Project Deployment Guide

Environment Requirements

- Python 3.9+
- pip
- Virtual environment tool (venv recommended)

Quick Start

1. Clone the Project

```
cd RuralMuralGenerator
```

2. Activate Virtual Environment

Windows:

```
maral-generator-env\Scripts\activate
```

3. Install Dependencies

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

Alternatively, install dependencies line by line as listed in `requirements.txt`.

4. Configure Environment Variables

```
# Copy environment variable template
copy .env.example .env  # Windows
# or
cp .env.example .env    # Linux/Mac

# Edit .env file and fill in your API keys
notepad .env  # Windows
# or
nano .env     # Linux/Mac
```

Baidu Qianfan Configuration Example

```
from openai import OpenAI

client = OpenAI(
    api_key="bce-v3/ALTAK-KZke********/f1d6ee*************",  # Qianfan bearer
token
    base_url="https://qianfan.baidubce.com/v2",  # Qianfan endpoint
    default_headers={"appid": "app-xxxxxx"}   # Your Qianfan app ID (optional)
)

completion = client.chat.completions.create(
    model="ernie-4.0-turbo-8k", # See model list for preset services; use API URL
for custom services
    messages=[{'role': 'system', 'content': 'You are a helpful assistant.'},
              {'role': 'user', 'content': 'Hello!'}]
)

print(completion.choices[0].message)
```

Required Configuration:

- `DASHSCOPE_API_KEY`: API key for Alibaba Cloud DashScope (Qwen/Tongyi Wanxiang)
    - Apply at: https://dashscope.aliyun.com/
    - Obtain the API key from the console after registration

5. Initialize ChromaDB Knowledge Base

```
python scripts/init_chromadb.py
```

Follow the prompts:

- On first run, enter `y` to reset the database
- Wait for knowledge base initialization to complete
- Optionally test retrieval functionality

6. Start Backend Service

Option 1: Use script (Windows)

```
scripts\start_backend.bat
```

Option 2: Manual start

```
cd backend
# Add project root to PYTHONPATH
powershell
```

```
$env:PYTHONPATH = "D:\03 lesson\05_微服务\RuralMuralGenerator"
powershell

python -m uvicorn main:app --reload --port 8000
```

After successful startup, visit:

- API documentation: http://localhost:8000/docs
- Health check: http://localhost:8000/api/health

7. Start Frontend Service

Open a new terminal window

Option 1: Use script (Windows)

```
scripts\start_frontend.bat
```

Option 2: Manual start

```
cd frontend
streamlit run app.py
```

Upon successful startup, your browser will automatically open: http://localhost:8501

Verify Installation

1. Check Backend Health Status

Visit: http://localhost:8000/api/health

Expected output:

```
{
  "status": "healthy",
  "version": "1.0.0",
  "api_keys_configured": {
    "dashscope": true,
    "government": false,
    "wenxin": false
  },
  "chromadb_status": "healthy (2 documents)"
}
```

2. Test Frontend Connection

In the Streamlit sidebar, you should see:

- ✅ Backend service is running

3. Test Full Workflow

1. Enter test data:
   - Village name: Xidi Village
   - Location: Huangshan City, Anhui Province
   - Historical story: A Ming-Qing ancient village renowned for horse-head walls and wood carvings
2. Click "Start Generation"
3. Wait for cultural analysis to complete
4. Review design proposals
5. Select a proposal to generate an image

Common Issues

Q1: "Backend service not connected" error

Solution:

1. Confirm the backend service is running (check terminal for errors)
2. Verify port 8000 is not occupied
3. Check firewall settings

Q2: ChromaDB initialization fails

Solution:

1. Ensure all dependencies are installed: `pip install -r requirements.txt`
2. Delete the `data/chromadb` directory and re-initialize
3. Verify Python version is 3.9+

Q3: Image generation fails or returns mock images

Causes:

- Tongyi Wanxiang API key not configured
- Insufficient API quota
- Network connectivity issues

Solution:

1. Verify `DASHSCOPE_API_KEY` in `.env` is correct
2. Check API quota in Alibaba Cloud console
3. Test network connectivity

Q4: LLM invocation fails

Solution:

1. Verify API key correctness
2. Check network connectivity
3. Inspect backend logs (`logs/app.log`) for detailed error messages

Q5: Dependency installation fails

Solution:

1. Upgrade pip: `pip install --upgrade pip`
2. Use a domestic mirror:

   ```
   pip install -r requirements.txt -i https://pypi.tuna.tsinghua.edu.cn/simple
   ```

Development Mode

View Logs

Backend log location: `logs/app.log`

View logs in real time:

```
# Windows
type logs\app.log

# Linux/Mac
tail -f logs/app.log
```

Reset Database

```
python scripts/init_chromadb.py
# Enter 'y' to reset the database
```

Test API

Use Swagger UI: http://localhost:8000/docs

Or use curl:

```
curl http://localhost:8000/api/health
```

Production Deployment

Using Docker (Recommended)

```
# Build image
docker-compose build

# Start services
docker-compose up -d

# View logs
docker-compose logs -f

# Stop services
docker-compose down
```

Manual Deployment

1. Use Gunicorn instead of Uvicorn:

```
pip install gunicorn
gunicorn backend.main:app -w 4 -k uvicorn.workers.UvicornWorker --bind
0.0.0.0:8000
```

2. Set up Nginx reverse proxy

3. Configure HTTPS certificate

4. Set environment variables:

```
export BACKEND_RELOAD=false
export LOG_LEVEL=WARNING
```

Performance Optimization

1. Enable Redis Cache

Install Redis and configure:

```
# .env
REDIS_URL=redis://localhost:6379/0
ENABLE_CACHE=true
```

2. Adjust LLM Parameters

Edit `.env`:

```
LLM_TEMPERATURE=0.7
LLM_MAX_TOKENS=2000
```