From Summative Multiple-Choice Testing to Formative Practice: Artificial Intelligence-Supported Question Generation, Cognitive Analytics and Adaptive Pacing

R. Bimberg, P. Kubica, K. Quibeldey-Cirkel

Technische Hochschule Mittelhessen (GERMANY)
IU International University (GERMANY)

Abstract: Multiple-choice (MC) exams are ubiquitous in higher education, yet most platforms provide little more than a summative score, limiting metacognitive reflection and self-regulated learning. We present *MC-Test*, a privacy-preserving web platform that turns MC testing into formative practice by integrating AI-supported item generation, structured feedback, learner-facing analytics, and adaptive pacing.

MC-Test uses a strict didactic data model that links each item to a learning objective, topic, weight, and an operationalized cognitive level (Bloom 1-3: reproduction, application, analysis). Item generation is context-engineered as a finite-state interaction and constrained to schema-valid JSON, enabling automated validation and downstream item statistics. After each session, *MC-Test* provides rationales for all options, just-in-time term clarification via a mini glossary, and exports derived learning objectives and study resources for revision.

Learner-facing dashboards triangulate performance through a Topic Performance Chart, a Cognitive Radar profile, Concept Mastery Columns, and a Topic × Cognitive Heatmap. To discourage low-effort responding (e.g., rapid guessing), configurable lockouts and reflection windows implement pedagogical control, with a time-aware Panic Mode that relaxes pacing under critical time pressure. *MC-Test* runs containerized on-premises with a local LLM backend so prompts and learner data remain within institutional infrastructure; pseudonyms support anonymous participation. A pilot usability study (SUS, N=20, mean=70.38/100, 'OK' [15]) confirms good acceptance. Open source (MIT): https://github.com/kqc-real/streamlit.

Keywords: Artificial Intelligence in Education, Multiple-Choice Question Generation, Bloom’s Taxonomy, Learner-Facing Learning Analytics, Self-Regulated Learning, Rapid Guessing Detection, Adaptive Pacing, Privacy-Preserving Local Language Models, Formative Assessment.

1. INTRODUCTION

Multiple-choice (MC) questions are widely used in university assessment because they scale well, support objective scoring, and are easy to administer in large cohorts. However, typical MC platforms provide little more than a total score and item-level correctness, offering limited support for metacognitive reflection and strategic exam preparation. Learners often receive feedback that is too coarse to inform their study strategies: they know *that* they failed, but not *why*.

Recent advances in large language models (LLMs) enable not only the automated generation of MC items but also richer internal representations of their cognitive demands. When item metadata such as topic, difficulty, and cognitive level are made explicit, MC tests can be reinterpreted as data sources for learner-facing analytics. Simultaneously, LLM-based coding assistants can substantially accelerate programming tasks and lower the barrier for domain experts to prototype tailored educational tools [1]. At the same time, studies have started to assess the psychometric characteristics of LLM-generated MCQs, including difficulty and discrimination indices. While such items can exhibit competitive measurement properties in some settings, the findings also underline the necessity of systematic item-level validation rather than assuming quality from generation alone [2].

This paper positions *MC-Test* as a concrete instantiation of these trends. It is a web-based platform integrating:

- A didactic data model with explicit cognitive-level labels.
- An LLM-based pipeline using a finite state interaction model for generating valid items.
- Learner-facing analytics visualizing performance by cognitive level.
- Pedagogical control mechanisms (dynamic cooldowns and pacing) to mitigate rapid guessing.

The platform was initially implemented using commercial LLM APIs and was later migrated to a local LLM backend via Ollama. The work is guided by the following research questions:

- RQ1: How can cognitive-level classifications be operationalized in a learner-facing assessment platform?
- RQ2: How can LLMs be embedded via robust prompt engineering to support psychometric quality checks and reliable item generation?
- RQ3: Which design considerations arise when migrating to a local LLM backend?

To address these questions, we present MC-Test as a **design and implementation study** with preliminary usability evidence (SUS, N=20, mean=70.38). While behavioral effects (rapid-guessing reduction) and learning outcomes are hypothesized and log-based evaluations planned (Section 6), the core contributions lie in the operationalized system design::

- An operational cognitive-level taxonomy (Bloom 1-3) embedded in a didactic item model and exposed through a four-view learner-facing analytics suite (Topic Performance Chart, Cognitive Radar Chart, Concept Mastery Columns, Topic × Cognitive Heatmap) (RQ1).
- A context-engineered, schema-constrained LLM item pipeline (finite-state interaction + JSON validation/repair) to support robust generation and downstream psychometric checks (RQ2).
- Design and deployment lessons from migrating from commercial APIs to a privacy-preserving local LLM backend (Ollama) in a containerized on-premises architecture, including robustness and data sovereignty considerations (RQ3).

MC-Test targets formative MC practice in STEM (Bloom 1–3), excluding higher-order tasks due to LLM limitations [3]. Usability is validated (SUS=70.38, "OK"), but learning effects require log analyses. Cognitive labels will be double-coded (kappa planned); human item review remains mandatory. Pacing is course-configurable for accessibility.

2. THEORETICAL BACKGROUND AND RELATED WORK

Cognitive Taxonomies and AI Validity: Bloom's taxonomy has long provided a framework for differentiating cognitive complexity. While LLMs excel at generating content for lower taxonomic levels (remember, understand), recent studies indicate significant reliability issues when models attempt to generate tasks for higher-order evaluation or creation without human-in-the-loop verification [3]. Consequently, automated MC generation requires strict constraints to ensure semantic validity [4].

Learner-Facing Learning Analytics & Self-Regulation: Learner-facing (often termed student-facing) learning analytics dashboards aim to provide learners with interpretable visualizations to support self-regulated learning [5]. However, research suggests that pure data visualization is often insufficient; effective systems must provide active guidance or "nudges" to translate insights into behavioral change [6]. In addition, prior work provides design and evaluation recommendations for learner-facing learning analytics dashboards [7]. *MC-Test* addresses this by combining analytics with active pacing mechanisms.

Test-Taking Effort and Rapid Guessing: In computer-based assessment, unusually short response times can indicate reduced test-taking effort (e.g., rapid guessing) rather than lack of knowledge. Response-time effort metrics and effort-moderated models use response times to flag or down-weight such responses, and provide practical guidance for setting response-time thresholds and interpreting rapid-guessing indicators in low-stakes contexts [8], [9].

Privacy in Generative AI: The use of commercial LLM APIs in education raises significant privacy concerns regarding data leakage and GDPR compliance [10]. Recent work highlights the "privacy paradox" of LLMs, suggesting that on-premises deployment (local LLMs) is a strong mitigation option for sensitive educational contexts [11].

3. SYSTEM DESIGN: THE *MC-TEST* PLATFORM

3.1 Pedagogical Requirements

MC-Test was designed for formative assessment in STEM courses. Key requirements included:

- **Cognitive transparency:** Items must be classified by cognitive level.
- **Actionable feedback:** Explanations must link to learning objectives.
- **Exam readiness:** Support for time management skills.
- **Didactic control:** Educators must retain control over item quality.

3.2 System Architecture

The application is implemented using the Python framework Streamlit and has been migrated from a cloud-based prototype to a robust, containerized microservice architecture. The entire stack is orchestrated via Docker Compose and deployed on an on-premises institutional server to ensure full data sovereignty. Key components include:

- **Data Persistence:** A PostgreSQL database replaces lightweight file-based storage to handle concurrent user sessions and complex relational data (user profiles, audit logs) reliably.
- **Privacy-Compliant Rendering:** To eliminate data leakage to external Content Delivery Networks (CDNs), mathematical formulas are rendered using a self-hosted local **MathJax** instance.

- **Local Inference Engine:** A central architectural shift is the integration of Ollama as a local LLM backend. This allows the platform to execute open-weight models (e.g., DeepSeek) entirely within the university's infrastructure, ensuring that neither student data nor prompt logic leaves the secure environment.

3.3 Didactic Data Model

The core is a strict JSON data model, with a schema excerpt shown in Fig. 1. Key fields include:

- `topic` and `learning_objective`.
- `weight` (1-3) and `cognitive_level` (reproduction, application, analysis).
- `mini_glossary`: Contextual definitions to support just-in-time learning.
- `rationales`: Detailed explanations for why each option is correct or incorrect.

3.4 Cognitive-Level Taxonomy for AI-Generated MCQs

MC-Test adopts a restricted three-level taxonomy inspired by Bloom: 1. Reproduction (Weight 1): Recall of facts, definitions, or simple algorithms. 2. Application (Weight 2): Use of known concepts in slightly varied contexts. 3. Analysis (Weight 3): Interpretation of data, comparison, or inference. Crucially, levels 4-6 (Evaluation, Creation) are explicitly excluded. Our analysis aligns with findings that current models lack the semantic grounding to reliably validate complex evaluation tasks in a closed MC format [3] and with work emphasizing that automated MC generation requires strict constraints to ensure semantic validity [4]. Restricting the system to levels 1-3 increases the verifiability of generated items and rationales and can reduce the risk of semantic hallucinations; however, human review and empirical item analysis remain necessary to ensure factual and psychometric quality.

3.4.1 Operationalization and Label Reliability

Learner-facing analytics are predicated on cognitive-level labels; consequently, *MC-Test* operationalizes the three levels with a brief rubric and a verification step. In the present workflow, the following three steps are taken. First, instructors define the learning objective and the intended cognitive level. Second, the LLM is instructed to generate an item that matches the specified level and to provide rationales aligned with the learning objective. Third, items are reviewed prior to release. To assess the reliability of the labeling process, a double-coding procedure will be implemented. This procedure will be executed by two instructors who will utilize a predetermined rubric. The interrater agreement, as measured by metrics such as percent agreement and Cohen's kappa, will be reported as part of the evaluation. This approach serves to prevent over-precise interpretations of learner-facing visualizations that lack the foundation of stable labeling.

3.5 Prompt Architecture

Item generation is controlled by a context-engineered system instruction set that frames the model as an “Interactive MCQ Generator.” Here, context engineering refers to the deliberate design of (i) an interaction protocol that elicits all generation parameters and (ii) hard output constraints that make results machine-checkable.

Concretely, *MC-Test* implements a finite-state workflow with three phases: (1) Configuration, in which the model collects required parameters stepwise (e.g., topic, target audience, item count, difficulty/weight distribution, option format, and optional curricular context) and requests explicit confirmation before generation; (2) Internal Blueprinting, in which the model performs a hidden planning and consistency-check step (e.g., validating that difficulty weights match the requested profile and that option/answer constraints are satisfiable) without exposing intermediate reasoning; and (3) Schema-Driven Output, in which the response is restricted to a single strictly parseable JSON object with explicit escaping and formatting rules (including stable question_ids and a schema_version for forward compatibility), Fig. 1.

For robustness across cloud and local backends, the JSON is subjected to automated schema and semantic validation (e.g., required fields, unique options, valid answer indices), with an optional repair loop that prompts the model to apply minimal edits when violations are detected.

Fig. 1. JSON schema for AI-generated question sets used in the web interface.

```
{
  "meta": {
    "schema_version": "1.1",
    "title": "string (from Configuration Step 1)",
    "created": "DD.MM.YYYY HH:MM",
    "target_audience": "string (from Configuration Step 2)",
    "question_count": number,
    "difficulty_profile": {
      "easy": number,
      "medium": number,
      "hard": number
    },
    "time_per_weight_minutes": {
      "1": 0.5,
      "2": 0.75,
      "3": 1.0
    },
    "additional_buffer_minutes": 5,
    "test_duration_minutes": number
  },
  "questions": [
    {
      "question_id": "Q001",
      "question": "string (Must start with '1. ', '2. ' etc.)",
      "options": ["string", "string", "string"],
      "answer": number,
      "explanation": "string (2–4 sentences)",
      "weight": number,
      "topic": "string",
      "concept": "string",
      "cognitive_level": "string (Reproduction | Application | Analysis)",
      "extended_explanation": null OR {"title": "string", "steps": ["string"], "content": "string"},
      "mini_glossary": [
        {"term": "TermKey", "definition": "Definition string"}
      ]
    }
  ]
}
```

Post-production learning objectives: For each generated question set, the LLM also proposes a small set of topic labels (typically ~10) that structure the item bank. In a second, dedicated post-production prompt, *MC-Test* derives learning objectives per cognitive level from each item, its assigned topic, and its cognitive label. The resulting objectives-phrased with Bloom-aligned action verbs-are compiled into a learner-facing study resource that summarizes the intended learning targets across topics and levels.

4. LEARNER-FACING ANALYTICS AND PEDAGOGICAL CONTROL

After completing a test, *MC-Test* provides a learner-facing analytics dashboard designed to foster metacognitive reflection. Instead of a single score, the dashboard offers four complementary visualizations that triangulate curricular coverage, cognitive processes, and concept mastery. Throughout this section, topics denote coarse-grained curriculum units, whereas concepts refer to fine-grained learning targets within a topic.

The views are designed as a diagnostic funnel from overview to detail. The heatmap supports pattern search across topics × cognitive levels (including potential avoidance), the topic bars and radar disentangle content gaps from process gaps, and the concept columns translate these patterns into concrete study targets for remediation. The intended effect is to move learners from undifferentiated self-assessments ("I'm bad at the subject") toward actionable diagnoses ("I can reproduce facts in Marketing, but I struggle with analysis questions because key concepts are missing") [12].

4.1 Topic Performance Chart

As shown in Fig. 2, the Topic Performance Chart offers a curricular perspective by aggregating results across subject areas (e.g., regression, classification, evaluation). Each bar visualizes the distribution of correct, incorrect, and unanswered items, and the x-axis labels report answered/total counts to make sparse evidence visible. This view supports prioritization of remediation by highlighting which content domains require attention and which domains may have been systematically skipped.

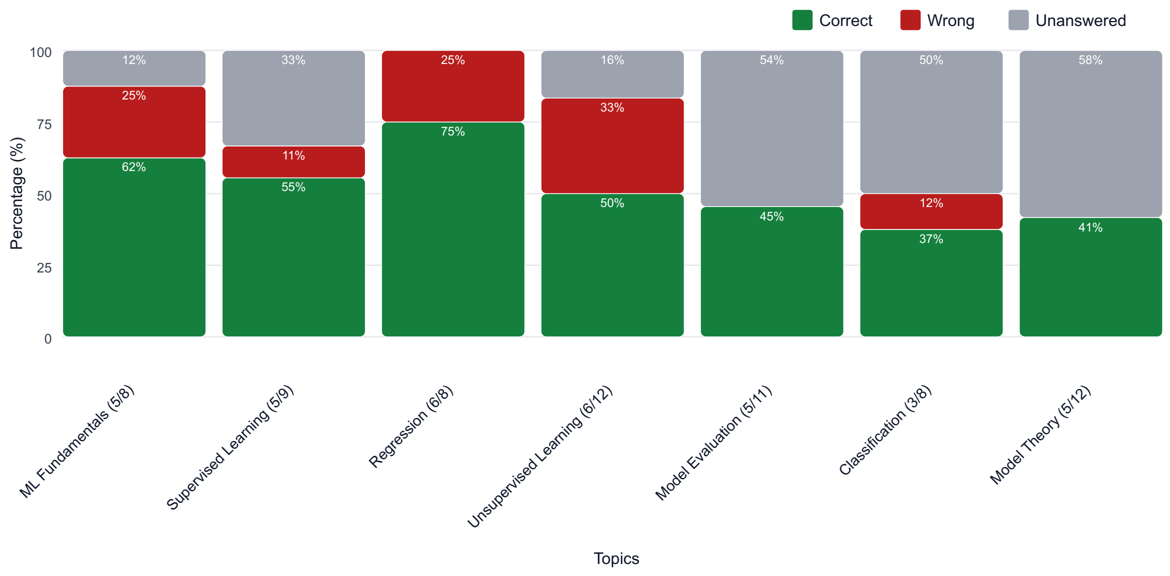


Fig. 2. Topic Performance Chart (stacked bars) summarizing learner performance across machine-learning topics. Bars show correct, incorrect, and unanswered responses; labels indicate answered/total items per topic.

4.2 Cognitive Radar Chart

As shown in Fig. 3, complementing the topic view, the Cognitive Radar Chart summarizes performance across three Bloom-aligned cognitive levels (reproduction, application, analysis), capturing whether a learner primarily recalls facts or can transfer and reason with knowledge. Balanced shapes indicate even development, whereas pronounced asymmetries signal targeted development zones. Based on characteristic patterns, the dashboard provides interpretive, non-diagnostic learner archetypes (e.g., Theorist, Practitioner, Analyst) to support reflection.

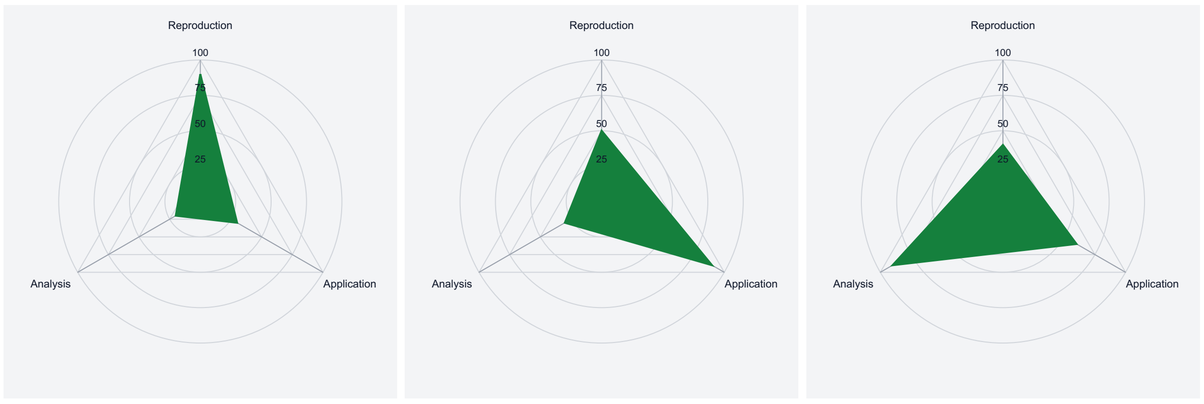


Fig. 3. Cognitive radar charts summarizing learner performance across the three cognitive levels (Reproduction, Application, and Analysis) for three learner archetypes: Crammer (left), Practitioner (center), and Theorist (right) (shown only when sufficient items per axis are available).

4.3 Concept Mastery Columns

As shown in Fig. 4, to provide fine-grained, action-oriented feedback, MC-Test groups items by tagged concepts (e.g., cross-validation, regularization) and categorizes each concept into understood, not understood, or not attempted. Concepts are marked as understood once at least 70% of their associated items were answered correctly; otherwise they are flagged to make misconceptions visible while reducing overinterpretation from single items. This micro-level view translates test outcomes into concrete study targets.

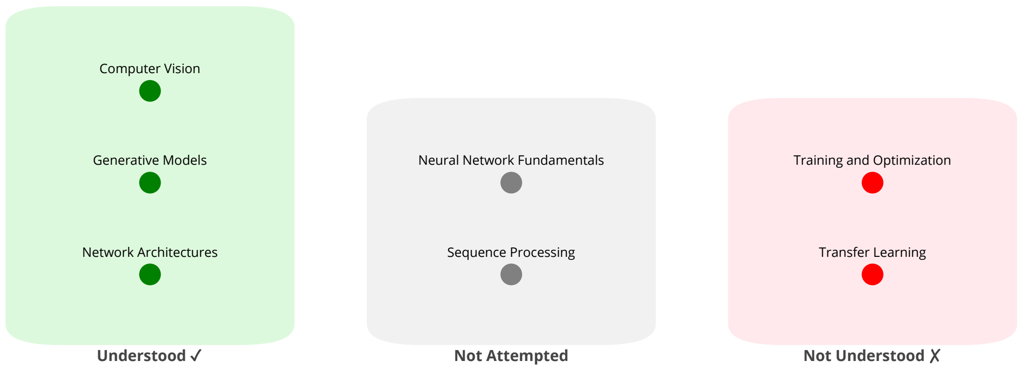


Fig. 4. Concept Mastery Columns grouping fine-grained concepts into understood (>= 70%), not understood, and not attempted categories.

4.4 Topic × Cognitive Heatmap

As shown in Fig. 5, the Topic × Cognitive Heatmap integrates curricular topics (rows) and Bloom-aligned cognitive levels (columns) in a single matrix. Cell colors encode performance while accounting for incompleteness: unanswered items are treated as 0 points, and each cell reports answered/total counts. This design discourages strategic skipping from inflating apparent competence and helps learners localize whether difficulties stem from content knowledge or higher-order processing within a topic.

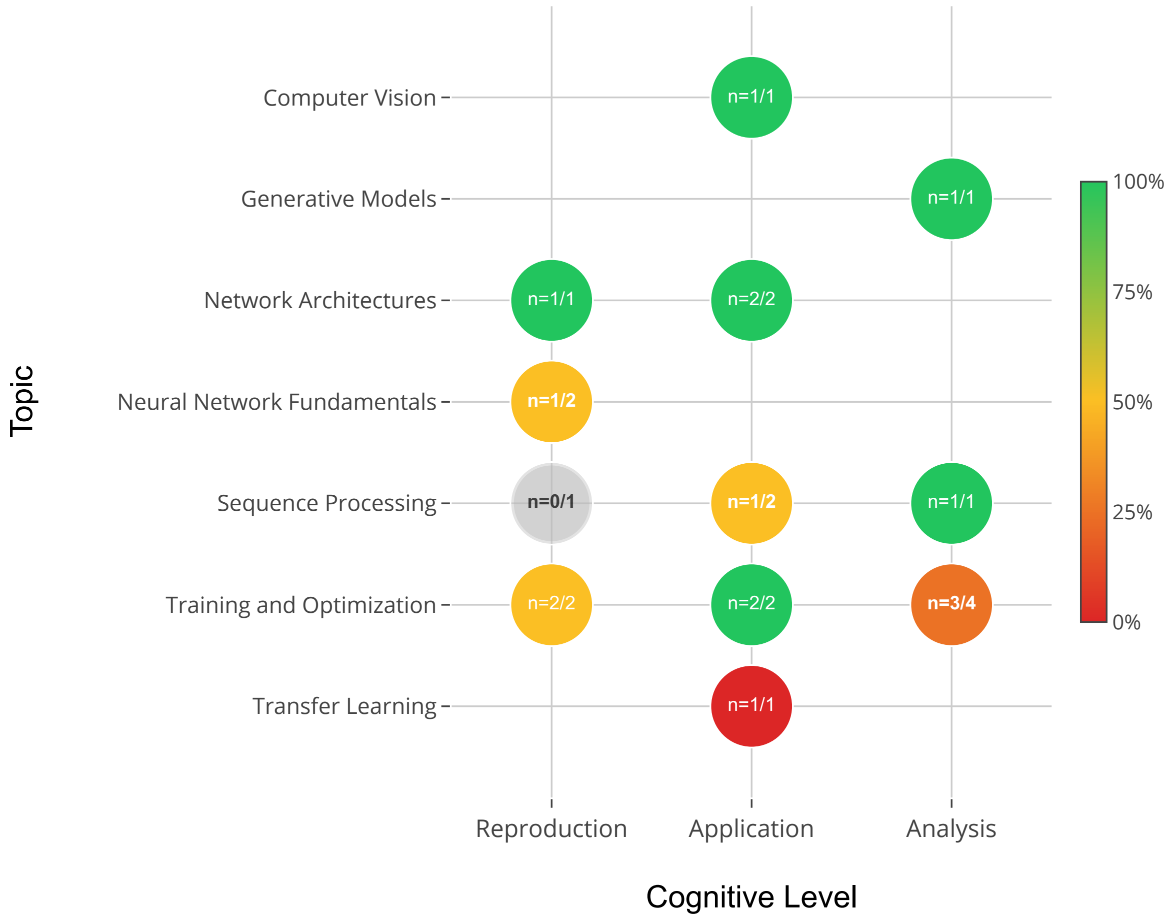


Fig. 5. Topic × Cognitive Heatmap combining topics (rows) and Bloom-aligned cognitive levels (columns). Each cell shows n = answered/total and the resulting score; unanswered items count as 0.

Taken together, the views are intentionally orthogonal: the Topic Performance Chart focuses on “what” (curricular content), the Cognitive Radar Chart focuses on “how” (cognitive processes), the Topic × Cognitive Heatmap links both, and the Concept Mastery Columns zoom into fine-grained learning targets. Triangulating these perspectives reduces blind spots and can surface strategic behavior such as systematically skipping higher-level items.

4.5 Feedback and Explanations

For complex items, explanations are extended with step-by-step reasoning. To support vocabulary acquisition, a two-tiered glossary concept is used: a mini-glossary field for immediate help and a summative PDF glossary for post-test review.

4.6 Pedagogical Control: Cooldowns and Panic Mode

A unique feature of MC-Test is its active management of learner behavior to encourage reading and reflection and to deter impulsive responding (e.g., rapid guessing). Rather than claiming to “enforce” deep processing, the platform implements configurable pacing nudges whose behavioral effects are treated as empirical questions to be evaluated.

The system implements a Pedagogical Control layer consisting of three mechanisms. **Pre-Answer Cooldown (reading support):** to increase the likelihood that students read the question before responding, the “Submit” button remains disabled for a dynamic duration upon loading a new item. This duration is calculated based on the item’s weight (complexity) and text length. **Post-Answer Cooldown (reflection support):** after submitting an answer, navigation to the next question is delayed, creating a mandatory window for reviewing the formative feedback [13]. The duration scales with the complexity of the explanation (e.g., +10 s for standard, +20 s for extended explanations). **Adaptive fairness (“Panic Mode”):** while pacing can support metacognitive reflection, it can be detrimental under time pressure.

Accordingly, the system continuously monitors the ratio of remaining time to remaining questions. If this drops below a critical threshold (e.g., <15 seconds per question), “Panic Mode” activates and overrides all cooldowns to allow the student to finish the attempt, prioritizing fairness over didactic constraint in critical moments. Because fixed pacing can disadvantage some learners (e.g., assistive-technology users or students with documented accommodations), all cooldown parameters are configurable at the course level and can be disabled or adjusted where required. Panic Mode further acts as a fairness safeguard under time pressure by prioritizing completion over pacing constraints.

5. IMPLEMENTATION AND MIGRATION TO LOCAL LLM BACKEND

The platform was migrated from commercial cloud APIs to an institutional server with a local LLM backend (Ollama) to address data protection and sustainability.

- **Prompt Refactoring:** The system prompt was adapted to local model characteristics, prioritizing JSON validity and robustness.
- **Privacy:** Local deployment ensures that no student data leaves the institution, adhering to “Privacy by Design” principles [10], [11].
- **Sustainability:** on-premises hosting mitigates the costs and rate limits associated with commercial API usage.
- **Data minimization and logging:** The deployment is designed to minimize sensitive data exposure. Only the information required for assessment and analytics is stored (e.g., answers, timestamps, aggregate scores), plus minimal account metadata and security audit logs needed to operate the system reliably. Prompts are constructed to avoid personal identifiers, and the local setup ensures that student data and prompt logic do not leave the secure environment. Access to logs and databases is restricted to authorized staff, and retention policies can be applied to limit long-term storage.
- Anonymity and pseudonyms: *MC-Test* is designed for anonymous participation. Users do not create personal accounts, and no real names are collected. At the start of a session, learners select a pseudonym from a predefined list of Nobel and Turing Award laureate names; this identifier is used only to keep session data consistent (e.g., to display progress and aggregate analytics) without enabling real-world identification of participants (see Fig. 7).

- **Open-source availability:** To support transparency and reproducibility, the *MC-Test* Streamlit application is released under the MIT License and publicly available on GitHub (https://github.com/kqc-real/streamlit) The repository includes documentation for deployment and configuration (e.g., Docker Compose) as well as the schema-driven artifacts used in the local inference pipeline.

6. EVALUATION DESIGN AND OUTLOOK

6.1 Planned Study Design

The evaluation consists of pilot deployments in STEM modules. Instruments include system log data (pacing choices, cognitive profiles, response times), pre- and post-questionnaires on metacognitive awareness and perceived usefulness, and interviews. A key next step is a comparative study design (e.g., A/B testing or within-course phased rollout) to test whether the Pedagogical Control features measurably reduce rapid-guessing indicators and whether they affect learning outcomes or study strategies. The planned evaluation also includes checks of the stability of cognitive-level labeling (double-coding) and basic item-level statistics to support psychometric interpretation of generated question sets.

To test whether Pedagogical Control changes test-taking behavior, we will derive rapid-guessing indicators from log data (e.g., item response times below item-specific thresholds, proportion of low-effort responses per learner, and changes in response-time effort before vs. after pacing is enabled) [8], [9]. Analyses will compare conditions (e.g., pacing on/off; staged rollout) while controlling for item difficulty and position effects, and will inspect downstream consequences such as feedback viewing time and option-review patterns.

To protect the validity of learner-facing analytics, we will treat cognitive-level labels as a measurable design assumption. The planned evaluation therefore includes double-coding with a rubric and reporting of interrater agreement (percent agreement and Cohen’s kappa), alongside basic item statistics (difficulty, discrimination) to identify low-quality or ambiguous items prior to interpreting cognitive profiles at scale.

Initial informal trials indicate appreciation for the cognitive-level breakdown and the transparency of AI-generated rationales. Reactions to the pacing helper were mixed, highlighting the need for the adaptive “Panic Mode,” which we subsequently implemented. These observations are exploratory and will be complemented by controlled comparisons using behavioral log metrics and learner-reported measures.

6.2 Preliminary Results: System Usability

To validate the user experience (UX) and technical acceptance of the platform, a standardized usability evaluation was conducted using the System Usability Scale (SUS) [14]. The survey was administered to a pilot cohort of 20 participants, yielding a mean SUS score of 70.38.

According to the adjective rating scale defined by Bangor et al. [15], this score places the MC-Test platform in the “OK” category, indicating good user acceptance despite the complexity of the integrated pedagogical features. Fig. 6 illustrates the distribution of individual participant scores, plotted against interpretive acceptability ranges (e.g., “Acceptable,” “Good,” “Excellent”). The visualization reveals a left-skewed distribution, suggesting that the majority of users perceived the system as easy to use.

Importantly, SUS primarily captures perceived usability; it does not establish learning benefits or behavioral effects. These aspects will be examined in subsequent studies using log-based rapid-guessing indicators, learner feedback on fairness/autonomy, and (where feasible) learning outcome measures.

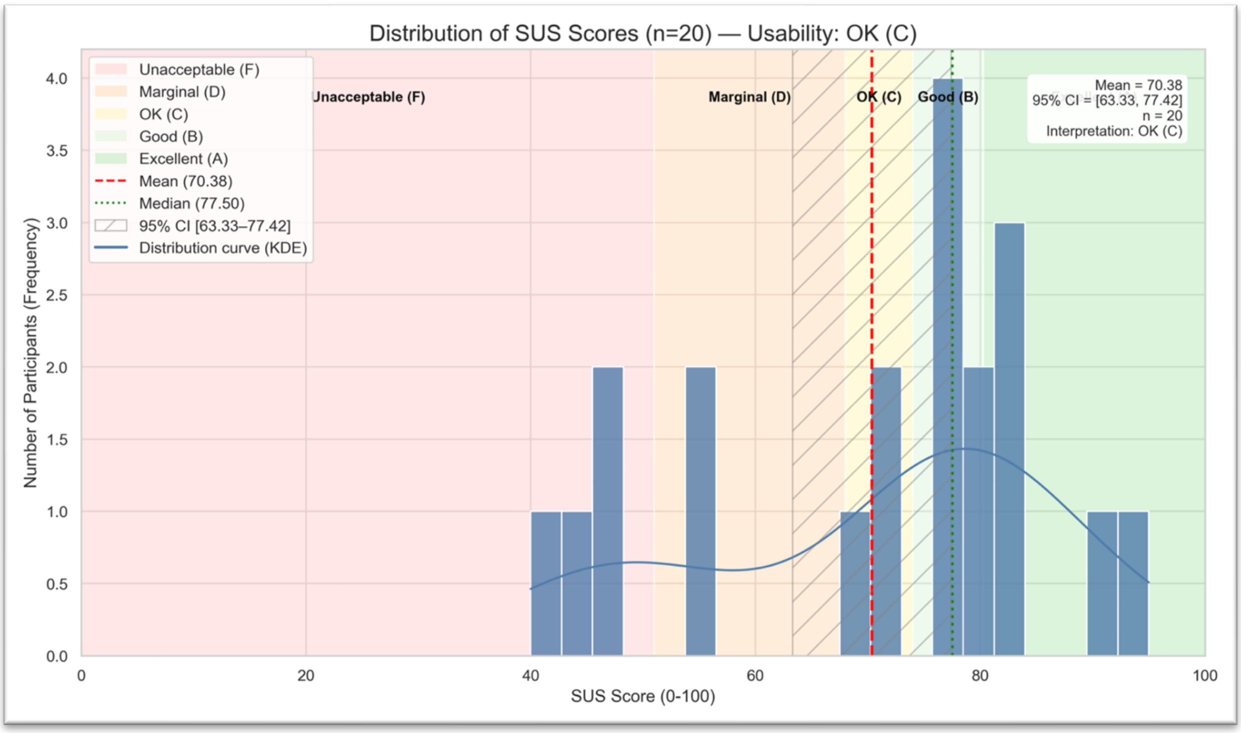


Fig. 6. Distribution of System Usability Scale (SUS) scores (N = 20). Background bands reflect the adjective rating scale [15]; the distribution is left-skewed (skewness = -0.625), indicating that higher usability scores are more common with some lower outliers.

6.3 Limitations

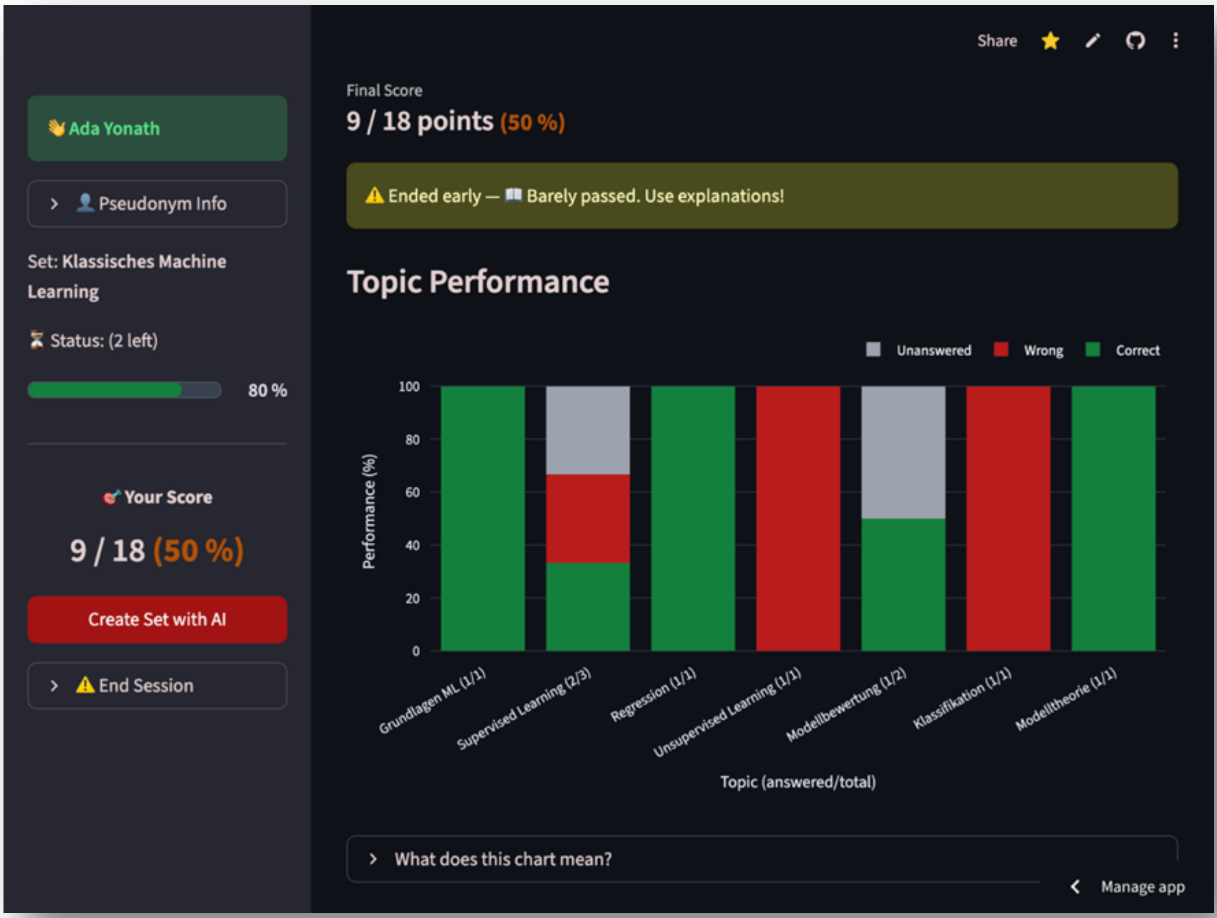
This paper primarily contributes a system design and preliminary usability evidence. The proposed benefits of cognitive profiling and pacing support (e.g., reduced rapid guessing, improved metacognitive regulation) require empirical validation beyond SUS. Cognitive-level labels can be interpretive; therefore, reliability checks (double-coding) and psychometric item analyses are necessary to avoid overconfident interpretations of learner-facing analytics. Finally, pacing mechanisms may affect learner autonomy and accessibility; configuration and accommodation pathways are required to avoid unintended disadvantages.

Recent studies explicitly assess the psychometric properties of LLM-generated questions, reinforcing the need to report item-level statistics when using AI-generated MCQs in practice [2]. Beyond classical item analysis, future work may also leverage "generative student" simulations, i.e., LLM-based profiles that approximate response patterns across different knowledge states, to flag ambiguous or low-discrimination items before deployment [16].

7. Conclusion

MC-Test demonstrates that effective AI-supported assessment requires more than simply calling an API. By combining context-engineered system instructions (finite-state interaction and schema-constrained generation) with a didactic data model and pedagogical pacing support (cooldowns and Panic Mode), the platform aims to transform multiple-choice testing from a summative drill into a formative, metacognitive tool. Restricting generation to verifiable cognitive levels (Bloom 1-3) and migrating to local LLMs strengthen both verifiability and data sovereignty while enabling robust, parseable outputs.

The primary contributions of this work are: (1) a didactic item data model linking learning objectives, cognitive levels, and formative rationales; (2) an operational concept of Pedagogical Control (configurable cooldowns plus Panic Mode) intended to nudge reading and reflection under realistic time constraints; and (3) an architecture for local, schema-constrained LLM inference in an assessment context. Initial usability results are encouraging; however, evidence for learning impact, behavioral effects (e.g., rapid-guessing reduction), label reliability, and psychometric item quality will be addressed in planned comparative and log-based evaluations.


Ada Yonath

Set: Klassisches Machine Learning

Status: (2 left)

80 %

Your Score

9 / 18 (50 %)

Final Score

9 / 18 points (50 %)

Ended early — Barely passed. Use explanations!

Topic Performance

Fig. 7. MC-Test interface showing end-of-session feedback and the Topic Performance dashboard.

REFERENCES

[1] T. Weber, M. Brandmaier, A. Schmidt, and S. Mayer, “Significant Productivity Gains through Programming with Large Language Models,” Proc. ACM Hum.-Comput. Interact., vol. 8, EICS, Art. 256, 2024, doi:10.1145/3661145.

[2] S. Bhandari, Y. Liu, Y. Kwak, and Z. A. Pardos, “Evaluating the psychometric properties of ChatGPT-generated questions,” *Computers and Education: Artificial Intelligence*, vol. 7, Art. 100284, 2024, doi: 10.1016/j.caeai.2024.100284.

[3] N. Scaria, S. D. Chenna, and D. Subramani, “Automated Educational Question Generation at Different Bloom’s Skill Levels Using Large Language Models: Strategies and Evaluation,” in *Artificial Intelligence in Education (AIED 2024)*, LNAI vol. 14830, pp. 165-179, 2024, doi: 10.1007/978-3-031-64299-9_12.

[4] P. Stalder, et al., “Ensuring Quality in AI-Generated Multiple-Choice Questions for Higher Education with the QUEST Framework,” in Communications in Computer and Information Science, Springer, 2024.

[5] R. Bodily and K. Verbert, “Trends and issues in student-facing learning analytics reporting systems research,” in *Proc. 7th Int. Conf. Learning Analytics & Knowledge (LAK ’17)*, Vancouver, BC, Canada, Mar. 2017, pp. 309-318, doi: 10.1145/3027385.3027403.

[6] L. de Vreugd, et al., “Learning Analytics Dashboard Design and Evaluation to Support Student Self-Regulation of Study Behavior,” Journal of Learning Analytics, vol. 11, no. 3, 2024.

[7] I. Jivet, M. Scheffel, M. Specht, and H. Drachsler, “License to evaluate: Preparing learning analytics dashboards for educational practice,” in *Proc. 8th Int. Conf. Learning Analytics & Knowledge (LAK ’18)*, Sydney, NSW, Australia, Mar. 2018, pp. 31-40, doi: 10.1145/3170358.3170421.

[8] S. L. Wise and C. E. DeMars, “An application of item response time: The effort-moderated model,” Journal of Educational Measurement, vol. 43, pp. 19-38, 2006.

[9] S. L. Wise, “Rapid-guessing behavior: Its identification, interpretation, and implications,” Educational Measurement: Issues and Practice, vol. 36, no. 4, pp. 52-61, 2017, doi:10.1111/emip.12165.

[10] H. Drachsler and W. Greller, “Privacy and Analytics: It’s a DELICATE Issue-A Checklist for Trusted Learning Analytics,” in Proceedings of the Sixth International Conference on Learning Analytics & Knowledge, Edinburgh, UK, 2016, pp. 89-98.

[11] Y. Shanmugarasa, M. Ding, M. A. P. Chamikara, and T. Rakotoarivelo, “SoK: The Privacy Paradox of Large Language Models: Advancements, Privacy Risks, and Mitigation,” arXiv preprint arXiv:2506.12699, 2025.

[12] Z. Yan and D. Carless, “Self-Assessment Is About More Than Self: The Enabling Role of Feedback Literacy,” Assessment & Evaluation in Higher Education, vol. 47, no. 7, pp. 1116-1128, 2022.

[13] N. E. Winstone and D. Carless, Designing Effective Feedback Processes in Higher Education: A Learning-Focused Approach. London: Routledge, 2019.

[14] J. Brooke, “SUS: A ‘Quick and Dirty’ Usability Scale,” in *Usability Evaluation in Industry*, P. W. Jordan, B. Thomas, B. A. Weerdmeester, and I. L. McClelland, Eds. London, U.K.: Taylor & Francis, 1996, pp. 189-194.

[15] A. Bangor, P. T. Kortum, and J. T. Miller, “Determining what individual SUS scores mean: Adding an adjective rating scale,” Journal of Usability Studies, vol. 4, no. 3, pp. 114-123, 2009.

[16] X. Lu and X. Wang, “Generative Students: Using LLM-Simulated Student Profiles to Support Question Item Evaluation,” in *Proc. 11th ACM Conf. on Learning @ Scale (L@S ’24)*, 2024, pp. 16-27, doi: 10.1145/3657604.3662031.