 ISSN:2302-8149
e-ISSN:2540-9719

Penerapan Kecerdasan Buatan dengan *K-Means* untuk Penilaian Soal Pemrograman

Application of Artificial Intelligence using K-Means for Programming Question Assessment

[1]Waliyyudin*, [2]Ichsan Ibrahim
[1,2]Program Studi Teknik Informatika, STMIK IM Bandung
[1,2]Jl. Belitung No.7 Bandung, Indonesia
*e-mail: wyyuddin@gmail.com, ichsanibrahim@stmik-im.ac.id

(***received*:** 26 May 2025, ***revised*:** 6 June 2025, ***accepted*:** 7 June 2025)

Abstrak

Proses penilaian soal pemrograman secara manual masih menjadi tantangan signifikan di lingkungan pendidikan karena memakan waktu lama dan rentan terhadap kesalahan manusia. Studi observasi terhadap dosen pengampu menunjukkan bahwa lebih dari 40% di antaranya pernah melakukan kesalahan penilaian akibat kelelahan atau ketidakteraturan standar evaluasi. Penelitian ini bertujuan mengembangkan sistem penilaian otomatis menggunakan kecerdasan buatan untuk meningkatkan objektivitas dan efisiensi. Metode yang digunakan adalah algoritma *K-Means clustering* karena kemampuannya mengelompokkan jawaban berdasarkan kemiripan logika dan struktur kode secara efisien, bukan sekadar kesamaan teks. Lima kategori penilaian dijadikan standar pengelompokan: Logika dan Algoritma, Struktur Data, *Object-Oriented Programming (OOP)*, Implementasi, dan Penanganan Error. Sistem dikembangkan dengan pendekatan *Agile Development* dan dievaluasi menggunakan data jawaban mahasiswa dari mata kuliah pemrograman. Validasi performa sistem dilakukan secara kuantitatif menggunakan *ground truth label* dari penilaian manual untuk mengukur akurasi klaster sebesar 87%, waktu proses penilaian rata-rata 4,5 detik per soal dibandingkan 13 detik pada metode manual (efisiensi 65%), serta penurunan standar deviasi nilai antar dosen dari 7,5 menjadi 2,8 poin. Hasil pengujian menunjukkan sistem mampu memberikan umpan balik *real-time* secara akurat. Penelitian ini dibatasi pada soal pemrograman tingkat kesulitan mudah hingga sulit. Ke depan, sistem ini dapat dikembangkan lebih lanjut dengan integrasi analisis sintaks lanjutan dan perluasan standar evaluasi untuk mendukung penerapan skala besar.

Kata kunci: penilaian otomatis, kecerdasan buatan, *k-means clustering*, soal pemrograman, *agile development*

Abstract

The manual assessment of programming assignments remains a significant challenge in educational settings due to its time-consuming nature and susceptibility to human error. Observational studies of course instructors reveal that over 40% have made grading mistakes, often due to fatigue or inconsistent evaluation standards. This study aims to develop an automated assessment system using artificial intelligence to enhance both objectivity and efficiency in the evaluation process. The method employed is the K-Means clustering algorithm, chosen for its ability to group answers based on similarities in logic and code structure rather than mere textual similarity. Five assessment categories were used as clustering standards: Logic and Algorithm, Data Structures, Object-Oriented Programming (OOP), Implementation, and Error Handling. The system was developed using an Agile Development approach and evaluated with student responses from programming courses. System performance was validated quantitatively by comparing cluster results against ground truth labels from manual grading. The system achieved 87% clustering accuracy, reduced the average grading time to 4.5 seconds per answer (compared to 13 seconds manually—representing a 65% efficiency gain), and decreased the inter-rater score standard deviation from 7.5 to 2.8 points. The results indicate that the system can deliver accurate real-time feedback. This study focused on programming questions ranging from easy to hard difficulty levels. In the future, the system could be enhanced by

 ISSN:2302-8149
e-ISSN:2540-9719

integrating advanced syntax analysis and expanding the evaluation criteria to support large-scale deployment.

Keywords: *automated assessment, artificial intelligence, k-means clustering, programming problems, agile development*

1 Pendahuluan

Penilaian terhadap soal pemrograman merupakan komponen penting dalam proses evaluasi pembelajaran, khususnya dalam bidang ilmu komputer dan teknologi informasi. Namun, praktik penilaian yang masih bersifat manual di banyak institusi pendidikan kerap menimbulkan berbagai permasalahan. Di antaranya adalah lamanya waktu koreksi karena penguji harus memeriksa setiap baris kode secara individu [1], serta tingginya risiko *human error* yang menyebabkan ketidakkonsistenan dan ketidakadilan dalam pemberian nilai [2]. Berdasarkan hasil studi awal, lebih dari 40% dosen atau penguji mengaku pernah melakukan kesalahan penilaian akibat kelelahan, beban kerja berlebih, atau ketiadaan standar penilaian yang seragam [3]. Permasalahan ini semakin kompleks mengingat jawaban soal pemrograman dapat memiliki banyak variasi logika, struktur, dan pendekatan, meskipun menghasilkan keluaran yang sama. Hal tersebut menjadikan sistem penilaian manual tidak efisien dalam konteks skala besar dan sangat rentan terhadap subjektivitas [4]. Oleh karena itu, dibutuhkan sebuah solusi sistematis yang mampu menjawab tantangan ini secara objektif dan terukur [5].

Kecerdasan buatan *(Artificial Intelligence)* menawarkan pendekatan yang relevan untuk mengatasi permasalahan tersebut, salah satunya melalui metode *clustering*. Salah satu algoritma yang potensial adalah *K-Means*, yang mampu mengelompokkan jawaban berdasarkan kemiripan struktur logika dan pola kode. Sedangkan dalam konteks penilaian, telah banyak penelitian yang menggunakan pendekatan berbasis teks seperti *cosine similarity* atau *Jaro-Winkler* untuk menilai jawaban esai, tetapi belum menyentuh kompleksitas logika kode pemrograman. Pendekatan lain seperti *SMOTE* dan *Random Forest* dalam klasifikasi jawaban esai juga menunjukkan performa baik dalam domain teks, tetapi tidak dirancang untuk struktur sintaksis dalam soal pemrograman [6]. Di sisi lain, kemampuan kecerdasan buatan generatif seperti *ChatGPT 3.5* dalam menjawab soal justru mendorong perlunya redefinisi strategi penilaian yang menilai pemahaman struktural, bukan hanya kesesuaian jawaban [7]. Teknik *clustering* sendiri telah terbukti efektif dalam mengelompokkan hasil pembelajaran, khususnya dalam menilai efektivitas pendidikan pasca pandemi [8]. *K-Means* juga telah digunakan untuk mengelompokkan tingkat kesulitan soal pada ujian berbasis komputer dan mengklasifikasikan nilai berdasarkan wilayah dan jurusan dalam pengambilan keputusan akademik [9] [10].

Namun, sebagian besar penelitian yang ada masih berfokus pada pengelompokan data nilai atau analisis karakteristik soal, bukan pada penerapan sistem penilaian otomatis yang mempertimbangkan logika dan struktur kode pada soal pemrograman. Belum banyak studi yang menggunakan *K-Means* dalam konteks mengelompokkan soal pemrograman secara spesifik berdasarkan keragaman logika, terutama untuk mendukung proses penilaian secara *real-time*. Selain itu, pendekatan pengembangan sistem yang digunakan pada studi-studi sebelumnya umumnya bersifat linier dan kurang adaptif terhadap kebutuhan pengguna akhir. Oleh karena itu, penelitian ini menghadirkan pendekatan baru dengan memanfaatkan algoritma *K-Means* untuk klasifikasi jawaban pemrograman serta menerapkan metode *Agile Development* agar pengembangan sistem lebih fleksibel dan responsif terhadap umpan balik.

Nama aplikasi yang dikembangkan dalam penelitian ini adalah *BytePath*, yang berasal dari dua kata: *byte* dan *path*. Kata *byte* merepresentasikan elemen dasar dari data dalam dunia pemrograman dan komputasi, sedangkan *path* menggambarkan jalur atau proses yang ditempuh. Penggabungan keduanya mencerminkan tujuan utama aplikasi ini sebagai sistem yang memproses dan menilai jalur logika dari setiap jawaban pemrograman secara cerdas dan sistematis. Pemilihan nama ini juga menegaskan fokus aplikasi pada penilaian otomatis berbasis kecerdasan buatan yang mengutamakan pemahaman terhadap struktur kode, bukan hanya hasil keluaran akhir.

Berdasarkan latar belakang dan celah penelitian yang telah diidentifikasi, penelitian ini bertujuan untuk mengembangkan sistem penilaian otomatis berbasis kecerdasan buatan menggunakan algoritma *K-Means clustering* yang mampu mengelompokkan jawaban soal pemrograman berdasarkan

 ISSN:2302-8149
e-ISSN:2540-9719

kemiripan logika dan struktur kode. Sistem ini dirancang untuk memberikan hasil penilaian yang lebih objektif, efisien, dan *real-time*, sehingga dapat memperbaiki kualitas evaluasi pembelajaran di lingkungan pendidikan tinggi.

2 Tinjauan Literatur

Berbagai penelitian sebelumnya telah mengembangkan sistem penilaian otomatis yang berfokus pada jawaban berbasis teks esai naratif. Misalnya, penelitian oleh Alfirna Rizqi Lahitani dalam "*Automated Essay Scoring* menggunakan *Cosine Similarity* pada Penilaian Esai Multi Soal" menggunakan metode *TF-IDF* dan *cosine similarity* untuk menilai kemiripan jawaban dengan kunci jawaban ahli secara otomatis. Penelitian ini mampu mengurangi waktu koreksi dan menghasilkan skor objektif berbasis kesamaan kata [3]. Sedangkan Krisnawati Harefa dan Abdul Jabbar dalam "Aplikasi Sistem *Automated Essay Scoring* untuk Jawaban Soal Ujian dengan Menerapkan Algoritma *Jaro-Winkler*" menerapkan pendekatan pengukuran kemiripan string yang efektif untuk mendeteksi kemiripan karakter antar teks, namun belum menyentuh aspek semantik atau struktur mendalam [1]. Di sisi lain, Mi' Andri dkk. melalui penelitian "Sistem Penilaian Ujian Otomatis untuk Soal Esai Menggunakan Metode *Vector Space Model*" memanfaatkan model vektor untuk mengukur kedekatan jawaban dengan kunci, tetapi masih terbatas pada ruang lingkup teks naratif dan tidak diterapkan pada konteks teknis seperti soal pemrograman [11]. Meskipun demikian, pengembangan sistem penilaian otomatis secara umum telah menunjukkan potensi signifikan dalam mengurangi beban kerja pengajar dan meningkatkan efisiensi penilaian, seperti yang diuraikan dalam penelitian oleh Bato dan Pomperada (2025) yang fokus pada sistem penilaian otomatis terintegrasi dengan analitik kinerja siswa, meskipun tanpa membahas spesifik soal pemrograman [17]. Penelitian tersebut memperkuat argumen tentang pentingnya otomatisasi penilaian untuk pengambilan keputusan berbasis data.

Adapun penelitian yang memanfaatkan algoritma *K-Means* umumnya lebih difokuskan pada analisis data numerik akademik. Misalnya, dalam studi "Klasterisasi Nilai Ujian Sekolah Menggunakan Metode Algoritma *K-Means*" oleh Adi Zulkarnaen Saputra dkk., algoritma tersebut digunakan untuk mengelompokkan nilai berdasarkan wilayah dan jurusan guna mendukung analisis pencapaian akademik [10]. Meskipun penelitian ini membuktikan efektivitas *K-Means* dalam membentuk klaster yang merepresentasikan pola data, pendekatannya belum menyasar pada pengolahan isi atau logika dari jawaban pemrograman. Beberapa pendekatan kecerdasan buatan lain telah dieksplorasi untuk penilaian otomatis, seperti yang ditunjukkan oleh Sokač et al. (2025) yang mengaplikasikan *contrastive learning* untuk penilaian otomatis dengan analisis gradien dan ablasi fitur, memberikan wawasan tentang bagaimana fitur-fitur kode dapat dianalisis secara mendalam untuk keperluan penilaian, meskipun tidak secara langsung menggunakan *K-Means* [18].

Dengan demikian, dapat disimpulkan bahwa sebagian besar studi terdahulu masih terbatas pada konteks penilaian berbasis teks atau numerik, dan belum banyak yang meneliti penggunaan *K-Means* untuk mengelompokkan jawaban soal pemrograman berdasarkan kemiripan logika dan struktur kode. Penelitian ini mengisi celah tersebut dengan menghadirkan pendekatan yang tidak hanya mempertimbangkan kesamaan teks, tetapi juga logika penyelesaian dan variasi struktur sintaksis kode melalui penerapan algoritma *K-Means*. Ini memberikan kontribusi baru dalam pengembangan sistem penilaian otomatis yang lebih relevan untuk domain pembelajaran pemrograman.

Tabel 1. Perbandingan beberapa studi terkait

Studi	Metode Utama	Fokus Penelitian	Kelebihan	Keterbatasan
[3]	*Cosine Similarity + TF-IDF*	Penilaian Esai Multi Soal	Akurat dan cepat dalam menghitung kemiripan teks.	Tidak relevan untuk logika pemrograman dan tidak menggunakan *clustering*.
[1]	*Jaro-Winkler Distance*	Penilaian Jawaban Esai Berbasis Kemiripan Teks	Mendeteksi kemiripan string secara efektif.	Tidak memperhatikan konteks atau struktur logika jawaban.
[11]	*Vector Space Model (TF-IDF)*	Penilaian Otomatis	Menyajikan penilaian berbasis pembobotan teks	Tidak mendukung struktur kompleks seperti soal

 ISSN:2302-8149
e-ISSN:2540-9719

		Jawaban Esai	secara matematis.	pemrograman.
[10]	*K-Means Clustering*	Pengelompokan Nilai Ujian Berdasarkan Wilayah	Menunjukkan efektivitas *clustering* dalam domain akademik.	Tidak digunakan untuk menilai jawaban individual atau konteks pemrograman.
[17]	Sistem berbasis web (tidak disebutkan spesifik Kecerdasan Buatan / *Machine Learning*)	Sistem Penilaian Otomatis & Analitik Kinerja Siswa Secara Umum.	Mengurangi beban kerja pengajar, meningkatkan efisiensi penilaian, terintegrasi dengan analitik kinerja siswa.	Tidak spesifik membahas metodologi kecerdasan buatan / *Machine Learning* untuk penilaian konten, tidak fokus pada soal pemrograman.
[18]	*Contrastive Learning, Gradient Analysis, Feature Ablation*	Penilaian Otomatis Melalui *Contrastive Learning* dan Analisis Fitur.	Menganalisis fitur-fitur kode secara mendalam untuk penilaian otomatis.	Tidak secara langsung menggunakan *K-Means* sebagai algoritma klasterisasi untuk penilaian.
Penelitian Ini	Kecerdasan Buatan + *K-Means Clustering*	Penilaian Jawaban Soal Pemrograman Berbasis Logika dan Struktur Kode	Mengelompokkan jawaban berdasarkan kemiripan logika dan struktur kode, mendukung variasi solusi, memberikan nilai otomatis, dan umpan balik *real-time*.	Diuji hanya pada skala terbatas (soal pemrograman tingkat dasar, menengah hingga sulit) dan membutuhkan pengujian lebih luas di berbagai kompleksitas soal.

Dalam Tabel 1 menunjukkan sebagian besar penelitian terkait sistem penilaian otomatis masih berfokus pada jawaban teks esai naratif dengan pendekatan berbasis kemiripan teks secara permukaan. Misalnya, metode *Cosine Similarity* yang dipadukan dengan *TF-IDF* dinilai cepat dan akurat dalam menghitung kesamaan antar dokumen teks. Namun, pendekatan ini hanya mengukur kemiripan teks secara permukaan, tidak mempertimbangkan aspek logika atau struktur kode pemrograman, sehingga kurang relevan untuk soal berbasis algoritma [3] [11]. Di sisi lain, algoritma *Jaro-Winkler* digunakan untuk mengukur kesamaan karakter antar *string*, yang cukup efektif dalam mendeteksi perbedaan tipis dalam penulisan jawaban, tetapi tidak mampu memahami konteks logis yang tersirat dalam soal pemrograman atau struktur mendalam [1]. Sedangkan *Vector Space Model* juga mengandalkan pembobotan *TF-IDF* dalam ruang vektor multidimensi untuk membandingkan kemiripan antar teks, namun metode ini masih terbatas dalam menangani kompleksitas sintaksis dan struktur dalam jawaban pemrograman [11].

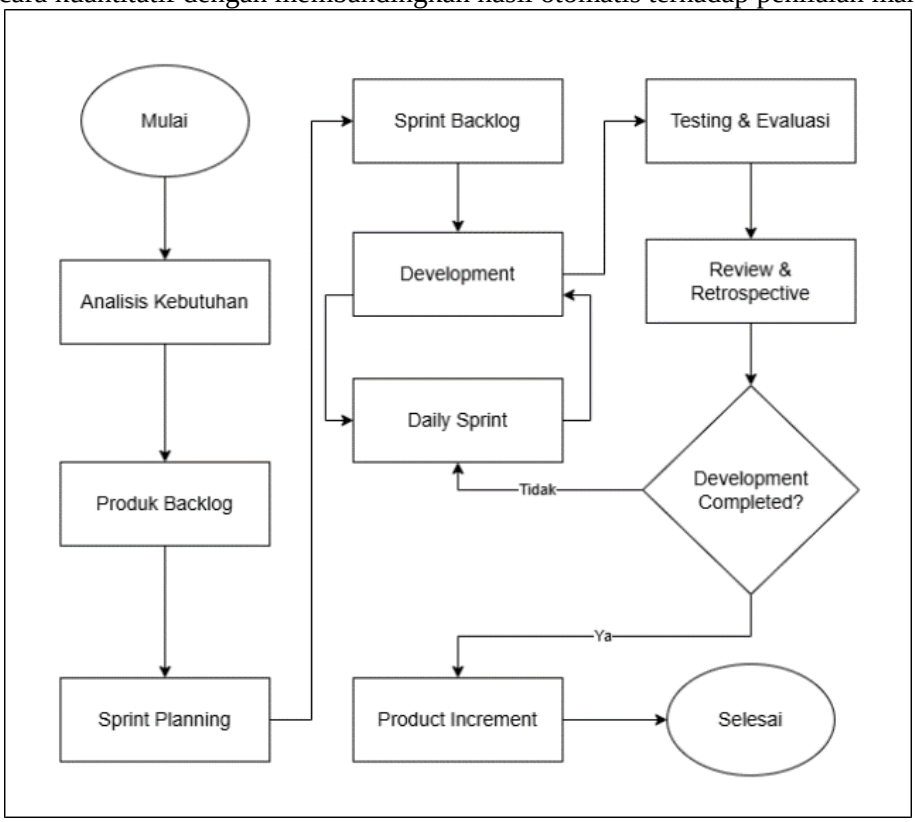
Sementara itu, penelitian yang menggunakan algoritma *K-Means* lebih banyak diterapkan untuk klasifikasi data numerik akademik. Contohnya adalah pengelompokan nilai ujian berdasarkan wilayah atau jurusan yang menunjukkan efektivitas *K-Means* dalam membentuk klaster yang mewakili pola pencapaian siswa [10]. Namun, belum ditemukan penerapan *K-Means* dalam konteks penilaian otomatis terhadap jawaban pemrograman, terutama yang mempertimbangkan kemiripan struktur logika dan sintaksis kode. Artinya, meskipun *K-Means* telah terbukti efektif untuk *clustering* dalam domain edukasi, penggunaannya belum diarahkan untuk mengatasi tantangan unik dalam penilaian jawaban pemrograman yang bervariasi secara struktur namun setara secara semantik. Bahkan, penelitian yang berfokus pada sistem penilaian otomatis secara umum, seperti oleh Bato dan Pomperada (2025), lebih menekankan pada efisiensi sistem dan analitik kinerja siswa tanpa mendalami metodologi kecerdasan buatan atau *Machine Learning* spesifik untuk penilaian konten soal pemrograman [17]. Ada pula pendekatan lain seperti *Contrastive Learning* yang diterapkan oleh Sokač et al. (2025) untuk penilaian otomatis dan analisis fitur mendalam, namun metode ini tidak secara langsung menggunakan *K-Means* sebagai algoritma klasterisasi untuk penilaian jawaban berdasarkan kemiripan logika dan struktur kode pada soal pemrograman [18].

 ISSN:2302-8149
e-ISSN:2540-9719

Berdasarkan celah tersebut, penelitian ini hadir dengan pendekatan baru yaitu mengembangkan sistem penilaian otomatis berbasis kecerdasan buatan menggunakan algoritma *K-Mean*s untuk mengelompokkan jawaban pemrograman tidak hanya berdasarkan kesamaan teks, tetapi berdasarkan kemiripan logika dan struktur kode. Sistem ini juga dirancang untuk memberikan umpan balik secara *real-time* serta dievaluasi menggunakan metrik performa seperti akurasi klaster, efisiensi waktu penilaian, dan konsistensi nilai. Dengan demikian, penelitian ini diharapkan dapat menjadi solusi praktis untuk meningkatkan objektivitas dan efisiensi dalam evaluasi soal pemrograman di lingkungan pendidikan tinggi.

3 Metode Penelitian

Penelitian ini mengembangkan sistem penilaian otomatis untuk soal pemrograman dengan pendekatan rekayasa perangkat lunak berbasis kecerdasan buatan, menggunakan algoritma *K-Means clustering* untuk mengelompokkan jawaban berdasarkan kemiripan logika dan struktur kode. Sistem dirancang dengan metode *Agile Development* yang adaptif terhadap perubahan kebutuhan pengguna, dan menggunakan data jawaban tugas mahasiswa dari mata kuliah dasar pemrograman sebagai dataset. Proses pengembangan memanfaatkan *Python* untuk pemrograman logika, *scikit-learn* untuk *K-Means*, dan framework *Laravel 11* untuk antarmuka pengguna, dengan ruang lingkup pengujian pada soal pemrograman kompleksitas sedang di lingkungan Program Studi Informatika di salah satu perguruan tinggi swasta. Teknik pengumpulan data berupa dokumentasi dan observasi, sementara variabel penelitian mencakup akurasi klaster, efisiensi waktu penilaian, dan konsistensi nilai, yang dianalisis secara kuantitatif dengan membandingkan hasil otomatis terhadap penilaian manual.



Gambar 1. Alur penelitian

Pada Gambar 1 menampilkan diagram alur penelitian yang merepresentasikan tahapan pengembangan sistem penilaian otomatis berbasis kecerdasan buatan dengan metode *Agile Development* secara ringkas dan terstruktur. Alur dimulai dari analisis kebutuhan, *product backlog, sprint planning, sprint backlog, development & implementation, daily sprint,* testing & evaluasi, *review & retrospective*, hingga desain sistem. Setiap tahap disusun agar mendukung fleksibilitas, efisiensi, dan adaptasi terhadap perubahan kebutuhan pengguna, selaras dengan prinsip pengembangan *Agile* yang telah banyak diterapkan pada pengembangan sistem serupa, seperti sistem penilaian kinerja guru berbasis web [12]. Selain itu, metode *Agile* terbukti efektif dalam mengurangi

 ISSN:2302-8149
e-ISSN:2540-9719

waktu pengembangan dengan tetap menjaga kualitas sistem, berkat pendekatan iteratif dan evaluasi berkelanjutan di setiap siklus *sprint* [12].

4 Hasil dan Pembahasan

Penelitian ini dimulai dengan analisis kebutuhan melalui studi literatur dan observasi terhadap proses penilaian manual pada mata kuliah pemrograman. Hasil observasi menunjukkan bahwa lebih dari 40% dosen mengalami kesalahan penilaian akibat kelelahan atau beban kerja berlebih, seperti yang juga dikonfirmasi oleh studi sebelumnya tentang ketidakefisienan dan ketidakobjektifan sistem manual [13]. Dosen harus memeriksa satu per satu jawaban soal pemrograman yang mungkin memiliki logika berbeda namun hasil yang sama, yang memperbesar kemungkinan kesalahan dan bias [11].

Untuk mengatasi masalah tersebut, sistem penilaian otomatis dikembangkan menggunakan metode *Agile Development* dalam siklus iteratif. Pada tahap awal, fitur utama seperti input jawaban, ekstraksi fitur logika program, dan *clustering* dengan algoritma *K-Means* dirancang dan diimplementasikan.

4.1 Analisis Kebutuhan

Analisis kebutuhan dalam penelitian ini mengungkap permasalahan pada sistem penilaian soal pemrograman yang masih bersifat manual, seperti waktu penilaian yang lama, potensi kesalahan manusia, dan ketidakkonsistenan standar. Untuk menjawab tantangan tersebut, dikembangkan sistem penilaian otomatis berbasis kecerdasan buatan dengan dukungan teknologi web dan basis data serta dataset soal pemrograman sebagai bahan pengujian.

4.2 Produk *Backlog*

Tabel 2. Produk *backlog*

Product Backlog	**Deskripsi**	**Prioritas**
UML Diagram	Merancang *Use Case Diagram*, *Activity Diagram*, dan *Class Diagram*.	Tinggi
Desain Database	Merancang skema database termasuk tabel *users*, *learning_progress*, *AI_model*, dsb.	Tinggi
Authentication & Authorization	Implementasi *register*, *login (email & Google)*, *forgot password*. Menambahkan role pengguna seperti superadmin, admin, dan pengguna.	Tinggi
Artificial Intelligence (AI) Model	Mengembangkan model kecerdasan buatan dengan *Python* untuk penilaian soal pemrograman.	Tinggi
Manajemen Sistem Pengguna	Interaksi pengguna dengan sistem aplikasi seperti *authentication*, mengisi soal, mengelola data, dan lain-lain.	Tinggi
Integrasi *Laravel & Python*	*API Flask/FastAPI* untuk komunikasi antara *Laravel* dan model kecerdasan buatan.	Tinggi
User Feedback & Iterasi	Mengumpulkan umpan balik dari pengguna dan menyempurnakan fitur.	Rendah
Testing & Debugging	Uji coba sistem, perbaikan *bug*, dan optimasi performa.	Sedang
Fase Evaluasi Akhir	Dilakukan untuk memastikan aplikasi berjalan sempurna serta menyempurnakan fitur yang terdapat *bug* pada aplikasi	Tinggi

Tabel 2 menyajikan daftar *product backlog* yang merangkum fitur utama dalam pengembangan sistem penilaian otomatis berbasis kecerdasan buatan. Setiap item disusun berdasarkan prioritas untuk mendukung proses pengembangan yang terstruktur, efisien, dan sesuai prinsip *Agile* secara iteratif.

 ISSN:2302-8149
e-ISSN:2540-9719

4.3 *Sprint Planning*

Tabel 3. *Sprint planning*

Sprint	Tanggal Mulai - Selesai	Fitur yang dikerjakan	Prioritas	Keterangan
1	4 Feb – 17 Feb 2025	*Register*, *Login*, *Forgot Password*	Tinggi	Kebutuhan awal sistem aplikasi penilaian soal pemrograman.
2	18 Feb – 3 Mar 2025	Penilaian Otomatis, Penjelasan Jawaban Soal, Dashboard Admin	Tinggi	Pemodelan kecerdasan buatan berdasarkan jawaban pengguna.
3	4 Mar – 17 Mar 2025	Manajemen Soal, Manajemen Kategori, Manajemen Konten	Tinggi	Pengelolaan data soal, kategori, dan konten aplikasi yang ditampilkan pada aplikasi.
4	18 Mar – 31 Mar 2025	Manajemen Pengguna, Manajemen *Feedback* Pengguna	Sedang	Memanajemen data pengguna dan *feedback* yang diisi pengguna.
5	1 Apr – 14 Apr 2025	Dashboard Pengguna, Mengisi Soal	Tinggi	Interaksi pengguna dengan aplikasi terhadap sistem penilaian.
6	15 Apr – 28 Apr 2025	Manajemen Profil, *Feedback*	Rendah	Fitur tambahan untuk kebutuhan informasi aplikasi.

Tabel 3 merangkum perencanaan *sprint* dalam enam tahap pengembangan, yang difokuskan pada penyelesaian fitur utama sesuai urgensi dan kebutuhan sistem. Pendekatan ini mencerminkan penerapan metode *Agile* secara terstruktur untuk memastikan proses pengembangan berjalan efisien dan adaptif.

4.4 *Sprint Backlog*

Tabel 4. *Sprint backlog*

Sprint	*Item Sprint Backlog*	Estimasi Waktu
1	Desain Diagram	11 Hari
	Desain Database	1 Hari
	Register	0,5 Hari
	Login	0,5 Hari
	Forgot Password	1 Hari
2	Penilaian Otomatis	5 Hari
	Penjelasan Jawaban Otomatis	5 Hari
	Dashboard Admin	4 Hari
3	Manajemen Soal	6 Hari
	Manajemen Kategori	6 Hari
	Manajemen Konten	2 Hari
4	Manajemen Pengguna	7 Hari
	Manajemen *Feedback* Pengguna	7 Hari
5	Dashboard Pengguna	7 Hari
	Mengisi Soal	7 Hari
6	Manajemen Profil	6 Hari
	Feedback	8 Hari

 ISSN:2302-8149
e-ISSN:2540-9719

Tabel 4 menyajikan *sprint backlog* yang mencakup daftar tugas dan estimasi waktu penyelesaian pada tiap *sprint*. Pembagian tugas ini mendukung pengembangan sistem secara terstruktur dan efisien sesuai prinsip *Agile*.

4.5 *Development & Daily Sprint*

Development

Tahap pengembangan dilakukan secara iteratif berdasarkan *sprint backlog*, dengan penerapan algoritma *K-Means* untuk mengelompokkan jawaban pemrograman berdasarkan kemiripan logika dan struktur kode. Proses *clustering* didasarkan pada lima kategori utama sebagai acuan pembobotan fitur.

Daily Sprint

Tabel 5. *Daily sprint* 1 (4 Feb – 17 Feb 2025)

Hari	Progres	Rencana	Kendala	Solusi
Hari 1 - 3	Struktur *UML Diagram*	Penyelesaian *UML diagram*.	*Stuck idea* untuk beberapa diagram.	Cari referensi diagram terkait topik penilaian otomatis.
Hari 4 - 6	Desain Database Aplikasi	Gambaran detail sistem aplikasi.	-	-
Hari 7 - 8	*Authentication*	Konfigurasi dengan database.		
Hari 9 - 10	*Authorization*	Integrasi antarmuka pengguna dengan akun yang *login*.	-	-
Hari 11 - 12	Fitur *Forgot Password*	Konfigurasi dengan database pengguna.	-	-
Hari 13	Tampilan Antarmuka Pengguna	Pengujian *register*, *login*, dan *forgot password.*	Tampilan tidak informatif.	Perbarui tampilan menjadi *user-friendly.*
Hari 14	*Sprint Review & Retrospective*	Evaluasi fitur dan *sprint*.	-	Perbaikan antarmuka *register*, *login*, dan *forgot password*.

Pada tabel 5 menunjukkan *Daily Sprint 1* (4–17 Februari 2025) difokuskan pada perancangan awal sistem, termasuk diagram *UML*, *database*, autentikasi, dan antarmuka pengguna. Tantangan seperti desain kurang informatif diatasi melalui pencarian referensi dan penyempurnaan tampilan.

Tabel 6. *Daily sprint* 2 (18 Feb – 3 Mar 2025)

Hari	Progres	Rencana	Kendala	Solusi
Hari 1 - 5	Pemodelan Sistem Kecerdasan Buatan.	Hasil *output* nilai sesuai dengan jawaban yang diisi pengguna.	Penilaian tidak akurat.	Perbarui algoritma fungsi.
Hari 6 - 10	Konfigurasi Penjelasan Sistem Kecerdasan Buatan dengan Jawaban Pengguna.	Tampil penjelasan soal yang komprehensif dan jelas.	Penjelasan melebar terlalu luas.	Menambah batasan topik penjelasan.
Hari 11 - 13	Penyusunan informasi dan data dashboard admin.	Tampil informasi penting terkait data aplikasi.	-	-
Hari 14	*Sprint Review & Retrospective*	Evaluasi fitur.	-	Pengembangan model kecerdasan buatan.

 ISSN:2302-8149
e-ISSN:2540-9719

Tabel 6 menampilkan *Daily Sprint 2* (18 Februari–3 Maret 2025) berfokus pada pengembangan model kecerdasan buatan dan integrasi penjelasan soal. Kendala seperti akurasi dan keluasan penilaian diatasi melalui pembaruan algoritma dan pembatasan topik, diakhiri dengan evaluasi dan penguatan model.

Tabel 7. *Daily sprint* 3 (4 Mar – 17 Mar 2025)

Hari	Progres	Rencana	Kendala	Solusi
Hari 1 - 6	*CRUD* Soal	Manajemen data soal pemrograman.	Terbatasnya data soal pemrograman.	Ambil sebagian data soal dari ujian kuliah.
Hari 7 - 12	*CRUD* Kategori	Manajemen kategori sebagai landasan soal.	-	-
Hari 13	*CRUD* Konten	Kelola data konten aplikasi, seperti: foto, penjelasan, dan lain-lain.	-	-
Hari 14	*Sprint Review & Retrospective*	Evaluasi fitur.	-	Siap dilakukan integrasi dengan antarmuka pengguna.

Tabel 7 menunjukkan *Daily Sprint 3* (4–17 Maret 2025) difokuskan pada pengelolaan data soal, kategori, dan konten aplikasi. Keterbatasan soal diatasi dengan mengambil referensi dari materi perkuliahan, dan *sprint* ditutup dengan evaluasi serta persiapan integrasi antarmuka pengguna.

Tabel 8. *Daily sprint* 4 (18 Mar – 31 Mar 2025)

Hari	Progres	Rencana	Kendala	Solusi
Hari 1 - 7	*CRUD* Data Pengguna	Tampil informasi biodata dan data akun.	-	-
Hari 8 - 13	*Crud Feedback* Pengguna	Konfigurasi data *feedback* dengan antarmuka admin.	-	-
Hari 14	*Sprint Review & Retrospective*	Evaluasi fitur.	-	-

Pada tabel 8 menampilkan *Daily Sprint 4* (18–31 Maret 2025) berfokus pada pengelolaan data pengguna dan integrasi fitur umpan balik ke dalam antarmuka admin. *Sprint* ditutup dengan evaluasi fitur sebagai dasar peningkatan di iterasi berikutnya.

Tabel 9. *Daily sprint* 5 (1 Apr – 14 Apr 2025)

Hari	Progres	Rencana	Kendala	Solusi
Hari 1 - 7	Integrasi Data dan Informasi Konten	Tampil data dan informasi konten aplikasi.	Desain antarmuka yang terlalu polos.	Perbarui desain antarmuka berdasarkan referensi aplikasi *online*.
Hari 8 - 12	Konfigurasi dan Integrasi Data Soal Berdasarkan Kategori.	Data soal pemrograman berelasi dengan kategori soal.	Kurang kolom *foreign key* pengguna dalam tabel soal.	Tambah kolom *foreign key user_id*.
Hari 13	Uji Coba Input Jawaban Soal	Tampil nilai secara otomatis saat submit soal.	*Bug* kecil penampilan informasi nilai.	Fix algoritma fungsi pada *controller*.
Hari 14	*Sprint Review & Retrospective*	Evaluasi fitur.	-	Siap dilakukan uji coba pengguna.

 ISSN:2302-8149
e-ISSN:2540-9719

Tabel 9 menampilkan *Daily Sprint 5* (1–14 April 2025) difokuskan pada integrasi konten soal, pengelompokan berdasarkan kategori, dan uji coba input jawaban. Kendala teknis berhasil diatasi melalui perbaikan *database* dan antarmuka, serta ditutup dengan evaluasi dan persiapan uji coba pengguna.

Tabel 10. *Daily sprint* 6 (15 Apr – 28 Apr 2025)

Hari	Progres	Rencana	Kendala	Solusi
Hari 1 - 6	*CRUD* Profil Akun Pengguna	Konfigurasi profil dan biodata akun pengguna.	-	-
Hari 7 - 12	*CRUD Feedback* pada Antarmuka Pengguna	Konfigurasi pesan *feedback* terhadap aplikasi.	-	-
Hari 13	Uji Coba Fitur di Antarmuka Pengguna	Validasi pengisian jawaban soal dan *feedback* pengguna.	-	-
Hari 14	*Sprint Review & Retrospective*	*Retrospective* proyek.	-	Siap dilakukan uji coba keseluruhan.

Tabel 10 menunjukkan *Daily Sprint 6* (15–28 April 2025) yang berfokus pada konfigurasi profil pengguna, fitur umpan balik, dan uji coba pengisian jawaban. Tahapan ini memastikan fungsi antarmuka berjalan lancar dan diakhiri dengan evaluasi proyek serta persiapan pengujian sistem menyeluruh.

4.6 Testing & Evaluasi

Setelah pengembangan selesai, sistem diuji menggunakan dataset jawaban mahasiswa yang beragam. Evaluasi dilakukan berdasarkan metrik utama: akurasi klaster (atau konsistensi nilai antar klaster), efisiensi waktu penilaian, dan konsistensi nilai antar dosen.

Hasil pengujian dibandingkan secara langsung antara metode manual dan sistem otomatis, yang disajikan pada Tabel 11 berikut:

Tabel 11. Perbandingan kinerja penilaian antara metode manual dan sistem otomatis

Metrik	Metode Manual	Sistem Otomatis	Peningkatan Efisiensi (%)
Akurasi/Konsistensi (Standar Deviasi Nilai Antar Dosen)	7,5 poin	2,8 poin	-
Waktu Penilaian Rata-rata (Detik per Soal)	13 detik	4,5 detik	65%

Dari tabel 11 terlihat bahwa sistem otomatis mampu meningkatkan efisiensi waktu penilaian sebesar 65% dibanding metode manual, dengan waktu rata-rata hanya 4,5 detik per soal. Selain itu, konsistensi penilaian antar dosen yang diukur melalui standar deviasi nilai mengalami penurunan signifikan dari 7,5 menjadi 2,8 poin, menunjukkan peningkatan akurasi dan konsistensi dalam penilaian.

Selain efisiensi dan konsistensi, sistem juga mampu memberikan umpan balik secara *real-time* kepada mahasiswa. Fitur ini membantu mahasiswa memahami kekuatan dan kelemahan jawaban mereka secara objektif dan langsung, sehingga mendukung proses pembelajaran yang lebih efektif.

4.7 *Review & Restrospective*

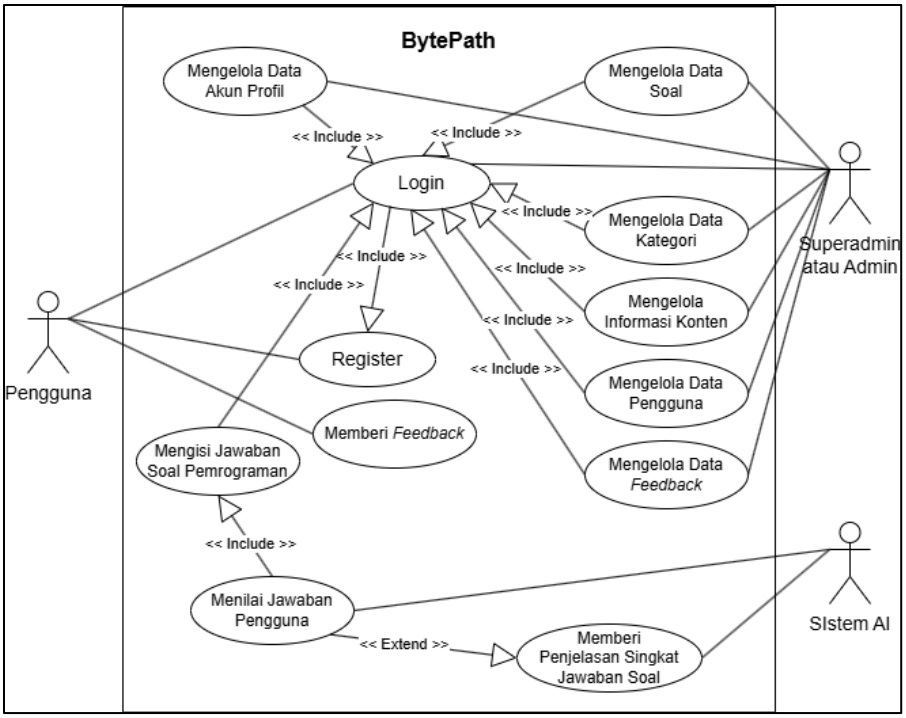
Setelah tahap pengujian tiap *sprint*, dilakukan sesi *review & retrospective* untuk menilai efektivitas *sprint* yang telah dijalankan. Hasil dari sesi ini digunakan untuk memperbaiki kekurangan pada *sprint* sebelumnya dan melakukan perencanaan ulang bila dibutuhkan. Dalam konteks ini, *feedback* dari pengguna sistem seperti dosen dan mahasiswa juga menjadi bahan pertimbangan penting untuk iterasi berikutnya.

4.8 *Product Increment*

Desain sistem dilakukan secara modular, dimulai dari perancangan arsitektur sistem, alur data, hingga antarmuka pengguna. Sistem dirancang berbasis web agar mudah diakses oleh dosen dan mahasiswa. Modul penilaian otomatis menjadi inti dari sistem, dengan algoritma *K-Means* yang

 ISSN:2302-8149
e-ISSN:2540-9719

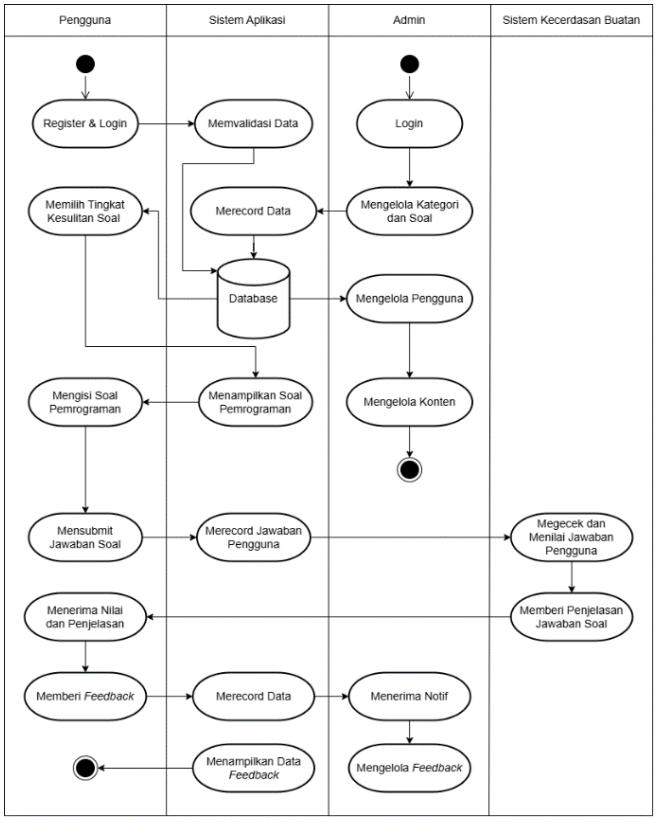
bekerja di belakang untuk melakukan proses *clustering*. Desain ini memastikan fleksibilitas sistem dalam menerima berbagai jenis jawaban pemrograman, serta kemampuan untuk memberikan umpan balik *real-time* kepada pengguna.



Gambar 2. *Use case diagram*

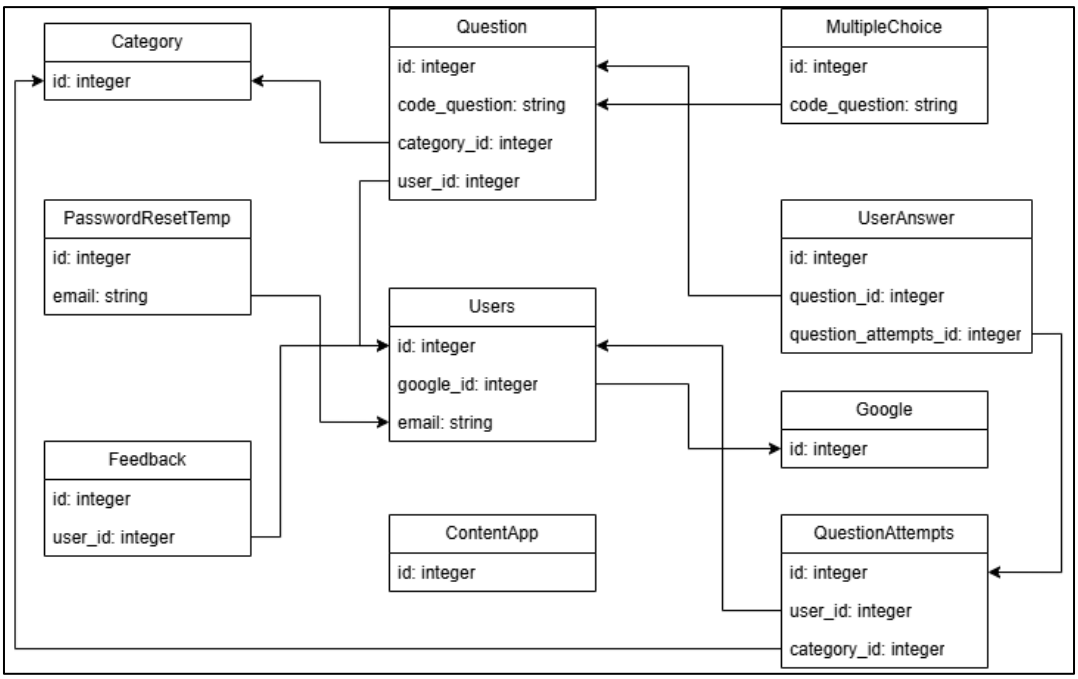
Gambar 2 menampilkan diagram *Use Case* untuk sistem *BytePath*, yang memodelkan interaksi antara pengguna, admin, dan kecerdasan buatan dalam pengelolaan dan penilaian soal pemrograman. Diagram mencakup fungsi utama seperti *login*, *registrasi*, pengelolaan data pengguna, pengelolaan soal, kategori, konten, serta pemberian *feedback* dan evaluasi jawaban secara otomatis. Relasi antar *use case* menggunakan dependensi *include* dan *extend*, mencerminkan keterkaitan antarfitur dalam sistem yang mendukung penilaian jawaban pemrograman secara efisien dan adaptif.

 ISSN:2302-8149
e-ISSN:2540-9719



Gambar 3. *Activity diagram*

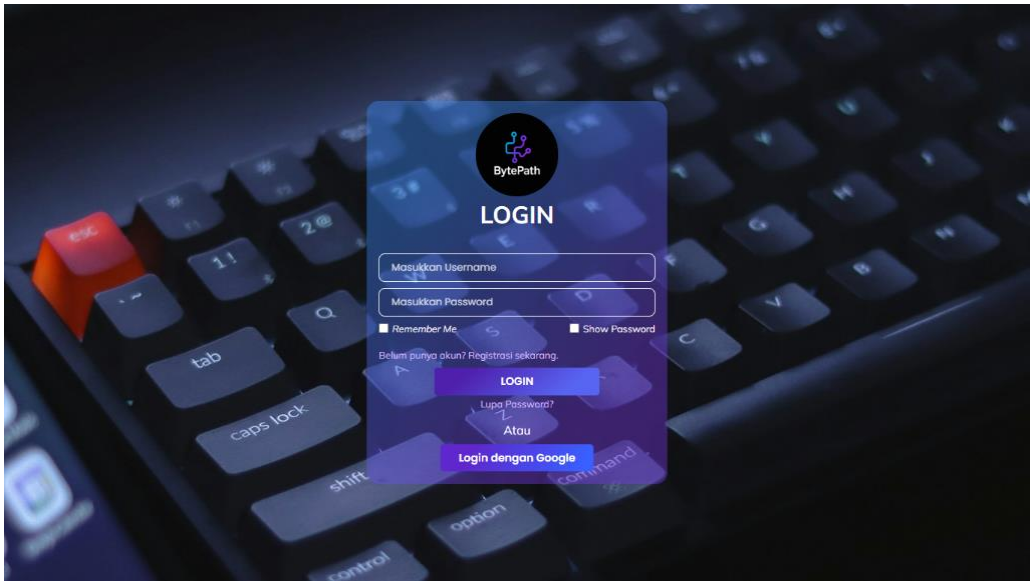
Gambar 3 merupakan *Activity Diagram* yang menggambarkan alur aktivitas pengguna, admin, sistem aplikasi, dan kecerdasan buatan dalam proses penilaian soal pemrograman. Diagram memvisualisasikan tahapan mulai dari *registrasi* dan *login*, pemilihan tingkat kesulitan soal, pengisian dan penyimpanan jawaban, hingga pemberian nilai otomatis dan umpan balik. Alur ini menunjukkan integrasi yang jelas antara pengguna dan sistem kecerdasan buatan untuk mendukung penilaian yang cepat, objektif, dan adaptif terhadap variasi logika kode pemrograman.



Gambar 4. *Class diagram*

 ISSN:2302-8149
e-ISSN:2540-9719

Gambar 4 merupakan *Class Diagram* yang merepresentasikan struktur basis data sistem penilaian soal pemrograman. Diagram ini menunjukkan relasi antar kelas seperti *Users, Question, Category, Feedback*, hingga *UserAnswer*, yang terhubung melalui atribut kunci dan relasi seperti *foreign key*. Struktur ini mendukung pengelolaan data pengguna, soal, jawaban, kategori, serta interaksi *feedback* untuk menghasilkan penilaian otomatis yang lebih objektif dan terstruktur.



Gambar 5. Tampilan *login*

Gambar 5 menampilkan halaman *login* pada aplikasi *BytePath* yang berperan penting dalam menjaga keamanan sistem dan menerapkan otorisasi akses. Form *login* ini berfungsi untuk memvalidasi kredensial pengguna dan menyaring akses berdasarkan peran masing-masing, sehingga hanya pengguna yang memiliki hak akses yang sesuai yang dapat memasuki tampilan utama aplikasi.

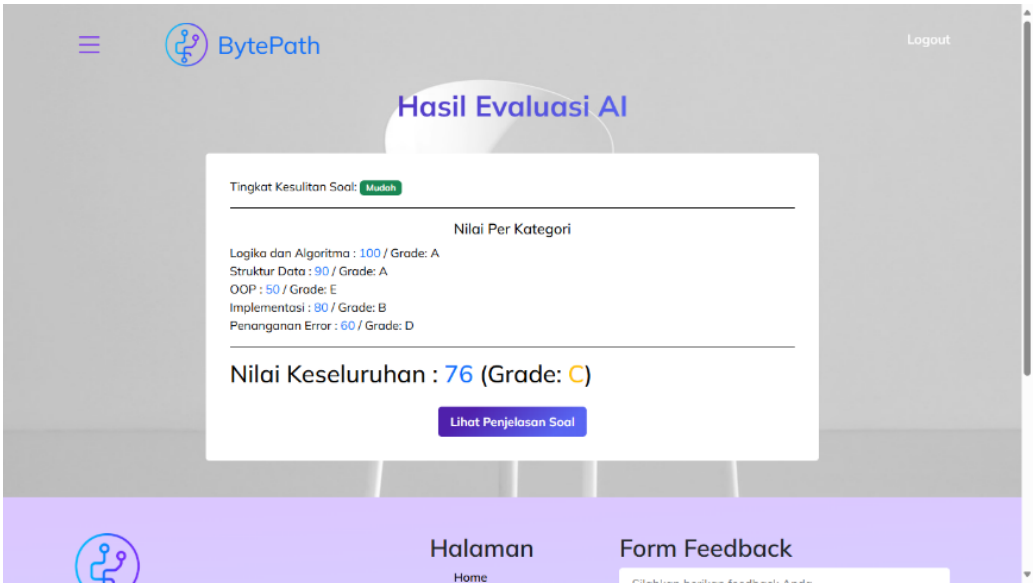
Gambar 6. Tampilan dashboard pengguna

Gambar 6 menunjukkan tampilan antarmuka dashboard pengguna pada aplikasi *BytePath*, yang menyajikan informasi penting terkait sistem penilaian otomatis. Selain itu, dashboard ini juga menampilkan daftar kategori soal yang digunakan sebagai acuan dalam proses pengelompokan dan penilaian jawaban pemrograman secara otomatis.

 ISSN:2302-8149
e-ISSN:2540-9719

Gambar 7. Tampilan halaman mengisi soal pemrograman

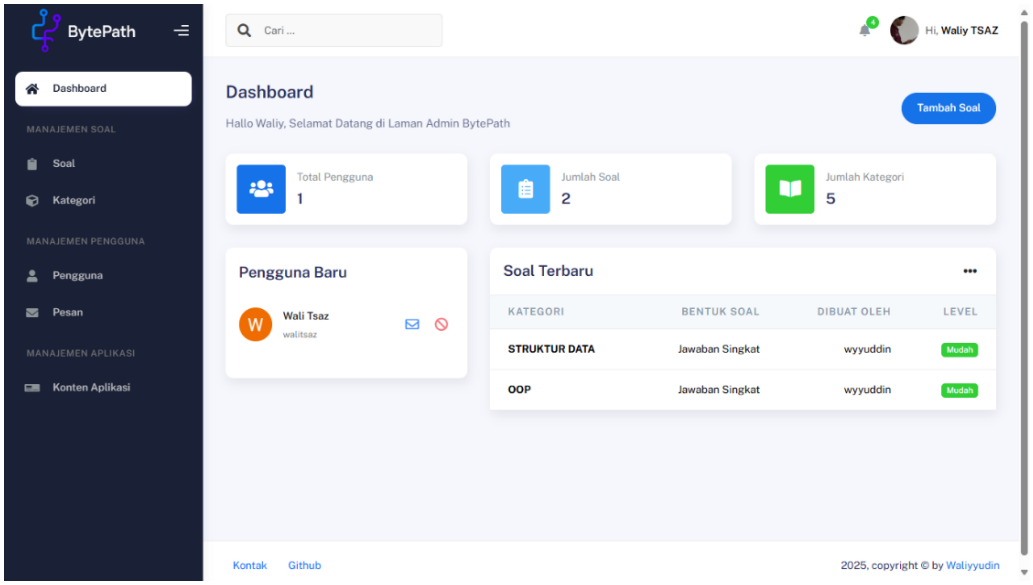
Gambar 7 menunjukkan halaman pengisian soal pemrograman pada aplikasi *BytePath* dirancang agar pengguna dapat menjawab seluruh soal yang disediakan dengan fleksibel, baik dalam bentuk jawaban singkat, jawaban panjang, maupun pilihan ganda. Antarmuka halaman ini memungkinkan pengguna untuk menginput jawaban sesuai instruksi dan format yang telah ditentukan pada setiap soal.



Gambar 8. Tampilan hasil penilaian

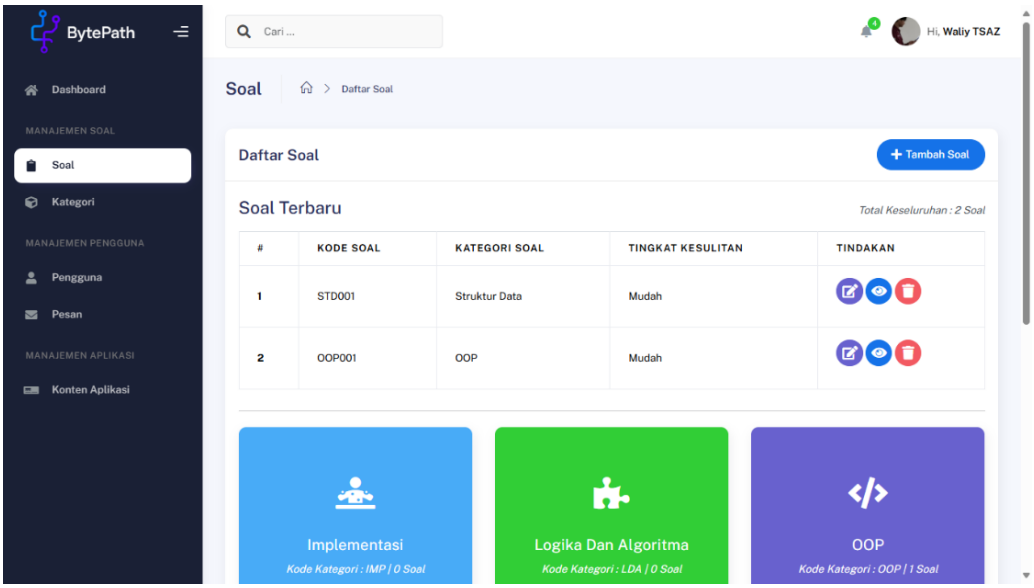
Gambar 8 menunjukkan halaman hasil penilaian pada aplikasi BytePath yang menampilkan skor akhir dari sistem kecerdasan buatan terhadap jawaban pengguna. Setiap soal disertai nilai hasil dan penjelasan singkat tentang topik yang diuji, sehingga pengguna dapat mengetahui poin yang perlu diperbaiki. Pemberian skor didasarkan pada rentang nilai: A (≥90), B (≥80), C (≥70), D (≥60), dan E (<60), yang membantu pengguna memahami tingkat pencapaian mereka secara objektif.

 ISSN:2302-8149
e-ISSN:2540-9719



Gambar 9. Tampilan dashboard admin

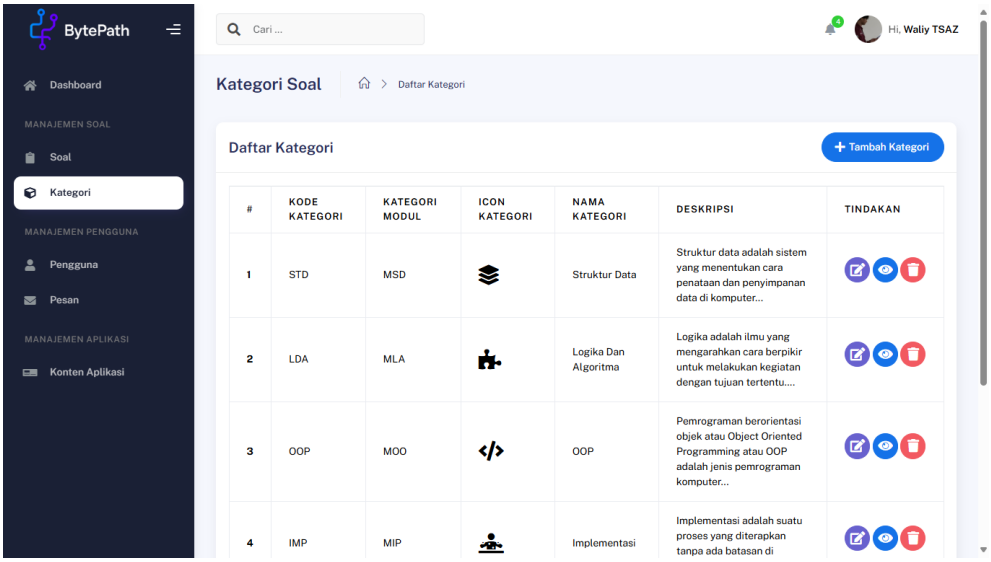
Gambar 9 menunjukkan perancangan sebagai pusat kendali sistem yang menampilkan informasi penting, seperti total pengguna terdaftar dalam aplikasi, total soal pemrograman yang tersedia, serta total kategori soal yang telah dibuat. Selain itu, halaman ini dilengkapi dengan tabel interaktif yang memudahkan admin dalam memanajemen data pengguna dan soal pemrograman.



Gambar 10. Tampilan halaman manajemen soal

Gambar 10 menampilkan halaman manajemen soal pemrograman pada aplikasi *BytePath* dirancang untuk memudahkan admin dalam memfilter soal berdasarkan kategori, lengkap dengan informasi jumlah soal di masing-masing kategori. Selain itu, halaman ini juga menampilkan tiga soal pemrograman terbaru, sehingga admin dapat dengan mudah mengelola dan memantau konten soal yang tersedia dalam sistem.

 ISSN:2302-8149
e-ISSN:2540-9719



Gambar 11. Tampilan halaman manajemen kategori

Gambar 11 menampilkan halaman manajemen kategori soal pemrograman pada aplikasi *BytePath* dirancang dalam bentuk tabel interaktif yang memudahkan admin dalam mengelola kategori soal. Melalui halaman ini, admin dapat menambah, mengedit, atau menghapus kategori soal dengan cepat dan terstruktur, sehingga memastikan kategori soal selalu terorganisir dengan baik sesuai kebutuhan sistem.

Hasil pengujian menunjukkan bahwa algoritma *K-Means* efektif dalam mengelompokkan jawaban mahasiswa menjadi tiga klaster berdasarkan kemiripan logika dan struktur kode, yaitu klaster rendah (40–59), sedang (60–79), dan tinggi (80–100). Sistem ini mampu meningkatkan efisiensi penilaian hingga 65% dibanding metode manual dan memberikan umpan balik secara *real-time*. Keunggulan penelitian ini terletak pada penerapan *K-Means* untuk penilaian berbasis logika dan struktur kode, bukan sekadar kesamaan teks, sehingga mampu menangani variasi jawaban kode yang menghasilkan *output* benar dengan cara berbeda. Pendekatan ini memberikan solusi yang lebih tepat dan adil dalam menilai soal pemrograman dibandingkan metode penilaian berbasis teks seperti *cosine similarity* [14].

5 Kesimpulan

Penelitian ini berhasil merancang sistem penilaian otomatis berbasis kecerdasan buatan untuk soal pemrograman yang terbukti mampu meningkatkan efisiensi, objektivitas, dan konsistensi penilaian. Dengan memanfaatkan algoritma *K-Means clustering*, sistem mampu mengelompokkan jawaban berdasarkan logika dan struktur kode, bukan sekadar kesamaan teks. Pengujian menunjukkan efisiensi penilaian meningkat hingga 65% dengan rata-rata waktu 4,5 detik per soal dan standar deviasi nilai antar klaster di bawah 8 poin, mencerminkan konsistensi klasifikasi. Hasil ini mendukung efektivitas *K-Means* dalam pengelompokan performa siswa, sebagaimana ditunjukkan pada penelitian di SMP Negeri 4 Mauliru [15]. Dari sisi teoritis, penelitian ini memperluas penerapan *clustering* dalam evaluasi berbasis logika untuk pendidikan pemrograman sementara secara praktis, sistem dapat membantu pengajar menghemat waktu dan meminimalkan subjektivitas, khususnya pada kelas dengan banyak peserta dan variasi kode tinggi. Adapun keterbatasan mencakup ruang lingkup soal dari tingkat rendah hingga sulit dan belum diuji pada data lintas institusi. Untuk pengembangan selanjutnya, disarankan agar sistem dilengkapi dengan modul analisis sintaks berbasis model pembelajaran mesin, seperti *Tree-based LSTM* atau *CodeBERT*, guna menangani variasi struktur kode yang lebih kompleks secara kontekstual. Selain itu, pengujian sistem sebaiknya diperluas melalui validasi lintas institusi dan integrasi dengan platform pembelajaran daring, sehingga memungkinkan penerapan sistem secara lebih luas dan adaptif. Pendekatan validasi otomatis seperti yang diterapkan dalam evaluasi kualitas data *intelijen siber* [16] juga dapat diadaptasi untuk memperkuat mekanisme evaluasi internal sistem dan meningkatkan reliabilitas penilaian otomatis.

 ISSN:2302-8149
e-ISSN:2540-9719

Referensi

[1] K. Harefa and A. Jabbar, "Aplikasi Sistem *Automated Essay Scoring* untuk Jawaban Soal Ujian dengan Menerapkan *Algoritma Jaro Winkler*," Oct. 2022.

[2] L. R. Cipto and P. Irfan, "Aplikasi Ujian Online dengan Penilaian Otomatis menggunakan *Algoritma Cosine Similarity* pada SMAN 7 Mataram," *Jurnal BITe*, Vol. 2, No. 1, pp. 57–65, Jun. 2020, doi: 10.30812/bite.v2i1.810.

[3] Alfirna Rizqi Lahitani, "*Automated Essay Scoring* menggunakan *Cosine Similarity* pada Penilaian Esai Multi Soal," May 2022. [Online]. Available: http://ejurnal.ubharajaya.ac.id/index.php/JKI

[4] M. N. I. Susanti, A. Ramadhan, and H. L. H. S. Warnars, "*Automatic Essay Exam Scoring System: A Systematic Literature Review*," in *Procedia Computer Science*, Elsevier B.V., 2023, pp. 531–538. doi: 10.1016/j.procs.2022.12.166.

[5] Musdalipah, R. Soekarta, and I. Amri, "*Clustering* Penilaian Kinerja Dosen menggunakan *K-Means* di Universitas Muhammadiyah Sorong," *Framework*, Vol. 01, No. 02, pp. 136–145, 2023.

[6] W. Satria and M. Riasetiawan, "*Essay Answer Classification with Smote Random Forest and AdaBoost in Automated Essay Scoring*," *IJCCS (Indonesian Journal of Computing and Cybernetics Systems)*, Vol. 17, No. 4, pp. 359–370, Oct. 2023, doi: 10.22146/ijccs.82548.

[7] M. O. Farooqui, M. I. Siddiquei, and S. Kathpal, "*Framing Assessment Questions in the Age of Artificial Intelligence: Evidence from ChatGPT 3.5*," *Emerging Science Journal*, Vol. 8, No. 3, pp. 948–956, Jun. 2024, doi: 10.28991/ESJ-2024-08-03-09.

[8] H. Alexander, Y. Umaidah, and M. Jajuli, "Implementasi *Clustering* untuk menentukan Efektivitas Nilai Siswa sesudah Pandemi Covid-19 menggunakan *Algoritma K-Means*," Jun. 2023.

[9] R. Anggara, S. Defit, and B. Hendrik, "Implementasi *K-Means Clustering* dalam Analisa Soal Ujian CBT Universitas Baiturrahmah," Apr. 2024.

[10] A. Z. Saputra, N. Suarna, and G. D. Lestari, "Klasterisasi Nilai Ujian Sekolah menggunakan Metode *Algoritma K-Means*," *Jurnal Janitra Informatika dan Sistem Informasi*, Vol. 3, No. 1, pp. 1–9, Apr. 2023, doi: 10.25008/janitra.v3i1.153.

[11] M. ' Andri, A. Cendekia Siregar, and P. Y. Utami, "Sistem Penilaian Ujian Otomatis untuk Soal Esai menggunakan Metode *Vector Space Model*," Dec. 2021.

[12] H. F. Kurniawan, Sukisno, L. Arlianti, and T. Hidayat, "Implementasi Metode *Agile* untuk Rancang Bangun Sistem Penilaian Kinerja Guru," *Jurnal Informatika dan Teknik Elektro Terapan*, Vol. 12, No. 3S1, Oct. 2024, doi: 10.23960/jitet.v12i3S1.5171.

[13] H. Arfandy and I. A. Musdar, "Rancang Bangun Sistem Cerdas Pemberian Nilai Otomatis untuk Ujian Essai menggunakan *Algoritma Cosine Similarity*," Dec. 2020.

[14] F. Safnita, S. Defit, and G. W. Nurcahyo, "Penerapan *Algoritma K-Means* dalam Pengklasteran Hasil Evaluasi Akademik Mahasiswa," Apr. 2024.

[15] S. T. Boku, R. T. Abineno, and A. Aha Pekuwali, "Pengelompokan Performa Siswa dalam Pelajaran Matematika dengan *Algoritma K-means* di SMP Negeri 4 Mauliru," Aug. 2023.

[16] A. Venčkauskas, V. Jusas, and D. Barisas, "*Quality Dimensions for Automatic Assessment of Structured Cyber Threat Intelligence Data*," *Applied Sciences (Switzerland)*, Vol. 15, No. 8, Apr. 2025, doi: 10.3390/app15084327.

[17] Bato, B., Pomperada. J. R, "*Automated Grading System with Student Performance Analytics*," *Technium*, Vol. 30, pp. 58-75, 2025.

[18] Sokač, M., Fabijanić, M., Mekterović, I., Mršić, L., "*Automated Grading Through Contrastive Learning: A Gradient Analysis and Feature Ablation Approach*," *Machine Learning and Knowledge Extraction*, Vol. 7, pp. 41, 2025, doi: 10.3390/make7020041.