Django REST Framework

# Overview

An API - Application Programming Interface - is a set of rules allowing different software applications to communicate with one another. The Django REST Framework (DRF) is a toolkit for building web APIs on top of the Django framework. It provides tools for interacting with Django models, CRUD operations, and other elements. Using DRF, we will create an API that will act as the communication link between the backend (Django) and the front-end (React), enabling secure data exchange. It will also directly handle other processes, such as user authentication. Since our web API will only act as the connection between the technologies mentioned above, we intend for our API to receive calls from only one source, the React server, where users of our web application will interact with it indirectly.

# Authentication

## Same Origin Policy

Our API will be limited to handling requests originating from the React server domain/IP address. Therefore, we authenticate API requests as they come from a trusted source, helping to prevent unauthorised third-party websites from accessing our Django server. We will adhere to this policy by using CORS (Cross-Origin Resource Sharing), enabling our Django server to specify that only the React server can access its resources.

## Secure Communication

In production, we will acquire an SSL/TLS certificate. This certificate will enable HTTPS, which allows for secure communication during the transmission of content sent to and from the server, helping to secure our API from nefarious actions.

# Rate Limits

We will enforce rate limits when accessing the Django server to ensure our API's optimal performance and reliability. By controlling how often a user can make a web request, we will prevent potential abuse and help manage server loads, stopping unexpected system outages and slower response times.

We will have an initial limit of five requests per minute per user. In events of exceedance, the server will return an error code to indicate the limit being exceeded.

# Status Codes

HTTP response codes will indicate the status of a request made to our API:

* **HTTP 200 OK** - For successful requests.
* **HTTP 400 Bad Request** - This is used for unsuccessful, malformed requests that the server cannot or will not process. Where needed, a message outlining what went wrong will accompany the request.
* **HTTP 403 Forbidden** - This is for situations where the user is not authenticated or doesn’t have the relevant permissions to make such a request.
* **HTTP 429 Too Many Requests** - For rate limit violations.

# Endpoints

Our API will be accessed through predefined endpoints - a specific URL that correlates to an action to be carried out by an API. Defined API endpoints that require a user to be authenticated will require the cookie created during authentication to be sent alongside the request. All endpoints belonging to our API will accept the ‘OPTIONS’ method to enable pre-flight checks, which are checks part of CORS that help validate requests (by checking if the endpoint supports the type of request).

## 

## User Management Endpoints

### /api/auth/login

* **Description:** This endpoint authenticates users. It verifies user credentials and logs them in if they are valid.
* **Supported Method Types:**
  + ‘POST’: is solely responsible for the authentication of a user with the provided credentials.
* **Permissions:** This endpoint is open to all users, regardless of their authentication status or permissions.
* **Resources:** Whilst authenticating users, the Django authentication system will access the set authentication user model, which will be our ‘Users’ table in our database. No resources are exposed, only accessed.

#### Example Request

| POST /api/auth/login Content-Type: application/json |
| --- |

…

| {  "email": "example@email.com",  "password": "password" } |
| --- |

#### Responses

The possible responses are:

* An **‘HTTP 429 Too Many Requests’**.
* An **‘HTTP 200 OK’**.
* An **‘HTTP 400 Bad Request’** accompanied by one of the following messages:

| [  "Email field cannot be empty." // Email field was not provided ]  [  "Password field cannot be empty." // Password field was not provided ]  [  "Incorrect username or password." // Incorrect credentials or user doesn't exist. ] |
| --- |

### /api/auth/logout

…