API Documentation - Medicine Recognition Backend

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

Table of Contents

Authentication Endpoints

1. Register New User

POST `/auth/register/`

Permission: Public (AllowAny)

Request Body:

```
JSON
{
  "username": "testuser",
  "email": "test@example.com",
  "password": "SecurePass123!",
  "password2": "SecurePass123!",
  "first_name": "Ahmed",
  "last_name": "Mohamed"
}
```

Success Response (201):

```
JSON
{
  "user": {
    "id": 2,
    "username": "testuser",
    "email": "test@example.com",
    "first_name": "Ahmed",
    "last_name": "Mohamed",
    "date_joined": "2025-11-29T19:45:00Z"
  },
  "message": "User registered successfully",
  "tokens": {
    "refresh": "eyJ0eXAiOiJKV1QiLCJhbGc...",
    "access": "eyJ0eXAiOiJKV1QiLCJhbGc..."
  }
}
```

Error Response (400):

```
JSON
{
  "username": ["A user with that username already exists."],
  "email": ["This email is already in use."],
  "password": ["Password fields didn't match."]
}
```

---

2. Login (Get JWT Token)

POST `/auth/login/`

Permission: Public (AllowAny)

Request Body:

```
JSON
{
  "username": "testuser",
  "password": "SecurePass123!"
}
```

Success Response (200):

```
JSON
{
  "refresh": "eyJ0eXAiOiJKV1QiLCJhbGc...",
```

```
  "access": "eyJ0eXAiOiJKV1QiLCJhbGc..."
}
```

Token Lifetime:

• Access Token: 1 hour

• Refresh Token: 7 days

---

3. Refresh Access Token

POST `/auth/token/refresh/`

Permission: Public (AllowAny)

Request Body:

```
JSON
{
  "refresh": "eyJ0eXAiOiJKV1QiLCJhbGc..."
}
```

Success Response (200):

```
JSON
{
  "access": "eyJ0eXAiOiJKV1QiLCJhbGc...",
  "refresh": "eyJ0eXAiOiJKV1QiLCJhbGc..."
}
```

---

4. Verify Token

POST `/auth/token/verify/`

Permission: Public (AllowAny)

Request Body:

```
JSON
{
  "token": "eyJ0eXAiOiJKV1QiLCJhbGc..."
}
```

Success Response (200):

```
JSON
{}
```

Error Response (401):

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

---

5. Logout

POST `/auth/logout/`

Permission: Authenticated

Headers:

```
Authorization: Bearer <access-token>
```

Request Body:

```
JSON
{
  "refresh": "eyJ0eXAiOiJKV1QiLCJhbGc..."
}
```

Success Response (200):

```
JSON
{
  "message": "Logout successful"
}
```

Profile Management

6. Get Current User Profile

GET `/auth/profile/`

Permission: Authenticated

Headers:

```
Authorization: Bearer <access-token>
```

Success Response (200):

```JSON
{
  "id": 2,
  "username": "testuser",
  "email": "test@example.com",
  "first_name": "Ahmed",
  "last_name": "Mohamed",
  "date_joined": "2025-11-29T19:45:00Z"
}
```

---

7. Update Profile

PUT/PATCH `/auth/profile/update/`

Permission: Authenticated

Headers:

```
Authorization: Bearer <access-token>
```

Request Body (PUT - all fields required):

```JSON
{
  "email": "newemail@example.com",
  "first_name": "Ahmed",
  "last_name": "Ali"
}
```

Request Body (PATCH - partial update):

```JSON
{
```

```
  "first_name": "Ahmed Updated"
}
```

Success Response (200):

```JSON
{
  "user": {
    "email": "newemail@example.com",
    "first_name": "Ahmed Updated",
    "last_name": "Ali"
  },
  "message": "Profile updated successfully"
}
```

Password Management

8. Change Password

POST `/auth/password/change/`

Permission: Authenticated

Headers:

```
Authorization: Bearer <access-token>
```

Request Body:

```JSON
{
  "old_password": "SecurePass123!",
  "new_password": "NewSecurePass456!",
  "new_password2": "NewSecurePass456!"
}
```

Success Response (200):

```JSON
{
  "message": "Password changed successfully"
}
```

Error Response (400):

```
JSON
{
  "old_password": ["Old password is incorrect."],
  "new_password": ["Password fields didn't match."]
}
```

---

9. Request Password Reset

POST `/auth/password/reset/`

Permission: Public (AllowAny)

Request Body:

```
JSON
{
  "email": "test@example.com"
}
```

Success Response (200):

```
JSON
{
  "message": "Password reset link generated",
  "reset_link": "http://localhost:3000/reset-password/Mg/b123-abc456/",
  "uid": "Mg",
  "token": "b123-abc456",
  "note": "In production, this would be sent via email"
}
```

Note: In production, remove `reset_link`, `uid`, and `token` from response and send via email.

---

10. Confirm Password Reset

POST `/auth/password/reset/confirm///`

Permission: Public (AllowAny)

URL Parameters:

• uidb64: Base64 encoded user ID

• token: Password reset token

Request Body:

```
JSON
{
  "new_password": "NewSecurePass789!",
  "new_password2": "NewSecurePass789!"
}
```

Success Response (200):

```
JSON
{
  "message": "Password reset successful"
}
```

Error Response (400):

```
JSON
{
  "error": "Invalid or expired token"
}
```

Medicine Upload API

11. Upload Medicine Image

POST `/api/uploads/new/`

Permission: Authenticated

Headers:

```
Authorization: Bearer <access-token>
Content-Type: multipart/form-data
```

Request Body (FormData):

```
image: <file> (JPEG, PNG, etc.)
```

Success Response (201):

```
JSON
{
  "id": 1,
  "image": "/media/uploads/2025/11/29/medicine_pic.jpg",
  "image_url": "http://127.0.0.1:8000/media/uploads/2025/11/29/medicine_pic.jpg",
  "result": "{'predicted_name': 'example-medicine', 'confidence': 0.75, 'description': 'Dumm
  "created_at": "2025-11-29T20:00:00Z"
}
```

cURL Example:

```
BASH
curl -X POST http://127.0.0.1:8000/api/uploads/new/ \
  -H "Authorization: Bearer YOUR_ACCESS_TOKEN" \
  -F "image=@/path/to/medicine.jpg"
```

PowerShell Example:

```
POWERSHELL
curl -X POST http://127.0.0.1:8000/api/uploads/new/ `
  -H "Authorization: Bearer YOUR_ACCESS_TOKEN" `
  -F "image=@C:\path\to\medicine.jpg"
```

---

12. List Users Uploads

GET `/api/uploads/`

Permission: Authenticated

Headers:

```
Authorization: Bearer <access-token>
```

Success Response (200):

```
JSON
[
  {
    "id": 3,
    "image": "/media/uploads/2025/11/29/medicine3.jpg",
    "image_url": "http://127.0.0.1:8000/media/uploads/2025/11/29/medicine3.jpg",
    "result": "{'predicted_name': 'aspirin', 'confidence': 0.95}",
```

```
    "created_at": "2025-11-29T20:05:00Z"
  },
  {
    "id": 2,
    "image": "/media/uploads/2025/11/29/medicine2.jpg",
    "image_url": "http://127.0.0.1:8000/media/uploads/2025/11/29/medicine2.jpg",
    "result": "{'predicted_name': 'paracetamol', 'confidence': 0.88}",
    "created_at": "2025-11-29T20:02:00Z"
  }
]
```


Account Management

13. Delete Account

DELETE `/auth/delete/`

Permission: Authenticated

Headers:

```
Authorization: Bearer <access-token>
```

Request Body:

```
JSON
{
  "password": "SecurePass123!"
}
```

Success Response (200):

```
JSON
{
  "message": "Account deleted successfully"
}
```

Error Response (400):

```
JSON
{
  "error": "Incorrect password"
}
```

Authentication Headers

For all authenticated endpoints, include the JWT access token in the header:

```
Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9...
```

Quick Test Flow

1. Register and Login

```bash
# Register
curl -X POST http://127.0.0.1:8000/auth/register/ \
  -H "Content-Type: application/json" \
  -d '{"username":"demo","email":"demo@test.com","password":"Demo123!@#","password2":"Demo12

# Login
curl -X POST http://127.0.0.1:8000/auth/login/ \
  -H "Content-Type: application/json" \
  -d '{"username":"demo","password":"Demo123!@#"}'
```

2. Use Token

```bash
# Save the access token, then:
curl -X GET http://127.0.0.1:8000/auth/profile/ \
  -H "Authorization: Bearer YOUR_ACCESS_TOKEN"
```

3. Upload Image

```bash
curl -X POST http://127.0.0.1:8000/api/uploads/new/ \
  -H "Authorization: Bearer YOUR_ACCESS_TOKEN" \
  -F "image=@medicine.jpg"
```

Admin Panel

Access: `http://127.0.0.1:8000/admin/`

Credentials:

• Username: admin

• Password: admin123

Notes

• Token Blacklist: Logout functionality uses token blacklist - once logged out, refresh tokens cannot be reused

• Token Rotation: New refresh tokens are generated on each refresh request

• Password Reset: Currently returns token in response - in production, send via email

• AI Integration: Edit core/ai_service.py to integrate your actual medicine recognition model

• CORS: Currently allows all origins - restrict in production

• Secret Key: Change DJANGO_SECRET_KEY in production

Error Codes

• 401 Unauthorized: Missing or invalid token

• 400 Bad Request: Validation error

• 404 Not Found: Endpoint doesnt exist

• 403 Forbidden: Insufficient permissions

- 500 Internal Server Error: Server error