PlantGuard API Endpoints Documentation

July 19, 2025

Generated for the PlantGuard Django Project

Contents

1 Introduction

This document provides detailed documentation for the API endpoints of the PlantGuard Django project. The project uses Django REST Framework (DRF) with JWT authentication for user management and a plant disease detection system. All endpoints are prefixed with `/api/`. The `account` app handles user authentication and profile management, while the `detection` app manages plant disease prediction and history.

Base URL: `http://127.0.0.1:8000/api/`

Authentication: Most endpoints (except registration and prediction) require JWT authentication. Obtain a JWT token via `/api/account/login/` and include it in the `Authorization` header as `Bearer <access_token>`.

2 Account App Endpoints

2.1 Register

URL `/api/account/register/`

Method POST

Description Creates a new user account.

Permissions `AllowAny` (no authentication required).

Request Body

```
{
    "username": "string",
    "email": "string (optional)",
    "password": "string",
    "first_name": "string (optional)",
    "last_name": "string (optional)"
}
```

Response

- **Success (201 Created)**:

```
{
    "username": "string",
    "email": "string",
    "first_name": "string",
    "last_name": "string"
}
```

- **Error (400 Bad Request)**:

```
{
    "username": ["A user with that username already
       exists."],
    "email": ["This field must be unique."]
}
```

```
curl -X POST http://127.0.0.1:8000/api/account/register/ \
-H "Content-Type: application/json" \
```

```
-d '{
    "username": "john_doe",
    "email": "john@example.com",
    "password": "securepassword123",
    "first_name": "John",
    "last_name": "Doe"
}'
```

2.2 Login (Token Obtain)

Example URL `/api/account/login/`

Method POST

Description Authenticates a user and returns JWT access and refresh tokens.

Permissions `AllowAny` (no authentication required).

Request Body

```
{
    "username": "string",
    "password": "string"
}
```

Response

- **Success (200 OK)**:

```
{
    "refresh": "string (refresh token)",
    "access": "string (access token)"
}
```

- **Error (401 Unauthorized)**:

```
{
    "detail": "No active account found with the given
       credentials"
}
```

```
curl -X POST http://127.0.0.1:8000/api/account/login/ \
-H "Content-Type: application/json" \
-d '{
    "username": "john_doe",
    "password": "securepassword123"
}'
```

2.3 Token Refresh

Example URL `/api/account/refresh/`

Method POST

Description Refreshes an expired access token using a valid refresh token.

Permissions `AllowAny` (no authentication required).

Request Body

```
{
    "refresh": "string (refresh token)"
}
```

Response

- **Success (200 OK)**:

```
{
    "access": "string (new access token)"
}
```

- **Error (401 Unauthorized)**:

```
{
    "detail": "Token is invalid or expired",
    "code": "token_not_valid"
}
```

```
curl -X POST http://127.0.0.1:8000/api/account/refresh/ \
-H "Content-Type: application/json" \
-d '{
    "refresh": "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9..."
}'
```

2.4 Update User Profile

Example URL `/api/account/profile/`

Method PUT

Description Updates the authenticated user's profile (first name, last name, and/or password).

Permissions `IsAuthenticated` (requires valid JWT token).

Request Body

```
{
    "first_name": "string (optional)",
    "last_name": "string (optional)",
    "old_password": "string (required if updating password)",
    "new_password": "string (required if updating password)"
}
```

Response

- **Success (200 OK)**:

```
{
    "message": "Profile updated successfully."
}
```

- **Error (400 Bad Request)**:

```
{
    "error": "Old password is incorrect."
}
```

```
curl -X PUT http://127.0.0.1:8000/api/account/profile/ \
-H "Authorization: Bearer <access_token>" \
-H "Content-Type: application/json" \
-d '{
    "first_name": "Jane",
    "last_name": "Doe",
    "old_password": "securepassword123",
    "new_password": "newpassword456"
}'
```

3 Detection App Endpoints

3.1 Predict Plant Disease

Example URL `/api/detection/predict/`

Method POST

Description Uploads an image of a plant leaf to predict the disease, confidence score, and suggested remedy. Saves prediction history for authenticated users.

Permissions `AllowAny` (no authentication required, but history saved only for authenticated users).

Request Body (multipart/form-data):

- `image`: Image file (e.g., JPEG, PNG)

Response

- **Success (200 OK)**:

```
{
    "disease": "string (e.g., Tomato___healthy)",
    "confidence": float (e.g., 0.9876),
    "remedy": "string (e.g., No disease detected.)"
}
```

- **Error (400 Bad Request)**:

```
{
    "error": "No leaf or disease found. Please provide a
       leaf image."
}
```

- **Error (500 Internal Server Error)**:

```
{
    "error": "string (exception message)"
}
```

```
curl -X POST http://127.0.0.1:8000/api/detection/predict/ \
-H "Authorization: Bearer <access_token> (optional)" \
-F "image=@/path/to/leaf.jpg"
```

3.2 List Prediction History

Example URL `/api/detection/history/`

Method GET

Description Retrieves the authenticated user's prediction history.

Permissions `IsAuthenticated` (requires valid JWT token).

Request Body None

Response

- **Success (200 OK)**:

```
[
    {
        "id": int,
        "user": int (user ID),
        "image": "string (URL to image)",
        "disease": "string",
        "confidence": float,
        "remedy": "string",
        "timestamp": "string (ISO 8601 format)"
    },
    ...
]
```

- **Error (401 Unauthorized)**:

```
{
    "detail": "Authentication credentials were not
       provided."
}
```

```
curl -X GET http://127.0.0.1:8000/api/detection/history/ \
-H "Authorization: Bearer <access_token>"
```

3.3 Retrieve Prediction History Detail

Example URL `/api/detection/history/<int:id>/`

Method GET

Description Retrieves details of a specific prediction record for the authenticated user.

Permissions `IsAuthenticated` (requires valid JWT token).

Request Body None

Response

- **Success (200 OK)**:

```
{
    "id": int,
    "user": int (user ID),
    "image": "string (URL to image)",
    "disease": "string",
    "confidence": float,
    "remedy": "string",
    "timestamp": "string (ISO 8601 format)"
}
```

- **Error (404 Not Found)**:

```
{
    "detail": "Not found."
}
```

- **Error (401 Unauthorized)**:

```
{
    "detail": "Authentication credentials were not
       provided."
}
```

```
curl -X GET http://127.0.0.1:8000/api/detection/history/1/ \
-H "Authorization: Bearer <access_token>"
```

3.4 Delete Prediction History Record

Example URL `/api/detection/history/<int:id>/delete/`

Method DELETE

Description Deletes a specific prediction record for the authenticated user.

Permissions `IsAuthenticated` (requires valid JWT token).

Request Body None

Response

- **Success (204 No Content)**: No body returned.

- **Error (404 Not Found)**:

```
{
    "detail": "Not found."
}
```

- **Error (401 Unauthorized)**:

```
{
    "detail": "Authentication credentials were not
       provided."
}
```

```
curl -X DELETE http://127.0.0.1:8000/api/detection/history/1/
   delete/ \
-H "Authorization: Bearer <access_token>"
```

3.5 Clear Prediction History

Example URL `/api/detection/history/clear/`

Method DELETE

Description Deletes all prediction history records for the authenticated user.

Permissions `IsAuthenticated` (requires valid JWT token).

Request Body None

Response

- **Success (204 No Content)**:

```
{
    "message": "X records deleted."
}
```

- **Error (401 Unauthorized)**:

```
{
    "detail": "Authentication credentials were not
       provided."
}
```

```
curl -X DELETE http://127.0.0.1:8000/api/detection/history/
   clear/ \
-H "Authorization: Bearer <access_token>"
```

4 Notes

- **Example Browsable API**: Django REST Framework provides a browsable API interface at each endpoint (e.g., `http://127.0.0.1:8000/api/detection/predict/`). Access it via a web browser to test endpoints interactively.
- **Media Files**: Image uploads for the `/predict/` endpoint are stored in the `media/predicted_imag` directory, accessible via `/media/predicted_images/<filename>`.
- **JWT Configuration**: Access tokens expire after 5 minutes, and refresh tokens expire after 1 day, as configured in `settings.py`.
- **Error Handling**: Ensure proper error handling for invalid image formats or missing fields in requests.
- **Model Loading**: The plant disease prediction model (`plant_disease_prediction_model.h5`) must be present in the `cnn_model` directory relative to the project root.