Balai Pengembangan Talenta Indonesia
Pusat Prestasi Nasional
Kementerian Pendidikan, Kebudayaan, Riset, dan Teknologi

MERDEKA BELAJAR

LKS SMK TINGKAT NASIONAL

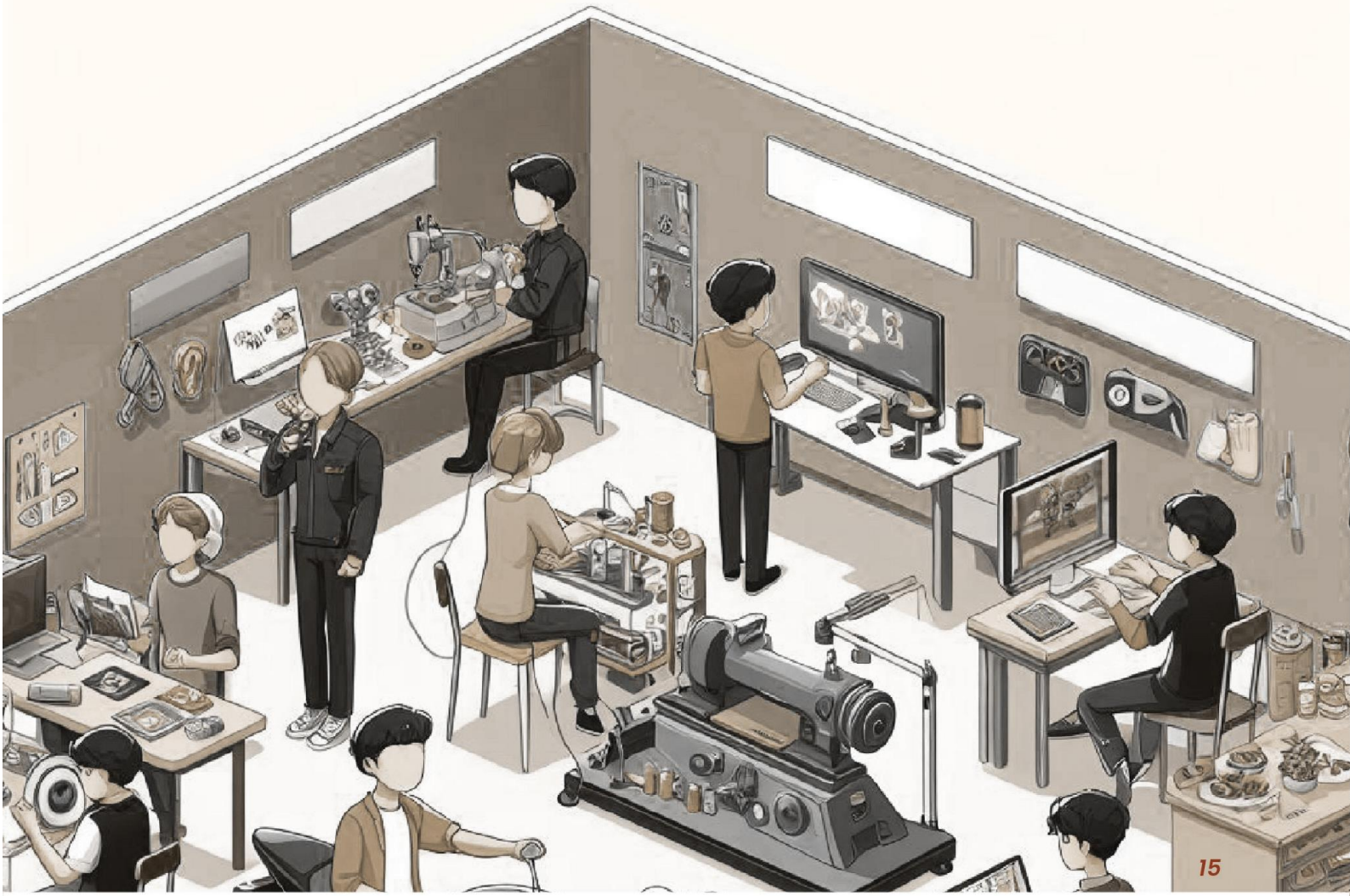
SMK

Kisi-Kisi

Lomba Kompetensi Siswa Nasional 2024

Teknik Desain Laman

(Web Technologies)



MERDEKA BERPRESTASI
Talenta Vokasi Menginspirasi

Contents

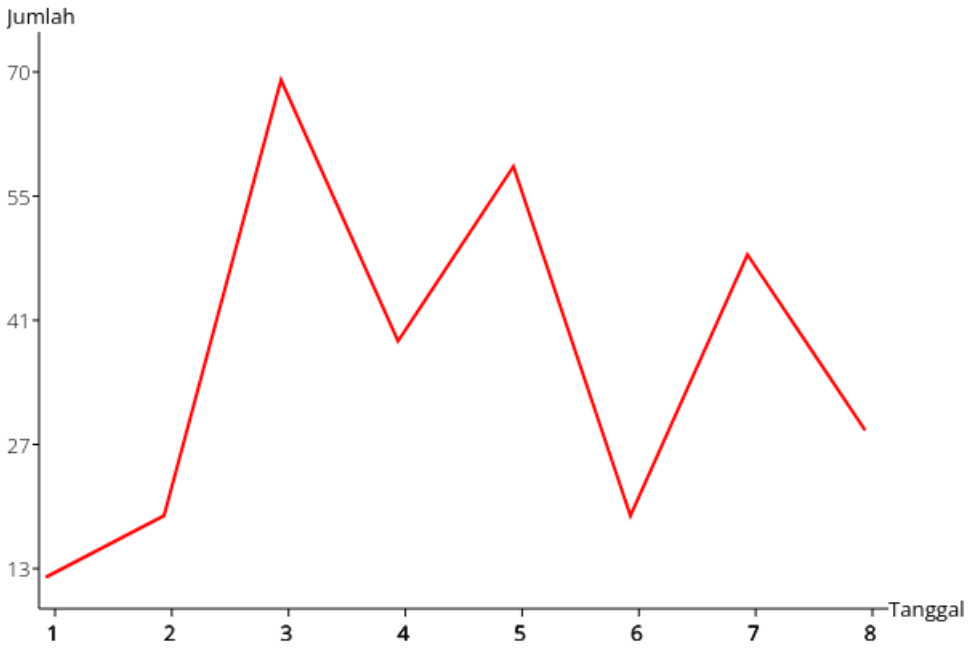
A1. Landing Page

Create a landing page consisting of navigation bar, logo, menu with dropdown, hero section with heading text, and call to action (CTA).

B1. Masonry Layout

In a centered container of 960px, you need to make a layout of 3 columns as shown in the image below.

C1. Line Chart



Make a line chart from the given data provided in media files. The canvas should be 600x400 in size and centered both vertically and horizontally in the browser. The vertical label, horizontal label, and the line should exist within the screen.

C2. Compare Image

Given an image in the media files, you need to draw the normal image and the grayscale version of the image. Users should drag the circle at the center to left and right in order to see the image difference. Please see the given video example provided in media files.

C3. Scope

You need to make a hidden fullscreen-sized text, and a box of 400x400 on top of it. The user can not see the text, but they can see it through the box. The box can be moved by users by dragging it. See example.mp4 for more details.

D1. Internationalization

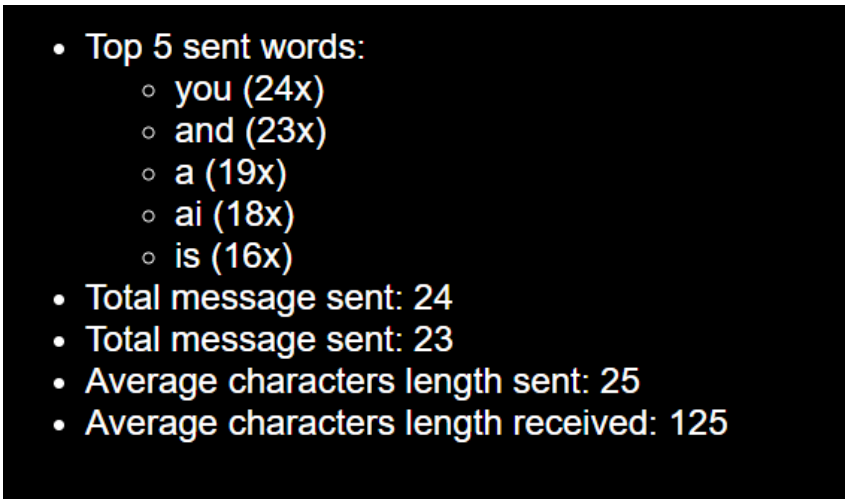
Provided a json file of languages, you need to translate the words that exist in the HTML file using PHP. There is a select form where user can change the language of the website.

D2. Chat Analytics

Provided a JSON file of user messages, you need to make the analytics of the messages sent which is consist:

- Top 5 sent words
- Total messages sent
- Total messages received
- Average character length sent
- Average character length received

Example:



D3. Watermark

Given an image and a logo (png file), the web should output an image with a watermark in the top right.

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 **World Head Football** 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 4 hours. Your first 2 hours must be used to create the initial layout using HTML/CSS. Your layout should follow the design that you created. The final 2 hours you will create the functionality of the game using JavaScript that allows the game to work correctly in different web browsers.

World Head Football game screen should have meet these requirements below:
1. Player Name
2. Gameboard
3. Player Character
4. Country Flag
5. Total Score
6. Timer
7. Match 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. **The design should be delivered in dark mode color.** You are free to choose dark color as long as it has the user convenience.
3. **You are free to decorate** the game screen design as long as it meets the requirements.
4. **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:
    1. one of three levels (easy, medium, hard).
    2. one of two balls to be used.

3. one of many countries as player 1 and one as player 2.
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.
    1. '30 seconds' for easy level
    2. '20 seconds' for medium level
    3. '15 seconds' for hard level
10. **The player 1 position** will be on the left and the player 2 will be on the right
11. **The ball will appear** from top to bottom in the middle of 2 characters.
12. **The ball will bounce** when it hits the body part of the character.
13. **Player 1 move the character** using the following buttons:
    1. "A" to move left
    2. "W" to jump
    3. "D" to move right
    4. "Space" to kick the ball
14. **Player 2 move the character** using the following buttons:
    1. "Left arrow" to move left
    2. "Right arrow" to move right
    3. "Up arrow" to jump
    4. "Enter" to kick the ball
15. **Items will drop** every 5 seconds.
16. **The items that will drop** are as follows:
    1. Increase Ball box to increase ball size
    2. Decrease Ball box to decrease ball size
    3. Diamond ice to freeze ball in 3 seconds
17. **Item will disappears** after being hit by the ball
18. **Walking animations are showing** as the characters are moving.
19. **The score will be increased** if the player can score a goal against the opponent.
20. **After a goal,** the ball will disappear and reappear from top to bottom in the middle of 2 characters.
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. **If the time is over but the score is still the same,** then one of the characters must score again, then the game is over.
26. **Show popup after game over** to display the player username, the countries, scores, save match history button, and restart button.
27. **Match results should be saved in the local storage** after the player clicks the "**Save Score"** button.
28. **Players are able to** see the match history after clicking the Match History Button.
29. **Match history can be sorted** by score or last matches based on user choices.
30. **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.


PLAYER NAME
MATCH HISTORY
Brazil
2 0
TIMER
30
England
WORLD HEAD FOOTBALL

SERVER SIDE MODULE

CONTENTS

This module has the following files:

1. MODULE_SERVER_SIDE.docx
2. MODULE_SERVER_SIDE_MEDIA.zip

INTRODUCTION

In this module, you are asked to create a social media website called “Facegram” where in the website a user can see other user's posts by following the user. Users can also register as a private account so other users who are not following could not see posts. The detailed description and tools that you can use will be described below.

DESCRIPTION OF PROJECT AND TASKS

This module is divided into 2 phases. In the first phase, you will create a REST Api then in the second phase you will create a frontend application with the following provided frameworks:

- Laravel (v10.x)
- React (v18.x with react-router and axios)
- Vue (v3.x with vue-router and axios)

Phase 1 - REST API

For all endpoints, the request header as default sets Content-type: application/json, but some endpoints have specified Content-type header so you have to adjust it.

1. Authentication

You can use **sanctum** for the token management and it must be valid when placed in the Authorization request header as a **Bearer token**.

a. Register

Endpoint	/api/v1/auth/register
Method	POST
Description	For user to register account
Request	Body: full_name (required) bio (required, max 100 chars) username (required, min 3 chars, unique, only alphanumeric, dot “.” or underscore “_” allowed) password (required, min 6 chars) is_private (boolean)

Response Success	HTTP Status Code: 201
	Body:
	{
	"message": "Register success",
	"token": "7|xuPEGo7jtV8IhoE2Mp3am1rwtIpYUTewXBTcH78Af922f116",
	"user": {
	"full_name": "Budi Budiman",
	"bio": "stop dreaming, start doing",
	"username": "budi.budiman",
	"is_private": true,

	<pre>"id": 101
	}
	}</pre>

Response Invalid Field	HTTP Status Code: 422
	Body:
	<pre>{
	"message": "Invalid field",
	"errors": {
	"full_name": [
	"The full name field is required."
	],
	"username": [
	"The username has already been taken."
	],
	"password": [
	"The password field must be at least 6 characters."
	],
	"bio": [
	"The bio field is required."
	]
	}
	}</pre>

b. Login

Endpoint	/api/v1/auth/login
Method	POST
Description	For user to log into system
Request	Body: username password

Response Success	HTTP Status Code: 200
	Body:
	<pre>{
	"message": "Login success",
	"token":
	"2|Qnt2aqHIi0EVwCz3rSH0yiFIKMqey38SdkUZntRvf0e507ff",
	"user": {
	"id": 1,
	"full_name": "John Doe",
	"username": "john.doe",
	"bio": "The best way to predict the future is to create
	it.",
	"is_private": 0,
	"created_at": "2023-10-14T15:04:33.000000Z"
	}
	}</pre>

Response Failed	HTTP Status Code: 401
	Body:
	<pre>{
	"message": "Wrong username or password"
	}</pre>

c. Logout

Endpoint	/api/v1/auth/logout
Method	POST
Description	For user to logout
Request	Headers: Authorization: Bearer <token> Body: username password

Response Success	HTTP Status Code: 200
	Body:
	{
	"message": "Logout success"
	}

Response Invalid Token	HTTP Status Code: 401
	Body:
	{
	"message": "Unauthenticated."
	}

2. Post

a. Create new post

Endpoint	/api/v1/posts
Method	POST
Description	For user to create a post
Request	Headers: Authorization: Bearer <token> Content-type: multipart/form-data Body: caption (required) attachments (required, array of image files jpg, jpeg, webp, png or gif)

Response Success	HTTP Status Code: 201
	Body:
	{
	"message": "Create post success"
	}

Response Invalid Field	Body:
	{
	"message": "Invalid field",
	"errors": {
	"caption": [
	"The caption field is required."
	],
	"attachments.0": [
	"The attachments.0 field must be a file of type: png, jpg, jpeg, webp."
	]

	} }

Response
Invalid Token
	HTTP Status Code: 401
	Body:
	{
	"message": "Unauthenticated."
	}

b. Delete post

Endpoint	/api/v1/posts/:id
Method	DELETE
Description	For user to delete a post
Request	Headers: Authorization: Bearer <token>
Response Success	HTTP Status Code: 204

Response Post Not Found	HTTP Status Code: 404
	Body:
	{
	"message": "Post not found"
	}

Response Try to delete another user's post	HTTP Status Code: 403
	Body:
	{
	"message": "Forbidden access"
	}

Response
Invalid Token
	HTTP Status Code: 401
	Body:
	{
	"message": "Unauthenticated."
	}

c. Get posts

Endpoint	/api/v1/posts
Method	GET
Description	For user to get all posts
Request	Headers: Authorization: Bearer <token> Params: page (integer, minimum 0 and default 0) size (integer, minimum 1 and default 10)
Response Success	HTTP Status Code: 200 Body: { "page": 0,

	"size": 10,
	"posts": [
	{
	"id": 366,
	"caption": "Letting my soul wander",
	"created_at": "2023-07-26 00:34:10",
	"deleted_at": null,
	"user": {
	"id": 61,
	"full_name": "Irving Greenfelder",
	"username": "talia94",
	"bio": "Success is not final, failure is not fatal: It is the courage to continue that counts.",
	"is_private": 0,
	"created_at": "2023-10-18 19:16:53"
	},
	"attachments": [
	{
	"id": 750,
	"storage_path": "posts/652fc6284eb76.jpg"
	}
	]
	},
	….
	]
	}

Response Invalid Params	HTTP Status Code: 422
	Body:
	{
	"message": "Invalid field",
	"errors": {
	"page": [
	"The page field must be at least 0."
	],
	"size": [
	"The size field must be a number."
	]
	}
	}

Response Try to delete another user's post	HTTP Status Code: 403
	Body:
	{
	"message": "Forbidden access"
	}

Response Invalid Token	HTTP Status Code: 401
	Body:
	{
	"message": "Unauthenticated."
	}

3. Following

a. Follow a user

Endpoint	/api/v1/users/:username/follow
Method	POST
Description	For user to follow another user

Request	Headers: Authorization: Bearer <token>

Response
Success
	HTTP Status Code: 200
	Body:
	{
	"message": "Follow success",
	"status": "following" | "requested"
	}

Response User not found	HTTP Status Code: 404
	Body:
	{
	"message": "User not found"
	}

Response Try to follow own user account	HTTP Status Code: 422
	Body:
	{
	"message": "You are not allowed to follow yourself"
	}

Response Already Followed	HTTP Status Code: 422
	Body:
	{
	"message": "You are already followed",
	"status": "following" | "requested"
	}

Response
Invalid Token
	HTTP Status Code: 401
	Body:
	{
	"message": "Unauthenticated."
	}

b. Unfollow a user

Endpoint	/api/v1/users/:username/unfollow
Method	DELETE
Description	For user to unfollow a followed user
Request	Headers: Authorization: Bearer <token>
Response Success	HTTP Status Code: 204

Response User not found	HTTP Status Code: 404
	Body:
	{
	"message": "User not found"
	}

Response Not Following the User	HTTP Status Code: 422
	Body:
	{
	"message": "You are not following the user"
	}

Response Invalid Token	HTTP Status Code: 401
	Body:
	{
	"message": "Unauthenticated."
	}

c. Get following users

Endpoint	/api/v1/following
Method	GET
Description	For users to get their following users
Request	Headers: Authorization: Bearer <token>

Response Success	HTTP Status Code: 200
	Body:
	{
	"following": [
	{
	"id": 91,
	"full_name": "Miss Ayana Russel",
	"username": "chartmann",
	"bio": "In a world where you can be anything, be kind.",
	"is_private": 0,
	"created_at": "2023-10-18 19:16:55",
	"is_requested": true | false
	},
	….
	]
	}

Response User not found	HTTP Status Code: 404
	Body:
	{
	"message": "User not found"
	}

Response Invalid Token	HTTP Status Code: 401
	Body:
	{
	"message": "Unauthenticated."
	}

4. Followers

a. Accept follow request

Endpoint	/api/v1/users/:username/accept
Method	PUT
Description	For users to get their following users
Request	Headers: Authorization: Bearer <token>

Response Success	HTTP Status Code: 200
	Body:
	{
	"message": "Follow request accepted"
	}

Response User not found	HTTP Status Code: 404
	Body:
	{
	"message": "User not found"
	}

Response Already Accepted	HTTP Status Code: 422
	Body:
	{
	"message": "Follow request is already accepted"
	}

Response User Not Following	HTTP Status Code: 422
	Body:
	{
	"message": "The user is not following you"
	}

Response Invalid Token	HTTP Status Code: 401
	Body:
	{
	"message": "Unauthenticated."
	}

b. Get follower users

Endpoint	/api/v1/users/:username/followers
Method	GET
Description	For users to get their following users
Request	Headers: Authorization: Bearer <token>

Response Success	HTTP Status Code: 200
	Body:
	{
	"followers": [
	{
	"id": 91,
	"full_name": "Miss Ayana Russel",
	"username": "chartmann",
	"bio": "In a world where you can be anything, be kind.",
	"is_private": 0,
	"created_at": "2023-10-18 19:16:55",
	"is_requested": true | false
	},
	….
	]
	}

Response User not found	HTTP Status Code: 404
	Body:
	{
	"message": "User not found"
	}

Response Invalid Token	HTTP Status Code: 401
	Body:
	{
	"message": "Unauthenticated."
	}

5. User

a. Get users

Endpoint	/api/v1/users
Method	GET
Description	Get all users that are not followed by logged in users. Logged in users should be hidden in the list.
Request	Headers: Authorization: Bearer <token>

Response Success	HTTP Status Code: 200
	Body:
	{
	"users": [
	{
	"id": 1,
	"full_name": "John Doe",
	"username": "john.doe",
	"bio": "The best way to predict the future is to create it.",
	"is_private": 0,
	"created_at": "2023-10-14T15:04:33.000000Z",
	"updated_at": "2023-10-14T15:04:33.000000Z"
	},
	….
	]
	}

Response Invalid Token	HTTP Status Code: 401
	Body:
	{
	"message": "Unauthenticated."
	}

b. Get detail user

Endpoint	/api/v1/users/:username
Method	GET
Description	For users to get detailed user data. Field posts should be hidden if the user is a private user and the follow status is not following or is requested.
Request	Headers: Authorization: Bearer <token>

Response Success	HTTP Status Code: 200
	Body:
	{
	"id": 2,
	"full_name": "Richard Roe",
	"username": "richard.roe",
	"bio": "Leave a little sparkle wherever you go.",
	"is_private": 1,
	"created_at": "2023-10-18 19:16:50",
	"is_your_account": false,
	"following_status": "not-following" | "requested" | "following",
	"posts_count": 6,
	"followers_count": 20,
	"following_count": 18,
	"posts": [
	{
	"id": 7,
	"caption": "Blissfully lost in the beauty of the moment.",
	"created_at": "2014-08-26 10:42:31",
	"deleted_at": null,
	"attachments": [
	{
	"id": 13,
	"storage_path": "posts/shutterstock_1464930743-scaled.webp"
	},
	….
	]
	},
	….
	]
	}

Response Invalid Token	HTTP Status Code: 401
	Body:
	{
	"message": "Unauthenticated."
	}

Response User not found	HTTP Status Code: 404
	Body:
	{
	"message": "User not found"
	}

Response Invalid Token	HTTP Status Code: 401
	Body:
	{
	"message": "Unauthenticated."
	}

Phase 2 - Frontend Development

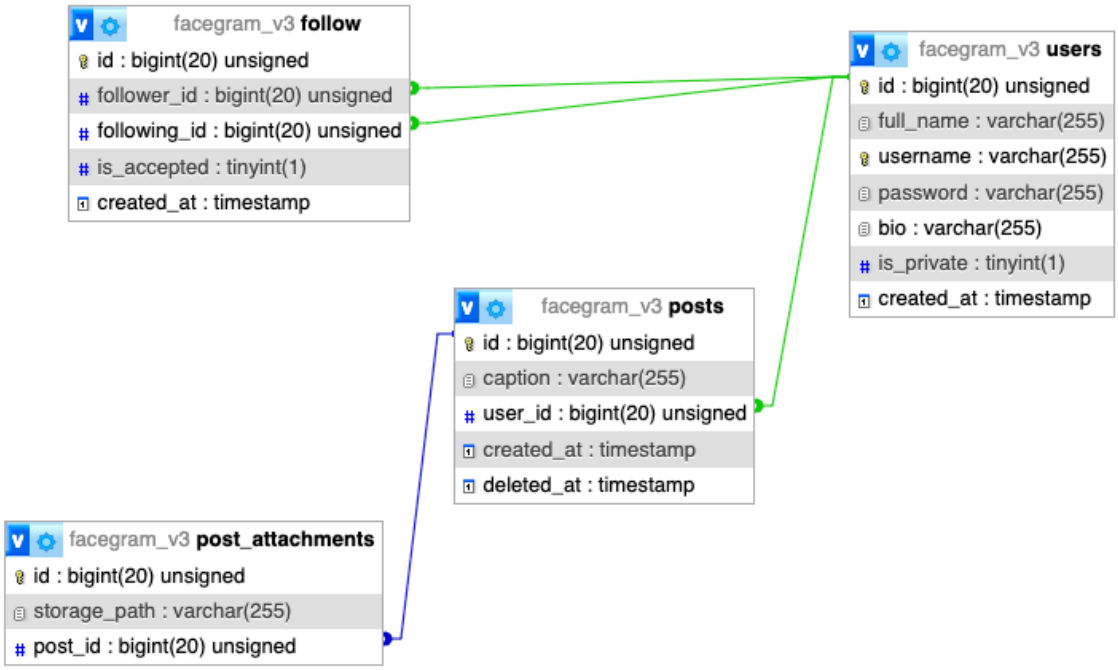
Template HTML is provided. You are required to use the template and you are recommended to modify the template design without affecting the functionality of the frontend.

Page	Description Tasks

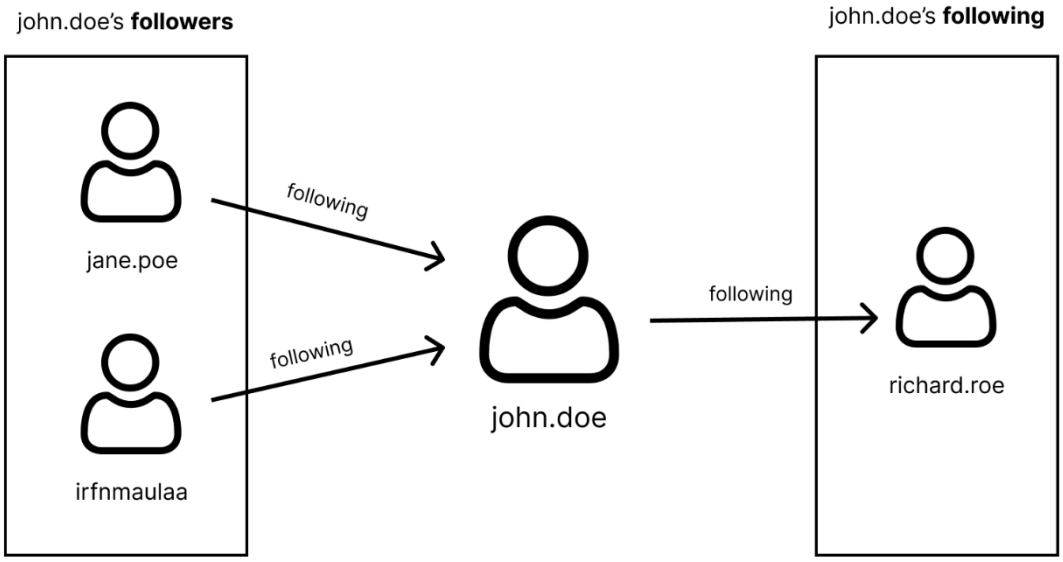
General	• Document title should be reflected to the current page • Display username and logout button on the navbar after login success • Clicking username will go to logged in user profile page • User can log out after clicking logout button on the navbar
Register	• If user logged in, the page will be redirected to own user profile page • User can register account • If register fails, display all of errors message • If logged in user try to access this page, the page should be redirected to own user profile page
Login	• If user logged in, the page will be redirected to own user profile page • User can login with existing credentials • If login fails, display all of errors message • If logged in user try to access this page, the page should be redirected to own user profile page
Homepage	• If user not logged in, the page will be redirected to login page News Feed section • Display following user's posts and own posts sorted by newest post on the newsfeed section • When page loads, display only first 10 newest posts • If the vertical scroll is stuck at the bottom, load the next 7 posts Explore people section • Display all users that are not followed by the logged in user • Clicking on username will go to that user profile page Follow requests section (for private account) • Display follow request users • The logged in user can accept/approve a follow request • If the logged in user is not a private account or there is no any follow request, hide this section

User profile	Page requirements: • If user not logged in, the page will be redirected to login page User information: • See profile (full_name, username, bio) • See posts list and count • See followers list and count • See following list and count Follow/unfollow: • User can follow another user • User can unfollow a followed user Posts section: • Display user's posts where the visited user is not a private account or following account • Users can delete their posts • Display "The account is private" if logged in user is not following the visited private user profile page
Create new post	• If user not logged in, the page will be redirected to login page • User can create new post with following fields: ○ caption (text, required) ○ attachments (file, can attach multiple files) • If create new post fails, display all of errors message

ER DIAGRAM



ADDITIONAL EXPLANATION



MEDIA FILES

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

BALAI PENGEMBANGAN TALENTA INDONESIA
PUSAT PRESTASI NASIONAL
KEMENTERIAN PENDIDIKAN, KEBUDAYAAN, RISET, DAN TEKNOLOGI

Jalan Gardu Rt. 10 Rw. 02, Srengseng Sawah, Kec. Jagakarsa, Kota Jakarta Selatan,
Daerah Khusus Ibukota Jakarta 12640