

University of Science and Technology of Hanoi

**Bachelor Thesis**

**ENGLISH EXAM PREPARATION WEBSITE**

**Student Name : Đào Quý Tùng**

**Student ID : 22BI13451**

**Major : ICT**

**Supervisor : Kiều Quốc Việt**

**Hanoi, VN, 2025**

To whom it may concern,

I, Kieu Quoc Viet certify that the thesis of Mr Dao Quy Tung is qualified to be presented in the Internship Jury 2024-2025.

Hanoi, 4 July 2025

**Supervisor’s signature**

# 

# 

# **Table of Content**

[Table of Content 3](#_Toc204650612)

[List of Figures 6](#_Toc204650613)

[List of Tables 7](#_Toc204650614)

[ABSTRACT 8](#_Toc204650615)

[ACKNOWLEDGEMENTS 9](#_Toc204650616)

[I. Introduction 10](#_Toc204650617)

[**1. Motivation and Research Background** 10](#_Toc204650618)

[**2. Objectives** 11](#_Toc204650619)

[**3. Thesis Structure** 12](#_Toc204650620)

[II. Requirements analysis 13](#_Toc204650621)

[**1. Functional requirements** 13](#_Toc204650622)

[**1.1. Student Features** 13](#_Toc204650623)

[**1.2. Teacher Features** 13](#_Toc204650624)

[**1.3. Administrative Features** 13](#_Toc204650625)

[**2. Non-functional requirements** 14](#_Toc204650626)

[**3. Use Case Diagram** 15](#_Toc204650627)

[**4. Functional Specification** 16](#_Toc204650628)

[**4.1. Register** 16](#_Toc204650629)

[**4.2. Login** 17](#_Toc204650630)

[**4.3. Take IELTS Test** 17](#_Toc204650631)

[**4.4. Practice Flashcards** 18](#_Toc204650632)

[**4.5. View Feedback and Grades** 19](#_Toc204650633)

[**4.6. Create Tests** 20](#_Toc204650634)

[**4.7. View Student Submissions** 20](#_Toc204650635)

[**4.8. Manage Flashcards** 21](#_Toc204650636)

[**4.9. Manage Users** 22](#_Toc204650637)

[**4.10. Manage Tests** 24](#_Toc204650638)

[III. Design 26](#_Toc204650639)

[**1. System Architecture** 26](#_Toc204650640)

[**2. Sequence Diagram** 27](#_Toc204650641)

[**2.1. Login & Authentication** 27](#_Toc204650642)

[**2.2. Take Test & Submit Answers** 29](#_Toc204650643)

[**2.3. Practice Flashcard** 30](#_Toc204650644)

[**2.4. Create new Test** 31](#_Toc204650645)

[**2.5. Grading System** 33](#_Toc204650646)

[**2.6. Manage System for admin** 35](#_Toc204650647)

[**2.6.1. Manage Tests** 35](#_Toc204650648)

[**2.6.2. Manage Users** 36](#_Toc204650649)

[**2.6.3. Manage Flashcards** 38](#_Toc204650650)

[**3. Database Design** 39](#_Toc204650651)

[**3.1. User Management** 40](#_Toc204650652)

[**3.2. Test and Question Structure** 40](#_Toc204650653)

[**3.3. Submissions and Grading** 40](#_Toc204650654)

[**3.4. Progress and Performance Tracking** 40](#_Toc204650655)

[**3.5. Flashcard Learning System** 41](#_Toc204650656)

[**3.6. Evaluation and Supporting Data** 41](#_Toc204650657)

[IV. Tools And Technologies 42](#_Toc204650658)

[**1. Backend Technology: Spring Boot and Java** 42](#_Toc204650659)

[**2. Database Technology: MySQL with JPA and Hibernate** 42](#_Toc204650660)

[**3. Frontend Technology: ReactJS and JavaScript** 42](#_Toc204650661)

[**4. Security and Authentication: JWT and Spring Security** 43](#_Toc204650662)

[**5. Audio Handling: MediaRecorder and Base64 Encoding** 43](#_Toc204650663)

[**6. Vocabulary Enrichment: Dictionary and TTS APIs** 43](#_Toc204650664)

[V. Implementation 44](#_Toc204650665)

[**1. Database connection** 44](#_Toc204650666)

[**2. Integration and System Flow** 45](#_Toc204650667)

[**3. Backend Implementation** 45](#_Toc204650668)

[**3.1. Technologies and Configuration** 46](#_Toc204650669)

[**3.2. Package Structure and Layered Design** 46](#_Toc204650670)

[**3.3. Authentication and Role-Based Authorization** 47](#_Toc204650671)

[**3.4. Audio Handling (Upload and Retrieval)** 47](#_Toc204650672)

[**3.5. Test Module** 47](#_Toc204650673)

[**3.6. Submission Module** 47](#_Toc204650674)

[**3.7. Flashcard Management** 48](#_Toc204650675)

[**3.8. User Management** 48](#_Toc204650676)

[**3.9. Repository & Database Queries** 48](#_Toc204650677)

[**4. Frontend Implementation** 48](#_Toc204650678)

[**4.1. Technologies and Project Setup** 48](#_Toc204650679)

[**4.2. Session and HTTP Request Management** 49](#_Toc204650680)

[**4.3. Audio Features** 49](#_Toc204650681)

[**4.4. Key Components and Pages** 49](#_Toc204650682)

[**4.5. UI/UX Handling** 49](#_Toc204650683)

[VI. Result and Testing 49](#_Toc204650684)

[**1. Discussion** 50](#_Toc204650685)

[**2. Testing** 51](#_Toc204650686)

[**2.1. Authentication & JWT Handling** 51](#_Toc204650687)

[**2.2. Role-Based Access Control** 52](#_Toc204650688)

[**2.3. Test Creation & Management** 52](#_Toc204650689)

[**2.4. Test Submission & Auto-Grading** 53](#_Toc204650690)

[**2.5. Manual Teacher Grading** 53](#_Toc204650691)

[**2.6. Audio Upload & Playback** 54](#_Toc204650692)

[**2.7. Flashcard Learning** 54](#_Toc204650693)

[VII. Conclusion and Future Works 55](#_Toc204650694)

[**1. Conclusion** 55](#_Toc204650695)

[**2. Future Development** 55](#_Toc204650696)

[VIII. References 57](#_Toc204650697)

[APPENDICES 58](#_Toc204650698)

# **List o****f Figures**

[Figure 1: Use Case Diagram 15](#_Toc204651815)

[Figure 2: System Architecture 26](#_Toc204651816)

[Figure 3: Login & Authentication Sequence Diagram 27](#_Toc204651817)

[Figure 4: Take Test & Submit Answers Sequence Diagram 29](#_Toc204651818)

[Figure 6: Create New Test Sequence Diagram 31](#_Toc204651819)

[Figure 7: Grading System Sequence Diagram 33](#_Toc204651820)

[Figure 8: Manage Tests Sequence Diagram 34](#_Toc204651821)

[Figure 9: Manage Users Sequence Diagram 35](#_Toc204651822)

[Figure 10: Manage Flashcards Sequence Diagram 37](#_Toc204651823)

[Figure 11: Entity Relational Diagram 38](#_Toc204651824)

[Figure 12: Home Page 57](#_Toc204651825)

[Figure 13: Taking Test Screen 57](#_Toc204651826)

[Figure 14: Flashcard Home Screen 58](#_Toc204651827)

[Figure 15: Practice Flash Card Screen 58](#_Toc204651828)

[Figure 17: Answer’s Details Screen 59](#_Toc204651829)

[Figure 18: User’s Profile 60](#_Toc204651830)

[Figure 19: Admin Dashboard 60](#_Toc204651831)

# **List of Tables**

[Table 1: Register Functional Specification Table 17](#_Toc204650716)

[Table 2: Login Functional Specification Table 17](#_Toc204650717)

[Table 3: Take IELTS Test Functional Specification Table 18](#_Toc204650718)

[Table 4: Practice Flashcards Functional Specification Table 19](#_Toc204650719)

[Table 5: View Feedback and Grades Functional Specification Table 19](#_Toc204650720)

[Table 6: Create Tests Functional Specification Table 20](#_Toc204650721)

[Table 7: View Student Submissions Functional Specification Table 21](#_Toc204650722)

[Table 8: Manage Flashcards Functional Specification Table 22](#_Toc204650723)

[Table 9: Manage Users Functional Specification Table 24](#_Toc204650724)

[Table 10: Manage Tests Functional Specification Table 25](#_Toc204650725)

[Table 11: Authentication & JWT Handling Testing Table 52](#_Toc204650726)

[Table 12: Role-Based Access Control Testing Table 52](#_Toc204650727)

[Table 13: Test Creation & Management Testing Table 52](#_Toc204650728)

[Table 14: Test Submission & Auto-Grading Testing Table 53](#_Toc204650729)

[Table 15: Manual Teacher Grading Testing Table 54](#_Toc204650730)

[Table 16: Audio Upload & Playback Testing Table 54](#_Toc204650731)

[Table 17: Flashcard Learning Testing Table 54](#_Toc204650732)

# 

# **ABSTRACT**

This project is about creating an online exam system to help teachers and students manage exams more easily. The system includes two main parts: a simple, user-friendly frontend built with React.js, and a powerful backend developed using Java and Spring Boot. Teachers can quickly create tests, add different kinds of questions, manage submissions, and grade students’ answers. Students can easily take tests, submit their responses, and review their grades and feedback.

The backend handles user accounts securely using JWT tokens, efficiently stores exam data, and supports various question types—including multiple-choice, essays, listening exercises, and speaking tasks. It also provides both automatic grading for objective questions and manual grading options for subjective or open-ended questions.

# **ACKNOWLEDGEMENTS**

First and foremost, I would like to express my heartfelt thanks to my supervisor, Mr. Kiều Quốc Việt, for his guidance, support and encouragement throughout this project. I truly appreciate the time and effort he spent helping me stay on track, offering insightful suggestions, and pushing me to improve the quality of my work. It has been a great opportunity and privilege to learn under his supervision.

To my friends, thank you for always being there — for studying together, motivating each other, and making these years truly memorable. And most importantly, to my family: thank you for your unconditional love, understanding, and support. Your belief in me has always been my greatest motivation.

This project would not have been possible without all of you.

# 

# 

# **I. Introduction**

## **1. Motivation and Research Background**

Over the past decade, online learning has rapidly expanded, becoming an essential method for language learners worldwide. Among various English proficiency tests, the International English Language Testing System (IELTS) remains one of the most important and widely accepted certifications for students and professionals aiming to study, work, or migrate abroad. In 2023, over 3 million individuals took the IELTS exam globally, highlighting its significance in academic and professional contexts.[5]

In Vietnam, the interest in IELTS has been increasing rapidly. The percentage of test-takers aged 16–18 rose from just 1.5% in 2018 to 30% in 2023, indicating that younger students are starting to prepare for IELTS earlier.[4]

Despite this growing interest, Vietnamese candidates still face challenges in certain skills. According to recent statistics, the average IELTS scores for Vietnamese test-takers in 2023–2024 were: Listening 6.3, Reading 6.4, Writing 6.0, and Speaking 5.7. These scores are slightly below the global averages, especially in Speaking and Writing.[1][2][3]

Several factors contribute to these challenges:

* Limited Speaking Practice: Many learners lack opportunities to practice speaking with proficient English speakers, making it difficult to receive feedback and improve.
* Writing Difficulties: Vietnamese students often struggle with IELTS Writing Task 2 due to differences in essay structures and expectations compared to their prior educational experiences. Issues include limited lexical resources, difficulties in organizing ideas, and challenges in meeting time constraints.

While various online platforms offer IELTS preparation resources, they often fall short in addressing these specific needs:

* Generic Content: Many platforms provide standardized materials that do not cater to the unique challenges faced by Vietnamese learners.
* Lack of Personalized Feedback: Automated systems may not effectively identify and address individual weaknesses, particularly in Speaking and Writing.
* Technical and Psychological Barriers: Students may encounter technical issues, lack digital literacy, or feel anxious when receiving feedback in online settings.

Given these challenges, we recognized the need for a more practical and inclusive system. Our primary goal was to build a web platform that is intuitive for both students and teachers: one that simplifies test creation, enables dynamic interaction, and supports diverse question types—from multiple-choice to audio-based speaking tasks. We aimed to bridge the gap by incorporating auto-grading for objective parts and manual scoring for subjective ones, all within a centralized dashboard.

In addition, we wanted to enhance the learning experience with a built-in flashcard system, allowing students to expand their vocabulary through spaced repetition—something that many current IELTS platforms either overlook or only offer in isolated modules. Security and fairness were also a top priority, leading us to implement role-based access control and token-based authentication.

By addressing the practical needs of IELTS learners and educators, our project aims to deliver a more complete, flexible, and user-friendly solution that overcomes the common shortcomings of today’s platforms. Through this initiative, we hope to contribute to the ongoing evolution of digital language learning tools.

## **2. Objectives**

The main goal of this project was to build an online IELTS practice platform designed to help students effectively prepare for the exam through interactive practice and structured feedback. Right from the start, our goal was not just to deliver content but to create a full learning environment that covers Listening, Reading, Writing, and Speaking skills.

Specifically, our objectives included:

* Simulating realistic IELTS test conditions, providing users with interactive and timed practice sessions for all four IELTS sections.
* Supporting both automatic and manual grading, offering automatic assessment for Listening and Reading sections, while enabling teachers to manually grade Writing and Speaking tasks.
* Integrating vocabulary learning tools, using flashcards combined with spaced repetition techniques and enriched with dictionary-based data (audio, IPA, and definitions).
* Providing role-based interaction, clearly defining separate functions and interfaces for students, teachers, and administrators, using secure authentication methods.
* Implementing basic web development best practices, such as building a RESTful API backend using Java Spring Boot and creating a responsive frontend using ReactJS.
* Ensuring secure data storage and management, using MySQL as the database system to handle users, tests, flashcards, and submissions effectively.
* Designing for usability and accessibility, making sure the interface remains user-friendly and works well on desktop browsers, even for non-technical users.

## **3. Thesis Structure**

This thesis is divided into five main chapters. The first chapter introduces the background and motivation for the project, along with the main goals of building an online IELTS preparation platform. Chapter 2 focuses on the system’s design, including feature requirements, system diagrams, and database structure. Chapter 3 explains the implementation in more detail, covering both backend and frontend development using Spring Boot, ReactJS, and MySQL. It also describes how core features like JWT authentication, test creation, audio handling, and flashcard learning were built. Chapter 4 presents the testing process and analysis, showing how we manually tested important features, recorded the results, and discussed any failed cases. Finally, Chapter 5 summarizes what has been achieved and suggests improvements for future development.

# 

# **II. Requirements analysis**

## **1. Functional requirements**

### **1.1. Student Features**

While practicing IELTS online, we noticed students often struggle to find realistic practice tests and have difficulty getting quick and clear feedback, especially for the Speaking and Writing sections. Therefore, we developed the following features to address these issues:

* Registration and Login: Each student has their own account to manage their learning individually. This helps students easily track their progress and keeps their personal information secure.
* Taking IELTS Practice Tests: Students can practice IELTS tests with timing similar to real exams. After completing the tests, they receive immediate scores for the Listening and Reading sections. For Writing and Speaking, teachers manually grade the tests and provide detailed feedback afterward.
* Viewing Scores and Feedback: Students can easily view their scores along with comments from teachers, clearly identifying areas they need to improve.
* Vocabulary Practice with Flashcards: I created an easy-to-use flashcard feature to help students effectively and quickly improve their vocabulary.

### **1.2. Teacher Features**

We realized that teachers typically spend a lot of time creating tests, managing exam materials, and especially grading Writing and Speaking assignments. To assist teachers, we implemented several supportive features:

* Test Creation and Management: Teachers can quickly and conveniently create new IELTS tests, update existing ones, or remove old tests. Audio files can also be directly uploaded to the system, making test creation easy.
* Grading and Evaluation: The system simplifies the process of reviewing and grading student submissions. Teachers can quickly assign scores and provide detailed feedback.

### **1.3. Administrative Features**

Administrators play a crucial role in managing and ensuring the system's smooth operation. To support administrators, we built these features:

* User Management: Admins can easily add, edit, or delete users, as well as adjust their roles and permissions within the system.
* Content Management: Admins have user-friendly tools to manage tests and flashcards, keeping the content up-to-date and relevant.
* System Security and Monitoring: The system logs important activities, allowing administrators to easily detect and address any irregularities, maintaining the system's security and stability.

## **2. Non-functional requirements**

Besides core functionalities, the system needs to satisfy several non-functional requirements to ensure it works properly in real use.

First, it must offer good performance, especially during test-taking. Pages should load quickly, and submissions must be processed without delays, even when many users are online at the same time. The system should also be easy to use. Both students and teachers must be able to navigate the interface without confusion. Key actions like starting a test, reviewing results, or grading submissions should be straightforward.

Security is another critical aspect. The platform uses JWT-based authentication and role-based access control to protect user data and ensure that users only access functions they are allowed to. In terms of reliability, the system should be stable during exam sessions and prevent data loss. Features like auto-save and proper error handling help make the platform more robust.

Lastly, the system should be easy to maintain and expand. Thanks to the clean structure of the backend (with clearly separated layers), new features or updates can be added without affecting existing parts.

## **3. Use Case Diagram**



Figure 1: Use Case Diagram

Figure 1 illustrates the main actors and the actions they can perform within the IELTS Practice System. There are three user roles: Student, Teacher, and Administrator, each interacting with the system in different ways based on their responsibilities.

* Students use the system primarily to practice and improve their English skills. They can register or log in to their account, take IELTS-style tests (which include Listening, Reading, Writing, and Speaking), and study vocabulary using flashcards. After completing tests, they can check their scores and review any feedback provided by teachers.
* Teachers are responsible for creating tests and evaluating student submissions. Since the Speaking and Writing parts require manual grading, teachers can listen to recordings or read essays, give scores, and provide comments.
* Administrators have full control over system management. They manage user accounts (both students and teachers), oversee the test and flashcard databases, and can access system logs to monitor activity. They are also in charge of maintaining security and making sure sensitive data stays protected.

## **4. Functional Specification**

### **4.1. Register**

|  |  |  |
| --- | --- | --- |
| **Use Case** | | **Register** |
| Actor | | Student |
| Precondition | | User is not yet registered |
| Main Flow | | |
| 1 | User | Opens the “Register” screen |
| 2 | User | Enters email, password, and confirmation password |
| 3 | System | Validates input (email syntax, password strength, match) |
| 4 | System | Checks that the email is not already in use |
| 5 | System | Creates a new user record with the “STUDENT” role |
| 6 | System | Issues a JWT and returns it in the response header |
| 7 | System | Redirects user to the login screen |
| Exceptions | | |
| 3a | System | “Password too weak” → show error |
| 3b | System | “Passwords do not match” → show error |
| 4a | System | “Email already in use” → show error |
| Postcondition | | A new user account exists and the user is authenticated |

Table 1: Register Functional Specification Table

### **4.2. Login**

|  |  |  |
| --- | --- | --- |
| **Use Case** | | **Register** |
| Actor | | Student, Teacher, Administrator |
| Precondition | | User already has an account and is not currently logged in |
| Main Flow | | |
| 1 | User | Opens the “Login” screen |
| 2 | User | Enters email and password |
| 3 | System | Validates input format (email syntax, non-empty password) |
| 4 | System | Verifies credentials against the database |
| 5 | System | Issues a JWT and stores it in the response header |
| 6 | System | Redirects user to their role-specific dashboard |
| Exceptions | | |
| 3a | System | “Invalid email format” → show error |
| 4a | System | “Incorrect email or password” → show error |
| Postcondition | | User holds a valid session token and is authenticated |

Table 2: Login Functional Specification Table

### **4.3. Take IELTS Test**

|  |  |  |
| --- | --- | --- |
| **Use Case** | | **Take IELTS** |
| Actor | | Student |
| Precondition | | Student is logged in |
| Main Flow | | |
| 1 | Student | Chooses “Take IELTS Test” |
| 2 | System | Displays available tests |
| 3 | Student | Starts selected test; time begins |
| 4 | Student | Answers Listening and Reading questions |
| 5 | System | Auto-grades Listening and Reading, shows immediate scores |
| 6 | Student | Submits Writing text and records Speaking response |
| 7 | Student | Submits entire test; marks Writing/Speaking as “Pending” |
| 8 | Student | Saves all responses and statuses |
| Exception | | |
| 5a | System | Time expires -> test auto-submits |
| Postcondition | | Listening/Reading scored; Writing/Speaking queued for manual grading |

Table 3: Take IELTS Test Functional Specification Table

### **4.4. Practice Flashcards**

|  |  |  |
| --- | --- | --- |
| **Use Case** | | **Practice Flashcards** |
| Actor | | Student |
| Precondition | | Student is logged in |
| Main Flow | | |
| 1 | Student | Selects “Practice Flashcards” |
| 2 | System | Loads due flashcards from the database |
| 3 | System | Displays the first card |
| 4 | Student | Reviews pronunciation/audio, views definition, then rates (Easy/Hard/Again) |
| 5 | System | Saves rating, schedules next review |
| 6 | System | Shows the next card or “All caught up!” |
| Exception | | |
| 2a | System | No due cards → displays “No cards due” |
| Postcondition | | Flashcard progress and next-review dates are stored |

Table 4: Practice Flashcards Functional Specification Table

### **4.5. View Feedback and Grades**

|  |  |  |
| --- | --- | --- |
| **Use Case** | | **View Feedback & Grades** |
| Actor | | Student |
| Precondition | | Student has at least one completed test |
| Main Flow | | |
| 1 | Student | Clicks “View Feedback & Grades” |
| 2 | System | Loads test history and available scores/feedback |
| 3 | System | Displays automatic scores and teacher comments |
| Exception | | |
| 2a | System | No completed tests → shows “No results available” |
| Postcondition | | Student can review scores and feedback details |

Table 5: View Feedback and Grades Functional Specification Table

### **4.6. Create Tests**

|  |  |  |
| --- | --- | --- |
| **Use Case** | | **Create Tests** |
| Actor | | Teacher |
| Precondition | | Teacher is logged in |
| Main Flow | | |
| 1 | Teacher | Selects “Create Tests” |
| 2 | System | Shows new-test creation form |
| 3 | Teacher | Enters title, instructions, question types |
| 4 | Teacher | Uploads audio if Listening/Speaking sections apply |
| 5 | System | Validates inputs |
| 6 | System | Saves test and confirms “Creation successful” |
| Exception | | |
| 5a | System | “Invalid data” → prompts teacher to correct fields |
| Postcondition | | New test is published and available to students |

Table 6: Create Tests Functional Specification Table

### **4.7. View Student Submissions**

|  |  |  |
| --- | --- | --- |
| **Use Case** | | **View Student Submissions** |
| Actor | | Teacher |
| Precondition | | Teacher is logged in |
| Main Flow | | |
| 1 | Teacher | Selects “View Submissions” |
| 2 | System | Loads all submissions with metadata |
| 3 | System | Displays list; teacher may open any for grading |
| Postcondition | | Teacher has visibility into student work |

Table 7: View Student Submissions Functional Specification Table

### **4.8. Manage Flashcards**

|  |  |  |
| --- | --- | --- |
| **Use Case** | | **Manage Flashcards** |
| Actor | | Administrator |
| Precondition | | Admin is logged in |
| Main Flow | | |
| 1 | Admin | Selects Manage Flashcards |
| 2 | System | Retrieves all flashcards from database |
| 3 | System | Displays the list in the admin UI |
| Add Flashcard | | |
| 4 | Admin | Clicks Add Flashcard |
| 5 | System | Shows blank flashcard creation form |
| 6 | Admin | Enters term, part of speech, difficulty |
| 7 | Admin | Clicks “Fetch Dictionary Data” |
| 8 | System | Call external Dictionary API and retrieve definition, IPA transcription, audio URL |
| 9 | System | Populate form fields with the fetched data |
| 10 | Admin | Clicks “Save” |
| 11 | System | Validate all inputs (term non-empty, IPA present, etc.) |
| 12 | System | Persist new flashcard in the database |
| Edit Flashcard | | |
| 13 | Admin | Clicks “Edit” on an existing flashcard |
| 14 | System | Show form pre-filled with current flashcard data |
| 15 | Admin | Modify any fields, or click “Fetch Dictionary Data” again |
| 16 | Admin | Clicks “Save” |
| 17 | System | Show “Flashcard updated successfully” |
| Delete Flashcard | | |
| 18 | Admin | Clicks “Delete” on a flashcard |
| 19 | System | Confirm deletion |
| 20 | System | Remove flashcard from the database |
| 21 | System | Show “Flashcard deleted successfully” |
| Exception | | |
| 7a | System | Dictionary API error → show “Unable to fetch IPA; please try again” |
| 11a | System | Input validation fails → show “Please fill in all fields correctly” |
| 20a | System | Deletion fails → show “Could not delete flashcard; retry later” |
| Postcondition | | The database reflects the admin’s changes (added, updated, or deleted flashcards), and the UI list is refreshed accordingly. |

Table 8: Manage Flashcards Functional Specification Table

### **4.9. Manage Users**

|  |  |  |
| --- | --- | --- |
| **Use Case** | | **Manage Users** |
| Actor | | Administrator |
| Precondition | | Admin is logged in |
| Main Flow | | |
| 1 | Admin | Selects “Manage Users” |
| 2 | System | Loads current user list |
| 3 | System | Displays table of users |
| Add User | | |
| 4 | Admin | Clicks “Add User” |
| 5 | Admin | Enters email, name, assigns role |
| 6 | System | Validates inputs |
| 7 | System | Creates account and confirms “User created” |
| Edit User | | |
| 8 | Admin | Clicks “Edit” on a specific user |
| 9 | Admin | Updates details (email, name, role, status) |
| 10 | System | Saves changes and confirms “Update successful” |
| Delete User | | |
| 11 | Admin | Clicks “Delete” on a specific user |
| 12 | System | Prompts for confirmation |
| 13 | Admin | Confirm deletion |
| 14 | System | Removes account and confirms “Deletion successful” |
| Exceptions | | |
| 6a | System | “Email already exists” |
| 10a | System | “Update failed, please retry” |
| 14a | System | “Deletion failed, please retry” |
| Postcondition | | User list in the database reflects all create/edit/delete operations. |

Table 9: Manage Users Functional Specification Table

### **4.10. Manage Tests**

|  |  |  |
| --- | --- | --- |
| **Use Case** | | **Manage Tests** |
| Actor | | Administrator |
| Precondition | | Admin is logged in |
| Main Flow | | |
| 1 | Admin | Selects Manage Tests |
| 2 | System | Retrieves all tests from database |
| 3 | System | Displays the list in the admin UI |
| Add Test | | |
| 4 | Admin | Clicks Create Test |
| 5 | System | Shows empty test-creation form |
| 6 | Admin | Enters title, duration, instructions |
| 7 | Admin | (If Listening/Speaking) uploads audio |
| 8 | System | Validates metadata and file formats |
| 9 | System | Saves new test and confirms success |
| Edit Test | | |
| 10 | Admin | Chooses a test and clicks Edit |
| 11 | System | Shows form with existing data |
| 12 | Admin | Modifies fields or replaces audio |
| 13 | System | Validates updates |
| 14 | System | Updates test record and confirms success |
| Delete Test | | |
| 15 | Admin | Chooses a test and clicks Delete |
| 16 | System | Prompts for confirmation |
| 17 | Admin | Confirm deletion |
| 18 | System | Removes test and confirms success |
| Exceptions | | |
| 8a | System | “Invalid input or audio format” → error |
| 14a | System | “Update failed” → prompt retry |
| 18a | System | “Deletion failed” → advise retry |
| Postcondition | | Test catalog reflects all admin changes |

Table 10: Manage Tests Functional Specification Table

# **III. Design**

## **1. System Architecture**

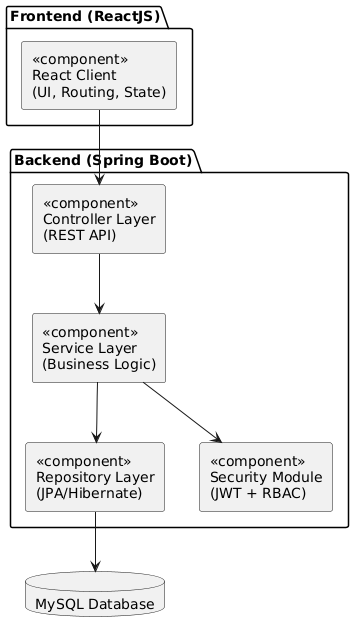


Figure 2: System Architecture

The system follows a three-tier architecture model, separating concerns across the frontend interface, backend application logic, and data storage. At the client side, the application is built using ReactJS, responsible for rendering the user interface, managing state, and handling routing. All user interactions such as logging in, taking tests, practicing flashcards, and submitting answers are sent to the backend through HTTP requests.

The backend is developed using Spring Boot and is structured into multiple layers. The Controller Layer receives and processes incoming API requests. These requests are then passed to the Service Layer, which contains the core business logic of the application. The Security Module, embedded within the backend, handles JWT-based authentication and role-based access control to ensure that only authorized users can access protected resources.

The Service Layer communicates with the Repository Layer, which uses JPA/Hibernate to interact with the underlying MySQL database. This database stores all persistent data, including user information, test content, results, and flashcards. Additionally, for functionalities involving audio (such as Listening and Speaking tests), the system connects to an external Audio Storage service either by uploading base64-encoded files to the database or integrating with Firebase for cloud-based storage.

## **2. Sequence Diagram**

### **2.1. Login & Authentication**

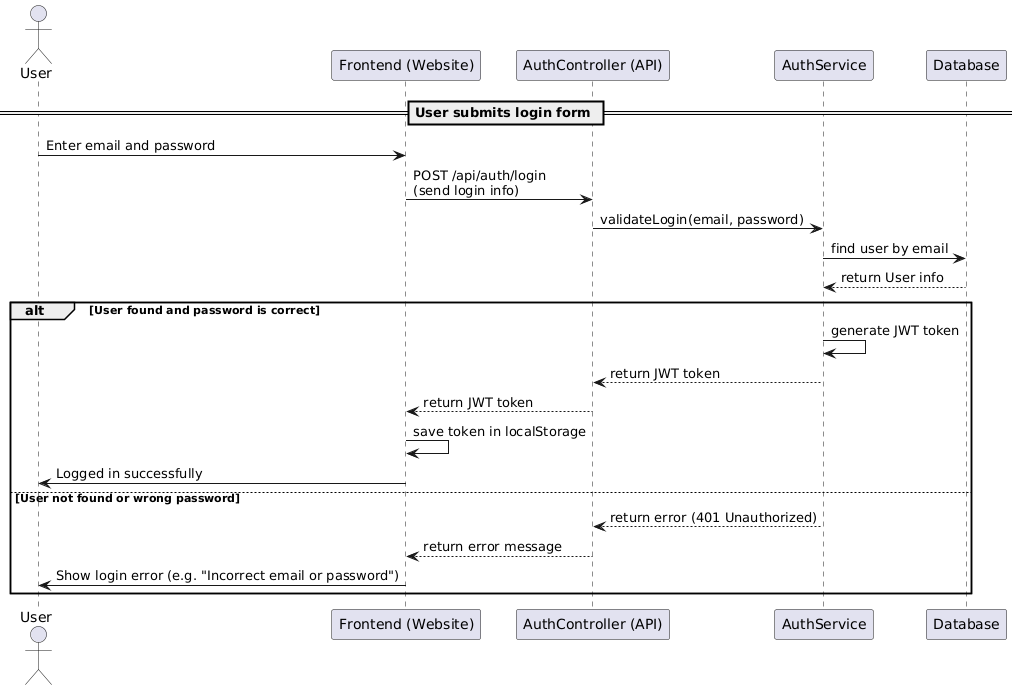


Figure 3: Login & Authentication Sequence Diagram

As illustrated in the figure 3, the login and authentication workflow of our IELTS Practice System is designed to be both secure and user-friendly. When a user begins the login process, they enter their email and password into the login form on the frontend interface. The frontend then sends this information to the backend via a POST request to the /api/auth/login endpoint.

Once the AuthController receives the request, it delegates the verification task to the AuthService. This service component interacts with the UserRepository to search for a matching user based on the provided email. If a user is found and the password matches, the system proceeds to generate a JWT (JSON Web Token), which serves as a digital access credential. The generated JWT is returned to the frontend and securely stored in the browser, typically in localStorage. This token is automatically attached to all future API requests, enabling the system to recognize and authenticate the user without requiring them to log in again during the session. In the event that the login credentials are invalid, such as an incorrect email or password, the backend returns a clear error message, and the frontend displays this feedback to inform the user.

We use the JWT not only for identity verification but also to enforce role-based access control. Each token includes the user’s role (Student, Teacher, or Admin), which allows us to restrict or grant access to specific features throughout the system. This logic is implemented modularly and consistently across all secure endpoints.

### **2.2. Take Test & Submit Answers**

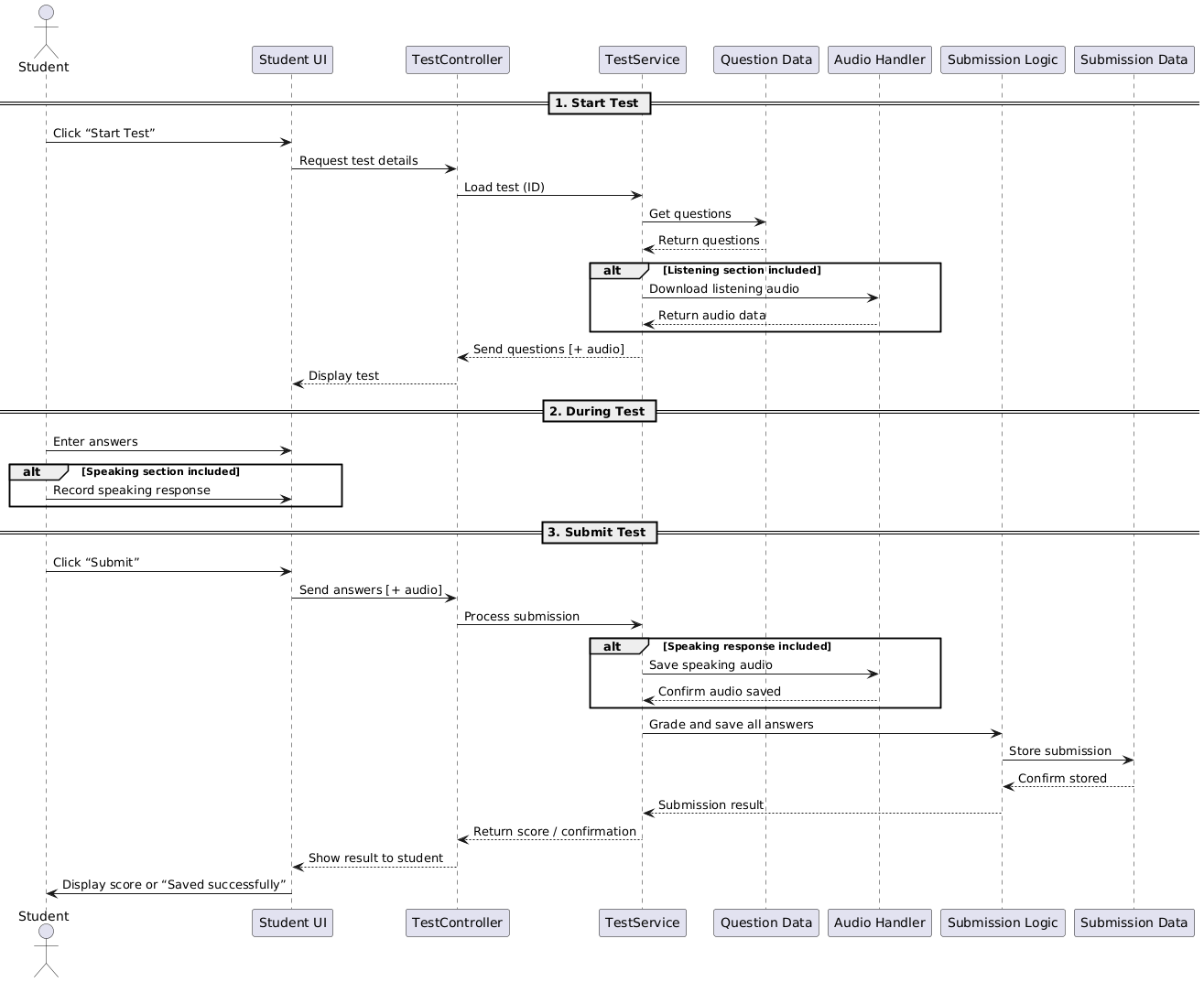


Figure 4: Take Test & Submit Answers Sequence Diagram

Figure 4 illustrates the end-to-end test-taking flow for a student using the IELTS preparation system. When students choose to take a test, the frontend sends a request to the backend to retrieve the relevant test structure, which includes questions, instructions, and audio (if applicable). We used conditional logic to determine whether the selected test is a Listening or Speaking test. For Listening tests, the backend returns Base64-encoded audio, which we decode and play using the browser's audio API. For Speaking sections, we allow students to record their voice directly in the browser and store that audio temporarily until submission.

Once students complete the test, their answers, both textual and audio, are compiled into a structured payload and submitted back to the backend. On the server side, we handle multipart requests and process audio files by encoding them into Base64 and saving them to the database. For objective test parts like Reading or Listening, we apply automated grading logic immediately, returning the calculated score. This sequence not only handles media but also considers asynchronous operations and provides feedback to users in near real-time.

### **2.3. Practice Flashcard**

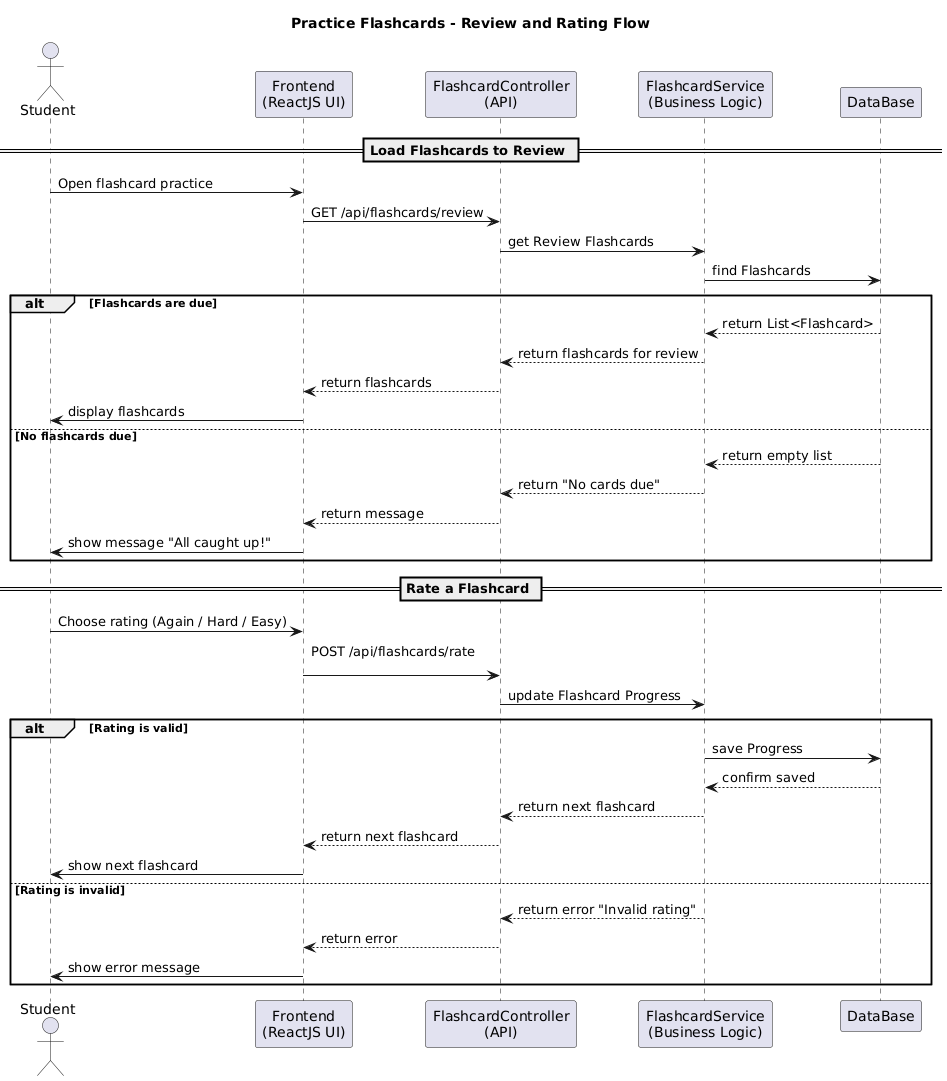
Figure 5: Practice Flashcard sequence diagram

Figure 5 illustrates the interaction flow of a student reviewing vocabulary flashcards using the spaced repetition technique.

When a student initiates a flashcard session, the frontend sends a request to fetch vocabulary that is due for review based on the student's previous performance. If any flashcards are ready, the system returns them and displays them one by one on the interface. In the event that no cards are scheduled for review, a friendly message like "You're all caught up!" is shown instead, so that learners clearly understand their review queue is empty. During the review, for each flashcard shown, the student selects a difficulty rating, either "Again", "Hard", or "Easy", to indicate how well they remember the word. This choice is then sent to the backend in an API request. Before updating the review schedule, the backend validates the rating to ensure it is a valid option. If an invalid value is received (for example, due to a client bug or manipulation), the system responds with an error message to prevent corrupt data from entering the database.

If the rating is valid, the backend recalculates the next review interval based on that difficulty level, following a simplified spaced repetition algorithm. It then updates the progress in the database, allowing the student to continue to the next flashcard.

### **2.4. Create new Test**

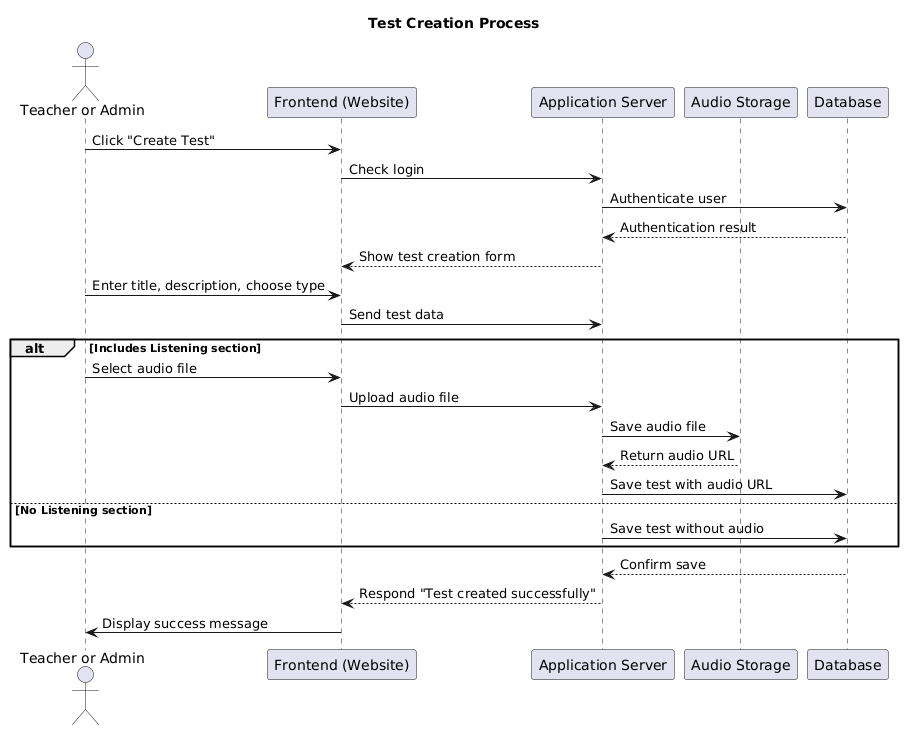


Figure 6: Create New Test Sequence Diagram

Figure 6 illustrated a streamlined test creation process that enables teachers or admin to design and manage assessments. When a user begins creating a new test, they are shown a simple interface where they can enter key details such as the test title, type (e.g., Reading, Listening, Speaking), time limits, and instructions.

If the test includes a Listening section, the interface allows users to upload an audio file using drag-and-drop or file selection. Once uploaded, the audio is saved in the server’s storage, and a link to that file is attached to the test. This way, the audio becomes part of the test without requiring any complicated steps.After all the information is filled in, the system sends the data to the backend, where it verifies the user’s identity, saves the test to the database, and returns a success message. If there’s no audio, the process is even faster since the upload step is skipped.

### **2.5. Grading System**

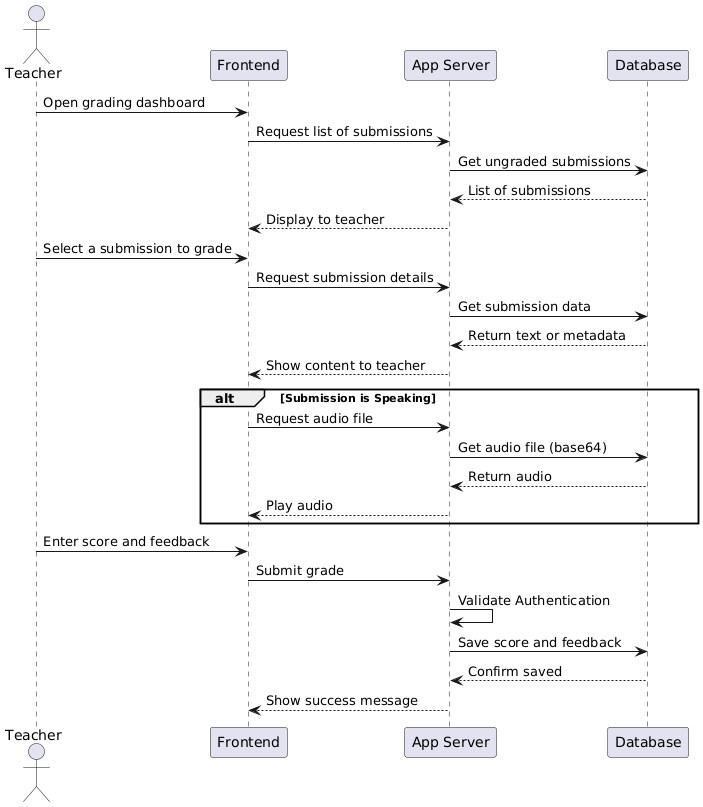


Figure 7: Grading System Sequence Diagram

Figure 7 clearly illustrates the step-by-step interaction involved in manual grading. Grading the Speaking and Writing parts of a test is done manually by teachers. After logging in, the teacher goes to their grading dashboard, where the system shows a list of all student submissions that haven’t been graded yet. When the teacher clicks on one of the submissions, the frontend sends a request to the server to get the student’s answers. If the submission is for a Speaking test, an additional request is sent to load the audio file.

The audio is fetched through a separate and secure API endpoint. Before sending the audio, the server checks the teacher’s identity and whether they are allowed to grade that submission. This ensures privacy and prevents unauthorized access. Once the data is loaded, the teacher can listen to the student’s response or read their essay directly on the website. They then enter a score and write comments. When the teacher submits their feedback, it is saved to the database and linked to the student’s original test attempt.

### **2.6. Manage System for admin**

#### **2.6.1. Manage Tests**

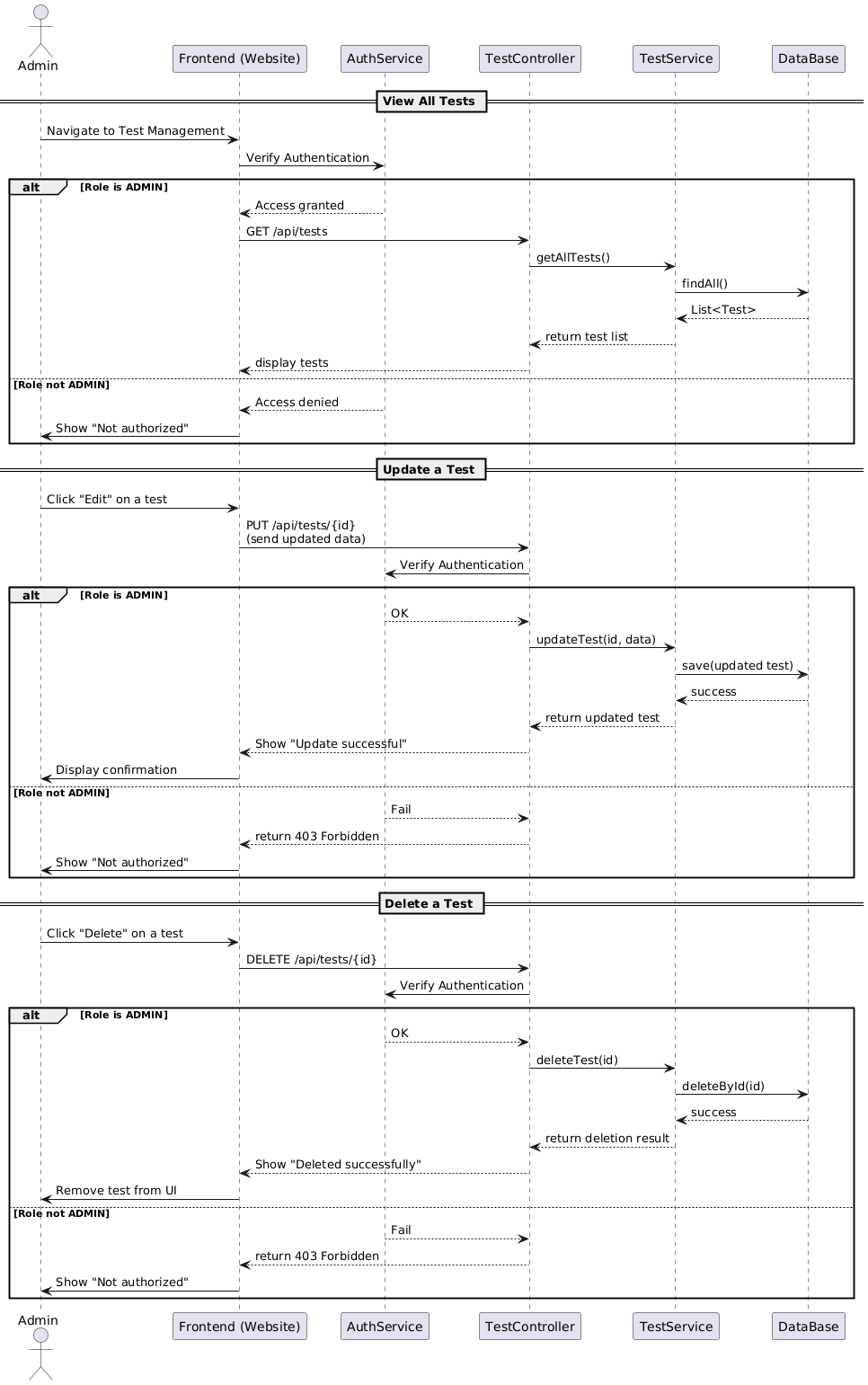


Figure 8: Manage Tests Sequence Diagram

Administrators can view and manage all tests in the system. Through the admin interface, they can sort tests by date, type, or author, and choose to edit or delete them. All actions are routed through protected endpoints that verify JWT tokens and user roles.

To ensure that editing a test does not disrupt students currently taking it, we implemented version tracking and lock mechanisms. Only tests that are inactive (not being attempted) are editable. If a test needs to be removed, it is flagged for archival instead of hard deletion, which helps us preserve system integrity. This management process allows admins to maintain control over the content lifecycle and keep the platform’s database consistent and clean.

#### **2.6.2. Manage Users**

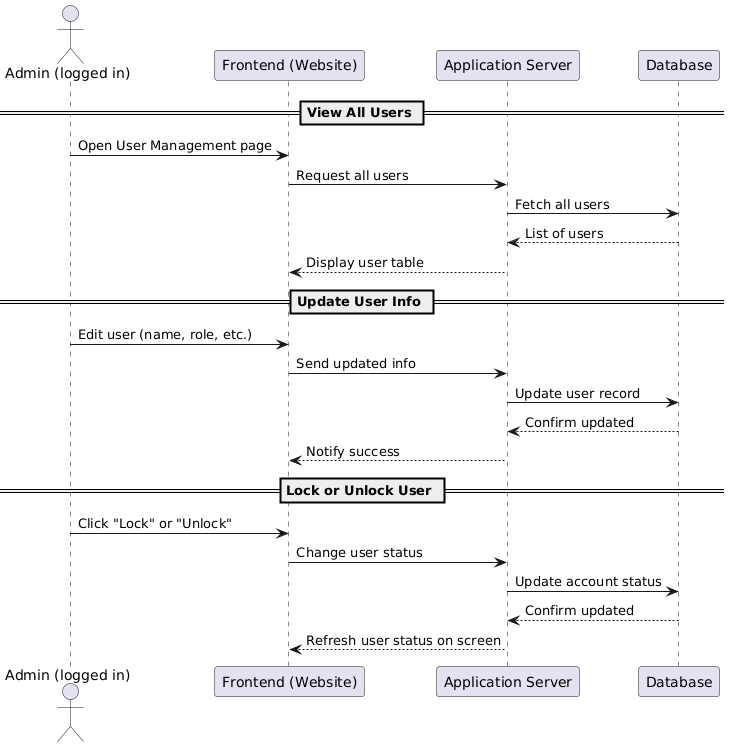


Figure 9: Manage Users Sequence Diagram

Administrators are responsible for managing the user base, and we provided them with a user-friendly interface to do so. When the admin dashboard loads, a GET request fetches all user profiles. The admin can then perform actions such as editing profile information, resetting passwords, modifying roles, and locking or unlocking user accounts. Each of these actions is tied to a secure API call, validated through the administrator’s JWT token.

We designed the backend to log each user management action for auditing purposes, and the frontend includes real-time feedback such as toast notifications and updated status indicators. These design choices helped us streamline the administrative workflow and strengthen system transparency.

#### **2.6.3. Manage Flashcards**

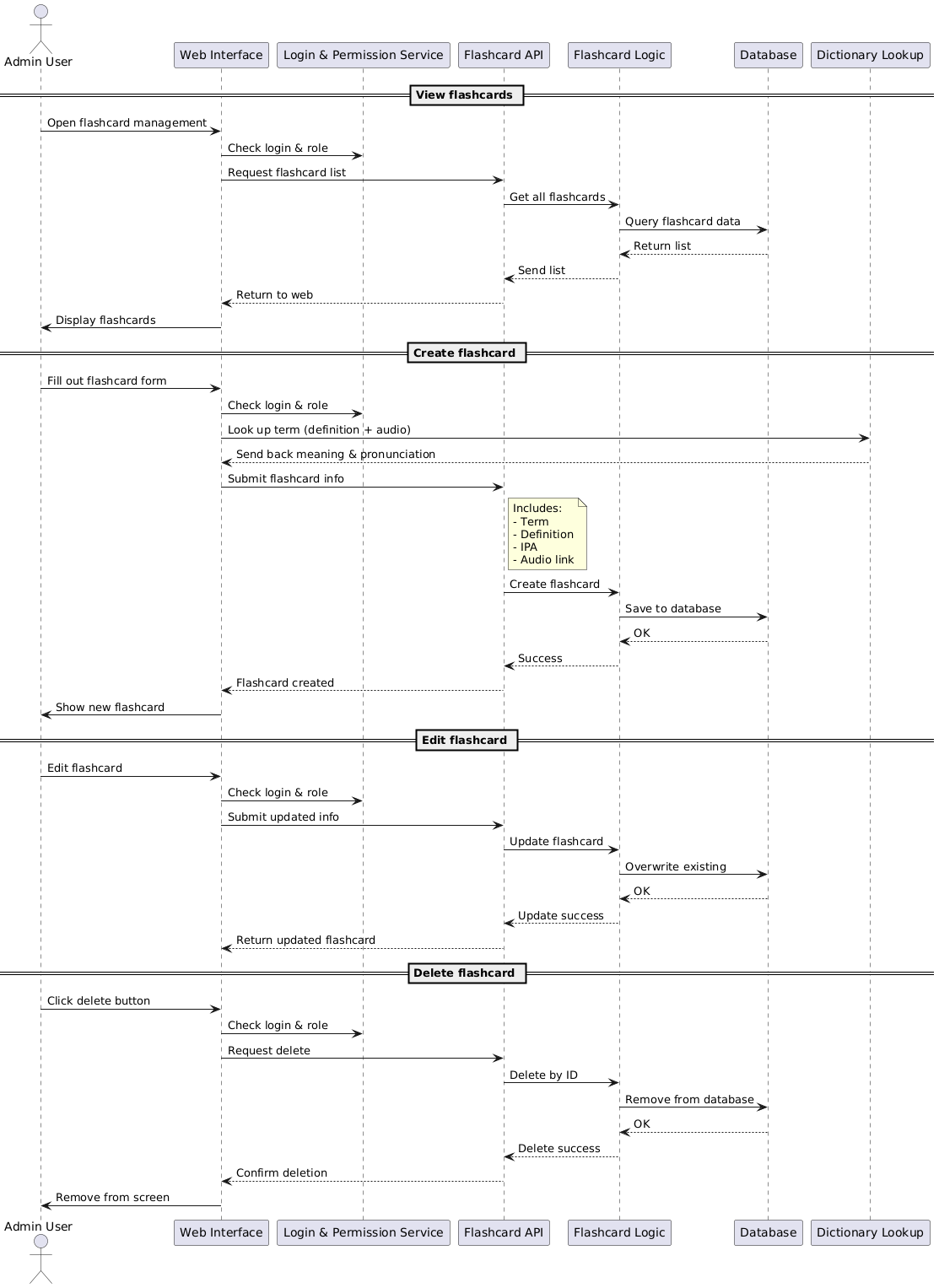


Figure 10: Manage Flashcards Sequence Diagram

Our flashcard system makes vocabulary learning easier by letting teachers and admins create and manage flashcards through a simple web interface. When a user wants to create a new flashcard, they just type in a keyword. Behind the scenes, our system automatically looks up that word using an online dictionary service, retrieving useful details like pronunciation (IPA), an audio file, and a definition.

These results are shown instantly to the user for review, then saved along with the flashcard into our system. This helps reduce manual work and ensures all flashcards have accurate and rich information. Users can also easily update or delete existing flashcards when needed.

## **3. Database Design**



Figure 11: Entity Relational Diagram

The database schema for our system was carefully designed to accommodate the complexity and diversity of features, including user management, test creation and participation, submission tracking, flashcard learning, and performance reporting. Built using a MySQL relational database, the schema emphasizes normalization, referential integrity, and flexibility for future expansion.

### **3.1. User Management**

- users: This is one of the most important tables in the database. It stores basic information about each user, like their name, email, hashed password, role (ADMIN, TEACHER, STUDENT), and whether their account is active.

- Many other tables, such as test\_attempts, student\_progress, and student\_flashcard\_progress, are linked to this table through the user\_id field so we can track who did what.

### **3.2. Test and Question Structure**

- tests: This table stores overall information about each test, like the title, skill type (Listening, Reading, etc.), difficulty level, and the user who created it.

- questions: Stores all the questions for each test. Each question has its content, type (e.g., multiple choice), and any media files attached (like audio or images).

- correct\_answers: Keeps the correct answers for each question. This is mainly used to automatically grade Reading and Listening questions.

- reading\_passage and listening\_audio: These tables store extra content like long reading texts or audio files that are linked to tests.

### **3.3. Submissions and Grading**

- test\_attempts: Every time a student takes a test, a record is saved in this table. It includes scores for each skill, how long the test took, and teacher feedback if available.  
- student\_responses: This table stores each individual answer to every question. For Speaking tasks, we store audio recordings as Base64-encoded strings here.

- speaking\_writing\_criteria\_scores: When teachers mark Speaking and Writing manually, their scores for each IELTS criterion (fluency, grammar, vocabulary, etc.) are saved in this table, along with any comments.

### **3.4. Progress and Performance Tracking**

- student\_progress: Helps us track each student’s learning journey, including their average scores, number of completed tests, and whether they’re improving over time.

- test\_score\_mapping: Translates raw scores (e.g., number of correct answers) into IELTS band scores. This ensures that results are standardized and easy to interpret.

### **3.5. Flashcard Learning System**

- flashcards: Stores vocabulary items along with their meaning, example sentence, IPA pronunciation, audio file, category, and difficulty level.

- student\_flashcard\_progress: Tracks how students are doing with their flashcards. For example, how many times they’ve reviewed each word, whether they’ve mastered it, and when the next review should happen (using spaced repetition).

### **3.6. Evaluation and Supporting Data**

- speaking\_writing\_criteria\_scores (as mentioned earlier): This table also holds teacher comments and detailed scores per criterion, which are useful for giving personalized feedback.

# 

# **IV. Tools And Technologies**

To build a secure, interactive, and scalable IELTS preparation system, we adopted a modern and well-integrated technology stack. The tools and technologies were selected based on their stability, community support, scalability, and suitability for real-time educational web platforms.

## **1. Backend Technology: Spring Boot and Java**

The backend of the system is developed using Spring Boot, a modern Java framework that simplifies the creation of standalone, production-grade RESTful web services. It follows the principle of convention over configuration and supports embedded servers such as Tomcat, making it ideal for rapid backend development.

We selected Spring Boot for its modular structure, allowing clear separation of logic through the Controller–Service–Repository layers. This architecture helped us effectively implement features such as user authentication, role-based access control, test management, grading, and flashcard handling. Furthermore, Java’s strong type system and mature ecosystem contributed to enhanced reliability and maintainability of the backend codebase.

## **2. Database Technology: MySQL with JPA and Hibernate**

For persistent data storage, we used MySQL, a widely-used relational database system known for its reliability and support for complex data relationships. MySQL stores all structured data, including user information, test results, audio submissions, vocabulary flashcards, and learning history.

To simplify data interaction, the backend integrates Spring Data JPA with Hibernate as the ORM (Object-Relational Mapping) provider. This enables seamless mapping between Java entity classes and relational tables, allowing us to perform queries and transactions using Java code instead of writing SQL manually. The database schema is automatically generated from the entity definitions, streamlining schema evolution and ensuring data consistency.

## **3. Frontend Technology: ReactJS and JavaScript**

The client-side interface is developed using ReactJS, a JavaScript library for building responsive, component-based user interfaces. React's Virtual DOM improves performance by minimizing direct DOM manipulation, which is essential for real-time features such as countdown timers, audio recording, and interactive flashcards.

React enables a modular structure where each part of the interface (login, dashboard, test screen, flashcard review) is implemented as an isolated component. We used React Router DOM for single-page navigation, allowing users to switch between views without full page reloads. All API interactions are handled via Axios, which also manages authentication tokens in request headers. To enhance user experience, React-Toastify is employed for real-time feedback through toast notifications.

## **4. Security and Authentication: JWT and Spring Security**

Security is a critical aspect of the system. We use Spring Security to enforce authentication and authorization on all backend API endpoints. Combined with JWT (JSON Web Token), this ensures a stateless authentication model where users log in once and receive a token used in subsequent requests.

Roles are assigned to users (e.g., STUDENT, TEACHER, ADMIN), and access to each endpoint is restricted based on these roles using Spring's method-level security annotations. Passwords are encrypted using the BCrypt hashing algorithm before being stored in the database, ensuring secure credential management.

## **5. Audio Handling: MediaRecorder and Base64 Encoding**

One of the core functionalities of the platform is recording student responses for the Speaking test. On the client side, we use the MediaRecorder API, which allows in-browser audio recording directly from the user's microphone.

Once the recording is complete, it is converted into a Base64-encoded string using the FileReader API. This encoded data is then sent to the backend and stored in the database. The use of Base64 ensures compatibility with JSON payloads and allows the audio to be embedded directly into the frontend interface using the standard <audio> element for playback.

## **6. Vocabulary Enrichment: Dictionary and TTS APIs**

To enhance the vocabulary learning experience, the system integrates external APIs:

* The DictionaryAPI.dev is used to retrieve definitions, IPA transcriptions, and audio pronunciation links for new flashcard entries.
* If no pronunciation audio is available from the dictionary, we utilize ResponsiveVoice, a text-to-speech (TTS) API, to generate the audio dynamically.

This integration allows teachers and administrators to create flashcards quickly and ensures a rich learning experience for students.

# **V. Implementation**

## **1. Database connection**

In our project, we connected the backend to a MySQL database using Spring Boot, and the process was straightforward thanks to Spring Boot’s built-in support for database configuration. We included two main dependencies in our pom.xml: one for Spring Data JPA, which also brings in Hibernate as the default ORM tool, and one for the MySQL JDBC driver.

All the configuration was done through the application.properties file. Here, we defined the JDBC URL pointing to our local MySQL server, set the username (root), and left the password empty for local development. We also enabled automatic schema updates by setting spring.jpa.hibernate.ddl-auto to update, which allowed Hibernate to create or modify tables based on our Java entity classes. For debugging purposes, we also chose to log all executed SQL statements.

Once the application starts, Spring Boot automatically reads these settings and sets up everything behind the scenes. It initializes a connection pool using HikariCP, scans for classes marked with @Entity, and syncs them with the corresponding tables in the database. This setup allowed us to avoid writing any manual JDBC code.

Instead, we worked entirely with Spring Data JPA by creating repository interfaces that extend JpaRepository. These interfaces gave us access to all the common database operations without having to write SQL ourselves. In our service layer, we simply injected the repositories and used them to perform database operations.

This setup made the data layer very clean and efficient. Spring handled everything under the hood, from managing connections, generating SQL queries, to mapping results back to Java objects. As a result, we were able to focus more on building the actual business logic of the system rather than spending time on repetitive or low-level code.

## **2. Integration and System Flow**

Our application integrates the frontend and backend through RESTful API endpoints protected with JWT authentication. During development, the React frontend typically runs on port 3000, while the Spring Boot backend runs on port 8080. Because these services operate on different origins, cross-origin request issues (CORS) naturally arise.

To handle this, we configured Cross-Origin Resource Sharing (CORS) in our Spring Boot backend by implementing a global configuration using WebMvcConfigurer**.** This setup allows our backend to securely accept HTTP requests from the frontend development server, our frontend directly calls the backend using absolute API URLs (e.g., http://localhost:8080/api/...). This explicit routing approach ensures transparency and matches production behavior more closely.

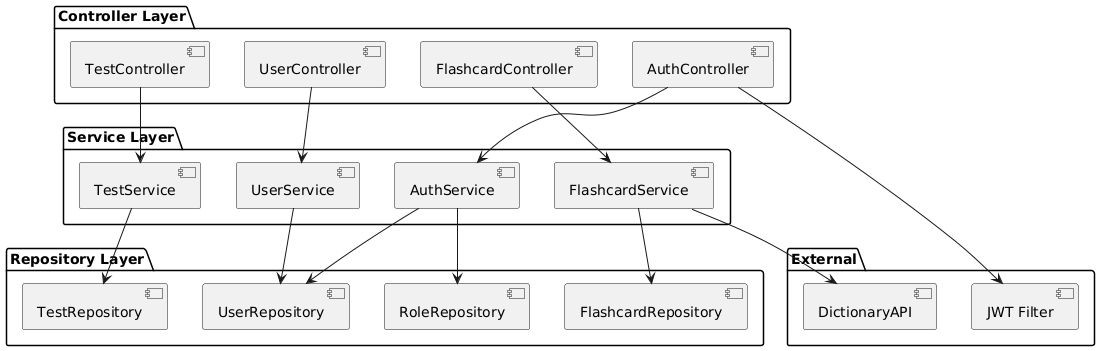
In production, the React frontend and Spring Boot backend are hosted separately, without relying on tools like Nginx. The React app is built with npm run build and deployed to a hosting provider, while the backend is served from its own domain or server. The frontend interacts with the backend by using its public URL (e.g., https://api.ourapp.com), and CORS must be enabled on the backend to accept these requests.

This architecture promotes decoupling between UI and business logic while maintaining secure, token-based communication. For instance, when a student completes a Speaking test, the React app converts the audio recording to a Base64-encoded string and submits it along with other answers via an authenticated HTTP request. The backend verifies the JWT token, saves the data, and responds with a confirmation. This consistent integration strategy supports scalability, debugging, and long-term maintenance.

This architecture ensures that our services remain decoupled and scalable, while also maintaining secure and reliable communication. For example, when a student submits a Speaking test with audio, the React app encodes the file in Base64 and sends it through a secure POST request. The backend validates the token, stores the audio and answers in the database, and returns a status update. Teachers later access this submission from their dashboards to provide manual grading.

## **3. Backend Implementation**

The backend architecture of our project is grounded in the Model-View-Controller (MVC) pattern, which promotes clean separation of responsibilities across components.



*Figure 12:Backend System Architecture*

This diagram emphasizes how each controller delegates incoming HTTP requests to its corresponding service, which encapsulates business logic. These services interact with the data persistence layer through repositories. Additionally, external integrations such as the Dictionary API (used to enrich flashcard content) and JWT security filters (used for authentication) are clearly represented. This modular structure ensures the backend is maintainable, scalable, and secure.

Before diving into each module, it's important to explain the architectural pattern our system is based on. The backend architecture adheres closely to the Model-View-Controller (MVC) paradigm. In this structure, the Model layer consists of JPA entities that reflect database tables. The Controller layer handles HTTP requests and user interactions, while the Service layer acts as an intermediary that encapsulates business logic. This clear separation helps us avoid code duplication and enhances testability.

### **3.1. Technologies and Configuration**

We implemented the backend using Java 17 with Spring Boot 2.6+. Our project is managed using Maven, which handles dependencies and project lifecycle. In our application.properties, we configured the database connection string, JWT security keys, CORS filters, and file upload limitations.

### **3.2. Package Structure and Layered Design**

To ensure code readability and maintainability, we adopted a layered architecture pattern. Each layer has a clearly defined responsibility:

* The controllerlayer handles HTTP requests and routes them to appropriate services.
* The service layer contains business logic such as grading, authentication, dictionary lookup, and audio processing.
* The repository layer communicates with the database via Spring Data JPA.
* We also structuredentity, dto, security, and exception packages to isolate domain models, data transfer representations, authentication filters, and error handlers respectively.

This separation made our system easier to scale, debug, and extend in the future.

### **3.3. Authentication and Role-Based Authorization**

We applied JWT-based authentication across our backend. Upon successful login, a token is issued and stored on the client side. This token is required for accessing all protected routes. We use Spring Security filters to intercept and validate tokens. Role-based access is enforced using method-level annotations like @PreAuthorize, which restrict access to endpoints based on roles such as ADMIN, TEACHER, or STUDENT. Unauthorized attempts result in a structured error response.

### **3.4. Audio Handling (Upload and Retrieval)**

To support the Speaking and Listening modules, we developed audio processing functionality. Audio files are submitted from the client in multipart/form-data format. The backend receives these files via MultipartFile, reads the byte stream, and converts it into a Base64-encoded string, which is stored in the database using a LONGTEXT field.

This method allows embedding audio directly into response payloads without using external storage. When needed, the backend retrieves the encoded string, embeds it in the response, and the frontend decodes and plays it back using a standard HTML element.

### **3.5. Test Module**

Test creation is managed via the TestController and TestService. Admins and teachers can create multi-section tests, each linked to a set of questions. If a Listening section is included, the audio file is uploaded and validated on the backend before being Base64-encoded and stored.

The system supports full CRUD operations for tests. Automated grading is applied to Listening and Reading sections using keyword comparison. All logic related to test creation, updates, and retrieval is centralized in the TestService.

### **3.6. Submission Module**

Students submit test responses through multipart forms, combining written answers with audio recordings (for Speaking). Submissions are stored in StudentResponse and TestAttempt entities. Reading and Listening responses are graded automatically, while Speaking and Writing submissions are marked as "Pending" for manual teacher review.

To maintain data integrity, we use the @Transactional annotation to ensure consistency across operations when saving test attempts and their corresponding responses.

### **3.7. Flashcard Management**

Our flashcard system allows users to input vocabulary words. Upon creation, the backend sends an HTTP request to the DictionaryAPI to fetch the word's definition, IPA transcription, and pronunciation audio. If the dictionary lacks audio, a fallback TTS URL is generated using ResponsiveVoice.

Parsed data is stored in the flashcard database. Users can rate flashcards or review them later. We implemented pagination and filtering mechanisms to allow sorting by difficulty and review date.

### **3.8. User Management**

Administrators manage users by activating/deactivating accounts, assigning roles, and viewing user details. Sensitive operations are protected by role checks and logged for traceability. We use soft deletion to mark users as inactive without permanently erasing data. This allows future reactivation and audit history.

### **3.9. Repository & Database Queries**

Database operations are implemented using Spring Data JPA’s repository interfaces. For complex operations such as retrieving pending submissions or filtering flashcards by difficulty we used custom JPQL queries via the @Query annotation.

These queries are optimized for performance and structured for pagination where appropriate. Entity relationships are fully mapped, including OneToMany and ManyToOne relations for users, tests, and submissions.

## **4. Frontend Implementation**

### **4.1. Technologies and Project Setup**

We built the frontend with ReactJS 17, using create-react-app for project scaffolding. We used react-router-dom for navigation between components and axios for HTTP communication. The frontend connects with the backend via a central API service module that handles token injection and response parsing.

### **4.2. Session and HTTP Request Management**

JWT tokens are stored in localStorage upon login. We configured axios interceptors to automatically attach the token to every outgoing request. If the backend returns a 401 status code, we redirect users to the login page and clear the local storage. This session handling logic ensures security and smooth user experience.

### **4.3. Audio Features**

To allow voice recording, we used the Web API MediaRecorder. Audio blobs are converted to Base64 strings using FileReader. These Base64 strings are then sent to the backend for storage. To play audio, we simply use an <audio> element and assign its src to a data:audio/mp3;base64,... URI.

### **4.4. Key Components and Pages**

We designed the frontend using functional components and CSS modules. The major pages include:

* Authentication Pages: Login and Register
* Dashboard: Dynamic dashboard showing options based on user role
* Test Interface: Displays test instructions, questions, and handles recording/submission
* Flashcard Interface: Vocabulary learning and rating
* Admin Panel: User list, role assignment, test and flashcard management

Each component was modularized and designed to maximize reusability.

### **4.5. UI/UX Handling**

We used libraries like react-toastify to display real-time notifications. Loading indicators were shown using spinners while fetching or submitting data. All form inputs are validated and error messages are displayed inline. We optimized the layout for clarity and made the design responsive across devices.

# **VI. Result and Testing**

By the end of the project, we were able to successfully turn our original idea into a fully working IELTS practice platform. The system covers all the core functionalities we set out to implement and provides a complete solution for students, teachers, and administrators.

For students, the platform offers a realistic IELTS testing experience. They can log in, take practice tests in all four skills—Listening, Reading, Writing, and Speaking—and receive detailed feedback. The Speaking and Writing sections, which require manual grading, are stored and later reviewed by teachers. Students can also study vocabulary using flashcards and track their progress through a personalized dashboard.

For teachers, the system includes a test management interface where they can create IELTS-style tests, upload audio files, and assign questions for each section. They can also view student submissions, grade subjective answers, and leave feedback. The grading interface is straightforward and designed to support efficient evaluation.

For administrators, we built a simple but powerful admin panel that allows them to manage users, control content (tests and flashcards), and monitor system activities. Admins can activate or deactivate accounts, assign roles, and ensure that only authorized users access certain features.

From a technical perspective, the backend was implemented with a focus on clear structure, security, and role-based access. Although we didn’t define exact technologies at the beginning, we managed to implement secure login, token-based authorization, audio handling for Speaking and Listening, flashcard review with dictionary support, and full API communication with the frontend. We also focused on making the platform user-friendly and responsive. On the frontend, the interface adapts to different screen sizes, displays helpful notifications, and guides the user through each process—whether it's doing a test, grading, or reviewing feedback.

Overall, the system meets the initial goals we had in mind: it supports meaningful IELTS preparation, gives teachers the tools they need to assess students, and keeps the platform manageable for admins. Most importantly, it works as a complete system that can be expanded or deployed in real-world environments with little additional setup.

## **1. Discussion**

Throughout the development of this project, we faced both technical and practical challenges that helped us gain valuable experience. From the beginning, we aimed to build a complete and useful IELTS practice system that supports students in all four skills while making it easier for teachers to manage and evaluate their performance. Although we did not have much experience with building large-scale systems before, the use of well-established tools like Spring Boot, ReactJS, and MySQL helped us structure the project in a clear and maintainable way.

One of the main things we learned during this process was how to properly separate responsibilities between backend and frontend, and how to secure an application using JWT authentication and role-based access. We also became more confident in handling database relationships, especially when designing features like test submissions, scoring, and flashcard tracking. Another important aspect was dealing with media files such as audio uploads and playback, which was new to us and required extra research to handle correctly in both the backend and frontend.

Due to time limitations, we focused on delivering core features that would work well and be usable by real students. This meant we had to make some compromises. For example, we did not include automated testing or mobile support in this version, and all tests were done manually. However, this gave us a chance to carefully check each part of the system and understand how everything works together.

What makes this project special is that it brings together multiple useful features into one single platform, which is something we noticed was missing in many existing IELTS practice websites. By combining tests, grading, audio, and flashcards into one place, we believe this platform provides a more complete learning experience. While there are still many ways to improve it, we are proud of what we achieved and confident that it could be used as a starting point for further development or real-world deployment in schools or English centers.

## **2. Testing**

### **2.1. Authentication & JWT Handling**

|  |  |  |  |
| --- | --- | --- | --- |
| Test Case | Expected Result | Actual Result | Status |
| Tampered JWT (altered token payload) | Server should detect invalid signature and return 401 Unauthorized | Token can be manually modified via browser dev tools. While some redirects occur, parts of the app still temporarily render before logout | Fail |
| Expired JWT Token | Token is invalid, system should redirect user to login or register with 401 | Expired session triggered login redirect | Pass |
| Logout and revisit dashboard | Access should be denied, returning 401 or redirect to login | Access blocked as expected, login required | Pass |
| Repeated failed login attempts | System allows retry with error message; no lockout expected | Error shown; login still available | Pass |

Table 11: Authentication & JWT Handling Testing Table

The tampered JWT test failed because, although the backend does detect the modified token eventually, there is a short delay where the frontend still renders content before the system forces logout. This could create confusion for users or allow brief unintended access. The frontend should ideally detect and reject altered tokens immediately on load, rather than waiting for user interaction.

### **2.2. Role-Based Access Control**

|  |  |  |  |
| --- | --- | --- | --- |
| Test Case | Expected Result | Actual Result | Status |
| Student accesses teacher-only page | Access blocked (403) or redirected. | Page access denied, 403 returned | Pass |
| Teacher tries admin functionality | Restricted features not visible or blocked with 403 | Admin panel hidden, forced URL blocked | Pass |
| Accessing another user’s data via modified URL | Request rejected or blank result | Unauthorized access prevented | Pass |
| Unauthenticated user accesses protected page | Redirect to login or 401 error | Login page shown, access denied | Pass |

Table 12: Role-Based Access Control Testing Table

### **2.3. Test Creation & Management**

|  |  |  |  |
| --- | --- | --- | --- |
| Test Case | Expected Result | Actual Result | Status |
| Submit incomplete test form | Validation errors shown, submission blocked. | Test not saved, user alerted. | Pass |
| Upload large or invalid audio file | Upload fails with size/format error. | File rejected, warning displayed. | Pass |
| Edit test and verify update on student side | Updated content visible to students. | Students saw a revised version. | Pass |
| Delete test with submissions | Either blocked with warning or handled gracefully. | Deleted test removed; no warning given. | Pass |

Table 13: Test Creation & Management Testing Table

### **2.4. Test Submission & Auto-Grading**

|  |  |  |  |
| --- | --- | --- | --- |
| Test Case | Expected Result | Actual Result | Status |
| Let timer expire during test | Auto-submit triggers; answer saved | Auto-submit executed on timeout | Pass |
| Attempt to submit after time’s up | Submission not accepted or blocked | Submit disabled after time expired | Pass |
| Submit with blank answer | Submission accepted; blanks scored zero | Graded as expected, zero for blanks | Pass |
| Check auto-grading with known correct answers | Score matches answer key | Accurate score calculation | Pass |
| Case-insensitive answer checking | Variants like "new york" match "New York". | Normalization confirmed. | Pass |
| Display scores immediately | Objective scores shown; subjective pending | Reading/Listening scores visible instantly | Pass |
| Submit long writing causes layout issue | Text should wrap or scroll properly | Overflow in grading UI | Fail |

Table 14: Test Submission & Auto-Grading Testing Table

The layout issue for long writing submissions failed because there was no proper scroll or wrap behavior defined for lengthy text. This can negatively affect grading visibility for teachers, especially when reviewing essays or extended answers.

### **2.5. Manual Teacher Grading**

|  |  |  |  |
| --- | --- | --- | --- |
| Test Case | Expected Result | Actual Result | Status |
| Enter invalid score (e.g. 10 or letters) | Input rejected or error shown | Validation worked correctly | Pass |
| Save after only grading one section | Partial save accepted; other pending | One section saved; grading incomplete | Pass |
| Student views updated feedback | Scores and comments shown | Results visible after grading | Pass |
| Modify previously saved score | Scores and comments shown | Revision applied correctly | Pass |
| Grade submission not owned by teacher | Access denied or record hidden | Access blocked, not visible | Pass |

Table 15: Manual Teacher Grading Testing Table

### **2.6. Audio Upload & Playback**

|  |  |  |  |
| --- | --- | --- | --- |
| Test Case | Expected Result | Actual Result | Status |
| Upload invalid file type | Upload fails, error shown. | Rejected as expected. | Pass |
| Play Listening audio | Audio plays smoothly | Worked as expected | Pass |
| Record and playback Speaking | Audio saved and played. | Clear audio, correct behavior. | Pass |
| Upload interrupted by network loss | Error shown, allow retry. | Upload error handled gracefully. | Pass |
| Access audio without login | 401/403 error returned. | Unauthorized request blocked. | Pass |
| Spinner after speaking upload | Confirmation should appear post-upload | Spinner stuck; audio uploaded successfully | Fail |

Table 16: Audio Upload & Playback Testing Table

The spinner remained visible after audio upload, suggesting that the upload status was not correctly updated in the frontend. This could confuse users who believe the process is still ongoing, even though the audio was successfully uploaded.

### **2.7. Flashcard Learning**

|  |  |  |  |
| --- | --- | --- | --- |
| Test Case | Expected Result | Actual Result | Status |
| Add flashcard from dictionary | Data fetched: meaning, IPA, audio. | Full flashcard created. | Pass |
| Review behavior: mark easy/difficult | Schedule adjusts review timing. | Spaced repetition worked. | Pass |
| Finish all due cards | Show "No more cards" message. | Review queue completed properly. | Pass |
| Audio playback for word | Plays if available; else disabled. | Pronunciation played or icon disabled. | Pass |

Table 17: Flashcard Learning Testing Table

# **VII. Conclusion and Future Works**

## **1. Conclusion**

After several months of development, we have successfully built a web-based IELTS preparation system that supports both learning and testing. The system includes full functionality for all four IELTS skills: Listening, Reading, Writing, and Speaking. Users can take practice tests online, submit their answers (including speaking recordings), and receive feedback either instantly (for objective sections) or from teachers (for subjective sections). Teachers can create and manage test content, grade student submissions, and view overall performance easily. In addition to test-taking, the system also provides a flashcard learning module, allowing students to review vocabulary using a spaced repetition approach.

From a technical perspective, the backend was developed using Spring Boot, with JWT authentication and role-based access to ensure secure interactions. The frontend, built with ReactJS, provides a responsive and user-friendly interface for all user roles (student, teacher, admin). Data is stored and managed through a MySQL relational database that is both well-organized and scalable. During the project, we manually tested all major features and handled various edge cases. Although we did not use automated tools, the testing results showed that the system worked reliably and handled different use scenarios correctly. Overall, this project has created a complete and practical platform that can support IELTS learners more effectively, while also providing useful tools for teachers and administrators.

## **2. Future Development**

Even though the system is stable and fully functional, there are still many directions for improvement. In future versions, we hope to add AI-powered scoring for Writing and Speaking so that students can receive automatic feedback without waiting for teacher input. This will save time for teachers and help students improve faster. The current user interface can also be redesigned to look more modern and be easier for new users to navigate. In addition, the system should be made mobile-friendly or developed into a mobile app so that learners can study and take tests on their phones.

Another important improvement would be to add progress tracking and visual reports. This would allow students to monitor their performance over time and focus on their weak areas. Teachers could also benefit from seeing which parts of the test students struggle with the most. Furthermore, if possible, we plan to include more realistic IELTS questions by using official content or questions that closely follow real exam formats.

By continuing to develop and improve the system, we believe it can become an even more helpful tool for IELTS learners and educators, and be ready for use in real-world learning environments.

# 

# **VIII. References**

[1] IELTSZIE. (n.d.). *Practice IELTS tests and resources online*. Retrieved from <https://ieltszie.com>

[2] Cathoven. (n.d.). *IELTS learning resources and platform*. Retrieved from<https://cathoven.com>

[3] IELTS First. (n.d.). *Which is better: Online or Offline coaching for IELTS?* Retrieved from [https://www.ieltsfirst.com/ielts-coaching/which-is-better-online-or-offline-coaching-for-ielts](https://www.ieltsfirst.com/ielts-coaching/which-is-better-online-or-offline-coaching-for-ielts/)

[4] Tuổi Trẻ Online. (n.d.). *Latest news and updates in education*. Retrieved from<https://news.tuoitre.vn>

[5] IELTS Tutors. (n.d.). *Online IELTS practice platforms and apps*. Retrieved from<https://www.ieltstutors.co.uk/other-practice-materials-online-practice-platforms-and-apps>

# **APPENDICES**

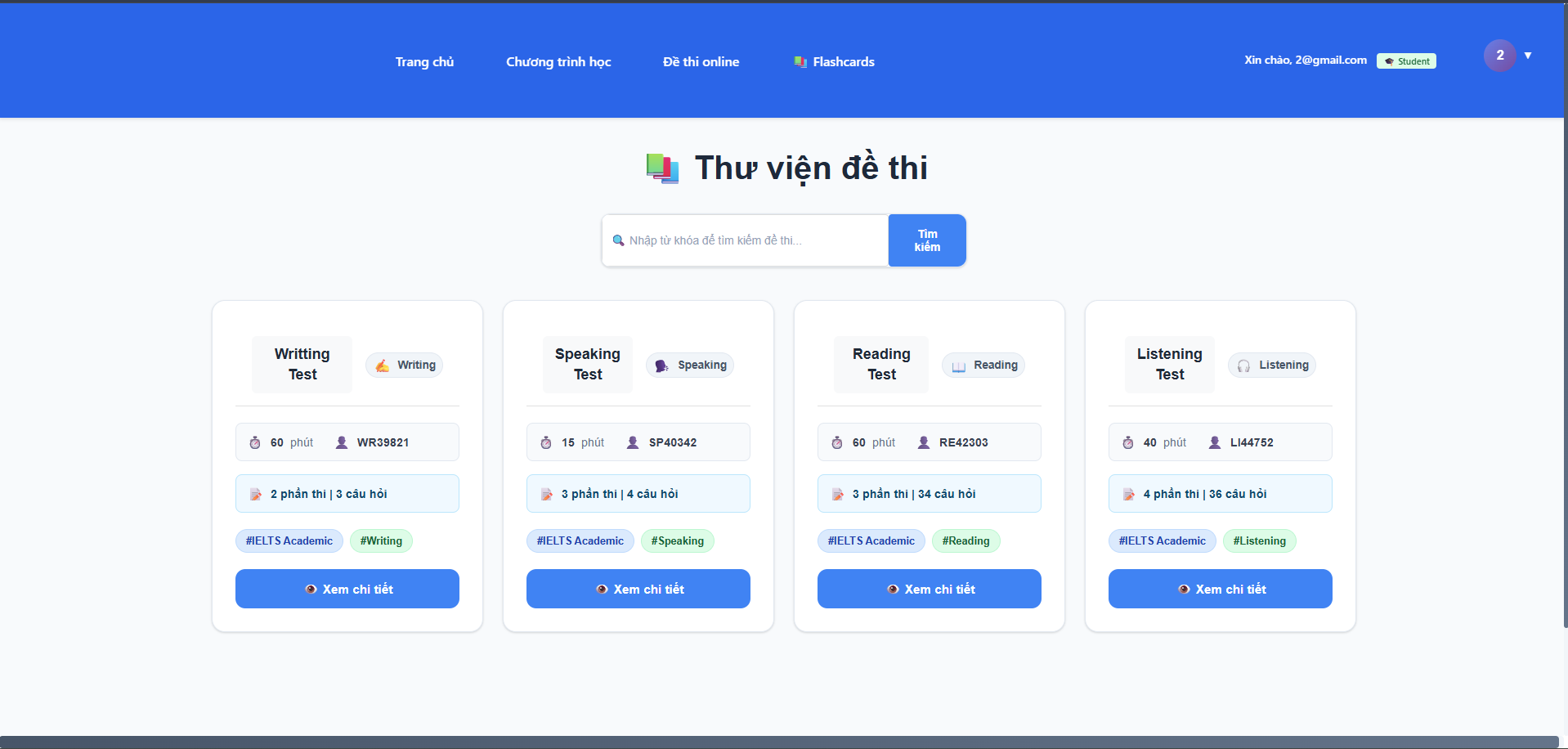


Figure 12: Home Page

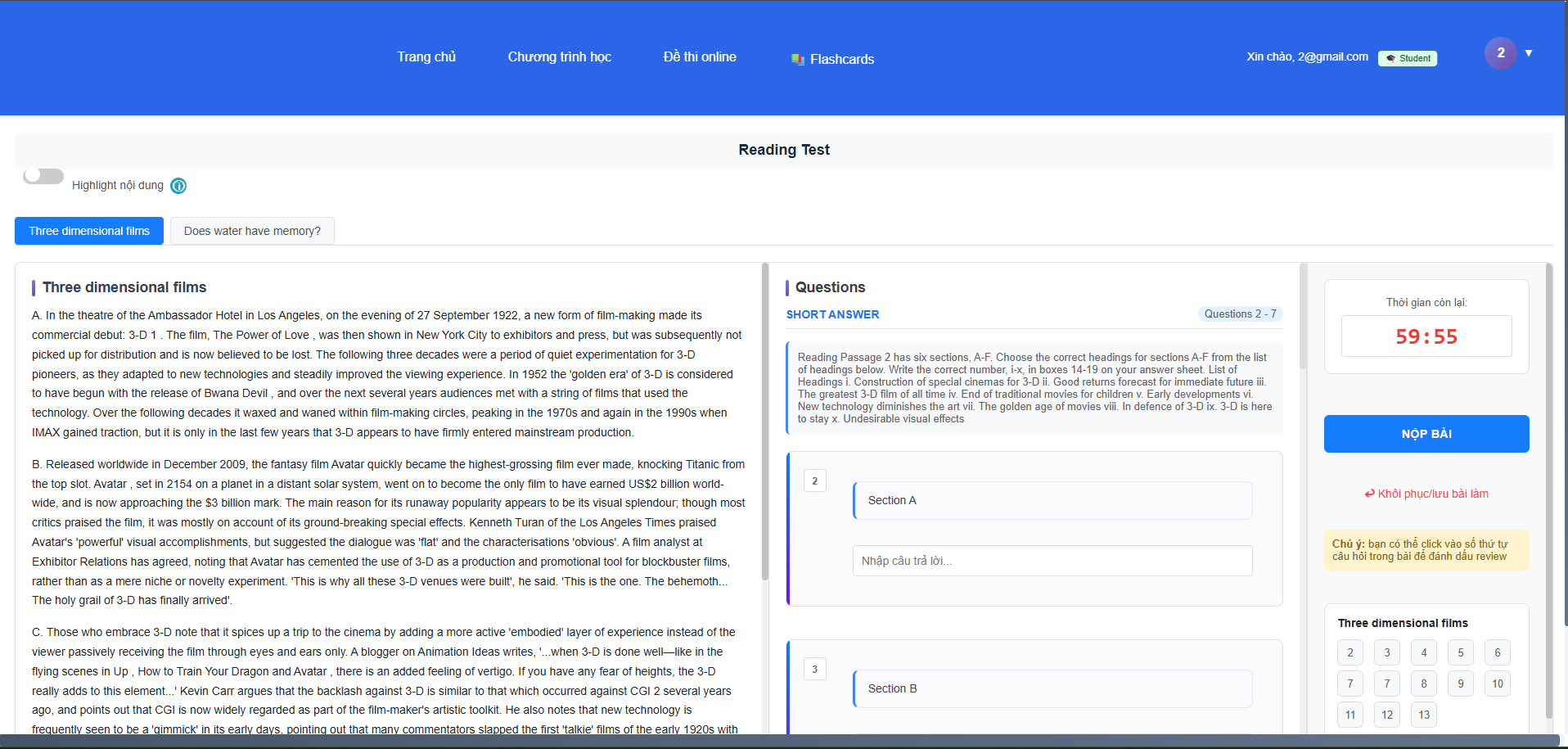


Figure 13: Taking Test Screen

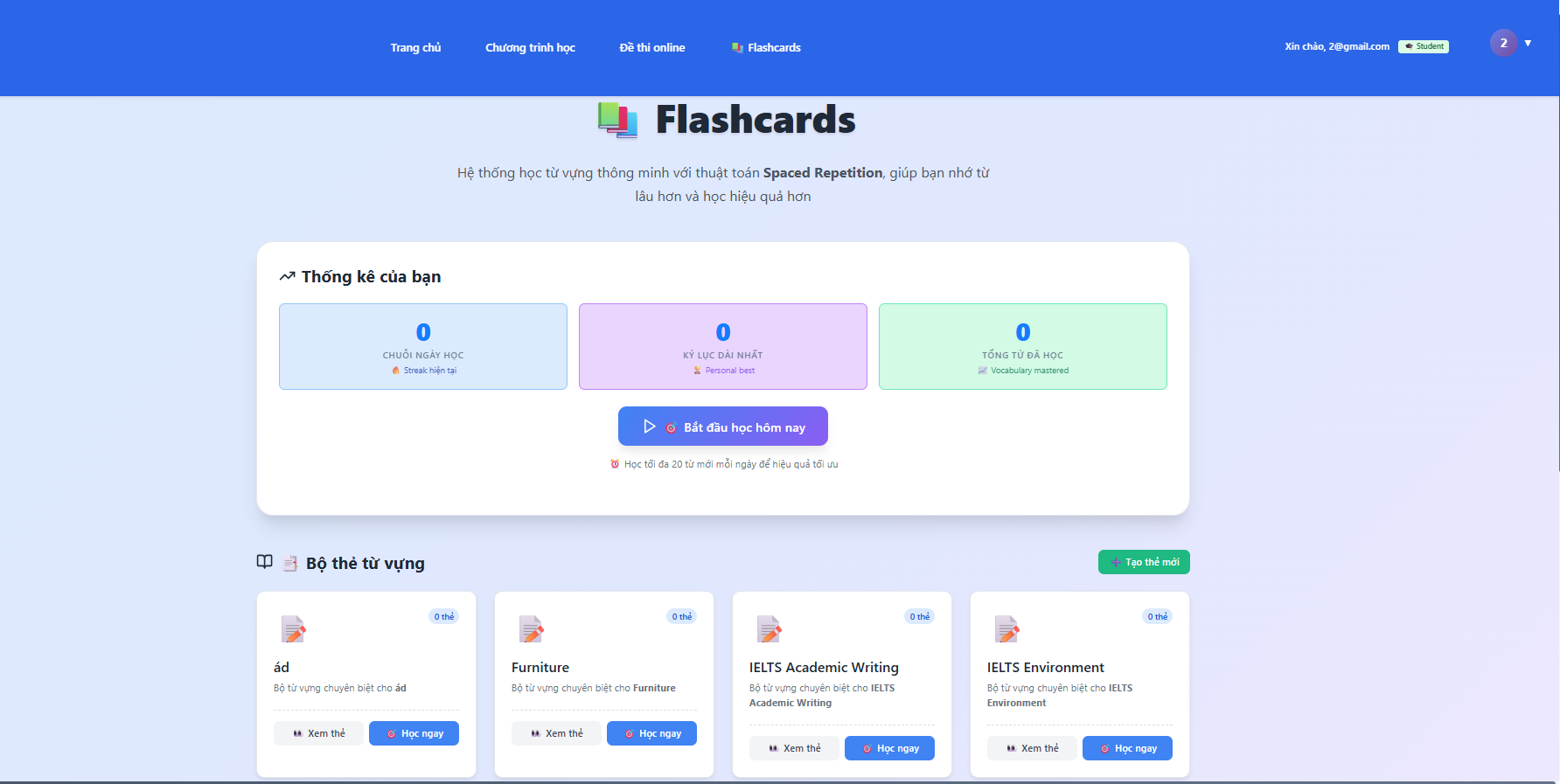


Figure 14: Flashcard Home Screen

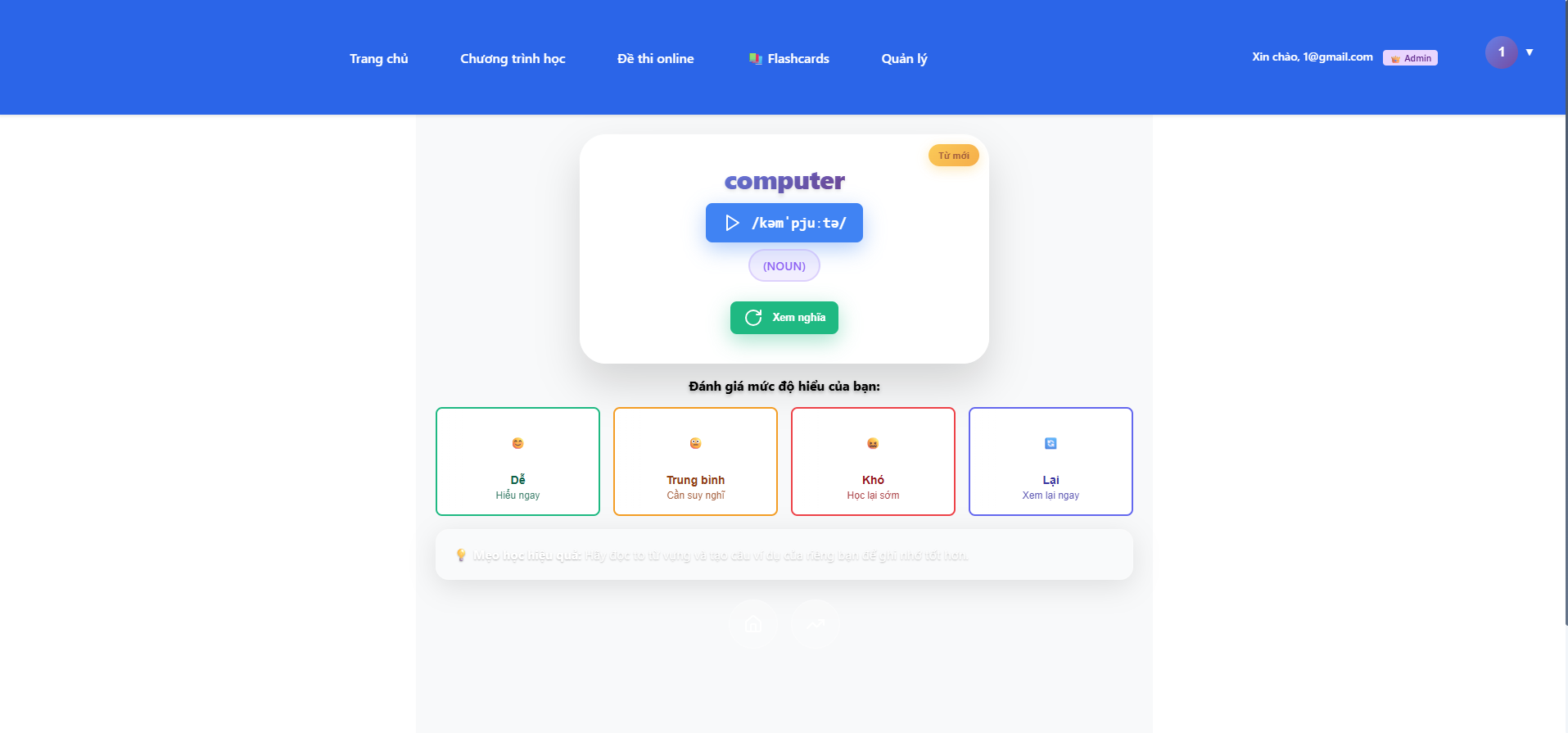
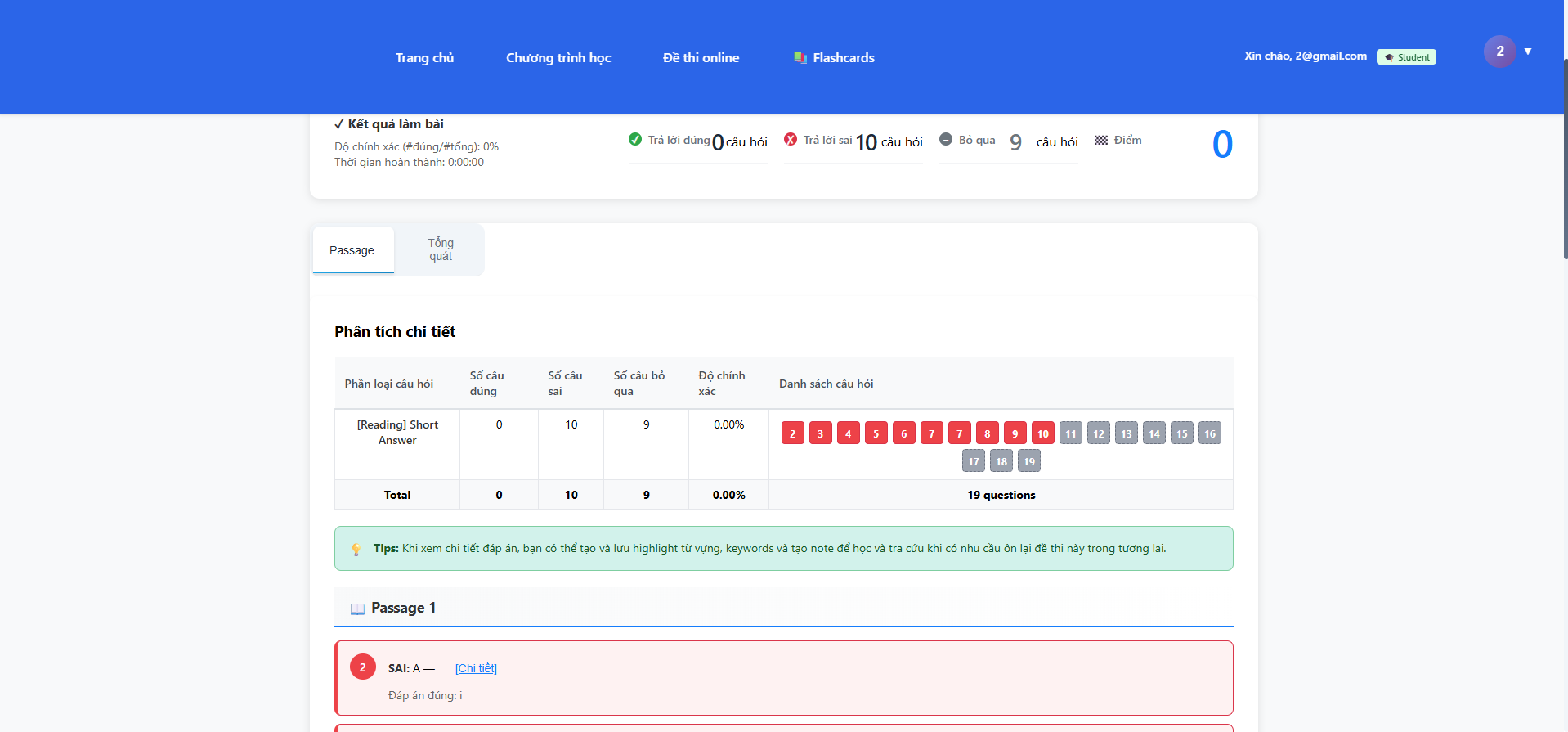


Figure 15: Practice Flash Card Screen

Figure 16: Test result Screen

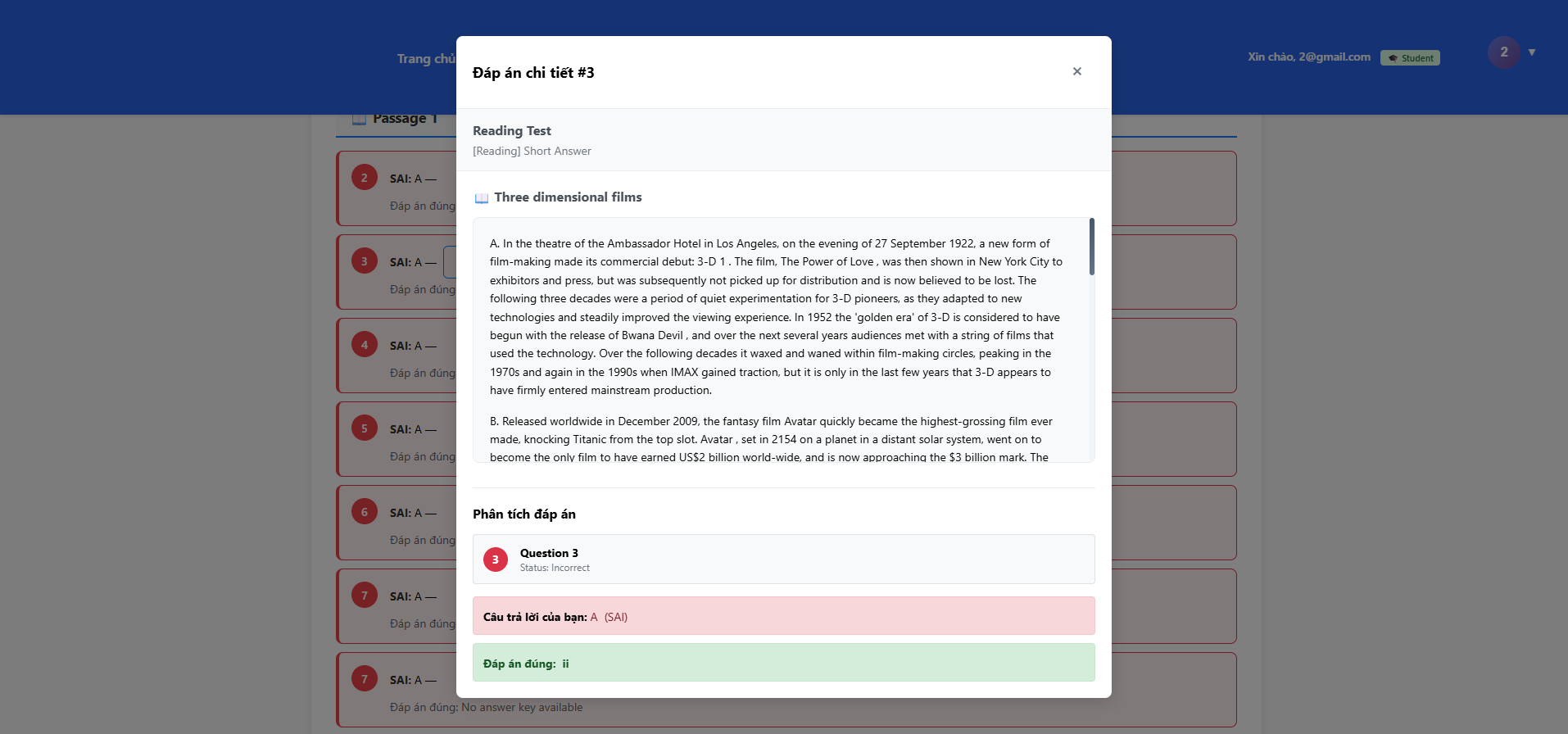


Figure 17: Answer’s Details Screen

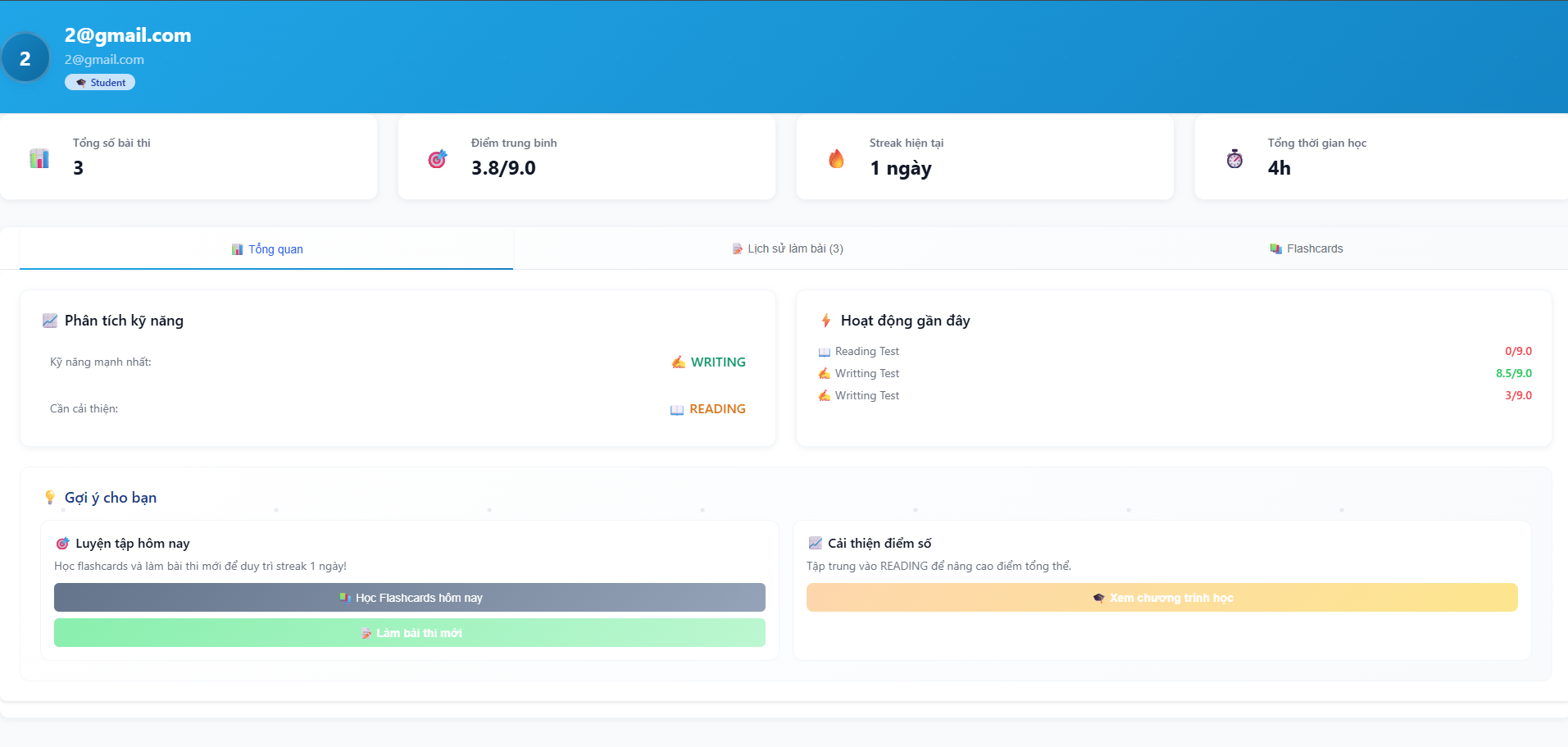


Figure 18: User’s Profile

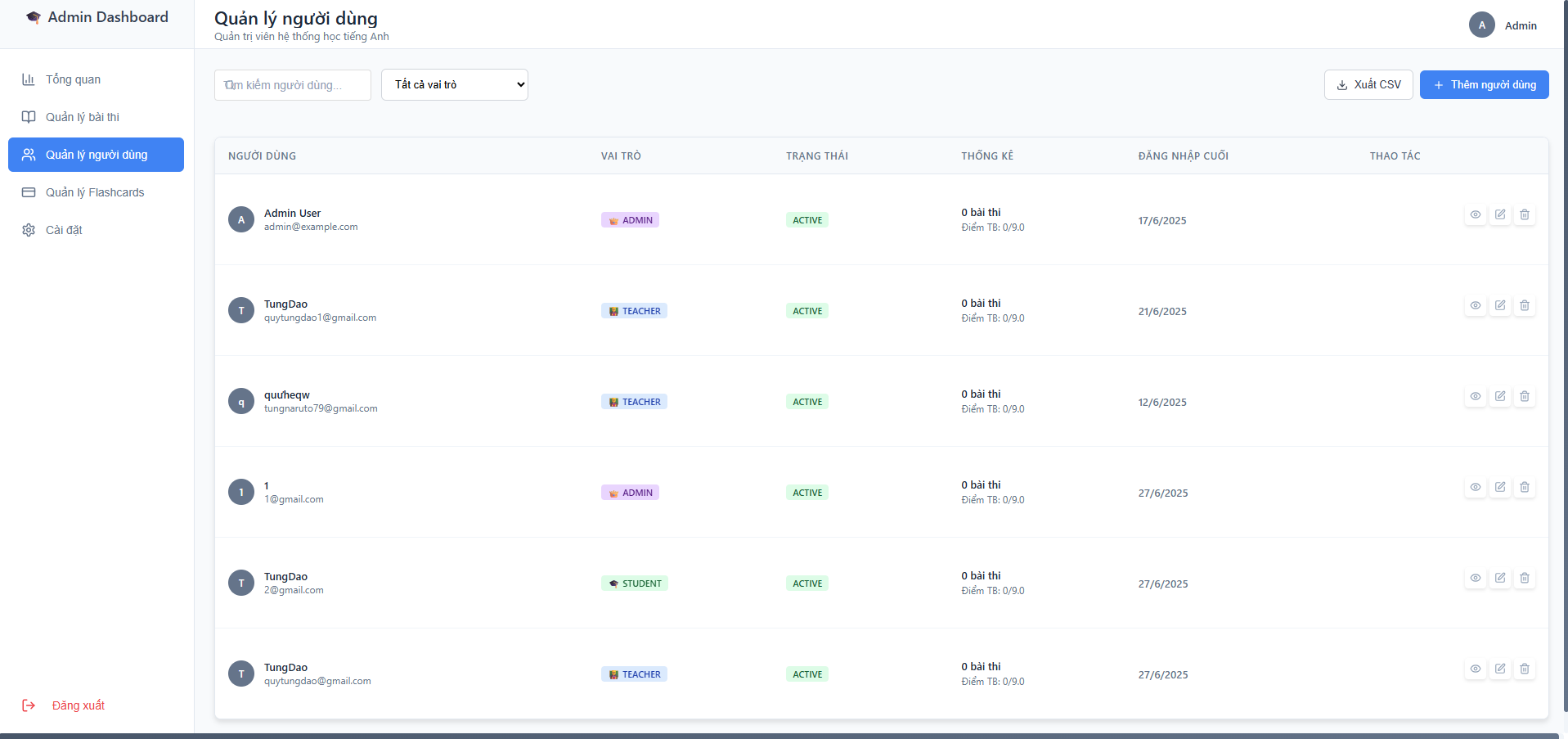


Figure 19: Admin Dashboard