Thiết kế kiến trúc LingoLeap

Version 1.0

Sinh viên thực hiện:

21810203 – Nguyễn Đức Cang

21810220 – Hồng Phương Tuấn

21810221 – Phan Thanh Tuấn

Bảng ghi nhận thay đổi tài liệu

|  |  |  |  |
| --- | --- | --- | --- |
| Ngày | Phiên bản | Mô tả | Người thay đổi |
| 16/12/2024 | 1.0 | Phân thích thiết kê ứng dụng LingLeap | 21810220 – Hồng Phương Tuấn |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Mục lục

[1. Tổng Quan Kiến Trúc 5](#_Toc185249020)

[1.1 Frontend 5](#_Toc185249021)

[1.2 Backend 5](#_Toc185249022)

[1.3 Cơ Sở Dữ Liệu 5](#_Toc185249023)

[1.4 Các Dịch Vụ Bên Ngoài 5](#_Toc185249024)

[2. Chi Tiết Các Module và Thành Phần Chính 6](#_Toc185249025)

[2.1 Frontend 6](#_Toc185249026)

[2.2 Backend 8](#_Toc185249027)

[2.3 Cơ Sở Dữ Liệu (Database) 11](#_Toc185249028)

[2.4 Các Mối Quan Hệ Giữa Các Collections 18](#_Toc185249029)

[2.5 Các Module Chính trong Backend 22](#_Toc185249030)

[2.6 Các Module Chính trong Frontend 24](#_Toc185249031)

[2.7 Các Thành Phần Bên Ngoài và Tích Hợp 26](#_Toc185249032)

[2.8 Bảo Mật và Tối Ưu 26](#_Toc185249033)

[2.9 Quy Trình Phát Triển và Triển Khai 27](#_Toc185249034)

[3. Kết Luận 28](#_Toc185249035)

# Tổng Quan Kiến Trúc

Ứng dụng Thi Trực Tuyến được thiết kế dựa trên kiến trúc phân tầng (Layered Architecture) với mục tiêu đảm bảo tính mở rộng, bảo trì dễ dàng, và hiệu suất cao. Hệ thống được chia thành hai phần chính: Frontend và Backend, kết nối với cơ sở dữ liệu MongoDB và các dịch vụ bên ngoài.

## Frontend

* Framework: React.js
* State Management: Redux Toolkit quản lý trạng thái
* Build Tool: Vite
* Libraries/Tools: React Router, Axios, Chart.js/D3.js

## Backend

* Runtime Environment: Node.js
* Framework: Express.js
* Database: MongoDB sử dụng Mongoose cho quản lý schema
* Authentication: JSON Web Tokens (JWT)
* Libraries/Tools: Bcrypt, Helmet, Express-Rate-Limit, Zod validation, socket.io

## Cơ Sở Dữ Liệu

* Database Engine: MongoDB
* Schema Management: Mongoose
* Collections Chính:
* Users & Profiles (users, userProfiles, userStatistics, loginSessions, subscriptions)
* Testing System (tests, questions, testAttempts, testResults, reviews)

## Các Dịch Vụ Bên Ngoài

* Email Service: Google Email Service
* Payment Gateway: Stripe
* Cloud Storage: AWS S3 hoặc MongoDB Atlas (tùy chọn)
* LLM API: Google TTS, OpenAI (gpt-4o-mini), DeepGram STT, LiveKit

# Chi Tiết Các Module và Thành Phần Chính

## Frontend

2.1.1. Structure and Organization

2.1.2. Thành Phần Chính

1. Components:
2. Mô tả: Các thành phần UI tái sử dụng được dùng để xây dựng giao diện người dùng của ứng dụng.
3. Thư Mục: src/components/
4. Các Subcomponents:

* Button:
* delete.jsx, test.attempt.jsx, view.jsx: Các thành phần nút chuyên biệt cho các hành động khác nhau.
* Form:
* Input.jsx, Select.jsx: Các thành phần đầu vào cho form.
* HomeComponent:
* ction.jsx, CourseHighlights.jsx, v.v.: Các thành phần đặc thù cho trang Home.
* Navigation:
* Logout.jsx, MenuPanel.jsx, Navigation.jsx, v.v.: Các thành phần quản lý điều hướng và menu người dùng.
* UI:
* avatar.jsx, button.jsx, input.jsx: Các thành phần UI chung được sử dụng trên toàn ứng dụng.
* Khác:
* Container.jsx, loader.jsx, logo.jsx, PageTransition.jsx, ProtectedRoute.jsx: Các thành phần tiện ích và bố cục khác.

1. Pages:

* Mô tả: Các view cấp cao tương ứng với các route khác nhau trong ứng dụng.
* Thư Mục: src/pages/
* Các Pages Chính:
* Xác Thực:
  + Auth.jsx, Login.jsx, Register.jsx: Xử lý các quy trình xác thực người dùng.
* Quản Lý Câu Hỏi:
  + AddQuestion/: Chứa các thành phần để thêm và quản lý câu hỏi.
* Quản Lý Kiểm Tra:
  + TestAttempt/, TestCreation/, TestResult/, Tests/: Quản lý quá trình tham gia kiểm tra, tạo kiểm tra và kết quả kiểm tra.
* Hồ Sơ Người Dùng:
  + Profile/: Quản lý hồ sơ người dùng và các chi tiết liên quan.
* Khác:
  + AiAssisstant.jsx, Chat.jsx, Contact.jsx, Courses.jsx, Home.jsx, MyQuestions/, NotFound.jsx, viewTestAttempt/: Các trang chức năng và điều hướng khác.

1. Redux

* Mô tả: Quản lý trạng thái toàn cục của ứng dụng sử dụng Redux Toolkit.
* Thư Mục: src/Redux/
* Các Submodules:
* Features:
* appSlice.js: Quản lý các trạng thái toàn cục như cài đặt giao diện.
* authSlice.js: Xử lý trạng thái xác thực, thông tin người dùng và token.
* testAnswerSlice.js: Quản lý câu trả lời của người dùng trong quá trình kiểm tra.
* Services:
* authApi.js, questionApi.js, v.v.: Định nghĩa các tương tác API sử dụng RTK Query cho các tài nguyên khác nhau.
* Cấu Hình Store:
* store.js: Cấu hình store Redux với middleware và reducers.

1. Services

* Tương tác với API backend để thực hiện các thao tác CRUD.
* api.js: Thiết lập Axios instance với cấu hình chung.
* authService.js: Xử lý các yêu cầu liên quan đến authentication.
* testService.js: Xử lý các yêu cầu liên quan đến bài thi.
* paymentService.js: Xử lý các giao dịch thanh toán thông qua Stripe.

1. Utils:

* validators.js: Định nghĩa các hàm kiểm tra dữ liệu nhập vào. Đảm bảo tính toàn vẹn và định dạng đúng cho dữ liệu trong tương tác người dùng
* auth.validator.js
* question.validator.js
* updateUserSchema.validator.js
* helpers.js: Các hàm tiện ích chung được sử dụng trong toàn bộ ứng dụng.

1. Styles:

* Tailwindcss, app.css.

## Backend

2.2.1. Structure and Organization

/backend

│

├── config

│   ├── mongoose.db.js

│   └── config.js

│

├── controllers

│   ├── authController.js

│   ├── userController.js

│   ├── testController.js

│   ├── courseController.js

│   ├── gradingController.js

│   └── adminController.js

│

├── middlewares

│   ├── authMiddleware.js

│   ├── errorHandler.js

│   ├── validationMiddleware.js

│   └── rateLimiter.js

│

├── models

│   ├── user.model.js

│   ├── userProfile.model.js

│   ├── userStatistics.model.js

│   ├── subscription.model.js

│   ├── loginSession.model.js

│   ├── course.model.js

│   ├── test.model.js

│   ├── question.model.js

│   ├── testAttempt.model.js

│   ├── testResult.model.js

│   └── review.model.js

│

├── routes

│   ├── auth.routes.js

│   ├── user.routes.js

│   ├── test.routes.js

│   ├── course.routes.js

│   ├── grading.routes.js

│   └── admin.routes.js

│

├── services

│   ├── emailService.js

│   ├── paymentService.js

│   └── notificationService.js

│

├── utils

│   ├── validators.js

│   ├── helpers.js

│   └── constants.js

│

├── app.js

├── server.js

├── package.json

└── .env

2.2.2. Thành Phần Chính

1. API

* Mô tả: Chứa các file .http định nghĩa các điểm cuối API cho các tài nguyên khác nhau như xác thực, câu hỏi, đánh giá, kiểm tra, v.v.
* Mục đích: Là giao diện cho các tương tác phía client với Backend, xử lý các yêu cầu và phản hồi HTTP.

1. Cấu Hình

* mongoose.db.js: Cấu hình kết nối đến cơ sở dữ liệu MongoDB sử dụng Mongoose.
* Mục đích: Tập trung cấu hình cơ sở dữ liệu, dễ dàng quản lý các môi trường khác nhau (phát triển, sản xuất).

1. Controllers

* Mô tả: Xử lý logic nghiệp vụ cho các tài nguyên khác nhau.
* Controllers Bao Gồm:
  + auth.controller.js: Quản lý các hoạt động liên quan đến xác thực.
  + questions.controller.js & questions.controller.v1.js: Xử lý các thao tác CRUD cho câu hỏi.
  + review.controller.js: Quản lý các chức năng liên quan đến đánh giá.
  + test.controller.js, testAttempt.controller.js, testResult.controller.js: Xử lý việc tạo kiểm tra, nỗ lực kiểm tra và kết quả kiểm tra.
  + user.controller.js: Quản lý các hoạt động liên quan đến người dùng.

1. Tài Liệu

* Mô tả: Chứa các tài liệu chi tiết về các khía cạnh khác nhau của ứng dụng, bao gồm các bước triển khai, sơ đồ, và tài liệu tổng quan.
* Các Tài Liệu Chính:
* app-overview.md: Cung cấp tổng quan về ứng dụng.
* implementation-steps.md & implementation-steps-v1.md: Hướng dẫn chi tiết về việc triển khai các tính năng khác nhau.
* question-schema.md: Định nghĩa sơ đồ cho câu hỏi.
* swagger/index.js: Cấu hình Swagger cho tài liệu API.
* use-case-diagram.md: Sơ đồ minh họa các trường hợp sử dụng.
* use-cases/: Các tài liệu trường hợp sử dụng chi tiết cho các vai trò người dùng khác nhau.

1. Models

* Mô tả: Định nghĩa cấu trúc dữ liệu sử dụng các sơ đồ Mongoose.
* Các Models Chính:
* Course, Question, Subscription, Test, TestAttempt, TestResult, User, UserProfile: Đại diện cho các thực thể cốt lõi của ứng dụng.
* Seed Data: Chứa các script để seed cơ sở dữ liệu với dữ liệu ban đầu.

1. Middleware

* Mô tả: Chứa các hàm middleware xử lý yêu cầu trước khi chúng tới các controllers.
* Các Middleware Chính:
* audio.middleware.js: Xử lý liên quan đến âm thanh.
* auth.middleware.js: Quản lý kiểm tra xác thực.
* clientInfo.middleware.js: Lấy thông tin khách hàng từ yêu cầu.
* error.middleware.js: Xử lý lỗi tập trung.

1. Routes

* Mô tả: Định nghĩa các điểm cuối và liên kết chúng với các controllers tương ứng.
* Các Routes Chính:
* auth.route.js: Các routes liên quan đến xác thực.
* question.route.js: Các routes quản lý câu hỏi.
* test.route.js: Các routes quản lý kiểm tra.
* Các routes khác xử lý đánh giá, seed dữ liệu, người dùng và chức năng giọng nói.

1. Services

* Mô tả: Chứa logic nghiệp vụ có thể tái sử dụng giữa các controllers.
* Các Services Chính:
* questions.service.js: Logic nghiệp vụ cho câu hỏi.
* testAttempt.service.js: Xử lý các nỗ lực kiểm tra.
* testResult.service.js: Quản lý kết quả kiểm tra.

1. Utilities

* similarity.js: Cung cấp các hàm tiện ích, có thể dùng để so sánh văn bản hoặc dữ liệu khác.

1. Validators

* Mô tả: Chứa logic xác thực cho các yêu cầu đến nhằm đảm bảo tính toàn vẹn của dữ liệu.
* Các Validators Chính:
* auth.validator.js: Xác thực dữ liệu xác thực.
* question.validator.js & question.validator.v1.js: Xác thực dữ liệu liên quan đến câu hỏi.
* review.validator.js, test.validator.js, user.validator.js: Xác thực dữ liệu đánh giá, kiểm tra và người dùng tương ứng.

## Cơ Sở Dữ Liệu (Database)

Hệ thống sử dụng MongoDB với các collections chính sau đây:

2.3.1. Users (users)

1. Mục Đích: Quản lý thông tin cơ bản của người dùng, bao gồm xác thực và vai trò trong hệ thống.
2. Thuộc Tính: \_id, email, username, password, firstName, lastName, dateOfBirth, phoneNumber, address , avatar, isVerified, isDeleted, role, userProfile, userStatistics, createdAt, updatedAt
3. Indexes: email, username, role

2.3.2. UserProfiles (userProfiles)

1. Mục Đích: Lưu trữ thông tin chi tiết về người dùng, bao gồm tiến trình học tập và kết nối xã hội.
2. Thuộc Tính:\_id, user, testTaken, testScore, achievements, preferences, learningGoals, targetExams, studyPlan, progressTracking, socialLinks, socialConnections, feedbackProvided, createdAt, updatedAt
3. Indexes: user, socialConnections.studyBuddies

2.3.3. UserStatistics (userStatistics)

1. Mục Đích: Theo dõi số liệu thống kê của người dùng về các bài kiểm tra đã làm và điểm số.
2. Thuộc Tính: \_id, User, testsCompleted, averageScores, questionStats, timeStats, progressOverTime, lastUpdated, createdAt, updatedAt
3. Indexes: user

2.3.4. LoginSessions (loginSessions)

1. Mục Đích: Quản lý các phiên đăng nhập của người dùng, bao gồm thông tin thiết bị và token.
2. Thuộc Tính: \_id, user, deviceInfo, ipAddress, refreshToken, isActive, lastActivity, expiresAt, refreshTokenUsed, clientId, createdAt, updatedAt
3. Indexes: refreshToken, user

2.3.5. Subscriptions (subscriptions)

1. Mục Đích: Quản lý các gói đăng ký và lịch sử thanh toán của người dùng.
2. Thuộc Tính:

\_id, user, plan, paymentHistory, isActive, createdAt, updatedAt

1. Indexes: user, plan.type, isActive

2.3.6. Tests (tests)

1. Mục Đích: Định nghĩa cấu trúc và nội dung của các bài kiểm tra.
2. Thuộc Tính: \_id, title, description, duration, sections, passages, totalPossibleScore, passingScore, difficulty, testType, createdBy, isPublished, averageRating, totalReviews, participantCount,
3. Thuộc Tính Phụ:

Các phương thức như updateAverageRating(), publish(), unpublish()

1. Indexes: createdBy, isPublished, difficulty,

2.3.7. Questions (questions)

1. Mục Đích: Lưu trữ các câu hỏi thuộc các bài kiểm tra khác nhau.
2. Thuộc Tính: \_id, Section, questionText, type, difficulty, points, createdBy,
3. Indexes: Section, Type, difficulty

2.3.8. TestAttempts (testAttempts)

1. Mục Đích: Quản lý các lần thử làm bài kiểm tra của người dùng.
2. Thuộc Tính: \_id, User, Test, startTime, endTime, maxEndTime, status, result, totalScore, answers,
3. Thuộc Tính Phụ:

Các phương thức như start(), isTimeUp(), complete(), calculateResult(), calculateScore(), sanitizeJsonString()

1. Indexes: User, Test, status

2.3.9. TestResults (testResults)

1. Mục Đích: Lưu trữ và phân tích kết quả chi tiết của các lần làm bài kiểm tra.
2. Thuộc Tính: \_id, testAttempt, user, test, totalScore, maxTotalScore, scorePercentage, totalQuestions, sectionScores, questionScores, writingQuestionResults, speakingResult, feedback, gradedBy, gradedAt
3. Thuộc Tính Phụ:

Phương thức addFeedback()

1. Indexes: testAttempt, user, test

2.3.10. Reviews (reviews)

1. Mục Đích: Quản lý đánh giá và phản hồi từ người dùng về các bài kiểm tra.
2. Thuộc Tính: \_id, Test, User, Rating, Comment, createdAt, updatedAt,
3. Thuộc Tính Phụ:

Middleware post("save") để cập nhật averageRating của bài kiểm tra sau khi lưu review

1. Indexes: Test, user

## Các Mối Quan Hệ Giữa Các Collections

2.4.1. Mô Tả Mối Quan Hệ

A diagram of a diagram

Description automatically generated with medium confidence

1. User → UserProfile (1:1)

* Mỗi người dùng (User) có một hồ sơ chi tiết (UserProfile).
* Tham chiếu hai chiều cho phép truy vấn nhanh chóng thông tin hồ sơ từ người dùng và ngược lại.
* Khi người dùng bị xóa, hồ sơ của họ cũng sẽ bị xóa (cascade deletion).

1. User → UserStatistics (1:1)

* Mỗi người dùng có một bộ thống kê (UserStatistics) liên quan đến hoạt động học tập và bài kiểm tra đã làm.
* Tham chiếu hai chiều để cập nhật và truy xuất dữ liệu thống kê dễ dàng.
* Tự động tạo bộ thống kê khi tạo người dùng mới.

1. User → Subscription (1:1)

* Mỗi người dùng có thể có một gói đăng ký (Subscription) duy nhất.
* Quản lý các gói đăng ký, lịch sử thanh toán và trạng thái hoạt động.
* Kiểm tra quyền truy cập vào các tính năng dựa trên gói đăng ký.

1. User → LoginSession (1:N)

* Một người dùng có thể có nhiều phiên đăng nhập (LoginSession) khác nhau.
* Ghi nhận thông tin thiết bị và địa chỉ IP của từng phiên đăng nhập.
* Hỗ trợ việc đăng nhập từ nhiều thiết bị.

1. User → TestAttempt (1:N)

* Một người dùng có thể có nhiều lần thử làm bài kiểm tra (TestAttempt).
* Theo dõi tiến trình và thời gian làm bài của từng lần thử.

1. User → Review (1:N)

* Một người dùng có thể viết nhiều đánh giá (Review) cho các bài kiểm tra khác nhau.
* Tính toán đánh giá trung bình (averageRating) cho mỗi bài kiểm tra dựa trên các review.

1. User → Test (1:N As Creator)

* Một người dùng (giáo viên hoặc admin) có thể tạo nhiều bài kiểm tra (Test).
* Quản lý các bài kiểm tra do mình tạo.

1. Test → Question (1:N)

* Một bài kiểm tra (Test) chứa nhiều câu hỏi (Question).
* Câu hỏi có thể thuộc nhiều phần khác nhau của bài kiểm tra.

1. Test → TestAttempt (1:N)

* Một bài kiểm tra có thể được thực hiện nhiều lần thử bởi nhiều người dùng khác nhau.

1. Test → Review (1:N)

* Một bài kiểm tra có thể nhận được nhiều đánh giá từ người dùng.

1. TestAttempt → TestResult (1:1)

* Mỗi lần thử làm bài kiểm tra (TestAttempt) sẽ tạo ra một kết quả chi tiết (TestResult).
* Kết quả chứa điểm số, phân tích và phản hồi từ các phần chấm điểm thủ công.

1. Inheritance in Questions

* Question là lớp cơ sở, từ đó kế thừa các loại câu hỏi cụ thể như MultipleChoiceQuestion, SingleChoiceQuestion, EssayQuestion, v.v...
* Các lớp con mở rộng các thuộc tính đặc thù cho từng loại câu hỏi.

## Các Module Chính trong Backend

2.5.1. User Management

* Mô Tả:
* Quản lý thông tin người dùng, bao gồm đăng ký, đăng nhập, xác thực email, quản lý mật khẩu.
* Quản lý hồ sơ cá nhân (UserProfile) và thống kê (UserStatistics).
* Xử lý các phiên đăng nhập và quản lý token JWT.
* Chức Năng:
* Đăng ký: Người dùng tạo tài khoản mới, nhận email xác thực.
* Đăng nhập: Người dùng đăng nhập và nhận token access và refresh.
* Quản lý Profile: Người dùng cập nhật thông tin cá nhân, học tập.
* Quản lý Subscription: Theo dõi và cập nhật các gói đăng ký, thanh toán qua Stripe.
* Logout: Chấm dứt phiên đăng nhập hiện tại.

2.5.2. Test Management

* Mô Tả:
* Quản lý toàn bộ vòng đời của bài kiểm tra, từ tạo, chỉnh sửa, xuất bản đến xóa.
* Quản lý các section và câu hỏi liên quan đến bài kiểm tra.
* Chức Năng:
* Tạo Test: Định nghĩa tiêu đề, mô tả, thời gian, các phần và câu hỏi.
* Chỉnh sửa Test: Cập nhật thông tin bài kiểm tra và cấu trúc câu hỏi.
* Xuất bản/Ẩn test: Kiểm soát bài kiểm tra có sẵn cho người dùng hay không.
* Xóa Test: Loại bỏ bài kiểm tra khỏi hệ thống.

2.5.3. Question Management

* Mô Tả:
* Quản lý ngân hàng câu hỏi, bao gồm tạo, chỉnh sửa, xóa và phân loại câu hỏi theo loại và độ khó.
* Chức Năng:
* Tạo Question: Thêm câu hỏi mới với các loại khác nhau (multiple choice, essay, etc.).
* Chỉnh sửa Question: Cập nhật nội dung hoặc cấu trúc của câu hỏi.
* Xóa Question: Loại bỏ câu hỏi khỏi ngân hàng.
* Phân loại và đặt độ khó: Quản lý loại câu hỏi và thiết lập mức độ khó cho từng câu.
* Import/Export: Nhập khẩu hoặc xuất khẩu câu hỏi từ/đến các định dạng khác nhau.

2.5.4. Test Attempt and Result

* Mô Tả:
* Quản lý các lần thử làm bài kiểm tra của người dùng và lưu kết quả chi tiết sau khi hoàn thành.
* Chức Năng:
* Bắt đầu TestAttempt: Khởi tạo một lần thử mới khi người dùng bắt đầu bài kiểm tra.
* Theo dõi tiến trình: Theo dõi thời gian, lưu lại câu trả lời, quản lý trạng thái của TestAttempt.
* Hoàn thành TestAttempt: Xác nhận khi người dùng hoàn thành và nộp bài thi.
* Tính toán TestResult: Chấm điểm tự động các câu hỏi khách quan và chuẩn bị dữ liệu cho chấm điểm thủ công.

2.5.5. Grading and Assessment

* Mô Tả:
* Xử lý chấm điểm tự động cho các câu hỏi khách quan và quản lý chấm điểm thủ công cho các câu hỏi chủ quan.
* Chức Năng:
* Auto-grading: Tự động chấm điểm các câu hỏi như multiple choice, true/false.
* Manual grading: Giao diện cho giáo viên/chấm điểm để đánh giá các câu hỏi như essay, speaking.
* Feedback: Cung cấp phản hồi chi tiết cho người dùng dựa trên kết quả chấm điểm.

2.5.6. Administration

* Mô Tả:
* Quản trị hệ thống từ góc nhìn của Admin, bao gồm quản lý người dùng, kiểm soát nội dung và cấu hình hệ thống.
* Chức Năng:
* Quản lý người dùng: Thêm, sửa, xóa người dùng, phân quyền.
* Quản lý nội dung: Kiểm tra chất lượng nội dung, duyệt bài kiểm tra mới.
* Giám sát hệ thống: Theo dõi hiệu suất, xử lý sự cố.
* Báo cáo: Tạo và xem các báo cáo thống kê về hoạt động của hệ thống.

## Các Module Chính trong Frontend

2.6.1. Authentication

* Mô Tả:
* chức năng đăng nhập, đăng ký, xác thực và quản lý token.
* Chức Năng:
* Login: Giúp người dùng nhập thông tin và nhận token JWT.
* Register: Đăng ký tài khoản mới và gửi email xác thực.
* Forgot Password: Khôi phục mật khẩu thông qua email.
* Protected Routes: Bảo vệ các route yêu cầu xác thực.

2.6.2. Dashboard

* Mô Tả:
* Cung cấp giao diện dành cho các vai trò khác nhau (Admin, Teacher, Student) để quản lý và truy cập các chức năng tương ứng.
* Chức Năng:
* AdminDashboard: Quản lý người dùng, hệ thống, và nội dung.
* TeacherDashboard: Tạo và quản lý bài kiểm tra, xem kết quả học sinh.
* StudentDashboard: Xem bài kiểm tra đã làm, kết quả, và theo dõi tiến trình học tập.

2.6.3. Test Management

* Mô Tả:
* giao diện để người dùng tham gia và quản lý các bài kiểm tra.
* Chức Năng:
* TestList: Hiển thị danh sách các bài kiểm tra có sẵn.
* TestDetail: Xem chi tiết bài kiểm tra, bao gồm mô tả, các phần và số điểm.
* TestTaking: Giao diện làm bài kiểm tra, bao gồm hiển thị câu hỏi và nhập câu trả lời.
* TestResult: Hiển thị kết quả sau khi hoàn thành bài kiểm tra, bao gồm điểm số và feedback.

2.6.4. Profile Management

* Mô Tả:
* thông tin cá nhân và các thiết lập của người dùng.
* Chức Năng:
* UserProfile: Cập nhật thông tin cá nhân, theo dõi tiến trình học tập.
* Preferences: Thiết lập thông báo, nhắc nhở học tập.
* LearningGoals: Đặt và theo dõi mục tiêu học tập.

2.6.5. Course Management

* Mô Tả:
* khóa học, bao gồm tạo, chỉnh sửa và tham gia khóa học.
* Chức Năng:
* CourseList: Hiển thị danh sách các khóa học có sẵn.
* CourseDetail: Xem chi tiết khóa học, bao gồm bài học và bài kiểm tra.
* CourseManagement: Công cụ quản lý khóa học dành cho giáo viên và admin.

2.6.6. Common Components

* Mô Tả:
* Các thành phần giao diện chung được sử dụng trong nhiều phần của ứng dụng.
* Chức Năng:
* Navbar: Thanh điều hướng trên đầu trang, tùy thuộc vào vai trò người dùng.
* Footer: Chân trang cho thông tin liên hệ và các liên kết quan trọng.
* ProtectedRoute: Bảo vệ các route yêu cầu người dùng đã đăng nhập.

## Các Thành Phần Bên Ngoài và Tích Hợp

2.7.1. Email Service (Google Email Sender)

* Chức Năng:
* Gửi email xác thực tài khoản mới.
* Gửi thông báo khôi phục mật khẩu.
* Gửi thông báo đăng ký và thanh toán.

2.7.2. Payment Gateway (Stripe)

* Chức Năng:
* Xử lý các giao dịch thanh toán cho các gói đăng ký.
* Quản lý lịch sử thanh toán.
* Tích hợp với subscriptions collection để cập nhật trạng thái đăng ký.

2.7.3. Cloud Storage (AWS S3 / MongoDB Atlas)

* Chức Năng:
* Lưu trữ các file media như ảnh đại diện, âm thanh trong câu hỏi, video bài thi.
* Quản lý tài nguyên lớn và đảm bảo khả năng mở rộng.

2.7.4. LLM API

* Chức Năng:
  + Truy cập và tương tác với các mô hình ngôn ngữ lớn: Google text to speech, OpenAI chat generation….

## Bảo Mật và Tối Ưu

2.8.1. Bảo Mật

* Xác Thực và Phân Quyền:
* Sử dụng JWT cho xác thực và phân quyền người dùng.
* Role-Based Access Control (RBAC) để kiểm soát quyền truy cập theo vai trò (user, teacher, admin).
* Bảo Vệ V against Attacks:
* Sử dụng Helmet để bảo vệ ứng dụng khỏi các lỗ hổng thông thường.
* Express-Rate-Limit để giới hạn số lượng yêu cầu từ một IP.
* Csurf để bảo vệ chống lại các cuộc tấn công CSRF.
* Data Validation:
* Sử dụng Zod validator để kiểm tra và xác thực dữ liệu nhập vào.

2.8.2. Tối Ưu Hiệu Suất

* Indexing: Sử dụng các chỉ mục (indexes) trên các trường thường xuyên truy vấn để tăng tốc độ truy vấn.
* Caching: Sử dụng Redis hoặc các giải pháp caching khác để lưu trữ tạm thời dữ liệu thường xuyên truy cập.
* Load Balancing: Triển khai cân bằng tải để phân phối các yêu cầu đến nhiều server backend.
* Minimizing Payloads: Giảm kích thước payload bằng cách sử dụng các phương pháp nén dữ liệu và chỉ gửi các trường cần thiết.

## Quy Trình Phát Triển và Triển Khai

2.9.1. Development Workflow

* Version Control: Sử dụng Git cho quản lý mã nguồn.
* Branching Strategy: Sử dụng chiến lược branching như GitFlow để quản lý các phiên bản phát triển và phát hành.
* Continuous Integration/Continuous Deployment (CI/CD): Thiết lập các pipeline CI/CD với các công cụ như GitHub Actions, Jenkins để tự động hóa quá trình kiểm tra và triển khai.

2.9.2. Testing

* Unit Testing: Viết các bài kiểm tra đơn vị cho các module backend và frontend.
* Integration Testing: Kiểm tra tương tác giữa các module trong hệ thống.
* End-to-End Testing: Sử dụng các công cụ như Cypress hoặc Selenium để kiểm tra toàn bộ hệ thống từ đầu đến cuối.

2.9.3. Monitoring và Logging

* Monitoring: Sử dụng các công cụ như Prometheus và Grafana để giám sát hiệu suất và sức khỏe của hệ thống.
* Logging: Sử dụng Winston hoặc Bunyan để ghi lại các log sự kiện và lỗi cho việc phân tích và gỡ lỗi.

# Kết Luận

Kiến trúc hệ thống của ứng dụng Thi Trực Tuyến được thiết kế với mục tiêu đảm bảo tính linh hoạt, bảo trì dễ dàng, và khả năng mở rộng cao. Bằng cách chia hệ thống thành các module rõ ràng và sử dụng các công nghệ hiện đại, ứng dụng có thể đáp ứng hiệu quả các yêu cầu của người dùng và dễ dàng phát triển thêm các tính năng mới trong tương lai.

Việc phân chia rõ ràng giữa frontend và backend, cùng với việc sử dụng các dịch vụ và thư viện mạnh mẽ, giúp hệ thống duy trì hiệu suất cao và đảm bảo bảo mật cho dữ liệu người dùng. Hệ thống cũng hỗ trợ quản lý nội dung và người dùng một cách hiệu quả, đồng thời cung cấp các công cụ phân tích và báo cáo mạnh mẽ để quản trị viên có thể theo dõi và cải thiện hệ thống liên tục.