

An AI-Anchored Pathway into HPC Practice: Design and Initial Evaluation of a Three-Day Summer School in Vietnam

Thanh Hoang Le Hai^{*†§}, Thin Nguyen Manh^{*†§}, Huy Nguyen Tuan^{*†}, Nhan Nguyen Phuc^{*†},
Thanh Nguyen Van Cong[†], Duong Nguyen Anh[†], Van Vo Tran Thi My[‡], Trang Nguyen Thi Thuy[‡],
Luong Phan Hao[†], Hao Thoai[†], Nhan Tran Thien[†], Khanh Vo Huy[†], Dang Tran Hai[†],
Nhien Ho Minh[†], Duc Le Dinh[†], and Nam Thoai^{*†§}

^{*}High Performance Computing Laboratory, Ho Chi Minh City University of Technology (HCMUT), VNU-HCM

[†]Faculty of Computer Science and Engineering, Ho Chi Minh City University of Technology (HCMUT), VNU-HCM

[‡]School of Industrial Management, Ho Chi Minh City University of Technology (HCMUT), VNU-HCM

{thanhhoang,nmthin,huy.nguyentuank22,nhan.nguyen2005phuyen, thanh.nguyen2313133,duong.nguyen14112005,
van.votran544,trang.nguyen165,luong.phanhaoluong,hao.thoaihao, nhan.tranthiennhan,khanh.vohuy,dang.tranhai2005,
nhien.hominh,duc.ledinh,namthoai}@hcmut.edu.vn

[§]Corresponding authors

Abstract—High Performance Computing (HPC) education increasingly serves students entering through artificial intelligence and data-intensive applications, despite uneven preparation in parallel programming and shared computing environments. This paper presents the design and initial evaluation of a three-day, practice-oriented HPC summer school for 36 selected students, of whom 34 attended at least one session, using a real Slurm cluster. The curriculum connected AI workloads with cluster access, resource management, OpenMP and MPI, GPU execution, reproducible environments, and an AI/HPC workflow challenge. We analyzed participation and assessment records, 25 post-school survey responses, and operational observations. Slurm and data-intensive AI workflows received the highest perceived-practicality ratings (4.12/5 and 4.08/5), while parallel programming presented the greatest knowledge load (3.36/5). The findings highlight the importance of teaching-assistant support, explicit instructional scaffolding, and reliable infrastructure, providing practical design lessons for institutions developing short, hands-on HPC training programs.

Index Terms—HPC education, parallel and distributed computing education, AI infrastructure, hands-on learning, summer school, workforce development

I. INTRODUCTION

Students increasingly encounter compute-intensive practice through deep learning, large language models, computer vision, and data analytics. These application areas create interest in larger computing resources, but familiarity with an AI framework or cloud notebook does not automatically prepare a student to work on a shared HPC system. Productive cluster use requires a different set of practices: command-line access, batch scheduling, explicit resource requests, software-environment management, interpretation of job states and logs, and reasoning about how parallelism and hardware choices affect performance. This gap is particularly visible in institutions where HPC is available through a research laboratory but is

not yet embedded as a complete pathway in the undergraduate curriculum.

HPC education has long addressed the barrier created by unfamiliar parallel computing environments. Scaffolded portals can lower the initial cost of accessing such environments [1], problem-based courses can connect accelerator programming to concrete computational problems [2], and curriculum integration studies show that HPC concepts can be introduced progressively across undergraduate programs [3]. However, contemporary learners also need to understand reproducible environments, accelerators, operational AI workflows, and the support practices surrounding shared infrastructure. Recent EduHPC work on GPU-based AI workflows and operational machine learning illustrates both the value and the delivery cost of authentic systems-oriented activities [4], [5].

HCMUT HPC Summer School 2026 was designed in this context. The program used student interest in AI as an entry point, but its instructional target was broader: to help learners begin operating responsibly on shared HPC infrastructure and connect application-level goals to resource-aware execution. Over three days, the School moved from cluster access and Slurm to OpenMP/MPI, performance trade-offs, GPU jobs, Apptainer, LLM serving, and a team challenge involving an AI workload evidence corpus. The first edition was intentionally compact and community-supported, combining faculty instruction, instructors, student teaching assistants, an LMS, and direct access to a Slurm cluster.

The study addresses three evaluation questions:

- RQ1: Which components of the AI-anchored HPC curriculum did participants perceive as most practical and immediately applicable?
- RQ2: What instructional and operational factors enabled or constrained hands-on participation on shared HPC infrastructure?

- RQ3: What mismatches emerged between the intended curriculum design and students’ experience within the intensive three-day format?

This paper makes three contributions:

- It presents an AI-anchored curriculum that progresses from cluster access and parallel programming to resource-aware and reproducible AI execution on shared HPC infrastructure.
- It documents a community-supported delivery model combining faculty instruction, student teaching assistants, an LMS, and authentic cluster access within a three-day format.
- It provides a multi-source initial evaluation identifying the components participants considered most practical and the scaffolding, staffing, and infrastructure constraints that shaped delivery.

The paper does not claim that the School caused measurable competence gains or outperformed a conventional HPC course. No objective paired pre/post assessment or comparison condition was available. Instead, it treats the first edition as an initial education-research case study from which bounded and transferable design lessons can be drawn.

II. RELATED WORK

A. Lowering the Entry Barrier to HPC and PDC

A persistent challenge in parallel and distributed computing education is that students must learn both computational concepts and an unfamiliar execution environment. OnRamp was designed to let students launch parallel applications early while gradually learning the native environment [1]. This separation between initial access scaffolding and eventual native practice is relevant to short HPC programs: an environment must be accessible enough for a learner to begin, but the scaffolding should still expose authentic concepts such as queues, resource requests, and job states. Neelima and Li showed how shared-memory, distributed-memory, accelerator, and hybrid programming concepts can be introduced progressively in an engineering curriculum [3]. Their longitudinal setting differs from a summer school, but it highlights that breadth must be sequenced rather than presented as a collection of unrelated tools.

B. Experiential Activities and Assessment

Problem-based and experiential approaches make parallel computing concepts concrete by tying them to observable computational behavior. Geist et al. used a substantial real-world problem as a vehicle for GPU design and performance reasoning [2]. Auto-grading work for parallel programs also demonstrates the need for assessment mechanisms that can handle nondeterminism and systems-level behavior rather than only final output correctness [6]. At a larger scale, the Student Cluster Competition immerses teams in the interdisciplinary and operational character of HPC [7]. These studies motivate the use of real infrastructure and a team activity, but they also imply that the observable student artifact and the concept

being assessed must be explicit. A challenge cannot be treated as evidence for every curriculum outcome simply because it occurs at the end of the program.

C. AI Workloads and Operational Computing as Educational Context

AI workloads offer a contemporary application context for teaching accelerators, performance analysis, and scalable execution. Srinivasan et al. describe a course that moves from GPU foundations toward AI workflow development on cloud GPU resources [4]. Fund et al. examine the substantial infrastructure and operational burden involved in teaching operational machine learning [5]. Recent work on modular workforce-development pathways similarly emphasizes reusable units and flexible learning structures [8]. Together, these studies suggest that AI can increase relevance, while access, reproducibility, and operational reliability remain part of the pedagogy rather than peripheral logistics.

AI is becoming part of HPC execution beyond model training: HPC centers now provide LLM inference services [9], while research connects LLM agents to parallel scientific workflows [10]. Previous educational initiatives include GPU-based AI workflow development [4], studies of the infrastructure costs of teaching operational machine learning [5], and hands-on agentic-workflow training on supercomputing infrastructure [11]. Our three-day undergraduate pathway differs in its instructional scope: it introduces native shared-cluster operation and parallel-computing foundations before connecting resource-aware GPU execution, reproducible environments, and a bounded LLM-workflow challenge. The evaluation also examines the alignment between intended learner activities and observable artifacts, alongside perceived practicality and operational barriers during delivery. We present this specific combination of curriculum design and evaluation evidence, rather than claiming to be the first to introduce AI or agentic workflows into HPC education.

III. PROGRAM CONTEXT AND DESIGN

A. Context, Cohort, and Constraints

The School was held from 17 to 19 June 2026 at HCMUT, VNU-HCM. It was organized by the High Performance Computing Laboratory and Big Data Club with support from the Faculty of Computer Science and Engineering, instructors, student organizers, and external speakers. The call received 125 valid applications from 11 institutions. Thirty-six students were selected to keep direct cluster access and live support manageable. Thirty-four attended at least one half-day session, 32 had recorded assessment scores, and 31 met the issued-certificate requirement of attending at least four of the six half-day sessions. These figures describe participation and should not be interpreted as evidence of learning. Table I summarizes the main cohort, infrastructure, instructional, and assessment characteristics of the first edition.

The applicant pool was dominated by students interested in AI/ML and HPC/parallel computing, but the records do not provide a complete pre-program profile for only the

TABLE I
PROGRAM AND DELIVERY CONTEXT.

Dimension	First-edition setting
Duration	Three days; six half-day blocks
Selection	125 valid applications from 11 institutions; 36 selected
Participation	34 attended at least one block; 31 met the certificate attendance requirement
Infrastructure	Real shared Slurm cluster with CPU and NVIDIA V100 GPU resources; VPN/SSH access; Apptainer; LMS
Instruction	Faculty-led lectures, guided labs, instructors, student teaching assistants, and team support
Assessment	Individual LMS activities plus a team AI/HPC workflow challenge
Primary constraint	Broad curriculum and heterogeneous preparation within a three-day format

selected cohort. The post-school survey asked respondents to retrospectively rate their prior confidence. The cleaned responses suggested relatively stronger preparation in Linux and AI/ML, moderate familiarity with HPC architecture, and the weakest preparation in OpenMP, MPI, and CUDA. Because the baseline was collected after the School and only from survey respondents, it is used as contextual evidence, not as a pretest.

B. Design Rationale: AI as an Anchor, Not a Substitute

The curriculum followed the theme “Start Local, Compute at Scale.” AI workloads were used as a familiar application anchor to motivate why a student should care about GPU allocation, reproducibility, profiling, and shared-resource policies. The design did not assume that AI familiarity could replace parallel-computing foundations. Instead, it attempted to connect a familiar application goal to progressively more explicit systems practices. Fig. 1 summarizes this intended pathway.

The three-day program comprised six half-day blocks containing multiple instructional sessions. The eight categories in the post-school survey grouped instructional topics for evaluation and did not map one-to-one onto the numbered teaching sessions. On Day 1, an introductory HPC lecture was followed by lecture-and-hands-on segments covering cluster access and Slurm resource management, OpenMP and MPI, and performance trade-offs. Day 2 combined a computer-vision lecture with lecture-and-hands-on segments on resource-aware GPU requests and reproducible execution with Apptainer, followed by a lecture and demonstration on LLM serving and multi-agent infrastructure. Day 3 centered on a team AI/HPC workflow challenge, followed by student presentations and talks connecting participants with broader HPC communities. Table II specifies the intended learner activities and observable artifacts associated with the hands-on components.

C. Community-Supported Delivery

The delivery model treated human support as part of the intervention. Faculty and instructors introduced concepts and led technical sessions. Teaching assistants supported account

access, VPN and SSH setup, Slurm jobs, code execution, LMS activities, and hackathon troubleshooting. A student committee handled technical preparation, participant support, documentation, and event operations. This layered model was necessary because a failure at any point, from account provisioning and network access through environment setup, job submission, output discovery, and LMS submission, could prevent a learner from completing the intended conceptual task.

IV. LEARNING ACTIVITIES AND ASSESSMENT ALIGNMENT

The School combined lectures, demonstrations, guided execution, parameter changes, and reflection questions. Table II links each intended learning outcome to a student activity and its observable evidence. For example, the Slurm exercise involves editing a batch script, submitting a job, and inspecting its state and output log.

A. Guided Cluster and Parallel-Computing Activities

Students accessed the GITC HPC Cluster from their personal laptops through VPN and SSH. For example, one guided CPU smoke test specified two CPU cores and 2 GB of memory; students were instructed to submit the batch script, monitor the job using `squeue`, and inspect its output log. Subsequent exercises introduced resource-allocation changes, hardware inspection, job arrays, GPU allocation checks, and common failure scenarios, including invalid partitions, out-of-memory errors, and timeouts. The OpenMP materials introduced compilation with `-fopenmp`, execution with different thread counts, parallel regions, work sharing, and data-sharing constructs. The MPI materials used code examples to introduce process ranks, point-to-point communication, and collective operations. These activities emphasized introductory execution and configuration rather than comprehensive coverage of parallel programming.

This progression was only partially realized in the first edition. Survey feedback showed that lab sheets were generally usable, but a majority of respondents reported missing intermediate steps. In a local programming exercise, a missing step may be recoverable through trial and error. On a shared cluster, it can prevent the student from reaching the measurement task at all. The first delivery therefore revealed that environment checks, expected outputs, common errors, and file locations must be specified as part of the educational material.

B. Reproducible AI Execution

In this School, resource-aware execution meant selecting and adjusting Slurm requests for CPUs, GPUs, memory, or time using workload requirements and observed execution evidence. Reproducible execution meant packaging and reusing a software environment with Apptainer while keeping workload code and persistent outputs identifiable. Guided exercises connected these practices to a computer-vision workload and introduced their relevance to subsequent LLM workflows. The emphasis was on making resource and environment choices

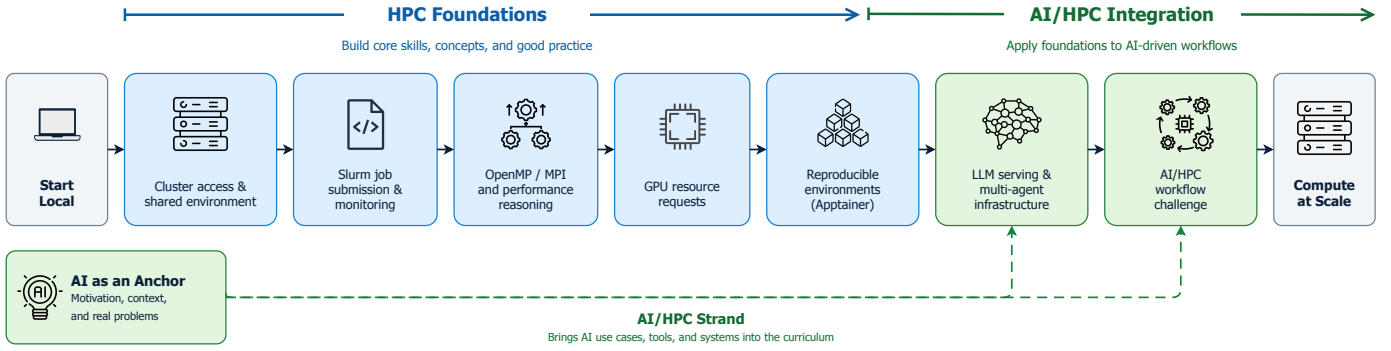


Fig. 1. AI anchored pathway into authentic HPC practice, progressing from cluster access to the final AI/HPC workflow challenge. The final challenge integrates the AI/HPC workflow strand rather than all curriculum outcomes.

TABLE II
INTENDED LEARNING OUTCOME–ACTIVITY–ARTIFACT–ASSESSMENT MAPPING.

Intended outcome	Student activity	Observable artifact	Evidence/assessment
Navigate a shared HPC environment	Connect through VPN/SSH; inspect files, modules, and partitions	Successful environment check and command output	Guided lab/LMS completion
Submit and monitor batch jobs	Edit a Slurm script; use sbatch and squeue; inspect logs	Script, job identifier, state, and output log	LMS activity and TA observation
Reason about parallel execution	Vary OpenMP threads or MPI processes and compare behavior	Runtime observations and short explanation	Lab response/quiz
Make resource-aware AI requests	Change CPU/GPU, memory, or time requests for a provided workload	Request configuration and observed execution behavior	Guided exercise
Use a reproducible environment	Run a supplied workload with Apptainer	Container command/configuration and reproduced output	Lab completion
Interpret performance evidence	Read runtime, utilization, or profiling information	Measurement record and interpretation	LMS prompt
Analyze an AI workload through an LLM workflow	Configure agents over a frozen evidence corpus	Answers, evidence references, execution trace, and runtime	Team hackathon rubric

explicit, not on demonstrating optimal resource allocation or bitwise-identical results.

The Day 2 GPU lab served as a smaller preparatory exercise for the Day 3 AI/HPC workflow challenge. Using a ResNet18 training workload, students were instructed to run two baseline configurations with batch sizes of 16 and 32, followed by a configurable third run. They then compared CSV summaries, including throughput and GPU-memory measurements, to justify a resource request based on observed evidence. This exercise introduced workload execution, performance interpretation, and resource-aware decision-making before the more complex team challenge. The GPU component did not include CUDA kernel development; its focus was on using shared GPU infrastructure effectively rather than implementing GPU kernels.

A subsequent LLM-workflow exercise reused the ResNet18 source package and performance evidence to introduce evidence-grounded resource prediction, bridging direct workload measurement and the final team challenge.

The Apptainer component extended reproducible execution beyond the use of prebuilt containers. Two guided labs introduced a progression from pulling an OCI image into the Singularity Image Format (SIF) and running it with `shell`, `exec`, and `run`, to constructing a custom SIF from a definition

file using `--fakeroot`. Learners were instructed to compare user identities inside and outside the container, inspect the read-only behavior of SIF, and use a bind mount to persist output on the host. Additional examples connected containerized execution to MPI, including instructions for running a Hello MPI program with multiple processes. The emphasis was on understanding how software environments are packaged and executed in HPC settings, rather than treating container use as a black-box deployment step.

C. AI/HPC Workflow Challenge

The final team activity was the culminating activity of the AI/HPC workflow strand. Students did not train the target image model. Each team received a frozen evidence corpus derived from an image-training workload, including source code, logs, runtime observations, and supporting artifacts. Teams configured a multi-agent LLM workflow that analyzed this corpus and answered workload-analysis questions. The submitted outputs included answers, references to supporting evidence, an execution trace, and runtime information. The activity required teams to execute and coordinate an AI workflow on the provided infrastructure while reasoning from recorded workload evidence.

The challenge was assessed at team level using criteria related to answer correctness, evidence use, workflow behavior, and execution. Its role in this paper is deliberately bounded. It provides evidence about participation in the AI/HPC strand and about operational barriers during a time-constrained team task. It does not directly assess mastery of OpenMP, MPI, or every outcome in Table II. Team composition, prior AI experience, and uneven attendance further limit any individual-level interpretation of the scores.

D. Assessment Model

The program combined individual LMS performance and a team hackathon score. When both components were available, the score reported on a ten-point scale was calculated as

$$\text{Final}_{10} = \frac{0.6 \times \text{LMS}_{100} + 0.4 \times \text{Hackathon}_{100}}{10}.$$

The LMS component included session activities and quizzes; the hackathon component was shared by team members. The composite score was used for program administration, not as a validated measure of overall HPC competence. LMS incidents, team-level scoring, and incomplete participation weaken its suitability as a research outcome, so the present analysis reports assessment records only as descriptive context.

V. EVALUATION METHODOLOGY

A. Data Sources

The evaluation triangulated four sources. First, application, selection, and attendance records described demand and the participation funnel. Second, LMS and hackathon records described whether students and teams produced the expected administrative submissions. Third, a voluntary post-school survey collected retrospective prior confidence, practicality and knowledge-load ratings across eight instructional topic categories, and feedback on labs, the hackathon, support, and infrastructure. The instrument combined five-point ratings, categorical and multiple-selection questions, and open-ended responses. The questionnaire and documented response scales are available in the public survey instrument (https://github.com/hpcc-hcmut/hpc-summer-school-leaderboard/blob/main/evaluation/post_event_survey.md); the accompanying Artifact Description covers the released resources. Fourth, internal operational records documented access, LMS, and delivery incidents. The program agenda and teaching materials were used to reconstruct the intended intervention and its outcome–activity alignment. Table III maps each research question to its primary evidence and interpretation boundary.

B. Survey Cleaning and Analysis

The survey workbook contained 26 submissions. One near-identical repeat submission, differing only in an optional free-text response and timestamp, was removed, producing 25 analyzed response records. The remaining submissions formed the cleaned analysis set. Because participation in the post-school survey was voluntary, the resulting sample should be

interpreted as respondent-level evidence rather than complete coverage of the full cohort.

We report descriptive counts, selection frequencies, and arithmetic means for numerical ratings. Session practicality and knowledge load were rated on five-point scales (1–5). For knowledge load, 3 indicates an appropriate amount of material, while higher values indicate greater perceived overload rather than better performance. Practicality measures perceived applicability, not demonstrated proficiency. We do not conduct significance testing because the sample is small, the cohort was selected, the survey was post-only, and no comparison condition was available. Percentages use the relevant denominator stated with each result.

VI. RESULTS

A. RQ1: Perceived Practicality and Immediate Applicability

As shown in Fig. 2, Slurm Job Scheduler received the highest perceived-practicality rating at 4.12/5, followed closely by Data-Intensive & AI Workflows at 4.08/5. HPC Workflows and Reproducible Environments each averaged 3.80/5, followed by Performance Measurement & Profiling at 3.72/5, GPU/CUDA at 3.68/5, Parallel Programming Models at 3.60/5, and HPC Systems at 3.52/5. The differences are descriptive rather than inferential, but the pattern remains consistent with the skills respondents most frequently identified as immediately useful, especially Linux CLI and Slurm job management, AI workflows at scale, reproducible environments, and OpenMP/MPI or debugging and optimization.

These findings indicate that respondents most readily recognized the value of operational cluster skills and an application context already connected to their interests. They do not show that students became more proficient in these areas. CUDA was selected less often as an immediately useful skill, which is consistent with the short duration and the retrospective baseline showing weaker prior preparation in parallel programming. CUDA may remain relevant as exposure, but the first edition did not provide evidence that kernel-level proficiency was a realistic outcome.

B. RQ2: Instructional and Operational Enablers and Constraints

Teaching-assistant enthusiasm and technical capability remained among the most positively rated instructional aspects, and technical support during the hackathon was also rated favorably. These responses, together with the range of observed access and execution issues, suggest that the support layer was central to keeping learners engaged with the intended tasks. The result should not be interpreted as proof that a particular staffing ratio caused better learning, but it identifies human support as a resource that future adopters must plan and cost.

Lab-sheet scaffolding was the clearest material constraint. Most respondents described the sheets as generally clear but missing intermediate steps; a smaller group found them very clear and step-by-step, while isolated feedback noted that some code did not match the actual environment. Infrastructure access (including VPN, SSH, server, and workspace stability)

TABLE III
EVALUATION QUESTIONS AND EVIDENCE.

Question	Primary evidence	Interpretation boundary
RQ1: Perceived practicality	Session practicality ratings; immediately useful skill selections	Perceived applicability, not demonstrated proficiency
RQ2: Enablers and constraints	TA-support ratings; lab clarity; infrastructure and LMS feedback; operational records	Conditions shaping participation, not causal effect sizes
RQ3: Design–experience mismatch	Knowledge load; pace; hackathon duration/rubric; open responses	First-edition design lessons for similar short formats

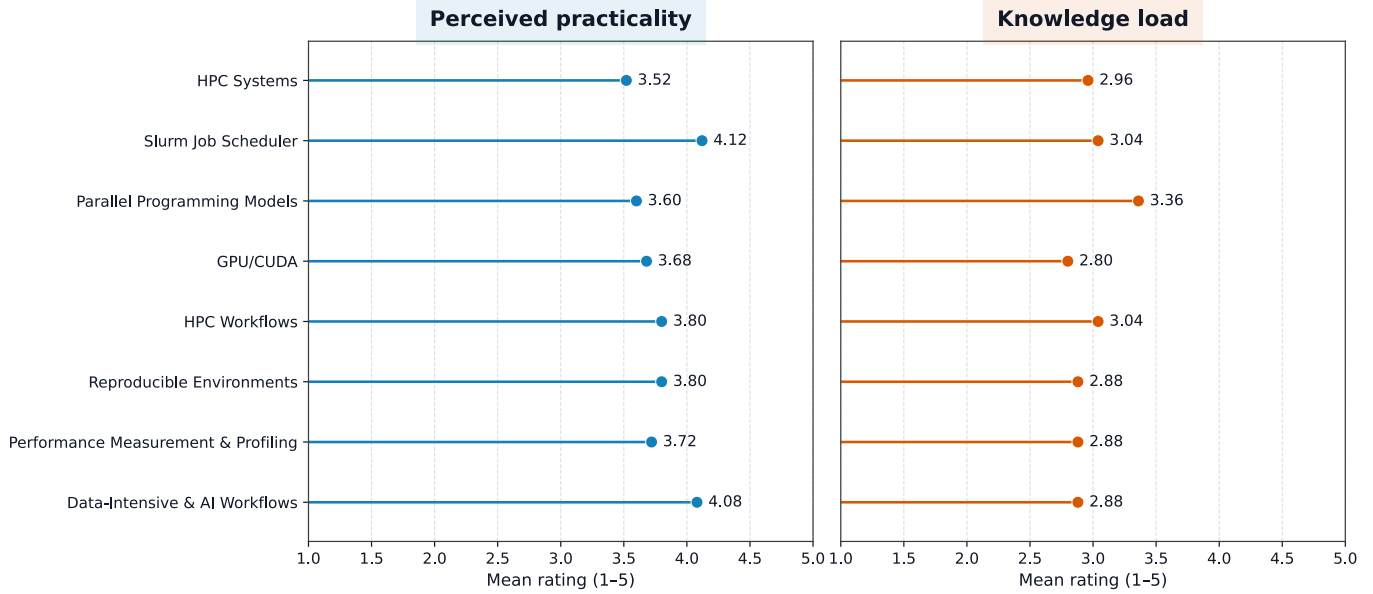


Fig. 2. Mean perceived practicality and knowledge-load ratings across eight topic categories ($n = 25$; both scales: 1–5). Practicality indicates perceived applicability; for knowledge load, 3 represents an appropriate amount of material, while higher values indicate greater perceived overload.

received a moderate rating. LMS materials and activities were widely used, but open responses identified quiz-submission problems, score-visibility issues, lag during simultaneous use, and concerns about generated explanations. When students needed help with missing lab steps, cluster access, or LMS submissions, they had less time within the three-day schedule to run alternative configurations, collect measurements, and interpret the results.

Participation remained high at the program level: 34 of 36 selected students attended at least one half-day block, and 31 met the certificate attendance threshold. These figures show that most selected students remained connected to the program, but attendance alone cannot distinguish motivation, support quality, selection effects, or learning. The evidence supports a claim of operational retention, not instructional effectiveness.

C. RQ3: Mismatches in the Three-Day Design

The principal mismatch was between curriculum breadth and heterogeneous readiness. As Fig. 2 also shows, knowledge load was closest to balanced for HPC Systems (2.96/5), Slurm (3.04/5), and HPC Workflows (3.04/5). Parallel Programming Models was the heaviest session at 3.36/5, while GPU/CUDA had the lowest mean load at 2.80/5. Containers, Performance

Measurement & Profiling, and AI Workflows each averaged 2.88/5. Pace responses nevertheless remained mixed: many respondents considered the theory–practice balance appropriate, while others found it slightly fast or wanted more advanced cases. The program was therefore near the density limit of a common path that attempted to serve both beginners and stronger students.

The second mismatch concerned the relationship between foundations and the final challenge. Most respondents regarded the hackathon as either challenging but appropriate or well matched to the knowledge covered in the School. However, open feedback also requested stronger OpenMP/MPI connections. This supports the interpretation that the challenge primarily integrated the AI/HPC workflow strand rather than the full curriculum. Several respondents wanted additional time, and rubric clarity remained a recurring concern. Thus, a positive difficulty assessment coexisted with weaknesses in time allocation and assessment transparency.

The third mismatch was between authentic access and operational reliability. Real Slurm and GPU resources made job states, resource requests, and environment constraints visible. The same authenticity introduced failure modes that a short program had little time to absorb. The first edition relied

on live TA intervention to compensate for incomplete lab scaffolding and access instability. That model worked for a small selected cohort, but it may not scale without improved materials, earlier account testing, and explicit support lanes.

VII. DISCUSSION AND DESIGN LESSONS

A. AI Familiarity Motivates Entry but Does Not Replace Foundations

The strongest perceived-practicality results were attached to Slurm and AI workflows. This combination supports the central design choice: an AI application can make resource management immediately meaningful. At the same time, the weakest retrospective preparation and heaviest knowledge load appeared around parallel programming. An AI anchor should therefore create motivation and continuity across the pathway, not compress or bypass OpenMP, MPI, performance measurement, and the mental model of shared resources. OpenMP and MPI introduced parallel execution and scaling trade-offs that were intended to provide a conceptual foundation for reasoning about workflow decomposition, coordination overhead, and resource costs in the multi-agent challenge. However, the challenge did not directly assess students' ability to apply OpenMP or MPI. Future editions should make this conceptual connection explicit while assessing foundational parallel-computing and AI-workflow outcomes separately.

B. Infrastructure Reliability Is Part of the Pedagogy

Authentic infrastructure has a dual role. It creates the educational value of seeing real queues, allocations, logs, software dependencies, and resource contention. It also makes access and environment failures part of the learner experience. The first edition shows that infrastructure preparation cannot be relegated to event logistics. Account provisioning, external VPN/SSH testing, a minimal known-good job, expected outputs, and fallback resources must be designed alongside the lesson. In a short program, the operational path to the concept is part of the concept's teachability.

C. Human Support Enables Heterogeneous Cohorts but Limits Scale

The high TA-support rating indicates that participants valued immediate help across several technical layers. Such support is especially important when students have uneven Linux, parallel-programming, and infrastructure experience. However, a support-intensive model cannot be assumed to scale automatically. Future implementations should publish a support plan covering accounts and network access, Slurm, software environments, GPU workloads, LMS activities, and hackathon submissions, with an escalation route to an instructor or cluster administrator. Materials should be improved not to remove human support, but to reserve it for conceptual and non-routine problems.

D. A Three-Day Format Supports Exposure, Not Broad Mastery

The three-day format is intended as an introductory pathway for learners who need initial exposure to shared HPC infrastructure, rather than a substitute for a semester-long course. It allows guided practice across several topics but cannot establish mastery or sustained learning. Institutions considering a similar format can use the reported lessons on account preparation, lab scaffolding, technical support, and curriculum density to assess their own delivery requirements; the participant ratings and outcomes should not be assumed to transfer across cohorts or institutions. Future editions should move basic setup before the School and protect a smaller core of hands-on activities, with optional extensions for more experienced learners.

E. Transferable Implementation Pattern

For institutions adapting this model, the reusable unit is not the agenda alone. It is a package containing: (1) stated outcomes for each strand; (2) a known-good environment check; (3) lab sheets with minimal path, expected output, common errors, measurement task, and reflection question; (4) a tested account and network plan; (5) a role-based support structure; (6) an assessment artifact aligned to each outcome; and (7) an evaluation instrument that separates perceived value from demonstrated performance. The first edition used a shared Slurm cluster with CPU and NVIDIA V100 GPU resources, VPN/SSH access, Apptainer, and an LMS (Table I). Access and environment problems, incomplete lab instructions, and LMS incidents interrupted some activities, as detailed in Section VI. The School's public recap links released teaching materials, including slides, lab handouts, and tutorials (<https://hpcc.hcmut.edu.vn/news/2026-hpc-school-conducted-successfully>). The accompanying Artifact Description documents the separately released mini-hackathon and evaluation resources. These materials support inspection and adaptation, but institutions must configure their own infrastructure, accounts, and support procedures; the release does not reproduce HCMUT's site-specific environment or individual learning outcomes.

VIII. THREATS TO VALIDITY

Construct validity is limited by the use of post-program perceptions. Practicality, self-reported progress, and intended use do not measure proficiency. The retrospective baseline may be affected by response shift after exposure to the topics. The composite administrative score mixes individual LMS activity with team performance and was affected by LMS incidents; it is therefore not used as a learning-gain measure. The hackathon assessed one strand of the curriculum and cannot represent OpenMP/MPI or overall HPC mastery.

Internal validity is limited because there was no pretest, comparison group, or random assignment. Students were selected from applicants and may have been unusually motivated. Attendance, TA support, prior experience, team composition, and infrastructure conditions were not independently

controlled. Because survey responses could not be linked to individual attendance records, we did not analyze associations between participation and perceived practicality or knowledge load. One respondent submitted two materially different responses, both retained under the response-level cleaning rule; consequently, the 25 analyzed records do not represent 25 verified distinct students. The records remain a voluntary and self-selected subset of the participant cohort.

External validity is limited to one small first-edition cohort at a single institution. The program used a particular cluster, teaching team, and local student community. Institutions with different account policies, GPU availability, staff capacity, or learner preparation may observe different results. Nevertheless, the detailed intervention mapping and the explicit constraints can support analytic transfer: readers can compare their own environment against the conditions reported here rather than treating the School as a universally effective model.

IX. CONCLUSION

This paper presented the design and initial evaluation of a three-day, AI-anchored HPC summer school delivered on authentic shared infrastructure. The pathway connected cluster access and Slurm, parallel programming and performance reasoning, GPU and containerized execution, and an AI/HPC workflow challenge. Participants perceived Slurm and data-intensive AI workflows as the most practical components, while teaching-assistant support, lab scaffolding, infrastructure reliability, program density, and rubric clarity strongly shaped the hands-on experience.

The first edition supports a bounded conclusion: AI-oriented applications, real cluster access, and community-supported instruction can be combined into a compelling short entry pathway, but the educational value of that pathway depends on explicit learner actions, observable artifacts, reliable access, and adequate human support. Future editions should introduce brief, matched pre/post concept checks and rubric-scored individual practical assessments for selected foundational outcomes, such as submitting a Slurm job and interpreting its output. These results should be reported separately from team hackathon scores. Future work should also improve selected-cohort baseline data, separate core and extension paths, and evaluate whether revised scaffolding reduces support load while preserving authentic practice.

ACKNOWLEDGMENT

The authors thank the HCMUT High Performance Computing Laboratory, the Faculty of Computer Science and Engineering, Big Data Club, the student organizing committee, lecturers, instructors, teaching assistants, technical supporters, invited speakers, partners, and participating students who contributed to the first edition of HCMUT HPC Summer School. We are particularly grateful to Prof. Fabrizio Gagliardi, George V. Neville-Neil, Prof. Worawan Diaz Carballo, and Dr. Chung Thanh Minh for their invited talks and for sharing their perspectives on HPC education and the international HPC community.

The authors used OpenAI ChatGPT to assist in restructuring and revising the prose in the Abstract and all sections, based on author-prepared source materials, results, and an initial complete manuscript. All research objectives, school design, data, analyses, interpretations, preferences, and conclusions were determined by the authors. The authors reviewed all AI-assisted text and take full responsibility for the final manuscript.

REFERENCES

- [1] S. S. Foley and J. Hursey, "Onramp to parallel and distributed computing," in *Proceedings of the Workshop on Education for High-Performance Computing (EduHPC)*, 2015, pp. 1–8, art. 1.
- [2] R. Geist, J. A. Levine, and J. Westall, "A problem-based learning approach to GPU computing," in *Proceedings of the Workshop on Education for High-Performance Computing (EduHPC)*, 2015, pp. 1–8, art. 5.
- [3] B. Neelima and J. Li, "Introducing high performance computing concepts into engineering undergraduate curriculum: A success story," in *Proceedings of the Workshop on Education for High-Performance Computing (EduHPC)*, 2015, pp. 1–8, art. 6.
- [4] S. Srinivasan, H. Alabsi, R. Obeidat, N. Ponnala, and A. Zenebe, "GPU programming for AI workflow development on AWS SageMaker: An instructional approach," in *Proceedings of the SC'25 Workshops of the International Conference for High Performance Computing, Networking, Storage and Analysis*, 2025, pp. 384–392.
- [5] F. Fund, K. Keahey, C. Hammock, M. Richardson, M. Powers, and M. Sherman, "The cost of teaching operational ML," in *Proceedings of the SC'25 Workshops of the International Conference for High Performance Computing, Networking, Storage and Analysis*, 2025, pp. 393–400.
- [6] M. Aziz, H. Chi, A. Tibrewal, M. Grossman, and V. Sarkar, "Auto-grading for parallel programs," in *Proceedings of the Workshop on Education for High-Performance Computing (EduHPC)*, 2015, pp. 1–8, art. 3.
- [7] S. L. Harrell, H. A. Nam, V. G. V. Larrea, K. Keville, and D. Kamalic, "Student cluster competition: A multi-disciplinary undergraduate HPC educational tool," in *Proceedings of the Workshop on Education for High-Performance Computing (EduHPC)*, 2015, pp. 1–8, art. 4.
- [8] K. Bondar, X. Aragonés, J. Carrabina, E. De Pelecijn, A. M. Espinosa, J. I. Gómez, F. Guim, T. Margalef, J. C. Moure, L. Piñuel, P. Reynaert, and I. Rodero, "Microcredentials for open hardware and HPC workforce development: The Openchip approach with RISC-V ecosystem," in *Proceedings of the SC '25 Workshops of the International Conference for High Performance Computing, Networking, Storage and Analysis*, ser. SC Workshops '25. New York, NY, USA: Association for Computing Machinery, 2025, pp. 416–423. [Online]. Available: <https://doi.org/10.1145/3731599.3767383>
- [9] Argonne Leadership Computing Facility, "Deploying inference services at ALCF," ALCF webinar announcement, Dec. 2025, describes HPC-hosted LLM inference endpoints; webinar dated Dec. 3, 2025. [Online]. Available: <https://www.alcf.anl.gov/events/deploying-inference-services-alcf>
- [10] H. Ma, A. Brace, C. Siebenschuh, I. T. Foster, and A. Ramanathan, "LangChain-Parsl: Connect large language model agents to high performance computing resource," in *Proceedings of the SC '25 Workshops of the International Conference for High Performance Computing, Networking, Storage and Analysis*, 2025, pp. 78–85.
- [11] National Energy Research Scientific Computing Center, "NERSC Agentic AI Bootcamp, april 2026," NERSC event and training announcement, Apr. 2026, remote bootcamp held April 23–24, 2026; hands-on activities using Perlmutter GPUs. [Online]. Available: <https://www.nersc.gov/news-and-events/calendar-of-events/agentic-ai-bootcamp-apr2026>

Appendix: Artifact Description

Artifact Description (AD)

I. OVERVIEW OF CONTRIBUTIONS AND ARTIFACTS

A. Paper’s Main Contributions

- C_1 HPC-AI capstone workflow: A four-hour mini-hackathon in which student teams configure a traceable multi-agent LLM workflow to analyze evidence from an HPC training workload and its Slurm execution. The activity exposes trade-offs among answer quality, evidence quality, workflow design, and computational cost.
- C_2 Adaptable instructional framework: A reusable activity design that makes model selection and routing, agent roles, concurrency, batching, context limits, evidence verification, and resource awareness inspectable through execution traces and a shared submission and scoring interface.

B. Computational Artifacts

Two public artifacts provide complementary student-facing and instructor-facing reproduction paths. The repository URLs and release tags below identify the public artifacts; no archival DOI is claimed for either release.

- A_1 Student-facing mini-hackathon package: assignment, instructional evidence, public questions, prompts and configuration, workflow, Slurm wrapper, mock execution, sample outputs, and adaptation guidance. Release `v1.1.1-eduhpc26` in the repository <https://github.com/hpcc-hcmut/hpc-school-mini-hackathon>
- A_2 Instructor-facing scoring, leaderboard, and evaluation package: submission API, scorer, leaderboard and administration interfaces, public evaluation material, deployment guidance, and the post-school survey instrument. Release `v1.1.2-eduhpc26` in the repository <https://github.com/hpcc-hcmut/hpc-summer-school-leaderboard>

Artifact	Contributions	Related paper elements
A_1	C_1, C_2	Capstone design and workflow (Secs. II–III); reusable instructional package (Sec. V).
A_2	C_1, C_2	Submission, scoring and evaluation workflow (Sec. III); reusable evaluation materials (Sec. V).

The releases enable workflow inspection, execution, and adaptation, but do not promise exact reconstruction of every classroom run or historical score. The instructor-facing artifact includes private competition questions and reference answers for scoring and evaluation; these materials should not be distributed to students during a live event. Identifiable student submissions, survey responses, and production data are excluded.

II. ARTIFACT IDENTIFICATION

A. Computational Artifact A_1

Relation To Contributions

A_1 implements C_1 and C_2 . Students modify prompts, agent roles and models, routing, concurrency, batching, context limits, and verification while preserving the evidence input and output contracts. The public dual-model baseline illustrates routing. Since classroom configurations varied, the release is a reusable teaching artifact rather than an exact snapshot of every historical run.

Expected Results

The portable mock path completes the Python workflow without live LLM calls and generates `results/hackathon_submission_<job_id>.json` and `results/hackathon_trace_<job_id>.jsonl`; `local` replaces the job identifier in mock mode. The submission contains answers, file/line evidence, agent/model and runtime metadata, and the trace records workflow events. Mock mode verifies execution structure and the submission contract, not LLM answer quality.

In a configured HPC environment, a Slurm job starts an Apptainer-hosted Ollama server on a GPU node with a pre-staged model cache; the Python client calls Ollama over the node’s loopback API, executes the configured agents, generates the JSON/JSONL outputs, and may POST the submission to A_2 . Inspecting these outputs exposes the choices and verification steps underlying C_1 and C_2 .

Expected Reproduction Time (in Minutes)

After checkout, setup and mock execution typically take a few minutes, followed by a few minutes for output inspection. Records from the local cluster accounting log show that jobs took approximately 1.5-6 minutes with staged images and models; cluster queueing, initial image/model provisioning, and site-specific configuration are excluded. Full-path setup time varies across cluster configurations.

Artifact Setup (incl. Inputs)

Hardware: Mock mode runs on an ordinary workstation and requires neither a GPU nor an HPC cluster. The historical classroom job requested one Slurm node, one task, four CPUs, one GPU, and a 15-minute walltime; a 32-GB NVIDIA V100 was allocated. An alternative CUDA-capable GPU may work if its memory accommodates the selected models, context, and concurrency. No multi-node interconnect is required.

Software: The mock path requires Python 3.10+ and a writable results directory, but not Slurm, CUDA, Apptainer, Ollama, or model weights. The full path requires a working Slurm installation, Apptainer with GPU passthrough, Ollama, compatible container images, and pre-staged models. The released baseline names `llama3.2:3b` and `qwen3:4b`. Site-specific Slurm, Apptainer, Ollama, and CUDA versions

are not fixed by this portable release; adopters should use compatible versions documented in their local deployment and validate GPU passthrough before execution. The client uses Ollama’s `/api/generate` API; OpenAI-compatible endpoints require an adapter. See the A_1 repository for source code and setup instructions.

Datasets / Inputs: Evidence is included under `evidence/repo/` and `evidence/results/`: source and configuration files, Slurm scripts, and saved benchmark summaries for the instructional HPC training workload. Public questions are supplied in `qa_public.json`. These are instructional inputs and are not student response data. Private competition questions and reference answers are not included in A_1 ; instructor-facing scoring materials are provided in A_2 . Original student submissions and production datasets are not included in A_1 .

Installation and Deployment: Clone or download the A_1 tagged release, retain its directory structure, and use Python 3.10+ to execute the portable mock check below. For the cluster path, an instructor must stage container images and model caches, supply site-specific partition/QoS and paths, and configure `CLIENT_IMAGE`, `SERVER_IMAGE`, `OLLAMA_CACHE`, and `ALLOWED_MODELS_FILE` before invoking the provided Slurm script. No local compilation is specified for the mock path.

Artifact Execution

Run the portable check from the repository root (line breaks are for readability):

```
python3 src/hackathon_workflow.py \
--source-dir . \
--config \
configs/hackathon_workflow.json \
--output-dir results \
--team-id test-team \
--team-name "Test Team" \
--mock
```

For the full cluster path, configure the institution-specific variables above and run `sbatch slurm/hackathon-parallel-workflow.slurm`.

The dependency chain is: T_1 load evidence and questions $\rightarrow T_2$ run configured evidence-extraction agents $\rightarrow T_3$ aggregate evidence, answer question batches and verify $\rightarrow T_4$ normalize answers and write submission/trace files $\rightarrow$ optional submission to A_2 .

The principal configurable parameters in `configs/hackathon_workflow.json` are model routing, maximum parallel agents, answer-batch count, context length, and temperature. They are teaching variables, not fixed parameter values for an exact rerun of the original competition. The release does not prescribe repetitions or statistical comparisons for the portable check.

Artifact Analysis (incl. Outputs)

Confirm that the JSON submission and JSONL trace are created. Inspect coverage of public questions, answer-to-file/line

evidence, recorded agent/model roles, workflow events, and runtime metadata. Compare configurations only under the same evidence, questions, and compute environment when discussing routing, concurrency, batching, or verification effects. The mock path establishes structural reproducibility, not original class scores or LLM accuracy; full-path outputs depend on available hardware, models, runtime configuration, and inference behavior.

B. Computational Artifact A_2

Relation To Contributions

A_2 consumes A_1 ’s shared submission format and implements the instructor-facing part of C_1 and C_2 : authenticated receipt and validation, evaluation against configured public or instructor reference sets, persisted results, score breakdowns, leaderboard and administrator views. Its public post-event survey instrument supports instructional adaptation and evaluation replication without sharing participant responses.

Expected Results

A deployed backend and frontend accept a valid A_1 submission through `/api/submissions`, return validity, total score, component breakdown, rank, and diagnostic data, persist the run, and expose it through leaderboard and run-detail views. The survey instrument provides the original Vietnamese questions, English translations, documented response formats, and verified rating-scale anchors. It supports reuse of the questionnaire but cannot reproduce the paper’s historical survey statistics without participant responses.

Expected Reproduction Time (in Minutes)

A Docker Compose-capable workstation or virtual machine is sufficient, with no GPU. Image build time depends on cache and connectivity; after prerequisites and configuration, service startup, health checking, and a test submission usually take a few minutes. Evaluation and inspection require additional minutes. No fixed end-to-end time is claimed across deployments.

Artifact Setup (incl. Inputs)

Hardware: A workstation or VM capable of running the backend and frontend containers and a persistent SQLite volume. No specialized HPC interconnect or GPU is required.

Software: The released Docker Compose configuration provides a Python 3.12/FastAPI backend, Next.js frontend, and SQLite storage. A supported Docker Engine with Compose is required; host-side engine and Compose versions are not prescribed by this artifact release. Consult the A_2 deployment and scoring/API documentation for compatibility and configuration details.

Datasets / Inputs: The primary input is a JSON submission conforming to the A_1 schema. The package includes public evaluation materials and instructor-facing private question and reference-answer sets. Instructors may replace these materials when adapting the artifact for a new event. The post-event survey instrument is available

at `evaluation/post_event_survey.md`. It contains the Vietnamese questions, English translations, documented response formats, and verified rating-scale anchors. Some categorical and multi-select option lists are not available in this release. Individual survey responses, identifying information, and original student submissions are not included. The public release alone does not reproduce the historical scores or survey statistics.

Installation and Deployment: Clone the tagged A_2 release, configure environment variables and a team identifier/token as directed in its deployment documentation, then run:

```
cp .env.example .env
docker compose build
docker compose up -d
```

Check the backend health endpoint once services are running. Use the supplied schema and API documentation to authenticate a team and POST an A_1 submission. For a new live event, institutions should configure their own storage and credentials and replace the included instructor-facing question and answer materials as needed; the public package does not contain production credentials.

Artifact Execution

Follow T_1 select A_1 public evidence/questions $\rightarrow T_2$ generate a JSON submission with the A_1 workflow $\rightarrow T_3$ authenticate and POST to the A_2 submission API $\rightarrow T_4$ validate and score against the configured evaluation set $\rightarrow T_5$ persist the run $\rightarrow T_6$ inspect the API response, run detail, and leaderboard. The released scoring and API documentation governs how reference material and component scores are configured. For survey reuse, retrieve the public questionnaire from the instructor-facing release and administer it separately; no student responses are supplied.

Artifact Analysis (incl. Outputs)

A successful test yields an accepted submission, validity and score fields including component breakdown, a persisted run, and a corresponding leaderboard entry. Check consistency between the API output, stored run, and frontend display. This verifies the instructor-facing processing and feedback mechanisms supporting C_1 and C_2 ; it does not reproduce the private event's final rankings or survey-derived results. For interpretation and deployment restrictions, consult the scoring/API documentation of the tagged release.