﻿API Documentation

**Base URL**

[https://rajiv.dikelasawan.my.id:8443/api](https://tinno.dikelasawan.my.id:8443/api/v1)

1. **Endpoint: User Registration**

* **URL**: /registration
* **Method**: POST
* **Description**: Registers a new user by validating the input data, saving the user's profile picture, hashing the password, and attaching a default role to the user. If the CAPTCHA is invalid, the registration will fail.

**Request Headers**

* Content-Type: application/json

**Request Body**

The request body should be a JSON object with the following fields:

{

"name": "string",

"email": "string",

"password": "string",

"profile\_picture": "file",

"captcha ": "string",

"token ": "string", (Captcha token, get from request captcha token endpoint)

}

1. **Endpoint: User Login**

* **URL**: /login
* **Method**: POST
* **Description**: Authenticates a user with their email and password. If the credentials are valid, a token is generated and returned to the user for authenticated sessions.

**Request Headers**

* Content-Type: application/json

**Request Body**

{

"username": "string",

"password": "string"

}

**Endpoint: Update Profile**

* **URL**: /update/user
* **Method**: POST
* **Description**: Updates the authenticated user's profile, including uploading a new profile picture. This endpoint requires authentication using Sanctum.

**Request Headers**

* Authorization: Bearer {token} (Required)  
  The token obtained during login should be sent in the Authorization header.
* Content-Type: multipart/form-data (Required for file uploads)

**Request Body**

The request body should be sent as multipart/form-data and may include the following fields:

* profile\_picture (Optional): A file input containing the new profile picture to be uploaded.

**12. Endpoint: Get Banners**

* **URL**: /banner
* **Method**: GET
* **Description**: Retrieves a list of all banners.

**Request Headers**

* No special headers are required.

**Request Body**

* This endpoint does not require a request body.

**13. Endpoint: Create Banner**

* **URL**: /banner
* **Method**: POST
* **Description**: Creates a new banner. This endpoint is restricted to admin users only.

**Request Headers**

* Authorization: Bearer {token} (Required)  
  The token obtained during login should be sent in the Authorization header.
* Content-Type: multipart/form-data (Required for file uploads)

**Request Body**

The request body should be multipart/form-data containing the following fields:

[

            'title' => ['required', 'string'],

            'description' => ['required'],

            'status' => ['required'],

            'image' => ['required', 'image', 'mimes:png,jpg,jpeg'],

];

**14. Endpoint: Update Banner**

* **URL**: /banner/{id}
* **Method**: post
* **Description**: Updates an existing banner identified by the banner ID. This endpoint is restricted to admin users only.

**Request Headers**

* Authorization: Bearer {token} (Required)  
  The token obtained during login should be sent in the Authorization header.
* Content-Type: application/json (Required for JSON data)

**URL Parameters**

* id (Required): The ID of the banner to be updated.

**Request Body**

The request body should be a JSON object containing the fields to be updated. For example:

[

            'title' ,

            'description' ,

            'status' ,

            'image' ,

];

**15. Endpoint: Delete Banner**

* **URL**: /banner/{id}
* **Method**: DELETE
* **Description**: Deletes an existing banner identified by the banner ID. This endpoint is restricted to admin users only.

**Request Headers**

* Authorization: Bearer {token} (Required)  
  The token obtained during login should be sent in the Authorization header.

**URL Parameters**

* id (Required): The ID of the banner to be deleted.

**16. Endpoint: Get Blogs**

* **URL**: /blog
* **Method**: GET
* **Description**: Retrieves a list of all blogs. This endpoint is public and does not require authentication.

**Request Headers**

* None required.

**17. Endpoint: Create Blog**

* **URL**: /blog
* **Method**: POST
* **Description**: Creates a new blog entry. This endpoint is restricted to admin users only.

**Request Headers**

* Authorization: Bearer {token} (Required)  
  The token obtained during login should be sent in the Authorization header.
* Content-Type: application/json (Required for JSON data)

**Request Body**

The request body should be a JSON object containing the details of the blog to be created. For example:

[

            'image' => ['required', 'image', 'mimes:png,jpg,jpeg'],

            'title' => ['required'],

            'description' => ['required'],

            'tags' => ['required', 'array' ],

];

**18. Endpoint: Update Blog**

* **URL**: /blogs/{id}
* **Method**: post
* **Description**: Updates an existing blog entry. This endpoint is restricted to admin users only.

**Request Headers**

* Authorization: Bearer {token} (Required)  
  The token obtained during login should be sent in the Authorization header.
* Content-Type: application/json (Required for JSON data)

**Request Parameters**

* **Path Parameter**:
  + id (integer): The unique identifier of the blog to be updated.

**Request Body**

The request body should be a JSON object containing the updated details of the blog. For example:

[

            'image',

            'title' ,

            'description' ,

            'tags' ,

];

**19. Endpoint: Delete Blog**

* **URL**: /blog/{id}
* **Method**: DELETE
* **Description**: Deletes an existing blog entry. This endpoint is restricted to admin users only.

**Request Headers**

* Authorization: Bearer {token} (Required)  
  The token obtained during login should be sent in the Authorization header.

**Request Parameters**

* **Path Parameter**:
  + id (integer): The unique identifier of the blog to be deleted.

**20. Endpoint: Get Blog Details**

* **URL**: /blogs/{id}
* **Method**: GET
* **Description**: Retrieves the details of a specific blog entry based on the provided ID.

**Request Parameters**

* **Path Parameter**:
  + id (integer): The unique identifier of the blog whose details are to be retrieved.

21. **Endpoint: Get Portfolios**

* **URL**: /portfolios
* **Method**: GET
* **Description**: Retrieves a list of all portfolios.

**22. Endpoint: Create Portfolio**

* **URL**: /portfolios
* **Method**: POST
* **Description**: Creates a new portfolio entry. This endpoint is restricted to admin users only.

**Request Headers**

* Authorization: Bearer {token} (Required)  
  The token obtained during login should be sent in the Authorization header.

**Request Body**

[

            'title' => ['required'],

            'description' => ['required'],

            'image' => ['image', 'required', 'mimes:png,jpg,jpeg']

];

**23. Endpoint: Update Portfolio**

* **URL**: /portfolio/{id}
* **Method**: post
* **Description**: Updates an existing portfolio entry. This endpoint is restricted to admin users only.

**Request Headers**

* Authorization: Bearer {token} (Required)  
  The token obtained during login should be sent in the Authorization header.

**Request Parameters**

* **Path Parameter**:
  + id (integer): The unique identifier of the portfolio to be updated.

**Request Body**

**Optional:**

[

            'title' ,

            'description' ,

            'image' ,

];

**24. Endpoint: Delete Portfolio**

* **URL: /portfolios/{id}**
* **Method: DELETE**
* **Description: Deletes an existing portfolio entry. This endpoint is restricted to admin users only.**

**Request Headers**

* **Authorization: Bearer {token} (Required)  
  The token obtained during login should be sent in the Authorization header.**

**Request Parameters**

* **Path Parameter:**
  + **id (integer): The unique identifier of the portfolio to be deleted.**