Endpoint: Register

**Description:** This endpoint allows a user to register a new account. Upon successful registration, the user is automatically logged in and an authentication token is provided.

**URL:**  
**POST /api/register/**

**Request:**

* Method: **POST**
* Headers:
  + Content-Type: application/json
* Body: JSON object with the following fields:
  + **username** (string): The desired username for the new account.
  + **password** (string): The password for the new account.
  + **email** (string): The email address of the user.

**Response:**

* Status Code: 201 Created
* Response Body: JSON object with the following fields:
  + **token** (string): An authentication token for the newly registered user.
  + **user** (object): User details in JSON format, including the user's ID, username, and email.

**Error Responses:**

* Status Code: 400 Bad Request
  + Response Body: JSON object containing validation errors if the request data is invalid.
* Status Code: 500 Internal Server Error
  + Response Body: JSON object with an error message if an exception occurs during registration.

Request:

POST /register/

{

"username": "john\_doe",

"password": "secure\_password",

"email": "john@example.com"

}

Response (201 Created):

{

"token": "your-auth-token",

"user": {

"id": 123,

"username": "john\_doe",

"email": "john@example.com"

}

}

**Note:**

* Upon successful registration, the user is automatically logged in.
* An authentication token is provided for subsequent authenticated requests.
* In case of an error during registration, the specific error message will be included in the response.

Endpoint: Login

**Description:** This endpoint allows a user to log in using their username and password. Upon successful login, an authentication token is provided for subsequent authenticated requests.

**URL:**  
**POST /api/login/**

**Request:**

* Method: **POST**
* Headers:
  + Content-Type: application/json
* Body: JSON object with the following fields:
  + **username** (string): The username of the user.
  + **password** (string): The user's password.

**Response:**

* Status Code: 200 OK
* Response Body: JSON object with the following fields:
  + **token** (string): An authentication token for the logged-in user.
  + **user** (object): User details in JSON format, including the user's ID, username, and email.

**Error Responses:**

* Status Code: 400 Bad Request
  + Response Body: JSON object with an error message if the provided credentials are invalid.
* Status Code: 500 Internal Server Error
  + Response Body: JSON object with an error message if an exception occurs during login.

Example:

Request:

POST /login/

{

"username": "john\_doe",

"password": "secure\_password"

}

Response (200 OK):

{

"token": "your-auth-token",

"user": {

"id": 123,

"username": "john\_doe",

"email": "john@example.com"

}

}

**Note:**

* Upon successful login, the user receives an authentication token.
* In case of invalid credentials, an error message is returned.
* Errors during login will include a specific error message in the response.

Endpoint: Logout

**Description:** This endpoint allows an authenticated user to log out, which invalidates their authentication token.

**URL:**  
**POST /logout/**

**Request:**

* Method: **POST**
* Headers:
  + Authorization: Token [your-auth-token]

**Response:**

* Status Code: 200 OK
* Response Body: JSON object with the message "Successfully logged out."

**Error Responses:**

* Status Code: 500 Internal Server Error
  + Response Body: JSON object with an error message if an exception occurs during the logout process.

**Note:**

* To perform a successful logout, include the authentication token in the **Authorization** header as "Token [your-auth-token]."
* After logging out, the token is deleted to invalidate it.
* Any errors during logout will include a specific error message in the response.

Endpoint: Upload Images

**Description:** This endpoint allows users to upload one or more image files. It processes the uploaded images, including associating labels and annotations with each image.

**URL:**  
**POST /api/upload-images/**

**Request:**

* Method: **POST**
* Headers:
  + Content-Type: multipart/form-data
* Form Data:
  + **images** (file list): List of image files to be uploaded.
  + **labels** (string, optional): Labels associated with the uploaded images.
  + **annotations** (string, optional): Annotations or descriptions for the uploaded images.

**Response:**

* Status Code: 201 Created
* Response Body: A list of JSON objects, each containing details of the uploaded images. Each object includes fields like:
  + **id**: The ID of the uploaded image.
  + **name**: The name of the image file.
  + **user**: The ID of the user who uploaded the image.
  + **labels**: The labels associated with the image.
  + **annotations**: Annotations or descriptions for the image.
  + **url**: The URL of the uploaded image.

**Error Responses:**

* Status Code: 400 Bad Request
  + Response Body: JSON object with the error message "No images provided" if no images are provided in the request.
* Status Code: 500 Internal Server Error
  + Response Body: JSON object with an error message if an exception occurs during image upload or processing.

**Example:**

* In your React frontend, create a form to allow users to select multiple image files and optionally input labels and annotations.
* Use the **multipart/form-data** content type for the POST request.
* After successful image upload, the response will contain details of the uploaded images, including labels and annotations if provided.
* In case of any issues during image upload or processing, specific error messages will be included in the response.

**Note:**

* This endpoint accepts multiple image files in the **images** field.
* Uploaded images must be of types: **.png**, **.jpg**, **.jpeg**, **.tiff**, **.bmp**, or **.gif**.
* Labels and annotations can be associated with the uploaded images.
* Images are associated with the user who made the upload.
* Uploaded images are accessible via their URL.

Endpoint: Update Image Annotations

**Description:** This endpoint allows an authenticated user to update the annotations or details of a specific image. Users can modify certain fields of an existing image record.

**URL:**  
**PATCH /api/update-annotations/<image\_id>/**

**Request:**

* Method: **PATCH**
* Headers:
  + Authorization: Token [your-auth-token]
* URL Parameters:
  + **image\_id** (integer): The ID of the image to be updated.
* Body: JSON object with the fields to be updated. You can partially update fields like **labels**, **annotations**, and others.

Example:

{

"annotations": "Updated description for this image."

}