Building a Secure RESTful API

Objective: Develop a fully functional and secure "Posts Management" API using PHP.

1. Requirements:

- **Resource:** `Post` (Fields: `id`, `title`, `content`, `user_id`, `created_at`, `updated_at`).
- **Framework:** None / Laravel (Latest version).
- **Authentication:** JWT / (For Laravel use `tymon/jwt-auth` or Laravel Sanctum).

2. The Routing Architecture (REST Principles):

You must implement the following routes:

Method	URI	Action	Description
GET	/api/posts	index	Fetch all posts (Public)
GET	/api/posts/{id}	show	Fetch a single post (Public)
POST	/api/posts	store	Create a new post (**Protected**)
PUT	/api/posts/{id}	update	Update a post (**Protected**)
DELETE	/api/posts/{id}	destroy	Delete a post (**Protected**)

3. Middleware Protection: Apply the `auth` middleware to ensure only logged-in users with a valid JWT can create / modify their own data.
4. Proper Error Handling:

- Return **404 Not Found** if the post ID doesn't exist.
- Return **422 Unprocessable Entity** for validation errors (e.g., missing title).
- Return **401 Unauthorized** if the JWT is missing or expired.
- Return **403 Forbidden** if the Auth user has no access to the resource.

5. JSON Consistency: Ensure every response follows a consistent structure:

```
{
  "success": true,
  "data": { ... },
  "message": "Post created successfully"
}
```

6. Standardized JSON Error Response Example (Status code 422)

```
{
    "success": false,
    "message": "Validation Failed",
    "errors": {
        "title": [
            "The title field is required.",
            "The title must be at least 5 characters."
        ],
        "content": [
            "The content field is required."
        ]
    }
}
```