《山海经》文化传承展示系统开发文档

一、项目概述

1.1 项目背景

基于《山海经》古籍的数字化展示需求，结合现代信息技术构建文化传承平台，实现文本解析、多模态可视化和 AI 图像生成功能。

1.2 技术栈

类别	技术选型
前端框架	HTML5/CSS3/JavaScript
后端框架	Django 4.2
可视化库	Pyecharts 2.0
AI 推理	DeepSeek API + Transformers
图像生成	Diffusers + PyTorch 2.0
数据库	MySQL 8.0 + PyMySQL
数据处理	Pandas 2.0 + OpenCV 4.8

二、系统架构设计

2.1 整体架构

用户端 (浏览器)
↑
Django Web 服务 (ASGI)
↑
服务层：数据处理 | AI 推理 | 可视化生成
↑
数据层：MySQL 数据库 | 文件存储

2.2 目录结构

shanhaijing/

```
├── apps/
│   ├── core/        # 核心功能
│   ├── visualization/ # 可视化模块
│   ├── ai_generation/ # AI 生成模块
│   └── users/       # 用户模块
├── config/          # 项目配置
├── static/          # 静态资源
├── templates/       # 前端模板
├── data/            # 原始数据
└── manage.py
```

三、数据库设计

3.1 数据表结构

sql

```sql
CREATE SCHEMA `shanhaijin` ;
CREATE TABLE `shanhaijin`.`shj` (
  `name` VARCHAR(100) NOT NULL,
  `appearance` VARCHAR(1000) NULL,
  `original_habitat` VARCHAR(1000) NULL,
  `morden_location` VARCHAR(1000) NULL,
  `source_excerpt` VARCHAR(1000) NULL,
  `source_translation` VARCHAR(1000) NULL,
  `abilities` VARCHAR(500) NULL,
  `symbolism` VARCHAR(100) NULL,
   `category` VARCHAR(30) NULL,
   `mountain_location` VARCHAR(10),
    `chapter` VARCHAR(20),
  PRIMARY KEY (`name`));
```

四、核心模块实现

4.1 古籍解析模块

python

```python
# apps/core/views.py
import openai

def analyze_text(request):
    text_chunk = get_text_from_db()
    response = openai.ChatCompletion.create(
        model="deepseek-chat",
        messages=[{
            "role": "user",
            "content": f"分析山海经文本：{text_chunk}"
        }]
    )
    save_analysis_result(response.choices[0].message['content'])
    return JsonResponse({'status': 'success'})
```

4.2 可视化模块

```
# apps/visualization/charts.py
from pyecharts.charts import Geo

def create_geo_map():
    geo = Geo()
    geo.add_schema(maptype="china")
    # 添加异兽地理位置数据
    for beast in MythicalBeast.objects.all():
        geo.add_coordinate(beast.name, *beast.location)
    return geo.render_embed()
```

4.3 AI 图像生成

```
# apps/ai_generation/generators.py
from diffusers import StableDiffusionPipeline
def generate_beast_image(description):
    pipeline = StableDiffusionPipeline.from_pretrained("stabilityai/stable-diffusion-2")
    image = pipeline(description).images[0]
    return save_image(image)
```

五、接口设计

5.1 RESTful API

端点	方法	功能描述
/api/beasts/	GET	获取异兽列表
/api/analysis/{id}/	GET	获取文本解析结果
/api/generate-image/	POST	提交生成图像请求
/api/visualization/{type}	GET	获取可视化数据

六、前端交互实现

6.1 搜索组件示例

javascript

```
// static/js/search.js
document.getElementById('beast-search').addEventListener('input', async (e) => {
    const response = await fetch(`/api/beasts/?q=${e.target.value}`);
    const results = await response.json();
    updateSearchResults(results);
});
```

七、部署配置

7.1 环境配置

python

```
# config/settings.py
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.mysql',
        'NAME': 'shanhaijing',
        'USER': 'db_user',
        'PASSWORD': 'secure_password',
        'HOST': '127.0.0.1',
        'PORT': '3306',
    }
}

# AI 配置
DEEPSEEK_API_KEY = 'your_api_key_here'
```

7.2 生产环境部署

bash

```
# 安装依赖
pip install -r requirements.txt

# 数据库迁移
python manage.py migrate

# 收集静态文件
python manage.py collectstatic

# 启动服务（生产环境）
gunicorn --workers 4 --bind 0.0.0.0:8000 config.asgi:application
```

八、测试方案

1. 单元测试：使用 Django TestCase 验证各模块功能
2. 集成测试：使用 Selenium 进行端到端测试
3. 性能测试：Locust 进行并发压力测试
4. 安全测试：SQL 注入/XSS 防护测试

九、项目进度计划

阶段	时间线	交付物
基础框架搭建	第 1-2 周	Django 项目骨架
核心模块开发	第 3-5 周	文本解析+可视化功能
AI 集成开发	第 6-7 周	图像生成模块

阶段	时间线	交付物
测试优化	第 8 周	测试报告+性能优化
部署上线	第 9 周	生产环境部署文档

十、注意事项

1. API 调用频率需遵守 DeepSeek 平台限制
2. 图像生成需要至少 8GB 显存的 GPU 支持
3. 古籍文本数据需进行 UTF-8 编码处理
4. 地理坐标数据采用 WGS84 坐标系
5. 用户敏感数据需进行加密存储

requirements.txt