Computer and Telecommunication Engineering 2025, 3(1), 3166.
https://doi.org/10.54517/cte3166

Article

Automated leaderboard system for hackathon evaluation using large language models

Bowen Li, Bohan Cheng, Patrick D. Taylor, Dale A. Osborne, Fengling Han[*], Robert Shen, Iqbal Gondal

Royal Melbourne Institute of Technology, Melbourne 3001, Australia
*** Corresponding author:** Fengling Han, fengling.han@rmit.edu.au

CITATION

Li B, Cheng B, Taylor PD, et al. Automated leaderboard system for hackathon evaluation using large language models. Computer and Telecommunication Engineering. 2025; 3(1): 3166. https://doi.org/10.54517/cte3166

ARTICLE INFO

Received: 18 December 2024
Accepted: 17 February 2025
Available online: 24 February 2025

Abstract: Evaluating large numbers of hackathon submissions quickly, fairly, and at scale is a persistent challenge. Existing automated grading systems often struggle with bias, limited scalability, and a lack of transparency. In this paper, we present a novel hybrid evaluation framework that leverages large language models (LLMs) and a weighted scoring mechanism to address these issues. Our approach classifies hackathon submissions using LLMs, converts Jupyter notebooks to markdown for consistent analysis, and integrates multiple evaluation factors—from technical quality to video presentations—into a single, balanced score. Through dynamic prompt engineering and iterative refinement against manually benchmarked evaluations, we mitigate prompt design biases and ensure stable, fair outcomes. We validate our method in a multi-campus GenAI and Cybersecurity hackathon, demonstrating improved scalability, reduced evaluator workload, and transparent feedback. Our results highlight the potential of hybrid AI-driven frameworks to enhance fairness, adaptability, and efficiency in large-scale educational and competitive environments.

Keywords: artificial intelligence; LLM-driven assessment; prompt engineering

1. Introduction

Kaggle competitions have gained popularity for hosting hackathon-style events, challenging participants to tackle real-world problems under time constraints, often involving cutting-edge technologies and complex scenarios, as noted by Gama et al. [1] and Steglich et al. [2]. However, as highlighted by Porras et al. [3], assessing a large number of submissions within strict deadlines remains a significant challenge. Traditional grading methods, while effective, often struggle with scalability according to Kumalakov et al. [4], are susceptible to bias as noted by Sadovykh et al. [5] and Steglich et al. [6], and can be exceedingly time-consuming for evaluators, as discussed by Gama et al. [7] and Farazouli et al. [8].

In recent years, integrating large language models (LLMs) into automated grading systems has attracted significant attention in educational research. Lagakis et al. [9] explore the application of LLMs in evaluating programming assignments, highlighting the challenges of manual grading in large-scale educational settings. They demonstrate how LLMs can provide accurate and timely feedback to students, improving the overall learning experience.

Another LLM-powered automated grading system was explored by Yousef et al. [10], which investigates the use of LLMs to provide detailed feedback on student submissions. Yousef demonstrates that LLMs can offer personalized and constructive critiques. For example, instead of simply labeling the solution as “incorrect” or “suboptimal”, the LLM offered a revised version of the code. The revised code features an iterative binary search algorithm and includes detailed comments explaining each change helping students improve their coding skills.

Although LLM-based grading systems offer considerable advantages in terms of efficiency and scalability, they also give rise to concerns regarding ethical considerations. A study by Farazouli et al. [8] addresses issues related to fairness, transparency, and potential biases in automated assessments, emphasizing the necessity for ethical guidelines and oversight in the implementation of such technologies.

All the studies above focus on grading systems for programming assignments, highlighting the utility of AI-powered systems in automating assessments and providing detailed feedback. However, a common concern across these works is the potential for ethical and bias-related issues in automated grading, particularly in ensuring fairness and transparency.

In contrast, instead of using LLMs purely for automated grading, we leverage LLMs to classify hackathon submissions and integrate a weighted scoring mechanism. This hybrid approach surpasses traditional AI-powered grading systems by combining the efficiency of automated classification with the fairness of manual review for top-performing submissions, as suggested by Mosqueira-Rey et al. [11]. The weighted scoring mechanism ensures a balanced evaluation by addressing potential biases and ethical concerns, providing a more transparent and reliable assessment process.

In this work, we have developed an automated evaluation system powered by large language models (LLMs). This system streamlines grading, reduces bias, saves time, and provides clear feedback, ultimately enhancing the overall competition experience. We demonstrate how LLMs can make hackathon grading faster, more reliable, and more transparent. By incorporating automated evaluations, we efficiently manage large volumes of submissions and offer feedback that earns participants’ trust. Our approach effortlessly adapts to various hackathon themes, assisting organizers in maintaining a fair and supportive environment.

Our primary contribution is a hybrid, weighted scoring methodology that integrates several evaluative factors, including technical quality, LLM-generated analysis, and video presentations, into a unified, balanced score. This approach refines traditional automated grading by carefully adjusting the impact of each component and refining prompt design for the LLM. Consequently, the evaluations maintain robustness and flexibility, meeting the diverse requirements of hackathon challenges.

2. Case study design

An in-person generative AI (GenAI) and Cyber Security hackathon [12] was organized by the School of Computing Technologies at RMIT University, taking place simultaneously at the RMIT Melbourne campus in Australia, Saigon South Campus in Vietnam, and Hanoi campus in Vietnam from the 8th to the 10th of November 2024. Entry was open to both RMIT and non-RMIT students. Participants were required to be full-time students enrolled in a bachelor’s, master’s, or PhD program.

The challenge tasks were developed using the SkywardAI platform, an open-source community providing local AI solutions. Competition teams were required to implement their defined functions and submit all work to the Kaggle [13] platform. Each team consisted of a maximum of 4 members, with one submission per team.

There were four challenge tasks the participants were required to complete.

1) Cloud environment deployment: Deploy an open-source local AI project to Amazon EC2.
2) Encryption and decryption implementation: Implement encryption and decryption mechanisms for the deployed open-source AI project.
3) Model fine-tuning for harmful web traffic detection: Refining a model [14] to effectively identify malicious web traffic.
4) Adversarial Attacks on large language models (LLMs): Employ adversarial techniques to challenge and test the robustness of LLMs.

In addition to the above main challenges, three additional mock tasks were provided for the participants to familiarize themselves with the AWS environments prior to the hackathon, ensuring the participants could focus on the main challenges themselves, rather than navigating a new environment.

2.1. Environment requirements

Table 1. This table outlines the compute configurations provided to each team, including the number of vCPUs, system memory, GPU memory (if available), and storage capacity. Amazon EC2 is used to deploy AI services, while amazon sagemaker offers a jupyter notebook-like environment for fine-tuning LLMs.

AWS Instance Type	vCPUs	Memory (GB)	GPU Memory (GB)	Storage (GB)
EC2	16	32	-	100
EC2 (GPU)	16	32	16	100
SageMaker	16	32	-	100
SageMaker (GPU)	16	32	16	100

2.2. LLM-based classification and weighted scoring

Our system uses a large language model (LLM) to classify the quality of hackathon submissions before applying a weighted scoring method to determine final grades. By converting Jupyter notebooks into markdown format, which included both code and documentation as solutions for hackathon tasks executed in the environment in **Table 1**, we ensure that the retrieval-augmented evaluation process, as demonstrated by González-Carrillo et al. [14] remains consistent and organized. Tailored prompts guide the LLM to focus on challenge-specific criteria, producing quality classifications that feed into our weighted scoring system. This approach boosts efficiency, scales easily, and delivers structured, transparent assessments that adapt to diverse hackathon settings.

2.3. Submission data overview

In our study, the data were obtained directly from our hackathon platform hosted on Kaggle. As shown in **Table 2**, 80 valid submissions were collected concurrently from three different cities across two countries. This diverse sample of 80 submissions provides a robust basis for evaluating our automated grading and feedback system, ensuring that our approach is effective across varied regional contexts and participant backgrounds.

Table 2. Valid hackathon submissions were collected from our Kaggle platform across three cities. Out of 80 submissions, 27 were received from Saigon, while Hanoi and Melbourne each contributed 25 submissions.

The Name of Cities	Submissions
SGS	27
Hanoi	25
Melbourne	25

2.4. Identified bias concerns

1) Prompt design bias: Prompts that are not carefully crafted may inadvertently emphasize certain aspects of submissions over others, leading to skewed evaluations.
2) Output variability: The probabilistic nature of LLMs can result in inconsistent grading outcomes for similar submissions, challenging the reliability of the evaluation process.

2.5. Mitigation strategies

To address these concerns, we implemented the following measures:

1) Dynamic prompt engineering: We developed a system that constructs long-context prompts, dynamically appending specific challenge criteria as system instructions. This approach ensures that all relevant aspects of each challenge are considered during evaluation, promoting comprehensive and balanced assessments.
2) Manual benchmarking and iterative refinement: We manually labeled high-performing submissions for each challenge, assigning scores based on expert evaluation. These scores served as a benchmark for testing the marking tool. The top three submissions for each challenge were then graded using the LLM-based system, and the results were compared to the manual scores. Through this iterative process, we refined the prompts and evaluation process until the automated scores closely aligned with the manual benchmarks, ensuring both stability and fairness in the marking system.

2.6. Marking strategy for hackathon

The top three teams from the Melbourne campus, the top three teams from the Saigon South campus, and the top two teams from the Hanoi campus were invited to create and submit a video proposing a solution for a startup focused on GenAI. The teams were evaluated using three distinct mechanisms. **Figure 1** shows how the entire system works.

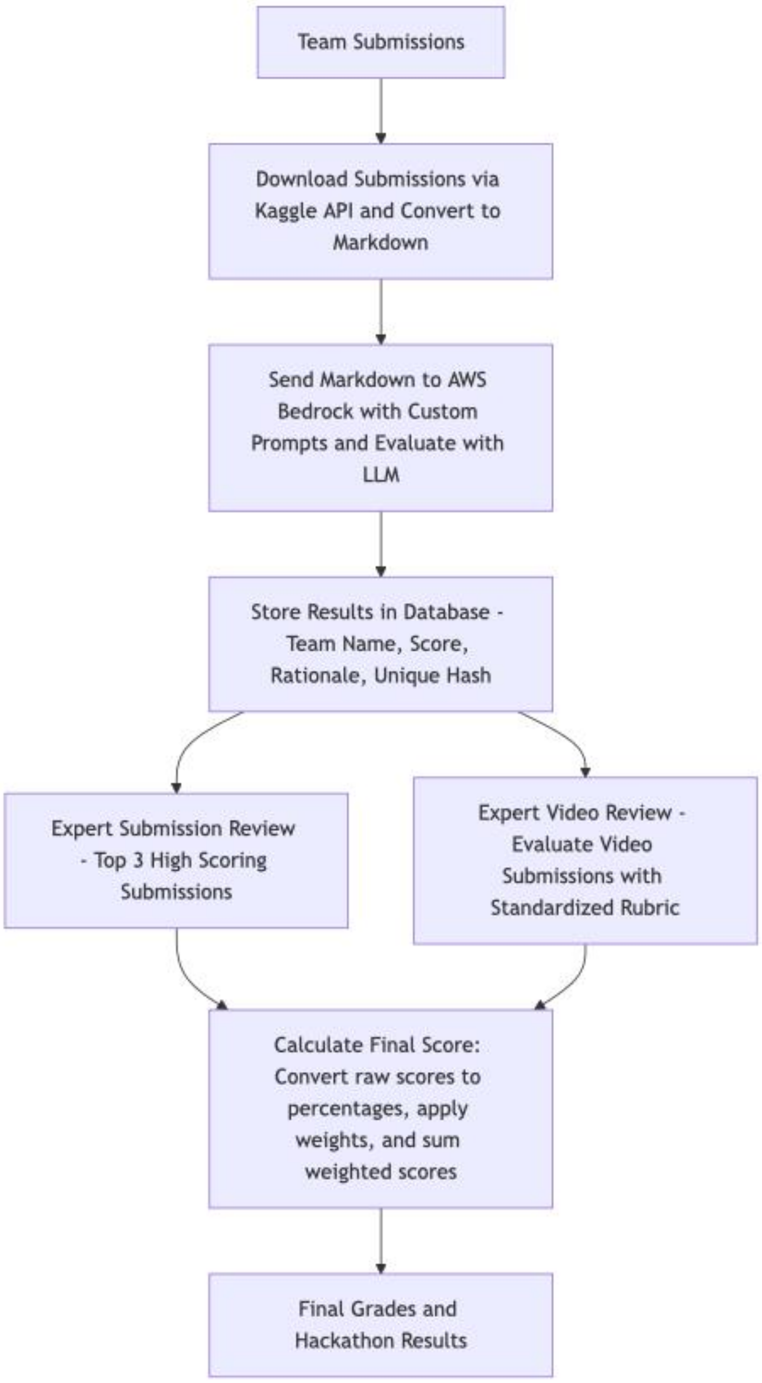
Team Submissions
Download Submissions via Kaggle API and Convert to Markdown
Send Markdown to AWS Bedrock with Custom Prompts and Evaluate with LLM
Store Results in Database - Team Name, Score, Rationale, Unique Hash
Expert Submission Review - Top 3 High Scoring Submissions
Expert Video Review - Evaluate Video Submissions with Standardized Rubric
Calculate Final Score: Convert raw scores to percentages, apply weights, and sum weighted scores
Final Grades and Hackathon Results

Figure 1. The flowchart shows the entire workflow of how the system works.

2.6.1. Automated LLM-based evaluation

Submissions are automatically downloaded using the Kaggle Competition API [15] and then converted into Markdown format. These Markdown files are forwarded to AWS Bedrock—a serverless LLM hosting service [16]—along with carefully designed system prompts that detail every aspect of the evaluation criteria. Through a multi-stage process, the LLM assigns a mark and provides a detailed rationale for each team's submission. The results, including the team's name, assigned mark, rationale, and a unique hash number (to ensure reliability and uniqueness), are then stored in our database. This evaluation focuses on assessing the functionality implemented, the supporting mechanisms, and the overall accuracy of the work.

2.6.2. Expert submission review

A panel of experts reviews the top three submissions with the highest automated marks. These experts assign additional scores based on our established assessment policy. The expert review serves as a validation step for the LLM-based evaluation: If the expert-assigned scores fall within a predetermined range, the LLM-based mark is confirmed as valid.

2.6.3. Expert video review

In addition to the submission reviews, a separate panel of experts evaluates the video submissions using a standardized rubric. This rubric is designed to assess key factors such as clarity, creativity, feasibility, and presentation quality. The video review process ensures that the overall evaluation captures not only the technical and functional aspects of the submission but also the effectiveness of its communication and presentation.

In our system, the technical score, LLM score, and video score have ranges of 400, 400, and 10, respectively. We designed an algorithm to calculate the final mark:

1) Convert each score to a percentage: Calculate each component's percentage relative to its maximum possible score. The Technical Percentage is computed as (Technical Score ÷ 400) × 100. The LLM Percentage is computed as (LLM Score ÷ 400) × 100. The Video Percentage is computed as (Video Score ÷ 10) × 100.
2) Apply weights: Assign weights to each percentage based on its relative importance to the final score. The weights are: Technical at 50%, LLM at 35%, and Video at 15%.
3) Calculate weighted score for each component: Multiply each percentage by its respective weight. The Weighted Technical score equals Technical Percentage × 0.5, the Weighted LLM score equals LLM Percentage × 0.35, and the Weighted Video score equals Video Percentage × 0.15.
4) Compute final total score: Sum the weighted scores of each component to obtain the final score, using the formula:

Total Score = Weighted Technical + Weighted LLM + Weighted Video

Let *T* be the technical score, *L* be the LLM score, and *V* be the video score. The final mathematical representation sees below:

$$F = \frac{T}{8} + 0.0875 \times L + 1.5 \times V$$

2.7. Challenge designing

Each task was marked out of 100 points. A live scoreboard was provided during the challenge and updated every hour during the main challenge day. The teams were issued a certificate upon successfully passing challenge 1 in recognition of their skills utilizing AWS EC2. This aligns with the AWS Certified Cloud Practitioner exam [17]. Challenge 2 was designed to test the participants knowledge in their understanding of cloud security to protect communications while using LLMs for inference on the public internet. Challenge 3 involved fine-tuning an AI model for binary classification to detect harmful web traffic logs. The model performance was evaluated using $F1$ metrics.

Challenge 4 was designed to analyze the impact of adversarial attacks on AI models by implementing attack techniques, comparing responses using cosine similarity, and providing a quantitative evaluation of the differences.

3. System design

The entire system architecture, shown in **Figure 2**, includes the frontend and a backend database for storing scores and reasons.

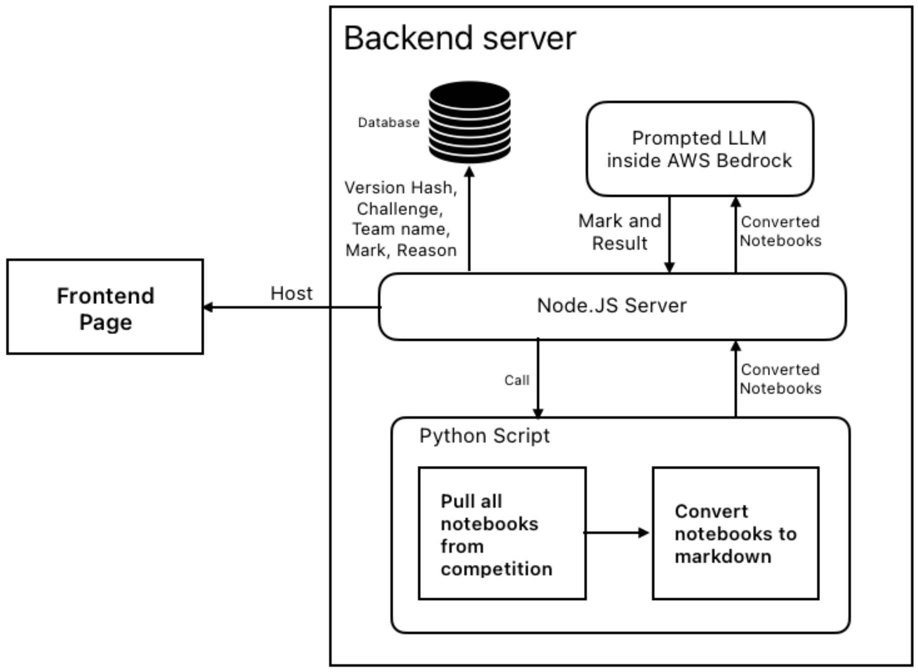
Backend server
Database
Prompted LLM inside AWS Bedrock
Version Hash, Challenge, Team name, Mark, Reason
Mark and Result
Converted Notebooks
Frontend Page
Host
Node.JS Server
Call
Converted Notebooks
Python Script
Pull all notebooks from competition
Convert notebooks to markdown

Figure 2. Overview of the system architecture illustrating the processing pipeline of the Jupyter notebook submissions, from initial raw data intake to the final predicted results. For the complete GitHub repository, see the Appendix.

4. Evaluation outcomes and impact

The AI-powered platform is designed to categorize submissions based on quality, ensuring unbiased initial evaluations. In our system, each category is defined by an LLM-driven rubric that assess key factors such innovation, practicality, technical implementation, enabling the platform to objectively assign each submission to a distinct quality tier. To address ethical concerns, we introduced a weighted scoring mechanism that combined automated scoring, manual review of top submissions, and evaluation of video presentations. This approach ensured fairness, stability, and comprehensive assessment by balancing AI efficiency with human oversight.

The results demonstrate that our real-time leaderboard system provides scalable and unbiased grading for hackathons, enabling consistent scoring and simplifying the tracking and analysis of participant performance across multiple campuses. The impact extends beyond hackathons, showcasing the potential of hybrid grading systems to accelerate evaluation processes in large-scale events and revolutionize educational assessments with scalable, transparent, and fair solutions.

Table 3. The evaluation of eight submissions is based on three scoring criteria: Technical Score, LLM Score, and Video Score, along with the calculated Gap (%) and Final Score. The Technical Score reflects human-assessed performance, while the LLM Score represents AI-based evaluation. The Video Score assesses the multimedia component of each submission. The Gap (%) column indicates the percentage difference between the Technical Score and the LLM Score, showing the level of agreement between human and AI assessments. Finally, the Final Score is a weighted combination of the three scores, providing an overall assessment of each submission.

Submission	Technical Score	LLM Score	Video Score	Gap (%)	Final Score
1	350	355	8	1.25	86.81
2	320	350	9	7.50	84.13
3	290	345	7	13.75	77.94
4	350	370	7	5.00	86.63
5	360	365	8	1.25	88.94
6	340	365	3	6.25	78.94
7	330	370	7	10.00	84.13
8	330	370	0	10.00	73.63

Percentage Gap: The relative difference mathematical formula:

$$\text{Gap} = \left(\frac{LLM\ Score - Technical\ Score}{400}\right) \times 100$$

Based on data in **Table 3**, we derive **Figure 3**, which shows a mean difference (bias) of 27.5 points—indicating that on average the LLM scores are 27.5 points higher than the technical scores—can be seen as acceptable in the context of automated grading ([18,19]). The 95% limits of agreement, ranging from approximately −6.85 to 61.85, show that for most submissions the difference between the two scoring methods falls within a span of about 68.66 points. Although the evidence from these studies is not definitive, our results suggest that the variability between the LLM and technical scores is comparable to the inter-rater variability typically observed in manual grading, thereby supporting the reliability of our hybrid evaluation approach.

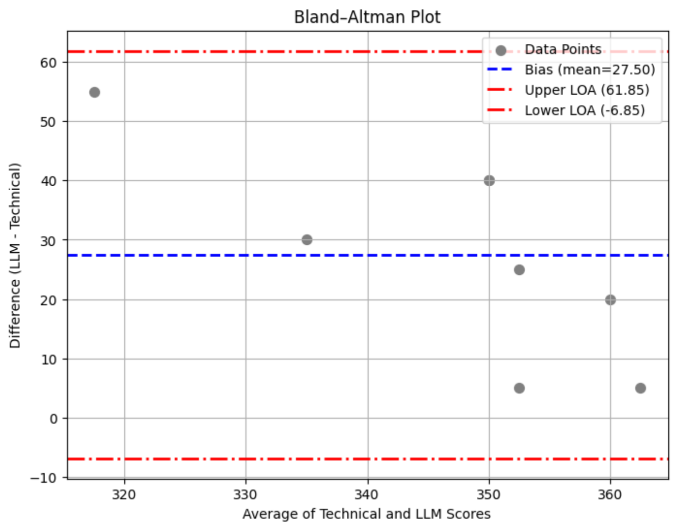

Figure 3. Bland-Altman analysis [20] table reveals a mean difference (bias) of 27.5 points—meaning the LLM scores are, on average, 27.5 points higher than the technical scores and represent roughly 6.9% of the maximum technical score. The 95% limits of agreement (−6.83 to 61.83) indicate that most differences fall within a 68.66-point range, which aligns with typical inter-rater variability in manual grading and supports the reliability of our hybrid evaluation approach.

As shown in **Table 4**, the results indicate the LLM-based marking system is more efficient than humans.

Table 4. A table shows eight submissions with the time taken for technical scoring and LLM scoring.

Submission	Technical Score Time (min)	LLM Score Time (min)
1	25	1.1
2	22	1.2
3	25	1.0
4	30	0.9
5	20	1.0
6	23	1.5
7	25	1.2
8	30	1.3

5. Discussion

In summary, our work demonstrates a novel hybrid evaluation framework that leverages LLM-driven classification and a weighted scoring mechanism to assess hackathon submissions fairly and transparently. By integrating automated analysis with expert reviews, our approach addresses challenges in scalability and bias while providing clear, constructive feedback. The successful validation of our system in a multi-campus hackathon setting highlights its potential to enhance the efficiency and fairness of evaluation processes in both educational and competitive environments.

Author contributions: Conceptualization, FH and RS; methodology, BL; software, BC; validation, BL, PDT and DAO; formal analysis, BL; investigation, BL; resources, BL; data curation, BL; writing—original draft preparation, BL; writing—review and editing, PDT; visualization, DAO; supervision, FH; project administration, IG; funding acquisition, IG. All authors have read and agreed to the published version of the manuscript.

Conflict of interest: The authors declare no conflict of interest.

References

1. Gama K, Valença G, Alessio P, et al. The Developers’ Design Thinking Toolbox in Hackathons: A Study on the Recurring Design Methods in Software Development Marathons. Software Engineering. 2022.
2. Steglich C, Marczak S, Guerra L, et al. An Online Educational Hackathon to Foster Professional Skills and Intense Collaboration on Software Engineering Students. In: Proceedings of the XXXV Brazilian Symposium on Software Engineering; 27 September–1 October 2021; Joinville, Brazil. pp. 388–397.
3. Porras J, Khakurel J, Ikonen J, et al. Hackathons in Software Engineering Education: Lessons Learned from a Decade of Events. In: Proceedings of the 2nd International Workshop on Software Engineering Education for Millennials; 27 May–3 June 2018; Gothenburg, Sweden. pp. 40–47.
4. Kumalakov B, Kim A, Mukhtarova S, et al. Hackathon as a Project-Based Teaching Tool: Employing Programming Challenge in the Class. In: Proceedings of the 2018 IEEE 12th International Conference on Application of Information and Communication Technologies (AICT); 17–19 October 2018; Almaty, Kazakhstan. pp. 1–5.
5. Sadovykh A, Beketova M, Khazeev M. Hackathons as a Part of Software Engineering Education: Case in Tools Example. In: Proceedings of the Frontiers in Software Engineering Education: First International Workshop; 11–13 November 2019; Villebrumier, France. pp. 232–245.
6. Steglich C, Salerno L, Fernandes T, et al. Hackathons as a Pedagogical Strategy to Engage Students to Learn and to Adopt Software Engineering Practices. In: Proceedings of the XXXIV Brazilian Symposium on Software Engineering; 21–23 October 2020; Natal, Brazil. pp. 670–679.
7. Gama K, Alencar B, Calegario F, et al. A Hackathon Methodology for Undergraduate Course Projects. In: Proceedings of the 2018 IEEE Frontiers in Education Conference (FIE); 3–6 October 2018; San Jose, CA, USA. pp. 1–9.
8. Farazouli A. Automation and Assessment: Exploring Ethical Issues of Automated Grading Systems from a Relational Ethics Approach. Springer; 2024.
9. Lagakis P, Demetriadis S, Psathas G. Automated Grading in Coding Exercises Using Large Language Models. Springer; 2024.
10. Yousef M, Mohamed K, Medhat W, et al. BeGrading: Large Language Models for Enhanced Feedback in Programming Education. Neural Computing and Applications. 2024.
11. Mosqueira-Rey E, Hernández-Pereira E, Alonso-Ríos D, et al. Human-in-the-Loop Machine Learning: A State of the Art. Artificial Intelligence Review. 2023; 56; 3005–3054.
12. RMIT GenAI and Cyber Security Hackathon. Available online: https://www.kaggle.com/competitions/rmit-gen-ai-and-cyber-security-hackathon (accessed on 2 December 2024).
13. Kaggle. Available online: https://www.kaggle.com/ (accessed on 2 December 2024).
14. González-Carrillo CD, Restrepo-Calle F, Ramírez-Echeverry JJ, González FA. Automatic Grading Tool for Jupyter Notebooks in Artificial Intelligence Courses. Sustainability. 2021; 13(21): 12050.
15. Kaggle. Kaggle API Documentation. Available online: https://www.kaggle.com/docs/api (accessed on 2 December 2024).
16. Amazon Bedrock. Amazon Bedrock Documentation. Available online: https://docs.aws.amazon.com/bedrock/ (accessed on 4 December 2024).
17. Amazon Web Services. AWS Certified Cloud Practitioner. Available online: https://aws.amazon.com/certification/certified-cloud-practitioner/ (accessed on 4 December 2024).
18. Guskey TR. Special Topic/The Case Against Percentage Grades. Educational Leadership. 2013; 71(1): 68–72.

19. National Center for Education Statistics, 2025. Scale Scores and NAEP Achievement Levels. Available online: https://nces.ed.gov/nationsreportcard/guides/scores_achv.aspx (accessed on 4 December 2024).
20. Bland JM, Altman DG. Statistical methods for assessing agreement between two methods of clinical measurement. The Lancet. 1986; 327(8476): 307–310.

Appendix

To access the code associated with this research, please visit the following repository: https://github.com/SkywardAI/hackathon-leaderboard.