阶段用户控制功能

```
**[Seminar Conclusion]**:

 <INFO>  Website

连接失败: 1045 (28000): Access denied for user 'root'@'localhost' (using password: Y
ES)

阶段 DemandAnalysis 的结论:
--------------------------------------------------
  Website
--------------------------------------------------

请选择操作:
1. 继续下一个阶段
2. 修改当前阶段
3. 查看文档
请输入选项 (1/2/3) : 3

文档已生成: C:\Users\25740\Desktop\软件工程\talkse2\software-engineering-project-mai
n\WareHouse\shopping_system_DefaultOrganization_2025-06-04_16-02-16\DemandAnalysis_d
ocumentation.md
```

Coding 阶段文档

结论

To develop an online shopping system using Python, we will structure our application into several core components. Below is the outline of the core classes, functions, and methods that will be necessary for our system:

Core Classes and Functions

1. **Main Application Class (`OnlineShoppingApp`)**
   - Purpose: To initialize and run the main application, manage the GUI, and handle user interactions.
2. **User Class (`User`)**
   - Purpose: To manage user information, including registration, login, and user profile.
3. **Product Class (`Product`)**
   - Purpose: To represent products available for purchase, including attributes like name, price, and description.
4. **Shopping Cart Class (`ShoppingCart`)**
   - Purpose: To manage the items added to the cart, including adding, removing, and calculating total price.
5. **Payment Class (`Payment`)**
   - Purpose: To handle payment processing and integration with a payment gateway.
6. **Database Class (`Database`)**
   - Purpose: To manage database connections and operations for storing user and product data.

File Structure

1. `main.py` - Main entry point for the application.
2. `user.py` - Contains the `User` class.
3. `product.py` - Contains the `Product` class.

一、已完成的修改

1. **Phase 类的增强**
   - 添加了 `role_prompts` 参数支持
   - 增加了状态管理相关属性（`phase_state`）
   - 添加了阶段状态管理方法（`get_phase_state`, `update_phase_state`, `reset_phase_state`）

2. **ComposedPhase 类的修改**
    - 在初始化 Phase 实例时添加了 `assistant_role_prompt` 和 `user_role_prompt` 参数
    - 从 `role_prompts` 字典中获取角色提示
3. **ChatChain 类的增强**
    - 在 `execute_step` 方法中添加了用户交互功能
    - 实现了阶段结论展示、修改和文档生成功能
    - 修复了时间格式处理问题

二、API 接口说明

1. **阶段状态管理接口**

```python
# 获取阶段状态
phase.get_phase_state()
# 返回格式：
{
    'task_prompt': str,
    'current_turn': int,
    'is_completed': bool,
    'needs_restart': bool,
    'restart_prompt': str
}

# 更新阶段状态
phase.update_phase_state(**kwargs)
# 参数示例：
{
    'task_prompt': '新的任务提示',
    'current_turn': 1,
    'is_completed': False
}

# 重置阶段状态
phase.reset_phase_state()
```

2. **阶段执行接口**

```python
# 执行阶段
phase.execute(chat_env, chat_turn_limit, need_reflect)
# 参数说明：
# - chat_env: 聊天环境对象
# - chat_turn_limit: 最大对话轮数
# - need_reflect: 是否需要反思
# 返回：更新后的 chat_env
```

3. **文档生成接口**

```
# 生成阶段文档
doc_path = os.path.join(chat_env.env_dict['directory'], f"
{phase}_documentation.md")
# 文档内容：
# - 阶段结论
# - 环境信息
# - 阶段状态
```

三、前后端开发建议

1. **前端开发**
   - 可以基于现有的用户交互界面，开发更友好的 Web 界面
   - 需要实现的功能：
     - 阶段执行状态显示
     - 结论展示和修改界面
     - 文档查看和下载
     - 阶段控制（继续/修改/查看文档）
2. **后端开发**
   - 可以基于现有的 Flask 服务器进行扩展
   - 需要实现的 API：

```
# 阶段控制 API
@app.route('/api/phase/execute', methods=['POST'])
@app.route('/api/phase/modify', methods=['POST'])
@app.route('/api/phase/document', methods=['GET'])

# 状态管理 API
@app.route('/api/phase/state', methods=['GET', 'POST'])
@app.route('/api/phase/reset', methods=['POST'])
```

3. **数据库设计**
   - 需要存储的信息：
     - 阶段状态
     - 阶段结论
     - 修改历史
     - 文档内容
4. **扩展建议**
   - 可以添加用户认证系统
   - 实现多项目并行开发
   - 添加版本控制功能
   - 实现实时协作功能

四、使用示例

1. **命令行模式**

```
python run.py --config Online_Shopping_System --task "开发一个在线购物系统" --name shopping_system
```

2. **API 调用示例**

```python
# 执行阶段
response = requests.post('http://localhost:5000/api/phase/execute',
    json={'phase': 'DemandAnalysis', 'task': '开发在线购物系统'})

# 修改阶段
response = requests.post('http://localhost:5000/api/phase/modify',
    json={'phase': 'DemandAnalysis', 'modification': '添加支付功能'})

# 获取文档
response = requests.get('http://localhost:5000/api/phase/document',
    params={'phase': 'DemandAnalysis'})
```

这些接口和功能为前后端开发提供了良好的基础，开发团队可以根据具体需求进行扩展和优化。

电商平台生成示例