# 甜品日记智能客服系统实现指南

## 1. 环境准备

### 1.1 Python环境配置

```
# 创建新的conda环境
conda create -n openai-env python=3.10
conda activate openai-env

# 安装核心依赖包（按顺序安装）
pip install langchain>=0.2.0 langchain-openai langchain-community langchain-core
langchain-text-splitters python-dotenv faiss-cpu dashscope gradio==3.50.2 -i
https://pypi.tuna.tsinghua.edu.cn/simple
```

### 1.2 环境变量配置

创建 `.env` 文件，添加以下内容：

```
# OpenAI API配置（使用Deepseek API）
OPENAI_API_KEY="sk-7198025cefc94e77a647d30c40f529db"  #for training
OPENAI_API_BASE="https://api.deepseek.com/v1"

# DashScope API配置（用于向量嵌入）,培训期间共用此key，你也可在
https://dashscope.aliyun.com/ 注册自己的key
DASHSCOPE_API_KEY="sk-62244f697212473f8624c61ab08b362f"

# LangSmith配置（可选，用于调试）
LANGCHAIN_TRACING_V2=true
LANGCHAIN_ENDPOINT="https://api.smith.langchain.com"
LANGCHAIN_API_KEY="your_langsmith_api_key"
LANGCHAIN_PROJECT="your_project_name"
```

### 1.3 项目结构设置

```
happyCake/
├── app/
|   ├── main.py
|   └── chains/
|       ├── rag_chain.py
|       └── chat_chain.py
├── knowledge_base/
|   ├── products/
|   |   └── xx.md
|   ├── orders/
|   |   └── ordering_guide.md
|   └── faq/
|       └── common_questions.md
```

```
├── .env
└── requirements.txt
```

### 1.4 验证环境配置

```
# 1. 测试环境变量加载
python -c "from dotenv import load_dotenv; load_dotenv(); import os;
print(os.getenv('OPENAI_API_KEY'))"

# 2. 测试LangChain导入
python -c "from langchain_openai import ChatOpenAI; from
langchain_community.embeddings import DashScopeEmbeddings"

# 3. 测试Gradio安装
python -c "import gradio; print(gradio.__version__)"
```

## 2. 系统部署步骤

### 2.1 知识库初始化

```
# 1. 确保知识库文档已放置在正确位置
# 2. 运行知识库加载器创建向量存储
python knowledge_base_loader.py
```

### 2.2 启动系统

选择以下任一方式启动系统：

```
# 命令行界面
python cli_interface.py

# 或者 Web界面
python gradio_interface.py
```

### 2.3 访问Web界面

- 本地访问：http://localhost:7860
- 局域网访问：http://本机IP:7860

## 3. 常见问题处理

### 3.1 依赖安装问题

```
# 如果安装依赖时出错，可以尝试：
pip install --upgrade pip
pip install -r requirements.txt --no-cache-dir

# 如果gradio安装有问题，可以尝试：
pip uninstall gradio
pip install gradio==3.50.2 --no-cache-dir
```

## 3.2 运行时错误处理

1. 向量存储加载错误：

```
# 重新生成向量存储
rm -rf knowledge_base/vector_store/*
python knowledge_base_loader.py
```

2. API密钥错误：

```
# 检查环境变量是否正确加载
python -c "from dotenv import load_dotenv; load_dotenv(); import os;
print('OPENAI_API_KEY:', os.getenv('OPENAI_API_KEY'));
print('DASHSCOPE_API_KEY:', os.getenv('DASHSCOPE_API_KEY'))"
```

3. Gradio界面无响应：

```
# 检查端口占用
netstat -ano | findstr :7860
# 如果端口被占用，可以在launch时指定其他端口
# 在gradio_interface.py中修改端口号：
demo.launch(server_port=7861)
```

# 4. 性能优化建议

1. 向量存储优化：
   - 适当调整chunk_size（默认500）和chunk_overlap（默认50）
   - 考虑使用更大的k值进行检索（默认k=3）
2. 模型响应优化：
   - 调整temperature参数（默认0.7）
   - 优化提示模板以获得更好的响应
3. 系统性能监控：
   - 使用LangSmith追踪链的执行情况
   - 监控API调用延迟和错误率

# 5. 安全注意事项

1. API密钥保护：
    - 不要将.env文件提交到版本控制
    - 定期更换API密钥
    - 在生产环境使用环境变量而不是.env文件
2. 向量存储安全：
    - 定期备份向量存储
    - 控制向量存储的访问权限
    - 注意FAISS反序列化的安全风险

```
```