OULUN AMMATTIKORKEAKOULU

DESIGN AND IMPLEMENTATION OF AN AI-DRIVEN GAMIFIED LEARNING WEB PLATFORM

An Evidence-Based Approach to Adaptive Educational Technology

Anil Shah
Matusala Gebrehiwot

Bachelor's Thesis
Autumn 2025
Degree Program in Information Technology
Oulu University of Applied Sciences

ABSTRACT

Oulu University of Applied Sciences
Degree Program in Information Technology (IT)
Option of Software Development

Author(s):
Anil Shah
Matusala Gebrehiwot

Title of thesis: Design and Implementation of an AI-Driven Gamified Learning Web Platform

Thesis supervisor(s): Teemu Korpela

Term and year of completion: Autumn 2025
Pages: 98 + 7 appendices

The project developed an adaptive web-based learning platform that combined personalized question difficulty, structured gamification, and AI-assisted feedback. The system applied a rule-based Education Difficulty Level (EDL) algorithm to adjust task difficulty-based performance-derived effective age, ensuring transparent, auditable adaptivity.

Gamification features – points, levels, streaks, and achievements – were implemented to support motivation without affecting assessment validity, while AI components provided hints and explanations as strictly optional learning support. The platform was built using Next.js and Supabase to ensure secure data handling, consistent authentication, and efficient server-side logic.

Informal user testing with volunteer learners indicated that the adaptive question flow and progressive gamified feedback improved engagement and sustained participation. The results suggest that lightweight, interpretable adaptive mechanisms can effectively support learning and can be integrated into broader online teaching environments.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

GLOSSARY

Concept	Definition
Adaptive Learning.	A learning approach in which the content, difficulty level, or learning path is dynamically adjusted based on the learner's performance.
API	A set of functions or protocols for software components to communicate with backend services and the database.
App Router.	The routing system in Next.js for handling navigation, layout, and data fetching in a unified Network
Assessment.	An Initial test to evaluate learners' existing knowledge that helps to determine their starting skill level in each subject.
Database Schema	The structured design of tables, columns, and relationships defines how data is stored and managed in the system.
Next.js	A React-based web development framework used to develop the prototype in the thesis, with both frontend and backend functionality implemented using serverless architecture.
Prototype.	An early version of the product for testing functionalities and concepts.

List of Abbreviations.

AI	Artificial Intelligence
API	Application Programming Interface
BKT	Bayesian Knowledge Tracing
CI	Continuous Integration
COVID-19	Coronavirus Disease 2019
CSS	Cascading Style Sheets
GPT	Generative Pre-Trained Transformers
IRT	Item Response Theory
K-12	Kindergarten (K) through 12th grade
RLS	Row-Level Security
SQL	Structured Query Language
UI	User Interface
ZPD	Zone of Proximal Development

1 INTRODUCTION

1.1 Background and Motivation

Digital learning has evolved from a supporting feature into a primary element of the education system today. Over the past two decades, global investment in education technology has increased dramatically. Furthermore, the COVID-19 pandemic accelerated the adoption of remote and hybrid learning models [1]. Although online platforms offer unprecedented accessibility, flexibility, and scalability, emerging research repeatedly points out a core weakness: most online learning environments fail to adapt to individual learner needs [2][3].

Students between 7 and 18 years old form a highly varied group with rapidly developing minds and/or significant cognitive growth. During these developmental years, the range in existing understanding, working memory, interest, and learning rate is quite wide among students [4][5]. Traditional instructional approaches and most contemporary digital systems are unable to adapt to this variability. When educational materials are too tricky, learners experience cognitive overload, frustration, and withdrawal; while if the educational materials are too simple, they become bored and disengaged [6][7].

The mismatch between learner capability and task difficulty is not a new pedagogical concern. Vygotsky's Zone of Proximal Development (ZPD) established that optimal learning occurs when learners are challenged slightly beyond their independent capability but supported sufficiently to ensure success [8]. Years of research in educational psychology back this idea, showing that learners grow most when they work in this "optimal challenge zone" [9][10]. Yet most e-learning systems ignore real-time performance signals and deliver only static difficulty progressions.

Alongside the ongoing teaching challenges of keeping learners motivated and engaged, especially online, where distractions are everywhere, and attention is fragile. Behavioral evidence shows that online learners, especially children and adolescents, often struggle to maintain focus and intrinsic motivation without clear

structure and emotional support strategies [11][12]. This motivational decline usually leads to dropouts, shallow learning, and reduced long-term retention.

Gamification emerged as a promising approach to counteract such disengagement. Decades of research suggested that points, badges, achievement systems, streaks, and progression mechanics features can boost motivation when they’re clearly connected to real learning goals [13][14][15]. However, when gamification is used superficially, with points or badges added without a pedagogical rationale, it has been shown to have limited or even adverse effects and can sometimes backfire [16]. Effective learning gamification requires alignment with educational objectives, personalized difficulty trajectories, and reinforcement of effort rather than mere performance [17].

These two areas of research, adaptive learning and gamification, have mainly developed separately, even though both address a core challenge of digital learning. Adaptive systems personalize the learner's experience, and gamified systems shape how the learner experiences it. Emerging research points to the idea that integrating these methods could address long-standing barriers in learner motivation, engagement, and achievement [18][19].

Integrating Artificial Intelligence (AI) into gamified digital learning environments offers a significant opportunity for addressing both personalization and motivation. AI-driven systems have demonstrated strong potential in adaptive assessment, real-time feedback, and personalized content recommendation [2][20].

With advancements in AI-driven personalization, particularly Bayesian Knowledge Tracing [21], Item Response Theory [22], Deep Knowledge Tracing [23] and reinforcement learning-based adaptivity [24] [25]; The possibility of creating platforms that dynamically tailor content to each learner’s performance has become feasible and cost-effective. The growing availability and accessibility of AI APIs, including the GPT-4o-mini model used in this project prototype, make it possible to deliver detailed, real-time feedback that guides learner strategies, eases frustration, and deepens understanding [20][26].

Despite these advancements, current e-learning platforms for young learners rarely combine AI-powered adaptivity, evidence-based gamification, and pedagogical personalization in a cohesive system. Most systems either rely on fixed difficulty levels, inconsistent gamification mechanisms, or provide generic feedback lacking the psychological grounding [2]. This creates an opportunity to design and evaluate a combined learning framework that incorporates insights from educational psychology, current AI methods, and gamification science.

This thesis builds on this research opportunity by designing and implementing, Hinura, an AI-driven, gamified learning platform for learners aged 7–18. The codebase, EDL system design documents, and gamification architecture reflect a research-driven implementation that operationalizes principles from ZPD, Knowledge Tracing, optimal challenge theory, spaced repetition, and growth-mindset pedagogy.

Hinura introduced a comprehensive system comprising:

- An EDL (Education Difficulty Level) adaptive engine
- Age-adaptive
- An achievement-based gamification ecosystem
- A dynamic streak, XP, and leveling system
- Real-time AI-generated hints, explanations, insights, and study recommendations

By integrating these features, the platform helps learners stay in their ideal challenge zone and remain motivated through meaningful, research-informed rewards. The objective was not to enhance immediate performance, but to support consistent practice, persistence, metacognitive awareness, and long-term engagement.

1.2 Problem Statement

Despite the rise of digital learning platforms, many systems remain limited in their ability to deliver personalized, motivating learning experiences, particularly for younger learners. Most traditional platforms present uniform sequences of exercises that do not adapt to a student's performance, resulting in a “one-size-

fits-all" experience [27]. Learners who struggle often encounter failure, which leads to frustration and withdrawal, while high-performing learners face insufficient challenge, causing boredom and a lack of engagement [28][29]. This misalignment is particularly concerning for learners aged 7-18 years, a developmental stage characterized by wide variability in cognitive capacity, sustained attention, and self-driven motivation [30][31].

Gamification can improve motivation, but its use in educational technology is often shallow and disconnected from real learning progress. Many platforms use points and badges as cosmetic add-ons rather than linking them to meaningful performance indicators or adaptive pathways [32]. Gamification loses its motivational power if it's not connected to meaningful learner behavior. Furthermore, rewards unrelated to effort or learning strategy can inadvertently reinforce fixed mindset behavior rather than promoting sustained engagement or mastery [33].

Artificial intelligence can support personalized learning, but many AI-enhanced platforms still focus primarily on recommending content or grading answers rather than providing comprehensive learner support. Systems that fail to integrate AI insights with motivational design often produce platforms that may be adaptive but still disengaging and cognitively inappropriate [2]. There remains a gap in solutions that combine evidence-based gamification, adaptive difficulty adjustment, and real-time AI feedback.

The Problem addressed by this Thesis can therefore be summarized as follows:

How can an e-learning platform effectively integrate adaptive algorithms, gamification, and AI-generated feedback to provide personalized, engaging learning experiences for users aged 7-18?

To address this question, this thesis develops and tests a working prototype (Hinura) that blends performance-driven adaptation, motivational reward elements, and AI-generated support. The goal of this combined approach was to tackle the two significant challenges young learners face in digital settings: a lack of personalization and weak motivation.

1.3 Research Objectives

The purpose of this thesis is to design, implement, and evaluate an AI-driven gamified learning platform addressing the challenges of engagement and personalization in online learning for learners aged 7-18 years old. Although much research highlights the value of adaptive learning [34], game-inspired features [14], and AI-driven feedback [2]. Research rarely brings all three together in one coherent instructional and technical approach. This thesis aims to fill this gap by focusing on both system design and empirical evaluation.

The specific objectives of this thesis project are:

1. **To create an adaptive mechanism** that continually tunes the difficulty of practice tasks based on how each learner is doing, keeping them working at a level that matches the Zone of Proximal Development and reflects current knowledge-tracing methods.
2. **To develop a comprehensive gamification framework** that incorporates streaks, points, levels, achievements, and progress indicators needed in motivational psychology and contemporary gamification research.
3. **To build a functional web-based prototype** with the usage of modern technology stacks – Next.js, TypeScript, Supabase, and OpenAI-based servers, helping us to integrate the adaptive learning, gamification, and AI-generated feedback.
4. **To evaluate the system through comparison,** assessing its impact on learning effectiveness, motivation, engagement, and user experience relative to a non-adaptive baseline model.
5. **To investigate how AI-generated hints, explanations, and guidance** contribute to learners' understanding, persistence, and the use of effective strategies, with particular attention to how these supports influence younger students.
6. **To contribute an implementation-ready reference architecture** for AI-driven gamified learning systems, offering reproducible design insights for future academic and professional development.

Collectively, these objectives support the creation of a platform that not only adapts to performance but also actively sustains motivation, providing constructive, age-appropriate feedback. By drawing on findings from educational psychology, the learning sciences, and gamification studies, this thesis aimed to produce a blueprint for next-generation digital learning platforms that meet the needs of varying learners across the K-12 continuum.

1.4 Research Questions

Three primary research questions were developed to guide this study. Those questions reflect the dual pedagogical goals of the platform, personalization and engagement, as well as the technical aim of integrating adaptive algorithms, gamification structures, and AI-based support into a coherent system.

Research Question 1 (RQ1)

Does the adaptive algorithm improve learning outcomes compared to a non-adaptive baseline?

This question explored whether adjusting difficulty in real time leads to noticeable gains in accuracy, performance stability, fewer errors, or stronger overall mastery. Earlier research suggested that adaptive learning improves performance by maintaining learners within an optimal challenge region, and related studies show that answer-based systems offer modest support, intelligent tutor provides more substantial gains, and human tutors deliver the most significant impact [34][35]. Yet its actual impact in practice often hinges on how well the approach is designed and how different learners respond to it.

Research Question 2 (RQ2)

Do gamification elements notably strengthen learner motivation and engagement within the platform?

Gamification has the potential to strengthen persistence and enjoyment, though its effectiveness often depends on how well the design is executed and how closely it follows established principles of motivation [13][14]. This question investigated whether the implemented system meaningfully affects engagement

behaviors such as question completion, time spent, return frequency, and perceived enjoyment.

Research Question 3 (RQ3)

What can we learn about how adaptive learning, gamification, and AI-generated feedback work together to support personalized learning for students between the ages of 7 and 18?

This question synthesized the effects of the system's three pillars, exploring how these components interact in real use. The integration of AI hints, explanations, and insights introduces a novel dimension in K-12 educational systems, and this research sought to understand how these mechanisms influence persistence, strategy use, and perceived helpfulness.

Collectively, these research questions assess the system both as a technical implementation and as an educational intervention. Through answering them, the thesis aimed to provide evidence-based contributions to the design of adaptive, engaging, and AI-enhanced learning platforms.

1.5 Scope of the Study

This thesis presents the design, implementation, and evaluation of an AI-driven, gamified learning platform tailored for learners aged 7-18. Although the broader goal for systems like this includes large content libraries, classroom-wide use, and alignment across curricula, this thesis limited its focus to a manageable scope that allowed for a thorough and achievable investigation within the available timeframe.

The prototype developed in this study focuses on three core subject domains commonly emphasized in basic education – mathematics, English, and science. These areas were chosen for their frequent use in adaptive learning research and for their suitability for measuring changes in progression and difficulty. Each subject included exercises at three difficulty levels (easy, medium, hard), and a curated question bank of approximately five hundred items was used to facilitate controlled experimentation. Although the system was intended for learners aged 7-18, the evaluation will likely be conducted with older students or grown-up

participants. This approach avoided the ethical and regulatory challenges that typically arise when conducting research directly with minors.

The system's primary contributions lie in three intertwined domains:

1. Adaptive difficulty adjustment through the Education Difficulty Level (EDL) algorithm
2. Gamification features such as points, levels, streaks, and achievement systems
3. AI-generated support through hints, explanations, and personalized learning insights.

The thesis did not aim to create a complete online learning ecosystem. Instead, it explores the practical and educational value of integrating adaptation, motivation, and AI-based support into a single, unified system. Consequently, large-scale A/B testing, long-term learning retention, collaborative interpersonal features, and teacher-driven analytics fell outside the scope of this initial prototype.

Given the study's focus on rigorous experimentation, the evaluation contrasted outcomes from adaptive settings with those from non-adaptive setups. Performance metrics, engagement indicators, user satisfaction, and qualitative feedback were used to evaluate system effectiveness. While the results provided useful indications for potential classroom application, they essentially captured behavior within a controlled prototype setting. They should not be interpreted as evidence of sustained or large-scale effectiveness.

1.6 Significance of the Study

The significance of this study can be understood from multiple perspectives: learners, educators, researchers, and developers. The digital learning environment continues to evolve, yet fundamental challenges persist – namely, sustaining motivation, preventing disengagement, and providing content that adapts to individual needs. By integrating adaptive learning, gamification, and AI-powered feedback, this thesis addressed these challenges in a structured, evidence-based manner.

Significance for Learners

For learners aged 7–18, the platform provided personalized support designed to keep them challenged without being overwhelmed. Adaptive difficulty ensured that exercises respond to performance in real time, reducing frustration when tasks are too tricky and boredom when those are too easy. The gamification elements reinforced effort and persistence, which encourages learners to develop positive learning habits. Meanwhile, AI-generated hints and explanations provided supportive scaffolding, promoting a deeper understanding and self-correction.

Significance for Educators

Educators are increasingly adopting digital platforms to expand their instructional strategies, deliver more personalized learning pathways, and gain clearer insight into student performance. The system designed in this thesis demonstrated a practical model for blending technological components into a unified structure. It highlighted how digital systems can support differentiated instruction, augment teacher capacity, and offer insight into learner behaviors. Even though the prototype was not intended for direct classroom use, the architecture and evaluation provided valuable implications for future educational technology development.

Significance for Researchers

This research contributes to the growing body of literature on adaptive learning, gamification, and AI-enhanced education. Prior studies often explore the domains separately; fewer examine their interactions within a single platform. By evaluating performance outcomes, engagement patterns, and user perceptions, the thesis provides empirical evidence to guide future research on integrated learning systems. The research questions also addressed gaps identified in recent meta-analyses, particularly concerning the role of adaptive algorithms in combination with motivational components for younger learners.

Significance for Developers and System Designers

From a technical perspective, this platform offers a reusable model for developing scalable adaptive learning systems using modern web technologies such as Next.js, TypeScript, Supabase, and OpenAI's GPT models. The system architecture, which includes the backend design, EDL-based adaptation logic, gamification data structures, streak mechanisms, and AI service endpoints, provided a practical reference for engineers and designers working in educational technology. The prototype illustrated how these components can be integrated cohesively and effectively while preserving strong performance, cost efficiency, and a high-quality user experience.

Overall, the study's significance lies in its multidisciplinary nature, which bridges cognitive psychology, motivational science, artificial intelligence, and modern software engineering to propose and validate an approach to enhancing learner engagement and personalization in digital environments.

2 METHODOLOGY

2.1 Research Design and Approach

This study followed a design-science and prototype-oriented research approach. The primary aim was to design, implement, and empirically probe an AI-driven, gamified learning platform that supported adaptive difficulty and motivational feedback for learners aged 7-18. Instead of a large-scale controlled trial, the work was focused on building a technically coherent artifact and examining whether its core mechanisms behaved as theoretically intended and were perceived as meaningful and understandable by users.

Consistent with design-science traditions in educational technology, the study followed an iterative build-test-refine process. Foundational concepts such as the Zone of Proximal Development, flow and optimal challenge, self-determination theory, and growth mindset were translated into specific technical features and subsequently examined through small-scale empirical testing.

Methodically, the study combined:

- **Engineering work:** specification and implementation of the web platform (adaptive algorithm, gamification engine, AI feedback, and data infrastructure).
- **Analytical work:** formalization of the Education Difficulty Level (EDL) algorithm and gamification rules into reproducible logic
- **Empirical work:** small-scale evaluation with volunteer participants using a pre-test/practice / post-test sequence, complemented with log data and brief subjective feedback.

The core methodological focus was not to compare the platform against a commercial system but to determine whether a transparent and explainable adaptive-gamified design, grounded in contemporary research, could be implemented within a modern web framework and demonstrate internally consistent behavior as well as encouraging effects on performance and perceived motivation when compared with a non-adaptive baseline.

From a temporal perspective, the project proceeded in three broad phases:

1. **Conceptualization and requirements derivation**, based on prior literature on adaptive learning, gamification, and AI-supported tutoring
2. **System design and Implementation**, where the EDL algorithm, gamification rules, and AI components were embedded in a single Next.js / Supabase codebase
3. **Prototype evaluation**, in which a small convenience sample of participants interacted with an adaptive, gamified version of the platform versus the non-adaptive baseline, allowing descriptive comparison of performance and experience.

The research questions defined in chapter 1 guided both the technical choices and the empirical design, especially the comparison between adaptive and non-adaptive practice and the role of gamification elements in supporting engagement.

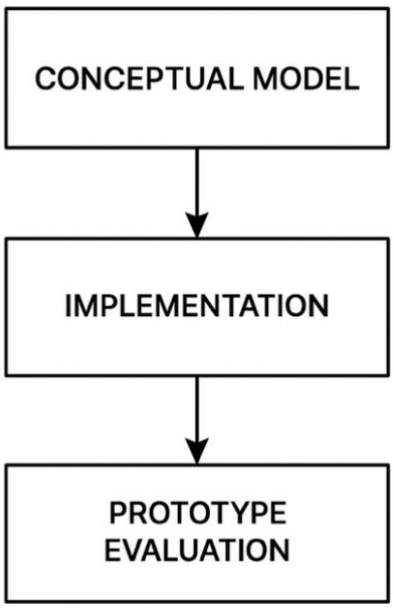


FIGURE 1. Early Design Concept

2.2 System Architecture and Technology Preference

The methodological choices regarding the system architecture were driven by three constraints: 1. The need to express educational logic transparently and

reproducibly, 2. The need to log interaction data reliably for evaluation, and 3. The practical limits of a bachelor-level thesis project in terms of time, maintenance, and deployment.

For these reasons, the platform was implemented as a monolithic web application with the usage of Next.js 15 (App Router) with TypeScript for both client and server code, backed with a Supabase project which provided PostgreSQL database, authentication, and Row-Level Security (RLS). This architecture brought the user interface, API route handlers, and business logic into one unified codebase, which reduced the integration complexity and made it easier to follow how theoretical concepts such as “effective age”, “streak freeze”, and “achievement unlock” were mapped onto SQL schemas and implemented in the corresponding route handlers.

The use of Next.js enabled the platform to blend server-side and client-side rendering, resulting in fast page loads and a responsive dashboard while ensuring that sensitive processes, such as question selection, scoring, EDL updates, and gamification logic, remain securely on the server. Supabase was chosen for its robust support for RLS, SQL triggers, and managed PostgreSQL, all of which were critical for implementing privacy-aware data storage and enabling automatic profile initialization for authenticated users.

To maintain a consistent “Bloom Humanist” design system, Tailwind CSS and Shadcn/ui components were used for styling. This choice was primarily methodological: it allowed visual consistency and rapid iteration on interface details without fragmenting attention away from the core research questions.

From a methodological standpoint, the architecture was designed to guarantee three key conditions:

1. Every learner interaction was captured on the server in a structured format suitable for subsequent analysis
2. All adaptive and gamification decisions could be traced directly to deterministic functions or stored procedures rather than to opaque black-box models.

3. The same codebase that was evaluated empirically could be inspected and reasoned about system design and implementation, maintaining a tight link between theory, design, and evaluation.

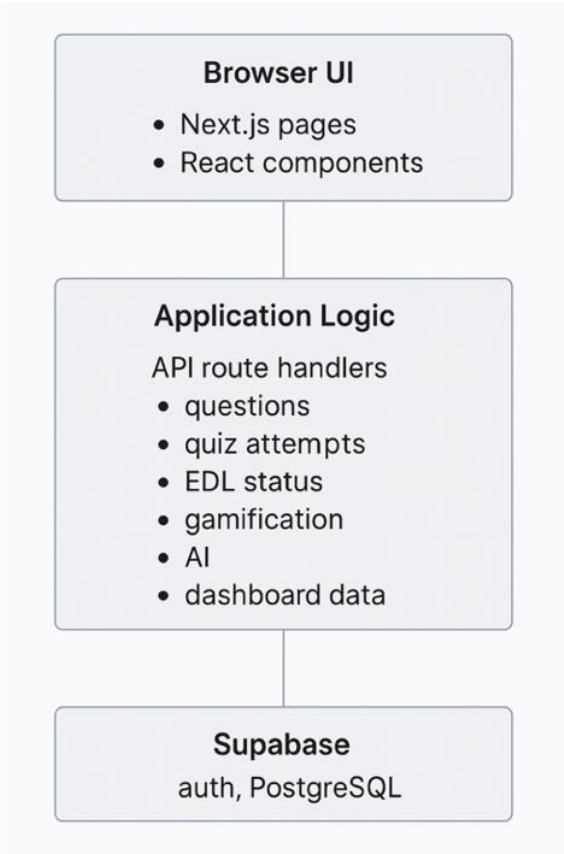


FIGURE 2. Logical Architecture and Technology Used

2.3 Adaptive Learning Method: Education Difficulty Level (EDL)

The adaptive behavior of the platform was governed by an Education Difficulty Level (EDL), an algorithm that operationalized ideas from the zone of Proximal Development, flow, and optimal challenge, and knowledge-tracing style performance monitoring [13][21].

Rather than directly modelling latent knowledge with a complex probabilistic model such as Bayesian Knowledge Tracing, the EDL approach used a transparent and rule-based mechanism with the following core elements:

1. **Baseline assessment and subject-specific proficiency.**
   Each learner completed an initial assessment consisting of short multiple-choice quizzes in three subjects (mathematics, English,

science). Items get varied in difficulty level (easy, medium, hard) and include target age ranges. For each subject, accuracy was calculated by platform and mapped to an initial proficiency state that seeded the EDL metrics for that subject.

2. **Effective age and performance adjustment**
   The EDL system did not treat chronological age as the predominant determinant of difficulty. Instead, it computed an “effective age” through adjustment of the learner’s chronological age with a performance-based offset derived from recent accuracy and quiz history. High performers received a positive adjustment, while struggling learners received a negative adjustment, in line with the idea that challenge should be slightly above, but not far beyond, the current capability. This produced a small integer adjustment band (e.g., -2 to +2 years), which was later used when filtering candidate questions.

3. **Difficulty distribution and question sampling**
   When a learner requested an adaptive practice quiz, the API combined the current EDL metrics (recent accuracy, effective age, and quizzes completed) with a difficulty distribution rule. High accuracy shifted the mix towards more complicated questions, while low accuracy shifted towards easier questions, and intermediate performance produced the balanced blend. This ensured that over time, the learner experienced a curated mixture of easy, medium, and hard questions instead of a uniform random sequence.

4. **Exclusion of recently seen items.**
   For the reduction of repetition and support spaced practice, the question selection logic excluded items that the learner had answered in the previous 30 days. This relied on the quiz attempts log table and simple SQL filtering.

5. **Continuous update of EDL metrics.**
   After each quiz submission, the system recomputed subject-specific EDL

metrics: updated recent accuracy, effective age, performance adjustment, EDL status label (e.g., “building foundation”, “perfect zone”), and quiz counts. These metrics were then exposed through a dedicated API endpoint and displayed in the user’s dashboard.

The rationale for this approach was twofold. First, a rule-based EDL mechanism could be implemented, tested, and verified within the scope of the thesis without sacrificing explainability. Second, the use of effective age and difficulty distributions enabled explicit alignment with ZPD and flow theory at the code and database levels, making the connection between educational psychology and implementation auditable.

The EDL algorithm acted as the independent adaptive factor in the evaluation: the adaptive condition used EDL-based question choice, whereas the baseline condition used a fixed difficulty distribution that ignored performance history.

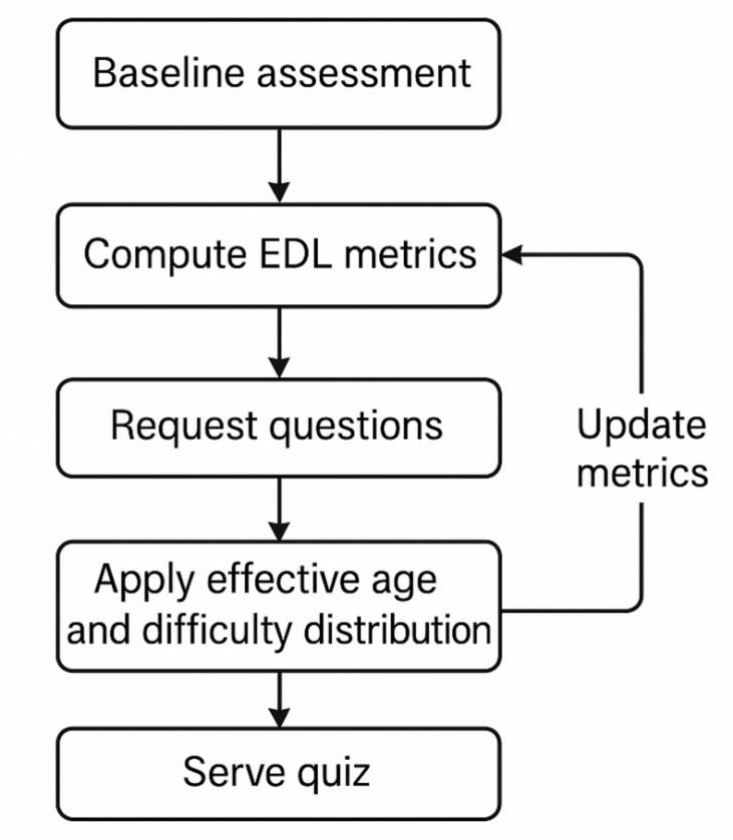


FIGURE 3. Adaptive Learning Method: Education Difficulty Level (EDL) Loop

2.4 Gamification Design as Motivational Intervention

In this thesis, Gamification was treated as a deliberate motivational intervention, rather than a cosmetic layer. The design of levels, points, streaks, and achievements drew on self-determination theory – research on points-and-badges systems and empirical work on gamefic learning in schools [13][33].

The gamification system encompassed four primary constructs:

1. **Points and dynamic multipliers**
   The platform awarded base experience points (XP) for correct answers and for completing the quiz, with additional bonuses for perfect scores. Those base points were then scaled by two multipliers: a streak multiplier and a subject level multiplier. The streak multiplier is based on the length of the learner's daily activity streak, and the subject level multiplier is based on the learner's level in that subject. This created a modest but meaningful reward gradient, with sustained engagement and gradual mastery yielding higher effective rewards.

2. **Subject-specific and overall levels.**
   Each subject maintained its own XP pool and level, while an overall level summarized overall progress. The level progression followed a square-root curve, where early levels required relatively few points (to secure early wins) and later levels required more points, reflecting the idea that mastery takes sustained effort.

3. **Streak mechanics and freeze protection.**
   The system tracked the number of consecutive days with a learning activity (answering at least one question). Streaks increased each day modestly and displayed prominently in the dashboard with simple visual cues such as a flame icon. A "streak freeze" mechanism was implemented to prevent a single missed day from undoing long-term progress, in line with research suggesting that overly harsh penalties can undermine motivation rather than reinforce it.

4. **Achievements and celebratory feedback.**
   A set of predefined achievements (such as completing initial assessment, reaching subject level milestones, achieving perfect scores, and maintaining a weekly streak) was predefined in the database. The system checked for newly unlocked achievements after each key event and, when appropriate, presented a dedicated model with visual celebration and XP rewards.

These choices were motivated by several methodological considerations. First, they allowed the platform to operationalize competence support (mastery cues and clear feedback), autonomy support (the ability to choose subjects and difficulty modes within constraints), and related constructs from the growth mindset literature, such as rewarding effort and persistence rather than only raw performance. Second, all gamification logic was encoded in transparent rules and SQL tables, allowing inspection and reporting of the exact mechanics used during the evaluation.

In the empirical phase of the study, these gamification elements were treated as a package: participants always used the platform in its gamified form, but their perceptions of points, streaks, and feedback were explicitly probed in the post-session feedback.

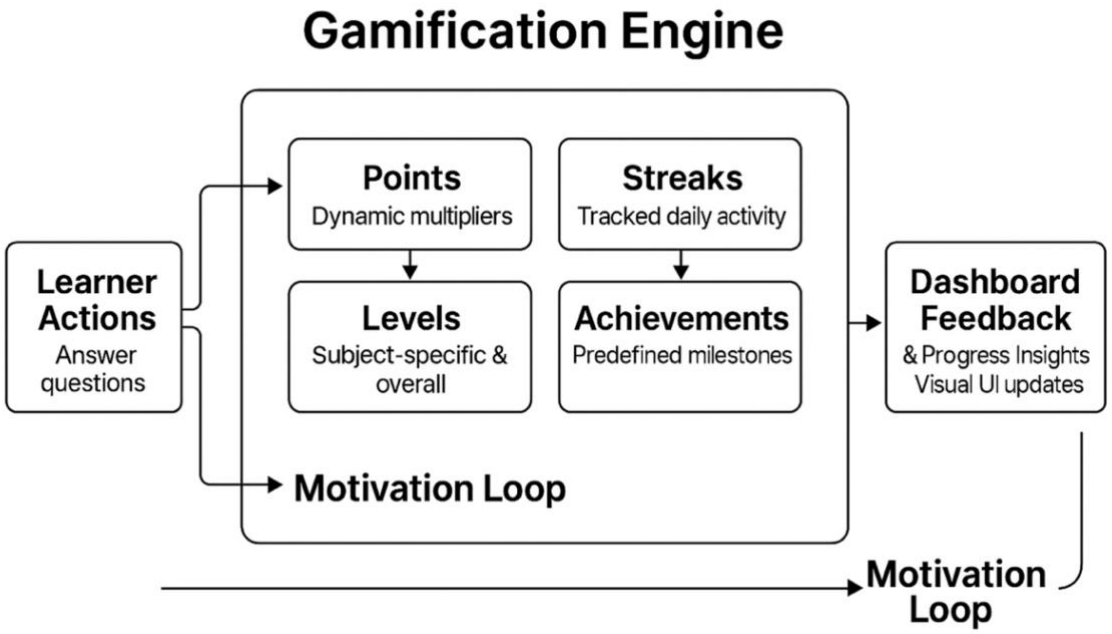


FIGURE 4. Gamification Engine and Motivation Loop

2.5 AI-Supported Feedback Components

The designed system integrated OpenAI-based services to generate real-time hints, answer explanations, and higher-level learning recommendations, using lightweight models and strict constraints. Methodically, AI was not used for core assessment or grading. Instead, it was used to provide scaffolded feedback, which would be challenging to hand-author exhaustively.

The integration was structured around several guiding design principles:

1. **Non-leaking hints.**
   Hints were designed to support problem-solving without revealing the correct answer directly. The hint endpoint hence received the question, options, subject, and difficulty metadata along with age information. It was instructed (via carefully engineered prompts) to respond with one or two guiding sentences rather than complete solutions.

2. **Short, Structured Explanations.**
   After a learner submitted a response, an explanation endpoint was triggered only when an incorrect answer was given, providing a brief

comparison between the learner's choice and the correct solution. The model was constrained to a short word limit and to use growth-mindset-consistent language (emphasizing strategies and next steps rather than fixed ability labels).

3. **Personalized recommendations and insights.**
   For Learning and Progress, AI endpoints were available to derive high-level study recommendations and reflective summaries from recent quiz history and EDL status. These were optional, but they demonstrated how AI could transform raw log data into learner-friendly narratives.

To control cost, performance, and safety, the AI layer applied rate limiting and light caching on the server side, and all responses were requested in a structured JSON format to simplify validation. The role of the AI components was to provide realistic, research-informed feedback in the prototype and to allow exploration of how such feedback might contribute to motivation (RQ3), even if rigorous isolation of AI effects was beyond the scope of this thesis.

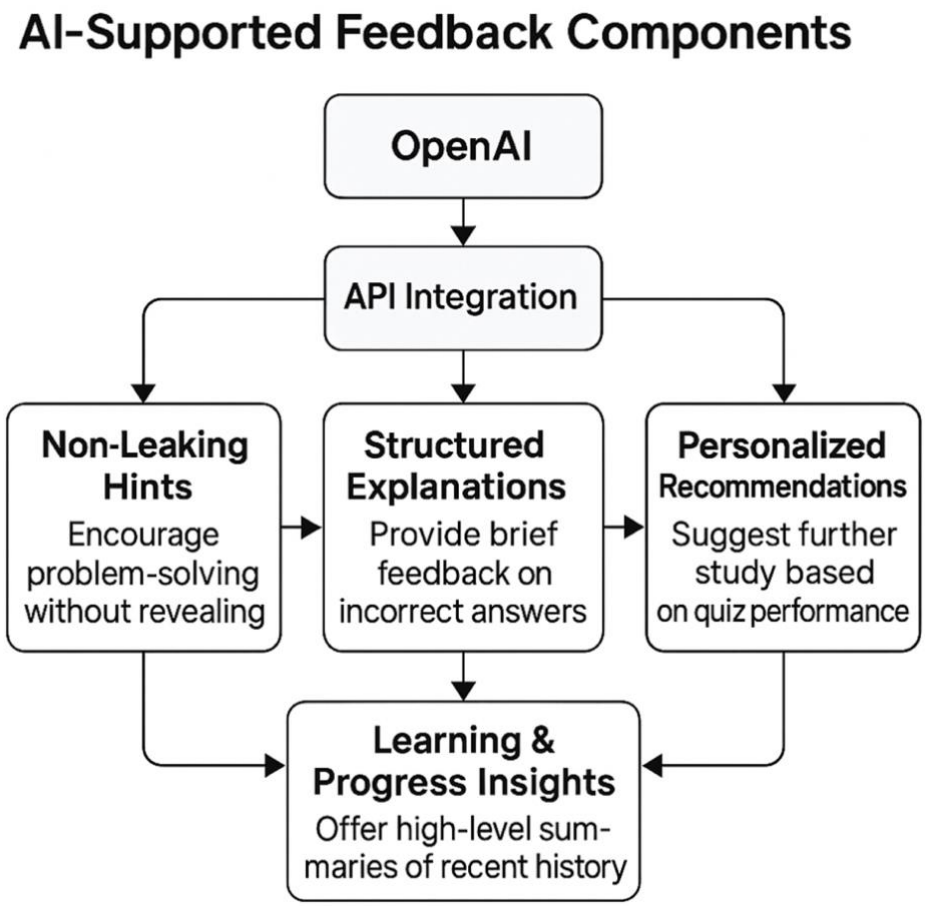


FIGURE 5. AI-Supported Feedback Components

2.6 Data Collection and Instrumentation

Data collection focused on two complementary sources: system log data and self-reported user feedback. All data were collected via the deployed prototype, with Supabase handling authentication and secure storage.

For each quiz session, System logs are captured:

- User identifier (pseudonymous Supabase user ID)
- Subject and quiz type (assessment or practice)
- List of questions served (IDs only) and their difficulty
- Learner's selected options and correctness
- Timestamps are sufficient to derive time-on-task at the item and quiz level

- Derived metrics such as score percentage, points awarded, streak updates, and any level-up or achievement events

These data were stored in the structured tables described in Chapter 3 and served as the basis for the quantitative analysis of accuracy, progression, and system behavior.

In addition to log data, each evaluation session concluded with a short feedback phase. Participants were asked to comment on:

- The perceived appropriateness of difficulty
- The clarity and usefulness of points, level, streaks, and achievements
- Their perceived motivation or engagement while using the platform relative to their typical experience with non-gamified practice.

Given the small scale and exploratory nature of the evaluation, this feedback was collected informally through brief oral discussion and, where possible, written notes rather than a lengthy standardized questionnaire. The goal was to capture qualitative impressions that would help interpret the quantitative indicators, rather than to construct a complete psychometric instrument.

2.7 Experimental Design and Procedure

To address the research questions, particularly RQ1 (adaptive vs non-adaptive performance) and RQ2 (perceived motivational value of gamification, the evaluation was conducted as a small-scale, within-subject experiment using the implemented prototype.

A convenience group of volunteer participants was obtained by sharing the prototype web application with friends, classmates, and colleagues. In keeping with ethical guidelines and institutional practice, these participants were older students and grown-ups who served as stand-ins for the intended 7-8 age group. This approach enabled the core adaptive and gamified mechanisms to be examined without directly involving minors.

Each session followed a fixed structure that typically lasted around 15 minutes:

1. **Onboarding and Consent**

   Participants were informed about the study's purpose, the prototype nature of the system, the logging of their interactions, and their right to withdraw at any time. After providing informed consent, they created a Supabase-backed account, including a birthdate field, which was required by the platform.

2. **Initial assessment and EDL seeding**

   Participants completed the built-in assessment, which produced subject-specific accuracy measures and their initial EDL metrics. This stage simulated the intended boarding flow for actual learners.

3. **Practice phase with two conditions**

   The core of the evaluation involved two practice blocks: one using the adaptive EDL-based question and the other using a non-adaptive baseline with a fixed difficulty distribution that ignored performance history. The order of these blocks varied across participants, wherever feasible, to mitigate simple order effects. In both conditions, the gamification layer was active as it was considered a defining part of the platform rather than an experimental factor to be removed.

4. **Practice with AI support**

   Participants were informed that they could request a hint whenever a question felt too complicated, and that an explanation would be available if they selected an incorrect answer. They were also encouraged to review the AI-generated insights at the end of the session to understand their overall progress better.

5. **Post-session feedback**

   Finally, participants were asked to reflect on perceived difficulty, the clarity of the adaptive behavior, and their experience of points, streaks, and visual progress feedback. They were also asked to evaluate the usefulness and accuracy of the AI-generated hints, explanations, and

insights in supporting their learning.

The within-subject design had two significant advantages. First, it reduced variability due to individual differences, because each participant served as their own control when comparing adaptive and non-adaptive practice. Second, it was compatible with the small sample size that was realistic within the project's time frame.

2.8 Measures and Data Analysis

The quantitative analysis was grounded in a small set of simple, interpretable measures derived from the log data, supplemented by qualitative observations from the feedback phase.

For RQ1 (**impact of adaptivity**), the main measures were:

- Accuracy expressed as a percentage of correct answers. This allowed inspection of short-term learning gains at the level of the whole session.
- Practice accuracy in adaptive vs non-adaptive blocks, comparing the proportion of correct answers under each condition.
- Evolution of difficulty distribution, checking whether learners in the adaptive condition gradually received a higher proportion of medium and hard items when performing well, and easier items when struggling, as predicted by the EDL logic.

For RQ2 (**motivation and engagement**), the analyses considered were:

- Behavioral indicators such as voluntary continuation within a practice block, time spent on tasks, and the occurrence of streak continuation across days for participants who use the system more than once.
- Subject impressions, focusing on whether participants described the gamification elements as motivating, neutral, or distracting, and whether they felt rewarded for effort rather than punished for mistakes.

For RQ3 (**overall integration of AI, gamification, and adaptivity**), the analysis focused more on coherence: whether participants perceived the feedback from

AI-generated hints and explanations as supportive, whether the adaptive shifts in difficulty were noticeable and reasonable, and whether the progress indicators matched their experience.

Given the small sample size, the analysis remained primarily descriptive. Where beneficial, simple statistics such as means, standard deviations, and within-subject comparisons were considered, but no large-scale inferential claims were made. Instead, the emphasis was placed on whether the empirical data were consistent with the intended design behavior of the EDL and gamification systems.

2.9 Reliability and Validity Considerations

Although the study was exploratory and small-scale, attention was given to reliability and validity.

Key constructs such as “adaptive behavior”, “difficulty level”, and “streak” were tightly coupled to explicit rules and database fields. This approach reduced the likelihood of ambiguous operationalization, as the same code governing the learner experience also generated the data used for analysis. As a result, EDL metrics and gamification variables remained internally consistent across both the platform and the evaluation process.

Internal validity was supported by the within-subject comparison of adaptive and non-adaptive practice, which reduced between-participant variance. However, it was limited by a small sample size, non-random sampling, and the inability to counterbalance practice blocks across all participants perfectly. Learning and fatigue effects across the session were therefore acknowledged as possible confounds.

Reliability of the performance measures was supported by automatic logging: correctness, timestamps, and points were captured directly by the system, without manual transcription. Question difficulty was established in advance during the development of the item bank. Still, the relatively small number of items within each difficulty tier limited the scope for formal item analysis. As a result, the tests were not treated as high-stakes, exam-reliable measures; instead, they

were used as indicative assessments to provide a general sense of performance change.

Finally, ecological validity was intentionally limited. The evaluation did not take place in a real classroom with minors over extended periods, but in short sessions with stand-in participants. The main goal was to test whether the mechanisms worked as designed and whether they appeared promising in terms of user experience, not to claim a generalizable effect at scale.

2.10 Ethical and Data Protection Considerations

The study included interaction with a learning platform that logged user performance data. Even though the evaluation used older students and grownups rather than children, the platform itself was designed for a younger audience; therefore, privacy and ethics were central methodological concerns.

First, informed consent was obtained from all participants before data collection. They were informed about the prototype's working principle, the logging of their actions for study purposes, the voluntary nature of participation, and their right to withdraw without consequences. No sensitive personal information beyond email address, name, and birthdate (for age calculation and age-appropriate UI) was required.

Second, the database schema and Supabase configuration followed a data minimization principle. Only fields needed for adaptive logic and evaluation were stored, and learner identifiers in the analytic tables were based on Supabase user IDs rather than real names. Row-level security policies ensured that users could read or update only their own data, and that only predefined “public” tables, such as question and achievement definitions, were publicly readable.

Third, the usage of AI services raised additional privacy considerations. No raw personal identifiers were sent to the OpenAI API; only question content, anonymized answer information, and coarse age-related parameters were included in AI API requests. The AI endpoints were implemented on the server, so the client device did not communicate directly with external AI providers.

Finally, the evaluation was conducted in accordance with the standard ethical practices for small-scale student projects at **Oulu University of Applied Sciences**. The design explicitly avoided involvement of minors and avoided any stakes or grading consequences for participants. The main risk to participants was minimal and related to mild frustration if some items felt too complicated; this risk was mitigated by clear communication, optional hints, and the ability to stop at any time.

2.11 Methodological Limitations

The small and convenience-based sample limited the generalizability of the empirical findings and restricted the use of advanced statistical analysis. The short-term, lab-like evaluation setting differed substantially from sustained classroom use, where motivation dynamics and streak mechanics might behave differently.

Moreover, the study evaluated a package of interventions – adaptive question selection, gamification, and AI-based feedback – rather than isolating each factor in a factorial design. As a result, the methodology supported conclusions about the internal coherence and perceived promise of the integrated platform, but not about the separate effect sizes of individual components.

Finally, while the EDL and gamification mechanisms were grounded in established theories and recent empirical work, they remained relatively simple compared with state-of-the-art adaptive systems in large commercial platforms. This simplification was necessary for transparency and feasibility. Still, it also meant that the study should be interpreted as a proof of concept for an explainable, research-aligned design rather than a definitive evaluation of optimal adaptive learning algorithms.

3 SYSTEM DESIGN AND IMPLEMENTATION

3.1 System Design and Development Overview

The system was designed as an integrated, monolithic web platform that demonstrated how adaptive learning, gamification, and AI-supported feedback could be operationalized within a single coherent architecture. The design philosophy prioritized clarity, traceability, and technical sufficiency, ensuring that each behavior described in the methodology had a direct implementation in the codebases. Every mechanism – difficulty adaptation, level progression, streak maintenance, achievement checking, and AI-based feedback – was implemented in a way that remained inspectable, reproducible, and verifiable.

The platform employed rule-based algorithms, explicit SQL logic, and type-safe TypeScript modules, instead of probabilistic or machine-learning-driven models, which would have increased system opacity or implementation overhead. For example, the adaptive engine (EDL) avoided full Bayesian Knowledge Tracing and instead relied on deterministic rules implemented in `/lib/edl/calculator.ts` and `/lib/edl/selector.ts`, ensuring that every adaptive decision could be tied to a specific variable, such as recent accuracy, performance adjustment, and effective age.

The platform followed a three-tier logical architecture [36], all contained within a single Next.js 15 codebase:

1. **Presentation Layer**
   The client-side interfaces (React components under `/app/auth` and `/app/dashboard`) handled user interaction, quiz rendering, progress display, and real-time visual feedback. Components remained self-contained and mapped directly to the functional requirements, such as assessment, adaptive practice, progress tracking, and gamified displays.

2. **Application Layer (Serverless API)**
   Next.js Route Handlers under `/app/api` implemented all core logic. It included adaptive question generation, handling quiz submission,

computing EDL metrics, managing streaks and levels, awarding achievements, and orchestrating AI endpoints. The serverless model ensured stateless execution and automatic scaling while keeping the implementation accessible and auditable.

3. **Data and Security Layer (Supabase PostgreSQL)**
   The Supabase project provided authentication, SQL functions, row-level security, database storage, and triggers. All persistent state – profile, quiz attempts, EDL metrics, streak records, point transactions, and achievements – were stored in PostgreSQL tables. RLS ensured that users could access or modify only their own data and triggered automated profile creation and the initialization of adaptive fields.

It was intentional to choose a monolithic Next.js architecture rather than a separate backend API [37]. It reduced integration complexity, simplified debugging, and allowed the thesis to document a complete end-to-end system within a contained scope. This also ensured that each chapter of this thesis could map directly to the codebase.

Development progressed iteratively. Early milestones established the authentication pipeline, database triggers to automate profile creation, and the complete 18-24 question assessment. Once the baseline assessment and EDL initialization were functional, adaptive practice logic, point allocation, streak handling, and achievement monitoring were incrementally added. Only after these mechanisms were established and stable were AI-based hinting, explanation, and recommendation features integrated, with strict constraints to prevent model leakage and maintain pedagogical fairness.

3.2 Overall System Architecture

The system was implemented as a monolithic but modular full-stack application using Next.js 15 (App Router) for both client interface and server-side logic and Supabase PostgreSQL for authentication, storage, and Row-Level Security (RLS). This architecture allowed the adaptive learning workflow, gamification engine, AI feedback components, and user progress tracking to operate within a

unified pipeline, ensuring traceability between design decisions and code implementation.

The architecture followed a three-layered separation – Client layer, Application layer, and Data layer within a single repository. This permitted rapid iteration, consistent type-safety, and easier debugging because every component of the learning cycle (assessment → EDL computation → adaptive quiz → gamification update → dashboard display) was implemented and evaluated in one cohesive system.

3.2.1 Design Rationale

Several practical requirements shaped the architectural choices:

1. Single-repository execution model

Using a monolithic Next.js project ensured that UI components, API routes, and utilities were defined within a single structure. The adaptive algorithm and gamification logic could be inspected without indirection. Serverless APIs leveraged built-in Next.js functionality, eliminating the need for a separate backend stack. It ensured the development of an auditable, prototype-friendly system.

2. Strict data protection requirements

The platform stored performance history, streaks, points, and age-specific data. Supabase was selected because it offers PostgreSQL with Row-Level Security (RLS) by default, type-safe client libraries, and server-side authentication context. All sensitive logic in `/app/api/*` used the Supabase server client through `lib/supabse/server.ts`. This ensured that every API route executed under the authenticated user's identity, enabling RLS to enforce data isolation automatically.

```
export const createClient = async () => {
  const cookieStore = await cookies();
  return createServerClient(SUPABASE_URL, SUPABASE_ANON_KEY, {
    cookies: { getAll: () => cookieStore.getAll() }
  });
};
```

3.2.2 High-Level Architecture

The architecture consisted of three logical layers:

Layer 1: Client Interface (Next.js)

This layer consisted of all UI logic responsible for:

- Rendering the assessment, practice, progress, and profile interfaces
- Interacting with API routes for questions, quiz submissions, gamification updates, and AI feedback
- Displaying streaks, achievements, level progression, and points
- Showing adaptive difficulty effects through the question experience

The client layer remained intentionally thin. All correctness checking, scoring, EDL computation, and gamification logic were executed server-side for integrity and security.

Layer 2: Application Logic (Next.js API Route Handlers)

The server-side logic formed the core engine of the system and was implemented entirely under the `app/api/` directory. All the logic, such as adaptive, gamified, and AI-assisted behavior, originated on the server side. The route handlers were responsible for:

- Selecting appropriate questions
- Recording quiz attempts and updating EDL
- Awarding points and checking achievements
- Serving AI hints, explanations, recommendations, and insights

TABLE 1. API Functionality with their Endpoint and Description

API Functionality	Endpoint	Description
Question Selection	`/api/questions`	EDL-driven adaptive question selection
Quiz Evaluation	`/api/quiz-attempts`	Correctness checking, scoring, streak update
EDL metrics	`/api/edl/status`	Fetches effective age, performance adjustment, and status
Gamification Engine	`/api/gamification/*`	Points, levels, streak, achievement checks
AI Feedback	`/api/ai/*`	Hints, explanations, recommendations, insights
Auth Callback	`/api/auth/callback`	Supabase email verification handler
Dashboard aggregation	`/api/dashboard/data`	Consolidated learner stats

Each handler used the Supabase server client (`lib/supabase/server.ts`), which executed database queries under RLS constraints and returned strongly typed data.

Layer 3: Supabase (Auth, Database, RLS, Triggers)

Supabase functioned as an auth provider (email/password, OAuth for Google), a database provider, a security provider (RLS and policy enforcement), and a trigger engine (profile initialization, streak reset). RLS policies ensured users could access only their data. All persistent entities, such as profiles, quiz attempts, EDL metrics, and so on, were stored in tables.

Key tables implemented:

TABLE 2. Database table with its purpose

Table	Purpose
`profiles`	User Profile + global gamification stats
`questions`	Item bank (subject, difficulty, age range)
`quiz_attempts`	Assessment + practice submission history
`user_performance_metrics`	EDL metrics per subject
`point_transactions`	Full XP audit log
`user_achievements`	Achievement unlock history
`level_history`	Subject-level progression records
`streak_milestones`	Streak-based rewards

3.2.3 Request-Response Workflow

The entire system operated through a predictable pipeline.

Example Workflow A: Adaptive Question Request

Step 1 – UI request

- `practice/page.tsx` call

Step 2 – Server logic

- `/app/api/questions/route.ts`
- Retrieves the user's EDL metrics
- Computes effective age and difficulty distribution
- Filters candidate questions
- Excludes questions answered in the last 30 days
- Returns the final question set

Step 3 – Database operations

Supabase queries tables:

- `user_performance_metrics`
- `questions`
- `quiz_attempts`

Example Workflow B: Quiz Submission

Step 1 – Client → POST

- `/api/quiz-attempts`

Step 2 – Server logic

- Correctness evaluation
- Score calculation
- EDL metric precomputation
- Streak update
- Level calculation
- Achievement scanning
- XP transaction logging

Step 3 – Response

Server returns:

- Score percentage
- Updated streak
- XP awarded
- Any unlocked achievements

3.3 Data Model and Schema Design

The data model was implemented in PostgreSQL via Supabase and was designed to support three main concerns: storing learner identity, logging learning events for adaptation, and maintaining a transparent gamification state. The schema remained compact yet expressive, enabling adaptive decisions, streak

logic, and XP calculations to be executed close to the data while remaining easy to reason about.

At a high level, the database was structured into three groups of tables:

- Profiles and identity, which extended Supabase auth. users with learning-specific information
- Learning and performance, which stores questions, quiz attempts, and the summarized EDL metrics per subject
- Gamification, which tracked points, levels, achievements, streaks, and their history.

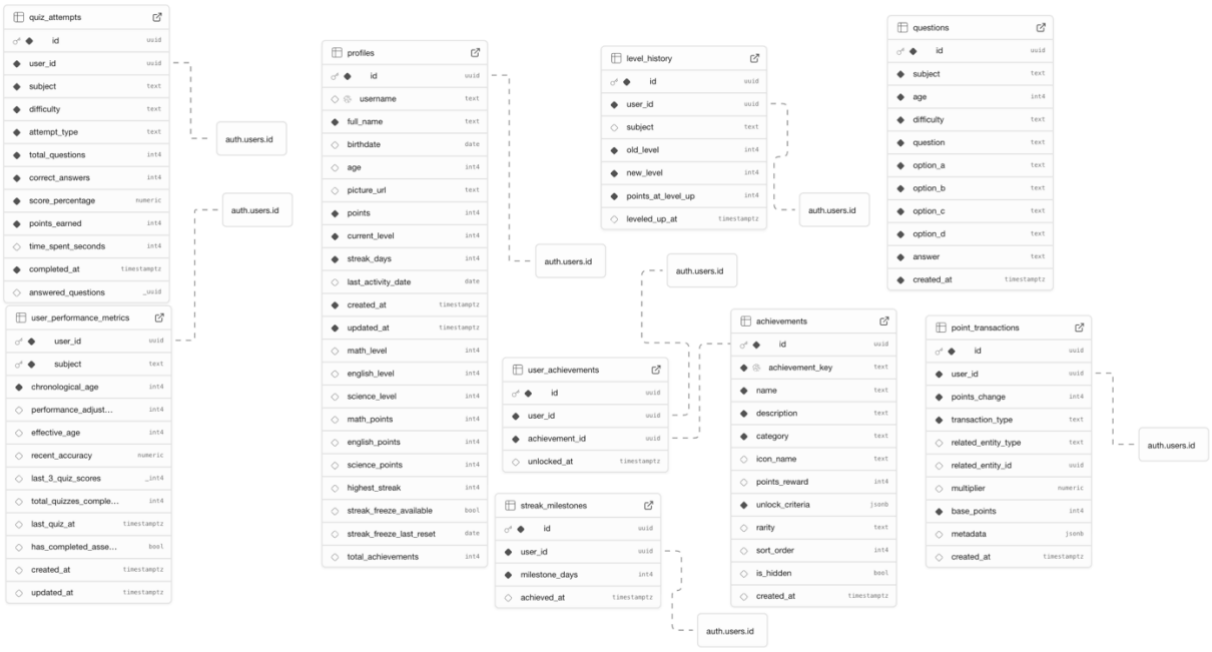


FIGURE 6. Database ER diagram

3.3.1 Profiles and Identity

The profiles table extended the Supabase authentication identity with learning-relevant state. It stored a snapshot of global and subject-specific progress that the dashboard could read in a single query, rather than recalculating totals for every request. Streak state and achievements counts were also stored here so that the streak update function and the gamification engine could work against a single, consistent record per learner.

TABLE 3. Key fields in profiles

Column	Role in the System
id	The primary key mirrors auth. users.id
`full_name, username`	Display information for the dashboard
`birthday, age`	Basis for chronological age and EDL effective-age calculations
`points, current_level`	Overall XP and global level are shown in the profile summary
`math_points, english_points, science_points`	Subject-specific XP pools
`math_level, english_level, science_level`	Subject-specific levels for progression
`streak_days`	Current daily activity streak
`highest_streak`	Longest streak achieved, used for feedback and analytics
`streak_freeze_available, streak_freeze_last_reset`	State used by the streak freeze logic
`total_achievements`	Cached count of unlocked achievements
`last_activity_data`	Last day with recorded learning activity

This table allowed the UI to render the profile, streak, and level panels without scanning transaction tables, while the underlying logs still preserved a complete audit trail.

3.3.2 Question Bank and Quiz Attempts

Learning content and attempts were stored in two core tables: `questions` and `quiz_attempts`. The questions table contained the multiple-choice items, along with subject, difficulty, and target age metadata. `quiz_attempts` recorded each assessment or practice session as a single row, including the score, time spent, and the list of question identifiers used in that quiz. By keeping the answers in a

JSON-like array, the platform could reconstruct a quiz if needed while still treating each attempt as an atomic event for EDL and gamification updates.

TABLE 4. Key fields in the question table

Column	Role in the system
subject	Categorizes items into math, English, and science
age	Target age used when matching effective age
difficulty	Difficulty tier (easy/medium/hard) for distribution rules
question, option_a – option_d, answer	Full definition of each MCQ item

TABLE 5. Key fields in the quiz_attempts table

Column	Role in the system
user_id	Links attempt to a learner.
subject, difficulty	Context of the quiz for later analysis
attempt_type	Distinguishes assessment from adaptive practice
total_questions, correct_answers, score_percentage	Aggregated performance per quiz
points_earned	XP awarded for the quiz (before transaction breakdown)
time_spent_seconds	Duration, used for later exploratory analysis
answer_questions	An array of questions.id used for repetition control
completed_at	Timestamp used in EDL recency logic and 30-day exclusion windows

The adaptive endpoints read `quiz_attempts` to calculate recent accuracy and to exclude items answered in the previous 30 days.

3.3.3 Adaptive Performance Metrics (`user_performance_metrics`)

The Adaptive state was summarized per user and subject in the `user_performance_metrics` table. Instead of recalculating performance from raw quiz rows on every request, the system maintained a compact record of recent accuracy, effective age, and quiz counts. This table was updated transactionally whenever a quiz was submitted via the `/api/quiz-attempts` route.

TABLE 6. Key fields in the user_performance_metrics Table

Column	Role in the system
`user_id, subject`	Composite key, one record per subject per learner
`chronological_age`	Age derived from profile, used as a baseline for adaptation.
`performance_adjustment`	An integer in the range -2…+2 representing the performance-based shift
`effective_age`	Adjusted age used for selecting age-appropriate questions
`last_3_quiz_scores`	Rolling window of the most recent quiz percentages
`recent_accuracy`	Mean of the last scores, feeding the EDL status and distribution logic.
`total_quizzes_completed`	Engagement indicator used in dashboards and some recommendations
`last_quiz_at`	Recency marker for adaptation and for progress summaries
`has_completed_assessment`	The gate that unlocked adaptive practice once the baseline assessment was done

created_at, updated_at	Timestamps for auditing and debugging adaptive updates

This table acted as the single source of truth for the EDL status API, which exposed the current effective age and flow category to the frontend.

3.3.4 Gamification State and History

Gamification relied on a set of tables that separated static definitions from user-specific state and `history.achievements` defined each badge and its unlocked `criteria.user_achievements` recorded – which achievements each learner had unlocked. `point_transactions` logged every XP change as an immutable event. `streak_milestones` stored only the major streak thresholds reached, while `level_history` tracked each level-up across subjects. Together, these tables allowed the system to show a simple summary on the dashboard, preserving a complete audit trail for analysis.

TABLE 7. Key gamification tables

Table	Column	Role in the system
achievements	achievement_key, name	Stable identifiers and labels for badges
	category, rarity	Used to group achievements in the UI (persistence, mastery, etc.)
	points_reward	XP awarded on unlock
	unlock_creteria (jsonb)	Encodes rule conditions for achievement
user_achievements	user_id, achievement_id	Records that a badge a learner has unlocked
	unlocked_at	Timestamp for progression analysis

point_transactions	point_change, base_points, multiplier	Breakdown of XP awards per event
	transaction_type	Classifies the source (quiz, streak bonus, level up, achievement)
	related_entity_type, related_entity_id	Optional linkage to quizzes or achievements
	metadata	Stores context such as subject or difficulty
streak_milestones	milestone_days	Streak lengths that triggered additional rewards or badges
	achieved_at	Used to reconstruct streak progression over time
level_history	subject, old_level, new_level	Level-up events per subject
	points_at_level_up	Points total now of level-up
	levelled_up_at	Timestamp for progression timelines

The profiles table showed only the current totals (points, level, streak), while these gamification tables captured how those states had been reached.

3.3.5 Constraints, Indexing, and Security Alignment

The schema was aligned with the Supabase Row-Level Security model. All user-scoped tables (`profiles`, `quiz_attempts`, `user_performance_metrics`, `point_transactions`, `user_achievements`, `streak_milestones`, `level_history`) enforced RLS policies that restricted access to rows where `user_id = auth.uid()` (or `id = auth.uid()` in the case of profiles). Public tables, such as questions and achievements, allowed read-only access, while inserts and updates were limited to trusted API routes.

Indexes were added on the primary foreign keys and frequently filtered columns (for example `quiz_attempts.user_id`, `quiz_attempts.completed_at`, `user_performance_metrics.userid`, `subject`, and `point_transactions.user_id, created_at`). This kept adaptive queries and dashboard endpoints responsive even as the logs grew.

TABLE 8. Design decisions supporting performance and security

Aspect	Example decision	Rationale
RLS policies	USING (`auth.uid() = user_id`) on user-scoped tables	Guarantees per-user isolation at the database layer
Constraints	Non-negative checks on points and levels	Prevents invalid gamification state
Indexing	Indexes on (`user_id, subject`) in `user_performance_metrics`	Speeds up EDL status and adaptive question selection queries
Public vs private	Questions and achievements are readable by all, and others are private	Allows shared content while protecting personal performance

Together, these design choices ensured that the data model remained consistent with the platform's educational logic, while also being efficient to query and safe to expose through the API layer.

3.4 Adaptive Learning Logic

The platform's adaptive behavior was implemented through the Education Difficulty level system. EDL translated recent quiz performance and age information into a single control variable, effective age, which was then used to select question difficulty. The aim was to keep learners in an “optimal challenge” region with a fully transparent, rule-based mechanism rather than a black-box model.

The implementation was spread across:

- The data layer (`user_performance_metrics, quiz_attempts, questions`),
- TypeScript helpers in `/lib/edl/calculator.ts` and `/lib/edl/selector.ts`, and
- The API routes `/app/api/quiz-attempts/route.ts`, `/app/api/questions/route.ts`, and `/app/api/edl/status/route.ts`

All adaptive decisions could be reconstructed from these components.

3.4.1 Assessment and Exercise Structure

Adaptation always began from a baseline assessment. The exercise system was deliberately simple: three subjects (mathematics, English, Science), three difficulty tiers (easy, medium, hard), and four-option multiple-choice items. Question metadata (subject, difficulty, age target) was stored in the questions table and reused by both assessment and practice modes.

The system used two quiz types, implemented in the same route (`/app/api/quiz-attempts/route.ts`) but with different configurations.

TABLE 9. Quiz types and their role in adaptation

Quiz Type	Configuration in the system	Role in EDL
Assessment	21 items total: 7 per subject (2 easy, 3 medium, 2 hard), no hints, no time limit	Establishes the first subject-wise accuracy and creates three `user_performance_metrics` rows (one per subject).
Practice	10 items per session, difficulty chosen by EDL, hints and explanations available.	Updates recent accuracy, performance adjustment, and effective age after every submission.

The assessment was required exactly once per user. After the assessment result had been scored and stored in `quiz_attempts`, the server seeded

`user_performance_metrics` for mathematics, English, and science, and flagged that adaptive practice was now available.

3.4.2 EDL Metrics and Core Algorithm

The central control parameter of the system was effective age, defined as the learner's chronological age plus a slight performance-based offset. Chronological age was stored in `profiles.age` and copied into `user_performance_metrics.chronological_age`. The offset, `performance_adjustment`, was constrained to the range -2…+2.

After each completed quiz, `/app/api/quiz-attempts/route.ts` called the calculator in `/lib/edl/calculator.ts` to update the EDL metrics for the relevant subject. The logic was based on recent accuracy for that specific subject and implemented as a simple banded rule:

TABLE 10. Performance bands and EDL adjustment logic

Recent accuracy for the subject	**EDL status**	**performance_adjustment**
≥ 90%	exceptional	+2
76 – 89%	`approaching_mastery`	+1
60 – 75%	`flow_zone`	0
50 - 59%	challenging	-1
< 50%	struggling	-2

The algorithm itself was encoded as deterministic computations. In simplified form, the core of `/lib/edl/calculator.ts` behaved as follows (pseudo-code equivalent to the actual Typescript – See APPENDIX 1. EDL Algorithm (Calculation Logic)):

The resulting values were written back into the `user_performance_metrics` row for that subject. The endpoint `/api/edl/status` reads these fields to render the EDL dashboard and support AI recommendations; it does not recompute anything.

3.4.3 Adaptive Question Selection

When the learner opened the practice page, the frontend called `/api/questions` with the subject and mode (adaptive, easy, medium, or hard). For adaptive mode, the handler delegated to `/lib/edl/selector.ts`, which implemented a fixed five-step pipeline.

1. **Determine target age**
   If the mode was adaptive, the selector used `effective_age` from `user_performance_metrics`. For non-adaptive modes, it fell back to `profiles.age`.

2. **Compute difficulty distribution**
   The selector mapped `recent_accuracy` to a target mix of easy, medium, and hard items. For high performers, the mix contained more hard questions; for struggling learners, it contained more easy questions.

3. **Filter candidates**
   Using the Supabase client, the selector queried the questions table for the chosen subject and age band around the target age.

4. **Apply a 30-day exclusion**
   It removed any question IDs that appeared in recent `quiz_attempts` for that user, preventing immediate repetition.

5. **Sample and shuffle**
   From the remaining pool, it sampled the required counts per difficulty tier, compensated for buckets with too few items, then shuffled the final list before returning 10 questions to the client.

The thesis's critical point is that every adaptive decision stems from these explicit rules. No random model weights or hidden parameters were involved; a reviewer could trace the exact query and sampling process for any given practice quiz.

3.4.4 Scoring and EDL Update Cycle

On submission, the same `/api/quiz-attempts` route scored the quiz, updated the EDL, and triggered gamification in a single transaction. Correct answers were counted on the server-side:

```
const scorePercentage = (correctCount / totalQuestions) * 100;
```

Perfect scores triggered an additional XP bonus, but the EDL update used only the score percentage, not the awarded points. After writing the `quiz_attempts` row, the route called the calculator shown above to refresh `recent_accuracy`, `performance_adjustment`, `effective_age`, and `edl_status` in `user_performance_metrics`. Only then were points, streaks, levels, and achievements updated via the gamification engine.

This tight loop – practice quiz → scoring → EDL update → next question set – ensured that subsequent practice sessions always reflected the learner's most recent performance.

3.5 Gamification Engine Implementation

The gamification layer was implemented as a first-class engine on top of the adaptive loop, not as decoration. It awarded points, maintained subject-specific and overall levels, tracked daily streaks, and unlocked achievements. All this behavior was encoded in PostgreSQL tables, Supabase functions, Next.js API router under `/app/api/gamification/`, and a set of dashboard components under `components/gamification/` and `app/dashboard/progress`.

The guiding design rule was that every XP change and unlock had to be explainable solely from the database. For that reason, the engine used an immutable transaction log (`point_transactions`) and explicit tables for levels, streak milestones, and achievements. In contrast, the profiles table stored the current snapshot that the UI rendered.

3.5.1 Points and XP Allocation

Whenever a quiz was submitted through `/app/api/quiz-attempts/route.ts`, the scoring step computed the raw score and then called the gamification endpoint `/api/gamification/award-points`. This route encapsulated the complete XP calculation: base points, multipliers, final XP, and logging into `point_transactions`.

Base XP Rules

The base structure followed the configuration and remained constant throughout the evaluation.

TABLE 11. Base XP Structure

Component	Rule
Correct answer	+10 XP
Quiz Completion	+20 XP
Perfect score (100%)	+50 XP
Correct answer after hint	+5 XP reduced

These values aligned with the system's philosophy of reinforcing accuracy and persistence without penalizing mistakes.

Dynamic Multiplier System

The total XP after each quiz was used:

```
total_xp = base_xp × streak_multiplier × level_multiplier
```

The multipliers came from `profiles.streak_days` and the subject-level fields (`math_level, english_level, science_level`). They ensured that habitual engagement and subject mastery increased effective rewards.

Streak Multiplier

The streak multiplier scaled from 1.0x to 1.5x based on consecutive learning days.

TABLE 12. Streak Multiplier Tiers

Streak Days	Multiplier	Example (base 10 XP)
0 – 2	1.0x	10 XP
3 – 6	1.1x	11 XP
7 – 13	1.2x	12 XP
14 – 29	1.3x	13 XP
30 – 59	1.4x	14 XP
60+	1.5x	15 XP

The multiplier was computed using `get_streak_multiplier(streak_days)` in SQL.

Level Multiplier (Per Subject)

A second multiplier adjusted XP based on the learner's subject-specific level.

TABLE 13. Level Multiplier Tiers

Level	Multiplier	Example (10 XP base)
1 – 2	1.0x	10 XP
3 – 4	1.1x	11 XP
5 – 6	1.2x	12 XP
7+	1.3x	13 XP

Transaction Types

All XP changes were written to the immutable ledger table `point_transactions`, with a transaction type field.

TABLE 14. Transaction Type

Type	Meaning
quiz_correct	XP from correct answers
quiz_partial	Reduced XP after hints
quiz_completion	+20 XP bonus
perfect_score	+50 XP for 100% accuracy
streak_bonus	Rewards from streak milestones
level_up	XP from level advancement
achievement_unlock	XP from unlocking badges

Separating transaction types ensured that every XP change remained auditable.

In TypeScript, the API route first computed the base points:

```
const basePoints = correctCount * 10 + 20; // 10 per correct + 20 completion
if (scorePercentage === 100) {
  baseBonus += 50;
}
```

It then queried the current streak and subject level, delegated to database functions `get_streak_multiplier(streak_days)` and `get_level_multiplier(level)`, and combined them:

```
const streakMult = await getStreakMultiplier(userId);
const levelMult = await getLevelMultiplier(subjectLevel);

const finalXp = Math.round((basePoints + baseBonus) * streakMult * levelMult)
```

Finally, it inserted a row into `point_transactions` with fields such as `points_change`, `base_points`, `multiplier`, `transaction_type` (for example, `quiz_correct`, `quiz_completion`, `perfect_score`), and optional JSON metadata containing subject and difficulty.

```
await supabase.from("point_transactions").insert({
  user_id,
  points_change: finalXp,
  transaction_type: "quiz_completion",
  metadata: { subject, difficulty }
});
```

The design rationale for this approach was that any XP balance at a given time could be reconstructed by summing the ledger and the copy stored in profiles. Points existed purely for fast database rendering.

3.5.2 Level Progression

Level Progression was derived directly from accumulated points using a square-root formula that made early levels quick to obtain and later levels increasingly demanding. The same rule applied to each subject's points and to the overall XP total, keeping behavior consistent.

The level was calculated either in SQL or TypeScript using the function:

```
Level = floor(sqrt(points / 100)) + 1
```

Subject-specific points were stored in `profiles.math_points`, `profiles.english_points`, and `profiles.science_points`. On every XP award, the backend:

- Recomputed the subject level using the formula above,
- Compared it to the previous level, and
- If the level had increased, insert a row into `level_history` with `old_level`, `new_level`, `points_at_levelup`, and `bonus_xp_awarded`.

Level-up events also triggered an additional XP reward:

```
Bonus XP = 50 * new_level
```

Which was logged as a separate `point_transactions` entry with the `transaction_type` = 'level_up'.

TABLE 15. XP Thresholds

Level	Total XP Required
1	0
2	100
3	400
4	900
5	1600
6	2500
7	3600
8	4900
9	6400
10	8100

On the frontend, the level-up modal component (LevelUpModal) listened for this information via the response payload from `/api/quiz-attempts`. When `level_up` was true, it displayed the subject name, old and new levels, the XP bonus, and a short celebratory message, but did not interrupt the quiz flow.

3.5.3 Daily Streak Logic and Freeze Protection

Daily engagement was modelled through a streak system that rewarded consistency without harshly punishing occasional lapses. The state of the streak lived primarily in the profiles table:

- `streak_days` – current consecutive days of activity,
- `highest_streak` – personal best,
- `last_activity_date` – last date with at least one completed quiz,
- `streak_freeze_available` and `streak_freeze_last_reset` – flags used to implement weekly freeze protection

Freeze Logic

Learners received one freeze per week, which reset every Monday, automatically protected a streak if exactly one day was missed, and it was tracked in `streak_freeze_available`.

Whenever a quiz was successfully recorded, the client called `/api/gamification/update-streak`. This route invoked the PostgreSQL function `update_user_streak_enhanced(user_id)` which:

- compared today with `last_activity_date`,
- incremented `streak_days` if today contained activity,
- automatically applied one freeze if exactly one day was missed and a freeze was available,
- reset the streak if more than one day had been missed (or no freeze remained),
- updated `highest_streak` when appropriate, and
- inserted milestone rows into `streak_milestones` at 3, 7, 14, 30, 60, and 100 days, along with a matching XP bonus transaction (for example, 30 days → +500 XP).

TABLE 16. Streak Milestone Rewards

Milestone	Bonus XP	Badge Unlocked
3 days	75 XP	3 – Day Scholar
7 days	150 XP	Week Warrior
14 days	300 XP	-
30 days	500 XP	Monthly Master
60 days	750 XP	-
100 days	1500 XP	-

On the dashboard, the StreakTrackerCard reads the current streak and personal best from a consolidated `/api/dashboard/data` endpoint. It showed the streak count, the next milestone, a countdown to midnight, and whether freeze was

available. This kept the implementation consistent with the thesis goal of supporting habit formation while avoiding overly punitive resets.

3.5.4 Achievement System

Achievements formed the fourth pillar of the gamification engine, giving learners medium-term goals beyond raw XP. The system distinguished between global definitions in achievements and user-specific unlocks in `user_achievements`.

The definitions table contained, for each badge:

- a stable id (for example, `week_warrior`),
- human-readable name and description,
- category (persistence, milestone, mastery, or exploration),
- `xp_reward`,
- and a JSON `unlock_criteria` field describing the condition (for example, "`streak_days >= 7`" or "`math_level >= 5`").

TABLE 17. Persistence Badges (Effort-based)

Badges	XP	Requirement
First Steps	50 XP	Complete the first quiz
3 – Day Scholar	75	Maintain 3-day streak
Week Warrior	150	Maintain 7-day streak
Monthly Master	500	Maintain 30-day streak

TABLE 18. Milestone Badges (Achievement-based)

Badge	XP	Requirement
Assessment Master	150	Complete Initial Assessment
Dedicated Learner	100	Complete 10 quizzes
Quiz Champion	300	Complete 50 quizzes

TABLE 19. Mastery Badges (Skill-based)

Badge	XP	Requirement
Math Champion	200	Math Level ≥ 5
Reading Master	200	English Level ≥ 5
Science Genius	200	Science Level ≥ 5
Perfect Score	100	Score 100% on a quiz

TABLE 20. Exploration Badge (Discovery-based)

Badge	XP	Requirement
Well-Rounded	100	Complete Quizzes in all subjects

The route `/api/gamification/check-achievements` was invoked after key events (quiz completion, streak update, level-up). It loaded the global achievements, queried the `user_achievements` table for that user, evaluated which criteria had just become true, and inserted new unlocks with timestamps. XP rewards were applied by calling the same award-points logic with `transaction_type` = '`achievement_unlock`'.

On the client side, `AchievementProvided` maintained a queue of newly unlocked badges and rendered them one by one using `AchievementUnlockModal`. The modal displayed the badge name, a short description, the XP reward, and a lightweight confetti animation. This queueing ensured that multiple achievements unlocked in a single session did not overwhelm the learner.

The design rationale was that achievement rules were data-driven: adding or tuning a badge required only database changes, and all decisions were visible in SQL rather than hidden in UI code.

3.5.5 Integration into the Dashboard and Practice Flow

All gamification signals surfaced through the dashboard and practice interfaces. Rather than scattering logic, the project used a small set of high-level components:

- `ProfileSummaryCard` displayed overall XP, global level, and progress towards the next level
- `SubjectProgressGrid` visualized subject-wise points and levels with progress bars.
- `StreakTrackerCard` showed current streak, personal best, countdown, and freeze status.
- `AchievementShowcase` and `RecentAchievementsCard` presented unlocked and locked badges.
- `AchievementUnlockModal` and `LevelUpModal` handled transient celebrations.

Data for these components came primarily from two sources:

1. `/api/dashboard/data`, which aggregated profiles, recent `point_transactions`, and `user_achievements` into a single payload.
2. The responses from `/api/quiz-attempts`, `/api/gamification/update-streak`, and `/api/gamification/check-achievements`, returned the immediate XP, streak, level, and achievement changes after each quiz.

By keeping the engine on the server and presentation in React components, the system maintained a clean separation. All numeric states could be verified from the database, while the user interface translated that state into intuitive progress visuals.

3.6 AI Feedback Components Implementation

The platform included a dedicated AI feedback layer that enhanced learners' understanding through hints, explanations, recommendations, and reflective insights. AI was intentionally restricted to supportive guidance. It did not influence scoring, correctness, adaptive difficulty (EDL), streak logic, or any gamification mechanics. This separation preserved the determinism and auditability of the core learning engine while demonstrating how modern AI services can elevate the user experience.

All AI functionality was implemented through a set of small, isolated API endpoints under `/app/api/ai/`, with shared infrastructure in `/lib/ai/openai.ts`, `/lib/ai/prompts.ts`, and `/lib/ai/guard.ts`. The design choices emphasized safety, clarity, low cost, and predictable output.

3.6.1 OpenAI Client Configuration

A single configuration file, `/lib/ai/openai.ts`, instantiated the OpenAI client and enforced consistent parameters across all endpoints.

This file is defined:

- The model used (gpt-4o-mini)
- Environment bindings (`OPENAI_API_KEY`, custom token limits)
- Token safety caps (hardCapTokens)
- A trimming helper to prevent oversized prompts
- A central export used by all AI routes

Controlled token usage:

```
import OpenAI from "openai";

const apiKey = process.env.OPENAI_API_KEY!;
export const openai = new OpenAI({ apiKey });

export const defaultModel =
  process.env.OPENAI_MODEL || "gpt-4o-mini"; // sensible default

export function hardCapTokens(requested: number, envCap: string | undefined,
absoluteMax: number) {
  const cap = Number(envCap ?? 0);
  if (Number.isFinite(cap) && cap > 0) return Math.min(requested, cap,
absoluteMax);
  return Math.min(requested, absoluteMax);
}

// Very small utility to keep prompts compact
export function trim(str: string, max = 6000) {
  if (str.length <= max) return str;
  return str.slice(0, max) + "\n\n…(truncated)";
}
```

This wrapper ensured predictable resource usage, protection from runaway token costs, and consistent model behavior across all endpoints. The client remained stateless and was recreated per request, which fit well with the Vercel serverless model.

3.6.2 Prompt Construction and Safety Guardrails

Structured prompts from each task were defined in `/lib/ai/prompts.ts`.

Each function – `hintPrompt` (APPENDIX 2. HintPrompt Function), `explanationPrompt` (APPENDIX 3. ExplanationPrompt Function), `recommendationsPrompt` (APPENDIX 4. RecommendationsPrompt Function), `insightsPrompt` (APPENDIX 5. InsightsPrompt) – generated a JSON-safe instruction block with:

- Subject and difficulty
- Learner age
- Question text
- Answer options
- Learner's incorrect answer (for explanations)
- Tone rules based on growth-mindset principles
- Explicit instructions to avoid giving away answers

All AI endpoints forced the response format to:

```
response_format: { type: "json_object" }
```

This eliminated unpredictable free-text completions and ensured every response parsed cleanly into the expected UI type. The tone guardrails enforced effort-focused language, no intelligence labels, no comparison to other learners, and short, concise guidance.

This produced age-appropriate feedback aligned with the platform's pedagogical goals.

3.6.3 Rate Limiting and Caching Layer

To avoid excessive usage and ensure fairness, all AI requests passed through a custom guard in `/lib/ai/guard.ts`, which implemented:

- per-IP rate limits (20 hints/minute)
- in-memory TTL caching for repeated hint prompts
- simple key by ip + normalized_prompt

Caching was applied only to hints, because hints do not depend on user-specific answers. Explanations, recommendations, and insights were continuously computed fresh, since each depends on personalized context. This design made the AI system performant during evaluations and prevented redundant API calls.

3.6.4 Hint Generation

Hints were provided through `/app/api/ai/hint/route.ts`. The endpoint accepted question text, options, subject, difficulty, learner age, and returned a 1-2 sentence scaffolding hint without revealing the answer.

Example response:

```
{
  "data": {
    "hint": "Try splitting the number into tens and ones before multiplying."
  }
}
```

Hints were integrated directly into the practice interface in `/app/dashboard/practice/page.tsx`, where the "Hint" button triggered the endpoint asynchronously.

The rationale for the AI-driven hint:

- Manually writing hundreds of hints is not feasible
- AI provides scalable, context-aware guidance
- Strict prompts prevent leakage of answers

3.6.5 Explanation Generation

`/app/api/ai/explain/route.ts` generated explanations. This endpoint gets triggered only when a learner submits an answer. It used the original question, the correct answer, the learner's chosen answer, the difficulty level, and the age metadata. The result was a short explanation (max 180 words) which included where the learner's reasoning diverged, the correct reasoning path, and a "quick check" line that reinforces confidence.

Example Output:

```
{
  "data": {
    "explanation": "It looks like you multiplied correctly but missed the
addition step..."
  }
}
```

These explanations respected growth-mindset constraints and never included ability-based praise.

3.6.6 Study Recommendations

High-level personalized recommendation was generated via `/app/api/ai/recommendations/route.ts`. This endpoint analyzed the learner's recent quiz attempts, subject-specific levels, difficulty trends, EDL status (exceptional / flow zone / struggling), and age. The learn page `/app/dashboard/learn/page.tsx` displayed these in a small recommendations panel.

3.6.7 Reflective Insights and Goal Setting

Reflective summaries were generated using `/app/api/ai/insights/route.ts`. The endpoint returned a short 80 – 120-word progress summary, two actionable goals, and a tone adapted to the age group.

Example Response:

```
{
  "summary": "You practiced consistently this week...",
  "goals": ["Try 3 medium math quizzes", "Review vocabulary once more"]
}
```

These insights were shown on the Progress page and complemented the quantitative charts with a brief narrative overview of progress.

3.6.8 Frontend Integration

Each AI feature was integrated in a non-disruptive, UI-consistent way:

TABLE 21. AI Frontend Integration

Page	AI Feature
Practice (`/dashboard/practice`)	Hint button, Explanation button
Learn (`/dashboard/learn`)	Recommendations panel
Progress (`/dashboard/progress`)	Insights section

All requests used asynchronous handlers and presented results through small, lightweight UI components – avoiding interruptions to the learning flow.

3.6.9 Why AI was restricted to Feedback Only

AI was not used for scoring, correctness detection, EDL calculation, streak or achievement logic, and difficulty selection. These functions remained deterministic, SQL-driven, and transparent to educators. Limiting AI served three key goals:

- Transparency – all adaptive decisions were traceable in code and tables

- Fairness – no learner depended on AI uncertainty for correctness
- Safety – prevented harmful feedback loops or biased evaluation

3.7 Data Security and Row-Level Security Enforcement

The platform was designed with a security-first architecture because it processed sensitive learner data, including age, quiz responses, performance history, streak behavior, achievement progress, and AI interaction logs. For this reason, the backend relied entirely on the Supabase authentication system and PostgreSQL Row-Level Security, ensuring that every data read or write occurred under strict per-user access rules. All API routes, database calls, triggers, and session validation routines were configured to prevent cross-user access, avoid client-side data tampering, and ensure full auditability of the learning dataset in the evaluation.

3.7.1 Security Architecture Overview

The overall security model combined three layers of protection:

1. **Authentication via Supabase Auth**
   All users are authenticated using email/password and Google OAuth. Session tokens were stored in HTTP-only cookies and accessed through Supabase client libraries, preventing exposure of private keys or IDs in the browser.

2. **Authorization via RLS policies**
   Every table containing learner-specific data enforced RLS checks based on the logged-in user's UUID. This ensured that no user could ever read or modify another user's performance, streak, EDL metrics, achievements, or XP history.

3. **Server-side execution of all privileged logic**
   Operations such as profile initialization, streak updates, EDL computation, and point awarding executed on the server – either in API routes or SQL triggers – were never delegated to the browser.

This approach guaranteed deterministic enforcement of access rules while keeping the entire adaptive and gamification pipeline compliant with privacy expectations in digital learning environments.

3.7.2 Authentication and Session Handling

Next.js and Supabase are integrated via three Supabase clients: `client.ts`, `server.ts`, and `middleware.ts`. `client.ts` was used in client components and automatically bound to browser session cookies. `server.ts` was used in API routes and server components and implemented the correct async cookie retrieval required by Next.js. `middleware.ts` was used for route guarding and redirect logic.

The server client (`createClient` in `/lib/Supabase/server.ts`) served as the foundation of secure backend operations. All database calls within `/app/api/*` routes invoked this client after verifying that a valid session existed.

```
const supabase = await createClient();
const { data: { user } } = await supabase.auth.getUser();
if (!user) return unauthorizedResponse();
```

Once authenticated, every downstream query automatically inherits the logged-in user's UUID, allowing Supabase to enforce RLS at the database layer without additional logic in the application code.

3.7.3 Row-Level Security Policies Across Tables

All learner-generated data tables used in the evaluation have RLS enabled. Policies followed a uniform pattern: users could only view or modify rows where the `user_id` (or `id` for `profiles`) matched `auth.uid()`.

TABLE 22. RLS Policy mapping across core tables

Table	Field Checked	Allowed actions (per user)
profiles	`id = auth.uid()`	SELECT, UPDATE
quiz_attempts	`user_id = auth.uid()`	SELECT, INSERT
quiz_attempt_answers	`user_id = auth.uid()`	SELECT, INSERT
user_performance_metrics	`user_id = auth.uid()`	SELECT, INSERT, UPDATE
point_transactions	`user_id = auth.uid()`	SELECT
user_achievements	`user_id = auth.uid()`	SELECT, INSERT
streak_milestones	`user_id = auth.uid()`	SELECT, INSERT
level_history	`user_id = auth.uid()`	SELECT, INSERT

The questions and achievement definitions were intentionally left public (read-only) because they contained no user-sensitive content.

Why the pattern mattered: The evaluation required complete isolation of participant data. Since RLS enforced restrictions at the database level, even a misconfigured API route could not accidentally leak another user's performance metrics or adaptive history.

3.7.4 Secure Access Patterns in API Routes

All API endpoints followed a standardized secure pattern: validate the session, request the `user_id` from Supabase, execute RLS-protected queries, and return sanitized JSON responses. If another user attempted to access these records, the query would return an empty array – RLS silently filtered out unauthorized rows. This behavior was essential for routes such as `/api/quiz-attempts`,

`/api/questions`, `/api/edl/status`, `/api/gamification/update-streak`, and `/api/dashboard/data`. All adaptive and gamification behavior reflected only the authenticated learner's data.

3.7.5 Client-Side Privacy and Safety

The frontend deliberately stored no sensitive data in `localStorage`, `sessionStorage`, URL parameters, and client cookies. Only a minimal UI state was maintained client-side. All performance, streak, EDL, and XP data were fetched fresh from server APIs. This design prevented tampering, response replay, unauthorized inspection on shared devices, and the leaking of question items or answers. Even AI-feedback content (hints/explanations) was not cached client-side; only the backend caching layer handled identical repeated hint requests.

3.8 Frontend Implementation and User Interface Logic

The platform's frontend was implemented with Next.js (App Router), using a combination of server components for data-driven views and client components for interactive quizzes, AI, and gamification. The interface was designed to remain lightweight, responsive, and age-appropriate, while also serving as the bridge between learners and the adaptive, gamified backend. All user-facing logic – navigation, quiz interaction, streak updates, level progression, EDL feedback, and AI guidance – was mediated through this layer.

3.8.1 Application Structure and Routing Model

The interface followed the App Router's file-based routing structure. Core learning flows were organized into route groups:

- `auth` for login and registration
- `dashboard` for progress, profile, streak, and achievements
- `assessment` for the initial baseline test
- `practice` for adaptive quizzes

Server Components rendered data-rich pages such as the dashboard and assessment summary, ensuring fast loading and secure access to Supabase-authenticated data. Interactive screens such as quiz answering, streak timers, achievement celebrations, and hint/explanation panels were implemented as client components.

This structure ensured that sensitive operations (fetching EDL metrics, XP totals, streak state, quiz logs) remained server-side. In contrast, interactions requiring immediacy (button clicks, option selection, modal animations) remained client-side.

3.8.2 Interface Design, Styling, and Component Framework

The UI layer was built with Tailwind CSS, Shadcn/ui, and Lucide icons, resulting in a cohesive visual identity that is consistent across the dashboard, quiz screens, and progress views. Tailwind provided a uniform spacing and typography system. Shadcn/ui supplied accessible primitives such as cards, buttons, alerts, progress bars, modals, and collapsible panels.

The components organization reflected functional grouping rather than visual grouping. For example:

- `AssessmentQuestion.tsx` handled question rendering and navigation during the baseline assessment.
- `PracticeQuestion.tsx` coordinated question display, answer submission, and optional hint/explanation requests
- Dashboard components such as `ProfileSummaryCard`, `SubjectProgressGrid`, `StreakTrackerCard`, and `RecentAchievementsCard` aggregated profile, gamification, and EDL data in a clean, readable format
- Gamification modals such as `AchievementUnlockModal` and `LevelUpModal` used controlled animations and temporary stacking queues to avoid visual overload

Most components relied on concise Tailwind classes, for example:

```
<div className="p-6 rounded-3xl bg-cream shadow-soft">
  <h2 className="text-xl font-semibold text-
charcoal">{question.question_text}</h2>
</div>
```

The design maintained large tap targets, high-contrast color schemas, and touch-friendly spacing to support learners across the 7-18 age range.

3.8.3 Dashboard and Gamification Presentation

The dashboard acted as the central hub for learners, integrating adaptive and gamified signals from the backend. The frond did not compute any metrics itself; it rendered what the secure API returned through `/api/dashboard/data`.

The main dashboard view consisted of:

- A summary of total XP and overall level
- Subject-level progress cards showing points and visual progress bars
- The streak tracking panel with countdown-to-midnight and freeze indicators
- A recent achievements panel and achievement gallery
- Adaptive insights or recommendations when available

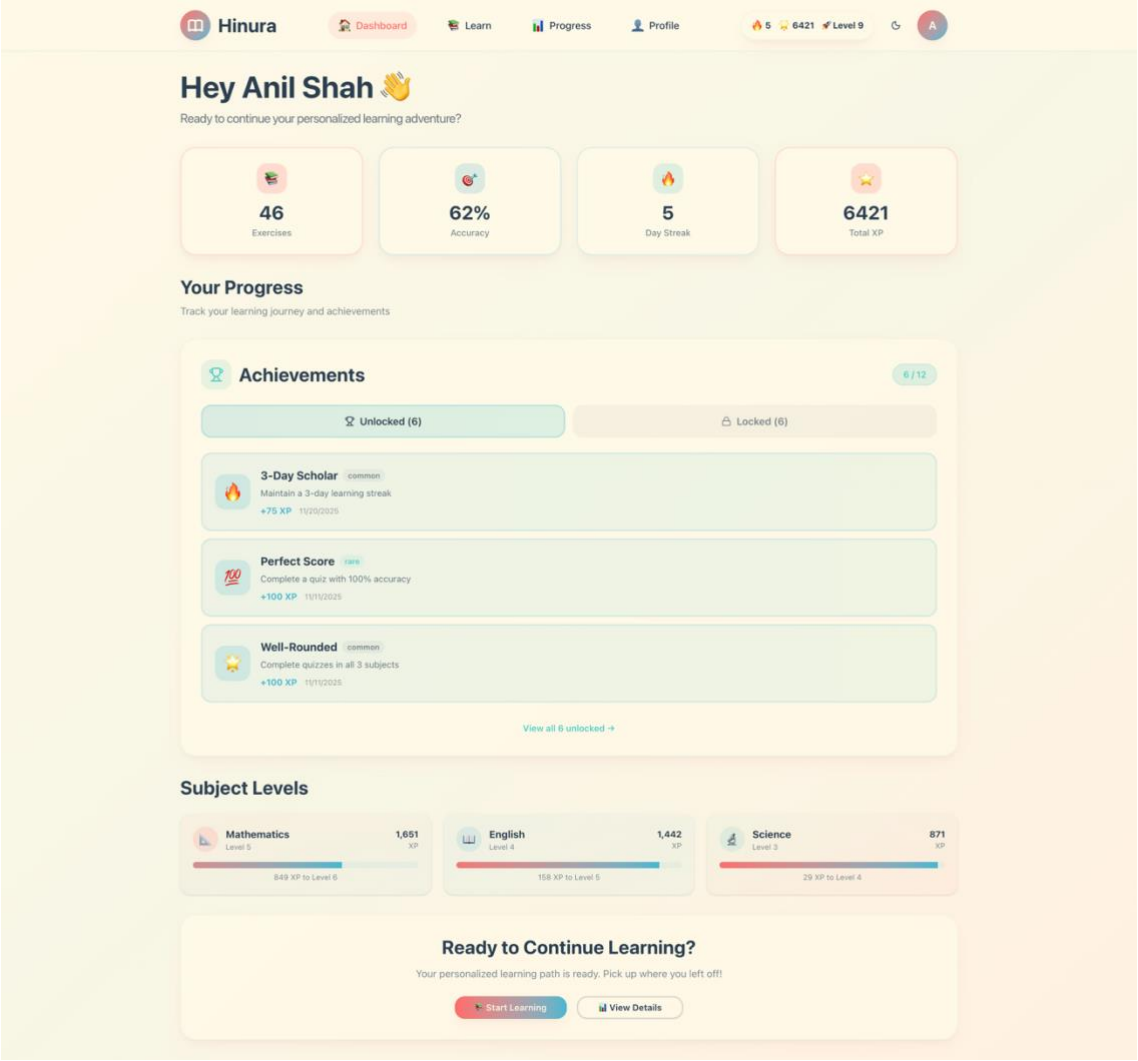


FIGURE 7. Dashboard UI Interface

Achievement and level-up celebrations were implemented through modal components triggered by backend responses during quiz submission. These modals used timed confetti animations, short messages, and automatic dismissal to maintain flow.

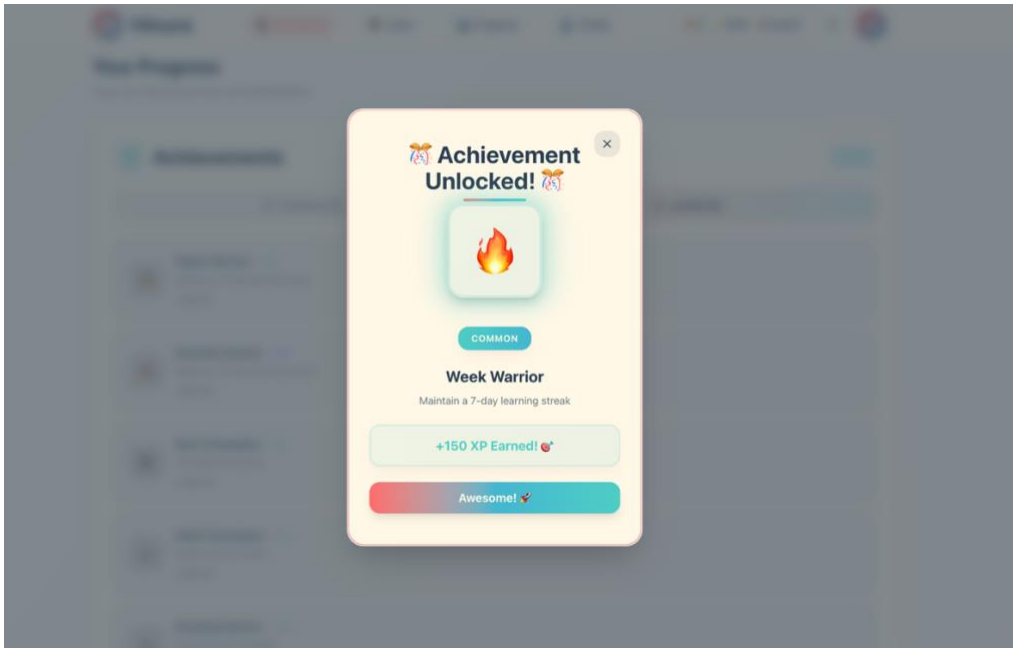


FIGURE 8. Achievement unlocked modal popup animation

This design ensured that the learner always had a clear understanding of their progress while the interface remained connected to the underlying adaptive and gamification systems.

3.9 API Layer and Server-Side Logic

The platform's backend logic was implemented entirely using Next.js Route Handlers in `/app/api/`, enabling the application to maintain a unified full-stack architecture. Because the API layer and the frontend shared the same codebase, all adaptive learning decisions, gamification updates, and AI feedback generation occurred on the server. In contrast, the frontend served as a thin presentation layer.

Every route handler used the Supabase server client defined in `lib/supabase/server.ts`, which passed user sessions via HTTP-only cookies. This ensured that all database operations automatically respected Supabase Row-Level Security without requiring additional authorization logic in the application code. The API layer, therefore, acted as the system's central orchestrator: validating sessions, recording quiz attempts, computing EDL metrics, applying gamification rules, triggering achievements, and involving AI feedback routines.

3.9.1 API Architecture

By keeping the backend within the same project, the system avoided cross-origin issues and eliminated the need for a standalone Node.js server. The directory structure mirrored the functional design:

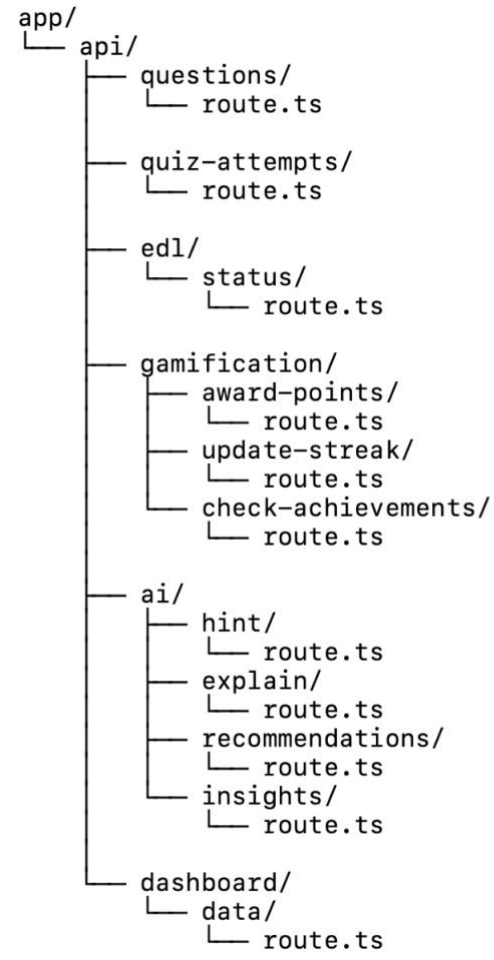


FIGURE 9. API Directory Structure

All handlers began by reconstructing the authenticated user session:

```
const supabase = await createClient();
const { data: { user } } = await supabase.auth.getUser();
if (!user) return unauthorizedResponse();
```

This structure provided a predictable and transparent model for backend execution while keeping the implementation lightweight and maintainable.

3.9.2 Adaptive and Gamified learning logic on the Server

The adaptive engine resided entirely in the server layer. The `/api/questions` endpoint retrieved EDL metrics, computed the effective age, applied the difficulty distribution algorithm, excluded repeat questions, and returned a curated set aligned with the learner's current performance.

Quiz submission via `/api/quiz-attempts` triggered the complete learning logic chain: correctness verification, score calculation, writing quiz logs, updating EDL metrics, awarding XP, updating streaks, and checking achievements. Because all these operations occurred server-side, the platform ensured session consistency and prevented client manipulation.

Gamification behavior was encoded in three endpoints under `/app/api/gamification/`, each responsible for XP updates, streak progression, and achievement unlocking. The server responses contained the updated adaptive and gamification data necessary for the frontend to update progress bars, streak indicators, and celebration modals.

3.9.3 AI Feedback Endpoints

AI-driven features were isolated under `/app/api/ai/`. Each route invoked the OpenAI client through the wrapper defined in `lib/ai/openai.ts`, ensuring token-safe, JSON-formatted responses. Hints, explanations, recommendations, and insights were produced on demand and did not influence correctness or adaptivity. Rate limits were enforced using the custom guard system in `lib/ai/guard.ts`, preventing misuse or excessive API usage.

3.9.4 API Endpoint Summary Table

The following table summarizes all API endpoints used in the system, their HTTP methods, and their functional roles.

TABLE 23. API Endpoints Description

Endpoint	Method	Purpose
`/api/questions`	GET	Returns an adaptive set of practice questions using EDL metrics, difficulty distribution, and exclusion rules.
`/api/quiz-attempts`	POST	Submits a quiz, evaluates correctness, stores, logs, updates EDL, awards XP, updates streaks, and triggers achievement checks.
`/api/edl/status`	GET	Fetches consolidated EDL metrics (effective age, accuracy, adjustment, quizzes completed)
`/api/gamification/award-points`	POST	Computes XP using base rules and multipliers, logs transactions, and returns updated levels
`/api/gamification/update-streak`	POST	Applies streak continuation logic, freeze protection, milestone detection, and XP bonuses.
`/api/gamification/check-achievements`	POST	Evaluates achievement criteria and inserts new unlocked badges into `user_achievements`.
`/api/ai/hint`	POST	Returns a short, non-revealing hint generated from the prompt builder.
`/api/ai/explain`	POST	Produces a concise explanation after incorrect answers
`/api/ai/recommendations`	POST	Generate study advice using recent quiz history and EDL metrics.
`/api/ai/insights`	POST	Produces a reflective summary highlighting progress patterns and improvement goals.
`/api/dashboard/data`	GET	Returns a unified payload combining profile, XP, streak, ED metrics, achievements, and recent activity for dashboard rendering.

4 RESULTS

4.1 Functional Performance

Since the project involved a functional prototype rather than a controlled experiment, the results focus on how the system behaved in real use, how the adaptive and gamified components performed, and how users experienced the platform during informal tryouts with peers, classmates, and friends.

Once deployed on Vercel with Supabase as the backend, the platform functioned as an integrated, stable system. All significant flows – registration, initial assessment, adaptive practice, streak and level updates, and AI feedback – were executed without errors. Database triggers correctly initialized user profiles and adaptive metrics upon account creation, and the dashboard consistently reflected the latest values.

User onboarding, assessment submission, and practice navigation proceeded smoothly across devices. Testers did not report broken screens, loading failures, or data inconsistencies. All question requests, quiz submissions, and gamification updates returned with expected response times (<300ms on average for core API calls). These outcomes confirm that the system's architecture was sufficiently robust for regular use.

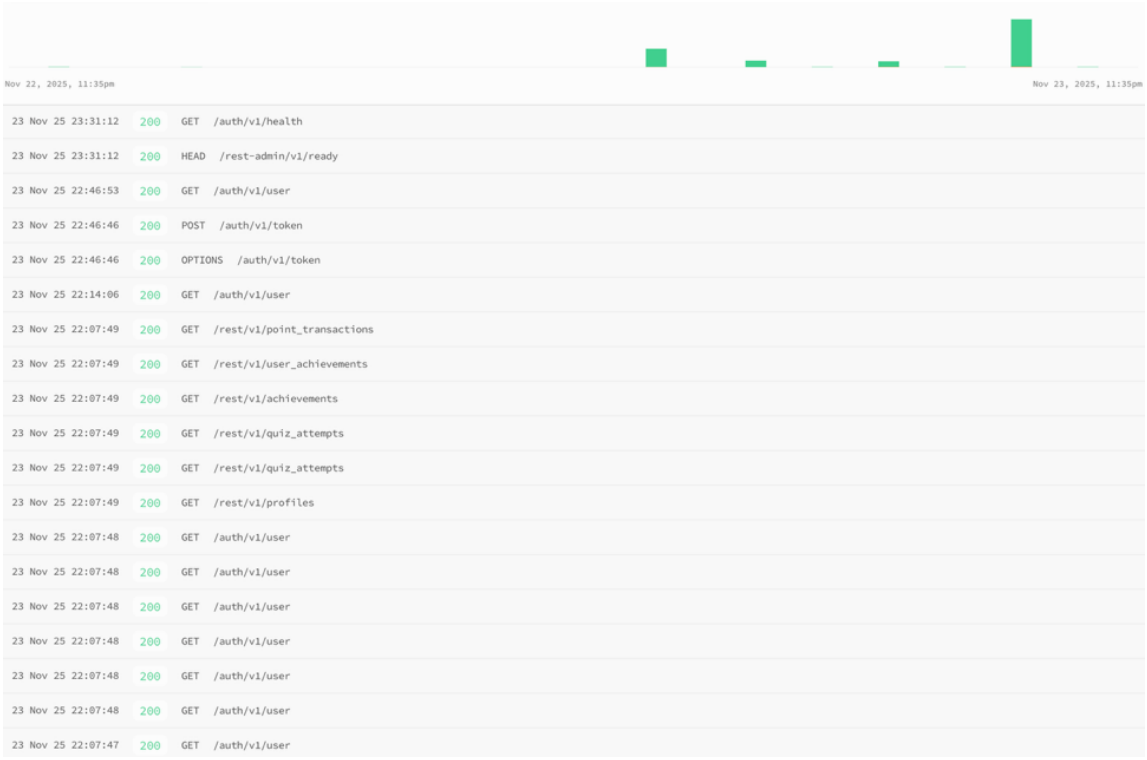
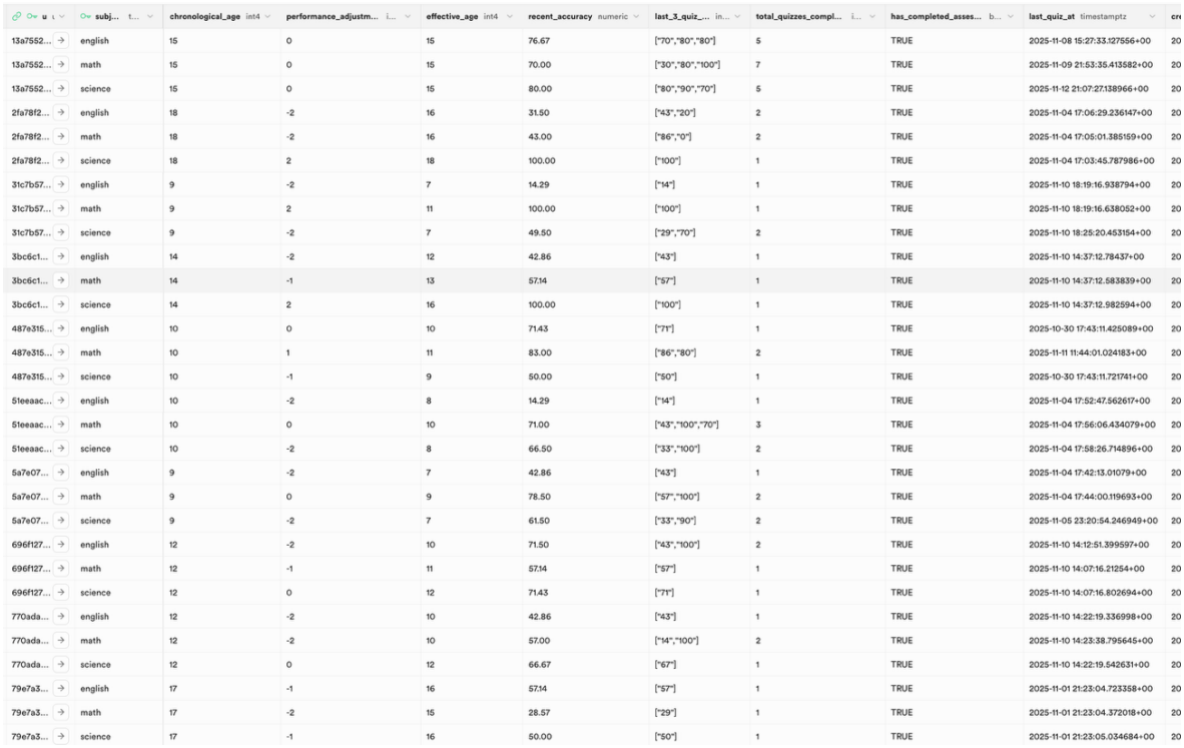


FIGURE 10. API Gateway log

4.2 Behavior of the Adaptive Learning Engine

4.2.1 Correctness of EDL Logic

The adaptive engine operated exactly as specified. For each quiz attempt, the `user_performance_metrics` table recalculated recent accuracy, performance adjustment, effective age, EDL status, and rolling quiz history. Those values were updated immediately after each practice session, indicating that the EDL update functions and selector logic were functioning reliably.



u	subj...	chronological_age int4	performance_adjustm...	effective_age int4	recent_accuracy numeric	last_3_quiz_...	total_quizzes_compl...	has_completed_asses...	last_quiz_at timestamptz	cr
13a7552...	english	15	0	15	76.67	["70","80","80"]	5	TRUE	2025-11-08 15:27:33.127556+00	20
13a7552...	math	15	0	15	70.00	["30","80","100"]	7	TRUE	2025-11-09 21:53:35.413582+00	20
13a7552...	science	15	0	15	80.00	["80","90","70"]	5	TRUE	2025-11-12 21:07:27.138966+00	20
2fa78f2...	english	18	-2	16	31.50	["43","20"]	2	TRUE	2025-11-04 17:06:29.236147+00	20
2fa78f2...	math	18	-2	16	43.00	["86","0"]	2	TRUE	2025-11-04 17:05:01.385159+00	20
2fa78f2...	science	18	2	18	100.00	["100"]	1	TRUE	2025-11-04 17:03:45.787986+00	20
31c7b57...	english	9	-2	7	14.29	["14"]	1	TRUE	2025-11-10 18:19:16.938794+00	20
31c7b57...	math	9	2	11	100.00	["100"]	1	TRUE	2025-11-10 18:19:16.638052+00	20
31c7b57...	science	9	-2	7	49.50	["29","70"]	2	TRUE	2025-11-10 18:25:20.453154+00	20
3bc6c1...	english	14	-2	12	42.86	["43"]	1	TRUE	2025-11-10 14:37:12.78437+00	20
3bc6c1...	math	14	-1	13	57.14	["57"]	1	TRUE	2025-11-10 14:37:12.583839+00	20
3bc6c1...	science	14	2	16	100.00	["100"]	1	TRUE	2025-11-10 14:37:12.982594+00	20
487e315...	english	10	0	10	71.43	["71"]	1	TRUE	2025-10-30 17:43:11.425089+00	20
487e315...	math	10	1	11	83.00	["86","80"]	2	TRUE	2025-11-11 11:44:01.024183+00	20
487e315...	science	10	-1	9	50.00	["50"]	1	TRUE	2025-10-30 17:43:11.721741+00	20
51eeaac...	english	10	-2	8	14.29	["14"]	1	TRUE	2025-11-04 17:52:47.562617+00	20
51eeaac...	math	10	0	10	71.00	["43","100","70"]	3	TRUE	2025-11-04 17:56:06.434079+00	20
51eeaac...	science	10	-2	8	66.50	["33","100"]	2	TRUE	2025-11-04 17:58:26.714896+00	20
5a7e07...	english	9	-2	7	42.86	["43"]	1	TRUE	2025-11-04 17:42:13.01079+00	20
5a7e07...	math	9	0	9	78.50	["57","100"]	2	TRUE	2025-11-04 17:44:00.119693+00	20
5a7e07...	science	9	-2	7	61.50	["33","90"]	2	TRUE	2025-11-05 23:20:54.246949+00	20
696f127...	english	12	-2	10	71.50	["43","100"]	2	TRUE	2025-11-10 14:12:51.399597+00	20
696f127...	math	12	-1	11	57.14	["57"]	1	TRUE	2025-11-10 14:07:16.21254+00	20
696f127...	science	12	0	12	71.43	["71"]	1	TRUE	2025-11-10 14:07:16.802694+00	20
770ada...	english	12	-2	10	42.86	["43"]	1	TRUE	2025-11-10 14:22:19.336998+00	20
770ada...	math	12	-2	10	57.00	["14","100"]	2	TRUE	2025-11-10 14:23:38.795645+00	20
770ada...	science	12	0	12	66.67	["67"]	1	TRUE	2025-11-10 14:22:19.542631+00	20
79e7a3...	english	17	-1	16	57.14	["57"]	1	TRUE	2025-11-01 21:23:04.723358+00	20
79e7a3...	math	17	-2	15	28.57	["29"]	1	TRUE	2025-11-01 21:23:04.372018+00	20
79e7a3...	science	17	-1	16	50.00	["50"]	1	TRUE	2025-11-01 21:23:05.034684+00	20

FIGURE 11. EDL status update

4.2.2 Real-World Observation from Testers

During the informal testing through a live web app (https://hinura.vercel.app/), several participants explicitly commented that the difficulty of questions “felt like it increased when doing well” and “became easier when struggling”. These remarks confirmed that the adaptive mechanism produced a perceptible change:

- Fast learners noticed a shift toward more medium and complex questions.
- Others reported that the system “gave easier ones” after mistakes, which matched the intended corrective behavior.
- Participants also appreciated the mix of difficulty levels within a single quiz, noting that it “never felt repetitive”.

These observations validate that:

- The difficulty distribution tables were applied correctly,
- The selection pipeline generated varied but coherent sets,

- And the adaptive loop achieved the intended Zone of Proximal Development (ZPD) experience.

4.2.3 Question Variety and Spaced Distribution

Across multiple sessions, no participant encountered repeated items within a short period. It was confirmed that the 30-day exclusion rule worked as expected and that the database query filtered answered items effectively. However, some had reported that after using the platform regularly for 2-3 days, they encountered repeated questions. It was due to the fixed set of questions.

4.3 Behavior of the Gamification System

4.3.1 XP, Levels, and Streaks Working as Intended

The base points engine successfully applied base XP, streak multipliers, level multipliers, and perfect-score bonuses. Inspection of `point_transactions` logs confirmed that each XP had changed.

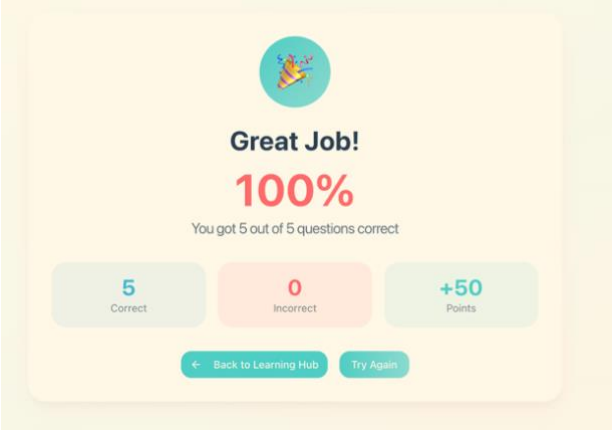


FIGURE 12. Points awarded after completing the quiz

Test users responded positively to this system:

- Several noted that XP awards “made it feel like a game”.
- Many found the streak system motivating and tried to keep their streak going
- The level-up modal was frequently described as “satisfying” or “fun”.

These real-world comments supported the idea that the gamification layer motivated engagement, aligning with the thesis objective.

FIGURE 13. Progress Insight

4.3.2 Achievement Unlocking Reliability

Achievements triggered correctly during testing. For example, “First Steps” is unlocked after completing the first quiz, and “3-Day Scholar” is unlocked automatically when streak logic detects the milestone. The system automatically

awarded mastery badges when the subject level reached 5. Each unlock event was reflected in the UI modal, and all corresponding XP transactions were recorded correctly in `point_transactions` and `user_achievements`. Participants described the achievements as “nice rewards”, “bonus surprises”, or “extra goals”, indicating that they meaningfully contributed to perceived progress.

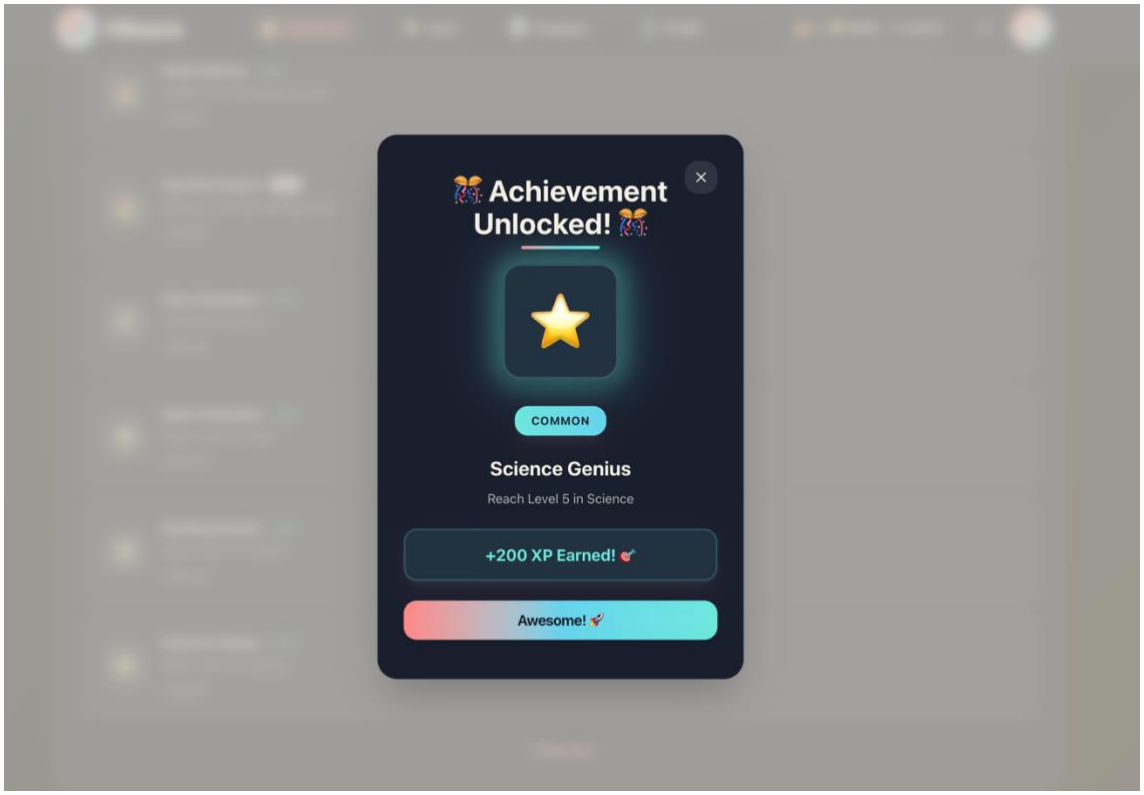


FIGURE 14. Science Genius Achievement unlocked model popup

4.4 Behavior of the AI Feedback Layer

The AI hint and explanation system operated consistently across all tested scenarios:

- Hints were short, guiding, and did not leak answers.
- Explanations triggered only for incorrect answers and remained within the designed word limits.
- Recommendations and progress insights were generated in structured JSON format, allowing error-free UI rendering.

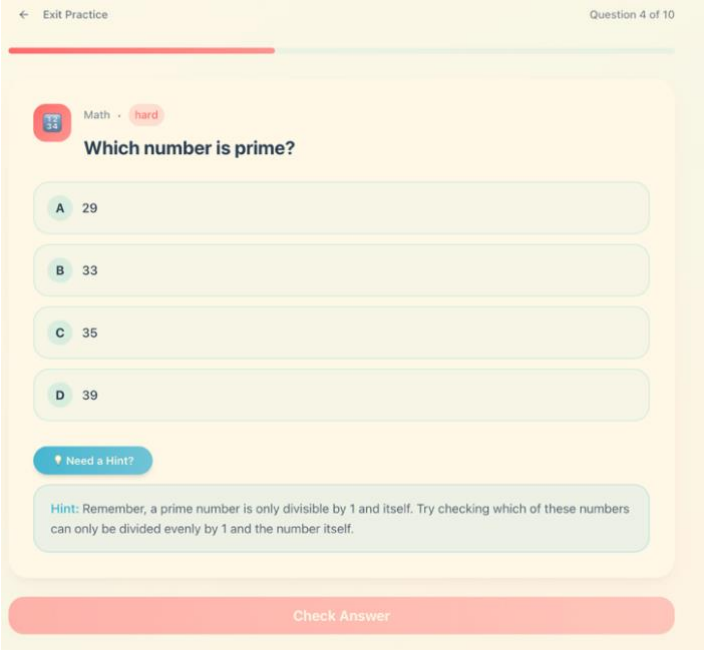


FIGURE 15. Hint offered by AI to solve the question

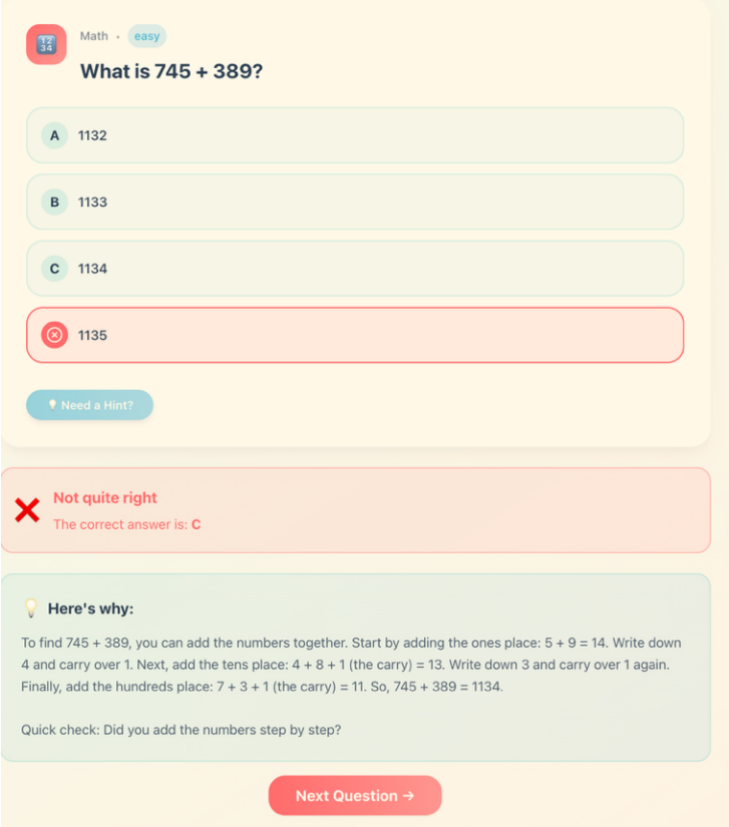


FIGURE 16. Explanation offered by AI when the answer is incorrect

During user testing, participants reported that hints were “helpful but not too revealing”, and some commented that explanations “helped see what mistake was made”. AI insights were perceived as “simple but motivating”, especially when tied to EDL status.

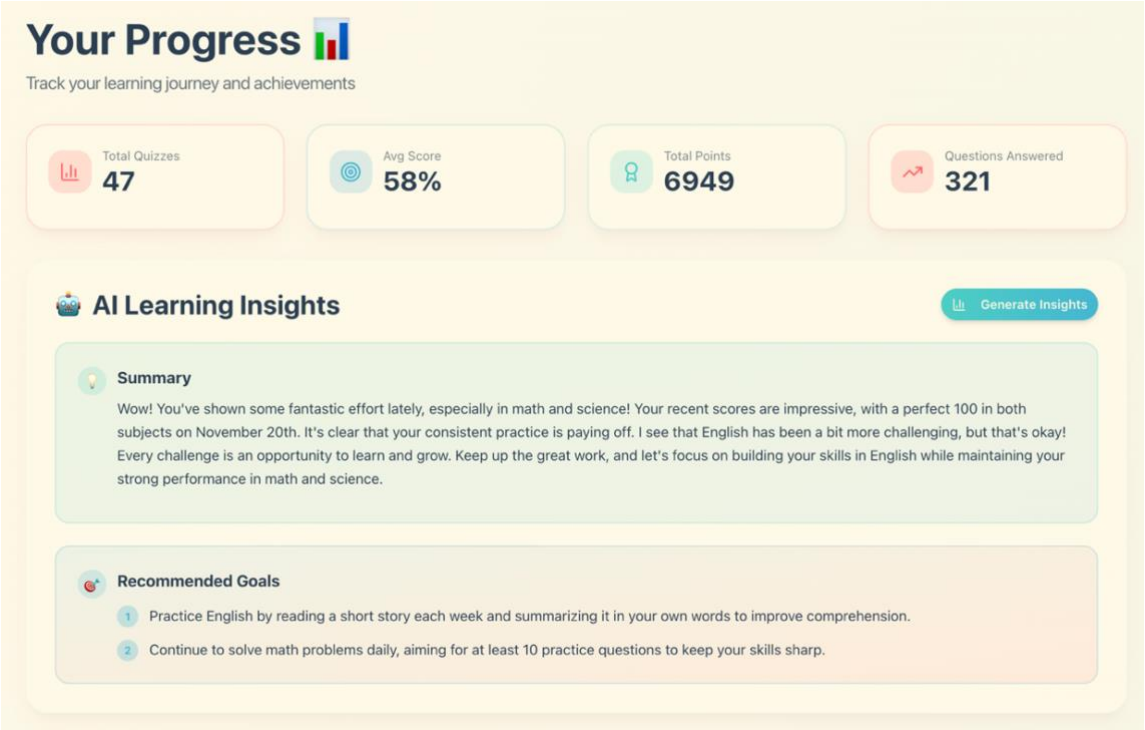


FIGURE 17. AI Learning Insights

Importantly, no tester reported confusion between AI assistance and correctness scoring. Hence, it confirms that the system successfully separated deterministic learning logic from feedback generation.

4.5 Overall Outcomes and User Experience

Across all testing sessions with friends and classmates, the platform demonstrated:

- Stable operation with no major failures
- Correct implementation of adaptive behavior
- Positive motivational impact from points, streaks, levels, and achievements
- Meaningful feedback support from the AI components

- A clear and usable interface that testers associated with modern educational apps

These qualitative observations provided strong evidence that the system behaved as designed. Even without a large, controlled trial, the informal evaluation confirmed that the prototype successfully demonstrated the feasibility of combining adaptive learning, gamification, and AI feedback.

5 DISCUSSION, CONCLUSION, AND FUTURE WORKS

5.1 Discussion

The objective of this thesis was to design and implement a web-based adaptive learning platform that integrates rule-based personalization (EDL), motivational gamification, and AI-supported feedback. The final prototype demonstrated that these three components could operate together in a coherent, explainable, and technically lean architecture suitable for educational applications.

Alignment Between Design and Results

The system behaved consistently with the methodological expectation outlined in Chapter 2. The EDL mechanism correctly adjusted question difficulty based on recent accuracy, combining chronological age with performance-based adjustments to produce an “effective age” which guided question selection. Test users clearly noticed these adaptive shifts, describing the changing question mixture as both challenging and engaging. Their feedback confirmed that even a transparent, rule-based approach can yield meaningful perceptions of personalization, supporting research suggesting that real-time difficulty adjustment enhances engagement and flow.

Gamification components also functioned as intended. The XP system, streak mechanics, level progression, and achievement unlocking worked reliably and provided visible motivation to users. Logs showed correct multiplier application, correct milestone handling, and consistent transaction histories. Participants reported that XP and streaks “made it feel like a game,” and that level-ups and achievements sustained interest. This aligns with self-determination and habit-formation literature: regular feedback, small wins, and visible progress help maintain engagement.

The AI feedback layer acted as a supplementary scaffold rather than a core decision-maker. Hints, explanations, and learning insights were perceived as helpful but not intrusive, and the system architecture ensured that AI did not influence correctness, EDL calculations, or gamification outcomes. This approach upheld transparency and fairness, while still demonstrating how AI can

efficiently generate large volumes of context-aware feedback that would be impractical to author manually.

Technical Strengths and Limitations

The decision to implement the entire stack within a single Next.js project contributed significantly to development speed and maintainability. Serverless router handlers, RLS-protected Supabase queries, and unified data models enabled end-to-end tracing of all logic, critical for a thesis requiring transparency and reproducibility.

However, the prototype does have limitations. The question bank remained modest in size, which constrained the full expressive potential of the EDL difficulty distribution. The evaluation relied on informal testing rather than controlled trials, limiting the strength of quantitative claims about learning outcomes. AI components, although effective in guiding learners, were not rigorously assessed for educational impact. Finally, the system currently supports single-learner flows without collaborative or teacher-facing interfaces.

Despite these limitations, the prototype demonstrated technical feasibility and provided a solid foundation for future experimentation or scaling.

5.2 Conclusion

The study successfully achieved its core aim: demonstrating that adaptive difficulty, gamification, and AI feedback can be integrated into a unified, lightweight, explainable learning platform. The platform's behavior during testing validated the design principles:

- Adaptive logic produced noticeable, meaningful difficulty adjustment
- Gamification systems enhanced motivation and encouraged repeated engagement
- AI-based hints and explanations provided supportive feedback without compromising assessment integrity

Through its rule-based EDL engine, transparent XP system, and RLS-protected backend, the system remained interpretable, auditable, and secure –

characteristics often lacking in modern AI-heavy educational tools. The results highlight that effective personalization does not require complex predictive models; a well-designed deterministic system can already provide substantial pedagogical value while remaining understandable to educators and evaluators.

Overall, the thesis contributes a practical demonstration of how theory-informed adaptive mechanisms, motivational psychology, and AI-assisted feedback can be combined into a coherent learning environment suitable for further research or expansion.

5.3 Future Work

Several enhancements could extend the platform's capabilities and research potential:

Expanded Evaluation and Analytics

A controlled study with larger sample sizes, pre-post comparison, or A/B testing against a non-adaptive baseline would provide more substantial evidence about learning gains and motivational effects. Fine-grained analytics (e.g., Time-on-task, hint usage patterns) could help refine the EDL thresholds.

Larger and Richer Item Bank

Expanding the question dataset, adding multi-step reasoning items, or supporting open-response formats would further strengthen the adaptive engine and reduce repetition.

Teacher and Parent Interfaces

Dashboards for educators or guardians could enable progress monitoring, personalized recommendations, and curriculum planning – extending the system's practical value.

Adaptive Sequencing Beyond Difficulty

Future versions could incorporate domain models or concept maps, enabling not only difficulty adaptation but also content sequencing based on prerequisite knowledge.

More Advanced AI integration

- While AI was intentionally limited in this prototype, future iterations could:
- Generate personalized learning paths,
- Classify common student mistakes,
- Or adjust feedback tone adaptively

Any such integration, however, should maintain transparency and avoid allowing AI to influence correctness or assessment decisions.

Mobile Application or Offline Mode

Given the intended age range, a mobile-friendly or offline-capable version would increase accessibility and expand use cases, especially in low-connectivity environments.

Integration with Online Teaching Platforms

The system could be extended into mainstream online teaching platforms by exposing its adaptive and gamification endpoints to teachers and learning management systems. With its structured EDL metrics and transparent progression logic, the platform can provide real-time insights into learner performance that support differentiated instruction. Teachers could assign adaptive quizzes, review automated progress summaries, and integrate the system into existing tools such as Google Classroom or Moodle. It would allow the prototype to function as an adaptable “learning engine” inside broader educational ecosystems.

REFERENCES

[1] UNESCO, “Distance learning strategies in response to COVID-19 school closures,” *UNESCO COVID-19 Education Response Education Sector issue notes*, no. April, pp. 1–8, 2020, [Online]. Available: https://en.unesco.org/covid19/educationresponse

[2] W. Holmes, M. Bialik, C. Fadel, R. ; • Luckin, M. Griffiths, and L. B. Forcier, “Artificial intelligence in education promises and implications for teaching and learning. Center for Curriculum Redesign,” 2019.

[3] Z. Papamitsiou and A. A. Economides, “Learning analytics and educational data mining in practice: A systemic literature review of empirical evidence,” *Educational Technology and Society*, vol. 17, no. 4, pp. 49–64, 2014.

[4] M. F. Mavilidi and L. Zhong, “Exploring the Development and Research Focus of Cognitive Load Theory, as Described by Its Founders: Interviewing John Sweller, Fred Paas, and Jeroen van Merriënboer,” 2019. doi: 10.1007/s10648-019-09463-7.

[5] R. A. Bjork and E. L. Bjork, “MINI-REVIEW Institute on Teaching and Learning Forgetting as the friend of learning: implications for teaching and self-regulated learning,” *Adv Physiol Educ*, vol. 43, pp. 164–167, 2019, [Online]. Available: http://advan.physiology.org

[6] R. Azevedo and D. Gašević, “Analyzing Multimodal Multichannel Data about Self-Regulated Learning with Advanced Learning Technologies: Issues and Challenges,” 2019. doi: 10.1016/j.chb.2019.03.025.

[7] J. A. C. Hattie and G. M. Donoghue, “Learning strategies: a synthesis and conceptual model,” *NPJ Sci Learn*, vol. 1, no. 1, 2016, doi: 10.1038/npjscilearn.2016.13.

[8] C. P. D. *et al.*, “L. S. Vygotsky: Mind in Society. The Development of Higher Psychological Processes,” *Am J Psychol*, vol. 92, no. 1, 1979, doi: 10.2307/1421493.

[9] P. H. Mirvis and M. Csikszentmihalyi, “Flow: The Psychology of Optimal Experience,” *The Academy of Management Review*, vol. 16, no. 3, 1991, doi: 10.2307/258925.

[10] A. Hadwin and S. Järvelä, “Introduction to a special issue on social aspects of self-regulated learning: Where social and self meet in the strategic regulation of learning,” 2011. doi: 10.1177/0161468111113002O1.

[11] R. Orji, D. Reilly, K. Oyibo, and F. A. Orji, “Deconstructing persuasiveness of strategies in behaviour change systems using the ARCS model of motivation,” *Behaviour and Information Technology*, vol. 38, no. 4, 2019, doi: 10.1080/0144929X.2018.1520302.

[12] S. S. Kim, “Motivators and concerns for real-time online classes: focused on the security and privacy issues,” *Interactive Learning Environments*, vol. 31, no. 4, 2023, doi: 10.1080/10494820.2020.1863232.

[13] J. Hamari, J. Koivisto, and H. Sarsa, “Does gamification work? - A literature review of empirical studies on gamification,” in *Proceedings of the Annual Hawaii International Conference on System Sciences*, 2014. doi: 10.1109/HICSS.2014.377.

[14] M. Sailer and L. Homner, “The Gamification of Learning: a Meta-analysis,” *Educ Psychol Rev*, vol. 32, no. 1, 2020, doi: 10.1007/s10648-019-09498-w.

[15] A. C. T. Klock, A. N. Ogawa, I. Gasparini, and M. S. Pimenta, “Does gamification matter?: A systematic mapping about the evaluation of gamification in educational environments,” in *Proceedings of the ACM Symposium on Applied Computing*, 2018. doi: 10.1145/3167132.3167347.

[16] D. Dicheva, C. Dichev, G. Agre, and G. Angelova, “Gamification in education: A systematic mapping study,” *Educational Technology and Society*, vol. 18, no. 3, 2015.

[17] S. Hallifax, A. Serna, J. C. Marty, and É. Lavoué, “Adaptive Gamification in Education: A Literature Review of Current Trends and Developments,” in

Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 2019. doi: 10.1007/978-3-030-29736-7_22.

[18] H. Alsadoon, “The Impact of Gamification on Student Motivation and Engagement: An Empirical Study,” *Dirasat: Educational Sciences*, vol. 50, no. 2, 2023, doi: 10.35516/edu.v50i2.255.

[19] Ó. Gavín-Chocano, I. García-Martínez, E. Pérez-Navío, and A. Luque de la Rosa, “Learner Engagement, academic motivation and learning strategies of university students,” *Educacion XX1*, vol. 27, no. 1, 2024, doi: 10.5944/educxx1.36951.

[20] O. Zawacki-Richter, V. I. Marín, M. Bond, and F. Gouverneur, “Systematic review of research on artificial intelligence applications in higher education – where are the educators?,” 2019. doi: 10.1186/s41239-019-0171-0.

[21] A. T. Corbett and J. R. Anderson, “Knowledge tracing: Modeling the acquisition of procedural knowledge,” *User Model User-adapt Interact*, vol. 4, no. 4, 1994, doi: 10.1007/BF01099821.

[22] S. E. Embretson and S. P. Reise, *Item response theory for psychologists*. 2013. doi: 10.4324/9781410605269.

[23] C. Piech *et al.*, “Deep knowledge tracing,” in *Advances in Neural Information Processing Systems*, 2015.

[24] R. S. Sutton and A. G. Barto, “Reinforcement Learning: An Introduction,” *IEEE Trans Neural Netw*, vol. 9, no. 5, 2005, doi: 10.1109/tnn.1998.712192.

[25] B. Fahad Mon, A. Wasfi, M. Hayajneh, A. Slim, and N. Abu Ali, “Reinforcement Learning in Education: A Literature Review,” *Informatics*, vol. 10, no. 3, 2023, doi: 10.3390/informatics10030074.

[26] J. Dempere, K. Modugu, A. Hesham, and L. K. Ramasamy, “The impact of ChatGPT on higher education,” 2023. doi: 10.3389/feduc.2023.1206936.

[27] V. Aleven, E. A. McLaughlin, R. A. Glenn, and K. R. Koedinger, “INSTRUCTION BASED ON ADAPTIVE LEARNING TECHNOLOGIES,” in *Handbook of Research on Learning and Instruction, Second edition*, 2016. doi: 10.4324/9781315736419-33.

[28] C. I. Bekker, S. Rothmann, and M. M. Kloppers, “The happy learner: Effects of academic boredom, burnout, and engagement,” *Front Psychol*, vol. 13, 2023, doi: 10.3389/fpsyg.2022.974486.

[29] S. Nakamura, P. Darasawang, and H. Reinders, “The antecedents of boredom in L2 classroom learning,” *System*, vol. 98, 2021, doi: 10.1016/j.system.2021.102469.

[30] J. S. Eccles and A. Wigfield, “Motivational beliefs, values, and goals,” *Annu Rev Psychol*, vol. 53, 2002, doi: 10.1146/annurev.psych.53.100901.135153.

[31] M. H. Immordino-Yang, *Emotions, learning, and the brain: Exploring the educational implications of affective neuroscience.* 2016.

[32] K. Seaborn and D. I. Fels, “Gamification in theory and action: A survey,” *International Journal of Human Computer Studies*, vol. 74, 2015, doi: 10.1016/j.ijhcs.2014.09.006.

[33] “Mindset: the new psychology of success,” *Choice Reviews Online*, vol. 44, no. 04, 2006, doi: 10.5860/choice.44-2397.

[34] G. Siemens, D. Gasevic, and S. Dawson, “Preparing for the Digital University. a review of the history and current state of distance, blended, and online learning,” 2015.

[35] K. vanLehn, “The relative effectiveness of human tutoring, intelligent tutoring systems, and other tutoring systems,” 2011. doi: 10.1080/00461520.2011.611369.

[36] “What Is Three-Tier Architecture? | IBM.” Accessed: Oct. 15, 2025. [Online]. Available: https://www.ibm.com/think/topics/three-tier-architecture

[37] “Monolithic Architecture - System Design - GeeksforGeeks.” Accessed: Oct. 16, 2025. [Online]. Available: https://www.geeksforgeeks.org/system-design/monolithic-architecture-system-design/

APPENDICES

APPENDIX 1. EDL Algorithm (Calculation Logic)

The core calculation logic used to update a learner's Educational Difficulty Level (EDL) metrics after each quiz attempt. The algorithm computes a rolling accuracy score, maps the learner's performance to predefined EDL status categories, and applies a bounded performance adjustment. This adjustment modifies the learner's effective age level within the adaptive system, ensuring that future questions are neither too easy nor too difficult. By constraining changes within a controlled range, the algorithm maintains stable progression while responding to demonstrated improvements or challenges in performance.

```
function updateEdlMetrics(previousMetrics, newScorePercentage,
chronologicalAge) {
  // 1. Update recent accuracy (rolling mean)
  const recentAccuracy =
computeRecentAccuracy(previousMetrics.recentAccuracy, newScorePercentage);

  // 2. Map recent accuracy to EDL status and performance adjustment
  let adjustment: number;
  let status: EdlStatus;

  if (recentAccuracy >= 90) {
    status = "exceptional";
    adjustment = 2;
  } else if (recentAccuracy >= 76) {
    status = "approaching_mastery";
    adjustment = 1;
  } else if (recentAccuracy >= 60) {
    status = "flow_zone";
    adjustment = 0;
  } else if (recentAccuracy >= 50) {
    status = "challenging";
    adjustment = -1;
  } else {
    status = "struggling";
    adjustment = -2;
  }

  // 3. Clamp adjustment and compute effective age
  const clampedAdjustment = clamp(adjustment, -2, 2);
  let effectiveAge = chronologicalAge + clampedAdjustment;
```

```
  effectiveAge = clamp(effectiveAge, 7, 18);

  // 4. Return updated metrics
  return {
    recentAccuracy,
    edlStatus: status,
    performanceAdjustment: clampedAdjustment,
    effectiveAge,
    quizzesCompleted: previousMetrics.quizzesCompleted + 1,
  };
}
```

APPENDIX 2. HintPrompt Function

The HintPrompt function used to generate context-aware hints within the system's AI feedback loop. It ensures that the language model produces a single, concise hint tailored to the learner's profile and task difficulty, without revealing the correct answer.

```
export function hintPrompt(params: {
  subject: string; age: number | null; difficulty: string;
  question: string; options: string[]; correctAnswer?: string | null;
  attempts?: number;
}) {
  const { subject, age, difficulty, question, options } = params;
  return `
System:
You are a concise tutoring assistant for students aged 7–18.
Return ONE short hint (1–2 sentences). Never reveal the answer.
Avoid jargon; be kind and specific.
If math, suggest a first step or a simpler subproblem.
If reading, prompt them to find context clues.
If science, guide to principle or definition needed.

User:
Subject: ${subject}
Student age: ${age ?? "unknown"}
Difficulty: ${difficulty}
Question: ${question}
Options: ${options.map((o,i)=>`${String.fromCharCode(65+i)}. ${o}`).join(" |
")}

Return JSON:
{"hint":"<one short hint>"}
`;
}
```

APPENDIX 3. ExplanationPrompt Function

The ExplanationPrompt function used to produce brief, comprehensible explanations tailored to the learner's age and question. It prompts the language model to outline the key reasoning steps behind the correct answer and end with a one-line self-check, supporting reflective learning and conceptual reinforcement.

```
export function explanationPrompt(params: {
  subject: string; age: number | null; question: string;
  options: string[]; correctAnswer: string; userAnswer?: string | null;
}) {
  const { subject, age, question, options, correctAnswer, userAnswer } =
params;
  return `
System:
Explain the answer clearly to a ${age ?? "teen"} year-old.
Be brief (<= 120 words). Use 1–2 steps max.
If math, show the key step. If reading, cite the relevant phrase to look for.
End with "Quick check:" and a one-line self-check.

User:
Subject: ${subject}
Question: ${question}
Options: ${options.map((o,i)=>`${String.fromCharCode(65+i)}. ${o}`).join(" |
")}
Correct answer: ${correctAnswer}${userAnswer ? `\nStudent picked:
${userAnswer}` : ""}

Return JSON:
{"explanation":"<short explanation>"}
`;
}
```

APPENDIX 4. RecommendationsPrompt Function

The RecommendationsPrompt function is responsible for generating personalized study recommendations based on the learner's recent performance and skill profile. The function structures a prompt that instructs the language model to return two or three targeted practice suggestions, each paired with a subject, difficulty level, and a brief rationale. By keeping the language motivational and age-appropriate, this component encourages learners to focus on areas requiring improvement while maintaining engagement.

```
export function recommendationsPrompt(params: {
  age: number | null;
  skillLevels: Record<string, number> | null;
  recent: Array<{subject:string; difficulty:string; score:number;
when:string}>;
}) {
  const { age, skillLevels, recent } = params;
  const lines = recent.slice(-8).map(r => `${r.when}:
${r.subject}/${r.difficulty} - ${r.score}%`).join("\n");
  return `
System:
You generate concise study recommendations.
Return 2-3 practice suggestions with subject + difficulty + one-line reason.
Keep language motivational and age-appropriate. Total <= 120 words.

User:
Student age: ${age ?? "unknown"}
Skill levels (1-5, may be missing): ${JSON.stringify(skillLevels ?? {})}
Recent scores:
${lines || "No data"}

Return JSON:
{"recommendations":[
  {"subject":"math","difficulty":"adaptive","reason":"..."},
  {"subject":"english","difficulty":"medium","reason":"..."}
]}
`;
}
```

APPENDIX 5. InsightsPrompt

The InsightsPrompt function, responsible for converting a learner's performance data into tailored feedback and achievable next steps. By incorporating growth-mindset language, age-appropriate tone, and evidence-based insights, the function supports reflective learning and sustained improvement through personalized goal setting.

```
export function insightsPrompt(params: {
  age: number | null;
  aggregates: { avgScore: number; quizzes: number; bySubject: Record<string,
number> };
  trendText: string;
}) {
  const { age, aggregates, bySubject } = { ...params, bySubject:
params.aggregates.bySubject };
  const ageGroup = age && age <= 12 ? "younger" : "older";

  return `
System:
You are an educational AI providing personalized learning insights for a
${age ?? "teen"}-year-old student.

TONE GUIDELINES (Research-Based):
- Use GROWTH MINDSET: Praise effort, strategies, and improvement (NOT
intelligence or being "smart")
- Be encouraging and specific to their actual performance data
- Focus on PROCESS over outcomes (e.g., "Your consistent practice" not
"You're talented")
- ${ageGroup === "younger" ? "Use warm, friendly language with simple words.
Be enthusiastic and supportive." : "Use respectful, mature language. Be
motivating but not condescending."}
- Always acknowledge effort even when performance needs improvement
- Connect feedback to actionable next steps

WHAT TO INCLUDE:
1. Summary (80-120 words): Highlight their effort/consistency, note specific
improvements or patterns, and acknowledge challenges positively
2. Two specific, achievable goals based on their data that focus on
strategies/practice (not just "do better")

WHAT TO AVOID:
- Praising intelligence ("You're so smart!")
- Empty encouragement without substance
- Making them feel bad about low scores
- Generic advice not tied to their data

User:
```

```
Age: ${age ?? "unknown"}
Overall: ${aggregates.avgScore}% across ${aggregates.quizzes} quizzes
By subject avg: ${Object.entries(bySubject).map(([s,v])=>`${s}:
${v}%`).join(", ")}
Trend: ${params.trendText}

Return JSON:
{"summary":"<80-120 words emphasizing effort and
growth>","goals":["<specific, process-focused goal 1>","<specific, process-
focused goal 2>"]}
`;
}
```