



# PRACTICE PAL

By  
**Joel S. Guti**  
H210028R

HIT400 Capstone project Submitted in Partial Fulfillment of the  
Requirements of the degree of  
Bachelor of Technology  
In  
**Software Engineering**  
In the  
**School of Information Sciences and Technology**  
Harare Institute of Technology  
Zimbabwe

Supervisor  
**Wellington Makondo**

June / 2025

## **ABSTRACT**

The increasing demand for personalised and adaptable educational tools has prompted the incorporation of artificial intelligence (AI) into learning platforms. This project introduces Practice Pal, a mobile-based AI-powered application that helps students create personalized practice tests, receive instant feedback, and track their academic progress over time. The system features a Flutter-based frontend and a Spring Boot backend, utilising Ollama to generate relevant and structured content based on user-selected courses and topics. It aims to enhance self-directed learning by offering features such as auto-marking, feedback analysis, performance tracking, and test retakes, all within an intuitive mobile interface.

The research and development process included a feasibility study, detailed system design, implementation, and rigorous testing phases such as unit, module, integration, system, and acceptance testing. Initial testing results showed that users could easily interact with the application and take advantage of its adaptive learning capabilities. However, limitations such as internet reliance, a lack of curriculum alignment, and the absence of educator moderation were identified. Despite these challenges, Practice Pal has proven to be effective in encouraging active learning and self-assessment, especially in resource-constrained environments such as Zimbabwe. The project concludes with recommendations to improve curriculum integration, offline capabilities, and teacher participation in content validation, highlighting the app's potential to support scalable, inclusive, and AI-enhanced education.

## **PREFACE**

This project report documents the design, development, and testing of Practice Pal, an artificial intelligence-powered mobile learning application that seeks to enhance student self-assessment and education accessibility. The motivation behind this project is the growing need for personalized, affordable, flexible, and accessible learning solutions, particularly in environments where traditional education resources are limited, such as in Zimbabwe. The report is a chronological account of the development of the project, starting from the concept and feasibility study through implementation, testing, and end recommendations. Each chapter guides the reader through the technical along with conceptual foundations of the system, the use of artificial intelligence in education, tools and technology, and results of various testing methods. It also incorporates user feedback and makes a comparison between the performance of the platform and existing solutions.

This project not only deepened my understanding of mobile application development, back-end integration, and AI services but also highlighted the societal impact technology can have on transforming student learning. The process involved collaboration, research, critical thinking, and lifelong learning, skills that will continue to be valuable in future professional and academic endeavours.

I hope this report serves as a valuable reference for those interested in educational technology, AI applications, and innovations in student-centred learning.

## **ACKNOWLEDGEMENT**

This project report documents the design, development, and evaluation of Practice Pal, an AI-powered mobile learning application designed to enhance student self-assessment and educational accessibility. The motivation for this project stems from the growing need for personalised, affordable, flexible, and accessible educational tools, particularly in environments where traditional educational resources are limited, such as in Zimbabwe.

The report provides a step-by-step account of the project's journey, from the initial concept and feasibility study to implementation, testing, and final recommendations. Each chapter is designed to guide the reader through the technical and conceptual foundations of the system, including the role of artificial intelligence in education, the tools and technologies used, and the results of various testing strategies. It also incorporates user feedback and evaluates the platform's performance in comparison to existing solutions.

This project not only deepened my understanding of mobile application development, back-end integration, and AI services but also highlighted the societal impact technology can have on transforming student learning. The process involved collaboration, research, critical thinking, and lifelong learning, skills that will continue to be valuable in future professional and academic endeavours.

I hope this report serves as a valuable reference for those interested in educational technology, AI applications, and innovations in student-centred learning.



This is to certify that HIT 400 Project entitled “**Practice Pal**” has been completed by **Joel S. Guti** (H210028R) for partial fulfillment of the requirements for the award of **Bachelor of Technology** degree in **Software Engineering**. This work is carried out by him under my supervision and has not been submitted earlier for the award of any other degree or diploma in any university to the best of my knowledge.

**Your Supervisor Name**

**Approved/Not Approved**

Project Supervisor

Project Coordinator

**Signature:** .....

**Signature:** .....

**Date:**.....

**Date:** .....



### **Certificate of Declaration**

This is to certify that work entitled “Practice Pal“ is *submitted in partial fulfillment of the requirements for the award of Bachelor of Technology (Hons) in Software Engineering, Harare Institute of Technology. It is further certified that no part of research has been submitted to any university for the award of any other degree.*

(Supervisor)	Signature.....	Date.....
(Mentor)	Signature.....	Date.....
(Chairman)	Signature.....	Date.....

## Table of Contents

ABSTRACT.....	2
PREFACE.....	3
ACKNOWLEDGEMENT.....	4
Chapter One: Introduction.....	10
Background.....	10
Problem Statement.....	10
Objectives.....	11
Hypothesis.....	11
Justification.....	12
Proposed Tools.....	12
1. Frontend (Mobile App).....	12
2. Backend (Server-side Logic).....	12
3. AI Integration.....	13
4. Hosting & Deployment.....	13
5. Development & Testing Tools.....	13
Feasibility Study.....	13
Technical Feasibility.....	13
Economic Feasibility.....	13
Operational Feasibility.....	14
Legal and Ethical Feasibility.....	14
Time Feasibility.....	15
Project Plan.....	15
Chapter Two: Literature Review.....	16
Introduction.....	16
Related Work.....	16
Quizlet.....	16
Khan Academy.....	17
Duolingo.....	17
Conclusion.....	18
Chapter Three: Analysis.....	19
Information Gathering Tools.....	19
Interview Questions.....	19
Questionnaires.....	20
Document Review.....	21
Description of System.....	21
Evaluation of Alternatives Systems.....	22
Existing System (Quizlet).....	23
Functional Analysis of Proposed.....	24
System Requirements.....	26
Functional Requirements.....	26
Non-Functional Requirements.....	26
Use-Case Diagram for Proposed System.....	27
Chapter Four: Design.....	28
System Diagrams:.....	28
Context Diagram.....	28
Data Flow Diagram (DFD) Level 1.....	28
Activity Diagram.....	29
Database Design:.....	30

ER Diagrams.....	30
Normalized Database.....	31
Program Design:.....	34
Class Diagram.....	34
Sequence Diagram.....	35
Interface Design:.....	36
Authentication Pages.....	36
Home Page.....	37
Courses Page.....	38
Generate Course Page.....	39
Tests Page.....	40
Generate Test Page.....	41
Test Results Page.....	42
Test Performance Page.....	43
Test Revision Page.....	44
Chapter Five: Implementation & Testing.....	45
Pseudocode for Major Modules for Java SpringBoot Backend.....	45
Generate and Save Course.....	45
Update and Save Course.....	45
Generate and Save Test.....	46
Generate Test from Existing Questions.....	46
Submit and Evaluate Test.....	47
Test Revision.....	48
Pseudocode for Major Modules for Flutter Frontend.....	49
Generate Course.....	49
Generate Test.....	49
Take and Submit Test.....	50
Revise Test.....	50
System Communication – JSON Payloads.....	52
Software Testing:.....	54
Unit Testing.....	54
Module Testing.....	55
Integration Testing.....	56
System Testing.....	57
Database Testing.....	58
Acceptance Testing.....	59
Chapter Six: Conclusions & Recommendations.....	60
Results and Summary.....	60
Recommendations.....	62
Future Works.....	63
Web-Based Version.....	63
Interactive Visual Analytics.....	64
Natural Language Querying.....	64
Collaborative Learning Features.....	64
Gamification Enhancements.....	64
Accessibility Support.....	64
References.....	66



## Table of Figures

Figure 1: Gannt chart.....	15
Figure 2: Questionnaire.....	20
Figure 3: Quizlet use-case diagram.....	23
Figure 4: Context diagram and DFD Level 1 for Quizlet.....	24
Figure 5: Use-case diagram.....	27
Figure 6: Context diagrams.....	28
Figure 7: Data Flow Diagram (DFD) Level 1.....	28
Figure 8: Activity diagram.....	29
Figure 9: ER diagram.....	30
Figure 10: Class diagram.....	34
Figure 11: Sequence diagram.....	35
Figure 12: Sign in page.....	36
Figure 13: Sign up page.....	36
Figure 14: Home page.....	37
Figure 15: Home drawer.....	37
Figure 16: Courses page.....	38
Figure 17: Generate course page.....	39
Figure 18: Tests page.....	40
Figure 19: Generate test page.....	41
Figure 20: Test results page.....	42
Figure 21: Test performance page.....	43
Figure 22: Test revision page.....	44

## Index of Tables

Table 1: Document review.....	21
Table 2: Users table.....	31
Table 3: Courses table.....	31
Table 4: Course topics table.....	31
Table 5: Tests table.....	32
Table 6: Test questions table.....	32
Table 7: Test results table.....	32
Table 8: JSON payloads.....	52
Table 9: Unit testing.....	54
Table 10: Module testing.....	55
Table 11: Integration testing.....	56
Table 12: System testing.....	57
Table 13: Database testing.....	58
Table 14: Acceptance testing.....	59

## **Chapter One: Introduction**

### **Background**

Zimbabwe's education system has long been recognised for its emphasis on literacy and formal academic success. While there is high literacy, many students have no consistent access to quality learning materials, personalised teaching, and practical assessment tools, particularly in poor or rural communities. Traditional methods of teaching and assessment frequently focus on rote learning and standardised testing, which does not address individual students' unique learning needs and rates.

With the increased use of mobile phones and growing digital literacy, there is greater scope for devising technology-driven solutions with the potential to close learning gaps. Zimbabwean primary school, high school, and college students are increasingly interested in platforms that facilitate personalised learning, offer timely feedback, and adapt to their own pace. Current systems are either too broad, lack comprehensive subject coverage, or intelligent feedback mechanisms.

Artificial intelligence holds great potential for improving learning experiences throughout the world. AI-powered platforms are able to develop adaptive content, automate assessments, and provide students with immediate feedback, all of which can be leveraged to improve learning outcomes in Zimbabwe's digitalization era.

### **Problem Statement**

Although Zimbabwe has done much to maintain a robust education sector, students are presented with problems such as class overcrowding, inadequate access to trained instructors, low one-on-one interaction, and outdated or static learning materials. Existing practice tests available tend to be hand-constructed, are not subject-specific to the indigenous curricula, or lack instant feedback and progress tracking mechanisms.

Besides, the majority of the students, particularly those who are distant from the school or college, lack readily available review materials that would allow them to learn from mistakes and refine their knowledge. Therefore, the students end up feeling unprepared for exams and are not confident in self-testing.

There is a discernible need for a smart, accessible, and personal solution by which Zimbabwean learners from primary to university levels can practice, evaluate, and construct their knowledge in an adaptive and engaging manner. The Practice Pal mobile app meets this need by providing AI-

based course authoring, question construction, auto-grading, and dynamic feedback that is tailored according to the learner's subject of interest and academic level.

## **Objectives**

- To generate test questions based on user-specified subjects and topics using AI.
- To automate the marking process and provide instant feedback upon test submission.
- To enable students to track their performance and learning progress over time.
- To allow users to revise and retake previous tests for continuous improvement.

## **Hypothesis**

The primary hypothesis of this project is that integrating AI into a mobile-learning system can be extremely effective in enhancing students' self-assessment abilities as well as their overall performance. Through the application of AI to automatically generate personalized practice tests, give instant feedback, and track user performance, the system addresses fundamental shortcomings of traditional and static approaches to assessment. Tailored tests enable learners to gain access to content aligned with their current skill level and subject of interest, thereby making learning more significant and relevant. Instant feedback completes the cycle between testing and learning, allowing students to reflect on errors and modify understanding in real time. Early intervention can reinforce correct concepts while preventing the formation of chronic misunderstandings. Apart from this, measuring performance over a period allows students to observe how they improve academically, identify weak areas, and make constructive choices to improve. This continuous feedback loop, courtesy of AI, makes learning more of an interactive and individualized process.

The sub-hypotheses defending the aforementioned more specifically describe the processes through which AI contributes to improved learning results. To begin with,  $H_1$  stipulates that practice tests created by the computer, dynamically generated according to a learner's chosen subject and level of difficulty, perform better than static, generic ones because they better fit the learner's immediate needs and circumstances. Personalization maximizes motivation and subjects learners to the appropriate amount of challenge. Second,  $H_2$  posits that the ability of the platform to automatically grade and provide instant, constructive feedback makes it possible for gaps in knowledge to be identified at a quicker pace, reinforcing retention and accelerating learning.  $H_3$  emphasizes reflection by giving students a chance to review their initial efforts and retake tests. Through this repetitive pattern, understanding is made stronger and mastery is achieved over time. Lastly,  $H_4$  asserts that a mobile platform enhances accessibility and student engagement, especially where

there is poor access to textbooks, internet-connected computers, or physical classrooms. As Zimbabwe, and similar countries, have good mobile phone penetration, such a platform can potentially empower students from diverse backgrounds to take ownership of learning at any place, anytime. Together, these hypotheses confirm the contention that AI-driven mobile learning platforms are an evolutionary tool for personalized, adaptive, and accessible learning.

## **Justification**

In Zimbabwe, there are different students who experience educational challenges due to the unavailability of customized learning tools, particularly in rural and underprivileged schools. Traditional ways of instruction and paper tests systematically fall short of providing instant feedback or addressing individualized student needs. This results in knowledge gaps and constrains learning development, particularly for those children who are unable to keep up with full classrooms and poor teacher support.

With the rising pervasiveness of smartphones and mobile connectivity throughout the nation, there is the potential to harness technology to bridge these gaps. Practice Pal addresses this gap by offering an AI-based mobile solution that creates customized courses and quizzes for students of any age group. The solution provides instant feedback, performance tracking, and revision facilities unavailable in most learning settings.

Through combining accessibility, automation, and intelligent content generation, Practice Pal makes it possible for independent learning, allowing all learners to practice and build at their own pace. This not only enhances educational achievement but also builds confidence, digital skills, and life-long learning capabilities in Zimbabwean learners.

## **Proposed Tools**

### **1. Frontend (Mobile App)**

- **Flutter:** A cross-platform framework ideal for building high-performance mobile applications on both Android and iOS from a single codebase.
- **Dart:** The programming language used by Flutter.

### **2. Backend (Server-side Logic)**

- **Spring Boot (Java):** A robust and scalable backend framework suitable for building REST APIs and managing business logic.
- **Spring Security:** For user authentication and authorization.

- **PostgreSQL:** A powerful open-source relational database system for managing users, courses, tests, and results.

### 3. AI Integration

- **LLaMA 2 7B Chat (via Ollama):** For generating course content, practice questions, and intelligent feedback.

### 4. Hosting & Deployment

- **Heroku:** For cloud deployment and hosting of the backend services.

### 5. Development & Testing Tools

- **Postman:** For API testing and validation.
- **GitHub:** Version control and project collaboration.
- **Docker** (optional): Containerization of backend services for easier deployment.

## Feasibility Study

### Technical Feasibility

Practice Pal development is, in theory, achievable given the available tools and infrastructure in Zimbabwe and the rest of the world. The project utilizes standard technologies, including Flutter for cross-platform mobile app development, Spring Boot for backend services, and PostgreSQL for secure database management. They are open-source, well-documented, and maintained by large development communities, which makes them adoptable even in places where cutting-edge technical support is not affordable. Moreover, the LLaMA 2 7B Chat model that one can install locally via Ollama or via API services facilitates incorporating artificial intelligence.

This flexibility enables the program to function suitably in bandwidth-constrained or offline environments, a main consideration in Zimbabwe, where pervasive internet connection cannot be guaranteed. As mid-range development equipment and connectivity to developer networks become more available, Zimbabwean institutions and developers are also positioned to deploy such a solution.

### Economic Feasibility

Economic viability of Practice Pal is strong because of free and open-source technologies. Tools like Flutter, Spring Boot, and PostgreSQL negate licensing fees, eliminating financial hurdles to creation and implementation. Running LLaMA 2 locally via Ollama eliminates the requirement to

sign up for paid AI APIs (e.g., OpenAI or Anthropic), saving significant long-term operating costs. This is especially useful in Zimbabwe, where institutions may have thin budgets and foreign exchange limit the ability to access commercial AI offerings. For development, the platform can be hosted on low-cost alternatives such as Heroku's free tier or Google Cloud Platform (GCP) using student credits or research grants. These cost-cutting measures ensure that the solution remains cost-effective for schools, students, and education providers, who are normally disadvantaged by limited ICT funds.

### **Operational Feasibility**

From an operational standpoint, Practice Pal addresses Zimbabwe's education sector's strategic issues. With the rapid shift towards e-learning and independent study, especially following the COVID-19 pandemic, students now more than ever require tools that enable independent, personalised study. The mobile-first strategy of the platform ensures usability on a wide range of devices, from low-end smartphones to tablets, which are used more widely than desktop computers in most Zimbabwean homes. Its user-friendly nature will be suitable for students at the foundation to tertiary levels, which are among national education ambitions that emphasize digital literacy and expanded access to learning resources. Teachers and education administrators will be able to utilize the platform to track progress and refine course delivery without requiring specialist technological knowledge. This makes the solution more sensitive and viable to Zimbabwe's educational institutions, the majority of which are already using ICTs within their teaching and learning processes.

### **Legal and Ethical Feasibility**

The site follows ethical and legal standards in the collection and use of student information, which is a critical component given the heightened sensitivity towards digital privacy and child protection at Zimbabwean schools. Practice Pal does not store or request unnecessary personal information, thus minimizing the risks of data abuse and breaches. In addition, student interactions and performance information are encrypted and stored securely and controlled by Spring Security access controls. The result is that only authorized users (e.g., the student and possibly a related teacher) can view test scores and accomplishments. By adhering to ethical design principles for AI and not attempting predictive profiling and discriminatory assessment, the platform is both transparent and free from bias in its functioning. While Zimbabwe does not yet have particular educational data privacy regulations in place, Practice Pal is compliant with best practice in international data security and can be easily adapted to adapt to future regulation standards as they emerge.

## Time Feasibility

Practice Pal development is possible in 3-6 months based on the size and experience of the development team. A small development team or an individual developer with a part-time or full-time designation can attain key milestones provided there is good planning and allocation of resources. The first month can be spent on UI/UX design and backend setup, where one can construct the Flutter interface and configure the Spring Boot server and PostgreSQL database.

In the following two months, key features such as AI-based test generation, automated scoring, and immediate feedback can be conceived and deployed. Extensive testing (unit, integration, and system) and ultimate release can be done in the last 1-2 months. This timeline is possible, considering the modular nature of the system and availability of frameworks that allow backend and frontend development to be accelerated. It also provides sufficient buffer time for feature polishing and customer feedback so that there is consistent performance in the real world, which is very important in view of Zimbabwe's bandwidth and infrastructure volatility issues.

## Project Plan

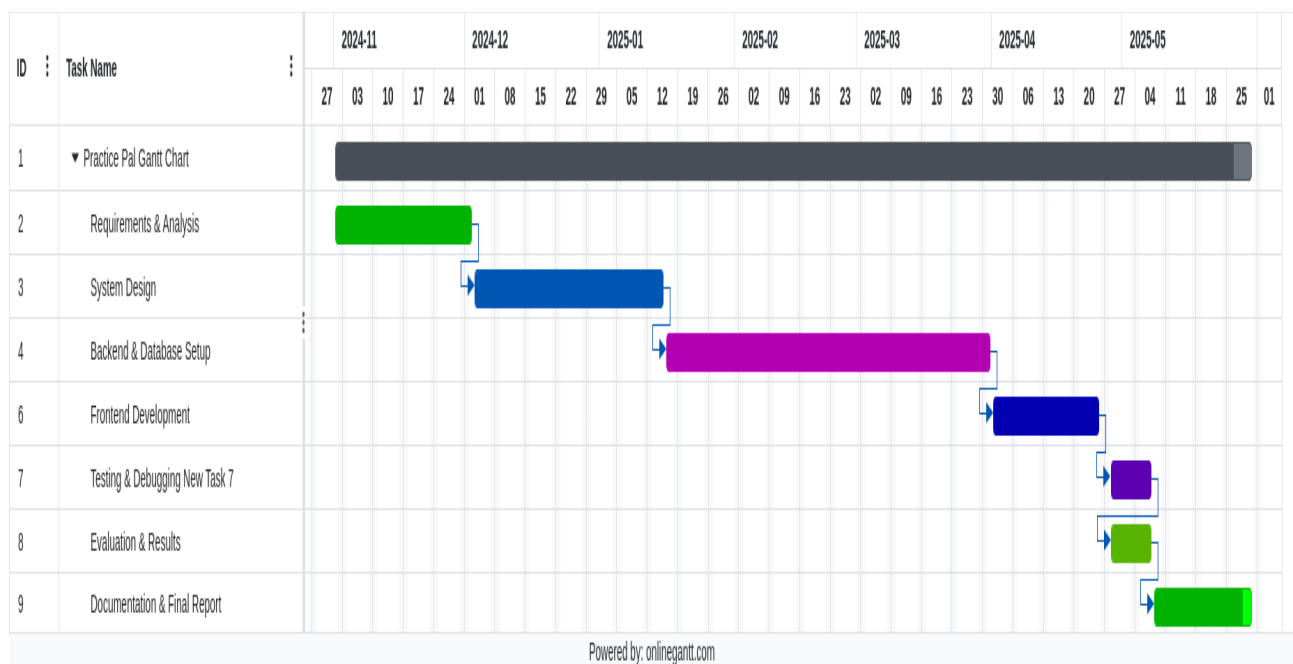


Figure 1: Gantt chart

## **Chapter Two: Literature Review**

### **Introduction**

In the recent past, the use of artificial intelligence in educational technologies has increased manifold. In their contemplation of the ethical challenges posed by AI to teaching in K-12, Akgun and Greenhow [1] emphasize the importance of designing student-centric technologies that facilitate learning without trading off data privacy. Programs like Quizlet and others have gained popularity with its simple-to-use interface and community-shared flashcards [3]. But they cannot offer dynamic, customized tests; instead, they rely on static user-provided content. Similarly, sites like Khan Academy provide existing classes and tests [4], but these cannot be customized to meet the needs of individual students.

Duolingo, being a language learning program, has been able to gamify learning by encouraging routine practice through short tasks and constructive feedback [5]. Its content is limited to language learning and does not allow learning on topics or courses of the user's choice. Further, adaptive learning systems, discussed by Aleven et al. [2], reflect utilization of AI to allow for personalised learning sessions although their typical application is limited to selected topics and educational institutions.

Despite all these developments, webpages that allow the users, basically primary to undergraduate level students, to enter their topics, receive organized knowledge created by AI, and engage in interactive, test-like learning still lag behind. Practice Pal shall complete this gap by employing the LLaMA 2.7 B Chat model to generate course maps and practice tests on demand, along with introducing features like test review, retesting, and automatic grading and feedback. This shall be utilized for making the self-assessment tool more flexible and intelligent so that students are empowered to steer their education.

### **Related Work**

#### **Quizlet**

Quizlet is a popular online learning platform that allows both students and teachers to create, study, and share customized flashcard decks. It offers various study modes, including standard flashcards, fill-in-the-blank quizzes, multiple-choice exams, and matching games. Users can also use pre-made decks from a shared community library. Quizlet has become especially popular for topics that involve memorisation, such as language, science facts, and historical dates.



For all its capabilities, Quizlet is restricted in terms of adaptability and intelligent feedback. Study material is pre-set; once a set of flashcards or quizzes is designed, the content doesn't alter unless changed manually. Lacking dynamic generation, the website cannot adjust automatically to a particular user's progress or generate customised tests. Furthermore, Quizlet does not leverage artificial intelligence to assess question difficulty or make targeted recommendations, making it inadequate for long-term, independent learning experiences that take place over a period of time.

### **Khan Academy**

Khan Academy is a free, nonprofit educational website that offers structured video lessons, interactive exercises, and practice problems in various subjects like mathematics, physics, economics, and humanities. It offers students a curriculum-based progression that includes progress monitoring, quizzes, and mastery-based learning features. Khan Academy is well known for its user-friendly education videos that break up challenging concepts into manageable chunks.

Its evaluation and testing modules, however, are generally inflexible and pre-designed and do not readily permit users to tailor tests or create content on the fly. The platform's structured nature is ideal for formal education support. It is less flexible, however, for students wanting to study non-traditional subjects or create personalized tests based on their study plan. Although it provides some adaptive learning through mastery tracking, it does not support user-defined question generation or AI-based test personalization, so its utility for open-ended self-testing is limited.

### **Duolingo**

Duolingo, the gamification-capable language learning software, is an active learning platform that instructs students in grammar, vocabulary, reading, writing, listening, and speaking. Its interactive little exercises, voice recognition pronunciation, spaced repetition memorization, and reward system entertain while teaching. The dynamic feedback systems within the software, which adjust lesson difficulty based on user performance, provide encouragement and reassurance, hence one of the most active learning platforms available in the language learning market.

While Duolingo is a great language-learning tool, it does come with its limitations. It is only utilized for the singular purpose of language learning and may not be altered or developed by users. What this means is that it cannot be utilized for creating unique tests or designing new topics outside of the pre-prescribed syllabi. For students or instructors looking to build tailored quizzes on subjects aside from language learning, like mathematics, science, or computer studies, Duolingo is not supportive. Also, its feedback is usually linguistic correctness and does not include higher-level judgments or performance ratings that can be translated to other learning subjects.

## **Conclusion**

Despite huge advances in ed-tech, present systems cannot deliver the maximum potential for furnishing students with closely personal, AI-supported learning and assessment experiences. Duolingo and Quizlet have been essential through play and fun, but they are of limited scope, tending to be attached to locked content and specialist subject areas. Similarly, Khan Academy and adaptive tutoring programs offer controlled learning but no user-created content adaptability and no way of presenting dynamically generated exams based on particular needs.

Practice Pal is different by addressing these shortfalls. With the users' topics inputted, the website uses the LLaMA 2.7 B Chat model to supply course material and practice tests on demand. Practice Pal addresses an inherent lack of test-based, personalized learning through the addition of features such as automatic scoring, immediate feedback, and revision functionality. This makes it a future-driven learning device that equips students from primary to undergraduate levels to take control of their academic life.

## **Chapter Three: Analysis**

### **Information Gathering Tools**

#### **Interview Questions**

1. What methods do you currently use to revise before exams?
2. Do you ever test yourself before a real test? If yes, how?
3. What do you find hardest about revising or preparing for tests?
4. Are you happy with the feedback you get after class tests or school exams?
5. What frustrates you about current study apps or websites?
6. If there was an app that could generate test questions from topics you choose, what would you want it to do?
7. Would you prefer an app that marks your test immediately and explains the correct answers
8. How important is it for you to track your performance over time (e.g., see improvement)
9. What would make you use an app regularly to help you study?
10. What kind of phone do you use (Android/iPhone)?
11. Do you have regular internet access or would you prefer something that works offline?

## Questionnaires

- 1 **What is your current level of education?**  

Primary School	Secondary School	College/University	Technical
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
  
- 2 **What subjects do you study or struggle with the most?**
  
- 3 **How do you prepare for tests and exams?**  

Reading notes	Use textbooks	Watch videos	Use apps	Other (Specify)
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/> _____
  
- 4 **Do you currently use any apps or platforms to test yourself or revise?**  

Yes	<input type="checkbox"/>
No	<input type="checkbox"/>
  
- 5 **What do you feel is lacking in the tools you currently use?**  

No practice questions in local curriculum	Not enough feedback after tests	No way to track performance	Boring or difficult to use	Other (Please Specify)
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/> _____
  
- 6 **If there was a free app that could generate custom test questions based on your topic or subject, how useful would it be to you?**  

Extremely useful	Useful	Not sure	Not useful
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
  
- 7 **What device would you use to access the app?**  

Mobile device	Tablet	Laptop/Desktop	Other (Please specify)
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/> _____
  
- 8 **Would you be willing to try such an app and give feedback during development?**  

Yes	<input type="checkbox"/>
No	<input type="checkbox"/>
Maybe	<input type="checkbox"/>

Figure 2: Questionnaire

## Document Review

Table 1: Document review

Application	Used for	Strengths	Limitations
Quizlet	Flashcards, study sets, and basic quizzes.	Quick revision, community content, flexible subjects.	No question generation, no progress tracking, lacks curriculum alignment, and minimal feedback.
Khan Academy	Watching subject-specific videos and answering pre-made questions.	Free, quality content, aligned with global standards.	Tests are rigid and not user-customized.
Duolingo	Language learning.	Fun interface, gamified progression, regular feedback.	Limited to language, cannot input your own topics, no support for academic testing.
Google Forms	Teacher-made quizzes or tests.	Customizable by teachers.	No instant feedback, no automated analysis, no performance tracking, not engaging.

## Description of System

Practice Pal is a mobile education platform that utilizes artificial intelligence to help students create and take personalized practice exams according to their needs and studies of interest. As they continue learning at their own pace, the system offers users a chance to input any secondary information or curriculum, beginning with primary math, and create a structured outline of the curriculum automatically. According to this plan, students can create personalized tests according to the number of predeterminations and the level of difficulty (simple, medium, or complex). Upon completion of the test, the system qualifies and provides full feedback immediately so that students can review their performance and identify areas where they must improve. Users are also able to revise their previous exam intentions, see right answers and retake the exams to redirect their learning and map a path on their academic development in the long term.

Internally, Practice Pal has a modern and scalable tech stack. The front end is built with Flutter, which delivers a seamless mobile experience on a wide range of Android devices. The backend is powered by Spring Boot, which handles core business logic, user authentication, and communication with the AI engine and database. For text generation, the site uses the LLaMA 2 7B model via Ollama to enable on-device or offline AI processing without invoking expensive third-party APIs. The AI model generates high-quality examination questions and schemas that are personalized to user input. It securely stores all data, including examination scores, user profiles, and progress indicators, in a PostgreSQL database. Overall, the system seeks to foster accessible,

scalable, and adaptable learning, particularly among students in Zimbabwe, who lack customized learning tools and technology aligned with their unique requirements.

### **Evaluation of Alternatives Systems**

To develop an effective AI-powered practice test platform, such as Practice Pal, existing education systems must be evaluated for strengths, weaknesses, and applicability to Zimbabwe's diverse population of learners. Several well-known systems, such as Quizlet, Khan Academy, Duolingo, and Google Classroom, provide learning support. None of them, however, provide a fully adaptive and personalised test-based learning experience that is customised to local needs.

Quizlet is a popular learning application that allows the creation and sharing of flashcards, as well as the use of various learning modes, including matching games and multiple-choice exams. Although it allows for collaborative learning and simple testing, its content is relatively static and user-generated, rendering it less dynamic. It lacks intelligent feedback mechanisms, does not support AI-driven question generation, and may not be curricula-aligned to Zimbabwe or provide offline access for children in low-connectivity areas.

Khan Academy offers video lectures with elaborate details, practice exercises with step-by-step instructions, and mastery tracking across a variety of subjects. While it offers a premeditated curriculum and progress tracking, its tests are pre-designed and not flexible. Students cannot make their own tests to suit their personal needs, and it is largely reliant on internet connectivity, which is a setback to poorly equipped schools and rural Zimbabwean students.

Duolingo provides lessons in small chunks, and instant feedback through gamification and rewards helps to reinforce language learning and increase engagement. However, its utility is limited to learning languages only and cannot be utilized for additional subjects such as mathematics or science. It also lacks the feature of creating user-generated exam content or learning courses outside of the predefined language curriculum.

Google Classroom, which is being used in the majority of Zimbabwean schools during remote learning seasons, is a classroom management system instead of a personalised learning system. Though it allows teachers to make quizzes, distribute resources, and communicate with students, it does not have AI-powered assessment generation, auto-marking, and dynamic content adaptation.

In this change, Practice Pal bridges these gaps to allow Zimbabwean students to generate practical exams with IA on demand, monitor their performance, and receive immediate and personalized feedback. A key difference among the alternatives is the complete personalization of the tests,

enabling the inference without an interdependence of the IA models discarded locally, for example, LLaMA 2, by the Ollama channels, and which prioritizes mobility, fundamental for accessibility in areas where smartphones are more accessible than computers. It also facilitates a wide range of signatures, embedded gaming and feedback loops. It is implemented with inexpensive code, making it more sustainable and suitable for the Zimbabwean school context.

### Existing System (Quizlet)

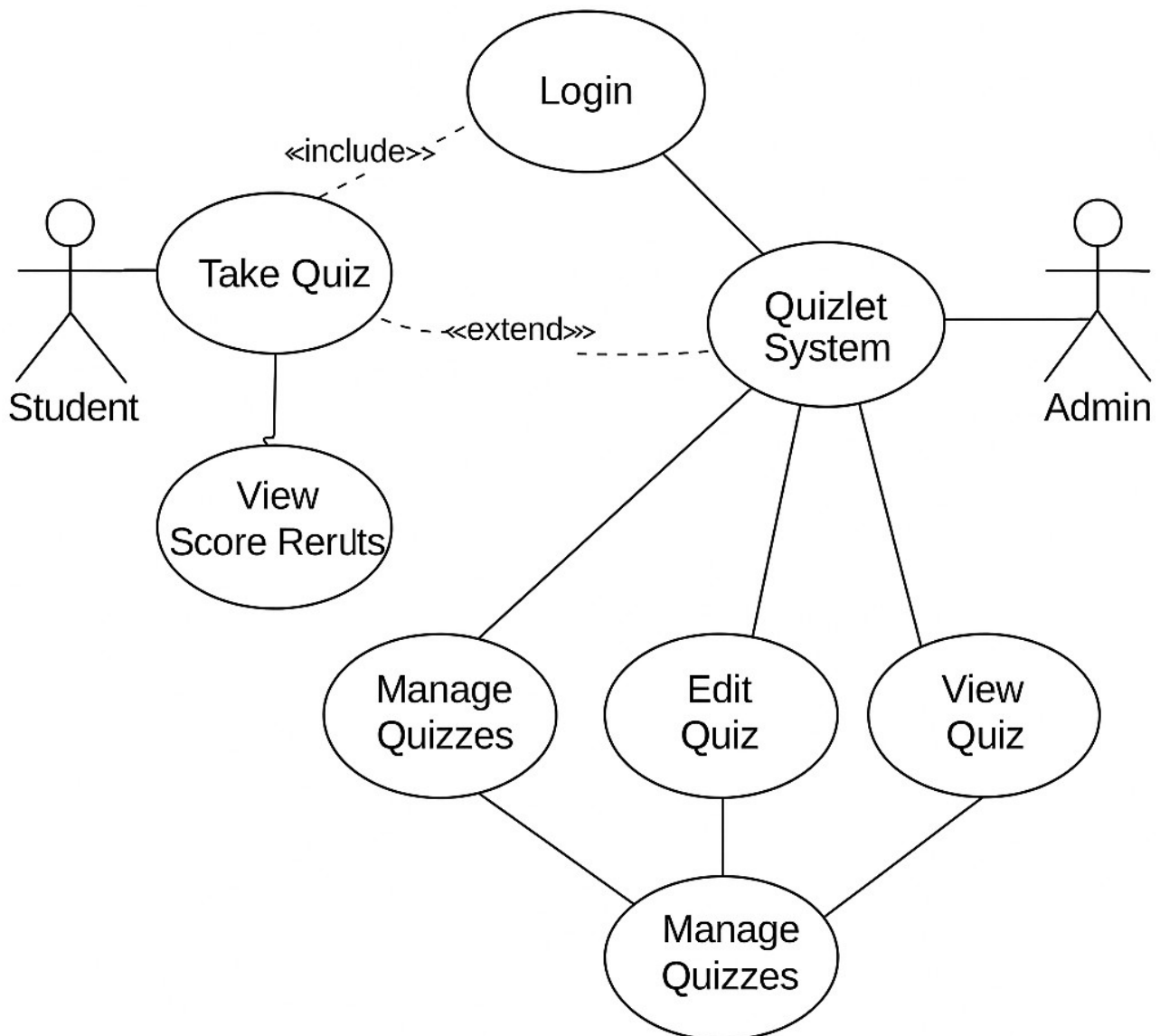


Figure 3: Quizlet use-case diagram

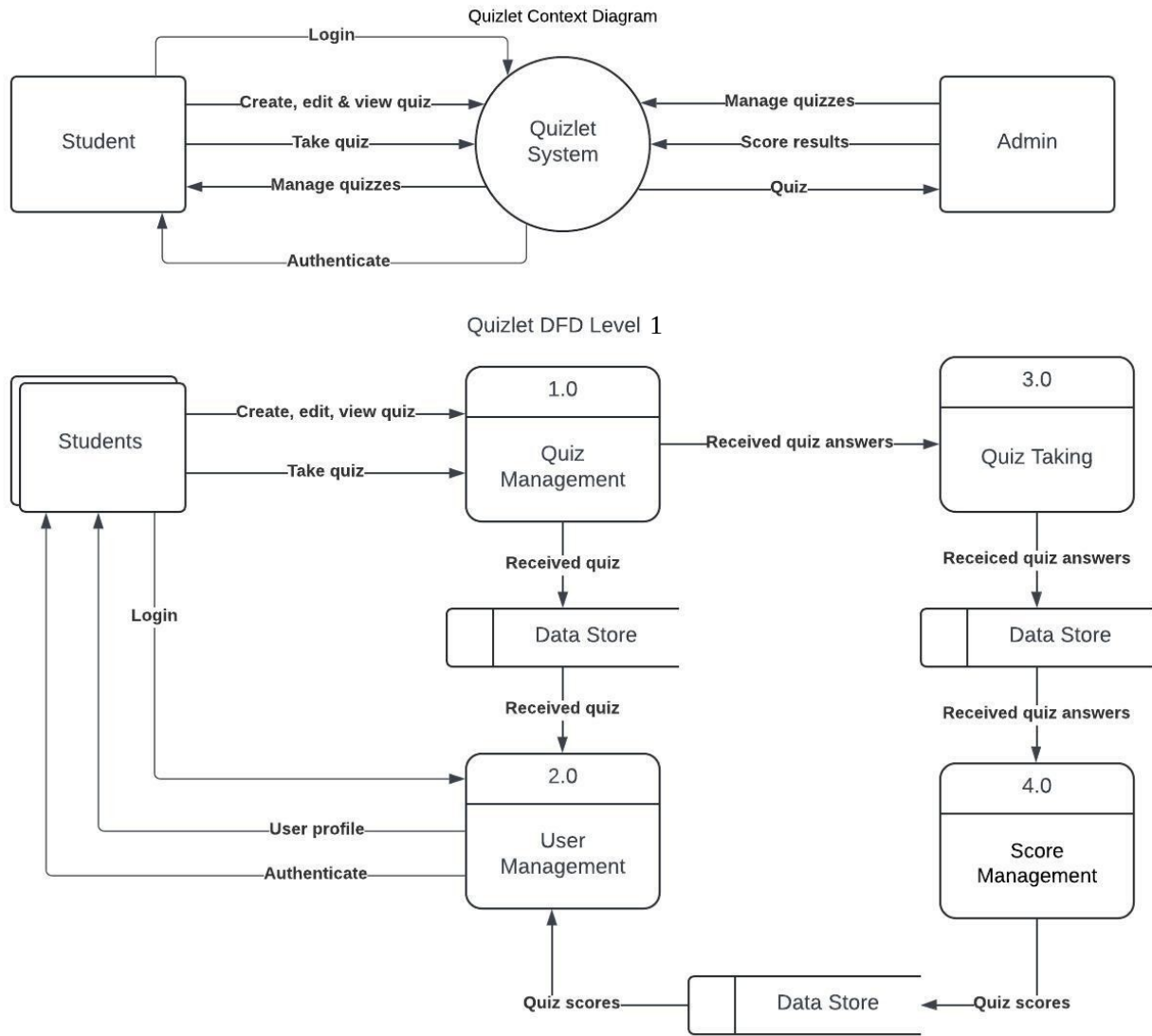


Figure 4: Context diagram and DFD Level 1 for Quizlet

## Functional Analysis of Proposed

In assessing Practice Pal's functional design, it is necessary to appreciate how its features compare to or exceed those of existing platforms. The technology applies artificial intelligence to develop customized practice tests, provides quick feedback through automated scoring, and helps the learner track and improve over time. This aspect is designed to overcome the inflexible, one-size-fits-all model in most systems currently available.

Quizlet, for example, is popular for interactive flashcards and easy quiz formats that allow repetition and memorization. It is largely user-generated content and lacks any adaptive learning functionality. It doesn't apply AI in the sense of generating question sets that are unique to the



learner's level or tracking progress in a substantive and systematic manner. There is no way to enter a subject and receive a full test prepared on the go, which hinders its value for serious exam preparation, especially in nations such as Zimbabwe, where compliance with domestic curricula is essential.

Khan Academy is very good at delivering properly structured and educationally valid lessons in a number of subjects, including science, math, and the humanities. It provides mastery tracking, instructional videos, and practice exercises. Its tests are static and pre-structured although they can be very penetrating. Students are not able to define topics for themselves or construct tests based on their own weaknesses. This restricts the scope of Zimbabwean students who may wish to study something outside the pre-packed syllabus or specialize in very specialized subjects from ZIMSEC or Cambridge syllabi.

Duolingo is perhaps the most popular gamified learning environment, and certainly for language learning. A daily goal, instant feedback, and micro-learning organization facilitate regular practice. Duolingo is usefully limited, though, in that it does not facilitate user-generated content and cannot extend to support learners in fields outside language subject areas. It is not possible for Zimbabwean science, maths or other non-language course learners to use Duolingo to prepare or take curriculum-based practice examinations.

By watching a student's behavior and adapting material accordingly, adaptive tutors—that are commonly driven by AI—deliver tailored learning experiences. Such software, such as Pearson's MyLab or Carnegie Learning, is beneficial but tends to be costly, centralized, and curriculum-bound. They rarely are available offline or for free and are mostly applied in formal schooling. Students in rural villages of Zimbabwe or schools with limited funding are faced with this obstacle.

Practice Pal takes the best from all these systems but lessens their drawbacks. It allows students to surf through any course topic, whether practical or theoretical and download a detailed and organized plan. From here, they can generate exams with parameters such as difficulty level and question number. It also offers automated marking, immediate feedback, performance tracking, as well as reviewing or repeating exams. It is mobile, web-based, and created using free and open-source technologies (Flutter, Spring Boot, LLaMA 2) so that it can be accessible to students across Zimbabwe regardless of the school setup or availability of internet. It also includes localized and adaptive features that distinguish it from standardized global systems and qualify it as a viable choice for Zimbabwean students.

## System Requirements

### Functional Requirements

- The system must allow users to input a course/topic and generate content using AI.
- It must provide a test generation interface that accepts parameters like difficulty and question count.
- It must store user data, including test history and performance analytics.
- The system must allow users to review and retake previous tests.
- It must automatically mark answers and return feedback.

### Non-Functional Requirements

- **Performance:** The system should respond within 2 seconds for all test generation and grading operations.
- **Scalability:** The backend should support hundreds of concurrent users without degradation.
- **Security:** All personal data and test results must be securely stored and transmitted.
- **Usability:** The app must offer a clean and intuitive interface accessible to students of all levels.
- **Portability:** The system must work seamlessly on both Android and iOS.
- **Maintainability:** The codebase must be modular and documented for future updates.

## Use-Case Diagram for Proposed System

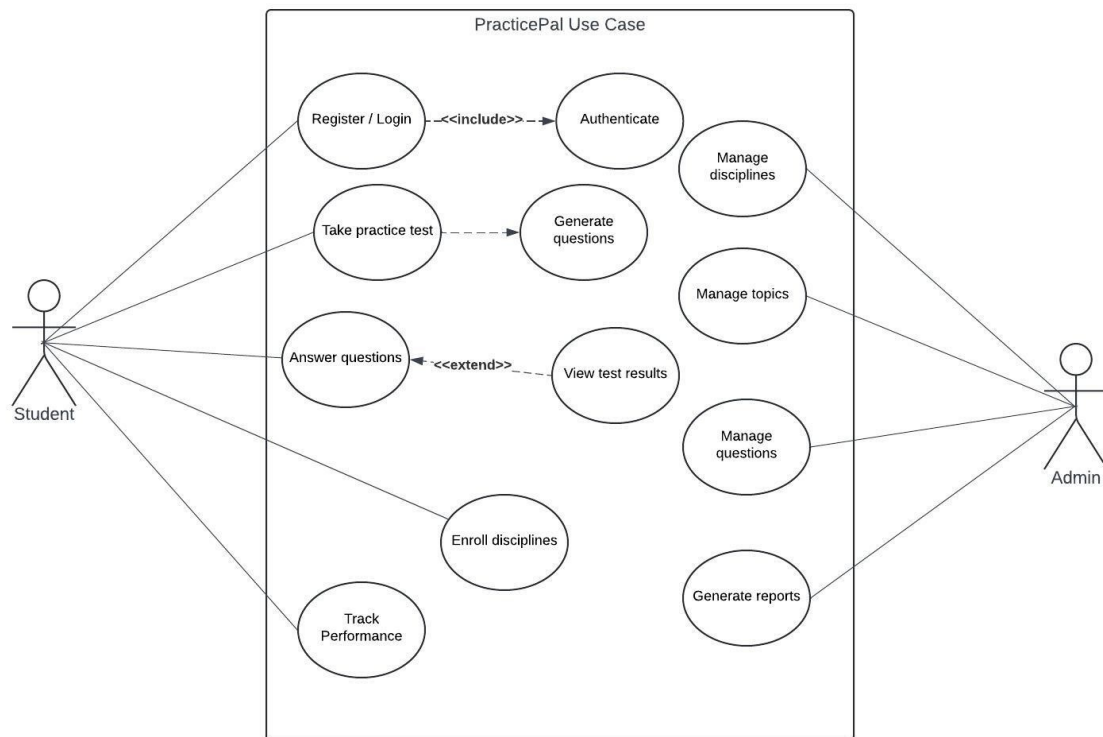


Figure 5: Use-case diagram

## Chapter Four: Design

### System Diagrams:

#### Context Diagram

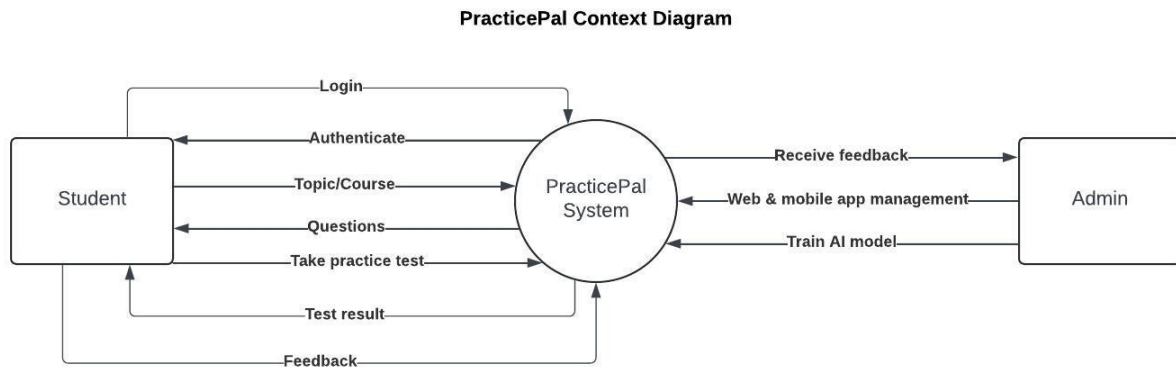


Figure 6: Context diagrams

#### Data Flow Diagram (DFD) Level 1

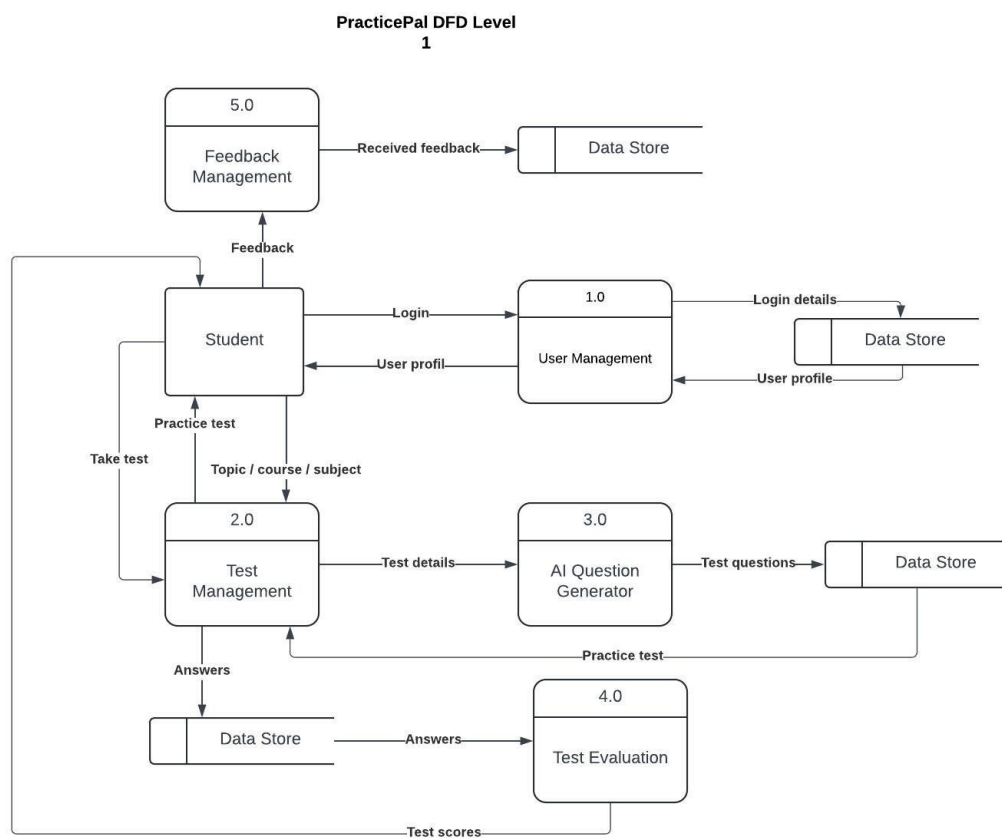


Figure 7: Data Flow Diagram (DFD) Level 1

## Activity Diagram

### Practice Pal

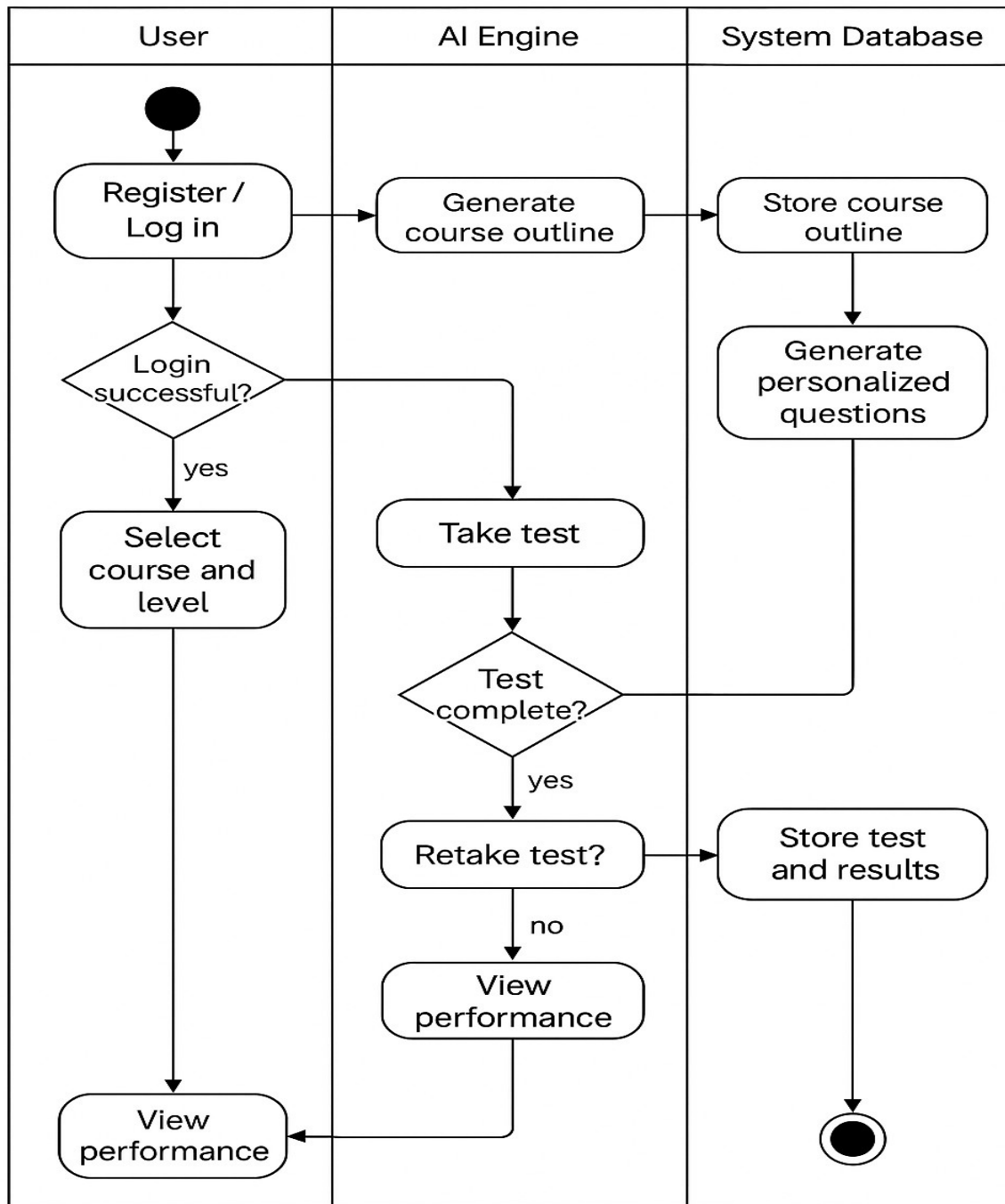


Figure 8: Activity diagram

## Database Design:

### ER Diagrams

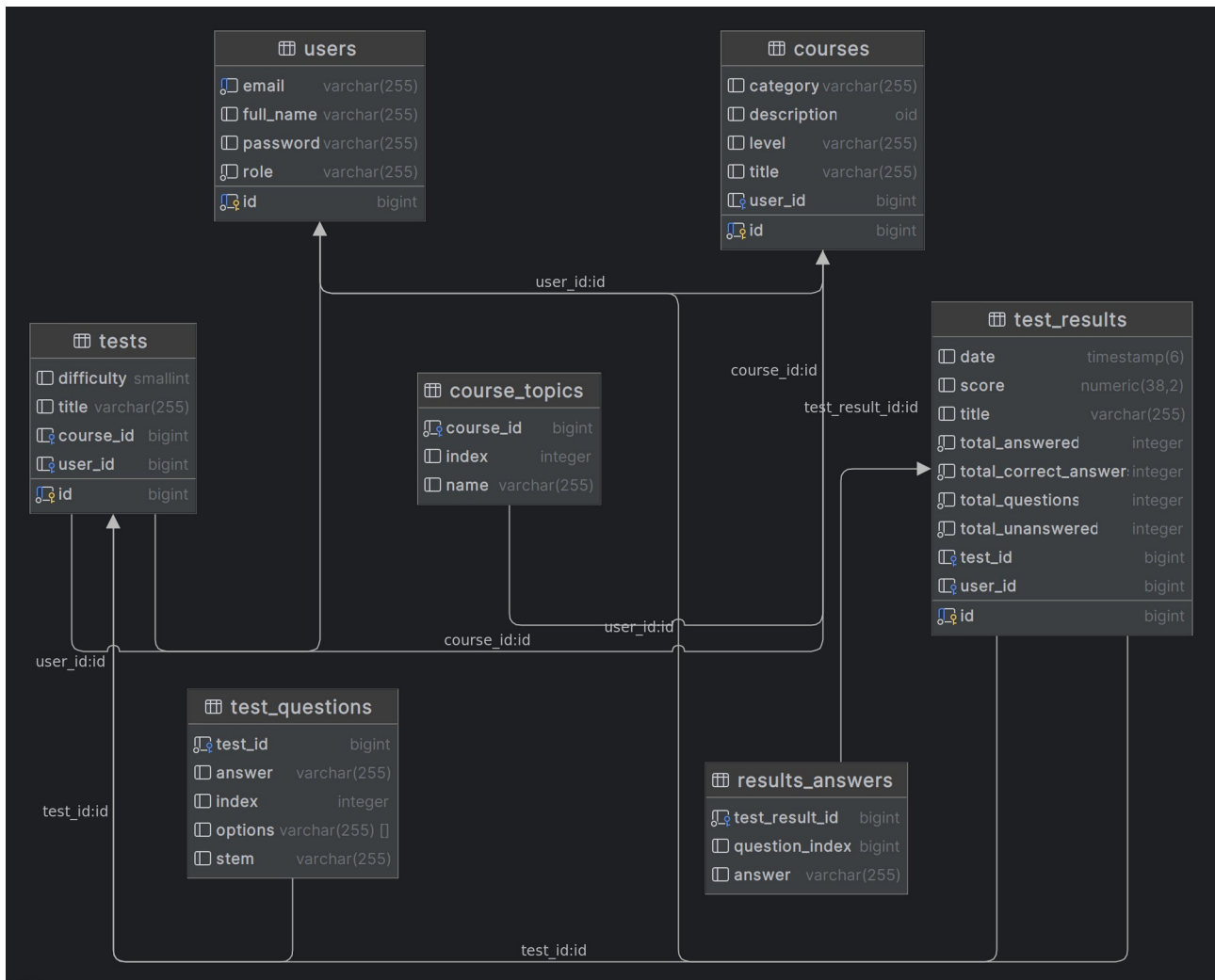


Figure 9: ER diagram

## Normalized Database

*Table 2: Users table*

Column Name	Data Type	Description
email	varchar(255)	User's email address
full_name	varchar(255)	User's full name
password	varchar(255)	Hashed password
role	varchar(255)	Role of the user (e.g., admin, student)
id	bigint	Primary key

*Table 3: Courses table*

Column Name	Data Type	Description
category	varchar(255)	Category of the course
description	oid	Description of the course
level	varchar(255)	Difficulty level
title	varchar(255)	Course title
user_id	bigint	ID of the user who created the course
id	bigint	Primary key

*Table 4: Course topics table*

Column Name	Data Type	Description
course_id	bigint	Associated course ID
index	integer	Topic index
name	varchar(255)	Topic name

Table 5: Tests table

Column Name	Data Type	Description
difficulty	smallint	Test difficulty level
title	varchar(255)	Title of the test
course_id	bigint	Associated course ID
user_id	bigint	ID of the user who created the test
id	bigint	Primary key

Table 6: Test questions table

Column Name	Data Type	Description
test_id	bigint	Associated test ID
answer	varchar(255)	Correct answer
index	integer	Question index
options	varchar(255)[[]	List of options
stem	varchar(255)	Question stem

Table 7: Test results table

Column Name	Data Type	Description
date	timestamp(6)	Date and time of the test
score	numeric(38,2)	Score obtained
title	varchar(255)	Title of the test result
total_answered	integer	Number of questions answered
total_correct_answers	integer	Number of correct answers
total_questions	integer	Total number of questions



total_unanswered	integer	Number of unanswered questions
test_id	bigint	Associated test ID
user_id	bigint	ID of the user who took the test
id	bigint	Primary key

## Program Design:

### Class Diagram

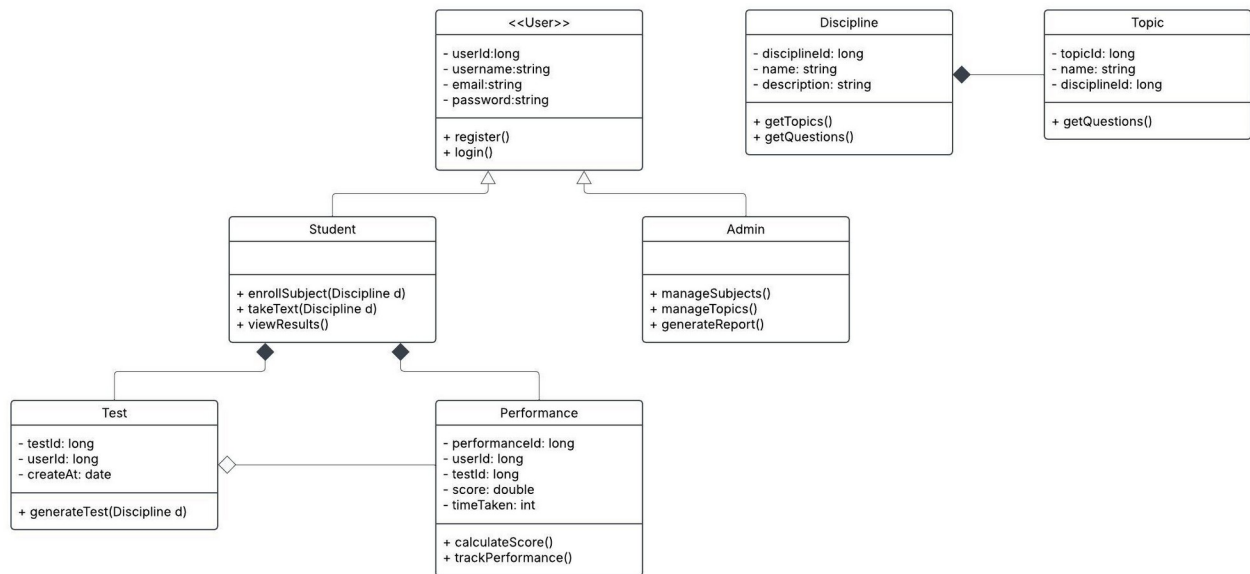


Figure 10: Class diagram

## Sequence Diagram

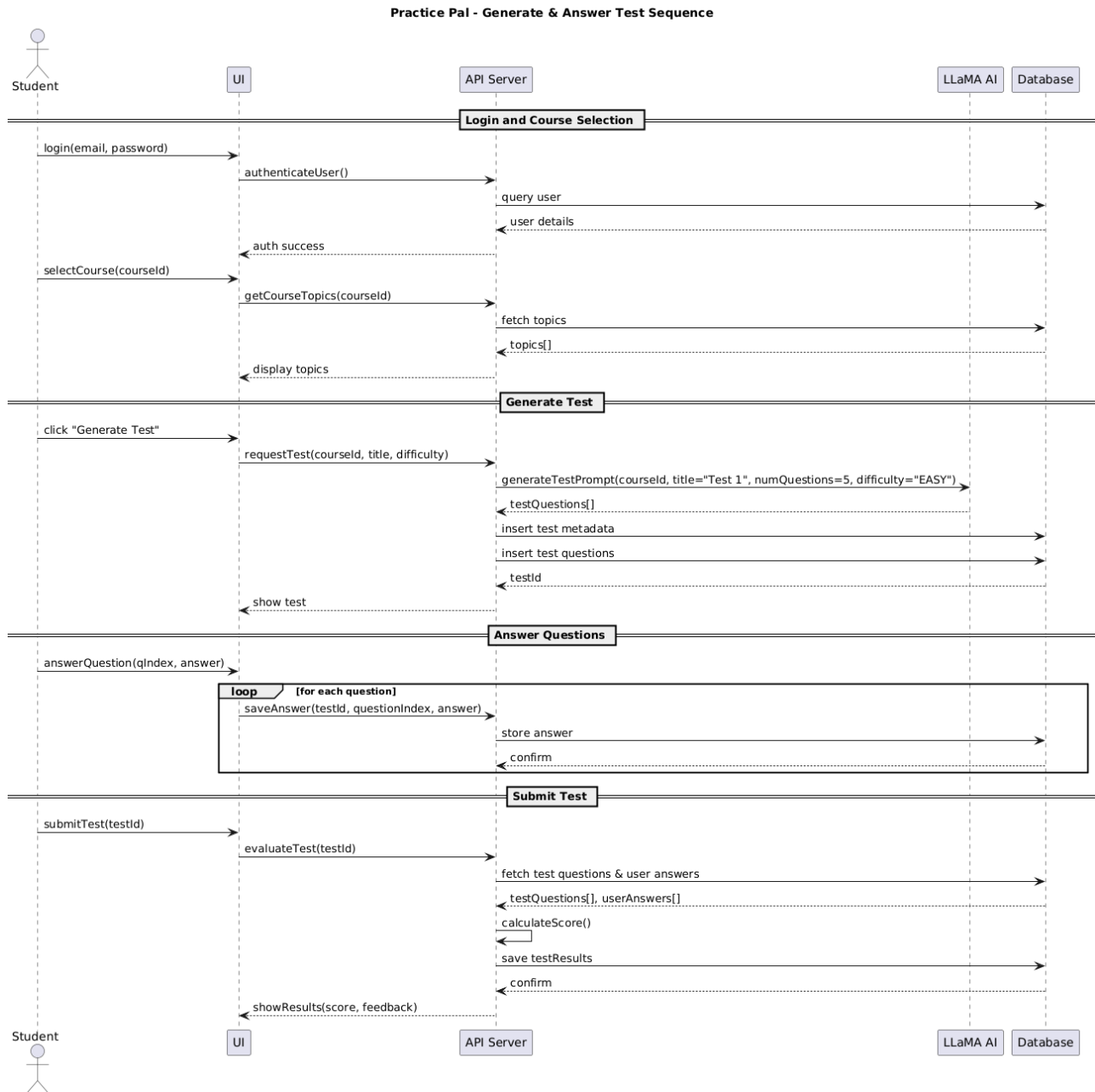
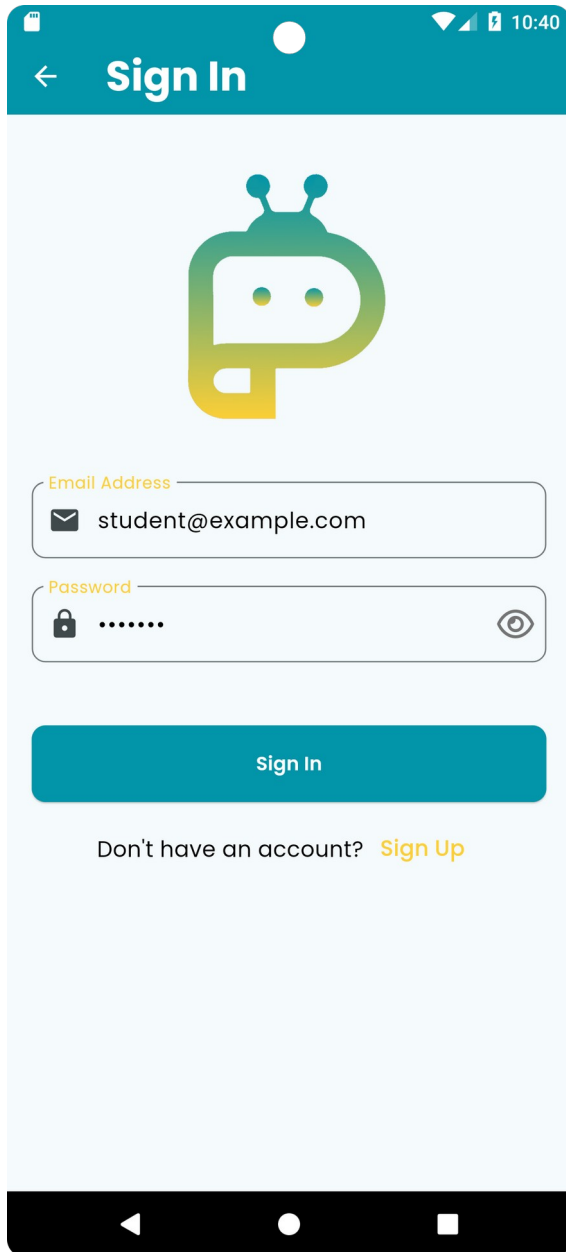


Figure 11: Sequence diagram

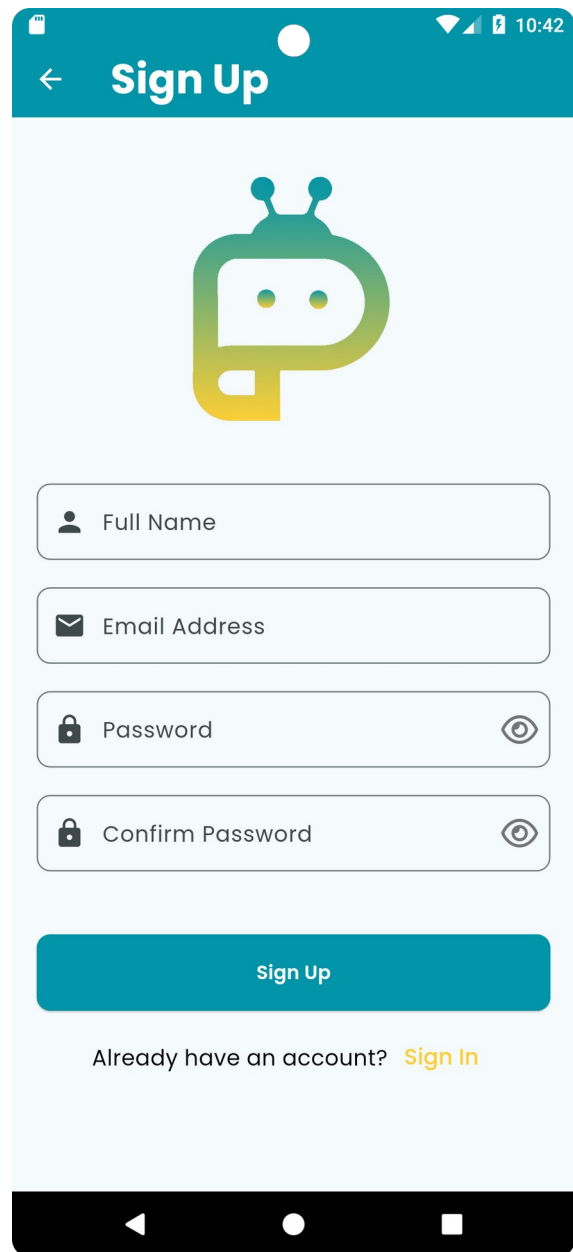
## Interface Design:

### Authentication Pages



The Sign In page features a teal header with a back arrow and the title "Sign In". Below the header is a large, stylized robot logo with a green-to-yellow gradient. The form consists of two input fields: "Email Address" with a mail icon and the text "student@example.com", and "Password" with a lock icon, masked dots, and a toggle eye icon. A teal "Sign In" button is positioned below the fields. At the bottom, a link reads "Don't have an account? Sign Up". The status bar at the top shows the time as 10:40.

Figure 12: Sign in page



The Sign Up page features a teal header with a back arrow and the title "Sign Up". Below the header is the same stylized robot logo. The form includes four input fields: "Full Name" with a person icon, "Email Address" with a mail icon, "Password" with a lock icon, masked dots, and a toggle eye icon, and "Confirm Password" with a lock icon, masked dots, and a toggle eye icon. A teal "sign Up" button is located below the fields. At the bottom, a link reads "Already have an account? Sign In". The status bar at the top shows the time as 10:42.

Figure 13: Sign up page

Home Page

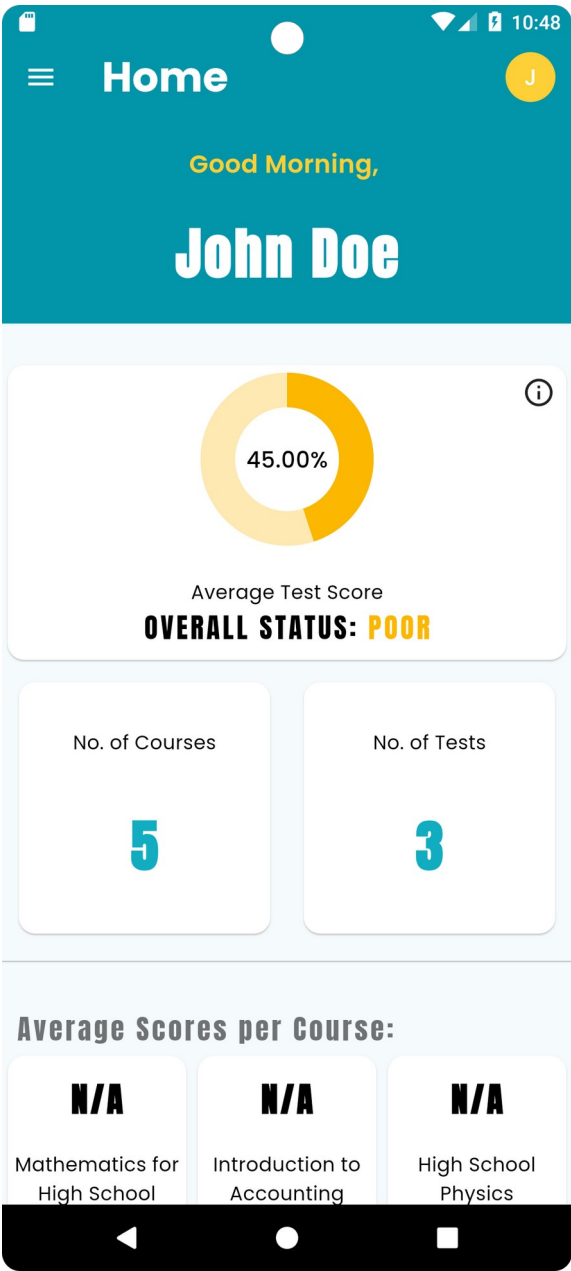


Figure 14: Home page

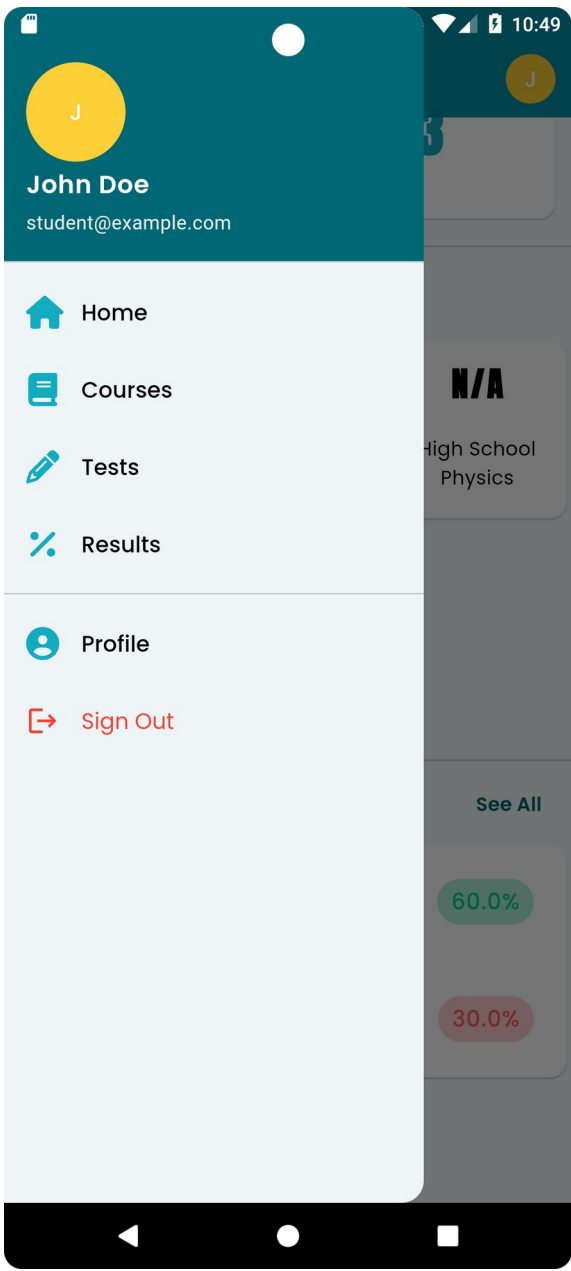


Figure 15: Home drawer

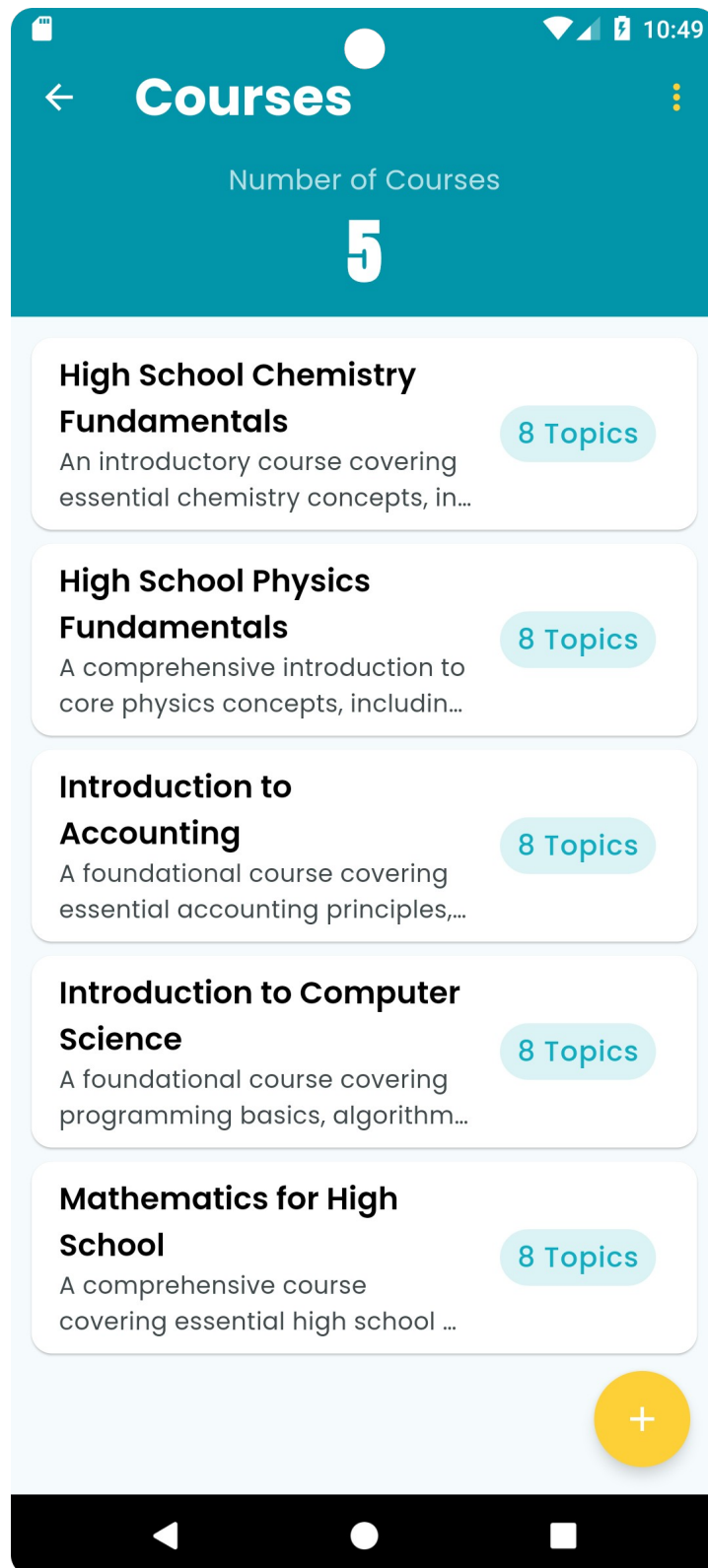
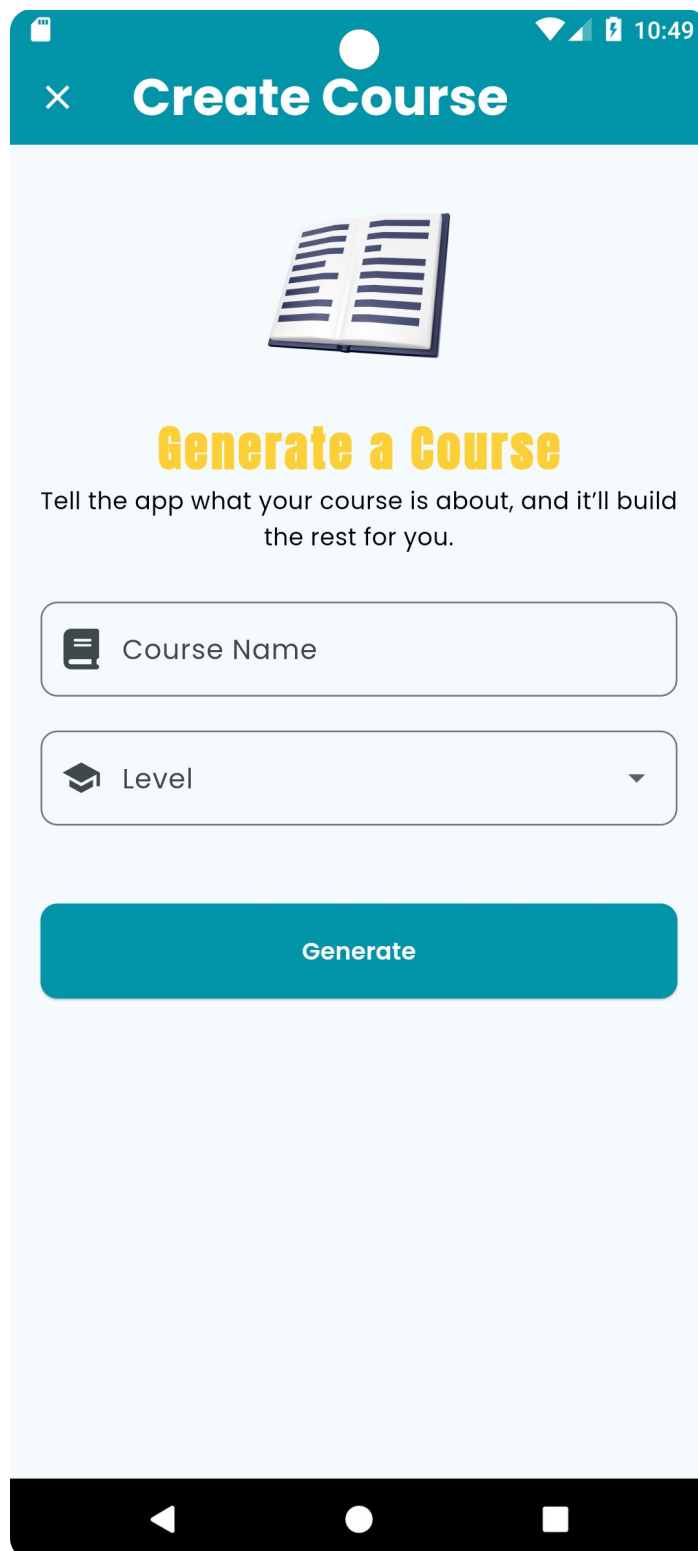



Figure 16: Courses page

## Generate Course Page




The image shows a mobile application interface for creating a course. At the top, there is a teal header with a white 'X' icon on the left and the text 'Create Course' in white. Below the header, there is a light blue background. In the center, there is an illustration of an open book. Below the book, the text 'Generate a Course' is displayed in a large, bold, yellow font. Underneath this, a smaller black font reads: 'Tell the app what your course is about, and it'll build the rest for you.' There are two input fields: the first is labeled 'Course Name' with a document icon on the left, and the second is labeled 'Level' with a graduation cap icon on the left and a dropdown arrow on the right. Below these fields is a large teal button with the word 'Generate' in white. At the bottom of the screen, there is a black navigation bar with three white icons: a back arrow, a circle, and a square.


× Create Course



### Generate a Course

Tell the app what your course is about, and it'll build the rest for you.

 Course Name

 Level ▼

Generate

Figure 17: Generate course page

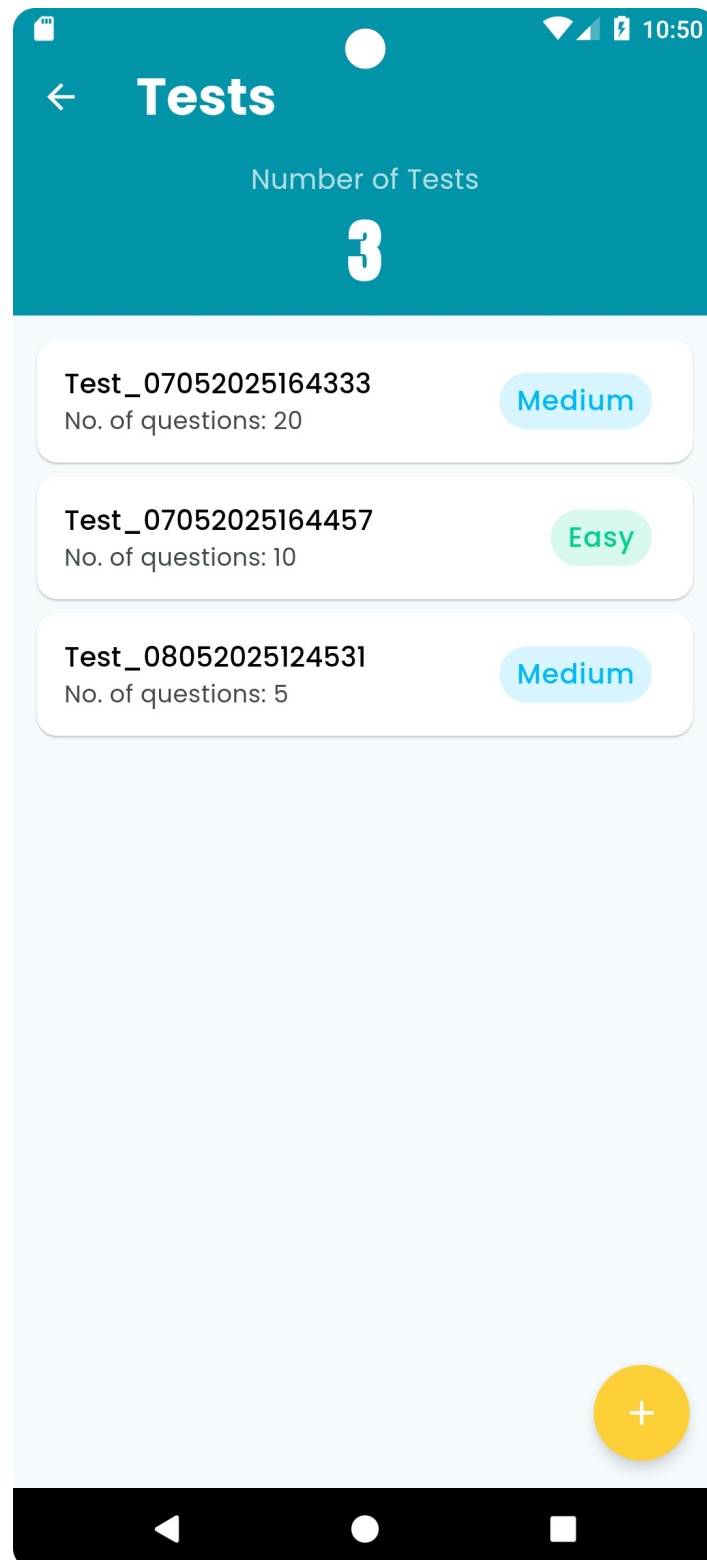
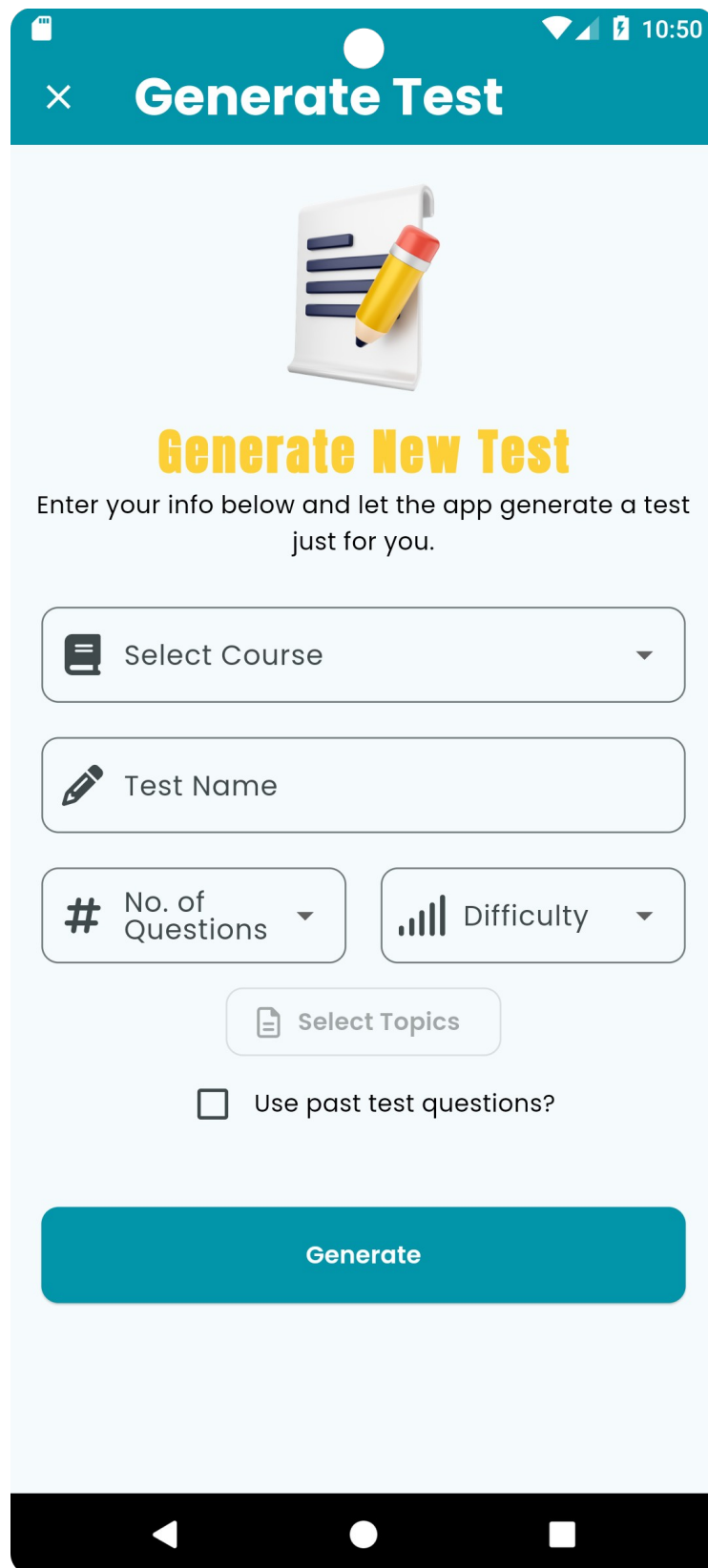



Figure 18: Tests page




A screenshot of a mobile application interface for generating tests. The app has a teal header with a close button (X) and the title "Generate Test". Below the header is a light blue area containing an illustration of a notepad and a yellow pencil. The main heading "Generate New Test" is in bold yellow text. Below it, a subtitle says "Enter your info below and let the app generate a test just for you." The form includes several input fields: a "Select Course" dropdown with a document icon, a "Test Name" text field with a pencil icon, a "No. of Questions" dropdown with a hash icon, and a "Difficulty" dropdown with a bar chart icon. There is also a "Select Topics" button with a document icon and a checkbox labeled "Use past test questions?". A large teal "Generate" button is at the bottom of the form. The bottom of the screen shows a black Android navigation bar with back, home, and recent apps icons. The status bar at the top shows signal strength, Wi-Fi, battery, and the time 10:50.


× **Generate Test**





**Generate New Test**


Enter your info below and let the app generate a test just for you.

 Select Course ▼

 Test Name

 No. of Questions ▼

 Difficulty ▼

 Select Topics

☐ Use past test questions?

**Generate**

Figure 19: Generate test page

## Test Results Page

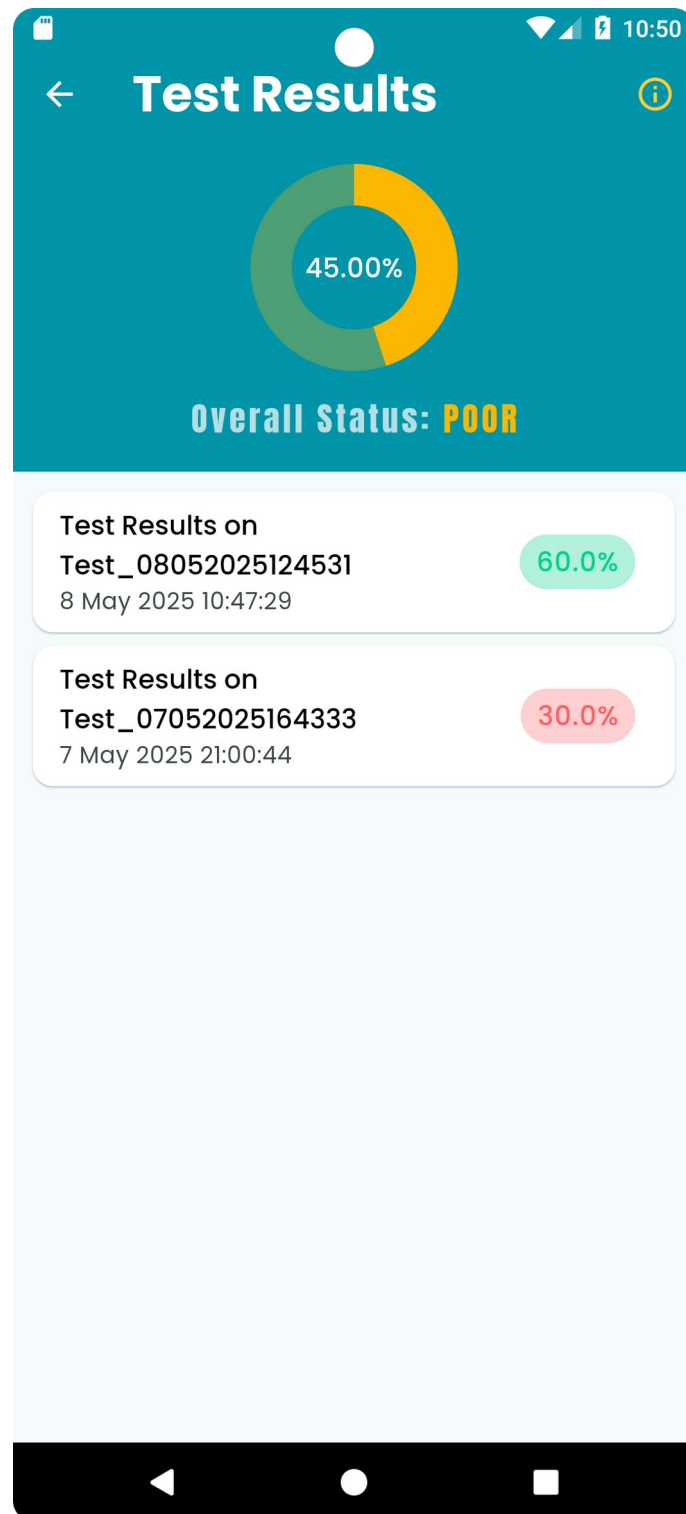


Figure 20: Test results page

## Test Performance Page

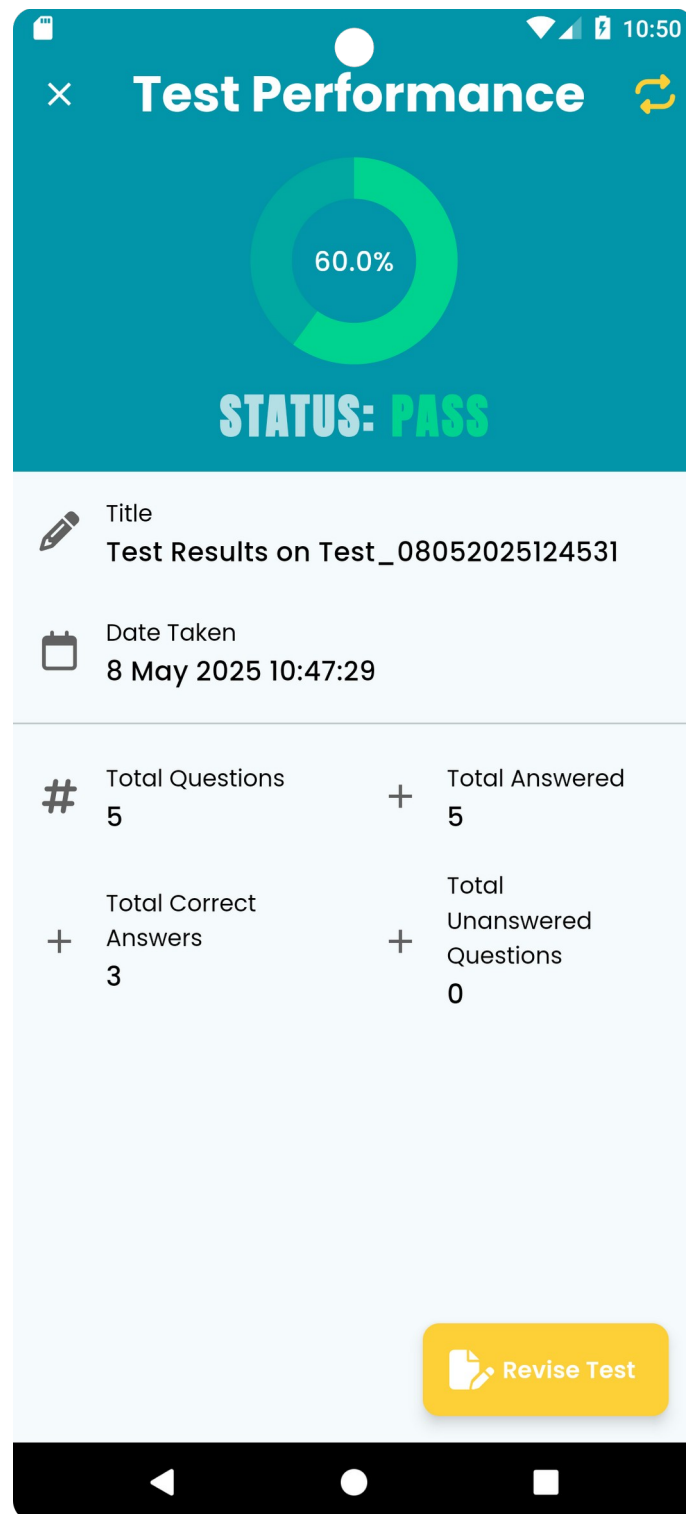


Figure 21: Test performance page

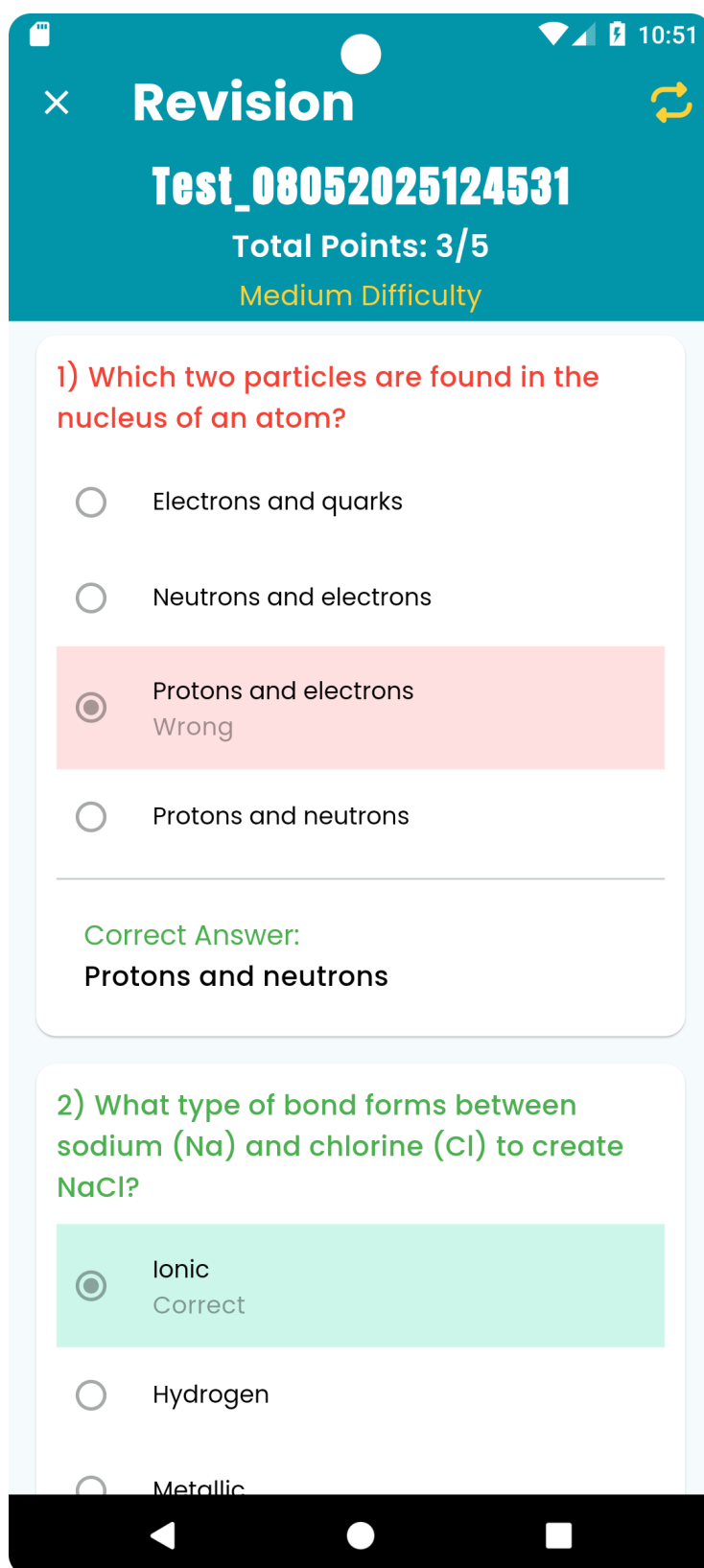


Figure 22: Test revision page

## Chapter Five: Implementation & Testing

### Pseudocode for Major Modules for Java SpringBoot Backend

#### *Generate and Save Course*

1. Generate a prompt using:
  - Course name
  - Course level
  - via `PromptGenerator.coursePrompt(...)`
2. Call the AI service (`chatClient`):
  - Send the prompt as a user input
  - Map the AI's response to `CourseAIResponse`
3. Print the AI's response for logging/debugging
4. Extract topics from the AI response:
  - For each topic string in the list:
    - Create a Topic object with:
      - ID = (index + 1)
      - Name = topic string
  - Collect these into a list
5. Retrieve the authenticated user from the security context
6. Build a new Course object with:
  - Title, category, and description from AI response
  - Level from request
  - Topics list
  - Authenticated user
7. Save the Course to the repository and return it.

#### *Update and Save Course*

BEGIN

SET user TO the authenticated user from security context

IF repository does not contain a course with ID = request.id AND user = user  
THEN

THROW "Course was not found" error with HTTP status 404

END IF

SET course TO new Course with:

id = request.id

title = request.title

description = request.description

level = request.level

topics = request.topics

user = user

RETURN repository.saveAndFlush(course)

END

### ***Generate and Save Test***

1. Fetch Course by ID:
  - If not found → throw 404 NOT\_FOUND ("Course not found")
2. Extract topics from the request:
  - If empty → throw 400 BAD\_REQUEST ("No topics were selected")
3. Initialize:
  - An empty list to store valid questions
  - A count variable (starting at 0)
4. While the count is less than (requested number of questions / 5):
  - a. Generate an AI prompt using:
    - Course title
    - Difficulty level
    - Comma-separated list of topics
    - Question index offset ((5 \* count) + 1)
  - b. Call the AI (via chatClient) and map the response to `TestAIResponse`
  - c. If response is non-null and has questions:
    - Filter out:
      - Questions with exactly 4 options
      - Questions where the answer is included in the options
    - Add valid questions to the list
    - Increment count
  - d. If any exception occurs during AI call:
    - Print the error
    - Throw 500 INTERNAL\_SERVER\_ERROR ("Failed to generate test!")
5. If after all attempts the question list is still empty:
  - Throw 500 INTERNAL\_SERVER\_ERROR ("No valid questions were generated!")
6. Retrieve the authenticated user from the security context
7. Save and return the test using `testRepository.save(...)`:
  - Set title, difficulty, createdAt, course, list of questions, and user

### ***Generate Test from Existing Questions***

1. Retrieve the currently authenticated user from the security context.
2. Get the course by course ID from the request.
  - If not found, throw a "Not Found" error.
3. Retrieve all tests associated with the course.
  - If there are no tests, throw a "Not Found" error.
4. Extract all questions from the course's tests.
5. Extract unique question stems from all questions to avoid duplicates.
6. IF the number of unique stems is less than or equal to the number of questions requested:
  - a. FOR each unique stem:
    - i. Find the first matching question with that stem.
    - ii. Add the question to a temporary list.
  - b. Shuffle the temporary list randomly.

- c. Add the first `n` questions to the `questions` list, creating new Question objects with proper indices.
- 7. ELSE (more unique stems than requested):
  - a. Use a greedy random selection strategy:
    - i. FOR each test:
      - IF desired number of questions is reached, STOP.
      - IF the test has no questions, SKIP.
      - Select a random question.
      - IF the question's stem is not already in the selected list, ADD it.
    - b. WHILE number of selected questions is still less than requested:
      - i. Randomly select a question from the entire pool.
      - ii. IF its stem is not in the selected list, ADD it.
- 8. IF the final number of selected questions is not equal to the number requested:
  - THROW a "Bad Request" error due to insufficient unique questions.
- 9. Save and return a new Test object with:
  - Title and difficulty from request
  - Current timestamp
  - Associated course
  - Selected list of questions
  - Authenticated user

### ***Submit and Evaluate Test***

1. Retrieve the test using the test ID from the request.
  - If not found, throw a "Not Found" error.
2. Calculate:
  - totalQuestions = number of questions in the test
  - totalAnswered = number of responses submitted
  - totalUnanswered = totalQuestions - totalAnswered
3. Initialize totalCorrectAnswers to 0.
4. Create a map (answerMap) from question index to correct answer using test questions.
5. FOR each response in the request:
  - a. Get the correct answer from answerMap using the response ID.
  - b. IF the response answer matches the correct answer:
    - Increment totalCorrectAnswers
6. Map each submitted response into an Answer object to form the givenAnswers list.
7. Retrieve the currently authenticated user.
8. Calculate the score as:
  - $(\text{totalCorrectAnswers} / \text{totalQuestions}) * 100$
  - Round to 2 decimal places (HALF\_UP rounding)
9. Build a new TestResults object with:
  - Title ("Test Results on " + test title)
  - Reference to the test
  - Total questions, answered, unanswered
  - Total correct answers
  - Calculated score

- Current date
- List of given answers
- Authenticated user

10. Save the TestResults object in the repository and return it.

### ***Test Revision***

1. Retrieve the currently authenticated user from the security context.
2. Find the test results by:
  - Result ID
  - Authenticated user
  - If not found, throw a "Not Found" error for test results.
3. Retrieve the test associated with the test result using the test ID.
  - If not found, throw a "Not Found" error for the test.
4. Create and return a new ReviseTest object using:
  - The retrieved test
  - The list of given answers from the test results
  - The number of correct answers from the test results
  - The total number of questions from the test results



## Pseudocode for Major Modules for Flutter Frontend

### *Generate Course*

1. Define a stateful widget named `CreateCoursePage`.
2. Inside its state (`CreateCourseController`):
  - Initialize a global form key.
  - Create a form validator provider instance.
  - Define two text controllers for course name and level.
  - Declare a `Level` object to hold the selected course level.
3. Define ``onSelect(value)`` function:
  - Set the selected `Level` by matching its name with the provided value.
4. Define ``create()`` function:
  - a. IF the form is valid:
    - i. Show a confirmation dialog asking the user if they want to build the course.
    - ii. IF the user confirms:
      - Prepare a payload containing:
        - Course name (trimmed)
        - Course level type
      - Send the payload to the `CourseAPIService` to create the course.
5. In ``build()``, return the `CreateCourseView` passing in this controller.

### *Generate Test*

1. Define a stateful widget named `GenerateTestPage`.
  - Accepts an optional `Course` object as a parameter.
2. Inside the state class (`GenerateTestController`):
  - Initialize state variables:
    - Future for fetching courses.
    - Form key and validator provider.
    - Text controller for test name.
    - Strings for question count and difficulty.
    - Selected Course and list of selected topics.
    - Boolean to determine if generating from past papers.
3. In ``initState()``:
  - If a course was passed in, use it and initialize its topics.
  - Load all available courses via `CourseAPIService`.
4. Define ``refresh()``:
  - Reload the list of courses and update the state.
5. Define ``onSelectCourse(value, list)``:
  - Find the course in the list that matches the selected value.
  - Set it as the selected course and extract its topics.
6. Define ``onSelectCount(value)`` and ``onSelectDifficulty(value)``:
  - Update question count and difficulty respectively.
7. Define ``onSelectTopics()``:
  - Open a dialog to let user select topics from the current course.
  - Save the selected topics.
8. Define ``toggle(value)``:

- Set the ``isPastPapers`` flag.
9. Define ``generate()``:
    - a. Validate the form.
    - b. If test name is empty, auto-generate one using the current datetime.
    - c. Show confirmation dialog to generate test.
    - d. If confirmed:
      - Build payload with courseId, testName, number of questions, difficulty, and topics.
      - Call TestAPIService to generate the test.
      - Return to previous screen with the new test result.
  10. In ``build()``, return the GenerateTestView passing in this controller.

### ***Take and Submit Test***

1. Define a stateful widget named TakeTestPage:
  - Takes a required Test object as a parameter.
2. In the state class (TakeTestController):
  - Define a ``test`` variable to hold the test passed in.
  - Initialize a list ``answers`` to store selected answers.
3. In ``initState()``:
  - Assign the passed test to the local ``test`` variable.
4. Define ``onSelect(Answer answer)``:
  - If an answer with the same question ID exists in the list, remove it.
  - Add the new answer to the list.
  - Log the updated answer list (for debugging).
5. Define ``submit()``:
  - a. Show a confirmation dialog:
    - If all questions are answered, prompt normally.
    - If some are unanswered, ask if the user wants to continue anyway.
  - b. If confirmed:
    - Build a payload with the test ID and user responses.
    - Call TestAPIService.attempt() to submit answers.
    - Navigate to ViewTestResultsPage with the returned results.
6. Define ``onPopInvoked(canPop)``:
  - If user tries to exit the page (``canPop == false``):
    - Show a confirmation dialog.
    - If the user confirms, exit the test page.
7. In ``build()``, return the TakeTestView and pass the controller.

### ***Revise Test***

1. Create a Scaffold with:
  - AppBar titled "Revision"
  - An IconButton in the AppBar for "Retake" action (calls state.retake)
2. Body is wrapped in ``RefreshableBody``:
  - Enables pull-to-refresh using ``state.refresh()``
3. Use ``FutureBuilder`` with ``state.future``:
  - a. If loading → show ``Loading()`` widget
  - b. If error → show ``ErrorInfo(snapshot.error)``
  - c. If data is loaded → proceed to display the revision

4. Once revision is loaded:
- Store the loaded data into ``state.revision``
  - Display a ``CustomScrollView`` with two main sections:
- === SliverAppBar ===
- Expanded header with:
    - Test title
    - Score
    - Difficulty level
- === SliverToBoxAdapter ===
- Column layout with:
    - A list of questions using ``ListView.builder``
      - For each question:
        - Find the matching user response from ``revision.responses``
        - Display a ``ReviseTestTile`` with the question and user's response
    - A gap
    - A "Done" button that pops the screen (calls ``Get.back()``)

## System Communication – JSON Payloads

Table 8: JSON payloads

Feature	Request	Response
Authentication	<pre>{   email: string,   password: string }</pre>	<pre>{   message: string,   data: {     token: string,     user: {       id: number,       fullName: string,       email: string,       role: string,       username: string,       enabled: boolean,       accountNonExpired: boolean,       accountNonLocked: boolean,       credentialsNonExpired: boolean     }   } }</pre>
Register	<pre>{   firstname: string,   lastname: string,   email: string,   password: string,   role: enum }</pre>	<pre>{   message: string,   data: {     token: string,     user: {       id: number,       fullName: string,       email: string,       role: string,       username: string,       enabled: boolean,       accountNonExpired: boolean,       accountNonLocked: boolean,       credentialsNonExpired: boolean     }   } }</pre>
Course Creation	<pre>{   courseName: string,   level: enum }</pre>	<pre>{   message: string,   data: {     id: number,     title: string,     category: string,     level: enum,     description: string,     topics: [       {         index: number,         name: string       },       ...     ]   } }</pre>

Test Generation	<pre>{   courseId: number,   testName: string,   noOfQuestions: number,   difficulty: string,   topics: [     {       index: number,       name: string     }   ] }</pre>	<pre>{   message: string,   data: {     id: number,     title: string,     difficulty: string,     createdAt: string,     questions: [       {         index: number,         stem: string,         options: [string],         answer: string       },       ...     ]   } }</pre>
Submission and Feedback	<pre>{   testId: number,   responses: [     {       id: number,       answer: string     }   ] }</pre>	<pre>{   message: string,   data: {     id: number,     title: string,     totalQuestions: number,     totalAnswered: number,     totalUnanswered: number,     totalCorrectAnswers: number,     score: number,     date: string,     testId: number   } }</pre>
Performance	empty	<pre>{   message: string,   data: [     {       id: number,       title: string,       totalQuestions: number,       totalAnswered: number,       totalUnanswered: number,       totalCorrectAnswers: number,       score: number,       date: string,       testId: number     },     ...   ] }</pre>

## Software Testing:

### Unit Testing

Table 9: Unit testing

Category	Component Tested	Test Description	Expected Outcome
<b>Course Generation</b>	Course Generator	Ensure the system uses the course name and academic level to generate a structured outline	Returned course includes relevant topics and is structured according to academic level
<b>Test Generation</b>	Prompt Parser	Ensure the input topic/subject is correctly parsed before sending to AI	Cleaned and structured prompt is generated
	AI Integration Service	Validate API or local LLaMA 2 model returns a list of questions based on input	Questions are relevant to topic and correctly formatted
	Question Formatter	Confirm questions have one correct answer and three valid distractors	All generated questions follow valid MCQ structure
<b>Test Submission &amp; Feedback</b>	Answer Validator	Check that submitted answers are correctly matched against correct options	Only correct answers are marked as such
	Score Calculator	Validate score calculation for combinations of correct/wrong/blank answers	Final score is accurately calculated
	Feedback Generator	Ensure feedback is generated based on score thresholds	Appropriate message displayed (e.g., “Excellent”, “Needs Review”)
<b>Performance Tracking</b>	Result Recorder	Check that every test submission is stored with user ID, score, and timestamp	Results are saved with accurate user and test associations
	Progress Analyzer	Verify calculation of performance trends across multiple tests	Average score and performance trend are correctly computed
	Chart Generator	Ensure data is formatted correctly for score graphs	Graphs correctly display historical scores
<b>Test Revisions</b>	Test Retrieval Handler	Ensure saved tests can be fetched by the original user	Test content and answers are restored accurately
	Retake Function	Verify user can initiate a retake and questions are reloaded	Retake test is correctly reset using original test content
	Improvement Tracker	Check that retake scores are compared with previous attempts	System indicates whether user's score has improved or declined

## Module Testing

Table 10: Module testing

Module Tested	Test Description	Expected Outcome
Test Creation Module	Ensure tests are linked to correct user and subject; valid question count	Test is created with correct metadata and associated questions
	Validate that only eligible users (e.g., authenticated students) can generate tests	Unauthorized users are denied access; tests only created by valid users
	Check test difficulty is respected in generated questions	Question set matches selected difficulty level (e.g., easy, medium, hard)
	Ensure duplicate test titles are handled or renamed	System prompts for unique titles or auto-renames with suffix
Feedback Module	Validate that feedback is generated and displayed based on the submitted score	Relevant, accurate feedback shown immediately after test submission
	Check that feedback messages are consistent with performance thresholds	Score-based categories (Excellent, Good, Needs Improvement) map to defined thresholds
	Ensure edge cases (e.g., score = 0 or 100) are handled properly	Correct feedback provided even at extremes
Performance Tracking Module	Verify performance data collection and chart generation	User sees correct performance trend over multiple test attempts
	Ensure tests are chronologically ordered in analytics display	Score chart and summaries follow correct timeline
	Validate average score, highest, lowest, and improvement rate calculations	All stats are computed accurately and update with each new test
User Authentication Module	Test registration with valid/invalid inputs (e.g., email format, password strength)	Valid users registered; invalid inputs show proper error messages
	Ensure secure login with token validation	Only users with correct credentials and valid tokens can access their dashboard
	Check role-based access control (student vs admin features)	Students and admins see only permitted screens/functions
Test Retake Module	Verify that users can only retake their own tests	Unauthorized retakes blocked; only original users may access retake option
	Check retake uses original questions without changes	Retake shows same questions in same order
	Validate that multiple attempts are tracked with correct timestamps and scores	History reflects every attempt clearly
Course Management Module	Ensure courses are created with correct level, title, and category	Valid course entries saved; duplicates avoided or flagged
	Verify only admin users can add or edit courses	Course creation/edit features restricted to admin role
	Test linking of topics to correct course and index order	Course topics ordered and displayed correctly in the app

## Integration Testing

Table 11: Integration testing

Integrated Features	Test Description	Expected Outcome
AI + Test Module	Ensure generated questions from AI are received and structured into the test format	Test contains AI-generated questions relevant to user input, correctly formatted
	Validate fallback mechanism if AI fails or gives an error	User sees helpful error and has option to retry or regenerate
Test Flow + Result Storage	Test completion should save results accurately to the database	Results saved with correct score, date, and question/answer history linked to user
	Test metadata (title, difficulty, topic) is correctly saved with results	Stored data matches selected options from user session
Authentication + Test Results + History	Test if logged-in users can view only their own test history and performance	Users only see their records; other users' results are not accessible
	Verify history displays test title, date, score, and retake option	Complete and accurate history available on dashboard
Performance Tracker + Chart Generator	Ensure performance data is updated after each test submission	Charts reflect up-to-date trends with new results included
	Data from multiple modules (tests, results, retakes) is merged into performance view	Dashboard shows clear and accurate visual progress indicators
Feedback + Score Calculator	Confirm feedback module responds correctly to score calculated from submitted test	Feedback messages correspond to defined score ranges (e.g., Excellent, Average)
	Ensure feedback updates on retakes	Feedback changes based on new scores
User Role + Access Control	Validate that only admins can create/edit courses, and students can only take tests	Role-based permissions enforced; unauthorized actions blocked with clear messages
Test Retrieval + Retake Function	Ensure old tests load correctly with saved answers	Test review feature displays original questions and student responses
	When retaking, new results are saved alongside old ones and compared	Retake data doesn't overwrite old data; improvement tracked



## System Testing

Table 12: System testing

Feature Tested	Test Description	Expected Outcome
Student Test Flow	Student registers, selects topic, takes test, gets feedback	Smooth end-to-end experience; accurate scoring and timely, relevant feedback
	Validate if test history is saved and viewable after submission	Test results persist in user dashboard with correct timestamps and scores
	Student retakes test from history	Retake initialized correctly with previous questions; updated score recorded
Admin Management	Admin creates/updates disciplines and manages question database	Admin privileges validated; new/edited data immediately available to users
	Check admin dashboard loads all courses and test content reliably	Admin sees full control panel and can manage users, courses, and questions
	Admin can delete or disable outdated content	Disabled items do not appear for student selection
AI Test Generation	AI returns questions based on difficulty and topic	Returned questions align with selected topic and difficulty (easy, medium, hard)
	Validate different levels (primary, secondary, tertiary) yield suitable content	Level-specific terminology and complexity matched appropriately
	Ensure fallback behavior if AI fails or gives empty response	User sees friendly error and retry option
Authentication and Roles	Users can register, login, and access correct dashboard	Students and admins redirected to their respective panels
	Passwords are hashed and secure in storage	No plain-text passwords in DB; system uses hashing (e.g., bcrypt)
	Invalid login shows appropriate messages	Invalid credentials return a user-friendly error message
Performance Tracking	Students view trend charts and average scores	Charts load accurately; metrics calculated and updated after each test
	Data from multiple tests aggregate correctly	Graphs reflect accurate average, highest, and lowest scores
Mobile Compatibility	App works seamlessly on different devices and screen sizes	Consistent UI/UX across Android and iOS; no layout issues or crashes
	Offline mode triggers when internet is unavailable	Offline message shown; previously saved tests still viewable
Course & Topic Navigation	Students can browse and choose from a list of subjects and topics	Subjects grouped by academic level and ordered clearly; topics correctly linked
Notifications & Feedback	Users receive alerts after test completion	Feedback displayed immediately with motivational or corrective messages

Feature Tested	Test Description	Expected Outcome
Database Consistency	Every test attempt links correctly to user, questions, and results	No orphan records; referential integrity maintained across all tables

## Database Testing

Table 13: Database testing

Area Tested	Test Description	Expected Outcome
Schema Validation	Ensure all entity relationships (e.g., <code>users</code> → <code>tests</code> → <code>results</code> , <code>tests</code> → <code>questions</code> ) are correctly mapped and enforced via foreign keys	No orphaned records exist; referential integrity is maintained across all tables
	Validate many-to-many and one-to-many mappings (e.g., users can take many tests)	Join tables and constraints are correctly implemented
Data Integrity	Check for NULL values in required fields (e.g., <code>username</code> , <code>question text</code> )	All mandatory fields contain valid values
	Detect invalid foreign key references or duplicate entries where uniqueness is required	All references are valid; constraints (e.g., unique email) are enforced
CRUD Operations	Insert new records using Data Access Object (DAO) layer for each major entity (user, test, result)	Data is correctly saved and visible in the database
	Retrieve existing records and validate data matches expected values	Data is fetched accurately and completely
	Update existing records (e.g., user profile, question edits)	Changes are saved and reflected correctly
	Delete records and ensure cascading effects or restrictions apply as designed	Deletions execute correctly without breaking relationships or leaving orphaned data
Query Performance	Test execution speed of frequent queries (e.g., user test history retrieval)	Queries return results within acceptable response times (especially on large datasets)
Transaction Handling	Simulate a transaction involving multiple inserts/updates (e.g., test + questions + results)	All steps succeed together or rollback fully on failure
Backup and Recovery	Test export/import of database and recovery from backup	Database state is accurately restored with no data loss or corruption

## Acceptance Testing

Table 14: Acceptance testing

Test Focus	Test Description	Expected Outcome
Real User Testing (Students)	Users simulate real-world scenarios including entering a course/topic, generating tests, taking the test, and reviewing feedback	Users complete the full learning cycle without issues; system meets user expectations
Educator Review	Educators evaluate the AI-generated content for relevance, difficulty, and educational alignment	Teachers validate that questions are appropriate and aligned with academic standards
Core Functionality Validation	Ensure all key features (course generation, test creation, grading, feedback, performance tracking) meet the defined user stories	All critical features work correctly and are accepted by students and educators
Cross-Device Functionality	Users access the app across different devices (Android, iOS) and test consistency of experience	Uniform and responsive behavior across supported platforms
Ease of Use	First-time users interact with the app to evaluate usability and intuitiveness	Navigation is clear, and onboarding helps users understand features quickly
Content Relevance	AI-generated questions are assessed against curriculum standards and different academic levels	Questions match the appropriate learning objectives and academic level
Performance Under Load	Multiple users interact with the app simultaneously	The app handles concurrent use without crashes or delays
Retake and History Access	Users attempt to review and retake previous tests	Functionality for revisions and performance tracking operates as intended

## **Chapter Six: Conclusions & Recommendations**

### **Results and Summary**

The Practice Pal initiative was to design an AI-powered learning environment that enables students to author their practice exams, build custom course content, receive immediate feedback and grading, and track progress over the long term. All of these objectives were discussed and refined during the development process, which justified the effectiveness and capability of the platform in supporting autonomous learning. The outcome confirms that a learner-centric system facilitating academic achievement through the integration of mobile technology, artificial intelligence, and real-time analytics was efficiently utilized.

Perhaps the most striking feature of the system was that the user could input a course and subject, after which it generated a preformatted course plan along with examination questions for the specified subject inputted. This was facilitated by the LLaMA 2 AI model, which was linked via Ollama, to ensure that the content remained relevant and at the chosen academic level. In trials, this functionality always produced well-organized, educationally sound, and logically formatted outlines and multiple-choice questions. This automated content creation significantly reduces time and expertise needed to create quality learning materials, particularly in low-resource educational environments.

The iOS and Android mobile application developed with Flutter successfully ran on both iOS and Android smartphones and had an intuitive interface. In user testing, its responsiveness, layout stability, and usability were highlighted. There were no issues with usability during student registration, subject selection, test taking, and results viewing. Zimbabwe is a country where this mobile-first approach is highly effective, as students are learning increasingly on the move and penetration via mobile devices is far greater than desktop access.

On the server side, the system employed Spring Boot for managing authentication, API business logic, and secure communication with the AI model and database. This kind of setup ensured strong performance during every stage of the test-taking process, from real-time question creation to storing results. The server side also set user authentication and authorization policies for protecting user data. During performance testing, the backend was found to handle simultaneous requests in a consistent and robust way, something that is fundamental for scalability when the system becomes used by more users in the future.

Application of auto-grading as well as instant feedback mechanisms was a major feature of the application. Test-takers received immediate feedback, including the questions answered correctly or incorrectly, and a score for the performance. This feedback system was effective in informing the learners of their mistakes immediately and urging them to take the test again with an improved approach. Primary school maths and high school level computer science contexts demonstrated that users learned more quickly from their errors and engaged in cycles of self-refinement.

Additionally, the system provided review and retake features for tests, allowing students to revisit past tests, examine their answers, and retest so that they could memorize them. This helped encourage reflective learning, allowing students to view improvement over time and focus on areas requiring more improvement. Having past performance data in the system allowed students to assume control of their academic progress and thus obtain confidence through measurable improvement.

Initial user testing took place with a small group of teachers and students from diverse academic fields. The feedback obtained was overall positive, particularly with respect to the relevance, adaptability, and speed of AI-generated content, as well as the overall usability of the system. The result of the study strongly suggests platforms like Practice Pal to play a vital role in autonomous learning, especially in the education of Zimbabwe as well as such regions with limited access to customized educational content. The success of the site at the prototype level strongly foretells prospective development and mass deployment.

While Practice Pal was successfully created and welcomed, there were several limitations that were observed during testing and implementation. One of the main issues was the requirement for an uninterrupted internet connection, as AI-supported content generation and cloud-based management of data require active internet connectivity, thus limiting its usage in rural or low-bandwidth settings. Besides that, although the AI model of LLaMA 2 generated informative and well-written questions for most topics, it sometimes generated superficial or inaccurate material for highly specialized or abstract topics. A second shortcoming was also limited local curriculum integration, i.e., the system is maybe not fully compatible with specific national exam formats or learning objectives without further customization. Besides, the system doesn't currently support collaborative functionality or direct teacher monitoring, which limits its suitability in traditional classrooms. These limitations further underscore the need for continued refinement in terms of offline support, teacher moderation facilities, and more integral curriculum planning for enhancing accuracy and context specificity.

In short, Practice Pal adequately demonstrated the potential of AI-driven learning support to tailor and enhance students' education through dynamic test generation, auto-grading assessment, and performance tracking. Though the core features operated flawlessly and the platform received encouraging feedback from initial users, addressing the found limitations will be essential to broader utilization and long-term success. With further growth to increase curriculum applicability, offline availability, and instructor engagement, Practice Pal can potentially turn into an influential platform for scalable, accessible, and adaptive learning, particularly in under-resourced learning settings.

## **Recommendations**

The AI-generated content needs to be in compliance with regional standards or national educational curricula, like Zimbabwe's ZIMSEC curriculum, to maximise PracticePal's impact and utility. Currently, many educational apps utilize generic content that may not accurately reflect local learning outcomes or test formats. By integrating with official curricula, the system can generate questions and lesson plans directly applicable to what students learn in school. This would increase the likelihood of adoption by schools and students, ensuring students are better prepared for national exams. In contrast to existing platforms like Quizlet or Duolingo, which offer static or globally generalized content, this feature would offer a clear advantage in terms of both local relevance and pedagogical accuracy.

Adding a layer of teacher review, customization, and approval of AI-generated content will bring credibility and trustworthiness to the system, particularly in formal learning contexts. Teachers can ensure the content is pedagogically appropriate, age-sensitive, and aligns with learning objectives. This recommendation addresses concerns about unbridled AI application in education by reintroducing human oversight and customization that are essential to student safety and academic integrity. Enabling the participation of teachers also makes it easy to utilize the platform within classrooms, rendering it a blended learning platform rather than a simple self-paced learning application. Other sites, on the other hand, disconnect teachers from the content development process, and as a result, they are not quite handy in the formal learning settings.

The addition of a comprehensive analytics dashboard would enable both educators and students to monitor their progress over time, identify strengths and weaknesses, and make intelligent, data-driven decisions about future study. This type of functionality could include visualizations of trending scores, time spent on each subject, and distributions of question accuracy. Unlike other tools that mark answers as correct or incorrect, Practice Pal is in a position to give insight into the

progress of students, allowing for personalized recommendations for improvement. It would also enable teachers to monitor entire classes and intervene where needed, creating a feedback-driven learning space that leads to improved long-term outcomes.

The addition of a comprehensive analytics dashboard would enable students and teachers to monitor their progress over time, identify strengths and weaknesses, and make data-driven, strategic choices about future study. Such a feature could include graphs of trend lines for scores, study time per subject, and distributions of question correctness. Unlike other software that marks answers as correct or incorrect, Practice Pal can offer actionable insights into student progress, with personalized recommendations on how to progress. It would also enable teachers to monitor entire classes and intervene where required, giving a feedback-led learning experience leading to improved long-term outcomes.

While tech-savvy people may find using the app a breeze, the majority of students and educators may struggle to utilize new interfaces or features. A well-designed onboarding experience that includes step-by-step tutorials, tooltips, and supporting documentation would go a long way in improving the experience for first-time users. This reduces the abandonment rate and ensures users fully understand how to use the platform to their benefit. Compared to competitors, many of whom assume prior digital literacy, Practice Pal would be distinct in being more accessible to users who do not have a lot of technical background. It would also be easier on instructors who are trying to get the app integrated into their classrooms.

By implementing these recommendations, Practice Pal can become a more complete, solid, and helpful learning tool. It would complement existing apps while offering new, context-aware functionality founded upon students' actual learning needs.

## **Future Works**

Suggested approaches for transforming Practice Pal into a comprehensive learning platform. Improved accessibility, engagement, and flexibility will enable the system to accommodate a diverse student group.

### **Web-Based Version**

Including a completely responsive web-based version of the platform will make it much more accessible. While the app already accesses both Android and iOS users, having a web version will make Practice Pal usable on desktop and laptop machines, which are used all over libraries, schools, and homes. Cross-platform compatibility ensures that students with differing devices types can

utilize the tool without interruption, making it very usable in both official and unofficial learning environments.

### **Interactive Visual Analytics**

Implementing a visual analytics dashboard will empower students to take ownership of their learning. By clearly displaying progress graphs, weak areas, topic mastery, and improvement suggestions, learners can identify what they need to focus on to achieve their goals. Visual cues, such as charts and colour-coded feedback, make it easier for users to interpret data and plan their study sessions more effectively, promoting a more personalized and strategic approach to learning.

### **Natural Language Querying**

Incorporating natural language processing (NLP) capabilities will allow users to interact with Practice Pal more intuitively. Instead of selecting rigid options, students could type or speak goals such as "Help me revise Grade 10 algebra" or "Create a test to prepare for my biology exam." The AI will interpret these inputs and generate personalized learning content. This approach lowers the barrier to technology use, especially for younger students or those unfamiliar with structured digital interfaces.

### **Collaborative Learning Features**

Introducing group-based features, such as shared tests, group leaderboards, and peer-reviewed answers, will foster a community of collaborative learners. These tools foster peer-to-peer learning, discussions, and support, which are essential components of educational development. Collaborative learning also mirrors classroom environments, promoting teamwork skills and making the app a valuable tool for both personal study and social learning.

### **Gamification Enhancements**

Gamification tactics such as leaderboards, badges, levels, and point systems can enhance student motivation and engagement. These aspects make learning more dynamic and gratifying. Gamification appeals to the learner's intrinsic motivation by giving goals, challenges, and a sense of accomplishment, all of which have been shown to improve retention and lower dropout rates on e-learning platforms.

### **Accessibility Support**

Finally, to ensure inclusivity, accessibility features such as voice-to-text (for individuals with motor impairments) and text-to-speech (for visually impaired learners or those with reading difficulties) must be integrated. These features make the platform accessible to students with diverse learning



needs and disabilities, aligning Practice Pal with universal design principles and ensuring that no learner is left behind due to technical or physical barriers.

## References

- [1] S. Akgun and C. Greenhow, "Artificial intelligence in education: Addressing ethical challenges in K-12 settings," *AI Ethics*, vol. 2, pp. 431–440, 2022. [Online]. Available: <https://doi.org/10.1007/s43681-021-00096-7>
- [2] V. Aleven, E. A. McLaughlin, R. A. Glenn, and K. R. Koedinger, "Instruction based on adaptive learning technologies," in *The Cambridge Handbook of the Learning Sciences*, R. E. Mayer and P. A. Alexander, Eds., 2nd ed., Cambridge University Press, 2016.
- [3] L. Hough, "Tool in School: Quizlet," *Ed. Magazine*, Harvard Graduate School of Education, May 22, 2018. [Online]. Available: <https://www.gse.harvard.edu/ideas/ed-magazine/18/05/tool-school-quizlet>
- [4] H. E. Vidergor and P. Ben-Amram, "Khan Academy effectiveness: The case of math secondary students' perceptions," *Computers & Education*, vol. 157, Art. no. 103985, Nov. 2020. [Online]. Available: <https://www.sciencedirect.com/science/article/abs/pii/S0360131520301834>
- [5] N. Guerrero, "Good, free, fun: The simple formula that has made Duolingo a daily habit for millions," *BBC Worklife*, Oct. 4, 2024. [Online]. Available: <https://www.bbc.com/worklife/article/20241004-the-simple-formula-that-made-duolingo-a-daily-habit-for-millions>
- [6] M. A. Cardona, R. J. Rodríguez, and K. Ishmael, *Artificial Intelligence and the Future of Teaching and Learning*. U.S. Department of Education, Office of Educational Technology, May 2023. [Online]. Available: <https://www.ed.gov/sites/ed/files/documents/ai-report/ai-report.pdf>
- [7] G. Wiggins, "Seven keys to effective feedback," *ASCD Education Leadership*, May 2012. [Online]. Available: <https://www.ascd.org/el/articles/sevenkeys-to-effective-feedback>
- [8] Walton Family Foundation, "Teachers and students embrace ChatGPT for education," *Walton Family Foundation*, Mar. 1, 2023. [Online]. Available: <https://www.waltonfamilyfoundation.org/learning/teachers-and-students-embracechatgpt-for-education>
- [9] Z. Swiecki, H. Khosravi, G. Chen, R. Martinez-Maldonado, J. M. Lodge, S. Milligan, B. Selwyn, and D. Gašević, "Assessment in the age of artificial intelligence," *Computers and Education: Artificial Intelligence*, vol. 3, 2022. [Online]. Available: <https://doi.org/10.1016/j.caeai.2022.100075>
- [10] I. Roll, V. Aleven, B. M. McLaren, and K. R. Koedinger, "Improving students' help-seeking skills using metacognitive feedback in an intelligent tutoring system," *Learning and Instruction*, vol. 21, no. 2, pp. 267–280, 2011. [Online]. Available: <https://doi.org/10.1016/j.learninstruc.2010.07.004>
- [11] S. Merrill, "In schools, are we measuring what matters?" *Edutopia*, 2020. [Online]. Available: <https://www.edutopia.org/article/schools-are-we-measuring-what-matters>

[12] W. Holmes and K. Porayska-Pomsta, Eds., *The Ethics of Artificial Intelligence in Education*. Abingdon, UK: Routledge, 2022. ISBN: 978-0367349721.

# Bibliography

## Practice Pal – User Guide (Mobile App)

### 1. Introduction

Welcome to Practice Pal – an AI-powered mobile app designed to help students practice tests in any subject and receive instant feedback. Whether you're in primary school or university, Practice Pal offers personalized test generation, performance tracking, and insightful learning paths.

### 2. Getting Started

#### 2.1 Installation

- Download the Practice Pal app from the Google Play Store (iOS coming soon).
- Launch the app and grant permissions (e.g., storage access, if prompted).

#### 2.2 Account Registration

- Tap on **Sign Up**.
- Enter:
  - First name
  - Last name
  - Email
  - Password
  - Select role (Student or Educator)
- Tap **Register**. You'll be redirected to the login page.

#### 2.3 Login

- Enter your **email** and **password**.
- Tap **Sign In**.
- Upon success, your dashboard will be shown.

### 3. Dashboard Overview

The dashboard provides a quick overview of your recent activity:

- **Start a New Test**
- **View Courses**
- **Past Test Results**
- **Recommended Topics**
- **Settings & Profile**

#### **4. Creating a Course (Educators/Advanced Students)**

##### **Steps:**

1. Navigate to **Courses** tab.
2. Tap **Create Course**.
3. Fill in:
  - Course Title
  - Level (e.g., PRIMARY, HIGH\_SCHOOL, UNIVERSITY)
  - Description
  - List of Topics (with optional descriptions)
4. Tap **Save**.

#### **5. Generating a Test**

##### **Steps:**

1. Tap **Start New Test**.
2. Select a course from the list.
3. Fill in:
  - Test Name
  - Number of Questions
  - Difficulty Level (EASY, MEDIUM, HARD)
  - Choose Topics
4. Tap **Generate**. AI will create relevant questions.

5. Review the generated test and tap **Begin Test**.

## 6. Taking a Test

- Each question is multiple choice.
- Tap your selected answer and tap **Next**.
- Submit at the end. Your test will be graded instantly.

## 7. Viewing Results

- Go to the **Results** tab.
- Tap any completed test to view:
  - Score
  - Time taken
  - Correct/Incorrect answers
  - Suggested topics for improvement

## 8. Feedback & Recommendations

- After completing a test, AI provides:
  - Your weak areas
  - Recommended study topics
  - Follow-up practice tests

## 9. Managing Your Account

- Tap on **Profile** in the navigation menu.
- Update your name, password, or view your role.
- Logout securely from the same menu.

## 10. Educator Tools

- Educators can:
  - Review AI-generated questions
  - Create shared group tests

- Monitor student performance

## 11. Accessibility Features

- Voice-to-Text input for question answers.
- Text-to-Speech support for students with visual impairments.

## 12. Tips for Best Use

- Always double-check topic selections before generating a test.
- Use feedback to retake weaker topic tests.
- Save progress frequently if on a low-battery device.

## 13. Troubleshooting

Issue	Solution
App crashes on load	Restart device and ensure updates are installed.
Can't sign in	Ensure credentials are correct, or reset password.
AI not generating questions	Ensure a course and topic are properly selected.
AI not creating course	Ensure the course name is correct and the level is selected, then try again.
Network error or timeout	Check your internet connection. Switch networks or retry in an area with stronger connectivity.
Test results not displaying	Refresh the app or clear cache. If the issue persists, log out and log back in.
App freezing during test	Close other background apps and ensure the device has sufficient RAM. Restart the app if needed.
Cannot hear text-to-speech	Ensure your device volume is up and TTS settings are enabled in your device's accessibility menu.
Incorrect role assigned	Log out and register again using the correct role, or contact support for assistance.
AI feedback seems inaccurate	Ensure questions were answered honestly. Regenerate test based on AI's suggested topics.

## 15. Version Info

- App Version: 1.0.0
- Platform: Android
- Backend: Spring Boot 3
- AI Model: LLaMA 2 7B Chat (via Ollama)



