NANYANG TECHNOLOGICAL UNIVERSITY

Long Term Tutor Bot with Memory and Educational-Level Awareness

Supervisor: Prof. Chee Wei Tan

by

Keith Lim En Kai

College of Computing and Data Science

2025

NANYANG TECHNOLOGICAL UNIVERSITY

Long-Term Tutor Bot with Memory and Educational-Level Awareness

Submitted in Partial Fulfillment of the Requirements

for the Degree of Bachelor of Computing in Computer Science of
Nanyang Technological University

by

Keith Lim En Kai

College of Computing and Data Science

2025

Abstract

Educational chatbots powered by large language models have shown strong potential in supporting learning through on-demand explanations and problem-solving assistance. However, most existing systems are general-purpose and lack alignment with specific educational curricula, often producing responses that are overly advanced, insufficiently contextualised, or misaligned with examination requirements. This limitation is particularly significant in structured education systems such as Singapore's, where curriculum scope and assessment standards vary distinctly across educational stages.

This work presents the design and development of a curriculum-aware AI tutor chatbot tailored for Singaporean students across multiple academic levels, including Primary School Leaving Examination (PSLE), GCE O-Level, GCE A-Level, and the International Baccalaureate (IB). The proposed system allows users to specify their educational level and subject, enabling the chatbot to generate responses that are syllabus-aligned and pedagogically appropriate. In addition, the chatbot supports the interpretation of uploaded handwritten questions or user-submitted answers through optical character recognition, allowing students to receive feedback on both problem statements and their own attempted solutions.

To ensure accuracy and curriculum relevance, the system integrates Retrieval Augmented Generation (RAG), grounding responses in official syllabi, school notes, and examination materials rather than relying solely on the language model's internal knowledge. This approach reduces hallucination, improves factual correctness, and enables rapid updates when curriculum content changes without retraining the underlying model. By combining level-aware prompting, document retrieval, and automated feedback generation, the proposed chatbot aims to provide personalised, scalable, and curriculum-aligned learning support for students within the Singapore education system.

Acknowledgments

I would like to express my sincere gratitude to my Final Year Project supervisor, Professor Chee Wei Tan, for his invaluable guidance, insightful feedback, and unwavering support throughout the course of this project. His expertise, encouragement, and generous provision of resources in the emerging field of Generative Artificial Intelligence were instrumental in the successful completion of this work.

I also wish to acknowledge Nanyang Technological University (NTU) and its faculty members for the extensive academic resources, rigorous training, and continuous support provided throughout my undergraduate studies. Their dedication and commitment to excellence have played a vital role in shaping my academic development and scholarly foundation.

Contents

Current to do list (Ideas and Features to implement):

1. UI to implement and improve:
    - https://v3.heroui.com/
2. User specific sessions (right now there is no memory and the agent only has context from the current input, the previous conversation is not passed in as context)
3. Update Main page, too short and little information, add information on specific documents in the Vector Database to supplement and add validity.
4. Change colour themes of app to something more modern, find inspiration
5. Update About page to contain developer information, add github and a circle with clickable link with github profile picture that when clicked leads directly to the github profile: https://github.com/keithfykai
6. Add dark/light mode

Chapter 1 Introduction

1.1 Background

In the past 5 years, Generative Artificial Intelligence (GAI) Chatbots have undergone a rapid evolution, from simple predictive text generators to advanced systems capable of web retrieval and producing high-quality synthetic images. Early-generation chatbots including OpenAI's widely viral GPT-3.5 model released in 2022, were built on a Transformer decoder-only architecture [1], which was originally derived from the initial design proposed in Attention Is All You Need [2].

These early GAI tools suffered from several issues, one of the most detrimental being hallucination, which is the generation of incorrect and/or misleading results and content [3]. This caused the tools to come up with unfactual information which impacted its accuracy and usefulness. For example, the GPT-3.5 model exhibited hallucination rates of around 39.6% when tasked to generate references used for systematic reviews, producing references with at least two or more bibliographic errors [4].

Gradually, numerous architectural innovations have been introduced to tackle these limitations. Some of which include Mixture-of-Experts (MoE) layers [5], long-context Transformer mechanisms [6], multimodal integration architectures [7], speculative decoding and distillation [8], and Retrieval-Augmented Generation (RAG) [9]. Together, these advancements have contributed to drastically reduced hallucination rates in OpenAI's latest flagship model, GPT-5, which reports rates of only 9.6% [10].

GAI tools have seen widespread adoption in virtually every corner of the world, with the education sector being no exception. In Singapore, more than 84% of pre-tertiary and tertiary students report using AI regularly for schoolwork [11]. Similarly, nearly 70% of students have attended tuition classes [12], and total household expenditures spent on private tuition reaching approximately S$1.8 billion in 2023 [13]. Through this extra help, it has often created positive reinforcement cycles of improved grades, greater confidence and stronger motivation to do well in other areas of life. However, the beneficiaries of this help has largely been limited to families who are able to afford it, leading to inequities in educational opportunities in today's society.

1.2 Motivation

Because many current GAI systems are trained primarily on global or non-Singaporean curriculum, they are unable to consistently provide tailored, syllabus-aligned responses to Singaporean students. Additionally, affordable alternatives for high-quality academic support remain limited. Thus, there

exists a need for an AI-driven solution that addresses this gap with greater accuracy and local curriculum alignment.

1.3 Objective

This project aims to explore the theory and algorithmic design of AI-native Large Language Models (LLMs) to develop a Retrieval-Augmented Generation (RAG) chatbot. The designed system will assess user queries and provide tailored responses to Singaporean students up to the pre-tertiary level, while minimizing hallucination.

1.4 Scope

This project focuses on the design and development of a Retrieval-Augmented Generation (RAG) chatbot to support pre-tertiary students in Singapore. Specifically, encompassing the collection, preprocessing, and integration of syllabus-aligned educational materials into a vectorized knowledge base, which will be used to augment a Large Language Model (LLM). The system will be fine-tuned to provide accurate, contextually relevant, and curriculum-specific responses to student queries.

Chapter 2 Literature Review

2.1 Large Language Models

Large Language Models (LLMs) are neural language models trained at scale to learn statistical and semantic regularities from large text corpora, enabling generation and understanding of natural language across diverse tasks. Contemporary LLMs are predominantly based on the Transformer architecture, introduced by Vaswani et al., which replaces recurrence with self-attention to model token dependencies in parallel and capture long-range contextual relationships efficiently [2]. Self-attention computes contextualised representations by projecting tokens into Query, Key, and Value vectors and weighting interactions via attention scores, allowing the model to integrate global context into each token representation [2].

Scaling Transformer-based language models has been shown to yield substantial gains in task performance, including instruction following and few-shot learning capabilities. For instance, Brown et al. demonstrate that increasing model and dataset scale improves general-purpose performance without task-specific fine-tuning, supporting the viability of large pre-trained models as general language engines [14]. Despite these capabilities, LLMs exhibit limitations that are particularly relevant in education settings: they may generate fluent but incorrect statements ("hallucinations"), especially when the prompt requires specific factual or curriculum-aligned content not reliably encoded in model parameters [15]. Such limitations motivate methods that integrate external sources of truth at inference time to improve factuality, traceability, and domain alignment.

2.1.1 Reasoning Models

The introduction of OpenAI's **o1 series of models in September 2024** marked a significant shift in the design philosophy of large language models (LLMs) toward **explicit reasoning-oriented training**. Unlike earlier generative models that focused primarily on next-token prediction, reasoning models are optimised to internally decompose complex tasks into structured intermediate steps before producing a final response. This paradigm aligns closely with earlier academic work on **chain-of-thought (CoT) reasoning**, where models are encouraged to articulate intermediate logical steps to improve performance on multi-step problems such as mathematics, scientific reasoning, and code generation [16].

Prior research has demonstrated that exposing or internally modelling intermediate reasoning steps substantially improves accuracy on tasks requiring compositional reasoning. Wei *et al.* showed that chain-of-thought prompting enables LLMs to solve problems that are otherwise inaccessible when

only short-form answers are generated [16]. Subsequent work by Wang *et al.* introduced **self-consistency decoding**, where multiple reasoning paths are sampled and aggregated, further improving robustness and correctness on reasoning-heavy tasks [17]. The o1 series builds upon these ideas by incorporating reasoning behaviours **directly into the training process**, reducing reliance on manually crafted prompts and making structured reasoning an intrinsic capability of the model rather than an external prompting technique.

An important consequence of reasoning-oriented training is the **reduction of hallucinations**, a known limitation of earlier LLMs. Hallucinations often arise when models generate fluent but unverified outputs without internal consistency checks. Reasoning models mitigate this by implicitly evaluating intermediate steps and validating conclusions against prior context before returning an answer. This aligns with research on **self-refinement and verification**, where models iteratively critique and revise their own outputs to improve factual correctness and logical consistency [18]. As a result, reasoning models demonstrate improved reliability in domains such as science education, mathematical problem-solving, and programming assistance, making them particularly suitable for educational applications where correctness and explainability are critical.

2.2 Retrieval Augmented Generation

Retrieval Augmented Generation (RAG) enhances an LLM by coupling it with an external retrieval component that supplies relevant documents at inference time. The core idea is to combine parametric memory (knowledge stored in model weights) with non-parametric memory (knowledge stored externally), thereby improving factual grounding and enabling easier updates to the underlying knowledge source [19]. Lewis et al. formalise RAG as a framework that retrieves passages from a dense index and conditions generation on the retrieved evidence, reporting improvements on knowledge-intensive NLP tasks and supporting the claim that retrieval can improve specificity and factual consistency [19].

In educational question-answering, the advantage of RAG is practical: curriculum materials (e.g., syllabi, school notes, worked solutions) may change over time, and it is not cost-effective to retrain or fine-tune a large model for every syllabus update. Instead, RAG supports incremental updates by re-indexing or appending new documents to the database. This design aligns with the broader observation that hallucination remains a key challenge in LLM deployment, and retrieval grounding is a widely adopted mitigation strategy (though not a complete solution) [15]. In a syllabus-aligned tutor scenario, a user query is embedded and used to retrieve relevant curriculum sections or examples; the LLM then produces an explanation conditioned on both the query and retrieved evidence, improving relevance to the examination scope and reducing unsupported generation.

Example of a Retrieval Augmented Generation (RAG) Workflow:

Workflow Step	**Description**
User Prompt	E.g “What is the quadratic formula?”
Vector Embedding + Retrieval	Convert query into a vector representation (e.g word2vec, GloVe, or transformer embeddings). Perform similarity search across indexed code files.
Context Passing	User asked: "What is the quadratic formula?" Formula: $x = [-b \pm \sqrt{(b^2 - 4ac)}] / 2a$
LLM Response Generation	The model uses both the prompt and retrieved context to produce a grounded answer: “The quadratic formula is: $x = [-b \pm \sqrt{(b^2 - 4ac)}] / 2a$”

2.2.1 Similarity Search in RAG

RAG retrieval typically relies on dense embeddings and similarity search. Dense retrieval methods such as Dense Passage Retrieval (DPR) represent queries and passages in the same vector space using a dual-encoder architecture and retrieve nearest neighbours by vector similarity, often outperforming traditional sparse retrieval baselines in open-domain QA settings [20]. For embedding sentences or short educational chunks, approaches such as Sentence-BERT (SBERT) are frequently used to produce semantically meaningful sentence embeddings that can be compared efficiently using cosine similarity [21]. These methods underpin the retrieval stage of many modern RAG systems.

Common similarity measures include cosine similarity and dot product, while common distance measures include Euclidean and Manhattan distance. In practice, cosine similarity is widely used for semantic embeddings due to its normalization effect, while dot product may be appropriate when magnitude encodes useful information depending on the embedding model and indexing strategy [21].

Common Similarity Search Methods:

Type of Similarity Search	**Description**	**Formula**
Cosine Similarity	A measure of how similar 2 vectors are, calculated as the cosine of the angle between them. Ranging from -1 to 1, where 1 indicates the vectors point in the same	$\text{cosine}(A, B) = (A \cdot B) / (\|\|A\|\| \, \|\|B\|\|)$

	direction, 0 represents orthogonal (unrelated), and -1 means they point in opposite directions and are completely opposite.	
Dot Product	A measure of vector similarity by calculating the sum of element-wise products, indicating both directional alignment and magnitude. A higher value represents greater similarity and is useful when the vector length matters, but it can be skewed by magnitude, often leading to cosine similarity which normalizes magnitude to be preferred.	DotProduct(x, y) = x · y = Σ(i=1 to n) (x_i y_i)
Euclidean	Measures straight-line distance between two vectors in multidimensional space. Smaller values indicate stronger similarity.	d_Euclidean(x, y) = ‖x − y‖_2 = sqrt(Σ(i=1 to n) (x_i − y_i)^2)
Manhattan	Measures the sum of absolute differences across dimensions. Effective for sparse vector spaces.	d_Manhattan(x, y) = ‖x − y‖_1 = Σ(i=1 to n) |x_i − y_i|

These similarity methods allow RAG systems to efficiently locate the most relevant information for any given user query.

2.3 Vector Database

A vector database is a specialised storage and indexing system designed to manage high-dimensional embeddings and support efficient nearest neighbour search. In RAG pipelines, the vector database stores embeddings corresponding to document chunks (e.g., syllabus sections, worked solutions, concept definitions) and enables retrieval of the most semantically relevant chunks given an embedded user query. Compared to keyword search, vector search better supports semantic matching (e.g., recognising that “completing the square” relates to “quadratic formula derivation”), which is important in educational question answering where student phrasing may vary substantially.

Due to the emergence of video, audio, and other unstructured data modalities, the need for a new class of data management systems has become increasingly apparent. Traditional relational databases are

optimised for structured, schema-bound data and exact-match queries, making them poorly suited to representing high-dimensional semantic information derived from unstructured sources such as text, images, or speech. Vector databases address this gap by storing dense numerical embeddings that capture semantic meaning, enabling similarity-based retrieval rather than strict equality matching. This paradigm is especially relevant for modern AI applications, including recommendation systems, semantic search, and RAG-based tutoring systems.

In educational contexts, vector databases allow heterogeneous learning materials—textbooks, lecture slides, OCR-extracted exam papers, and even transcribed video content—to be represented in a unified embedding space. This abstraction enables semantically meaningful retrieval across different formats and levels of granularity. By decoupling knowledge storage from the language model itself, vector databases support scalable, updateable, and syllabus-aligned knowledge access, which is critical for maintaining correctness and relevance in academic settings.

Unlike normal databases that rely on deterministic indexing structures such as B-trees or hash tables, vector databases are designed around approximate similarity search in high-dimensional spaces. Exact nearest neighbour search becomes computationally infeasible as dimensionality and dataset size grow, a phenomenon commonly referred to as the “curse of dimensionality.” Vector databases therefore prioritise probabilistic retrieval techniques that trade a small amount of recall for substantial gains in speed and scalability.

Furthermore, vector databases typically integrate tightly with machine learning pipelines, supporting operations such as batch embedding ingestion, metadata filtering, and hybrid query execution. This contrasts with conventional databases, where machine learning functionality is often layered externally. In RAG systems, this tight integration enables efficient end-to-end workflows, from content ingestion and indexing to real-time retrieval and generation, all within latency constraints suitable for interactive educational applications.

2.3.2 How a Vector Database is built

A typical vector database ingestion pipeline consists of:

1. **Document acquisition and normalisation:** collecting source materials and converting them into machine-readable text (including OCR when needed).
2. **Chunking:** splitting documents into smaller semantically coherent units (e.g. headings, paragraphs, stepwise solutions). Chunking helps to improve retrieval specificity and reduces irrelevant context being passed into the LLM.

3. **Embedding**: converting each chunk into a dense vector representation using an embedding model (e.g DPR-style encoders for retrieval, or SBERT-like models for sentence-level semantics) [20], [21].
4. **Indexing**: building an index structure to accelerate similarity search at scale. Large-scale systems commonly use Approximate Nearest Neighbour (ANN) methods to balance retrieval speed with recall.

2.3.3 How is Retrieval is Performed

At inference time, the user query is embedded using the same embedding model used for indexing, and the vector database performs nearest neighbour search to return the top-k most similar chunks. ANN indexing methods are commonly applied because exact search becomes computationally expensive as the number of vectors grows. Johnson et al. demonstrate billion-scale similarity search with GPU acceleration and compressed representations, illustrating practical strategies for scaling vector retrieval [22]. Graph-based ANN approaches such as Hierarchical Navigable Small World (HNSW) graphs provide strong performance by organising vectors in a multi-layer graph that supports efficient navigation toward nearest neighbours [23]. These retrieval methods enable low-latency access to relevant context, which is then inserted into the LLM prompt to generate a grounded response.

This retrieval process supports operational requirements typical in educational RAG systems: updateability (add/remove content without model retraining), traceability (retrieved evidence can be logged and inspected), and domain alignment (retrieval constrained to syllabus-approved content).

2.3.4 Standard and Hybrid Querying

Standard querying in vector databases typically refers to pure vector similarity search, where a query embedding is compared against stored embeddings to retrieve the top-k nearest neighbours based solely on distance metrics such as cosine similarity. This approach is effective when semantic relevance is the primary criterion, such as retrieving conceptually similar explanations or examples regardless of exact wording.

However, many real-world applications require additional constraints beyond semantic similarity. Hybrid querying addresses this need by combining vector-based retrieval with traditional structured filters, such as metadata constraints, keyword matching, or access control rules. For example, an educational RAG system may retrieve only syllabus-approved documents for a specific examination board or academic level while still ranking results by semantic similarity.

Hybrid querying improves both precision and controllability, particularly in domains where correctness and scope boundaries matter. By integrating symbolic filters with neural similarity search, vector databases enable more reliable and context-aware retrieval, reducing the risk of out-of-scope or inappropriate content being surfaced during generation.

2.3.5 Approximate Nearest Neighbour Search

Approximate Nearest Neighbour (ANN) search is a foundational technique in vector databases, enabling scalable similarity search over millions or billions of high-dimensional vectors. Rather than guaranteeing exact nearest neighbours, ANN algorithms aim to retrieve vectors that are sufficiently close with high probability, dramatically reducing query latency.

This approximation is essential because exact k-nearest neighbour search scales linearly with dataset size and dimensionality, making it impractical for large embedding collections. ANN methods exploit structural properties of vector spaces, such as clustering tendencies or graph connectivity, to prune the search space efficiently.

In educational RAG systems, ANN search enables real-time interaction even as curriculum repositories grow over time. The slight loss in recall is typically acceptable given the redundancy of educational content and the downstream LLM's ability to synthesise information from multiple retrieved chunks.

2.3.6 Cluster and Graph-Based Indexes

Cluster-based indexing methods partition the embedding space into regions, allowing queries to be evaluated only against the most relevant clusters. Techniques such as k-means clustering or inverted file indexes (IVF) are commonly used to reduce the search space before performing local similarity comparisons.

Graph-based indexes, such as HNSW, represent embeddings as nodes in a proximity graph where edges connect nearby vectors. Queries navigate this graph by iteratively moving toward closer neighbours, achieving logarithmic or sublinear search complexity. These approaches have become dominant in modern vector databases due to their strong balance between recall, latency, and memory usage.

2.3.7 Liveness Layers

Liveness layers refer to architectural mechanisms that allow newly ingested vectors to become searchable immediately without requiring full index rebuilds. This is particularly important in

dynamic systems where content is frequently updated, such as educational platforms that continuously ingest new notes, past papers, or syllabus revisions.

By separating mutable “live” data from more static base indexes, vector databases can support low-latency updates while maintaining high-performance retrieval. This design ensures that newly added educational materials are instantly available for RAG, improving responsiveness and content freshness.

2.3.8 Disk-Resident Indexing

Disk-resident indexing enables vector databases to scale beyond the limitations of main memory by storing indexes partially or fully on disk. While memory-resident indexes offer lower latency, they become prohibitively expensive at large scales.

Modern systems mitigate disk access costs through compression, caching, and sequential access patterns. Disk-resident indexing is particularly relevant for institutional educational repositories, where long-term storage of large corpora must be balanced against cost and performance constraints.

2.3.9 Quantization

Quantization reduces the memory footprint of vector embeddings by representing them with lower-precision numerical formats. Techniques such as product quantization compress vectors into compact codes while preserving approximate distance relationships.

This compression allows vector databases to store and search significantly larger datasets within fixed hardware budgets. In RAG systems, quantization enables large-scale curriculum indexing without degrading retrieval quality beyond acceptable thresholds.

2.3.10 Updatable Indexes

Updatable indexes support insertions, deletions, and modifications without requiring full re-indexing. This capability is essential in educational environments, where content must be revised to reflect curriculum changes, corrected solutions, or updated learning objectives.

Index updateability directly supports maintainability and governance, ensuring that outdated or incorrect materials can be removed promptly. This aligns with the broader goal of reducing hallucinations by ensuring retrieval sources remain accurate and authoritative.

2.3.11 GPU-Based Indexing

GPU-based indexing leverages massive parallelism to accelerate both index construction and query execution. By offloading vector similarity computations to GPUs, systems can achieve significant throughput improvements, particularly for large batch queries.

In educational platforms serving many concurrent users, GPU acceleration helps maintain low response times during peak usage, such as exam preparation periods. This scalability is critical for deploying RAG systems at institutional or national levels.

2.3.12 Multi-Tenancy

Multi-tenancy allows a single vector database deployment to serve multiple users, organisations, or courses while maintaining isolation between datasets. This is achieved through logical namespaces, metadata filters, or access control mechanisms.

In education, multi-tenancy supports scenarios where different schools, curricula, or examination boards share infrastructure without data leakage. It also enables personalised retrieval experiences while preserving administrative separation and security.

2.3.13 VDBMS Architecture

A Vector Database Management System (VDBMS) typically consists of ingestion pipelines, embedding services, indexing layers, query engines, and storage backends. These components are orchestrated to support high-throughput ingestion and low-latency retrieval under dynamic workloads.

From a systems perspective, VDBMS architecture reflects a shift toward AI-native data infrastructure. By treating embeddings as first-class data objects and integrating tightly with machine learning workflows, vector databases form a critical backbone for RAG-based educational systems.

2.4 Optical Character Recognition

Educational content frequently exists in image-based formats, including scanned exam papers, diagrams, handwritten notes, and PDF documents with embedded raster images. Optical Character Recognition (OCR) converts such visual text into machine-readable representations, enabling subsequent chunking, embedding, and indexing for RAG. Recent OCR research has increasingly adopted transformer-based architectures. For example, TrOCR proposes an end-to-end Transformer OCR approach that combines a Transformer-based vision encoder with a text decoder for sequence generation [24]. Beyond plain OCR, document AI models such as LayoutLMv3 integrate text and image representations to better handle complex layouts (e.g., multi-column notes, tables, question

sheets), which is relevant for robust ingestion of real-world educational documents prior to embedding and retrieval [25].

A strong ingestion pipeline—combining OCR, layout-aware parsing, and chunking—can therefore improve downstream retrieval quality by preserving semantic structure and reducing noise introduced during text extraction, which in turn improves the grounding quality of the RAG response.

OCR plays a pivotal role in bridging the gap between raw educational materials and vector-based retrieval systems. By converting scanned or image-based content into text, OCR enables documents that would otherwise be inaccessible to LLMs to be embedded and indexed within vector databases. This process expands the retrievable knowledge base to include legacy materials such as handwritten notes, printed worksheets, and past examination papers.

Advanced OCR pipelines often incorporate layout-aware parsing, confidence scoring, and post-correction techniques to improve text quality before embedding. Clean, well-structured OCR output improves embedding fidelity, which in turn enhances similarity search accuracy within the vector database. As a result, OCR is not merely a preprocessing step but a foundational component that directly impacts retrieval quality and the reliability of downstream RAG responses.

Chapter 3 Data Collection

To ensure that the proposed RAG-based educational chatbot provides accurate and curriculum-aligned responses, the vector database was constructed using authoritative and syllabus-relevant educational materials.

Official syllabus documents were obtained from recognized examination authorities, including the Ministry of Education (MOE) Singapore, the Singapore Examinations and Assessment Board (SEAB), Cambridge Assessment, and the International Baccalaureate Organization (IBO). These syllabi define the scope, learning objectives, and assessment requirements for each educational level and serve as the primary reference for curriculum alignment.

However, official syllabi typically specify learning outcomes rather than detailed instructional content. Furthermore, many officially prescribed textbooks for PSLE and O-Level subjects are proprietary and not publicly accessible, while A-Level subjects do not mandate a single standardized textbook and instead rely on school-developed materials. To address this limitation, supplementary learning materials in the form of school-produced notes, student notes, and publicly available past-year examination papers were incorporated.

Multiple sources were used for supplementary materials to improve content coverage and robustness. Although this introduced redundancy across documents, such duplication was acceptable within the RAG framework, as it increases the likelihood of retrieving relevant context during inference. All supplementary materials were constrained and validated against the corresponding official syllabi to reduce the risk of misalignment, oversimplification, or inclusion of out-of-scope content.

The collected materials span PSLE, O-Level, A-Level, and International Baccalaureate subjects across mathematics and sciences, forming a comprehensive corpus for embedding into the vector database.

"Authoritative syllabus documents and supplementary learning materials were obtained from official examination authorities and publicly accessible educational repositories. A detailed list of data sources is provided in ***Appendix A****."*

Chapter 4 Technical Implementation

This section describes the technical design and implementation decisions behind the proposed system. The application adopts a modern full-stack web architecture, integrating a responsive frontend, a lightweight backend, and a large language model (LLM) enhanced with retrieval-augmented generation (RAG). Each technology choice was guided by considerations of scalability, development efficiency, maintainability, and cost-effectiveness.

4.1 Application Architecture

The application is designed as a full-stack web-based system consisting of a **decoupled frontend and backend architecture**. The frontend is implemented using **Next.js with React**, while the backend is built using **FastAPI**, a modern Python web framework designed for high-performance RESTful APIs. Communication between the frontend and backend occurs through HTTP-based REST API endpoints, enabling a clear separation of concerns between user interface logic and server-side processing.

This architectural approach improves scalability and maintainability, as frontend and backend components can be developed, tested, and deployed independently. The backend is containerised and deployed on **Google Cloud**, allowing it to scale automatically based on request demand, while the frontend is hosted separately on a specialised frontend platform. This separation enables future extensions—such as mobile applications or alternative client interfaces—to reuse the same backend services without requiring significant architectural changes.

FastAPI was selected due to its strong performance characteristics, native support for asynchronous request handling, and automatic request/response validation through Python type hints. These features are particularly well-suited for AI-driven workflows that involve I/O-bound operations such as large language model inference and document retrieval. Additionally, FastAPI automatically generates OpenAPI-compliant documentation, simplifying API testing and integration during development.

The architecture also supports seamless integration with external services, particularly the **OpenAI API** for large language model inference and retrieval-augmented generation. By centralising all model interactions, prompt construction, and document retrieval logic within the backend, the system ensures consistent behaviour across clients, improves security through backend-only API key management, and enables easier monitoring of system performance and usage when deployed on Google Cloud infrastructure.

4.1.1 Tools and Software

This project was developed using a modern full-stack toolchain that supports rapid prototyping, scalable deployment, and maintainable iteration. The selected tools cover source control, containerisation, authentication, database services, backend APIs, large language model integration, and cloud deployment.

Tools	**Description**
Visual Studio Code (VS Code)	Primary development environment used for both frontend and backend implementation. VS Code provided integrated debugging, code formatting, linting, and extension support for JavaScript/TypeScript, Python, Docker, and Git workflows, improving development efficiency and code quality.
Git/Github	Version control and collaboration platform used to track code changes, manage branches, and maintain a reliable project history. GitHub also served as a central repository for source code, enabling reproducible development and facilitating structured commits and rollback when necessary.
Docker	Used to containerise backend services and standardise the execution environment across machines. Docker improved reproducibility by packaging application code together with dependencies, reducing "works on my machine" issues and simplifying deployment to cloud infrastructure.
Firebase Authentication	Provided secure user authentication and account management, supporting common sign-in flows (e.g., email/password and/or OAuth depending on configuration). This reduced the need to implement authentication from scratch while ensuring standard security practices such as token-based session handling.
Firestore Database	Cloud-hosted NoSQL database used to store application data such as user profiles, chat/session metadata, and other structured records required by the frontend/backend. Firestore's managed scaling and real-time capabilities simplified database operations and reduced infrastructure overhead.
FastAPI	Backend framework used to implement RESTful API endpoints that connect the frontend to system services (e.g., authentication checks, request validation, chatbot query routing). FastAPI was selected for its strong performance, automatic OpenAPI documentation generation, and type-safe request/response modelling using Python type hints.
OpenAI	Used to access LLM inference capabilities and manage application-level integrations. The platform also supports document upload workflows for retrieval and vector store management, enabling Retrieval-Augmented Generation (RAG) without maintaining a separate vector database infrastructure.
GPT-5-mini	The primary large language model used for chatbot response

	generation. GPT-5-mini was chosen for low latency and cost efficiency, making it suitable for interactive educational chat use where frequent queries are expected.
Next.JS	Frontend framework used to build the web application interface. Next.js enabled structured routing, component-based UI development with React, and strong performance optimisations, supporting a responsive chat experience across devices.
Vercel	Hosting platform used to deploy the Next.js frontend. Vercel provides seamless CI/CD integration, fast global content delivery, and optimised deployment workflows tailored for Next.js applications.
Google Cloud	Cloud infrastructure used to deploy and run the backend services. Google Cloud provided scalable compute and networking for the API server, allowing the chatbot backend to be hosted reliably and scaled based on usage demand.

4.2 Frontend (Rationale for Frontend Selection)

The frontend is implemented using Next.js and React, which together provide a robust framework for building modern, responsive, and performant web applications. React enables the creation of reusable UI components and efficient state management, which is particularly beneficial for interactive applications such as chat-based educational systems. This component-based approach simplifies development and enhances code readability and maintainability.

Next.js was chosen on top of React due to its built-in support for server-side rendering (SSR) and static site generation (SSG). These features improve initial page load times and search engine optimisation (SEO), while still allowing dynamic client-side interactions. For an educational chatbot, fast load times and a smooth user experience are important to ensure accessibility and sustained student engagement.

Furthermore, the Next.js ecosystem provides seamless routing, API handling, and deployment compatibility with popular cloud platforms. Its strong community support and extensive documentation reduce development friction and enable rapid prototyping. Overall, this frontend stack strikes a balance between performance, developer productivity, and long-term extensibility.

4.2.1 Vercel Deployment

The frontend application is deployed using **Vercel**, a cloud hosting platform optimised for Next.js applications. Vercel provides a fully managed deployment environment with built-in support for continuous integration and continuous deployment (CI/CD), global content delivery, and automatic performance optimisations.

Deployment is carried out by linking the GitHub repository containing the Next.js frontend to Vercel. Whenever changes are pushed to the main branch, Vercel automatically builds and deploys the latest version of the application. This workflow ensures that updates are consistently deployed without manual intervention, reducing deployment errors and improving development velocity.

Vercel also offers features such as serverless functions, edge caching, and automatic HTTPS configuration, which contribute to improved reliability and security. By hosting the frontend on Vercel, the system benefits from low-latency global access and a highly scalable infrastructure that can handle varying user traffic with minimal configuration.

4.3 Backend (Rationale for Backend Selection)

The backend of the system is implemented using **FastAPI**, a modern Python web framework designed for building high-performance RESTful APIs. FastAPI was selected over more traditional frameworks due to its strong performance characteristics, ease of development, and native support for asynchronous programming.

One of the key advantages of FastAPI is its use of Python type hints to automatically validate request and response data. This enables early detection of errors, improves code reliability, and ensures that API contracts between the frontend and backend are well-defined. Additionally, FastAPI automatically generates interactive API documentation using the OpenAPI standard, simplifying testing and debugging during development.

Python was chosen as the backend language due to its extensive ecosystem for artificial intelligence, natural language processing, and data handling. This makes FastAPI particularly suitable for orchestrating large language model (LLM) workflows, managing retrieval-augmented generation (RAG) pipelines, and integrating with external AI services such as the OpenAI API.

From a system design perspective, FastAPI provides excellent scalability when deployed in containerised environments and supports asynchronous request handling, which is beneficial for I/O-bound operations such as API calls to LLM services. These characteristics make it well-suited for production deployment while remaining flexible enough for rapid iteration during development.

4.3.1 Google Cloud Run Deployment

The backend API is deployed using **Google Cloud Run**, a fully managed serverless container platform provided by Google Cloud. Cloud Run allows containerised applications to be deployed

without the need to manage servers or underlying infrastructure, making it a cost-effective and scalable solution for backend services.

The FastAPI backend is packaged into a Docker container, which includes all required dependencies and runtime configurations. This container image is then deployed to Google Cloud Run, where it is executed in a stateless, on-demand environment. Cloud Run automatically scales the number of running containers based on incoming request traffic, scaling down to zero when idle to minimise operational costs.

Deployment is typically triggered through a container build and push process, followed by a Cloud Run service update. Once deployed, the backend is exposed via a secure HTTPS endpoint, which the frontend communicates with through RESTful API calls.

By using Cloud Run, the system benefits from:

- Automatic scaling based on request load
- Built-in security features such as HTTPS and IAM-based access control
- Reduced operational overhead due to serverless infrastructure management

This deployment strategy ensures that the backend remains responsive under varying usage conditions while maintaining low infrastructure complexity and cost.

4.4 Large Language Model

The system utilises **OpenAI's GPT-5-mini** model via the OpenAI API as its primary large language model. GPT-5 Mini is one of OpenAI's smallest and most cost-efficient model, optimised for **low-latency, high-throughput tasks** such as summarisation, classification, and conversational response generation. With a large context window of up to **400,000 tokens**, the model is capable of processing extensive educational materials while maintaining fast response times.

The affordability and performance characteristics of GPT-5 Mini make it particularly well-suited for an **educational chatbot**, where frequent user queries and real-time interaction are expected. By prioritising speed and cost efficiency over deep multi-step reasoning, the model enables scalable deployment without incurring excessive operational expenses. This is critical for applications intended for sustained student use.

In addition to the model itself, OpenAI provides a comprehensive **Graphical User Interface (GUI)** for managing models, API usage, and knowledge sources. This GUI simplifies system configuration, monitoring, and iteration, reducing development overhead. The integration between the LLM and

OpenAI's retrieval tools further streamlines the overall workflow, allowing developers to focus on application logic rather than infrastructure management.

4.5 Vector Database

To support retrieval-augmented generation (RAG), the system leverages **OpenAI's File Search feature**, which functions as an integrated vector database solution. File Search allows developers to upload documents through a user-friendly graphical interface, automatically converting them into vector embeddings that form a searchable knowledge base. These documents can then be queried by the LLM during inference to provide context-aware and accurate responses.

This approach significantly reduces the complexity typically associated with setting up and maintaining a separate vector database such as FAISS, Pinecone, or Chroma. By using File Search, the project avoids additional infrastructure overhead, configuration effort, and synchronisation issues between different platforms. All data ingestion, embedding, and retrieval processes are handled within the same ecosystem as the LLM.

From a development and cost perspective, centralising the vector database and language model within the OpenAI platform simplifies deployment and monitoring. It also ensures compatibility between embeddings and the model's internal representations, leading to more reliable retrieval performance. As a result, File Search provides an efficient and scalable RAG solution that aligns well with the project's goals of ease of use, rapid development, and cost control.

4.6 Application Features

The system provides two core user-facing features designed to support independent learning and formative assessment: an Educational Chatbot and an AI-powered Quiz Mode. Both features are driven by a large language model enhanced with retrieval-augmented generation (RAG), ensuring that responses and questions are grounded in authoritative educational materials rather than purely generative output.

The features are designed with flexibility in mind, allowing students to customise their learning experience based on their educational level (PSLE, O Level, or A Level) and subject domain. This ensures that responses are pedagogically appropriate and aligned with curriculum expectations for each level.

4.6.1 Chatbot

The Educational Chatbot serves as the primary interactive component of the system, enabling students to ask free-form questions related to their academic subjects. Upon initiating a chat session, students are first prompted to specify their current educational level. This information is used to route the request to a level-specific workflow (e.g. PSLE Agent, O Level Agent, or A Level Agent), ensuring that the language complexity, depth of explanation, and retrieved materials are appropriate for the learner.

When a student submits a question, the backend performs the following steps:

1. **Input processing**: The system accepts either text input, image input, or a combination of both. Image uploads are particularly useful for scenarios where students wish to submit scanned questions, textbook excerpts, or handwritten answers.
2. **Retrieval-Augmented Generation (RAG)**: The processed input is used to perform semantic search over the relevant vector database containing curated educational materials and past examination papers.
3. **Response generation**: The retrieved context is injected into the LLM prompt, allowing the model to generate a response that is both contextually relevant and factually grounded.
4. **Response delivery**: The generated output is returned to the frontend and displayed to the student in a conversational format.

The chatbot supports several common learning use cases, including:

- Clarification of concepts and doubts across supported subjects.
- Generation of practice questions or revision prompts for a given topic.
- Verification of student answers, where students can input or upload their attempted solutions and receive feedback.
- Multimodal interaction through photo uploads, allowing richer contextual understanding and more accurate feedback.

By combining conversational interaction with curriculum-aligned retrieval, the chatbot functions as an always-available academic assistant that supports personalised, self-paced learning.

4.6.2 Feature - Quiz Mode

Quiz Mode is designed to provide structured, exam-style practice through automatically generated multiple-choice questions (MCQs). Students begin by selecting their educational level, subject, and

the desired number of questions. Based on these parameters, the system generates a quiz that closely mirrors the style and difficulty of real examination papers.

The MCQs are generated using the same RAG pipeline as the chatbot, drawing from a database of past exam papers and curated educational resources. This ensures that the questions are realistic, syllabus-aligned, and pedagogically meaningful rather than purely synthetic.

Key features of Quiz Mode include:

- **Topic tagging**: Each question displays its associated topic, helping students identify which syllabus area is being tested.
- **Immediate feedback**: When a student answers a question incorrectly, an explanation is provided to clarify the correct reasoning and address misconceptions.
- **Score tracking**: The system maintains a running score throughout the quiz, providing real-time performance feedback.
- **Post-quiz summary**: Upon completion, the system generates a personalised summary analysing the student's performance, highlighting strengths and identifying topics that require further revision.

This feature transforms the application from a passive information tool into an active assessment platform. By offering instant feedback and targeted revision guidance, Quiz Mode approximates the benefits of personalised tuition while remaining accessible and cost-effective.

4.7 API Endpoints

All core application logic related to large language model inference, retrieval workflows, and quiz generation is exposed through a set of RESTful HTTP API endpoints. These endpoints are hosted on a cloud-based backend and act as the communication layer between the frontend interface and the underlying AI services.

This API-driven design provides a clean separation between presentation logic and backend processing, improving scalability, maintainability, and security. The primary endpoints implemented in the system are described below.

4.7.1 / Health Check

The health check endpoint provides a simple mechanism to verify that the backend service and LLM pipeline are operational. It is primarily used for deployment validation, uptime monitoring, and debugging.

A successful response confirms that the API server is running and able to accept requests.

4.8.2. /chat/stream Chatbot Feature

The /chat/stream endpoint implements the core educational chatbot functionality with **real-time streaming response delivery**. It accepts user input in the form of textual prompts and optionally uploaded images, together with the selected educational level (PSLE, O-Level, or A-Level), enabling multimodal and curriculum-aware interactions.

Upon receiving a request, the endpoint performs the following operations:

1. **Input Validation**
   The request parameters are validated to ensure that a valid educational level is specified and that at least one form of input (text or image) is provided.
2. **Level-Aware Workflow Selection**
   Based on the selected educational level, the system routes the request to the corresponding level-specific processing workflow. This ensures that syllabus scope, difficulty, and response style remain aligned with the learner's academic stage.
3. **Retrieval-Augmented Generation (RAG)**
   The system executes a semantic similarity search over the appropriate level-specific vector database to retrieve relevant syllabus notes, worked examples, or contextual reference material. These retrieved documents are incorporated into the prompt context for the language model.
4. **Streaming Response Generation**
   Instead of waiting for the full response to be generated, the endpoint streams partial output tokens incrementally to the frontend using a server-sent events (SSE) style mechanism. As the language model produces new tokens, they are immediately forwarded to the client, allowing the user to observe the response being constructed in real time.
5. **Response Finalisation**
   Once generation is complete, the stream terminates gracefully, signalling the end of the response. Any associated metadata, such as processing completion status, is also communicated to the client.

This streaming-based design significantly improves perceived responsiveness compared to traditional request-response APIs, particularly for longer explanations. It allows students to begin reading and reasoning about the solution while the remainder of the response is still being generated. The /chat/stream endpoint therefore supports multimodal inputs, level-aware reasoning, and real-time interaction, forming the primary conversational interface of the educational chatbot system.

4.8.3 /quiz/start

The /quiz/start endpoint is responsible for generating quizzes in Quiz Mode. It accepts structured JSON input specifying the educational level, subject, and number of MCQs.

After validating the request, the endpoint triggers the quiz generation workflow, which:

- Retrieves relevant exam-style materials.
- Generates MCQs with answer options, correct answers, topic labels, and explanations.
- Returns the complete quiz payload to the frontend for rendering.

This design allows the frontend to remain lightweight while delegating all content generation logic to the backend.

4.8.3 /quiz/summary

The /quiz/summary endpoint handles post-quiz analysis. It accepts the completed quiz data, including selected answers and final score, and generates a performance summary.

The summary includes:

- An overall assessment of the student's performance.
- Identification of weak topics or frequently missed question types.
- Actionable revision recommendations.

By separating quiz generation and quiz analysis into distinct endpoints, the system maintains modularity and allows future extensions, such as analytics dashboards or longitudinal performance tracking.

Chapter 5 Chatbot Application Demonstration

This chapter presents a practical demonstration of the developed curriculum-aware educational chatbot, illustrating how the system operates from an end-user perspective. While the previous chapters focused on the theoretical foundations, data preparation, and technical implementation of the system, this chapter shifts attention to the realised application interface and user interaction flow.

Through a sequence of annotated screenshots, the chapter showcases the key functionalities of the chatbot as experienced by a student. These include the homepage layout, the chat interface, educational level selection, text-based query submission, real-time response generation, and image-based input handling via uploaded photographs of questions. Each screenshot captures a distinct interaction state of the application, allowing the progression of a typical user session to be clearly observed.

The demonstration aims to highlight how curriculum alignment and educational-level awareness are reflected not only in backend processing but also in the frontend design and user experience. In particular, the screenshots illustrate how the chatbot enforces level selection before interaction, provides visual feedback during response generation, and supports multimodal inputs to accommodate real-world student usage scenarios such as handwritten or printed questions.

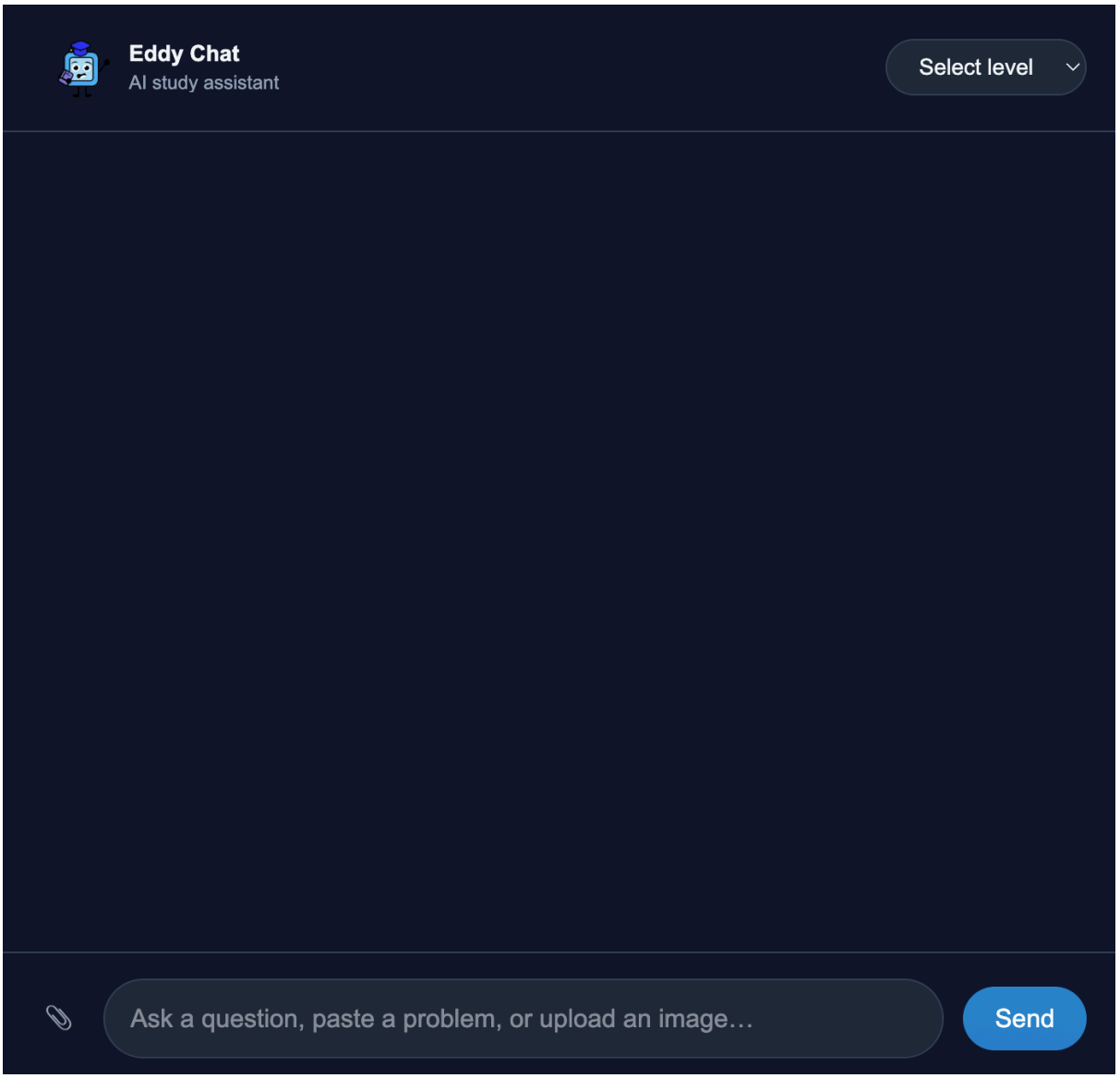
By presenting the application in operation, this chapter bridges the gap between system design and practical deployment, providing concrete evidence of how the proposed chatbot functions as an accessible, interactive, and syllabus-aligned learning support tool within the Singapore education context.

5.1 Website Home Page

Screenshot of the Application's Home Page

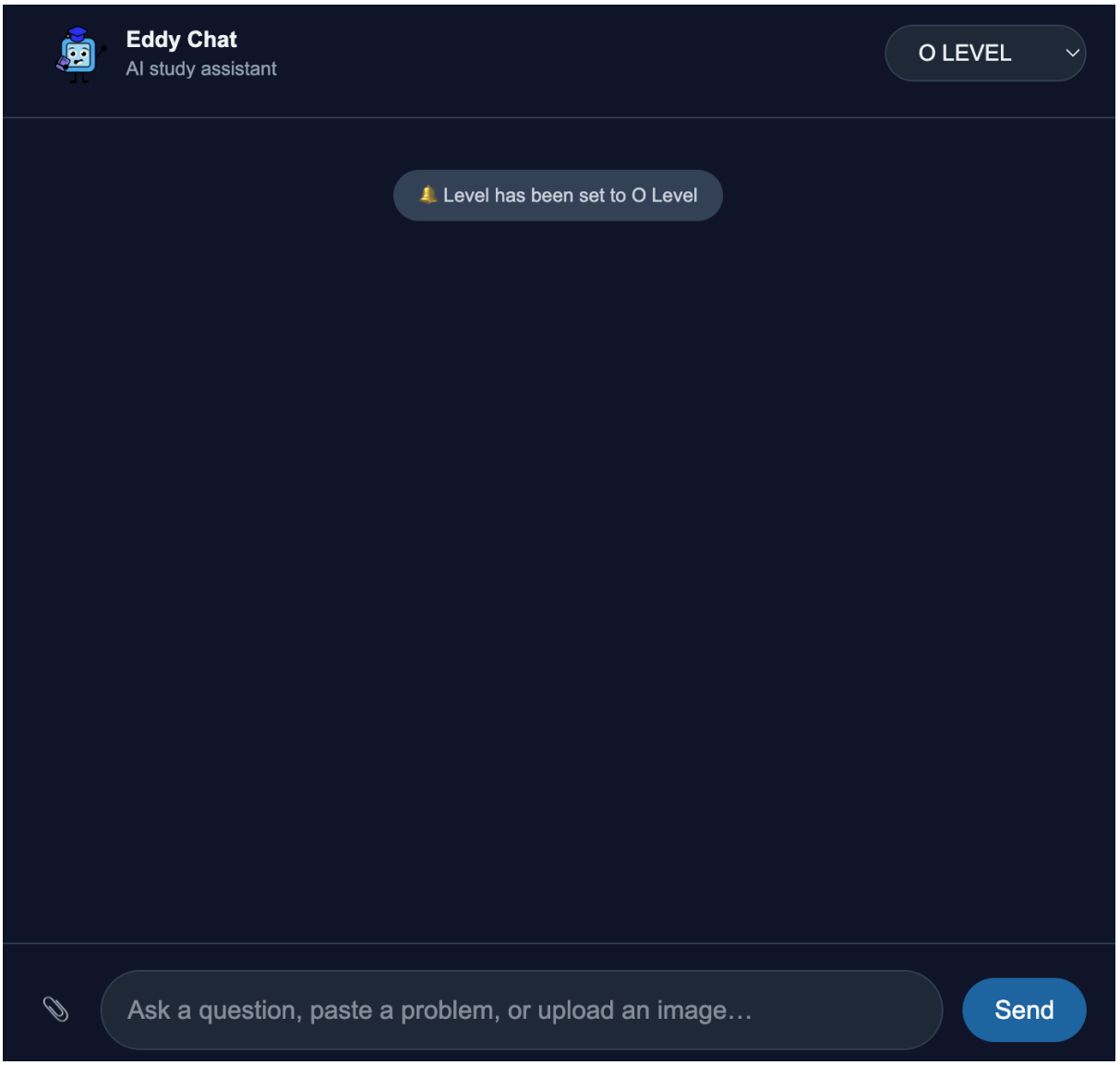
5.2 Chatbot Page

5.2.1 No Input or Level Selected



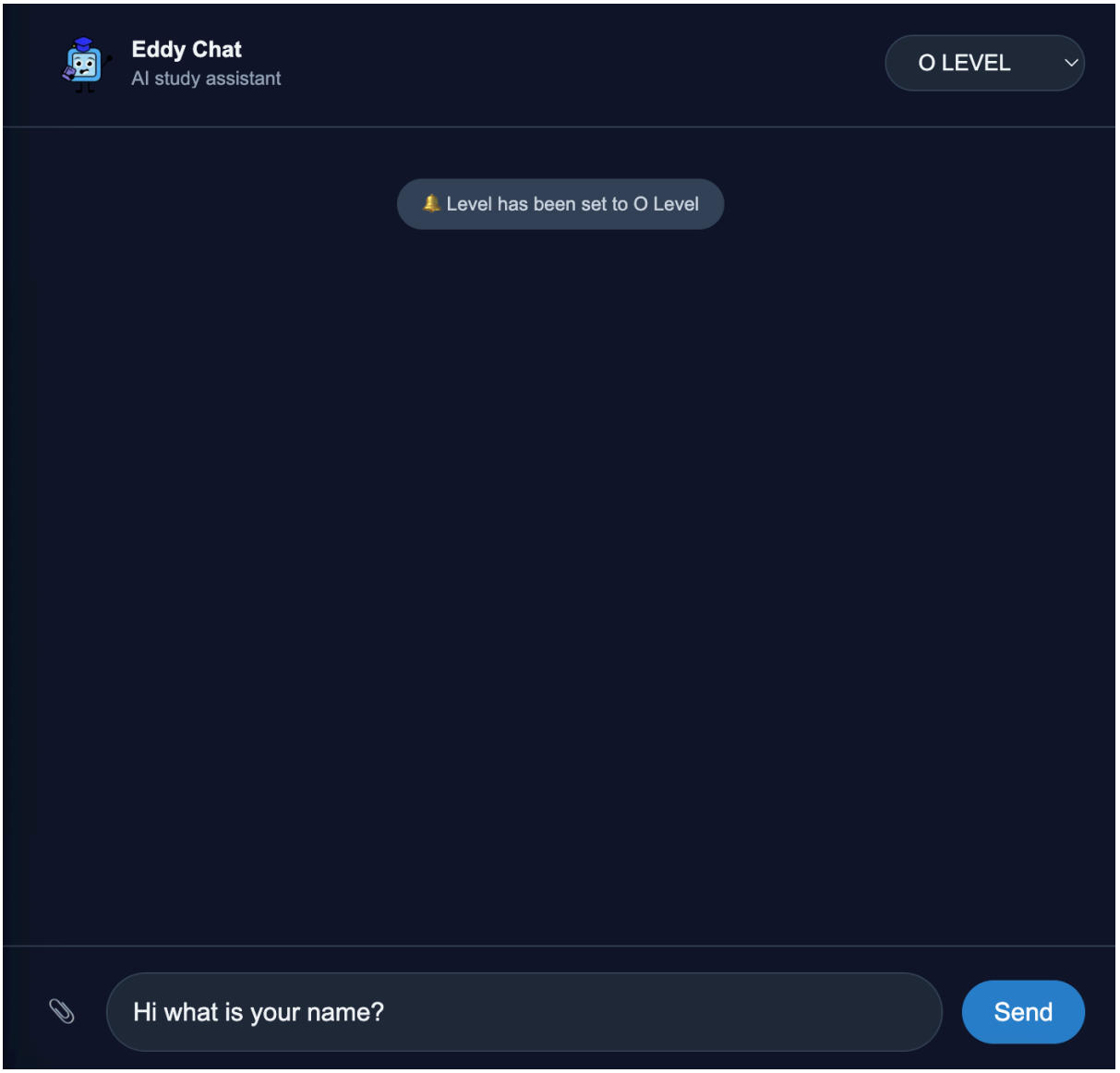
Screenshot of the Chatbot Page - No Input or Level Selected yet

5.2.2 Level Selected



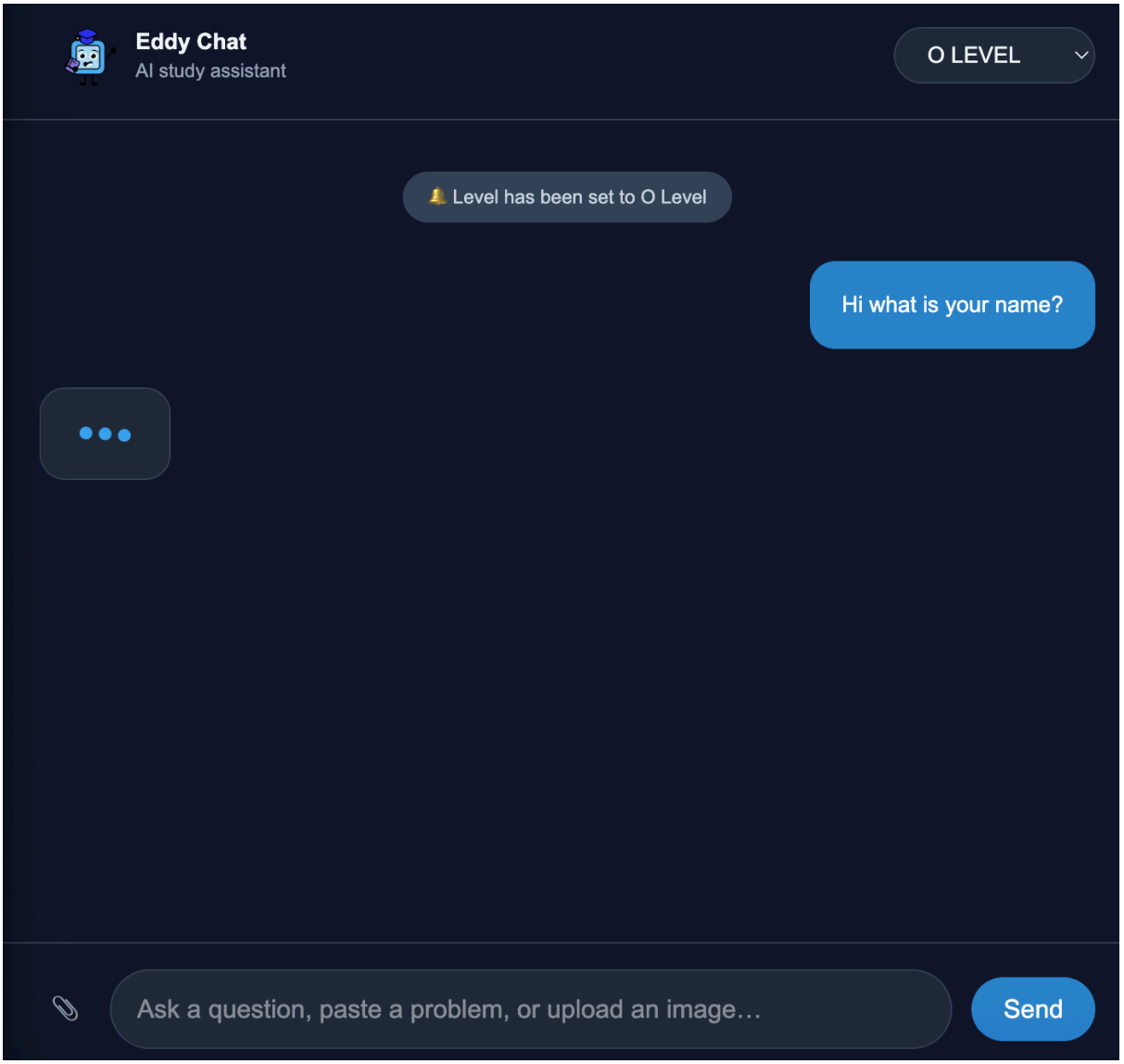
Screenshot of the Chatbot Page - Level Selected

5.2.3 Level Selected and Input Entered



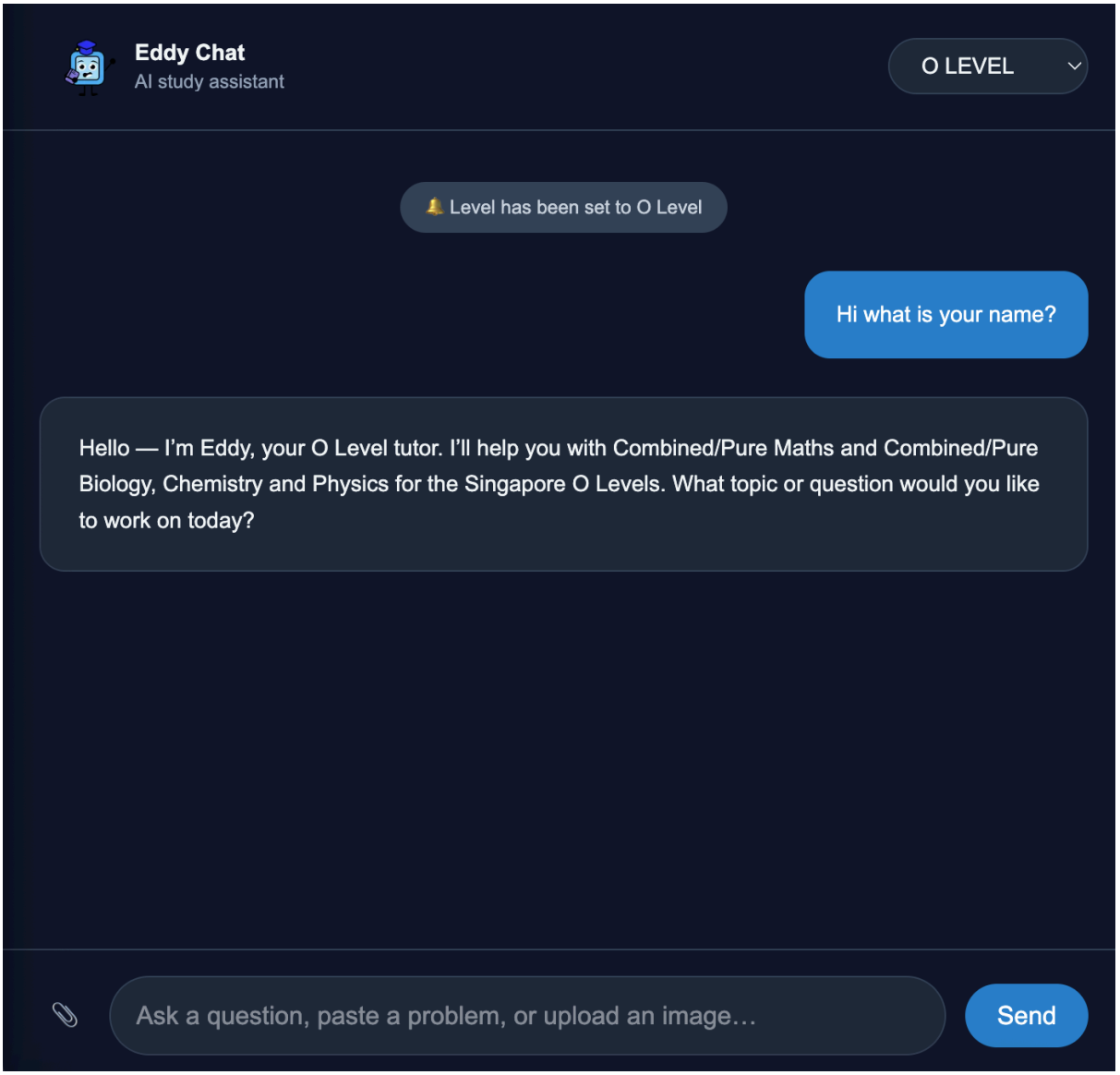
Screenshot of the Chatbot Page - Level Selected and Input entered, but message not sent yet

5.2.4 Message sent and Agent Thinking



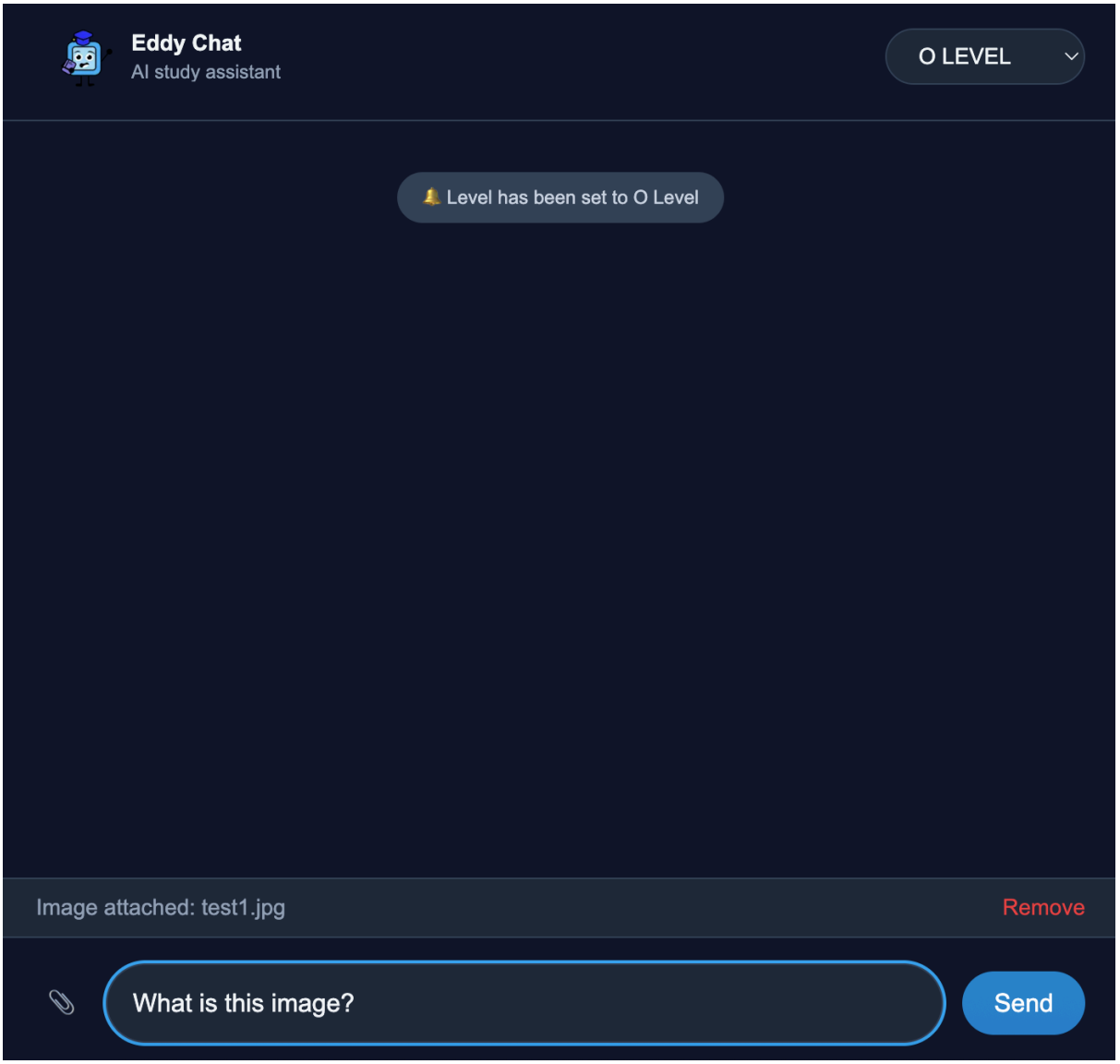
Screenshot of the Chatbot Page - Message sent and Agent thinking

5.2.5 Reply Received



Chatbot Page - Reply Received (Eddy the tutor)

5.2.6 Photo Attached



Screenshot of Chatbot Page - Photo Attached, Level Selected, Input Entered

Question 11: Acids, Bases & Salts

Meili has been instructed by her teacher to make sodium chloride solution by adding hydrochloric acid to sodium hydroxide solution

(a) Identify the process taking place in this reaction? [1]

(b) How could Meili decide when she had added enough hydrochloric acid to react with the sodium hydroxide solution since both hydrochloric solution and sodium hydroxide solution are colourless? [2]

(c) Sodium chloride can be separated from the sodium chloride solution by evaporation. Draw a fully labelled diagram of the apparatus you would use.[1]

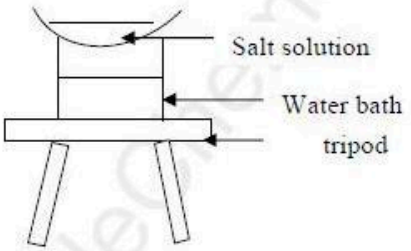
(d) Sodium chloride is an ionic compound and its formula is NaCl.
Draw the 'dot' and 'cross' diagram of this compound. [3]

Student's Answers:

(a) Neutralisation Reaction

(b) Add equivalent amount of hydrochloric acid and sodium hydroxide solutions so that sodium chloride salts solution is formed

(c)



(d)

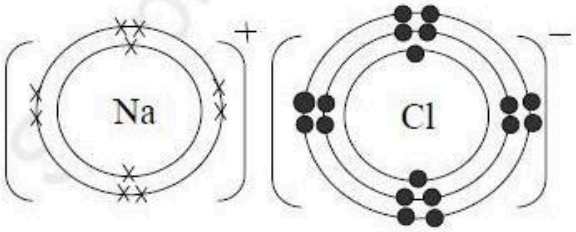


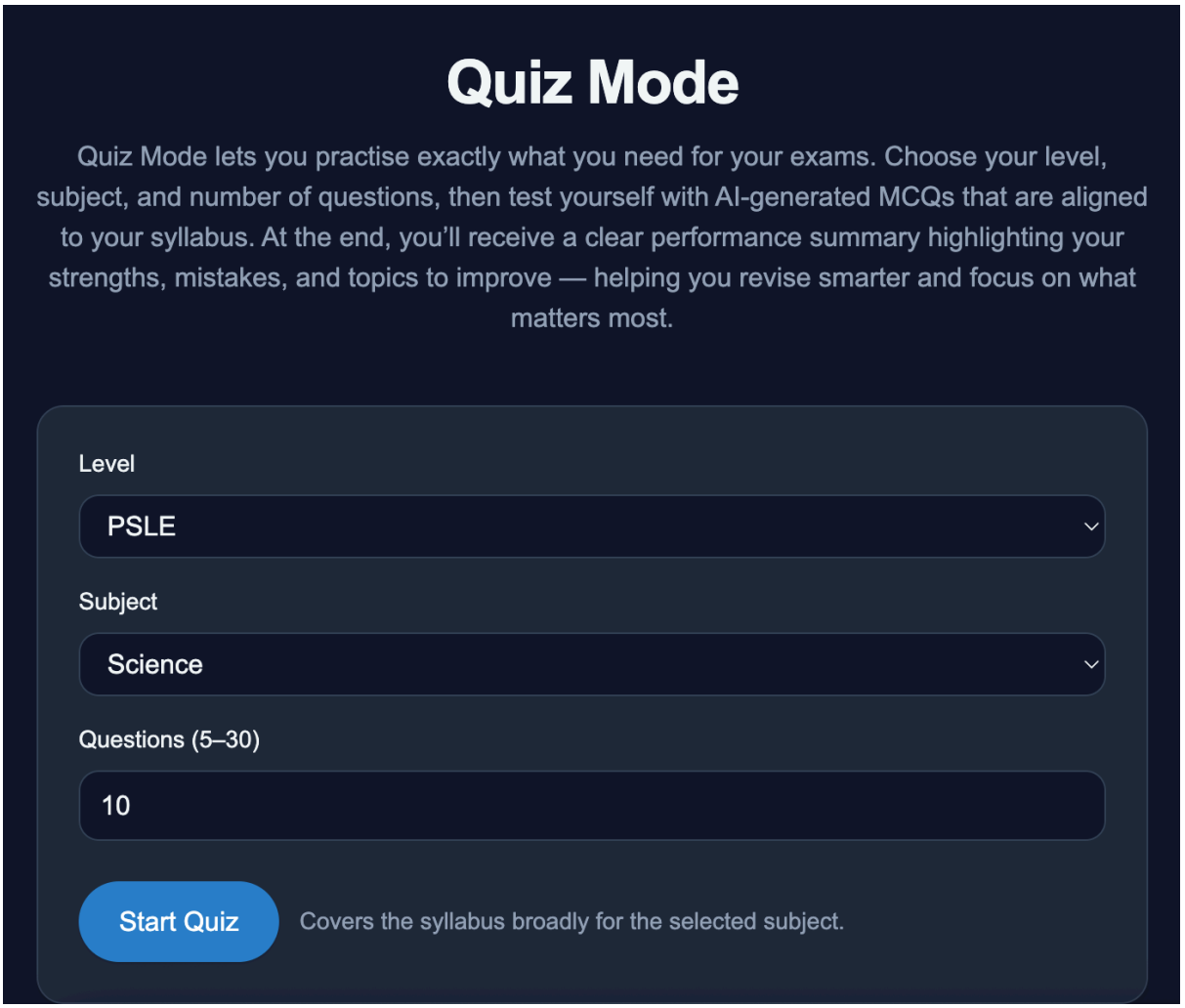
Photo of test1.jpg attached in input

5.2.7 Reply Received with Photo Input

Chatbot Page - Reply Received from Agent (Photo Attached in input)

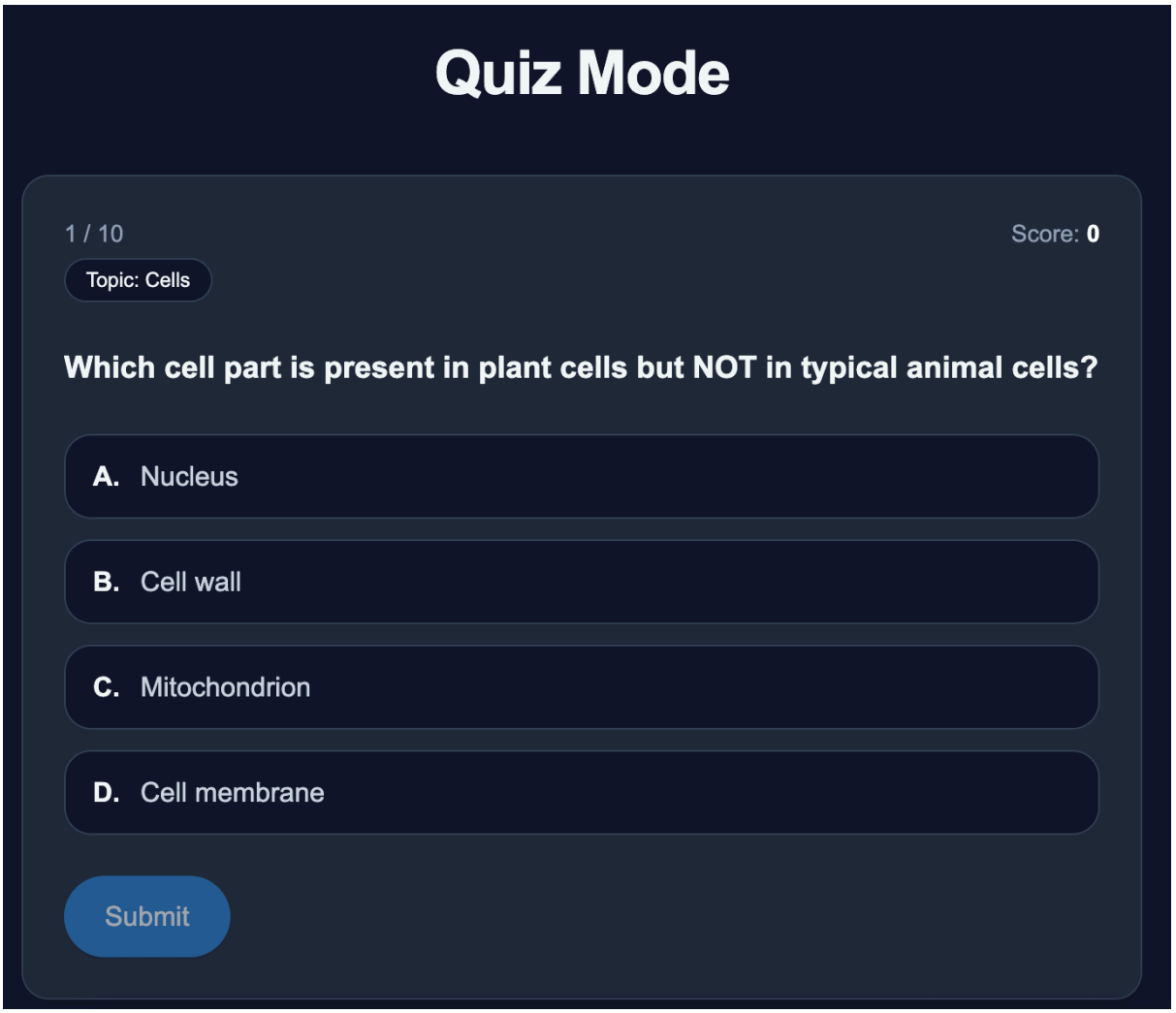
5.3 Quiz Mode Page

5.3.1 Quiz Mode Selection Screen



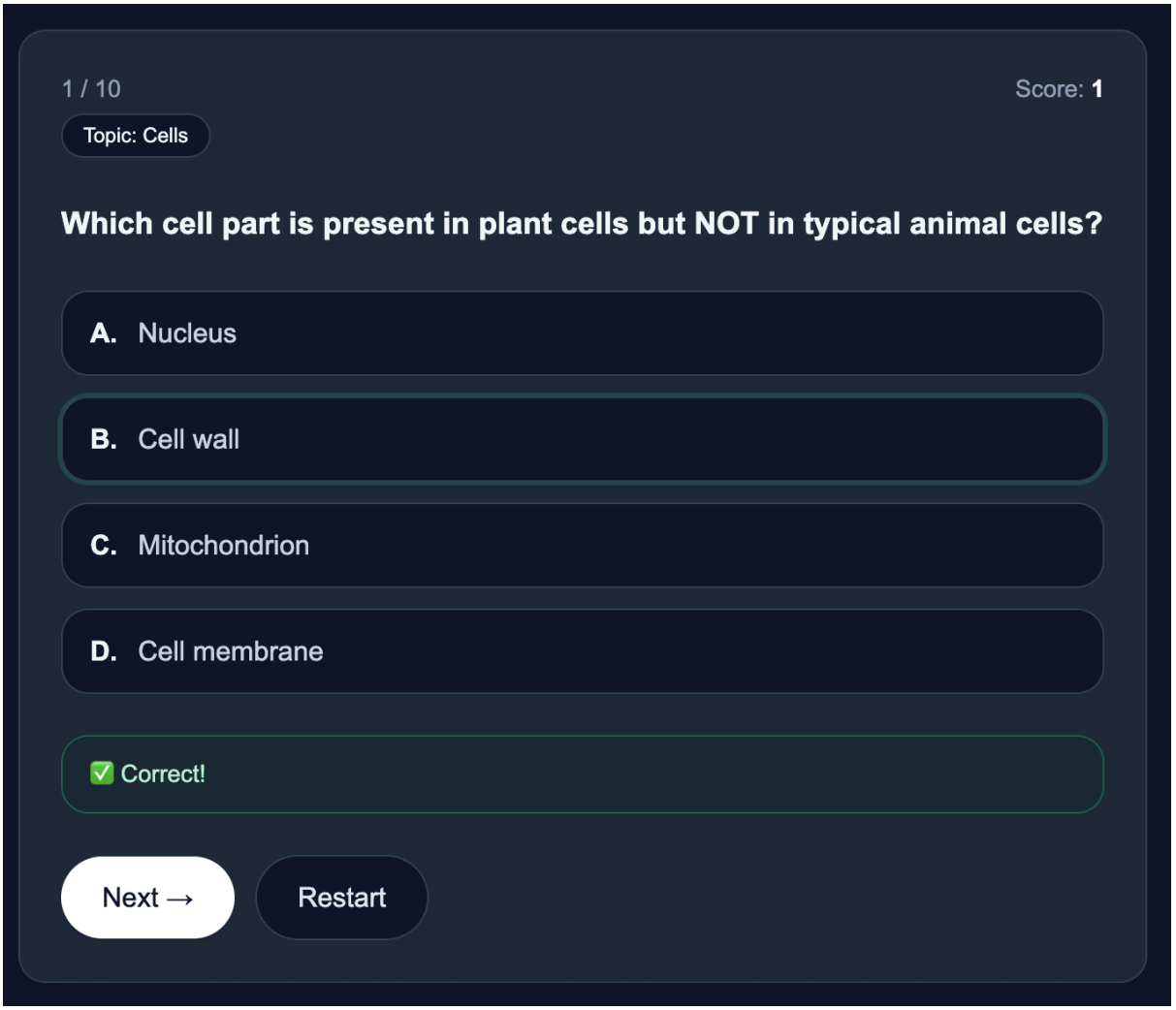
Screenshot of selection screen with options for Education Level, Subject and Number of MCQ questions to generate

5.3.2 Quiz Mode - Multiple Choice Questions



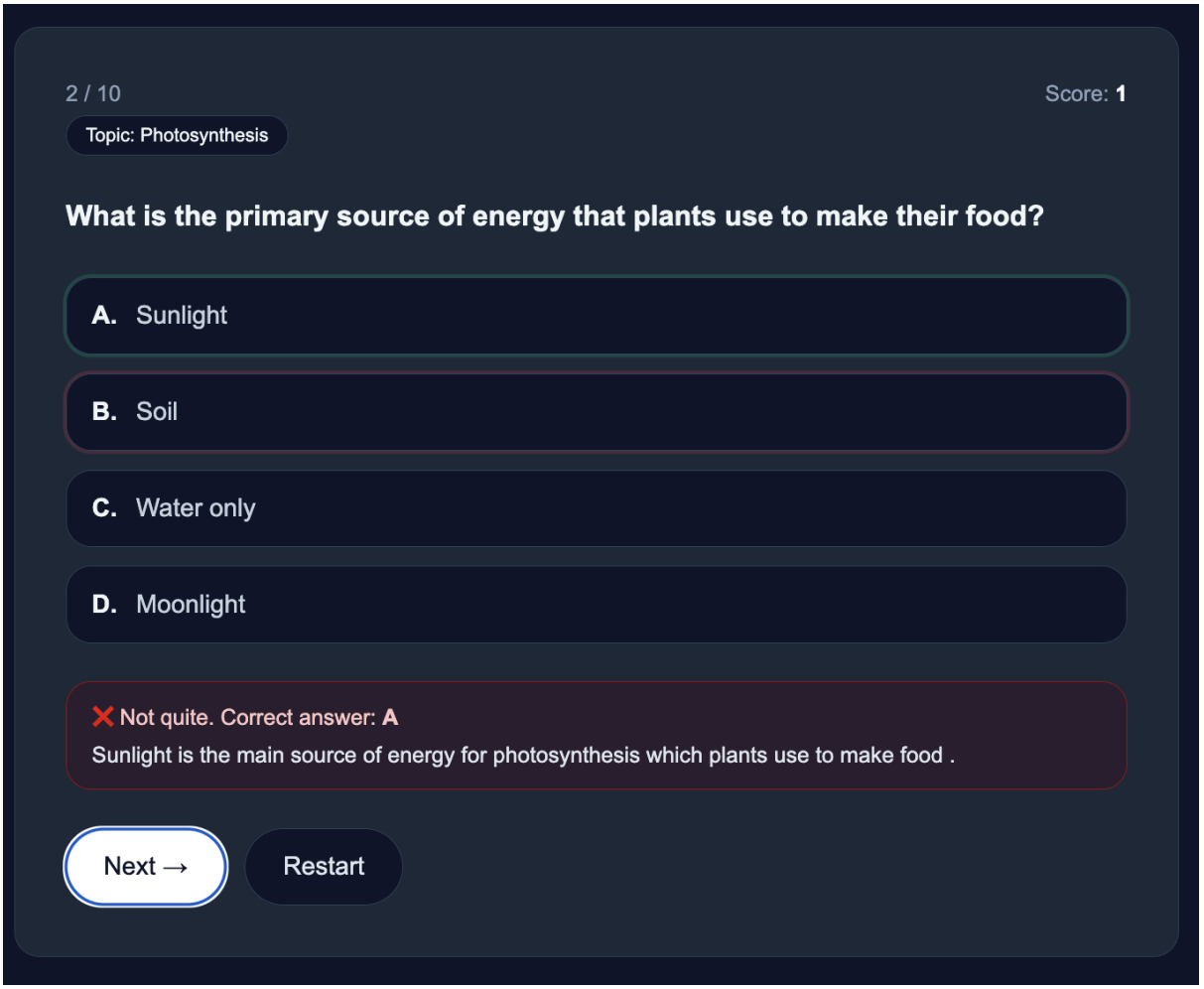
Screenshot of Multiple Choice Questions (MCQs) generated by the LLM displayed on Frontend

5.3.3 Quiz Mode - Correct Answer Selected



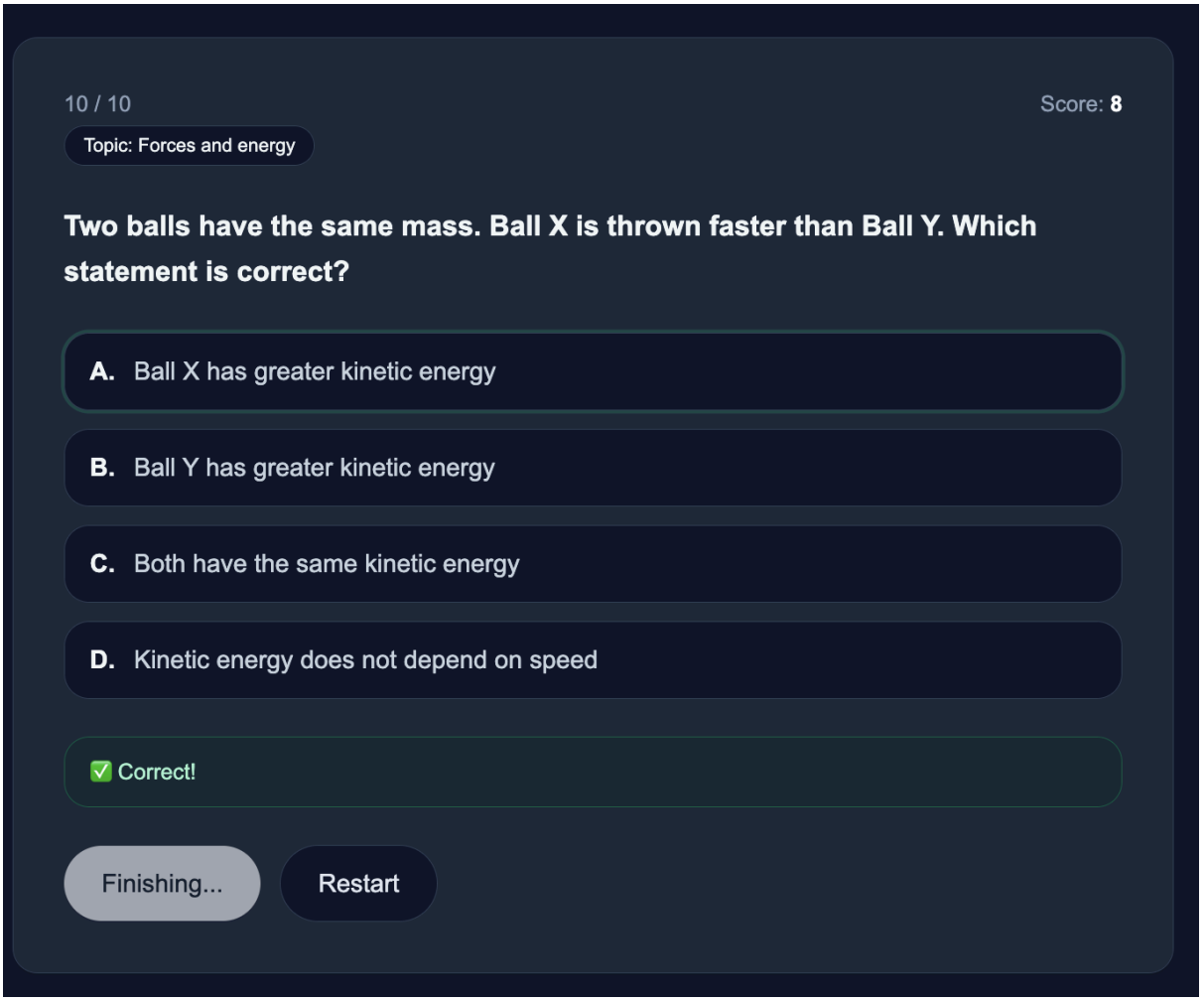
Screenshot of Correct Answer Selected

5.3.4 Quiz Mode - Wrong Answer Selected



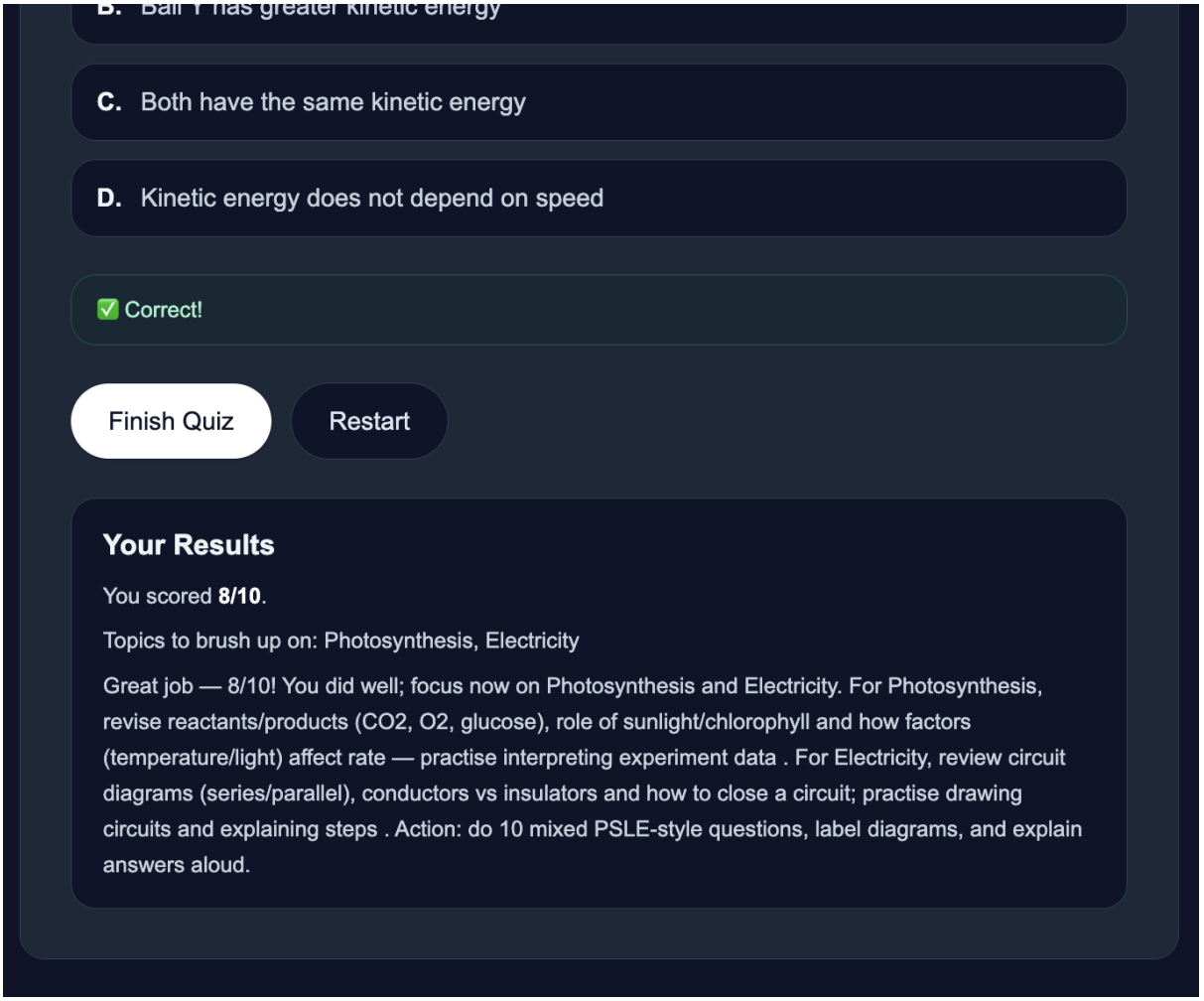
Screenshot of Wrong Answer Selected

5.3.5 Quiz Mode - Quiz Finished and Submission



Screenshot of Quiz Finished and Submission of Quiz to generate summary

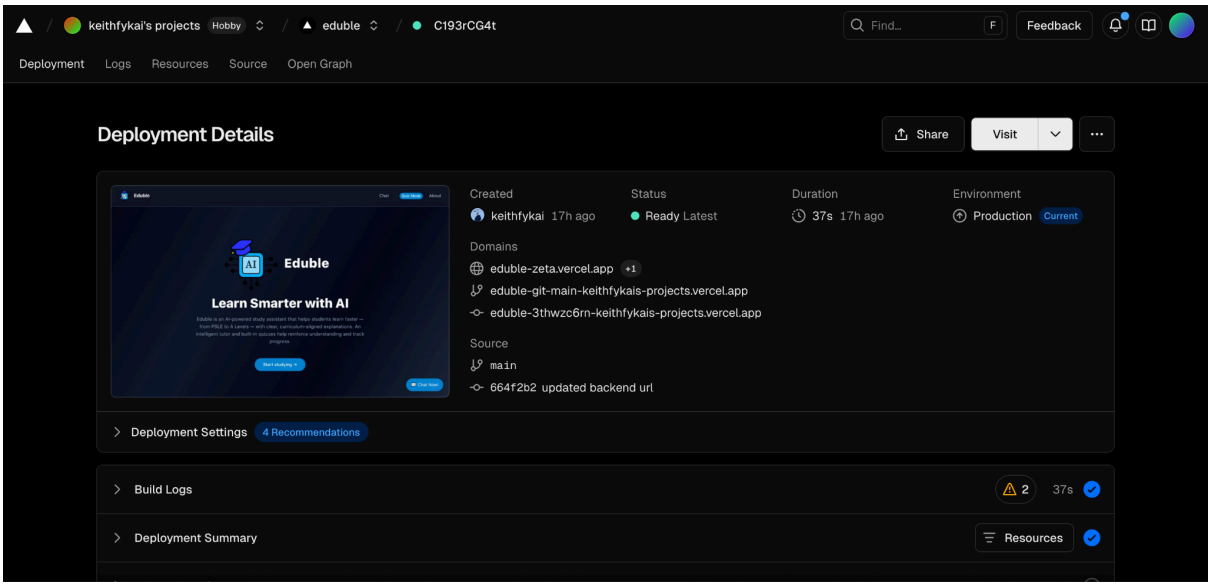
5.3.6 Quiz Mode - Quiz Summary Generated



Screenshot of LLM Generated Quiz Summary

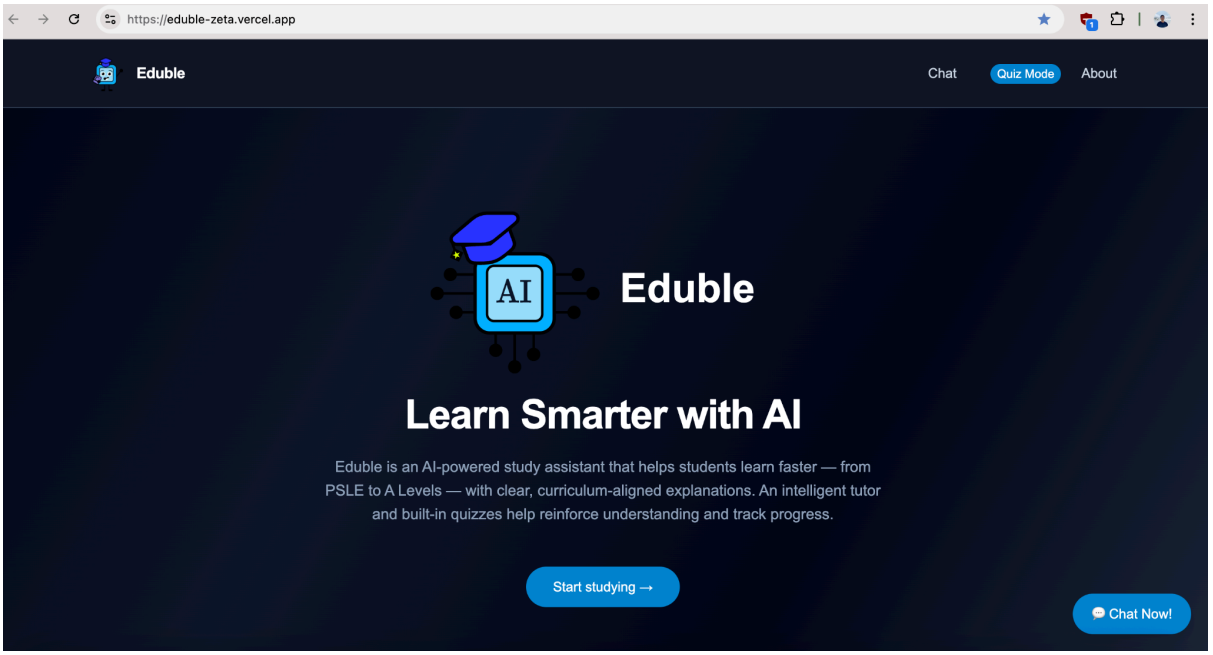
5.4 Vercel Frontend Deployment

5.4.1 Vercel Console



Screenshot of Vercel Console used for deployment of Frontend Code, site now available at https://eduble-zeta.vercel.app/

5.4.2 Deployed Website



Screenshot of Deployed Website

Chapter 6 Testing and Results of Chatbot

The chatbot was evaluated using a curated set of 50 standard multiple-choice questions for each subject and level. The questions were designed to align with the Singapore SEAB/Cambridge syllabus and reflected the level's difficulty. Questions covered all major examinable topics. Accuracy was calculated as the proportion of correct responses over the total number of questions.

6.1 Testing Results:

PSLE			
Subject	**Score**	**Accuracy (%)**	**Grade**
Math	49/50	98	A
Science	50/50	100	A
O Level			
Subject	**Score**	**Accuracy (%)**	**Grade**
Elementary Mathematics	50/50	100	A1
Additional Mathematics	46/50	88%	A1
Pure Chemistry	50/50	100	A1
Combined Chemistry	50/50	100	A1
Pure Physics	50/50	100	A1
Combined Physics	50/50	100	A1
Pure Biology	49/50	98	A1
Combined Biology	50/50	100	A1
A Level			
H1 Mathematics	49/50	98	A
H2 Mathematics	48/50	96	A
H1 Chemistry	50/50	100	A
H2 Chemistry	50/50	100	A

H1 Physics	50/50	100	A
H2 Physics	48/50	96	A
H1 Biology	50/50	100	A
H2 Biology	50/50	100	A

6.1.1 PSLE Results Analysis

At the PSLE level, the chatbot achieved **near-perfect to perfect performance**, with accuracies of **98% in Mathematics** and **100% in Science**, corresponding to an overall **Grade A** standard.

This performance indicates that the chatbot has a **strong mastery of foundational concepts**, including basic numeracy, problem-solving skills, and elementary scientific reasoning. The single error in Mathematics suggests a minor lapse rather than a conceptual weakness, as coverage spanned all major examinable topics.

Overall, the results suggest that the chatbot is **highly reliable for primary-level academic support**, capable of handling PSLE-standard questions with a level of accuracy comparable to top-performing students.

6.1.2 O Level Results Analysis

At the O Level, the chatbot demonstrated **exceptionally strong and consistent performance across all subjects**, with most subjects achieving **100% accuracy (A1 grade)**.

Notably:

- **Pure and Combined Sciences (Chemistry, Physics, Biology)** recorded **near-perfect or perfect scores**, indicating robust subject understanding across both depth-focused and integrated curricula.
- **Elementary Mathematics** achieved **100% accuracy**, reflecting strong procedural fluency and problem-solving skills.
- **Additional Mathematics**, while still achieving an **A1 grade**, recorded a slightly lower accuracy of **88%**, highlighting that higher-order algebraic manipulation and calculus-based reasoning introduce increased complexity relative to Elementary Mathematics.

Despite this small drop, the chatbot's overall O Level performance shows **excellent syllabus coverage, conceptual clarity, and computational accuracy**, suitable for upper-secondary academic use and examination preparation.

6.1.3 A Level Results Analysis

At the A Level, the chatbot maintained **very high performance across both H1 and H2 subjects**, with accuracies ranging from **96% to 100%**, corresponding uniformly to **Grade A outcomes**.

Key observations include:

- **H1 subjects** (Mathematics, Chemistry, Physics, Biology) consistently achieved **98–100% accuracy**, suggesting strong performance on breadth-focused syllabi.
- **H2 subjects**, which place greater emphasis on depth, abstraction, and multi-step reasoning, showed marginally lower but still excellent results in **H2 Mathematics and H2 Physics (96%)**.
- **H2 Chemistry and H2 Biology** achieved **perfect scores**, demonstrating strong handling of concept-dense topics such as reaction mechanisms, energetics, molecular biology, and data interpretation.

The small reduction in accuracy for some H2 quantitative subjects is expected given their **higher cognitive demand**, but the chatbot's results remain firmly within top-grade performance bands.

6.1.4 Overall Evaluation of Educational Chatbot

Across all three educational levels, the chatbot demonstrates:

- **High consistency** in accuracy across subjects and syllabi
- **Strong scalability**, performing reliably from foundational (PSLE) to advanced pre-university (A Level) content
- **Robust syllabus alignment**, with coverage spanning all major examinable topics

The results suggest that the chatbot is **well-suited as an academic support tool** for Singapore's national examinations, with particular strength in science subjects and strong performance in both procedural and conceptual mathematics.

Chapter 7 Future Improvements

While the developed system has been fully deployed and demonstrates strong performance across PSLE, O-Level, and A-Level curricula, there remain several areas where the system can be further enhanced. These improvements focus on increasing pedagogical effectiveness, personalisation, and long-term educational value rather than infrastructure or deployment concerns.

7.1 Chatbot

Although the chatbot achieves high accuracy and strong syllabus alignment, future improvements can enhance its role as an intelligent tutoring system rather than a reactive question-answering tool.

One potential improvement is **adaptive pedagogical scaffolding**. Currently, explanations are generated at a level appropriate to the selected educational stage, but future iterations could dynamically adjust explanation depth based on student interaction patterns. For example, repeated follow-up questions or incorrect attempts could trigger more step-by-step explanations, worked examples, or simplified analogies.

Another enhancement involves **explicit source grounding and transparency**. While the chatbot internally uses Retrieval-Augmented Generation (RAG), future versions could optionally display the syllabus sections or retrieved documents used to generate responses. This would increase student trust, improve explainability, and support independent verification of answers.

Finally, the chatbot could benefit from **long-term learning memory**, where recurring misconceptions, frequently asked topics, or historical performance are stored and used to personalise future interactions. This would allow the chatbot to provide more targeted assistance over time, further approximating the behaviour of a human tutor.

7.2 Quiz Mode

Quiz Mode represents a significant extension beyond conversational learning by introducing active assessment and immediate feedback. However, several enhancements can further improve its effectiveness and educational impact.

A key limitation of the current implementation is that quizzes are generated at the **subject-wide level**, rather than being constrained to specific topics or subtopics. Future improvements could allow

students to explicitly select topics (e.g. “Photosynthesis”, “Quadratic Equations”, or “Kinetics”), enabling more focused revision and targeted practice.

Another potential enhancement is **adaptive difficulty adjustment**. By analysing student responses in real time, the system could dynamically adjust question difficulty, increasing complexity when a student performs well and revisiting foundational concepts when errors are detected. This approach aligns with established principles of adaptive learning and formative assessment.

Additionally, Quiz Mode could be extended beyond multiple-choice questions to include **short-answer or structured-response questions**, particularly for higher-level subjects such as A-Level Mathematics and Sciences. This would better reflect real examination formats and allow assessment of deeper reasoning rather than recognition-based answering.

Finally, future versions could incorporate **longitudinal performance tracking**, allowing students to view progress over time, identify persistent weak areas, and receive personalised revision recommendations across multiple quiz sessions. Such analytics would further strengthen Quiz Mode as a diagnostic and revision tool rather than a one-off testing mechanism.

Chapter 8 Conclusion

This project presented the design, implementation, and deployment of a curriculum-aware educational chatbot powered by Retrieval-Augmented Generation (RAG), tailored to Singapore's pre-tertiary education system. By integrating syllabus-aligned educational materials with a large language model, the system addresses key limitations of general-purpose generative AI tools, particularly hallucination and misalignment with examination requirements.

In addition to conversational academic support, the system incorporates an AI-powered **Quiz Mode**, enabling structured, exam-style practice with immediate feedback and personalised performance summaries. This feature extends the system beyond passive assistance into active learning and formative assessment, reinforcing understanding through retrieval practice and targeted revision.

Experimental evaluation across PSLE, O-Level, and A-Level subjects demonstrates consistently high accuracy, with performance corresponding to top-grade outcomes across mathematics and science disciplines. These results highlight the effectiveness of combining educational-level awareness with retrieval grounding to support learning across varying levels of academic complexity.

Overall, this work demonstrates that curriculum-aligned, RAG-based educational systems can function as scalable and accessible learning support tools. By integrating conversational tutoring, automated assessment, and personalised feedback within a deployed full-stack application, the project establishes a strong foundation for future intelligent tutoring systems that aim to enhance educational equity and learning outcomes within structured education environments.

References

[1] Data Science Stack Exchange. (2023). ChatGPT's architecture: Decoder-only or encoder-decoder? https://datascience.stackexchange.com/questions/118260/chatgpts-architecture-decoder-only-or-encoder-decoder

[2] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., … & Polosukhin, I. (2017). Attention is all you need. Advances in Neural Information Processing Systems, 30. https://arxiv.org/abs/1706.03762

[3] Google Cloud. (2023). What are AI hallucinations? https://cloud.google.com/discover/what-are-ai-hallucinations

[4] Salvagno, M., Taccone, F. S., & Gerli, A. G. (2024). Reliability of references generated by ChatGPT for systematic reviews: cross-sectional study. JMIR Medical Informatics, 12, e50514. https://pubmed.ncbi.nlm.nih.gov/38776130/

[5] Hugging Face. (2022). Mixture of experts explained. https://huggingface.co/blog/moe

[6] Sun, Y., Dong, L., & Wei, F. (2023). LongNet: Scaling transformers to 1,000,000,000 tokens. arXiv preprint. https://arxiv.org/abs/2311.12351

[7] Google Research. (2021). Multimodal bottleneck transformer (MBT): A new model for modality fusion. https://research.google/blog/multimodal-bottleneck-transformer-mbt-a-new-model-for-modality-fusion/

[8] Chen, M., et al. (2023). Looking back at speculative decoding. Google Research Blog. Retrieved from https://research.google/blog/looking-back-at-speculative-decoding/

[9] Lewis, P., Perez, E., Piktus, A., Karpukhin, V., Goyal, N., Küttler, H., … & Riedel, S. (2020). Retrieval-augmented generation for knowledge-intensive NLP tasks. Advances in Neural Information Processing Systems, 33, 9459–9474. https://arxiv.org/abs/2005.11401

[10] OpenAI. (2025). GPT-5 system card. https://cdn.openai.com/gpt-5-system-card.pdf

[11] Channel News Asia. (2024, April 30). Secondary students in Singapore using AI tools for homework and grades.

https://www.channelnewsasia.com/cna-insider/secondary-students-using-artificial-intelligence-ai-homework-grades-singapore-5146911

[12] The Straits Times. (2016, July 25). 7 in 10 parents send their children for tuition: ST poll. https://www.straitstimes.com/singapore/education/7-in-10-parents-send-their-children-for-tuition-st-poll

[13] The Straits Times. (2024, February 19). Singapore families spent $1.8 billion on private tuition for children in 2023. https://www.straitstimes.com/singapore/spore-families-spent-1-8-billion-on-private-tuition-for-children-in-2023

[14] T. B. Brown *et al.*, "Language Models are Few-Shot Learners," *arXiv:2005.14165*, 2020. Available: https://arxiv.org/abs/2005.14165

[15] L. Huang *et al.*, "A Survey on Hallucination in Large Language Models: Principles, Taxonomy, Challenges, and Open Questions," *arXiv:2311.05232*, 2023. Available: https://arxiv.org/abs/2311.05232

[16] J. Wei, X. Wang, D. Schuurmans, *et al.*, "Chain-of-Thought Prompting Elicits Reasoning in Large Language Models," *arXiv preprint arXiv:2201.11903*, 2022. Available: https://arxiv.org/abs/2201.11903

[17] X. Wang, J. Wei, D. Schuurmans, *et al.*, "Self-Consistency Improves Chain of Thought Reasoning in Language Models," *arXiv preprint arXiv:2203.11171*, 2022. Available: https://arxiv.org/abs/2203.11171

[18] M. Madaan, S. Tandon, P. Gupta, *et al.*, "Self-Refine: Iterative Refinement with Self-Feedback," *arXiv preprint arXiv:2303.17651*, 2023. Available: https://arxiv.org/abs/2303.17651

[19] P. Lewis *et al.*, "Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks," *arXiv:2005.11401*, 2020. Available: https://arxiv.org/abs/2005.11401 Available: https://arxiv.org/abs/2005.11401

[20] V. Karpukhin *et al.*, "Dense Passage Retrieval for Open-Domain Question Answering," *arXiv:2004.04906*, 2020. Available: https://arxiv.org/abs/2004.04906

[21] N. Reimers and I. Gurevych, "Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks," *arXiv:1908.10084*, 2019. Available: https://arxiv.org/abs/1908.10084

[22] J. Johnson, M. Douze, and H. Jégou, “Billion-scale similarity search with GPUs,” *arXiv:1702.08734*, 2017. Available: https://arxiv.org/abs/1702.08734

[23] Y. A. Malkov and D. A. Yashunin, “Efficient and robust approximate nearest neighbor search using Hierarchical Navigable Small World graphs,” *arXiv:1603.09320*, 2016. Available: https://arxiv.org/abs/1603.09320

[24] M. Li *et al.*, “TrOCR: Transformer-based Optical Character Recognition with Pre-trained Models,” *arXiv:2109.10282*, 2021. Available: https://arxiv.org/abs/2109.10282

[25] Y. Huang *et al.*, “LayoutLMv3: Pre-training for Document AI with Unified Text and Image Masking,” *arXiv:2204.08387*, 2022. Available: https://arxiv.org/abs/2204.08387

Appendix A: Data Sources and Educational Resources

A.1 Primary School Leaving Examination (PSLE)

Official Syllabi

- Ministry of Education (MOE), Singapore. *Primary Mathematics Syllabus (Primary 1–6)*, updated October 2025.
  Available at:
  https://www.moe.gov.sg/-/media/files/primary/2021-primary-mathematics-syllabus-p1-to-p6-updated-october-2025.pdf
- Ministry of Education (MOE), Singapore. *Primary Science Syllabus*, updated May 2024.
  Available at:
  https://www.moe.gov.sg/-/media/files/primary/syllabus/primary-science-syllabus-2023_may24.pdf

Examination Rules and Regulations

- Singapore Examinations and Assessment Board (SEAB). *PSLE Rules and Regulations*.
  Available at: https://file.go.gov.sg/seab-rulesandregulations-psle.pdf

Past Year Examination Papers

- Mendaki. *PSLE Past Year Examination Papers*.
  Available at: https://www.mendaki.org.sg/resources/?mode=exams-paper

---

A.2 Singapore-Cambridge GCE O-Level

Official Syllabi

- Singapore Examinations and Assessment Board (SEAB). *GCE O-Level Subject Syllabi*.
  Available at: https://www.seab.gov.sg/

Past Year Examination Papers

- Public educational repositories providing O-Level past year examination papers and sample answers.
  Example source: https://grail.moe/library

Note: Official O-Level textbooks are proprietary and not publicly accessible. Therefore, syllabus documents were used as the primary reference, supplemented by publicly available examination papers.

A.3 Singapore-Cambridge GCE A-Level

Official Syllabi

- Singapore Examinations and Assessment Board (SEAB). *GCE A-Level Subject Syllabi*. Available at: https://www.seab.gov.sg/

Supplementary Learning Materials

- School-developed notes and publicly available examination resources compiled from multiple institutions, aligned to SEAB syllabi.

Past Year Examination Papers

- Public educational repositories providing A-Level past year examination papers. Example source: https://grail.moe/library

Note: A-Level subjects do not mandate a single official textbook. Schools select or develop their own instructional materials.

A.4 International Baccalaureate (IB)

Syllabi, Past Year Papers, and Mark Schemes

- International Baccalaureate Organization (IBO). *IB Subject Guides and Examination Materials*.
  Available via public archival repository: https://dl.ibdocs.re/

Note: Due to repository size constraints, only subject-relevant documents were selected for inclusion.