode yang dikirimkan mahasiswa mencoba untuk menyelesaikan soal dengan meminta input dua bilangan bulat (x dan y) dan menampilkan bilangan ganjil di antara keduanya. Secara umum, pendekatan yang digunakan sudah sesuai dengan tujuan soal, yaitu menggunakan loop untuk iterasi dan kondisi untuk mengecek bilangan ganjil. Namun, terdapat beberapa kesalahan yang perlu diperbaiki.

Identifikasi Potensi Masalah:

1. **Kesalahan Sintaks:** Ada kesalahan penulisan `input,nextInt()` seharusnya `input.nextInt()`.
2. **Logika Kondisi Salah:** Kondisi `if (i % 2 = 0)` seharusnya `if (i % 2 != 0)` karena yang ingin ditampilkan adalah bilangan ganjil, bukan genap.
3. **Duplikasi Kode:** Terdapat duplikasi kode yang sama persis (kelas `BilanganGanjil` ditulis dua kali).
4. **Format Output:** Output seharusnya menampilkan bilangan ganjil dalam satu baris dipisahkan spasi, bukan per baris.

Saran dan Pembimbingan:

1. Perbaiki kesalahan sintaks pada pembacaan input.
2. Periksa kembali logika kondisi untuk mengecek bilangan ganjil. Ingat bahwa bilangan ganjil adalah bilangan yang tidak habis dibagi 2.
3. Hapus duplikasi kode yang tidak diperlukan.
4. Pertimbangkan untuk menggunakan `System.out.print(i + " ")` agar output sesuai dengan contoh soal (bilangan dipisahkan spasi).
5. Coba jalankan program dengan berbagai input (x < y, x > y, atau x = y) untuk memastikan program bekerja dalam semua kasus.

Pernyataan Evaluasi Singkat:
Status: incorrect

Conclusion:
Incorrect ×

Score: 50/100

Gambar A. 16 Hasil *Feedback* AI Responden 4 Nomor 1

AI Feedbacks:

Analisis Fungsional:
Kode yang dikirimkan oleh mahasiswa secara umum sudah mencoba untuk memenuhi tujuan soal, yaitu menghitung faktorial dari sebuah bilangan bulat `n`. Program meminta input dari pengguna, menggunakan loop `for` untuk menghitung faktorial, dan menampilkan hasilnya. Pendekatan yang digunakan sudah masuk akal untuk menyelesaikan masalah ini.

Identifikasi Potensi Masalah:

1. **Kesalahan Penulisan Variabel:** Terdapat kesalahan penulisan nama variabel `intput` (seharusnya `input`) dan `faktorial` (seharusnya `Faktorial`). Ini akan menyebabkan error saat kompilasi karena Java bersifat case-sensitive.
2. **Penanganan Input Negatif:** Program tidak menangani kasus jika pengguna memasukkan bilangan negatif. Faktorial tidak terdefinisi untuk bilangan negatif, sehingga sebaiknya ada pengecekan untuk hal ini.
3. **Format Output:** Format output tidak persis sesuai dengan contoh yang diberikan (contoh: "Masukkan bilangan:", sedangkan kode menulis "masukan angka:").

Saran dan Pembimbingan:

1. Periksa kembali penulisan nama variabel untuk memastikan konsistensi dan kebenaran penulisan. Java membedakan huruf besar dan kecil, jadi pastikan semua referensi variabel sesuai dengan deklarasinya.
2. Pertimbangkan untuk menambahkan pengecekan input, misalnya memastikan `n` tidak negatif. Anda bisa menggunakan kondisi `if` untuk menangani kasus ini.
3. Perhatikan format output sesuai dengan contoh yang diberikan dalam soal untuk memastikan kesesuaian dengan permintaan.

Pernyataan Evaluasi Singkat:
Status: incorrect

Conclusion:
Incorrect ×

Score: 65/100

Gambar A. 17 Hasil *Feedback* AI Responden 4 Nomor 2

AI Feedbacks:

Analisis Fungsional:
Kode yang dikirimkan oleh mahasiswa sudah mencoba untuk memenuhi tujuan soal dengan baik. Program meminta input dua bilangan bulat `a` dan `b`, kemudian menghitung jumlah semua bilangan genap dalam rentang tersebut (termasuk batasnya). Pendekatan yang digunakan, yaitu menggunakan loop `for` dan pengecekan kondisi `if (i % 2 == 0)`, secara umum masuk akal dan sesuai dengan logika yang dibutuhkan.

Identifikasi Potensi Masalah:

1. **Penanganan Kasus Khusus:** Kode tidak mempertimbangkan kasus di mana nilai `a` lebih besar dari `b`. Jika `a > b`, loop tidak akan berjalan sama sekali, dan outputnya akan menunjukkan jumlah 0, yang mungkin tidak diharapkan. Soal tidak secara eksplisit menyebutkan batasan ini, tetapi penanganan kasus seperti ini bisa menjadi nilai tambah.
2. **Input Negatif:** Program tidak memeriksa apakah input adalah bilangan bulat. Meskipun soal hanya meminta bilangan bulat, penanganan input yang tidak valid (misalnya, karakter atau string) bisa menyebabkan error.
3. **Efisiensi:** Untuk rentang yang sangat besar, loop dari `a` ke `b` mungkin tidak efisien. Namun, untuk kasus sederhana seperti ini, hal ini tidak menjadi masalah.

Saran dan Pembimbingan:

1. **Kasus `a > b`:** Coba pikirkan apa yang seharusnya terjadi jika `a` lebih besar dari `b`. Apakah program harus menukar nilai `a` dan `b`, atau memberikan pesan error? Diskusikan dengan diri sendiri apakah penanganan kasus ini diperlukan.
2. **Validasi Input:** Pertimbangkan untuk menambahkan validasi input untuk memastikan bahwa input yang dimasukkan adalah bilangan bulat. Bagaimana cara menangani input yang tidak valid?
3. **Komentar Kode:** Tambahkan komentar singkat untuk menjelaskan logika utama, seperti tujuan loop dan kondisi `if`. Ini akan membuat kode lebih mudah dipahami oleh orang lain.

Pernyataan Evaluasi Singkat:
Status: correct

Conclusion:
Correct✓

Score: 90/100

Gambar A. 18 Hasil *Feedback* AI Responden 4 Nomor 3

AI Feedbacks:

Analisis Fungsional:
Kode yang dikirimkan oleh mahasiswa sudah mencoba menjawab seluruh tujuan dari soal dengan baik. Program meminta input suhu dalam Celcius, mengkonversinya ke Fahrenheit dan Kelvin, serta menampilkan hasilnya sesuai format yang diminta. Pendekatan yang digunakan secara umum masuk akal dan sesuai dengan rumus konversi suhu yang benar.

Identifikasi Potensi Masalah:

1. **Penanganan Input Non-Numerik:** Program tidak menangani kasus jika pengguna memasukkan input yang bukan angka (misalnya huruf atau simbol). Ini bisa menyebabkan program error.
2. **Presisi Output:** Output untuk Fahrenheit dan Kelvin sudah benar, tetapi tidak ada pembatasan jumlah digit desimal. Meskipun tidak diminta dalam soal, ini bisa menjadi pertimbangan untuk konsistensi tampilan.
3. **Validasi Input Negatif:** Program bisa menerima suhu negatif (valid untuk Celcius), tetapi tidak ada penanganan khusus jika ini menjadi masalah (misalnya untuk kasus Kelvin yang seharusnya tidak negatif).

Saran dan Pembimbingan:

1. **Penanganan Error Input:** Coba pikirkan bagaimana cara menangani input yang tidak valid (misalnya dengan menggunakan `try-catch` atau loop untuk memastikan input adalah angka).
2. **Format Output:** Pertimbangkan untuk menambahkan pembatasan digit desimal (misalnya menggunakan `String.format`) agar tampilan lebih konsisten.
3. **Validasi Suhu:** Untuk Kelvin, suhu tidak boleh negatif. Bagaimana jika pengguna memasukkan suhu di bawah -273.15°C? Apa yang seharusnya program lakukan?

Pernyataan Evaluasi Singkat:
Status: correct

Conclusion:
Correct✓

Score: 90/100

Gambar A. 19 Hasil *Feedback* AI Responden 4 Nomor 4

AI Feedbacks:

Analisis Fungsional:
Kode yang dikirimkan oleh mahasiswa secara umum sudah mencoba untuk memenuhi tujuan soal, yaitu mencetak pola segitiga siku-siku bintang sebanyak `n` baris. Program menerima input `n` dengan benar dan menggunakan perulangan bersarang untuk mencetak bintang sesuai dengan jumlah baris. Pendekatan yang digunakan sudah masuk akal dan sesuai dengan logika dasar untuk membuat pola segitiga bintang.

Identifikasi Potensi Masalah:

1. **Format Output:** Output yang dihasilkan tidak sepenuhnya sesuai dengan contoh yang diberikan dalam soal. Contoh soal menunjukkan output dengan spasi antara bintang (misalnya: `* * * ** * *** * **** *`), sedangkan kode mahasiswa mencetak bintang tanpa spasi (misalnya: `* ** *** **** *****`). Ini mungkin disebabkan oleh perbedaan interpretasi terhadap contoh soal.
2. **Penanganan Input Negatif:** Program tidak menangani kasus jika pengguna memasukkan nilai `n` yang negatif atau nol. Ini bisa menyebabkan program tidak menghasilkan output apa-apa atau berperilaku tidak sesuai harapan.

Saran dan Pembimbingan:

1. **Periksa Kembali Contoh Output:** Coba bandingkan output programmu dengan contoh yang diberikan dalam soal. Apakah ada perbedaan dalam format atau spasi? Jika iya, pikirkan bagaimana cara menyesuaikannya.
2. **Tambahkan Validasi Input:** Pertimbangkan untuk menambahkan pengecekan jika `n` bernilai negatif atau nol. Kamu bisa menggunakan pernyataan `if` untuk memastikan input valid sebelum melanjutkan ke perulangan.
3. **Optimasi Kode:** Kode sudah cukup rapi, tetapi kamu bisa menambahkan komentar untuk menjelaskan langkah-langkah penting, terutama jika kode ini akan dibaca oleh orang lain.

Pernyataan Evaluasi Singkat:
Status: correct

Conclusion:
Correct ✓

Score: 85/100

Gambar A. 20 Hasil *Feedback* AI Responden 4 Nomor 5

AI Feedbacks:

Analisis Fungsional:
Kode yang diberikan oleh mahasiswa secara umum mencoba untuk memenuhi tujuan soal, yaitu menampilkan bilangan ganjil dari `x` sampai `y`. Mahasiswa menggunakan loop `for` dan kondisi `if` untuk memeriksa bilangan ganjil, yang merupakan pendekatan yang masuk akal. Namun, ada beberapa masalah yang perlu diperhatikan.

Identifikasi Potensi Masalah:

1. **Kesalahan Variabel:** Variabel yang digunakan dalam loop `for` adalah `x` dan `y`, tetapi variabel yang diinput adalah `awal` dan `akhir`. Ini akan menyebabkan error karena `x` dan `y` tidak dideklarasikan.
2. **Format Output:** Output menggunakan `println` yang akan menampilkan setiap bilangan ganjil dalam baris baru, sedangkan contoh output soal menampilkan bilangan dalam satu baris dipisahkan spasi.
3. **Penanganan Kasus Khusus:** Kode tidak mempertimbangkan kasus di mana `awal` lebih besar dari `akhir`. Program akan gagal menampilkan output yang benar dalam kasus ini.
4. **Kesalahan Nama Method:** Nama method `Main` seharusnya `main` (huruf kecil) karena Java bersifat case-sensitive.

Saran dan Pembimbingan:

1. Periksa kembali nama variabel yang digunakan dalam loop `for`. Pastikan konsistensi antara variabel yang diinput dan yang digunakan dalam perhitungan.
2. Pertimbangkan untuk menggunakan `print` daripada `println` agar output sesuai dengan format yang diminta (satu baris dipisahkan spasi).
3. Tambahkan penanganan kasus di mana `awal` lebih besar dari `akhir`. Anda bisa mempertimbangkan untuk menukar nilai `awal` dan `akhir` atau memberikan pesan error.
4. Perhatikan penulisan nama method `main` yang benar sesuai aturan Java.
5. Tambahkan komentar jika diperlukan untuk menjelaskan logika kode, meskipun tidak wajib.

Pernyataan Evaluasi Singkat:
Status: incorrect

Conclusion:
Incorrect ×

Score: 50/100

Gambar A. 21 Hasil *Feedback* AI Responden 5 Nomor 1

AI Feedbacks:

Analisis Fungsional:
Kode yang dikirimkan oleh mahasiswa sudah mencoba untuk memenuhi tujuan soal, yaitu menghitung faktorial dari sebuah bilangan bulat yang diinputkan oleh pengguna. Pendekatan yang digunakan secara umum masuk akal dengan menggunakan loop `for` untuk menghitung faktorial dan penanganan kasus khusus untuk input negatif.

Identifikasi Potensi Masalah:

1. **Penanganan Input Negatif:**
   - Kode sudah menangani kasus input negatif dengan menampilkan pesan error, tetapi pesannya kurang spesifik. Sebaiknya dijelaskan bahwa faktorial tidak terdefinisi untuk bilangan negatif.
   - Program tidak meminta input ulang jika input negatif, sehingga pengguna harus menjalankan program lagi.
2. **Kasus Faktorial 0:**
   - Faktorial dari 0 seharusnya menghasilkan 1, dan kode ini sudah benar menanganinya karena loop `for` tidak akan dijalankan jika `faktor = 0` (karena `i <= 0` tidak terpenuhi), sehingga `jumlah` tetap 1.
3. **Overflow untuk Input Besar:**
   - Kode tidak menangani kemungkinan overflow jika input terlalu besar (misalnya, `faktor = 20` atau lebih). Nilai faktorial bisa melebihi kapasitas `int` dan menyebabkan hasil yang salah.
4. **Komentar dan Struktur Kode:**
   - Kode cukup rapi dan terstruktur, tetapi tidak ada komentar yang menjelaskan logika atau bagian penting dari program.

Saran dan Pembimbingan:

1. **Perbaiki Pesan Error:**
   - Pertimbangkan untuk membuat pesan error lebih informatif, misalnya: "Faktorial tidak terdefinisi untuk bilangan negatif. Silakan masukkan bilangan bulat non-negatif."
2. **Loop atau Validasi Input:**
   - Coba pikirkan cara untuk meminta input ulang jika pengguna memasukkan bilangan negatif, misalnya dengan menggunakan loop `while` sampai input valid.
3. **Penanganan Overflow:**
   - Coba cari tahu apa yang terjadi jika input terlalu besar. Apakah tipe data `int` cukup? Mungkin perlu menggunakan `long` atau `BigInteger` untuk menangani angka yang sangat besar.
4. **Tambahkan Komentar:**
   - Tambahkan komentar singkat untuk menjelaskan bagian penting, seperti inisialisasi variabel `jumlah` atau logika perhitungan faktorial.

Pernyataan Evaluasi Singkat:
Status: correct

Conclusion:
Correct✓

Score: 85/100

Gambar A. 22 Hasil *Feedback* AI Responden 5 Nomor 2

AI Feedbacks:

Analisis Fungsional:
Kode yang dikirimkan oleh mahasiswa sudah mencoba untuk memenuhi tujuan soal dengan baik. Program meminta input dua bilangan bulat (`a` dan `b`), kemudian menghitung jumlah semua bilangan genap di antara keduanya (termasuk batasnya). Pendekatan yang digunakan, yaitu loop dengan pengecekan modulus, secara umum masuk akal dan sesuai dengan kebutuhan soal.

Identifikasi Potensi Masalah:

1. **Penanganan Input Terbalik:** Kode tidak mempertimbangkan kasus di mana nilai `a` lebih besar dari `b`. Dalam kondisi ini, loop tidak akan berjalan sama sekali karena kondisi `i <= akhir` tidak terpenuhi, sehingga hasilnya akan selalu 0. Ini mungkin bukan perilaku yang diharapkan.
2. **Validasi Input:** Tidak ada pengecekan apakah input yang dimasukkan benar-benar bilangan bulat. Jika pengguna memasukkan input non-integer, program akan error.
3. **Efisiensi:** Meskipun tidak kritikal untuk kasus sederhana, loop dari `a` ke `b` bisa dioptimasi dengan langsung iterasi bilangan genap saja (misalnya, mulai dari bilangan genap terdekat setelah `a` dan increment +2).

Saran dan Pembimbingan:

1. **Pertimbangkan Kasus Input Terbalik:** Coba pikirkan apa yang terjadi jika `a` lebih besar dari `b`. Apakah output 0 adalah hasil yang diinginkan? Jika tidak, bagaimana cara menanganinya?
2. **Validasi Input:** Tambahkan pengecekan untuk memastikan input adalah bilangan bulat. Anda bisa menggunakan `hasNextInt()` dari `Scanner` sebelum membaca input.
3. **Optimasi Loop:** Untuk latihan tambahan, coba modifikasi loop sehingga hanya mengiterasi bilangan genap saja. Ini akan mengurangi jumlah iterasi dan membuat kode lebih efisien.
4. **Komentar Kode:** Meskipun kode sudah cukup jelas, tambahkan komentar sederhana untuk menjelaskan logika utama (misalnya, "Loop untuk menghitung jumlah bilangan genap dari a ke b").

Pernyataan Evaluasi Singkat:
Status: correct

Conclusion:
Correct ✓

Score: 85/100

Gambar A. 23 Hasil *Feedback* AI Responden 5 Nomor 3

AI Feedbacks:

Analisis Fungsional:
Kode yang diberikan oleh mahasiswa secara umum sudah mencoba menjawab tujuan soal, yaitu mengkonversi suhu dari Celcius ke Fahrenheit dan Kelvin. Pendekatan yang digunakan masuk akal dengan menggunakan rumus konversi yang benar untuk kedua satuan suhu.

Identifikasi Potensi Masalah:

1. **Ketepatan Input:** Program menggunakan `nextInt()` untuk membaca input, yang berarti hanya menerima bilangan bulat. Jika pengguna memasukkan bilangan desimal (misalnya 25.5), program akan error. Seharusnya menggunakan `nextDouble()` untuk menangani input desimal.
2. **Format Output:** Output yang diminta dalam soal mencakup prompt input ("Masukkan suhu dalam Celcius:") dan hasil konversi dalam satu baris (contoh: "Fahrenheit: 77.0 Kelvin: 298.15"). Kode saat ini tidak menampilkan prompt input dan menampilkan hasil dalam baris terpisah.
3. **Penanganan Kasus Khusus:** Program tidak menangani kemungkinan input yang tidak valid (misalnya, input bukan angka).

Saran dan Pembimbingan:

1. Pertimbangkan untuk menggunakan `nextDouble()` agar program dapat menerima input bilangan desimal.
2. Tambahkan prompt input (`System.out.print("Masukkan suhu dalam Celcius: ");`) sebelum membaca input untuk memandu pengguna.
3. Sesuaikan format output agar sesuai dengan contoh yang diberikan, yaitu menampilkan hasil dalam satu baris.
4. (Opsional) Tambahkan penanganan error jika input tidak valid, misalnya dengan menggunakan `try-catch` atau pemeriksaan tipe data.

Pernyataan Evaluasi Singkat:
Status: incorrect

Conclusion:
Incorrect ×

Score: 65/100

Gambar A. 24 Hasil *Feedback* AI Responden 5 Nomor 4

AI Feedbacks:

Analisis Fungsional:
Kode yang diberikan oleh mahasiswa mencoba untuk menyelesaikan soal dengan membuat pola segitiga siku-siku bintang berdasarkan input `n`. Secara umum, pendekatan yang digunakan sudah masuk akal dengan menggunakan nested loop untuk mencetak bintang. Namun, ada beberapa bagian yang perlu diperbaiki agar kode dapat berfungsi dengan benar.

Identifikasi Potensi Masalah:

1. **Kesalahan Sintaks:**
   - `in.nextInt` seharusnya `in.nextInt()` karena `nextInt` adalah method yang memerlukan tanda kurung.
   - Operator `=<` seharusnya `<=` untuk perbandingan "kurang dari atau sama dengan".
   - Penggunaan tanda kutip tunggal (`'*'`) seharusnya tanda kutip ganda (`"*"`) karena `System.out.print` menerima parameter bertipe `String`.
2. **Logika Perulangan:**
   - Loop dalam (`for (int j=1; j<i; j++)`) akan mencetak bintang sebanyak `i-1`, dan diikuti dengan `System.out.println('*')`, sehingga total bintang per baris akan menjadi `i`. Ini sebenarnya benar, tetapi cara mencetaknya bisa disederhanakan.
3. **Input dan Output:**
   - Tidak ada pesan yang memandu pengguna untuk memasukkan input (contoh: "Masukkan jumlah baris: ").
   - Output tidak sesuai dengan contoh yang diberikan karena tidak ada spasi antara bintang.

Saran dan Pembimbingan:

1. Perbaiki sintaks yang salah, seperti penggunaan `nextInt()`, operator `<=`, dan tanda kutip untuk string.
2. Pertimbangkan untuk menambahkan pesan panduan saat meminta input agar lebih user-friendly.
3. Cek kembali logika perulangan dan pastikan output sesuai dengan contoh yang diminta (termasuk spasi antara bintang jika diperlukan).
4. Gunakan `System.out.println()` untuk pindah baris setelah mencetak bintang pada satu baris.

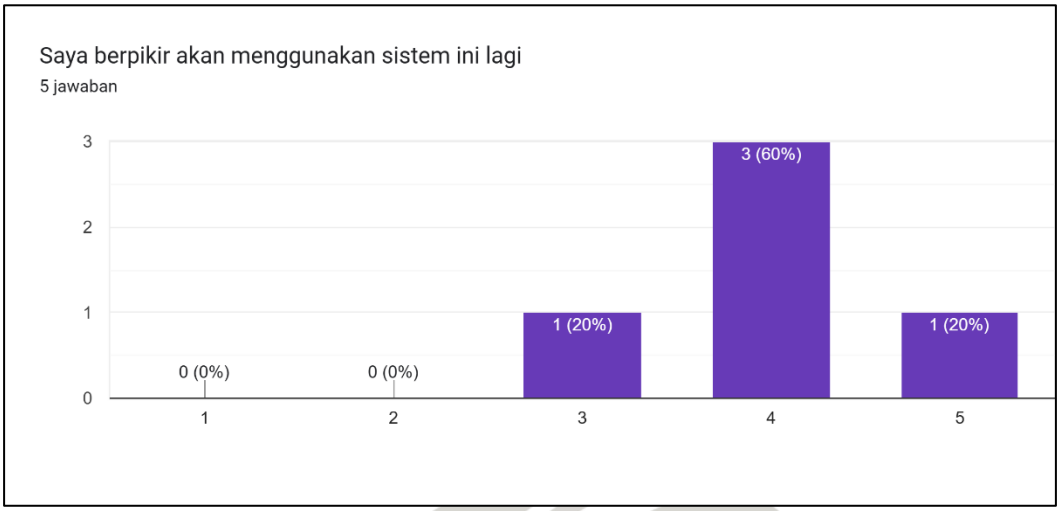
Pernyataan Evaluasi Singkat:
Status: incorrect

Conclusion:
Incorrect ×

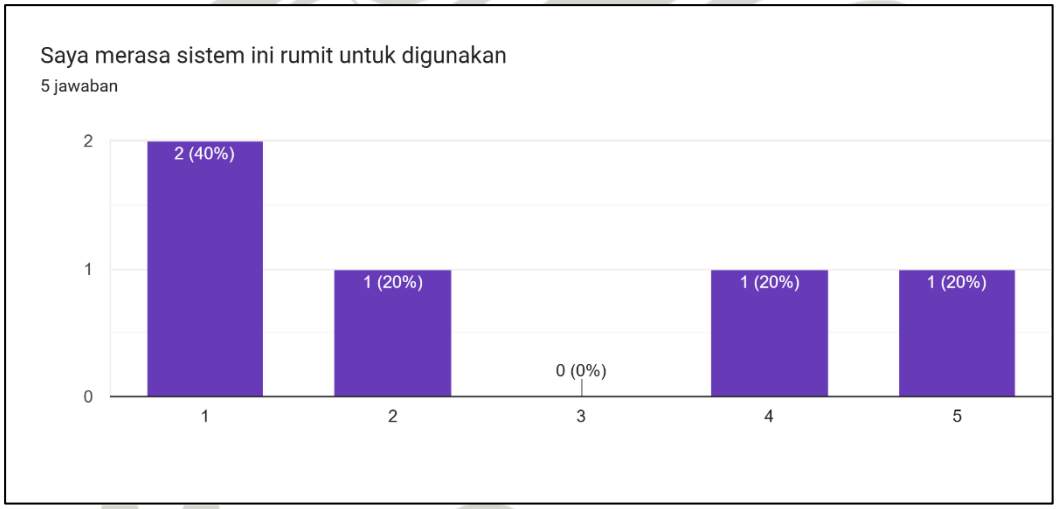
Score: 50/100

Gambar A. 25 Hasil *Feedback* AI Responden 5 Nomor 5

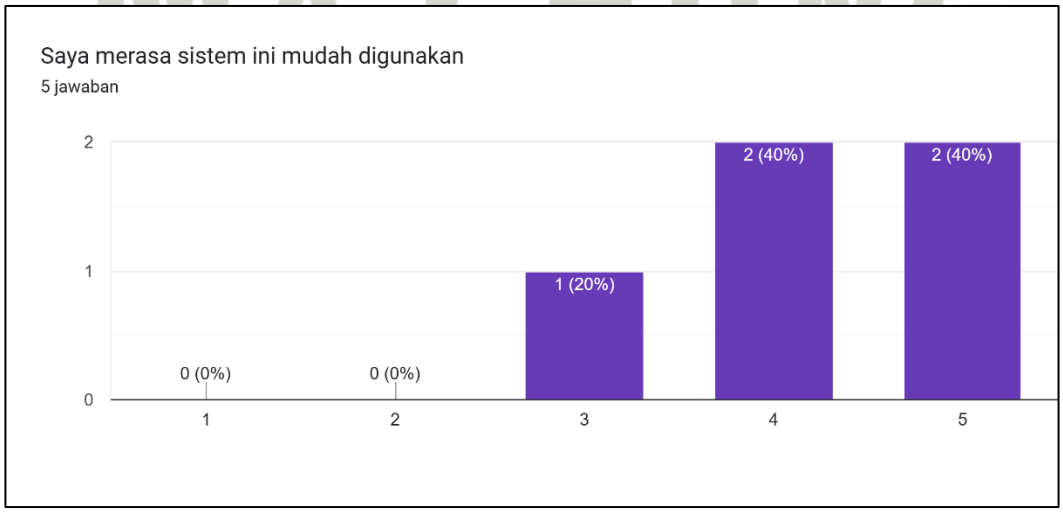
LAMPIRAN B: HASIL SURVEI SUS



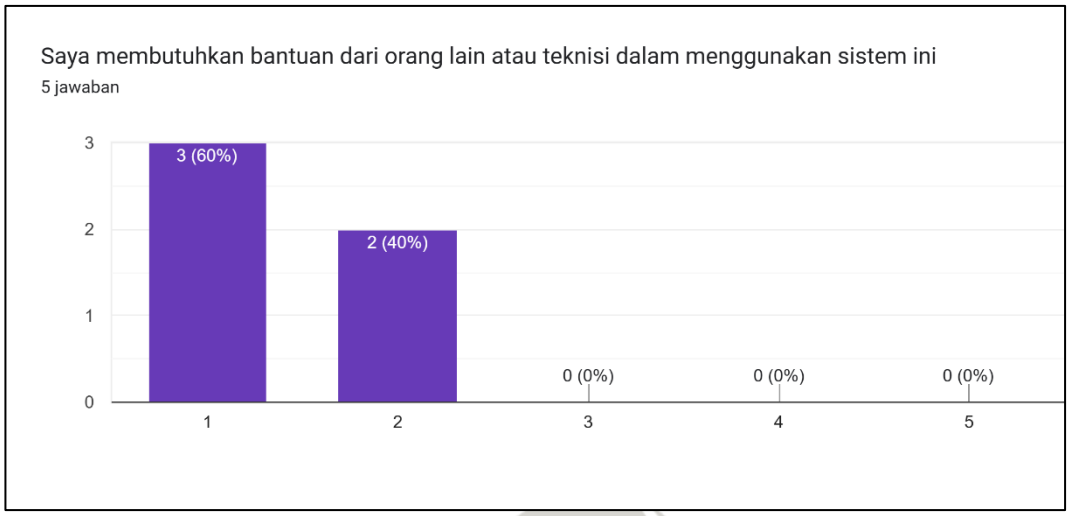
Gambar B. 1 Hasil Survei SUS Nomor 1



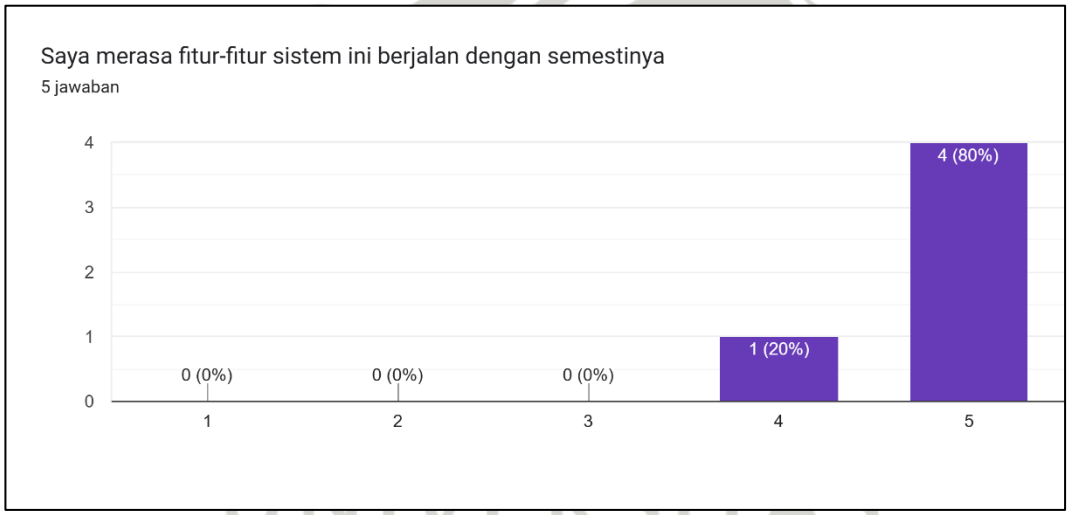
Gambar B. 2 Hasil Survei SUS Nomor 2



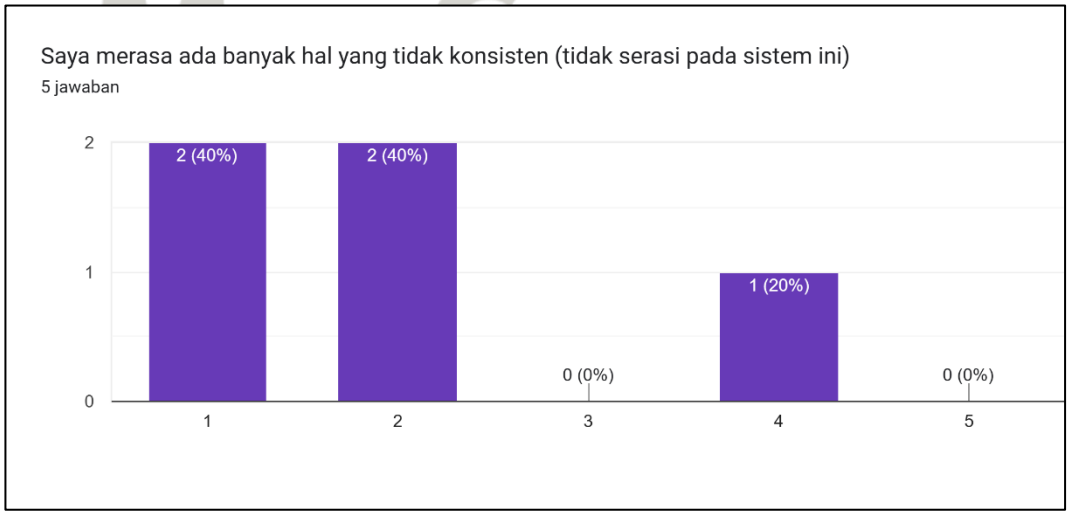
Gambar B. 3 Hasil Survei SUS Nomor 3



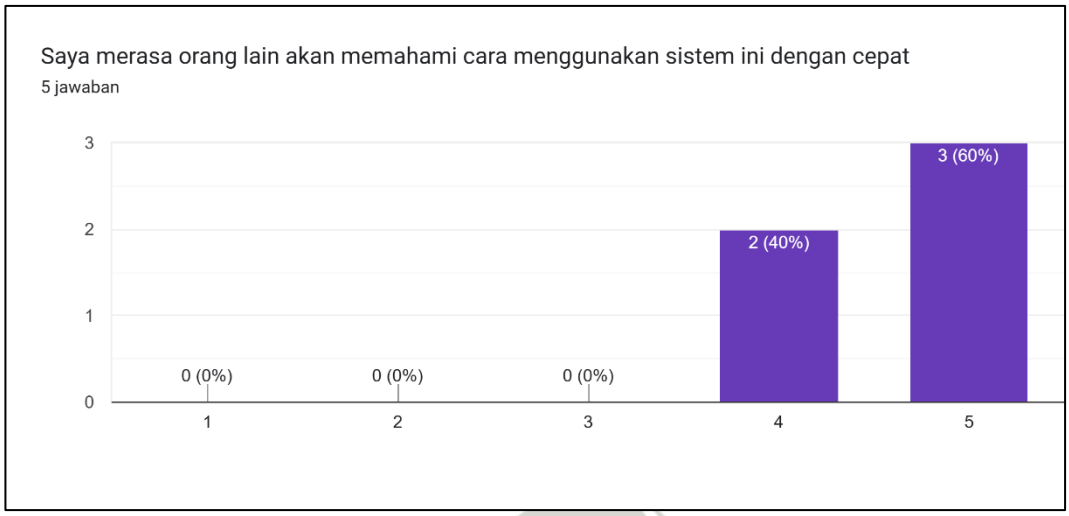
Gambar B. 4 Hasil Survei SUS Nomor 4



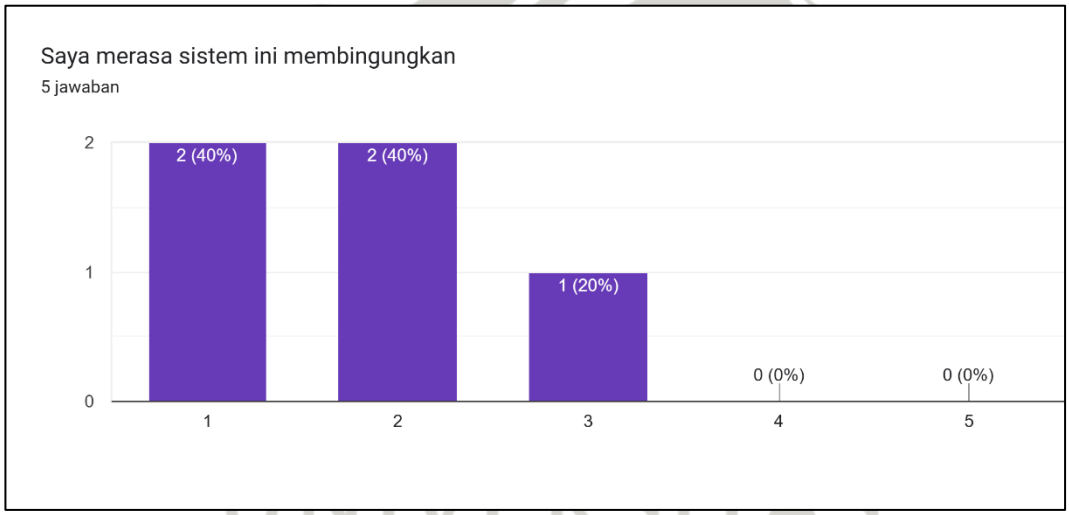
Gambar B. 5 Hasil Survei SUS Nomor 5



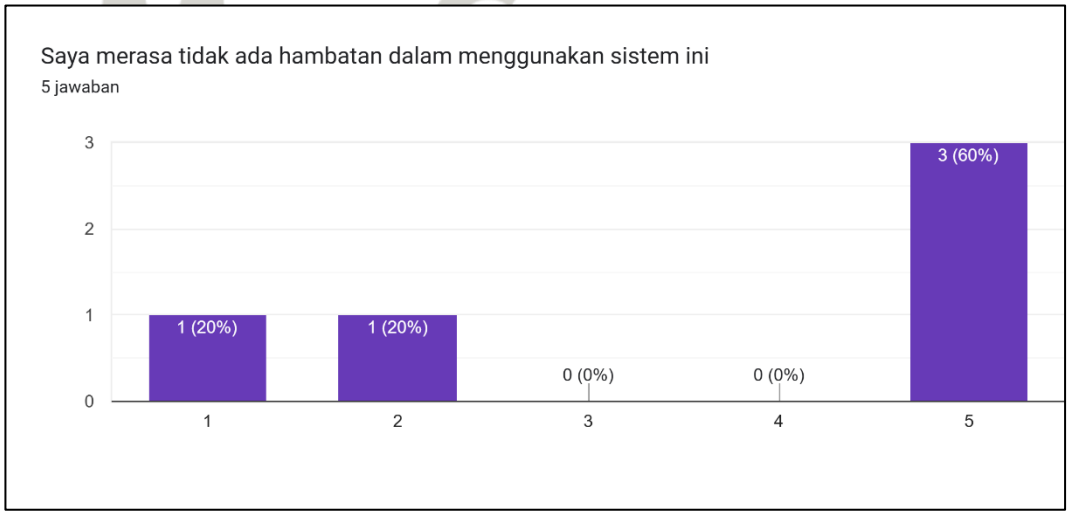
Gambar B. 6 Hasil Survei SUS Nomor 6



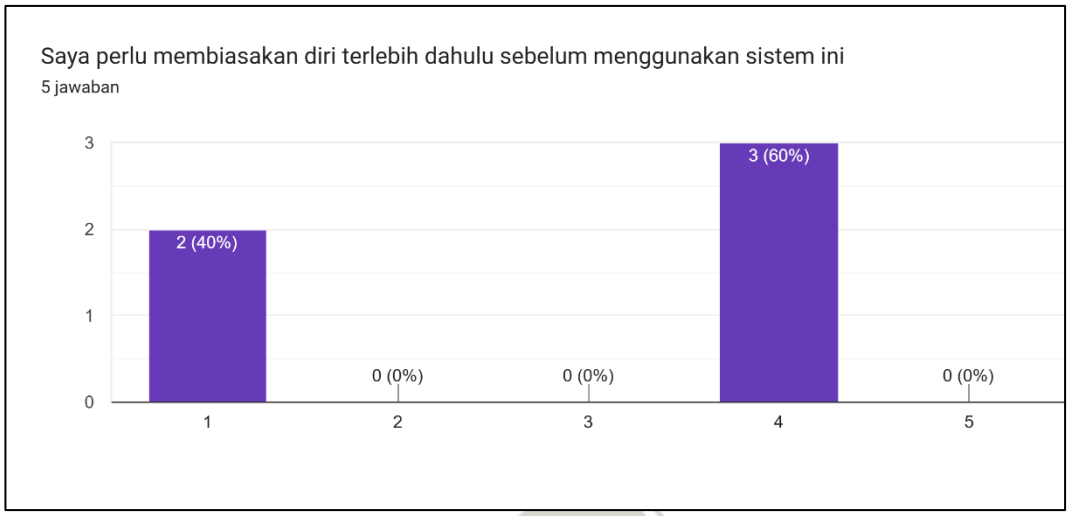
Gambar B. 7 Hasil Survei SUS Nomor 7



Gambar B. 8 Hasil Survei SUS Nomor 8

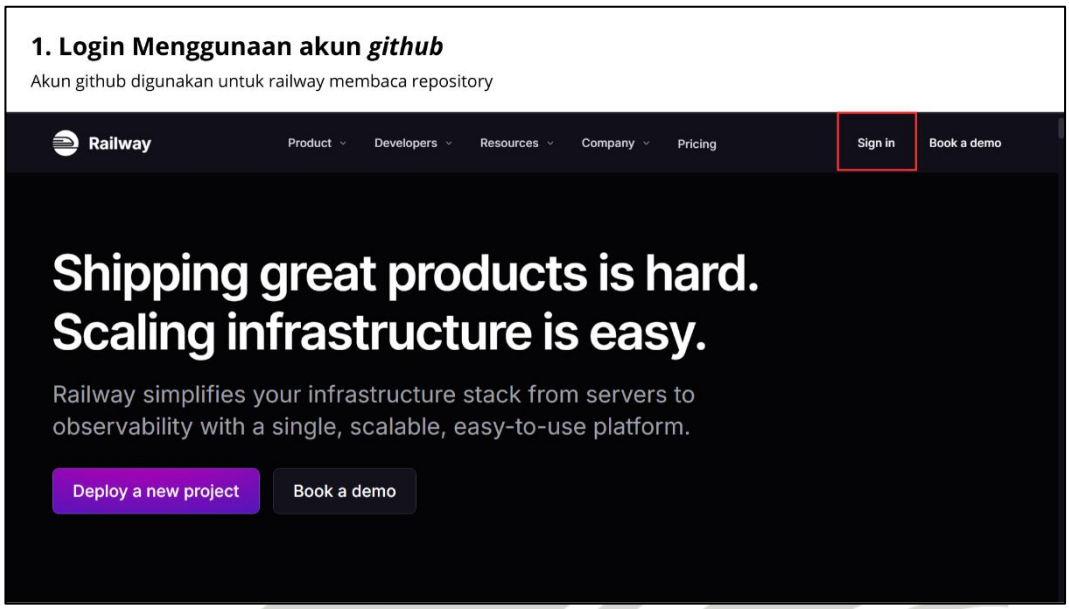


Gambar B. 9 Hasil Survei SUS Nomor 9

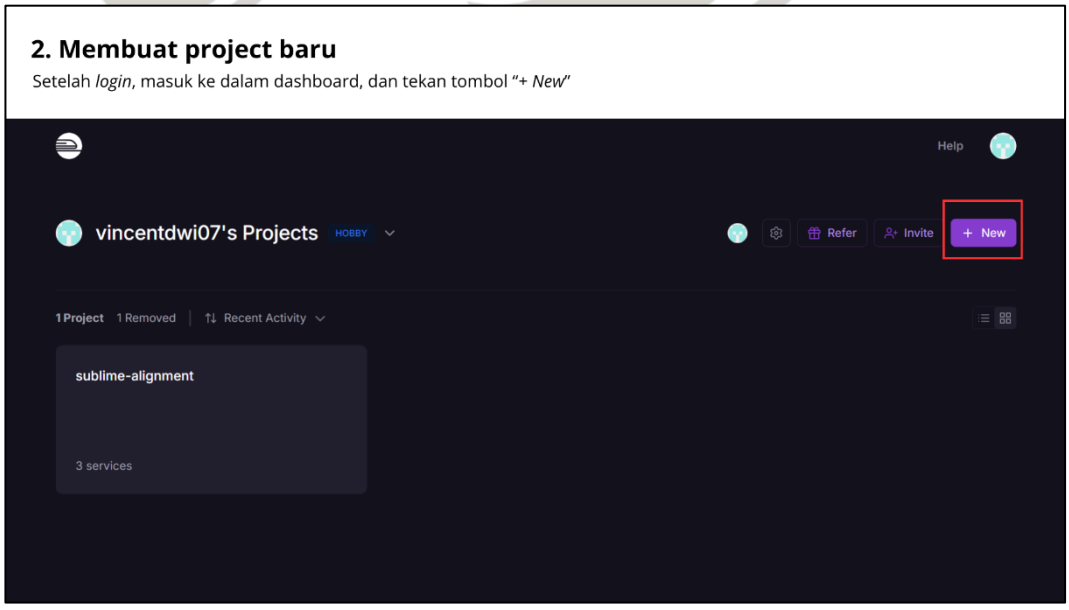


Gambar B. 10 Hasil Survei SUS Nomor 10

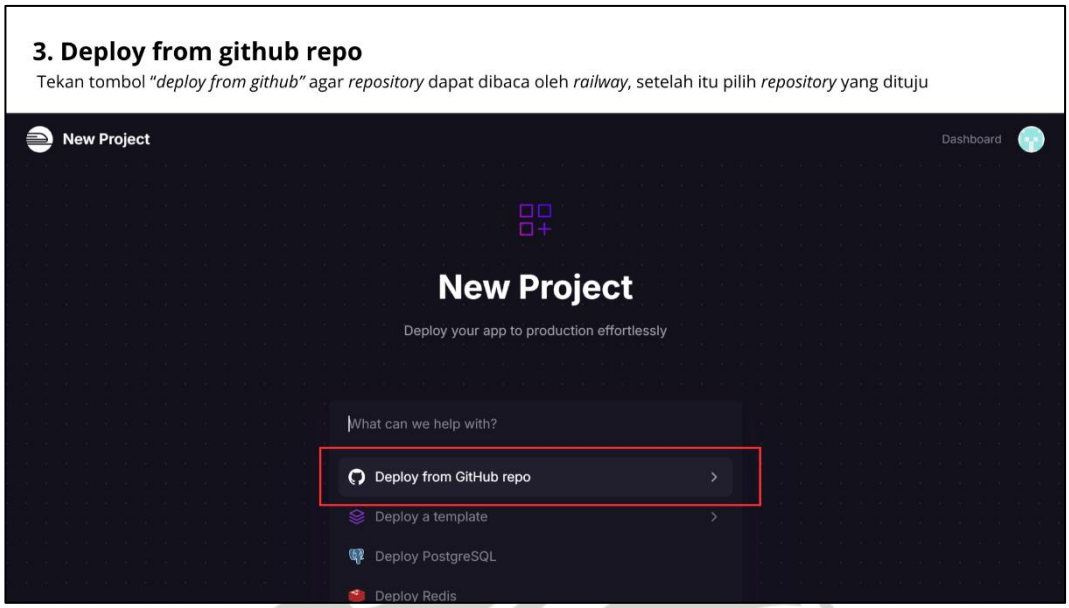
LAMPIRAN C: DOKUMENTASI PENGGUNAAN RAILWAY DAN DOMAIN



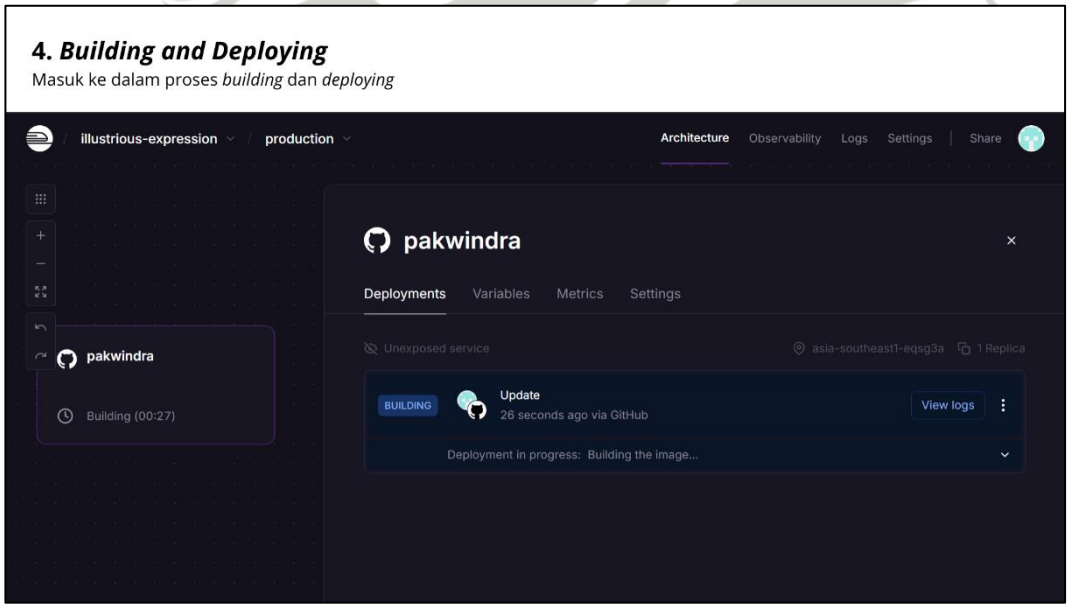
Gambar C. 1 Dokumentasi Langkah 1



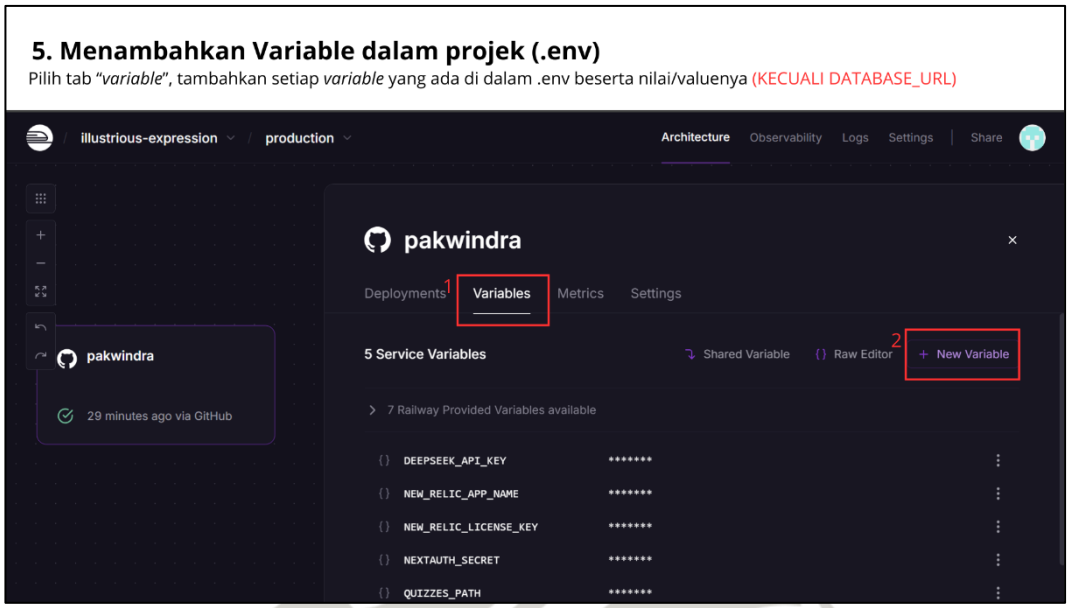
Gambar C. 2 Dokumentasi Langkah 2



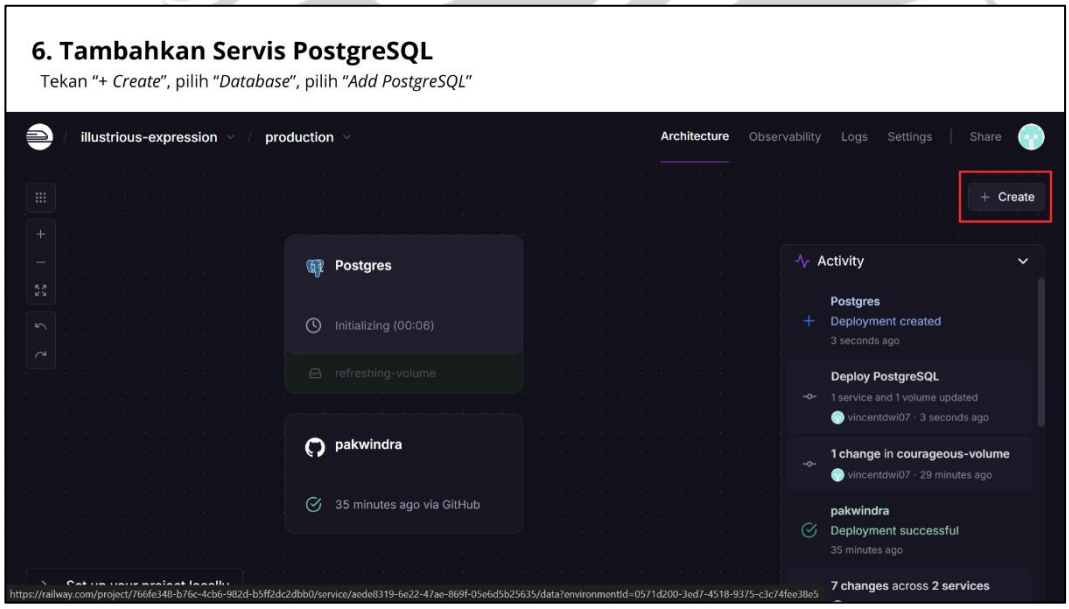
Gambar C. 3 Dokumentasi Langkah 3



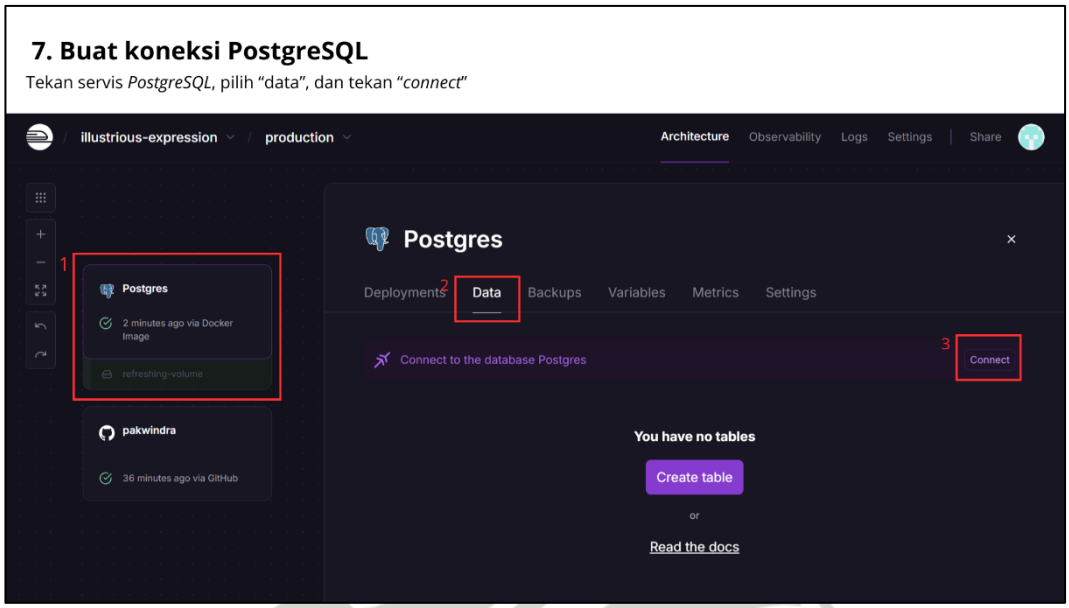
Gambar C. 4 Dokumentasi Langkah 4



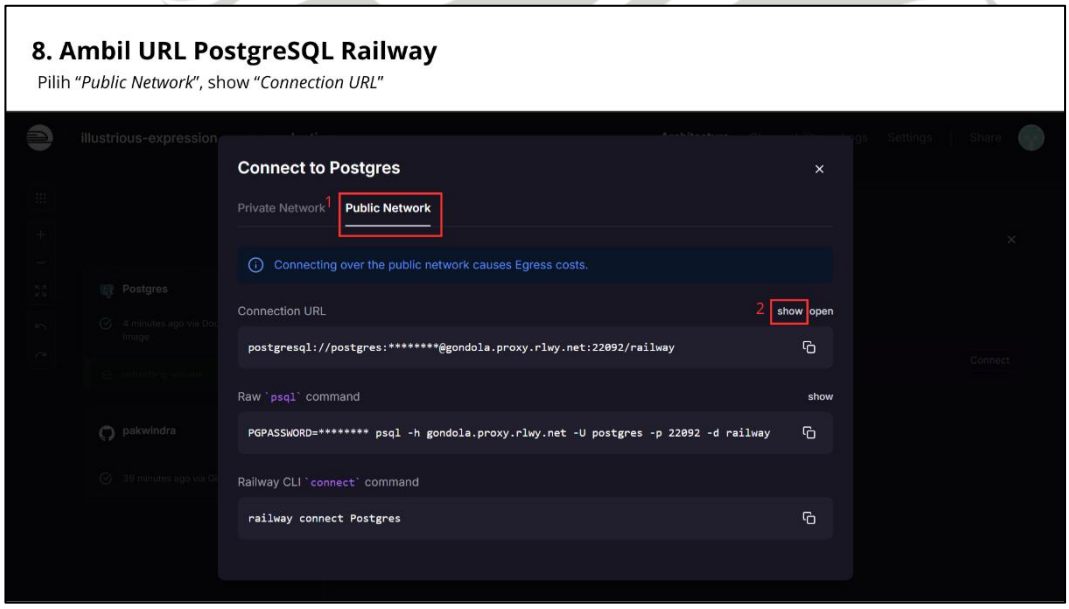
Gambar C. 5 Dokumentasi Langkah 5



Gambar C. 6 Dokumentasi Langkah 6



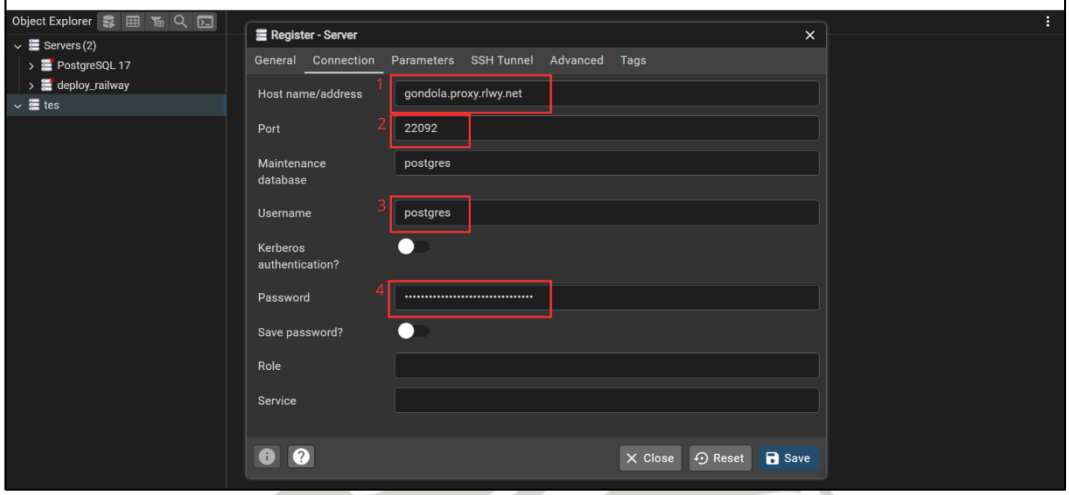
Gambar C. 7 Dokumentasi Langkah 7



Gambar C. 8 Dokumentasi Langkah 8

9. Sambungkan postgreSQL railway ke local

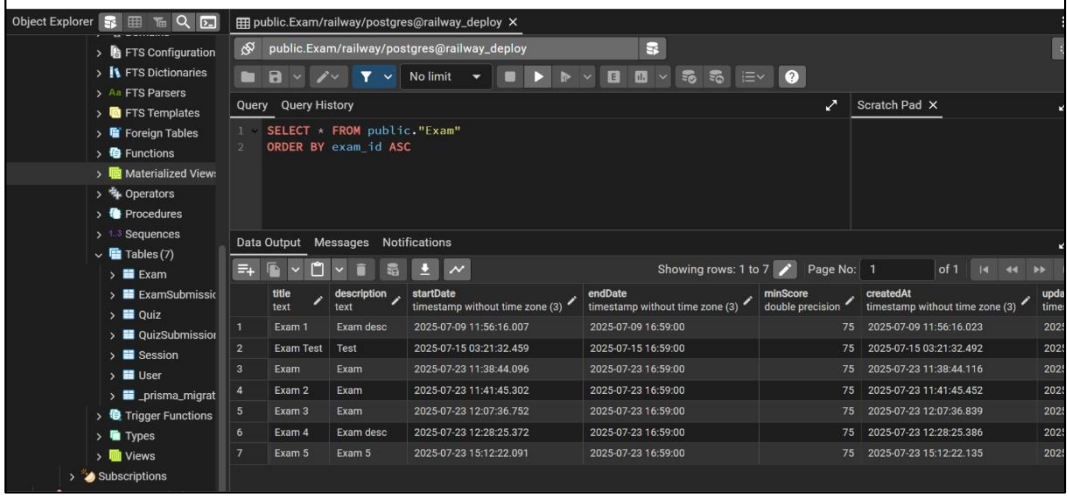
Isi setiap data dari *connection URL* ke *server postgreSQL local* (*host name*, *port*, *username* dan *password*)



Gambar C. 9 Dokumentasi Langkah 9

10. Ambil data local ke database railway

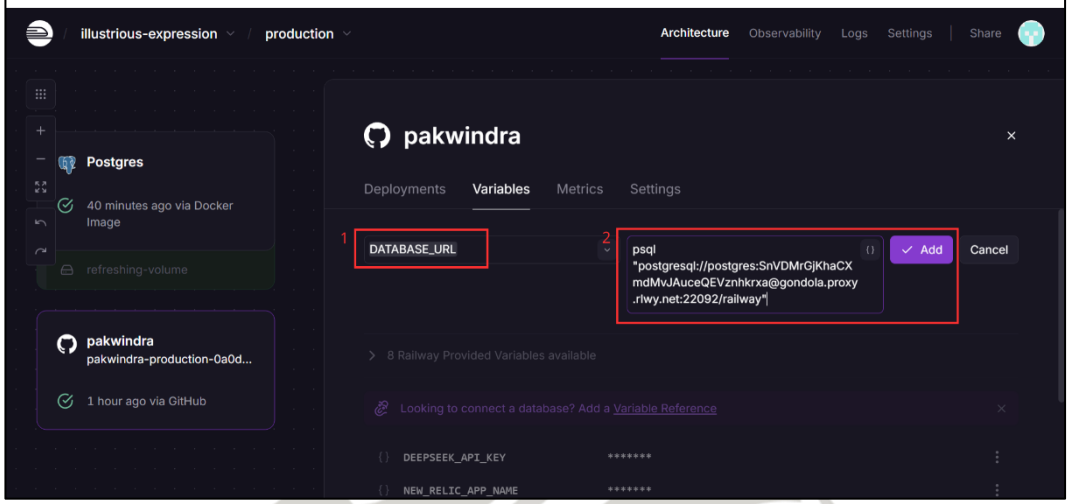
Backup data *local* sehingga mendapatkan *file* .sql, lalu *import* ke dalam *database* yang telah dibuat dari *railway*



Gambar C. 10 Dokumentasi Langkah 10

11. Penambahan variabel database_url

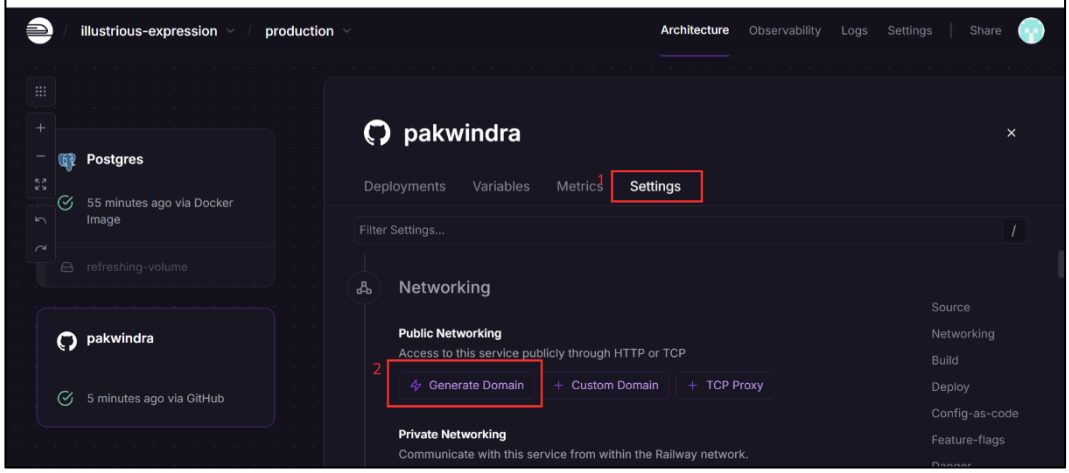
Tambahkan *variable database_url* ke dalam projek dengan value/nilai berisi *connection URL* sebelumnya



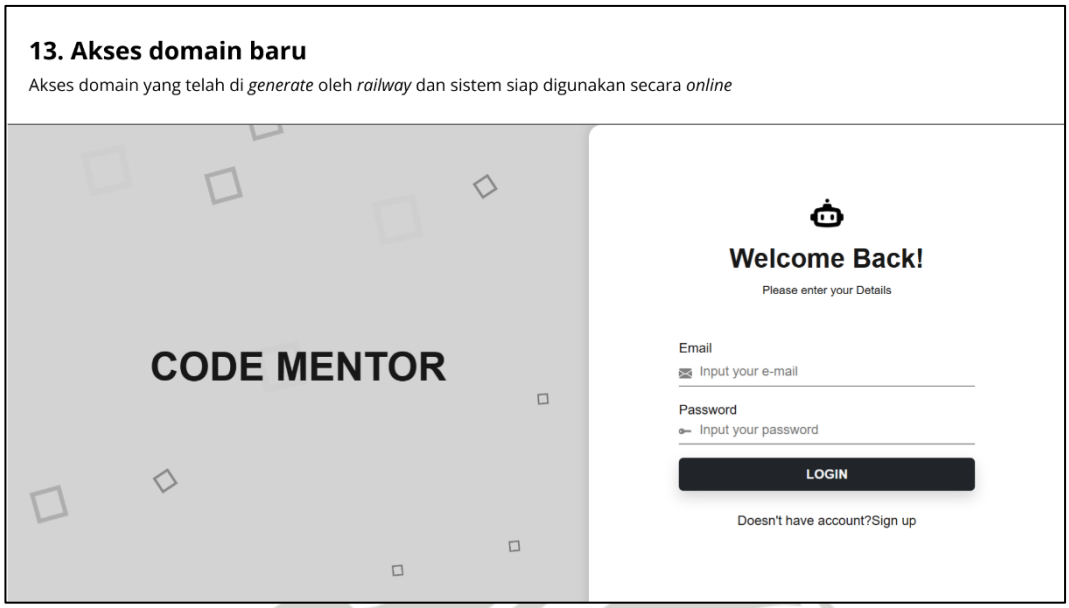
Gambar C. 11 Dokumentasi Langkah 11

12. Penambahan domain

Pilih *Settings*, scroll ke bawah hingga menemukan *Networking*, dan tekan *generate domain*, jika memiliki *custom domain*, dapat disambungkan menggunakan *nameserver* dari *provider*



Gambar C. 12 Dokumentasi Langkah 12



Gambar C. 13 Dokumentasi Langkah 13