Panduan Instalasi dan Penggunaan API

Prasyarat

Sebelum memulai, pastikan Anda sudah menginstall:

- Python 3.8 atau lebih baru
- pip (Python package manager)
- Git (optional, untuk clone repository)

Langkah 1: Setup Project

Clone Repository (Jika dari GitHub)

```
git clone https://github.com/DoItMark/TST-API-DDD.git
cd TST-API-DDD
```

Atau Buat Directory Baru

```
mkdir item-management-api
cd item-management-api
```

Langkah 2: Install Dependencies

Cek File requirements.txt

Pastikan file `requirements.txt` berisi:

```
fastapi==0.104.1
uvicorn[standard]==0.24.0
pydantic==2.5.0
pyjwt==2.8.0
passlib[bcrypt]==1.7.4
```

Install Packages

```
pip install -r requirements.txt
```

Output yang diharapkan:

```
Collecting fastapi==0.104.1
  Downloading fastapi-0.104.1-py3-none-any.whl
Collecting uvicorn[standard]==0.24.0
  Downloading uvicorn-0.24.0-py3-none-any.whl
...
Successfully installed fastapi-0.104.1 uvicorn-0.24.0 pydantic-2.5.0 pyjwt-2.8.0 pass
  lib-1.7.4 bcrypt-4.1.1
```

Verifikasi Instalasi

```
python -c "import fastapi; print('FastAPI:', fastapi.__version__)"
python -c "import jwt; print('PyJWT:', jwt.__version__)"
python -c "import passlib; print('Passlib: OK')"
```

Langkah 3: Jalankan API Server

Menggunakan Python Langsung

```
python listing_api.py
```

Atau Menggunakan Uvicorn

```
uvicorn listing_api:app --reload
```

Parameter:

- `listing_api`: Nama file Python (tanpa .py)
- `app`: Nama variable FastAPI app di dalam file
- `--reload`: Auto-restart server saat code berubah (untuk development)

Output yang diharapkan:

```
INFO:     Will watch for changes in these directories: ['C:\\...\\TST-API-DDD']
INFO:     Uvicorn running on http://127.0.0.1:8000 (Press CTRL+C to quit)
INFO:     Started reloader process [12345] using StatReload
```

```
INFO:     Started server process [12346]
INFO:     Waiting for application startup.
INFO:     Application startup complete.
```

Akses API Documentation

Buka browser dan kunjungi:

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

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

Langkah 4: Testing API dengan cURL

4.1 Register User Baru

```
curl -X POST "http://localhost:8000/register" ^
  -H "Content-Type: application/json" ^
  -d "{\"username\": \"johndoe\", \"password\": \"password123\"}"
```

Response:

```
{
  "message": "User registered successfully",
  "user_id": "a1b2c3d4-e5f6-7890-abcd-ef1234567890",
  "seller_id": "b2c3d4e5-f6a7-8901-bcde-f12345678901"
}
```

Catat seller_id untuk digunakan nanti!

4.2 Login dan Dapatkan Token

```
curl -X POST "http://localhost:8000/login" ^
  -H "Content-Type: application/json" ^
  -d "{\"username\": \"johndoe\", \"password\": \"password123\"}"
```

Response:

```
{
  "access_token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiJqb2huZG9lIiwiZXhwI
  joxNzAxNDM2ODAwfQ.xyz123...",
  "token_type": "bearer"
}
```

Simpan access_token ini untuk request berikutnya!

4.3 Create Listing (Dengan Token)

Ganti `YOUR_TOKEN` dengan access_token yang didapat dari login:

```
curl -X POST "http://localhost:8000/listings" ^
  -H "Content-Type: application/json" ^
  -H "Authorization: Bearer YOUR_TOKEN" ^
  -d "{\"title\": \"Kamera Vintage Canon\", \"price\": {\"amount\": 2999999, \"curren
  cy\": \"IDR\"}, \"condition\": {\"score\": 8, \"detailed_description\": \"Kondisi b
  aik dengan sedikit bekas pakai\", \"known_defects\": [\"Goresan kecil di tutup lens
  a\"]}, \"attributes\": [{\"name\": \"Brand\", \"value\": \"Canon\"}, {\"name\": \"T
  ahun\", \"value\": \"1985\"}]}"
```

Response:

```
{
  "listing_id": "c3d4e5f6-a7b8-9012-cdef-234567890123",
  "seller_id": "b2c3d4e5-f6a7-8901-bcde-f12345678901",
  "title": "Kamera Vintage Canon",
  "item_state": "Active",
  "price": {
    "amount": 2999999,
    "currency": "IDR"
  },
  "condition": {
    "score": 8,
    "detailed_description": "Kondisi baik dengan sedikit bekas pakai",
    "known_defects": ["Goresan kecil di tutup lensa"]
  },
  "attributes": [
    {
      "attribute_id": "d4e5f6a7-b8c9-0123-def4-567890123456",
      "name": "Brand",
      "value": "Canon"
    },
    {
      "attribute_id": "e5f6a7b8-c9d0-1234-ef56-789012345678",
      "name": "Tahun",
      "value": "1985"
    }
  ],
  "created_at": "2025-12-01T10:30:00",
  "updated_at": "2025-12-01T10:30:00"
}
```

Catat listing_id untuk digunakan nanti!

4.4 Get Listing (Public, Tanpa Token)

```
curl -X GET "http://localhost:8000/listings/c3d4e5f6-a7b8-9012-cdef-234567890123"
```

4.5 List All Listings (Public)

```
curl -X GET "http://localhost:8000/listings"
```

Dengan filter seller:

```
curl -X GET "http://localhost:8000/listings?seller_id=b2c3d4e5-f6a7-8901-bcde-f123456
  78901"
```

Dengan pagination:

```
curl -X GET "http://localhost:8000/listings?skip=0&limit=10"
```

4.6 Update Price (Dengan Token)

```
curl -X PATCH "http://localhost:8000/listings/c3d4e5f6-a7b8-9012-cdef-234567890123/pr
  ice" ^
  -H "Content-Type: application/json" ^
  -H "Authorization: Bearer YOUR_TOKEN" ^
  -d "{\"new_price\": {\"amount\": 2499999, \"currency\": \"IDR\"}}"
```

4.7 Activate Listing (Dengan Token)

```
curl -X PATCH "http://localhost:8000/listings/c3d4e5f6-a7b8-9012-cdef-234567890123/ac
  tivate" ^
  -H "Authorization: Bearer YOUR_TOKEN"
```

4.8 Delist Listing (Dengan Token)

```
curl -X PATCH "http://localhost:8000/listings/c3d4e5f6-a7b8-9012-cdef-234567890123/de
  list" ^
  -H "Content-Type: application/json" ^
  -H "Authorization: Bearer YOUR_TOKEN" ^
  -d "{\"reason\": {\"reason_type\": \"SellerRequest\", \"detail\": \"Barang sudah ti
  dak tersedia\"}}"
```

4.9 Delete Listing (Dengan Token)

```
curl -X DELETE "http://localhost:8000/listings/c3d4e5f6-a7b8-9012-cdef-234567890123"
  ^
  -H "Authorization: Bearer YOUR_TOKEN"
```

Response: Status 204 No Content

4.10 Search Listings (Public)

```
curl -X GET "http://localhost:8000/search?query=kamera&min_relevance=1.0"
```

4.11 Health Check (Public)

```
curl -X GET "http://localhost:8000/health"
```

Response:

```
{
  "status": "healthy",
  "listings_count": 5,
  "search_index_count": 5
}
```

Langkah 5: Testing dengan Swagger UI

Swagger UI lebih mudah untuk testing interaktif:

1. Buka browser: http://localhost:8000/docs
2. Klik endpoint `/register` → "Try it out"
3. Isi data user → "Execute"
4. Klik endpoint `/login` → "Try it out"
5. Isi credentials → "Execute"
6. Copy access_token dari response
7. Klik tombol "Authorize" di pojok kanan atas
8. Paste token (tanpa prefix "Bearer ")
9. Klik "Authorize" → "Close"
10. Sekarang Anda bisa test semua protected endpoints!

Langkah 6: Testing Error Cases

6.1 Register dengan Username yang Sudah Ada

```
curl -X POST "http://localhost:8000/register" ^
  -H "Content-Type: application/json" ^
  -d "{\"username\": \"johndoe\", \"password\": \"newpassword\"}"
```

Response (400 Bad Request):

```
{
  "detail": "Username already registered"
}
```

6.2 Login dengan Password Salah

```
curl -X POST "http://localhost:8000/login" ^
  -H "Content-Type: application/json" ^
  -d "{\"username\": \"johndoe\", \"password\": \"wrongpassword\"}"
```

Response (401 Unauthorized):

```
{
  "detail": "Incorrect username or password"
}
```

6.3 Akses Protected Endpoint Tanpa Token

```
curl -X POST "http://localhost:8000/listings" ^
  -H "Content-Type: application/json" ^
  -d "{\"title\": \"Test\", ...}"
```

Response (403 Forbidden):

```
{
  "detail": "Not authenticated"
}
```

6.4 Update Listing Milik Orang Lain

1. Register user kedua
2. Login dengan user pertama, dapatkan token1
3. Login dengan user kedua, dapatkan token2
4. Create listing dengan token1
5. Coba update listing tersebut dengan token2

```
curl -X PATCH "http://localhost:8000/listings/{listing_id}/price" ^
  -H "Authorization: Bearer TOKEN_USER_2" ^
  -d "{\"new_price\": {\"amount\": 1000, \"currency\": \"IDR\"}}"
```

Response (403 Forbidden):

```
{
  "detail": "You can only update price of your own listings"
}
```

Langkah 7: Testing dengan Python Script

Buat file `test_api.py`:

```
import requests
import json

BASE_URL = "http://localhost:8000"

# 1. Register
print("1. Registering user...")
register_response = requests.post(
    f"{BASE_URL}/register",
    json={"username": "testuser", "password": "test123456"}
)
print(f"Status: {register_response.status_code}")
print(f"Response: {register_response.json()}\n")

# 2. Login
print("2. Logging in...")
login_response = requests.post(
    f"{BASE_URL}/login",
    json={"username": "testuser", "password": "test123456"}
)
token = login_response.json()["access_token"]
print(f"Token: {token[:50]}...\n")

# 3. Create Listing
print("3. Creating listing...")
headers = {"Authorization": f"Bearer {token}"}
listing_data = {
    "title": "Laptop Gaming ASUS ROG",
    "price": {"amount": 15000000, "currency": "IDR"},
    "condition": {
        "score": 9,
        "detailed_description": "Seperti baru, garansi 1 tahun",
        "known_defects": []
    },
    "attributes": [
        {"name": "Brand", "value": "ASUS"},
        {"name": "RAM", "value": "16GB"},
        {"name": "Storage", "value": "512GB SSD"}
    ]
}
create_response = requests.post(
    f"{BASE_URL}/listings",
    json=listing_data,
    headers=headers
)
listing = create_response.json()
listing_id = listing["listing_id"]
print(f"Created listing ID: {listing_id}\n")

# 4. Get Listing
print("4. Getting listing...")
get_response = requests.get(f"{BASE_URL}/listings/{listing_id}")
print(f"Title: {get_response.json()['title']}\n")

# 5. Update Price
print("5. Updating price...")
update_response = requests.patch(
    f"{BASE_URL}/listings/{listing_id}/price",
    json={"new_price": {"amount": 14500000, "currency": "IDR"}},
    headers=headers
)
print(f"New price: {update_response.json()['price']}\n")

# 6. List All Listings
print("6. Listing all...")
list_response = requests.get(f"{BASE_URL}/listings")
print(f"Total listings: {len(list_response.json())}\n")

# 7. Health Check
print("7. Health check...")
health_response = requests.get(f"{BASE_URL}/health")
print(f"Status: {health_response.json()}\n")

print("All tests completed!")
```

Jalankan:

```
python test_api.py
```

Tips dan Troubleshooting

Server Tidak Bisa Start

Error: Port 8000 already in use

```
# Cek process yang menggunakan port 8000
netstat -ano | findstr :8000

# Kill process (ganti PID dengan nomor dari output di atas)
taskkill /PID 12345 /F

# Atau gunakan port lain
uvicorn listing_api:app --port 8001
```

Token Expired

JWT token expire setelah 30 menit. Jika mendapat error 401, login kembali untuk mendapat token baru.

Import Error

```
# Pastikan semua dependencies terinstall
pip install -r requirements.txt --upgrade
```

Connection Refused

Pastikan server sedang running di terminal lain:

```
# Terminal 1: Jalankan server
python listing_api.py

# Terminal 2: Test dengan curl
curl http://localhost:8000/health
```

Production Deployment Notes

Untuk production, lakukan perubahan berikut:

1. **Ganti SECRET_KEY**

```
# Generate secure key
import secrets
SECRET_KEY = secrets.token_urlsafe(32)
```

2. **Gunakan Database Real**

• PostgreSQL, MySQL, atau MongoDB

• Ganti in-memory dict dengan database connection

3. **HTTPS Only**

• Deploy dengan reverse proxy (Nginx)

• Force HTTPS untuk secure token transmission

4. **Environment Variables**

• Jangan hardcode SECRET_KEY

• Gunakan .env file atau environment variables

5. **Rate Limiting**

• Add rate limiting untuk prevent abuse

• Gunakan library seperti slowapi

6. **Logging**

• Add proper logging

• Monitor failed login attempts

Kesimpulan

Anda sekarang dapat:

• ✓ Install dan run API server

- ✓ Register user baru
- ✓ Login dan mendapat JWT token
- ✓ Create, read, update, delete listings
- ✓ Test dengan cURL, Swagger UI, atau Python
- ✓ Memahami error handling

Selamat menggunakan API Item Management!