AI Engineering Employee Platform V2

项目代号

AI Engineering Employee

一、项目目标

构建一个能够自主完成软件研发工作的 AI 工程团队。

能力范围：

- 项目分析
- 架构设计
- 代码开发
- 自动测试
- Bug修复
- Code Review
- 文档生成
- Git提交
- PR生成
- 多Agent协同

最终实现：

```
需求
 ↓
Supervisor
 ↓
Architect
 ↓
Developer
 ↓
Tester
 ↓
Reviewer
 ↓
Reporter
 ↓
Git PR
 ↓
Human Approval
```

二、当前状态

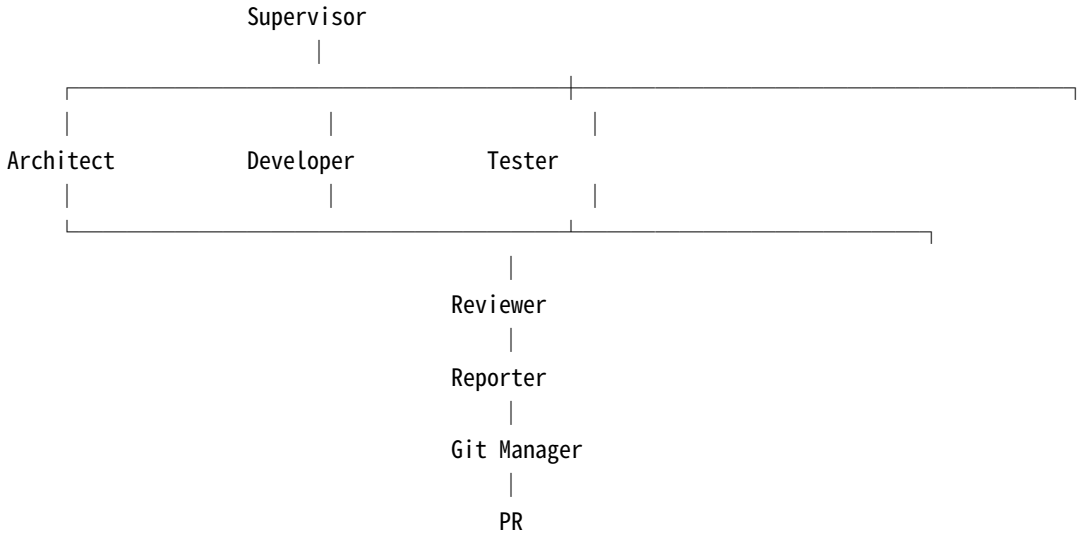
已完成：

- LangGraph工作流
- Sandbox隔离
- Claude Code接入
- DeepSeek接入
- 条件路由
- 测试执行
- 报告输出

当前流程：

```
prepare_sandbox
 ↓
review_project
 ↓
execute_code
 ↓
run_tests
 ↓
generate_report
```

三、V2总体架构



四、目录结构

```
app/

├── main.py
├── state.py

├── graphs/
│
├── engineering_graph.py
├── review_graph.py
├── bugfix_graph.py
├── release_graph.py

├── agents/
│
├── supervisor.py
├── architect.py
├── developer.py
├── tester.py
├── reviewer.py
├── reporter.py

├── executors/
│
├── base_executor.py
├── claude_executor.py
├── codex_executor.py
├── openhands_executor.py
├── gemini_executor.py

├── tools/
│
├── sandbox.py
├── git_manager.py
├── docker_manager.py
├── report_generator.py

├── memory/
│
├── project_memory.py
├── vector_store.py

├── configs/
│
├── settings.py

└── reports/
```

五、State设计

```
class EngineeringState(TypedDict):

    task_id: str

    user_request: str

    repo_path: str

    sandbox_path: str

    selected_executor: str

    project_analysis: dict

    architecture_plan: dict

    implementation_plan: list

    code_result: dict

    test_result: dict

    review_result: dict

    diff_result: dict

    report: str

    status: str

    attempts: int
```

六、Agent设计

Supervisor Agent

职责：

```
任务拆解
Agent分配
流程控制
失败重试
```

输入：

```
用户需求
```

输出：

```
{
 "tasks": [
  "设计数据库",
  "开发接口",
  "开发前端",
  "编写测试"
 ]
}
```

Architect Agent

职责：

```
项目分析
架构设计
目录规划
数据库设计
接口设计
```

输出：

```
{
 "architecture": {},
 "database": {},
 "api": {}
}
```

Developer Agent

职责：

```
代码开发
代码修改
功能实现
```

执行器：

```
Claude Code
Codex
OpenHands
```

Tester Agent

职责：

```
自动测试
自动验证
生成覆盖率报告
```

支持：

```
pytest
npm test
pnpm test
go test
cargo test
```

Reviewer Agent

职责：

```
代码Review
Bug检测
安全检查
性能检查
```

输出：

```
{
 "bugs": [],
 "security": [],
 "performance": []
}
```

Reporter Agent

职责：

```
生成最终报告
生成PR说明
生成交付文档
```

七、Executor抽象

定义：

```python
class BaseExecutor:

    def run(
        self,
        repo_path: str,
        task: str
    ):
        pass
```

Claude Executor

```
ClaudeExecutor
```

使用：

```
Claude Code
DeepSeek Anthropic API
```

Codex Executor

```
CodexExecutor
```

使用：

```
OpenAI Codex CLI
```

OpenHands Executor

```
OpenHandsExecutor
```

使用：

```
OpenHands Runtime
```

八、Sandbox系统

目标：

```
永远不修改原项目
```

流程：

```
原项目
 ↓
复制
 ↓
sandbox/task_xxx
 ↓
Agent工作
 ↓
测试
 ↓
生成diff
```

结构：

```
sandboxes/

task_001
task_002
task_003
```

功能：

```
create_sandbox()
cleanup_sandbox()
```

九、Git Manager

新增：

```
app/tools/git_manager.py
```

功能：

```
git_diff()
git_commit()
git_branch()
git_pr()
changed_files()
```

报告中输出：

```
Modified Files

app/graph.py
app/state.py

Git Diff

+124
-28
```

十、Workflow设计

Engineering Graph

```
prepare_sandbox
 ↓
analyze_project
 ↓
architect
 ↓
developer
```

```
 ↓
tester
 ↓
reviewer
 ↓
reporter
 ↓
git_manager
```

Bugfix Graph

```
analyze_bug
 ↓
fix_bug
 ↓
run_tests
 ↓
review_fix
 ↓
report
```

Review Graph

```
scan_code
 ↓
security_review
 ↓
performance_review
 ↓
report
```

十一、数据库设计

建议：

```
PostgreSQL
```

表：

```
tasks
agents
executions
reports
projects
```

十二、API设计

FastAPI

```
POST /tasks

GET /tasks/{id}

GET /reports/{id}

POST /projects

GET /projects
```

十三、队列系统

Redis

Celery

任务：

```
开发任务
测试任务
Review任务
报告任务
```

十四、长期路线图

Phase 1 ✓ MVP

Phase 2 ✓ Sandbox ✓ Git Diff

Phase 3

- Codex
- OpenHands
- Gemini

Phase 4

- Web UI
- FastAPI
- Redis

Phase 5

- 多Agent协同

Phase 6

- AI研发团队

```
1 Supervisor

2 Architect

3 Developer

2 Tester

1 Reviewer
```

实现全自动软件研发流水线。