Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

Leveraging Large Language Models for Assessments Grading and Personalised Feedback Generation: A Case of Software Engineering

Mamoona Tasadduq

mamoona.tasadduq@gmail.com

Information Technology University https://orcid.org/0000-0002-6987-9581

Fakhra Amjad

COMSATS University Islamabad

Javed Ali Khan

University of Hertfordshire

Hafsa Shareef Dar

University of Gujrat

Research Article

Keywords: Automated grading, Formative feedback, Large Language Models, Software engineering education, Software artefacts

Posted Date: August 25th, 2026

DOI: https://doi.org/10.21203/rs.3.rs-10804567/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: The authors declare no competing interests.

Leveraging Large Language Models for Assessments Grading and Personalised Feedback Generation: A Case of Software Engineering

Fakhra Amjad, Mamoona Tasadduq, Javed Ali Khan, Hafsa Shareef Dar

Received: date / Accepted: date

Abstract

The emergence of Artificial Intelligence and Large Language Models (LLMs) is transforming the education sector. Researchers have recently started exploring LLMs for evaluating educational assessments and generating personalised feedback; however, their evaluation and effectiveness in evaluating complex Software Engineering artefacts remain underexplored. In the proposed approach, we used LLMs, including ChatGPT-4 and Gemini 2.5, to identify rubric-based grading of students' assessments and to generate personalised feedback using integrated zero-shot and chain-of-thought prompting. For this purpose, 54 students' assessments from the 'Software Design and Architecture' course were evaluated, including UML-based artefacts such as use case diagrams, class diagrams, and three-tier architecture. Each assignment was evaluated using an instructor-defined rubric, and two different LLMs generated scores. The LLMs' generated scores were compared with the instructor-defined grades using Mean Absolute Error (MAE), Root Mean Square Error (RMSE), Pearson correlation, Spearman correlation, error tolerance analysis, and paired-samples (t)-tests. Student perceptions of AI-generated feedback were also examined through a questionnaire-based survey. The results show that both LLMs generated structured rubric-based evaluations and meaningful feedback. Gemini 2.5 achieved a slightly lower MAE than GPT-4, while GPT did not show a significant mean-level difference from instructor scores. GPT-4 was also highly preferred by the students for feedback quality, personalization, and similarity to the instructor's feedback. However, both models demonstrated weak correlation and score compression with instructor grading. These findings suggest that LLMs can provide effective feedback and preliminary grading, but should not replace human instructors in high-stakes assessments. A hybrid assessment model where LLMs assist instructors, while the final authority remains with the instructor, is therefore recommended. The proposed approach is supported by a web-based tool.

Keywords: Automated grading · Formative feedback · Large Language Models · Software engineering education · Software artefacts

1 Introduction

Over the past decade, information technology has rapidly evolved, affecting various domains, including education (Alier Forment et al. 2024), and has led to the emergence of smart education to improve learning and teaching experiences (Badshah et al. 2023). More recently, with the development of Large Language Models (LLMs) that have revolutionised

various domains, including natural language processing (NLP) (Min et al. 2023) and software engineering (Hou et al. 2024; J. A. Khan, Qayyum, et al. 2025). Similarly, LLMs have been widely adopted in education and hold the potential to improve the learning experience by automating evaluation, enabling personalised learning, and enhancing learner engagement, while considering associated ethical concerns (Guizani et al. 2025). To support this, researchers have recently developed approaches based on LLM agents to address educational challenges, such as feedback generation (Chu et al. 2025).

To address these challenges, LLMs pose significant potential in assessment evaluation and personalised feedback generation (Yavariabdi et al. 2025). For this purpose, researchers have proposed several approaches for assessment evaluation and generating personalised feedback. For example, Xie et al. (2025) propose an LLM-based grading approach that evaluates learners' assessments using several prompting methods to first refine rubrics, generate scores and personalised feedback, and conduct post-grading review to ensure the accuracy of LLM-generated results. Similarly, Schneider et al. (2023) used ChatGPT 3.5 to evaluate learners' short-text questions by classifying them as Extremely good, Very good, Good, Ok, Bad, Very bad, or Extremely bad. The study also reported several limitations of LLM-generated feedback, including a lack of specificity, potential bias, and generic feedback. J. A. Khan, Yaqoob, et al. (2025) proposed the LLM-MATE approach to evaluate software diagrams and generate personalised feedback using zero-shot learning. Riazi and Rooshenas (Riazi and Rooshenas 2025) proposed an LLM-based approach to support conceptual design learning by converting learners' Entity-Relationship Diagrams (ERDs) into JSON format, identifying relationships, and extracting related requirements to generate detailed feedback. Poličar et al. (2025) propose an LLM-based automated grading and feedback approach for textual assessment, employing several LLMs via few-shot prompting to generate personalised feedback with a satisfactory learner response rate. In the literature, there has been growing interest in LLM-based approaches for assessment evaluation and personalised feedback generation due to their satisfactory results in improving learners' learning and knowledge. Still, based on our knowledge, there are limitations and challenges in existing approaches that warrant consideration by the research community to improve the learner experience and learning. For example, LLM performance needs to be evaluated using multimodal assessments that include diagrams, text, and code samples. Also, an end-to-end assessment, evaluation, and feedback generation tool is required to provide learners with a better learning experience. Moreover, thorough feedback from the learners is required to evaluate the personalised feedback generated by the LLMs and its implication on their learning.

In line with this, we proposed an LLM-based approach for learner assessment evaluation and personalised feedback generation to improve the learning experience. The approach contributes by developing a dataset comprising 54 students' assessments from the 'Software Design and Architecture' course, including UML-based artefacts such as use cases, class diagrams, and three-tier architecture. Also, a combined prompt using zero-shot and chain-of-thought learning is developed to evaluate each assessment using an instructor-defined rubric, and two different LLMs generated scores using a multi-run strategy to overcome hallucination. The LLMs' generated scores were compared with the

instructor-defined grades using Mean Absolute Error (MAE), Root Mean Square Error (RMSE), Pearson correlation, Spearman correlation, error tolerance analysis, and paired-samples (t)-tests. Moreover, the student perceptions of AI-generated feedback were examined through a questionnaire. The results show that both LLMs generated structured rubric-based evaluations and meaningful feedback. Gemini 2.5 achieved a slightly lower MAE than GPT-4; whereas, GPT showed no significant mean-level difference from the instructor scores. GPT-4 was also strongly preferred by the students for feedback quality, personalisation, and similarity to instructor feedback. However, both models demonstrated weak correlation and score compression with instructor grading. These findings suggest that LLMs can provide effective feedback and preliminary grading, but should not replace human instructors in high-stakes assessments. A hybrid assessment model in which LLMs assist instructors while the final authority remains with the instructor is therefore recommended.

To guide the study, the following research questions were formulated:

- RQ1: To what extent do LLMs align with instructor-assigned grades when evaluating assignments in software design and architecture?
- RQ2: Which LLM demonstrates stronger agreement with instructor's grading?
- RQ3: How do students perceive the clarity, usefulness, and learning value of AI-generated feedback?
- RQ4: Which LLM-generated feedback do students prefer for future assessments?

The rest of the paper is organized as follows. Section 2 presents a review of the relevant literature on Large Language Models in software engineering education, automated assessment, and AI-generated feedback. Section 3 describes the research methodology, including the assignment design, grading rubric, LLM evaluation process, and data collection procedures. Section 4 presents the experimental results and discusses the findings, including the agreement between LLM-based and instructor-based grading as well as student perceptions of AI-generated feedback. Section 7 discusses the threats to validity and limitations of the study. Finally, Section 8 concludes the paper, summarizes the key findings, discusses limitations, and outlines directions for future research.

2 Literature Review

This section discusses the existing literature on generative artificial intelligence in general and its use for automated educational assessments in particular, establishing the theoretical foundation for this study. The literature also provides a review on the role of Large Language Models (LLMs) as evaluators in software engineering education, their effectiveness for grading and providing feedback, and the challenges associated with their deployment in higher education.

The recent research on LLMs in higher education has primarily focused on automated code generation, intelligent tutoring, debugging assistance, and personalized learning support (Fan et al. 2023; Frankford et al. 2024). Despite their increasing adoption in educational settings, the use of LLMs for formative assessment and evaluation remains comparatively underexplored. Existing studies have demonstrated the potential of LLMs to provide personalised feedback and adaptive learning environments; however, the questions regarding their consistency, fairness and reliability remain, particularly for descriptive software engineering assessments (Wang et al. 2025; M. Khan et al. 2025).

LLMs for assessment systems introduce both opportunities and challenges. Advanced generative AI models such as ChatGPT-4o, DeepSeek, and Claude have demonstrated the ability to interpret complex classification rubrics, provide detailed feedback, and reduce the administrative burden associated with manually grading assessments (Elvira et al. 2025; Chang and Hsu 2025; Mazzone 2025). Yousef et al. (2025) suggests that fine-tuned LLMs can achieve grading consistency similar to that of experienced human evaluators. However, concerns regarding academic integrity, student over-reliance on generative tools, and the difficulty of distinguishing between student-generated and AI-assisted work remain (M. Khan et al. 2025). Consequently, researchers have emphasised the need for assessment frameworks that explicitly account for varying degrees of AI assistance while preserving the validity of learning outcomes.

LLMs have recently been explored as automated evaluators. Duong and Meng (2024) proposed a framework for automated grading for short-answer questions in a software engineering course using embedding and completion-based approaches built upon GPT models. Their findings demonstrated that completion-based grading, particularly when enhanced through Retrieval Augmented Generation (RAG), achieved strong agreement with instructor-assigned scores while requiring no task-specific model training. Similarly, benchmarking studies have shown that modern reasoning-oriented models can achieve levels of agreement with expert graders that approach human performance under carefully defined rubric conditions (Pecuchova et al. 2025). However, this performance remains highly dependent on prompt design, rubric clarity, domain context, and task characteristics (Wang et al. 2025). Researchers have also highlighted challenges related to bias propagation, inconsistencies in feedback generation, and difficulties in assessing ambiguous or unconventional solutions (Grande et al. 2024; Naman et al. 2025).

In software engineering education, most assessment research has concentrated on textual responses, conceptual questions, essays, or programming assignments. In contrast, little attention has been devoted to evaluating higher-level software engineering artifacts including use case diagrams, class diagrams, software architecture designs, and project documentation. These artifacts require subjective interpretation, design reasoning, and contextual judgment that extend beyond simple correctness-based evaluation. Consequently, the capacity of LLMs to reliably assess such deliverables remains insufficiently understood. Furthermore, although several studies have investigated grading accuracy, relatively few have examined students' perceptions of AI-generated feedback or compared learner satisfaction across different LLM evaluators.

Reihanian et al. (2025) argued that hybrid assessment approaches provide a good balance between scalability and pedagogical rigor, where LLMs perform preliminary grading and feedback generation while instructors retain oversight of complex evaluations. Complementary strategies such as reflective journals, oral examinations, and justification-based assessments have also been recommended to ensure students remain accountable for their work and develop deeper conceptual understanding (Pham et al. 2023). These approaches acknowledge that AI should function as a supportive educational tool rather than a substitute for human judgment and critical thinking.

Although prior research demonstrates the promise of LLMs in educational assessment, empirical evidence regarding their effectiveness in evaluating authentic software engineering design artifacts remains limited. Existing LLM-based grading studies have primarily focused on textual responses (Yeung et al. 2025), essays (Seßler et al. 2025), and short-answer questions (Duong and Meng 2024), and programming-oriented tasks (Yousef et al. 2025). Limited evidence exists regarding the ability of LLMs to evaluate complex software engineering design artifacts such as UML diagrams, use case models, class diagrams, and architectural designs. Furthermore, limited research has examined students' satisfaction with AI-generated feedback from different LLMs. Therefore, this study investigates the performance of multiple contemporary LLMs in grading software engineering assignment artifacts and generating feedback, comparing their evaluations against instructor grading while simultaneously examining student perceptions of the resulting feedback. By doing so, the study aims to provide empirical evidence regarding the suitability of LLM-assisted assessment in software engineering education.

3 Proposed Methodology

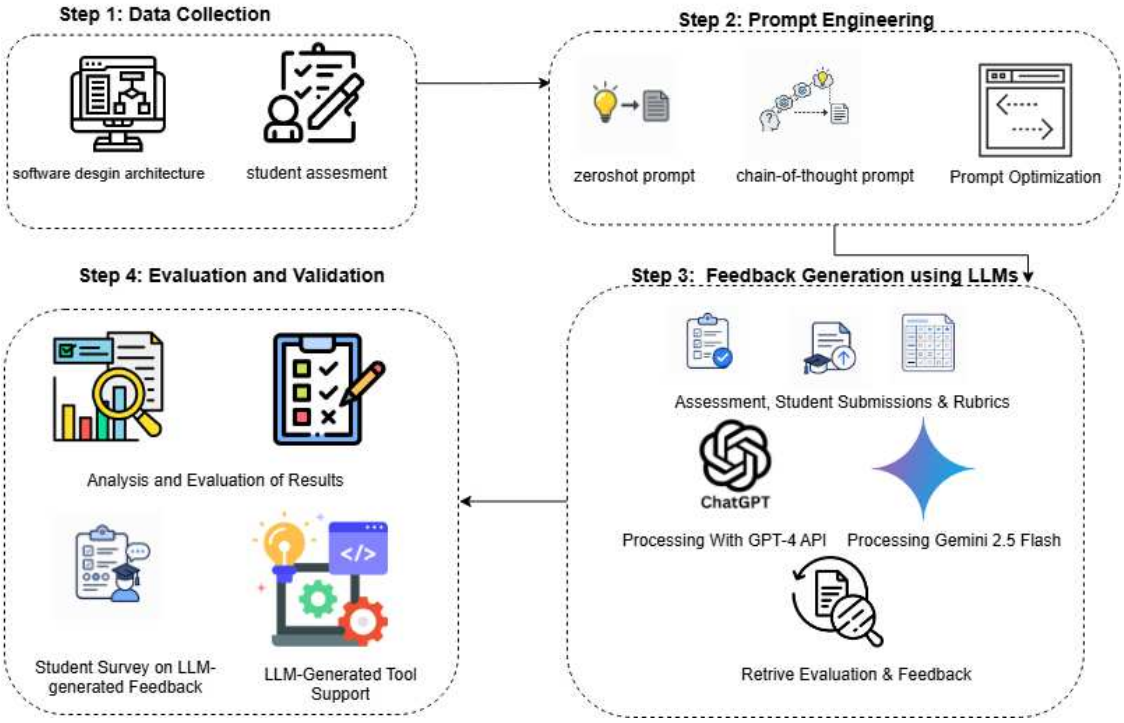
This study proposes a Large Language Model (LLM)-based assessment framework for grading software engineering assignments. This framework utilises LLMs to evaluate student submissions against instructor-defined rubrics and to generate formative, personalised feedback. To evaluate the performance of the proposed approach, we used various analyses and evaluations, including error and correlation-based methods. Furthermore, student perceptions of the quality and usefulness of LLM-generated feedback are examined using a questionnaire-based survey. The questionnaire consisted of 15 items adapted to measure student engagement, confidence, motivation, usefulness, trust, and satisfaction with AI-generated feedback. In addition, three comparative questions were included to determine students' preferred LLM and perceived similarity to instructor feedback. The complete questionnaire is provided in Appendix 9. The overall research approach is illustrated in Figure 1. The approach mainly comprises four steps. Firstly, student assessments are collected and processed with LLMs to grade them and generate personalised feedback. In the second step, various prompting approaches are used to optimise prompts in response to student assessments. In the third step, the assessment scenario, the instructor-defined rubric, and the student submission are used with the optimised prompt to be processed by multimodal LLMs, which generate scores based on defined criteria and qualitative feedback. To improve grading reliability, each

submission is evaluated independently three times, and the resulting scores are averaged. The final grades are subsequently compared against the instructor evaluations and analysed using statistical measures. Moreover, the proposed approach was supported by a web-based application developed using Django to facilitate assignment management, submission handling, and the delivery of LLM-generated assessment grading and personalised feedback to students. However, the primary focus of this study is the assessment methodology rather than the underlying software platform. Below, we elaborate on each methodological step.

3.1 Assessment Dataset

To implement the proposed methodology, a dataset of students' assessments in Software Engineering Concepts (SEC) from the Software Design Architecture (SDA) course, offered by the Department of Software Engineering at the University of Gujarat[1], Pakistan, was collected. Unlike existing approaches, which mainly select programming exercises or short-answer assessments, we selected a complex SDA exercise in which students need to submit a Use Case Diagram, a Class Diagram, and a Three-Tier Architecture Design by analyzing a case study. These artefacts require interpretation of UML notation, an understanding of design and architectural logic, and alignment with software requirements. Hence, the dataset provides a benchmark for evaluating LLMs' ability to assess complex software engineering artefacts. The dataset comprised 54 student submissions of software engineering design artefacts. Each assignment was manually evaluated by the instructor using predefined grading criteria/rubric.

[1] https://uog.edu.pk/



Proposed LLM-based assessment framework for rubric-based grading and personalised feedback generation.

For each submission, the dataset contained:

- Assessment question
- Student submission in PDF format
- Instructor-defined grading rubric
- Instructor-assigned marks
- Instructor-written feedback

Rubric-Based Assessment: To maintain consistency, artefacts were assessed using the same rubric employed by the module instructor. The proposed LLM-based approach comprises three major evaluative categories

- Use Case Model (15 marks)
- Class Diagram (15 marks)
- Three-Tier Architecture Model (20 marks)

The Use Case Model rubric assessed UML notation, scenario alignment, and the appropriate use of `<<include>>` and `<<extend>>` relationships. The Class Diagram rubric evaluated class identification, attributes, methods, associations, and the use of

collections. The Three-Tier Architecture rubric assessed architectural separation, component allocation, dependencies, and adherence to UML standards. Moreover, the rubric was incorporated directly into the evaluation prompts to ensure consistency with instructor evaluations and grading.

3.2 Prompt Engineering

For LLM-based assessment evaluation and personalised feedback generation, well-structured prompts are essential to ensure that responses remain relevant to the given domain and potentially overcome hallucination (J. A. Khan, Yaqoob, et al. 2025). The literature shows that effective feedback generation using LLMs is closely aligned with accurate prompt design (White et al. 2024). In the proposed approach, prompt engineering was employed to encourage consistent, rubric-aligned assessment. A single prompting strategy was adopted that combined elements of zero-shot and chain-of-thought prompting rather than treating them as separate, independently compared conditions. This hybrid prompting approach was adopted to address limitations observed when using either strategy in isolation during preliminary testing. Prompts without explicit reasoning instructions produced holistic, impression-based scores that did not reliably reflect rubric-specific evidence, particularly for design-oriented artifacts such as UML diagrams, where correctness is distributed across multiple interdependent elements. Conversely, prompts that exposed the full reasoning trace as free-form output introduced inconsistency in output structure, complicating the automated parsing and aggregation of scores required for the multi-run evaluation strategy. Requiring internal step-by-step reasoning, including a scenario-validation step prior to any criterion-level scoring, while constraining the final output to a fixed structured format, was intended to preserve the reliability benefits of staged reasoning without sacrificing the consistency needed for automated processing.

The prompt required the model to perform internal step-by-step reasoning, including validating the scenario/domain alignment between the student's submission and the assignment question, and systematically analyzing each rubric criterion for evidence of required elements, consistent with chain-of-thought reasoning. However, rather than exposing this full reasoning trace as free-form output (as is typical in conventional chain-of-thought prompting), the model was instructed to produce only a final, structured evaluation output, consistent with the constrained, task-focused response format typical of zero-shot prompting. This hybrid design was intended to combine the reasoning benefits of chain-of-thought prompting, namely more consistent, evidence-based rubric adherence, with the output consistency and machine-readability benefits of a constrained response format, which was necessary for downstream automated score aggregation across the three evaluation runs described in Section 3.5. Separate prompt implementations were developed for GPT-4 and Gemini 2.5 Flash to accommodate differences in input handling (PDF-based input for GPT-4 versus rendered image input for Gemini 2.5 Flash, as detailed in Section 3.4). While the two prompts were not worded identically, they were designed to be functionally equivalent, enforcing the same scenario-validation step, the same evidence-based rubric adherence rules, the same error-severity classification and penalty

structure, and the same structured output format required for automated score aggregation across both models.

Prompt Optimisation: Prompt refinement was conducted manually rather than via automated prompt optimisation techniques (e.g., DSPy or automated prompt search), primarily due to budgetary and API cost constraints associated with repeated large-scale evaluation runs across two multimodal models. The refinement process was iterative and empirically driven: an initial version of the prompt was run on a subset of submissions, and the resulting scores and feedback were manually compared with instructor-assigned grades to identify systematic discrepancies. The instructor-defined rubric itself remained fixed throughout to preserve rubric-instructor alignment; refinement was instead directed at how the prompt instructed the model to interpret and apply the rubric, and at how its internal reasoning process was structured. Particular attention was paid to submissions that produced outlier scores relative to instructor grades, both overly lenient evaluations, in which the model awarded high marks despite missing or superficial rubric elements, and overly harsh evaluations, in which minor or cosmetic issues resulted in disproportionately low scores. Submissions receiving mid-range scores were also reviewed, as these often revealed subtler inconsistencies in how the model applied the rubric or in how it assigned partial credit. Based on these observations, adjustments were made to the scenario-validation instructions, the evidence requirements for awarding credit, the classification and severity of penalised errors, and the phrasing used to calibrate grading harshness against instructor conventions. Each revised prompt version was re-tested against the same set of outlier and mid-range submissions before being applied to the full dataset, and this cycle of testing, comparison, and adjustment was repeated across multiple iterations for both the GPT-4 and Gemini 2.5 Flash prompt implementations until model-generated scores and feedback demonstrated closer, more consistent agreement with instructor evaluations. An example of an optimised prompt is shown below;

Prompt for student assessment evaluation and personalised feedback generation You are a deterministic grading system evaluating a student's Software Engineering submission against an instructor-defined rubric. Reason through the evaluation step by step internally, but output only the final structured evaluation. ## Instructions: Step 1 — Scenario validation: Before scoring, confirm that the submission addresses the same system/domain specified in the assignment question. If the domain does not match, assign a score of zero across all criteria and stop; otherwise proceed. Step 2 — Criterion-by-criterion reasoning: For each rubric criterion, identify all required elements and check the submission for explicit evidence of each, reasoning through completeness and correctness before assigning any score. Base all judgments strictly on evidence visible in the submission; do not infer or assume the presence of unstated elements. Classify identified errors by severity (e.g., severe, critical, major, minor) using a fixed penalty scale, and apply proportional deductions accordingly. Step 3 — Score calculation: Combine completeness and error-severity penalties to compute a final score for each criterion, rounded to the nearest half-mark increment, applying consistent and realistic grading severity rather than lenient or impression-based scoring. ## Output Response Format: Do not include the step-by-step reasoning in the response. Provide only a structured

evaluation for each criterion (strengths, weaknesses, final score), followed by a machine-readable summary of per-criterion and total scores in JSON format for automated aggregation. Evaluate the following submission: Assignment question: question_text Student submission: student_answer Grading rubric: rubric

3.3 Assessments Evaluation and Feedback Generation

Model Selection: For improved student assessment evaluation and personalised feedback generation, we shortlisted two state-of-the-art multimodal Large Language Models to investigate their performance, i.e., GPT-4 (OpenAI) and Gemini 2.5 Flash (Google). These models were selected because of their strong reasoning capabilities and support for multimodal inputs. These models are capable of analyzing textual content and visual representations, making them suitable for evaluating UML-based software engineering assignments. To ensure fairness, both models were evaluated using identical assignment materials and grading rubrics. Furthermore, inference temperature was fixed at zero to minimize stochastic variation and encourage deterministic outputs.

Input Preparation: To evaluate student assessments and generate personalised feedback, each evaluation required three inputs, i.e., assessment question, instructor grading rubrics, and student submission. The evaluation prompt was constructed by incorporating relevant textual data from the three input files. Student submissions were provided in PDF format containing both textual explanations and UML diagrams. Moreover, two different approaches were used according to the capabilities of each model. GPT-4 received the original PDF submission directly through the OpenAI API. Whereas PDF pages were rendered as images and supplied to Gemini 2.5 Flash. This approach preserved the visual information contained within UML diagrams and architectural representations while enabling multimodal reasoning.

Assessment Evaluation and Feedback Generation: To evaluate assessments and generate personalised feedback, the assessment question, instructor grading rubrics, and student submission are processed with both Gemini 2.5 Flash and GPT-4 using an optimised prompt developed in Section 3.2. The details of the assessment, evaluation, and feedback generation are elaborated in Section 4.

3.4 Evaluation and Validation

In this section of the proposed approach, the LLM-based assessment and feedback generation is evaluated, validated and analysed.

Evaluation: Instructor-assigned grades served as the reference benchmark for evaluating the performance of the proposed framework.

The grading performance of each model was assessed using Mean Absolute Error (MAE), Pearson correlation coefficient (r), and Spearman rank correlation coefficient (ρ). MAE measures the average deviation between LLM-generated scores and instructor-assigned scores, while Pearson and Spearman correlations evaluate linear agreement and ranking

consistency, respectively.
The Mean Absolute Error (MAE) was calculated as:

$$MAE = \frac{1}{n}\sum_{i=1}^{n} |y_i - \hat{y}_i|$$

where y_i represents the instructor-assigned score and $\hat{y}_i$ represents the score generated by the LLM.

Student Perception Survey (Validation): To investigate student perceptions of AI-generated feedback, a questionnaire-based survey was conducted after completion of the assessment process. The questionnaire was adapted from the Intrinsic Motivation Inventory (IMI) (McAuley et al. 1989) and extended with additional items related to AI-generated feedback quality, trust, and LLM preference. Students reviewed the feedback generated by GPT-4 and Gemini 2.5 Flash and subsequently completed a questionnaire using a seven-point Likert scale ranging from Not at all true (1) to Very true (7).

The questionnaire evaluated several dimensions, including feedback clarity, usefulness, personalization, trustworthiness, motivation, learning support, and overall satisfaction. Students were also asked to indicate their preferred LLM and identify the LLM generated feedback closely to that of the instructor.

Data Analysis: The data was analyzed statistically to gauge the effectiveness of the LLMs generated feedback. The scores from GPT-4 and Gemini 2.5 Flash were compared against the instructor's grades using error rates (MAE) and correlation tests. For the survey data, frequencies, percentages, mean scores, and standard deviations were calculated for each questionnaire item.

The combined analysis enabled a comprehensive overview of both - the effectiveness of LLM-based grading and the AI-generated feedback in software engineering education.

4 Results and Discussion

This section presents the LLM-based assessment evaluation and feedback generation process and results of the proposed framework. The analysis is divided into two parts. First, the grading performance of GPT-4 and Gemini 2.5 Flash is compared with instructor-assigned marks using error-based and correlation-based metrics. Second, student perceptions of AI-generated feedback are analyzed using questionnaire responses.

4.1 Systematic Evaluation of Student Assessments and Personalised Feedback Generation

In this subsection, we elaborate on the preprocessing, assessment evaluation, and personalised feedback generation process using Gemini 2.5 Flash and ChatGPT-4.

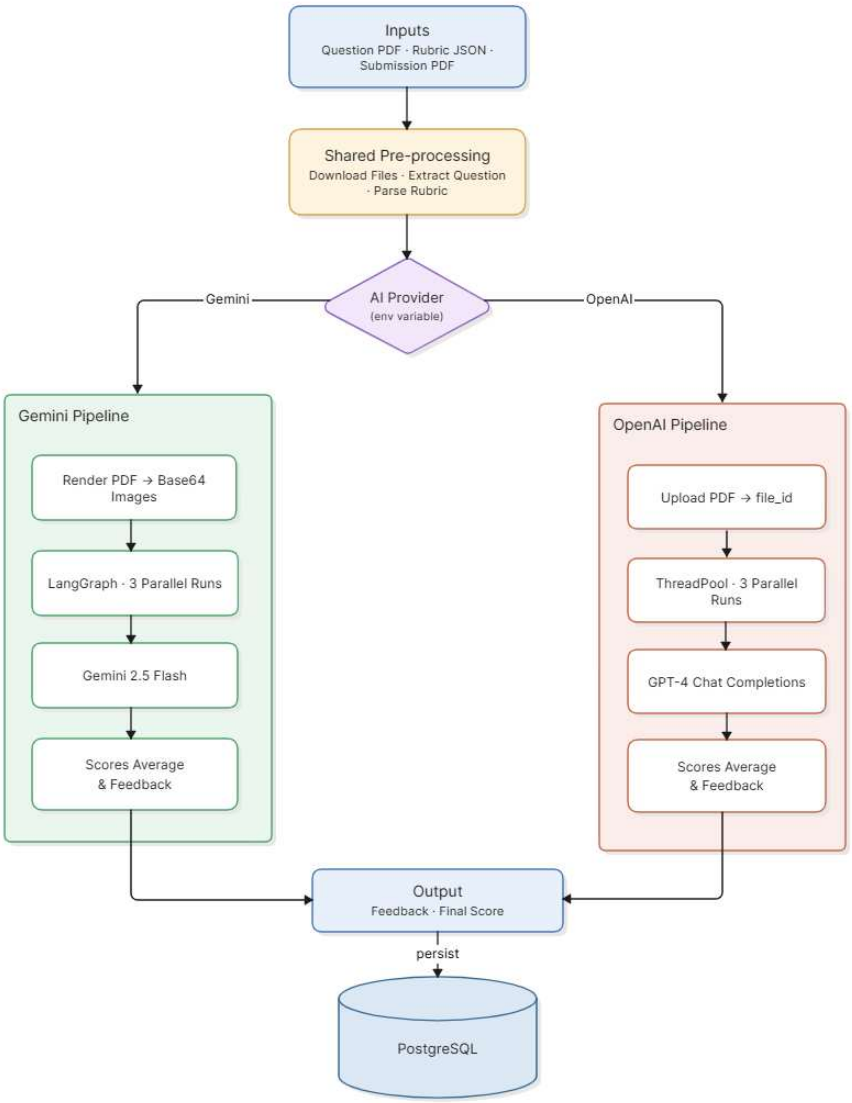
Shared Pre-Processing Pipeline: Prior to model-specific evaluation, all assessment submissions underwent a common preprocessing stage, applied identically across downstream models. For each submission, the corresponding assignment question, grading rubric, and student submission file were retrieved from cloud storage. Textual content was then extracted from the assignment question document, and the grading rubric was parsed into a structured format to enable consistent injection into the evaluation prompt across both models. This shared pre-processing step ensured that both GPT-4 and Gemini 2.5 Flash received identically sourced and identically structured question and rubric content, with any divergence between the two evaluation pipelines confined strictly to model-specific input handling and inference, as described in the following subsections. Following evaluation, results from both pipelines were persisted through a common results-handling stage, in which the aggregated score, structured feedback, and run-level metadata were stored against the corresponding submission record, and downstream notifications were triggered to inform students that feedback was available. The overall assessment evaluation and feedback generation process is shown in Figure 2.

Gemini 2.5 Flash Pipeline: For Gemini 2.5 Flash, non-blank pages of each student submission were rendered as images at 150 DPI resolution and converted into PIL Image objects. Each rendered page was subsequently encoded using Pillow and $base64$ encoding, producing image data in the format for direct inclusion in the model input. The three independent evaluation runs were executed in parallel using a LangGraph StateGraph, comprising three parallel nodes. Each run issued a single call via the langchain-google-genai library, constructing a prompt input containing the evaluation prompt alongside the encoded page images. Model inference was configured with $temperature = 0.0, top_k = 1, and top_p = 0$ to minimise stochastic variation and encourage deterministic scoring. Following the three parallel runs, results were aggregated by computing the per-criterion mean score and selecting the feedback corresponding to the median-scoring run.

GPT-4 Pipeline: For GPT-4, student submissions were uploaded directly as PDFs via the OpenAI Files API, with a deduplication mechanism to prevent duplicate submissions across concurrent evaluation runs. The three evaluation runs were executed concurrently using a thread-based execution pool, each independently invoking the evaluation call. Each call used the OpenAI API with the GPT-4 model, temperature = 0, and a fixed seed to ensure deterministic outputs. Aggregation of results was performed by computing the per-criterion mean score and selecting feedback from the middle-scoring run. Although the underlying evaluation approach, as shown in Figure 2, included three independent runs per submission, per-criterion score averaging, and median-based feedback selection, was identical across both pipelines, the two models were integrated using different orchestration mechanisms: a thread-based concurrency model for GPT-4, and a graph-based orchestration framework (LangGraph) for Gemini 2.5 Flash. Both approaches achieve the same underlying goal of executing three evaluation runs concurrently and independently, and the consistency of results across two structurally

different implementations suggests that the proposed multi-run evaluation methodology is not dependent on a particular orchestration framework or software architecture. This is a practically relevant consideration for the educational context of this study, as it indicates that the proposed assessment framework can be flexibly integrated to meet the requirements of different institutional systems, such as varying Learning Management Systems (LMSs) or provider-specific APIs, rather than being tied to a single technical implementation.

Multi-Run Evaluation Strategy: Despite deterministic generation settings, LLM outputs may exhibit minor variations across repeated evaluations. To improve reliability and reduce the influence of anomalous responses, a multi-run evaluation strategy was adopted. The choice of three independent evaluation runs was informed by prior work highlighting the importance of repeated-trial testing when using LLMs for educational assessment. The authors used ChatGPT-4 to recall the same 54 student responses across ten separate iterations to test recall precision and identify hallucinations before evaluation, finding that repeated trials were necessary to surface inconsistencies that a single pass would not reveal (Jauhiainen and Garagorry Guerra 2025). While that study used 10 iterations specifically to test recall accuracy prior to evaluation, the proposed approach used a similar approach of repeated trials directly in the scoring process, with 3 independent evaluation runs per submission to reduce the influence of anomalous or hallucinated outputs on the final grade. A higher number of runs, such as ten, was not adopted due to the substantially increased computational cost and API expenditure associated with multi-modal evaluation of image-based UML submissions across two models; three runs were selected as a practical compromise that still allows outlier detection and averaging, given the budgetary constraints and scale of the dataset (54 submissions evaluated across two models).



LLMs Evaluation pipelines

Each submission was evaluated three times independently using identical inputs and prompt instructions. Let S_1, S_2, and S_3 denote the scores generated by the three evaluation runs. The final score was computed as:

$$S_{\text{final}} = \frac{S_1 + S_2 + S_3}{3}$$

The arithmetic mean was selected to reduce the influence of outlier scores and improve grading stability. Although the models were configured with deterministic-oriented settings, large language models remain fundamentally non-deterministic in practice. Minor variations in internal computation, floating-point operations, and model-serving

infrastructure can still produce different outputs across repeated calls with identical inputs, even in nominally deterministic settings. This non-determinism means that a single evaluation run cannot be assumed to reliably represent the model's "true" judgment of a submission, and is also the underlying reason a multi-run strategy was necessary in the first place, since any individual run may occasionally diverge from the others due to this inherent variability rather than a genuine difference in the submission being evaluated.

This variability has direct consequences for feedback quality. Across three independent evaluation runs of the same submission, occasional divergence was observed in the consistency with which the model identified specific rubric-relevant elements. For example, in one run the model might correctly identify that a Use Case diagram omitted a required <<include>> relationship and penalise accordingly, while in another run of the same submission it might overlook this same omission entirely, resulting in an inflated score accompanied by generic, overly generous feedback that failed to flag the missing element to the student. Conversely, a run might misinterpret an ambiguous or unconventional but technically valid notation choice as an error, resulting in an unusually harsh assessment and feedback that incorrectly instructed the student to revise a component that was, in fact, acceptable. Selecting the highest-scoring run risks propagating the first type of anomaly, providing the student with falsely reassuring feedback that omits genuine weaknesses in their submission. Selecting the lowest-scoring run risks propagating the second type of error, delivering unnecessarily discouraging or misleading feedback based on a misjudged criterion. The median run was therefore considered the most representative and stable choice: because it corresponds to the middle score across the three runs, it is by definition not the product of the single most lenient or single most severe interpretation of the submission and is more likely to reflect a typical, balanced evaluation outcome, filtering out the influence of any one run's extreme or hallucinated judgment without requiring manual review of every submission. The personalised feedback generated by the proposed approach is shown in Figures 8, 9, 10, 11, 12, and 13, in the appendices section [appendix:personalisedfeedback].

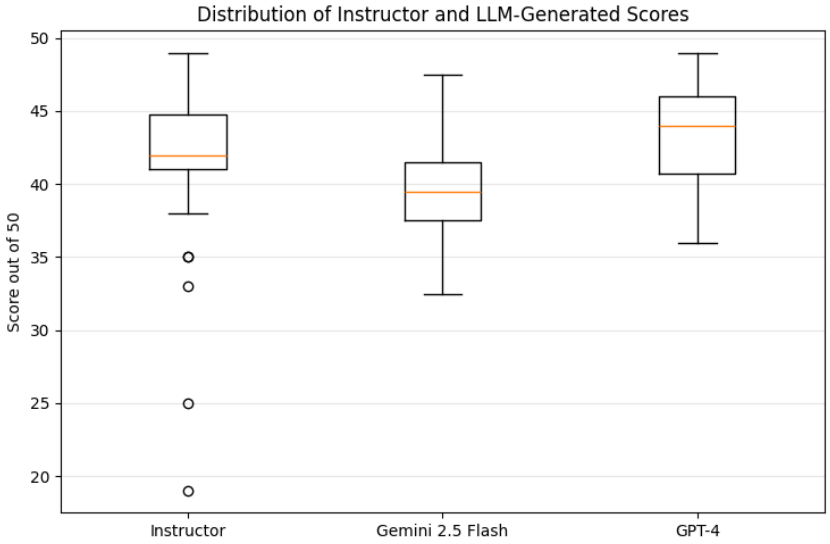
4.2 Descriptive Analysis of Assessments Grading Scores

The instructor-assigned scores were used as the reference for evaluating the performance of both LLMs. The assignment was graded out of 50 marks. As shown in Table 1, the instructor scores ranged from 19 to 49, with a mean score of 42.09 and a standard deviation of 5.37. In comparison, Gemini 2.5 Flash produced scores ranging from 32.5 to 47.5, with a mean of 39.39 and a standard deviation of 3.16, while GPT-4 produced scores ranging from 36.0 to 49.0, with a mean of 43.11 and a standard deviation of 3.84.

These results indicate that both LLMs generated scores within a narrower range than the instructor, suggesting a tendency toward score compression, in which very low and very high performing submissions were not separated as strongly as in human grading. Gemini 2.5 Flash generally assigned lower scores than the instructor, while GPT-4 assigned slightly higher scores, as illustrated in Figure 3.

Descriptive statistics of instructor and LLM-generated scores

Evaluator	Mean	SD	Minimum	Maximum
Instructor	42.09	5.37	19.00	49.00
Gemini 2.5 Flash	39.39	3.16	32.50	47.50
GPT-4	43.11	3.84	36.00	49.00



Distribution of instructor and LLM-generated scores.

4.3 Agreement Between LLM and Instructor Grading

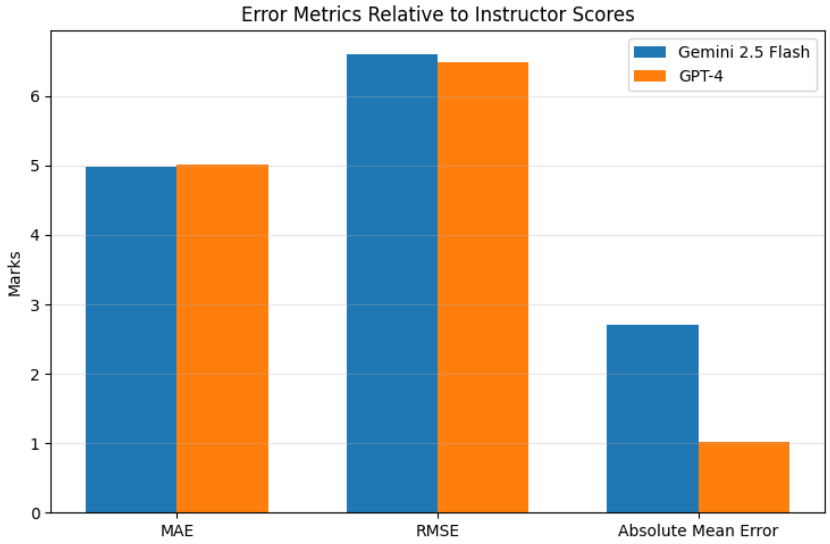
Table 2 presents the main grading performance metrics for both LLMs. Mean Absolute Error (MAE) was used to measure the average grading deviation from instructor scores. Root Mean Square Error (RMSE) was also reported because it penalizes larger grading deviations more heavily. Pearson correlation coefficient measured linear agreement, while Spearman correlation coefficient measured rank-based agreement.

Comparison of LLM-generated scores with instructor scores

Metric	Gemini 2.5 Flash	GPT-4
Mean Absolute Error (MAE)	4.98	5.02
RMSE	6.61	6.49
Mean Error	-2.70	1.02
Median Absolute Error	4.25	4.00
Pearson Correlation (r)	0.05	0.04
Spearman Correlation (ρ)	0.12	0.08

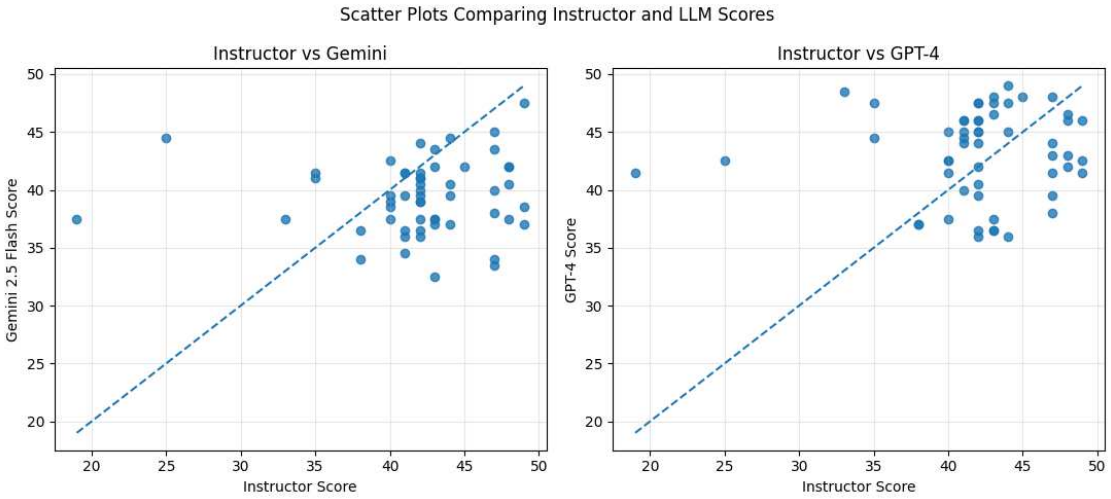
The MAE values show that both models differed from instructor scores by approximately five(5) marks on average, which corresponds to about 10% of the total assignment marks. Gemini 2.5 Flash achieved a slightly lower MAE than GPT-4, indicating a marginally smaller

average grading deviation. However, GPT-4 achieved a slightly lower RMSE, suggesting that its overall squared error was marginally lower, as shown in Figure 4.



Comparison of error metrics for Gemini 2.5 Flash and GPT-4 relative to instructor scores.

The mean error reveals different scoring tendencies. Gemini 2.5 Flash had a mean error of -2.70, indicating that it generally under-scored submissions relative to the instructor. In contrast, GPT-4 had a mean error of 1.02, indicating a slight tendency toward over-scoring. This difference is important because systematic under-scoring or over-scoring can affect student grades if LLM-based assessment is used without instructor moderation. Although the MAE values indicate that both models provided reasonable approximate scores, the Pearson and Spearman correlations were low. This suggests that neither model strongly preserved the instructor's score distribution or student ranking. As shown in Figure 5, the LLM-generated scores do not closely follow the diagonal agreement line, indicating weak score-level alignment with instructor grading. This pattern can be explained partly by score compression. The instructor assigned a wider range of scores, including very low scores for weak submissions, whereas both LLMs tended to assign scores within a compressed range. Several low-scoring submissions were therefore assigned comparatively higher marks by the LLMs, suggesting difficulty in identifying incomplete or weak UML artifacts. Such submissions may appear visually structured but fail to satisfy deeper rubric criteria such as scenario alignment, correct include/extend relationships, meaningful class associations, and appropriate architectural dependencies. Therefore, while LLMs can support approximate rubric-based grading and formative feedback, the observed score compression and weak correlation suggest that final grading should remain under instructor supervision, and that LLM-based assessment can serve as an alternative resource to help instructors better understand the assessment and expedite the process for a large cohort.



Scatter plots comparing instructor-assigned scores with LLM-generated scores. The dashed diagonal line represents perfect agreement between instructor and model scores.

4.4 Error Tolerance Analysis

To further examine grading agreement, the number of submissions falling within different error thresholds was calculated. Table 3 shows the percentage of submissions for which each LLM's score was within ± 1, ± 2, and ± 5 marks of the instructor score.

Percentage of submissions within selected error thresholds

Error Threshold	**Gemini 2.5 Flash**	**GPT-4**
Within ± 1 mark	18.52%	11.11%
Within ± 2 marks	33.33%	20.37%
Within ± 5 marks	59.26%	64.81%

Gemini 2.5 Flash performed better at stricter thresholds, with 18.52% of submissions falling within one mark and 33.33% falling within two marks of instructor scores. GPT-4 performed better at the broader ± 5 threshold, with 64.81% of submissions falling within five marks of the instructor score. Overall, Gemini 2.5 Flash was closer to the instructor in 29 out of 54 submissions, while GPT-4 was closer in 21 submissions. Both models produced equal absolute error in four cases. These findings indicate that neither model consistently dominated the other across all evaluation criteria. Gemini 2.5 Flash was slightly better for close agreement, while GPT-4 showed slightly better performance under a broader tolerance margin. Therefore, the results suggest that model selection should depend on whether the assessment goal prioritizes strict score matching or general feedback-oriented evaluation.

4.5 Paired-Samples Statistical Comparison

To determine whether the differences between instructor-assigned grades and LLM-generated grades were statistically significant, paired-samples t-tests were conducted. A paired test was appropriate because each student submission was evaluated by all three evaluators: the instructor, GPT-4, and Gemini 2.5 Flash. Thus, the analysis compared dependent score pairs across the same 54 submissions, as summarized in Table 4.

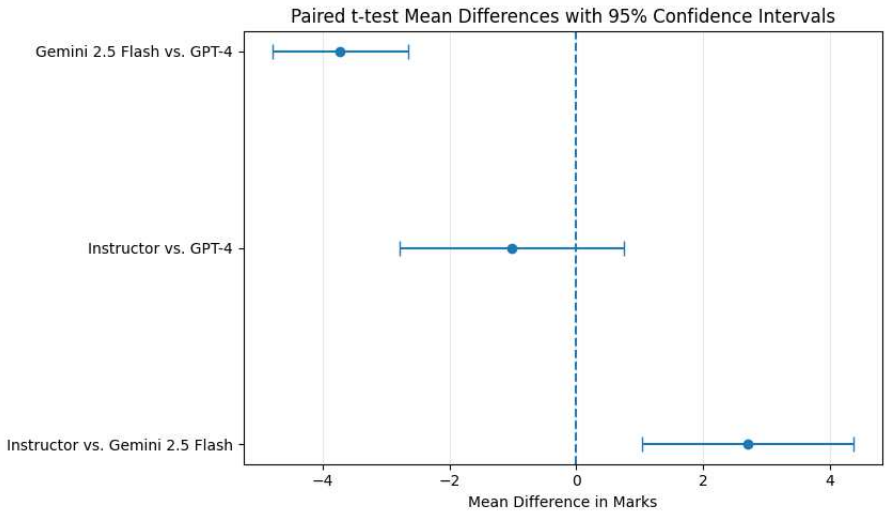
Paired-samples t-test comparison of instructor and LLM scores

Comparison	**Mean Diff.**	**95% CI**	t	p	**Cohen's d_z**
Instructor vs. Gemini 2.5	2.70	[1.04, 4.37]	3.26	0.002	0.44
Instructor vs. GPT-4	-1.02	[-2.79, 0.75]	-1.16	0.253	-0.16
Gemini 2.5 vs. GPT-4	-3.72	[-4.79, -2.65]	-6.98	< 0.001	-0.95

The mean differences and their 95% confidence intervals are visualized in Figure 6. The tests revealed distinct grading behaviors between the two models:

- **Gemini 2.5 Flash tended to under-score:** The instructor gave significantly higher scores than Gemini 2.5 Flash ($t(53) = 3.26, p = 0.002$, Mean Diff. = 2.70). The effect size ($d_z = 0.44$) indicates a moderate under-scoring tendency.
- **GPT-4 aligned well on average:** The difference between the instructor and GPT-4 was not statistically significant ($t(53) = -1.16, p = 0.253$). Although GPT-4 scored slightly higher on average (Mean Diff. = -1.02), the small effect size ($d_z = -0.16$) suggests limited mean-level bias.
- **The two LLMs differed significantly from each other:** GPT-4 assigned significantly higher scores than Gemini 2.5 Flash ($t(53) = -6.98, p < 0.001$, Mean Diff. = -3.72). The large effect size ($d_z = -0.95$) indicates a clear difference in their scoring behavior.

These results show that GPT-4 was closer to the instructor in terms of average score level, while Gemini 2.5 Flash followed a more conservative scoring pattern. However, the low Pearson and Spearman correlations reported earlier indicate that similarity in average scores does not necessarily mean the models ranked students the same way as the instructor. Therefore, paired t-tests should be interpreted together with MAE, RMSE, and correlation metrics. Overall, the paired t-test results support the use of LLMs as assistive tools for preliminary grading and formative feedback. However, because neither model reliably preserved instructor-based student rankings, final summative grading should remain under instructor supervision.



Mean score differences with 95% confidence intervals for paired-samples t-tests.

4.6 Student Perceptions of AI-Generated Feedback

A total of 48 students responded to the perception questionnaire. The questionnaire used a seven-point Likert scale, where 1 represented 'Not at all true'' and 7 represented 'Very true''. The items were grouped into five constructs: engagement, confidence and learning support, feedback quality, negative perception, and motivation and trust.

Student perception of AI-generated feedback

Construct	**Mean**	**SD**
Engagement and Interest	3.79	1.26
Confidence and Learning Support	4.20	1.31
Feedback Quality and Usefulness	4.53	1.35
Negative Perception *(Reverse Coded)*	5.18	1.37
Motivation and Trust	3.76	1.24

The highest mean score was observed for feedback quality and usefulness (M = 4.53, SD = 1.35), indicating that students generally found the AI-generated feedback helpful and actionable. Confidence and learning support also received a positive mean score (M = 4.20, SD = 1.31), suggesting that students perceived the feedback as useful for understanding mistakes and improving future work.

The negative perception construct had a relatively high mean score (M = 5.18, SD = 1.37), indicating that students generally did not feel highly anxious or overwhelmed by receiving LLM-generated feedback. This is an encouraging finding because it suggests that students were not strongly resistant to AI-supported assessment. However, motivation and trust received a mean score close to the midpoint (M = 3.76, SD = 1.24), indicating that while

students found AI feedback useful, their trust in LLM-generated feedback was not equivalent to full confidence in instructor feedback.

4.7 Comparison of Student Preference Between LLMs

Students were also asked to compare the feedback generated by the two LLMs. Table 6 presents the comparative responses.

Student preference between LLM-generated feedback

Question	**GPT-4**	**Gemini 2.5 Flash**
Closer to instructor feedback	85.42%	14.58%
More personalized and constructive	68.75%	31.25%
Preferred for future feedback generation	72.92%	27.08%

The results show a clear student preference for GPT-4 feedback. A large majority of students perceived GPT-4 feedback as closer to instructor feedback, more personalized and constructive, and more suitable for future feedback generation. This is particularly interesting because the grading metrics did not show a strong quantitative advantage for GPT-4 over Gemini 2.5 Flash. In fact, Gemini 2.5 Flash achieved a slightly lower MAE, while GPT-4 achieved a slightly lower RMSE and higher within-± 5 agreement.

This contrast suggests that numerical grading accuracy and perceived feedback quality are not necessarily equivalent. Students may value clarity, tone, specificity, and constructive guidance more than exact score alignment. GPT-4 may have produced feedback that students found more understandable, personalized, or similar to human instructor comments, even when its numerical scores were not substantially closer to instructor marks.

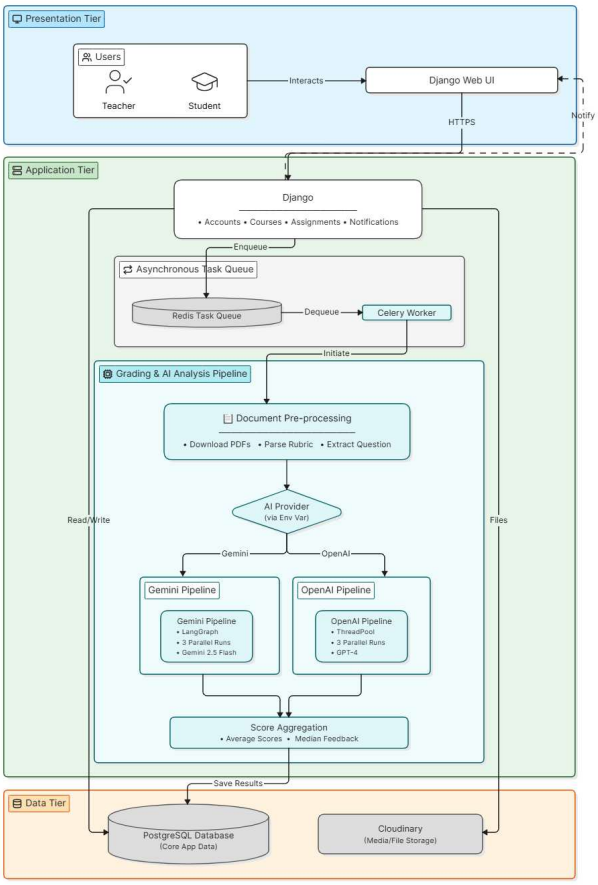
5 Software Tools and Technology

To investigate the feasibility, reliability, and consistency of the proposed LLM-based assessment, evaluation, and feedback generation, we developed a web-based learning management system (LMS) using the Django web framework. The proposed system is capable of reproducing the full conditions under which such grading would actually operate: course structures, evaluation rubrics, concurrent student submissions, and operational failure modes such as timeouts, API quota exhaustion, and worker crashes that any deployed grading system must withstand.

The platform spans four interrelated modules (accounts, courses, assignments, notifications), coordinates two independently orchestrated AI grading backends behind a single provider-agnostic interface, and implements a three-tier architecture: a synchronous web layer, asynchronous task-processing layer, and AI evaluation layer, as shown in Figure [fig:systemarchitecture]. Individual design decisions throughout the system, such as deferring task dispatch until after database commit, using Redis-backed

atomic operations to coordinate API rate limits across multiple concurrent workers, and running an autonomous recovery task to detect and resolve stuck submissions, were necessitated by problems that only emerge once AI evaluation is deployed at a realistic scale with real concurrency and real third-party API constraints.

Django was selected as the foundation for this platform because its object-relational mapper, authentication scaffolding, and administrative tooling allowed the substantial engineering effort required elsewhere in the system. The dual grading pipelines, the ensemble methodology, the asynchronous task infrastructure, and the reliability mechanisms surrounding it could be built without having to construct these more routine subsystems from first principles. This allowed the development effort to be concentrated where it mattered most for the research: the design, orchestration, and comparative evaluation of the AI grading pipelines themselves. The remainder of this section describes the specific tools and technologies used to develop the system, organised by architectural role, followed by a detailed account of the AI evaluation layer, which constitutes the core research contribution.



Architecture Diagram

Programming Language and Web Framework: The system is developed in Python, chosen for its mature ecosystem of web development, data-processing and machine-

learning library support, including Django, Celery, pdfplumber, PyMuPDF, LangGraph, Google's Gemini and OpenAI's GPT-4 APIs. A single language across the web layer, task-processing layer and AI evaluation layer avoids the integration overhead of a polyglot stack, allowing business objects such as rubric structures and submission metadata to be shared directly between Django views, Celery tasks and grading modules. Moreover, Django is used as the core web framework, providing Uniform Resource Locator routing, an object-relational mapper, session-based authentication, an administrative interface and a server-side templating engine. Django was selected over lighter-weight microframeworks because the system is create-read-update-delete (CRUD) intensive across five interrelated modules: accounts, courses, assignments, notifications and platform analytics, and Django's built-in object-relational mapper, admin panel and authentication scaffolding substantially reduce the boilerplate required to implement these domains relative to assembling equivalent functionality from smaller libraries.

Database Technologies: PostgreSQL-14 is used as the production relational database, storing users, courses, assignments, submissions, rubrics, notifications, and activity logs. PostgreSQL was selected because it provides native JavaScript Object Notation (JSON) field support for storing rubric structures and per-submission processing metadata, and because it tolerates the concurrent write load generated by multiple Celery workers writing grading results simultaneously. Also, it is the standard managed database offering on the Platform-as-a-Service targets used for deployment.

Asynchronous Task Processing: Celery is used as a distributed task queue to execute AI grading outside the Hypertext Transfer Protocol request-response cycle. Celery was selected because a single automated assessment comprising three parallel grading passes can take several seconds to minutes to complete, which would exceed the synchronous web server's timeout window. Celery decouples submission acceptance from evaluation, and provides built-in retry semantics, configurable timeouts and horizontal scaling of worker processes. Task dispatch is deferred until after the associated database transaction commits, preventing a worker from acting on a submission record before it is guaranteed to exist. Reliability under failure conditions is further reinforced through late task acknowledgement, automatic requeuing on worker failure, and a constrained task-prefetch policy that prevents uneven workload distribution across workers. Celery Beat, Celery's periodic task scheduler, runs an autonomous recovery task every 5 minutes that detects and resolves submissions stuck in an intermediate processing state, allowing the system to recover from failures without manual intervention. Moreover, Redis is used as the Celery message broker and result backend, queuing submission-evaluation tasks and storing their results for status tracking, and additionally as a distributed state store for Gemini API key rotation across worker processes. Redis was selected for this role because it is lightweight, fast, and well-supported by Celery's broker and result-backend interfaces. Also, its atomic operations allow multiple concurrently running Celery workers to share a single, consistent view of API key usage, cooldown periods, and quota state.

Deployment and Development Environment: The application is served in production by Gunicorn, a Web Server Gateway Interface server suited to concurrent request handling, with static assets delivered through WhiteNoise to avoid the need for a separate content delivery network on Platform-as-a-Service hosting. Docker and Docker Compose are used to containerise the application and its PostgreSQL and Redis dependencies for local development, ensuring that the development environment mirrors the production service topology and remains reproducible across machines.

Authentication and Email Delivery: django-allauth is used to implement email-based registration, mandatory email verification and password reset, replacing Django's default username-based authentication with an email-only login flow. django-allauth was selected because it implements verification-token generation, expiry and security best practices out of the box, which would otherwise require a non-trivial custom implementation. Also, django-celery-email is used to route all outgoing email, including verification, password reset, and course invitation messages, through the Celery task queue rather than sending them synchronously within the request-response cycle. This was adopted because Simple Mail Transfer Protocol delivery can be slow or intermittently unavailable, and routing email through Celery prevents such delays from blocking user-facing signup or invitation requests.

File Storage and Document Processing: Cloudinary, accessed through django-cloudinary-storage, is used as the durable storage location for all uploaded files, including question papers, rubrics and student submissions. Cloudinary was selected because the application's deployment targets provide only an ephemeral filesystem that does not persist uploaded files across request cycles or worker restarts, whereas Cloudinary provides durable, uniform resource locator-addressable storage that both the Django application and independent Celery worker processes can access without sharing a local disk. Document text and image parsing is performed using pdfplumber for text-layer extraction from PDF pages and PyMuPDF for page rendering; both libraries are applied to question papers, rubrics and student submissions. Pillow is used for image validation, encoding and page-rendering output in support of the multimodal grading pipeline. These libraries were selected jointly because student submissions are predominantly PDF documents that may contain embedded diagrams.

LLM Evaluation Technologies: The LLM evaluation layer constitutes the primary research contribution of the proposed approach and is deliberately provider-agnostic, switchable between two independently orchestrated backends via a single environment variable, enabling direct comparison of grading approaches within a shared experimental infrastructure without modification to the surrounding application code. LangGraph is used to orchestrate the Gemini evaluation pipeline as a directed state graph, in which a shared preparation step fans out into three parallel grading nodes and fans back into a single score-aggregation node. LangChain-Google-GenAI and langchain-core are used as the client library and message-construction layer for Gemini, providing a standardized interface for constructing multimodal messages that combine text prompts with base64-encoded page images. Google Gemini 2.5 Flash is invoked once per parallel run with

deterministic generation settings of temperature 0.0, top_k 1 and top_p 0. The OpenAI Python GPT-4 model is used as another evaluation backend. The OpenAI Files API is used to upload the submission PDF directly and obtain a file identifier that is referenced in a subsequent chat-completion call, avoiding the need to render the document to images. A ThreadPoolExecutor configured with three worker threads is used to execute the three parallel grading calls in this backend. A SHA-256 hash is computed for each uploaded file's contents so that the same file is not uploaded three times across the three parallel threads, with a lock guarding this upload cache against race conditions. Both pipelines apply an identical ensemble grading methodology: three independent evaluation runs are generated per submission, per-criterion scores are averaged and rounded to the nearest 0.5 marks, and narrative feedback is selected from the run judged most representative of the group, a design intended to reduce the variance inherent in single-run LLM-based assessment.

Front-End Technologies: The user interface is rendered server-side using Django's built-in templating engine rather than a separate single-page-application framework, which was adopted to keep authentication reliant on standard session cookies and to avoid the additional build pipeline and deployment complexity of a decoupled frontend for what is primarily a forms- and dashboard-driven academic platform. Tailwind CSS is used as the utility-first styling framework for all dashboards and forms, compiled from source during the Docker build using Node.js 20. Tailwind was selected for its speed, allowing a consistent design system to be applied across a large number of templates without maintaining a parallel stylesheet of custom class names. Moreover, the Python-Markdown library is used to convert LLM-generated grading feedback, which is produced in markdown format, into Hypertext Markup Language for display to students and teachers, with additional handling to escape angle brackets originating from UML stereotype notation so that they are not misinterpreted as Hypertext Markup Language tags. This was necessary because the grading models return structured Markdown rather than plain text, and server-side rendering avoids introducing a client-side Markdown parsing dependency.

Environment Configuration and Supporting Utilities: django-environ is used for type-safe parsing of environment variables in the Django settings module, and dj-database-url is used to parse a single DATABASE_URL connection string into Django's structured database configuration. These libraries were adopted so that the identical codebase can be deployed to local Docker Compose, Heroku targets purely by changing environment variables, without any code-level branching between environments. The requests library is used throughout the Celery evaluation task to download question papers, rubrics and submissions from their Cloudinary uniform resource locators prior to processing.

6 Discussion

With respect to RQ1, the results indicate that both GPT-4 and Gemini 2.5 Flash can approximate instructor grading to some extent, with average deviations of approximately five marks out of 50. However, the low correlation values suggest that the models did not

strongly preserve the instructor's ranking of students. Therefore, LLMs should be viewed as supportive grading tools rather than replacements for instructor judgment.

For RQ2, the comparison does not identify a single clearly superior model across all metrics. Gemini 2.5 Flash achieved slightly lower MAE and stricter threshold agreement, whereas GPT-4 achieved slightly lower RMSE and was preferred by students for feedback quality. This suggests that Gemini 2.5 Flash may be more suitable when the priority is score closeness, while GPT-4 may be more suitable when the priority is student-facing feedback.

For RQ3, the questionnaire results show that students generally perceived AI-generated feedback positively, particularly in terms of usefulness and learning support. The relatively low negative perception scores suggest that students were not strongly uncomfortable with LLM-generated feedback. However, moderate motivation and trust scores indicate that students may still prefer human confirmation for final assessment decisions.

For RQ4, students clearly preferred GPT-4 feedback over Gemini 2.5 Flash feedback. This preference suggests that feedback quality should be evaluated separately from grading accuracy. An LLM may not always be the closest numerical grader but may still be perceived as more helpful by students due to the clarity and constructiveness of its comments.

Overall, the findings support the use of LLMs as assistive tools for rubric-based assessment in software engineering education. The models were able to generate structured scores and feedback for complex UML-based artifacts, including use case diagrams, class diagrams, and three-tier architecture designs. Nevertheless, the observed score compression and weak rank correlation highlight important limitations. Therefore, LLM-generated grades should be reviewed by instructors before being used in summative assessment. A hybrid assessment model in which LLMs provide preliminary grading and formative feedback while instructors retain final authority appears to be the most appropriate approach.

7 Threats to Validity

Several threats to validity should be considered when interpreting the findings of this study. First, the dataset consisted of 54 assignments from a single course and a single institution. Although the assignments were authentic and represented a real classroom, the sample size and single course may limit the generalizability to other software engineering courses. Second, the instructor-assigned grades were used as the reference for evaluating the LLM-generated score. While this reflects the actual grading, it relies on a single instructor's evaluation, which may be influenced by their grading style, expectations, or perception of the rubric. Third, the student questionnaire was designed to compare feedback generated by the LLMs and the instructor, rather than to evaluate grading accuracy. Although item C1 in the questionnaire asked students which LLM-generated feedback was closer to the instructor, the survey did not ask the students to compare numerical scores. This design

was intentional because exposing students to score comparisons could introduce bias; students may prefer the model that awards higher marks rather than the one that provides more valuable feedback. Therefore, the survey results should be interpreted as a perception of feedback quality and usefulness.

8 Conclusion and Future Work

This study investigated the effectiveness of multimodal Large Language Models (LLMs) for automated grading and feedback generation in software engineering education. GPT-4 and Gemini 2.5 Flash were evaluated on authentic Software Engineering Concepts (SEC) assignments containing UML-based artifacts, including use case diagrams, class diagrams, and three-tier architecture designs. The models were assessed using instructor-defined rubrics, and their generated scores were compared with instructor-assigned grades. Student perceptions of AI-generated feedback were also examined through a questionnaire-based survey.

The findings show that both LLMs can generate structured rubric-based evaluations and meaningful feedback for software engineering design artifacts. However, both models differed from instructor grading by approximately five marks on average out of 50. Gemini 2.5 Flash achieved a slightly lower Mean Absolute Error, whereas GPT-4 showed no statistically significant mean-level difference from instructor scores and was more strongly preferred by students in terms of feedback quality, personalization, and similarity to instructor feedback. This indicates that numerical grading accuracy and perceived feedback usefulness should be evaluated separately.

The results also revealed important limitations. Both models compressed scores into a narrower range than the instructor and showed weak rank correlation with instructor grading, indicating difficulty in preserving student performance rankings. These findings suggest that although LLMs can support rubric-based assessment, they should not be used as fully autonomous graders in high-stakes contexts. Instead, a hybrid assessment model is more appropriate, where LLMs provide preliminary grading and formative feedback while instructors retain final authority.

Future work should evaluate the framework using larger datasets collected from multiple courses, institutions, and instructors. Involving multiple human evaluators would also enable inter-rater reliability analysis and provide a stronger benchmark for assessing LLM performance. Further research should examine criterion-level agreement to identify which rubric components, such as UML notation, class associations, or architectural dependencies, are most difficult for LLMs to assess. In addition, prompt calibration, retrieval-augmented prompting, and analysis of how students revise their work after receiving AI-generated feedback may further improve the reliability and educational value of LLM-based assessment.

Acknowledgements: The authors acknowledge the use of the writing assistance tools (Grammarly and ChatGPT) to improve the writing quality of this paper. Following its use,

the authors thoroughly reviewed and revised the content, and they take full responsibility for the final version of the paper.

Authorship contributions: F.A., M.T., and J.A.K. developed the method, detailed investigation, and manuscript writing; H.S.D. curated the research data set; M.T. conducted the analysis, F. A. developed the tool, M.T. and J.A.K. revised the methodology and supervised and revised the manuscript writing (revised draft). All authors have read and agreed to the published version of the manuscript.

Funding: The authors have not disclosed any funding.

Data Availability and Replication: The replication package is available at [2]

Conflict of interest: The authors do not have any conflict of interest.

Ethical approval: This work involved human participation and was performed in accordance with ethical principles. Furthermore, prior to data collection, ethical approval was obtained from the University of Gujarat's Institutional Ethics Committee. Participants were informed of the study's purpose, and participation was voluntary. All collected data were used solely for research purposes, ensuring participant privacy and confidentiality.

9 Student Perception Questionnaire

Students evaluated the AI-generated feedback using a seven-point Likert scale where 1 = Not at all True and 7 = Very True.

Student Perception Questionnaire

ID	Question
Q1	Receiving feedback from an LLM is more interesting than traditional grading methods.
Q2	The immediacy of LLM-generated feedback made the assessments feel more engaging.
Q3	I would prefer LLM-generated feedback over traditional methods for future assessments.
Q4	The feedback from LLMs increased my curiosity about the topic.
Q5	The LLM-generated feedback helped me feel more confident in improving my learning abilities.
Q6	I understood my mistakes better because of the detailed personalized LLM-generated feedback.
Q7	The LLM-generated feedback made me feel capable of mastering the subject.

[2] https://doi.org/10.5281/zenodo.21514637

ID	Question
Q8	The LLM-generated feedback felt highly relevant to my learning goals.
Q9	I found the LLM-generated feedback more actionable than traditional assessment feedback.
Q10	The detailed suggestions from LLMs are helpful for future assessments.
Q11	I felt anxious receiving feedback from an LLM instead of an instructor. (R)
Q12	The LLM-generated feedback overwhelmed me with too much irrelevant information. (R)
Q13	I would put more effort into my work knowing I would receive LLM-generated feedback in future.
Q14	The LLM-generated feedback motivated me to revise my assessments immediately.
Q15	I trust LLM-generated feedback as much as I trust human instructor feedback.

Comparative Evaluation Questions

ID	Question
C1	Which LLM-generated feedback is closer to the instructor's feedback?
C2	Which LLM-generated feedback is more personalized and constructive?
C3	Which LLM-generated feedback would you prefer for future feedback generation?

10 LLM-based Personalised Feedback using web-based application

Below are the screenshots for detailed personalised feedback generated using the proposed approach, showing strengths, areas for improvement, and future recommendations in the assessment, as shown in Figure 8, 9, and 10. Also, the system shows detailed personalised feedback for each UML diagram, including use case, class and architecture diagrams using predefined rubrics, as shown in Figures 11, 12, and 13.

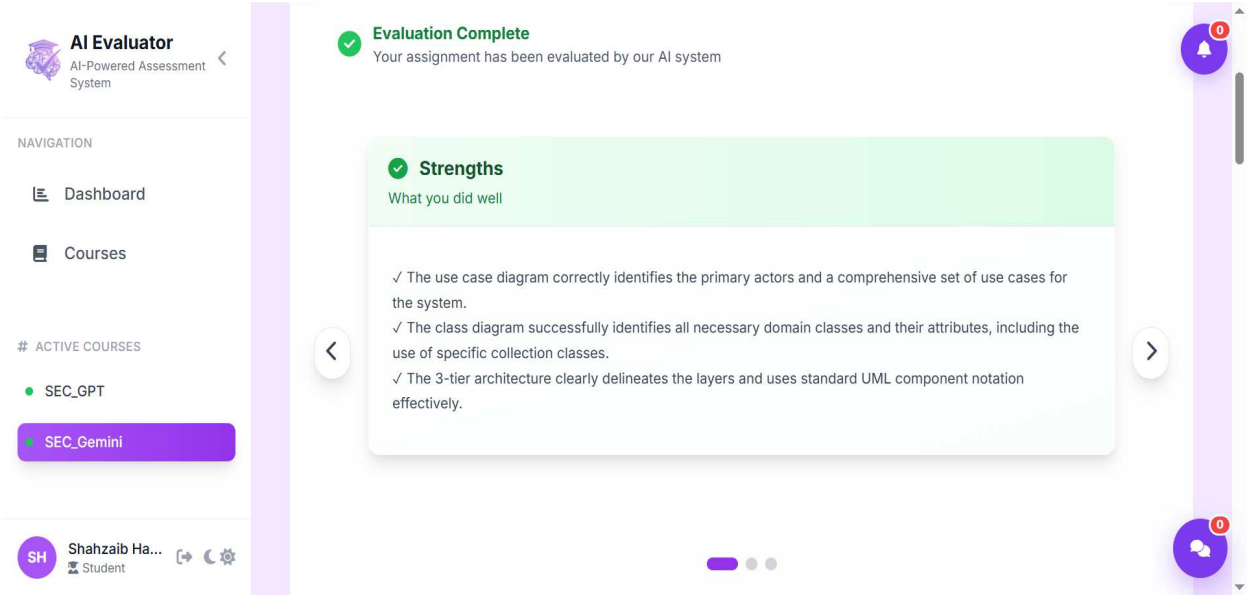


Fig. 8: Personalised Feedback Generated by LLM: Showing strengths of the student assessment

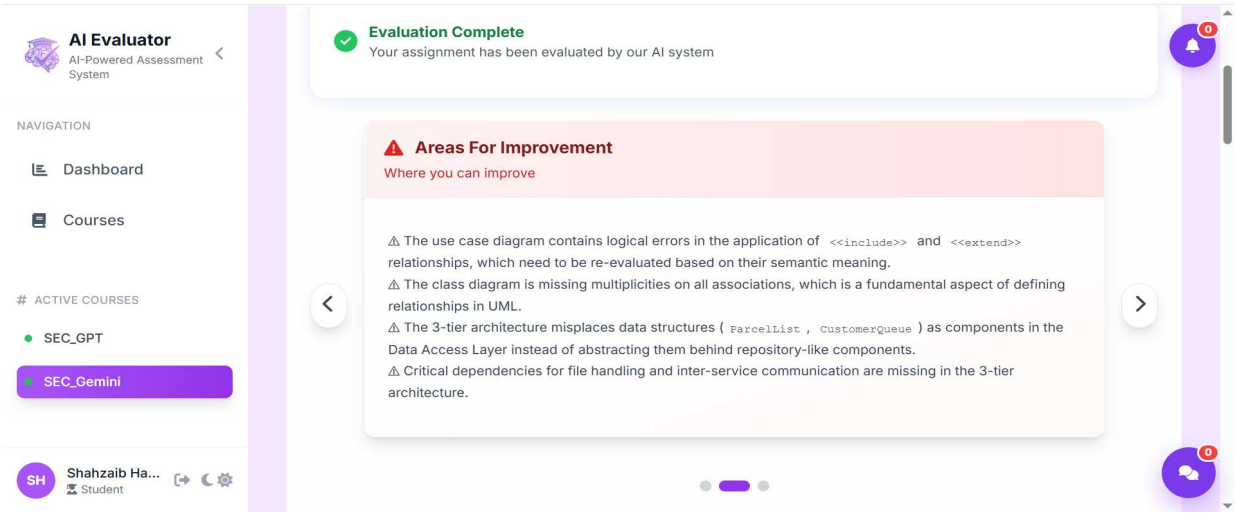


Fig.9: Personalised Feedback Generated by LLM: Showing area of improvements in the student assessment

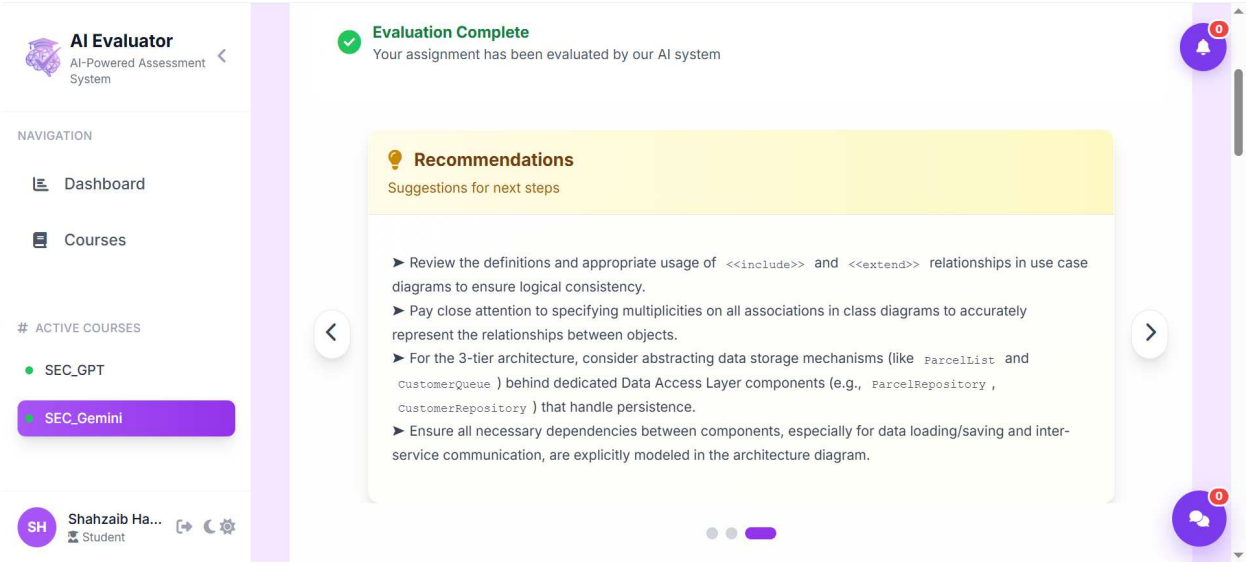


Fig. 10: Personalised Feedback Generated by LLM: Recommending further improvements in the student assessment

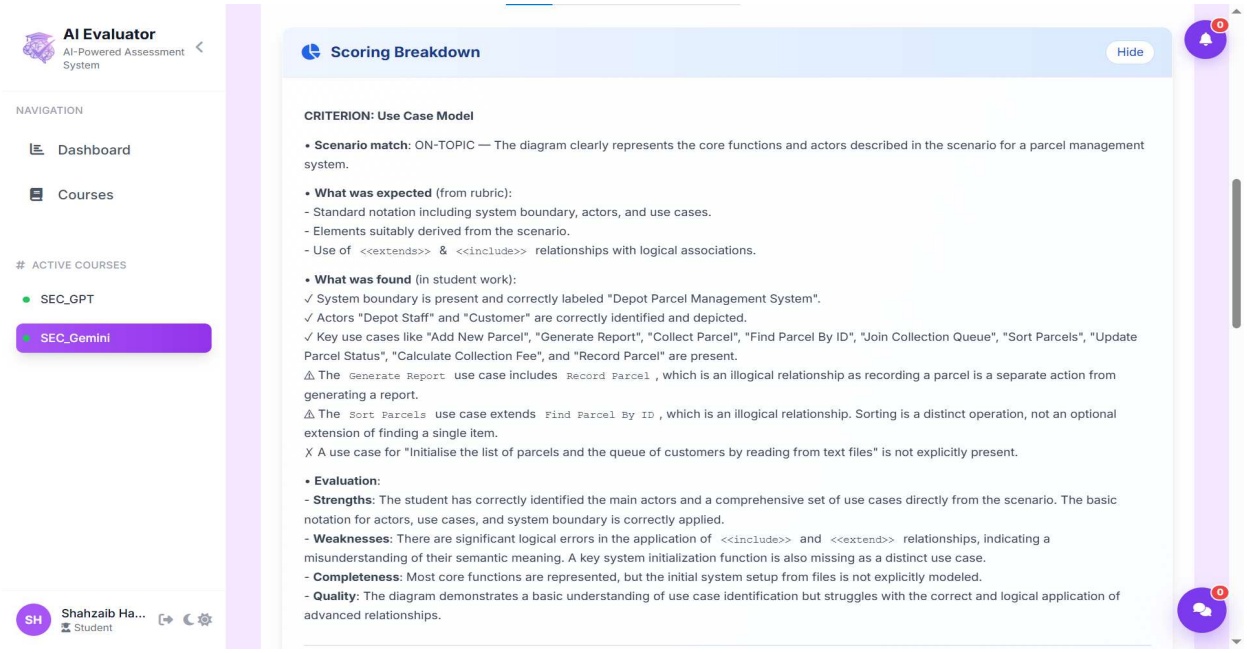


Fig.11 :Personalised Feedback Generated by LLM: Detailed feedback to the student's assessment on use case diagram based on the pre-defined rubrics

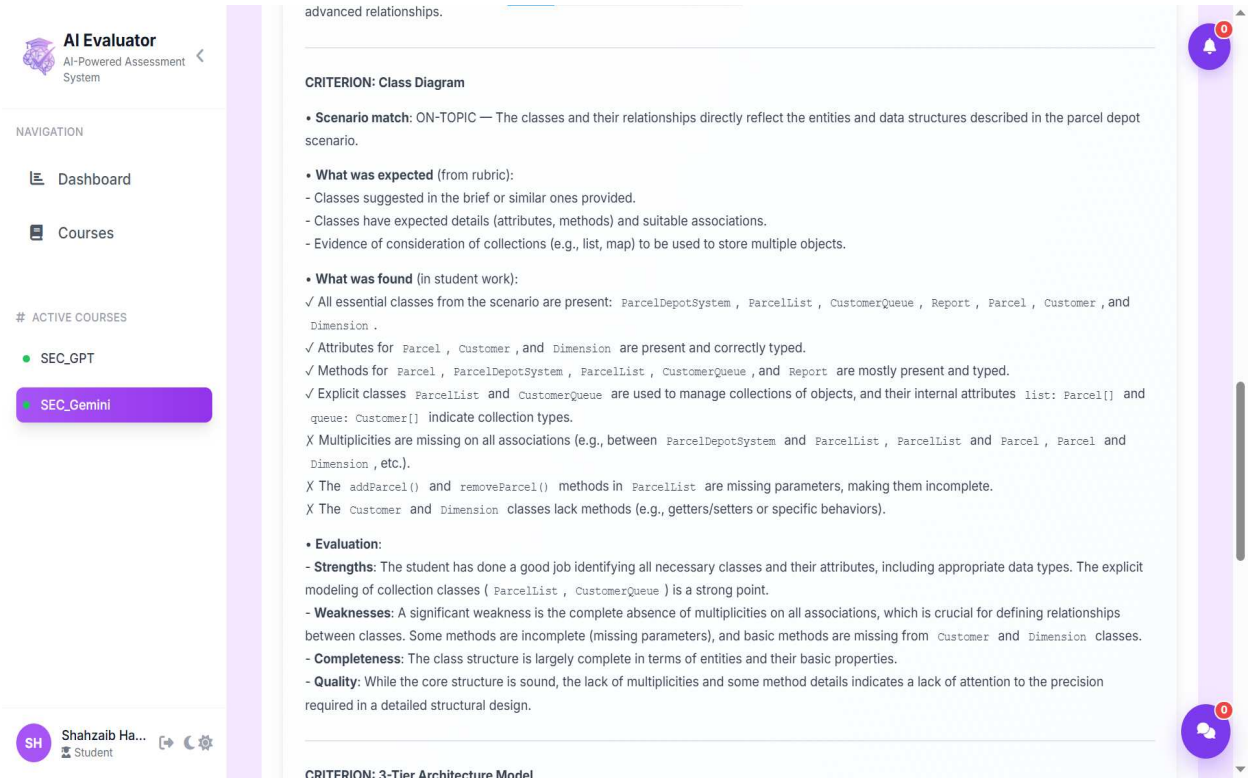


Fig. 12: Personalised Feedback Generated by LLM: Detailed feedback to the student's assessment on class diagram based on the pre-defined rubrics

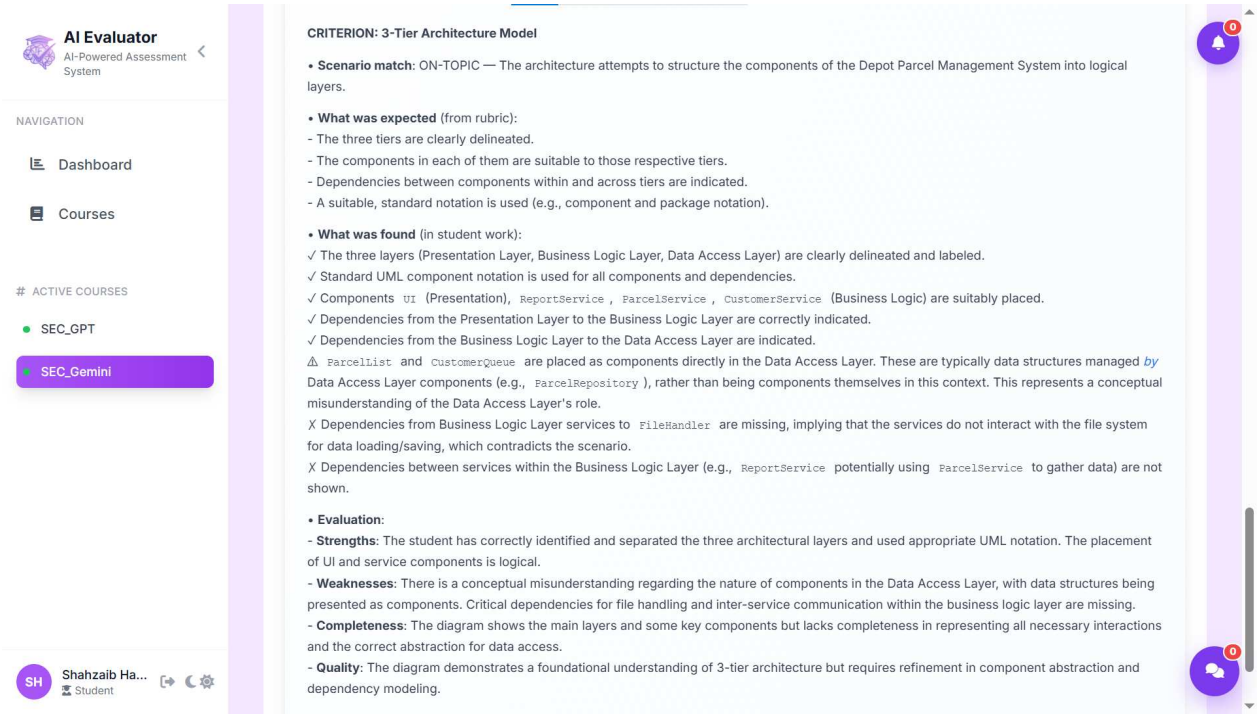


Fig.13: Personalised Feedback Generated by LLM: Detailed feedback to the student's assessment on use case diagram based on the pre-defined rubrics

References:

Alier Forment, Marc, Francisco José García Peñalvo, and Jorge D Camba. 2024. "Generative Artificial Intelligence in Education: From Deceptive to Disruptive." *International Journal of Interactive Multimedia and Artificial Intelligence* 8 (5): 5–14.

Badshah, Afzal, Anwar Ghani, Ali Daud, Ateeqa Jalal, Muhammad Bilal, and Jon Crowcroft. 2023. "Towards Smart Education Through Internet of Things: A Survey." *ACM Computing Surveys* 56 (2): 1–33.

Chang, Ju-Ping, and Kuo-Hsun Hsu. 2025. "Exploring the Automation of Software Engineering Assignment Scoring with LLMs." *IET Conference Proceedings CP929* 2025: 295–97.

Chu, Zhendong, Shen Wang, Jian Xie, et al. 2025. "Llm Agents for Education: Advances and Applications." *arXiv Preprint arXiv:2503.11733* 2.

Duong, Ta Nguyen Binh, and Chai Yi Meng. 2024. "Automatic Grading of Short Answers Using Large Language Models in Software Engineering Courses." *2024 IEEE Global Engineering Education Conference (EDUCON)*, 1–10.

Elvira, Timothy, Omar Ochoa, and Juan Ortiz Couder. 2025. "Evaluating the Efficacy of Using LLMs in Assessing Software Quality Assurance Projects." *2025 IEEE Frontiers in Education Conference (FIE)*, 1–5.

Fan, Angela, Beliz Gokkaya, Mark Harman, et al. 2023. "Large Language Models for Software Engineering: Survey and Open Problems." *2023 IEEE/ACM International Conference on Software Engineering: Future of Software Engineering (ICSE-FoSE)*, 31–53.

Frankford, Eduard, Clemens Sauerwein, Patrick Bassner, Stephan Krusche, and Ruth Breu. 2024. "Ai-Tutoring in Software Engineering Education." *Proceedings of the 46th International Conference on Software Engineering: Software Engineering Education and Training*, 309–19.

Grande, Virginia, Natalie Kiesler, and María Andreína Francisco R. 2024. "Student Perspectives on Using a Large Language Model (Llm) for an Assignment on Professional Ethics." In *Proceedings of the 2024 on Innovation and Technology in Computer Science Education v. 1*.

Guizani, Sghaier, Tehseen Mazhar, Tariq Shahzad, Wasim Ahmad, Afsha Bibi, and Habib Hamam. 2025. "A Systematic Literature Review to Implement Large Language Model in Higher Education: Issues and Solutions." *Discover Education* 4 (1): 35.

Hou, Xinyi, Yanjie Zhao, Yue Liu, et al. 2024. "Large Language Models for Software Engineering: A Systematic Literature Review." *ACM Transactions on Software Engineering and Methodology* 33 (8): 1–79.

Jauhiainen, Jussi S, and Agustín Garagorry Guerra. 2025. “Educational Evaluation with Large Language Models (LLMs): ChatGPT-4 in Recalling and Evaluating Students’ Written Responses.” *Journal of Information Technology Education: Innovations in Practice*.

Khan, Javed Ali, Shamaila Qayyum, and Hafsa Shareef Dar. 2025. “Large Language Model for Requirements Engineering: A Systematic Literature Review.” *Preprint Research Square Https://Doi.org/10.21203/Rs.3.rs-5589929/V1*.

Khan, Javed Ali, Muhammad Yaqoob, Mamoona Tasadduq, Hafsa Shareef Dar, and Aitezaz Ahsan. 2025. “Personalized and Constructive Feedback for Computer Science Students Using the Large Language Model (LLM).” *arXiv Preprint arXiv:2510.11556*.

Khan, Maryam, Muhammad Azeem Akbar, and Jussi Kasurinen. 2025. “Integrating Llms in Software Engineering Education: Motivators, Demotivators, and a Roadmap Towards a Framework for Finnish Higher Education Institutes.” *Proceedings of the 2025 29th International Conference on Evaluation and Assessment in Software Engineering Companion*, 182–91.

Mazzone, Samuel B. 2025. “The Great Grade-Off: LLMs Vs. Humans.” *Proceedings of the ACM Global on Computing Education Conference 2025 Vol 2*, 418–18.

McAuley, Edward, Terry Duncan, and Vance V Tammen. 1989. “Psychometric Properties of the Intrinsic Motivation Inventory in a Competitive Sport Setting: A Confirmatory Factor Analysis.” *Research Quarterly for Exercise and Sport* 60 (1): 48–58.

Min, Bonan, Hayley Ross, Elior Sulem, et al. 2023. “Recent Advances in Natural Language Processing via Large Pre-Trained Language Models: A Survey.” *ACM Computing Surveys* 56 (2): 1–40.

Naman, Agrawal, Ridwan Shariffdeen, Guanlin Wang, Sanka Rasnayaka, and Ganesh Neelakanta Iyer. 2025. “Analysis of Student-Llm Interaction in a Software Engineering Project.” *2025 IEEE/ACM International Workshop on Large Language Models for Code (LLM4Code)*, 112–19.

Pecuchova, Janka, L’ubomír Benko, and Martin Drlik. 2025. “Automated Grading of Open-Ended Questions in Higher Education Using GenAI Models.” *International Journal of Artificial Intelligence in Education*, 1–34.

Pham, Thanh, Thanh Binh Nguyen, Son Ha, and Ngoc Thanh Nguyen Ngoc. 2023. “Digital Transformation in Engineering Education: Exploring the Potential of AI-Assisted Learning.” *Australasian Journal of Educational Technology* 39 (5): 1–19.

Poličar, Pavlin G, Martin Špendl, Tomaž Curk, and Blaž Zupan. 2025. “Automated Assignment Grading with Large Language Models: Insights from a Bioinformatics Course.” *Bioinformatics* 41 (Supplement_1): i21–29.

Reihanian, Iman, Yunfei Hou, and Qingquan Sun. 2025. “From Pilots to Practices: A Scoping Review of GenAI-Enabled Personalization in Computer Science Education.” *AI* 7 (1): 6.

Riazi, Sara, and Pedram Rooshenas. 2025. “Llm-Driven Feedback for Enhancing Conceptual Design Learning in Database Systems Courses.” *Proceedings of the 56th ACM Technical Symposium on Computer Science Education v. 1*, 1001–7.

Schneider, Johannes, Bernd Schenk, and Christina Niklaus. 2023. “Towards Llm-Based Autograding for Short Textual Answers.” *arXiv Preprint arXiv:2309.11508*.

Seßler, Kathrin, Maurice Fürstenberg, Babette Bühler, and Enkelejda Kasneci. 2025. “Can AI Grade Your Essays? A Comparative Analysis of Large Language Models and Teacher Ratings in Multidimensional Essay Scoring.” *Proceedings of the 15th International Learning Analytics and Knowledge Conference*, 462–72.

Wang, Yue, Zhenyu Chen, Yuan Zhao, Chunrong Fang, Ziyuan Wang, and Song Huang. 2025. “RUM: Rule+ LLM-Based Comprehensive Assessment on Testing Skills.” *arXiv Preprint arXiv:2508.12922*.

White, Jules, Sam Hays, Quchen Fu, Jesse Spencer-Smith, and Douglas C Schmidt. 2024. “Chatgpt Prompt Patterns for Improving Code Quality, Refactoring, Requirements Elicitation, and Software Design.” In *Generative Ai for Effective Software Development*. Springer.

Xie, Wenjing, Juxin Niu, Chun Jason Xue, and Nan Guan. 2025. “Grade Like a Human: Rethinking Automated Assessment with Large Language Models.” *Proceedings of the International Conference on Research in Adaptive and Convergent Systems*, 1–8.

Yavariabdi, Amir, Bhuwan Paudel, Tamara Carleton, and Carlos Diego Andrade De Almeida. 2025. “Generative AI in Assessment and Feedback Generation in Higher Education: A Systematic Review.” *2025 International Conference on Education Technology and Computers (ICETC)*, 361–71.

Yeung, Calvin, Jeff Yu, King Chau Cheung, et al. 2025. “A Zero-Shot LLM Framework for Automatic Assignment Grading in Higher Education.” *arXiv Preprint arXiv:2501.14305*.

Yousef, Mina, Kareem Mohamed, Walaa Medhat, Ensaf Hussein Mohamed, Ghada Khoriba, and Tamer Arafa. 2025. “BeGrading: Large Language Models for Enhanced Feedback in Programming Education.” *Neural Computing and Applications* 37 (2): 1027–40.