项目分工

- **区块链底层组 (2人)**: 负责构建简易区块链核心与虚拟机。
- **智能合约组 (2人)**: 负责实现运维 SOP 逻辑与 Token 经济模型。
- **Agent 适配组 (2人)**: 负责 Agent 与区块链的对接及 Prompt 改造。
- **前端交互组 (2人)**: 负责可视化界面与交互控制台。

详细任务分配

成员	角色	核心负责文件/模块	详细任务描述
成员 1	**区块链核心开发 (数据层)**	`mABC/core/blockchain.py` `mABC/core/types.py`	1. **数据结构定义**: 实现 `Block` (区块)、`Transaction` (交易)、`Account` (账户) 类。 2. **密码学基础**: 实现 SHA256 哈希计算、Merkle Tree 构建与验证逻辑。 3. **链式存储**: 实现区块链的主链维护逻辑，确保区块哈希链接的正确性。
成员 2	**区块链核心开发 (执行层)**	`mABC/core/vm.py` `mABC/core/state.py`	1. **虚拟机 (VM)**: 实现一个简易的状态机执行引擎，用于处理交易并更新状态。 2. **交易执行**: 实现交易的签名验证、Nonce 检查及 Gas 扣除逻辑。 3. **状态管理**: 维护全局状态 (World State)，确保数据的一致性。
成员 3	**智能合约开发 (流程控制)**	`mABC/contracts/ops_contract.py`	1. **SOP 状态机**: 编写合约逻辑，严格定义运维流程状态 (`Init` -> `Data_Collected` -> `Root_Cause_Proposed` -> `Consensus` -> `Solution`)。 2. **前置校验**: 实现每个步骤的准入条件检查 (如：只有数据上链后才能进行分析)。 3. **事件日志**: 定义关键步骤的 Event，供前端查询。
成员 4	**智能合约开发 (经济模型)**	`mABC/contracts/token.py` `mABC/contracts/governance.py`	1. **Token 系统**: 实现 `OpsToken` 的发行、转账、余额管理逻辑。 2. **共识机制**: 实现 PoS 质押 (Staking)、投票 (Voting) 逻辑。 3. **奖惩机制**: 实现提案通过后的奖励分发，以及被证伪后的罚没 (Slashing) 逻辑。
成员 5	**Agent 中间件开发**	`mABC/agents/base/dao_run.py` `mABC/core/client.py`	1. **DAOExecutor**: 开发新的运行器替代 `ThreeHotCotRun`，负责与链交互。 2. **交易封装**: 实现将 Agent 的决策 (Thought/Action) 自动封装为 `Transaction` 对象。 3. **回执处理**: 监听链上事件，将合约执行结果 (Receipt) 反馈给 Agent。
成	**Agent**	`mABC/agent`	1. **身份管理**: 为每个 Agent 分配模拟的钱包地址与私

成员	角色	核心负责文件/模块	详细任务描述
员6	**适配与Prompt工程**	`s/base/profile.py` `mABC/utils/prompts.py`	钥。 2. **Prompt 改造**: 重写 System Prompt，让 Agent 理解“质押”、“交易”、“Gas”等概念，并输出结构化决策。 3. **行为调优**: 调整 Agent 的 Temperature 等参数，使其在涉及 Token 质押时更加谨慎。
成员7	**前端开发 (区块链浏览器)**	`frontend/src/components/Explorer`	1. **区块可视化**: 展示区块链的链式结构、区块高度、哈希值。 2. **交易追踪**: 展示每笔交易的发送方、接收方、Payload 及执行结果。 3. **审计视图**: 实现 Merkle Proof 的可视化验证，生成“可信审计报告”。
成员8	**前端开发 (运维控制台)**	`frontend/src/components/Dashboard`	1. **状态监控**: 实时展示当前运维 SOP 所处的阶段 (状态机视图)。 2. **经济看板**: 展示各 Agent 的 Token 余额、信誉分排行、质押情况。 3. **交互中心**: 可视化展示 Agent 之间的投票博弈过程及最终决策路径。

详细说明：

1. 区块链底层组 (成员 1 & 2)

目标：从零实现一个轻量级的 Python 区块链，不需要复杂的 EVM 兼容，只需满足存证和简单的状态流转。

技术选型：

语言：Python 3.9+
数据校验：Pydantic（定义区块/交易结构），hashlib（SHA256），ecdsa 或 eth-keys（签名/验签）。
存储：LevelDB（plyvel）或简单的 SQLite（用于持久化区块和状态）。
通信：gRPC 或 FastAPI（如果节点间需要通信，或者仅作为单机模拟环境则不需要网络层）。
实现思路：

数据层（blockchain.py）：定义 Block 类（包含 Header 和 Body），Header 包含 prev_hash，timestamp，merkle_root。实现一个 Chain 类管理区块列表，提供 add_block() 方法。
执行层（vm.py）：不需要完整的图灵完备虚拟机。实现一个 StateProcessor，接收 Transaction，根据交易类型（如：PROPOSE_ROOT_CAUSE，VOTE）调用对应的 Python 函数更新内存中的 WorldState（字典结构）。

2. 智能合约组 (成员 3 & 4)

目标：用 Python 代码模拟智能合约逻辑，控制运维流程和经济模型。

技术选型：

语言：Python（作为“预编译合约”直接嵌入在链代码中）。
状态机：transitions 库（用于管理 SOP 状态流转）。
实现思路：

SOP 合约（ops_contract.py）：设计一个类，内部维护一个状态变量 current_state。定义方法如 submit_analysis(agent_id, content)，方法内检查 current_state == 'Data_Collected'，成功则更新状态并记录日志。
Token 经济（token.py）：维护一个全局账本 balances = {address: amount}。实现 stake(address, amount) 锁定代币，slash(address, amount) 扣除代币。
共识逻辑：在 Consensus 阶段，统计各 Agent 对某个提案的投票权重（权重 = 质押量 * 信誉分），超过阈值则通过。

3. Agent 适配组 (成员 5 & 6)

目标：让现有的 AIOps Agent (基于 LLM) 学会像区块链用户一样“签名交易”和“支付 Gas”。

技术选型：

LLM 框架：沿用项目中现有的（可能是 LangChain 或自定义的 llm.py）。
接口适配：Python 装饰器或 Wrapper 模式。
实现思路：

中间件（dao_run.py）：改造 Agent 的 Action 输出。当 Agent 决定“诊断故障”时，不再直接打印结果，而是构造一个 Transaction 对象，用 Agent 的私钥签名，发送给区块链模块。
Prompt 工程（profile.py）：
System Prompt："你是一个去中心化运维专家。你的每一个分析都需要质押 Token，如果分析错误，Token 将被罚没。请谨慎决策。"
Output Format：强制 Agent 输出 JSON 格式，包含 action_type, payload, stake_amount。

4. 前端交互组 (成员 7 & 8)

目标：可视化展示链上数据和 Agent 的博弈过程。

技术选型：

框架：React（推荐）或 Vue 3。
UI 组件库：Ant Design 或 Material UI。
可视化：
React Flow 或 Mermaid.js：绘制 SOP 流程图和 Agent 决策树。
ECharts 或 Recharts：展示 Token 余额变化、投票统计。
API：后端使用 FastAPI 暴露区块链数据接口。
实现思路：

区块链浏览器：调用后端 /get_blocks 接口，渲染区块列表。点击区块显示交易详情（JSON 树状展示）。
运维控制台：
左侧：实时日志流（显示 Agent 的 Thought 和 Action）。
中间：拓扑图或状态机视图，高亮当前 SOP 阶段。
右侧：经济看板，显示各 Agent 的 Token 盈亏排行。

后端通过 main.py 启动一个 Loop，模拟“出块”和“交易执行”。
前端独立在 frontend/ 目录下，通过 REST API 读取后端状态。

协作流程建议

1. **第一阶段 (基础建设)**: 成员 1、2 完成区块链底层；成员 3、4 完成合约逻辑设计。
2. **第二阶段 (中间件对接)**: 成员 5、6 基于底层接口，改造 Agent 逻辑，使其能跑通“发起交易 -> 上链 -> 状态更新”的闭环。
3. **第三阶段 (前端对接)**: 成员 7、8 对接后端 API，展示实时数据。
4. **第四阶段 (联调与测试)**: 全员参与，进行全链路测试，验证“幻觉消除”与“自动奖惩”效果。