Tuijin Jishu/Journal of Propulsion Technology
ISSN: 1001-4055
Vol. 46 No. 2 (2025)

GEN - AI Autograding for Academic Evaluation

Siriwardhana S.M.R.R, Kulathunga R.G.G.P, Rashen W.G.M, Jayasinghe J.A.P.M, Prof. Nuwan Kodagoda, Dr. Kalpani Manathunga

Department of Computer Science and Software Engineering, Sri Lanka Institute of Information Technology, Malabe, Sri Lanka

Abstract: Academic institutions face challenges in providing rapid, consistent, and accurate grading across various assignments. This paper introduces a multi-faceted GenAI-based automatic grading system incorporating four microservices: Programming Assignments, Diagram-based Assignments, Technical Subject-related Essays, and English Essays. Developed using the MERN stack with a TypeScript front-end, the system integrates large language models (LLMs) for automatic, discipline-specific evaluation. Each microservice individually assesses criteria such as code correctness, diagrammatic accuracy, technical precision, and linguistic depth, providing numerical grades and feedback. Programming assignments utilize collaborative LLMs for Java code evaluation, while the diagram assessment employs Generative AI, zero-shot classification (DeBERTa-v3-large), vision-language modeling (Qwen2.5-VL-7B-Instruct), and SpaCy-based semantic analysis. Essay evaluations leverage NLP techniques and GPT-3.5 models for detailed grammar, vocabulary, and coherence analysis. Pilot tests demonstrated significant improvements in grading speed, consistency, and reduced bias, enabling scalable, context-aware grading through aggregated dashboard results.

Keywords*: Automatic grading, Educational technology, GenAI large language models, Microservices*

1. Introduction

Educational institutions increasingly face pressures to provide timely, consistent, and unbiased assessments across diverse assignment types, such as coding, diagrams, and essays. Manual grading remains time-intensive, subjective, and prone to inconsistency [1]. Advances in Generative AI (GenAI) and Large Language Models (LLMs) [2] have opened new possibilities for automating and standardizing evaluations, significantly reducing instructor workload and inconsistencies.

This research proposes a multi-microservice GenAI-based grading platform built on the MERN stack with a TypeScript front-end, providing integrated evaluations and enabling teachers to review AI-generated grades before publication. Four specialized microservices employ advanced language models.

Manual grading of programming tasks often suffers from inconsistency and inefficiency, particularly at scale. Existing test case-based systems struggle with incomplete or incorrect programs [8]. Recent AI advancements offer improved assessments by analyzing code structure and logic, yet lack adaptability across different educational contexts [10], [11]. Our system utilizes CodeBERT, CodeT5, and GPT-based models to evaluate Java programs against dynamic rubrics, handling both complete and partial submissions while allowing manual review.

This microservice automatically grades UML diagrams (Class, Use Case, ER, Sequence) through Generative AI and NLP. Employing DeBERTa-v3-large for diagram identification, Qwen2.5-VL-7B-Instruct for interpretation, and SpaCy for textual analysis, the system provides structured, rubric-based feedback efficiently, improving accuracy and reducing manual grading efforts.

Grading technical essays demands subject-specific knowledge and involves subjective interpretation [3]. Our module uses a fine-tuned GPT-3.5-turbo model trained on domain-specific data from the Sri Lanka Institute of Information Technology, significantly enhancing accuracy. This quantitative evaluation is complemented by a

baseline GPT-3.5-turbo model generating detailed qualitative feedback, thus providing comprehensive, transparent assessments and actionable student feedback [4], [5].

The English Essays microservice evaluates student essays for grammar, vocabulary, coherence, and creativity using advanced transformer-based NLP methods beyond traditional rule-based systems [7]. It dynamically identifies linguistic patterns and offers real-time, detailed corrections and contextually relevant feedback, scalable within the integrated multi-microservice platform.

2. Literature Review

Recent demand for automated marking software has increased due to growing class sizes and the need for timely student feedback. Traditional grading methods based on rule-based or keyword-matching approaches struggle with open-ended tasks like essays and diagrams [6]. Recent advancements in Generative AI (GenAI) and Large Language Models (LLMs) offer new capabilities for context-aware assessment, enabling comprehensive evaluation of conceptual correctness and expressive precision [2].

Automated grading for programming assignments initially relied on test-case-based systems, limited in assessing incomplete or non-executable programs and requiring manual test-case preparation [8],[12]. Efforts such as static and dynamic analysis with JUnit [13], and statistical comparison methods [9], improved grading efficiency but did not adequately address conceptual correctness or incomplete submissions. Machine learning approaches using Ridge Regression, SVM, Random Forests [11], and transformer-based models like BERT and CodeBERT [10] enhanced grading accuracy but still lacked flexibility and adaptability to dynamic rubrics. To address these issues, the proposed GenAI-based autograder utilizes CodeBERT, CodeT5, and GPT-based models, supporting dynamic rubrics and incomplete code grading.

Assessment of UML diagrams, crucial in software engineering education, faces challenges with manual grading inefficiencies and inconsistencies. Previous studies employed Java-based structured comparisons [7], explored diagram similarity measures [8], and leveraged image-processing techniques (ReSECDI model) [9]. Graph-based methods [12],[13] and NLP-based diagram generation and evaluation (Smart UML system) [14] were also investigated. However, these approaches largely focus on single diagram types, rely on predefined solutions, and lack Generative AI integration, limiting their scalability, adaptability, and comprehensive feedback capabilities.

Automated essay grading historically utilized surface-level linguistic analysis, as seen in PEG and e-rater systems, which struggled with contextually nuanced tasks [15]. Subsequent developments, including statistical models like Intelligent Essay Assessor (IEA) and LSA-based approaches, improved semantic detection but not coherence or reasoning [3]. Recent advances using Transformer-based models like BERT and GPT demonstrated significant improvements in contextual understanding and rubric alignment [6],[7]. Fine-tuned transformer models proved effective for specialized domains like technical essays, enabling accurate assessment of coherence, argumentation, and rhetorical effectiveness [8],[19].

Grammar correction remains challenging for automated systems, which frequently generate false positives or misunderstand context [9]. Traditional AES methods fail to capture domain-specific terminologies critical for technical essays [16]. Recent NLP advances facilitated fine-tuned GPT models and prompt engineering techniques, significantly enhancing domain-specific assessment precision [19],[20]. Despite these developments, current systems inadequately distinguish general writing skills from analytical content demands. The presented research fills this gap by implementing finely-tuned GPT-3.5 models with sophisticated prompt engineering to enhance grading accuracy, consistency, and feedback quality.

Modular architectures using microservices, essential for scalable and adaptable educational technologies [21],[22], remain underexplored in automated grading solutions. Existing systems typically handle assignment types separately, lacking integrated solutions for coding, diagrams, and various essay formats. This research introduces an integrated GenAI-based microservice framework, accommodating diverse assignment types within a unified, scalable system that optimizes efficiency, accuracy, and instructor workflow management.

3. Methodology

This research proposes a GEN-AI-based automatic grading system developed as a web application, structured using a microservice architecture. Each type of assignment is managed through a dedicated microservice, ensuring modularity and scalability. The frontend was built with React and TypeScript, while backend services were developed in Python using Flask and FastAPI. MongoDB was chosen as the database for its flexibility in handling diverse data formats, and RESTful API endpoints facilitated communication among components [23].

The technical subject essay auto-grading component evaluates student-written responses by comparing them with expected answers and assigning scores based on a defined marking guide. Two GPT-3.5 Turbo models are used: a fine-tuned version (ID: ft:gpt-3.5-turbo-0125:personal:genaiautograderv1:AwStpEZl) for score generation and a default GPT-3.5 model for qualitative feedback. Training data was sourced from 527 student submissions related to 21 essay questions from the "Secure Software Development – SE4030" course at the Sri Lanka Institute of Information Technology. Data was structured into JSONL format as prompt–completion pairs, including question context, expected answers, and lecturer-assigned marks. Fine-tuning used approximately 12.75 million tokens, applying a learning rate multiplier of 0.1, a batch size of 1, and four epochs. Cross-entropy loss with the Adam optimizer ensured the model learned to return responses in strict JSON format. The model achieved a convergence loss of 0.0047, indicating strong optimization [24].

During inference, a two-stage process is applied. First, a detailed prompt containing the question, expected answer, grading instructions, and the student's essay is fed into the fine-tuned model, which returns the allocated marks in JSON format. Second, a prompt is sent to the standard GPT-3.5 Turbo model to produce detailed qualitative feedback, enhancing transparency and aiding student learning.

The programming assignment auto-grading module utilizes a multifaceted evaluation strategy, incorporating syntax analysis, functional correctness, code similarity, and qualitative aspects using machine learning and large language models (LLMs). Syntax checking is handled through static parsing (Javalang) and compilation (javac), with 0.5-point deductions for each error. Several LLMs—including CodeBERT, RoBERTa, DistilBERT, and XLNet—were benchmarked for syntax evaluation, with DistilBERT and CodeBERT delivering the best results. Functional correctness is assessed by executing student code against predefined test cases and calculating output match percentages. Additionally, structural and semantic code similarity is analyzed using TF-IDF and cosine similarity for text, while CodeBERT is employed for semantic understanding. Similarity scores contribute 10% to the final grade. For qualitative analysis, DeepSeek and CodeT5 evaluate readability, efficiency, and adherence to coding best practices, assigning scores based on defined rubrics. CodeT5 outperformed other models in processing Java, Python, and JavaScript submissions. The final programming grade is computed as a weighted aggregate of all evaluation components.

Diagram-based assignment auto-grading involves a comprehensive pipeline comprising a Question Management Module, an Answer Processing Module, a Transformation Engine, and a Grading Engine. Instructors create questions via an interface, and the system uses DeBERTa-v3-large for zero-shot classification to determine the diagram type—class, use case, ER, or sequence. The Transformation Engine uses Qwen2.5-VL-7B-Instruct, a vision-language model (VLM), to convert diagram images into structured JSON based on type-specific prompts. Students submit diagrams which are processed similarly to generate comparable JSON structures. The Grading Engine compares student and reference diagrams using semantic similarity metrics via SpaCy, with a threshold of 0.9. Element comparisons and rubric-based evaluations are then applied using dynamically generated templates. The calculate_marks() function assigns scores based on correctness percentages, while calculate_total() computes the final grade. Special logic handles edge cases such as partial matches and naming inconsistencies to ensure fairness and reliability.

The English essay auto-grading module operates as a standalone FastAPI microservice, utilizing WebSockets for real-time evaluation. It consists of grammar correction, vocabulary analysis, and creativity scoring. Grammar is checked using a refined transformer model trained to identify deep syntactic and contextual errors beyond simple spelling mistakes [25]. Vocabulary analysis involves calculating lexical diversity, word rarity using WordNet and

TF-IDF, and evaluating vocabulary sophistication [27]. Creativity scoring assesses sentence variety, figurative language, and sentiment shifts to measure expressiveness [28]. Essays are tokenized using NLTK and processed sentence-by-sentence, with feedback and scores streamed to the user in real time. Results include corrected essays, grammar and vocabulary scores, creativity evaluations, and overall grading summaries. Performance benchmarks show a grading accuracy of 85% compared to human evaluators and response latency under one second per essay [26].

4. Testing and Evaluation

The GENAI-Based Autograder underwent comprehensive testing and evaluation to ensure its reliability, efficiency, and user acceptance. The system was rigorously tested through multiple layers: unit testing for individual module functionality, integration testing to confirm seamless interactions among microservices, and performance testing to measure grading efficiency on large-scale datasets. Its performance was evaluated using diverse metrics such as grading accuracy, execution time, syntax error detection precision, and model-specific assessments. To validate accuracy, AI-generated scores were compared with human-assigned grades across a dataset of 500 assignments using Pearson correlation, revealing a strong alignment. Execution time was benchmarked against manual grading processes, highlighting the system's potential for significantly reducing assessment turnaround. The precision, recall, and F1-score of syntax checking components were also computed to assess error detection reliability. In the domain of code similarity, models such as TF-IDF and CodeBERT were evaluated based on the same metrics to determine their effectiveness in identifying structural and semantic equivalence in programming submissions.

Comparative benchmarking against traditional autograders and manual grading methods showed the GENAI-based system offered a superior approach by not only evaluating output correctness but also assessing algorithmic complexity, code structure, and conceptual clarity—features often overlooked in conventional autograders. A user study was conducted involving 10 instructors and 50 students. Results indicated that 85% of instructors felt AI grading aligned well with human assessments, while 78% of students reported that the generated feedback helped them improve their coding practices. This high satisfaction rate reinforced the system's utility in academic environments and its capacity to complement or potentially replace manual grading under appropriate supervision.

Specific testing of the Technical Essay Grading Module involved feeding a subset of previously graded student responses into the full grading pipeline. The fine-tuned GPT-3.5 Turbo model generated numeric marks in JSON format, while the default GPT-3.5 Turbo produced qualitative feedback. The key evaluation metric, Mean Absolute Error (MAE), was calculated to compare auto-generated grades with lecturer-assigned marks. Results demonstrated extremely low MAE—0.0 for most entries and 0.5 for a few borderline cases—indicating near-perfect grading accuracy. Grade-matching accuracy, which measures the percentage of perfect matches between system-generated and human-assigned marks, reached 85%. Additionally, the latency of the two-stage inference pipeline was monitored to ensure the system remained within acceptable bounds for real-time or near-real-time classroom use. Subject-matter experts reviewed the qualitative feedback generated, confirming that it was coherent, contextually relevant, and helpful for understanding the rationale behind the assigned scores. A sample comparison (Table 1) supported these findings, illustrating minimal deviations between automated and human grading results.

Table 1: Comparison of auto-graded marks with lecturer-assigned marks

Question ID	Allocated Marks	Lecturer Marks	Auto Graded Marks	MAE
Q1	3	3	3	0
Q2	2	2	2	0
Q3	4	4	3.5	0.5

The English Essay Grading Module was similarly tested using a separate dataset of lecturer-assigned marks. Essays were passed through the two-stage pipeline, where GPT-3.5 Turbo first generated a numerical score and then provided a written explanation. The MAE for this module averaged 0.2, with a maximum deviation of 0.6 in only a few cases. Grade-matching accuracy reached 82%, with most discrepancies falling within a ±1 mark margin, demonstrating consistency with human evaluation standards. Inference time was also evaluated, with an average latency of under 2.5 seconds per essay, confirming the model's suitability for real-time applications. The feedback quality was reviewed by academic experts, who found it logically structured and informative. This affirmed the model's capacity to enhance both student learning and instructor oversight.

In summary, the GENAI-Based Autograder demonstrated high grading accuracy, fast response times, and strong alignment with human judgment across multiple assignment types. The combination of low error margins, high exact-match percentages, and positive user feedback underscores the system's effectiveness and readiness for deployment in academic environments. These results validate the technical design, training strategy, and inference framework while also identifying areas for improvement, such as refining prompts and expanding training data for better generalization in edge cases [25][26][27][28].

5. Conclusion

This research introduces a robust, multi-microservice GenAI-based grading platform designed to automate the assessment of programming tasks, diagram-based assignments, technical essays, and English essays. By leveraging advanced language models such as GPT-3.5, CodeBERT, CodeT5, DeBERTa-v3-large, and Qwen2.5-VL-7B-Instruct, the system achieves high accuracy in grading while offering rich, context-aware feedback. Implemented using the MERN stack with a TypeScript frontend and a modular microservice backend, the system ensures scalability, flexibility, and seamless integration into academic workflows. Rigorous testing and validation—through unit tests, integration tests, and performance benchmarks—demonstrated high correlation with human grading, low mean absolute error (MAE), and real-time feedback capabilities. User studies further affirmed the system's alignment with instructor expectations and its effectiveness in improving student learning outcomes. The modular design and use of advanced NLP and vision-language techniques mark a significant step toward scalable, unbiased, and adaptive academic evaluation. Future enhancements will focus on expanding dataset diversity, refining prompt engineering, and integrating domain-specific feedback generation to further increase grading precision and adaptability across broader educational contexts.

References

[1] S. P. Jones, "Challenges of Large-Class Grading in Higher Education," IEEE Transactions on Education, vol. 64, no. 2, pp. 123-131, 2021.
[2] O. Team, "GPT-3.5: The Next Generation in Natural Language Processing," OpenAI, 2023.
[3] M. C. C. L. J. Burstein, "CriterionSM: Online Essay Evaluation," in Workshop on Innovative Use of NLP for Building Educational Applications, 2003.
[4] A. S. J. Doe, " Challenges in Manual Assessment: Time and Consistency Issues in Higher Education," IEEE Transactions on Education, vol. 64, no. 3, pp. 123-130, 2020.
[5] J. B. Y. Attali, "Automated Essay Scoring with e-rater® V.2," Journal of Technology, Learning, and Assessment, vol. 4, no. 3, 2006.
[6] J. C. a. J. English, "Automated grading of programming assignments using tool-focused evaluation," ACM SIGCSE Bulletin, vol. 31, no. 3, pp. 76-80, 1999.
[7] S. Modi, H. A. T. and H. Mahmud, "“A Tool to Automate Student UML diagram Evaluation," Academic Journal of Nawroz University vol. 10, p. 189–198, Jun. 2021.
[8] D. S. S. R. E. T. R. Fauzan, "Automated Class Diagram Assessment using Semantic and Structural Similarities," International Journal of Intelligent Engineering and Systems, vol. 14, p. 52–66, Apr. 2021.
[9] L. Z. X. L. N. N. F. Chen, "Automatically recognizing the semantic elements from UML class diagram images," Journal of Systems and Software, vol. 193, Nov. 2022.
[10] N. S. K. G. W. P. G. Thomas, "Computer assisted assessment of diagrams," in ACM SIGCSE Bulletin, New York, Jun. 2007.

[11] K. P. K. B. Narasimha Bolloju, "FORMATIVE AND SUMMATIVE ASSESSMENT OF CLASS DIAGRAMS - Development and Evaluation of a Prototype," Proceedings of the 3rd International Conference on Computer Supported Education, vol. vol. 2, pp. 402-410, Jan. 2011.
[12] T. M. L. A. O. Anas, "New method for summative evaluation of UML class diagrams based on graph similarities," International Journal of Electrical and Computer Engineering (IJECE), vol. vol. 11, p. 1578 1578, Apr. 2021.
[13] K. T. H. J. P. O. N. U. S. R. A. K. D. N. H. Weerasinghe, "Smart UML - Assignment Management Tool for UML Diagrams," in International Conference on Advancements in Computing (ICAC), Colombo, Dec. 2022.
[14] A. L. M. T. A. Outair, "Towards an Automatic Evaluation of UML Class Diagrams by Graph Transformation," International Journal of Computer Applications, vol. vol. 95, pp. 36-41, Jun. 2014.
[15] J. B. J. Attali, "Automated Essay Scoring with e-rater® V.2," ETS Research Report Series, 2006.
[16] J. B. D. Shermis, Handbook of Automated Essay Evaluation, Routledge, 2013.
[17] M.-W. C. K. L. K. T. J. Devlin, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," in NAACL-HLT, 2019.
[18] T. B. e. al., "Language Models are Few-Shot Learners," in NeurIPS, 2020.
[19] OpenAI. [Online]. Available: https://platform.openai.com/docs/guides/fine-tuning.
[20] D. B. e. al., "On the Opportunities and Risks of Foundation Models," arXiv:2108.07258, 2021. arXiv preprint, vol.
[21] M. Richards, "Scaling education technology with microservices," IEEE Software, vol. 32, no. 3, pp. 2129, 2015.
[22] J. D. e. al., "Microservices: yesterday, today, and tomorrow," in Springer LNCS (Lecture Notes in Computer Science), vol. 10502, 2017, pp. 195-208.
[23] Aggarwal, S. Srikant and V., "A system to grade computer programming skills using machine learning," in Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2014.
[24] OpenAI, 2023. [Online]. Available: https://platform.openai.com/docs/models/gpt-3-5 turbo.
[25] A. A. e. al., "Automated Essay Scoring Using Natural Language Processing Techniques," IEEE Transactions on Education, pp. vol. 63, no. 2, pp. 123–132, 2020., 2020.
[26] B. A. a. C. Author, "Challenges in Evaluating Technical Essays: A Survey,” in Proc. IEEE Int. Conf. Educ. Technol., 2019," IEEE Int. Conf. Educ. Technol, 2019.
[27] E. W. a. D. Lee, "Domain-Specific Fine-Tuning of Language Models for Automated Essay Scoring," in Educational Data Mining (EDM), 2022.
[28] M. S. a. B. Hamner, "Contrasting State-of-the-Art Automated Scoring of Essay," in 28th Annual Meeting of the Modern Language Journal, 2012.