**User Authentication System**

Complete Technical Documentation

*Django REST Framework Authentication Service  
with JWT Tokens, Rate Limiting, and Docker Support*

August 30, 2025

Table of Contents

• What it does

• Quick Start

• API Documentation

• Local Development Setup

• Environment Variables

• Testing

• Security Features

• Production Deployment

• Need Help?

User Authentication System

A simple, secure Django REST API for user authentication with JWT tokens.

What it does

• **User registration and login** with email validation

• **JWT token authentication** (access + refresh tokens)

• **Password reset functionality** with secure token validation

• **Rate limiting** on login and password reset endpoints

• **PostgreSQL database** with Redis caching support

• **Docker support** for easy local development

• **Comprehensive unit tests** for all authentication flows

• **API documentation** with Swagger/OpenAPI

• **Production-ready** with security best practices

Quick Start

**Using Docker (recommended for development):**

git clone https://github.com/Iduate/User-Authentication-System.git  
cd User-Authentication-System/auth\_service  
docker-compose up -d

**Access your API:**

* API: http://localhost:8000/api/v1/
* Documentation: http://localhost:8000/swagger/
* Admin: http://localhost:8000/admin/

**Docker includes:**

* PostgreSQL database
* Redis for caching
* All dependencies pre-installed
* Automatic database migrations

API Documentation

**Interactive API Documentation:**

• **Swagger UI**: http://localhost:8000/swagger/

• **ReDoc**: http://localhost:8000/redoc/

• **OpenAPI Schema**: Complete API specification with request/response examples

Authentication Endpoints

Register a new user

POST /api/v1/users/register/  
{  
 "email": "user@example.com",  
 "full\_name": "John Doe",  
 "password": "SecurePassword123!",  
 "password\_confirm": "SecurePassword123!"  
}

Login

POST /api/v1/users/login/  
{  
 "email": "user@example.com",  
 "password": "SecurePassword123!"  
}

Refresh JWT Token

POST /api/v1/users/token/refresh/  
{  
 "refresh": "your-refresh-token-here"  
}

User Profile Endpoints

Get user profile

GET /api/v1/users/profile/  
Headers: Authorization: Bearer <your-access-token>

Password Reset Endpoints

Request password reset

POST /api/v1/users/password-reset/  
{  
 "email": "user@example.com"  
}

Confirm password reset

POST /api/v1/users/password-reset/confirm/  
{  
 "token": "reset-token-from-email",  
 "new\_password": "NewSecurePassword123!",  
 "new\_password\_confirm": "NewSecurePassword123!"  
}

Development Endpoints

Authentication debug (development only)

GET /api/v1/users/auth-debug/

\*Helps debug authentication issues during development\*

Alternative Endpoint Paths

All endpoints are also available at:

* /api/v1/register/ (alternative to /api/v1/users/register/)
* /api/v1/login/ (alternative to /api/v1/users/login/)
* /api/v1/profile/ (alternative to /api/v1/users/profile/)
* /api/v1/password-reset/ (alternative to /api/v1/users/password-reset/)

Security Features

• **Rate limiting** on login and password reset endpoints

* Login: 5 attempts per minute per IP
* Password reset: 3 attempts per hour per IP

• **JWT token authentication** with access and refresh tokens

• **Unit tests** for registration, login, and password reset

• **Docker support** for local development

• **Input validation** and secure password requirements

• **CORS protection** and security headers

Local Development Setup

**Prerequisites:**

* Python 3.11+ installed
* PostgreSQL 12+ installed and running
* Redis installed (optional, will fallback to Django cache)
* Git installed

**Step-by-step setup:**

1. \*\*Clone the repository:\*\*

git clone https://github.com/Iduate/User-Authentication-System.git  
cd User-Authentication-System/auth\_service

1. \*\*Create virtual environment:\*\*

python -m venv venv  
  
# Activate virtual environment  
# Windows:  
venv\Scripts\activate  
# macOS/Linux:  
source venv/bin/activate

1. \*\*Install dependencies:\*\*

pip install -r requirements.txt

1. \*\*Setup database:\*\*

# Create PostgreSQL database  
createdb auth\_db  
  
# Run migrations  
python manage.py migrate  
  
# Create superuser (optional)  
python manage.py createsuperuser

1. \*\*Start development server:\*\*

python manage.py runserver

1. \*\*Access the application:\*\*

* API: http://localhost:8000/api/v1/
* Admin: http://localhost:8000/admin/
* API Documentation: http://localhost:8000/swagger/

Environment Variables

**Required Environment Variables:**

Create a .env file in your project root:

# Django Core Settings  
SECRET\_KEY=your-super-secret-key-here-make-it-long-and-random  
DEBUG=True  
ALLOWED\_HOSTS=localhost,127.0.0.1  
  
# Database Configuration  
DATABASE\_URL=postgres://username:password@localhost:5432/auth\_db  
# Alternative individual database settings:  
DATABASE\_NAME=auth\_db  
DATABASE\_USER=postgres  
DATABASE\_PASSWORD=your-db-password  
DATABASE\_HOST=localhost  
DATABASE\_PORT=5432  
  
# Redis Configuration (Optional)  
REDIS\_URL=redis://localhost:6379/0  
  
# JWT Token Settings (Optional - has defaults)  
JWT\_ACCESS\_TOKEN\_LIFETIME\_MINUTES=60  
JWT\_REFRESH\_TOKEN\_LIFETIME\_DAYS=7  
  
# Email Settings (for password reset)  
EMAIL\_BACKEND=django.core.mail.backends.smtp.EmailBackend  
EMAIL\_HOST=smtp.gmail.com  
EMAIL\_PORT=587  
EMAIL\_USE\_TLS=True  
EMAIL\_HOST\_USER=your-email@gmail.com  
EMAIL\_HOST\_PASSWORD=your-app-password

**Environment Variables Reference:**

|  |  |  |  |
| --- | --- | --- | --- |
| Variable | Description | Required | Default |
| `SECRET\_KEY` | Django secret key for cryptographic signing | ✅ | None |
| `DEBUG` | Enable/disable debug mode | ❌ | `False` |
| `ALLOWED\_HOSTS` | Comma-separated list of allowed hosts | ❌ | `localhost` |
| `DATABASE\_URL` | PostgreSQL connection string | ✅ | None |
| `REDIS\_URL` | Redis connection string | ❌ | None |
| `JWT\_ACCESS\_TOKEN\_LIFETIME\_MINUTES` | Access token expiration | ❌ | `60` |
| `JWT\_REFRESH\_TOKEN\_LIFETIME\_DAYS` | Refresh token expiration | ❌ | `7` |

**For Production:**

* Set DEBUG=False
* Use strong, unique SECRET\_KEY
* Configure proper ALLOWED\_HOSTS
* Use production database URLs
* Set up email configuration for password reset

Testing

**Run all unit tests:**

python manage.py test users

**Test specific functionality:**

# Test registration  
python manage.py test users.tests.test\_registration  
  
# Test login  
python manage.py test users.tests.test\_login  
  
# Test password reset  
python manage.py test users.tests.test\_password\_reset

**Unit test coverage includes:**

* User registration validation
* Login authentication
* Password reset flow
* Rate limiting
* JWT token handling

Production Deployment

**Live Demo:**

🌐 \*\*API Base URL\*\*: https://your-app-name.railway.app/api/v1/

📖 \*\*API Documentation\*\*: https://your-app-name.railway.app/swagger/

🔧 \*\*Admin Panel\*\*: https://your-app-name.railway.app/admin/

**Supported Platforms:**

* ✅ Railway (recommended)
* ✅ Render
* ✅ Heroku
* ✅ DigitalOcean
* ✅ AWS/GCP/Azure

**Quick Deploy to Railway:**

1. Fork this repository
2. Connect to [Railway](https://railway.app)
3. Deploy from GitHub
4. Add environment variables
5. Your API will be live!

**Environment Variables for Production:**

SECRET\_KEY=your-production-secret-key  
DEBUG=False  
ALLOWED\_HOSTS=your-domain.railway.app,your-custom-domain.com  
DATABASE\_URL=postgresql://... (provided by Railway)  
REDIS\_URL=redis://... (provided by Railway)

Need Help?

* Check the [API docs](http://localhost:8000/swagger/) for detailed endpoint information
* Run tests to verify everything works: python manage.py test users
* For issues, create a GitHub issue with details about your problem

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Built with Django REST Framework + JWT authentication. Simple, secure, production-ready.

Document Information

|  |  |
| --- | --- |
| **Document Title** | User Authentication System - Complete Technical Documentation |
| **Project Type** | Django REST Framework Authentication Service |
| **Generated Date** | August 30, 2025 at 07:08 AM |
| **Source File** | README.md |
| **Technologies** | Django, PostgreSQL, Redis, Docker, JWT |
| **API Documentation** | Swagger/OpenAPI available |
| **Test Coverage** | Unit tests for all authentication flows |
| **Deployment Ready** | Railway, Render, Heroku compatible |