Benchmarking AI Models for Grading CS Assignments Across Multiple Domains

Marek Horváth, Lukáš Tomaščík, Nikola Geciová, Norbert Ádám and Emília Pietriková

Technical University of Košice, Letná 1/9, 042 00 Košice, Slovakia
marek.horvath@tuke.sk, lukas.tomascik@tuke.sk, nikola.geciova@tuke.sk, norbet.adam@tuke.sk, emilia.pietrikova@tuke.sk

Abstract: This study examines the use of large language models for grading assignments in computer science education. A dataset of authentic student submissions, including source code, written documents, and image-based content, was evaluated using a controlled prompt strategy designed to enforce uniform numeric scoring. The models were assessed based on their ability to produce grades aligned with human evaluations while adhering to strict output constraints. The analysis focuses on grading accuracy, consistency, and sensitivity to task structure and prompt formulation. The results indicate that language models can support grading in structured assignments with clearly defined expectations, while their reliability decreases in open-ended or loosely specified tasks. Prompt formulation influenced output stability, particularly for incomplete or ambiguous submissions. Overall, the findings suggest that large language models can assist instructors in scaling assessment, provided they are deployed with clear procedures and human oversight to maintain fairness and alignment with educational objectives.

Keywords: AI-assisted; AI-based assessment; Automated grading; Computer Science Education; Large Language Models; Programming evaluation

1 Introduction

Automated assessment in computer science education faces challenges due to rising student enrolments and limited instructor capacities. Educators must evaluate large quantities of student submissions, making consistent grading increasingly difficult. Manual grading methods are slow, inconsistent, and often unable to provide timely, formative feedback, particularly when managing hundreds of assignments within short deadlines. Conventional automated systems perform well with precise and deterministic outputs, such as multiple-choice tests or straightforward programming exercises. However, complex, subjective tasks create difficulties for these methods. Evaluating open-ended programming assignments without explicit outcomes,

assessing design rationales, analyzing written reports, or reviewing presentations and diagrams requires judgment and context-based evaluation, which standard automated graders cannot readily provide.

Recent progress in Large Language Models (LLMs), such as GPT-4o, LLaMA-3, DeepSeek, and Qwen, presents opportunities for addressing these challenges. These models interpret natural language, analyze source code structures and evaluate visual content. This suggests a substantial potential for improving assessment practices. Practical application requires validation to ensure fairness, reliability, and alignment with human grading standards. Without proper validation, reliance on LLMs might introduce biases or inaccuracies affecting student outcomes. Empirical determination of model accuracy in replicating human grading across various assignment types is essential. Additionally, LLMs can generate detailed feedback, particularly beneficial for source code evaluation, assessing programming conventions, readability, and security practices.

Our study evaluates four major LLMs using historical course submissions, including source code files in C, Java, and SQL, documents and presentations in PDF and PPTX formats, and image-based assignments. Historical grades from instructors serve as benchmarks to evaluate the accuracy and reliability of LLM-generated grades. We analyze numerical grading accuracy, consistency across varied prompts, and reliability in replicating human grading patterns. By systematically testing prompt types, we aim to identify optimal practices for LLM deployment. Findings will inform educators about effectively using AI-driven grading tools to manage assessments efficiently, ensuring consistency, fairness, and enhanced feedback quality.

1.1 Objectives and Research Questions

The main objective of this study is to systematically examine the applicability of large language models for grading diverse computer science assignments. We aim to evaluate if these models can deliver accurate and fair assessments aligned with human judgments, including assignments traditionally challenging for automated grading, such as handwritten tests, visual diagrams, and software design documents. The study also investigates the impact of different prompt instructions on the consistency and quality of evaluations. We compare various open source LLM architectures to identify their strengths and limitations in the context of automated grading. The following research questions guide our investigation:

- **RQ1 -** To what extent can LLMs replicate human-assigned grades across different computer science assignment formats?
- **RQ2 -** How does the inclusion of explicit grading instructions affect the quality and consistency of evaluations produced by LLMs?

- **RQ3 -** What are the comparative strengths and limitations of different LLM architectures for automated assessment?
- **RQ4 -** Which types of student submissions (code, text, images, SQL, or hybrid formats) are most suitable for LLM-based evaluation, and which require substantial human oversight?

Our study assesses four openly licensed large language models: *GPT-4o*, *LLaMA-3-Instruct*, *DeepSeek-Instruct-7B,* and *Qwen-2-Instruct*. The models were selected to reflect realistic assessment settings in computer science education, combining one commercially available multimodal model with several locally deployable instruction-tuned models. This setup allows comparison across different model families, access modes, and levels of instruction-following behavior under comparable grading conditions. The evaluation uses an authentic dataset of student submissions covering various modalities and formats commonly encountered in computer science education. Assignments span several practical categories:

- Short algorithm implementations and memory-managed *C* projects
- Object-oriented extensions for a *Java* game
- *SQL* schema designs and query optimization
- Scanned handwritten tests from electrical engineering labs
- User interface prototypes with usability analyses
- Technical slide presentations for stakeholder briefings

Outputs are limited to numeric grades and brief rationales, enabling direct comparison with historical instructor scores through metrics such as *accuracy, mean absolute error, quadratic weighted kappa, macro-averaged F1, and Pearson correlation.* The study intentionally excludes plagiarism detection, code generation, and conversational tutoring to focus strictly on grading functionality. The analysis emphasizes practical considerations relevant to computer science departments, including limited GPU resources, strict data privacy constraints, and short grading turnaround times. These realistic parameters provide practical guidance for educators and curriculum developers on deploying AI-driven grading systems efficiently and responsibly.

2 LLM-based Grading

Increasing enrolment in computer science courses continues to raise pressure on teaching staff to deliver useful feedback while handling a large number of student submissions. Manual grading, even with detailed rubrics, often leads to inconsistent results and slow turnaround [1]. Traditional automated tools help in checking simple correctness through test cases [2] but do not address other important aspects such

as structure, style, or explanation clarity [3]. Attempts to combine automated scores with human judgment sometimes bring more inconsistency and create new problems with time and effort.

Language models offer a possible compromise by bringing scalability with some level of understanding. Experiments with *GPT-4* show that it can reproduce instructor scores for open questions with less variation than peer evaluation [4], and *ChatGPT* reached a strong correlation with human results in medical short-answer tests [5]. In programming tasks, *GPT-4* gave useful feedback in most cases when properly prompted [3]. Specialized models such as *Codex* have been used in tutoring systems and have reduced the demand for direct consultations without affecting student success. Benchmark studies confirm that *ChatGPT* and similar models can perform similarly to beginner-level graders in many common programming tasks, including code checks and feedback [6-8]. Other reports show that language models can generate valid answers to standard exercises in undergraduate teaching [9] [10], though they remain weaker in tasks that require deeper understanding or reasoning.

At the same time, several problems remain. Reviews have shown that language models can focus on superficial traits that are not related to actual knowledge or skill [11]. The explanations that accompany scores are not always correct or aligned with what the model actually used to produce the grade [12]. In some studies, a significant number of *GPT-4* code comments were either incorrect or confusing [13]. Tasks that involve diagrams or other non-text input also remain difficult. For example, grading flowcharts still requires an additional step to turn shapes into structured information that the model can process [14]. Also, many of the best-performing models rely on extra training with human feedback [15], which can introduce hidden errors or dependence on the training data. Prompt design remains a challenge because even small changes in wording can lead to different results [16]. At a broader level, automatic assessment of code structure or diagrams still lacks a standard and reliable solution [17]. In addition, cost factors such as computing resources, data handling policies, and integration into existing systems all limit the use of these tools at scale.

Earlier studies [18] [19] show that LLMs can match human scores in narrow and clearly defined tasks. What is less clear is how they behave across different formats, languages, or assignment types. This work addresses that question by testing several open-source language models on a real dataset that includes different formats and content types. The analysis focuses on accuracy, variation, and resource use under a consistent set of prompts. The results are relevant not only for designing assessment systems but also for thinking about how feedback can be scaled in a way that remains fair and useful for students.

3 Dataset Preparation and Processing

Our analysis is based on a dataset collected from computer science courses between 2019 and 2024, reflecting syllabus updates, technological changes, and instructional variations. Submissions were preserved in their original format to retain authenticity, including build scripts, auxiliary files, and notes. Each submission was anonymized and standardized to ensure privacy and consistency. Assignments were grouped as follows:

1. Semester - Introductory C programming and electrical engineering worksheets (n = 79) with breadboard diagrams and traces.
2. Semester - Procedural C projects with memory management, hardware integration, and recursion. (n = 3322 graded submissions, each project consisting of 6–10 assignments)
3. Semester - Java-based OOP tasks extending a game engine, with inheritance and unit testing. (178 projects, 4 graded assignments per project; n = 712)
4. Semester - SQL schema design. (collected but not included in the final evaluation)
5. Semester - UX design Figma prototypes and usability reports. (178 prototypes and 193 reports; n = 371)
6. Master's course - Requirements engineering with structured documentation and stakeholder presentations. (12 projects with 4 graded submissions per project and 21 standalone reports; n = 69)

To protect privacy and ensure consistency, all submissions were processed in three stages. Identifiers were replaced with salted hashes, directory structures normalized, and timestamps rounded. Slides were converted to *PDF*, and *Figma* prototypes were exported as high-resolution images. The dataset was enriched with compilation logs, syntax trees, and control flow graphs. OCR was used on scanned submissions, and instructor grades and feedback were preserved. Common technical issues such as encoding errors, unsupported libraries, and faint diagrams were resolved manually when needed. SQL assignments were excluded from the final evaluation due to data availability constraints and are therefore not reflected in the reported results. This dataset provides a realistic basis for evaluating language models in automated grading scenarios and supports the development of robust, AI-assisted assessment workflows.

To support reproducibility, the evaluation pipeline, prompt templates, and analysis scripts used in this study are publicly available in a dedicated repository[1]. Due to the educational nature of the dataset and the reuse of assignments across course

[1] https://github.com/marek-horvath/Benchmarking-LLM-assignements

editions, the original student submissions cannot be released publicly, as this could enable future students to access and reuse assessment materials. The fully anonymized dataset is therefore stored on institutional infrastructure and can be made available to researchers upon reasonable request for verification purposes. The public repository includes executable code, prompt definitions, configuration files, and sample outputs that allow the evaluation process to be inspected and replicated without direct access to the original submissions.

4 Prompt Design and Evaluation Setup

To ensure that large language models produced useful and consistent grading outputs, we developed a controlled prompt engineering workflow built on a set of predefined templates. The goal was not only to extract reliable scores but also to observe how models respond to variations in instruction detail and tone. Since different models interpret instructions with varying degrees of strictness, prompt design was critical to maintain format consistency, reduce unnecessary verbosity, and avoid behaviors such as hallucinating code or rewriting submissions. The prompt system consists of three templates, each tailored to test a different aspect of the model's scoring behavior.

1. **Numeric score only -** Persona is defined as a strict grader. If the submission is empty, the model returns 0; otherwise, a whole number from 0 to 100. Output must contain only the score, without any explanation or formatting. This format is used to test raw scoring logic and enforce compliance with fixed format output.
2. **Score with brief justification** - Same strict grader framing. Model prints the score followed by a one-sentence explanation limited to twenty words. This variant confirms that the model has actually parsed the content and highlights primary scoring criteria without excessive output.
3. **Partial credit enabled** - Grader persona is less strict. The model is instructed to allocate scores based on partial correctness, but must not rewrite the submission. Output remains a single number, without commentary.

Each prompt is generated by a preprocessing script that inserts the task description, appends relevant rubric content, maps class and method names with Slovak equivalents in parentheses, and adds an explicit note when the submission is empty. The automation guarantees consistency across the entire evaluation set, and every prompt variant is stored with a *SHA-256* hash to support later auditing. Where detailed rubrics were available, their items were decomposed into atomic criteria and embedded directly into the prompt. When rubrics were missing or incomplete, a fallback was used that split points equally between functionality and structure,

maintaining a uniform approach across the dataset. Since all student work was originally submitted in Slovak, and models occasionally penalized unfamiliar terms, key identifiers were listed in both Slovak and English to improve model interpretation without exceeding token limits. Early tests revealed format-related errors. Some models returned values with decimals, added markdown formatting, or responded in Slovak despite English prompts. Format restrictions for *Prompt 1* removed almost all of these deviations. The word cap in *Prompt 2* reduced verbose outputs and forced concise feedback. *Prompt 3* instruction to avoid rewriting code was critical to prevent models from *"fixing"* the student's submission, which had been a recurring issue.

Rubric data were stored as structured records that paired each criterion with a corresponding weight. These records allowed prompt, specific content insertion. *Prompt 1* excluded rubrics to test baseline behavior, *Prompt 2* included rubric items in compact form to guide rationales, and *Prompt 3* embedded rubric language without weights to test scoring without direct arithmetic hints. Subjective criteria such as visual clarity or creativity were abstracted to simplified proxies or excluded if they could not be represented clearly. A maximum of three proxies was used per prompt to remain within context limits.

To validate the effectiveness of this setup, we ran two minimal baselines:

- **Zero shot -** Only the assignment and submission were provided. Models defaulted to speculative generation, often inventing or correcting content. No open model produced reliable scores, and *GPT-4o* returned numeric output in just 22% of cases.
- **No scale -** Assignment and rubric bullets were included, but no score range was specified. Models generated results like *7 / 10* or added decimals, confirming that explicit scale definition is required.

These baselines highlighted that structured prompts are essential for reproducible grading. The three-template strategy was selected because it balances output control with insight into how models interpret scoring rules. Fewer than two percent of all runs required retries due to format violations. If a response failed to meet constraints, the prompt was repeated once, and persistent issues were flagged for manual review. Overall, this prompt design provides a consistent framework for large-scale evaluation. With prompt templates, rubric-aware content, and automated format checks, the setup ensures that grading outputs are both repeatable and aligned with expectations. This approach is not only about shaping output but about assessing whether models apply grading instructions accurately, without being distracted by language or superficial formatting details.

5 Evaluation Strategy and Grading Metrics

This section describes the evaluation setup used to assess the grading performance of large language models across multiple assignment types. Four models were tested *GPT-4o*, *Meta-LLaMA-3.1-8B-Instruct*, *Qwen2.5-7B-Instruct-1M*, and *DeepSeek-R1*. All student submissions were pre-graded by human instructors. The dataset included programming assignments, technical presentations, and written reports. Submissions were anonymized and preprocessed to standardize input formats. Files were extracted from archives, parsed, and, where needed, text was retrieved via *OCR*. A custom *Python* tool handled data extraction, formatting, and *API* communication.

Prompts were selected to reflect three styles: numeric-only grading, numeric with justification, and proportional grading without explanation. All outputs were capped in token length, with numeric-only prompts limited to 3 or 10 tokens and prompts with rationale limited to 250. Input prompts included task descriptions and rubric elements and were kept consistent across all models. *GPT-4o* was used through a web interface and accepted native *PDF* and archive inputs. Local models were tested on plain text extracted from original submissions. Prompt instructions emphasized the output format and discouraged any attempt to solve or correct student work. Models varied in compliance and behavior. *Qwen* produced consistently formatted outputs and adhered closely to prompt rules. *DeepSeek* often exceeded length constraints and introduced internal reasoning steps that interfered with prompt objectives. *LLaMA* frequently ignored instructions and attempted to complete or improve student responses, even in numeric-only tasks. *GPT-4o* maintained prompt discipline throughout, regardless of input modality. For each model, output formatting, prompt fidelity, and response structure were evaluated as part of the grading assessment.

Evaluation metrics were selected to measure both absolute scoring accuracy and consistency across categories. Exact match accuracy is calculated as the percentage of identical scores. *Mean Absolute Error* measures numeric deviation. *Cohen's Kappa* assessed agreement on categorical levels (grades A–FX) and accounted for chance alignment. Macro averaged *F1* score was based on per-grade precision and recall. It mitigated class imbalance by weighting each category equally. *Pearson correlation* measured whether the model preserved ordinal ranking across submissions. To enable categorical analysis, numerical grades were converted into grade labels. For each label, the number of correctly predicted instances and total predictions were computed. These were used to derive per-label precision, recall, and F1. Averaging across labels produced the macro F1. Kappa was computed using the same label set, aligning predictions and true values. Pearson correlation used raw scores to quantify alignment in grading trends, especially in borderline cases.

Models were tested across multiple assignment types. In object-oriented programming tasks, *Qwen* aligned most closely with human marks on interface use

and inheritance. *LLaMA* struggled to evaluate partial structure and often gave full credit to incomplete solutions. In presentation-based assignments, *GPT-4o* identified usability language but underreported visual issues. In written reports, all models demonstrated bias toward high scores when the rubric information was partial. Output compliance also impacted reliability. *DeepSeek* regularly exceeded token constraints and appended speculative text. In multi-part inputs, *LLaMA* sometimes dropped sections. *GPT-4o* and *Qwen* returned consistently bounded responses, even on ambiguous input. When faced with incomplete files, *Qwen* penalized appropriately under strict prompts, while *LLaMA* defaulted to average scores. These discrepancies underscore the role of prompt design in controlling output variability and output structure.

Format parsing influenced performance. Locally hosted models processed plain text only. This excluded the layout and visual cues available in the original documents. *GPT-4o*, accepting native formats, metadata, and layout when scoring reports. This difference was apparent in slides with structured bullets or annotated diagrams. Where text-based models missed section boundaries or failed to recognize annotations, *GPT-4o* maintained context awareness. Inference time and stability were also monitored. Local models, constrained by hardware and token limits, required repeated restarts on large files. *DeepSeek* crashed occasionally on nested prompts. *GPT-4o* responded within seconds for all formats. This gap in throughput and failure rate limits the practical deployment of open models without further tuning.

Taken together, the evaluation setup provided comparative insight into model behavior, prompt compliance, and scoring reliability. Metrics covered absolute and relative accuracy, categorical agreement, and format sensitivity. Observations from prompt-based differences and model tendencies informed the interpretation of results across assignment types. These findings serve as a basis for evaluating the potential of large language models as grading assistants under realistic academic conditions.

6 Result and Discussion

The experimental results provide an evaluation of the grading performance of each tested model. We report both raw score deviation and categorical agreement with human-assigned grades, as well as internal consistency and alignment trends. This section presents findings through both visualizations and summary tables, organized by metric and assignment type.

1.1.1 Quantitative Results

This subsection details the application of several evaluation metrics, including *MAE*, *accuracy*, *F1*, *Cohen's Kappa*, *Pearson correlation,* and *prompt consistency*.

Each of these metrics provides a different perspective on the quality and reliability of the models' grading behavior.

Figure 1 shows the average mean absolute error (MAE), scaled to a 10-point scale, across assignments. This normalization was necessary due to the varying original score scales across tasks. *GPT-4o* consistently outperformed the other models, particularly in OOP-related assignments. *Qwen-2.5* showed moderate accuracy, with performance degradation on more complex tasks, while *LLaMA-3.1* exhibited the highest MAE overall.

Figure 2 presents MAE grouped by prompt type. Again, *GPT-4o* demonstrates lower error regardless of prompt variation, while *Qwen* and *LLaMA* show a mild performance drop when switching from Prompt 1 to other prompts.

Figure 3 presents the average grading accuracy across different assignments. *GPT-4o* consistently achieved the highest accuracy. *LLaMA* struggled across all assignments, rarely exceeding 15% accuracy. These results highlight the variability in grading reliability depending on the complexity and structure of the assignment.

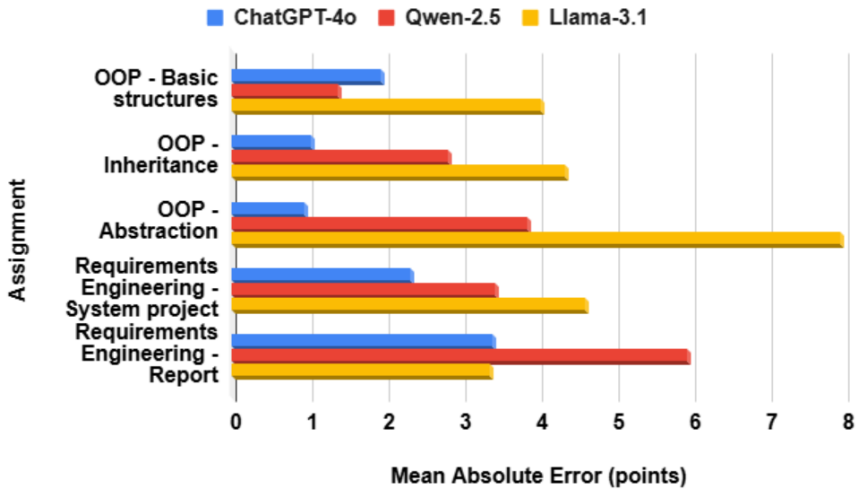

Figure 1
Average mean absolute error by assignment, scaled to 10 points. (lower = better)

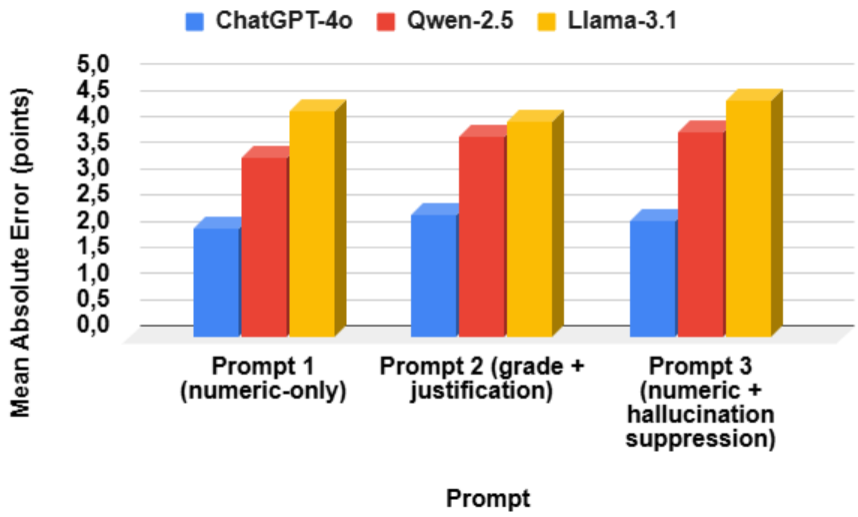

Figure 2
Average mean absolute error prompt scaled to 10 points (lower = better)

Figure 4 shows the effect of prompt type on accuracy. Prompt 1, which instructed models to output only a numeric score, consistently produced higher accuracy than Prompts 2 and 3. This suggests that reducing output complexity lowers the chance of format errors or hallucinations, especially in lower-performing models.

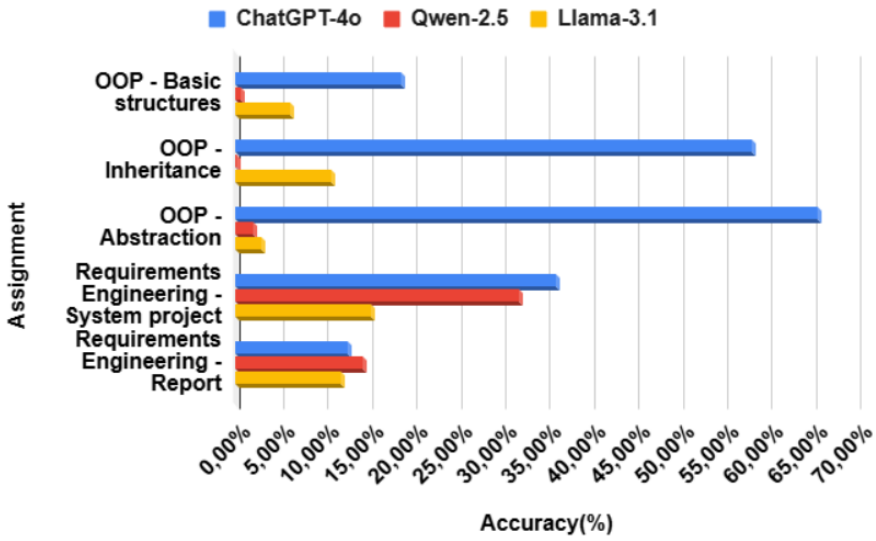

Figure 3
Average accuracy by assignment (higher = better)

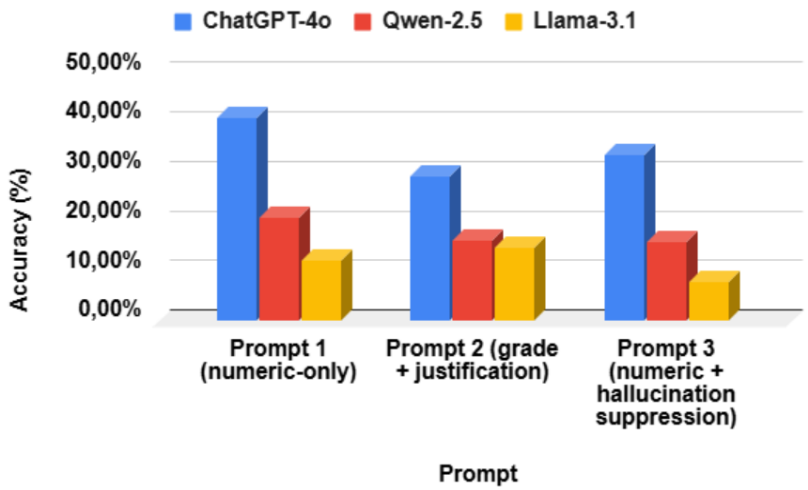

Figure 4
Average accuracy by prompt (higher = better)

To evaluate the consistency of model judgments under different prompt formulations, we computed average Pearson correlations between prompt pairs within each assignment. These correlations were transformed using *Fisher's Z* and averaged to produce an estimate of intra-model grading consistency. High correlation suggests that the model has an internal grading policy, while low or negative correlation implies prompt sensitivity or stochastic behavior. In educational or automated grading scenarios, such internal consistency is essential to ensure fairness and reproducibility of results, regardless of prompt phrasing.

Table 1
Intra-model grading consistency (average Pearson correlation across prompts; higher = better)

Assignment	**GPT-4o**	**Qwen 2.5**	**LLaMA 3.1**
OOP - Basic structures	0.927	0.895	0.337
OOP - Inheritance	0.982	0.828	0.434
OOP - Abstraction	0.514	0.651	0.278
Requirements Engineering - System project	0.327	0.474	0.355
Requirements Engineering - Report	0.012	0.267	0.302

As shown in Table 1, *GPT-4o* exhibited the highest consistency in nearly all assignments, with correlations exceeding 0.9 in OOP - Basic structures and OOP - Inheritance. *Qwen* also showed stable behavior, albeit slightly lower. In contrast, *LLaMA* demonstrated weak consistency, especially in the OOP and reports tasks, suggesting it was more prone to prompt-sensitive grading variability.

These results indicate that prompt design affects grading behavior, but does not fully explain the observed performance differences. While changes in prompt

structure influenced score variability and output format, the relative ordering of the models remained largely stable across prompts. Models with higher internal consistency, such as GPT-4o and Qwen, consistently outperformed the others regardless of prompt variant. This suggests that model quality plays a larger role than prompt design in determining grading reliability, while prompt engineering mainly helps reduce variability rather than compensate for model limitations.

To understand how well models track human graders, we calculated the Pearson correlation between model-assigned and teacher-assigned scores for each assignment and prompt. Table 2 summarizes these correlations.

Table 2
Correlation between AI and human scores by assignment and prompt (higher = better)

Assignment Prompt	**GPT-4o**	**Qwen 2.5**	**LLaMA 3.1**
OOP - Basic structures - Prompt 1	0.092	0.031	0.039
OOP - Basic structures - Prompt 2	0.076	0.058	0.098
OOP - Basic structures - Prompt 3	0.067	0.062	0.146
OOP - Inheritance - Prompt 1	0.029	0.012	0.167
OOP - Inheritance - Prompt 2	0.028	0.124	0.101
OOP - Inheritance - Prompt 3	0.039	0.057	0.062
OOP - Abstraction - Prompt 1	0.541	0.214	0.132
OOP - Abstraction - Prompt 2	0.256	0.179	0.028
OOP - Abstraction - Prompt 3	0.120	0.232	0.116
Requirements Engineering - System project - Prompt 1	0.237	0.557	0.336
Requirements Engineering - System project - Prompt 2	0.302	0.557	0.119
Requirements Engineering - System project - Prompt 3	0.123	0.401	0.411
Requirements Engineering - Report - Prompt 1	0.105	0.092	0.202
Requirements Engineering - Report - Prompt 2	0.121	0.067	0.356
Requirements Engineering - Report - Prompt 3	0.182	0.067	0.385

ChatGPT showed moderate correlation in most assignments. *Qwen* generally performed better than *LLaMA*, especially on the Requirements Engineering - System project, but performance varied considerably between prompts. Negative correlations suggest poor alignment with human expectations and may indicate either misinterpretation of task structure or overfitting to superficial cues in the submission.

Table 3 reports the macro-averaged F1 scores for each model across assignments, averaged over all prompts. The F1 score provides a balance between precision and recall across classification categories, which is especially useful when dealing with imbalanced grade distributions.

Table 3
Macro averaged F1 scores per assignment (higher = better)

Assignment	GPT-4o	Qwen 2.5	LLaMA 3.1
OOP - Basic structures	0.124	0.115	0.056
OOP - Inheritance	0.192	0.076	0.098
OOP - Abstraction	0.204	0.062	0.022
Requirements Engineering - Reports	0.377	0.296	0.470

No F1 score is reported for the Requirements Engineering – System project due to the low number of submissions, which made meaningful classification unreliable. The Requirements Engineering – Report task used binary labels (PASS/FAIL) instead of letter grades (A–FX), which tends to inflate F1 scores because of the simpler class structure. As a result, while F1 values are highest for the Report assignment (see Table 3), they are not directly comparable to the multi-class tasks.

Across the OOP assignments, *ChatGPT* consistently achieved the highest F1 scores, though they remained modest in absolute terms. *LLaMA* struggled to capture class boundaries, frequently misclassifying submissions across grade levels. These findings reinforce that high F1 in binary setups does not necessarily reflect performance in more complex grading scenarios.

Cohen's Kappa evaluates agreement with human grades while adjusting for chance. Table 4 lists Kappa values for each assignment and prompt. Requirements Engineering - System project is excluded for the same reasons as above, insufficient data.

Table 4
Cohen’s Kappa between AI and human grades (higher = better)

Assignment Prompt	GPT-4o	Qwen 2.5	LLaMA 3.1
OOP - Basic structures - Prompt 1	0.002	0.014	0.015
OOP - Basic structures - Prompt 2	0.057	0.001	0.021
OOP - Basic structures - Prompt 3	0.031	0.018	0.007
OOP - Inheritance - Prompt 1	0.037	0.024	0.057
OOP - Inheritance - Prompt 2	0.106	0.003	0.017
OOP - Inheritance - Prompt 3	0.096	0.004	0.014
OOP - Abstraction - Prompt 1	0.206	0.020	0.002
OOP - Abstraction - Prompt 2	0.102	0.018	0.005
OOP - Abstraction - Prompt 3	0.064	0.016	0.010
RE- Report documentation - Prompt 1	0.019	0.358	0.019
RE - Report presentation - Prompt 1	0.185	0.056	0.145
RE - Report documentation - Prompt 2	0.192	0.000	0.222
RE - Report presentation - Prompt 2	0.031	0.195	0.208
RE - Report documentation - Prompt 3	0.320	0.263	0.195
RE - Report presentation - Prompt 3	0.167	0.100	0.157

The values in Table 4 are generally low, with several negative scores on open-ended tasks. Negative Kappa indicates worse than random agreement, pointing to misalignment with human grading and possible systematic biases. *Qwen* showed inconsistent performance across prompts, while *LLaMA* failed to align meaningfully with human evaluations. These results underscore a key limitation of current LLMs in replicating human grading judgment.

These findings reinforce the need for more fine-tuning, task-specific calibration, or human oversight before LLMs can be trusted for high-stakes assessment tasks.

1.1.2 Qualitative Case Studies

To complement the quantitative analysis, we examine specific cases where the LLMs demonstrated notable grading behaviors. These case studies highlight instances of accurate grading, misjudgments, and divergent assessments among models, offering a deeper understanding of their decision-making processes.

In several cases, the models accurately assessed student submissions. For example, in the OOP - Basic structures assignment, a student provided a well-structured code with appropriate comments and error handling. *ChatGPT* awarded a grade of *A*, aligning with the human grader's assessment. The model's explanation cited the code's clarity and adherence to best practices, reflecting a sound understanding of programming principles.

Conversely, there were instances where models incorrectly graded submissions. In the Requirements Engineering - Report assignment, a student submitted a report lacking a clear hypothesis and with minimal data analysis. *Qwen* assigned a grade of *B*, citing *"comprehensive data interpretation,"* which was not present. This suggests the model generated an evaluation based on expected report structures rather than the actual content, a phenomenon known as hallucination.

In several cases, the same student submission received markedly different grades depending on the model and the prompt type. For instance, in the OOP - Abstraction assignment, a student submitted code with clear logical flaws but employed an unconventional and creative approach to problem-solving. *ChatGPT* (using Prompt 2) awarded a grade of *B*, citing the originality and partial correctness. In contrast, *LLaMA* (under Prompt 1) assigned a grade of *D*, focusing solely on functional errors without considering creativity. This divergence illustrates how models apply different implicit grading criteria, some prioritizing correctness, others accounting for effort or intent.

Even within a single model, prompt structure influenced the outcome. For the same submission, *ChatGPT* gave a *B* under Prompt 2 (justification allowed), but only a *C* under Prompt 1 (numeric only). The difference appears to stem from the presence or absence of explanation. When asked to justify, the model recognized creativity as a partial compensating factor, whereas the numeric-only prompt gave a harsher score. This suggests that prompt design not only shapes how a model communicates

but can also subtly shift its evaluation logic. Such intra-model variability is a key limitation in current LLM grading and reinforces the need for careful prompt engineering and consistency in deployment.

The models also faced challenges with nonstandard submissions. In cases where students submitted incomplete assignments or used unconventional formats, such as integrating multimedia elements, the models struggled to provide accurate assessments. For instance, a student included a video explanation alongside minimal written content. The models either ignored the video or penalized the lack of text, indicating limitations in processing diverse submission formats.

In assignments encouraging creativity, such as open-ended project proposals, the models occasionally misinterpreted imaginative submissions. A student proposed an unconventional solution to a problem, which was innovative but deviated from standard methodologies. *ChatGPT* appreciated the creativity and awarded a high grade, while *Qwen* penalized the deviation from traditional approaches, reflecting differing capacities to evaluate creative thinking.

These qualitative analyses reveal that while LLMs can effectively grade standard submissions, they encounter difficulties with unconventional formats, creative content, and when their internal evaluation criteria diverge from human expectations.

1.1.3 Educational and Practical Implications

The integration of large language models into grading workflows can change how instructors manage feedback and evaluation. Our results show that in structured tasks, such as programming assignments, models like ChatGPT can support instructors by identifying submissions for closer review or generating initial feedback. Their ability to include short explanations may also help students understand the grading better and support reflection.

At the same time, there are clear limits. LLMs often struggle in open-ended or creative tasks where originality or ambiguity play a role. They may misjudge intent or miss important parts of the work. Reproducibility is also a problem, especially when grading depends on prompt wording or input format. Some models penalized imaginative submissions simply because they did not follow a familiar structure, which goes against common educational goals.

Instructors who decide to use LLMs should see them as support tools, not replacements. Human oversight is still necessary to keep grading fair and sensitive to the assignment's goals. These models may help save time and allow more meaningful contact with students, but without clear workflows and control, they can lead to mechanical evaluation that ignores learning needs.

Ultimately, alignment with human grading depends not only on content understanding, but on instruction following to input variation. Human graders bring subjectivity, often awarding partial credit for effort or structure, while LLMs tend

to apply rules. Neither is perfect. These findings suggest the value of combining LLM evaluation with human oversight for fair and pedagogically sound assessment.

1.1.4 Limitations and Threats to Validity

The evaluation showed that large language models can support grading, but several limitations must be considered. Some models produced incomplete or overlong responses, leading to reruns or exclusions. The dataset included a mix of structured text and complex inputs such as handwritten notes, circled answers, and scanned images. Optical character recognition was applied to these cases, but was often unreliable, especially with handwritten or non-English content. This led to the exclusion of some submissions and a bias toward digital, well-formatted inputs. Since student submissions were in Slovak and not all models supported the language, class and method names were translated into English with Slovak originals kept in parentheses. This improved recognition but introduced extra steps and may not be transferable to other multilingual settings.

Prompt construction also affected output quality. Structured prompts ensured consistency but sometimes caused models to rely on repeated patterns or ignore submission-specific content. Strict word and token limits controlled verbosity but also led to vague or incomplete explanations. In some cases, models gave low scores when the structure was unexpected, even if the submission showed effort or partial correctness. Human graders often apply more flexible judgment in such cases, considering intent and rewarding partial solutions. The models, by contrast, followed instructions literally, leading to a rigid grading style that may not reflect the full context of student work.

Finally, the base truth used for evaluation, grades assigned by instructors, was not immune to subjectivity. Despite using rubrics, small variations in interpretation are inevitable. Differences between model and human scores may reflect these nuances rather than model errors. These issues underline the difficulty of treating human grades as the only reference point. While models can be useful for grading structured tasks, their deployment requires thoughtful prompt design, preprocessing, and review. Even when models behave consistently, they lack the contextual awareness and flexibility that human graders apply in real-world scenarios.

Conclusions

This study systematically evaluated LLMs for automated grading in computer science education, focusing on accuracy, consistency, prompt compliance and multimodal handling. With respect to the research questions, the results show that LLMs can approximate human grading most reliably in structured programming and report-based tasks, while performance degrades for open-ended, creative, or visually complex submissions (RQ1). Explicit and strictly formulated prompts improve output consistency and format compliance, but do not eliminate differences in grading quality across models (RQ2). Clear differences were observed between

model architectures in terms of instruction-following behavior, stability, and multimodal handling, with stronger models showing higher internal consistency across prompts (RQ3). Overall, deterministic and well-specified assignments are most suitable for LLM-assisted grading, whereas hybrid, visual, and creative tasks require substantial human oversight to maintain fair evaluation (RQ4).

Four models were tested on a diverse dataset of programming tasks, reports, presentations, and images, simulating a realistic academic setting. Model outputs were compared to instructor grades using both quantitative and qualitative methods. A consistent prompt strategy provided a controlled framework for assessing model behavior across varied inputs.

Results show that while LLMs can't fully replicate human judgment, they can approximate it under certain conditions. ChatGPT achieved the highest accuracy and consistency, especially on structured tasks like procedural programming. Among local models, Qwen showed reliable format compliance and scores close to human grading. LLaMA and DeepSeek often produced verbose or incomplete responses, disregarding formatting or hallucinating justifications, especially on open-ended or underspecified tasks.

Evaluation of prompt design revealed that the structure and specificity of the instructions significantly influenced model output. Prompts demanding a single numeric response improved format compliance and reduced hallucinations. Asking for justifications decreased output stability, particularly for weaker models. The presence or absence of rubric cues also impacted behavior. Clearly presented rubrics improved alignment with grading criteria, though models still struggled with incomplete, unconventional, or visually complex submissions. These findings underscore the necessity of prompt tuning not just for format control but for maintaining evaluation fidelity.

In practice, LLMs are most useful for scaling grading in tasks with deterministic or rule-based evaluations, like OOP, schema validation, or memory safety. Their effectiveness drops with open-ended reasoning, layout-sensitive tasks, or visual inputs. This supports a division of labor: models can handle routine assessments, flag edge cases, and identify common errors, while instructors retain control over subjective or creative tasks. Even in structured tasks, human oversight is needed to catch anomalies and maintain long-term reliability. Without constraints, models may overestimate performance or default to safe answers, causing grading drift.

For institutions aiming to integrate AI-based assessment, the findings offer several recommendations. Prompt engineering should be central to deployment, as phrasing directly impacts performance. Educators should be trained to construct clear, enforceable prompt templates and understand how LLMs interpret rubric language. Infrastructure should include logging, auditing, and fallback systems for cases where model output is noncompliant or ambiguous. When using open models, preprocessing pipelines must ensure input formatting consistency, anonymization, and resource compatibility.

As language models evolve, future research should explore calibration techniques, multilingual support, and strategies for multimodal evaluation. Importantly, the presence of AI in the grading process must not be perceived as removing educator responsibility but rather as augmenting capacity, especially in large-scale or repetitive assessment settings. When used thoughtfully, LLMs can improve feedback, reduce turnaround time and enhance consistency, while preserving human judgment, where it matters most.

Acknowledgement

This work was supported by project KEGA No. 061TUKE-4/2025 Building Bridges between University and High School ICT Education.

References

[1] M. Messer, N. C. C. Brown, M. Kölling, and M. Shi, "Automated Grading and Feedback Tools for Programming Education: A Systematic Review", ACM Transactions on Computing Education (TOCE), 24(1), pp. 1-43, 2024

[2] P. Biro, T. Kádek, M. Kósa and J. Pánovics, "A New Method to Increase Feedback for Programming Tasks During Automatic Evaluation," *Acta Polytechnica Hungarica,* No. 19, pp. 103-116, 2022

[3] D. Lohr, H. Keuning, and N. Kiesler, "You're (Not) My Type – Can LLMs Generate Feedback of Specific Types for Introductory Programming Tasks?", Journal of Computer Assisted Learning, 2025

[4] N. Garuda, S. Golchin, S. Stamer, M. Wenger, and C. Impey, "Using Large Language Models for Automated Grading of Student Writing about Science". University of Arizona, 2025

[5] L. Morjaria et al., "Examining the Threat of ChatGPT to the Validity of Short Answer Assessments in an Undergraduate Medical Program", Journal of Medical Education and Curricular Development, Vol. 10, 2023

[6] T. Phung et al., "Generative AI for programming education: benchmarking ChatGPT, GPT-4, and human tutors", in Conference on International Computing Education Research, Vol. 2, pp. 41-42, 2023

[7] J. Savelka, A. Agarwal, C. Bogart, Y. Song, and M. Sakr, "Can generative pre-trained transformers (gpt) pass assessments in higher education programming courses?", in Proceedings of the 2023 Conference on Innovation and Technology in Computer Science Education, Vol. 1, pp. 117-123, 2023

[8] N. Kiesler and D. Schiffner, "Large Language Models in Introductory Programming Education: ChatGPT's Performance and Implications for Assessments", arXiv preprint arXiv:2308.08572, 2023

[9] E. L. Ouh, B. K. S. Gan, K. Jin Shim, and S. Wlodkowski, "ChatGPT, Can You Generate Solutions for my Coding Exercises? An Evaluation on its

Effectiveness in an undergraduate Java Programming Course.", in Conference on Innovation and Technology in CS Education, Vol. 1, pp. 54-60, 2023

[10] M. E. Ellis, K. M. Casey, and G. Hill, "ChatGPT and Python programming homework", Decision Sciences Journal of Innovative Education, 22(2), pp. 74-87, 2024

[11] I. X. Weissburg, S. Anand, S. Levy, and H. Jeong, "LLMs are Biased Teachers: Evaluating LLM Bias in Personalized Education", arXiv preprint arXiv:2410.14012, 2024

[12] P. Y. Win Myint, S. L. Lo, and Y. Zhang, "Harnessing the power of AI-instructor collaborative grading approach: Topic-based effective grading for semi open-ended multipart questions", Computers and Education: Artificial Intelligence, Vol. 5, 2024, doi: 10.1016/j.caeai.2023.100128

[13] P. Silva and E. Costa, "Assessing Large Language Models for Automated Feedback Generation in Learning Programming Problem Solving", in 8th Brazilian Workshop on Teaching of Software Engineering (iRAISE), 2025

[14] R. T. U. Bouali N. Gerhold M. and A. F, "Toward Automated UML Diagram Assessment: Comparing LLM-Generated Scores with Teaching Assistants", in Conference on Computer Supported Education (CSEDU 2025), 2025

[15] L. Ouyang et al., "Training language models to follow instructions with human feedback", Advances in neural information processing systems, vol. 35, pp. 27730-27744, 2022

[16] M. Messer, N. C. Brown, M. Kölling, and M. Shi, "Machine learning-based automated grading and feedback tools for programming: a meta-analysis", in Conference on Innovation and Technology in Computer Science Education, Vol. 1, pp. 491-497, 2023

[17] Y. Boubekeur, G. Mussbacher, and S. McIntosh, "Automatic assessment of students' software models using a simple heuristic and machine learning", in International Conference on Model Driven Engineering Languages and Systems: Companion Proceedings, pp. 1-10, 2020

[18] M. Jukiewicz, "The future of grading programming assignments in education: The role of ChatGPT in automating the assessment and feedback process", Thinking Skills and Creativity, Vol. 52, 2024

[19] J. Flodén, "Grading exams using large language models: A comparison between human and AI grading of exams in higher education using ChatGPT", British educational research journal, 51(1), pp. 201-224, 2025