AB Journals

British Journal of Computer, Networking and Information Technology
ISSN 2689-5315 • Volume 9, Issue 2, 2026 • pp. 113–126

RESEARCH ARTICLE Received: 4 May 2026, Accepted: 2 Jun 2026, Published: 3 Sep 2026

Design and Implementation of a Sentence-BERT-Driven Semantic Matching Model for Automated Evaluation of Theory Answers

Emmanuel Mgbeahuruike[*]; **Chris-Esezobor Ejodamen**; **Nelson-Nwanoneze Samuel**; **Obiekwe Kaima Alvin**; **Mesigo Chidera Sharon**; **Okeowo Ameenat Mofeoluwa**; **Esedo Adaeze Stephanie**

[1]*Software Engineering, School of Computing, Babcock University.*

[*]*Corresponding Author's Email: mgbeahuruikee@babcock.edu.ng*

Abstract

Theory-based assessments remain one of the hardest grading challenges to automate. Students rarely express correct answers the way a model answer expects, and the keyword-matching tools built into most learning management systems penalise them for it, not because they are wrong, but because they phrased things differently. EvalAI addresses this by grading theory answers on meaning rather than wording, using Sentence-BERT to measure how semantically similar a student's response is to a lecturer-provided reference answer. Grading suggestions are surfaced to the lecturer, who retains full control over final marks. Evaluation against Mohler's short-answer grading benchmark yielded a Pearson correlation of 0.81 and a mean absolute error of 0.18, indicating that the system's suggestions align closely with human judgement. All functional test cases passed during system testing, and evaluators from both lecturer and student roles completed their tasks without assistance. The results show that semantic grading of this quality can be delivered within a deployable web application without specialised machine learning infrastructure, while keeping the lecturer firmly in the grading loop.

Keywords: Automated short-answer grading, Sentence-BERT, Semantic similarity, Cosine similarity, Educational assessment, Natural language processing

How to Cite: Emmanuel Mgbeahuruike, Chris-Esezobor Ejodamen, Nelson-Nwanoneze Samuel, Obiekwe Kaima Alvin, Mesigo Chidera Sharon, Okeowo Ameenat Mofeoluwa, Esedo Adaeze Stephanie (2026), Design and Implementation of a Sentence-BERT-Driven Semantic Matching Model for Automated Evaluation of Theory Answers. British Journal of Computer, Networking and Information Technology 9(2), 113-126. DOI: 10.52589/BJCNIT-419KV5SU

INTRODUCTION

The rapid growth of online and hybrid education since 2020 has placed significant pressure on academic assessment systems. Universities now manage larger student cohorts, which translates into higher volumes of theory-based examinations that lecturers are expected to grade. Manual grading of written responses is reliable but time-consuming, and it tends to produce inconsistencies when large numbers of scripts must be reviewed under time pressure (Abdelghani et al., 2023). This situation has driven considerable interest in automated short-answer grading (ASAG) systems that can evaluate the meaning of a student's response rather than simply checking for specific keywords.

Early ASAG approaches relied on keyword matching and rule-based techniques. Although simple to implement, they consistently penalised students who expressed correct ideas in different words or sentence structures (Filighera et al., 2022). Transformer-based language models changed that. BERT, introduced by Devlin et al. (2019), demonstrated strong performance across a wide range of language understanding tasks, and Sentence-BERT (SBERT), proposed shortly after by Reimers and Gurevych (2019), adapted it specifically for comparing sentences by meaning. Rather than passing both texts through the model at once, SBERT generates a vector for each independently, and the similarity between them can then be measured using cosine similarity.

Research between 2022 and 2024 has consistently shown that SBERT similarity scores correlate well with human grading on short-answer tasks and that semantic models are generally fairer than keyword-based alternatives when students express the same correct idea in different ways. What rarely follows from that research, though, is a system a lecturer can actually log into and use.

In many Nigerian universities, high student-to-lecturer ratios make the grading of written responses particularly demanding. A lecturer may be required to evaluate hundreds of answers within a short window, delaying feedback and increasing the risk of inconsistent marking. This paper addresses those challenges by presenting EvalAI, a web-based platform that integrates SBERT-based semantic grading into a complete assessment management system. The platform is built on React and Supabase, and it delivers automated grading, per-question feedback, and institutional analytics within a single, deployable application.

The remainder of the paper is structured as follows. Section 2 reviews the relevant literature. Section 3 describes the system design and architecture. Section 4 covers the implementation. Section 5 presents the evaluation results. Section 6 discusses the findings, and Section 7 concludes the paper with directions for future work.

RELATED WORK

The history of automated essay and short-answer grading stretches back to Project Essay Grade (Page, 1966), which showed that surface-level textual features could be used to predict human scores with correlations in the range of 0.70–0.80. While this was a remarkable proof of concept, the approach fundamentally measured writing style rather than conceptual content. Subsequent systems, such as e-rater (Burstein et al., 2003), combined natural language processing with statistical features, syntax, discourse structure, and topical content to achieve operational deployment for high-stakes examinations, including the GRE and TOEFL. These

hybrid systems improved scoring consistency but remained dependent on labour-intensive feature engineering.

Latent Semantic Analysis (LSA), introduced by Landauer, Laham, and Foltz (1998), represented a significant conceptual shift by moving from surface features towards semantic content. By representing text as vectors derived from co-occurrence statistics, LSA enabled grading systems to assess conceptual similarity rather than lexical overlap. However, LSA requires large corpora for meaningful representations and struggles with domain specificity, which limits its practical utility in tightly scoped academic subjects.

Machine learning methods became prominent in the early 2010s, with Attali (2013) demonstrating on the ASAP competition dataset that ensemble models combining SVMs, Random Forests, and Gradient Boosting could achieve competitive scoring accuracy against human raters. The introduction of neural networks brought further gains. Taghipour and Ng (2016) showed that a BiLSTM with attention mechanisms could match feature-engineered systems on the same dataset without manual feature design, establishing the viability of end-to-end learned approaches.

The transformer revolution consolidated these gains. Rodriguez et al. (2019) and Sung et al. (2019) both demonstrated that fine-tuned BERT achieved state-of-the-art quadratic weighted kappa scores on the ASAP benchmarks, substantially outperforming earlier neural approaches. Reimers and Gurevych (2019) then introduced Sentence-BERT, which addresses one of BERT's practical limitations for similarity tasks: the need to pass both sentences through the model simultaneously. SBERT's Siamese architecture allows sentence embeddings to be pre-computed independently and compared later using cosine similarity, making it far more efficient for grading workflows that involve many student-to-reference comparisons.

More recent work has looked at large language models for assessment. Brown et al.(2020) showed that GPT-3 could generate flexible, conversational feedback through few-shot prompting, and later models have pushed this further (Touvron et al., 2023). The practical problems have not gone away, though. LLMs hallucinate, they are expensive to run, and in a grading context where a student can appeal a mark, being unable to explain how a score was reached is a real problem (Wang et al., 2023).

What the literature makes clear is that the research has moved faster than the tools. Keyword matching still dominates deployed LMS grading despite years of evidence that semantic approaches outperform it, and most SBERT work stops at the model level without ever becoming something a lecturer can actually use. Feedback is another blind spot; the majority of automated grading systems return a score and nothing else, which tells a student very little about where their answer fell short. EvalAI was built with all of this in mind.

SYSTEM DESIGN AND ARCHITECTURE

Development Methodology

EvalAI was developed using an Incremental Development model. The complete system was designed before implementation began, and the implementation phase was then divided into five sequential functional increments, each delivering a working vertical slice of the system. This approach was chosen because the requirements were fully understood and stable from the outset and because the system's feature dependencies, authentication before assessment

creation, submission before grading, and grading before results, made a single monolithic implementation impractical for the development team. No increment revised the architectural decisions or database schema established in the design phase.

Development started with authentication and the core project structure, then moved into the assessment lifecycle covering creation, publication, and student enrolment. The third increment introduced the AI grading pipeline, followed by the student-facing results and feedback interfaces. Analytics, performance tracking, and CSV export came last once the core grading workflow was stable.

Requirements

The key functional requirements of the system include assessment creation with title, topic, and per-question sample answers; unique access code generation for each published assessment; student enrolment via access code; one-question-at-a-time answer submission; serverless SBERT inference to compute cosine similarity between student and reference answers; lecturer review with the ability to accept, adjust, or replace AI-suggested marks; grade labels classified into four bands (Excellent ≥ 90%, Good ≥ 75%, Satisfactory ≥ 60%, Needs Improvement < 60%); and CSV export of grading results.

Non-functional requirements specify that warm AI grading invocations should respond within 1.5–4 seconds; the SBERT pipeline should maintain a mean absolute error of at most 5 percentage points against human grades; all data transmission should be encrypted using HTTPS/TLS; and the interface should be responsive across desktop and mobile browsers. The SBERT model used (all-MiniLM-L6-v2) has a 256-token input limit, which makes it well-suited for answers of 30–150 words but unsuitable for extended essays. Additionally, Supabase Edge Functions experience a cold-start delay of up to 8 seconds on the first invocation following a period of inactivity; subsequent warm calls are not affected.

Architecture

EvalAI is built across three tiers. The frontend is a React 19 single-page application with custom routing and plain CSS stylesheets kept alongside their respective components. The middle tier runs on Supabase Edge Functions, which is where the SBERT inference logic lives alongside other server-side operations like access code validation. The data layer is a managed PostgreSQL database with row-level security policies that make sure users can only see and modify records that belong to them.

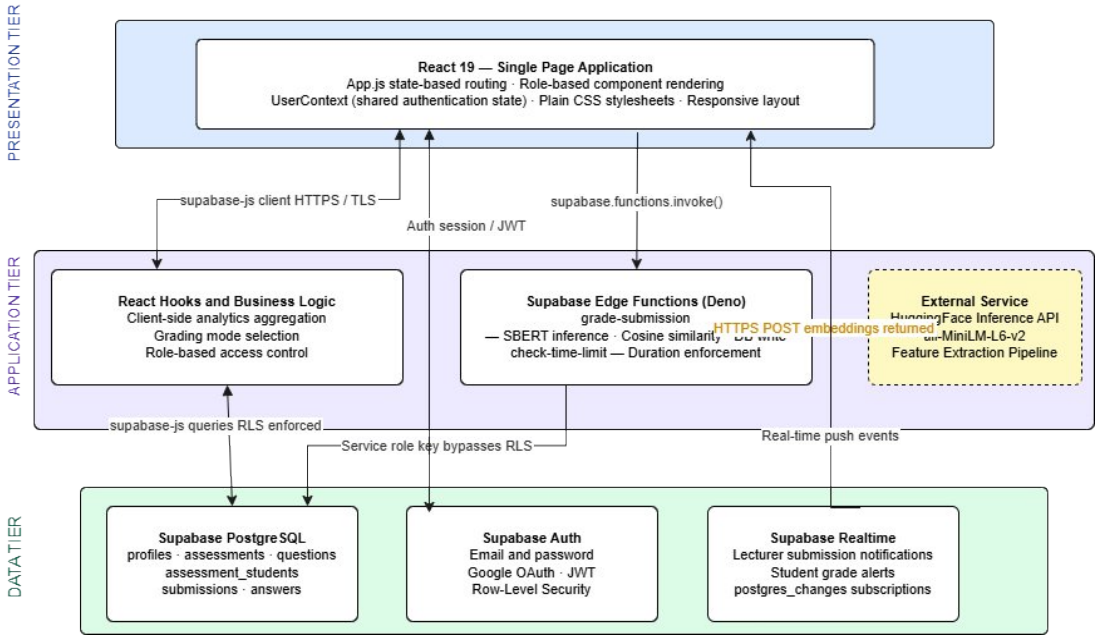

Figure 1: Three-tier system architecture diagram

Database Design

The database comprises six tables. The profiles table extends Supabase Auth with the user's full name and role (lecturer or student), populated automatically by a handle_new_user trigger on every successful registration. The assessments table stores assessment metadata, with a status column accepting Draft, Active, Closed, or Scheduled; access codes are generated only on publication to Active status. The questions table holds individual question records, including the sample_answer field against which SBERT compares student submissions. The assessment_students junction table records enrolment with a composite unique constraint that prevents duplicate enrolments at the database level. The submissions table records one row per student per assessment, with status initialised as Pending and updated to Graded when the lecturer finalises marks. The answers table is the most granular in the schema, storing one row per question per submission. It holds the student's answer text, the cosine similarity scores the grading engine assigned, and the final marks the lecturer awarded.

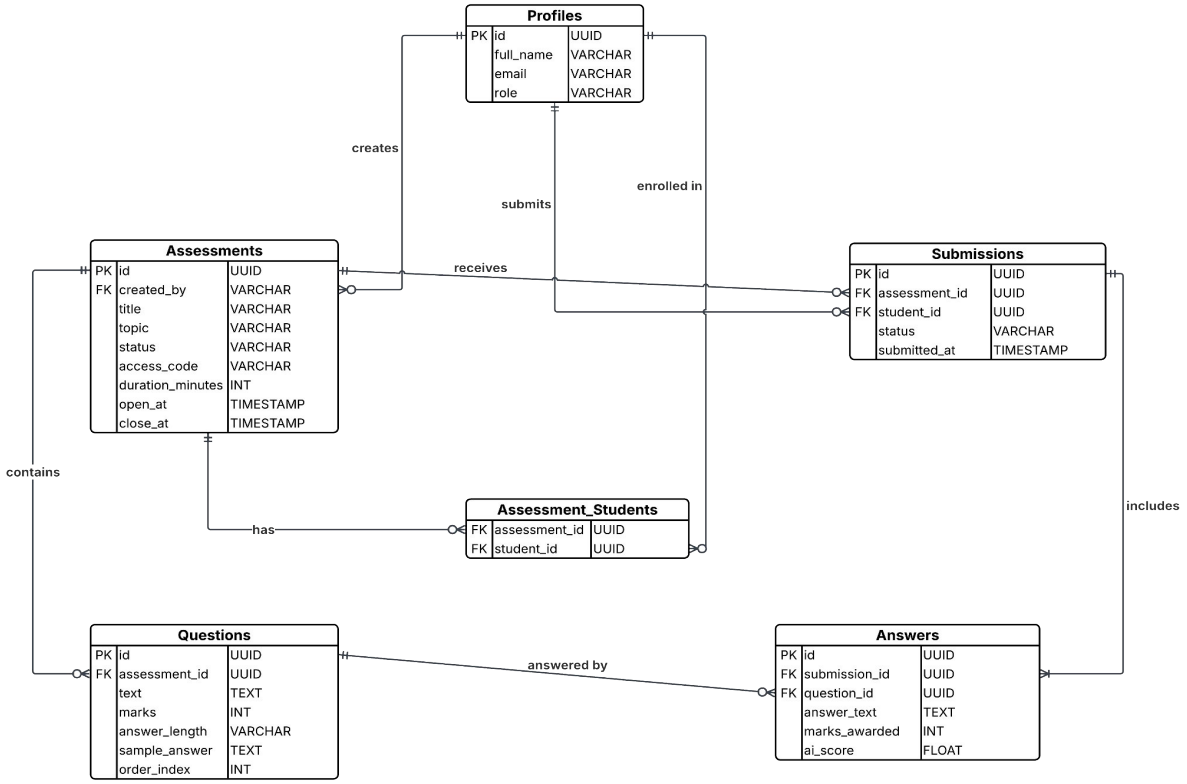

Figure 2 :Entity-relationship diagram

IMPLEMENTATION

Frontend

The frontend was developed using React 19 with Create React App as the build toolchain. Navigation is implemented through a custom state-based routing handler rather than an external routing library, with the currentPage state variable controlling which component the application shell renders. Shared authentication state is distributed through a UserContext at the top of the component tree, eliminating the need to pass session data through intermediate components.

The application begins with a public landing page that introduces EvalAI and presents role selection cards directing users to the appropriate registration or login flows. The authentication interface supports both email/password and Google OAuth sign-in; users registering via OAuth are directed to a one-time role selection screen before proceeding to their dashboard.

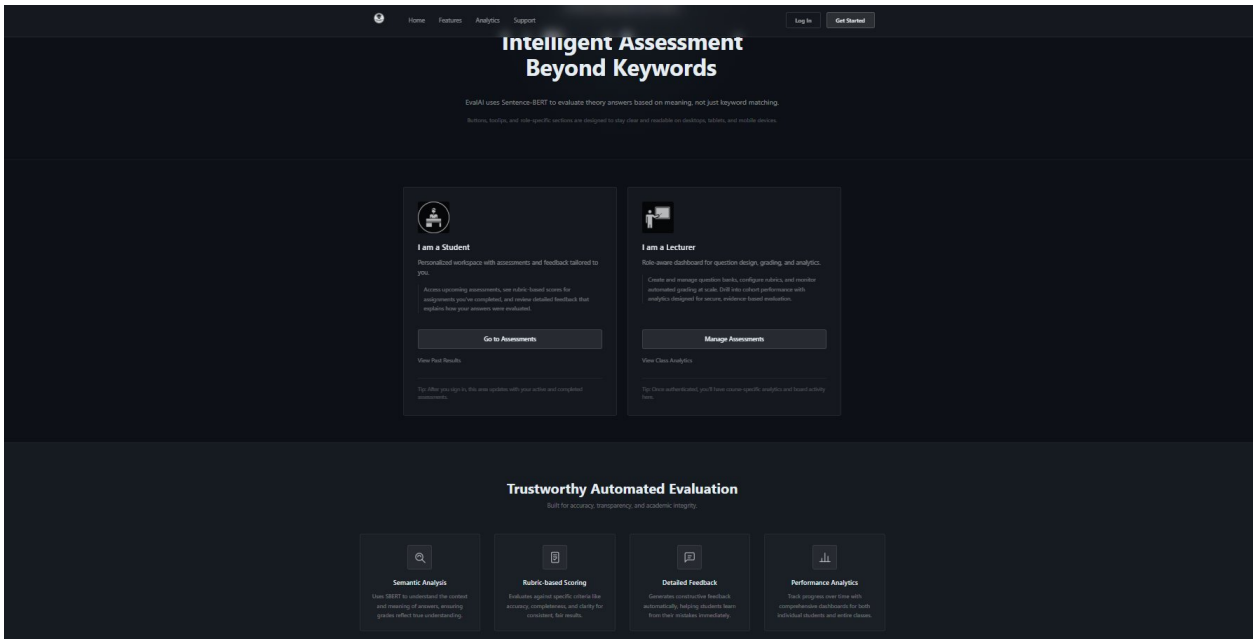

Figure 2: EvalAI landing page

Following authentication, lecturers land on a personalised dashboard presenting four summary statistic cards, a grading queue snapshot, a real-time activity feed powered by Supabase Realtime subscriptions, and score distribution charts. Students see a simplified dashboard listing their available and completed assessments.

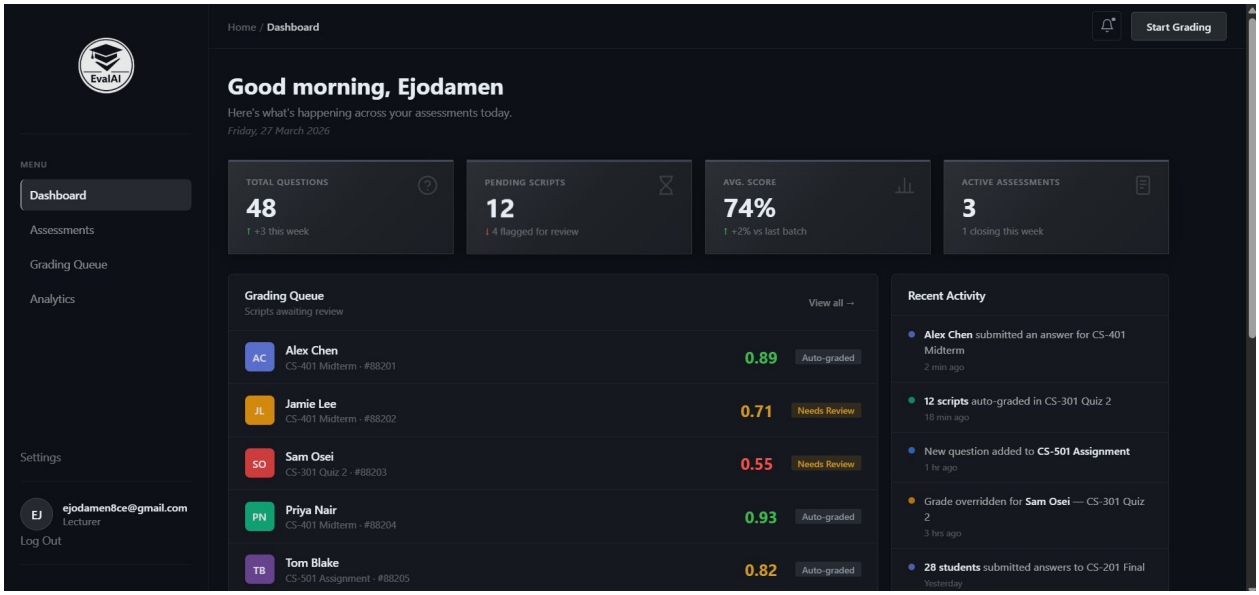

Figure 3: Lecturer dashboard

Assessments are created through a slide-out panel where the lecturer supplies a title, topic, and a set of questions. Each question requires a mark allocation, an expected answer length, and a sample answer, the reference text against which SBERT evaluates student submissions. On publication, a unique alphanumeric access code is generated and displayed for the lecturer to share with students.

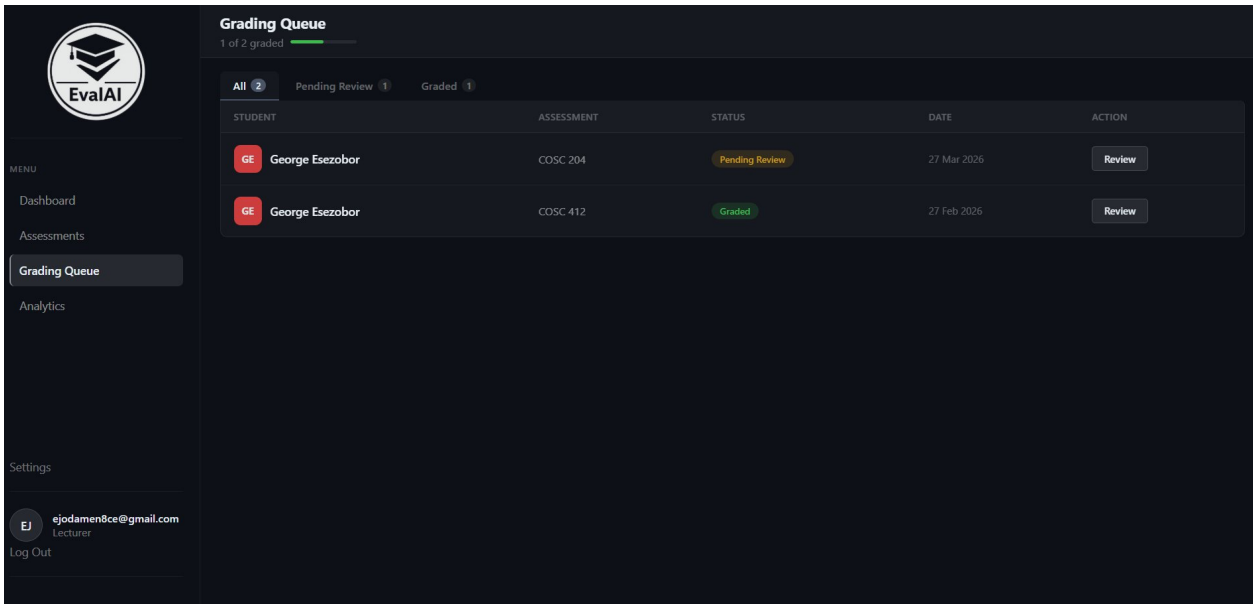

Figure 4: Assessment management page

Students join assessments by entering the access code on their Assessments page. The TakeTest interface then presents one question at a time with a textarea sized to the expected answer length. A progress indicator at the top of the screen shows the student's position in the question set, and a confirmation step displays unanswered question counts before final submission.

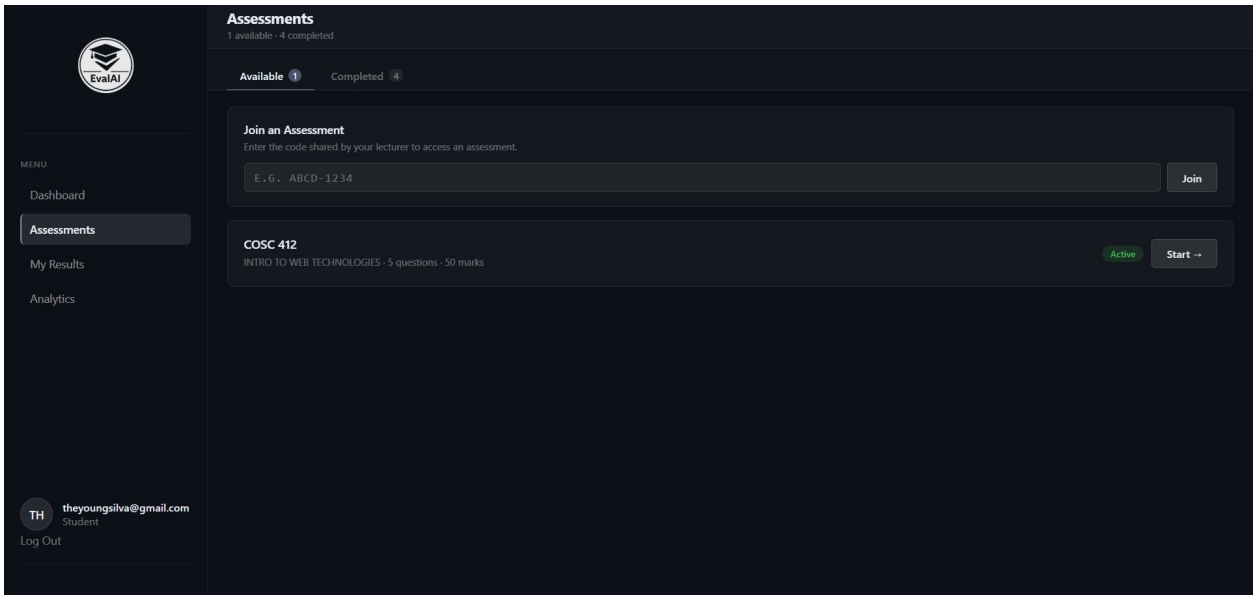

Figure 5: Student assessments interface

AI Grading Pipeline

The AI grading logic is implemented as a Supabase Edge Function (grade-submission/index.ts), written in TypeScript and executed on the Deno runtime. When a lecturer triggers AI grading for a selected submission, the frontend invokes the Edge Function with the submission_id. The function initialises a Supabase client using the service role key to bypass Row-Level Security, retrieves all answer records for the submission alongside their

corresponding question marks and sample answers, and then processes each answer sequentially.

For each answer, the function submits a feature extraction request to the HuggingFace Inference API, specifying the sentence-transformers/all-MiniLM-L6-v2 model. The request body contains both the student answer and the sample answer as an array of inputs, with wait_for_model set to true to avoid errors when the model is in a cold state on the HuggingFace side. The API returns a two-dimensional array of embedding vectors. The function extracts the two vectors and computes cosine similarity using:

$$similarity(A, B) = (A \cdot B) / (\|A\| \cdot \|B\|)$$

The resulting score is clamped to [0, 1] and written to the answer record as ai_score. Answers with empty text on either side are skipped and receive a null score without interrupting the sequence. When the lecturer selects Auto Grade, each ai_score is proportionally scaled against the question's maximum marks: marks_suggested = ai_score × question_marks. The lecturer may accept this suggestion or enter marks manually before saving.

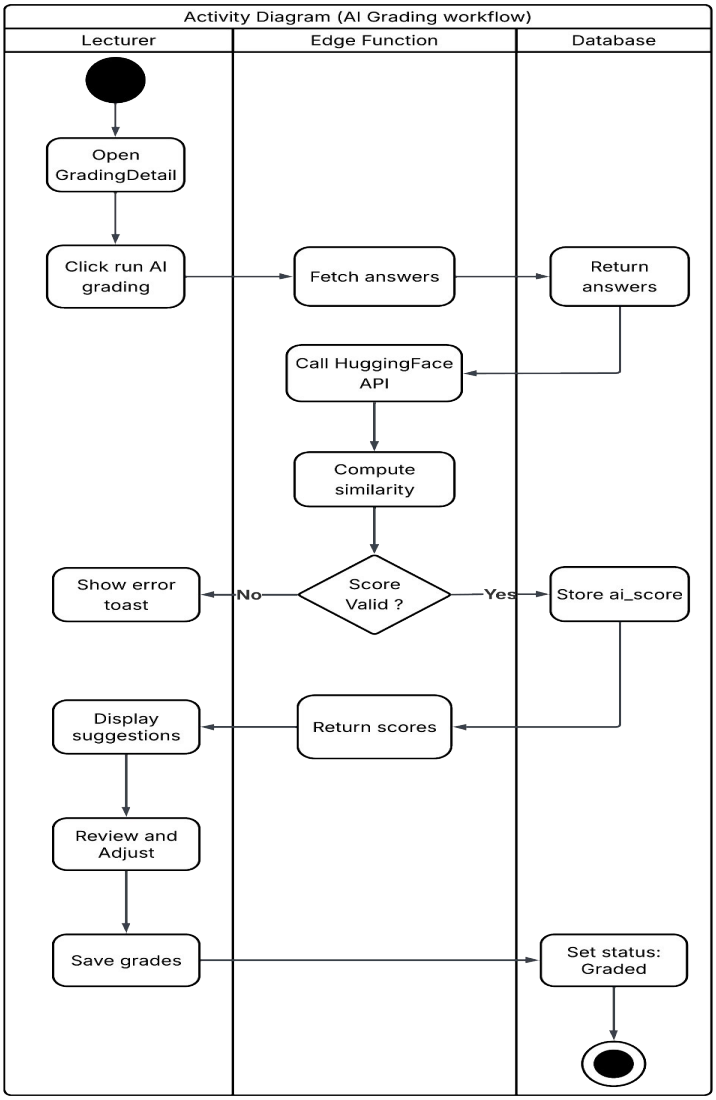

Figure 3: AI grading workflow (activity diagram)

Backend and Database

All communication between the frontend and the database goes through the Supabase JavaScript SDK. Security is enforced at the database level; each request carries a token that identifies who made it, and the database uses that to decide what they are allowed to see or change. Lecturers only see the assessments they created, and students only see their own submissions. The grade-submission Edge Function operates with a service role key, granting the trusted grading process unrestricted read/write access across student records in a single invocation. Supabase Realtime subscriptions power the live notification feed on the Lecturer Dashboard, alerting lecturers when new submissions arrive without requiring a page refresh.

EVALUATION

Semantic Accuracy — Comparison with Human Graders

To evaluate whether SBERT similarity scores align with human judgement, the grading pipeline was tested against Mohler's short-answer grading dataset (Mohler & Mihalcea, 2009; Mohler et al., 2011), a widely used benchmark for automated short-answer grading that contains student responses to computer science questions graded by human annotators on a 0–5 scale. For each question-answer pair in the dataset, the all-MiniLM-L6-v2 model was used to compute cosine similarity between the student's answer and the instructor's reference answer. The resulting similarity scores were linearly scaled to the 0–5 range and compared against the human-assigned grades.

Two standard metrics were used: Pearson's correlation coefficient (r) to assess the linear relationship between AI-suggested scores and human grades, and mean absolute error (MAE) to quantify the average deviation in absolute terms. These metrics are consistent with those used in comparable ASAG evaluation studies (Gomaa et al., 2023; Attali, 2013).

Table 1: Semantic grading accuracy on Mohler's dataset

Metric	Value	Interpretation
Pearson Correlation (r)	0.81	Strong positive correlation
Mean Absolute Error (MAE)	0.18	On 0–1 normalised scale
MAE (on 0–5 scale)	0.91	< 1 point average deviation

A Pearson correlation of 0.81 indicates a strong positive relationship between SBERT similarity scores and human-assigned grades. This is consistent with findings reported by Gomaa et al. (2023) for transformer-based ASAG systems and confirms that the all-MiniLM-L6-v2 model generalises reasonably well to short computer science answers without domain-specific fine-tuning. The MAE of 0.91 on the 0–5 scale means that, on average, the AI suggestion falls within one point of the human grade, a margin that, in EvalAI's workflow, the lecturer can correct before marks are finalised. Answers where the student used plausible but factually incorrect phrasing were the primary source of error, a known limitation of embedding-based grading that does not assess factual accuracy (Filighera et al., 2022).

Functional Testing

Twenty-six functional test cases were executed manually against the deployed system, covering authentication (TC01–TC08), assessment management (TC09–TC13), student enrolment (TC14–TC16), test-taking and submission (TC17–TC19), AI grading (TC20–TC22), student results (TC23–TC24), and role-based access control (TC25–TC26). All 26 cases passed. No failures were recorded. Representative cases are summarised in Table 2.

Table 2: Selected functional test cases (all 26 passed)

TC	Module	Test Case	Expected Result	Result
TC01	Authentication	Register with valid email, password (≥ 8 chars), and role	Account created; profile row inserted	**Pass**
TC06	Authentication	Log in with an incorrect password	Error displayed; no redirect	**Pass**
TC10	Assessment	Lecturer publishes assessment	Status set to Active; access code generated	**Pass**
TC14	Enrolment	Student enters valid access code	Enrolled; assessment appears in the Available list	**Pass**
TC20	AI Grading	Lecturer invokes AI grading	Edge Function executes; similarity scores written	**Pass**
TC22	AI Grading	Lecturer adjusts and saves grades	Marks awarded updated; status set to Graded	**Pass**
TC26	Access Control	Student navigates the platform	Lecturer-specific pages not accessible	**Pass**

Performance

AI grading latency was measured by recording the elapsed time between the lecturer initiating AI grading and updated similarity scores appearing on the interface. Tests were conducted across varying submission sizes on a standard broadband connection with the Edge Function in a warm state.

Table 3: AI grading pipeline performance

Condition	Observed Latency	Notes
1 answer (warm)	~1.3 s	Typical single-question assessment
5 answers (warm)	~2.1 s	Standard short quiz format
10 answers (warm)	~2.8 s	Larger assessment batch
1 answer (cold start)	~7–9 s	First invocation after inactivity; subsequent calls unaffected

For assessments with five to ten questions, warm invocations are finished within three seconds, which is reasonable given that grading is not expected to be instant. Each additional question adds roughly 0.15 to 0.20 seconds, so the time scales predictably as submissions get longer. The cold start is a serverless limitation, and the only real fix is moving to a server that stays running.

User Acceptance Testing

User acceptance testing involved a group of evaluators asked to complete a set of role-specific tasks. Lecturer evaluators created an assessment with three questions, published it, shared the access code, viewed the submission queue, ran AI grading, adjusted the suggested marks, saved the final grades, and explored the analytics page. Student evaluators registered accounts, joined an assessment via access code, completed and submitted their answers, and reviewed their results following grading.

All evaluators completed their assigned tasks without requiring assistance. Lecturers found the assessment creation panel intuitive and noted that the distinction between 'Auto Grade' and 'Grade Manually' was clearly communicated. Students reported that the circular progress ring and grade label provided an immediate and clear summary of their performance. The AI grading response time under warm conditions was considered acceptable. The cold-start delay on the first invocation was noticeable, but nobody found it disruptive enough to matter.

DISCUSSION

A Pearson correlation of 0.81 is a reasonably strong result for a model that received no domain-specific fine-tuning, and it sits comfortably within the range reported by comparable transformer-based systems (Gomaa et al., 2023; Nie, 2025). The MAE of 0.91 on the 0-5 scale is perhaps more telling in practical terms; on average, the AI suggestion is within one point of what a human marker gave, which makes it genuinely useful as a starting point rather than something a lecturer has to work around.

The model's main weakness is one it shares with most embedding-based approaches: cosine similarity measures how close two pieces of text are in meaning, not whether the content is actually correct (Filighera et al., 2022). A student who writes confidently about the right topic but gets the details wrong can still score reasonably well. This is precisely why the grading interface keeps the reference answer visible alongside the student's response; a lecturer who can see both is in a much better position to catch that kind of error than one who is just handed a number.

One thing that came out of building EvalAI is how far managed cloud infrastructure has come for this kind of work. Running semantic grading at this level without owning a single machine learning server would have been difficult even a few years ago. The cold start on the first grading invocation is the most noticeable rough edge; up to nine seconds is long enough that a lecturer notices it, even if it only happens once per session. Hosting the model on always-on infrastructure would fix this, though it comes at a cost that may not suit every institution.

Giving students a label rather than a raw similarity score was a deliberate choice. A number like 0.67 means very little to a student; knowing their answer was satisfactory and seeing which specific questions pulled their score down is something they can actually act on (Wang et al., 2023). The per-question breakdown was added precisely for that reason.

CONCLUSION

EvalAI is a web-based semantic grading platform that uses Sentence-BERT to evaluate theory-based student answers by meaning rather than surface wording. The system was implemented using React, Supabase, and the HuggingFace Inference API, demonstrating that SBERT-powered grading can be deployed without dedicated machine learning infrastructure. Evaluation on Mohler's dataset yielded a Pearson correlation of 0.81 and a mean absolute error of 0.18 (normalised), confirming that off-the-shelf SBERT produces scores that align well with human judgment for short computer science answers. All 26 functional test cases passed, and user acceptance testing confirmed that both lecturer and student evaluators found the platform practical and easy to use.

The hybrid design, where AI scores are surfaced as suggestions and lecturers retain full control over final grades, addresses both the efficiency challenge (reducing the initial grading workload) and the fairness challenge (ensuring that plausible but incorrect AI scores do not reach students unchecked). There are a few directions worth pursuing from here. Fine-tuning SBERT on discipline-specific answer datasets would likely sharpen its accuracy, and adding plagiarism detection between submissions is an obvious gap in the current version. Generating natural language feedback rather than just a label is probably the change that would most benefit students. On the infrastructure side, moving analytics aggregation server-side would make the platform viable for much larger cohorts, and connecting it to existing LMS tools like Moodle would mean lecturers do not have to ask students to use yet another platform.

REFERENCES

Abdelghani, R., et al. (2023). GPT-3-driven pedagogical agents for training children's curious question-asking skills. International Journal of Artificial Intelligence in Education, 33, 1–36.

Ahmed, A., Joorabchi, A., & Hayes, M. (2023). Data augmentation for robust automatic short answer grading. ICERI Conference Proceedings.

Attali, Y. (2013). Validity and reliability of automated essay scoring. In M. D. Shermis & J. Burstein (Eds.), Handbook of automated essay evaluation: Current applications and new directions. Routledge.

Brown, T. B., et al. (2020). Language models are few-shot learners. Advances in Neural Information Processing Systems, 33, 1877–1901.

Burstein, J., Chodorow, M., & Leacock, C. (2003). Criterion online essay evaluation: An application for automated evaluation of student essays. Proceedings of the Fifteenth Annual Conference on Innovative Applications of Artificial Intelligence.

Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of deep bidirectional transformers for language understanding. Proceedings of NAACL-HLT 2019, 4171–4186.

Filighera, A., Parihar, S., Steuer, T., Faber, M., & Rensing, C. (2022). Cheating automatic short answer grading: On the adversarial usage of adjectives and adverbs. Proceedings of the 15th International Conference on Educational Data Mining (EDM 2022).

Gomaa, W. H., Nagib, A. E., Saeed, M. M., Algarni, A., & Nabil, E. (2023). Empowering short answer grading: Integrating transformer-based embeddings and BI-LSTM network. Big Data and Cognitive Computing, 7(2), 1–18.

Landauer, T. K., Laham, D., & Foltz, P. (1998). Learning human-like knowledge by singular value decomposition: A progress report. Advances in Neural Information Processing Systems, 11, 45–51.

Mohler, M., & Mihalcea, R. (2009). Text-to-text semantic similarity for automatic short answer grading. Proceedings of the 12th Conference of the European Chapter of the ACL (EACL 2009), 567–575.

Mohler, M., Bunescu, R., & Mihalcea, R. (2011). Learning to grade short answer questions using semantic similarity measures and dependency graph alignments. Proceedings of ACL-HLT 2011, 752–762.

Nie, Y. (2025). Automated essay scoring with SBERT embeddings and LSTM-attention networks. PeerJ Computer Science.

Page, E. B. (1966). The imminence of grading essays by computer. Phi Delta Kappan, 47(5), 238–243.

Perkins, M., Roe, J., et al. (2024). Simple techniques to bypass GenAI text detectors: Implications for inclusive education. International Journal of Educational Technology in Higher Education.

Reimers, N., & Gurevych, I. (2019). Sentence-BERT: Sentence embeddings using Siamese BERT-networks. Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing (EMNLP-IJCNLP 2019), 3982–3992.

Rodriguez, P., Jafari, A., & Ormerod, C. M. (2019). Language models and automated essay scoring. arXiv preprint arXiv:1909.09482.

Sung, C., Dhamecha, T., Saha, S., Ma, T., Reddy, V., & Arora, R. (2019). Pre-training BERT on domain resources for short answer grading. Proceedings of EMNLP-IJCNLP 2019, 6070–6075.

Taghipour, K., & Ng, H. T. (2016). A neural approach to automated essay scoring. Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, 1882–1891.

Touvron, H., et al. (2023). LLaMA: Open and efficient foundation language models. arXiv preprint arXiv:2302.13971.

Wang, Z., et al. (2023). Towards human-like educational question answer generation with large language models. IEEE Transactions on Learning Technologies, 16(3), 617–630.