Enhancing Formative E-Assessments through Substituting the Teacher-User with Large Language Models

---

Author: Samuel Braham
Student ID: 2123325
Program Name: B.Sc. Computer Science
Credits: 40 Credit Project
Supervisor's name: Wendy Yanez Pazmino
Word count: 10931

Abstract

Formative e-assessments are an effective tool for learning and revision. However, these tools are often dependent on the work of teachers to create, mark and provide feedback. Without this input, the e-assessments wouldn't be an effective tool for students. Large language models (LLMs) could be used to create questions for the student, mark the questions, and then give feedback on how the student has done and how they could improve in the future. This work proposes the development of a student-led LLM-powered formative e-assessment web application to improve students' educational independence and free more time for teachers.

The development followed an iterative development methodology using Behaviour Driven Development. Each sprint followed the creation of unit tests, the development of features and the subsequent refactoring of code and integration testing. Development included the use of LLMs in the creation of quizzes, marking of answers and giving of feedback. In addition, visualisations of student progress and information about the app's functionality were added. Following this development, an evaluation was carried out to assess the quality of the application.

The application confirmed that LLMs can successfully be used in an e-assessment application for the development of these features. Despite this, due to current limitations in LLMs, such as hallucinations and their need for high-performance machines, the application has a chance to present inaccurate facts which could harm learning. With this in mind, the project concludes it would be irresponsible to use this application in its present state for teaching purposes. The limitations and future work to mitigate this are discussed.

Git repository: https://git.cs.bham.ac.uk/projects-2023-24/sxb1567

Contents

Introduction

Assessments are a core element of student education. As explained in the National Foundation for Education Research's (NFER) "Assessment 101" [1] there are two kinds of assessment: summative assessment and formative assessment. Summative assessment is separate from learning, coming after a subject has been taught, to evaluate understanding. Formative assessment is integrated into the learning process to practise understanding and help students and teachers guide that learning.

Formative assessments positively affect the learning process [2, 3], linked to their recall practice [4], and provide useful feedback about where a pupil is on their learning journey. However, they are hugely time-consuming with time needed to understand what part of the topic assesses all the class, to think of a broad range of questions, and then write, format and print the questions. After the assessment, more time is spent marking, and then, finally, feedback is given, which is often brief due to this lengthy process. All this only adds to the excessive workload already on teachers today.

Formative e-assessment technology can standardise and streamline the process, making it quicker and easier for both teachers to create assessments and students to take them. This is done through the standardisation of processes and well-designed systems. This can be seen in existing formative e-assessment applications. The most well-known of these would be Kahoot! [5] (Appendix 1) created in 2013, with other options being Quizizz [6], MyMaths [7], Quizlet [8] or as a feature of the multi-purpose application Canvas [9]. Although having a variety of different qualities, all applications share the core element of an e-assessment application: the ability for a user to take a formative assessment on a certain topic, intending to improve their understanding.

Most current applications follow a teacher-led model[1] dependent upon the engagement of a teacher or expert before a student can effectively utilise it (Figure 1). If students were to use Kahoot!, for example, to assess their understanding, they can only do so if a teacher/expert has created a quiz on that topic. Students, still learning, would be unable to create an effective test for themselves. This dependency does little to alleviate the time input, as a teacher is still the central component in that process.

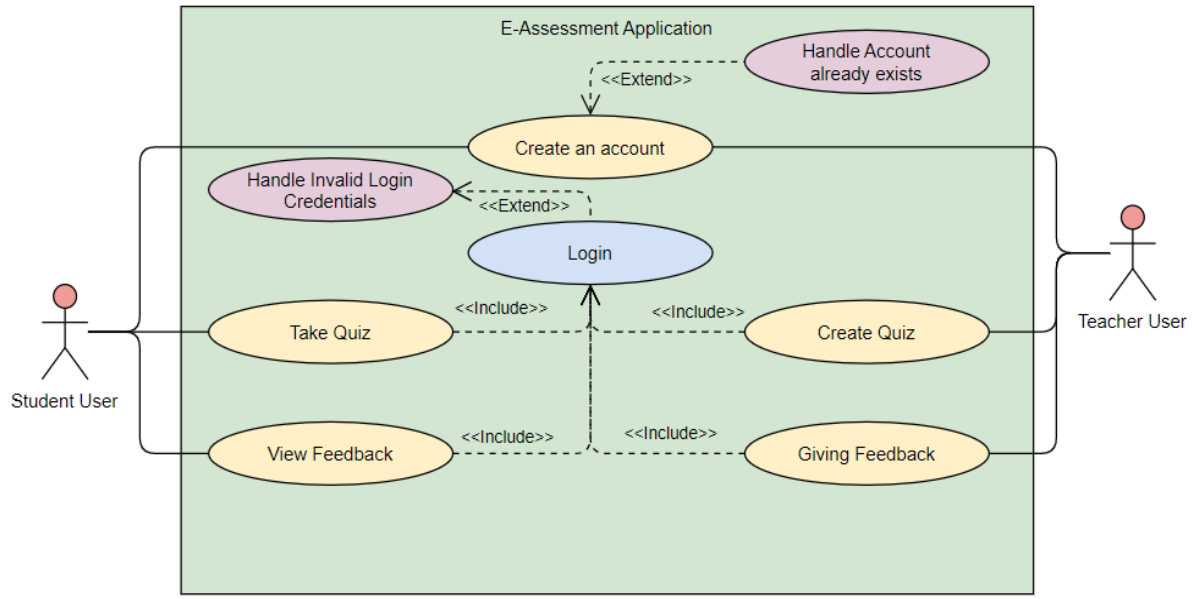

Figure 1. Teacher-led Use Case Diagram

[1] See appendix 1

In contrast, a student-led model is not dependent on any other users and instead facilitates the use of features by the student users alone. This reduces the need for teachers' time, greatly improving student choice in how and what they learn, facilitating self-directed learning and autonomy, and improving both performance and well-being [10]. This substitution has precedent in existing applications. MyMaths, for example, has been designed by experts in mathematics to allow the student to engage with the application independently. However, this only serves to make assessments static and little can be done by the user to access more content. The core problem of student-led applications is what will substitute the teacher. One possible solution is generative artificial intelligence in the form of large language models (LLMs) (Figure 2).

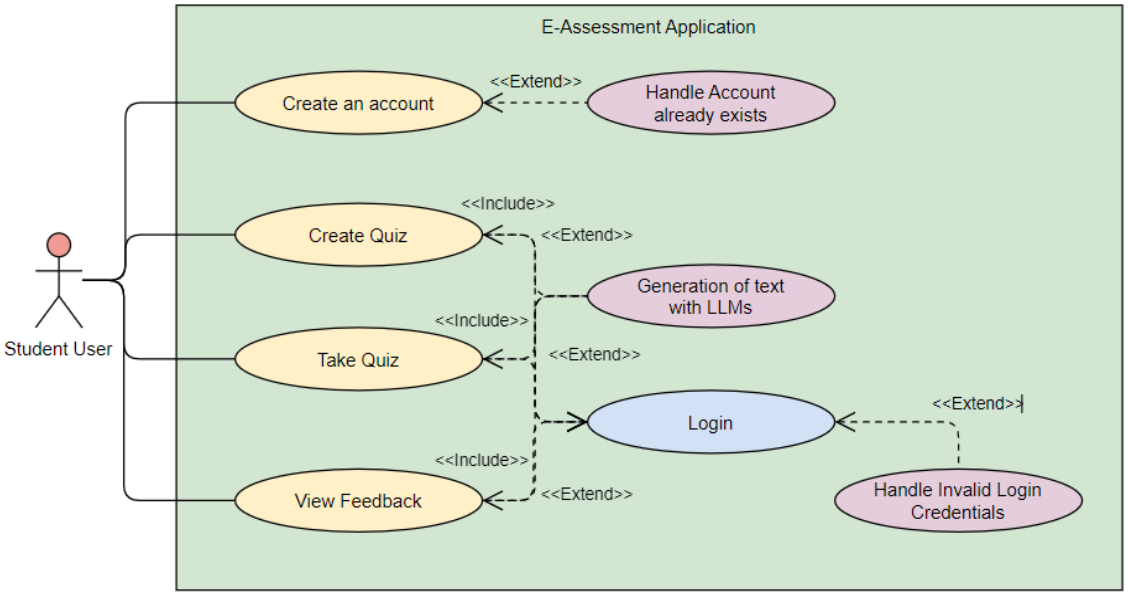

Figure 2. LLM-facilitated Student-Led Use Case Diagram

Recent advances in LLM technology, centrally the use of transformer models [11], can have a major impact on the ecosystem of e-assessment tools. Many multi-task learning (MTL) models have been developed such as BERT [12] and ChatGPT [13]. There are uses for both MTL and single-task LLMs in educational technology [14]. Focusing on formative assessments, LLMs can be used to enhance e-assessments by automating or aiding the generation of questions, marking long answer questions, and giving feedback to students. This would alleviate these tasks from a teacher's workload and allow them to focus on other key aspects of their job such as planning lessons and pastoral care.

Existing applications have begun to utilise LLMs in recent updates but with limited implementation. Fundamentally, they remain teacher-led, focusing on assisting question creation for teachers [15] rather than attempting to move control towards the student. As well as this, they have been known to produce unexpected outputs, often called hallucinations. Within academic text generation, hallucinations are instances when the outputs of an LLM "lacks fidelity to factual accuracy, reality, or the intended context" [16]. The existence of this in a system attempting to support students' learning is potentially damaging, leading to students learning the wrong material or losing trust in the application. Similarly, if questions are irrelevant to context then time is wasted answering them.

Aim and Objectives

This work proposes the development of a student-led LLM-powered formative e-assessment web application to improve students' educational independence and free more time for teachers. Research into existing users and applications, as well as a review of the academic literature, is carried out and, following this, an application is developed.

The core aims of the LLM formative e-assessment application consist of:

1) Use LLMs to automate the generation of quizzes and store these in a database.
2) Have a user be able to take multiple-choice and short-answer questions.
3) Use LLMs to mark the student's answers for short-answer questions.
4) Use LLMs to automate the adaptive feedback of students' answers.

The following additional aims are considered to ensure the effective utilisation of the technology and improve the experience for the user.

5) Use prompt and response handling to allow a streamlined and effective use of the LLM, making the experience quick and convenient to use.
6) Investigate and utilise hallucination mitigation techniques to minimise the number of erroneous LLM outputs.
7) Track the progress of the user over each attempt of a quiz, to cultivate and motivate the development of learning over time.
8) Add clear information about the function of the LLM integration and how it works to help users understand its functionality and limitations to ensure effective, explainable and ethical use.

The application is tested and evaluated through a variety of means focusing on how well it addresses the aim of the project.

Project Limitations

It is important to address the scope of this project and the limitations in cost, time, and hardware access. State-of-the-art models, such as OpenAI's GPT-4 are out of scope due to their cost per token sent and received [17].

This moves attention towards two options using open-source models. The first is using off-the-shelf open-source MTL-LLMs as they come, using prompt engineering to focus their task where it is needed. The second is to fine-tune these open-source LLMs to be more optimal at the specific tasks. Although the latter would certainly result in better quality outputs [18] fine-tuning multiple models would be highly time-consuming, costly and would detract from the overall aim of the project[2]. Likewise, the effects of fine-tuning on question creation have already been explored[3]. Considering these points, off-the-shelf open-source MTL-LLMs are used in this project.

Lastly, the application could theoretically be used by any student at any level, although another limitation of the potentially usable models is the level of content they can get correct. GCSE and A-Level content is accurate while university-level content is less so. For this reason, the project focuses on GCSE and A-level as well as some general knowledge areas.

Social and Ethical Considerations

There must be clear ethical considerations when talking about the replacement of a human with AI. The specific case investigated by the project would potentially involve removing teachers from the role of formative assessment creators, however, this is only one of the many things teachers do. Currently, teachers have very limited time so removing this task from them would give them more time to fulfil their other requirements, improving their capacity as teachers. For that reason, I do not consider this replacement to be of ethical concern.

[2] This is especially true when we consider that a separate fine-tuned model would need to be made for question generation, marking, and feedback, either greatly increasing the work required or reducing the scope of the application.
[3] See literature review for sources on this

Background and Literature Review

Survey of Potential Users

A pre-survey[4] to collect user information was carried out by asking several expected users what e-assessment applications they have used, what they think of them and what they would want to see in a future e-assessment application. It is important to state this research first as it motivates the specifics of later research.

The key findings are that, out of the people who took the survey, Kahoot! [5], Quizlet [8], Quizizz [6], MyMaths [7], Canvas [9], and Mr Hogisty Maths are the e-assessment applications people have used before, with significantly varying use (Figure 3). This data is used to assess which apps to investigate, looking at the top five.

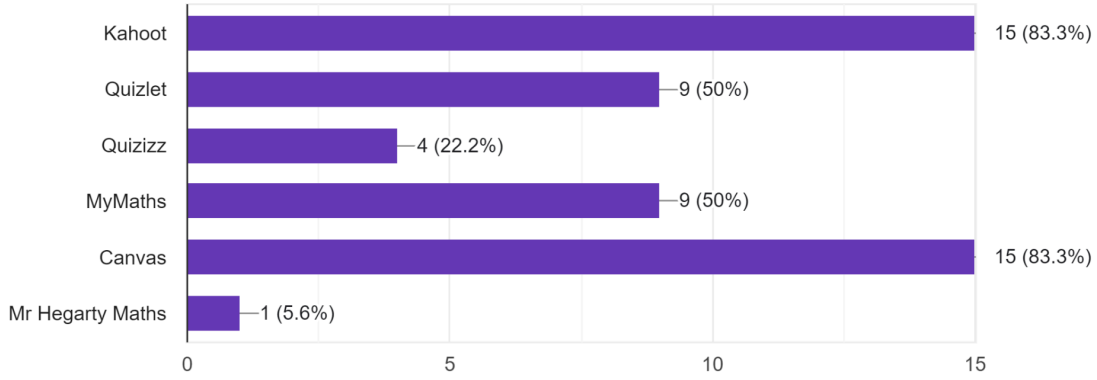

Figure 3. Which of these e-assessment applications have you used before? Please list any others you have used.

Figure 4 sets out the main features people want to see in an application with users able to choose from a list or add their own features. The majority of responses are in favour of the app being *quick and convenient to use*, that it *has questions that match a user's skill level* and that it gives *great feedback on the questions that have been taken*. These are important areas of investigation and things to consider when creating requirements. Interestingly, when asked what people thought of the feedback provided by existing applications (Figure 5) most people responded *poor* or *very poor* highlighting this as a key area of improvement over existing applications.

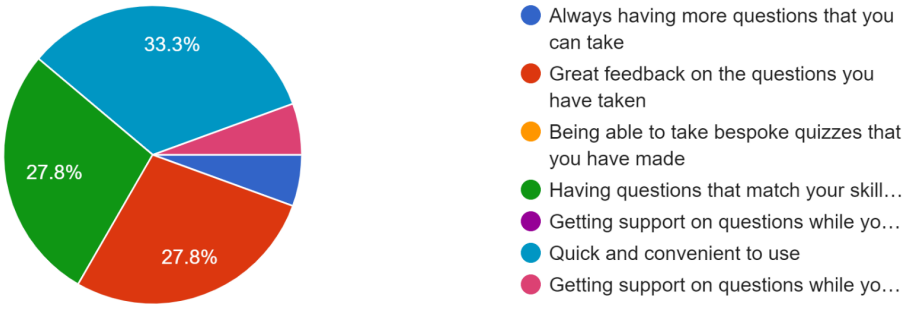

Figure 4. What is the most important feature of E-Assessments?

[4] Full details of pre-survey in appendix 1

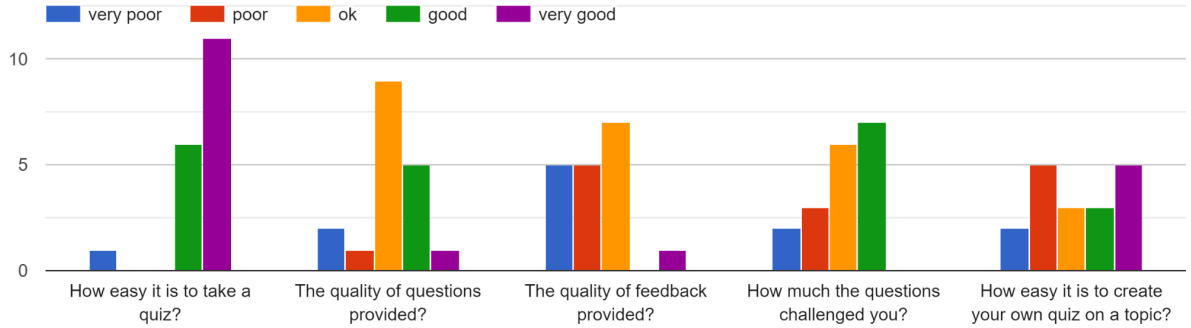

Figure 5. How would you rate the experience with e-formative assessment applications on these different criteria?

A final note from the survey is the belief that existing applications were more teacher-led than student-led (Figure 6), echoing the prior sentiment that existing applications are dependent on the teacher user and their expertise to create quizzes.

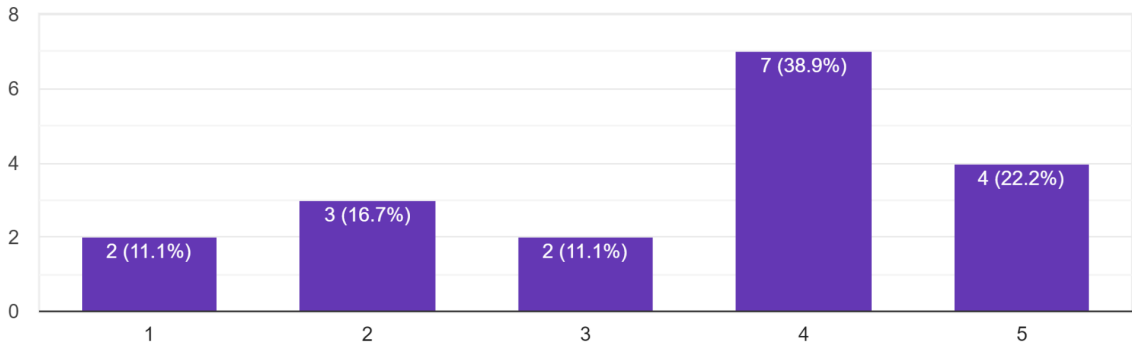

Figure 6. On a scale from 1 to 5, how much do you feel that these applications are more Teacher-led, made to help teachers make assessments and grade their students, (5) or Student-led, made to support students learning independently and revision (1)

Software Review

The number of existing e-assessment applications is vast, with most having a very similar set of features [19, 20]. The top 5 by usage from the survey are investigated.

Kahoot! & Quizizz

Kahoot! is potentially the most iconic formative e-assessment tool, being well known for its gamification whilst Quizziz has many of the same features (Figure 7).

These applications centre on a teacher or expert user creating a quiz that is played in a large group where players compete against each other for the best score [21, 22]. Quizzes can also be shared, allowing other individuals access. This successfully maintains a fun and engaging environment [23].

This central dependence on teacher users illustrates the limitations of the teacher-led design model. Without the participation of the teacher user in creating quizzes student users are unable to effectively use the learning applications.

One of Kahoot!'s key features is its use of Gamification: using game-based engagement techniques, such as attractive visuals and music or the use of competition, in non-traditional game settings. Competition is cited as one of the key drivers of Kahoot!'s high engagement and success [23]. Other less gamified motivation techniques, such as visualisations of clear improvement for the student, are

not used. Competition is also a core element of Quizziz although much less music and colour makes the application less overwhelming. This reduces extraneous cognitive load allowing for a greater capacity for student learning[5].

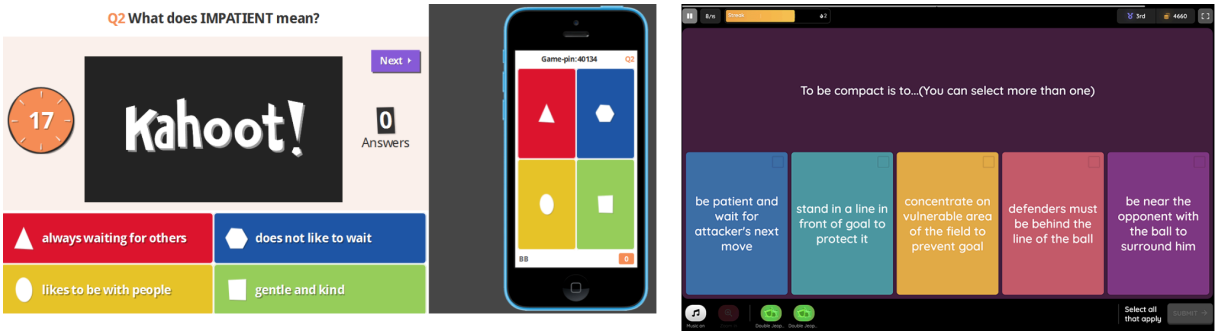

Figure 7. Kahoot! Quiz (left) and Quizizz Quiz (right)

Both Kahoot! and Quizizz implement Artificial Intelligence and LLMs in their application. For Kahoot!, this is only accessible via a paid subscription [24]. These "AI Assistants" allow a teacher user to input text or upload a document, and then a quiz is created based on this input. The teacher can then review the questions created before publishing the quiz [25]. This is limited to the teacher user as it is all done through the quiz creation tool, meaning that answers are visible. It is also clear that these applications intend an expert to review the created questions as their form of human hallucination detection (Figure 8).

Figure 8. An example of a hallucination on Quizizz AI quiz generation. Both A and D are the motto of the University of Birmingham, A is the motto and D is the motto in English. This would require a teacher-user to spot and change before a student-user took this quiz.

Neither Kahoot! nor Quizizz clearly explain how their AI integrations work. Doing so would help their user understand its capacity and limitations. This additional feature is important to ensure the explainability and appropriate use of intelligent interactive systems [26].

Canvas

Canvas is a multi-purpose learning management system (LMS) that provides a variety of features including the sharing of module content, documents and videos, the creation of submit discussions, integration with some other learning applications and, most importantly, the creation, assignment and submission of assessments [27].

Similarly, quizzes can be created by teachers and assigned to students. They can also form part of a module allowing the summation of assignment scores into a final grade. Quizzes can have assignment dates and due dates and can be set up to allow multiple attempts. There are many quiz types to

[5] See literature review for more information about cognitive load

choose from including multiple-choice and ‘essay’ questions. An important feature of Canvas is its ability for teachers to give adaptive feedback to users [28, 29].

Canvas quizzes are still subject to the design limitations created by the need for teacher users. No LLM features are used.

Quizlet

Quizlet is a studying and learning application. Features include document note-taking, revision cards and a variety of assessment applications, such as matching games and quizzes. In March 2023, Quizlet started to integrate AI into many of its features, as well as introducing "Q-Chat", a virtual AI tutor powered by OpenAI's ChatGPT [30].

The process of creation on Quizlet follows a different, more student-led approach. The user will input text or upload a document or notes and this will create a “Study Set” full of ChatGPT-generated “magic notes” summarising the material and revision cards. From this, ChatGPT is also used to create example essay questions and quizzes [31]. The quiz features are only accessible via a paid subscription [32][6].

The abundance of similar but distinct features makes Quizlet hard to both use and describe. For example, by using the flashcards you can access a “Learn” feature, which functions like a multiple-choice quiz of indeterminate length. This is somehow distinct from the “Test” feature which is a quiz of a set size that could be true/false, multiple-choice, written (short-answer) or “matching” (which itself is distinct from the flashcard “Match” game). It should be noted that all the quizzes are created using flashcards and very little additional content is created, other than distractor answers in the case of multi-choice. Likewise, long answer questions require students to input the exact answer, rather than marking them semantically, relying on the student to mark it correctly if the answer is semantically correct but syntactically different.

It is unclear to what extent hallucinations have been mitigated. The creation of study sets does not appear to have obvious problems, although details can be oversimplified in the “magic notes”. The "Q-Chat" is the most problematic element and struggles to maintain conversation. Q-Chat has four modes, including “Teach Me”, “Quiz Me”, “Apply my knowledge” and “Ask a question”. When using the “Quiz Me” mode the Q-Chat insists on ending every response with a question, meaning that it’s hard to explore questions after you have answered them or get feedback on your performance on the quiz.

Overall, Quizlet is one of the best student-led existing applications and makes extensive use of AI integration, although its abundance of similar features makes it confusing to use and hard to know what to focus on when using.

MyMaths

MyMaths, a mathematics learning website, also attempts to circumvent the need for teacher involvement. It provides a range of teaching applications as well as content-specific information. The content has been designed by maths teachers so there is no need for additional teacher involvement, although the application still does require the school to set up an account before students can use it. It is subscription-based with the school paying the cost.

The website contains many pre-made assessments for users to study, although no feature allows teachers to create new quizzes [33].

[6] Note: users do get one free quiz

Features	Kahoot!	Quizizz	Canvas	Quizlet	MyMaths	Proposed application
Quiz creation	✔	✔	✔	✔		✔
Quiz editing	✔	✔	✔			
Quiz taking	✔	✔	✔	✔	✔	✔
Multiple choice questions	✔	✔	✔	✔		✔
Short answer questions	✔	✔	✔	✔		✔
More question types	✔	✔	✔	✔	✔	
Gamification	✔					
Progress tracking				✔		✔
Adaptive feedback			✔			✔
LLM integration	✔	✔		✔		✔
Free use of LLM integration		✔		✔*		✔
Automated hallucination mitigation				✔*		✔
AI integration explained						✔
Can the student use the application without the need for a teacher?				✔	✔	✔

Figure 9. Table details the different features present in researched applications compared with the features present in the planned application.
** Some, but it does not cover all features*

Literature Review

Formative Assessment

Einig recognised the importance of e-assessment for students' learning, as well as the lack of empirical evidence supporting the use of multiple-choice questions (MCQs) [2]. During their study, undergraduate students were given regular formative assessments to support their learning. The paper found that feedback was widely positive. It also found that MCQ usage positively correlates to examination performance, although only when many questions had been attempted. Similar findings have been found by Black and Wiliam [3].

Generally, we can understand the benefits of Formative Assessment through the practice of retrieval. Ebbinghaus illustrated memory through a forgetting curve [4, 34]. Memory reduces over time, although if we *retrieve* the information from long-term memory multiple times following the initial learning period (often called “spaced repetition”) then we can improve retention of this information [35]. A key aspect of retrieval practice is repetition, explaining why the prior study by Einig only found a positive effect when many questions had been attempted by the student. E-assessment applications provide students with the resources they need to do an effective and challenging form of retrieval practice; resources they could not create on their own.

In addition to retrieval, the design of e-assessments can also be influenced by cognitive load theory. There are three types of cognitive load: *intrinsic cognitive load,* the effort associated with a specific

topic, *extraneous cognitive load*, the way information or tasks are presented to a learner, and *germane cognitive load* the work put into creating a permanent store of knowledge (a schema) [36]. By minimising extraneous cognitive load through simple and clear design, as well as the student curating the quiz's intrinsic cognitive load, we allow for the germane cognitive load to construct permanent understanding. Interestingly, this aligns with a desire from surveyed users for the application to be *quick and easy to use*.

Returning to the NFER's Assessment 101 [1], one of their three core principles for assessment is *Valid*: the assessment must validly relate to the desired topic of study, if it doesn't it will not have a positive effect on the student. This will be important to keep in mind when evaluating the quality of assessments produced and ensuring suitable use of hallucination detection.

Large Language Models in Formative Assessment

Many papers exist on Natural Language Processing (NLP) in educational technology [14, 37, 38]. Many go further to focus on specific uses in creating assessments [18, 39, 40, 41, 42, 43, 44]. For example, Dijkstra et. al. trained the fine-tuned model, EduQuiz, to generate multiple-choice questions based on text inputs using an educational quiz dataset[7] [18]. This was compared to other general-purpose models, Macaw-11b and GPT-3. The result shows that fluent and useful multiple-choice questions can be created using LLMs, although not perfect every time. These findings align with the findings of other papers.

Large Language Models for Generating Adaptive Feedback

Providing an expert solution or the expected answer to a question is static feedback [45]. Static feedback is the same regardless of how the student performs, easily showing students what the correct answer is. Adaptive feedback is guidance given to the student about how they specifically could improve, based on their recent performance in the quiz [45]. Adaptive feedback is much harder to create, requiring an understanding of the question, the correct answer and why the student may have got it wrong. LLMs could be used to generate this adaptive feedback.

This use of automated adaptive feedback has been investigated [45, 46, 47] finding positive results regarding the capacity of NLP technology to generate adaptive feedback. However, these studies don't investigate the use of MTL-LLMs for this particular problem. One simplistic way in which this technology can be used will be investigated and its effectiveness evaluated in this paper.

Hallucination Mitigation

Hallucination mitigation attempts to address the source of hallucination and reduce its likelihood of affecting output. It is a vital consideration for this project. In a student-led application, all information given to the student must be accurate to prevent teaching incorrect facts. In the words of the NFER, outputs must be *Valid*. The two main ways for mitigating hallucinations undertaken in the project are prompt engineering and hallucination detection[8].

Prompt engineering is the field of research concerned with finding the optimal prompt to encourage precise and useful responses from LLMs [49]. Many prompt engineering techniques exist. One of these methods is *In-Context Learning (ICL)* [50] where an example of the desired output is given in the prompt. This has shown significant improvements in the quality of answers given by LLMs. Another example of prompt engineering would be the use of Role, Instructions and Seed-Word [51]. Simply put, this is the process of clearly detailing what you want LLM to do (Instructions), particular things for the model to focus on (Seed-words) and how the model should respond (Role)

[7] https://github.com/jemmryx/EQG-RACE

[8] Other approaches of hallucination detection use formal verification to detect logical inconsistency in the output and give feedback to the LLM [48]. This was deemed out of scope for this project.

Once the prompt has been sent and the output generated, the next thing we can do is attempt to catch hallucinated outputs. This could be carried out by a human expert, such as the method used by Kahoot!, although this is labour-intensive and lacks the benefit of student-led design.

An interesting way of solving this problem is bypassing the output back into the LLM, but this time asking it to check for hallucinations. Chen Qian *et. al.* found this approach effective when they used a "team" of "software agents" (LLMs) to form a "virtual chat-powered software technology company – ChatDev" [52]. Here, code written by software agents was reviewed by other software agents and feedback was given on how to improve the solution. "On average, ChatDev generated 17.04 files per software, alleviated potential code vulnerabilities caused by code hallucinations 13.23 times, had a software production time of 409.84 seconds, and incurred a manufacturing cost of $0.2967."

A similar system is used by "SelfCheckGPT" [53]. Here, when an output is being checked, several other stochastically generated outputs are created and then each is compared with the originally generated sentence. The theory is that, if a program is hallucinating then there should be high variation in output. If a large number of stochastically generated outputs do not conform with the original output then this output is deemed a hallucination.

Project Management and Requirements

Methodology and Project Management

The methodology used during the development of this application is the Behaviour-Driven-Development (BDD) Methodology [54], a form of agile development focused on using a Red-Green-Refacor cycle (Figure 10).

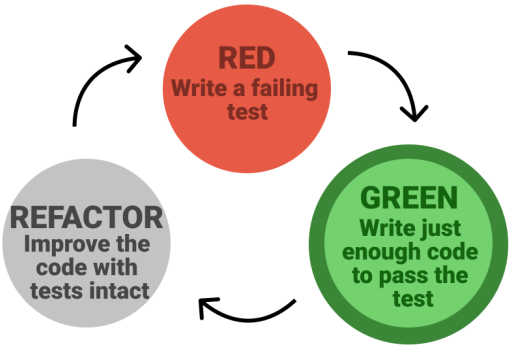

Figure 10. Behaviour-Driven-Development's Red-Green-Refactor Cycle

First, the most important behaviour that is not yet present in the application is considered. A test for the behaviour is written which will, of course, fail. Then the smallest amount of code will be written which will satisfy the test, followed by, finally, refactoring the code. Refactoring includes improving the readability or efficiency of code, as well as catching bugs in old code that new code may have changed. Once one cycle has finished, the next one starts. Each development cycle is called a sprint.

This methodology leads to a quick, focused and effective development cycle that constantly considers what is needed and achieves this as quickly as possible. Red-Green ensures the aim is clear when writing code, while Refactoring ensures the good upkeep of well-written code.

This project is well suited for this methodology due to its small scale and focus on the exploration of novel features, leading it towards a behaviour-centred methodology. Additionally, any lack of prior experience with web development and LLM is mitigated by an iterative methodology that tackles behaviours one step at a time, learning these skills on moving through the project.

Requirements

Functional Requirements

Despite the user's wishes for many different things, it is key to focus on the most important first. This involves considering centrality, impact and cost of development. To handle this, the MoSCoW method is utilised to prioritise the most important features and focus on them in earlier sprints. This uses user stories[9] to create several requirements and then ranks requirements as *must-have*, *should-have*, *could-have* or *won't-have*. The distinction between *should-have* and *could-have* is subjective; however, this is a problem inherent to the MoSCoW method [55]. Decisions are made by considering the estimated cost of development against the impact of implementation on the user. The relative significance of interest that people gave towards features during the pre-survey is also taken into account.

[9] This application has only one user to consider, the student-user (see Figure 2)

The **Student-User,**

1. **(must)** ...be able to create and log in with a personal account.
    - Users must be able to create an account through the use of an email and a password.
    - Passwords must never be shown or stored in plaintext.
    - Passwords must be double-checked before the account is created by asking the users to enter them twice.
    - Passwords must be restricted by security requirements (minimum length, inclusion of number, inclusion of special character).
    - Users must be able to log in using their email and password.
    - All webpages must reroute to the login page (except the create account page) when the user is not logged in.

2. **(must)** …be able to create their own **multiple-choice** quizzes on whatever topic they want.
    - Users must be able to create quizzes, with questions that are made up of a question, an answer and three distractors (semantically wrong but syntactically similar answers).
    - Users must be able to input a title and a description to influence the content the questions are about.
    - Users must be able to specify the number of questions the quiz will contain.

3. **(must)** …be able to take their quizzes and view their results.
    - Users must have access only to quizzes that they have created.
    - Users must be able to take the quiz through a dedicated page.
    - Users must be given questions one at a time.
    - Users must have the option to return to questions after they have answered them.
    - Users must not be able to see what questions they have got right or wrong while doing the quiz.
    - Users must be able to leave the quiz at any time, return to the dashboard, and be informed that they will lose their progress if doing so.
    - Upon quiz completion, users must be taken to a dedicated page to view the results of the quiz.

4. **(should)** …be able to create their own **short-answer** quizzes on whatever topic they want.
    - All requirements for multiple-choice quizzes should also be satisfied by short answer quizzes (without the addition of distractor answers).
    - Users' answers should be evaluated on semantic rather than syntactic bases, not needing their answers to identically match the expected answers.

5. **(should)** …get adaptive feedback on questions they have answered wrong.
    - After completing the quiz, users should be provided with feedback on questions they got wrong.
    - This feedback should be informative, explaining what the correct answer is and what the given answers relate to (if anything).
    - This feedback should be instructive, explaining how the student can improve in the future.
    - This feedback should be positive, with a constructive tone with no harmful or demoralising comments used.

6. **(should)** …be able to track their progress over time for a given quiz.
    - Users should be able to see a record of past attempts for each quiz.
    - Users should be able to see what answers they gave on those attempts.
    - Users should be able to see a visualisation of their progress over time.

7. **(could)** …be able to select from a list of predefined topics (e.g. History, Geography, Physics) when creating a quiz.
    - Questions could be sorted by topic.
    - The topic could influence the content of the questions created.
    - The list could include the option ‘other’, leading to no topic being picked, and do not affect the question created.

8. **(could)** …be able to take a quiz with an endless amount of questions, generating more as it goes along.
    - When creating a quiz, users could select the option for an endless quiz.
    - Endless quizzes could make new questions as the user takes the quiz indefinitely.
    - Users could be able to end the quiz on a question of their choosing, at which point the quiz will be saved at that length.

9. **(could)** …be able to learn and get guidance on how best to use the application.
    - An information page could exist that is accessible to the user.
    - Information could detail how the application works.
    - Information could detail how to structure the title and description.
    - The user could use a form to send an email to a page editor if they are unsure about the information details on the page.

Non-functional Requirements

The following are the non-functional requirements that will guide design and development.

Usability

- **The application should be convenient and intuitive**. This is something that is both important to users and aligns with educational cognitive load literature.
    - All elements should have a consistent style.
    - The application will follow a minimalist design style, limiting the number of features on a page to only the most essential.
    - 90% of users should not feel confused while using the application.
    - The application should inform the user when an error has occurred, informing the users what has happened and why (if that would be useful information).
- **The application should be quick**. Something also highly desired by potential users.
    - Login verification (success and failure) should take less than 2 seconds.
    - All pages should load within 5 seconds.
    - The generation of questions should take less than 30 seconds per question.
    - Verification of answers should take less than 1 minute. Generation of feedback should take less than 2 minutes.
- **Explainability**
    - Users should be able to explain, in simple terms, how the application creates questions using the LLM integration

Accuracy

- Information provided by the application must be correct at least 90% of the time. This will be evaluated by having field experts review the generated questions to determine their accuracy.

Accessibility.

- The application should be freely accessible. This means that no architectural elements should run at a cost to the user, and free tools should be used where possible.

Security.

- Passwords should never be stored or represented in plain text.

- All inputs should be verified before being used, especially when these inputs are stored in memory or used in LLM prompts to mitigate malicious inputs.
- Only necessary data should be stored in the application. Email is taken so the user can be contacted if we need to. Users are advised not to include personal data in any other input.
- Pages should only be accessible through a HTTPS connection, with any HTTP connections being routed to HTTPS.
- The systems should not display any unnecessary system data to the user, as this may be used by a malicious party.

Compatibility.

- The application should be compatible with most modern web browsers. This includes both desktop/laptop and mobile browsers. The application will be tested against Edge, Safari, Chrome, IE, and Firefox on Windows and macOS, as well as browsers on Android and iOS.

Maintainability.

- Unit tests should be written for each behaviour and integration testing should be carried out following the stage in the development.
- Code should be written using consistent formatting, and should be well documented.
- Errors should be reported to a developer and recorded if not fixed in a known bugs list.
- The application should follow a modular design, such that if any one feature needs to change it has minimal effect on all other features. This also holds that if an API used needed to change only the direct requests would need to be re-written.

Design and Implementation

The product is a web application allowing it to be accessible from anywhere with an internet connection with no additional setup. In addition, this approach allows the quizzes to be saved on the server and shared wherever the user accesses them.

The application is made up of a front-end user interface and a back-end that hosts the website and stores all the necessary files and data. An application programming interface (API) is also integrated into the architecture to utilise LLM functionality on the front end (Figure 11).

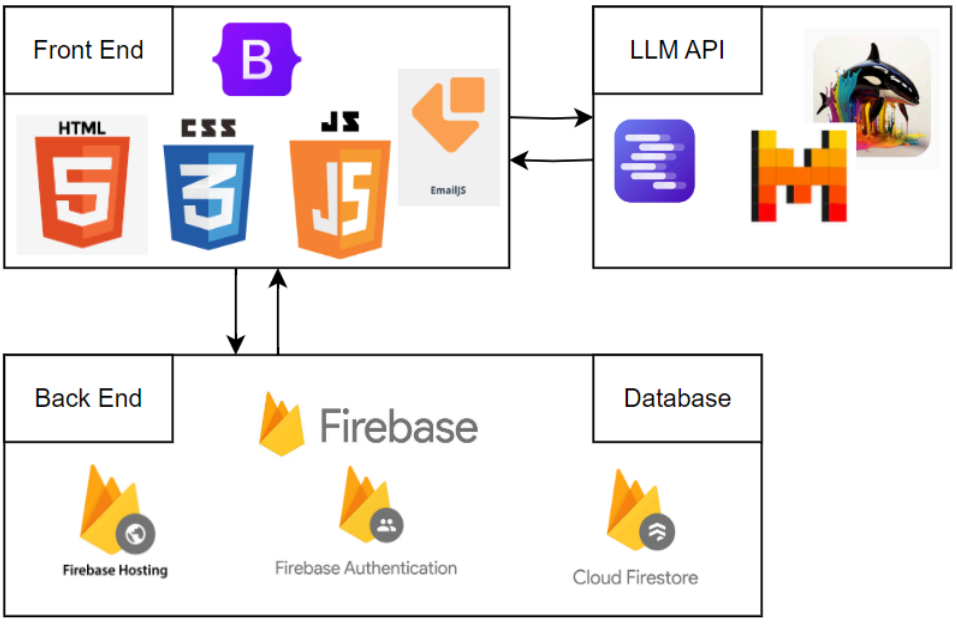

Figure 11. The Stack

The code is developed following a modular design method allowing parts of functionality to be abstracted from each other. This facilitates a well-organised body of code that is robust to changes. This design methodology aligns with the non-functional requirement of maintainability.

In addition to the modular design, effort is made to ensure that the code has consistent formatting and is as readable as possible. Consistent naming conventions are used throughout the code: variables use snake_case, functions use camelCase (starting with a verb), and files use kabab-case. Abbreviations are avoided. This makes it clear to know what each keyword is and how it is used. Comments are made throughout the code to increase readability, as well as using JSDoc [56] to annotate functions. Code is written using Visual Studio Code's IDE [57].

Frontend

The development of this client-server web application uses the HTML/CSS/JavaScript ecosystem, the industry standard for the development of web applications. HTML/CSS are excellent for designing the front-end client-side user interface, while JavaScript can handle the logic on both the client side and on the server. This also provides access to many other tools that are designed to integrate with this ecosystem, such as Bootstrap 5 [58]. Using Bootstrap 5, a CSS design library, speeds up development and assent in creating a uniform design, aligning with the non-functional requirements. To allow users to engage with the application further and ask questions about how it works, EmailJS [59] has also been integrated into the front end to allow the automatic sending of emails. This is used in the information page, allowing users to ask questions about the application giving them an understanding beyond a basic description.

Backend

Google's Firebase *Backend-as-a-service (BaaS)* [60] is used to handle all the necessary back-end functionality providing web hosting, user authentication and file storage, and interacting with the front-end via JavaScript on the client through an API. Users and quizzes are stored in separate databases managed via Authentication and Firestore respectively. Firestore stores data in a JSON format. All of a user's quizzes are stored in Firestore under that user's ID.

LLM API

Firebase does not satisfy the project requirements when it comes to LLM integration so an additional API, LM-Studio [61], an application that facilitates the local downloading and running of off-the-shelf open-source LLMs, is used. LM-Studio hosts a Local Inference Server that is requested using a JQuery [62] Ajax request. This same architecture could be used if the LLM hosting was moved to a centralised server rather than being hosted locally.

There is a wide range of models accessible via LM-Studio. Models are gathered from HuggingFace [63], an open-source model-sharing platform. An informal analysis of different models on the platform considered a range of the core MTL-LLMs models on the platform (Meta's LLaMa 2 [64] and Mistral's Mistral AI [65]) as well as several different versions of these models. Some models were too large to be investigated due to technical limitations. It was found that the clearest and most consistent responses were from *'TheBloke / Mistral-7B-OpenOrca-GGUF / mistral-7b-openorca.Q5_K_M.gguf'.* This is a quantised version of Mistral's 7-billion parameter model, trained on the OpenOrca dataset [66] and stored in GGUF format.

User Interface (UI)

The design of the UI is consistent and minimalist, supported by both users (quick and convenient to use) and theory (reducing extraneous cognitive load). This aligns with the use of Bootstrap.

Figure 12 shows a state diagram that illustrates the movement between different pages and the design of those pages. The application uses alerts where possible to indicate to the user when an action has occurred or needs to be made that wasn't part of the user interface. Examples include the confirmation of significant actions, like deleting a quiz.

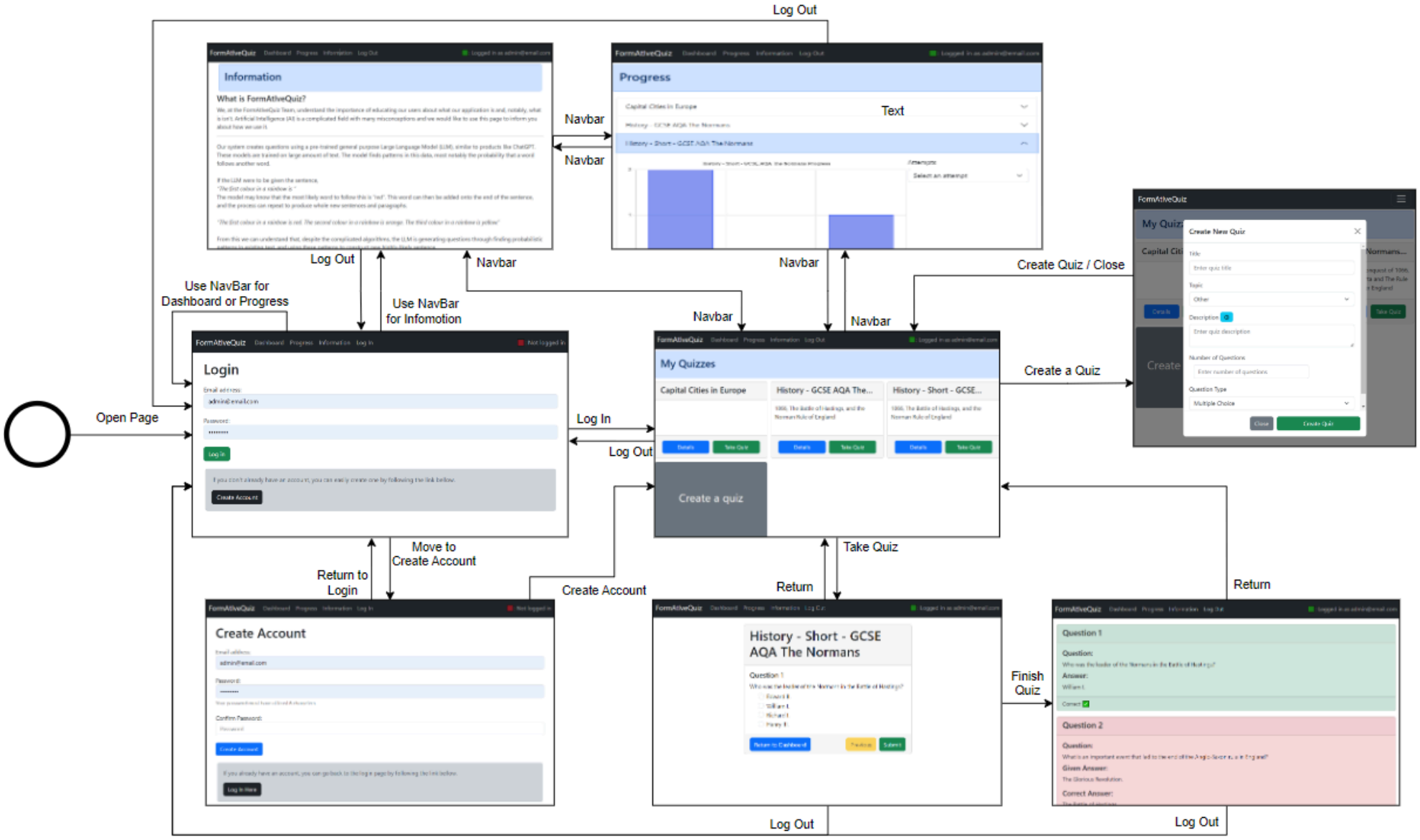

Figure 12. A state diagram for the application

Accessibility considerations influenced the design of the application. The elements are large and the font is chosen to be easy to read. The use of symbols without text is avoided to minimise confusion, and text instructions are short to reduce the need for reading where possible. Colour, although used through the app at a prompt (e.g. a delete button being red), is never needed to understand how an aspect of the application works in case colour cannot be seen for any reason or the colours are interpreted differently.

Quiz Generation

Quizzes are stored as objects, incorporating an Object Oriented Programming (OOP) approach (Figure 13). OOP allows multiple instances of quizzes to exist with different data, but all having the same structure, defined by the Quiz Class. Each Quiz object contains the attributes defined by the user (title, topic, description, number of questions to be generated, and the type of quiz, *'multi-choice'* or *'short-answer'*) as well as some others initialised on creation and edited later (*id*, containing a unique identifier for storage in memory, *questions*, an array of Question objects created through question generation process, and *attempts*, an array of Attempt objects added to each time the user completes the quiz). Question objects contain two strings, a question and an answer, and, in the case of multiple choice, a list of options. The Quiz object also contains methods utilised during the running of the quiz. Attempt objects, created when a user completes a quiz, contain an index, the date the attempt was made, how long this attempt took, a score, and an array of the given answers (Answer Objects)

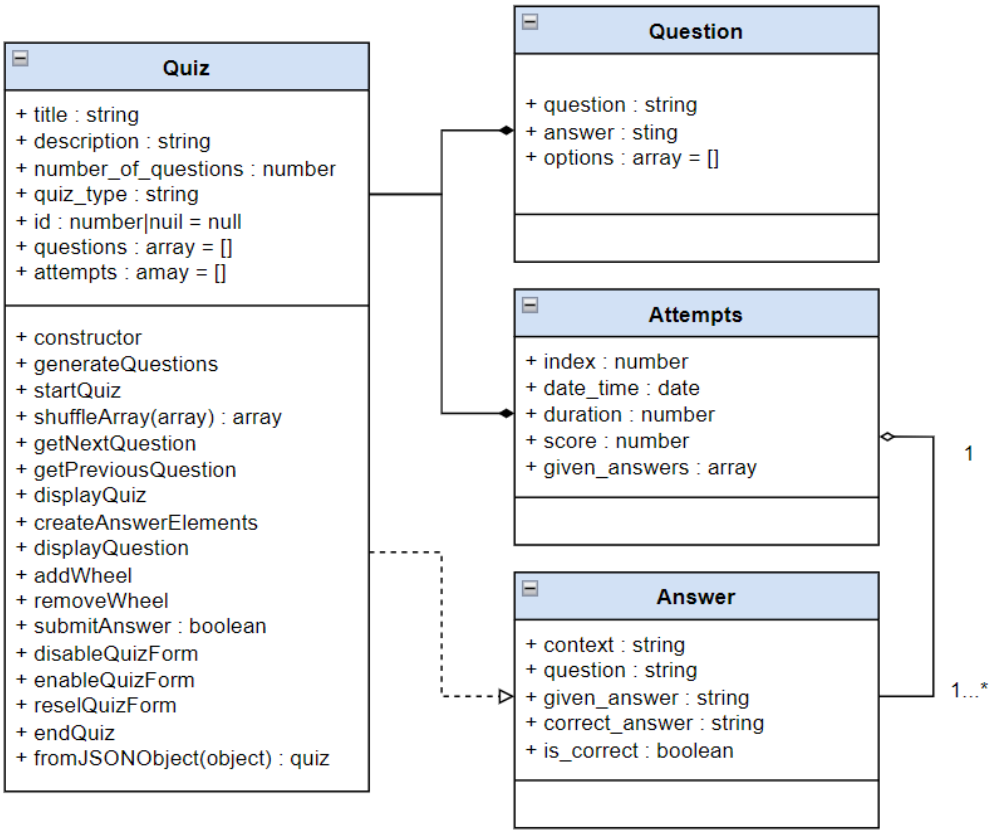

Figure 13. Class Diagram for the quiz-related features

Questions are stored after being created using Firebase Firestore using JSON under the user's ID.

When a user logs on, they are taken to a dashboard. A query is sent to Firebase Firestore to fetch all quizzes under that user's ID, and then cards for them are created and added to the dashboard. The page lists all their quizzes as cards, and one extra card is used to create new quizzes. The function fromJSONObject(object) allows quiz objects to be constructed from the JSON objects stored in memory.

For generating the quizzes, the user inputs at title (e.g. '*GCSE History AQA Normans*') and descriptions (e.g. '*The Norman Rule of England Following the Battle of Hastings*'). These are then used to set the "context" of the generated quizzes.

Following the work of Dijkstra *et al*, the method initially used in question generation is Step-Wise-Question-Generation (SWQG). Instead of generating a whole quiz or whole question in one request, multiple individual requests are made that each build on the last; step 1 generates the question, step 2 generates the answer, and step 3 generates distractor answers in the case of a multiple choice quiz. Each step would have a function with its prompt and handling. This had the advantage of making the code componentized, allowing for efficient refactoring and testing, as well as clear use of each component.

This method, however, creates the Repeated Question Problem. Due to each question being generated independently through the same prompt, the model tended to generate similar or identical questions each time (Figure 14). One attempted solution to this problem was explicitly telling the model not to generate questions that are similar to any existing question although this did little to mitigate the issue.

Context	History - GCSE AQA Normans - The Battle of Hastings
Question 1	"When was The Battle of Hastings?"
Question 2	"What year was The Battle of Hastings?"
Question 3	"When was The Battle of Hastings?"

Figure 14. The Repeated Question Problem

The effective solution to The Repeated Question Problem is the development of a new question generation framework, Batch-SWQG. Batch-SWQG works by generating all the quiz questions at once in a batch. This batch of N questions is then broken up. The answers to each question are generated independently for each question. Finally, multiple-choice distractors are generated, these are also generated in a batch to ensure that they are unique. Duplicate and correct answers are also manually removed from the list of options. Figure 15 shows a flowchart detailing Batch-SWQG.

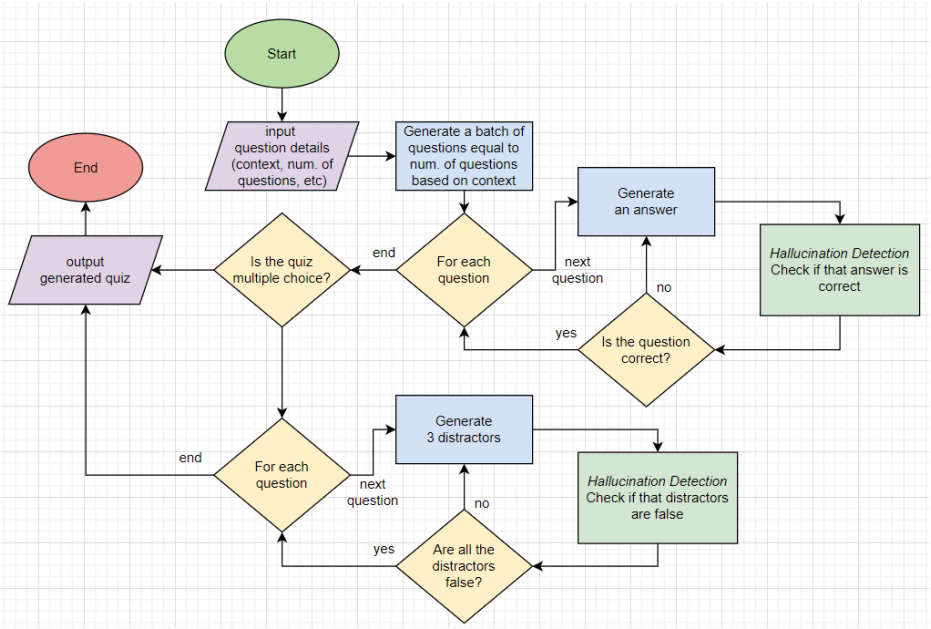

Figure 15. shows a flowchart detailing Batch-SWQG

One drawback of using this methodology is that, once a batch of questions has been created, more questions can't be added to the quiz. Attempting this would run into the Repeated Question Problem as the new batch would clash with the original batch. This means that the requirement 'be able to take a quiz with an endless amount of questions, generating more as it goes along' cannot be developed in the current system. Considering that this is not a high-priority requirement, it is omitted from the final product and will not be developed in future sprints.

Given all questions need to be created before any answers are created, a limit is put on the number of questions that could be created. This is set to 30, the length of quiz primarily desired by expected users in the pre-survey after unlimited quiz length[10].

It's possible to create quizzes with just a title (distractors and topics are optional) due to the use of input handling. The user inputs a simple title like "Computers" which becomes the context of the quiz. This context is then inserted into a prompt which is passed to the LLM to create questions. A similar process is used for creating answers and destructors. This process utilised the prompt engineering method of User/Instruction/Keyword, as Context, Questions and Answers are all used as keywords within the prompt, and instructions are listed clearly. The prompts used are shown in Figure 16.

```
        let system_content = `Given the context, generate exactly ${number_of_distractors +
2} FALSE distractor answers to the following question. The true answer is "${answer}". Each
distractor must be different from the true answer, as well as from each other. Distractors
should look similar to the answer. Do not state in any way that the answer is false, or
that it is a distractor. `;

        system_content += `Format the distractors as a list, separated by bars "|". For
example, if the question was "What is the capital of France?", a valid response might be
"|Lyon|Berlin|London|Madrid|". Do not number the distractors. `;

        let user_content = `Context: ${context}. Question: ${question}. `;
```

Figure 16. Formatted code snippet from src\text-generation\quiz-generation\3-distractor-generation.js. The system prompt uses clear instructions and keywords for effective prompting. These keywords are then used in the user prompt.

When creating questions or distractors batches they must be listed with a bar character, |, between them and separated into a list. Figure 17 also shows the use of in-context learning for this process. An example of how the output should be formatted is given, allowing the LLM to copy this format. If questions are not given in this format then the request is resent and tried again. Output handling also consists of standardising the punctuation and, in the case of distractors, checking to see if any of them are the answers and removing them, also seen in Figure 17.

```
let response = await callLMStudio(system_content, user_content, 1000);

// Split the response into an array of potential distractors
let potential_distractors = response.split('|');
// Convert to string
potential_distractors = potential_distractors.map(distractor => String(distractor))
// Remove leading and trailing punctuation
potential_distractors = potential_distractors.map(distractor => distractor.trim())
```

[10] See appendix 1

```
// Remove strings that don't contain letters, numbers or mathematical symbols (including
empty strings)
potential_distractors = potential_distractors.filter(distractor =>
/[a-zA-Z0-9+\-*/^()]/.test(distractor))
// Remove leading and trailing punctuation
potential_distractors = potential_distractors.map(distractor =>
distractor.replace(/^[.,?!]+|[.,?!]+$/g, ''));
// Add a period to the end of each answer if it contains a text character
potential_distractors = potential_distractors.map(distractor => /[a-zA-Z]/.test(distractor)
? distractor + "." : distractor);
// Remove the true answer from the list of potential distractors
potential_distractors = potential_distractors.filter(distractor => distractor !== answer);
// Remove any distractors that have already been added to the list
potential_distractors = potential_distractors.filter(distractor =>
!distractors.includes(distractor));

if (potential_distractors.length > number_of_distractors) {
    potential_distractors = potential_distractors.slice(0, number_of_distractors);
}

distractors = potential_distractors;
```

Figure 17. Formatted code snippet from src\text-generation\quiz-generation\3-distractor-generation.js. The responses are thoroughly handled to ensure uniform formatting and check for erroneous outputs.

The last thing that happens before a response is saved is hallucination detection to ensure that answers are true and distractors false. After an answer or distractor is generated, it is checked to see if it is true or false respectively, by asking another instance of the LLM, similar to ChatDev. This LLM instance has a temperature[11] of zero to reduce the likelihood of it hallucinating [67]. If the answer is deemed false (or distractor true) then it is regenerated and rechecked. This process is iterative and is attempted 5 times. If it fails on the 5th attempt the current model will then accept the most recent given answer and use it, although the proportion of these instances is highly dependent on the context of the question and what information the model has been trained on. A future development of this application could reject this whole question and generate a new one, although due to The Repeated Question Problem and the desire to keep questions at the length specified by the user, this is not implemented.

This process of hallucination detection can be understood through three possible outcomes:

1. The LLM “knows” that answer and gives the right answer, at which point the answer is deemed correct and used.
2. The LLM “knows” that answer and hallucinates, giving a wrong answer, at which point the answer is rejected and another one is created
3. The LLM doesn’t “know” the answer and guesses at an answer, at which point the answer is rejected and another one is created

If the LLM “knows” the answer it can be assumed that it will generate the correct answer more than it hallucinates, so getting it to regenerate the answer is likely to result in a correct answer being given instead of the wrong one it initially went for. If the LLM doesn’t “know” the correct answer then all five attempts will fail. This process of repeated generation and comparison is similar to that of SelfCheckGPT.

Hallucination detection slows down the creation of questions. During development, there was a trade-off between time and accuracy. The focus was put on implementing the hallucination detection features, resulting in the creation of a question taking, on average, 5 minutes 34 seconds. This time is

[11] Temperature is the level of variation in the outputs. High temperatures create very random outputs. A temperature of zero would mean that, given the same input, every output would be the same.

primarily hampered by the hardware used to run the LLM, i.e. my local machine. The same process running on a higher-performance CPU would take less time.

Marking of questions using LLMs

Quizzes either use a multiple-choice or short-answer format. Multiple-choice questions are marked automatically when the user selects from a list of four radio buttons. Short-answer questions could be marked automatically by matching the input exactly with the expected answer, although it is unlikely that the user would do that without first knowing the exact wording of the expected answer. Instead, these questions are marked using the LLM API.

When a student answers the question, an answer object is created containing the context of the question, the question, the expected answer and the given answer. This is then passed to a prompt handler that creates a prompt for the LLM. The LLM is asked to mark the question giving either a YES or a NO. Similar to hallucination detection, this is done with a temperature of zero to reduce hallucinations. If the output doesn't contain the string YES or NO, the answer is considered to be false.

Adaptive feedback using LLMs

After the student has taken the test, they can see all their results and feedback for the questions. All answers receive static feedback in the form of a model answer (Figure 19). Generated adaptive feedback is also given if the student gets the question wrong (Figure 20). This reduces the time taken to generate feedback. Feedback is generated individually starting from the first question they got wrong. Generating feedback takes an answer object, using all of its elements.

The prompt used in generating feedback is another example of instruction/key-work/role prompt engineering as shown in Figure 18.

```
export async function generateFeedback(answer_object) {
    let system_content = `You are a supportive and constructive teacher.
        Given the following context, question, expected answer and given answer,
        please give the student some advice on what was wrong in their given answer and
some ways they could improve.`;

    let user_content = `Context: ${answer_object.context}. `;
    user_content += `Question: ${answer_object.question}. `;
    user_content += `Expected Answer: ${answer_object.correct_answer}. `;
    user_content += `Given Answer: ${answer_object.given_answer}. `;
    user_content += `Is Correct: ${answer_object.is_correct}. `;

    try {
        let response = await callLMStudio(system_content, user_content);
        return response;
    }

    catch (error) {
        console.error(`generateFeedback: error:`, error);
        throw error;
    }
}
```

Figure 18. Feedback generation utilising instruction/key work/role prompt engineering. ***Role:*** *"You are a supportive and constructive teacher."* ***Keywords:*** *"Context", "Question", "Expected Answer", "Given Answer."* ***Instruction:*** *"Give the student some advice on what was wrong in their given answer, and some ways they could improve."*

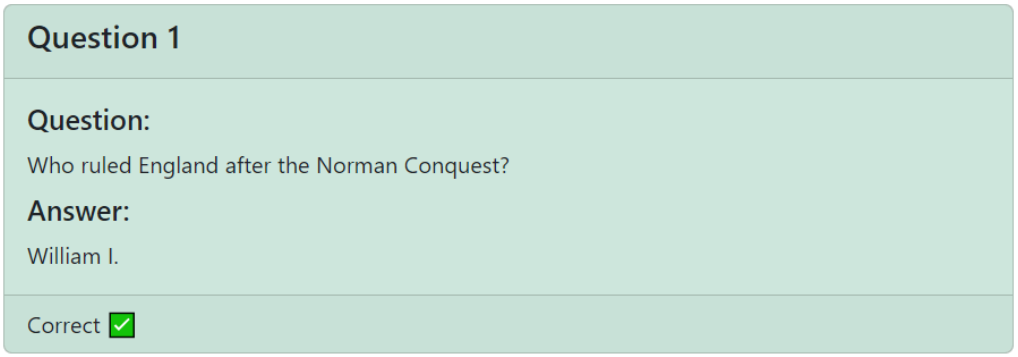

Figure 19. Example of feedback for correct answer

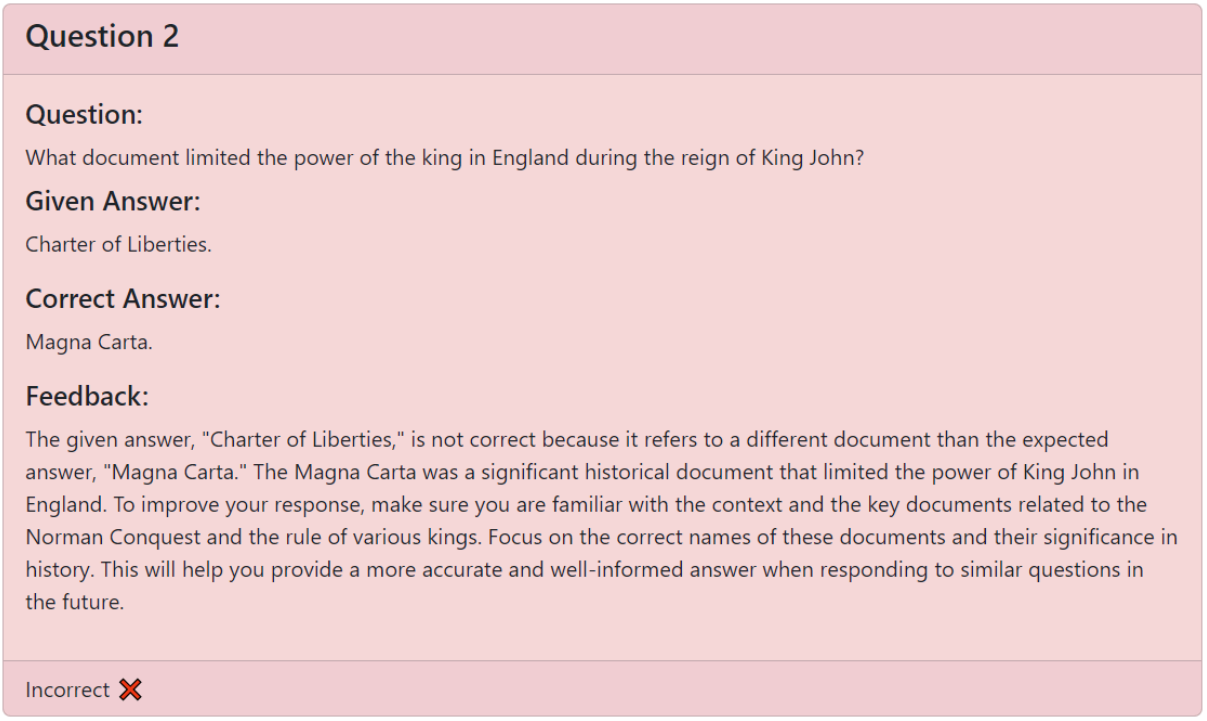

Figure 20. Example of feedback for incorrect answer

Progress tracking

After a user has finished a quiz, an Attempt object is created and added to the Quiz object. The attempt object consists of an index, the current date and time, the “duration” (the time it took for the student to complete the quiz), their “score” (how many questions they answered correctly) and a list of their given answers.

These attempt objects are used in the progress page to display a user's progress in the quizzes they have attempted. If a quiz has been attempted, a drop-down will show how the user has performed in past attempts. These attempts can then be further investigated to find the date and duration, as well as what the student put for each answer. It doesn’t show the correct answers although it does show whether the question was right or wrong.

The visualisation of a student's progress helps the student to understand the effect of spaced repetition and motivates the student for further work (Figures 21 and 22).

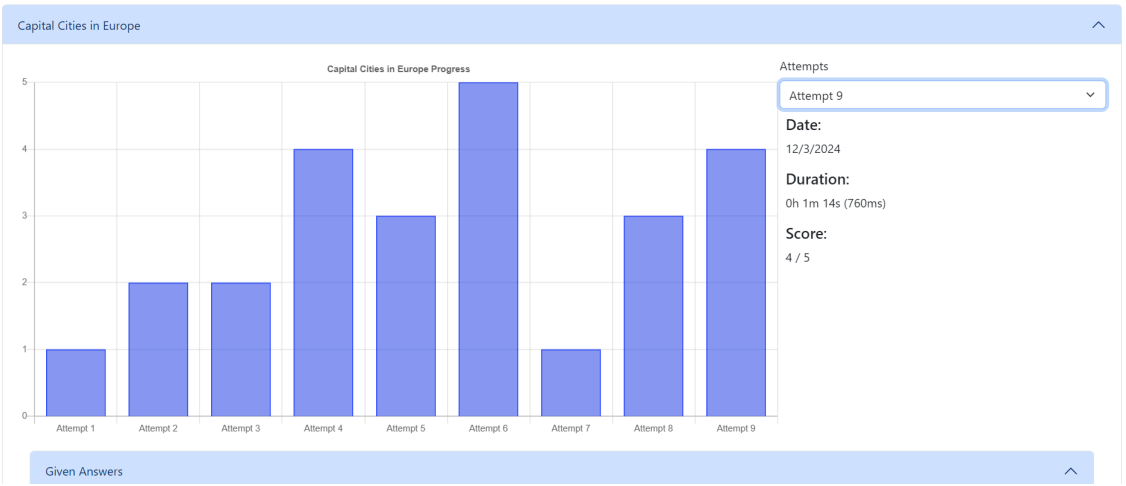

Figure 21. Visualisation of students' progress over time

Question	Answer Given	Result
1. What is the capital city of Germany?	Madrid	✗
2. What is the capital city of Netherlands?	Amsterdam	✓
3. What is the capital city of Poland?	Warsaw	✓
4. What is the capital city of Norway?	Oslo	✓
5. What is the capital city of Switzerland?	Bern	✓

Figure 22. Details of questions and given answers for past attempts

LLM Information

A page details the nature of the LLM's integration with the system to help the user understand how the application works to better use it and understand its capacities and limitations. The majority of this page is just text. This text has gone through several iterations through user analysis. An initial draft was written and reviewed by potential users who had a range of background knowledge surrounding AI. Feedback was given and then acted upon to re-write further drafts.

An additional feature of this page is the ability for users to ask questions through an email submission box. If a user wants to ask a question about the application then they can do so by filling out the form at the bottom of the page.

This form uses the email.js library which can be set up through an EmailJS account. This sends an email containing their feedback to SXB1567@student.bham.ac.uk from the email given.

By allowing the user to input an email the user can put any preferred email. In addition, we can allow this page meaning that users can access it without needing to have an account.

Results and Evaluation

Testing and evaluation determine how successful the development process has been in addressing the aims of the project. Testing is carried out to assess the functional and non-functional requirements of the application. A more thorough evaluation was also carried out for certain specific features to gain a better understanding of how they perform.

Testing

BDD involves testing iteratively throughout development. Unit tests[12] are written at the start of each sprint, the sprint ending when all these tests pass. Previous unit tests are carried out in a process of integration testing in the refactoring phase, ensuring that no existing behaviour has been damaged by recent development. During this, tests may also be updated if a sprint has removed or altered old behaviour.

Unit tests focused development on getting behaviours implemented. Time is saved by working on one feature and not getting distracted by non-essential elements. This focus still allows for space to be left for future development, ensuring that code is written in a clear modular manner, and comments made where future features could be added to guide future development. They also ensure that behaviour is never lost, integration tests considering the behaviour and functionality of all elements of the system.

On compilation of development, potential users carried out end-to-end testing, being asked to carry out a particular use case, as well as exploratory testing, using the application with no guidance. This test is also used as part of the whole system evaluation, discussed in a later section. The results of this testing catch elements missed by the unit tests. An example of this is unflagged errors. If the system had an issue, like the LLM not correcting, then errors were logged but not relayed clearly to the users. Following this, error alerts were an important behaviour added to development and a sprint was carried out to implement this behaviour.

Some systems require more than unit testing. This enables a more qualitative view of key features, based on judgement from potential users and experts. All of this evaluation is important when considering acceptance testing (done in the final part of this section) where the product is compared against the requirements.

Evaluation of Questions Generation

As stated by Dijkstra *et al*, human evaluation is the “gold standard” for evaluating LLM outputs of this kind and was the main source of evaluation in researching the vast majority of existing literature. Question generation is evaluated through Dijkstra’s methodology [18], with similar methods and ranking criteria also used in other papers [68].

The evaluation of question generation has two aims, to evaluate the use of LLMs in generating questions[13], and to evaluate the effectiveness of hallucination detection. To test these things, first, the test is done with hallucination detection disabled, then carried out a second time with hallucination detection enabled. The results of the two tests are considered independently (Figures 24 and 25) and compared (Figure 26).

[12] See appendix 3 for the full list of unit tests

[13] Note: no evaluation as done without the use of prompt engineering. In-context learning became an essential part of the input/output handling to ensure the output was formatted correctly. Almost no questions would be acceptable without prompt engineering enabled.

One hundred multiple-choice questions were generated[14] on ten different topics: Computer Science, English literature, Physics, Philosophy, Art, History, Mathematics and French (in French), as well as general knowledge topics Seventies Music and The Premier League. Topics were chosen based on the expertise of the human evaluators. Some topics also had a description added when generating questions. These descriptions varied in length, with some topics having no description and others having descriptions that were paragraphs long.

The one hundred questions were shuffled and divided into groups of twenty. Evaluators, including potential users and teaching experts, were assigned a group and asked to evaluate each part of the question against six different criteria, using a binary classification (Figure 23). The criteria, taken from Dijkstra *et al,* also align with NFER's Assessment 101 principle of Validity.

Sub-part	*Metric*	*Description*
Question	**Fluency**	Whether the question is grammatically correct and fluent
	Relevancy	Whether the question is semantically relevant to the context (title/description).
	Answerability	Whether it is possible to give a correct answer to the question
Answer	**Fluency**	Whether the answer is grammatically correct and fluent
	Correctness	Whether the answer contains correct information
	Validity	Whether the answer is a correct answer to the question
Distractors	**Fluency**	Whether the distractors are grammatically correct and fluent
	Coherence	Whether the distractors are relevant to the context and the question
	Distracting Ability	Whether the distractors can mislead the learner and if there is no true information in the distractors

Figure 23. The table shows the criteria used to evaluate the questions generated. This was the same criteria used by Dijkstra et al. The aims of each criterion were explained in detail to each evaluator before the start of the evaluation.

Figure 24 shows the results of the evaluation without hallucination detection, and Figure 25 the results of the evaluation with hallucination detection. Figure 26 then shows a more direct comparison of the most important fields when thinking about hallucination detection.

The evaluation shows that the quality of question creation varies across each sub-part of the generation process. Fluency is high across all generations, which aligns with results found in Dijkstra *et al*. Question generation is generally good, although there are issues around the answerability of questions, with some questions having no clear answer. For both evaluations, Distractor generation was marked quite low, with distractors being distracting only 69% of the time with hallucination detection enabled. Although this was the case, there were cases where distractors were effective, with one of the Teacher evaluators describing them as "really plausible, which makes participants have to

[14] Note: there is no need to evaluate short-answer questions separate from multiple-choice questions as short answer question generation simply uses two of the three parts used in multiple choice questions generation.

think". An example of this was the generated question *Which Queen song was a hit in 1975?* Options: *Bohemian Rhapsody, We Will Rock You, Another One Bites the Dust, Stairway to Heaven*. Three of these are queen songs although only *Bohemian Rhapsody* was released in 1975. The third is a song from a different band, indicating a sort of very wrong answer.

Potentially, the most alarming result might be the 74% of questions which had factually wrong answers and 64% whose answers didn't answer the question. This is significantly worse than the target set by the accuracy requirements of 90%. However, with hallucination detection these figures improve to 86% and 80%, illustrating the effectiveness of Hallucination Detection. Looking further into this evaluation, when generating the 100 questions with Hallucination Detection 44% of questions were caught as having hallucinations and re-generated. The effects of hallucination Detection are less significant, and likely negligible, when it comes to improving distractor generation, due to the high variety in the evaluation. This should be considered in future work, potentially looking at the prompts used when trying to catch distractor hallucinations.

	Questions			Answers			Distractors		
	Fluency	Relevancy	Answer ability	Fluency	Correctness	Validity	Fluency	Coherence	Distracting Ability
Mean	*97%*	*100%*	*84%*	*95%*	*74%*	*64%*	*97%*	*89%*	*62%*
Stand. Div	*±2.74%*	*±0.00%*	*±9.62%*	*±5.00%*	*±10.84%*	*±11.40%*	*±4.47%*	*±12.45%*	*±16.05%*

Figure 24. Results of question generation evaluation with no hallucination detection

	Questions			Answers			Distractors		
	Fluency	Relevancy	Answer ability	Fluency	Correctness	Validity	Fluency	Coherence	Distracting Ability
Mean	*97%*	*98%*	*93%*	*98%*	*86%*	*80%*	*96%*	*96%*	*69%*
Stand. Div	*±4.91%*	*±4.97%*	*±4.68%*	*±4.47%*	*±11.83%*	*±17.68%*	*±8.94%*	*±6.52%*	*±25.21%*

Figure 25. Results of question generation evaluation with hallucination detection

	Correctness	Valid	Distractibility
No Hallucination Detection	74%	64%	62%
Hallucination Detection	86%	80%	69%

Figure 26. Direct comparison between results on certain criteria between question generation with and without hallucination detection

Evaluation of the marking of questions using LLMs

Automated question marking is evaluated by having the system mark the same questions with different answers, and then comparing the mark given against the expected mark. First, twenty of the questions are generated by the model, then four curated answers are written in an attempt to answer them: one identical to the model answer, one semantically correct but syntactically different, one semantically incorrect but syntactically or contextually similar, and one completely wrong (Figure 27).

The program was asked to evaluate all the answers. The accuracy of each evaluation is done by counting the number of false positives (either expecting true marking false, or expecting false and marking true). The results indicated that the accuracy of question marking is around **93%**, i.e. 93% of marks as expected. The most accurate marking is when the answer is wrong with **100%** accuracy

during the test. The worst accuracy is when marking questions that are semantically wrong but syntactically similar, although still with an accuracy of **85%**.

This evaluation shows that LLMs can be used to mark short-answer questions with a high level of accuracy although there is still significant room for improvement. The most common and best case for marking is likely marking questions that are semantically wrong but syntactically similar - this represents when a student has an understanding of the subject area but has got this particular question wrong, having the best chance of improving their understanding. More work should be done to improve the accuracy of that case particularly.

Question: What is the setting of Macbeth? Expected Answer: Scotland		Expected Mark
Model Answer	Scotland.	True
Semantically correct but syntactically different	The Kingdom of Scotland.	True
Semantically incorrect but contextually similar	The Globe Theater	False
Completely wrong	qwertyuiopasdfghjklzxcvbnm	False

Figure 27. Different types of answers were used to evaluate the automated marking of short-answer questions. This table illustrates an example of one of the questions used in the evaluation.

Evaluation of adaptive feedback using LLMs

There is little literature on the creation of feedback using MTL-LLMs so a similar evaluation is done to that of question generation. Feedback is generated for several answers ranging in quality. This feedback is then evaluated by both expected users and experts to evaluate their quality against some criteria (Figure 28).

Metric	*Description*
Fluency	whether the feedback is grammatical and readable
Informative	whether the feedback explains what the correct answer is and what the given answers relate to (if anything)
instructive	whether the feedback explains how the student can improve in the future
Positive	whether the feedback uses a constructive tone with no harmful or demoralising language

Figure 28. The criteria used to evaluate feedback generation.

	Fluency	Informative	instructive	Positive
Mean	*100%*	*95%*	*91%*	*91%*
Stand. Div.	*±0.00%*	*±5.00%*	*±9.62%*	*±13.12%*

Figure 29. The results of feedback evaluation.

Feedback performs well in all categories (Figure 29), successfully informing and instructing students positively and fluently.

An interesting comment is that there is no consistency to whether the LLM thinks the feedback is being given to the student (“I noticed that your answer was not correct”) or a third party (“The student's given answer is not correct”). This is a detail that is not made clear in the prompt and could be added in future development.

A comment given by the expert while evaluating feedback is also that feedback should aim to explain why the answer was wrong. Most feedback satisfied the conditions considered in the requirements, including being informative in both the given answer and explanation, although the feedback rarely linked these two pieces of information to explain why it was wrong. This would be an additional consideration in future feedback generation.

A final comment given by the teacher evaluator is that she felt it was clear that the feedback was written by a machine. She thought that sometimes feedback was unnecessarily superfluous or obvious. Instructions to “proofread” your answer are instructive, although when the answer given is one word long, it makes little sense to suggest this as a way of improving your answer. This comment illustrates that, although the LLM-generated feedback was not bad, it still lacks an understanding of exactly what good feedback consists of which may be off-putting for some users. More work could be done to make sure the feedback is warm and engaging and sounds more like human feedback. This would likely need fine-tuning of the model.

System Evaluation

Users carrying out the end-to-end and exploration testing are asked several questions through a post-survey[15] covering all the application aims as well as the central project aim.

Each evaluator is asked about both the perceived efficacy and their satisfaction with each feature on a scale from *very good, good, ok, poor,* and *very poor*. Each feature has an average ranking for both criteria of at least *ok*, with certain features like account creation/login and progress tracking ranking *very good* and *good* respectively.

All evaluators consider the application to be intuitive, with only one report of the application being confusing.

When considering the final question about the central project aim, *“Do you feel the use of LLMs in the application, in its current state, effectively substitutes the need for a teacher in the creation, marking and feedback of formative assessments?”* Most evaluators responded no. In discussion with the evaluators, the primary reason given for this was the time taken to create quizzes and give feedback. If these times were reduced they would have answered differently. This should be a primary factor when considering future work.

Acceptance Testing

(must) ...be able to create and log in with a personal account.	
Users must be able to create an account through the use of an email and a password.	✔
Passwords must never be shown or stored in plaintext.	✔
Passwords must be double-checked before the account is created by asking the users to enter them twice.	✔

[15] Full details of post-survey in appendix 2

Passwords must be restricted by security requirements (minimum length, inclusion of number, inclusion of special character).	✔
Users must be able to log in using their email and password.	✔
All webpages must reroute to the login page (except the create account page) when the user is not logged in.	✔*
(must) …be able to create their own **multiple-choice** quizzes on whatever topic they want.	
Users must be able to create quizzes, with questions that are made up of a question, an answer and three distractors (semantically wrong but syntactically similar answers).	✔
Users must be able to input a title and a description to influence the content the questions are about.	✔
Users must be able to specify the number of questions the quiz will contain.	✔
(must) …be able to take their quizzes and view their results.	
Users must have access only to quizzes that they have created.	✔
Users must be able to take the quiz through a dedicated page.	✔
Users must be given questions one at a time.	✔
Users must have the option to return to questions after they have answered them.	✔
Users must not be able to see what questions they have got right or wrong while doing the quiz.	✔
Users must be able to leave the quiz at any time, return to the dashboard, and be informed that they will lose their progress if doing so.	✔
Upon quiz completion, users must be taken to a dedicated page to view the results of the quiz.	✔
(should) …be able to create their own **short-answer** quizzes on whatever topic they want.	
All requirements for multiple-choice quizzes should also be satisfied by short answer quizzes (without the addition of distractor answers).	✔
Users' answers should be evaluated on semantic rather than syntactic bases, not needing their answers to identically match the expected answers.	✔
(should) …get adaptive feedback on questions they have answered wrong.	
After completing the quiz, users should be provided with feedback on questions they got wrong.	✔
This feedback should be informative, explaining what the correct answer is and what the given answers relate to (if anything).	✔
This feedback should be instructive, explaining how the student can improve in the future.	✔
This feedback should be positive, with a constructive tone with no harmful or demoralising comments used.	✔
(should) …be able to track their progress over time for a given quiz.	
Users can see a record of past attempts for each quiz.	✔
Users can see what answers they gave on those attempts.	✔
Users can see a visualisation of their progress over time.	✔
(could) …be able to select from a list of predefined topics (e.g. History, Geography, Physics) when creating a quiz.	

Questions could be sorted by topic.	✔
The topic could influence the content of the questions created.	✔
The list could include the option ‘other’, leading to no topic being picked, and do not affect the question created.	✔
(could) …be able to take a quiz with an endless amount of questions, generating more as it goes along.	
When creating a quiz, users could select the option for an endless quiz.	N/A**
Endless quizzes could make new questions as the user takes the quiz indefinitely.	N/A**
Users could be able to end the quiz on a question of their choosing, at which point the quiz will be saved at that length.	N/A**
(could) …be able to learn and get guidance on how best to use the application.	
An information page could exist that is accessible to the user.	✔
Information could detail how the application works.	✔
Information could detail how to structure the title and description.	✔
The user could use a form to send an email to a page editor if they are unsure about the information details on the page.	✔

Figure 30. The results of the acceptance testing

**The information page does not reroute to the login page when the user is logged in. This aim was changed during development. No reason could be found for it to re-route to the login page.*
*** The endless quiz feature was deprecated due to the Repeated Questions Problem*

This evaluation shows that the application meets most of the requirements set at the start of the project, except the qualitative evaluation of accuracy, explained in prior sections.

Comparing the project against the non-functional requirements shows that there are many successes, as well as some failures. The application is usable in regards to convenience and intuitiveness, as shown by the post-survey. Users successfully answered questions about how the system worked during the user evaluation, confirming the application as explainable. The application remains freely accessible despite the limitations that this puts on development. The security of the system was ensured to a level deemed suitable for this project. The application is compatible with several different browsers and upholds maintainable design practices throughout development.

On the other hand, the usability of the system in regards to speed is significantly limited by the hardware that the LM-Studio runs on. This causes question generation, marking and feedback generation to all be slower than desired. A further consideration should be made towards the accuracy of question generation. Although the accuracy of generated questions is promising, the current instance of the application would not be a viable product to use for educational purposes, and significant improvement would be needed before it does replace the teacher user.

Discussion

Deficiencies

It was initially planned that a Client-Server Architecture would be used in this project although issues arose during the sprint when the LLM API was integrated in the app. Before this point, sprints had been carried out building the app using Firebase and it had been effectively used throughout authentication and database management. It was then realised that, to handle API calls from the server, a paid version of Firebase was needed. Due to the lack of budget in this project, the options were to handle the API calls on the front end or to find a new back end and re-implement the existing features. It was decided not to go back and re-develop features, so the API was integrated with the front end. This was a breach of the intended Client-Server Architecture. The problems for this were only detected later in development. These problems could have been foreseen if the Client-Server Architecture was better understood and, through reflection on the experience has provided a better understanding and appreciation for sticking to planned architecture.

This issue of the breach in Client-Server Architecture is representative of another lesson that was learnt: the hyper-focus on working features. Although the BDD development methodology focuses on the implementation of behaviour, the success criteria used for behaviour is passing the unit tests, very similar to the for-runner to BDD: Test Driven Development. During the iterative development process, features are checked to ensure they perform certain behaviours, but the quality of these behaviours is not thoroughly checked from the users' perspective, user evaluation only occurs at the end of the development process. This means that the quality of features is not clear throughout development. Features such as distractor generation pass unit tests although fail in the face of user evaluation. In the future, it would be better to conduct user evaluations throughout the development process to get a better understanding of how well behaviours are implemented and how they can be improved.

A final issue is the volatility of Large Language Models, especially multi-task learning open-source large language models. Lots of time is spent trying to understand what prompts cause what outputs and trying to determine the best prompts to use. This process is hard to predict and often inconsistent, resulting in it taking more time than planned. Similarly, there are several problems inherent to open-source MTL-LLMs, such as certain undesirable topics being accepted inputs. Although there are “guard rails” to stop some topics from being discussed, it is still possible to generate quizzes on some of these, for example, a quiz titled “How to make a bomb” resulting in informative questions. Hallucinations are another one of these inherent problems discussed. Although these can be reduced in many ways there is always a possibility of them slipping through the gaps even when the techniques previously discussed are used.

Achievements

This project enables experience with full-stack web development outside of theory and explores the range of technologies. The app uses Firebase and Bootstrap as central tools in its development. I now feel comfortable, not just in web development, but also in exploring more complex web development technologies such as React or Angular.

Throughout the development of this project, good coding practice is upheld, despite some architectural failings. Modular design is a cornerstone of good agile development and ensures that the code can easily be improved in future development. A uniform and categorical naming convention is an effective way to quickly understand what name is being used, a technique that will be taken forward into future projects. Similarly, JSDoc is a useful standard with great benefits from learning it.

The project's main flaw is its technical limitations, although this paper has set out the framework that could be replicated on higher-performance systems to produce something closer to the desired result. This project is a good foundation for future work. This paper demonstrates the problems that are most likely to be encountered by future work tackling this issue (e.g. the Repeated Question Problem) and some ways of tackling them. It also collates and presents research necessary for understanding this problem.

Conclusion

This work proposes the development of a student-led LLM-powered formative e-assessment web application to improve student's educational independence and free more time for teachers. The application successfully develops all functional requirements, although user evaluation shows that its performance in more qualitative measurements was insufficient to be considered a complete success. The final results are promising and show that this technology does have the potential to achieve the aims set out by this project. However, the application developed is not presently a viable product for real-world educational purposes. I suggest that any application used with current technology still relies on the teacher user to ensure that questions are correct and useful to learning students, and questions are marked accurately. That being said, I do still believe this technology has the potential to achieve this aim as models improve, improving the quality of outputs, and the technology becomes cheaper and more accessible, facilitating greater study and use of higher-performance models.

Future Work

A major possibility for future work is the development of a more marketable version of this product. This would include several features deemed to be outside of the research focus of this project, such as account management, more thorough security testing and a more friendly website for new users (such as the first page being an informative and welcoming landing page rather than just a login page). Another major consideration would be the movement of the LLM API from local hardware to a server, as well as better networking between API servers and backend servers. This could also include the improvement of hardware and computing power used to run the LLM. As shown by Dijkstra et al, their fine-tined model EduQuiz performed significantly better than none fine-tuned models meaning that if there was not this technical limitation, and the project was done again using a different model and LLM API the output quality of outputs would be significantly improved. Generation times could also be greatly reduced by a higher-performance machine. The quality of outputs would be better if a fine-tuned model was used with a larger set of parameters.

Many other features could be considered in future development. These would include a solution to the Repeating Question Problem that allows for parallelism and atomic question generation, allowing the endless question feature to be developed. As well as this, the editing of existing quizzes, the sharing of quizzes with other users and the ability to create quizzes that match the user’s ability would all be interesting to research and develop, as well as having an interest to users of existing applications and potential users.

The generation of feedback using generative LLMs is an area that lacks research and could benefit from a more comparative study. This could involve a collection of data and an example of feedback to find and tune a model or an exploration into effective prompt engineering. This would also need an analysis of pedagogical literature to understand what makes good feedback good.

References

[1] National Foundation for Educational Research, “Assessment 101”, 2020. (Online). Available: https://www.nfer.ac.uk/media/yjnhhvrz/assessment_101_a_traineeearly_career_teachers_handbook_to_primary_assessment_in_england.pdf (Accessed: 01/04/2024).

[2] S. Einig, “Supporting Students’ Learning: The Use of Formative Online Assessments,” Accounting Education, vol. 22, no. 5, 2013, doi: 10.1080/09639284.2013.803868.

[3] P. Black and D. Wiliam, “Inside the black box: Raising standards through classroom assessment,” Phi Delta Kappan, vol. 92, no. 1, 2010, doi: 10.1177/003172171009200119.

[4] H. Ebbinghaus, “Memory: A Contribution to Experimental Psychology,” Annals of Neurosciences, vol. 20, no. 4, 2013, doi: 10.5214/ans.0972.7531.200408.

[5] Kahoot!. Kahoot. https://kahoot.com/ (Accessed: 03/04/2024)

[6] Quizizz Inc. Quizziz. https://quizizz.com/ (Accessed: 03/04/2024)

[7] Oxford University Press. MyMaths. https://www.mymaths.co.uk/ (Accessed: 03/04/2024)

[8] Quizlet Inc. Quizlet. https://quizlet.com/gb (Accessed: 03/04/2024)

[9] Instructure. “What is Canvas?” Canvas LMS Community. https://community.canvaslms.com/t5/Canvas-Basics-Guide/What-is-Canvas/ta-p/45 (Accessed: 03/04/2024)

[10] F. Guay, “Applying Self-Determination Theory to Education: Regulations Types, Psychological Needs, and Autonomy Supporting Behaviors,” *Canadian Journal of School Psychology*, vol. 37, no. 1, 2022, doi: 10.1177/08295735211055355.

[11] A. Vaswani et al., “Attention is all you need,” in Advances in Neural Information Processing Systems, 2017.

[12] J. Devlin, M. W. Chang, K. Lee, and K. Toutanova, “BERT: Pre-training of deep bidirectional transformers for language understanding,” in NAACL HLT 2019 - 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies - Proceedings of the Conference, 2019.

[13] OpenAI. “ChatGPT”. OpenAI. https://openai.com/chatgpt (Accessed: 03/04/2024)

[14] E. Kasneci et al., “ChatGPT for good? On opportunities and challenges of large language models for education,” Learning and Individual Differences, vol. 103. 2023. doi: 10.1016/j.lindif.2023.102274.

[15] J. Håvaldsrud. “Kahoot! launches new AI features for more powerful learning” Kahoot. https://kahoot.com/blog/2023/06/07/kahoot-launches-new-ai-features/ (Accessed: 03/04/2024)

[16] N. Maleki, B. Padmanabhan, and K. Dutta. “AI Hallucinations: A Misnomer Worth Clarifying.” in arXiv, 2024. doi: 10.48550/arXiv.2401.06796

[17] OpenAI. “Pricing”. OpenAI. https://openai.com/pricing (Accessed: 03/04/2024)

[18] R. Dijkstra, Z. Genç, S. Kayal, and J. Kamps, “Reading Comprehension Quiz Generation using Generative Pre-trained Transformers,” in CEUR Workshop Proceedings, 2022.

[19] D. Restifo. “Best Free Formative Assessment Tools and Apps” https://www.techlearning.com/how-to/formative-assessment-tools-and-apps (Accessed: 03/04/2024)

[20] K. Bell. “27 Formative Assessment Tools for Your Classroom.” Shake Up Learning. https://shakeuplearning.com/blog/20-formative-assessment-tools-for-your-classroom/ (Accessed: 03/04/2024)

[21] Kahoot!. “What is Kahoot!?” Kahoot. https://kahoot.com/what-is-kahoot/ (Accessed: 03/04/2024)

[22] Quizizz Inc. “Free online quiz maker” Quizziz. https://quizizz.com/home/quiz-maker?lng=en (Accessed: 03/04/2024)

[23] A. I. Wang, “The wear out effect of a game-based student response system,” Computers and Education, vol. 82, 2015, doi: 10.1016/j.compedu.2014.11.004.

[24] Kahoot!. “Choose your Kahoot! for schools plan” Kahoot. https://kahoot.com/schools/#plans (Accessed: 03/04/2024)

[25] Kahoot!. “How to use Kahoot! AI Question generator” Kahoot Support. https://support.kahoot.com/hc/en-us/articles/17152945038355-How-to-use-Kahoot-AI-Question-generator (Accessed: 03/04/2024)

[26] P. A. Pekka, W. Bauer, U. Bergmann, M. Bieliková, C. Bonefeld-Dahl, Y. Bonnet, L. Bouarfa, et al. “The European Commission’s High-Level Expert Group on Artifcial Intelligence: Ethics Guidelines for Trustworthy AI.” 2018. (Online) Available: https://digital-strategy.ec.europa.eu/en/library/ethics-guidelines-trustworthy-ai (Accessed: 03/04/2024).

[27] Instructure. “Instructor Getting Started Resources” Canvas LMS Community. https://community.canvaslms.com/t5/Instructor-Guide/Instructor-Getting-Started-Resources/ta-p/579378 (Accessed: 03/04/2024)

[28] Instructure. “How do I create a quiz using New Quizzes?” Canvas LMS Community. https://community.canvaslms.com/t5/Instructor-Guide/How-do-I-create-a-quiz-using-New-Quizzes/ta-p/1173 (Accessed: 03/04/2024)

[29] Instructure. “How do I add feedback to a question in New Quizzes?” Canvas LMS Community. https://community.canvaslms.com/t5/Instructor-Guide/How-do-I-add-feedback-to-a-question-in-New-Quizzes/ta-p/591 (Accessed: 03/04/2024)

[30] L. Bayer, “Welcome to Quizlet’s AI Study Era: Studying will never be the same.” Quizlet. https://quizlet.com/blog/ai-study-era (Accessed: 03/04/2024)

[31] A. Saccone, “Quizlet 101: A beginners guide and how to get started.” Quizlet. https://quizlet.com/blog/a-beginners-guide-to-quizlet (Accessed: 03/04/2024)

[32] Quizlet Inc. “QuizletPluz.” Quizlet. https://quizlet.com/upgrade?showPlus (Accessed: 03/04/2024)

[33] Oxford University Press. “Try MyMaths” MyMaths. https://www.mymaths.co.uk/try-mymaths.html (Accessed: 03/04/2024)

[34] J. M. J. Murre and J. Dros, “Replication and analysis of Ebbinghaus’ forgetting curve,” PLoS ONE, vol. 10, no. 7, 2015, doi: 10.1371/journal.pone.0120644.

[35] N. J. Cepeda, H. Pashler, E. Vul, J. T. Wixted, and D. Rohrer, “Distributed practice in verbal recall tasks: A review and quantitative synthesis,” Psychological Bulletin, vol. 132, no. 3, 2006, doi: 10.1037/0033-2909.132.3.354.

[36] J. Sweller, “Cognitive load during problem solving: Effects on learning,” *Cognitive Science*, vol. 12, no. 2, 1988, doi: 10.1016/0364-0213(88)90023-7.

[37] S. Skrabut, “80 Ways to Use ChatGPT in the Classroom.” 2023. ISBN:9798985553741.

[38] T. Adiguzel, M. H. Kaya, and F. K. Cansu, “Revolutionizing education with AI: Exploring the transformative potential of ChatGPT,” *Contemporary Educational Technology*, vol. 15, no. 3. 2023. doi: 10.30935/cedtech/13152.

[39] X. Jia, W. Zhou, X. Sun, and Y. Wu, “EQG-RACE: Examination-Type Question Generation,” in *35th AAAI Conference on Artificial Intelligence, AAAI 2021*, 2021. doi: 10.1609/aaai.v35i14.17553.

[40] X. Zhou, S. Luo, and Y. Wu, “Co-attention hierarchical network: Generating coherent long distractors for reading comprehension,” in *AAAI 2020 - 34th AAAI Conference on Artificial Intelligence*, 2020. doi: 10.1609/aaai.v34i05.6522.

[41] E. Gabajiwala, P. Mehta, R. Singh, and R. Koshy, “Quiz Maker: Automatic Quiz Generation from Text Using NLP,” in *Lecture Notes in Electrical Engineering*, 2022. doi: 10.1007/978-981-19-5037-7_37.

[42] H. A. Nguyen, S. Bhat, S. Moore, N. Bier, and J. Stamper, “Towards Generalized Methods for Automatic Question Generation in Educational Domains,” in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2022. doi: 10.1007/978-3-031-16290-9_20.

[43] F. Qu, X. Jia, and Y. Wu, “Asking Questions Like Educational Experts: Automatically Generating Question-Answer Pairs on Real-World Examination Data,” in *EMNLP 2021 - 2021 Conference on Empirical Methods in Natural Language Processing, Proceedings*, 2021. doi: 10.18653/v1/2021.emnlp-main.202.

[43] R. Rodriguez-Torrealba, E. Garcia-Lopez, and A. Garcia-Cabot, “End-to-End generation of Multiple-Choice questions using Text-to-Text transfer Transformer models,” *Expert Systems with Applications*, vol. 208, 2022, doi: 10.1016/j.eswa.2022.118258.

[44] V. Raina, and M. J. Gales, “Multiple-Choice Question Generation: Towards an Automated Assessment Framework.” ArXiv abs/2209.11830. 2022. doi:10.48550/arXiv.2209.11830

[45] M. Sailer *et al.*, “Adaptive feedback from artificial neural networks facilitates pre-service teachers’ diagnostic reasoning in simulation-based learning,” *Learning and Instruction*, vol. 83, 2023, doi: 10.1016/j.learninstruc.2022.101620.

[46] J. Pfeiffer *et al.*, “FAMULUS: Interactive annotation and feedback generation for teaching diagnostic reasoning,” in *EMNLP-IJCNLP 2019 - 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, Proceedings of System Demonstrations*, 2019. doi: 10.18653/v1/d19-3013.

[47] J. P. Bernius, S. Krusche, and B. Bruegge, “Machine learning based feedback on textual student answers in large courses,” *Computers and Education: Artificial Intelligence*, vol. 3, 2022, doi: 10.1016/j.caeai.2022.100081.

[48] S. Jha, S. K. Jha, P. Lincoln, N. D. Bastian, A. Velasquez, and S. Neema, “Dehallucinating Large Language Models Using Formal Methods Guided Iterative Prompting,” in *Proceedings - 2023 IEEE International Conference on Assured Autonomy, ICAA 2023*, 2023. doi: 10.1109/ICAA58325.2023.00029.

[49] F. Petroni, P. Lewis, A. Piktus, T. Rocktäschel, Y. Wu, A. H. Miller, and S. Riedel. “How Context Affects Language Models' Factual Predictions.” ArXiv abs/2005.04611. 2020. doi:10.24432/C5201W

[50] Q. Dong *et al.*, “A survey for in-context learning,” *Computer Science*, 2022.

[51] A. J. Spasic and D. S. Jankovic, “Using ChatGPT Standard Prompt Engineering Techniques in Lesson Preparation: Role, Instructions and Seed-Word Prompts,” in *2023 58th International Scientific Conference on Information, Communication and Energy Systems and Technologies, ICEST 2023 - Proceedings*, 2023. doi: 10.1109/ICEST58410.2023.10187269.

[52] C. Qian, X. Cong, C. Yang, W. Chen, Y. Su, J. Xu, Z. Liu, M. Sun, and W. Liu, “Communicative Agents for Software Development.” ArXiv, abs/2307.07924. 2023. doi:10.48550/arXiv.2307.07924

[53] P. Manakul, A. Liusie, and M. J. Gales, “SelfCheckGPT: Zero-Resource Black-Box Hallucination Detection for Generative Large Language Models.” ArXiv, abs/2303.08896. 2023. doi:10.48550/arXiv.2303.08896

[54] D. North, “Behavior Modification.” Stickyminds. https://www.stickyminds.com/better-software-magazine/behavior-modification (Accessed: 03/04/2024)

[55] Hot PMO, “MoSCoW or Kano Models – how do you prioritize?” Hot PMO. https://www.hotpmo.com/management-models/moscow-kano-prioritize (Accessed: 03/04/2024)

[56] JSDoc, “@use JSDoc.” JSDoc. https://jsdoc.app/ (Accessed: 03/04/2024)

[57] Microsoft, Visual Studio Code. https://code.visualstudio.com/ (Accessed: 03/04/2024)

[58] Bootstrap, “Bootstrap 5” Get Bootstrap. https://getbootstrap.com/ (Accessed: 03/04/2024)

[59] EmailJS. https://www.emailjs.com/ (Accessed: 03/04/2024)

[60] Google for Developers, Firebase. https://firebase.google.com/ (Accessed: 03/04/2024)

[61] LM Studio. https://lmstudio.ai/ (Accessed: 03/04/2024)

[62] jQuery. https://jquery.com/ (Accessed: 03/04/2024)

[63] Hugging Face. https://huggingface.co/ (Accessed: 03/04/2024)

[64] Meta, “Llama 2: open source, free for research and commercial use.” Llama. https://llama.meta.com/llama2/ (Accessed: 03/04/2024)

[65] Mistral AI, Mistral. https://mistral.ai/ (Accessed: 03/04/2024)

[66] OpenOrca, “OpenOrca.” Hugging Face. https://huggingface.co/Open-Orca (Accessed: 03/04/2024)

[67] S. Sergei, “Fixing Hallucinations in LLMs.” Better Programming. https://betterprogramming.pub/fixing-hallucinations-in-llms-9ff0fd438e33 (Accessed: 03/04/2024)

[68] S. Moore, H. A. Nguyen, N. Bier, T. Domadia, and J. Stamper, “Assessing the Quality of Student-Generated Short Answer Questions Using GPT-3,” in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2022. doi: 10.1007/978-3-031-16290-9_18.

Appendix

Appendix 1: Pre-Survey
Pre-survey Google Form View Analytics

Appendix 2: Post-Survey
Post-Survey Google Form View Analytics

Appendix 3: Unit tests
Italics: denotes a large number of possible inputs
* denotes inputs that are not exhaustive but have been tested are the extremities.
E.g. for the number of questions 2-30*, we test 2, 3, 10, 20, 29, 30

Test	Input	Expected Result
Feature 1: Create an account		
Navigate to create-account page	On the login screen, click Create account	The user is taking to log in page
Successfully creating account	Email: tom.black@gmail.com Password: #Password123	Account, 'tom.black@gmail.com' is created. 'tom.black@gmail.com' becomes the active user. The user is taken to their Dashboard. Their passwords are never shown in plaintext as part of this process
Unsuccessfully creating account	Email: Password: Re-enter Password:	Alert → The email is a required field, please fill it in. An account is not created
	Email: email Password: Re-enter Password:	Alert → The password is a required field, please fill it in. An account is not created
	Email: email Password: 123 Re-enter Password: 123	Alert → The email must be a valid email input, please try again. An account is not created
	Email: jerry.white@gmail.com Password: 123 Re-enter Password: 123	Alert → The password must contain at least 6 characters and include at least one letter, number and special character. An account is not created
	Email: jerry.white@gmail.com Password: password Re-enter Password: password	Alert → The password must contain at least 6 characters and include at least one letter, number and special character. An account is not created
	Email: jerry.white@gmail.com Password: 12345678 Re-enter Password:12345678	Alert → The password must contain at least 6 characters and include at least one letter, number and special character. An account is not created
	Email: jerry.white@gmail.com Password: Password123 Re-enter Password: Password123	Alert → The password must contain at least 6 characters and include at least one letter, number and special character. An account is not created
	Email: jerry.white@gmail.com Password: #Password123 Re-enter Password: different	Alert → The two entries of the password do not match. Both password fields are cleared. An account is not created
	Email: tom.black@gmail.com Password: #Password123 Re-enter Password: #Password123	Alert → This email already has an attached user, please try a different email, or log in. An account is not created

Navigate to the login screen	Using the navbar, click log in	The user is taken to the login page
Page re-routing	When the user is not logged in, using the navbar, click Dashboard	The user is rerouted to the login page
	When the user is not logged in, using the navbar, click Progress	The user is rerouted to the login page
Successful login	Email: tom.black@gmail.com Password: #Password123	Then the account, 'tom.black@gmail.com', becomes the active user. The user is taken to their dashboard. Their password is never shown in plain text.
Unsuccessful login	Email: Password:	Alert → The email is a required field, please fill it in. And the user is not logged in.
	Email: email Password:	Alert → The password is a required field, please fill it in. And the user is not logged in.
	Email: email Password: 123	Alert → The email and/or password you have entered was incorrect, please try a different email and/or password or create a new account. And the user is not logged in.
	Email: jerry.white@gmail.com Password: #WrongPassword123	Alert → The email and/or password you have entered was incorrect, please try a different email and/or password or create a new account. And the user is not logged in.
	Email: tom.black@gmail.com Password: #WrongPassword123	Alert → The email and/or password you have entered was incorrect, please try a different email and/or password or create a new account. And the user is not logged in.
The user navigates to the Dashboard	Given the user is logged in Using the navbar, click Dashboard	The user is taken to their dashboard containing all the quizzes that they have made and a button to create more quiz
		Quiz are sorted using Topic then Title
successfully creating a quiz	When the user clicks Create a Quiz	A form appears allowing the user to input a Title (text), Topic (dropdown), Description (text), Number of questions (number), Type of Quiz [Multiple-choice, Short-answer] (Dropdown)
	Title: *any** Topic: *any* | other Description: *any** | *None* Number of questions: *1-30** Type of Quiz: *Multiple Choice* | *Short Answer*	A quiz is created based on the topics given with a specified length. Each question contains a question and answer pair, and [if multiple-choice] 3 distractors. This quiz is stored in the database. This quiz is visible in the user's dashboard.
unsuccessfully creating a quiz	Title: *None*	Alert → Title field must be filled out. And the quiz is not created.
	Number of questions: *-1, 0, 1, 31, 50*	Alert → The number of questions must be between 2 and 30. And the quiz is not created.
	It should also not be possible to input any unique Topics or Types of Quiz	
Navigate a quiz in the dashboard	In the dashboard on a quiz, click details	Users are shown the details of that quiz, including its full description, number of questions, type and details about its past attempts.
	In the dashboard on a quiz, click Take Quiz	The user is taken to the quiz page for that quiz

	In the details page, click Take Quiz	The user is taken to the quiz page for that quiz
Deleting Quiz	On the details page, click Delete quiz	Quiz is deleted from the database and removed from the user's dashboard
Taking quiz	Users are given questions one at a time, showing the question and [if multiple choice] the options or [if short answer] a text box.	
	User can input their answers and submit	The user is taken to the next question
	Users can go to previous questions	They are shown what they put previously and have the option to change their answer
	On the last question, users can input their answers and submit	They are asked if they are happy with all their answers. If they are, they are taken to the feedback page
	Users should not be able to see the results of their answers until they have finished the quiz and are on the feedback page	
Prematurely ending the quiz	On the quiz page, the user clicks to return to the dashboard	The user is informed that their quiz progress will be lost and asked if they are sure they want to return to the dashboard. If yes, they are taken to their dashboard.
Feedback	On the feedback page, the user is presented with each question that they have answered. They show the question, their given answer and the expected answer. If they get the question wrong, they are also given bespoke feedback.	
	On the feedback page, click Return to dashboard	The user is taken to their dashboard.
Navigate to Progress Tracking	Given the user is logged in, using the navbar, click Progress	The user is taken to their progress page.
	The progress page contains an accordion for every quiz the user has attempted. These accordions detail all their attempts in a visualised graph, including when they were done, what answers were given and what score they had.	
Navigating to the Information Page	From the navbar, click information	The user is taken to the information page
Information Page	The information page contains details about how the application works	
Successful user feedback	In the feedback form, Email: tom.black@gmail.com Feedback: *any** Click submit	An email is sent to SXB1567@student.bham.ac.uk from the given email address (tom.black@gmail.com) containing the given feedback. Submission disabled Alert → Feedback successfully sent The form is cleared and the submission re-enabled
Unsuccessful user feedback	Email: *None* | email	Alert → Please enter a valid email
	Correct information but network issues	Submission disabled Alert → Feedback failed to send. Please check your internet connection and try again later. Submission disabled