基于我对希望杀文档的阅读，我现在为您制定一个详细的开发计划：

**希望杀卡牌游戏开发计划**

**项目概述**

这是一个基于Flask的多人卡牌游戏，类似于三国杀，包含作业牌、体术牌、事件牌、装备牌、状态牌、时间牌和魔法牌等多种卡牌类型。

**技术架构**

**后端 (Python Flask)**

1. **核心框架**: Flask + Flask-SocketIO (用于实时通信)
2. **数据库**: SQLite (初期) / PostgreSQL (后期扩展)
3. **卡牌逻辑引擎**: 自定义Python类实现
4. **房间管理系统**: 支持2人对战
5. **用户认证**: 简单的session管理

**前端**

1. **基础布局**: HTML5 + CSS3
2. **交互逻辑**: JavaScript + Socket.IO客户端
3. **UI设计**: 现代化卡牌游戏界面
4. **响应式设计**: 适配不同屏幕尺寸

**开发阶段规划**

**第一阶段：基础架构 (1-2周)**

1. **Flask应用搭建**

* 基础路由设置
* 静态文件服务
* 跨域配置

1. **数据库设计**

* 用户表
* 游戏房间表
* 卡牌数据表
* 游戏状态表

1. **基础卡牌系统**

* 卡牌基类设计
* 简单卡牌效果实现（作业牌、体术牌）
* 卡牌效果触发机制

**第二阶段：核心游戏逻辑 (2-3周)**

1. **游戏状态管理**

* 回合制系统
* 玩家状态管理
* 血量系统 (san值)

1. **基础卡牌实现**

* **作业牌**: 一套卷子、驳回等
* **体术牌**: 闪避、反击等
* **装备牌**: 基础装备效果
* **状态牌**: 基础状态效果

1. **角色系统**

* 原皮角色实现（根据文档中的角色技能）
* 角色技能触发机制

**第三阶段：前端界面 (2-3周)**

1. **游戏界面设计**

* 卡牌展示区域
* 玩家信息面板
* 手牌区域
* 装备/状态区域

1. **交互功能**

* 卡牌拖拽使用
* 目标选择
* 效果确认

1. **实时通信**

* WebSocket连接
* 游戏状态同步
* 玩家操作广播

**第四阶段：完善和测试 (1-2周)**

1. **功能完善**

* 更多卡牌效果
* 游戏平衡性调整
* 错误处理

1. **测试和优化**

* 功能测试
* 性能优化
* 用户体验改进

**核心功能实现重点**

**卡牌系统**

* **作业牌**: 造成伤害，目标选择
* **体术牌**: 防御、反击效果
* **装备牌**: 持续效果，装备栏管理
* **状态牌**: 临时效果，状态管理

**角色系统**

* 实现原皮角色的特殊技能
* 技能触发时机管理
* 角色平衡性

**游戏机制**

* 回合制流程
* 卡牌使用时机
* 效果结算顺序
* 胜负判定

**技术难点和解决方案**

1. **实时同步**: 使用WebSocket确保游戏状态实时同步
2. **卡牌效果**: 设计灵活的效果系统，支持复杂交互
3. **状态管理**: 使用状态机管理游戏流程
4. **数据一致性**: 服务器权威，客户端只负责展示

**文件结构规划**

**text**

希望杀/

├── app/

│   ├── \_\_init\_\_.py

│   ├── models/

│   │   ├── user.py

│   │   ├── game.py

│   │   └── card.py

│   ├── routes/

│   │   ├── main.py

│   │   └── game.py

│   ├── game\_logic/

│   │   ├── card\_effects.py

│   │   ├── game\_state.py

│   │   └── character\_skills.py

│   └── static/

│       ├── css/

│       ├── js/

│       └── images/

├── templates/

├── config.py

└── run.py