# Documentation

## Documentation

Welcome to the Twitter Backend Project!

## AUTHENTICATION

### Create User - POST /auth/register/

```
{
    "username": "test",
    "password": "1234"
}
```

> - Handles POST requests to create a new user registration.
> - Validates and saves user information.
> - Returns the data of the newly created user.

## Login - POST /auth/login/

```
{
    "username": "test",
    "password": "1234"
}

"""
TOKEN
{
    "refresh": "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJ0b2tlbl9
    "access": "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJ0b2tlbl9
}
"""
```

- Handles POST requests for user login.
- Retrieves the username and password, finds the user in the database.
- Raises authentication error if the user does not exist or if the password is incorrect.
- Provides a JWT (JSON Web Token) to the user upon successful login.
- Embeds this JWT into an HTTP cookie and sends it as a response.
- The response containing the JWT includes both access and refresh tokens.

## Profile - GET  /auth/profile/

```
    {
        "id": 4,
        "username": "test",
        "bio": ""
    }
```

- Only allows authenticated users.
- Handles GET requests and returns the profile of the requesting user.

## Log Out- POST /auth/logout/

```
{
    "message": "Log out success"
}
```

- Only allows authenticated users.
- Handles POST requests to log the user out.
- Deletes the JWT cookie and returns a logout success message.

## USERS

### All Users- GET  /users/users/

```
[
    {
        "id": 1,
        "username": "urfat",
        "bio": ""
    },
    {
        "id": 2,
        "username": "fatih",
        "bio": ""
    },
    {
        "id": 3,
        "username": "yunus",
        "bio": "cyber kid"
    },
    {
        "id": 4,
        "username": "test",
        "bio": "npc :)"
    }
]
```

- Extends `ModelViewSet` to handle CRUD operations for user profiles.
- Specifies the queryset to include all user profiles.
- Uses `UserSerializer` for serialization and deserialization of user data.
- Requires authentication for all operations (`IsAuthenticated` permission).
- Uses `SearchFilter` for filtering user profiles based on the username field.

## Users Detail- GET /users/users/4

```
    {
        "id": 4,
        "username": "test",
        "bio": "npc :)"
    }
```

- Provides detailed views for individual user profiles.
- Requires authentication for all operations (`IsAuthenticated` permission).
- Handles GET requests to retrieve a specific user profile by its primary key (`pk`).

- Handles PUT requests to update a specific user profile by its primary key (`pk`).
- Handles DELETE requests to delete a specific user profile by its primary key (`pk`).
- Returns appropriate error responses if the requested user profile does not exist.

### Query - GET /users/users/?search=test

```
{
    "id": 4,
    "username": "test",
    "bio": "npc :)"
}
```

## TWEETS

### All Tweets - GET /api/tweets

```
[
{
    "id": 1,
    "content": "bohemian rhapsody",
    "created_at": "2024-06-11T17:56:44.418290Z",
    "author": 1,
    "hashtags": []
},
{
    "id": 2,
    "content": "billie jean",
    "created_at": "2024-06-11T17:57:03.793156Z",
        "author": 1,
        "hashtags": [
            1
        ]
    },
    {
        "id": 3,
        "content": "star wars",
        "created_at": "2024-06-11T17:57:12.643521Z",
        "author": 1,
        "hashtags": [
            2
        ]
    },
    {
        "id": 4,
        "content": "espresso",
        "created_at": "2024-06-11T17:57:26.395110Z",
        "author": 1,
        "hashtags": [
            1
        ]
    }
]
```

- Extends `ModelViewSet` to handle CRUD operations for tweets.
- Specifies the queryset to include all tweets.
- Uses `TweetSerializer` for serialization and deserialization of tweet data.
- Requires authentication for all operations (`IsAuthenticated` permission).
- Utilizes `SearchFilter` for filtering tweets based on the content field.
- Overrides the `perform_create` method to automatically set the author of the tweet to the current authenticated user.

- Implements custom filtering to allow filtering tweets by hashtag name, if provided in the request query parameters.

## Tweet Details - GET  /api/tweets/3

```
{
    "id": 3,
    "content": "star wars",
    "created_at": "2024-06-11T17:57:12.643521Z",
    "author": 1,
    "hashtags": [
        2
    ]
}
```

## Create Tweet - POST /api/tweets/

```
{
    "content": "hello",
    "hashtags": [
        "1"
    ]
}

"""
{
    "id": 5,
    "content": "hello",
    "created_at": "2024-06-12T08:27:31.086971Z",
    "author": 4,
    "hashtags": [
        1
    ]
}
"""
```

## Create Hashtag- POST /api/hashtags/

```
{
    "name": "language"
}
```

## All Hashtags - GET /api/hashtags/

```
[
    {
        "id": 1,
        "name": "music"
    },
    {
        "id": 2,
        "name": "movie"
    },
    {
            "id": 3,
            "name": "language"
        }
]
```

- Extends `ModelViewSet` to handle CRUD operations for hashtags.
- Specifies the queryset to include all hashtags.
- Uses `HashtagSerializer` for serialization and deserialization of hashtag data.
- Requires authentication for all operations (`IsAuthenticated` permission).

## Endpoint to list tweets with specific hashtags - GET /api/tweets/?hashtag=movie

```
[
    {
        "id": 3,
        "content": "star wars",
        "created_at": "2024-06-11T17:57:12.643521Z",
        "author": 1,
        "hashtags": [
            2 #movie
        ]
    },
     {
        "id": 5,
        "content": "Lupen",
        "created_at": "2024-06-11T17:57:12.643521Z",
        "author": 2,
        "hashtags": [
            2 #movie hashtag
        ]
    }
]
```

- Specifies the queryset to include all hashtags.
- Requires authentication for all operations (`IsAuthenticated` permission).

> 💡 There may be a design problem. tweets are under the api path(api/tweets), users are under the users path(users/users)😇

😃 **Thank you!**

Any questions or thoughts? You can contact me at info**@yunusolcar**.com