KISI-KISI SOAL / DESKRIPSI TEKNIS LOMBA KOMPETENSI SISWA (LKS) DIKMEN (SMK/SMA/MA/MAK) TINGKAT PROVINSI JAWA TIMUR TAHUN 2025

BIDANG LOMBA :

Web Technologies

LKS DIKMEN
TINGKAT PROVINSI JAWA TIMUR
(Jenjang SMK/SMA/MA/MAK)

DINAS PENDIDIKAN PROVINSI JAWA TIMUR
Jl. Gentengkali No. 33 Surabaya, Jawa Timur, 60275
Telp. (031) 5342706 / 5342709, Website : dindik@jatimprov.go.id

Kisi-kisi LKS Jawa Timur
Web Technologies
2025

SPEEDTEST MODULE

CONTENTS

This module has the following files:

1. MODULE_SPEEDTEST.docx
2. MODULE_SPEEDTEST_MEDIA.zip

A1. Table Pricing

Create a table pricing design consisting of 3 tables, title, price, description list, purchase button.

A2. Landing News Page

Create a vacation landing page consisting of navigation bar, logo, menu, list of 3 destinations (title, picture, short description, total views, total comments, read more button).

B1. Particle Button

Create an interactive animation using CSS only. When user click the “Click Me!” Button, many particles with diffusion effect will be displayed around the button.
You can only edit the style.css file.

B2. 5 Stars Rating Animation

Create a 5 Stars Rating UI using the provided "stars.png" image.
When you hover your mouse over a star, an animation that smoothly fills in from the left to the star's score proceeds.
Score 0.5 point unit: 0, 0.5, 1, 1.5, 2, 2.5, 3, 3.5, 4, 4.5, 5
Example.

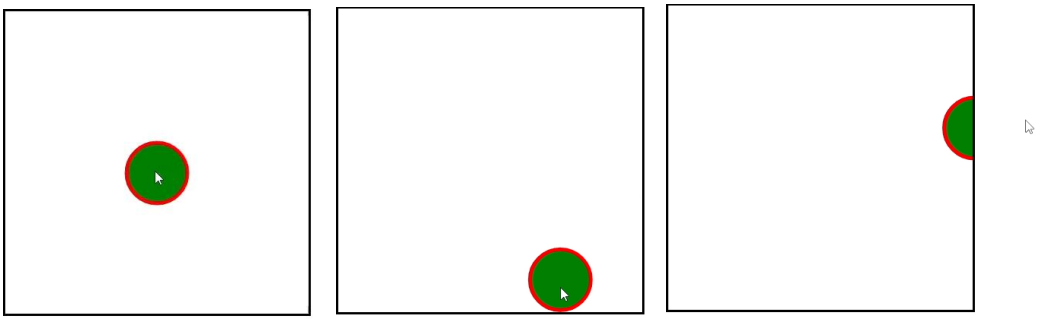
Refer to **animation.mp4** for the star filling animation.

C1. Mouse Style

Change the style of the mouse. When clicked, it should display an animation. See **mouse.mp4** in the given folder to understand the functionality.

C2. Canvas Hover

Draw a ball with border in a container of 400px height and 400px width using canvas. Make "ball" follow mouse on canvas.

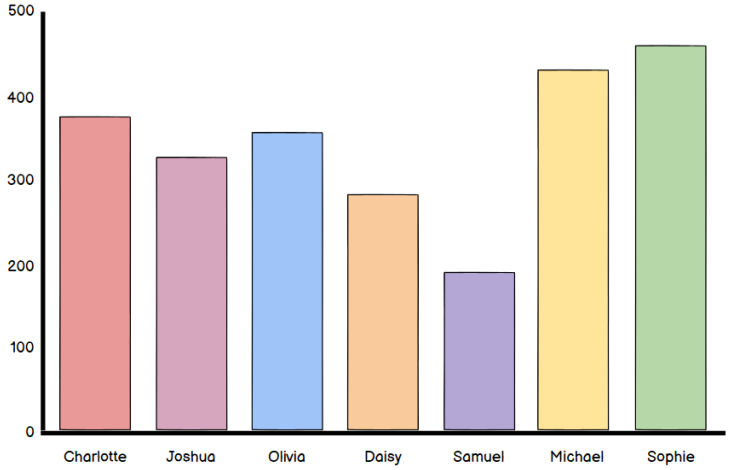


D1. Bar Chart

"random.php" that generates random data to create charts is provided.
Draw a bar chart using all data from "random.php".

- Y tick labels (vertical axis number) must not have a decimal point, and there must be at least 5.
- The number of bars is randomly selected between 5 and 15.
- The color of the bars is applied randomly.
- For X Axis Data Labels, name must appear

random.php data
[{ name: "Charlotte", value: 380, }, ...]
Example



D2. Generate File DOC

- In a form you must insert a title and then its content, when submitting the form you must download a DOC document where you can open it and view the content in the DOC file.
- The file must be called as the title of the form.
- The content must be included in the file with its respective title and paragraph.

Example video: **create_file.mp4**

An HTML_TO_DOC PHP class is provided.

E1. Answer Checker

You are provided with two CSV files. One file contains the actual answers, and the other file contains the submitted answers. You are expected to develop a web page, displaying a table showing the question number, actual answer and submitted answer. At the bottom of the table, display the scores for this submission. Eg, if there are 8 correct answers and there are 10 questions. Display the score as “8/10”.

Screen shot:

Question	Actual Answer	Submitted Answer
1	A	A
2	B	A
3	B	C
4	C	C
5	D	D
6	A	C
7	B	B
8	C	A
9	D	D
10	C	C

Score: 6/10

E2. Array Manipulation

Create an array containing the numbers 1 to 40. When the file is run using index.php?factor=n, the content of the array changes to indicate which are the numbers that are multiple of n.

Example: For example index.php?factor=4

```
Modified Array
(
    [0] => 1
    [1] => 2
    [2] => 3
    [3] => 4 is a multiple of 4**
    [4] => 5
    [5] => 6
    [6] => 7
    [7] => 8 is a multiple of 4**
    [8] => 9
    [9] => 10
    [10] => 11
    [11] => 12 is a multiple of 4**
    [12] => 13
    [13] => 14
    [14] => 15
    [15] => 16 is a multiple of 4**
    [16] => 17
    [17] => 18
    [18] => 19
    [19] => 20 is a multiple of 4**
    [20] => 21
    [21] => 22
    [22] => 23
    [23] => 24 is a multiple of 4**
    [24] => 25
    [25] => 26
    [26] => 27
    [27] => 28 is a multiple of 4**
    [28] => 29
    [29] => 30
    [30] => 31
    [31] => 32 is a multiple of 4**
    [32] => 33
    [33] => 34
    [34] => 35
    [35] => 36 is a multiple of 4**
    [36] => 37
    [37] => 38
    [38] => 39
    [39] => 40 is a multiple of 4**
)
```

CLIENT SIDE MODULE

CONTENTS

This module has the following files:

1. MODULE_CLIENT_SIDE.docx
2. MODULE_CLIENT_SIDE_MEDIA.zip

INTRODUCTION

You are asked to develop a game called **Shooter** 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 needs to be developed 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 module of 3 hours. 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.

Shooter game screen should have meet these requirements below:

1. Player Name
2. Gameboard
3. Gun Type
4. Target
5. Total Score
6. Timer
7. Shoot History

Design and Initial Layout

1. **Develop the initial markup (HTML + CSS) of your game application.** Overall screen must be within 1000 x 600 pixels and centered on the screen.
2. **You are free to decorate** the game screen design as long as it meets the requirements.
3. **The HTML and CSS** code must be valid in the W3C standards for HTML5 and CSS3 rules in accordance with the WCAG and standard ARIA (Accessible Rich Internet Applications Suite)

Game Functionalities

1. **Show game welcome** in the center after pages are loaded.
2. **Players can go to the game** after filling the username field and click the "**Play Game**" button at the bottom of the welcome page.
3. **The "Play Game" button should be disabled** if the user did not input the username.
4. User must choose one from each option:
   a. one of three levels (easy, medium, hard).
   b. one of two guns to be used.
   c. one of three targets to be used.
5. **Game instructions should be shown** after the page is loaded.
6. **After clicking the "Instruction"** button, it shows game instructions.
7. **Users can close instructions** after clicking the **"X"** button.
8. **Show countdown for three seconds in the center of screen** after the user clicked the play button before the game started playing.
9. **When the game starts,** the timer will start with time according to level.
   a. '30 seconds' for easy level
   b. '20 seconds' for medium level
   c. '15 seconds' for hard level
10. **Add target randomly** every 3 seconds.
11. **The target will appear** with animation.
12. **When the game starts, there will be 3 random targets**
13. The pointer will appear in the center of the screen and move with the mouse
14. **gun** will move follow pointer motion.
15. **Player can shoot with click**
16. **Target will disappears** after being hit by the shoot.
17. **If the shot does not hit the target, the time will be reduced by 5 seconds**
18. **The score will be increased** if the player can shoot to target.
19. Players can change guns after pressing the space button
20. There is an animation when changing guns.
21. **Players can pause** the game.
22. Press **Esc** to open the **pause popup**. The game should be in a paused state when opening the popup.

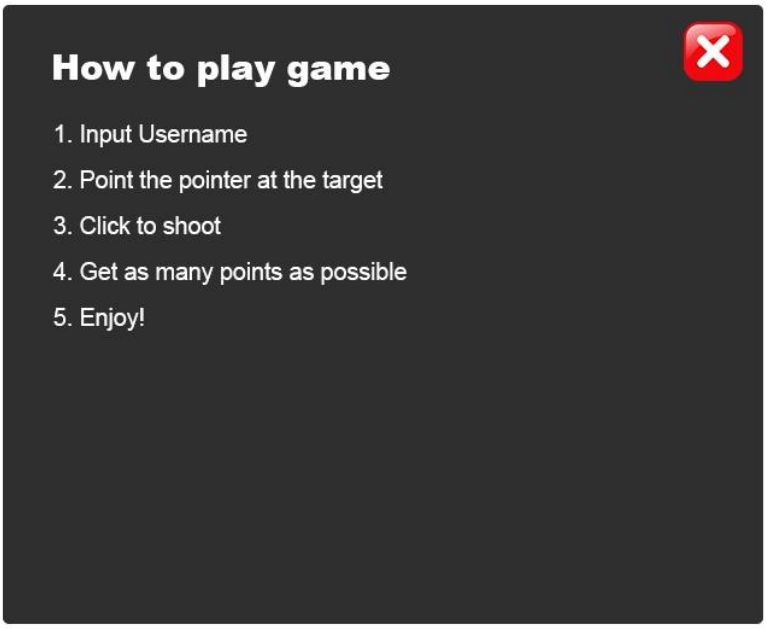
23. Press **Esc** again to **continue** or click the "**continue"** button and the countdown should appear before the game continues.
24. **Game Over** when the timer time is up.
25. **Show popup after game over** to display the player username and scores, save match history button, and restart button.
26. **Match results should be saved in the local storage** after the player clicks the "**Save Score"** button.
27. **Players are able to** see the match history after clicking the Match History Button.
28. **Shoot history can be sorted** by score or last matches based on user choices.
29. **The game needs to work correctly** in Google Chrome.

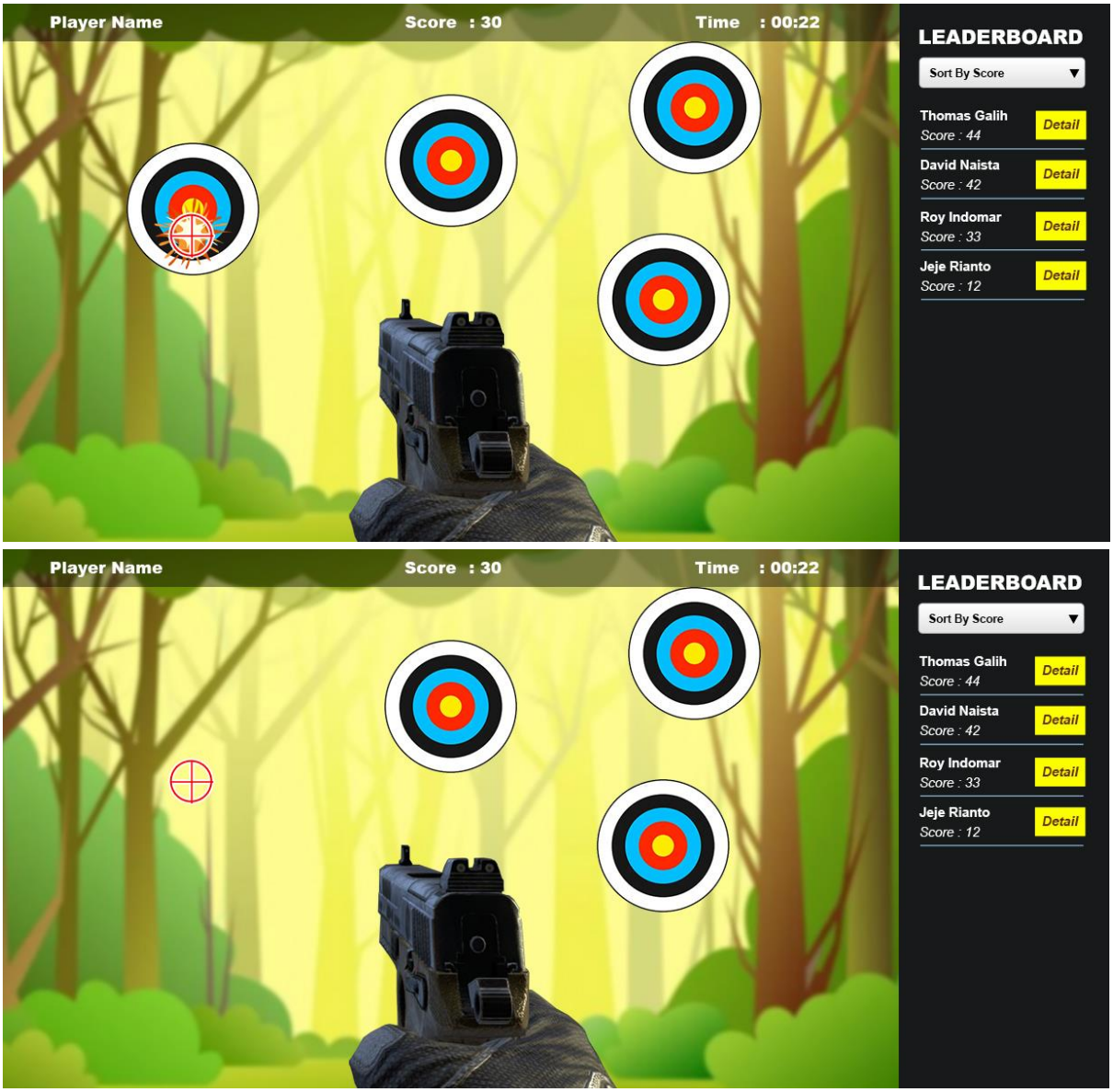
EXAMPLE

These following images are for example purposes only. You may design your own game layout.

Welcome Screen

Show Instruction





INSTRUCTION FOR COMPETITORS

1. The media files are available in the ZIP file. You can modify the supplied files and create new media files to ensure the correct functionality and improve the application.
2. The entry file should be '**XX_CLIENT_SIDE/index.html'**
3. You should create additional images for each of the requested resolutions to highlight hidden elements, animations, interactions, or any additional information that will assist in the presentation of the game design.
4. Save the working game to the directory on the server named "**XX_CLIENT_SIDE**". Be sure that your main file is called **index.html**.
5. You are responsible for the time management in your development. If you finalize some tasks you can continue to other tasks.

SERVER SIDE MODULE

Contents

1. MODULE_SERVER_SIDE.docx
2. MODULE_SERVER_SIDE_MEDIA.zip

Introduction

A new founded company is looking for full stack developers to create an online browser gaming platform. Game developers can upload their games to the platform and users can play them online in the browser.
There are three parts to the platform:

- Developer Portal: A web application for game developers to upload their games to the platform.
- Administrator Portal: A web application for administrators to manage the platform with its users and games.
- Gaming Portal: A web application for players to play games online in the browser.

The company wants to create a minimum viable product (MVP) for the platform. The MVP should already contain the aforementioned parts, but it is acceptable that the Game Developer Portal and the Administrator.
Portal are not fleshed out yet. The Gaming Portal should be fully functional, so that users can play games online in the browser.

Description of Projects

The project is split into two phases:

- Phase one for building the API and static pages using a PHP framework and MySQL database.
- Phase two for building the frontend parts using HTML/CSS and a JavaScript framework, consuming the API developed in phase one.

You can find the provided media files to support your work:

- Provided frameworks (laravel, vuejs, reactjs)
- Postman collection and environment
- Template GUI (to build frontend UI)
- lks-server.sql (a database with structures and dummy data)

Phase 1 : RESTful API

In this phase, you should build a RESTful API using the Laravel framework according to the documentation below.

Ensure users can login using credentials below:

Administrator

Username	Password
admin1	hellouniverse1!
admin2	hellouniverse2!

Developer

Username	Password
dev1	hellobyte1!
dev2	hellobyte2!

Players

Username	Password
player1	helloworld1!
player2	helloworld2!

REST API Specification

General information:

- The response bodies contain some static example data. Dynamic data from the database should be used.
- Placeholder parameters in the URL are marked with a preceding colon (e.g. :slug or :id).
- The order of properties in objects does not matter, but the order in an arrays does.
- The Content-Type header of a request must always be application/json for POST, PUT, PATCH.
- The Content-Type header of a response is always application/json unless specified otherwise.
- Timestamps are formatted as ISO-8601 strings. E.g. 2032-01-31T21:59:35.000Z.
- The given URLs are relative to the base URL of the API. E.g. /api/v1/games is the URL to get all games.
- The API URLs must not end in .php or .html or any other file extension. The game files are an exception to this.
- The token for protected endpoints must be specified as a Bearer token in the `Authorization` header. I.e. `Authorization: Bearer <token>`

1. Authentication

You should create Login and Logout endpoints. The accessToken must be generated by sanctum and will be placed in the request headers Authorization Bearer.

Sign Up

POST [domain]/api/v1/auth/signup

This endpoint creates a new user and returns a session token.

Request Body:

```
{
   "username": "testuser",
   "password": "asdf1234"
}
```

PROPERTY	COMMENT
username	required, unique, min length 4, max length 60
password	required, min length 5, max length 10

Response:

Successful creation response:

Status Code: 201

Response Body:

```
{
   "status": "success",
   "token": "xxx"
}
```

Sign In

POST [domain]/api/v1/auth/signin

This checks the username and password against all known users. If found, a session token is returned.

Valid response:

Status Code: 200

Request Body**:**

```
{
   "username": "testuser",
   "password": "asdf1234"
}
```

PROPERTY	DESCRIPTION
username	required, unique, min length 4, max length 60
password	required, min length 5, max length 10

Response Body:

```
{
```

```
    "status": "success",
    "token": "xxx"
}
```


Wrong username / password response:

Status Code: 401

Response Body:


```
{
    "status": "invalid",
    "message": "Wrong username or password"
}
```


Sign Out

POST [domain]/api/v1/auth/signout
Deletes the current session token.

Valid response:

Status Code: 200

Response Body:


```
{
    "status": "success"
}
```


2. Users

Get all admin data
GET [domain]/api/v1/admins
Returns an admin data.

Response:

Status Code: 200


```
{
  "totalElements": 2,

  "content": [
    {
      "username": "admin1",
      "last_login_at": "",
      "created_at": "2024-04-05 20:55:40",
      "updated_at": "2024-04-05 20:55:40",
    },
    {
      "username": "admin2",
      "last_login_at": "2024-04-05 20:55:40",
      "created_at": "2024-04-05 20:55:40",
      "updated_at": "2024-04-05 20:55:40",
    }
  ]
}
```


User is not administrator response:

Status Code: 403

Response Body:

```
{
   "status": "forbidden",
   "message": "You are not the administrator"
}
```

POST [domain]/api/v1/users

This endpoint can be used to create a user.

Request Body:

```
{
   "username": "testuser",
   "password": "asdf1234"
}
```

PROPERTY	COMMENT
username	required, unique, min length 4, max length 60
password	required, min length 5, max length 10

Response:

Successful creation response:

Status Code: 201

Response Body:

```
{
   "status": "success",
   "username": "testuser"
}
```

Existing username:

If the username is not unique, the admin user cannot be created and instead the following response is returned.

Response:

Status Code: 400

```
{
   "status": "invalid",
   "message": "Username already exists"
}
```

User is not administrator response:

Status Code: 403

Response Body:

```
{
```

```
    "status": "forbidden",
    "message": "You are not the administrator"
}
```


GET [domain]/api/v1/users

Returns a user details.

Response:

Status Code: 200


```
{
  "totalElements": 2,

  "content": [
    {
      "username": "player1",
      "last_login_at": "",
      "created_at": "2024-04-05 20:55:40",
      "updated_at": "2024-04-05 20:55:40",
    },
    {
      "username": "player2",
      "last_login_at": "2024-04-05 20:55:40",
      "created_at": "2024-04-05 20:55:40",
      "updated_at": "2024-04-05 20:55:40",
    }
  ]
}
```


User is not administrator response:

Status Code: 403

Response Body:


```
{
    "status": "forbidden",
    "message": "You are not the administrator"
}
```


PUT [domain]/api/v1/users/:id

This endpoint can be used to update a user.

Request Body:


```
{
    "username": "testuser",
    "password": "asdf1234"
}
```


PROPERTY	COMMENT
username	required, unique, min length 4, max length 60
password	required, min length 5, max length 10

Response:

Successful creation response:

Status Code: 201

Response Body:

```
{
   "status": "success",
   "username": "testuser"
}
```

Existing username:

If the username is not unique, the admin user cannot be created and instead the following response is returned.

Response:

Status Code: 400

```
{
   "status": "invalid",
   " message": "Username already exists"
}
```

User is not administrator response:

Status Code: 403

Response Body:

```
{
   "status": "forbidden",
   "message": "You are not the administrator"
}
```

DELETE [domain]/api/v1/users/:id

This endpoint can be used to update a user.

Successful deletion response:

Status Code: 204

This returns an empty body. The `Content-Type` header does not have to be

`application/json`.

User is not found response:

Status Code: 403

Response Body:

```
{
   "status": "not-found",
   "message": "User Not found"
}
```

User is not administrator response:

Status Code: 403

Response Body:

```
{
   "status": "forbidden",
   "message": "You are not the administrator"
}
```

3. Games

GET [domain]/api/v1/games
Returns a paginated list of games.

Query Parameters:

PROPERTY	DESCRIPTION	DEFAULT
page	Page number. Starts at 0	0
size	Page size. Must be greater or equal than 1	10
sortBy	Field to sort by. Must be one of "title", "popular", "uploaddate"	title
sortDir	Describes sort direction. Must be one of "asc" or "desc"	asc

The sort fields are explained here:

FIELD	DESCRIPTION
title	Game title
popular	Counts the total number of scores per game and sorts by this count
uploaddate	Latest game version upload timestamp

In `content`, the fields `thumbnail` and `uploadTimestamp` refer only to the latest version.

The field `scoreCount` is the sum of scores over all versions.

Response:

Status Code: 200

Response Body:

```
{
  "page": 0,
  "size": 10,
  "totalElements": 15,

  "content": [
    {
      "slug": "demo-game-1",
      "title": "Demo Game 1",
      "description": "This is demo game 1",
      "thumbnail": "/games/:slug/:version/thumbnail.png",
```

```
    "uploadTimestamp": "2032-01-31T21:59:35.000Z",
    "author": "dev1",
    "scoreCount": 5
   }
 ]
}
```

Response page fields explained:

FIELD	DESCRIPTION
page	The requested page number. Starts at 0
size	The actual page size. Must be less or equal than the requested page size
totalElements	The total number of elements regardless of page
content	An array of the games in the page

It can be computed how many pages there are:

pageCount = ceil(totalElements / requestedSize)

It can also be computed if the returned page is the last page by multiplying the (page+1) by requested page size and checking if the result is less than or equal to the total elements.

isLastPage = (page + 1) * requestedSize >= totalElements

Note 1: If there is a game that has no game version yet, it is not included in the response nor the total count.
Note 2: If there is no thumbnail, the thumbnail field is null.

POST [domain]/api/v1/games

This endpoint can be used to create a game. However, the game version needs to be uploaded in a separate step. If a game does not have a version yet, it is not returned in this endpoint.

Request Body:

```
{
   "title": "Demo Game 3",
   "description": "This is demo game 3"
}
```

PROPERTY	DESCRIPTION
title	required, min length 3, max length 60
description	required, min length 0, max length 200

Response:

Successful creation response:

Status Code: 201

Response Body:

```
{
   "status": "success",
   "slug": "generated-game-slug"
}
```

Existing slug:

If the generated slug is not unique, the game cannot be created and instead the following response is returned.

Response:

Status Code: 400

```
{
   "status": "invalid",
   "slug": "Game title already exists"
}
```

GET [domain]/api/v1/games/:slug
Returns a games details.

Response:

Status Code: 200

```
{
  "slug": "demo-game-1",
  "title": "Demo Game 1",
  "description": "This is demo game 1",
  "thumbnail": "/games/:slug/:version/thumbnail.png",
  "uploadTimestamp": "2032-01-31T21:59:35.000Z",
  "author": "dev1",
  "scoreCount": 5,
  "gamePath": "/games/demo-game-1/1/"
}
```

If there is no thumbnail, the thumbnail field is null.

The `gamePath` field points to a URL path that browsers can use to render the game. This means this is a reachable asset path.

Game file upload

POST [domain]/api/v1/games/:slug/upload
The user can upload a new version of a game if they are the author of that game.

- This is not a REST endpoint and rather it accepts a file upload. The parameter name is `zipfile`.
- The version of the game is an integer and incrementing. The first version is `1`. The user cannot control the version.
- The session token needs to be provided as form parameter `token`.
- The path has to be stored in the game record, so that players can find the game files.

If the upload fails because of one of these possible reasons, the response must be a plain text explanation of the error.

- User is not author of the game

Serve game files

GET /games/:slug/:version/

The game files that were uploaded are served under that path which is public.

PUT [domain]/api/v1/games/:slug

This endpoint allows the author of the game to update the game title and description.

Request Body:

```
{
   "title": "Demo Game 1 (updated)",
   "description": "Updated description"
}
```

Note: This does not update the game's slug.

Response:

Successful update response:

Status Code: 200

Response Body:

```
{
   "status": "success"
}
```

User is not game author response:

Status Code: 403

Response Body:

```
{
   "status": "forbidden",
   "message": "You are not the game author"
}
```

DELETE [domain]/api/v1/games/:slug

The author can delete their game. This deletes the game, all versions and all scores.

Response:

Successful deletion response:

Status Code: 204

This returns an empty body. The `Content-Type` header does not have to be `application/json`.

User is not game author response:

Status Code: 403

Response Body:

```
{
   "status": "forbidden",
   "message": "You are not the game author"
}
```

GET [domain]/api/v1/users/:username

Returns the user details.
Response:

Status Code: 200

Response Body:

```
{
   "username": "dev1",
   "registeredTimestamp": "2032-01-31T21:59:35.000Z",
   "authoredGames": [
      {
         "slug": "demo-game-1",
         "title": "Demo Game 1",
         "description": "This is demo game 1"
      }
   ],
   "highscores": [
      {
         "game": {
            "slug": "demo-game-1",
            "title": "Demo Game 1",
            "description": "This is demo game 1"
         },
         "score": 15,
         "timestamp": "2032-01-31T21:59:35.000Z"
      }
   ]
}
```

The authoredGames is an array that returns all games with at least one version where this user is the author. If the user requesting the user details is the user itself, this returns also games that have no version yet.

The highscores is an array of highest scores per game played.

GET [domain]/api/v1/games/:slug/scores

Returns the highest scores of each player that played any version of the game, sorted by score (descending).

Response:

Status Code: 200

Response Body:

```
{
   "scores": [
      {
         "username": "player2",
         "score": 20,
         "timestamp": "2032-01-31T21:59:35.000Z"
      },
      {
         "username": "player1",
         "score": 15,
         "timestamp": "2032-01-31T21:59:35.000Z"
```

```
      }
   ]

}
```

POST [domain]/api/v1/games/:slug/scores

When a user ends a game run, the score can be posted to this endpoint.

Request Body:

```
{
   "score": 100
}
```

The game version associated to the score is the latest one available.

Response:

Successful creation response:

Status Code: 201

Response Body:

```
{
   "status": "success"
}
```

Invalid request body

If the POST or PUT request had invalid fields, they are validated and a response is returned that lists the violations.

Status Code: 400

Response Body:

```
{
   "status": "invalid",
   "message": "Request body is not valid.",
   "violations": {
      "field_name": {
         "message": "required"
      },
      "field_name": {
         "message": "must be at least 4 characters long"
      },
      "field_name": {
         "message": "must be at most 60 characters long"
      }
   }
}
```

In the above example, all possible violations are shown. The actual returned violations should only be the fields which were actually invalid. At most one validation per field is shown. Validations are executed in order of appearance in the example above.
field_name must be replaced with the actual field name.
The messages for length must include the actual length requirement value.

Missing or invalid auth header

If the consumer makes a call to a path that requires the auth header to be present, this must be the response:

Status Code: 401

Response Body:

```
{
  "status": "unauthenticated",
  "message": "Missing token"
}
```

If the consumer makes a call to a path that requires the auth header to be present, but provides an invalid or not existing token, this must be the response:

Status Code: 401

Response Body:

```
{
  "status": "unauthenticated",
  "message": "Invalid token"
}
```

If the consumer makes a call to a path that requires the auth header to be present, but the user is blocked, this must be the response:

Status Code: 403

Response Body:

```
{
  "status": "blocked",
  "message": "User blocked",
  "reason": "You have been blocked by an administrator"
}
```

Note: The reason is a dynamic value, chosen by the admin that blocks the user.

The following method and path patterns require a valid session header to be present:

- POST /api/v1/auth/signout
- POST, PUT, DELETE /api/v1/games/**
- GET, POST, PUT, DELETE /api/v1/users/**
- GET /api/v1/admins/**

Non-existing API path

If the consumer makes a call to a non-existing path, or a resource that does not exist, this must be the response:

Status Code: 404

Response Body:

```
{
  "status": "not-found",
  "message": "Not found"
}
```

Phase 2 : Front-end Development

In this part, you should build a front-end application using one of the provided frameworks (vue js or react js). You can use the provided template gui on the media files to build the front-end ui.

Pages / Features	Description
Sign Up, Sign In, Sign Out	- User can sign in using the correct credentials (username and password) - Alert errors should be displayed when login failed - Users can sign out by clicking the logout button on the navbar. - 2 level user can login (administrator and user) - Administrator menu (List Admin, List User) - User menu (Discover Games, Manage Games, User Profile)
Home	- Users can see their login info.
List Admin	- The admin can see all admin users in the database with username, created timestamp and last login timestamp.
List User	- The admin can see all platform users with username, registration timestamp, last login timestamp and click a link to their profile page - The admin can block and unblock users - The admin can create, update and delete users

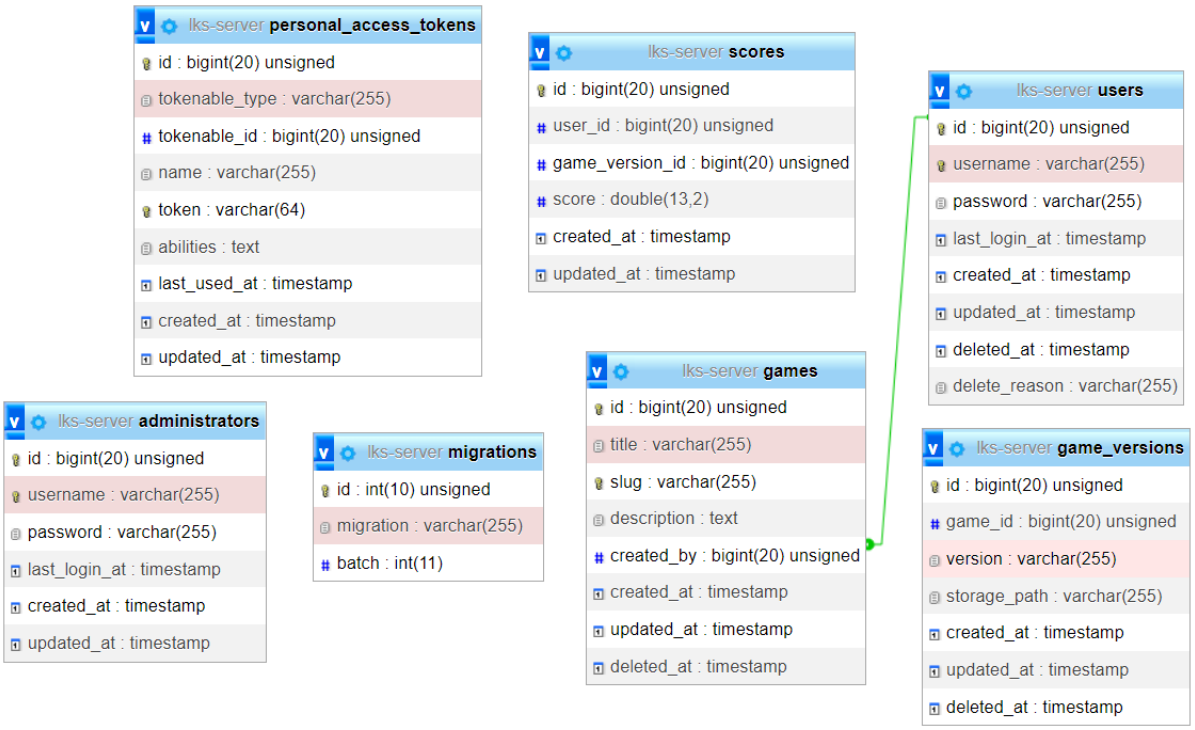
Discover games	- See a list of games. For each game, the following is shown:
	  ○ Title
	  ○ Score Count: Number of scores submitted
	  ○ Description: The description provided by the author.
	  ○ Thumbnail: If available, otherwise a default graphic.
	  ○ The list can be sorted by popularity, recently updated, or title. Both ascending and descending options can be selected by the user.
	- The link to the game should have link-only link purpose applied for accessibility
	
	The games are shown in an infinitely scrolling list. With an infinitely scrolling list, there are no pages, but scrolling the page triggers loading of more games if the scroll position gets close to the end of the currently loaded games. In infinite scroll mode, there are no pages, but scrolling triggers loading of more games.

Detail Game	- This page renders the game shows the current highscores (which are automatically updated). - If the user is within the top ten, the current user is highlighted and marked as being the current user (for example by showing it bold). If the current user has a score that ranks below the top ten, then the score is appended, but shown without rank. - The game description is shown as well.

User Profile	This page shows the user profile with:
	- Username
	- Authored Games
	  ○ The list is sorted by when the last version was uploaded. The most recently updated game is at the top.
	  ○ If the user has not uploaded any games, the "Authored Games" section is omitted.
	  ○ Per authored game, the following is shown: Game Title, Description, Thumbnail, The number of submitted scores
	- The author is not shown per game as this is redundant information.
	- Highscores
	  ○ The list shows the highest score per game.
	  ○ The list is sorted by game title alphabetically.
	  ○ Each game has a link that can be clicked to go to the game.

Manage Game	On this page, developers can update title and description of a game, delete the game, or upload a new version. On this page, developers can do the following: - Update title and description. - Upload a new version. This requires the user to upload a ZIP file. - Delete the game. The game is only deleted after the user confirms. If a user tries to reach this page without being the author, they are redirected to the game itself.

ERD



Instructions

- 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_MODULE_SERVER** 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_MODULE_SERVER**
- REST API should be reached on **<host>/XX_MODULE_SERVER/BACKEND** where **XX** is PC number. **No port and no /public suffix path**.
- Frontend app should be reached on **<host>/XX_MODULE_SERVER/FRONTEND**. **No port allowed**
- **Zip** root folder **XX_ MODULE_SERVER** and submit to the submission page. **Make sure to exclude the vendor** and **node_modules** folder when zipping files.