**SERVER MODULE**

# 

# CONTENTS

This module has the following files:

1. MODULE\_SERVER.docx
2. MODULE\_SERVER\_MEDIA.zip

# INTRODUCTION

In this module, you are asked to create an online learning application called “**Web Tech Studio**” where in the application a user can register and complete their course to get a certificate. The detailed description and tools that you can use will be described below.

# DESCRIPTION OF PROJECTS AND TASKS

This module is divided into two phases. In the first phase, you will create a REST API. In the second phase, you will build a frontend application. You must use the provided frameworks:

* Laravel (v11.x)
* React (v18.x with react-router-dom and axios)
* Vue (v3.x with vue-router and axios)

# 

# PHASE 1: REST API

## Authentication

Authentication serves as the gateway for users to access all courses within the application. There are three endpoints: register, login, and logout. **Sanctum** should be used to create tokens. The generated token will be used to access all available endpoints. The logout function should only deactivate the token from the specific device making the request.

### A1: Register

**POST** /api/register

For users to create an account in the application. When users create an account in the application, their role should be assigned as 'user' by default.

**Request Headers**

| Accept | application/json |
| --- | --- |

**Body (JSON)**

| full\_name | required. |
| --- | --- |
| username | required, min 3 chars, unique, accept only alphanumeric, dot “.” and underscore “\_”. |
| password | required, min 6 chars. |

**A1a: Response Success**

| Status Code | 201 |
| --- | --- |
| Body | {  "status": "success",  "message": "User registration successful",  "data": {  "full\_name": "Richard Roe",  "username": "richard.roe",  "updated\_at": "2024-08-08T02:26:52..",  "created\_at": "2024-08-08T02:26:52..",  "id": 2,  "token": "1|qgV2cTisoRVSwX6R2iPpxd..",  "role": "user"  } } |

**A1b: Response Invalid Field**

| Status Code | 400 |
| --- | --- |
| Body | {  "status": "error",  "message": "Invalid field(s) in request",  "errors": {  "full\_name": [  "The full name field is required."  ],  "username": [  "The username has already been taken."  ],  "password": [  "The password field is required."  ]  } } |

### A2: Login

**POST** /api/login

For admin and users login into the application using username and password.

**Request Headers**

| Accept | application/json |
| --- | --- |

**Body (JSON)**

| username | required. |
| --- | --- |
| password | required. |

**A2a: Response Success**

| Status Code | 200 |
| --- | --- |
| Body | {  "status": "success",  "message": "Login successful",  "data": {  "id": 1,  "username": "admin",  "created\_at": "2024-08-08T02:26:48..",  "updated\_at": "2024-08-08T02:26:48..",  "token": "2|vKXiM2qjuLIvgTsWjzJ0p6..",  "role": "admin"  } } |

**A2b: Response Invalid Credentials**

| Status Code | 400 |
| --- | --- |
| Body | {  "status": "authentication\_failed",  "message": "The username or password you entered is incorrect" } |

### A3: Logout

**POST** /api/logout

For logged-in admin and user logout from the application. It specifically deactivates the authentication token associated with the current device making the request.

**Request Headers**

| Accept | application/json |
| --- | --- |
| Authorization | Bearer <TOKEN> |

**A3a: Response Success**

| Status Code | 200 |
| --- | --- |
| Body | {  "status": "success",  "message": "Logout successful" } |

**A3b: Response Invalid Token**

| Status Code | 401 |
| --- | --- |
| Body | {  "status": "invalid\_token",  "message": "Invalid or expired token" } |

## Course

### B1: Add Course

**POST** /api/courses

For admin to add a new course. When an admin adds a new course, it should be set to 'draft' and not published by default.

**Request Headers**

| Accept | application/json |
| --- | --- |
| Authorization | Bearer <TOKEN> |

**Body (JSON)**

| name | required. |
| --- | --- |
| description | nullable. |
| slug | required, unique. |

**B1a: Response Success**

| Status Code | 201 |
| --- | --- |
| Body | {  "status": "success",  "message": "Course successfully added",  "data": {  "name": "Web API Architecture",  "description": "Design and implement..",  "slug": "web-api-architecture",  "updated\_at": "2024-08-08T02:45:04..",  "created\_at": "2024-08-08T02:45:04..",  "id": 10  } } |

**B1b: Response Invalid Field**

| Status Code | 400 |
| --- | --- |
| Body | {  "status": "error",  "message": "Invalid field(s) in request",  "errors": {  "name": [  "The name field is required."  ],  "slug": [  "The slug has already been taken."  ]  } } |

**B1c: Response Forbidden**

| Status Code | 403 |
| --- | --- |
| Body | {  "status": "insufficient\_permissions",  "message": "Access forbidden" } |

**B1d: Response Invalid Token**

| Status Code | 401 |
| --- | --- |
| Body | {  "status": "invalid\_token",  "message": "Invalid or expired token" } |

### B2: Edit Course

**PUT** /api/courses/:course\_slug

For admin to edit an existing course. An admin can publish the course by setting 'is\_published' to true.

**Request Headers**

| Accept | application/json |
| --- | --- |
| Authorization | Bearer <TOKEN> |

**Body (JSON)**

| name | required. |
| --- | --- |
| description | nullable. |
| is\_published | nullable, boolean. |

**B2a: Response Success**

| Status Code | 200 |
| --- | --- |
| Body | {  "status": "success",  "message": "Course successfully updated",  "data": {  "id": 10,  "name": "Web API Architecture (published)",  "slug": "web-api-architecture",  "description": "Design and implement..",  "is\_published": **true**,  "created\_at": "2024-08-08T02:45:04..",  "updated\_at": "2024-08-08T02:53:43.."  } } |

**B2b: Response Invalid Field**

| Status Code | 400 |
| --- | --- |
| Body | {  "status": "error",  "message": "Invalid field(s) in request",  "errors": {  "name": [  "The name field is required."  ],  "is\_published": [  "The is published field must be true or false."  ]  } } |

**B2c: Response Not Found**

| Status Code | 404 |
| --- | --- |
| Body | {  "status": "not\_found",  "message": "Resource not found" } |

**B2d: Response Forbidden**

| Status Code | 403 |
| --- | --- |
| Body | {  "status": "insufficient\_permissions",  "message": "Access forbidden" } |

**B2e: Response Invalid Token**

| Status Code | 401 |
| --- | --- |
| Body | {  "status": "invalid\_token",  "message": "Invalid or expired token" } |

### B3: Delete Course

**DELETE** /api/courses/:course\_slug

For admin to delete existing courses. When an admin deletes a course, it should be removed from the database.

**Request Headers**

| Accept | application/json |
| --- | --- |
| Authorization | Bearer <TOKEN> |

**B3a: Response Success**

| Status Code | 200 |
| --- | --- |
| Body | {  "status": "success",  "message": "Course successfully deleted" } |

**B3b: Response Not Found**

| Status Code | 404 |
| --- | --- |
| Body | {  "status": "not\_found",  "message": "Resource not found" } |

**B3c: Response Forbidden**

| Status Code | 403 |
| --- | --- |
| Body | {  "status": "insufficient\_permissions",  "message": "Access forbidden" } |

**B3d: Response Invalid Token**

| Status Code | 401 |
| --- | --- |
| Body | {  "status": "invalid\_token",  "message": "Invalid or expired token" } |

### B4: Get All Published Courses

**GET** /api/courses

For logged-in users to get all of the published courses.

**Request Headers**

| Accept | application/json |
| --- | --- |
| Authorization | Bearer <TOKEN> |

**B4a: Response Success**

| Status Code | 200 |
| --- | --- |
| Body | {  "status": "success",  "message": "Courses retrieved successfully",  "data": {  "courses": [  {  "id": 1,  "name": "Web Dev Fundamentals",  "slug": "web-dev-fundamentals",  "description": "Grasp the core..",  "is\_published": 1,  "created\_at": "2024-08-08T02...",  "updated\_at": "2024-08-08T02..."  },  ...  ]  } } |

**B4b: Response Invalid Token**

| Status Code | 401 |
| --- | --- |
| Body | {  "status": "invalid\_token",  "message": "Invalid or expired token" } |

### B5: Get Course Details

**GET** /api/courses/:course\_slug

For logged-in users to get course details. There are two content types: 'learn' and 'quiz.' For the 'quiz' type, there is an additional 'options' field. Sets, lessons, and contents should be ordered in ascending order based on the 'order' field.

**Request Headers**

| Accept | application/json |
| --- | --- |
| Authorization | Bearer <TOKEN> |

**B5a: Response Success**

| Status Code | 200 |
| --- | --- |
| Body | {  "status": "success",  "message": "Course details retrieved successfully",  "data": {  "id": 1,  "name": "Web Dev Fundamentals",  "slug": "web-dev-fundamentals",  "description": "Grasp the core technologies of HTML...",  "is\_published": 1,  "created\_at": "2024-08-08T02:44:52...",  "updated\_at": "2024-08-08T02:44:52...",   "sets": [  {  "id": 1,  "name": "HTML Basics",  "order": 0,  "lessons": [  {  "id": 1,  "name": "Introduction to HTML",  "order": 0,   "contents": [  {  "id": 1,  "type": "learn",  "content": "HTML stands for...",  "order": 0  },  {  "id": 2,  "type": "quiz",  "content": "What does HTML stand for?",  "order": 1,  "options": [  {  "id": 1,  "option\_text": "Hyper Text"  },  {  "id": 2,  "option\_text": "Home Tool Markup Language"  },  {  "id": 3,  "option\_text": "Hyper Transfer Markup Language"  },  {  "id": 4,  "option\_text": "Hyperlink and Text Markup Language"  }  ]  },  ]  },  ]  }  ]  } } |

**B5b: Response Invalid Token**

| Status Code | 401 |
| --- | --- |
| Body | {  "status": "invalid\_token",  "message": "Invalid or expired token" } |

## Set

### C1: Add Set

**POST** /api/courses/{course}/sets

For admin to add a new set of courses. When a set is added, its 'order' value should be automatically incremented.

**Request Headers**

| Accept | application/json |
| --- | --- |
| Authorization | Bearer <TOKEN> |

**Body (JSON)**

| name | required. |
| --- | --- |

**C1a: Response Success**

| Status Code | 201 |
| --- | --- |
| Body | {  "status": "success",  "message": "Set successfully added",  "data": {  "name": "Web API Architecture Fundamentals",  "order": 2,  "id": 5  } } |

**C1b: Response Invalid Field**

| Status Code | 400 |
| --- | --- |
| Body | {  "status": "error",  "message": "Invalid field(s) in request",  "errors": {  "name": [  "The name field is required."  ]  } } |

**C1c: Response Forbidden**

| Status Code | 403 |
| --- | --- |
| Body | {  "status": "insufficient\_permissions",  "message": "Access forbidden" } |

**C1d: Response Invalid Token**

| Status Code | 401 |
| --- | --- |
| Body | {  "status": "invalid\_token",  "message": "Invalid or expired token" } |

### 

### C2: Delete Set

**DELETE** /api/courses/{course}/sets/:set\_id

For admin to delete a set of courses. When an admin deletes a set, it should be removed from the database.

**Request Headers**

| Accept | application/json |
| --- | --- |
| Authorization | Bearer <TOKEN> |

**Body (JSON)**

| name | required. |
| --- | --- |

**C2a: Response Success**

| Status Code | 200 |
| --- | --- |
| Body | {  "status": "success",  "message": "Set successfully deleted" } |

**C2b: Response Invalid Field**

| Status Code | 400 |
| --- | --- |
| Body | {  "status": "error",  "message": "Invalid field(s) in request",  "errors": {  "name": [  "The name field is required."  ]  } } |

**C2c: Response Forbidden**

| Status Code | 403 |
| --- | --- |
| Body | {  "status": "insufficient\_permissions",  "message": "Access forbidden" } |

**C2d: Response Invalid Token**

| Status Code | 401 |
| --- | --- |
| Body | {  "status": "invalid\_token",  "message": "Invalid or expired token" } |

## Lesson

### D1: Add Lesson

**POST** /api/lessons

For admin to add a new lesson of a set. When a lesson is added, its 'order' value should be automatically incremented.

**Request Headers**

| Accept | application/json |
| --- | --- |
| Authorization | Bearer <TOKEN> |

**Body (JSON)**

| name | required. |
| --- | --- |
| set\_id | required, valid set\_id data. |
| contents | array of objects, each object must contain following fields:   * type (‘learn’ or ‘quiz’) * content * options: (when type is ‘learn’). it’s object of:   + option\_text   + is\_correct (true/false) |

**D1a: Response Success**

| Status Code | 201 |
| --- | --- |
| Body | {  "status": "success",  "message": "Lesson successfully added",  "data": {  "name": "Introduction to Web API..",  "order": 1,  "id": 10  } } |

**D1b: Response Invalid Field**

| Status Code | 400 |
| --- | --- |
| Body | {  "status": "error",  "message": "Invalid field(s) in request",  "errors": {  "name": [  "The name field is required."  ],  "contents": [  "The contents field must be an array."  ]  } } |

**D1c: Response Forbidden**

| Status Code | 403 |
| --- | --- |
| Body | {  "status": "insufficient\_permissions",  "message": "Access forbidden" } |

**D1d: Response Invalid Token**

| Status Code | 401 |
| --- | --- |
| Body | {  "status": "invalid\_token",  "message": "Invalid or expired token" } |

#### 

### 

### D2: Delete Set

**DELETE** /api/lessons/:lesson\_id

For the admin to delete a lesson from a set. When an admin deletes a lesson, it should be removed from the database.

**Request Headers**

| Accept | application/json |
| --- | --- |
| Authorization | Bearer <TOKEN> |

**D2a: Response Success**

| Status Code | 200 |
| --- | --- |
| Body | {  "status": "success",  "message": "Lesson successfully deleted" } |

**D2b: Response Not Found**

| Status Code | 404 |
| --- | --- |
| Body | {  "status": "not\_found",  "message": "Resource not found" } |

**D2c: Response Forbidden**

| Status Code | 403 |
| --- | --- |
| Body | {  "status": "insufficient\_permissions",  "message": "Access forbidden" } |

**D2d: Response Invalid Token**

| Status Code | 401 |
| --- | --- |
| Body | {  "status": "invalid\_token",  "message": "Invalid or expired token" } |

### D3: Check Answer

**POST** /api/lessons/:lesson\_id/contents/:content\_id/check

For logged-in users to check the answer of a lesson content quiz.

**Request Headers**

| Accept | application/json |
| --- | --- |
| Authorization | Bearer <TOKEN> |

**Body (JSON)**

| option\_id | required, valid option\_id. |
| --- | --- |

**D3a: Response Success**

| Status Code | 200 |
| --- | --- |
| Body | {  "status": "success",  "message": "Check answer success",  "data": {  "question": "What does HTML stand for?",  "user\_answer": "Hyper Text Markup..",  "is\_correct": **true**  } } |

**D3b: Response Only for Quiz Type**

| Status Code | 400 |
| --- | --- |
| Body | {  "status": "error",  "message": "Only for quiz content" } |

**D3c: Response Not Found**

| Status Code | 404 |
| --- | --- |
| Body | {  "status": "not\_found",  "message": "Resource not found"  } |

**D3d: Response Forbidden**

| Status Code | 403 |
| --- | --- |
| Body | {  "status": "insufficient\_permissions",  "message": "Access forbidden" } |

**D3e: Response Invalid Token**

| Status Code | 401 |
| --- | --- |
| Body | {  "status": "invalid\_token",  "message": "Invalid or expired token" } |

### 

### D4: Complete Lesson

**PUT** /api/lessons/:lesson\_id/complete

For logged-in users, track when they have completed all of the lesson contents.

**Request Headers**

| Accept | application/json |
| --- | --- |
| Authorization | Bearer <TOKEN> |

**D4a: Response Success**

| Status Code | 200 |
| --- | --- |
| Body | {  "status": "success",  "message": "Lesson successfully completed" } |

**D4b: Response Not Found**

| Status Code | 404 |
| --- | --- |
| Body | {  "status": "not\_found",  "message": "Resource not found" } |

**D4c: Response Forbidden**

| Status Code | 403 |
| --- | --- |
| Body | {  "status": "insufficient\_permissions",  "message": "Access forbidden" } |

**D4d: Response Invalid Token**

| Status Code | 401 |
| --- | --- |
| Body | {  "status": "invalid\_token",  "message": "Invalid or expired token" } |

## 

## User

### E1: Register to a Course

**POST** /api/courses/:course\_slug/register

For logged-in users to register for a course.

**Request Headers**

| Accept | application/json |
| --- | --- |
| Authorization | Bearer <TOKEN> |

**E1a: Response Success**

| Status Code | 201 |
| --- | --- |
| Body | {  "status": "success",  "message": "User registered successful" } |

**E1b: Response Already Registered**

| Status Code | 400 |
| --- | --- |
| Body | {  "status": "error",  "message": "The user is already registered for this course" } |

**E1c: Response Not Found**

| Status Code | 404 |
| --- | --- |
| Body | {  "status": "not\_found",  "message": "Resource not found" } |

**E1d: Response Invalid Token**

| Status Code | 401 |
| --- | --- |
| Body | {  "status": "invalid\_token",  "message": "Invalid or expired token" } |

#### 

### E2: Get User Progress

**GET** /api/users/progress

For logged-in users to get their registered courses and completed lessons of the course.

**Request Headers**

| Accept | application/json |
| --- | --- |
| Authorization | Bearer <TOKEN> |

**E2a: Response Success**

| Status Code | 201 |
| --- | --- |
| Body | {  "status": "success",  "message": "User progress retrieved successfully",  "data": {  "progress": [  {  "course": {  "id": 1,  "name": "Web Dev Fundamentals",  "slug": "web-dev-fundamentals",  "description": "Grasp the core technologies..",  "is\_published": 1,  "created\_at": "2024-08-08T02:44:52.000000Z",  "updated\_at": "2024-08-08T02:44:52.000000Z"  },  "completed\_lessons": [  {  "id": 1,  "name": "Introduction to HTML",  "order": 0  }  ]  }  ]  } } |

**E2b: Response Invalid Token**

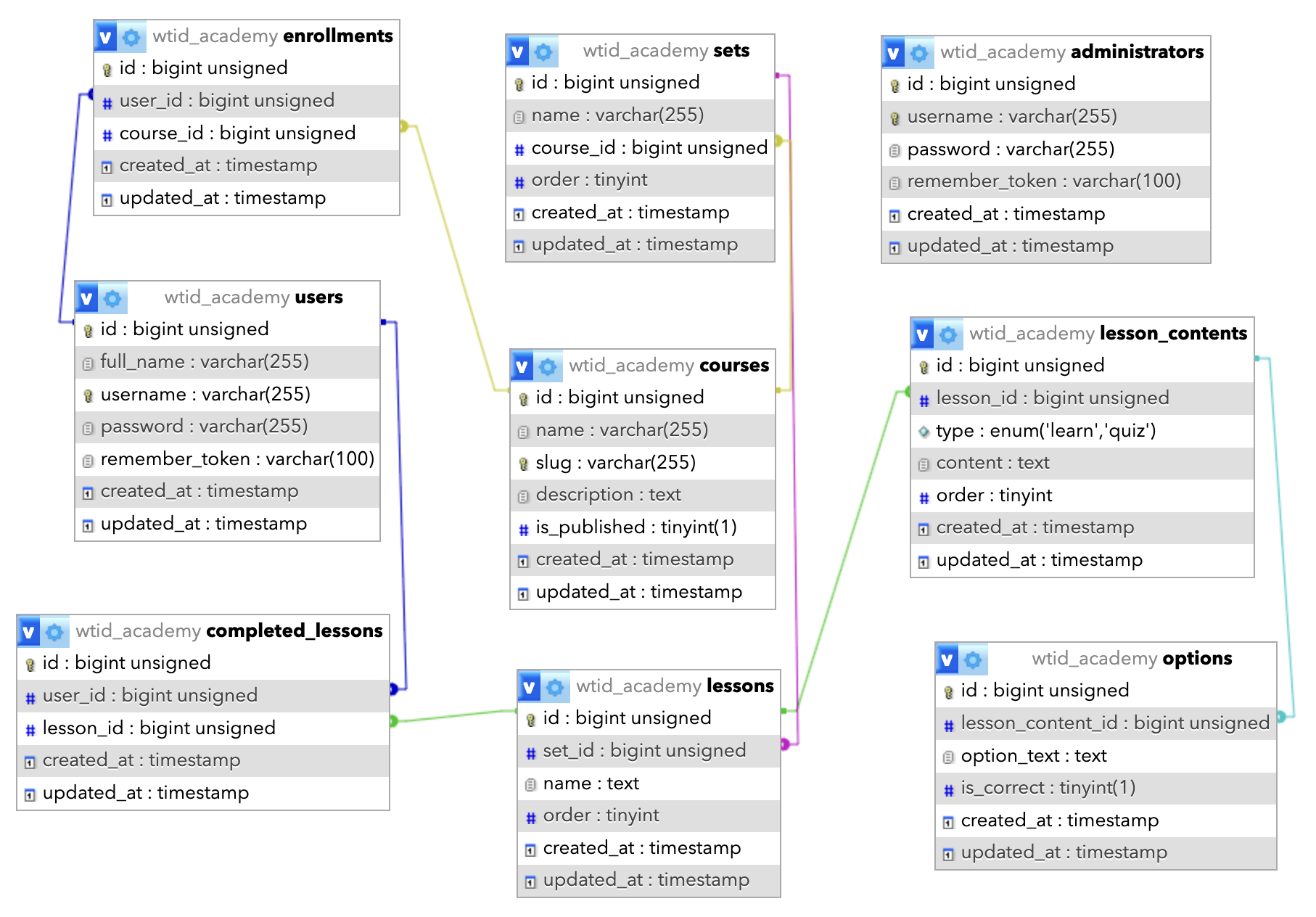
| Status Code | 401 |
| --- | --- |
| Body | {  "status": "invalid\_token",  "message": "Invalid or expired token" } |

### 

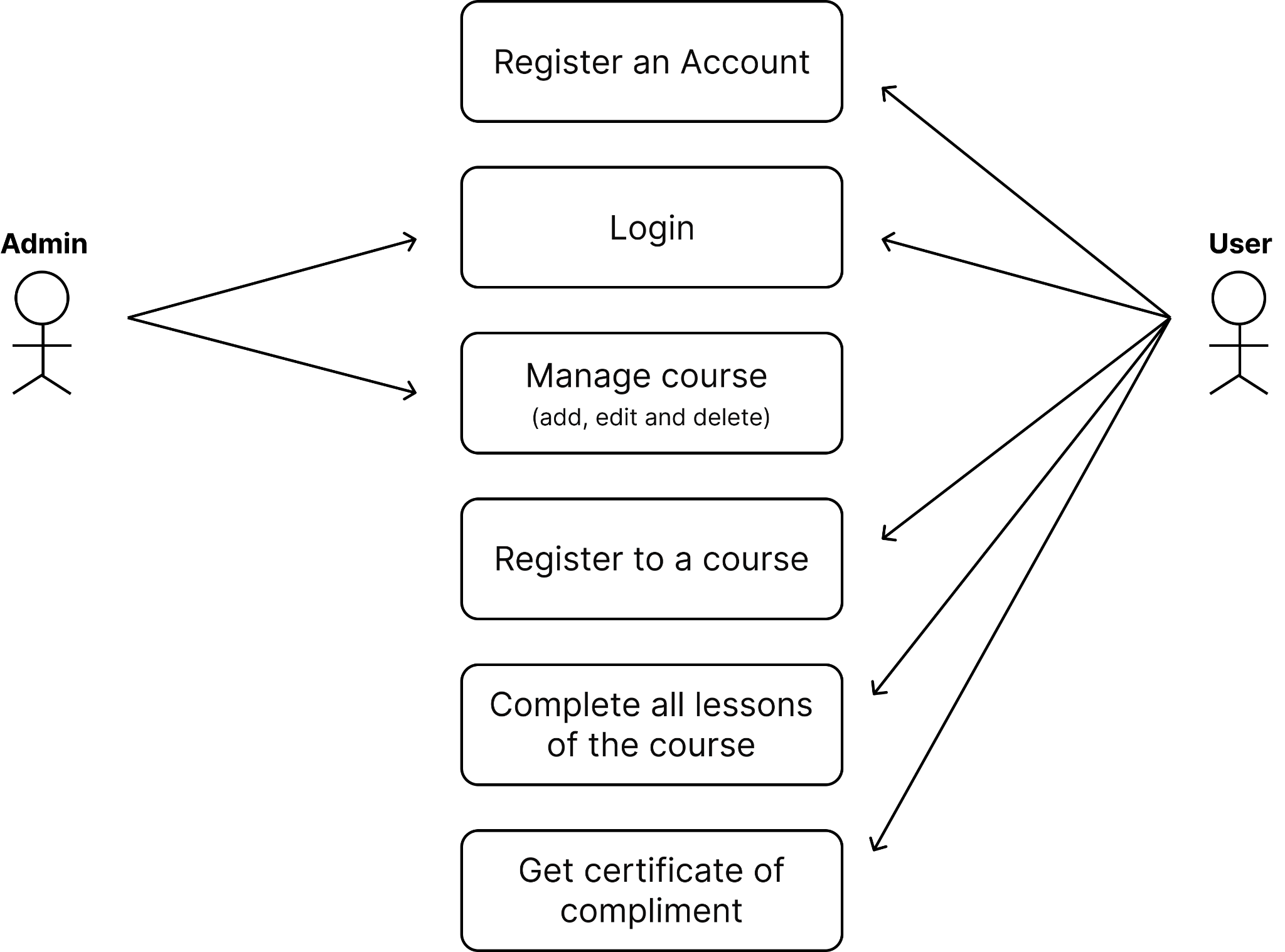
# PHASE 2: Front-End Development

| **Pages** | **Features** |
| --- | --- |
| General | * Document title should be reflected to the current page * After a successful login, display the full name and a logout button on the navbar. * Users can log out by clicking the logout button on the navbar. |
| Register | * Path: /register * Users can register using following fields:   + Full Name   + Username   + Password * Display error messages corresponding to the specific error fields when a failure occurs. * After a successful registration, the user will be redirected to the home page. * If a logged-in user tries to access this page, they will be redirected to the home page. |
| Login | * Path: /login * Users can log in using their existing credentials: username and password. * Display error messages corresponding to the specific error fields when a failure occurs. * After a successful login, the user will be redirected to the home page. * If a logged-in user tries to access this page, they will be redirected to the home page. |
| Home Page | * Path: / * Only logged-in user can access this page * Display the user's registered courses in one section and other available courses in a separate section. * Display course name and course description correctly |
| Course Page | * Path: /:course\_slug * Only logged-in user can access this page * If the user is not registered for the course, display:   + Course Title   + Description   + 'Register to this course' button.   + Outline (name of sets) * Users can register for the course by clicking the 'Register to this course' button. * If the user is already registered for the course, display:   + Course Title   + Learning progress (percentage)   + All lessons grouped by the set * Display a badge on each lesson indicating its status: 'completed,' 'current,' or 'locked,' based on the condition. * Users only can access the 'completed' or 'current' lessons. * Users cannot access a locked lesson until they have completed the preceding lessons. * Display a ‘Jump Here’ button for the set that contains the current lesson. * When all lessons are completed, a certificate will be displayed with the user's full name and the course name. |
| Lesson Page | * Path: /:course\_slug/lessons/:lesson\_id * Only logged-in user can access this page * Show lesson name and progress (percentage) * There are two lesson content types: ‘learn’ and ‘quiz’. * For learn-type lesson contents, display only the content text and a 'Continue' button. Clicking the button will take the user to the next content * For quiz-type lesson contents, there should be four choices, and the user must select the correct answer to proceed to the next content. * Users must complete lesson contents in sequence. * If all lesson contents are completed, the lesson will be marked as 'complete.' and the user will be redirected to Course Page |
| Jump Page | * Path: /:course\_slug/jump/:lesson\_id * Only logged-in user can access this page * This page will allow users to complete all lessons in the set by answering all of the quizzes in each lesson * Show set name and progress (percentage) * Users must complete lesson contents in sequence. * If all lesson contents are completed, all of the lessons in the set will be marked as 'complete.' and the user will be redirected to Course Page |
| 404 Page | * Path: /\* (all paths except pages above) * Display text “Page Not Found” with button “Back to Home” |

# ER DIAGRAM



# USE CASE DIAGRAM



# MEDIA FILES

* Frameworks (laravel, react and vue)
* SQL File (structure and records. allowed to modify)
* Postman collection
* Template HTML (based on bootstrap v5.3)

# INSTRUCTION FOR COMPETITORS

* You may create additional endpoints if necessary, but you must fulfil all of the requirements listed above.
* You can add additional fields to the response body if needed, but ensure that all required fields are included and the structure is displayed correctly.
* Create a root folder called **XX\_SERVER\_MODULE** in your local computer, where **XX** is your **computer number**.
* Database (**db-dump.sql**) and ER diagram (**db-diagram.pdf**) should be exported and saved under the root folder **<host>/XX\_SERVER\_MODULE**
* REST API should be reached on **<host>/XX\_SERVER\_MODULE/BACKEND** where xx is PC number. **No port and no /public suffix path**.
* Frontend app should be reached on **<host>/XX\_SERVER\_MODULE/FRONTEND**. **No port allowed**
* **Zip** root folder **XX\_SERVER\_MODULE** and submit to the submission page. **Make sure to exclude the vendor** and **node\_modules** folder when zipping files.