Assimilate: Automated Assessment and Bayesian Knowledge Modelling in Teacher Independent Adaptive Learning: A Feasibility Study

By: Adam Doyle/ N00222244

Project Supervisor: Cyril Connolly

Second Reader: Michael McAndrew

Submission Date: 1st May 2026

DL863 Bsc [Hons] Creative Computing

Declaration of Authorship

Incorporating material without formal and proper acknowledgement (even with no deliberate intent to cheat) can constitute plagiarism.

You should document this in your submitted work if you have received significant help with a solution from one or more colleagues. If you doubt what discussion/collaboration is acceptable, consult your lecturer or the Course Director.

WARNING: Take care when discarding program listings lest they be copied by someone else, which may well bring you under suspicion. Do not leave copies of your files on a hard disk where others can access them. Remember that removable media used to transfer work may be removed and/or copied by others if left unattended.
Plagiarism is an act of fraudulence and an offence against the Institute's discipline.

Alleged plagiarism will be investigated and dealt with appropriately by the Institute. Please refer to the Institute Handbook for further details of penalties.

The following is an extract from the B.Sc. in Computing (Hons) course handbook. Please read carefully and sign the declaration below.

Collusion may be defined as more than one person working on an individual assessment. This would include jointly developed solutions, with one individual giving a solution to another, who then makes some changes and hands it up as their work.

DECLARATION

I know the Institute’s policy on plagiarism and certify that this thesis is my work.

Adam Doyle

Abstract

Traditional classroom environments are unable to provide real-time personalised feedback to students. Existing adaptive tools often fail to target specific skill deficiencies of learners.

This paper discusses Assimilate, a teacher independent adaptive learning system designed to automate assessment and provision questions adaptively using Bayesian Knowledge Tracing. This system was implemented using a system of microservices. This was compromised of a Spring Boot gateway, a Fast API template engine, LLM grading service that utilises a Mathstral model and a the BKT Api that uses knowledge tracing to predict mastery.

The systems success was evaluated against a set of predefined criteria. The system successfully automates the grading of submissions and adapts question difficulty in response to performance. Various limitations such as LLM reliability and small skill set are discussed in detail. The project concludes that the systems approach is feasible and sets the groundwork for future work.

Acknowledgements

The author would like to thank Cyril Connolly for supervising this project from start to finish. Cyril provided valuable feedback on the direction of the project and assisted in containing its scope. Without this guidance, the project would not have reached its current form.

Thanks are also extended to Michael McAndrew for being the second reader for this project.

Finally, gratitude is expressed to all survey participants and user testers for their time and contributions.

Table of Contents

1. Introduction

1.1 Context

Traditional educational settings have existed for millennia, typically involving a teacher and students within a classroom. While this system has been a reliable method for providing education, it generally follows a uniform pace and learning style that can be coined as a “one size fits all” approach to learning. Consequently, this form of learning is unable to offer a diversified experience to each individual learner as it does not concern itself with individual needs, abilities and preferences (Meylani, 2024; Wu et al., 2024).

In today’s world, there is a growing demand for personalised education systems. This demand has only been propelled by the most recent developments within the field of artificial intelligence (A.I.) (Cristian Randieri, 2024). These developments enabled the creation of complex systems that possess the ability to analyse student performance, understand student needs and provide instantaneous feedback. This has propelled the industry substantially and the market continues to expand (Precedence Research, 2025). This use of AI has driven the enhancement of the traditional education system into a more personalised and flexible form of learning that could potentially resolve the issues of traditional learning systems. (Farhood et al., 2025)

The proposed solution to the established issues of traditional education is Assimilate. Assimilate is an E-learning application that involves question generation, automated assessment and Bayesian knowledge modelling to create an adaptive learning system. Assimilate provides a highly personalised form of education that caters to the individual learners skill level. It also removes the physical requirement of being within a traditional education setting to learn and receive feedback, this creates a teacher independent learning cycle.

1.2 Objectives & Aims

The aim of this project is to implement and evaluate Assimilate. It aims to assess the feasibility of providing a personalised, automated educational experience without the involvement of a human instructor/tutor.

The main objectives of this project are as follows:

1. Implement a threshold based system that provides LaTeX templates that are aligned with Leaving Cert standard questions.
2. Integrate a Large Language Model (L.L.M.) that is able to grade and examine student submissions.
3. Employ a BKT model that models a learners knowledge over time based on results from LLM grading.
4. Evaluate the feasibility of Assimilate through functional, user and performance testing.

1.3 Success Criteria

- Pipeline operates end to end without human intervention.
- BKT mastery score influences question difficulty.
- LLM grading system provides deterministic results aligned with Leaving Cert stepwise scoring system

1.4 Application Area

This falls under many applications within the realm of technology, the main application areas involved are Artificial Intelligence (A.I.) and E-Learning. Assimilate will be comprised of a closed loop pipeline that uses threshold based question generation. The submission of these questions will be sent to an LLM for grading, it will then be sent to a Bayesian Knowledge Tracing Model (BKT) to provide a prediction on the students mastery level of skills involved. This will then shape the next question that is presented to the user.

2. Research & Background

2.1 Literature Review

2.1.1 Traditional Learning

Traditional learning has been the standard model of education for centuries. Traditional learning is classroom based and teacher-led. Follows a strict curriculum on a set time line for a school year. This system assumes that within a classroom each student possesses an identical ability level and fails to diversify for the requirements of the students.
The curriculum is delivered in a fixed manner, each subject is generally allocated a proportion of weeks within the year. This system does not encourage mastery of a subject rather familiarity with said subject. Despite this, some students perform and exceed expectations, but there are also many students who cannot keep up. (Yue, 2024)
Another issue within this system is that a classroom can range from 10 – 40 students, while there may be only one teacher per group. In theory, one teacher's attention has to be shared amongst each student equally; however, in reality, this often is not the case. Teachers may have biases towards students or simply the inability to teach the student in a way that caters towards the student's needs (CAMPBELL, 2015).

Another aspect to note is the feedback cycle. Feedback is vital to a student's progression within said subject. Students regularly complete work, but receiving personalised feedback from their teacher can be a high-latency process as other students require the same personalised feedback. (Gan et al., 2021) Finally, the last issue that surrounds traditional learning is that it requires attendance.

2.1.2 E-Learning

E-learning can be described as learning that is done online. E-learning was one of the first breakthroughs in changing the traditional learning structure. It allows for the removal of students and teachers from the same setting. This allows for a more flexible way to teach students. A major prevalence within e-learning is the advent of Learning management systems (L.M.S.). LMS can deliver content to students and enable online-based lectures. Despite E-learning being an improvement on the traditional approach by removing the requirement of being present, it still follows the basic principles of traditional learning through teacher-involved feedback.

E-learning applications typically contain predefined courses that have multiple-choice answers. This system is effective for determining a correct answer from a wrong answer; however, it does not possess the ability to explain why the answer was wrong without teacher's input. An example of this can be seen in mathematical assignments. A student is presented with multiple options to answer. Regardless of the option picked, a student's uploaded solution will not be taken into account by the system, but rather the answer they chose. Furthermore, if a student required additional feedback on their solution, it would require the input from an external source, such as a teacher.

LMSs are static systems. They do not generate content, rather act as an accessible file storage for work that the teacher has uploaded. LMS also suffer from the one size fits all approach; it changes depending on the geographical location of the teaching, but it does not change the original problems that traditional learning suffers from regarding to pacing of content and acknowledging each student's individual needs.

2.1.3 Intelligent Tutoring Systems

An Intelligent Tutoring System (ITS) is a computer program that uses Artificial Intelligence (AI) to deliver a personal level of tutoring. The structure of an ITS can vary drastically depending on the system design, but a common pattern can occur. This pattern has four models that create a structure of an ITS. This structure consists of an “expert knowledge model”, this model is responsible for various tasks such as generating questions for a student, providing answers, and explanations. Next is the student model which is responsible for representing and tracking the knowledge of the student as it progresses. The tutoring model exists to provide activities to the student based on the students ability while staying in line with the knowledge model’s parameter’s. The final model is the communication model, this model is the direct translation of all models to the user. This can appear in various ways but as general it would appear as Graphical User Interface. This enables the system to present itself to a user in a suitable manner (Nwana, 1990). Intelligent tutoring systems are closed loop systems, meaning that they do not need any external action to complete the systems goals. This closed loop system is the main support of a teacher-independent system

In the proposed pipeline, the expert knowledge model is a library of defined LaTeX templates that allow for question generation while ensuring integrity. The student model uses Bayesian Knowledge Tracing (BKT) to maintain a probabilistic assumption of the students mastery level. The tutoring model utilises the BKT scores to assign work to students based on difficulty groups that are defined within the LaTeX templates. Finally the communication model is responsible for rendering the LaTeX based problems to the user. The student will answer the questions via LaTeX based input fields to remove cognitive load and create a standard of communication across the system. The proposed pipeline begins with a question being provided from the template system, this is then answered and graded by the LLM system; the results are fed into the BKT model. BKT then returns a mastery level that is then used to dictate what difficulty of template is provided to the user. This system continuously feeds data through itself and updates based on new data without human intervention, this is the basis of the closed loop design.

2.1.4 Adaptive Learning System

According to Montclair State University (n.d.) adaptive learning can be described as "a technique to use data-driven instruction to adjust and tailor learning experiences to meet the individual needs of each student." Learners have different cognitive abilities and experiences that can affect how they learn. Within a traditional e-learning application that mimics a classroom setting a "one-size fits all" style of curriculum can lead to undesirable results (Gligorea et al. 2023). Through the use of Adaptive learning which consists of using AI to analyse the student abilities, empowers students to progress at their own pace. This can be achieved by generating personalised content that is tailored to their knowledge and abilities.

2.1.5 Zone of Proximal Development

The Zone of Proximal development can be defined as the gap between what a learner can accomplish independently and what a learner can accomplish with the help of a knowledgeable person. This is considered the most optimal zone where a student can learn. (Baker et al., 2020). This system uses adaptive learning to constantly keep the student within the scope of their abilities.

2.2 Technical Research

2.2.1 Latex

LaTeX is a typesetting system that is considered the industry standard for communication of mathematical and scientific documents. LaTeX offers precise mathematical typography and equation composition. LaTeX is vital for the delivery of data throughout the system. It was chosen as it standardises the communication between the LLM grading system, AQG templates, and the front end UI.

2.2.2 Template Based Question Generation

Automatic Question Generation uses natural language processing to facilitate unsupervised content generation. This functions without the intervention of a human and works completely autonomously. This system excels in regards to linguistic generation, however it lacks the symbolic logic for mathematics. This can lead to hallucinations within generated content and unsolvable equations (Bender et al., 2021). The proposed solution to this issue is the use of deterministic template-based question generation. This prevalent difference in this system is that it removes the generative aspect of AQG and instead uses a template based system. These templates are predefined and are created in line with the Leaving Certificate Exam questions. This allows for a controlled output that mitigates the chance of hallucinations and follows the Department of Educations' standard for mathematical problems (Lewkowycz et al., n.d.). These templates are defined in LaTeX which ensures mathematical integrity within the templates. This is able to reduce format errors within the system and aligns with the standard communication of data within the pipeline.

Furthermore, template based AQG is directly correlated with the previously mentioned BKT system. Templates can be categorized into a hierarchical difficulty tiers. These tiers act as thresholds of mastery and can only be accessed by reaching the associated mastery score. This architecture is the foundation of adaptive learning. If a student receives a lower mastery score upon completion, the system will detect this and provide templates that are easier to solve. Inversely if mastery score increases, templates of advanced difficulty will be provided to the student. This system exists to keep the student in an optimal zone of proximal development.

2.2.3 Large Language Models for Grading

Building upon the limits of traditional e-learning applications that prioritise the correct answer as opposed to the methodology of a solution, this project integrates Large Language Models (L.L.M.s) to provide a more methodical analysis of a student's answer.
In relation to this project, LLMs will not be used in a generative context, rather it is a deterministic examiner of submitted work. According to Poličar et al. (2025) LLMs acting as automatic assignment graders have similar abilities to that of a teaching assistants.

The proposed system is comprised of an LLM that aims to provide process-oriented assessment rather than the traditional result only assessment. Each step a student answers to reach a solution is sent to an LLM where it simulates heuristic evaluation. This step-wise analysis enables the system to award partial credits for attempted work.

This system tackles the previously established issue of feedback latency. Since the LLM is analysing the submitted work, it is able to identify any methodological issues to the student instantaneously.

The final benefit of this granular assessment is that it corelates with BKT values. It analyses step-wise actions and can more accurately determine if a student understands the subject examined. (Lightman et al., 2024) If a student does reach the final answer but had a correct process it indicates a slip event as opposed to a lack of mastery. Inversely if a student provides the correct answer but does not show the methodology to achieve it, it could be assumed as a guess event which would indicate that the student may not have achieved mastery.

Challenges of LLMs

Using generative systems such as LLMs to grade students work imposes challenges such as hallucinations and inconsistencies. These issues can lead to inaccurate information being presented to the user (Shao, 2025). This is crucial to the central system the pipeline relies on accurate data. Despite the challenges associated there have been strategies implemented to reduce these issues.

One mitigation strategy is setting a limitation on the LLMs temperature. Temperature is a parameter that controls an LLMs creativity/randomness. The temperature will be set to 0.2. This constraint ensures that the model is deterministic in its grading process. In turn this maintains pedagogical integrity.

2.2.4 Knowledge Tracing

Knowledge Tracing can be summarised as modelling a student's changing knowledge state within a subject. Knowledge tracing holds two main assumptions one is that the student has two states of knowledge(learned/unlearned). The other assumption that knowledge tracing makes is that the student's performance is used to dictate the knowledge state. This is done through using the students previous correct/incorrect actions to decide upon state. Corbett and Anderson (1995) noted from their experiment that they abandoned the traditional knowledge tracing model, instead opting to use their own model which yielded higher rates of success.

This initial form of knowledge tracing, despite being basic and ineffective without addition is the core underlying mechanism of the adaptive system. The flaws within this design is that it strictly relies on a binary value to gauge the students' knowledge level in a subject. It does not concern itself with a student's ability to learn, forget, or make mistakes. It assumes that the student either knows/doesn't know the subject. This can lead to unreliable results in assuming one's ability. Despite this system tracking the students' knowledge over time, it does so in an unflexible way, that isn't granular and rather instant due to its binary nature.

2.2.5 Bayesian Knowledge Tracing

Bayesian Knowledge Tracing(BKT) is a probabilistic model. Unlike basic knowledge tracing B.K.T. follows the principles of Baye's theorem which is a formula that updates probabilities based on new evidence (Joyce, 2003). In regards to knowledge tracing BKT contains 4 parameters for a skill, this includes:

- Prior (students probability of prior knowledge),
- Learn (probability a student learns the skill),
- Guess(probability the student guessed the correct answer)
- Slip (probability the student makes a mistake despite knowing the skill).

These four parameters are the core of BKT and differ from traditional knowledge tracing as it allows for a more fluid learning process and provides continual assessment of a student. This in turn allows this form of knowledge tracing to provide an improved prediction capability in regards to traditional knowledge tracing. (Bulut et al., 2023)

Using these four parameters enables the system to keep track of the students ability based on multiple factors. The inclusion of certain parameters such as slip, enable the system to be less harsh toward the student and provide a degree of forgiveness toward human error. The guess parameter can discern whether the student understood the assignment or simply answered correctly but did not understand what they had answered. The prior parameter also makes an assumption of a student's mastery by acknowledging the fact that a student may have prior knowledge within a subject. This accounts for a more realistic value when trying to understand someone's level of ability. Finally the learn parameter estimates the students chance of learning said skill. These four parameters in combination are able to provide what is coined as a "mastery level". This is a probabilistic score ranged from 0 to 1. 0 being no mastery where 1 is mastery achieved. This score scheme provides more accurate data, and a more discernible understanding of where a student is within learning a subject compared to the binary value scheme within traditional Knowledge tracing.

2.2.5.1 Cold Start Problem

In adaptive learning systems an issue arises known as the “Cold Start” problem. This can be defined as “if there is no prior information about the capabilities of a learner or about the difficulty of a set of materials, even an adaptive system can initially be poorly attuned to the needs of a learner.” (van der Velde et al., 2021). There is no certain way to gauge a student’s understanding capabilities before having being examined, similarly there is no way to gauge a subjects difficulty with no data from students. Despite this, there are certain methods to solve this issue. A Bayesian model can mitigate this problem by using initial estimates within the beginning phase that are derived from either learner difficulty or subject difficulty. Bayesian models can also infer a prediction of the probability of forgetting before having been fed data. This type of inferred data leads to what is referred to as a “warm start” and can be optimized from the very beginning unlike other knowledge tracing models. Within the context of BKT the issue of cold start cannot be eliminated, but can be significantly reduced through probabilistic reasoning and through continuous updating of data.

2.2.5.2 Comparison of Knowledge Tracing

There are different variations of modern knowledge tracing. One form of knowledge tracing that was considered for this project was Deep Knowledge Tracing (DKT). Deep knowledge tracing is comprised of Recurrent neural networks and long-short term memory(LSTM) models. The advantage of DKT over BKT is that DKT can autonomously learn the structure of data and categorize it, whereas BKT needs to be categorised and structured with human oversight. DKT also has the ability to predict complex relationships of how one aspect of knowledge can affect another. Furthermore DKT possesses the ability to create more accurate predictions of a learners future ability. (Piech et al., n.d.)

The comparison of BKT vs DKT is one that researchers debate the trade-offs of each, however this implementation of the project has used BKT. BKT was chosen as it possesses deeper insights into cognitive states that are more quantifiable to humans. (Khajah et al., n.d.) In BKT the use of the four parameters can be used to easily understand what is affecting the students score.(An example of this could be when a student has a higher slip percentage rather than low mastery percentage, this indicates a student knows how to complete the problem, but instead shows they have made a mistake in the process. Contrast to BKT, DKT

uses vectors of neurons that are harder for humans to gauge what the cause of the issue the student is having. (Ding & Larson, n.d.)

Another Reason BKT was chosen is due to the previously mentioned cold start problem. Creating a DKT system requires a vast amount of resources and data, this system cannot work as affectively as BKT without an initial influx of data and may take longer to be considered accurate. Whereas BKT allows for predefined prior information that is set to mitigate cold start issues.

3. Requirement Analysis

3.1 Requirement Gathering

3.1.1 Surveys

A survey was conducted to gauge the feasibility of this project. It was created using Microsoft Forms and was distributed via a class group chat. The survey garnered 4 responses in total. Within this survey it prompted participants to provide potential features they would like to see within an application of this nature. The full survey can be found within the appendices submitted alongside this report.

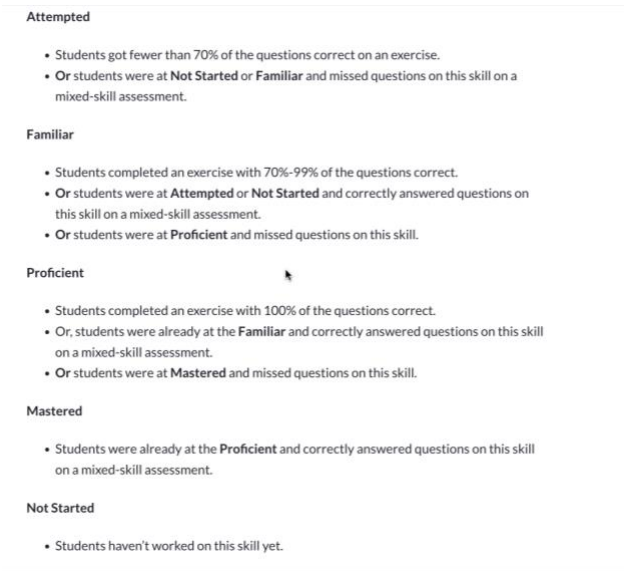
Observing the responses from the survey it is clear that the majority of participants felt that traditional classroom settings account for different learning styles. A large portion of participants felt behind in their learning and felt as if they were struggling to keep up. All participants said that they would find system that is able to predict their knowledge helpful.

3.1.2 Similar Applications

There are numerous sites existing currently that involve adaptive systems to enhance users performance. This chapter will delve into two mainstream applications that are currently using this type of system and what can be learned from it.

Khan Academy

Khan Academy is a not-for-profit online E-learning application that contains a wide range of subjects for students to choose from. It uses a system known as Self-Paced Mastery learning which defines certain threshold that a student can be placed in based on their results. This system is designed to give teachers a better understanding on where their students currently sit within their learning journey (*What Is Self-Paced Mastery?*, 2024). Figure 1 describe these thresholds in detail.



Attempted

- Students got fewer than 70% of the questions correct on an exercise.
- **Or** students were at **Not Started** or **Familiar** and missed questions on this skill on a mixed-skill assessment.

Familiar

- Students completed an exercise with 70%-99% of the questions correct.
- **Or** students were at **Attempted** or **Not Started** and correctly answered questions on this skill on a mixed-skill assessment.
- **Or** students were at **Proficient** and missed questions on this skill.

Proficient

- Students completed an exercise with 100% of the questions correct.
- Or, students were already at the **Familiar** and correctly answered questions on this skill on a mixed-skill assessment.
- **Or** students were at **Mastered** and missed questions on this skill.

Mastered

- Students were already at the **Proficient** and correctly answered questions on this skill on a mixed-skill assessment.

Not Started

- Students haven't worked on this skill yet.

Figure 1- Description of Khan Academy mastery levels

This type of thresholding students into categories is what makes Khan Academy an adaptive application. What separates Assimilate from this is that it uses skill oriented mastery tracking rather than question tracking and thresholding. From this fine grained approach Assimilate is able to offer the users the ability to target micro skills used in solving problems. It also focuses on stepwise grading and process oriented solutions rather than the final answer being marked as correct .Another major difference is that within math questions the UI for khan academy is a standard input box with symbols that can be placed and typed out. Assimilate offers the user the freedom to write their maths using their cursor mimicking a traditional

notebook structure. Assimilate aims to provide a more insightful review of what skills within a problem the student might be struggling with and offers them the ability to work directly on the skills rather than the problem as a whole.

Duolingo

Duolingo is a free to use application that provides user with content on learning a language of their choice. Duolingo uses an adaptive learning algorithm that is able to dynamically render questions for a user based on their shown ability level. If a user obtains a winning streak(a serious of questions answered correctly) the system dynamically changes the content of the questions and increases the difficulty. The inverse of this applies if the user provides a series of incorrect answers. Duolingo also utilizes spaced intervals between question types and content to promote long term retention of knowledge (Geiger-Wolf, 2025).

While a direct comparison between Duolingo and Assimilate based of content types isn't feasible due to the difference in topic nature. The adaptive systems employed within Duolingo informed parts of the design decisions within Assimilate. An immediate example of this is the question dynamically changing based on the users ability level. Despite the question changing based on increased correct answer in series, Assimilate opted to log metrics about the user using BKT to track their mastery scores. This then provides the user with a prompt for their next question to ultimately let them decide how to accomplish their learning adaptively. Another Difference between Assimilate and Duolingo can be found in its repetition model, Duolingo uses spaced repetition to identify if learning is taking place. In contrast Assimilate tracks specific sub skills used across a variety of mathematical problems and can track what sub skills need to be worked upon.

3.2 Requirement Modelling

3.2.1 Functional & Non-Functional Requirements

Below is a list of functional and non-functional requirements they are in no assorted order. These were derived from the project proposal, user survey and similar application analysis.

Functional Requirements

1. The System allows users to log in/register securely.
2. The system presents a user with a question.
3. The system shall convert the students submission into latex.
4. The system shall grade student submissions in a stepwise manner.
5. The system will provide feedback to the user on their submission.
6. The system shall update a student's mastery score after each submission.
7. The mastery score is used to define what question the user receives next.

Non-Functional Requirements

1. The trained BKT model will possess a minimum of 0.60 Area Under Curve metric.
2. The system will use open source alternatives of software to reduce cost.
3. The system will allow users to focus on their weakest/strongest skill.
4. The system will provide insights into the user's mastery over time.
5. The system will be maintained through a decoupled architecture of microservices.
6. The system has a professionally designed frontend.

3.3.2 Use-Case Diagrams

A use case diagram was formed to describe the different interactions that the user will have when using the application. Figure 2 describes the use cases for this project.

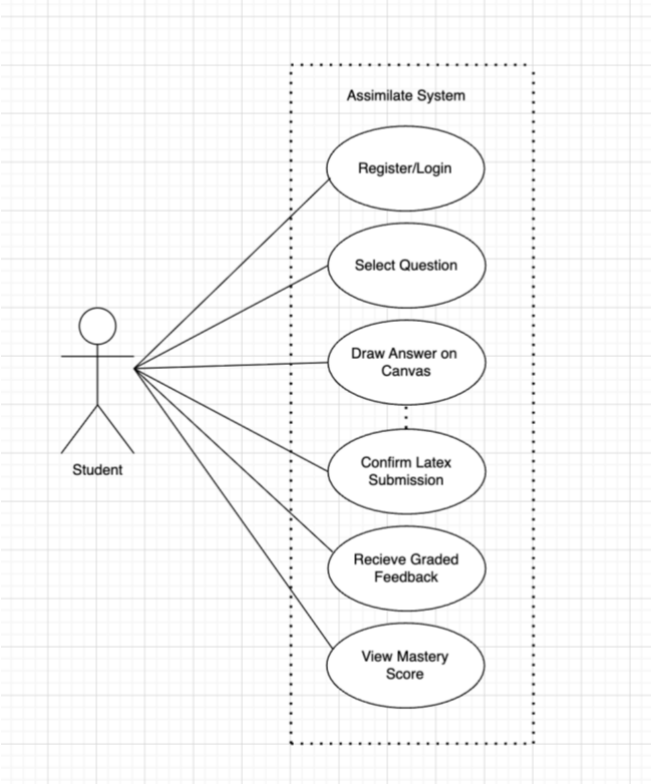


Figure 2- Use Case Diagram

4. Design

4.1 System Architecture

The architecture of assimilate consists of a system of microservices that are connected via a centralised gateway service that handles authentication and routing. The rationale for this decision is described further.

The primary reason for this architecture pattern was fault tolerance. Due to this project containing multiple points of failure (i.e external database calls, LLM calls etc) separating these services increases fault tolerance. If one service is experiencing downtime, it should not affect the other services from performing their operations. Within Assimilate, if the grading service is down the user can still view questions and query their previous history of mastery, this means that Assimilate should be expected to have continuous uptime regardless of service failure.

The second reason for implementing this architecture was due to separation of concerns. Due to the complex nature of each service, running all of them together in a traditional monolithic structure would add additional complexity and make it more challenging to maintain as well as debug. Using microservices also enabled each service to possess their own database, ensuring that each service has full authority and ownership of the data without creating a dependant shared data store. This architecture patten contains an innate ability to scale as required. Each microservice within the system can be scaled or reduced based on demand. This meant more cost efficiency across the platform as only the services that required more received more. This type of scaling would not have been feasible within a monolithic structure.

The final reason for this architecture pattern is due to the ability to work on different services in different languages to fit the requirement of the service itself. This led to different services written in different languages able to communicate to each other using the R.E.S.T. protocol. The gateway utilized Java/Spring Boot for its large auth framework whereas the microservices were made using FastApi which enabled for rapid prototyping of service features.

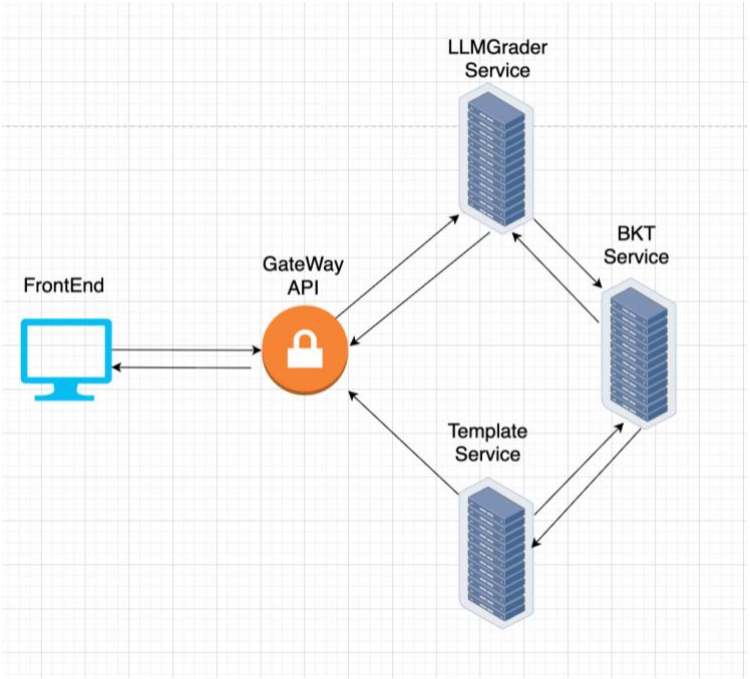


Figure 3 -Proposed System Architecture

4.1.1 UML Diagrams

The following section presents the system architecture of Assimilate. Each diagram is accompanied by a discussion of the design decisions made and the rationale behind them. These diagrams were essential for mapping out the internal structure of Assimilate and its associated functionality.

User Flow Diagram

Figure 4 depicts a diagram of the user flow. This shows the steps and interactions the student will have throughout the duration using the application. It starts with login/register then proceeds to question selection. This ends with a user submitting an answer. The user confirms that their submission is correct before being graded. The submission is returned and the results are then used to provide the next question.

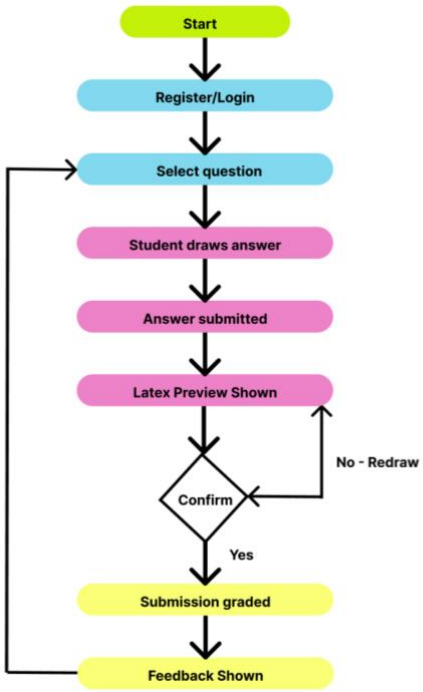


Figure 4 - User Flow Diagram

Component Diagram

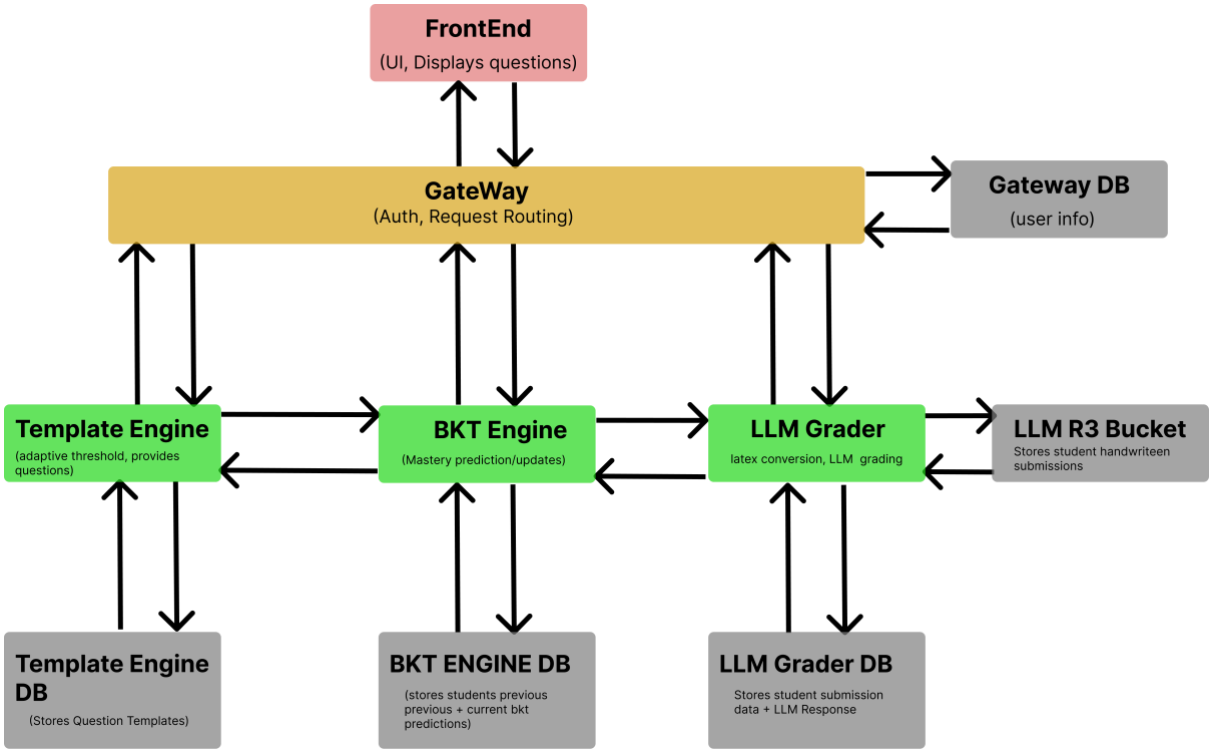


Figure 5- Component Diagram

Sequence Diagram 1: Submission Process

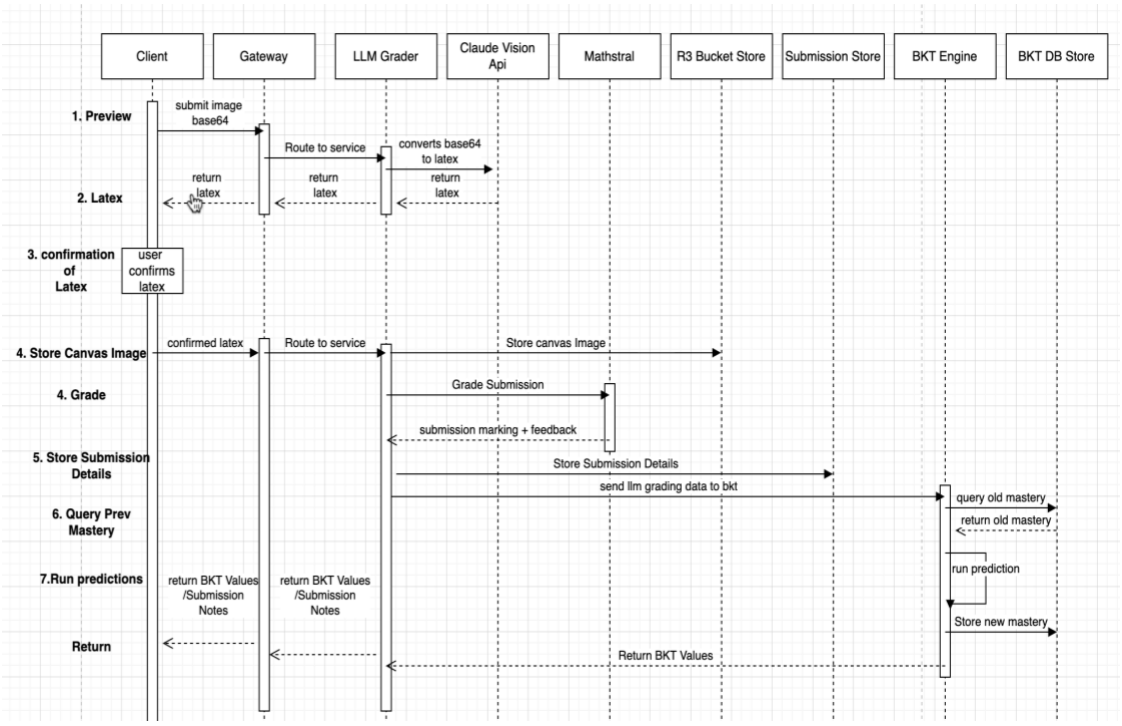


Figure 6 - Submission Sequence Diagram

Figure 6 illustrates the sequence process for the submission and grading pipeline within the application. It describes the request's lifecycle from submission throughout to BKT mastery prediction.

The process starts when the client submits a request to the gateway. The gateway routes this request to the LLM Grader service which then sends the submitted base64 of the user's drawing to Claude Vision API. The base64 is converted to latex and returned to the client for confirmation of submission.

Upon confirmation another request is sent to the gateway which again routes to the LLM Grader service, this time the base64 of the canvas image is stored within an R3 Bucket Store. The LaTeX is then sent to a locally running Mathstral model which grades and reviews the submitted answer. The returned marking is stored within the services database alongside the R3's link to the image submission. This data is then sent to the BKT Engine.

The BKT Engine service receives this data and instantly queries the user's previous mastery from its own database. Using the historical data and the newly provided data, the BKT Engine runs a prediction on what it believes to be the student's new mastery score. This new prediction is stored within the DB and the new BKT data is returned back to the client alongside the Grader LLM's results. This closed loop system resembles the ITS architecture discussed previously.

Sequence Diagram 2: Get Question Process

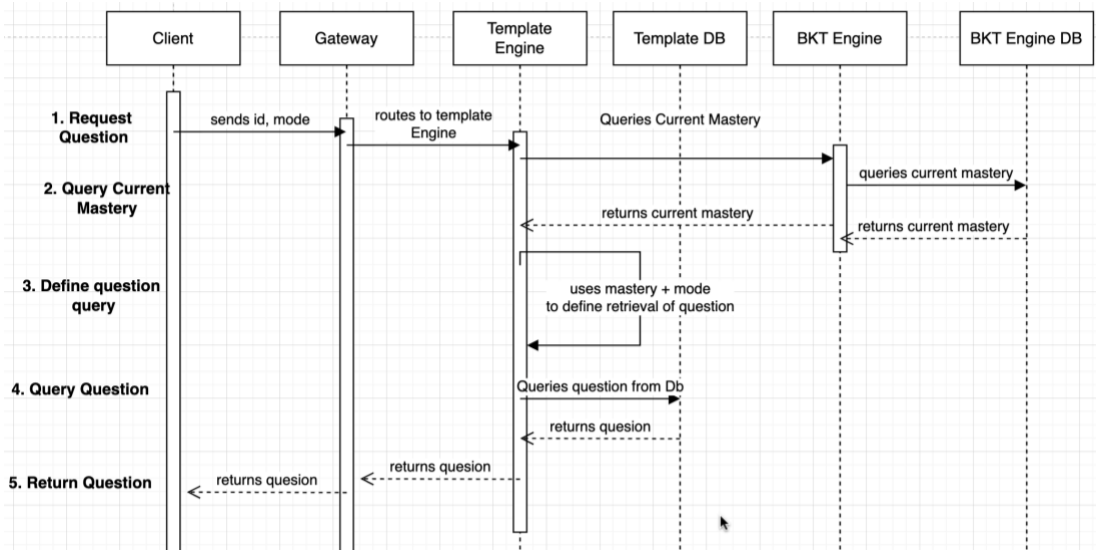


Figure 7 -Get Question Sequence Diagram

Figure 7 describes the process for retrieving a question from the database. The process starts when a user sends request to retrieve a question. The gateway then routes this request to the template engine.

Upon receiving this request the Template Engine queries the BKT Engine to retrieve the users current mastery score. Dependant on what mode the user has sent within the initial request the template engine user certain threshold to then query its own database. The queried template is then returned back to the client ending the process.

4.3 Process Design

4.3.1 Frameworks & Libraries

Throughout the duration of the project there were a multitude of libraries and frameworks used. Below is a list of all primary frameworks used to create this project, included is the rationale as to why it was chosen.

Spring Boot

Spring Boot is a java based web framework. Spring Boot is used within the gateway and handles authentication and request routing. Spring Boot was chosen for its extensive security ecosystem. This enabled implementation of security features without the need of exhaustive custom code.

FastApi

FastApi is a python based web framework that was used for the creation of the different microservice APIs. The reason FastApi was chosen was due to its ability to rapidly prototype and create services, which was critical given the time constraints. FastApi was chosen over other contenders such as Flask due to its native support for asynchronous requests and automatic API documentation. Alongside that FastApi provides built-in support for a multitude of external python services and frameworks such as PyTest for testing, SQLModel for database interactions (Vebrina, 2025).

Mathstral

Mathstral is a seven billion parameter large language model (L.L.M.) developed by Mistral AI. It specialises in mathematical and scientific operations. Mathstral is the LLM that was used within the LLM Grader service for reviewing student submissions. It was chosen for this project due to its multi-step logical reasoning which is critical in mirroring the stepwise grading structure used within Irish Leaving Cert marking scheme. This choice is also supported by Mathstral's performance on industry standard benchmarks, achieving 56.6% on the MATH benchmark and 63.47% on the MMLU benchmark (Mistral AI, 2024).

PyBKT

PyBKT is a python based implementation of Bayesian Knowledge Tracing(BKT) . The PyBKT library was used to train a model to provide parameters for BKT calculations within the BKT service. The reason PyBKT was chosen as it is an accessible framework based in python that offers multiple variants and algorithms to train different BKT models. A major factor in choosing this library is that it was developed citing original BKT papers and has been validated within multiple reports, justifying its use as a practical implementation of BKT (Badrinath et al., 2021).

Claude Vison API

Claude Vision is a vision model that was created and is maintained by Anthropic. The Claude Vision Api was used within the project to transform the base64 from student drawings into LaTeX. Claude vision was chosen over other Optical Character Recognition models(O.C.R.) as it excels when applied to transforming handwritten math notation (*GPT-4o Mini vs. Claude 3.5 Sonnet: A Detailed Comparison for Developers*, 2025).

4.3.2 Algorithms

There are two core algorithms are used to create the adaptive functionality of Assimilate. The Bayesian Knowledge Tracing update, and the adaptive template selection.

Bayesian Knowledge Tracing Update

```
11  # Thoerem observes wheter student gets correct or incorrect then updats its belife on knowledge of skill
12  def updateBKTModel(p_l,correct,p_t, p_f, p_g, p_s ):
13      if correct == 1:  # if answer is correct
14          #evidence that student knows the skill and doesnt slip + doesnt know but guess correctly
15          p_e = p_l * (1 - p_s) + (1-p_l) * p_g
16          p_l = (p_l * (1- p_s)) / p_e
17      else: #if answer is incorrect
18          #evidenve that student does know skill but slips + evidenve that student doesnt know skill and guess incorectly
19          p_e = p_l * p_s + (1 - p_l) * (1 - p_g)
20          p_l = (p_l * p_s) / p_e
21      # Learnign and forgeting transition to update new mastery
22      p_lt = p_l * (1 - p_f) + (1 - p_l) * p_t
23      return  p_lt
```

Figure 8 - Code Snippet of BKT algorithm

Figure 8 displays the algorithm implemented for updating knowledge state within the project. It follows the previously discussed Bayes formula. The algorithm accepts 6 parameters, they are as follows (Line 12):

- P(L) = current mastery
- P(T) = learns
- P(F) = forgets
- P(G) = guesses
- P(S) = slips
- P(e) = evidence (probability of observed response)

When this method is called the algorithm immediately checks if the answer is correct or incorrect. (Line 13 & 17)

If a submission is observed as correct, the evidence is calculated as the probability that a student knew the skill and didn't slip combined with the probability that the student did not know the skill but guessed correctly. (Line 15)

Posterior mastery is then calculated by dividing the probability of knowing the skill and not slipping by the total evidence. (Line 16)

If the submission is observed as incorrect, the evidence is calculated as the probability that the student knew the skill but slipped combined with the probability that the student did not know the skill and did not guess correctly. (Line 19)

The posterior mastery is then calculated by dividing the probability of knowing the skill and slipping by the total evidence. (Line 20)

Finally the new mastery is always calculated the same regardless of answer observation. It is calculated by combining the probability the student knew the skill and didn't forget with the probability the student didn't know the skill but learned it on this attempt. (Line 22)

Adaptive Template Threshold Algorithm

```python
    # Returns template based on user choic using adaptive measures
    async def getAdaptiveTemplate(self, user_id: str, mode: str) -> Template | None:

        # call to bkt engine get mastery endpoint
        async with httpx.AsyncClient() as client:
            response = await client.get(
                f"{os.getenv('BKT_ENGINE_URL')}/mastery/{user_id}"
            )
            masteryData = response.json()

        mastery = masteryData.get("mastery", {})

        # if theres no data for the user just send a random template
        if not mastery:
            templates = self.db.exec(select(Template)).all()
            return random.choice(templates) if templates else None


        # defines the range for zone of proximal development
        zpdSkills = {
            skill: m for skill, m in mastery.items()
            if 0.35 <= m <= 0.85
        }


    # adaptive == if mode chosen is adaptive  choose the template w the highes value in the range
    # weakest == choose the skill with the lowest mastery score
    # Strogest == choose the skill witht the highest mastery score
        if mode == "adaptive":
            if zpdSkills:
                target_skill= min(zpdSkills, key=lambda s: min(abs(zpdSkills[s] - 0.50), abs(zpdSkills[s] - 0.75)))
            else:
                below_zpd = {skill: m for skill, m in mastery.items() if m < 0.30 }
                target_skill = max(below_zpd, key = lambda s: below_zpd[s]) if below_zpd else None
```

Figure 9 - Adaptive Algorithm Code Snippet

```python
    elif mode == "weakest":
        target_skill = min(mastery, key=lambda s: mastery[s]) if mastery else None
    elif mode =="strongest":
        target_skill = max(mastery, key=lambda s: mastery[s]) if mastery else None
    else:
        target_skill = None


    if target_skill is None:
        templates = self.db.exec(select(Template)).all()
        return random.choice(templates) if templates else None




    current_mastery = mastery.get(target_skill, 0)
    if current_mastery < 0.35:
        difficulty = "easy"
    elif current_mastery < 0.65:
        difficulty = "medium"
    else:
        difficulty = "hard"


    template = self.db.exec(select(Template).where(Template.difficulty == difficulty).where(Template.skills.contains([{"skill_name": target_skill}])).order_by(func.random())).first()

    if not template:
        templates = self.db.exec(select(Template).where(Template.difficulty == difficulty)).all()
        template = random.choice(templates) if templates else None


    return template
```

Figure 10 - Adaptive Algorithm Code Snippet Continued

Figures 9 & 10 describe the algorithm that is used to make this an adaptive learning system. This algorithm is vital in defining what question the user gets next using their previous history alongside an additional set of personal choices.

This algorithm has 2 parameters, the user's id and the mode that they chose on the frontend (Line 37). The argument is a value passed from UI element the that user chose on the homepage. The UI element provides a description of what direction the user would like take their learning. The three modes are weakest skill, adaptive and strongest skill. These modes and their functionality will be discussed later in this section.

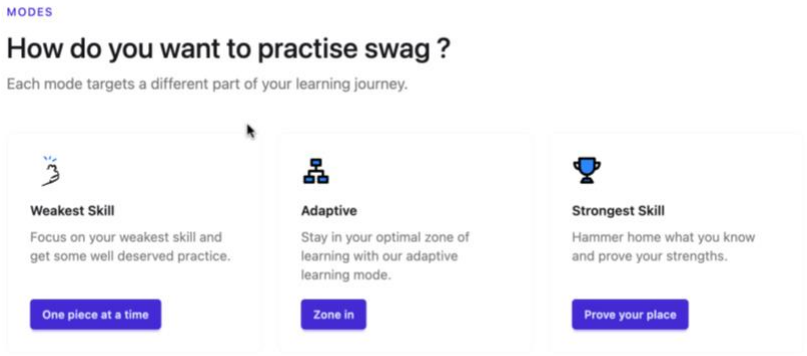


Figure 11- Screenshot of mode selection screen

The next phase of the algorithm queries the BKT Engine service (Line 40 – 46) , this returns the students history. If there is no previous history regardless of mode the algorithm exits early and returns a random template from the database until the student has more mastery data. (Line 49-51)

After querying it loops through each skill within the returned history, it then places any skill within the Zone of Proximal Development (Z.P.D) range into a dictionary (Line 54 - 58). The Z.P.D. range is defined as any skill with a mastery between 0.35 and 0.85. Skills above this threshold can be considered as almost mastered and skill below can be considered underdeveloped.

This is where the algorithm branches functionality based on mode selected (Line 64-83).

If the mode chosen is adaptive, the algorithm searches for a skill that is within the Z.P.D. range. It then uses the absolute difference between the skills mastery score and each threshold to identify the skill to work on. This was chosen as it identifies which skill is closest to reaching a breakthrough to a different threshold. If no skills exist within the Z.P.D band it defaults to the lowest skill that is closest to entering the band (Line 64-69).

If the weakest skill mode is chosen the system will search the history data for the skill with the lowest mastery (Line 73 -74). The strongest mode is the inverse of this and returns the skill from the students history that has the highest mastery from the data (Line 75-76).

If there is no skill selected from this process, the system will default to returning a random template from the database (Line 81 -83).

Next the algorithm defines certain bands to select the question difficulty. This is done by taking the students current mastery and comparing it against the threshold values (Line 88-94)

Finally the algorithm builds a query using the previous variables. It will select a template from the database where the difficulty is set to what was received from the difficulty thresholds. It then will match the target skill to be searched for. To ensure the user doesn't receive the same question from this process, the response from this query is ordered randomly and the first question is chosen.

If no template is found from this the system will return a random template to ensure the user is provided with something from the database. This is then returned to the user on the front end.

4.4 Database Design

4.4.1 Intro

There were multiple databases used throughout the development of Assimilate. Each microservice contained its own isolated database following the best practice standards for a system of microservices as outlined previously. The different types of databases and rationale for their selection will be discussed in detail in this chapter.

4.4.2 MongoDB

MongoDB is a NoSQL database that utilises JSON documents to manage storage of data. It excels in storing complex and unorganised data exceedingly well. Alongside this, it is extremely flexible and doesn't enforce structure (MongoDB, n.d.). MongoDB was used as the primary data storage for two services; the BKT Engine and the Gateway Auth Storage.

Within the BKT Engine MongoDB was chosen due to large volume of unstructured JSON data that is being stored for each users current and historic mastery scores. Each skill acts as a key to their mastery values and nested within these skills is their current mastery alongside the nested history of previous mastery. This structure varies depending on skill and is only updated if the according skill is being examined making a structured key related database such as PostgreSQL unsuitable.

MongoDB was also used within the gateway for storing information about the user. It was chosen for its quick search capabilities without creating complex queries to search users.

4.2.3 PostgreSQL

PostgreSQL is an SQL based database that enforces structured schemas for data storage and enable complex querying. (Scott, 2024) It was selected for services that have structured data that will remain consistent providing quick search and insertion methods.

The Template Engine service utilizes PostgreSQL to store the LaTeX templates that are presented to the user. Each template contains set fields, data types and attributes. Due to this structured nature PostgreSQL was a natural fit over MongoDB.

The LLM Grader service primary database storage was PostgreSQL. This was used to store submission data from the user. Each submission contained a fixed set of fields and datatypes similar to the Template Engine service. Alongside this using PostgreSQL ensured data integrity across submissions making it the logical choice.

4.2.3 Cloudflare R2 Bucket Storage

A Cloudflare R2 bucket was used to store the students handwritten canvas drawings. Bucket storage was chosen as storing large binary files within databases can increase latency (Golas, 2021). Once submitted into the bucket file a URL is generated and then stored within the submission database table for quick access to the students work.

4.4.5 Entity Relationship Diagrams

There were no explicit relationships defined within the project due to the isolated nature of each services database. This section will highlight the main tables schema.

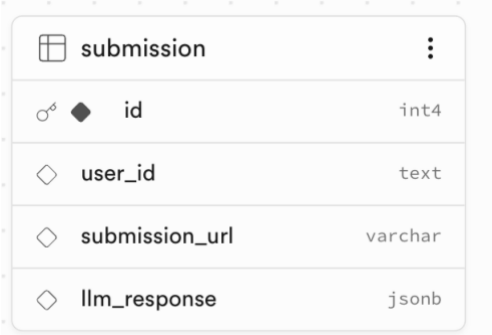


Figure 12 - Submission entity diagram

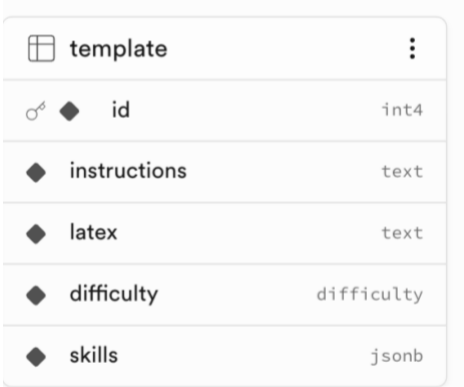


Figure 13 - Template Entity Diagram

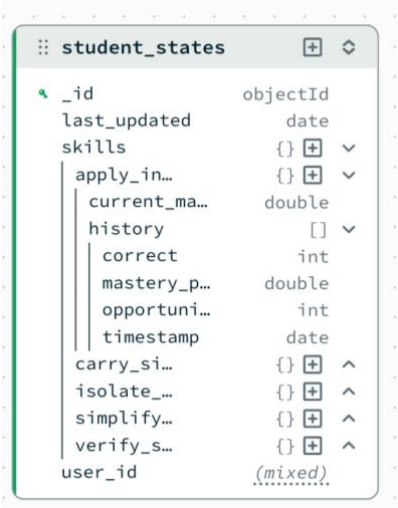


Figure 14 – student state entity diagram

5. Implementation

5.1 Methodology

This chapter will discuss the implementation of the project from January 26th to May 1st. Implementation of Assimilate followed an Agile methodology which consisted of several 2-week sprints. However in practice this methodology had fallen short throughout the project due the dependant nature of interconnected services. Alongside this issue, certain sprints were initially planned in order of completion that would not make logical sense as there were prerequisites that were required for their implementation.

Due to these issues the sprint methodology was altered slightly, each sprint was followed accordingly with modifications to the initial plans set out in the proposal that will be discussed in detail further. Despite the exact sprint roadmap not being abided by, this roadmap provided a framework for time keeping of feature development and progress tracking throughout development.

Sprint 1: Gateway Service (January 26th – February 8th)

The goal of the first sprint was to develop the gateway service. The gateway is the central point of this application and acts as the middle layer between the front end client and microservices.
During this sprint I implemented auth functionality within the Spring Boot service. This part of the project is the one that went the smoothest as there are vast amount of learning materials online and tutorials for setting up authentication within the Spring framework. One difficulty I had during this sprint was that I opted to use the newest version of Spring Security, which did not have as much publicly available resources on the new features and ways to do things. The Java/Spring ecosystem documentation is quite unsightly and navigating the documentation did take up a substantial amount of time throughout this sprint. At the end of this sprint I had created basic auth functionality that contained registration and login functionality, alongside user information being stored within a locally running MongoDB Instance.

One issue raised during this sprint was the choice of authentication. Initially a JSON web token (J.W.T.) method of auth was chosen due to it being stateless and not requiring to be stored or queried from a database other forms of auth such as session cookies. Despite J.W.T being chosen, it brough its own risks as standard implementations of J.W.T store tokens within local storage on the client side. This leaves the client open to a myriad of attacks, one such being cross site scripting (X.S.S). To mitigate this vulnerability, J.W.T. tokens were stored within H.T.T.P. - only secure cookies and sent to the user. Opting to use this method does bring its own issue such as C.S.R.F. attacks but with a strictly defined C.O.R.S. policy on the gateway it mitigates the issue at hand.

Sprint 2: Template Engine Service (February 9th - February 22nd)

The second sprint consisted of developing the Template Engine service. The main objective of this sprint was to implement a system where a user could send a GET request to this service and receive a template question from the backend.

There were two main issues experienced during this sprint. The main issue encountered was due to opting to use an online instance of MongoDB as the development environment is entirely on a laptop which only contains 8 gigabytes of ram, making it unsustainable to run multiple local instances of MongoDB. The hosted MongoDB instance worked perfectly apart from when in College. This problem was narrowed down to the college's ICT department rules. Any request made to the hosted instance would instantly fail unless external networks or mobile data. This made development challenging when demonstrating or working on the project as using hotspot which halted progress immensely due to latency.

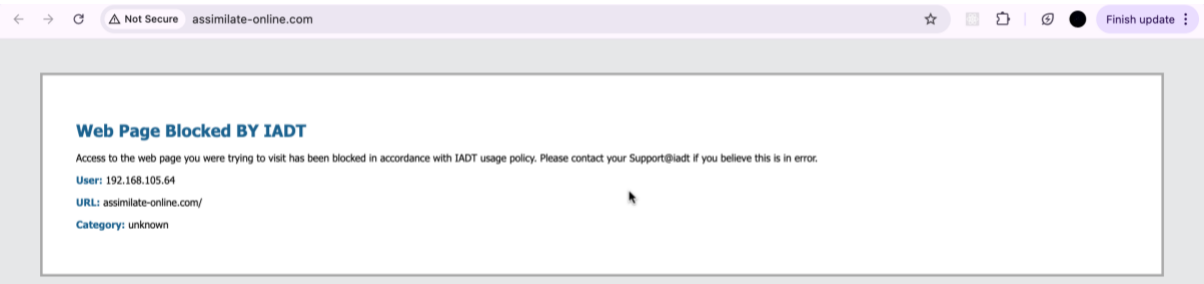


Figure 15 - Error message

The second issue encountered within this sprint was that I couldn't fully implement the adaptive template section as I hadn't defined the other services that fed the data. I could plan and hypothesize what the functions would look like but without the actual data being fed into this service it would be inadequate to define this service fully. This led me to changing how I implemented this project.

By the end of this sprint basic retrieval of question had been implemented alongside the storage and creation of different types of questions that could be presented to the user. Alongside this I opted to redefine how I implemented this project moving forward.

Restructuring of Implementation

After the complications of a sprint based implementation became prevalent, implementation of Assimilate transferred toward incremental development. This decision was made to ensure that the core functionality of the pipeline was preserved and prioritised. This also reduced idling throughout the development process. This required following a strict mapping of the functionality for future phases within the project. Using the previously established architecture diagrams this transition did not impact or urge a major redesign. The original sprints were kept in place to be used as a time keeping system and provide a more general overview of where the system should be by each sprint. Further discussion with the project supervisor about this issue validated the decision to transition to an incremental development approach.

Sprint 3: LLM Grader Service (February 23rd - March 9th)

The 3rd Sprint consisted of developing the LLM Grader Service. This is one of the core objectives outlined within the project aims. Before implementing the LLM Service a simple frontend was created that contained a canvas component. This was done to provide the vision model with image data to convert. Claude Vison Api docs recommend images be sent as base64 strings. This is implemented by converting the users canvas drawing as a PNG then encoding the PNG as a base64 string. This is then routed to the grading service to get sent to the Claude Vision Api.

```
// Sends request to grading system
const previewSubmission = async () => {
  const PNG = await canvasRef.current.exportImage("png");
  //console.log(PNG);

  const base64 = PNG.split(",")[1];
```

Figure 16 - Code Snippet of Base64 Conversion

The next service implemented within the LLM grader service was the grading service. This is the pivotal function of this service as it takes the returned latex from Claude Vision and calls the Mathstral model to review the users submitted answer returning a graded response. This

service proved to be a challenge when implementing it. Due to the risk of hallucination from Mathstral, prompt engineering was required to ensure that Mathstral returned the exact data required to match the response model. This involved multiple revisions of the system prompt which accumulated a lot of time testing the responses returned from Mathstral. This prompt can be seen in figure 17. A more accurate explanation of the Mathstral responses will be discussed in testing. Despite the challenges presented within this chapter, Sprint 3 was completed successfully on time with all major functionality working correctly.

```
37
38      try:
39          systemPrompt = """You are a Leaving Certificate Examiner.
40          Solve the question yourself first. Your answer is the ground truth.
41          Assess the student's submission against YOUR answer line by line.
42          A skill is only correct if the student wrote the correct numbers - wrong numbers or wrong answer means 0.
43          Return only valid JSON. No commentary, no markdown, no code fences."""
44
45          userPrompt = f"""
46      QUESTION: {template['instructions']}: {template['latex']}
47
48      SKILLS TO ASSESS:
49      {json.dumps(template['skills'], indent=2)}
50
51      STUDENT SUBMISSION (LaTeX): {latex}
52
53      Solve the question yourself first, then check the student's submission against your answer.
54      For each skill in SKILLS TO ASSESS return 1 if the student applied it correctly with the right numbers, 0 if not.
55      Only return observations for skills listed in SKILLS TO ASSESS, nothing else.
56
57      Return only this JSON with no trailing commas:
58      {{
59        "bkt_observations": [
60          {{
61            "skill_name": "<exact skill_name from SKILLS TO ASSESS>",
62            "correct": <1 or 0>
63          }}
64        ],
65        "notes": "brief feedback, if correct return Great job!"
66      }}"""
```

Figure 17 - Mathstral Prompt

Sprint 4: Training BKT Model (March 10th - March 29th)

Sprint 4 focused on training the BKT model to be used within the project. This involved installing the PyBKT model and gathering data to be used to train the model. PyBKT allows for multiple variants of the model to be trained and tested. Following the literature the base model + forgets variant is considered the most accurate variant as it providing the highest Area Under Curve (A.U.C.) metric at 0.83. (Badrinath et al., 2021) This metric measures how well the model can distinguish between two outcomes, in relation to BKT this is whether a student has mastered a skill or not.

One major issue during this sprint was installing the library within the development environment. PyBKT initially released on April 8th 2020. Due to the fact that this library is as of writing, 6 years old, the versions of software required to run the library were not within the development environment as it contained modern versions the software used. This required creating a virtual environment to run the library with the intended software. During this process certain python versions, and binaries were unavailable to find online this lead to certain core functionality within the library being rendered impossible to use without altering the source code. This was resolved eventually but to ensure that the replication of this project is feasible for others use, a detailed read.me file was created which documented all the required versions of software tools and any internal changes made to the PyBKT source code with explanations of what was being changed.
This process took a long time and this was before any training was accomplished despite this the project continued ahead behind schedule.

The second issue encountered during this sprint was the lack of data to train the model. The libraries documentation listed two helpful sample datasets to understand the model, however these were not applicable to Assimilate as these datasets focused solely on large problems rather than the subset of skills used to solve these problems. To navigate this issue the dataset used to train the model has to be synthesized. This is becoming a common solution to this problem in training LLMs due to the lack of new content to train LLMs (IBM, 2023).

The final issue during the training of the BKT model was the dev environments computational power. As previously mentioned the dev environment consisted of a laptop with no graphical processing unit(G.P.U.). Training the BKT model is heavy on

computational resources. This made the process of training the model lengthy, rendering any other development impossible during training batches.

Throughout this sprint multiple model variants were trained and had cross validation ran on them to ensure accuracy. As the literature suggested the base model + forgets yielded the highest A.U.C score coming in at 0.73%. This difference between the literature and the projects models projection can be assumed to the difference in datasets used to train the models. Despite this difference from the literature, the model exceeds the desired accuracy outlined within the project proposal.

Sprint 5: BKT Service (March 30th – April 5th)

This sprint oversaw the development of the BKT service. This services sole responsibility was to take the values from the LLM Grader service and run predictions on it using the BKT formula. This sprint was less complicated compared to other sprints as it simply took the parameters obtained from training the BKT model and applied them to the formula. The previous chapter on Algorithms covered how this worked in detail. Once this system was implemented, the adaptive threshold service was added to the template engine.

Sprint 6: Front End & Wiring & Hosting (April 7th – April 20th)

Sprint 6 focused on routing the different endpoints of the microservices through the gateway as throughout development each endpoint was queried directly. Also within this sprint the front end was worked upon, this involved designing the way data was displayed and creating functions that queried the gateway.

After this was completed the application was then containerised via docker, this was to ensure each system worked in isolation and allowed easy traversal of components to different providers.

The biggest challenge in this sprint was hosting the application, due to the complex structure of the project, an Amazon EC2 instance was used to host the application. This worked well upon till trying to configure the backend of the system with the frontend. Netlify was chosen for the frontend hosting provider due to its ease of use, however this brought the issue protocol mismatch. Netlify uses the H.T.T.P.S. secure protocol where EC2 instances run using a H.T.T.P protocol. This meant a proxy server was required to convert the request sent from the EC2 instance H.T.T.P.S. to suit Netlify's requirements.

5.2 AI Usage

This section will discuss the usage of A.I. throughout developing the project. A.I. tools were utilised for various tasks within the project. One major use of it was for synthesising data to train the B.K.T. model due to the lack of sufficient training data. Another use for A.I. within the project was for rapid template question generation. The questions were then stored within the template database.

Throughout the implementation of Assimilate, A.I. was treated as if it were a senior engineer. One major use was reviewing code and discussing potential edge cases that would need to be considered from implemented code. Another use during implementation was for advanced error fixing. The final use case for AI during development was for implementing grid design on the front end.

AI was also used during testing to hypothesize certain edge cases for functions that wouldn't be immediately apparent. All suggestions from A.I. were reviewed, researched, and tested before implementation.

6. Testing & Evaluation

There were various tests ran throughout the duration of the project. Some of these tests included unit tests, feature/integration tests as well as model accuracy tests. These will be discussed in this chapter alongside notes of testing.

PyTest was used to run test cases against various methods within the backend. These tested if the functions provided what it was meant to and to fail if edge cases were hit. This ensured the core functionality was working and could be relied upon and ran during each update to the codebase to ensure baseline functionality.

The next tests carried out were integration/feature tests. These were tests done within insomnia to test endpoints of the project to see if the expected request and response structure was being sent and received. Alongside this incorrect request bodies were sent to ensure that the system would reject any format it didn't expect. This was done through the use of defining request bodies within the microservices, if the body of the request didn't match the expected structure it would fail to process the request.

User tests were carried out once a working prototype was built. There were three user tests done due to the time constraints. The task plan and questionnaire/feedback can be found within the appendices folder. From this user testing it is clear that the project is functional however it does have its limitations and requires editions that need to be added. The main concerns were:

- Efficiency of hand drawn submissions on laptops.
- Full submission history not being present on the application.
- UI needs more user feedback (Success/Error messages)

The final tests carried out were model accuracy tests. These tests were used to test the accuracy of the different trained models before implementing one into the BKT engine. This was done to ensure a better performing model was chosen which will increase the overall accuracy of the system.

6.2 Evidence of Tests

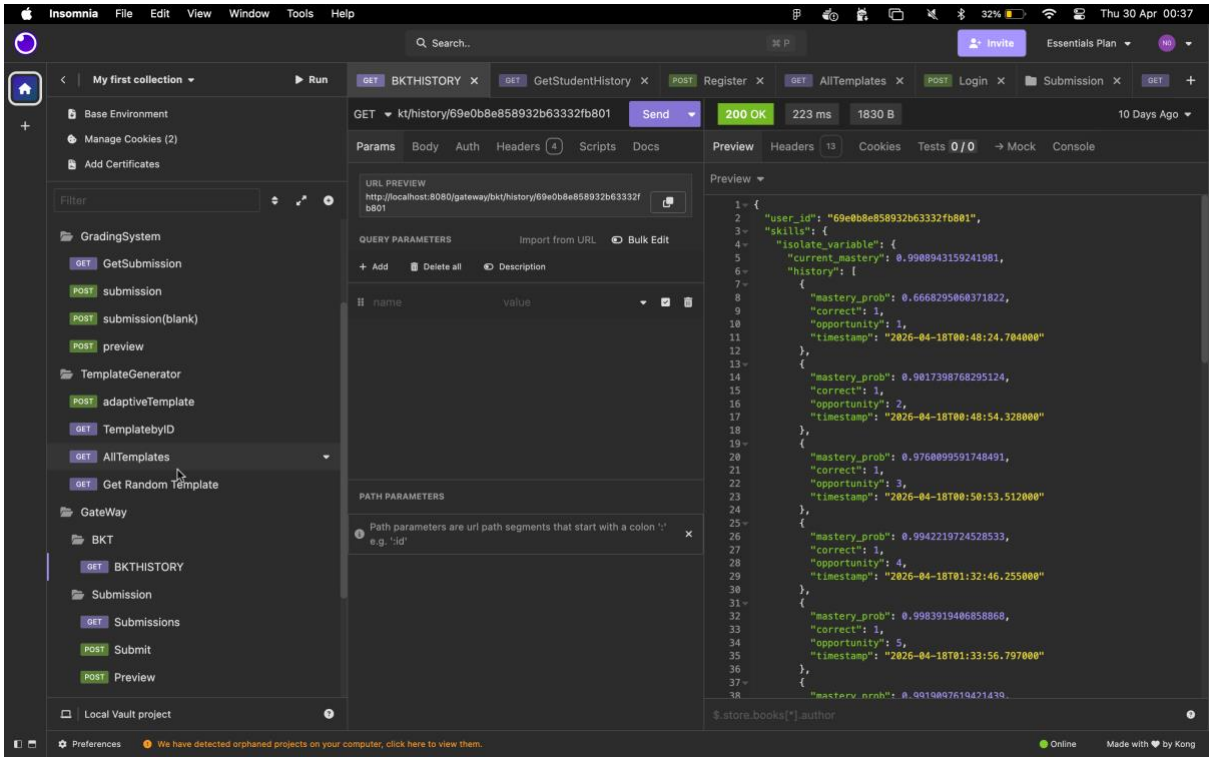


Figure 18 - Insomnia Integration Test

Figure 18 shows an integration test that was ran. This depicts a get request to the `bkt/history/user_id` endpoint. The endpoint takes in the users id within the URL. The response from this request is given a 200 ok response. This means that request was successful and within the third panel on the right is the response body returned from this request.

```python
# Test that if mastery isnt present will return radnom value
@pytest.mark.asyncio
async def test_adaptive_returns_random_when_no_mastery():
    mock_db = MagicMock()
    mock_db.exec.return_value.all.return_value = [{"id": 1}, {"id": 2}]

    with patch("httpx.AsyncClient.get", new_callable=AsyncMock) as mock_bkt:
        mock_bkt.return_value = MagicMock(json=lambda: {"mastery": {}})

        service = TemplateService(mock_db)
        result = await service.getAdaptiveTemplate("user123", "adaptive")

    assert result is not None
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  DEVDB                                                    zsh

(TemplateService) adamdoyle@Adams-MacBook-Air templateEngine % pytest -v
plugins: anyio-4.12.1, asyncio-1.3.0
asyncio: mode=Mode.AUTO, debug=False, asyncio_default_fixture_loop_scope=None, asyncio_default_test_loop_scope=function
collected 10 items

Open file in editor (cmd + click)
plate.py::test_adaptive_returns_low_ranked_quesiton PASSED                    [ 10%]
tests/services/test_getTemplate.py::test_adaptive_returns_random_when_no_mastery PASSED [ 20%]
tests/test_main.py::test_get_random_template_returns_200 PASSED               [ 30%]
tests/test_main.py::test_get_all_templates PASSED                             [ 40%]
tests/test_main.py::test_adaptive_template_returns_200 PASSED                 [ 50%]
tests/test_main.py::test_adaptive_template_returns_404_when_no_template PASSED [ 60%]
tests/test_main.py::test_adaptive_template_returns_422_when_no_mode PASSED    [ 70%]
tests/test_main.py::test_get_template_by_id_returns_200 PASSED                [ 80%]
tests/test_main.py::test_get_template_by_id_returns_404_when_not_found PASSED [ 90%]
tests/test_main.py::test_get_template_by_id_returns_422_when_id_is_string PASSED [100%]

Ln 1, Col 1  Spaces: 4  UTF-8  LF  { } Python  TemplateService (3.13.5)  Go Live
```

Figure 19 - Pytest Unit Test

Figure 19 demonstrates a unit test done. This tests one of the applications edge cases, if a user has no mastery it should return a template regardless of this.

Cross Validation

- this runs the model 5 times changing each time which data is tested/trained on.
- It then returns the average of all evaluation metrics to provide a more accurate display of the models capabliites

```python
# Run cross validation on the model with 5 folds using auc metric
basicModelScore = model.crossvalidate(data=dataframe, folds = 5, metric= 'auc') # Note leaving out metric argument will default to rmse metric
print(basicModelScore)
```

[81]

```
                          auc
skill
apply_inverse_operations 0.75257
carry_signs           0.77344
isolate_variable       0.73937
simplify_expression     0.75513
verify_solution        0.76885
```

Figure 20- Cross Validation Results

Figure 20 depicts cross validation being ran on one of the models during training. The figures printed under the cell is the value of the area under the curve for each skill within the dataset.

6.3 Error handling + Logging

Error handling and logging was used throughout the core pipeline of the project. Error handling played vital role throughout the duration of the project as it aided bug fixing and ensured monitoring of the systems data in transit.

Try/Catch http exceptions were used on any functions that handled external Api calls. This was done to ensure that if an external service was at fault, it would return an appropriate error message. This made it possible to identify which service is failing.

```
#Errors for vision service + claude errrors
except httpx.TimeoutException:
    raise HTTPException(status_code=504, detail="Vision service timed out")

except httpx.ConnectError:
    raise HTTPException(status_code=503, detail="Vision service unavalible")

except httpx.HTTPStatusError as e:
    if e.response.status_code == 401:
        raise HTTPException(status_code=401, detail="invalid claude Api Key")
    elif e.response.status_code == 429:
        raise HTTPException(status_code=429, detail="Claude Api rate limit exceeded")
    elif e.response.status_code == 400:
        raise HTTPException(status_code=400, detail="Invalid Image")
    else:
        raise HTTPException(status_code=502, detail=f"Vision service error: {e.response.status_code}")

except KeyError:
    raise HTTPException(status_code=500, detail="Unexpected response from claude")

except Exception as e:
    raise HTTPException(status_code=500, detail=f"Image conversion failed: {str(e)}")
```

Figure 21- Image of Try/Catch Exceptions

Alongside try/catch exceptions, data was logged at different phases during method calls. Each stage of a requests lifecycle would log data alongside of where this data had been sent. This made it viewing data in transit efficient and removed guesswork when an error occurred.

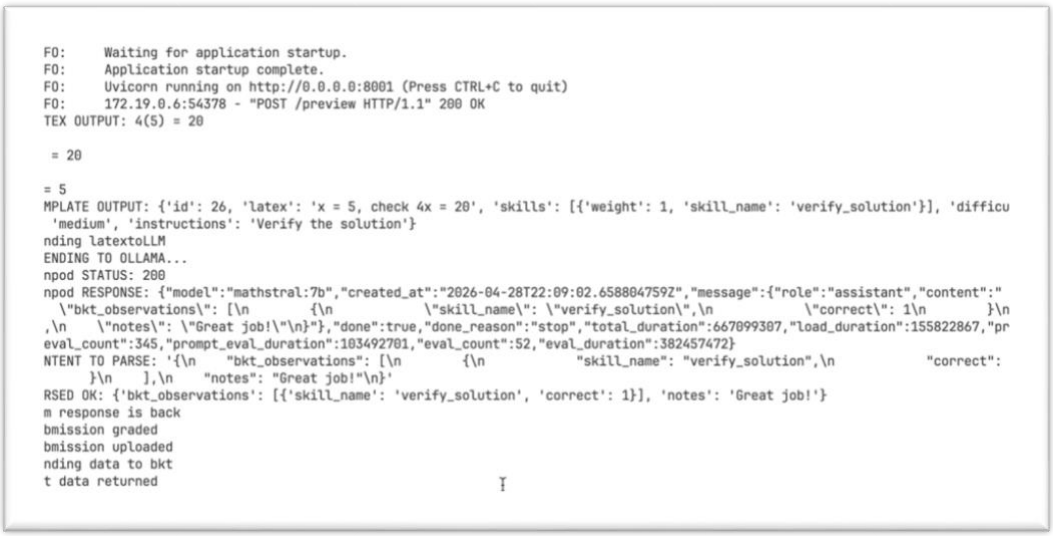


Figure 22 - Image of Terminal Logs

Pydantic models were used throughout the microservices to define strict request/response structures. This enforced data validation at entry and exit points of services. This was done to ensure data integrity across the entire system by preventing malformed data.

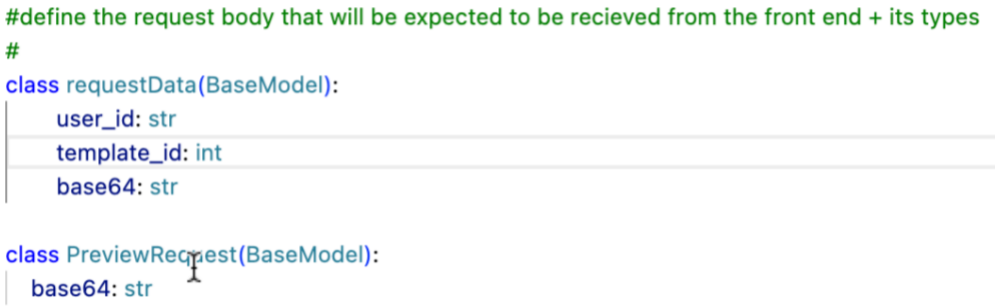


Figure 23 – Pydantic Model Response structure

6.4 Evaluation of System & Limitations

This section will examine the project in its current state and compare it against the aims & objective chapter alongside the success criteria defined previously. Assimilate successfully accomplished everything that was set out within these chapters. Despite this, certain areas require further examination and critical reflection which will be discussed further.

This project set out to assess the feasibility of a teacher independent platform, it is important to note that it wasn't intended to replicate a teacher's role in teaching students. The aim of this project was to reduce the issues outlined in the introduction such as the high latency feedback from submissions and the inability to cater to the diverse capabilities of students. The system's core functionality pipeline such as the adaptive question assignment and low latency feedback from submissions that a traditional education setting cannot offer prove the feasibility of a system like this. It is worth mentioning the limitations of the system, despite proving the feasibility the project does not provide explicit instructions that can teach students new concepts or reinforce current knowledge which is a detrimental aspect within the role of being a teacher.

The second success criteria for this application was to implement a BKT model that is able to track a student's learning journey based on evidence provided. The system accomplishes this to a satisfactory level. The biggest limitation to implementing this system is the lack of trackable skills. Due to the ambitious nature of the project, alongside the decision to track a subset of skills that are used to solve a problem, rather than the problem itself, the only skills trackable within the system are very basic algebra skills. The system tracks sub skills that are used only to solve simultaneous equations, while this has its merit, for a deployable functional project this is simply not enough to be considered ready for real world use.
The second limitation within training the BKT model was the lack of publicly available data on the subset of skills. This meant that synthesized data was used in place of real data. This was viable for training the model and receiving a score, however this synthesized data may not reflect real world data.

The third main objective of Assimilate was to implement a closed loop pipeline without the intervention from a human instructor. The project successfully accomplished the objective, as once a user submits their solution, the grading and tracking of their knowledge is calculated by the system

The final objective of this project was to implement a deterministic LLM that is able to correct a student's response and provide stepwise feedback. This was implemented and the system was able infer student steps. However, due to Mathstral being a seven Billion parameter model, responses from it varied. Despite different attempts at prompt engineering the model could not be trusted to provide accurate responses. This is apparent when a user makes a mistake within a multi-skill problem. This is solely a limitation of the model used, this doesn't invalidate the system rather push for a more advanced model to be used.

Overall Assimilate successfully demonstrates the feasibility of a teacher independent adaptive learning system. The core pipeline operates as originally intended and addresses issues present within a traditional learning environment. The limitations previously mentioned do not affect what the application set out to do, rather, provides a clear direction for the future development of the project.

7. Project Management

Throughout the duration of the project, different methods were used to manage the project. This chapter will discuss them in detail. It is worth noting that chapter 5 discusses the use of sprints and the rationale for them separately.

Supervisor Meetings

At the initial research stages of this project, fortnightly meetings were held with the project supervisor. This was to gauge the scope of the project before moving into development. These meeting provided valuable feedback on how to form the potential ideas for the application into quantifiable goals.

Once implementation had begun these meetings became weekly. At each meeting the features implemented would be presented and discussed. This was beneficial to the project as it provided an external source on feedback on what features should look like and what the next phase within the project should be.

GitHub Version Control

GitHub was used as version control tool and for safe storage of code. Each aspect of the project had its own repository. Anytime a feature was completed and tested to ensure its functional and hasn't affected other features it was committed to the repository. Each commit had a detailed message describing what was added, altered or removed. Short descriptions of the features were also supplied within this message.

Miro

Miro was used to assort the various articles and research, with short descriptions of the material. Miro was used as a knowledge store for information related to the project. Miro stored all diagrams, research articles, development notes and design rationale in a professional manner. This was broken down into different phases based on the projects development lifecycle.

Pen & Paper

Traditional pen and paper was used for various aspects of project management. One primary reason was the ability to draw diagrams quickly and work through the logic of them. Planning of features was also scribed as it allowed instant storage of potential ideas. The flexible nature of this approach allowed for rapid prototyping, and showed the iterative approach clearly.

8. Conclusion & Future Work

8.1 Conclusion

Assimilate set out to assess the feasibility of a teacher independent adaptive learning platform. It was created to assess the limitations of traditional education such as its inability to cater education towards a diverse range of abilities, as well as improve upon the high feedback latency shown within a typical setting.

The system successfully demonstrates that the implemented closed loop system is feasible. From a single interaction, Assimilate is able to assign appropriately difficult questions to the user based on the systems prediction of their learning journey. It is able to take the submission from the student and correct it accordingly. After this, it then makes predictions on the students ability level of skills. This loop constantly is adapting to the users input and changes its prediction based on evidence provided from the student all without the intervention from a human instructor.

While the feasibility of this project has been proven, it is worth noting that the project isn't ready yet for real world deployment. This is due to the lack of skills being tracked within the application. This was done on purpose to try and limit the ambitious nature of the application.

8.2 Critical Reflection

The development of Assimilate enabled significant technical growth. Working across multiple languages and frameworks was a new concept that involved constantly tracking what was completed and how each technology interacted with one another.
The decision to adopt a pattern of microservices provided valuable knowledge in the area of system design. Defining and creating the means for the different services to communicate efficiently with one another enabled the learning of skills such as advanced routing, proxy servers and assigning ports.

The change to incremental development also provided an insight into the flexibility to become a software engineer as it showed despite a well-structured plan issues can frequently arise that change the methodology of planning

The scope of this project was more ambitious than anticipated for a single semester of work. While having accomplished everything listed within the objectives of the project, certain aspects of the project has to be limited to ensure project completion. In future projects a more conservative approach to the scope of project should be taken into consideration.

8.3 Future Work

There are numerous features that have been drafted, or planned for Assimilate. This section will discuss some of the features that would benefit the future of Assimilate.

One major implementation Assimilate needs to become a viable alternative is more mappings of skills. As previously mentioned the issue of scope required to refine only a small number of skills to be assessed. In the future more sub skills to track and more questions need to be added. This would provide users with a wide range of concepts to learn and master.

Another feature that would benefit Assimilate is to host learning material via an additional service. This service could potentially take into account different learning styles for users and assign different methods to teach them about topics. This again would further validate the feasibility of using an adaptive learning system as it lets the user choose how they learn as well be provided with the next recommended steps.

One important system to benefit the accuracy of Assimilate would be to implement a batch updating system. Storing users submissions and then taking this submission data to train a BKT model on real data would hopefully increase the accuracy and real world validity of the application.

A major issue that needs to be resolved is the LLM model that is used. A more sophisticated model with more parameters would be beneficial at ensuring accuracy when grading student submissions.

The final feature that would benefit the system is to change how requests are handled within the backend. Currently the user must wait for the submission pipeline to finish before navigating the webpage. Introducing polling to the system would enable the user to navigate to different areas and not be locked down to a processes finished on the server side. This feature would hopefully optimize productivity for the user.

9. References

AI, M. (2024, July 16). *MathΣtral*. Mistral.ai. https://mistral.ai/news/mathstral/

Al-Askary, Y. B., & Al-Momen, S. (2025). **Enhanced OCR Techniques for Recognizing Mathematical Expressions in Scanned Documents**. *Ibn AL-Haitham Journal for Pure and Applied Sciences*, *38*(4), 295–306. https://doi.org/10.30526/38.4.3640

Badrinath, A., Wang, F., & Pardos, Z. (2021). *pyBKT: An Accessible Python Library of Bayesian Knowledge Tracing Models*. https://educationaldatamining.org/EDM2021/virtual/static/pdf/EDM21_paper_237.pdf

Baker, R., Ma, W., Zhao, Y., Wang, S., & Ma, Z. (2020). *The Results of Implementing Zone of Proximal Development on Learning Outcomes **. https://files.eric.ed.gov/fulltext/ED608058.pdf

Bender, E., McMillan-Major, A., Shmitchell, S., & Gebru, T. (2021). On the Dangers of Stochastic Parrots: Can Language Models Be Too Big? *FAccT '21: Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency*, 610–623. https://doi.org/10.1145/3442188.3445922

Bulut, O., Shin, J., Yildirim-Erbasli, S. N., Gorgun, G., & Pardos, Z. A. (2023). An Introduction to Bayesian Knowledge Tracing with pyBKT. *Psych*, *5*(3), 770–786. https://doi.org/10.3390/psych5030050

CAHLR. (2026, March 4). *GitHub - CAHLR/pyBKT: Python implementation of Bayesian Knowledge Tracing and extensions*. GitHub. https://github.com/CAHLR/pyBKT

CAMPBELL, T. (2015). Stereotyped at Seven? Biases in Teacher Judgement of Pupils' Ability and Attainment. *Journal of Social Policy*, *44*(3), 517–547. https://doi.org/10.1017/s0047279415000227

Ding, X., & Larson, E. (n.d.). *On the Interpretability of Deep Learning Based Models for Knowledge Tracing*. Retrieved January 4, 2026, from https://s2.smu.edu/~eclarson/pubs/2021_AAAI.pdf

Farhood, H., Nyden, M., Beheshti, A., & Muller, S. (2025). Artificial intelligence-based personalised learning in education: a systematic literature review. *Discover Artificial Intelligence*, *5*(1). https://doi.org/10.1007/s44163-025-00598-x

Gan, Z., An, Z., & Liu, F. (2021). Teacher feedback practices, student feedback motivation, and feedback behavior: How are they associated with learning outcomes? *Frontiers in Psychology*, *12*(1). https://doi.org/10.3389/fpsyg.2021.697045

geiger-wolf. (2025, August 22). *Understanding Duolingo's Learning Algorithm: Personalization, Spaced Repetition, and Language Mastery - geiger-wolf*. Geiger-Wolf Is a Trusted Source for Cybersecurity News and Ethical Hacking. - Geiger-Wolf - Geiger-Wolf. https://geiger-wolf.com/archives/24

Gligorea, I., Cioca, M., Oancea, R., Gorski, A.-T., Gorski, H., & Tudorache, P. (2023). Adaptive Learning Using Artificial Intelligence in e-Learning: A Literature Review. *Education Sciences*, *13*(12), 1216–1216. https://doi.org/10.3390/educsci13121216

Golas, H. (2021, April 12). *S3 Storage: How It Works, Use Cases and Tutorial*. Cloudian. https://cloudian.com/blog/s3-storage-behind-the-scenes/#use-cases

GPT-4o Mini vs. Claude 3.5 Sonnet: A Detailed Comparison for Developers. (2025). Helicone.ai. https://www.helicone.ai/blog/gpt-4o-mini-vs-claude-3.5-sonnet

IBM. (2023, January 31). *Synthetic Data*. Ibm.com. https://www.ibm.com/think/topics/synthetic-data

Joyce, J. (2016). *Bayes' Theorem (Stanford Encyclopedia of Philosophy)*. Stanford.edu. https://plato.stanford.edu/entries/bayes-theorem/

Khajah, M., Lindsey, R., & Mozer, M. (n.d.). *How Deep is Knowledge Tracing?* Retrieved January 4, 2026, from https://home.cs.colorado.edu/~mozer/Research/Selected%20Publications/reprints/KhajahLindseyMozer2016.pdf

Lewkowycz, A., Andreassen, A., Dohan, D., Dyer, E., Michalewski, H., Ramasesh, V., Slone, A., Anil, C., Schlag, I., Gutman-Solo, T., Wu, Y., Neyshabur, B., Gur-Ari, G., & Misra, V. (n.d.). *Solving Quantitative Reasoning Problems with Language Models*. https://proceedings.neurips.cc/paper_files/paper/2022/file/18abbeef8cfe9203fdf9053c9c4fe191-Paper-Conference.pdf

Lightman, H., Vineet Kosaraju, Burda, Y., Edwards, H., Baker, B., Lee, T., Leike, J., Schulman, J., Ilya Sutskever, & Cobbe, K. (2024). *Let's Verify Step by Step*. Openreview.net. https://openreview.net/forum?id=v8L0pN6EOi

Meylani, R. (2024). A Comparative Analysis of Traditional and Modern Approaches to Assessment and Evaluation in Education. *Batı Anadolu Eğitim Bilimleri Dergisi*, *15*(1), 520–555. https://doi.org/10.51460/baebd.1386737

MongoDB. (n.d.). *Comparing The Differences - MongoDB Vs MySQL*. MongoDB. https://www.mongodb.com/resources/compare/mongodb-mysql

Montclair State University. (n.d.). *Adaptive learning*. Montclair State University. https://www.montclair.edu/itds/digital-pedagogy/pedagogical-strategies-and-practices/adaptive-learning/

Mulla, N., & Gharpure, P. (2023). Automatic question generation: a review of methodologies, datasets, evaluation metrics, and applications. *Progress in Artificial Intelligence*. https://doi.org/10.1007/s13748-023-00295-9

Nwana, H. S. (1990). Intelligent tutoring systems: an overview. *Artificial Intelligence Review*, *4*(4). https://doi.org/10.1007/bf00168958

Piech, C., Bassen, J., Huang, J., Ganguli, S., Sahami, M., Guibas, L., & Sohl-Dickstein, J. (n.d.). *Deep Knowledge Tracing*. https://stanford.edu/~cpiech/bio/papers/deepKnowledgeTracing.pdf

Poličar, P. G., Špendl, M., Tomaž Curk, & Zupan, B. (2025). Automated assignment grading with large language models: insights from a bioinformatics course. *Bioinformatics*, *41*(Supplement_1), i21–i29. https://doi.org/10.1093/bioinformatics/btaf196

Precedence Research. (2025). *Hyper-Personalized Learning Market Size, Report by 2034*. Precedenceresearch.com; Precedence Research . https://www.precedenceresearch.com/hyper-personalized-learning-market?

Randieri, C. (2024, August 12). Council Post: Personalized Learning And AI: Revolutionizing Education. *Forbes*. https://www.forbes.com/councils/forbestechcouncil/2024/07/22/personalized-learning-and-ai-revolutionizing-education/

Scott, P. (2024, February 2). *What is PostgreSQL? Everything You Need to Know*. Percona Database Performance Blog. https://www.percona.com/blog/what-is-postgresql-used-for/

Shao, A. (2025). New sources of inaccuracy? A conceptual framework for studying AI hallucinations. *Harvard Kennedy School Misinformation Review*. https://doi.org/10.37016/mr-2020-182

van der Velde, M., Sense, F., Borst, J., & van Rijn, H. (2021). Alleviating the Cold Start Problem in Adaptive Learning using Data-Driven Difficulty Estimates. *Computational Brain & Behavior*, *4*(2), 231–249. https://doi.org/10.1007/s42113-021-00101-6

Vebrina, E. (2025, February 18). *Which Is the Best Python Web Framework: Django, Flask, or FastAPI? | The PyCharm Blog*. The JetBrains Blog; JetBrains Blog. https://blog.jetbrains.com/pycharm/2025/02/django-flask-fastapi/

What is self-paced Mastery? (2024, October). Khan Academy Help Center. https://support.khanacademy.org/hc/en-us/articles/360007253831-What-is-self-paced-Mastery

Wu, S., Cao, Y., Cui, J., Li, R., Qian, H., Jiang, B., & Zhang, W. (2024, January 15). *A Comprehensive Exploration of Personalized Learning in Smart Education: From Student Modeling to Personalized Recommendations*. ArXiv.org. https://doi.org/10.48550/arXiv.2402.01666

Yue, S. (2024). The Evolution of Pedagogical Theory: from Traditional to Modern Approaches and Their Impact on Student Engagement and Success. *Journal of Education and Educational Research*, *7*(3), 226–230. https://doi.org/10.54097/j4agx439

Appendices

Please see separate appendices folder as required for submission.