



**KISI - KISI SOAL
LOMBA KOMPETENSI SISWA
TINGKAT KABUPATEN PEMALANG
TAHUN 2025
BIDANG WEB TECHNOLOGIES**

CLIENT MODULE

CONTENTS

This module has the following files:

1. MODULE_CLIENT.docx
2. MODULE_CLIENT_MEDIA.zip

INTRODUCTION

You are asked to develop a game called **BlindMaze** using HTML and CSS and develop client-side programming using JavaScript. Some media files are available to you in a zip file. You can create more media and modify anything in the media if you want. Your game should be able to be played in a tablet resolution (1000 x 600 pixels). In bigger resolution, the game must be centered in the screen both horizontally and vertically.

DESCRIPTION OF PROJECTS AND TASKS

This is a **3 hours module**. You can create the layout using HTML/CSS and create the functionality of the game using JavaScript that allows the game to work correctly in different web browsers. **EcmaScript 6 / Javascript Modules** not allowed for this module. This module will be marked with directly opening the **index.html** page using **Google Chrome**.

BlindMaze game screen should have meet these requirements below:

1. Username
2. Gameboard
3. Player Lives
4. Maze
5. Timer
6. Stage number

Game Functionalities

1. Game should be on a single page application (index.html). No refresh / reload page and additional html page for any action.
2. **Show game welcome** in the center after pages are loaded.
3. **Players can Start the game** after filling the username field and click the **"Play Game"** button at the bottom of the welcome page.
4. **The "Play Game" button should be disabled** if the user did not input the username.
5. **Game instructions should be shown** when the user clicks the "Show instructions" button.
6. **Leaderboard modal should be open and show saved score descending** when the user clicks the "Leaderboard" button.
7. **Users can close the leaderboard modal** by clicking the **"X"** button.
8. Click the **"Play Game"** button to start the game.
9. **These elements should present** when the game started:
 - a. 10x10 grid
 - b. Stage number
 - c. Time left
 - d. 5 HP
10. The start point is a **single random block on the most left side column**.
11. The finish point is a **single random block on the most right side column**.
12. When the game starts, **generate random walls in the grid**. Walls are represented by **gray color blocks**.
13. You are **not allowed** to generate walls **statically**.
14. Generated walls **must not block ways** from start to finish.
15. There are two phases in a stage:
 - a. **Memorizing time**: player should **memorize the walls** and the route to finish point for 10 seconds. Player cannot move at this time.
 - b. **Move time**: after memorizing time ends, the player has 20 seconds to move to the finish point without hitting the wall.
16. **These walls are only visible in memorizing time**, then disappear when the moving time starts.

17. Players can move with **arrow keys**.
18. **If a player hits a wall**, show an alert, reduce the number of lives by 1 and restart the stage.
19. **If a player runs out of time**, show an alert, reduce the number of lives by 1 and restart the stage.
20. **If a player reaches the finish point**, show a success alert then the player will move to the next stage.
21. Player has **one chance to get a hint** by clicking the "**Hint button**".
22. When the hint button is clicked at the Move Time, the walls **become visible for one second** and then disappear.
23. Every time the user is moving to the next stage, **generate new random walls** and memorizing time starts.
24. Every new stage will generate **new random walls, random starting point, and random finish point**.
25. Game will be over **if the user loses all their lives**.
26. **A confirmation popup appears** when the game is over.
27. The game over popup should show:
 - a. Username
 - b. Stage number
 - c. Button to save score
28. **You can develop your own algorithm** to generate the walls. The generated walls should not prevent the player from moving to the finish point.
29. The game should work correctly on Google Chrome without **console error**.
30. You may see the video provided (**example.mp4**) for more details.

INSTRUCTION FOR COMPETITORS

1. Create a root folder called **XX_CLIENT_MODULE** in your local computer, where **XX** is your **computer number**.
2. Place your works inside the root folder. Make sure your works are working well when directly opening the **index.html** file.
3. Zip your root folder **XX_CLIENT_MODULE** and submit to the submission page.\

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

A. 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	<pre>{ "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" } }</pre>

A1b: Response Invalid Field

Status Code	400
Body	<pre>{ "status": "error", "message": "Invalid field(s) in request", "errors": { "full_name": ["The full name field is required."] } }</pre>

	<pre>], "username": ["The username has already been taken."], "password": ["The password field is required."]] } } </pre>
--	---

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	<pre> { "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" } } </pre>

A2b: Response Invalid Credentials

Status Code	400
Body	<pre>{ "status": "authentication_failed", "message": "The username or password you entered is incorrect" }</pre>

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	<pre>{ "status": "success", "message": "Logout successful" }</pre>

A3b: Response Invalid Token

Status Code	401
Body	<pre>{ "status": "invalid_token", "message": "Invalid or expired token" }</pre>

B. 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	<pre>{ "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 } }</pre>

B1b: Response Invalid Field

Status Code	400
Body	<pre>{ "status": "error", "message": "Invalid field(s) in request", "errors": {</pre>

	<pre> "name": ["The name field is required."], "slug": ["The slug has already been taken."] } } </pre>
--	--

B1c: Response Forbidden

Status Code	403
Body	<pre> { "status": "insufficient_permissions", "message": "Access forbidden" } </pre>

B1d: Response Invalid Token

Status Code	401
Body	<pre> { "status": "invalid_token", "message": "Invalid or expired token" } </pre>

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	<pre>{ "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.." } }</pre>

B2b: Response Invalid Field

Status Code	400
Body	<pre>{ "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."] } }</pre>

B2c: Response Not Found

Status Code	404
Body	<pre>{ "status": "not_found", "message": "Resource not found" }</pre>

B2d: Response Forbidden

Status Code	403
Body	<pre>{ "status": "insufficient_permissions", "message": "Access forbidden" }</pre>

B2e: Response Invalid Token

Status Code	401
Body	<pre>{ "status": "invalid_token", "message": "Invalid or expired token" }</pre>

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	<pre>{ "status": "success", "message": "Course successfully deleted" }</pre>

B3b: Response Not Found

Status Code	404
Body	<pre>{ "status": "not_found", }</pre>

	<pre>"message": "Resource not found" }</pre>
--	--

B3c: Response Forbidden

Status Code	403
Body	<pre>{ "status": "insufficient_permissions", "message": "Access forbidden" }</pre>

B3d: Response Invalid Token

Status Code	401
Body	<pre>{ "status": "invalid_token", "message": "Invalid or expired token" }</pre>

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	<pre>{ "status": "success", "message": "Courses retrieved successfully", "data": { "courses": [{ "id": 1, "name": "Web Dev Fundamentals", "slug": "web-dev-fundamentals", </pre>

	<pre> "description": "Grasp the core..", "is_published": 1, "created_at": "2024-08-08T02...", "updated_at": "2024-08-08T02..." }, ...] } }</pre>
--	---

B4b: Response Invalid Token

Status Code	401
Body	<pre> { "status": "invalid_token", "message": "Invalid or expired token" }</pre>

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	<pre> { "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..." } }</pre>

```

    "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"
                  }
                ]
              }
            ]
          }
        ]
      }
    ],
  },

```

	<pre>], },] } } </pre>
--	--------------------------------

B5b: Response Invalid Token

Status Code	401
Body	<pre> { "status": "invalid_token", "message": "Invalid or expired token" } </pre>

C. 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	<pre> { "status": "success", "message": "Set successfully added", "data": { "name": "Web API Architecture Fundamentals", "order": 2, </pre>

	<pre>"id": 5 } }</pre>
--	------------------------

C1b: Response Invalid Field

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

C1c: Response Forbidden

Status Code	403
Body	<pre>{ "status": "insufficient_permissions", "message": "Access forbidden" }</pre>

C1d: Response Invalid Token

Status Code	401
Body	<pre>{ "status": "invalid_token", "message": "Invalid or expired token" }</pre>

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	<pre>{ "status": "success", "message": "Set successfully deleted" }</pre>

C2b: Response Invalid Field

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

C2c: Response Forbidden

Status Code	403
Body	<pre>{ "status": "insufficient_permissions", }</pre>

	<pre>"message": "Access forbidden" }</pre>
--	--

C2d: Response Invalid Token

Status Code	401
Body	<pre>{ "status": "invalid_token", "message": "Invalid or expired token" }</pre>

D. 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: <ul style="list-style-type: none"> • type ('learn' or 'quiz') • content • options: (when type is 'learn'). it's object of: <ul style="list-style-type: none"> ◦ option_text ◦ is_correct (true/false)

D1a: Response Success

Status Code	201
Body	<pre>{ "status": "success", }</pre>

	<pre> "message": "Lesson successfully added", "data": { "name": "Introduction to Web API..", "order": 1, "id": 10 } </pre>
--	--

D1b: Response Invalid Field

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

D1c: Response Forbidden

Status Code	403
Body	<pre> { "status": "insufficient_permissions", "message": "Access forbidden" } </pre>

D1d: Response Invalid Token

Status Code	401
Body	<pre> { "status": "invalid_token", "message": "Invalid or expired token" } </pre>

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	<pre>{ "status": "success", "message": "Lesson successfully deleted" }</pre>

D2b: Response Not Found

Status Code	404
Body	<pre>{ "status": "not_found", "message": "Resource not found" }</pre>

D2c: Response Forbidden

Status Code	403
Body	<pre>{ "status": "insufficient_permissions", "message": "Access forbidden" }</pre>

D2d: Response Invalid Token

Status Code	401
Body	<pre>{ "status": "invalid_token", "message": "Invalid or expired token" }</pre>

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	<pre>{ "status": "success", "message": "Check answer success", "data": { "question": "What does HTML stand for?", "user_answer": "Hyper Text Markup..", "is_correct": true } }</pre>

D3b: Response Only for Quiz Type

Status Code	400
Body	<pre>{ "status": "error", "message": "Only for quiz content" }</pre>

D3c: Response Not Found

Status Code	404
Body	<pre>{ "status": "not_found", "message": "Resource not found" }</pre>

D3d: Response Forbidden

Status Code	403
Body	<pre>{ "status": "insufficient_permissions", "message": "Access forbidden" }</pre>

D3e: Response Invalid Token

Status Code	401
Body	<pre>{ "status": "invalid_token", "message": "Invalid or expired token" }</pre>

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	<pre>{ "status": "success", "message": "Lesson successfully completed" }</pre>

D4b: Response Not Found

Status Code	404
Body	<pre>{ "status": "not_found", "message": "Resource not found" }</pre>

D4c: Response Forbidden

Status Code	403
Body	<pre>{ "status": "insufficient_permissions", "message": "Access forbidden" }</pre>

D4d: Response Invalid Token

Status Code	401
Body	<pre>{ "status": "invalid_token", "message": "Invalid or expired token" }</pre>

E. 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	<pre>{ "status": "success", "message": "User registered successful" }</pre>

E1b: Response Already Registered

Status Code	400
Body	<pre>{ "status": "error", "message": "The user is already registered for this course" }</pre>

E1c: Response Not Found

Status Code	404
Body	<pre>{ "status": "not_found", "message": "Resource not found" }</pre>

E1d: Response Invalid Token

Status Code	401
Body	<pre>{ "status": "invalid_token", "message": "Invalid or expired token" }</pre>

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	<pre>{ "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 }] }] } }</pre>

E2b: Response Invalid Token

Status Code	401
Body	<pre>{ "status": "invalid_token", "message": "Invalid or expired token" }</pre>

PHASE 2: Front-End Development

Pages	Features
General	<ul style="list-style-type: none">• 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	<ul style="list-style-type: none">• Path: /register• Users can register using following fields:<ul style="list-style-type: none">○ 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	<ul style="list-style-type: none">• 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	<ul style="list-style-type: none">• Path: /• Only logged-in user can access this page

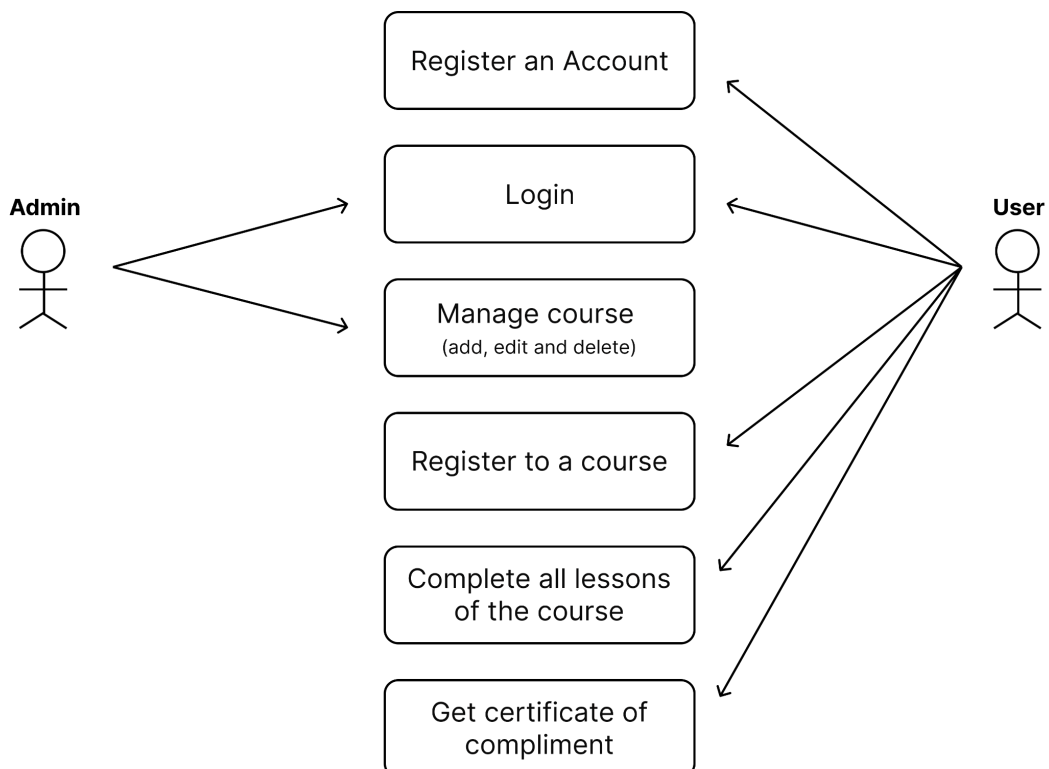
	<ul style="list-style-type: none"> • Display the user's registered courses in one section and other available courses in a separate section. • Display course name and course description correctly
Course Page	<ul style="list-style-type: none"> • Path: <code>/:course_slug</code> • Only logged-in user can access this page • If the user is not registered for the course, display: <ul style="list-style-type: none"> ◦ 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: <ul style="list-style-type: none"> ◦ 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	<ul style="list-style-type: none"> • Path: <code>/:course_slug/lessons/:lesson_id</code> • 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

	<p>the user to the next content</p> <ul style="list-style-type: none"> • 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	<ul style="list-style-type: none"> • Path: <code>/:course_slug/jump/:lesson_id</code> • 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	<ul style="list-style-type: none"> • Path: <code>/*</code> (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.**