INTERNATIONAL HELLENIC UNIVERSITY

SCHOOL OF ENGINEERING

DEPARTMENT OF INFORMATION AND ELECTRONIC ENGINEERING

THESIS

Development of an Intelligent Platform for Lifelong Learning with Adaptive Review and AI Assistance

Student
Theologos Bourloglou
St. Number: 2020107

Supervisor
Dr. Periklis Chatzimisios
Professor

May 2026

Thesis Title: Development of an Intelligent Platform for Lifelong Learning with Adaptive Review and AI Assistance

Thesis Number: 25174

Student Full Name: Theologos Bourloglou

Supervisor Full Name: Periklis Chatzimisios

Date of Thesis Undertaking the Thesis: 23 March 2025

Date of Thesis Completion: 31 May 2026

I certify that I am the author of this work and that any assistance I received in its preparation is fully acknowledged and referred to in the work. I have also listed any sources from which I used data, ideas, images and text, whether they are cited exactly or paraphrased. Furthermore, I certify that this work was prepared by me personally, specifically as a diploma thesis, at the Department of Information and Electronic Engineering of the International Hellenic University.

«To my mother»

Abstract

In the modern digital era, traditional e-learning platforms frequently suffer from low student completion rates and poor long-term knowledge retention due to an over-reliance on passive, static content delivery mechanisms. To address these pedagogical inefficiencies, this thesis presents the comprehensive design, development, and verification of an intelligent, research-informed lifelong learning platform engineered to dynamically personalize the educational experience. The system is built using a modern full-stack web application model featuring PHP 8 and Laravel 12 on the backend, Vue 3 on the frontend, and Inertia.js serving as a server-driven Single-Page Application (SPA) bridge to deliver high client-side interactivity without the architectural duplication of a separate REST API (Representational State Transfer Application Programming Interface). Grounded in cognitive psychology and motivational science, the platform successfully integrates several advanced learning modules: a spaced repetition system utilizing a geometric doubling interval for long-term memory consolidation, an expansive gamification layer (awarding Experience Points, daily tracking streaks, and milestone badges) to satisfy intrinsic psychological needs for competence and persistence, and an adaptive "Weakness Map" analytical layer. This adaptive tool aggregates localized temporal failure signals to prioritize individual knowledge gaps and generate direct, recommended recovery actions such as lesson re-reads or quiz retries. Orchestrated strictly server-side via the Laravel AI SDK (Software Development Kit) to enforce verification and data safety, the platform leverages OpenAI's GPT-4o model using structured outputs to empower both creators and students. Content creators benefit from automated lesson quality analysis alongside algorithmic quiz and flashcard generation, while learners receive real-time, scaffolded support through contextual quiz hints and detailed post-failure conceptual explanations. Furthermore, a verifiable certification workflow automatically creates portable, print-ready digital credentials mapped to unique, public links, facilitating external verification while keeping unshared records private. Finally, validated through a robust suite of automated PHPUnit feature tests to guarantee statistical and transactional integrity, this platform effectively demonstrates how converging observable behavioral metrics with generative artificial intelligence transitions digital education into an active, adaptive, and highly accountable learning paradigm.

Ανάπτυξη Ευφυούς Πλατφόρμας για Δια Βίου Μάθηση με Προσαρμοστική Αναθεώρηση και Υποστήριξη Τεχνητής Νοημοσύνης

Μπουρλόγλου Θεολόγος

Περίληψη

Στη σύγχρονη ψηφιακή εποχή, οι παραδοσιακές πλατφόρμες ηλεκτρονικής μάθησης συχνά υποφέρουν από χαμηλά ποσοστά ολοκλήρωσης των μαθημάτων από τους εκπαιδευόμενους και ασθενή μακροπρόθεσμη συγκράτηση της γνώσης, λόγω της υπερβολικής εξάρτησης από παθητικούς, στατικούς μηχανισμούς παράδοσης περιεχομένου. Για την αντιμετώπιση αυτών των παιδαγωγικών ανεπαρκειών, η παρούσα διπλωματική εργασία παρουσιάζει τον ολοκληρωμένο σχεδιασμό, την ανάπτυξη και την επαλήθευση μιας ευφυούς, ερευνητικά τεκμηριωμένης πλατφόρμας δια βίου μάθησης, η οποία έχει σχεδιαστεί για να εξατομικεύει δυναμικά την εκπαιδευτική εμπειρία. Το σύστημα είναι κατασκευασμένο με βάση ένα σύγχρονο μοντέλο διαδικτυακής εφαρμογής πλήρους στοίβας (full-stack) που διαθέτει PHP 8 και Laravel 12 στο backend, Vue 3 στο frontend, και το Inertia.js να λειτουργεί ως μια καθοδηγούμενη από τον διακομιστή γέφυρα εφαρμογής μιας σελίδας (SPA) για την παροχή υψηλής διαδραστικότητας στην πλευρά του χρήστη, χωρίς την αρχιτεκτονική επικάλυψη ενός ξεχωριστού REST API (Representational State Transfer Application Programming Interface). Βασισμένη στη γνωστική ψυχολογία και την επιστήμη των κινήτρων, η πλατφόρμα ενσωματώνει με επιτυχία αρκετές προηγμένες ενότητες μάθησης: ένα σύστημα επανάληψης σε διαστήματα που χρησιμοποιεί ένα γεωμετρικό διάστημα διπλασιασμού για την ενίσχυση της μακροπρόθεσμης μνήμης, ένα εκτεταμένο επίπεδο παιχνιδοποίησης (απονομή πόντων εμπειρίας, καθημερινών σερί παρακολούθησης και σημάτων ορόσημων) για την ικανοποίηση εγγενών ψυχολογικών αναγκών για ικανότητα και επιμονή, και ένα προσαρμοστικό αναλυτικό επίπεδο υπό τη μορφή ενός «Χάρτη Αδυναμιών». Αυτό το προσαρμοστικό εργαλείο συγκεντρώνει τοπικά χρονικά σήματα αποτυχίας για να ιεραρχήσει τα ατομικά κενά γνώσης και να προτείνει άμεσες, συνιστώμενες ενέργειες αποκατάστασης, όπως η εκ νέου μελέτη μαθημάτων ή η επανάληψη κουίζ. Ενορχηστρωμένη αυστηρά από την πλευρά του διακομιστή μέσω του Laravel AI SDK(Artificial Intelligence Software Development Kit) για την επιβολή της επαλήθευσης και της ασφάλειας των δεδομένων, η πλατφόρμα αξιοποιεί το μοντέλο GPT-4o της OpenAI χρησιμοποιώντας δομημένες εξόδους για να ενδυναμώσει τόσο τους δημιουργούς όσο και τους μαθητές. Οι δημιουργοί περιεχομένου επωφελούνται από την αυτοματοποιημένη ανάλυση ποιότητας μαθήματος παράλληλα με τη δημιουργία αλγοριθμικών κουίζ και καρτών εξάσκησης (flashcards), ενώ οι μαθητές λαμβάνουν υποστήριξη σε πραγματικό χρόνο μέσω σχετικών με το πλαίσιο υποδείξεων κουίζ και λεπτομερών εξηγήσεων μετά από αποτυχημένες προσπάθειες. Επιπλέον, μια επαληθεύσιμη ροή εργασίας πιστοποίησης δημιουργεί αυτόματα φορητά, έτοιμα για εκτύπωση ψηφιακά διαπιστευτήρια που αντιστοιχίζονται σε μοναδικούς, δημόσιους συνδέσμους, διευκολύνοντας την εξωτερική επαλήθευση, διατηρώντας παράλληλα τα μη κοινόχρηστα αρχεία ιδιωτικά. Τέλος, αυτή η πλατφόρμα καταδεικνύει αποτελεσματικά πώς η σύγκλιση παρατηρήσιμων μετρήσεων συμπεριφοράς με την παραγωγική τεχνητή νοημοσύνη μετατρέπει την ψηφιακή εκπαίδευση σε ένα ενεργό, προσαρμοστικό και εξαιρετικά αξιόπιστο μαθησιακό παράδειγμα.

Table of Contents

List of Figures

Chapter 1: Introduction

1.1 Introduction

These days, e-learning and digital education have completely transformed the way people think about learning. In the past, learning was something that mostly happened inside a physical classroom, during a specific time of a person's life, like when they were children or university students. Today, this old model is changing because of the fast pace of modern technology, shifting job markets, and the continuous evolution of professional industries. Because of this, learning cannot stop after graduation. People now need to acquire new skills and refresh their knowledge throughout their entire lives, a concept widely known as lifelong learning. Digital web platforms have become the primary tools to help people achieve this because they remove the major limitations of traditional schooling. With a simple internet connection, a computer, or a mobile phone, anyone can study from any location in the world at any hour of the day. E-learning platforms have made education highly flexible and accessible to global audiences, offering low-cost alternatives to expensive university degrees.

However, even though online learning has grown incredibly fast and is now highly popular, modern web-based platforms still suffer from deep structural and pedagogical issues. The most significant problem is that a massive number of existing educational websites operate purely as static content delivery players. They behave like digital filing cabinets where teachers just upload text documents, presentation slides, or pre-recorded video lectures. After the files are uploaded, the system presents the exact same learning path, the exact same order of materials, and the same tests to every single student. This one-size-fits-all approach completely ignores the natural differences between individual human beings. Every person who visits an online course comes with a completely unique background, a different learning speed, and specific areas where they understand things quickly or slowly. When a platform ignores these differences and relies heavily on passive reading or watching, students lose their focus, get bored, and feel disconnected. This widespread lack of interactive personalization and immediate feedback is the primary reason why digital courses suffer from huge drop-out rates. For instance, historical data shows that popular massive open online courses regularly have full completion rates below 10%.

Another critical challenge in current e-learning models is the severe lack of long-term knowledge retention. Most students interact with digital courses by reading the text just once and then immediately taking a quiz or an exam, an ineffective studying method commonly called "cramming." Cognitive science research proves that human memory decays in a very sharp, predictable curve right after a person learns a new concept. If a student does not actively recall or practice that information at mathematically optimized times, they will quickly forget a huge percentage of it within a few days. Traditional digital tools do absolutely nothing to track these memory gaps or to help the user practice effortful retrieval.

To fix these serious digital teaching inefficiencies, online education must transition away from static content repositories and move toward truly intelligent, adaptive learning environments. We need to build web systems that can actively observe student performance, listen to behavioral signals, automatically discover exactly where a user is struggling, and utilize generative artificial intelligence to offer customized help in real time. By combining modern web software architectures with proven cognitive psychology and game mechanics, digital platforms can become highly interactive spaces that make lifelong learning deeply engaging, efficient, and successful for long-term memory consolidation.

1.2 Research Contributions and Platform Overview

To directly solve the engagement and memory retention issues of traditional online education, this thesis describes the full design, implementation, and code architecture of an intelligent full-stack web application built specifically for lifelong learning, adaptive review and AI assistance. The main goal of this project was to move completely past passive text presentation and create a high-performance system that adapts directly to the learner's actions. During the preparation of this thesis, theories across computer programming, memory psychology, human motivation, and generative artificial intelligence were deeply studied to assemble a cohesive development platform.

The completed platform features a distinct two-role perspective for content creators and students, and it successfully introduces five core pillars that work together to personalize the digital educational journey:

- **Spaced Repetition System (SRS)**: Grounded in the cognitive science of human memory and the spacing effect, the Spaced Repetition System (SRS) features an automated review queue powered by a geometric doubling interval algorithm. When a student marks a flashcard as correct, the system multiplies the previous review interval by two. Conversely, if a flashcard is marked as incorrect, the interval immediately resets to one day, requiring the student to promptly re-study the forgotten fact.
- **Adaptive Analytics via the Weakness Map**: The platform utilizes a dedicated background engine that aggregates localized "failure signals" from the student's database history. It specifically tracks failed quiz attempts and incorrect review marks where a student swiped a flashcard to the left. The system uses this raw data to calculate a dynamic Weakness Score for every lesson, ranking the most problematic topics at the top of a visual map and providing one-click recommended recovery actions, such as re-reading the source text or retrying the specific quiz.
- **Generative Artificial Intelligence Assistance**: OpenAI’s GPT-4o and gpt-4.1-mini models are securely integrated into the backend using the Laravel AI SDK. To ensure data safety, all AI operations are handled on the server side using precise JSON schemas to generate structured outputs. For creators, the AI analyzes lesson text to provide quality feedback or automatically generates draft quizzes and flashcards to reduce authoring time. For students, it functions as a real-time tutor, providing smart contextual hints during quizzes and custom conceptual explanations for incorrect answers.
- **Gamification Layer**: To drive daily engagement and support the formation of lasting study habits, the platform incorporates a gamified tier designed around human motivation psychology. Students accumulate specific Experience Points (XPs): earning 50 XPs for finishing a lesson, 100 XPs for passing a quiz, 20 XPs for a failed quiz attempt to reward effort, and 10 XPs for completing a flashcard review session. The system also tracks daily activity streaks and automatically evaluates eligibility for 12 distinct milestone badges across consistency, effort, and completion.
- **Verifiable Certification Workflow**: To provide genuine career value to adult continuous learners, the platform implements an automated completion workflow. The moment a student finishes all lessons in a course, the server creates a permanent certificate record with a unique identifier mapped to a public, unauthenticated link. Anyone holding the URL can verify the credential instantly. The integration of custom CSS print media queries makes the certificate instantly print-ready or downloadable as a clean PDF without server-side file overhead.

From a technical perspective, the architecture was designed to deliver this feature-rich environment while avoiding a complex, fragmented system. A highly productive, full-stack, single-codebase web application model was selected. The system utilizes PHP 8 and Laravel 12 for primary backend services, including Eloquent object-relational mapping, validation requests, and authorization policies.

The frontend leverages Vue 3 alongside Tailwind CSS 4 for rapid styling, Vite 7 for compilation, and specialized client libraries such as Unovis for charts and motion-v for smooth flashcard swipe gestures. Crucially, Inertia.js serves as the architectural bridge, enabling a server-driven Single-Page Application (SPA) where users move seamlessly between screens without full browser reloads, thereby keeping routing and access control securely within Laravel. Finally, absolute system correctness is verified through an automated suite of HTTP feature and unit tests developed with PHPUnit 11 to validate everything from quiz score calculations to transactional data safety.

1.3 Structure of the Thesis

The text of this diploma thesis is organized into six comprehensive chapters that systematically explain the project from the underlying academic concepts down to the implemented database logic and code execution. The contents of the remaining chapters are summarized below:

- **Chapter 2: Theoretical Background** – This section presents the academic literature and cognitive research that forms the foundation of our application. It covers the historical evolution of e-learning platforms, the mathematics behind memory retention algorithms, motivational psychology inside gamification frameworks, natural language processing for question generation, and the practical needs of modern continuous adult learners.
- **Chapter 3: Technology Stack and Architectural Approach** – This chapter focuses on the architectural rationale behind our programming tools. It breaks down why the combination of Laravel 12, Inertia.js, and Vue 3 was selected, how the Laravel AI SDK manages prompt structures, and the implementation details of supporting tools for database persistence, code linting, and front-end interface charting.
- **Chapter 4: Core Platform Implementation** – This chapter walks through the concrete functional details of the application's core modules. It describes the user authentication system, the content authoring tools where creators organize lessons and plain-text flashcards, the marketplace discovery search bar, and the interactive swipe gesture code used during student flashcard reviews.
- **Chapter 5: Intelligent and Adaptive Features** – This section details the advanced, data-driven back-end systems of the platform. It explains how the server communicates asynchronously with OpenAI's APIs, how the analytics engine aggregates failure logs to compute live Weakness Scores, the event-driven programming behind the gamification rewards, and the structural design of our automated PHPUnit testing suite.
- **Chapter 6: Summary, Conclusions, and Future Extensions** – The final chapter brings the entire work to a close. It provides a full summary of the development outcomes, evaluates the platform's educational success, and lays out explicit plans for future versions, including shifting AI prompts to queued background workers and integrating multimodal data inputs.

Chapter 2: Theoretical Background

2.1 Introduction to the Theoretical Background

2.1.1 Purpose of this Chapter

This chapter presents the theoretical foundation on which the platform described in this thesis is built. A system that combines e-learning, adaptive feedback, spaced repetition, gamification, AI-generated content, and lifelong learning certification cannot be properly understood without first examining the research that justifies each of those design choices. The goal here is not simply to review literature for the sake of it, but to explain why each theoretical area is relevant and how it shapes the structure and behaviour of the platform.

The review draws on recent work in adaptive learning systems, cognitive science, educational technology, and AI-assisted content creation. The references selected are representative of both established findings and more recent developments, roughly from 2019 to 2025. Where topics have a longer research tradition, such as the psychology of memory and spaced repetition. The chapter traces enough of that background to make the platform's design decisions clear. For areas that are more recently developed, such as generative AI in education, the focus stays on current research [1],[2].

This chapter is organized to move from the broader context of e-learning toward increasingly specific topics: personalization algorithms, memory science, motivation through game mechanics, AI content tools, and finally lifelong learning and certification. Each section builds on the previous one, which reflects how the platform itself is structured.

2.1.2 How the Theory Connects to the Platform

The platform described in this thesis is a web-based learning environment built with Laravel, Inertia.js, and Vue.js. It supports two roles, creators and students, and offers a range of features including quiz generation, flashcard-based review sessions, gamification mechanics, a weakness map, and course completion certificates. Each of these features has a grounding in existing research.

The adaptive and AI-driven aspects of the platform connect directly to the literature on adaptive learning systems and intelligent tutoring [1],[3]. The spaced repetition system, which adjusts review intervals based on whether a learner answered correctly or not, reflects the memory science covered in sections 2.3 and 2.4. The gamification layer (Experience Points - XPs, badges, and streaks), draws from evidence on motivation and engagement in digital learning environments [4],[5]. AI-generated quizzes and flashcards are grounded in recent work on automated content creation [6]. Finally, the certification feature and the platform's general design philosophy reflect ideas from the lifelong learning literature [7].

The intention is that by the end of this chapter, the reader will understand not just what the platform does, but why it was designed the way it was.

2.2 E-Learning and Adaptive Learning Systems

2.2.1 What E-Learning is and how it has Evolved

E-learning refers broadly to learning that is facilitated through digital technologies, typically over the internet. It covers a wide range of formats, from static course pages and downloadable resources to

fully interactive platforms with live video, automated feedback, and personalized learning paths. The term has been in use since the late 1990s, but the field has changed significantly in the years since. Early e-learning systems were essentially digital versions of traditional course materials: the content was online, but the structure and delivery were largely the same as in a classroom [2].

Over time, two things changed that significantly. First, improvements in web technologies allowed platforms to become more interactive and responsive. Second, the availability of large amounts of learner data made it possible to study how people actually learn online and to design systems that respond to those patterns. This led to a shift from static course delivery toward more dynamic, data-driven approaches. Platforms stopped just presenting content and started tracking behavior, like which lessons were completed, how long learners spent on each section, and which questions they got wrong. They then started using that data to make decisions about what to show next [1],[2].

The scale of e-learning also increased dramatically. Massive Open Online Courses (MOOCs) brought university-level content to audiences of hundreds of thousands. At the same time, research showed that raw scale does not automatically lead to good outcomes. Completion rates for MOOCs remained persistently low, and the lack of personalized support was often cited as a key reason [7]. This is why the field gradually moved toward adaptive systems that try to account for individual learner differences rather than delivering the same experience to everyone.

2.2.2 Adaptive Learning: Personalizing the Experience

Adaptive learning systems adjust the content, pace, or difficulty of a course based on data collected about the learner. The basic idea is that different learners come with different backgrounds, learning speeds, and areas of weakness, and a one-size-fits-all curriculum cannot serve all of them equally well. An adaptive system tries to close that gap by responding to what it observes [1].

In practice, adaptation can happen at different levels. At the simplest level, a system might skip content that a learner already knows, based on a pre-assessment. At a more sophisticated level, it might continuously update a model of the learner's knowledge state and select the next piece of content to maximize learning efficiency. The underlying technique often involves some form of knowledge tracing: estimating how well a learner knows each concept and how likely they are to retain it over time [1],[8].

The main benefit of adaptive learning is that it reduces wasted time. Learners do not sit through material they already understand, and they do not progress past material they have not yet grasped. Research generally shows that adaptive learning improves both engagement and academic performance compared to non-adaptive alternatives [1],[8]. The challenges are real too, though. Building an accurate model of a learner's knowledge is not straightforward, and doing so requires collecting significant amounts of behavioral data, which raises privacy concerns [1],[8].

For the platform in this thesis, adaptation is implemented through the weakness map and the spaced repetition system. Rather than a full predictive model of learner knowledge, the platform uses observable failure signals like quiz failures and review errors, to surface which topics need attention and to schedule them for repeated practice.

2.2.3 AI in Adaptive Learning Platforms

Artificial intelligence has become a central tool in modern adaptive learning, primarily because it enables the kind of dynamic, data-driven decisions that manual rule-based systems struggle with at

scale. Machine learning algorithms can identify patterns in learner behavior that would be invisible to a human observer reviewing logs manually, and they can do this continuously and in real time [1].

There are several ways AI is applied in this context. Recommendation systems suggest what to study next based on past behavior and similarity to other learners. Natural language processing is used to analyze learner-written responses and provide feedback. Reinforcement learning has been explored as a way to optimize content sequencing over time by treating the instructional choice as a decision problem [1],[4]. More recently, large language models have been applied to generate learning materials automatically, a development that is particularly relevant to this thesis [6].

One important distinction is between AI that operates in the background (choosing what content to show) and AI that is visible to the learner (providing hints, explanations, or feedback). Both are present in the research literature, and both are used in this platform. The quiz hint system and the explanations provided after a failed question are examples of visible AI interaction. The weakness map and the review scheduling algorithm use behavioral signals in the background without requiring the learner to interact with an AI interface directly [2],[3].

A common concern raised in the literature is the risk of algorithmic bias. If a system is trained on data that reflects historical inequalities, for example, if certain types of learners have historically had less success on standardized assessments, the model may systematically disadvantage those same groups. Transparency and human oversight in AI-driven platforms are therefore important, especially in educational settings [8].

2.2.4 Examples and Comparison of Existing Platforms

Several commercially deployed platforms have been studied in the literature as examples of AI-driven adaptive learning. Carnegie Learning applies AI to mathematics education and uses a combination of cognitive modeling and hints to guide learners through problem-solving. DreamBox Learning focuses on primary and middle school mathematics, with a system that adapts in real time to how learners approach individual problems, not just whether they get the right answer. Smart Sparrow is a more general authoring environment that allows educators to design adaptive lessons, and Knewton (now part of Wiley) applies a recommendation layer on top of existing content from educational publishers [8].

Each of these platforms takes a different approach to adaptation and content delivery. What they share is a data-intensive design: they all collect detailed interaction data and use it to make instructional decisions. They also all face the same structural challenge: balancing the automation of instruction with the role of human educators. When a platform makes decisions about what a learner should study, it takes on a function that was previously the teacher's, and this creates questions about accountability and pedagogical oversight [8].

A system described in the literature called Skillify represents a different angle: it uses generative AI and a course recommendation engine to help learners find appropriate courses rather than just adapting content within a course [9]. This is closer to the approach taken in the platform in this thesis, where OpenAI's GPT-4o model is used to generate quizzes and flashcards from lesson content, and where the weakness map functions as a recommendation mechanism for where to focus study effort.

2.3 Spaced Repetition and Memory Theory

2.3.1 The Forgetting Curve and why Spacing Matters

The scientific study of memory and forgetting goes back to the late nineteenth century, when Hermann Ebbinghaus conducted systematic experiments on himself to measure how quickly learned material is forgotten. His results showed that memory decays in a predictable, non-linear way. Most forgetting happens relatively soon after learning, and the rate slows over time. This pattern is typically represented as the forgetting curve, which shows a steep initial decline followed by a gradual leveling off [10].

The implication for learning is significant. If a learner studies something once and does not return to it, they will have forgotten a large portion of it within a few days. Studying the same material multiple times, with gaps between sessions, slows down this forgetting process. The reason is that each retrieval attempt (each time the learner successfully recalls something) strengthens the memory trace and effectively resets the forgetting clock. This is the spacing effect, and it is one of the most consistently replicated findings in cognitive psychology [10],[11].

Spacing works in part because it forces the brain to work harder to retrieve information that has been partially forgotten. When material is reviewed immediately after learning, it is still fresh and retrieval is easy. When it is reviewed after an interval, the learner has to reconstruct the memory more actively, and this effortful retrieval is what produces stronger long-term retention. This is sometimes called the "desirable difficulty" principle. making learning slightly harder in the short term produces better outcomes over the long term [10].

The relevance for digital learning platforms is clear. A system that schedules review sessions at intervals calibrated to the forgetting curve can dramatically improve long-term retention compared to a system that simply lets learners review whenever they feel like it, or not at all.

2.3.2 How Spaced Repetition Algorithms Work

The earliest practical implementation of spaced repetition as a study method is the Leitner system, developed by Sebastian Leitner in the 1970s. In this system, flashcards are organized into groups (or boxes), and the review frequency of each card depends on which group it is in. A card answered correctly moves to the next group and is reviewed less frequently. A card answered incorrectly goes back to the first group and is reviewed more often. The system is simple to implement with physical cards and captures the core intuition of spacing [10].

The SM2 algorithm, developed by Piotr Wozniak in the 1980s and used in the software SuperMemo, formalized this approach with a mathematical model. SM2 assigns each card an "easiness factor" and an interval. After each review, both are updated based on whether the answer was correct and how confidently it was given. Correct answers extend the interval by a factor that depends on the easiness score, while wrong answers reset the interval. This was the algorithm that made spaced repetition practical as a software feature and influenced many tools that came after [10].

More recent developments in the field have moved toward data-driven approaches that go beyond the fixed formulas of SM2. Systems based on machine learning, including models that use long short-term memory (LSTM) networks and probabilistic approaches like the Half-Life Regression (HLR) model, attempt to learn individual forgetting rates from data rather than assuming they follow a universal curve [10]. Research published in PNAS framed optimal spaced repetition as a control problem, in

which the goal is to find a reviewing schedule that maximizes the probability of recall at a future test date. The study showed that for two well-known memory models, the optimal schedule outperforms fixed-interval heuristics [11].

The platform in this thesis uses a simpler but practical variant: a doubling interval. When a learner answers a flashcard correctly, the time until the next review is doubled. When they answer incorrectly, the interval resets to one day. This mirrors the basic logic of SM2 without the full parameter model, making it easy to implement and reason about while still delivering meaningful spacing benefits.

2.3.3 Spaced Repetition in Digital Learning Tools

Digital tools have made spaced repetition far more accessible than it was in the era of physical flashcard boxes. Applications like Anki, Duolingo, and Quizlet all incorporate some form of spaced review, though they vary in how closely they follow formal memory models and how transparent they are with learners about the scheduling logic [12],[13].

One challenge in digital spaced repetition systems is adoption. Research shows that when learners are given access to a spaced repetition tool, not all of them use it as intended. A study in a physics thermodynamics course found that only about half of the learners who used the provided app actually spaced their reviews. The rest concentrated their use closer to the exam. Learners who used the app in a truly spaced manner scored notably higher on the final exam: an average of 70% compared to 64% for massed users and 61% for non-users [14]. This finding highlights an important design issue: providing the tool is not enough, the system needs to encourage or enforce spaced use.

Another gap in the literature is the focus on learners rather than instructors. Most spaced repetition tools are designed for learners who create or import their own review material. Research on a template-based spaced repetition API specifically noted this imbalance, arguing that educators need tools that help them build spaced repetition into their courses and track how learners are engaging with review material [12]. The platform described in this thesis addresses this partly through the creator role, which allows course authors to write flashcards that are then served to learners through the automated review queue.

2.3.4 Flashcards as a Learning Tool

Flashcards are one of the oldest and most widely used study tools. Their basic format, a question or prompt on one side, an answer on the other, supports active retrieval practice, which research consistently finds to be more effective for long-term memory than passive re-reading [10]. The act of looking at a prompt, attempting to recall the answer before flipping the card, and then checking whether the recall was correct provides immediate feedback and engages memory in a way that simply re-reading notes does not.

Digital flashcards extend this basic format in several ways. They can incorporate images, audio, and other media alongside text. Research examining the role of audio and visual inputs in digital flashcards found that combining different types of information, what the study called "referential stimuli" that bridge verbal and nonverbal content, can further support memorization, particularly for vocabulary learning in a second language [15]. The study applied Dual Coding Theory, which holds that information is better retained when it is encoded in both verbal and visual channels simultaneously, to flashcard design for computer-assisted language learning [15].

For the platform in this thesis, flashcards are authored per lesson and are only accessible to learners after they have completed the relevant lesson. This design choice ensures that flashcards function as a consolidation tool, reinforcing material that has already been introduced, rather than as a first point of contact with new content. The front/back structure is kept simple, but the connection to the spaced repetition system means that each flashcard is reviewed at intervals that are calibrated to the learner's performance history.

2.4 Gamification in Education

2.4.1 What Gamification is and why it is Used

Gamification is the application of game design elements to non-game contexts. In education, this typically means adding mechanics like points, levels, badges, and leaderboards to a learning environment that is not itself a game. The idea is to borrow the motivational power of games (their ability to create engagement, reward persistence, and make progress feel visible and meaningful) and apply it to activities that learners might otherwise find repetitive or difficult to sustain [4],[5].

The theoretical basis for gamification in education draws from motivational psychology. Self-determination theory, for example, identifies autonomy, competence, and relatedness as the three basic psychological needs whose satisfaction supports intrinsic motivation. Well-designed gamification can address all three: learners feel a sense of control over their progress, they receive clear signals of growing competence as they earn XPs (Experience Points) and badges, and social elements like leaderboards create a sense of relatedness to other learners. The goal is not to replace intrinsic motivation with external rewards, but to create conditions where engagement and motivation reinforce each other [4].

It is important to distinguish gamification from educational games. An educational game is a game designed with learning as the primary objective, the learning happens through play. Gamification adds game elements to an existing learning activity without turning the activity itself into a game. A learner studying for a certification exam is still studying. The points and badges are added to make the process more engaging, not to turn the exam into a game [5].

2.4.2 Game Elements: XPs, Badges, Streaks, and Leaderboards

The most commonly studied game elements in educational technology research are points (or XPs), badges, leaderboards, progress bars, challenges, and levels [5]. Each serves a different psychological function, and their effectiveness can vary depending on the context and the learner.

XPs provide a continuous signal of progress. Every action that earns XPs (completing a lesson, passing a quiz, finishing a review session) contributes to a total that grows over time. This makes progress visible even when broader milestones feel distant. In the platform described here, completing a lesson awards 50 XPs, passing a quiz awards 100 XPs, failing a quiz still awards 20 XPs (to reward the attempt), and completing a review session awards 10 XPs. The different values reflect differences in effort and achievement, and the fact that failure still earns some XPs is a deliberate design choice to avoid discouraging learners who struggle.

Badges are discrete rewards that mark specific achievements. Unlike XPs, which accumulates continuously, badges are earned at identifiable moments. A "First Lesson" badge after the student completes their first lesson, a "Course Completer" badge after finishing an entire course, a "Streak 7" badge for maintaining a seven-day activity streak. Research suggests that badges are most effective

when they represent meaningful achievements rather than trivial ones, and when they are tied to behaviors the platform wants to encourage [5].

Streaks track consecutive days of activity and are particularly effective at building habits. The psychological mechanism is similar to the "don't break the chain" principle: once a student has maintained a streak for several days, the cost of losing it becomes a motivator in itself. Leaderboards introduce a social comparison dimension: students can see how their XPs total compares to others. This can increase motivation for some learners but may discourage others who find themselves consistently at the bottom. For this reason, some platforms display only the top performers, or show each learner their position relative to nearby ranks rather than the full list [5].

2.4.3 Evidence for Gamification's Effectiveness

The research evidence on gamification's effectiveness in education is generally positive, though not uniformly so. A large systematic literature review of empirical gamification studies in higher education found that the most commonly used elements (points, badges, and leaderboards) do tend to improve engagement and motivation when implemented well [5]. At the same time, the review noted that the effects are not consistent across all learners and contexts, and that there is no single design that works for everyone.

A structural equation modelling study comparing a gamified university mathematics course to a non-gamified version found that motivation was the factor with the largest effects on cognitive and metacognitive outcomes [4]. The gamified course reduced learner demotivation, and when learners actually used the gamified platform, their motivation had a positive and significant influence on academic performance. This is a useful finding because it separates two things that are sometimes conflated: gamification's effect on motivation, and motivation's effect on performance. Both links need to be present for gamification to improve learning outcomes [4].

One consistent pattern in the literature is that gamification works better when the game elements are aligned with the learning goals. Points awarded for superficial actions, clicking through slides, for example, can produce engagement with the platform without producing genuine learning. On the other hand, points awarded for meaningful activities, like passing a quiz, completing a spaced review session or finishing a course, connect the reward to the behavior that actually produces learning. This is the approach taken in the platform described here [4],[5].

2.4.4 Gamification in E-Learning Platforms

Gamification has become a standard feature in many commercial e-learning platforms. Duolingo is perhaps the most widely cited example, using streaks, XPs, leaderboards, and a live system to drive daily engagement for language learning. Kahoot! uses competitive quizzing with real-time leaderboards in classroom settings. Khan Academy uses badges and energy points. Coursera and edX have experimented with certificates and peer recognition as social reward mechanisms [5],[13].

The combination of gamification and spaced repetition is particularly interesting because the two approaches address different problems. Spaced repetition targets the cognitive mechanics of memory consolidation: it makes the learning process more efficient. Gamification targets motivation: it makes the learning process more appealing and easier to sustain over time. When the two are combined, the gamification layer can drive the repeated engagement that makes spaced repetition work [13]. Research specifically examining this combination in a K-12 STEM mobile learning context found that

both strategies independently improved outcomes, and that their combination, delivered through a mobile app, amplified the benefits of each [13].

The platform described in this thesis integrates gamification and spaced repetition in exactly this way. The XPs reward for completing a review session creates an incentive to return to the spaced repetition queue each day. The streak mechanic reinforces daily activity. The badge for "Review Starter" marks the beginning of a habit. This design is deliberate and reflects what the research suggests about how these mechanics interact.

2.5 AI-Assisted Content Creation in Education

2.5.1 Automatic Quiz and Question Generation

Creating high-quality assessment questions is one of the most time-consuming parts of developing an online course. Writing questions that are clear, relevant, and appropriately challenging requires expertise and significant effort. As a result, many online courses have limited or low-quality assessments, which reduces their educational value. Automatic Question Generation (AQG) addresses this problem by using natural language processing to extract key information from a text and turn it into quiz questions [6].

Early approaches to AQG used rule-based methods, like identifying named entities, key terms, or syntactic patterns and generating questions based on templates. These produced reasonable results for factual content but struggled with more complex conceptual material. More recent approaches use large language models, which can generate questions that are more varied in form and better matched to the meaning of the source text rather than just its surface features [1],[2].

The quality of automatically generated questions depends heavily on the quality of the input text. If the source material is well-written and clearly organized, the generated questions tend to be relevant and accurate. If the source is poorly structured or ambiguous, the generated questions often reflect those problems. This is one reason why the platform in this thesis includes a content quality analysis feature: before generating quiz questions, the AI assesses the lesson content and provides feedback to the course creator on its clarity and structure.

On the student side, the platform generates quizzes from lesson content using OpenAI's GPT-4.1-mini model. For each quiz question the learner answers incorrectly, the platform provides an AI-generated explanation of why the correct answer is correct. This closes the feedback loop that purely automated assessment systems often leave open: it is not enough to tell a student they got something wrong. The system needs to explain what the correct understanding looks like [3].

The quiz hint feature works similarly, when a learner requests a hint on a question, the AI generates a contextual clue that points toward the answer without giving it away directly. This type of scaffolded support is well aligned with research on adaptive tutoring systems, which suggests that hints and feedback should guide learners toward correct understanding rather than simply correcting them [8].

2.5.2 Automatic Flashcard Generation

Flashcard creation, like quiz creation, is labor-intensive. Learners who create their own flashcards spend significant time on the process, time that could be spent studying. One study estimated that learners spend roughly half of their study time on methods that are less effective than active retrieval

[6]. If flashcard creation can be automated, learners can get the benefit of retrieval practice without the overhead of content preparation.

The FlashMe system, described in the research literature, uses a multi-stage NLP pipeline based on transformer models to generate flashcards from dense text. The pipeline extracts key concepts, formulates them as question-answer pairs, and filters the output for quality [6]. This is similar in spirit to the flashcard generation feature in the platform described here, which uses GPT-4.1-mini to generate front-back flashcard pairs from lesson content. The creator can review and edit the generated flashcards before they are published, which keeps a human in the loop and allows for quality control.

Automatic generation does not eliminate the need for human judgment, but it significantly reduces the time required to produce study materials. A course creator who might otherwise spend hours writing flashcards for a ten-lesson course can instead review and refine AI-generated drafts in a fraction of the time. This makes comprehensive flashcard coverage of course content much more achievable in practice [6].

2.5.3 AI Feedback and Content Analysis

Beyond question and flashcard generation, AI can be used to analyze and improve learning content itself. The platform described in this thesis includes a content quality analysis feature, in which the AI reviews lesson text and provides the creator with feedback on its educational quality, looking at factors like clarity, completeness, appropriate difficulty, and whether the content is likely to support the learning objectives stated in the course.

This type of AI-assisted editorial feedback reflects a broader trend in educational AI toward tools that support educators rather than replacing them [3],[16]. Research on AI-based learning platforms for college English listening, for example, identified the lack of personalized guidance and feedback as a major weakness of existing systems, and proposed a three-layer architecture that includes an AI component dedicated to evaluation and feedback [16]. The insight that AI should evaluate and improve content quality, and not just deliver it, is directly relevant to the design of the platform in this thesis.

From the student’s perspective, AI feedback is most valuable when it is timely and specific. A general comment that an answer is wrong is less useful than an explanation that identifies the specific misconception and points toward the correct understanding. The platform's explanation feature, which generates a targeted explanation after a failed quiz question, is designed to provide exactly this kind of specific, actionable feedback. Research on AI in adaptive learning suggests that personalized feedback is one of the strongest mechanisms through which AI improves learning outcomes [1],[8].

2.6 Lifelong Learning and Skills Certification

2.6.1 What Lifelong Learning Means in the Digital Age

Lifelong learning refers to the idea that education does not end with formal schooling. It is an ongoing process that extends through adulthood and across different phases of a person's career and life. In the current economic context, this is not just a philosophical position but a practical necessity. Technological change is reshaping labor markets at a pace that formal education systems struggle to keep up with, and many workers need to acquire new skills (or update existing ones) repeatedly over the course of their careers [7].

Digital platforms are particularly well suited to supporting lifelong learning because they remove many of the barriers that make traditional education inaccessible to adults in work: fixed schedules, physical location, high cost, and the need to commit to multi-year programs. An online platform can be accessed at any time, from anywhere, at a fraction of the cost of a university program. For someone who needs to learn a specific skill quickly to respond to a change in their job or industry, this flexibility is extremely valuable [7].

At the same time, the track record of digital lifelong learning is mixed. MOOCs were widely promoted as a solution to the global education gap, but research and data showed that their completion rates were very low, often below 10%, and that they were disproportionately used by people who already had high levels of formal education [7]. The design of early MOOCs replicated the lecture-based format of university courses, without the social structures, accountability mechanisms, and personalized support that help learners complete formal programs. Rethinking what digital lifelong learning should look like (moving away from passive content delivery toward active, personalized, and socially connected learning) is one of the central challenges in the field [7].

Research on designing models for massive digital lifelong learning identifies AI-based instructional assistants, multi-dimensional learner data, and the development of cross-cutting skills, such as critical thinking and problem-solving, as key priorities for next-generation platforms [7]. The platform described in this thesis addresses several of these priorities: it uses AI to support learners through hints, explanations, and personalized review scheduling, it collects and uses behavioral data through the weakness map and the dashboard, and it is designed around active learning through quizzes and spaced repetition rather than passive content consumption.

2.6.2 Skills Certification and Micro-Credentials

Certification has always been an important part of formal education. A degree or certificate signals to employers and others that the holder has acquired a defined set of skills or knowledge. In lifelong learning and professional development, certification plays a similar role: it makes learning visible and verifiable to third parties who were not present for the learning process [7].

Micro-credentials are a relatively recent development that adapts the certification idea to shorter, more focused learning experiences. Instead of a full degree that represents years of study, a micro-credential certifies completion of a specific course or skill module. They are designed to be stackable: learners can accumulate multiple micro-credentials over time, each representing a different competency, and together they form a picture of the learner's skills profile [7].

The platform described in this thesis issues certificates on completion of a full course. Each certificate is generated with a unique UUID-based public URL, which makes it verifiable. Anyone who receives the certificate link can confirm that the named learner completed the named course. The certificate is also print-ready, supporting physical distribution if needed. This design is simple but reflects the core function of certification: making a learning achievement portable and credible beyond the platform itself.

One challenge with platform-issued certificates is credibility. A certificate from a well-known institution or accreditation body carries more weight than one from an unfamiliar platform. Building trust in the certification system requires transparency about what the certificate represents, what content was covered, how learning was assessed, and what the passing criteria were. The platform addresses this by tying certification to full course completion and quiz performance, so the certificate reflects a verifiable learning process rather than just enrollment.

2.6.3 Platform Design for Continuous Learners

Learners who engage in lifelong learning have different needs than typical students in a formal program. They often have limited time, specific and practical learning goals, existing knowledge in some areas, and gaps in others. They need to be able to enter a course at a level appropriate to their background, make efficient use of their study time, and apply what they learn relatively quickly. A platform designed for this audience should prioritize flexibility, efficiency, and clear feedback about progress [7].

The dashboard in the platform described here is designed with this in mind. It shows the learner's current XPs total, their streak, their quiz pass rate, a preview of their weakness map, and their progress through enrolled courses. This gives learners a clear and immediate picture of where they stand and what they should focus on. The weakness map in particular is designed to direct attention efficiently: it aggregates quiz failure and review error data per lesson, tracks whether performance on each topic is improving or worsening over time, and recommends a specific action: re-read the lesson, retry the quiz, or review the flashcards. For a learner with limited study time, this kind of directed recommendation is more useful than a general overview of course content [2],[3].

The two-role structure of the platform (creator and student) also supports the lifelong learning context. Creators in this context are not necessarily professional teachers. They might be subject-matter experts in a company, professionals who want to share knowledge with a community, or students who have completed a course and want to create their own materials. This reflects a broader trend in lifelong learning toward community-driven and peer-produced content, where the line between learner and creator is not fixed [7],[9].

2.7 Chapter Summary

2.7.1 Key Theoretical Concepts and their Relation to the Platform

This chapter has covered five main areas of theory: adaptive e-learning, spaced repetition and memory science, gamification, AI-assisted content creation, and lifelong learning with skills certification. Each of these areas is directly reflected in one or more features of the platform.

Adaptive learning research [1],[8] motivates the weakness map and the way quiz and review data are used to identify and address individual learner weaknesses. The spaced repetition literature [10],[11] provides the scientific basis for the interval-based review system and explains why regular, spaced review of flashcards produces better long-term retention than studying once. Gamification research [4],[5] supports the use of XPs, badges, streaks, and the leaderboard as tools for sustaining motivation and building regular study habits. The AI content generation literature [6] underpins the quiz and flashcard generation features, as well as the content quality analysis tool for creators. Finally, the lifelong learning and certification literature [7] shapes the overall design philosophy: the platform is built for learners who are motivated by practical goals, need efficient feedback, and want their achievements recognized in a portable and verifiable way.

The theoretical grounding is not just academic. Each design choice in the platform can be traced back to a research finding or a well-established principle. This is important because it means the design is accountable and can be evaluated against the claims that motivated it and refined if the evidence changes.

2.7.2 Transition to the Technology and Implementation Chapters

With the theoretical background established, the following chapters turn to the practical question of how the platform was actually built. Chapter 3 covers the technology stack and architecture: why Laravel was chosen as the backend framework, how Inertia.js enables a single-page application experience without a separate API layer, and how Vue.js is used for the frontend interface. Chapters 4 and 5 describe the implementation in detail, walking through each major feature quiz generation, spaced repetition, gamification, the weakness map, and certification and explaining how the theoretical ideas from this chapter are translated into working code and system design. The goal of those chapters is to show not just what was built, but that what was built reflects a coherent and research-informed approach to the problem of building an intelligent platform for lifelong learning.

Chapter 3: Technology Stack and Architectural Approach

3.1 Introduction to the Technology Chapter

3.1.1 Purpose and Scope of the Chapter

This chapter presents and explains the technology choices made during the development of this platform. The purpose is not simply to list tools and libraries, but to explain how each choice relates to the overall goals of the project and why one option was preferred over another. Technology decisions in a project like this are never purely technical. They also reflect constraints like development time, maintainability, and how well a stack can accommodate both conventional web application needs and AI-driven features.

The scope of the chapter covers the full technology stack: the backend framework, the frontend layer, the bridge between them, the AI integration approach, supporting libraries, data persistence, testing infrastructure, and code quality tooling. Each component is discussed in enough detail to make its role in the system clear, without going into implementation specifics that belong in later chapters. The goal is to give a complete picture of what the platform is built with and why this set of choices works well together as a whole.

3.1.2 Criteria for Technology Selection

Several factors guided technology selection throughout the project. The first was productivity, given the scope of the platform, it was important to use tools that reduce boilerplate and provide solid built-in functionality. Related to this was the maturity and stability of each technology. Choosing well-established frameworks and libraries lowers the risk of hitting undocumented edge cases or having to work around missing features.

A second criterion was how well the technologies integrate with each other. A stack where components fit naturally together avoids a lot of glue code and makes the codebase easier to understand. In this case, the combination of Laravel, Inertia.js, and Vue 3 is a well-known pattern that has proven compatibility.

A third factor was AI integration. One of the main requirements of the platform is support for AI-driven features such as quiz generation, flashcard creation, and content analysis. The chosen stack needed to accommodate this without requiring a fundamentally different architectural approach for the AI parts. As will be explained later in this chapter, the Laravel AI SDK makes it possible to define and invoke AI agents using the same patterns used elsewhere in the backend.

Finally, long-term maintainability was considered. The code should remain readable and manageable as the project grows. This pointed toward technologies with strong community support, clear documentation, and good tooling for code quality.

3.1.3 Overview of the Final Stack

The platform is built as a full-stack web application using PHP 8 and Laravel 12 on the backend, with Vue 3 on the frontend. Inertia.js serves as the bridge between these two layers, enabling a

server-driven single-page application model without the overhead of building a separate REST API. Tailwind CSS 4 handles styling, and Vite 7 is the build tool.

For AI capabilities, the platform uses the Laravel AI SDK, with OpenAI's gpt-4.1-mini model powering all implemented agent classes. Supporting libraries cover rich-text editing, data visualization, animation, and component primitives. Data is managed via Eloquent ORM over a PostgreSQL relational database. Code quality is maintained through Laravel Pint, ESLint 9, and Prettier 3, and automated testing uses PHPUnit 11.

3.2 Overall Architectural Approach

3.2.1 Full-Stack Web Application Model

The platform follows a full-stack web application model where a single Laravel application handles both the backend logic and the delivery of frontend assets. This is different from a separated architecture where a frontend application consumes a backend API independently. In this project, Laravel owns the routing, middleware, authentication, authorization, data access, and response preparation. The frontend, built with Vue 3, is responsible for rendering the user interface based on data that Laravel has already prepared and validated.

This model has several practical advantages. There is a single codebase to maintain, a single deployment unit, and a natural place for cross-cutting concerns like authentication and authorization. The frontend does not need to re-implement any of these concerns, it receives pre-processed, already-authorized data from the server.

3.2.2 Server-Driven SPA Approach

While the application uses Vue 3 to render interactive page components on the client side, it is not a traditional SPA where routing and data fetching are handled entirely by the frontend. Instead, Inertia.js implements a server-driven SPA approach: navigation events are intercepted by Inertia, which sends a lightweight request to the server, receives a JSON response containing the component name and its props, and replaces the current page component without a full page reload.

This means that page routing is defined in Laravel's route files, not in Vue Router. Controllers load the necessary data through Eloquent and return an Inertia response specifying which Vue component to display and what props to pass to it. The result is that the application behaves like a modern SPA from the user's perspective, with fast, seamless navigation, while the server remains the authoritative source for routing, data, and access control.

Shared props are a key feature of this setup. Inertia makes certain pieces of data available to every page without requiring each controller to explicitly pass them. In this project, shared props include the authenticated user object, flash messages, the application name, route metadata and sidebar state. These values are available to any Vue component that needs them.

3.2.3 Separation Between Page Delivery and Asynchronous Operations

Most of the platform's interactions follow the standard Inertia request cycle: the user navigates or submits a form, a Laravel route handles the request, a controller processes it and returns an Inertia response, and the Vue layer updates accordingly. This covers the vast majority of CRUD operations: creating courses, editing lessons, enrolling in a course, attempting a quiz, and so on.

However, there is a second category of interactions that does not fit this model well: AI-driven operations and file uploads. These are handled through a separate set of API routes that return JSON responses, with Axios used on the client side to make the requests asynchronously. This separation is intentional. AI operations can take a variable amount of time to complete, and it is better to handle them as explicit async requests rather than as page navigations. File uploads for lesson images follow the same pattern for similar reasons.

The result is two clearly defined interaction surfaces. The first is the web route layer, which drives Inertia page responses and handles all standard application flows. The second is the API route layer, which handles Axios-based requests for AI operations and file management. These two surfaces are distinct in their routes, their response format, and their client-side handling, but they share the same authentication infrastructure and the same Eloquent data layer.

3.2.4 Why this Architecture was Selected

The server-driven SPA model with Inertia.js was selected because it provides the interactivity and responsiveness of a modern SPA without the architectural complexity of maintaining a separate, independently deployed frontend application. Building such an application that consumes a REST API would have required duplicating concerns like routing and access control on both sides, and would have added significant development overhead for features that do not benefit from that separation.

At the same time, a traditional server-rendered application with full page reloads would not have been suitable for the interactive features the platform requires, in particular, the animated flashcard review deck, the dynamic quiz interface, and the real-time dashboard. Inertia.js provides a practical middle ground: clean server-side routing and data preparation combined with a Vue-based component layer that can deliver the interactivity users expect.

This architecture also made AI integration straightforward. Because the backend remains the central authority, AI agent classes can be invoked directly from controllers, with no need for a separate service tier or message broker for the standard use cases.

3.3 Backend Technologies

3.3.1 PHP 8

PHP 8 is the base language for all backend logic in this project. While PHP has sometimes been viewed as a legacy language, PHP 8 introduced a significant number of modern features that make it much more suitable for well-structured, maintainable applications. Features used in this project include enums, for example, to represent question types, spaced repetition intervals, and course theme color options, as well as typed properties and typed method signatures throughout the codebase.

PHP remains one of the most practical choices for web platforms that combine conventional request-response cycles with more complex logic like AI orchestration. The language's ecosystem, and in particular the Laravel framework, is mature and well-maintained. For a project of this scope, PHP 8 with Laravel provides a highly productive development environment without sacrificing code quality.

3.3.2 Laravel 12

Laravel 12 is the main backend framework. It handles route registration for both the web and API layers, middleware configuration, controller-based request handling, authentication flows including

email verification and password reset, form request validation, authorization policies, filesystem management for uploads, and testing support.

The choice of Laravel as the backbone of this project is justified by the breadth of functionality it provides out of the box. Routing, persistence, security, storage, testing, and AI integration all work within the same framework, which significantly reduces the integration cost compared to assembling these capabilities from separate packages.

3.3.3 Eloquent ORM

Eloquent is Laravel's built-in ORM and is used throughout the project as the primary interface to the relational database. The domain model is expressed as a set of Eloquent model classes, each corresponding to a database table, with relationships defined using Eloquent's built-in relationship methods.

The relational model covers a range of entity types and their associations. Users can author courses (a one-to-many relationship) and also enroll in courses (a many-to-many relationship managed through a course_user pivot table). Courses contain lessons, which in turn contain flashcards and quizzes. Quizzes have questions, and questions have answers. Reviews connect users to flashcards, and quiz attempts connect users to quizzes through the user_quiz_attempts table, with incorrect answers stored as a JSON field within each attempt record. Badges and certifications are also linked to users through their respective pivot and direct relationship tables.

The domain model makes use of custom query scopes for common filtering patterns, for example, retrieving publicly visible courses, finding cards that are due for review, or computing a user's progress through a course. Computed attributes are used for things like enrollment state and lesson completion status. For operations that update multiple related entities at once, the logic is wrapped in database transactions to ensure consistency.

Image uploads are stored on the filesystem rather than as binary blobs in the database, which keeps the database lean and avoids performance issues associated with storing large binary data relationally.

3.3.4 Validation, Authorization, and Authentication Mechanisms

Data validation in this project is handled at the framework level. Important write operations, such as creating or updating courses, lessons, quizzes, and cards, use dedicated Form Request classes that define validation rules declaratively. This keeps controller methods clean and ensures that validation logic is not scattered across the codebase.

Authorization is managed through Laravel's policy system. Each major resource type has a corresponding policy class that governs what the authenticated user is allowed to do with it. For example, a user can only edit a course if they are its author, they can only access a lesson if they are enrolled in the parent course or are the course author. This is important because the application serves multiple user roles in different contexts: a user may be an instructor in one course and a learner in another.

Authentication is handled by Laravel's built-in auth system. The middleware stack enforces authentication on all protected routes, and email verification is required before users can access the main application. Password management, profile updates, appearance preferences, and account deletion are all handled through dedicated settings routes.

3.3.5 Error Handling and Logging Facilities

Error handling in the platform follows Laravel's standard exception handling infrastructure, with additional controller-level try/catch blocks around operations that are more likely to fail, in particular, file upload handling and AI requests. Image upload controllers return structured JSON responses with explicit success or error payloads, making it straightforward for the client to handle either outcome.

AI controllers follow a similar pattern: if an AI request fails, the controller catches the exception and returns a controlled JSON error response rather than letting the application throw an unhandled error. AI explanation flows also log errors when requests fail, which helps with debugging during development.

The error handling approach in this project is practical and appropriate for the application's current scale. There is no claim of enterprise-grade monitoring or advanced observability infrastructure, the focus is on predictable, controlled failure behaviour for the operations that are most likely to encounter errors.

3.4 Frontend Technologies

3.4.1 Inertia.js

Inertia.js version 2 is not a minor utility in this stack, it is the architectural bridge between the Laravel backend and the Vue frontend. Without Inertia, building this application with both server-side routing and a reactive Vue interface would require either a decoupled API architecture or a hybrid approach with Blade templates and mounted Vue components. Inertia makes it possible to use Laravel's routing and controller patterns as-is, while rendering Vue page components on the client.

From the controller's perspective, returning an Inertia response looks similar to returning a Blade view: the controller calls Inertia, passes a component name and a props array, and Inertia takes care of the rest. On subsequent navigations, Inertia intercepts the browser's navigation events, makes a background request to the same Laravel route, and swaps the current Vue page component for the new one without a full page reload.

Form submissions also benefit from Inertia. The Inertia form helper manages form state, submission, and error display, integrating naturally with Laravel's validation error format. This covers most standard CRUD operations in the application.

3.4.2 Vue 3

Vue 3 is used as the frontend component framework. Page components live under the resources/js/pages directory and are rendered by Inertia based on the component name returned by the controller. A set of reusable layout components provides the authenticated shell, including the sidebar and navigation structure, that wraps most page components.

Vue 3's Composition API is used extensively, with the <script setup> syntax being the standard approach for most components. This makes component logic more concise and easier to organize compared to the Options API, particularly for components that combine reactive state, computed values, and side effects.

Vue components in this project handle a range of interface responsibilities: the interactive quiz-taking interface, the animated flashcard review deck, the dynamic course management dashboard, and the

nested course and lesson editing forms. These are the parts of the application where the frontend framework's reactive capabilities make the biggest difference to the user experience.

3.4.3 Tailwind CSS 4

Tailwind CSS version 4 handles all styling in the frontend layer. A notable difference between Tailwind CSS 4 and earlier versions is its CSS-first configuration approach: there is no tailwind.config.js file. Instead, configuration is done directly in CSS, and design tokens are defined as CSS custom properties (variables).

Per-course theme colors add another layer to this system. Each course can have a color theme stored in the themes database table, and these values feed into the component-level styling for course-specific views.

3.4.4 Axios

Axios is used in this project as the HTTP client for asynchronous requests that fall outside the Inertia navigation model. It is important to be clear about where Axios is and is not used. Standard page navigation and most CRUD operations go through Inertia, which handles its own HTTP requests internally. Axios is only used for a specific set of operations: AI-driven features (quiz generation, quiz hints, explanations, flashcard generation, and lesson analysis) and file uploads through the TipTap editor.

This selective use keeps the two interaction surfaces clearly separated. AI requests and file uploads return JSON responses and need to be handled as async operations in the component, which is where Axios fits naturally. Using Axios for everything would undermine the Inertia model and introduce unnecessary complexity in how the server needs to respond.

3.4.5 Vite

Vite 7 is the build tool and development server for the frontend. It is configured with the official Laravel plugin, the Vue plugin, and the Tailwind Vite plugin. The standard client entry point is resources/js/app.ts. Vite's hot module replacement makes the development experience fast. Changes to Vue components and CSS are reflected in the browser almost immediately, without a full rebuild.

3.5 Artificial Intelligent Technologies

3.5.1 Laravel AI SDK

The Laravel AI SDK is the main technology used to integrate AI capabilities into the platform. It should be understood as an architectural choice, not just a convenience library. The SDK allows AI features to be defined as agent classes within the Laravel backend, following patterns that are consistent with the rest of the application. Prompts, model selection, and structured output expectations are all defined close to the backend logic, making it easy to reason about what each AI feature does and how it fits into the application's flow.

Controllers invoke agent prompts directly and process the responses. This means there is no need for a separate AI service layer or an external orchestration system for the use cases implemented in this project. The AI logic lives in the same codebase as the rest of the backend, benefits from the same testing infrastructure, and is subject to the same code quality standards.

This integration approach lowers architectural overhead considerably. Adding a new AI-assisted feature means writing a new agent class and calling it from a controller, a familiar pattern for any developer already working in the Laravel codebase.

3.5.2 AI Model and Provider Configuration

The Laravel AI SDK can be configured with a large set of AI providers: OpenAI, Gemini, Anthropic, Cohere, Azure OpenAI, DeepSeek, Groq, Mistral, Ollama, OpenRouter, VoyageAI, xAI, ElevenLabs, Jina, and others. This configuration reflects the SDK's multi-provider design and makes it straightforward to switch providers in the future.

All implemented agent classes, use as an active provider OpenAI, and the model used is gpt-4.1-mini. This applies uniformly across all AI features in the platform. The broader provider configuration represents future flexibility, not the current operational setup.

3.5.3 Structured-Output Design with JSON Schemas

One of the most intentional design decisions in the AI integration is the use of structured outputs rather than free-form text generation. Each agent class defines a JSON schema-like output contract that specifies the exact structure the AI response should follow. For example, a quiz generation request does not return a paragraph of text from which questions need to be extracted. It returns a structured object containing an array of questions, each with its text, type, and answer options.

This approach makes AI responses much easier to validate, parse, and integrate into the application. The controller receives a predictable data structure that can be directly used to create database records or return to the client as a JSON payload. It also reduces the risk of unexpected or malformed responses causing errors in the processing pipeline, while keeping the cost of the response generation lower.

Across the implemented features, structured outputs cover quiz questions with answer options and correct answer indicators, hint text for specific questions, explanation content for failed question attempts, flashcard front/back pairs, and lesson content feedback with a numerical quality score.

3.5.4 High-Level AI-Supported Capabilities of the Platform

The platform implements five AI-assisted capabilities, each corresponding to a dedicated agent class in the backend:

The first is quiz generation from lesson content. Given the text content of a lesson, the platform can generate a set of quiz questions with multiple-choice answer options. This reduces the manual effort required for instructors to create assessment content.

The second is quiz hints. When a learner is attempting a quiz and requests a hint for a specific question, an AI agent generates a contextual hint without revealing the answer directly. This supports learning without undermining the assessment value of the quiz.

The third is explanation generation for failed quiz questions. After a learner submits a quiz and answers one or more questions incorrectly, the platform can generate an explanation for why a given answer was wrong and what the correct reasoning should be. This turns errors into learning opportunities.

The fourth is flashcard generation from lesson content. Similar to quiz generation, this feature takes lesson content as input and produces flashcard pairs: a front side with a term or question and a back side with an explanation or answer.

The fifth is lesson content quality analysis. An instructor can request an analysis of a lesson's content, and the AI agent returns a quality score along with structured feedback. This helps instructors improve their material before publishing.

3.5.5 Strengths and Limitations of the AI Approach

The main strengths of the AI integration in this project are practical. The Laravel AI SDK allows AI features to be added using native Laravel patterns, which keeps the codebase consistent and avoids introducing a parallel architectural paradigm just for AI. Structured outputs make the AI results programmatically reliable. The separation of each AI capability into its own agent class creates a clean boundary and makes each feature independently testable and maintainable.

At the same time, the current implementation has some limitations that are worth acknowledging. AI calls are currently made synchronously, they are not queued or processed in the background. For features where response time is not critical, queuing would provide a better user experience, but this has not been implemented in the current version. The AI capabilities are also text-generation focused; multimodal inputs are not used in practice, even though some configured providers support them.

3.6 Supporting Libraries and Interface Technologies

3.6.1 TipTap Rich-Text Editor

TipTap version 3 is used as the rich-text editor for lesson content authoring. When an instructor creates or edits a lesson, they interact with a TipTap editor instance embedded in the Vue page component. The editor supports formatting options standard for educational content: headings, lists, emphasis, and links, as well as image insertion.

Images inserted into lesson content are uploaded to the Laravel backend and stored on the server filesystem. The editor sends the image file to a dedicated API endpoint, receives a URL in response, and embeds that URL in the HTML content. Lesson content is stored and retrieved as HTML, which makes it straightforward to render in the learner-facing lesson view without additional transformation.

TipTap was selected over simpler textarea-based approaches because educational content benefits from structure and formatting. A plain text input would be insufficient for lessons that include step-by-step explanations, structured lists, or visual content.

3.6.2 Charting and Data-Visualization Libraries

Data visualization in the platform is handled by Unovis, a component-based charting library. Unovis is used in the dashboard to render two specific visualizations: a weekly activity chart showing the user's recent engagement, and a course completion donut chart that summarizes overall progress across enrolled courses. These two charts serve a specific informational purpose on the dashboard, and Unovis was chosen for its lightweight footprint and compatibility with Vue 3.

3.6.3 Motion and Interaction Libraries

Two libraries handle animation and GPU-rendered visual effects in the platform. The first is motion-v, which is used specifically for the animated swipe-based flashcard review interaction. When a user is reviewing flashcards, they can swipe or drag cards to indicate whether they remembered the content, with smooth animated transitions between cards. This interaction is an important part of the user experience because the spaced repetition review system is a core feature of the platform.

The second is OGL, a lightweight WebGL library used to render an Aurora-style animated visual background effect. This is used in specific parts of the interface where visual atmosphere is part of the design intent. OGL operates at the GPU level, which means it can produce smooth visual effects without impacting the performance of the rest of the page.

Custom drag-and-drop behavior in the application is implemented using the browser's native drag-and-drop API rather than a third-party library. This keeps the dependency count lower and avoids compatibility layers around a relatively well-supported browser API.

3.6.4 Components and Icon Libraries

The component layer is built on a combination of Reka UI and shadcn-vue style component scaffolding. Reka UI provides accessible primitive components, things like dialogs, dropdowns, tabs, and form controls, that handle keyboard navigation and ARIA attributes correctly out of the box. These primitives are the foundation on which the application's visible UI components are built.

shadcn-vue style architecture means that UI components are not consumed as black-box library dependencies. Instead, their source code lives directly in the project and can be customized freely. This is particularly useful for a project with a specific design system, where tight control over component appearance and behavior is needed.

Icons are provided by Lucide Vue Next, a Vue 3 compatible icon library with a consistent visual style. Composable utility logic, things like scroll detection, element sizing, event listeners, and storage access, is handled by VueUse, which provides a large collection of Vue 3 composables that follow the Composition API pattern.

3.7 Data Persistence and Quality Assurance Technologies

3.7.1 Relational Database Approach

The platform uses a relational database as its primary persistence layer, accessed through Eloquent ORM. The default configuration uses PostgreSQL, which simplifies local development and makes the project easy to run without additional database server setup.

The schema covers all major domain entities: users, courses, lessons, cards, quizzes, questions, answers, reviews, course enrollments, lesson completions, quiz attempts, certifications, badges, user and badges. The choice to use a relational model reflects the structured, interconnected nature of the domain data. Relationships like course-lesson-card hierarchies and user-enrollment-progress tracking are a natural fit for relational tables with foreign key constraints.

Image data is stored on the filesystem rather than in the database. Binary blobs in relational databases create performance and maintenance problems at scale, and using filesystem storage with database-stored URLs is a cleaner separation of concerns.

3.7.2 Automated Testing with PHPUnit

The project uses PHPUnit 11 as its automated testing framework. Tests are organized at two levels: unit tests for individual services and logic components, and feature tests for HTTP-level flows that exercise routes, controllers, and database interactions together.

Test coverage spans a range of functional areas: authentication and settings flows, the dashboard and weakness map features, course and lesson management, quiz and review functionality, certification generation, AI endpoints, and gamification services. This coverage ensures that the core application flows work correctly and that changes to the codebase do not break existing functionality without being detected.

Tests run against an in-memory PostgreSQL database, which keeps test execution fast and avoids any dependency on an external database server. The in-memory database is created fresh for each test run, so tests are fully isolated from each other and from any development data.

3.7.3 Code Quality Tools: Pint, ESLint, and Prettier

Three tools are used to maintain code formatting and quality standards across the codebase. On the PHP side, Laravel Pint is used as the code formatter. Pint is opinionated and requires minimal configuration, running it reformats PHP files according to a consistent style that aligns with Laravel conventions.

On the frontend side, ESLint 9 handles linting for JavaScript, and Vue files. It enforces code quality rules and catches common errors like unused variables, incorrect Vue component usage, and type issues. Prettier 3 is used for frontend code formatting, it handles whitespace, indentation, and line length consistently across all frontend files.

An important addition is the Tailwind-aware Prettier plugin, which ensures that Tailwind utility classes are sorted in a canonical order. This makes Tailwind usage more consistent and easier to read, particularly in components with many utility classes.

3.8 Chapter Summary

3.8.1 Main Reasons for the Chosen Stack

The technology stack described in this chapter was selected to serve three main goals: a productive and maintainable full-stack development experience, a capable and interactive frontend that can support the platform's learning-focused features, and a practical AI integration that fits inside the existing application architecture rather than beside it.

Laravel 12 with Inertia.js and Vue 3 covers the first two goals well. Laravel provides a mature, comprehensive backend framework that handles routing, data access, security, and testing without requiring external integrations for each concern. Inertia.js allows the Vue frontend to deliver a modern SPA experience while keeping routing and data authority on the server side. Vue 3 with the Composition API and Tailwind CSS 4 provides a responsive, themeable interface layer.

The Laravel AI SDK covers the third goal. It brings AI capabilities into the same codebase using the same patterns, making it possible to add focused AI assistance without a separate AI service architecture. Structured JSON outputs from the AI agents make results reliable and easy to integrate.

Supporting libraries fill specific roles without adding unnecessary complexity. TipTap handles rich-text editing, Unovis handles dashboard visualizations, motion-v and OGL handle the more interactive and visual parts of the interface, and Reka UI with shadcn-vue component architecture provides a solid, accessible UI foundation.

3.8.2 Transition to the Implementation Chapters

With the technology stack established, the following chapters focus on the actual implementation of the platform's features. Chapter 4 covers the basic application structure and interface while Chapter 5 addresses the AI integration in more depth, covering how each agent class is implemented and how AI outputs are used within the application.

Chapter 4: Core Platform Implementation

4.1 Introduction to the Implementation

This chapter describes the implementation of the main modules of the lifelong learning with adaptive review and AI assistance, platform. First, it explains the basic access layer, and then moves on to course creation, how students view the courses, the assessment process, and finally the spaced-repetition review system. For each part, we explain its structure, how it works, and how it connects to the rest of the application.

4.1.1 Scope of the Implemented System

This platform is a full-stack web application built for creating, delivering, and testing educational content. Its main features include user management, course creation, student enrollment, progress tracking, quizzes, and a flashcard system that uses a spaced-repetition algorithm.

There are two main types of users: content creators and students, and each group has its own interface. The system works completely in a web browser, so users do not need to install any software. When navigating between pages, the application updates without a full page reload, keeping the application state intact. All data is stored in a relational database, and the backend does not have a public API, it only communicates directly with the frontend.

This chapter does not cover deployment, performance optimization, or the advanced analytics described in Chapter 5. Instead, it focuses on the core user features: access control, content management, student engagement, and the assessment process.

4.1.2 Main User Roles

The platform defines two primary user roles (Figure 4.1). The first is the “creator” role, which encompasses all users who create and publish educational content. A creator can create courses, structure them into ordered lessons, create lesson content using a rich text editor, attach flashcards to individual lessons, and define quiz questions for assessment. The second is the “student” role, which encompasses users who discover and enroll in published courses, progress through lesson content, complete quizzes, and engage with the spaced-repetition review system.

In the current implementation, the two roles are not mutually exclusive: a registered user may function as both a creator and a student within the same account. Role-specific views are accessed through shared navigation, and the system presents the appropriate interface based on the context of the action being performed.

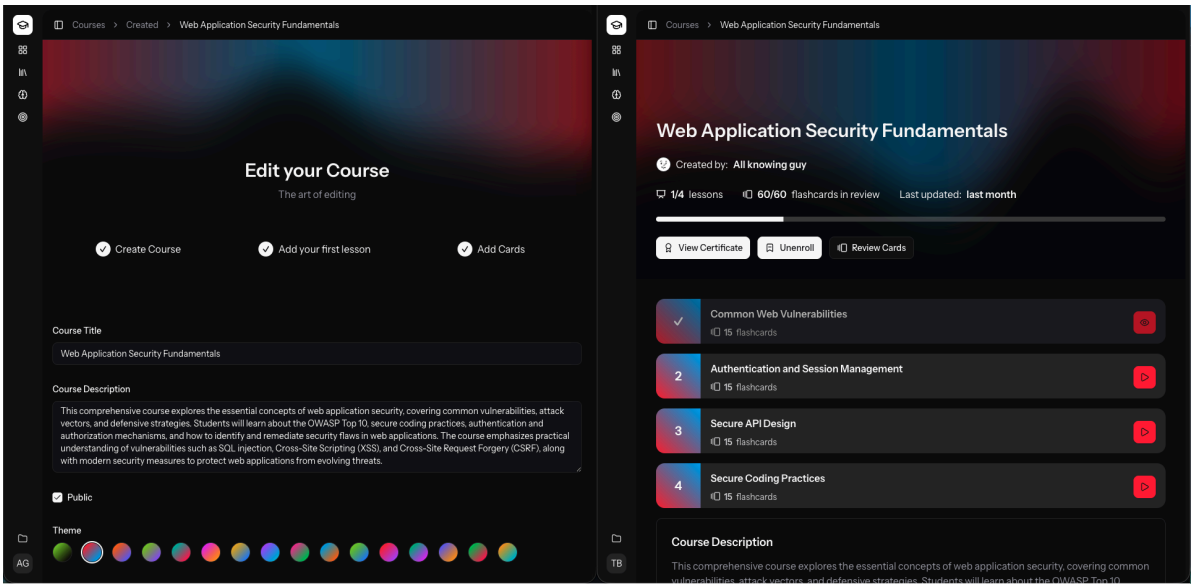

Figure 4.1 The two main perspectives of the platform: (left) course authoring interface for creators, (right) lesson view for students.

4.1.3 Organization of the Implementation Discussion

The rest of this chapter follows the order a user would actually experience the system. Section 4.2 gives an overview of the modules and the main tasks for each role. Section 4.3 explains user login, registration, email verification, and the main navigation menu. Sections 4.4 and 4.5 cover how creators make courses, lessons, and flashcards. Section 4.6 describes how students find and join courses, while Section 4.7 explains how they view lessons and track their progress. Section 4.8 looks at creating, taking, and scoring quizzes. Finally, Section 4.9 covers the review system, including how the flashcard queue works, the swipe controls, and the spaced-repetition algorithm. Each section focuses on what the user sees and does, mentioning the technical background only when it helps explain how a feature works.

4.2 Functional Overview of the Application

Before looking closely at each module, it helps to understand how the platform works as a whole. The application is made up of several connected parts, each handling a specific task. How these parts work together shapes the entire teaching and learning experience for both creators and students.

4.2.1 Main System Modules

The platform is organized into seven main modules:

- **Authentication and access:** Handles user accounts, logins, and sessions.
- **Course and lesson creation:** Allows creators to build and organize their educational content.
- **Flashcard creation:** Lets creators attach review cards directly to lessons.
- **Marketplace and enrollment:** Helps students discover and join published courses.
- **Progress tracking:** Monitors how students advance through the lesson material.
- **Quizzes:** Supports the creation and grading of tests.
- **Spaced-repetition review:** Manages when and how flashcards are shown to students for review.

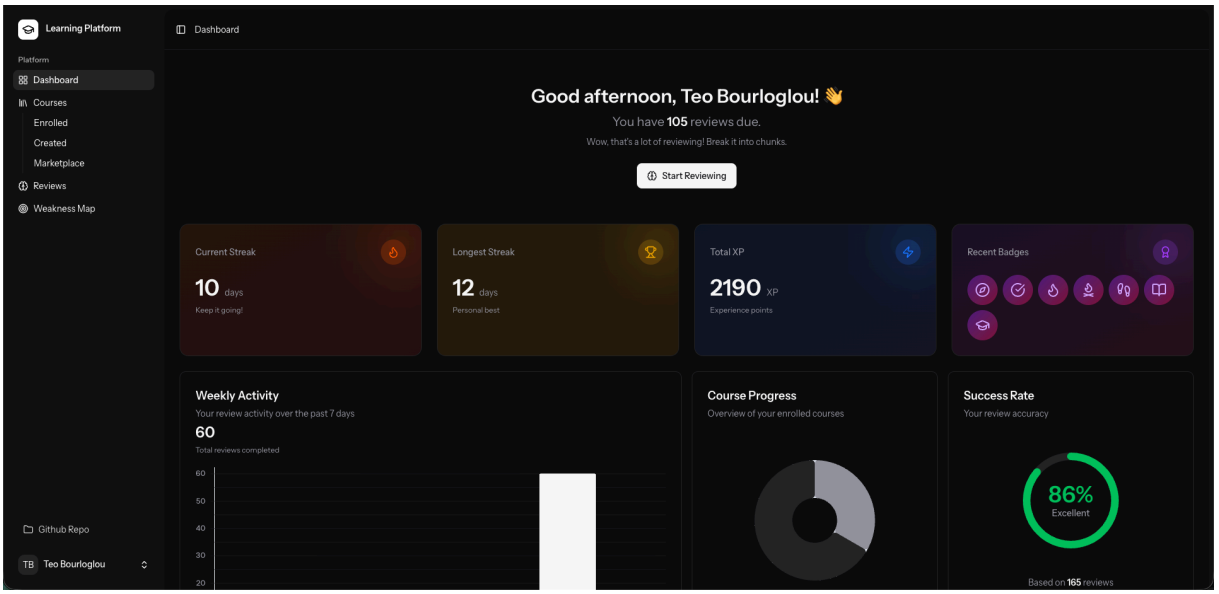

Figure 4.2 Main application shell after login, showing sidebar navigation and available modules.

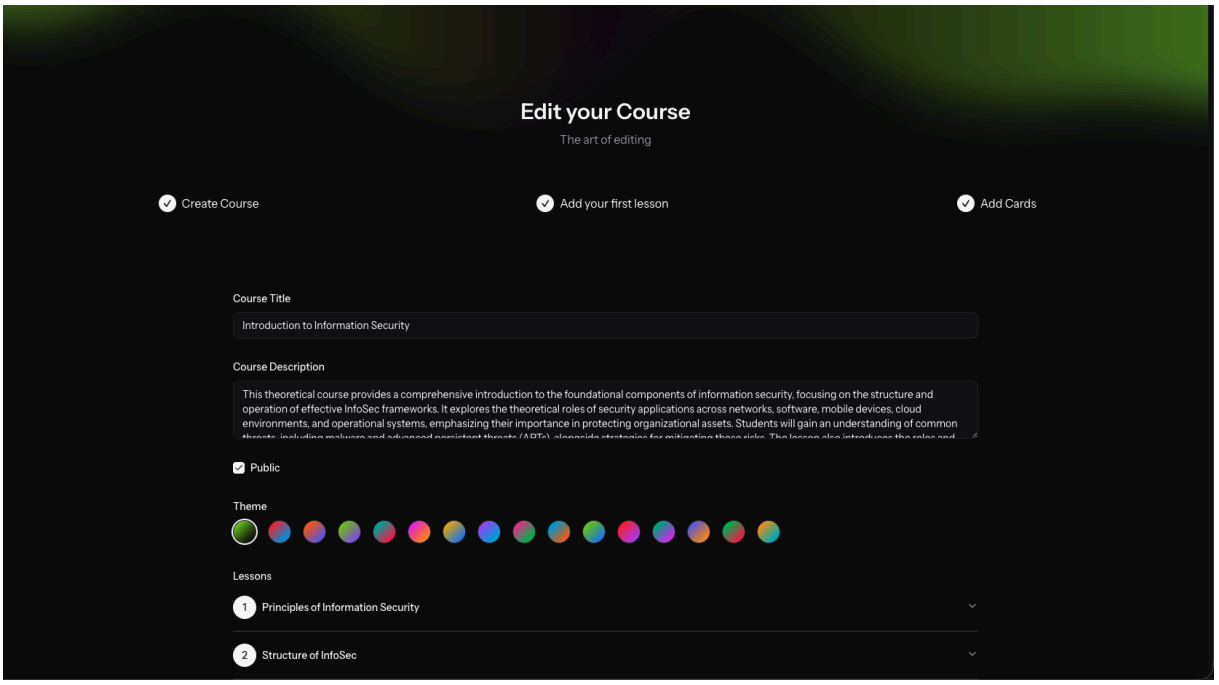

Figure 4.3 Creator workflow: Course editor page, highlighting the tools available to content creators.

These modules do not work in isolation. For example, flashcards depend on lessons, the review system relies on progress tracking, and quiz scores feed directly into the Weakness Map (described in Chapter 5). These connections are built directly into the database design and the system's access permissions.

4.2.2 Creator-Side Workflow Overview

The creator's workflow (Figure 4.3) starts by setting a course title, description, and category. Once the course is created, the creator adds a sequence of lessons, each with its own text and media. Inside each lesson, the creator can attach flashcards (simple question-and-answer pairs for future review) and create a quiz with multiple-choice questions. When the course is finished, the creator publishes it, making it visible in the marketplace for students to join.

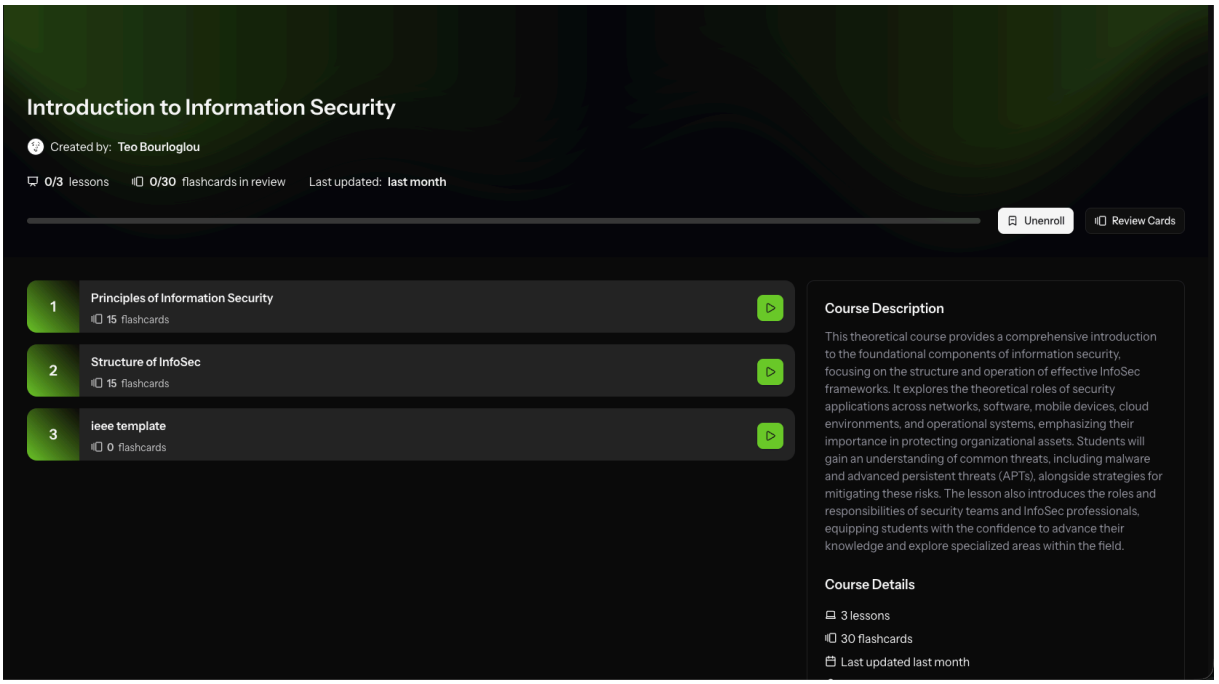

Figure 4.4 Student workflow: Course page, illustrating the course interface for students.

Throughout this process, the system saves changes without reloading the page. Any errors or feedback appear directly next to the input fields, ensuring the creator's work is never interrupted.

4.2.3 Student-Side Workflow Overview

For students, the process starts in the marketplace, where they can browse and search for courses by title. When they select a course, they see a details page (Figure 4.4) with the description and a list of lessons. Enrolling gives them access to the content. The student goes through the lessons in order, marking each one as complete when they finish.

Completing a lesson unlocks its flashcards, adding them to the student's personal review queue, and also unlocks any quizzes for that lesson. When a student takes a quiz, the system records whether they pass or fail.The flashcard review system works separately from the individual courses. Flashcards from all enrolled courses gather in one place, allowing the student to review them at any time.

Finally, the student's overall course progress is shown as a percentage, which updates instantly whenever they mark a lesson as complete or incomplete.

4.2.4 Progress and State Flow Across Modules

Some data is shared across different parts of the system and needs to be updated together. For example, when a student finishes a lesson, it triggers three things: it updates their course progress, unlocks the lesson's flashcards, and allows them to take the quiz. The system handles all of these changes at the exact same time. The backend processes everything in a single step before updating the user's screen.

Also, quiz results are permanently saved rather than just shown once. Every time a student takes a quiz, the system saves the attempt and shows their latest score on the lesson page. Even failed attempts are kept in the database to be used by the analytics tools described in Chapter 5. Finally, the flashcard review queue runs automatically based on time. When a user starts a review, the system simply checks which cards are scheduled for today.

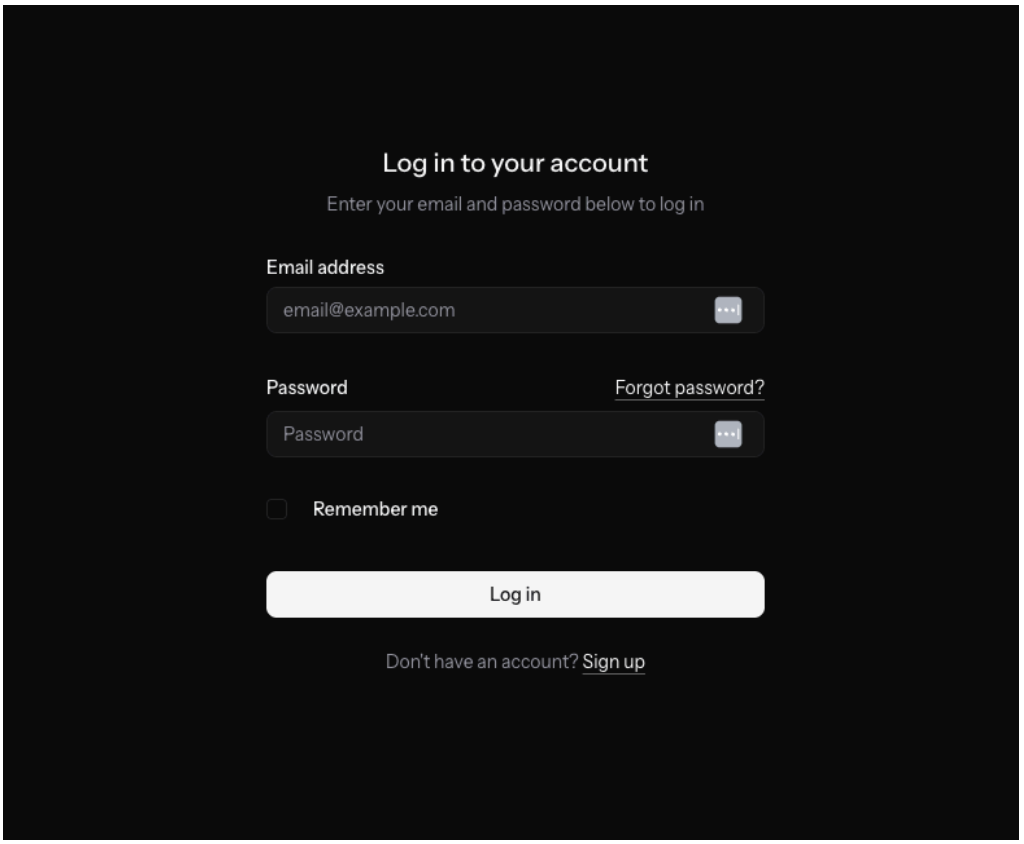

Figure 4.5 Registration screen, showing how new users join the platform.

4.3 Authentication, Access, and Application Shell

Access to the platform is managed by a standard authentication system. This handles user registration, logging in (Figure 4.5), email verification, and keeping users securely signed in. This section explains each part of the login process, as well as the main navigation menu that users see after they successfully log in.

4.3.1 Registration and Login

To register, users fill out a form with their display name, email address, and password. Each account must have a unique email. If a user tries to sign up with an email that is already registered, the form shows an error message. For security, passwords are never stored as plain text, they are hashed using bcrypt function. Once the form is submitted, the system creates the account and sends a verification email.

To log in, users enter their email and password. The system checks these details, and if they are correct, the user is logged in and taken to the dashboard. If they are wrong, an error message appears. Following standard security practices, this message does not reveal whether the email or the password was incorrect.

If a user forgets their password, they can request a reset link sent to their email. This link contains a secure, temporary token. Clicking it takes the user to a form where they can enter and confirm a new password. Once updated, the system saves the new password and securely refreshes their session.

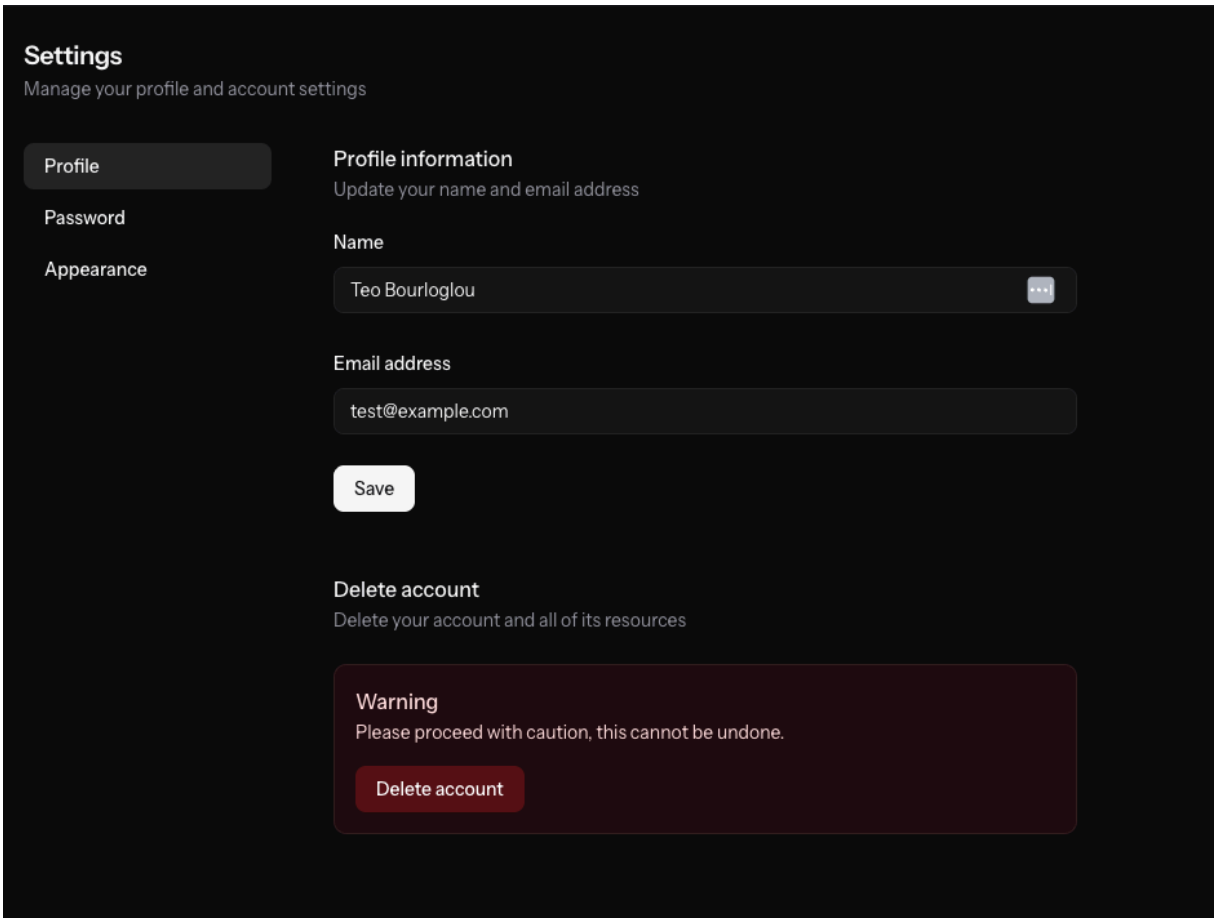

Figure 4.6 Account management page, demonstrating user options.

4.3.2 Verified Access and Account Management

After registering, new users see a message asking them to verify their email. The system blocks unverified users from accessing the main parts of the platform. The verification email includes a secure link. Clicking it confirms the email in the database and gives the user full access.

If a user later changes their email address in their settings, their verified status is removed, and they must verify the new address. This ensures the system always has a valid email for every account.

Users can manage their accounts directly from their profile page. Here, they can update their display name, email, or password. To change a password, the user must first enter their current one. This simple security measure prevents someone else from taking over the account if the user leaves their device unlocked.

4.3.3 Authenticated Navigation and Page Structure

Once users log in and verify their email, they see the main application layout, which stays the same across all pages. This layout has two main parts: a sidebar menu and a top header.

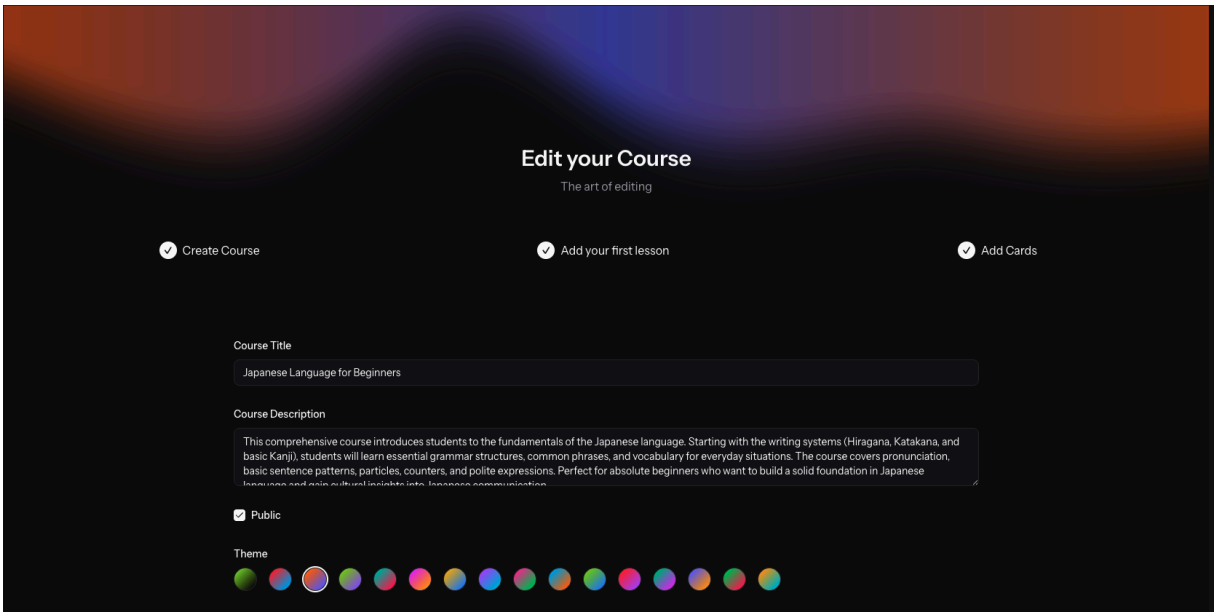

Figure 4.7 Course editor interface, with fields for title, description, theme, and visibility controls.

The sidebar includes links to the main sections of the app: the Dashboard, Courses (created, enrolled and courses marketplace), Flashcard Reviews, and the Weakness Map. This menu stays on the screen and does not reload when the user clicks on different links. The header at the top shows the user's name and profile picture. On mobile phones, the sidebar hides behind a menu button so the app fits perfectly on smaller screens.

Overall, the page structure is very consistent. The sidebar stays fixed on the left, and the main content appears on the right. When a user navigates to a new section, only the main content changes, updating smoothly without a full browser reload.

4.4 Course and Lesson Authoring

The authoring system provides the tools creators need to build their educational content (Figure 4.7). In this system, courses are the main category, and lessons are the specific units of content within each course. This section explains how to create and edit both of these, as well as how to arrange the order of lessons and save changes.

4.4.1 Creating and Editing Courses

To create a new course, creators fill out a form with a title, description, and category. For the category, they can choose from a list or type their own, allowing them to classify their content however they like. When the form is submitted, the course is saved as a 'draft.' In this state, the course is hidden from students; the creator must manually publish it before it appears in the marketplace.

After a course is created, all its details can be edited. The editing form shows the current title, description, and category, and the creator can change any of them. Saving the form updates the course information immediately. The option to change a course from a draft to published is also found in this editing section.

Creators can find all their work in the 'My Courses' section. From there, they can choose any course to edit, view the list of lessons, or publish their content.

Figure 4.8 Lesson organization view, showing the list of lessons within a course.

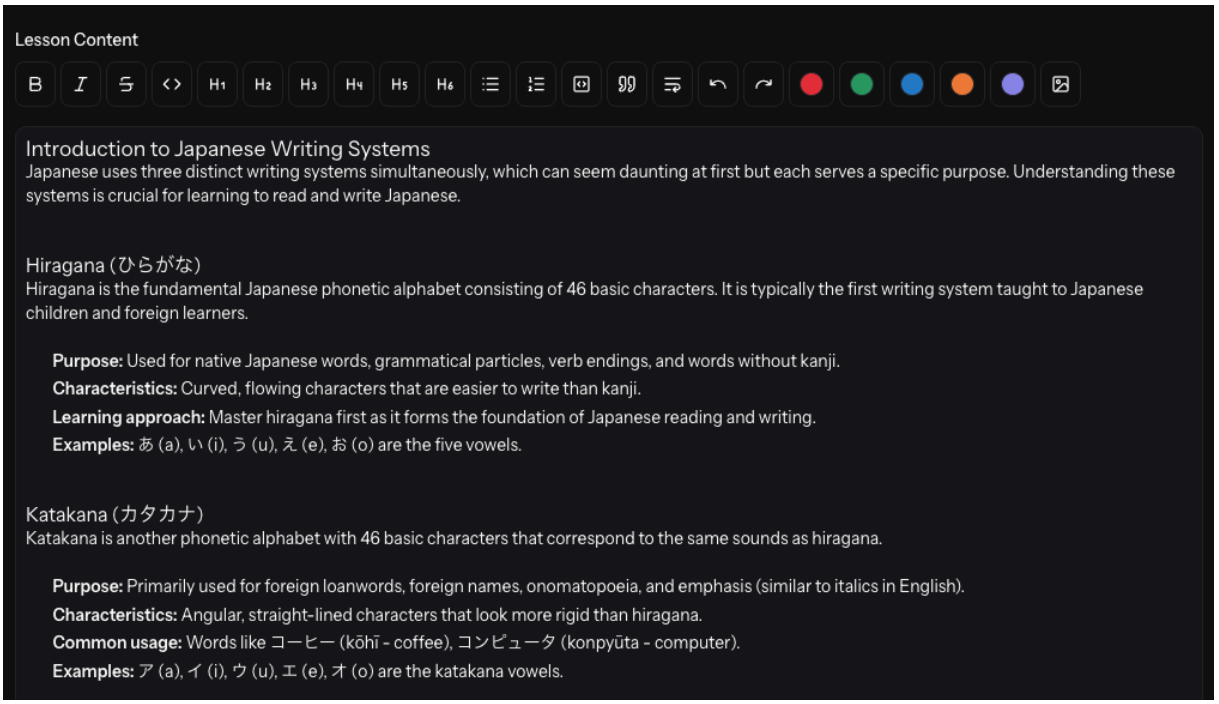

Figure 4.9 Lesson content editor with rich-text area.

4.4.2 Organizing Lessons Within a Course

Lessons are created within a course and are the main building blocks of the content. Each lesson is part of one specific course, but a single course can have as many lessons as the creator wants. When adding a new lesson, the creator provides a title and initial text. Each lesson is also given a position to decide where it sits in the list.

The lesson list (Figure 4.8) shows the creator exactly how the lessons are ordered. This is the same order that students will see once they enroll. From this view, the creator can check all the lessons, open the editor for any of them, and see the overall structure of the course at a glance.

New lessons are added through a form in the course management area. When a new lesson is created, it is automatically added to the end of the list. The creator can change this order later using the tools described in Section 4.4.4.

Figure 4.10 Lesson reordering controls.

4.4.3 Editing Lesson Content

Each lesson has its own editing page with a title field and a rich text editor built using the Tiptap framework (Figure 4.9). This editor allows creators to format their text easily, including headings, bold and italic text, lists, and blockquotes, as well as code blocks. The content is saved as HTML and looks exactly the same when shown to students.

When a creator opens an existing lesson, the current title and text are loaded automatically. They can make any changes they like, and when they click save, the system updates the database in the background. No page reload is needed, and if there are any errors, they appear directly on the screen without interrupting the work. This page also acts as a hub: from here, creators can access the flashcard system (Section 4.5) and the quiz editor (Section 4.8), keeping all the tools for one lesson in one place.

4.4.4 Managing Lesson Order and Persistence

The order of lessons within a course is managed through a set of reordering buttons in the course management view (Figure 4.10). Each lesson in the list has controls that allow the creator to move it up or down in the sequence. These changes happen instantly on the screen, allowing the creator to organize the course structure visually.

Unlike other changes that might happen automatically, these new positions are only saved to the database when the creator clicks the 'Save' button. This sends the entire updated list to the backend, which processes the changes. This ensures that the new order is saved correctly and prevents any errors. Once the save is successful, the page confirms the update without needing a full reload.

As with other parts of the authoring system, this process uses Inertia.js. This keeps the experience smooth and fast, while still allowing the server to validate the data and show any errors directly on the page.

4.5 Flashcard Authoring

Flashcards are simple study items linked to individual lessons. They are the main source of content for the spaced-repetition review system. A lesson can have many flashcards or none at all, and creators manage them directly on the lesson's editing page.

4.5.1 Creating Lesson Flashcards

The flashcard management screen is accessed directly from the lesson editor. This interface displays the existing cards for the lesson alongside a creation form (Figure 4.11). To generate a card, the creator provides the 'front' text (the question or term) and the 'back' text (the answer or explanation).

Figure 4.11 Flashcard grid within the lesson editor.

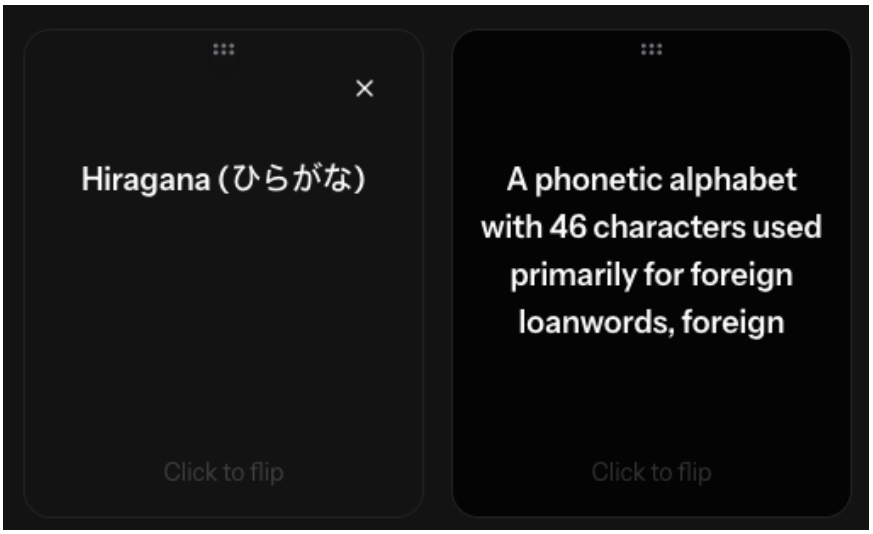

Figure 4.12 Editing a flashcard: close-up of front and back content fields.

The system allows for the creation of multiple cards in sequence without navigating away from the page. Upon each submission, the new card is added to the lesson's collection and the form fields are cleared for the next entry. These cards do not appear in a student's review queue immediately. Instead, as explained in Section 4.7.3, they only enter the queue once a student enrolled in the course has completed the corresponding lesson.

4.5.2 Editing Front and Back Content

Existing flashcards can be edited directly on the flashcard management screen (Figure 4.12). When a creator chooses to edit a card, the static text is replaced by input fields containing the current front and back content. In the current version of the platform, these fields only support plain text, with no rich-text formatting.

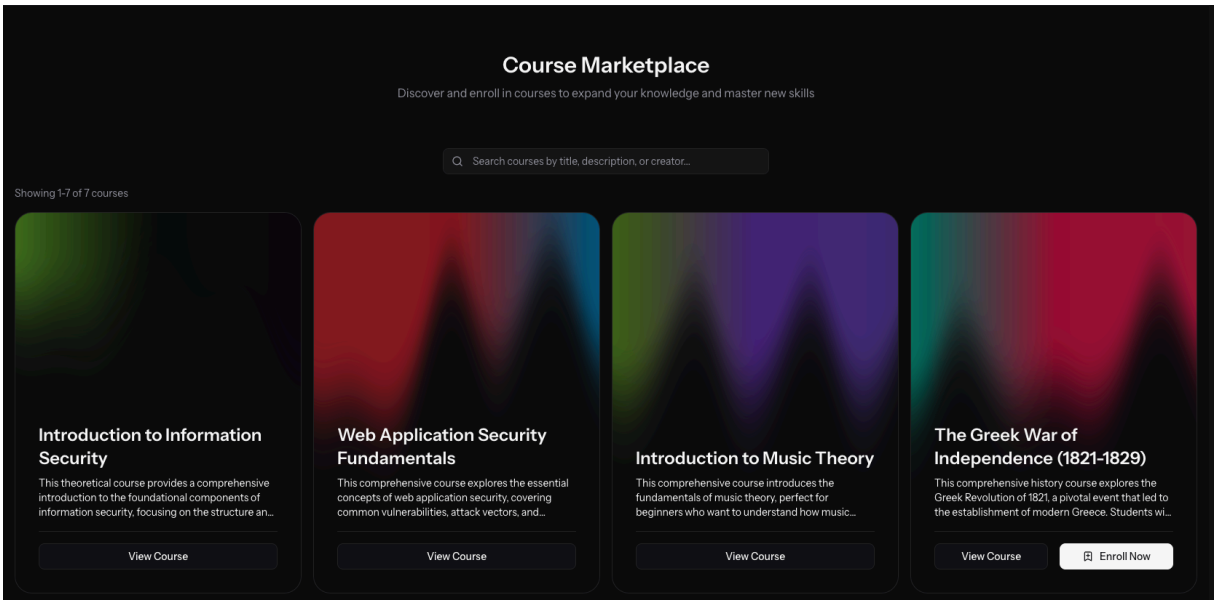

Figure 4.13 Course marketplace.

Once the changes are submitted, the system updates the database and the screen reflects the new text immediately without a page reload. If the creator submits an empty field or invalid data, the system displays an error message on the screen and keeps the editor open until the input is corrected. This simple front-and-back structure was an intentional design choice to follow the traditional flashcard format, where a single question or stimulus is paired with a single answer.

4.5.3 Reordering and Removing Cards

The order of flashcards within a lesson is managed through a drag-and-drop interface. The cards are displayed in a sortable list, allowing the creator to click and move any card to a new position. When a card is dropped into its new place, the system sends a request to the backend to update the positions of all affected cards at once. This sequence is important because it determines the order in which the cards will appear during a student's review session.

Creators can also remove individual cards by selecting the delete button. In the current implementation, deletion happens immediately without a confirmation prompt. There is no undo feature, so if a card is removed by mistake, it must be manually recreated. To maintain system consistency, deleting a card also removes it from any student's review queue where it was previously active.

4.6 Course Discovery and Enrollment

The course discovery and enrollment system connects the creator and student workflows. It provides the tools students use to find courses that interest them and to enroll in them to access the content.

4.6.1 Marketplace Browsing and Search

The course marketplace (Figure 4.13) is a page where all logged-in users can browse available content. Each published course is shown as a card containing the title, the creator’s name, the category, and the total number of lessons. Courses that are still in 'draft' status are hidden from this list.

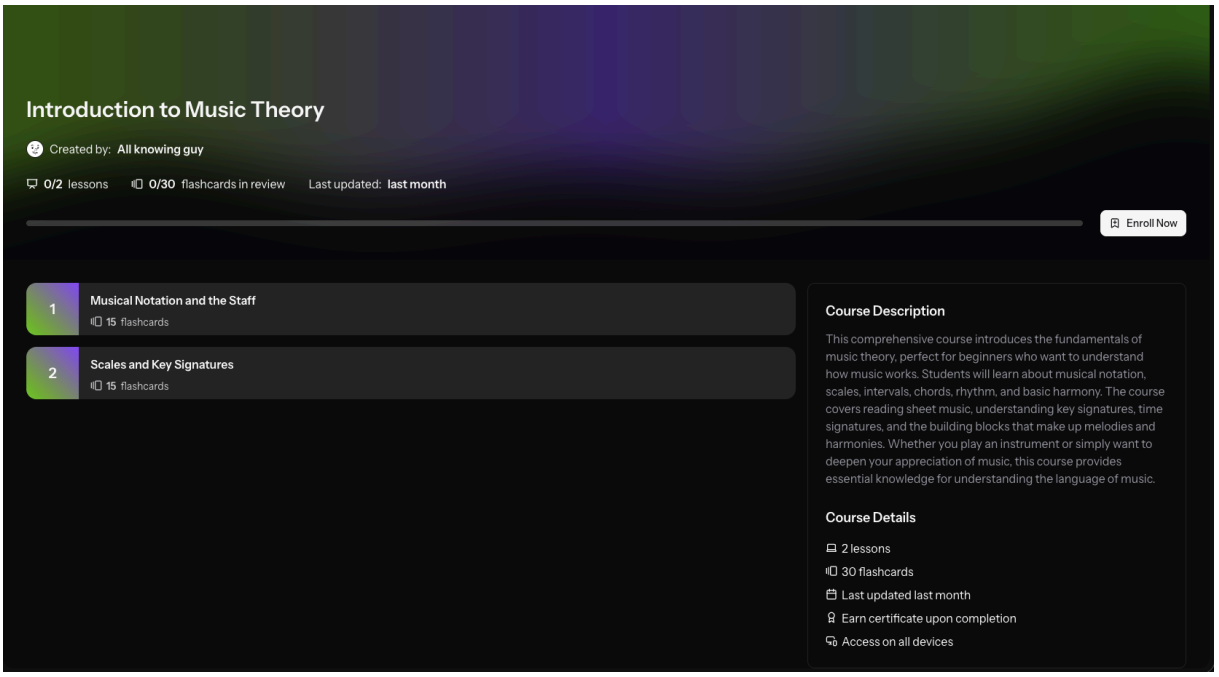

Figure 4.14 Course overview page before enrollment.

At the top of the page, a search bar allows students to filter courses by their title. This search happens instantly within the browser; as the student types, the list of courses updates immediately without needing to wait for the server to respond. This makes the search experience feel very fast and responsive.

The marketplace is the main place for students to find new material. At this stage, they cannot see the actual lesson content yet, they must first click on a course to see more details before they decide to enroll.

4.6.2 Course Overview Page

When a student selects a course from the marketplace, they are taken to the course overview page (Figure 4.14). This page shows the full description and a list of lesson titles. At this stage, the actual content of the lessons is hidden, ensuring that it is only accessible after enrollment. This allows the student to see the overall structure and topics of the course before deciding to join.

A clear enrollment button is shown on the page. If the student is not yet enrolled, they will see an ‘Enroll' button. If they are already enrolled, the button changes to an ‘Unenroll’ option, and they will see links that take them directly into the course lessons.

4.6.3 Enrollment and Unenrollment

Clicking the ‘Enroll’ button triggers a request to the backend that creates an enrollment record in the database, associating the authenticated user with the selected course. The enrollment record serves as the gate for access to all content within the course: lesson content, quizzes, and flashcard review material are accessible only to users with a corresponding enrollment record. After the enrollment record is created, the student is redirected to their enrolled course view, from which they can begin accessing lesson content.

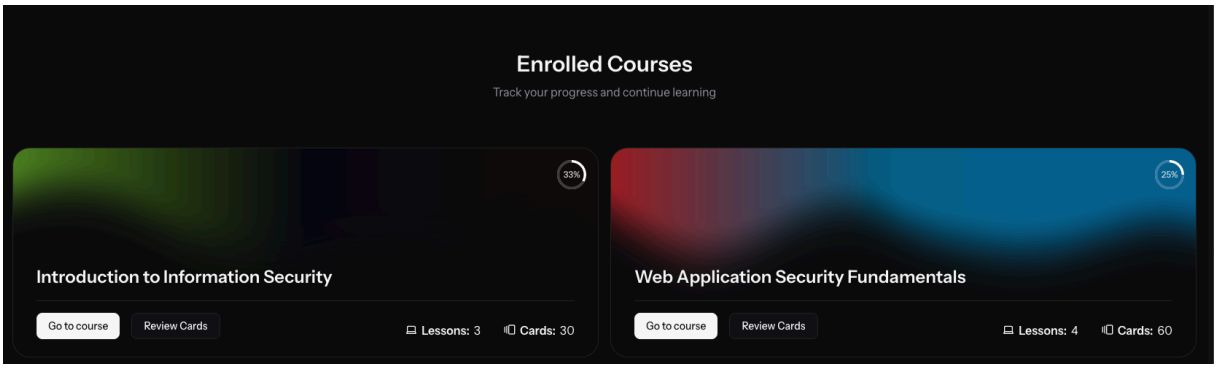

Figure 4.15 Enrolled courses page.

Unenrollment is performed by clicking the ‘Unenroll’ button present on the course overview page when the student is already enrolled. This action deletes the enrollment record from the database. As a consequence of the deletion, the student immediately loses access to all lesson content and review material associated with the course. Any flashcards from the course that were present in the student's review queue are also removed. This action cannot be undone through the application interface; re-enrolment would create a new enrolment record but would not restore previously accumulated review history.

4.6.4 Student Access After Enrollment

Once a student enrolls, the course page changes from the public overview to a private view that shows more details. Each lesson in the list now includes a status icon showing if it has been marked as finished. Additionally, the overall course progress is shown as a percentage at the top of the page.

From this view, the student can click on any lesson to start studying. Even though the lessons are listed in a specific order, the system does not force the student to follow that sequence. They can open any lesson at any time. However, the order still reflects how the creator intended the course to be taught. Access to these lessons depends on the student staying enrolled. If they unenroll, they can no longer view the content and will be redirected away from the page.

4.7 Lesson Consumption and Progress Tracking

The lesson system manages how students view course content and how the platform tracks their progress. It also controls how finishing a lesson opens up other features, specifically giving the student access to flashcards and allowing them to take quizzes.

4.7.1 Reading Lesson Content

The lesson page is the main area where enrolled students view course material. It displays the lesson title as a heading with the rich-text content formatted below. Since the content was created using the Tiptap editor, the page displays it exactly as the creator intended, preserving all formatting, lists, and code blocks.

The lesson content is the central focus of this page (Figure 4.16), while the sidebar and header stay in place to provide easy access to other parts of the platform.

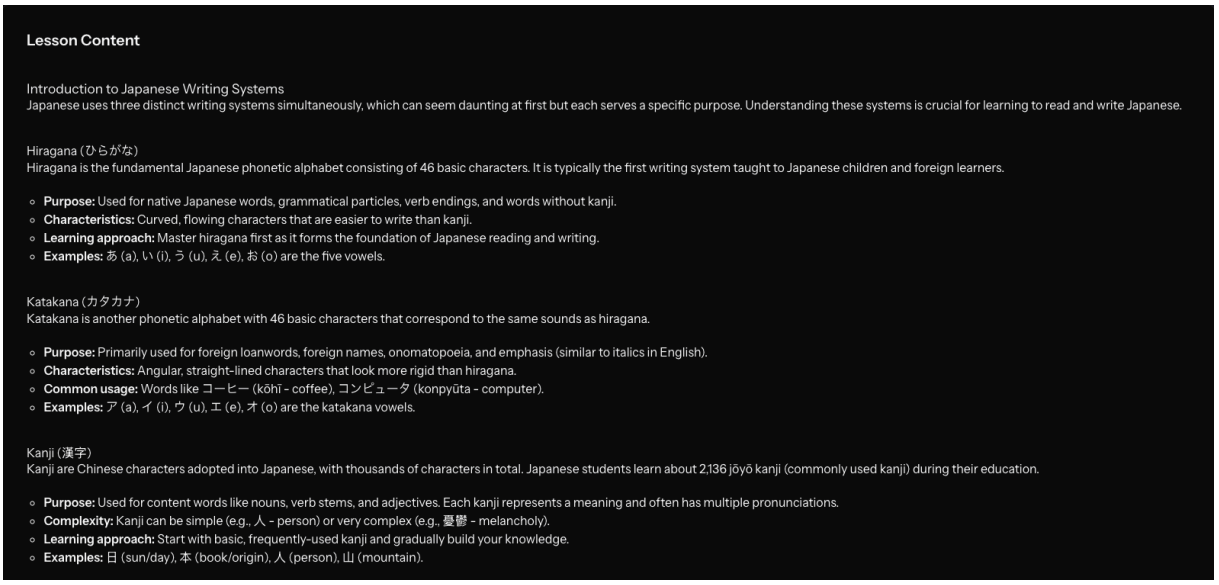

Figure 4.16 Student lesson page with instructional content.

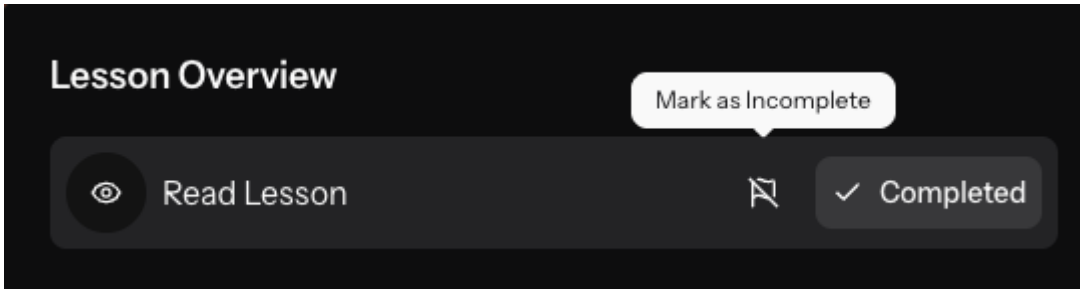

Figure 4.17 Lesson completion control.

4.7.2 Marking Lessons as Complete or Incomplete

At the top of every lesson, enrolled students find a completion toggle (Figure 4.17). If a lesson has not been finished, the button allows the student to mark it as complete. If it is already finished, the same button can be used to mark it as incomplete. This allows students to manage their own progress if they decide they need to revisit the material before moving on.

Clicking this button sends a request to the backend to update the user's progress record for that specific lesson. If a record does not exist yet, the system creates one, otherwise, it simply switches the status. The change is reflected on the page immediately without a reload.

Marking a lesson as complete does more than just update the progress bar. It also unlocks the associated flashcards for review (Section 4.7.3) and updates the overall course completion percentage (Section 4.7.4). If a student marks a lesson as incomplete again, these effects are reversed, and the cards are removed from the review system.

4.7.3 Unlocking Flashcards Through Lesson Progression

Flashcards linked to a lesson only become available for review once a student marks that lesson as complete (Figure 4.18). When this happens, the backend identifies all flashcards for that lesson and adds them to the student's personal review queue. Each card is scheduled for review on the current date, meaning they appear in the review system immediately.

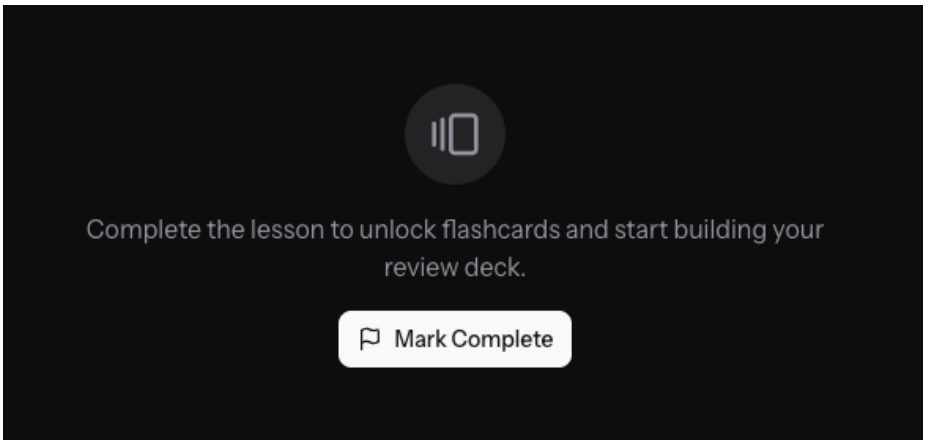

Figure 4.18 Flashcards are locked before lesson completion.

If a student changes a lesson back to incomplete, the system reverses this process and removes the flashcards from the review queue. This ensures the review queue always matches the student's current progress in the course. This design creates a clear incentive: students must finish reading the lesson material before they can access the tools for long-term memorization.

4.7.4 Updating Course-Level Progress

A progress bar is visible at the top of the page to show the student's overall standing in the course. This bar shows progress as a percentage, and is updated in real-time, helping the student keep track of their progress without needing to go back to the main course page. The progress percentage is determined by the following formula:

$$progress = \frac{number\ of\ completed\ lessons}{total\ number\ of\ lessons\ in\ the\ course} \times 100 \quad (4.1)$$

4.8 Quiz Creation and Execution

The quiz system allows creators to build assessments for each lesson and tracks student scores across multiple attempts. This section explains the data structure behind the quizzes, the interface used to create them, and the process students follow when taking a test. It also details the logic used for scoring, how the system determines a passing grade, and the rules for retaking a quiz.

4.8.1 Creating and Editing Quizzes

Each lesson on the platform can have only one quiz. Creators start the process from the lesson page, where they can access the quiz editor. This editor allows them to build the list of questions that will make up the assessment (Figure 4.19).

If a lesson does not have a quiz yet, the creator starts with an empty list and adds questions one by one. Each question is created using a form where the creator enters the question text and as many answer choices as they need. After saving, the quiz can be modified at any time. Creators can add new questions, edit existing ones, or remove them entirely. These changes are saved through the standard form system and take effect immediately.

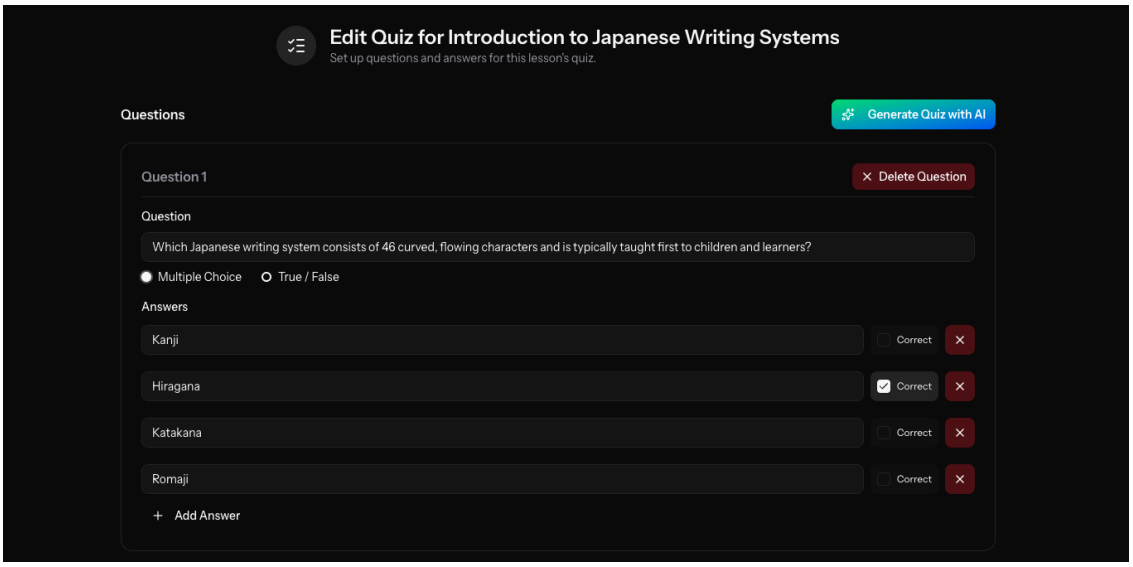

Figure 4.19 Quiz editor overview, with questions and answers list.

In the database, the quiz is linked to its lesson through a specific ID. Because each lesson is limited to a single quiz, the button to create a new quiz disappears once one is made, and the creator is instead given the option to edit the existing one.

4.8.2 Question Types and Answer Structures

Each quiz question consists of the question text and its associated answer options. The platform supports two specific question types: Multiple Select and True/False (Figure 4.20). The system determines which type to display based on the answers provided by the creator.

Multiple Select questions allow for one or more correct answers. These are rendered with checkbox inputs, enabling the student to select any combination of options they believe to be right. This format is used for most standard questions where the student must identify the correct information from a list.

True/False questions, on the other hand, are designed for simple binary choices. These are presented as a statement where the student must choose between two fixed options. By using these two formats, the platform can handle both complex knowledge checks and quick conceptual validations within the same quiz. The interface updates dynamically for each question to ensure the student sees the appropriate selection method.

4.8.3 Taking Quizzes as a Student

The quiz section displays all questions one by one for the student (Figure 4.21). Each question is shown with its text followed by the possible answers. True/False questions and those with only one correct answer use radio buttons, while Multiple Select questions use checkboxes. This choice of input informs the student whether they should select only one option or can choose a combination of several.

Students complete the entire quiz before submitting their answers in a single action at the bottom of the page. The system does not use a question-by-question progression, instead, all selections are captured and sent at once.

When the student submits the quiz, the application sends the data to the backend to be scored. The system then evaluates the answers and immediately returns the result, showing the student their total score and whether they passed or failed the assessment.

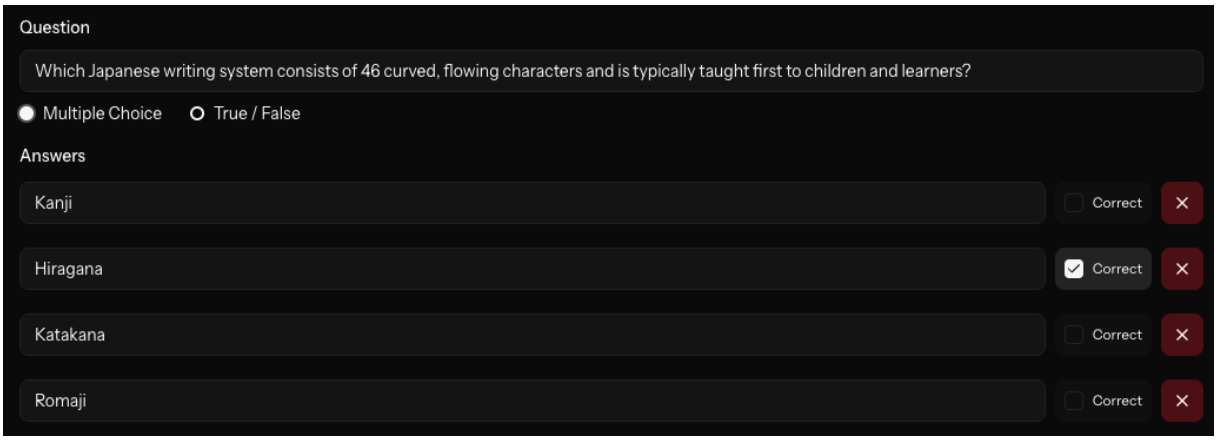

Figure 4.20 Multiple-choice vs. true/false question authoring.

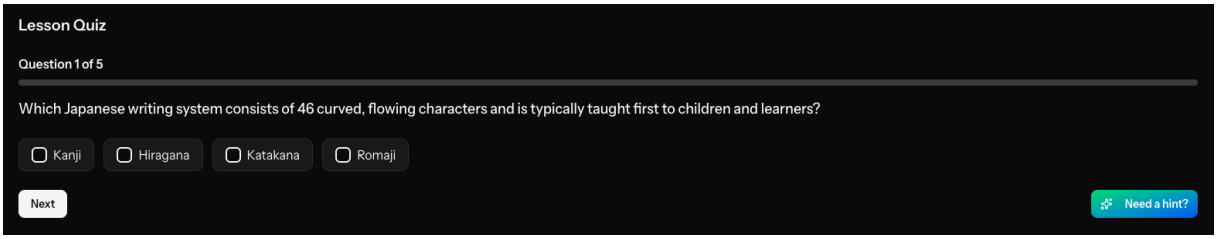

Figure 4.21 Student quiz interface during an attempt.

4.8.4 Scoring, Pass Criteria, and Attempt Persistence

Quiz scoring is handled on the server to ensure accuracy. For each question, the system compares the student's chosen answers with the correct options stored in the database. A question is only marked as correct if the student's selection matches the correct set exactly. The student must pick every correct option and none of the incorrect ones. The platform does not award partial credit. For example, missing one correct choice in a multiple-select question results in the entire question being marked wrong.

The total score is expressed as a percentage using the following formula:

$$score = \frac{number\ of\ correctly\ answered\ questions}{total\ number\ of\ questions} \times 100 \tag{4.2}$$

Whether a student passes a quiz depends on a passing percentage set by the lesson's creator. During the quiz authoring process, the creator can choose the minimum score required to pass. This means that one lesson might require a 60% score to proceed, while another more difficult lesson might require 90%.

Every time a student submits their answers, the system saves a new record of the attempt in the database. This record includes the time of the test, the score, and whether the student met the passing mark set by the creator. Even though the system keeps a permanent history of every try, the student only sees the score of their latest attempt on the lesson page. This prevents the interface from becoming cluttered while still allowing for backend data analysis of student performance over time.

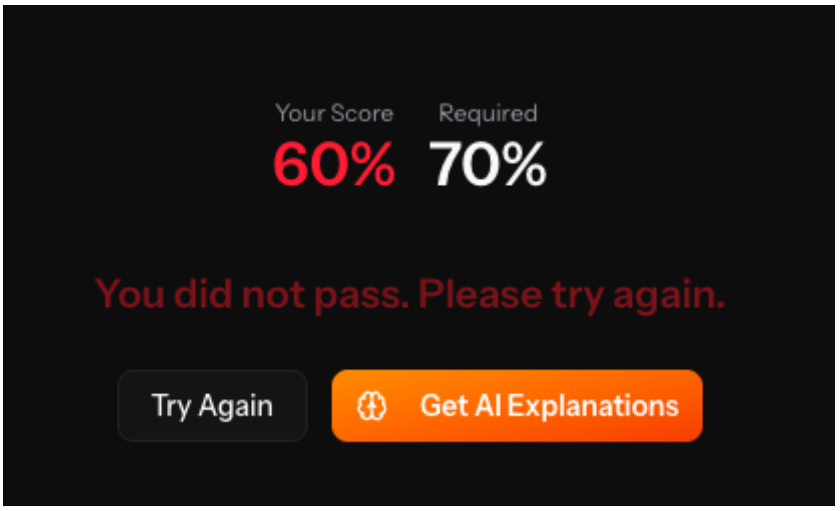

Figure 4.22 Quiz result with score, fail status and retry option.

4.8.5 Retry Flow and Failure Handling

When a quiz attempt falls below the passing threshold set by the creator, the results page displays the score along with a 'Try Again' button (Figure 4.22). Notably, the student is not shown which answers were correct or incorrect. This is a deliberate design choice: by withholding the correct answers, the platform ensures the quiz remains an effective assessment for future attempts. It encourages students to revisit the lesson material rather than simply memorizing the correct sequence of answers.

Clicking the 'Retry' button clears all previous selections and returns the student to the start of the quiz. They can then review the questions and submit a fresh attempt. Each of these tries is saved as a separate record in the database, preserving the full history of the student's progress.

These failed attempts are more than just a record of a mistake. They provide the data for the 'Weakness Map' feature (detailed in Chapter 5). By analyzing patterns in these incorrect answers over time, the system can identify specific topics where a student is struggling. In this way, a failed attempt becomes a valuable piece of data that helps the platform adapt to the student's needs.

4.9 Review System and Spaced Repetition

The review system helps students strengthen their memory of course content through periodic exposure to flashcards. This process is managed by a spaced-repetition scheduling algorithm that organizes a personal review queue for every user. The system allows for various study sessions and delivers cards through an interactive, swipe-based interface. After each card is reviewed, the algorithm updates the scheduling interval based on how well the student remembered the information.

4.9.1 Adding Cards to the Review Queue

Flashcards are introduced into a student's review queue through the lesson completion process detailed in Section 4.7.3. When a lesson is marked as complete, the system identifies all associated flashcards and generates entries in the student's personal queue. Each new entry is scheduled for the current date, ensuring that these cards are immediately available for study in the student's next review session.

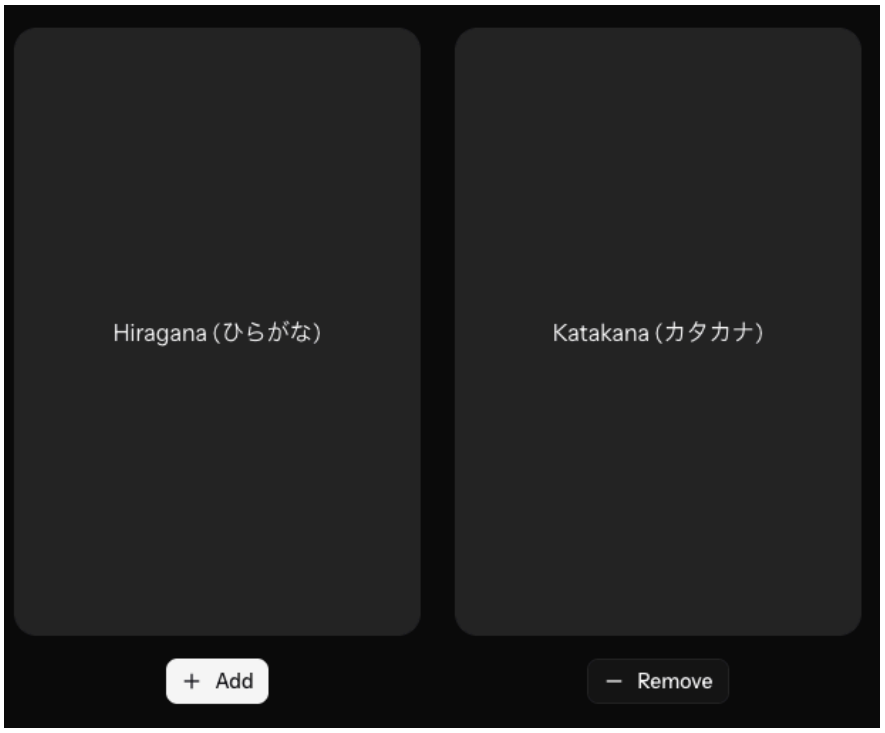

Figure 4.23 Adding a flashcard to the review queue from the lesson page.

Each entry in the review queue tracks four key data points: references to the specific flashcard and student, the current scheduling interval (measured in days), and the next scheduled review date. Upon initial creation, the interval is set to one day and the review date is set to the current date. As the student interacts with the cards, the spaced-repetition algorithm (Section 4.9.4) dynamically updates these values.

A card is removed from the queue if the student reverts a lesson to 'incomplete' or chooses to unenroll from the course. In both scenarios, the corresponding queue entries are deleted from the database, immediately excluding those cards from future review sessions.

4.9.2 Review Session Modes

The Reviews page offers two distinct options for launching a study session, categorized by their scope (Figure 4.24). The 'All Reviews' option aggregates due cards from all courses in which the student is enrolled. Conversely, 'Course Reviews' focuses exclusively on a single course, allowing the student to concentrate their efforts on a specific subject area.

In both modes, the system identifies the cards for the session at the moment of launch by querying the database for entries with a scheduled review date equal to or earlier than the current date. Cards scheduled for the future are excluded, regardless of the chosen scope. This query creates a fixed set of cards for the session.

This dual-mode approach provides flexibility for students who want to practice broadly or prepare for a specific assessment. While the initial filtering process differs, both modes utilize the same review interface and the same underlying scheduling algorithm.

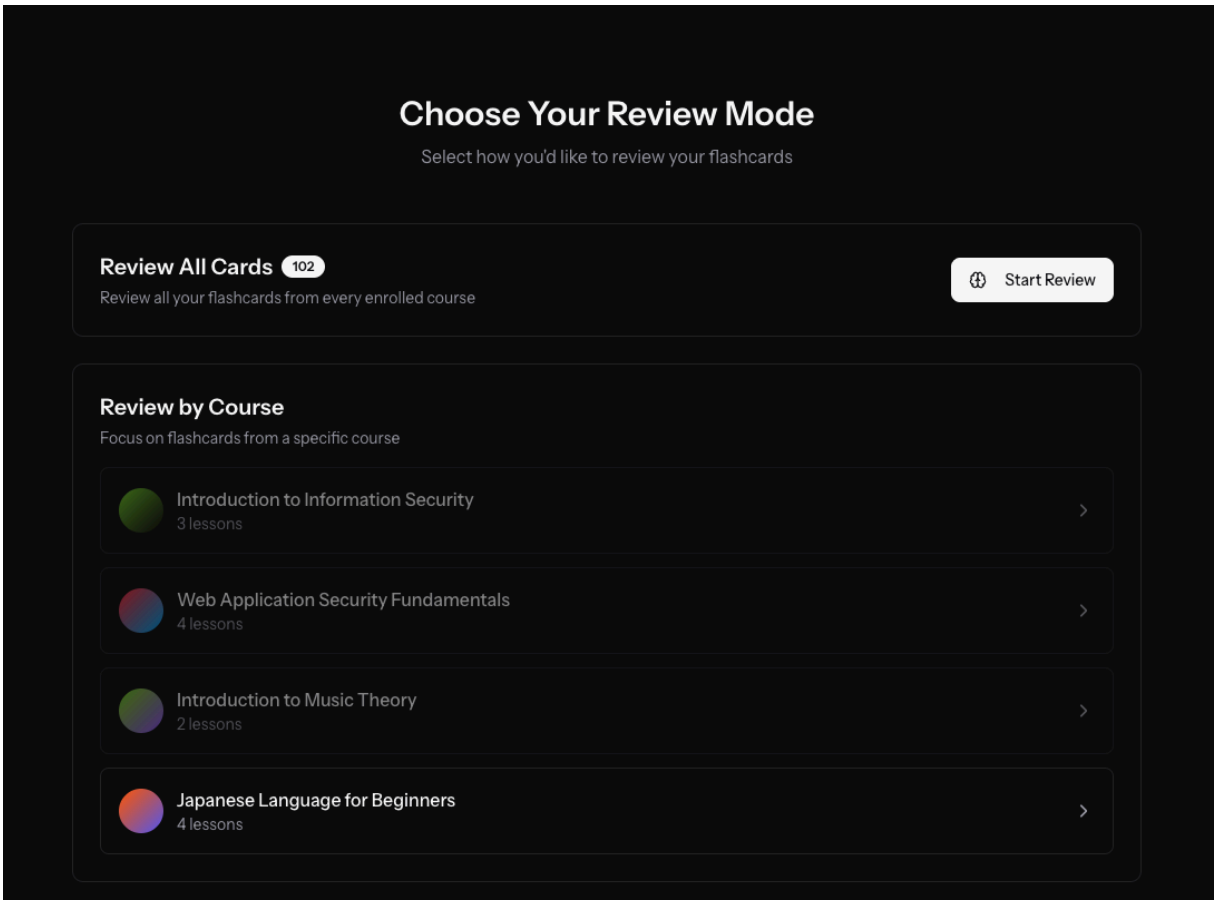

Figure 4.24 Review mode selection screen, with options for all cards, by course, or by lesson.

4.9.3 Swipe-Based Review Interaction

The review session interface is designed to focus on one card at a time. When a session begins, the student is shown the 'front' of the first card, which contains the question or term they must recall. The student then 'flips' the card by tapping it or clicking a button to reveal the answer on the back.

After viewing the answer, the student records their performance using either a swipe gesture or response buttons (Figure 4.25). Swiping the card to the right signals a successful recall, while swiping to the left indicates a failed attempt. Technically, this gesture is managed by a drag-event handler that tracks horizontal movement; once the movement passes a specific threshold, the system registers the response and loads the next card.

This swipe-based model was chosen to make the review process feel intuitive and physically engaging. By using a binary 'correct or incorrect' choice rather than a complex rating scale, the platform simplifies the self-assessment process and keeps the student focused on the material.

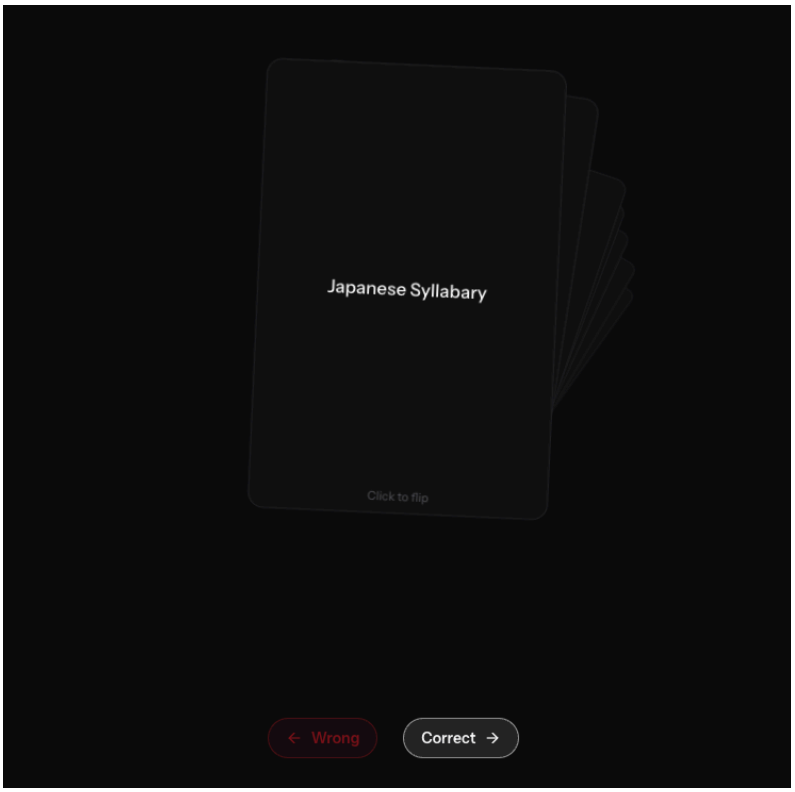

Figure 4.25 Swipe-based review stack during a session.

4.9.4 Interval Updates After Correct and Incorrect Responses

The scheduling algorithm is a simplified variant of the spaced-repetition principle, where the time between reviews increases after success and resets after failure.

When a student records a correct response, the current interval of the card is multiplied by a factor of 2. The next review date is then set by adding this new interval to the current date. For example, a card with a 1-day interval that is answered correctly will be scheduled for review in 2 days. If answered correctly again, the interval increases to 4 days, then 8, 16, and so on, following a geometric progression:

$$I_{n+1} = I_n \times 2$$

$$d_{next} = d_{today} + I_{n+1}$$

where I_n is the interval after the n-th correct review.

(4.3)

If a card is marked as incorrect, the interval is reset to 1 day regardless of its previous value, and the next review is scheduled for the following day. This reset ensures that forgotten material is revisited promptly, preventing it from remaining in a long-term scheduling horizon.

This model is intentionally straightforward. It does not use complex variables like recall latency or the ease-factors found in algorithms like SM-2. This provides predictable behavior and makes the system easier to audit. Every response triggers an immediate database write, ensuring the student's progress is saved even if they close the session early.

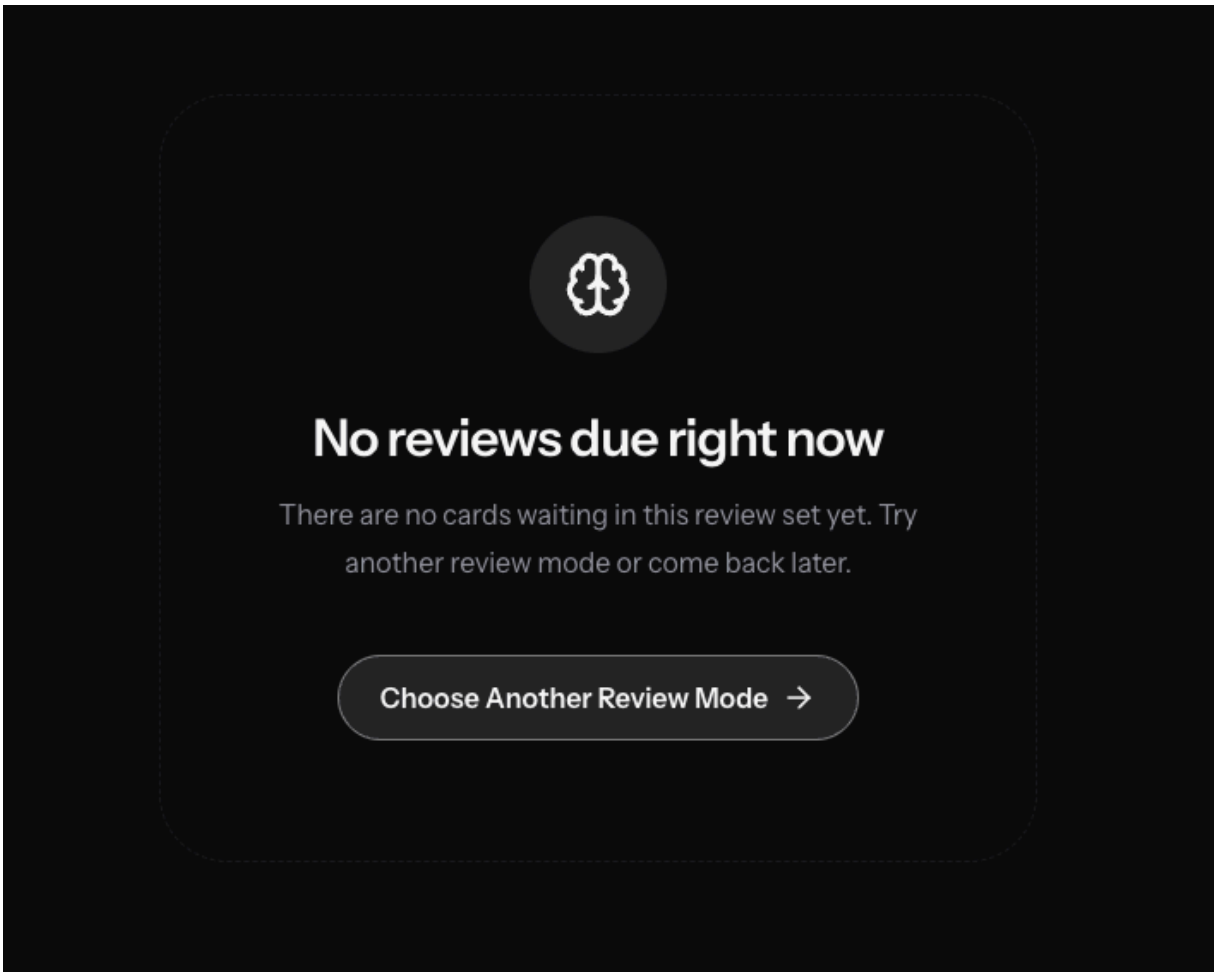

Figure 4.26 Empty-state screen after all reviews are completed.

4.9.5 Empty States and Session Completion

Two terminal states can be reached within the review workflow. The first is the 'empty state,' which occurs when a student initiates a session but no cards within the selected scope are currently due. Rather than launching a session, the system displays an informational screen explaining that no reviews are scheduled for the day. This clearly communicates that the lack of cards is a function of the spaced-repetition algorithm rather than a system error.

The second terminal state is 'session completion,' reached once all cards in the current session snapshot have been reviewed (Figure 4.26). After the final card is processed, the interface transitions to a summary screen, displaying the appropriate message. These two states ensure the review loop closes logically, whether the student has no work to perform or has just finished a study session. By providing clear, informative feedback at these points, the system avoids ambiguous end states and ensures the student is always aware of their current progress.

Chapter 5: Intelligent and Adaptive Features

This chapter explores the intelligent and adaptive features of the platform. It describes how each component is implemented, the way data flows through the system, and how the various subsystems interact to provide a personalized learning experience. The following sections cover AI-assisted tools for content and assessment, adaptive analytics used to find student weaknesses, and gamification elements designed to keep users engaged. Additionally, this chapter details the student dashboard, the certification process, system-wide security and access controls, and the testing suite used to verify the platform's most critical logic.

5.1 AI-Supported Educational Features

The platform integrates OpenAI's GPT-4o model to enhance both content creation and the learning experience. To ensure security and control, all interactions with the OpenAI API are managed server-side through Laravel controllers. The frontend is never allowed to communicate with the API directly. For every AI-powered feature, the system builds a structured prompt, sends it to the model, and then delivers the generated response to the user interface. The following sections detail how each of these AI features is implemented.

5.1.1 Lesson Content Feedback for Creators

Creators can request automated feedback on their lessons directly within the editor. By clicking the 'Get AI Feedback' button, the system converts the lesson's rich-text HTML into plain text and sends it to the backend via an asynchronous request.

The backend then instructs GPT-4o to critique the lesson based on four criteria: language clarity, structural flow, completeness, and pedagogical value. The model provides a detailed response highlighting both strengths and weaknesses. This critique is passed back to the frontend and displayed in a panel below the editor (Figure 5.1).

Because the feedback is never saved to the database, it is entirely ephemeral. Each time the creator clicks the button, a fresh analysis is generated. This approach ensures that the creator always receives guidance based on their most recent edits without cluttering the database with outdated AI responses.

5.1.2 AI-Assisted Quiz Generation

To streamline the creation of assessments, the platform includes an AI-powered quiz generator. By clicking 'Generate Quiz' within a lesson, the creator sends the full text of that lesson to the backend.

The system directs GPT-4o to generate a diverse set of assessments, including both Multiple Select and True/False questions, derived specifically from the lesson content. Each item must include a question stem and the appropriate number of choices: four options for Multiple Select questions or two fixed options for True/False queries. The AI also identifies the correct answer markers for each item. This response is converted into a structured data format and transmitted to the frontend, where it automatically populates the quiz creation form with the suggested questions (Figure 5.2).

This allows the creator to review every question and make any necessary changes before saving. The quiz is only added to the database when the creator clicks the final save button, ensuring that no AI-generated content reaches students without the approval of a human expert.

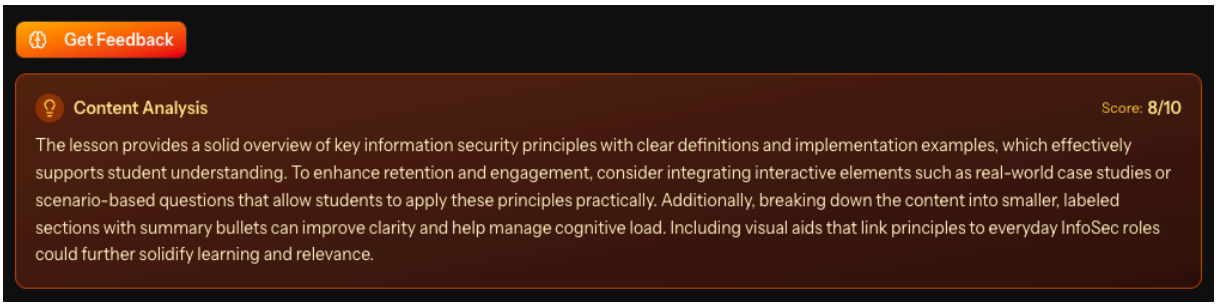

Figure 5.1 Lesson content feedback card, with AI-generated score and improvement suggestions for creators.

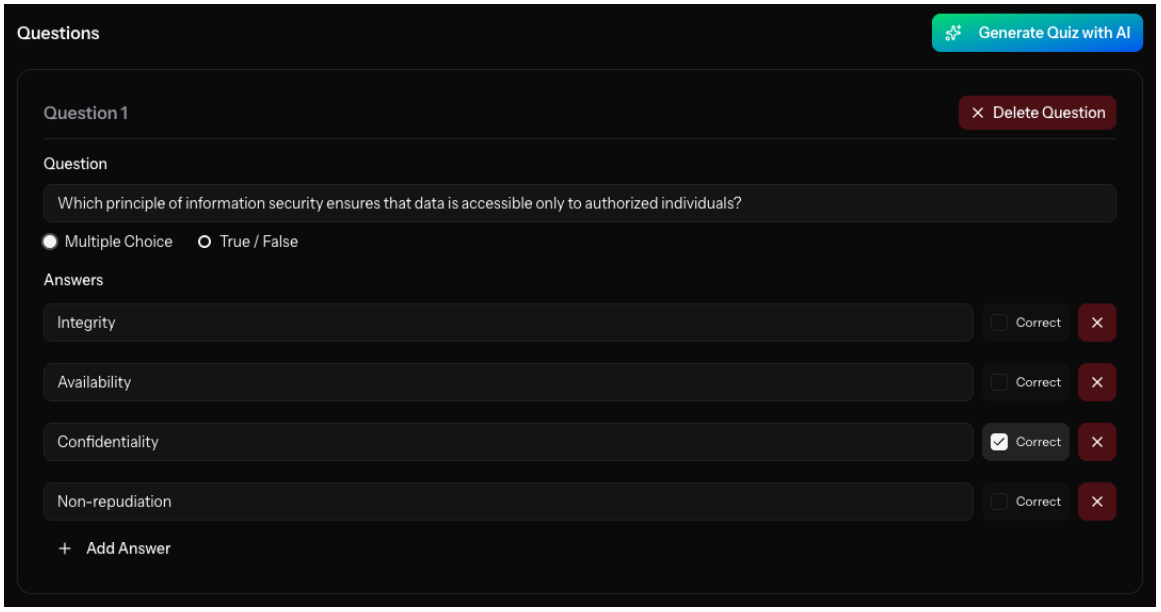

Figure 5.2 Quiz editor with AI-generated questions, demonstrating automatic quiz creation from lesson content.

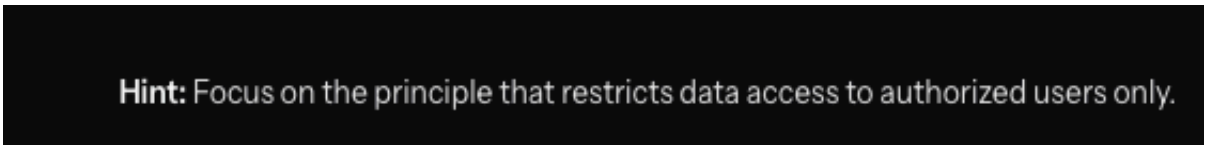

Figure 5.3 AI-generated hint displayed during quiz solving, supporting students in real time.

5.1.3 Hint Generation During Quiz Solving

During a quiz, students can request a contextual hint for any question by clicking the 'Get Hint' button (Figure 5.3). When activated, the system sends the question and all possible answers to the backend.

The system then asks GPT-4o to generate a hint that guides the student toward the correct answer without revealing it directly. The prompt specifically instructs the AI to use relevant concepts to help the student's reasoning rather than simply identifying the right choice. The resulting hint is displayed immediately below the question, keeping it visible as the student considers their options.

Like the creator feedback, these hints are not saved in the database. This keeps the database structure simple and ensures that if a student tries the quiz again later, they can receive a fresh hint that matches their current understanding. This design provides immediate support without permanently altering the assessment data.

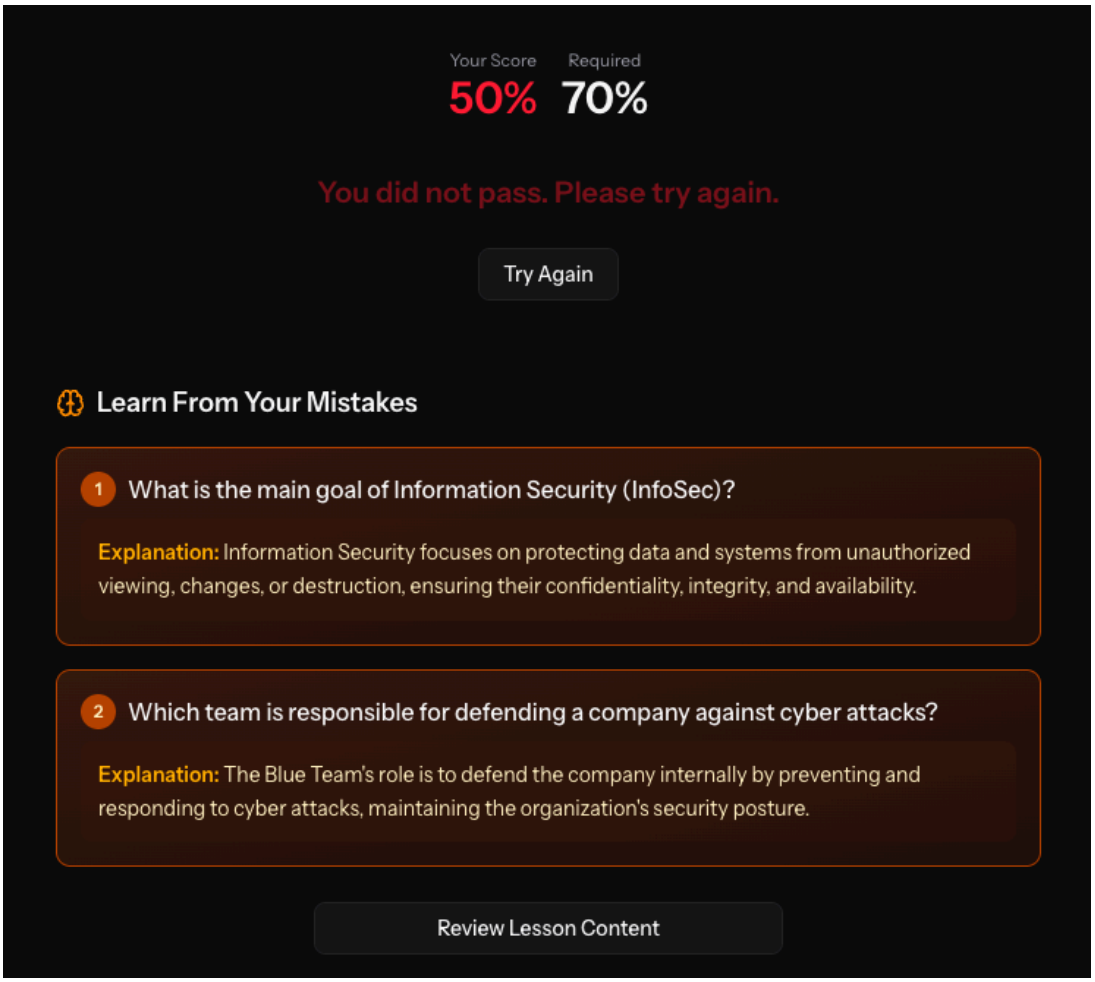

Figure 5.4 Explanations panel for failed quiz questions, showing how AI helps students understand mistakes.

5.1.4 Explanations for Failed Quiz Questions

After failing a quiz, students can review their attempt and see which questions they missed. For every incorrect answer, the system offers an 'Explain Answer' button. When clicked, the question, all answer choices, and the correct answer are sent to the backend.

The system then asks GPT-4o to write a concise explanation of why the correct answer is right and why it is better than the other options. This ensures the feedback serves a corrective function, helping the student understand the underlying logic rather than just memorizing a fact.

The explanation appears directly below the quiz results (Figure 5.4). Like the hints used during the quiz, these explanations are generated on demand and are not saved in the database. This feature is only available after a failed attempt, ensuring that in-depth feedback is provided exactly when a knowledge gap is identified.

5.1.5 AI-Assisted Flashcard Generation

To help creators build study materials more quickly, the platform includes an AI-assisted flashcard generator. By clicking 'Generate Flashcards' in the management screen, the creator sends the lesson’s text to the backend for analysis.

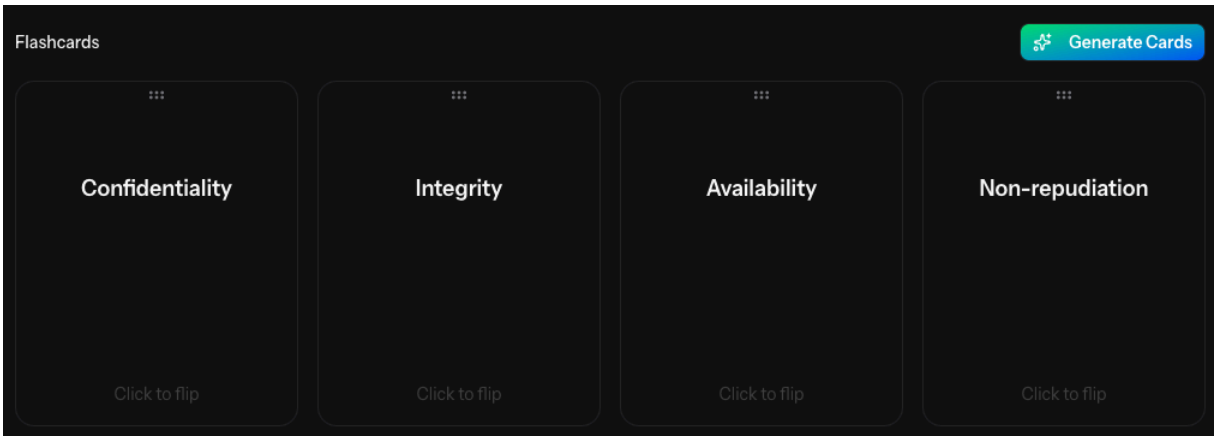

Figure 5.5 AI-generated flashcards added to the lesson editor, illustrating assisted flashcard creation.

The system then instructs GPT-4o to identify the core concepts within the lesson and format them as flashcard pairs. Each pair consists of a front-side prompt and a back-side answer. The AI's structured response is parsed by the backend and sent to the frontend, where it pre-fills the flashcard list with draft entries (Figure 5.5).

This workflow allows the creator to review every card, edit the text for accuracy, or delete irrelevant items before saving. No AI-generated cards are stored in the database until the creator explicitly submits the form. This approach significantly reduces the manual labor of creating review sets while ensuring the creator remains in full control of the final content.

5.2 Weakness Map and Adaptive Analytics

The Weakness Map is the platform's adaptive analytics tool (Figure 5.6). It collects 'failure signals' from various activities, such as incorrect quiz answers and forgotten flashcards, to show students exactly where they are struggling. By aggregating this data, the system provides a prioritized view of the lessons and cards that need the most attention. The following sections describe how the platform collects this data, ranks the difficulty of different topics, calculates trends over time, and uses a recommendation engine to guide the student's next steps.

5.2.1 Aggregating Review and Quiz Failures

The Weakness Map analyzes two types of failure signals, both of which are stored in the database with timestamps. The first is the incorrect review mark, recorded whenever a student swipes a flashcard to the left. The second is the failed quiz attempt, recorded when a student scores below the passing threshold.

Each signal is linked to the student, the specific card or quiz, and the parent lesson. This allows the system to trace every mistake back to the lesson level, the primary unit of analysis for the Weakness Map. To calculate a lesson's 'Weakness Score,' the system sums the total number of forgotten flashcards and failed quiz attempts associated with that lesson. A higher score signifies a deeper knowledge gap, marking that lesson as a high priority for review.

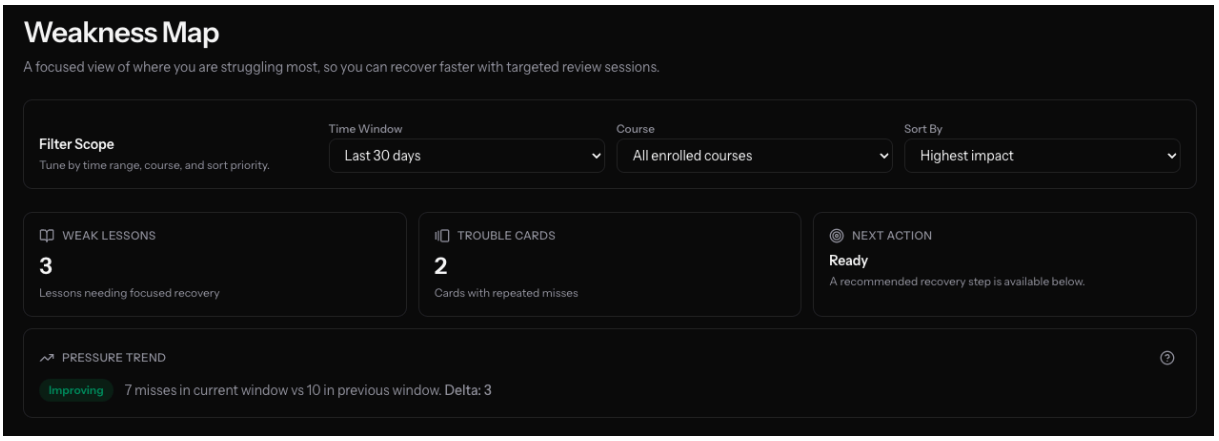

Figure 5.6 Weakness map overview with filters, showing how students can focus on specific problem areas.

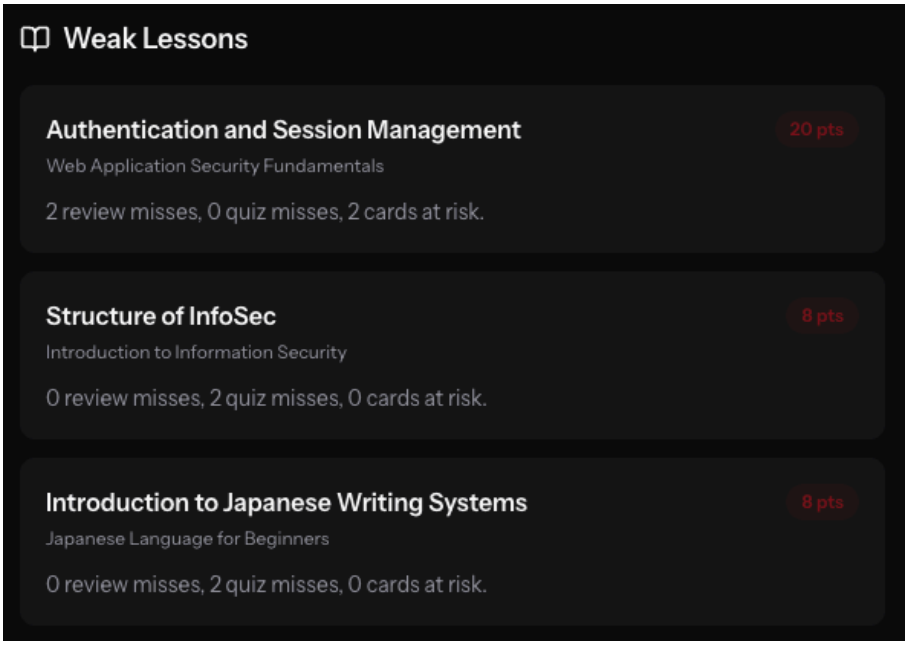

Figure 5.7 Weak lessons panel, highlighting lessons that need more attention.

5.2.2 Ranking Weak Lessons and Cards

Once the weakness scores are calculated, the platform ranks lessons in descending order, placing the most difficult topics at the top of the list. This ordered view allows students to immediately see which areas require the most effort (Figure 5.7).

Next to this high-level view, the interface also provides a granular ranking of individual flashcards. In this view, cards are ranked by how many times they have been marked incorrect (Figure 5.8). This is particularly useful because it highlights specific 'problem cards' even within a lesson that the student otherwise understands. By distinguishing between reliably recalled facts and persistently difficult ones, the system helps students pinpoint the exact concepts that are slowing their progress.

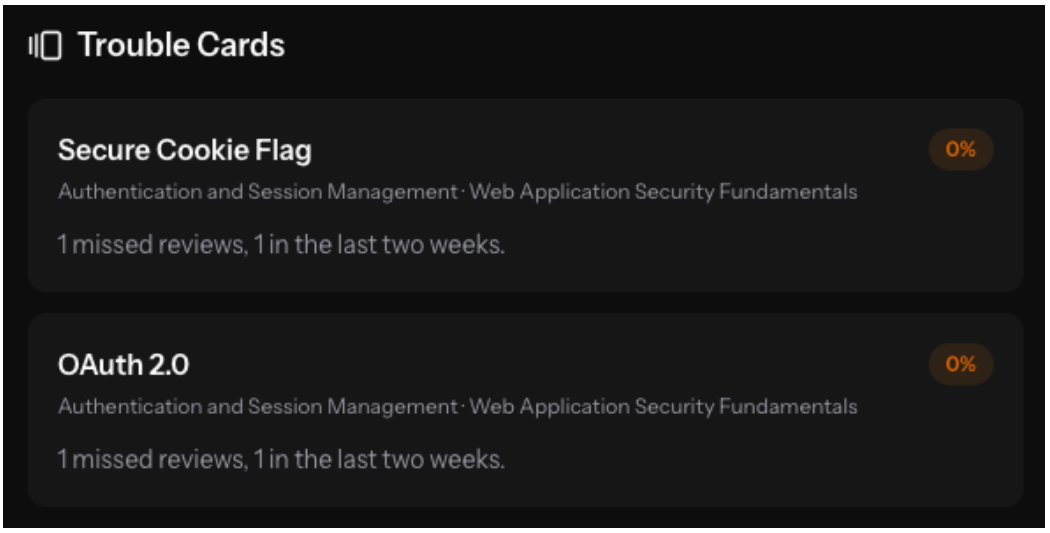

Figure 5.8 Weak cards panel, listing flashcards with the most errors or failures.

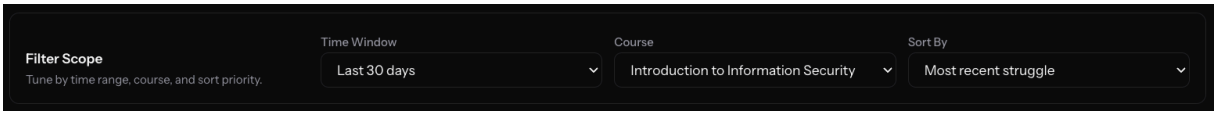

Figure 5.9 Filtering and sorting panel, helping the user find the information needed.

5.2.3 Filtering and Sorting Analytical Results

The Weakness Map includes powerful controls that allow students to filter and reorder their results instantly, without needing to wait for a server response (Figure 5.9). Using a 'single-fetch' architecture, the platform loads the full dataset into the Vue.js component once.

Filtering by course is handled entirely on the client side: When a student selects a specific subject, the interface updates immediately without a new API call. Sorting controls also allow students to switch between ranking by 'Weakness Score' or by the date of their most recent mistake. This allows students to choose between tackling long-term struggles or reviewing things they failed just today. Because all of these operations happen in the browser, the interface remains highly responsive and fast, even for students with a large history of study data.

5.2.4 Trend Calculation Across Time Windows

To give the weakness scores more meaning, the platform calculates a trend indicator for each lesson. This trend is found by comparing failure counts over two back-to-back seven-day periods: the 'previous window' (8 to 14 days ago) and the 'current window' (the last 7 days).

If failures have increased in the current window, the trend is marked as 'worsening.' If they have decreased, it is 'improving,' and if the counts are the same, it is 'stable.' These indicators are shown next to each lesson, giving students a clear sign of whether their performance is getting better or worse over time.

Because this calculation uses timestamps already saved in the database, it doesn't require any extra data storage. The platform performs this check instantly whenever the Weakness Map is loaded, ensuring that the trend always reflects the student's most recent activity.

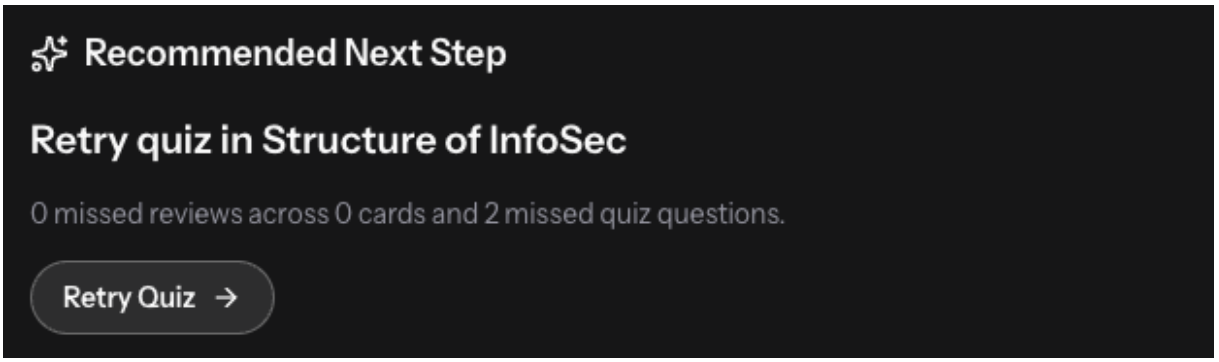

Figure: 5.10 Recommended actions and trend area, providing actionable insights and progress trends.

5.2.5 Recommended Recovery Actions

To help students act on their data, the system provides a specific 'Recommended Next Step' in the Weakness Map (Figure 5.10). The recommendation is determined by a set of priority rules that evaluate the student's history and the available materials for a lesson.

The system follows a clear hierarchy to choose the best intervention:

- **Flashcard Review:** If the student has unlocked flashcards for the lesson, the system prioritizes a review session. This is considered the most targeted way to fix memory gaps.
- **Quiz Retry:** If no flashcards are available but the student recently failed the lesson's quiz, the system recommends retrying the quiz to encourage active retrieval.
- **Lesson Re-read:** If neither a quiz nor flashcards apply, the system suggests re-reading the source material to build a stronger foundational understanding.

Each recommendation includes a direct link to the activity. This 'one-click' approach removes any friction, allowing the student to move instantly from identifying a weakness to fixing it.

5.3 Gamification and Progress Motivation

Gamification features throughout the platform encourage steady engagement and provide students with instant feedback on their progress. The system operates through four main tools: experience points (XPs) accumulation, daily streaks and badge award system for milestones.

5.3.1 XPs Accumulation

XPs serve as the primary metric for tracking learning activity. Students earn XPs through three specific actions: completing a lesson (10 XPs) recognizes the effort spent on new content, passing a quiz (20 XPs) rewards actual comprehension, and finishing a review session (5 XPs) encourages regular practice through spaced repetition.

These points are stored permanently in the user profile and never decrease due to failure or inactivity. The total XPs is displayed on the Dashboard (Figure 5.11), where it ranks students within the community. In this way, XPs acts as both a personal progress tracker and a tool for social comparison, tapping into both internal and external motivation.

To ensure system integrity, XPs is awarded on the server side the moment a task is verified. This prevents manual manipulation and ensures that the database accurately reflects genuine learning activity.

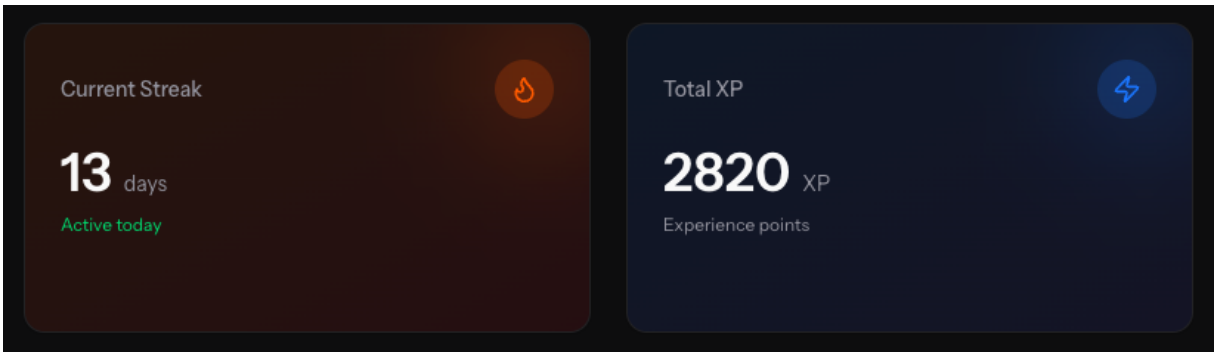

Figure: 5.11 XPs and streak cards, visualizing progress and daily engagement.

5.3.2 Streak Tracking

The platform tracks a daily learning streak to encourage consistent study habits. This streak represents the number of consecutive days a student logs at least one activity, such as finishing a lesson, attempting a quiz, or completing a review session. Including a wide range of activities is an intentional design choice to ensure that students are rewarded for the simple habit of showing up rather than being forced to complete a specific type of task every day.

The system updates the streak counter on the server whenever an activity is recorded. When a student logs their first activity of the day, the system compares the current date to the date of their last session. If the previous activity occurred exactly one day prior, the counter increases by one. If more than a day has passed, the streak resets to one to represent the start of a new daily sequence. If the student has already logged an activity earlier that same day, the counter remains unchanged to prevent multiple increments within a single twenty-four-hour period.

The current streak count is clearly displayed on the Dashboard to keep the student informed of their progress. The reset process happens automatically behind the scenes: if a day is missed, the counter is not adjusted until the student returns for their next session. At that moment, the system identifies the gap in the timeline and restarts the streak at one, reflecting the break in consistency.

5.3.3 Badge Awarding

The platform features a series of badges (Figure 5.12) designed to recognize specific milestones across three categories: consistency, total effort, and content completion. To reward daily engagement, the system tracks streaks through the Getting Started (3 days), On Fire (7 days), Dedicated (14 days), and Unstoppable (30 days) badges. Overall progress is measured through XPs accumulation, with milestones including First Steps (100 XPs), Knowledge Seeker (500 XPs), Scholar (1,000 XPs), and Master (5,000 XPs). Finally, students are rewarded for completing curriculum content via the Lesson Complete (1 lesson), Course Explorer (5 lessons), and Fast (10 lessons) badges.

The system evaluates badge eligibility on the server immediately after any triggering event. For example, once a lesson is finished or a quiz is passed, the system automatically checks if the requirements for a new badge have been met. If the student qualifies and has not previously received that specific award, a new badge record is created in the database. This event-driven approach ensures that students receive their rewards instantly, reinforcing the connection between their effort and the platform's recognition without the need for delayed background processing.

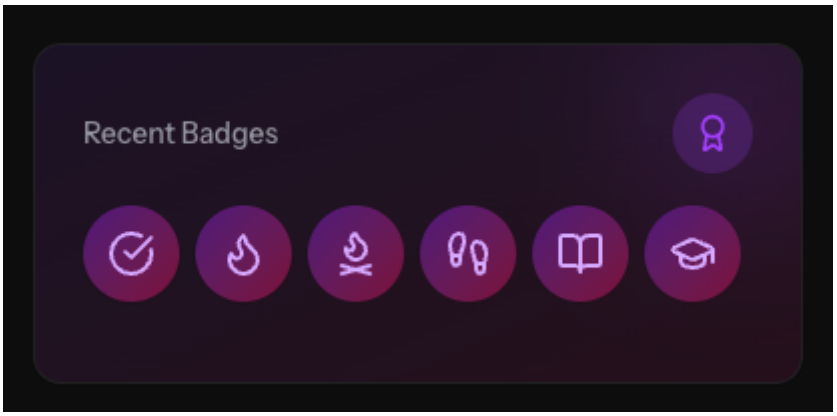

Figure: 5.12 Badges and achievements area, displaying earned milestones.

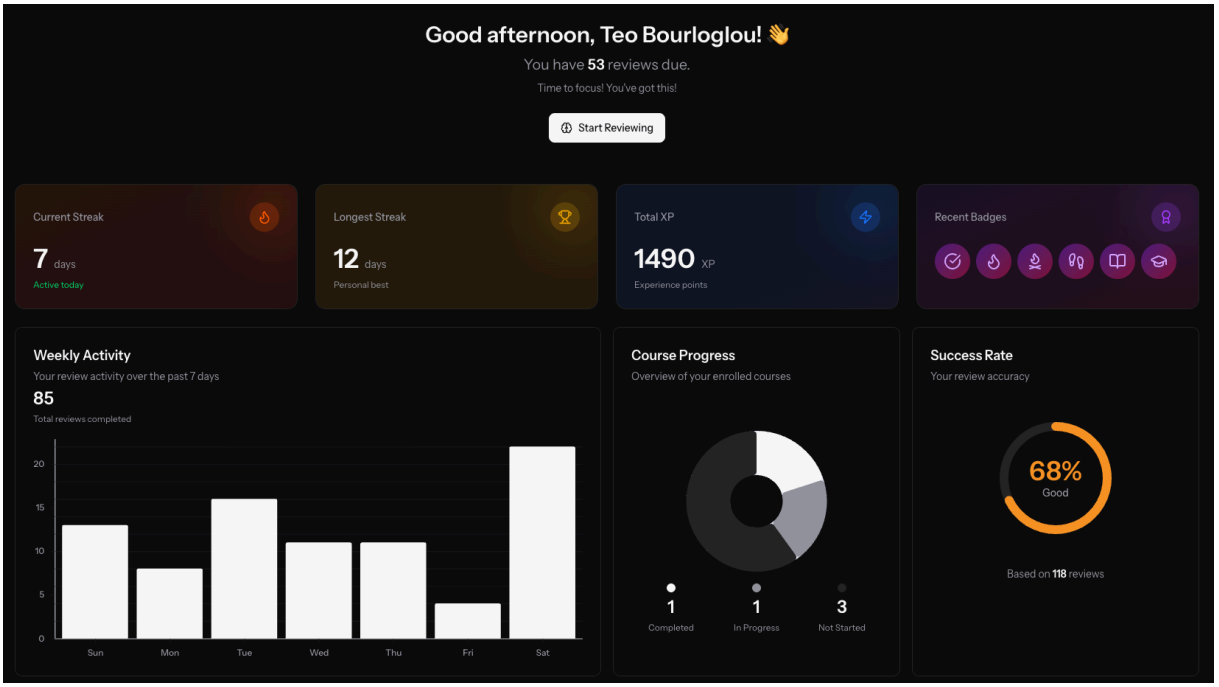

Figure: 5.13 Full dashboard overview after login, summarizing the student's status.

5.4 Dashboard and Monitoring

The Dashboard serves as the platform's central hub and the first screen a student sees after logging in (Figure 5.13). By bringing together data from various parts of the system, it provides a clear, unified snapshot of a student's current progress, their latest activities, and what they should focus on next. The following sections break down each component of the Dashboard and explain how they support the learning experience.

5.4.1 Personalized Overview

The top of the Dashboard provides a summary of the student's overall progress through four key indicators: total accumulated XPs, current daily streak, longest streak recorded, and the 8 most recent badges awarded. Together, these figures offer a clear picture of the student's total investment in the platform, the consistency of their study habits, and the overall scope of their learning activity.

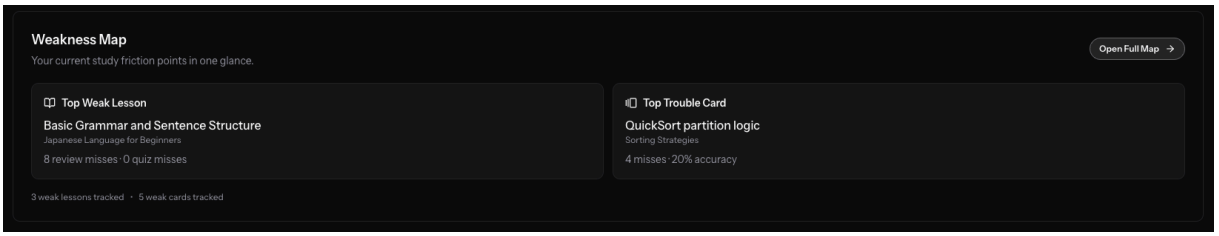

Figure: 5.14 Lower dashboard area with weakness preview.

To make this data easy to read at a glance, each indicator is housed in its own visually distinct card. This information is retrieved from the server the moment the page loads, ensuring that the student's record is accurately reflected. Because the Dashboard refreshes upon every visit, these values remain current and provide an up-to-date snapshot of the student's achievements.

5.4.2 Activity, Completion and Success Indicators

Below the overview, the Dashboard highlights indicators of short-term activity. A count of review sessions completed today provides immediate feedback on how well the student is keeping up with their spaced-repetition practice. Similarly, a diagram of lessons completed during the current week helps the student see their progress within the context of a typical weekly study cycle.

The primary success metric in this section is the overall quiz pass rate, which is the percentage of passed quizzes compared to total attempts across the entire platform. This figure gives the student a straightforward way to track their assessment performance over time. When a low pass rate is visible alongside the "weakness score" data further down the page, it acts as a subtle prompt for the student to use the platform’s review and remediation tools.

5.4.3 Weakness Map Preview

The Dashboard features a condensed preview of the Weakness Map (Figure 5.14), highlighting the one lesson with the highest weakness score and the one flashcard with the most failed reviews. For the lesson, the number of failed reviews and the number of quiz misses is shown. For the flashcard the number of missed reviews and the percentage of accuracy are displayed under the name of the card. This provides an immediate view of the specific areas that require the most attention. A link in the top right side of this preview directs the student to the full Weakness Map page, which contains the complete rankings, filters, and recommended study actions.

This preview serves two main purposes. First, it introduces the platform’s adaptive analytics to students who might not otherwise visit the full map on their own. Second, by placing this data directly at the daily entry point, it keeps "weakness awareness" front and center. This integration makes it much more likely that a student will see their problem areas and take the necessary steps to review and improve.

5.4.4 Quick Access to Active Courses

The bottom of the Dashboard displays individual cards for every course the student is currently taking. Each card shows the course title and the student's current completion percentage, which is based on how many lessons they have finished. A "Go to course" link is also provided, which takes the student directly to the next incomplete lesson. This eliminates the need to manually search through a course list to find where they last left off.

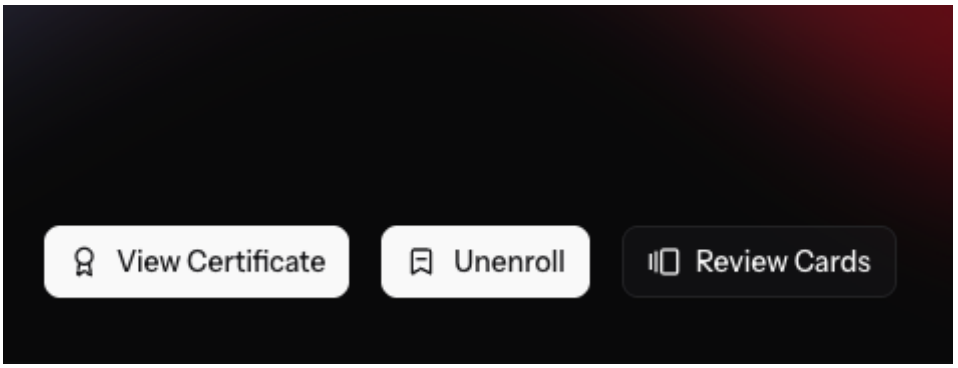

Figure 5.15 Course page showing certificate availability after completion, marking the student's achievement.

This quick-access design is intended to reduce "navigational friction," making it as easy as possible for students to jump back into their studies. By shortening the path from the login screen to an active lesson, the platform encourages more frequent study habits.

5.5 Certification Workflow

The platform uses a structured certification workflow that automatically generates a verifiable certificate once a student finishes a course. This process covers everything from initial detection and record creation to public display and print-ready rendering. The following sections describe each of these stages in detail.

5.5.1 Detecting Full Course Completion

The certification process begins the moment a student’s status changes from partial to full course completion. The system defines "full completion" as every lesson within a specific course being marked as finished. To ensure this is tracked accurately, the server runs a check every time a student completes a lesson.

During this check, the backend compares the total number of lessons in the course against the number of lessons the student has actually finished. If these two numbers match, meaning no incomplete lessons remain, the system triggers the certification workflow. This process is designed to be efficient and error-proof: if a certificate already exists for that student and course, the system recognizes it and avoids creating a duplicate.

5.5.2 Creating Certification Records

When a student finishes a course, the backend automatically creates a permanent certificate record in the database. This record stores all the details needed to verify the achievement later, such as the student’s name, the course title, and the date they finished. It also includes a unique ID, which acts as a public identifier for that specific certificate.

Figure: 5.16 Public certificate page, as seen by the student or a third party.

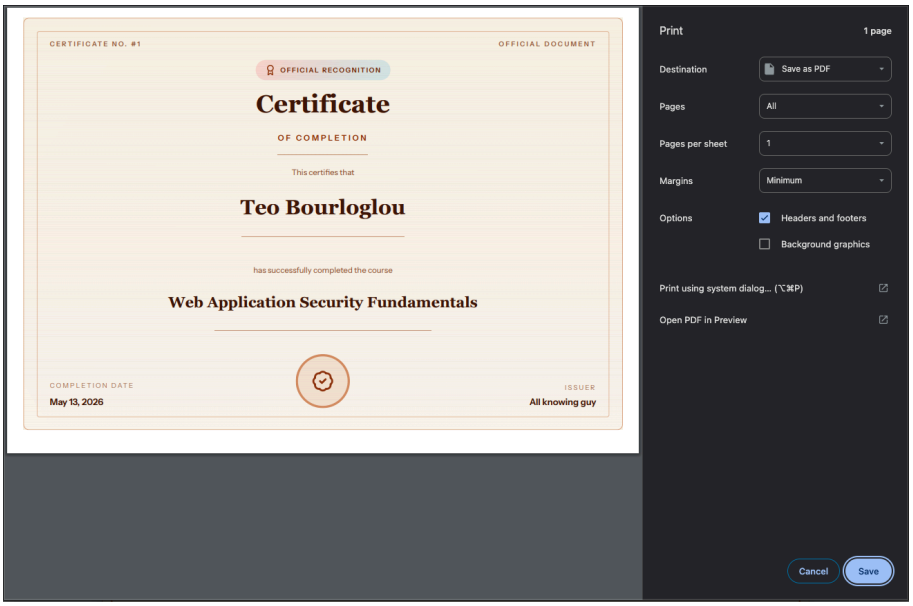

Figure: 5.17 Print-ready certificate layout or print preview, demonstrating the final polished artifact.

5.5.3 Public Certificate Presentation

Every certificate has its own public web link that looks like /certificates/{id}. This link is open to everyone, so students can easily share their achievement with job recruiters or schools (Figure 5.16). Because people don't need to log in to see it, it works as a simple way for anyone to verify that the certificate is real.

The public page shows the student’s name, the course they finished, the date, and the platform's logo in a clean, professional layout. Even though the page is public, it stays private unless the student shares the link.

5.5.4 Print-Ready Certificate View

The certificate page features a "Print" button that opens the browser's standard print window (Figure 5.17). To make sure the document looks professional, the page uses special CSS rules called "print media queries." These rules automatically hide website parts that don't belong on paper, such as navigation bars and buttons. It then resizes the text and adjusts the spacing to fill the page perfectly.

This setup allows the student to either print a physical copy or use the "Save as PDF" option on their computer to keep a digital file. Because this is done entirely with CSS code, the system doesn't need to create a separate file on the server. This ensures that the printed version always matches the information on the website exactly, keeping the record consistent and accurate.

5.6 Validation, Authorization and Data Integrity

To keep the platform safe and reliable, the system uses several layers of security. These protections make sure that only the right people can change content and that all information entered into the system is accurate and complete.

First, every piece of information sent to the server goes through server-side validation. This check makes sure that no incorrect or malicious data is saved, even if someone tries to bypass the website's front interface.

The system also uses enrollment-gated access. This ensures that a student can only see the quizzes, flashcards, and lessons for courses they are actually signed up for. Finally, for important actions that involve multiple steps, the system uses transactional guarantees. This is an "all or nothing" rule: if one part of a save fails, the whole process is canceled. This prevents the database from ever ending up with half-finished or broken records.

5.6.1 Access Control Across Core Actions

To keep the platform secure, the system uses strict rules to control who can see or change information. These rules are divided into two main categories: creator controls and student controls.

For creators, any action that involves creating, changing, or deleting a course, lesson, quiz, or flashcard is protected. These actions are only allowed for the specific person who created the course. The system checks this at the server level using Laravel policies. If someone else tries to make a change, the system automatically blocks the request and sends a "403 Forbidden" error. This happens before any data is even processed, making sure the content stays safe.

For students, the system uses enrollment-gated access. This means a student can only mark a lesson as finished, take a quiz, or use flashcards if they are actually signed up for that course. Before the server processes any of these actions, it looks for an active enrollment record. If the student isn't enrolled, they are redirected to the signup page or shown an error message.

These two security layers are completely separate. A creator cannot use the student features (like taking a quiz for points) unless they also enroll in the course as a student. Similarly, a student can never access the authoring tools to change a lesson. This "separation of concerns" ensures that every user stays within their proper role.

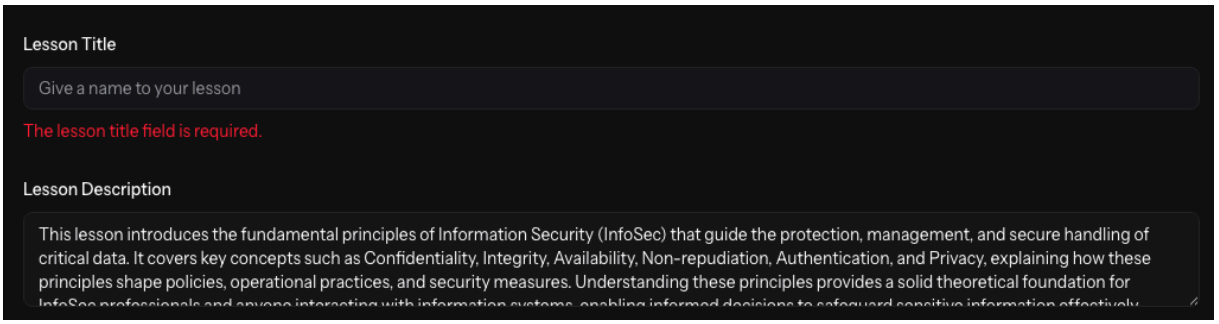

Figure: 5.18 Validation error messages in the course editor, showing how incorrect input is handled.

5.6.2 Validation of Nested Authoring Data

All information sent to the platform is checked by the server to ensure it is correct and complete. This is done using Laravel's validation framework, which acts as a gatekeeper. Before any data reaches the main part of the system, the server checks it against a set of strict rules. If the data is missing or incorrect, it is rejected immediately (Figure 5.18).

For basic items, the rules are straightforward: Courses must have a title and a description. Lessons must have a title and a body of content. Flashcards must have text on both the front and the back. For Quizzes, the validation is more detailed because the data is more complex. The system looks inside each question to make sure it follows the right structure. Every question must have a clear "stem" (the question itself) and at least two possible answers. The system also checks to make sure that at least one of those answers is marked as correct.

If there is a mistake, for example, if a user forgets to provide a second answer for the third question, the system points exactly to that spot. Instead of showing a generic error message, the platform gives the creator precise feedback so they know exactly which question or answer needs to be fixed. This makes it much easier to correct errors and save the work successfully.

5.6.3 Protection of Student-Only and Enrolled-Only Flows

On top of general security, the platform has extra protections for the most important parts of the learning process. These checks make sure that students only interact with the lessons, quizzes, and flashcards that are part of their own courses.

When a student tries to open a lesson, the system first confirms they are actually enrolled in that course. The same thing happens when a student submits a quiz. The server checks that the quiz belongs to the correct lesson and that the student is signed up for that specific course. This stops anyone from trying to "cross-submit" answers to quizzes in courses they aren't taking.

The flashcard review system works the same way. It verifies that every card being studied is part of a lesson within an enrolled course. All of these security checks happen on the server at the exact moment the request is made. Because the server makes the final decision, it is impossible for someone to bypass these rules by trying to trick the website or sending fake data directly to the system.

Finally, the platform protects all private pages by requiring users to log in. If someone who isn't logged in tries to access a lesson or a dashboard, they are sent back to the login page. The only exception is the public certificate page, which is intentionally left open so that students can share their achievements with others.

5.6.4 Transactional Consistency and Safe State Changes

Some actions on the platform change many database records at once. To keep the data accurate, the system uses "transactions." This is an "all or nothing" rule that ensures the database never gets stuck with half-finished or broken information.

One example is lesson reordering. When a teacher changes the order of lessons in a course, the system has to update a number field for every single lesson at the same time. Because this involves many updates, the system wraps them in a transaction. If even one lesson fails to update, the whole process is canceled, and the lessons stay in their original order. This prevents the course from ending up with missing or overlapping lesson numbers.

The second example is certificate creation. As mentioned before, when a student finishes a course, the system saves all the certificate details in one go. This ensures that a certificate record is either 100% complete or not saved at all. These protections show a core design rule of the platform: any task that could cause confusion or errors if it was interrupted is always protected by a database transaction.

5.7 Testing and Verification

The platform includes a set of automated tests built with PHPUnit to ensure everything works correctly. These tests focus on the most important parts of the system, such as the core learning path, the math behind the Weakness Map, and the accuracy of quiz records. The following sections describe what each category of tests covers and how they are organized.

5.7.1 Verifying Core Workflows

The core workflow tests check the student's journey from start to finish. A typical test creates a course with several lessons, signs up a test student, and marks each lesson as finished. It then verifies that the system correctly creates a certificate and saves it in the database. This confirms that the progress tracking, the completion logic, and the certificate system all work together perfectly.

Other tests in this group check the opposite: they make sure that if a student is removed from a course, they can no longer see the lessons, take quizzes, or study the flashcards. These tests try to access the protected pages after unenrollment to confirm the system blocks the request. Together, these tests prove that the security rules for the learning path are working as intended.

5.7.2 Tests for Analytical Correctness

These tests check that the Weakness Map's math and logic are accurate. Each test sets up a specific list of "fake" study records and quiz scores with exact times and dates. The system then runs its calculations, and the test compares the results to the correct answers to make sure they match perfectly (Figure 5.19).

For the ranking tests, the system is given lessons with different amounts of failures. The test checks if the Weakness Map puts these lessons in the right order based on their scores. For the trend tests, failure records are placed into the two seven-day windows. The test then confirms that the system correctly labels each lesson as "worsening," "improving," or "stable" based on the data. These tests make sure that any future changes to the code won't accidentally break how the Weakness Map works.

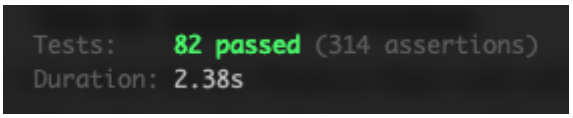

Figure: 5.19 Test run output or test summary, providing evidence of automated verification.

5.7.3 Tests for Attempt Data Integrity

These tests check three important rules for how quiz scores are saved. First, they make sure that when a student finishes a quiz, the system creates exactly one record in the database. The test verifies that this record is linked to the right student and the right quiz so that no data is lost or duplicated.

Second, the tests check that the score itself is calculated correctly. A set of answers with a known number of correct and incorrect responses is sent to the system, and the test confirms that the final score saved in the database matches the expected math.

Third, the tests verify that the "pass" or "fail" label is accurate based on the passing rule. The tests specifically check "boundary" cases, for example, such as a score of exactly 70%, one just below it at 69%, and others well above or below. These checks are important because small mistakes in rounding or logic often happen at these cutoff points. By testing these boundaries, the system ensures that every student gets the correct result.

5.7.4 Why Verification Matters for Adaptive Features

The accuracy of the Weakness Map’s advice and trends depends completely on the quality of the data. If quiz scores are calculated incorrectly, the "pass" or "fail" labels will be wrong, which ruins the data used to find a student's weaknesses. If the system fails to record when a flashcard was missed, the ranking will not show what the student is actually struggling with. Even worse, if the trend logic is broken, a student who is falling behind might be told they are "improving," which could stop them from getting the help they need.

These mistakes are more than just small errors. Because the system is designed to adapt to the student, giving bad advice or wrong trends can actually hurt their learning by pointing them in the wrong direction. The test suite prevents this by checking every single step, from the moment a student clicks an answer, to the moment a trend appears on their map. If any part of the math or logic breaks, the tests will catch it immediately so the platform remains a reliable and helpful tool.

5.8 Final Implementation Summary

5.8.1 Recap of Implemented Functionality

Chapter 5 has detailed the full range of intelligent and adaptive features built into the platform. These features work across five main areas. First, the AI-supported features (such as lesson feedback, quiz and flashcard generation, and real-time hints) connect the platform to GPT-4o. This allows the AI to help both teachers and students, though all AI-generated content must be approved by a human before it is saved. Second, the Weakness Map collects data from quizzes and study sessions to show students exactly what they need to work on through ranked lists and trend labels.

Third, the gamification system uses points, streaks, and badges to keep students motivated and reward their hard work. Fourth, the Dashboard brings all of this information together onto a single screen so students can see their progress as soon as they log in. Finally, the certification system automatically

detects when a student finishes a course, creates a secure certificate with a unique ID, and provides a professional copy that can be shared or printed.

All of these features are supported by the security and data rules described in section 5.6. These protections make sure that all information is correct, that only the right people can change the content, and that important data is always saved safely and completely.

5.8.2 Most Significant Implementation Outcomes

Among the features described in this chapter, a few stand out as the most important results of the project. The Weakness Map is the most unique part of the platform's design. It combines failure logs, data from different sources, and trend analysis to help students automatically. It works on its own by watching how a student learns, so the user doesn't have to manually enter what they find difficult.

The AI tools for creators, especially for creating quizzes and flashcards, greatly reduce the time and effort needed to build a course. By keeping a human in the loop to review the AI's work, the platform ensures the teaching material remains high quality. The certification system adds long-term value by providing a secure, unique link that students can use to prove their skills to others. Finally, the automated tests make the entire system reliable. This is vital for an adaptive platform, as it ensures that the data and advice given to students are always accurate and helpful.

5.8.3 Chapter Summary

The implementation details described in this chapter establish the functional foundation of the platform. By integrating AI-driven content creation, automated progress tracking through the Weakness Map, and a structured gamification framework, the system moves beyond a traditional static course player into a truly adaptive learning environment.

The successful deployment of these features, supported by a robust security architecture and an extensive suite of automated tests, proves that complex learning platforms can be built into a responsive, high-performance web application. These design choices ensure that the platform is not only technically stable but also educationally effective, providing a verified and reliable baseline for supporting modern students.

Chapter 6: Summary, Conclusions, and Future Extensions

6.1 Summary

This thesis presented the complete design, implementation, and automated verification of an intelligent e-learning web platform built for lifelong learning with adaptive review and AI assistance. To solve the widespread challenges of low student completion rates and poor long-term knowledge retention found in traditional digital tools, a modern full-stack web application was implemented. The system relies on a single-codebase software architecture consisting of PHP 8 and Laravel 12 on the backend, Vue 3 on the frontend, and Inertia.js acting as the server-driven single-page application bridge. This specific technical choice delivers high client-side interactivity and fast screen transitions without the architectural duplication and complexity of creating a separate, decoupled REST Application Programming Interface application.

The developed platform combines five major learning pillars to establish a highly personalized, active environment for users:

- An automated **Spaced Repetition System (SRS)** review queue that uses a geometric doubling interval algorithm to combat the human forgetting curve.
- An interactive **Gamification Layer** that features Experience Points (XPs), daily activity streaks, and milestone badges to satisfy intrinsic psychological needs, keep students engaged, and build regular study habits.
- An adaptive **Weakness Map** analytics tool that automatically aggregates localized temporal failure signals (specifically quiz errors and forgotten cards) to rank difficult topics and provide direct recovery actions.
- A suite of **Generative AI Tools** powered by OpenAI's GPT-4o model via the Laravel AI SDK, delivering structured automated quiz and flashcard generation for creators alongside real-time hints and explanations for students.
- A **Verifiable Certification Workflow** that automatically creates unique, public web links and print-ready layouts for users who finish all lessons in a course.

Finally, the complete system was verified using a robust suite of automated feature and unit tests with PHPUnit 11 to guarantee operational consistency and statistical accuracy.

6.2 Conclusions

Several important conclusions can be drawn from the design, development, and automated testing of this intelligent learning platform. First, the project proves that static online learning materials can be successfully turned into active, adaptive spaces by monitoring observable user actions. In traditional e-learning, every student follows the exact same path, which completely ignores individual differences and background knowledge. The platform's Weakness Map module demonstrates that an educational system can track a learner's knowledge gaps implicitly in the background without forcing the user to manually report their struggles. By using simple, observable behavioral failure signals, like failed quiz attempts and forgotten flashcards, instead of massive, hidden predictive models, the platform showcases a highly practical approach to knowledge tracing that reduces wasted study time and guides learners exactly where their focus is required most.

From a cognitive psychology perspective, the platform highlights the danger of traditional study behaviors like "cramming," which fail because human memory naturally decays along a steep, predictable forgetting curve. We can conclude that integrating an automated spaced repetition system directly addresses this limitation by enforcing the spacing effect. By forcing students to review flashcards at geometric doubling intervals, the platform puts the theoretical "desirable difficulty" principle into direct practice. This effortful retrieval forces the brain to reconstruct memories actively when they are partially forgotten, which results in much stronger long-term knowledge retention compared to passive reading. Furthermore, locking flashcards until a lesson is marked complete ensures that these review tools are used for knowledge consolidation rather than as a first point of contact with new content.

Regarding student engagement, the thesis validates key principles of motivational psychology, particularly Self-Determination Theory. Gamification mechanics are often criticized as superficial distractions, but this project concludes that game elements like Experience Points (XPs), daily streaks, and milestone badges can successfully satisfy intrinsic psychological needs for competence and autonomy. However, a crucial conclusion is that gamification only works well when the game elements are aligned directly with meaningful learning goals, such as completing a spaced review session or passing an assessment quiz, rather than superficial actions like clicking through slides. When designed this way, the gamification layer provides the vital psychological drive that encourages students to return to the platform daily, which directly sustains the repetitive engagement needed for spaced repetition to be effective over time.

On the side of artificial intelligence, the platform leads to the conclusion that generative models are best utilized when they provide active pedagogical scaffolding instead of just delivering raw text layouts. Features like the automated quiz hints and post-failure mistake explanations work as an on-demand virtual tutor, guiding students toward correct conceptual reasoning rather than simply giving away correct answers. For content creators, the study shows that a "human-in-the-loop" architecture is the safest and most efficient way to apply large language models in educational settings. Using backend JavaScript Object Notation (JSON) schemas to force structured artificial intelligence outputs guarantees that generated quizzes and flashcards match exact programmatic expectations, drastically lowering course development labor while keeping human experts in full control of the final published material.

From a technical and architectural standpoint, the selection of a server-driven single-page application model using Inertia.js is highly justified for research-informed educational web software. It successfully balances client-side interactive requirements, like fluid flashcard swipe gestures and real-time dashboard charts, with centralized server-side authority over database routing, data validation, and asset security. Finally, the project proves that comprehensive automated verification using tools like PHPUnit is absolutely necessary for data-driven adaptive systems. Because the platform automatically updates student metrics, alters review queues, and issues certifications based on behavioral logs, a rigorous testing suite is the only way to make the underlying code architecture accountable, verifiable, and instructionally reliable for the continuous learner.

6.3 Future extension

Although the completed platform successfully implements all designed intelligent and adaptive features, there are several boundaries and limitations in the current version that present clear opportunities for future technical extensions. The first and most critical change that would truly

revolutionize the educational value of the platform is moving from a diagnostic system to a fully automated generative learning path model. Right now, the Weakness Map aggregates localized failure signals to tell students which lessons need attention and recommends basic actions like re-reading the content or retrying quizzes. However, if a student fails an assessment, simply telling them to re-read the exact same source material might not solve their confusion, because the original explanation did not work for them the first time. A future change that would make a massive difference is the implementation of Dynamic Curriculum Generation. When a student triggers a high weakness score, the server could instruct the artificial intelligence to analyze the exact pattern of their incorrect quiz answers. Instead of just showing a link to the old lesson, the system would dynamically compile a completely personalized, new "remedial micro-lesson." This custom text would feature alternative analogies, simplified breakdowns, and targeted practice problems designed specifically to correct that student's unique misconception, transforming the platform from a system that merely tracks mistakes into an active engine that automatically reshapes the curriculum in real time.

Another valuable path for future extension involves expanding the multi-dimensional data capabilities of the educational tools. The current implementation of the platform is entirely focused on text generation and text processing, meaning it cannot process rich media formats or use multimodal inputs in practice. However, rich instructional materials frequently rely on diagrams, mathematical formatting, and visual images. Since several of the configured AI providers natively support multimodal operations, a valuable future extension will be opening the platform to visual inputs. For content creators, this would allow them to upload textbook charts, infographics, or slide images, and the AI agent could automatically extract the key concepts to generate contextual quiz questions or flashcard pairs. For students, multimodal support would enable them to snap a picture of a handwritten math problem or a complex coding diagram and receive immediate, step-by-step assistance from the AI hint and explanation engines.

Furthermore, the underlying memory algorithms can be enhanced to achieve higher precision. The current spaced repetition system relies on a simplified geometric doubling interval rule. While this provides predictable behavior that is very easy to implement, audit, and save to the database, it assumes a single universal rate of forgetting for every user. Future iterations of the platform can move beyond these fixed formulas toward advanced, data-driven machine learning models. By transitioning to probabilistic models like Half-Life Regression or recurrent neural network architectures like Long Short-Term Memory networks, the scheduling engine can actively learn individual forgetting rates from actual performance data. The algorithm would analyze aggregate failure logs, response latencies, and user habits over time to predict the exact moment a specific student is about to forget a concept, making review sessions even more efficient.

Finally, the platform can expand toward collaborative and peer-produced content spaces. In the current application, the line between content creators and students is separated by course enrollment boundaries, even though a single registered account can technically function in both roles within the platform. However, modern digital lifelong learning shows a clear structural trend toward community-driven and peer-produced material, where the boundary between the teacher and the learner is completely fluid. A major future upgrade will involve building social collaboration features. Students should be allowed to share their own custom flashcard decks, publish public study guides for courses they have successfully completed, and rate community-contributed materials. Additionally, the gamification layer can expand to include cooperative challenges, such as group leaderboards or peer-review rewards, which would foster a deeper sense of social relatedness and motivate adult learners to maintain their daily consistency together.

REFERENCES

[1] I. Gligorea, M. Cioca, R. Oancea, A.-T. Gorski, H. Gorski, and P. Tudorache, "Adaptive Learning Using Artificial Intelligence in e-Learning: A Literature Review," Education Sciences, vol. 13, no. 12, p. 1216, 2023, doi: 10.3390/educsci13121216

[2] L. Zhaidakbayeva, M. Dildabayeva, and K. Tynyshbek, "Review: Using AI Learning Models to Develop E-Learning Platforms," in Proc. 2024 Int. Conf. Automatics and Informatics (ICAI), 2024, doi: 10.1002/9781119256151.ch17.

[3] B. Zhao, "Construction of Personalized Intelligent Learning Service Platform for Universities Based on Artificial Intelligence," in Proc. 2023 Int. Conf. Industrial IoT, Big Data and Supply Chain (IIoTBDSC), 2023, doi: 10.2478/amns-2025-0411.

[4] I. M. García-López, E. Acosta-Gonzaga, and E. F. Ruiz-Ledesma, "Investigating the Impact of Gamification on Student Motivation, Engagement, and Performance," Education Sciences, vol. 13, no. 8, p. 813, 2023, doi: 10.3390/educsci13080813.

[5] A. Khaldi, F. Nader, and R. Bouzidi, "Gamification of e-Learning in Higher Education: A Systematic Literature Review," Smart Learning Environments, vol. 10, no. 1, 2023, doi: 10.1186/s40561-023-00227-z.

[6] N. Nair, S. Pikle, S. Save, R. Varghese, and K. Sonawane, "FlashMe: Automatic Flashcard Generation," in Proc. 2023 14th Int. Conf. Computing Communication and Networking Technologies (ICCCNT), 2023, doi: 10.1109/ICCCNT56998.2023.10308164.

[7] C. Dede, "Designing a Model for Massive Digital Lifelong Learning," in Proc. ACM Conf., 2024, doi: 10.1145/3657604.3665449.

[8] S. Mishra, S. Dutta, S. Ranjan, V. Sharma, P. Hewage, and C. Iwendi, "Enhancing Educational Adaptability: A Review and Analysis of AI-Driven Adaptive Learning Platforms," in Proc. 4th Int. Conf. Innovative Practices in Technology and Management (ICIPTM), 2024, doi: 10.1109/ICIPTM59628.2024.10563448.

[9] Dhairya, J. Hrishikesh, G. P. Sonu, D. Vaishnavi, L. Pallavi, and S. Sanapala, "Skillify: Enhanced Learning Management System Using Generative AI," in Proc. 2024 Int. Conf. Circuit Power and Computing Technologies (ICCPCT), 2024, doi: 10.23939/sisn2025.17.330.

[10] M. Huang, "Spaced Repetition and Retrieval Practice: Efficient Learning Mechanisms from a Cognitive Psychology Perspective and Their Empowerment by AI," Int. J. Asian Social Science Research, vol. 2, no. 6, 2025, doi: 10.70267/ijassr.250206.3137.

[11] B. Tabibian, U. Upadhyay, A. De, A. Zarezade, B. Schölkopf, and M. Gomez-Rodriguez, "Enhancing Human Learning via Spaced Repetition Optimization," Proc. Natl. Acad. Sci., vol. 116, no. 10, pp. 3988–3993, 2019, doi: 10.1073/pnas.1815156116.

[12] A. Hajja and A. J. Hunt, "A Template-Based Spaced Repetition Learning Solution Designed for Educators," in Proc. 2021 IEEE Frontiers in Education Conf. (FIE), 2021, doi: 10.1109/FIE49875.2021.9637051.

[13] M. K.-C. Yeh, A. Toshtzar, L. Guertin, and Y. Yan, "Using Spaced Repetition and Gamification to Enhance K-12 Student Science Literacy with On-Demand Mobile Short Reads," Penn State University, 2023, doi: 10.1109/FIE.2016.7757361.

[14] A. Voice and A. Stirton, "Spaced Repetition: Towards More Effective Learning in STEM," Research Directions, 2023, doi: 10.29311/ndtps.v0i15.3376.

[15] Y. Zhu and A. S. L. Fung, "Revisiting the Effects of Audio and Visual Inputs of Digital Flashcards on Memorization of Chinese New Words by Foreign Beginners," 2024, doi: 10.1109/ICEBEG.2011.5887162.

[16] J. Zhou, "Design of AI-based self-learning platform for college English listening," *2020 2nd International Conference on Machine Learning, Big Data and Business Intelligence (MLBDBI)*, Taiyuan, China, 2020, pp. 544-547, doi: 10.1109/MLBDBI51377.2020.00114.