

TRƯỜNG ĐẠI HỌC THỦY LỢI  
KHOA CÔNG NGHỆ THÔNG TIN

# BÁO CÁO MÔN HỌC THỰC TẬP TỐT NGHIỆP CNTT

Võ Việt Anh – 63CNTT2

Công ty CP Công nghệ Aido - 12/24/99, Kim Đồng, Giáp Bát, Hoàng Mai, HN

# MỤC LỤC

<b>LỜI NÓI ĐẦU .....</b>	<b>3</b>
<b>NHỮNG KẾT QUẢ THU HOẠCH ĐƯỢC SAU ĐỢT THỰC TẬP .....</b>	<b>4</b>
<b>I. VỀ KIẾN THỨC.....</b>	<b>4</b>
<b>Tuần 1 (17-2 – 23/2): Tìm hiểu và cài đặt công nghệ sử dụng: .....</b>	<b>4</b>
<b>Tuần 2 (24/2 – 2/3): Xây dựng project với API cơ bản: .....</b>	<b>8</b>
<b>Tuần 3 (3/3 – 9/3): Phân quyền, hoàn thiện API với JWT .....</b>	<b>14</b>
<b>Tuần 4 (10/3 - 16/3): Public API – Hợp tác với FrontEnd .....</b>	<b>16</b>
<b>II. VỀ KỸ NĂNG .....</b>	<b>17</b>
<b>III. PHẨM CHẤT ĐẠO ĐỨC NGHỀ NGHIỆP.....</b>	<b>18</b>
<b>IV. NHỮNG THU HOẠCH BỔ ÍCH NHẤT ĐỐI VỚI BẢN THÂN .....</b>	<b>18</b>
<b>V. NHỮNG HẠN CHẾ CỦA BẢN THÂN:.....</b>	<b>19</b>
<b>VI. KIẾN NGHỊ.....</b>	<b>19</b>
<b>VIII. ĐÁNH GIÁ CỦA GVHD TẠI TRƯỜNG ĐHTL.....</b>	<b>19</b>

## **LỜI NÓI ĐẦU**

Trong quá trình thực tập tốt nghiệp, em đã nhận được sự giúp đỡ rất nhiệt tình của Trường Đại học Thủy Lợi, Công ty cổ phần công nghệ Aido, thầy Nguyễn Văn Nam - giáo viên hướng dẫn thực tập

Để có thể hoàn thành tốt đợt thực tập tốt nghiệp, em xin gửi lời cảm ơn chân thành đến Ban giám hiệu Trường Đại học Thủy Lợi, thầy Nguyễn Văn Nam và quý công ty đã tận tình giảng dạy và cung cấp những kiến thức cần thiết, bổ ích để em có thể hoàn thành tốt quá trình học tập và thực tập của mình.

Em xin gửi lời cảm ơn chân thành đến thầy Nguyễn Văn Nam đã trực tiếp, hỗ trợ và giúp đỡ em trong quá trình thực tập.

## **THÔNG TIN CHUNG**

### Thông tin cá nhân:

- Họ tên sinh viên: Võ Việt Anh
- Mã SV: 2151062715
- Lớp: 63CNTT2

### Thông tin doanh nghiệp:

- Tên cơ sở thực tập: Công ty CP Công nghệ Aido
- Địa chỉ thực tập: Số 12/24/99, Kim Đồng, Giáp Bát, Hoàng Mai, HN
- Người hướng dẫn tại doanh nghiệp:
  - Họ tên: Nguyễn Văn Nam
  - Vị trí công tác: PM
  - Điện thoại liên lạc: 0988862626
  - Email: namnv.dev@gmail.com

# NHỮNG KẾT QUẢ THU HOẠCH ĐƯỢC SAU ĐỢT THỰC TẬP

## I. VỀ KIẾN THỨC

### Các kiến thức được doanh nghiệp yêu cầu tự học:

- Tìm hiểu khái niệm, các bước xây dựng một API backend.
- Tìm hiểu về NodeJS, ExpressJS, cơ sở dữ liệu (MySQL, MongoDB...), và các công nghệ liên quan (JWT, RESTful API...).

### Các phần mềm sử dụng trong đợt thực tập:

- Zalo để trao đổi công việc.
- Visual Studio Code để viết code backend.
- Git/Github để quản lý code.
- Postman để test API.
- Công cụ quản lý cơ sở dữ liệu (MySQL Community Server - Dbeaver).

**Link Github:** <https://github.com/quyenbui220103/TLCOIN/tree/backend>

### Tuần 1 (17-2 – 23/2): Tìm hiểu và cài đặt công nghệ sử dụng:

*Làm quen với Backend, API, RESTful API:*

- Backend, hay còn gọi là "phía máy chủ", là phần "hậu trường" của một ứng dụng web, đóng vai trò xử lý logic và dữ liệu.
- Nó bao gồm máy chủ (server), cơ sở dữ liệu (database) và các ứng dụng chạy trên máy chủ.
- Backend chịu trách nhiệm xử lý các yêu cầu từ người dùng, tương tác với cơ sở dữ liệu và trả kết quả về cho frontend.
- Ví dụ, khi bạn đăng nhập vào một trang web, backend sẽ xác thực thông tin đăng nhập của bạn và kiểm tra xem bạn có quyền truy cập vào trang web hay không.

*API (Application Programming Interface):*

- API là giao diện lập trình ứng dụng, cho phép các ứng dụng khác nhau giao tiếp và trao đổi dữ liệu với nhau.
- Nó định nghĩa các quy tắc và phương thức mà các ứng dụng có thể sử dụng để truy cập và sử dụng các chức năng của nhau.
- API giúp đơn giản hóa việc tích hợp các ứng dụng và cho phép xây dựng các ứng dụng phức tạp từ các thành phần nhỏ hơn.
- Ví dụ, một ứng dụng thời tiết có thể cung cấp API cho phép các ứng dụng khác truy cập và hiển thị thông tin thời tiết.

➤ RESTful API:

- RESTful API là một kiểu kiến trúc API phổ biến, tuân theo các nguyên tắc của kiến trúc REST (Representational State Transfer).
- Nó sử dụng các phương thức HTTP (GET, POST, PUT, DELETE...) để thực hiện các thao tác CRUD (Create, Read, Update, Delete) trên dữ liệu.
- RESTful API giúp xây dựng các API dễ hiểu, dễ sử dụng và dễ mở rộng.
- Ví dụ, một RESTful API quản lý sản phẩm có thể có các endpoint như /products (GET để lấy danh sách sản phẩm, POST để tạo sản phẩm mới), /products/{id} (GET để lấy thông tin sản phẩm theo ID, PUT để cập nhật sản phẩm, DELETE để xóa sản phẩm).

*NodeJS và ExpressJS:*

1. NodeJS

- NodeJS là một môi trường chạy JavaScript phía server, được xây dựng trên nền tảng V8 JavaScript engine của Google Chrome.
- Nó cho phép sử dụng JavaScript để xây dựng các ứng dụng backend, giúp đơn giản hóa việc phát triển web full-stack.
- NodeJS có hiệu năng cao và khả năng mở rộng tốt, phù hợp cho việc xây dựng các ứng dụng web thời gian thực (real-time).

Ưu điểm của Nodejs là nó dùng ngôn ngữ Javascript, mà Javascript lại là ngôn ngữ rất phổ biến ở phía frontend, nên khi dùng nodejs chúng ta có thể dùng 1 ngôn ngữ để làm fullstack.

2. ExpressJS:

- Đây là một framework web phổ biến cho NodeJS, cung cấp các công cụ và tính năng để xây dựng các ứng dụng web và API một cách nhanh chóng và dễ dàng.
- Nó giúp đơn giản hóa việc định tuyến (routing), xử lý yêu cầu (request) và trả lời (response), quản lý middleware và nhiều tác vụ khác.
- ExpressJS là một framework linh hoạt và có cộng đồng hỗ trợ lớn.
- Expressjs giúp đơn giản hóa việc xây dựng các api, giúp chúng ta không cần phải viết quá nhiều code thuần nodejs.

*Hệ quản trị cơ sở dữ liệu MySQL:*

- MySQL là một hệ quản trị cơ sở dữ liệu quan hệ (RDBMS) mã nguồn mở, được sử dụng rộng rãi trong các ứng dụng web và các hệ thống quản lý dữ liệu khác. Nó lưu trữ dữ liệu trong các bảng, cho phép bạn truy vấn, cập nhật và quản lý dữ liệu một cách hiệu quả.

- MySQL có tính ổn định, hiệu năng cao và dễ sử dụng, là lựa chọn phổ biến cho các ứng dụng web nhỏ và lớn.
- Nó hỗ trợ nhiều tính năng mạnh mẽ, bao gồm giao dịch, khóa ngoại và các chỉ mục, giúp đảm bảo tính toàn vẹn và hiệu suất của dữ liệu.
- MySQL được sử dụng rộng rãi trong các ứng dụng web dựa trên PHP, Python, NodeJS và các ngôn ngữ lập trình khác.

*Cài đặt môi trường làm việc:*

## 1. Download NodeJS:

- Truy cập trang web chính thức của NodeJS tại: [nodejs.org](https://nodejs.org).
- Tải xuống phiên bản NodeJS phù hợp: Chọn phiên bản NodeJS phù hợp với hệ điều hành của bạn (Windows, macOS hoặc Linux). Nên chọn phiên bản LTS (Long Term Support) để đảm bảo tính ổn định.
- Cài đặt NodeJS: Chạy trình cài đặt và làm theo hướng dẫn. NPM sẽ được cài đặt cùng với NodeJS.
- Kiểm tra cài đặt: Mở cửa sổ dòng lệnh (Command Prompt trên Windows hoặc Terminal trên macOS/Linux) và chạy lệnh `node -v` và `npm -v` để kiểm tra phiên bản NodeJS và NPM đã được cài đặt thành công

## 2. Cài đặt NPM và Express:

NPM là trình quản lý gói mặc định cho môi trường NodeJS. Nó đóng vai trò trung tâm trong việc quản lý các thư viện và công cụ (gói) được sử dụng trong các dự án NodeJS. Với NPM, bạn có thể dễ dàng:

- Cài đặt gói: Tìm kiếm và cài đặt các gói từ kho lưu trữ NPM, một kho lưu trữ trực tuyến khổng lồ chứa hàng ngàn thư viện và công cụ JavaScript.
- Quản lý phụ thuộc: Theo dõi và quản lý các phụ thuộc của dự án, đảm bảo rằng tất cả các gói cần thiết đều được cài đặt và tương thích với nhau.
- Cập nhật và gỡ cài đặt gói: Dễ dàng cập nhật các gói lên phiên bản mới nhất hoặc gỡ cài đặt các gói không cần thiết.
- Tạo và chia sẻ gói: Tạo các gói của riêng bạn và chia sẻ chúng với cộng đồng thông qua kho lưu trữ NPM.

NPM giúp đơn giản hóa quá trình phát triển NodeJS, cho phép bạn tập trung vào việc viết mã ứng dụng thay vì lo lắng về việc quản lý các thư viện:

- Khởi tạo dự án NPM: Chạy lệnh `npm init` để khởi tạo một dự án NPM mới. Lệnh này sẽ tạo một tệp `package.json`.
- Cài đặt ExpressJS: Chạy lệnh `npm install express` để cài đặt ExpressJS và thêm nó vào danh sách các phụ thuộc của dự án trong tệp `package.json`.

- Kiểm tra cài đặt: Sau khi cài đặt thành công, kiểm tra bằng cách xem tệp package.json hoặc thư mục node\_modules để đảm bảo rằng ExpressJS đã được cài đặt

### 3. Cài đặt MySQL Community Server và kết nối qua Dbeaver:

#### 3.1 Cài đặt MySQL Community Server:

- Tải xuống : Truy cập trang web chính thức của MySQL: [dev.mysql.com](https://dev.mysql.com).
- Tìm đến phần "Downloads" và chọn "MySQL Community Server".
- Chọn hệ điều hành của bạn và tải xuống trình cài đặt phù hợp.
- Cài đặt: Chạy trình cài đặt đã tải xuống.
- Làm theo hướng dẫn trên màn hình.
- Lưu ý: Trong quá trình cài đặt, bạn sẽ cần: Thiết lập mật khẩu cho tài khoản root. Hãy ghi nhớ mật khẩu này.
- Chọn các thành phần cần cài đặt (ví dụ: MySQL Server, MySQL Workbench).
- Hoàn tất quá trình cài đặt.

#### 3.2 Dbeaver:

DBeaver là một công cụ quản lý cơ sở dữ liệu đa nền tảng miễn phí và mã nguồn mở. Nó hỗ trợ nhiều loại cơ sở dữ liệu khác nhau, bao gồm MySQL, PostgreSQL, Oracle, SQL Server, v.v. DBeaver cung cấp giao diện đồ họa trực quan, giúp bạn dễ dàng:

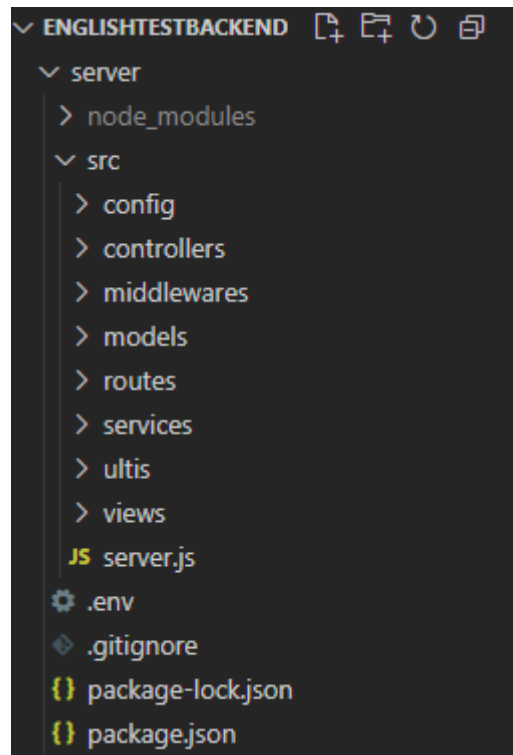
- Kết nối đến cơ sở dữ liệu.
- Truy vấn và chỉnh sửa dữ liệu.
- Quản lý lược đồ cơ sở dữ liệu.

#### Kết nối MySQL với Dbeaver:

- Mở DBeaver: Khởi chạy ứng dụng DBeaver.
- Tạo kết nối mới: Trong DBeaver, nhấp vào biểu tượng "New Database Connection" (hoặc vào menu "File" -> "New" -> "Database Connection").
- Chọn "MySQL" từ danh sách các loại cơ sở dữ liệu.
- Nhấn "Next".
- Nhập thông tin kết nối: Host: localhost (nếu MySQL được cài đặt trên máy tính của bạn). Hoặc IP của máy chủ MySQL.
- Port: 3306 (cổng mặc định của MySQL).
- Username: root (hoặc tên người dùng khác).
- Password: Mật khẩu bạn đã thiết lập khi cài đặt MySQL.
- Nhấn "Test Connection" để kiểm tra kết nối.
- Nếu kết nối thành công, nhấn "Finish".

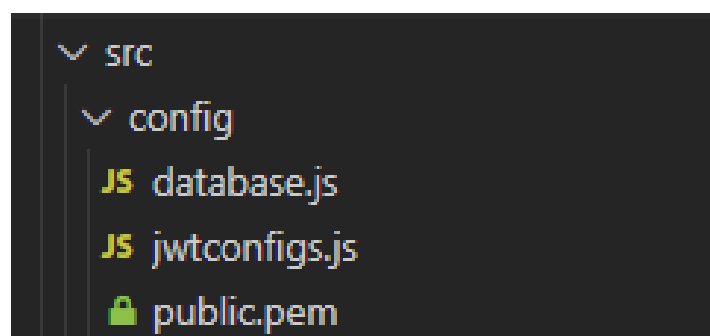
## Tuần 2 (24/2 – 2/3): Xây dựng project với API cơ bản:

### 1. Thiết lập cấu trúc dự án:



- config/

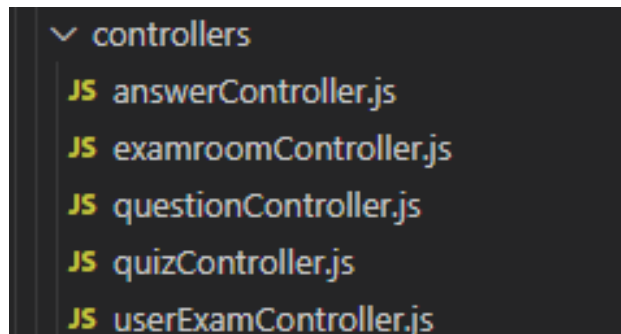
Thư mục config chứa các file cấu hình của ứng dụng, như cấu hình kết nối cơ sở dữ liệu, các biến môi trường, các cài đặt toàn cục, và các thông tin khác mà ứng dụng cần để hoạt động đúng. Việc giữ các cấu hình này trong thư mục riêng giúp cho việc quản lý và thay đổi các thông số trở nên dễ dàng hơn mà không ảnh hưởng đến mã nguồn chính





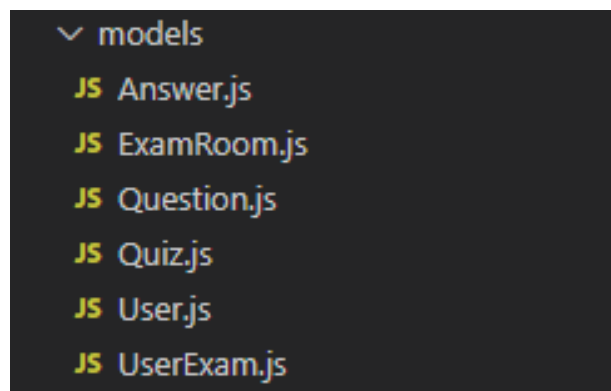
- controllers/

Thư mục controllers chứa các file Controller, nơi chịu trách nhiệm xử lý các logic nghiệp vụ cho từng endpoint của API. Controllers thực hiện việc nhận yêu cầu từ người dùng thông qua các đường dẫn (routes), sau đó gọi các hàm trong Services để thực hiện các tác vụ nghiệp vụ, từ đó xử lý dữ liệu và trả kết quả về cho người dùng. Đây là thành phần trung gian giữa Routes và Services, giúp tách biệt các logic xử lý nghiệp vụ khỏi phần giao tiếp với người dùng và HTTP. Một controller có thể đảm nhận nhiều endpoint khác nhau hoặc nhiều controller có thể cùng xử lý một nhóm các endpoint liên quan. Điều này giúp phân chia mã nguồn một cách hợp lý, dễ dàng mở rộng và bảo trì.



- models/

Thư mục models chứa các file Model, đại diện cho cấu trúc dữ liệu của các đối tượng trong ứng dụng. Mỗi model thông thường tương ứng với một bảng trong cơ sở dữ liệu, giúp mô tả chi tiết các thuộc tính và các mối quan hệ giữa các đối tượng dữ liệu trong ứng dụng.



- routes/

Thư mục routes chứa các file Route, nơi định nghĩa các đường dẫn API (URL) và ánh xạ chúng đến các hàm xử lý trong Controllers. Các routes xác định cách ứng dụng API phản ứng với các yêu cầu HTTP từ người dùng, dựa trên phương thức HTTP (GET, POST, PUT, DELETE,...) và đường dẫn URL. Mỗi route sẽ ánh xạ một hoặc nhiều endpoint đến các hàm xử lý cụ thể trong các controllers đã được định nghĩa trước đó. Bằng cách phân chia các route theo từng phần của ứng dụng, bạn có thể dễ dàng quản lý và thay đổi các endpoint mà không làm ảnh hưởng đến các phần khác trong ứng dụng. Các routes đóng vai trò rất quan trọng trong việc duy trì sự tổ chức của ứng dụng khi các tính năng và endpoint ngày càng trở nên phức tạp hơn.

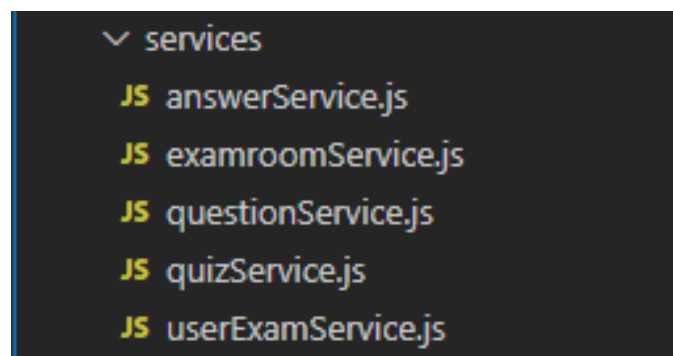
```
const express = require('express')
const router = express.Router()
const { getAllQuiz, addNewQuiz, deleteQuiz, createCompletedQuiz, getFullQuizById } = require('../controllers/quizController');
const { addNewQuestions, updateQuestions, deleteQuestions, getQuestionsByQuiz } = require('../controllers/questionController');
const { addNewAnswers, updateAnswers, deleteAnswers, getAnswersByQs } = require('../controllers/answerController');
const { createExamRoom, getExamRoomById, startDoQuiz, getAllRooms } = require('../controllers/examroomController');
const { submitTest, getResultTest, getAllTestsResult } = require('../controllers/userExamController');
const { authMiddle } = require('../middlewares/auth');
const { checkRole } = require('../middlewares/checkRoles')

router.use(authMiddle) // áp middleware cho tất cả route

//quiz
router.get('/quiz', getAllQuiz)
router.post('/quiz', checkRole('Admin'), addNewQuiz)
router.delete('/quiz', checkRole('Admin'), deleteQuiz)
router.post('/quiz-create', checkRole('Admin'), createCompletedQuiz)
router.get('/quiz/:id', checkRole('Admin'), getFullQuizById)
```

- services/

Thư mục services chứa các file Service, nơi chứa logic nghiệp vụ chính của ứng dụng. Các services giúp tách biệt các logic nghiệp vụ phức tạp ra khỏi controllers, đảm bảo rằng các controllers chỉ tập trung vào việc nhận và trả dữ liệu từ người dùng mà không xử lý quá nhiều công việc phức tạp. Mỗi service có thể chứa nhiều hàm thực hiện các tác vụ liên quan đến dữ liệu, như truy vấn cơ sở dữ liệu, thao tác với các model hoặc thực hiện các phép toán, xử lý dữ liệu. Việc chia tách logic nghiệp vụ vào trong services giúp mã nguồn dễ dàng kiểm thử, tái sử dụng và bảo trì. Các services thường được gọi từ controllers khi cần xử lý các yêu cầu từ người dùng.



```
▼ services
  JS answerService.js
  JS examroomService.js
  JS questionService.js
  JS quizService.js
  JS userExamService.js
```

- server.js

File server.js là điểm khởi đầu của ứng dụng NodeJS. Đây là nơi bạn sẽ khởi tạo ứng dụng ExpressJS, thiết lập các middleware (như xử lý body của yêu cầu, CORS, logging, v.v.), và định nghĩa các route của API. Sau khi cấu hình xong, file này sẽ khởi động server và bắt đầu lắng nghe các yêu cầu HTTP từ người dùng. File server.js đóng vai trò quan trọng trong việc kết nối các thành phần trong ứng dụng, từ các route, controller, services cho đến các cấu hình môi trường, giúp server có thể phục vụ các yêu cầu từ người dùng một cách hiệu quả..

```
JS server.js
server > src > JS server.js > ...
1  const express = require('express')
2  const app = express()
3  const port = 8080
4  const cors = require('cors');
5  const path = require('path')
6  const pool = require('./config/database')
7  const User = require('./models/User')
8  const Quiz = require('./models/Quiz')
9  const Questions = require('./models/Question')
10 const Answers = require('./models/Answer')
11 const router = require('./routes/routers');
12 const ExamRoom = require('./models/ExamRoom');
13 const UserExam = require('./models/UserExam');
14
15 app.use(cors());
16 app.use(express.json());
17 app.use(express.urlencoded({ extended: true }));
18 app.use('/api', router)
19
20 > const initTable = async () => { ...
28   }
29
30 // auto connect db
31 > (async () => { ...
39   })()
40
41
42 > app.get('/', (req, res) => { ...
44   })
45
46 > app.listen(port, () => { ...
48   })
```

- File package.json

File package.json là file cấu hình quan trọng của dự án NodeJS. Nó chứa thông tin về dự án như tên, phiên bản, mô tả, các script sử dụng để chạy ứng dụng, cũng như các phụ thuộc (dependencies) mà ứng dụng cần sử dụng. File này giúp quản lý các thư viện và gói phần mềm (package) mà ứng dụng cần, đồng thời cho phép cài đặt và quản lý các gói này thông qua các công cụ như npm hoặc yarn. Bên cạnh đó, file package.json cũng định nghĩa các script, chẳng hạn như script khởi động ứng dụng, script chạy ứng dụng trong chế độ phát triển (với nodemon, chẳng hạn), hay các script kiểm tra (test). Ngoài ra, package.json cũng là nơi quản lý các thông tin về các phiên bản các gói phụ thuộc, giúp đảm bảo ứng dụng luôn sử dụng phiên bản phù hợp của các thư viện.

```
{
  "name": "englishtest",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1",
    "dev": "nodemon src/server.js"
  },
  "author": "",
  "license": "ISC",
  "dependencies": {
    "body-parser": "^1.20.3",
    "cors": "^2.8.5",
    "dotenv": "^16.4.7",
    "express": "^4.21.2",
    "jsonwebtoken": "^9.0.2",
    "moment-timezone": "^0.5.47",
    "mysql2": "^3.12.0"
  },
  "devDependencies": {
    "nodemon": "^3.1.9"
  }
}
```

## 2. Danh sách API cơ bản:

EnglishTestWeb
Quiz
GET Get All Quiz
POST Add Quiz
DEL Delete Quiz
POST Create Completed Quiz
GET Get Completed Quiz By Id
Question
GET Get All Questions by Quiz
POST Add Questions
PUT Update Questions
DEL Delete Questions
Answer
GET Get All Answers
POST Add Answers
PUT Update Answers
DEL Delete Answers
Rooms
GET Get Room By Id
POST Create Room
POST Start Do Quiz
GET Get All Rooms
Test Result
PUT Submit Test
GET Get Result Test
GET Get Tests History

### Tuần 3 (3/3 – 9/3): Phân quyền, hoàn thiện API với JWT

Thực hiện tích hợp xác thực access\_token qua public key và điều chỉnh API để hỗ trợ phân quyền người dùng bằng cách kết hợp với đồng nghiệp phát triển phần Auth :

*Xây dựng các middleware xác thực và phân quyền:*

Để thực hiện xác thực và phân quyền hiệu quả, các middleware sẽ được xây dựng trong hệ thống ExpressJS. Các middleware này sẽ thực hiện các nhiệm vụ chính như:

- Xác thực token: Middleware đầu tiên sẽ kiểm tra xem có token trong header của yêu cầu hay không và xác thực token thông qua public key. Nếu token không hợp lệ hoặc hết hạn, middleware sẽ trả về lỗi 401 (Unauthorized).
- Giải mã thông tin người dùng: Sau khi xác thực thành công, token sẽ được giải mã để lấy thông tin người dùng, bao gồm các quyền và vai trò của người dùng.
- Kiểm tra quyền truy cập: Middleware sẽ kiểm tra quyền của người dùng dựa trên vai trò trong token và so sánh với yêu cầu quyền của API endpoint. Nếu người dùng không có quyền truy cập, hệ thống sẽ trả về lỗi 403 (Forbidden).

```
1  const jwt = require('jsonwebtoken');
2  const { verifyToken } = require('../config/jwtconfigs')
3
4  const authMiddle = (req, res, next) => {
5      const token = req.headers.authorization?.split(" ")[1]; //Lấy access_token sau Bearer
6
7      if (!token) return res.status(401).json({ message: "No access_token" });
8
9
10     try {
11         const decoded = verifyToken(token);
12         req.user = { // đính thêm info user vào mỗi req trước khi gửi
13             id: decoded.nameid,
14             role: decoded.role
15         };
16         next(); // req thành công
17     } catch (error) {
18         return res.status(403).json({ message: "Invalid or expired token" });
19     }
20 }
21
22 module.exports = { authMiddle }
```

```
const checkRole = (...allowedRoles) => { // tự tạo mảng nếu truyền vào nhiều role
    return (req, res, next) => {
        if (!allowedRoles.includes(req.user.role)) {
            return res.status(403).json({ message: "Bạn không có quyền truy cập" });
        }
        next();
    };
}

module.exports = { checkRole };
```

Các middleware này sẽ được áp dụng cho các route cần bảo vệ, đảm bảo rằng mỗi yêu cầu HTTP phải vượt qua các bước xác thực và phân quyền trước khi được xử lý. Các API sẽ được bảo vệ và yêu cầu quyền truy cập phù hợp dựa trên vai trò của người dùng trong token. Nếu người dùng không có quyền truy cập, hệ thống sẽ trả về mã lỗi 403 (Forbidden).

```
router.use(authMiddle) // áp middleware cho tất cả route

//quiz
router.get('/quiz', getAllQuiz)
router.post('/quiz', checkRole('Admin'), addNewQuiz)
router.delete('/quiz', checkRole('Admin'), deleteQuiz)
router.post('/quiz-create', checkRole('Admin'), createCompletedQuiz)
router.get('/quiz/:id', checkRole('Admin'), getFullQuizById)

//question
router.get('/question', getQuestionsByQuiz)
router.post('/question', checkRole('Admin'), addNewQuestions)
router.put('/question', checkRole('Admin'), updateQuestions)
router.delete('/question', checkRole('Admin'), deleteQuestions)

//answer
router.get('/answer', getAnswersByQs)
router.post('/answer', checkRole('Admin'), addNewAnswers)
router.put('/answer', checkRole('Admin'), updateAnswers)
//router.delete('/answer', deleteAnswers)

//examroom
router.get('/exam-room/:id', checkRole('User'), getExamRoomById)
router.get('/exam-room', checkRole('User'), getAllRooms)
router.post('/exam-room', checkRole('Admin'), createExamRoom)
router.post('/exam-room/:id/start', checkRole('User'), startDoQuiz)

//userExam
router.put('/exam-submit', checkRole('User'), submitTest)
router.get('/exam-result', checkRole('User'), getResultTest)
router.get('/exam-history', checkRole('User'), getAllTestsResult)

module.exports = router
```

*Cập nhật tài liệu API và mô tả phân quyền:*

Sau khi hoàn thành việc điều chỉnh API để hỗ trợ phân quyền, tài liệu API sẽ được cập nhật để mô tả rõ ràng yêu cầu phân quyền cho từng endpoint. Mỗi endpoint sẽ có phần mô tả các quyền cần thiết để truy cập, với các quyền hạn phân chia rõ ràng giữa User và Admin.

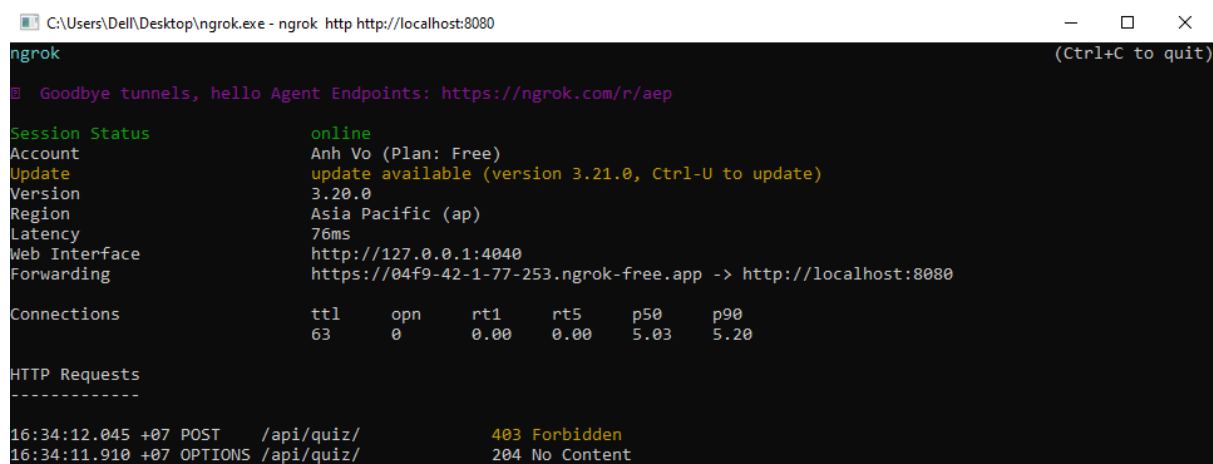
## Tuần 4 (10/3 - 16/3): Public API – Hợp tác với FrontEnd

### 1. Triển khai Public API qua Ngrok:

Triển khai API công khai (“public API”) sử dụng Ngrok để hỗ trợ team Frontend trong việc tích hợp API vào giao diện web. Ngrok là một công cụ hỗ trợ tạo một đường dẫn tạm thời truy cập API nội bộ từ internet, giúp Frontend test API ngay cả khi backend chưa deploy lên server:

Cài đặt Ngrok và đăng ký tài khoản để lấy API key.

- Chạy Ngrok để expose cổng API backend (Node.js/Express) ra internet.
- Kiểm tra các endpoint API và test kết nối bằng Postman.
- Chia sẻ URL Ngrok với team Frontend để hỗ trợ việc fetch dữ liệu.
- Xử lý các vấn đề bất định do Ngrok thay đổi URL sau mỗi lần khởi chạy.



```
C:\Users\ DELL\Desktop\ngrok.exe - ngrok http http://localhost:8080
ngrok
Goodbye tunnels, hello Agent Endpoints: https://ngrok.com/r/aep

Session Status      online
Account             Anh Vo (Plan: Free)
Update              update available (version 3.21.0, Ctrl-U to update)
Version             3.20.0
Region              Asia Pacific (ap)
Latency             76ms
Web Interface       http://127.0.0.1:4040
Forwarding           https://04f9-42-1-77-253.ngrok-free.app -> http://localhost:8080

Connections
  ttl    opn    rt1    rt5    p50    p90
    63     0     0.00  0.00  5.03  5.20

HTTP Requests
-----
16:34:12.045 +07 POST    /api/quiz/          403 Forbidden
16:34:11.910 +07 OPTIONS /api/quiz/          204 No Content
```

Qua việc sử dụng Ngrok, bản thân hiểu rõ hơn về cách làm việc với API trong giai đoạn phát triển, giúp đồng bộ giữa Backend và Frontend trước khi deploy chính thức.

### 2. Trao đổi, hợp tác với Frontend trong việc gọi API:

Trong quá trình làm việc, em nhận ra rằng backend và frontend cần giao tiếp liên tục với nhau để tránh những sai sót khi gọi API. Ban đầu, frontend gửi request nhưng nhận lại HTTP 400 hoặc 500 do khác biệt trong dữ liệu gửi nhận. Sau khi trao đổi với team, em đã cùng Frontend điều chỉnh lại các vấn đề:

- Xác định rõ request/response format: Quy định frontend gửi request theo JSON có cấu trúc chuẩn, backend trả response theo format thống nhất.
- Test API trước khi frontend gọi: Dùng Postman để kiểm tra các request trước khi tích hợp frontend.
- Ghi chép API document: Viết tài liệu API bằng Swagger/Postman để frontend dễ tra cứu.
- Trao đổi thường xuyên: Tạo nhóm chat với frontend để giải quyết nhanh các bug khi fetch API.



Kết quả là sau khi chỉnh sửa, frontend gọi API một cách đồng nhất, giảm sai sót và backend dễ debug hơn. Việc giao tiếp và tương tác giữa backend và frontend đóng vai trò rất quan trọng trong quá trình phát triển sản phẩm.

## II. VỀ KỸ NĂNG

### *Debug:*

Trong quá trình phát triển ứng dụng, việc gỡ lỗi là một kỹ năng quan trọng để tìm và sửa các lỗi trong mã nguồn. Kỹ năng này giúp xác định được các vấn đề trong ứng dụng và tìm ra nguyên nhân gây ra lỗi, từ đó sửa chữa và cải thiện mã nguồn. Sử dụng các công cụ gỡ lỗi cơ bản như console.log trong NodeJS để theo dõi giá trị các biến và luồng xử lý của chương trình, sử dụng Chrome DevTools để debug mã nguồn phía server, giúp theo dõi từng bước thực thi của ứng dụng, từ đó dễ dàng nhận diện được những điểm sai sót hoặc các hành vi không mong muốn trong chương trình.

### *Đọc hiểu tài liệu và xác định API cần thiết:*

Một trong những kỹ năng quan trọng mà tôi đã phát triển trong suốt thời gian thực tập là khả năng đọc hiểu tài liệu kỹ thuật. Khi làm việc với các dự án phần mềm, việc nắm bắt các yêu cầu và hướng dẫn từ tài liệu API giúp tôi dễ dàng xác định những tính năng hoặc API cần được xây dựng. Tôi đã học được cách làm quen với các tài liệu kỹ thuật, từ đó hiểu rõ các yêu cầu chức năng và logic của từng API, đảm bảo việc phát triển phù hợp với thiết kế ban đầu của hệ thống.

### *Tiếp cận với Microservice và làm việc với JWT:*

Làm quen với kiến trúc microservice, tiếp xúc với các khái niệm như phân chia các dịch vụ thành những phần nhỏ, độc lập và giao tiếp giữa chúng qua API. Đặc biệt, việc làm việc với JSON Web Token (JWT) để xác thực và phân quyền người dùng, giải mã và kiểm tra token giúp bản thân hiểu về các dịch vụ có thể bảo vệ các API và quản lý người dùng giữa các service khác nhau. Đây là một phần quan trọng khi ứng dụng cần được phân chia thành nhiều dịch vụ độc lập nhưng vẫn cần tương tác với nhau một cách an toàn.

### *Kỹ năng teamwork:*

Học được cách làm việc nhóm hiệu quả thông qua hợp tác với team Frontend, thường xuyên tham gia là hỗ trợ team Frontend bằng cách điều chỉnh các API sao cho phù hợp với yêu cầu giao diện người dùng. Điều này không chỉ giúp các phần của ứng dụng hoạt động mượt mà còn giúp cải thiện hiệu quả công việc nhóm, nâng cao các kỹ năng giao tiếp, kỹ năng giải quyết vấn đề.

### III. PHẨM CHẤT ĐẠO ĐỨC NGHỀ NGHIỆP

- ✓ **Tính kỷ luật:** Luôn nỗ lực tuân thủ đầy đủ các quy định và nội quy của công ty, làm việc đúng giờ, thực hiện các yêu cầu công việc một cách nghiêm túc và đúng tiến độ, duy trì sự chuyên nghiệp và có tác phong làm việc hiệu quả trong môi trường công ty.
- ✓ **Tác phong nhanh nhẹn, chủ động:** Cố gắng hoàn thành nhiệm vụ được giao một cách nhanh chóng và đảm bảo chất lượng công việc, Chủ động giáp tiếp đầy đủ với các bạn đồng nghiệp nhằm đảm bảo rằng công việc luôn đi đúng hướng và kịp thời giải quyết các vấn đề phát sinh.
- ✓ **Thái độ trong làm việc nhóm:** Ngoài việc hoàn thành tốt công việc, bản thân luôn cố gắng giữ thái độ hòa đồng, tôn trọng và lễ phép đối với đồng nghiệp trong team, giúp duy trì mối quan hệ tốt đẹp với mọi người và làm việc trong một môi trường hợp tác, thân thiện, tạo điều kiện cho công việc nhóm diễn ra thuận lợi hơn.
- ✓ **Tinh thần học hỏi và trau dồi kiến thức:** Trong quá trình thực tập, bản thân luôn giữ tinh thần học hỏi, tìm hiểu thêm về các vấn đề chuyên môn từ các đồng nghiệp xung quanh, cố gắng tiếp thu những Kiến thức mới và áp dụng vào công việc để cải thiện bản thân.
- ✓ **Nhiệt huyết và đam mê với công việc:** Luôn giữ một tinh thần nhiệt huyết với công việc và tìm hiểu nhiều tài liệu chuyên môn để nâng cao kiến thức. Việc luôn tìm tòi, học hỏi và trau dồi kiến thức đã giúp phát triển kỹ năng cá nhân và đóng góp tích cực cho công việc của team.

### IV. NHỮNG THU HOẠCH BỔ ÍCH NHẤT ĐỐI VỚI BẢN THÂN

Tuy thời gian thực tập tại công ty Công ty cổ phần công nghệ Aido không nhiều nhưng để lại cho em những kiến thức bổ ích và những kinh nghiệm có thể giúp em sau này làm việc tốt hơn. Em đã học được rất nhiều kiến thức mới từ thầy Nguyễn Văn Nam và các bạn trong team làm project:

- Được làm việc trong môi trường doanh nghiệp chuyên nghiệp.
- Học và tìm hiểu về NodeJS, MySQL, Postman, kỹ năng viết API chuẩn RESTful, xây dựng hệ thống theo mô hình Client – Server
- Học được cách làm việc theo đúng thời gian, phân bổ công việc cho phù hợp.
- Kỹ năng mềm: giao tiếp, ứng xử được cải thiện trong môi trường làm việc chuyên môn.
- Có thêm những người đồng nghiệp, những người bạn tốt, giúp bản thân được học hỏi và có cơ hội phát triển về sự nghiệp trong tương lai.

## V. NHỮNG HẠN CHẾ CỦA BẢN THÂN:

- Kiến thức:
  - + Mới chỉ tiếp cận với NodeJS ở mức cơ bản
  - + Làm quen với viết API theo chuẩn RESTful nhưng vẫn còn đơn giản, chưa đầy đủ
  - + Chưa hiểu về các vấn đề như tối ưu hiệu suất, bảo mật, ...
  - + Tìm hiểu cơ bản để có thể xây dựng những thao tác đơn giản của website, chưa tối ưu trong việc thiết kế hệ thống
- Kỹ năng: khả năng lập trình chưa tốt, cần học hỏi, rèn luyện và thực hành nhiều hơn.
- Ngoại ngữ: ở mức đọc hiểu tài liệu Tiếng Anh, cần bổ sung kiến thức về Tiếng Anh chuyên ngành, kỹ năng nói, viết để phục vụ tốt cho công việc.
- Kỹ năng mềm: kỹ năng giao tiếp, thuyết trình chưa thực sự tốt.

## VI. KIẾN NGHỊ

- Về hình thức thực tập: không có.
- Về thời gian: thời gian thực tập tương đối ngắn, nên sinh viên chưa học tập được nhiều kiến thức và trau dồi kinh nghiệm làm việc từ môi trường làm việc doanh nghiệp.
- Về GVHD: không có.
- Các kiến nghị khác (nếu có): không có.

## VIII. ĐÁNH GIÁ CỦA GVHD TẠI TRƯỜNG ĐHTL

Điểm số	Nhận xét	GVHD tại ĐHTL (Ký và ghi rõ họ tên)



TRƯỜNG ĐẠI HỌC THỦY LỢI  
KHOA CÔNG NGHỆ THÔNG TIN

CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM  
Độc lập – Tự do – Hạnh phúc

PHIẾU ĐÁNH GIÁ QUÁ TRÌNH THỰC TẬP TẠI CƠ SỞ THỰC TẬP

Họ tên sinh viên thực tập: Võ Viết Anh  
Mã SV: 215.106.2215 Lớp: CS CNTT 2 Ngày sinh: 27.10.1999  
Cơ sở thực tập: Làng ủy ban Phường Long Nghé An Đô  
Phố 12.1.29.199 Kim Đồng, Quận Bình, Thành Phố Hồ Chí Minh

Phản đánh giá

Tiêu chí đánh giá	Điểm tối đa	Điểm do Cơ sở thực tập đánh giá
Ý thức chấp hành nội quy, kỷ luật tại đơn vị thực tập	2.0	2.0
Tinh thần trách nhiệm, thái độ đối với công việc	2.0	2.0
Khả năng vận dụng kiến thức chuyên môn, kỹ năng mềm vào thực tiễn	2.0	1.5
Kết quả, mức độ hoàn thành công việc được giao	4.0	3.5
<b>Tổng</b>	<b>10</b>	<b>9.0</b>

(Điểm đánh giá được làm tròn đến 1 chữ số thập phân)

Điểm trung bình chung:

Bảng số: 9.0 điểm

Bảng chữ: Chun

Ngày 16 tháng 08 năm 2025

GVHD của Cơ sở thực tập

(Ký và ghi rõ họ tên)

Nguyễn Văn Nam