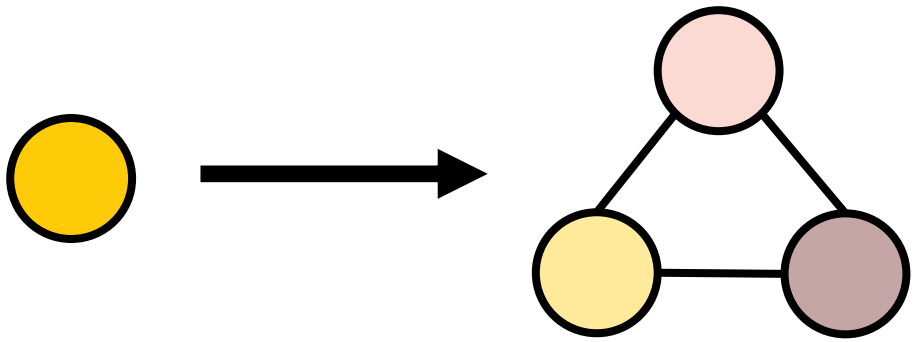
MetaGPT: Meta Programming For A Multi-Agent Collaborative Framework

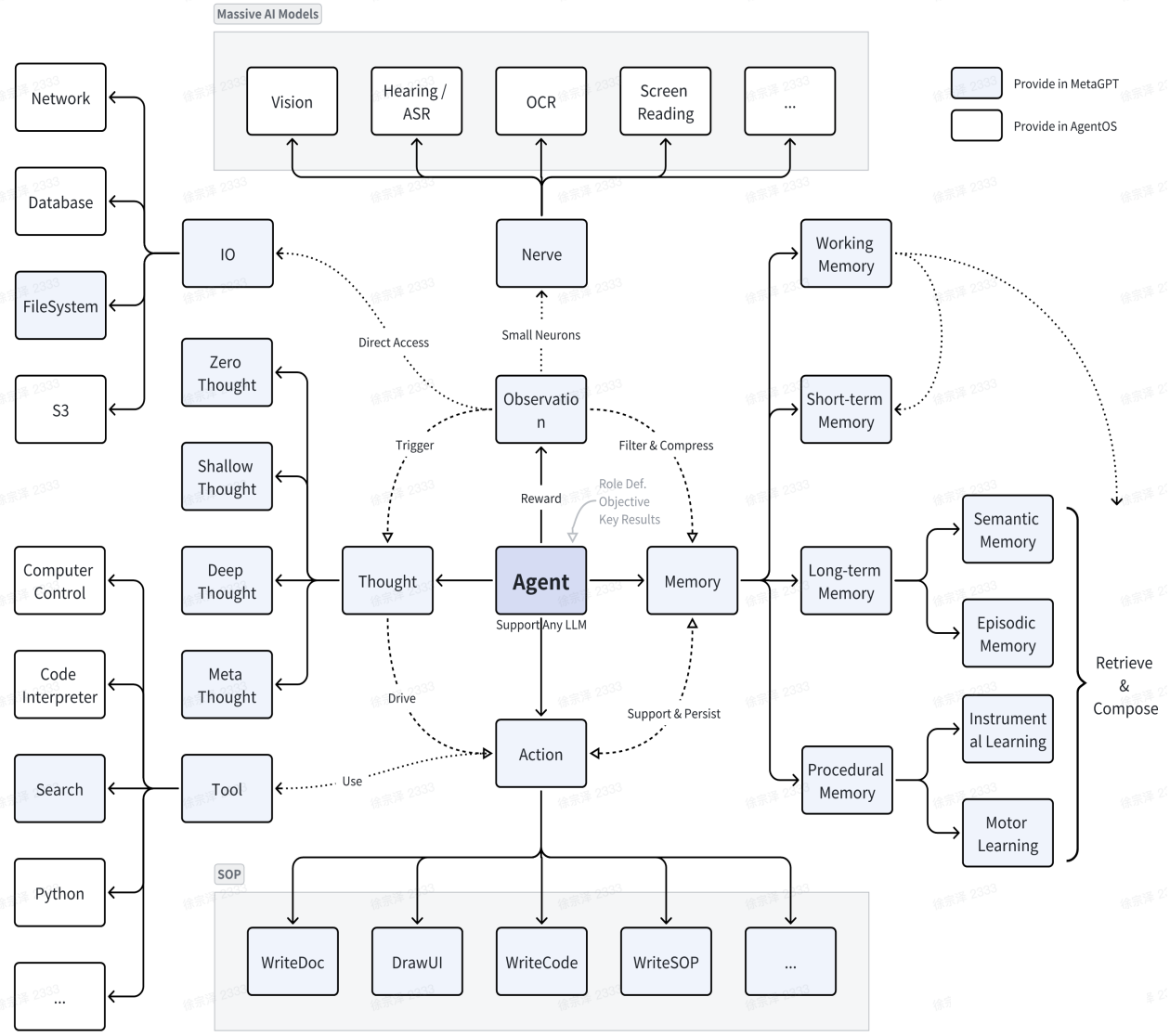
Sirui Hong, Mingchen Zhuge*, (equal contributions)*
Jonathan Chen, Xiawu Zheng, Yuheng Cheng, Ceyao Zhang, Jinlin Wang, Zili Wang,
Steven Ka Shing Yau, Zijuan Lin, Liyang Zhou, Chenyu Ran, Lingfeng Xiao,
Chenglin Wu†, Jürgen Schmidhuber (email: alexanderwu@deepwisdom.ai)

1. Introduction



What is an agent?

What is the multi-agents?



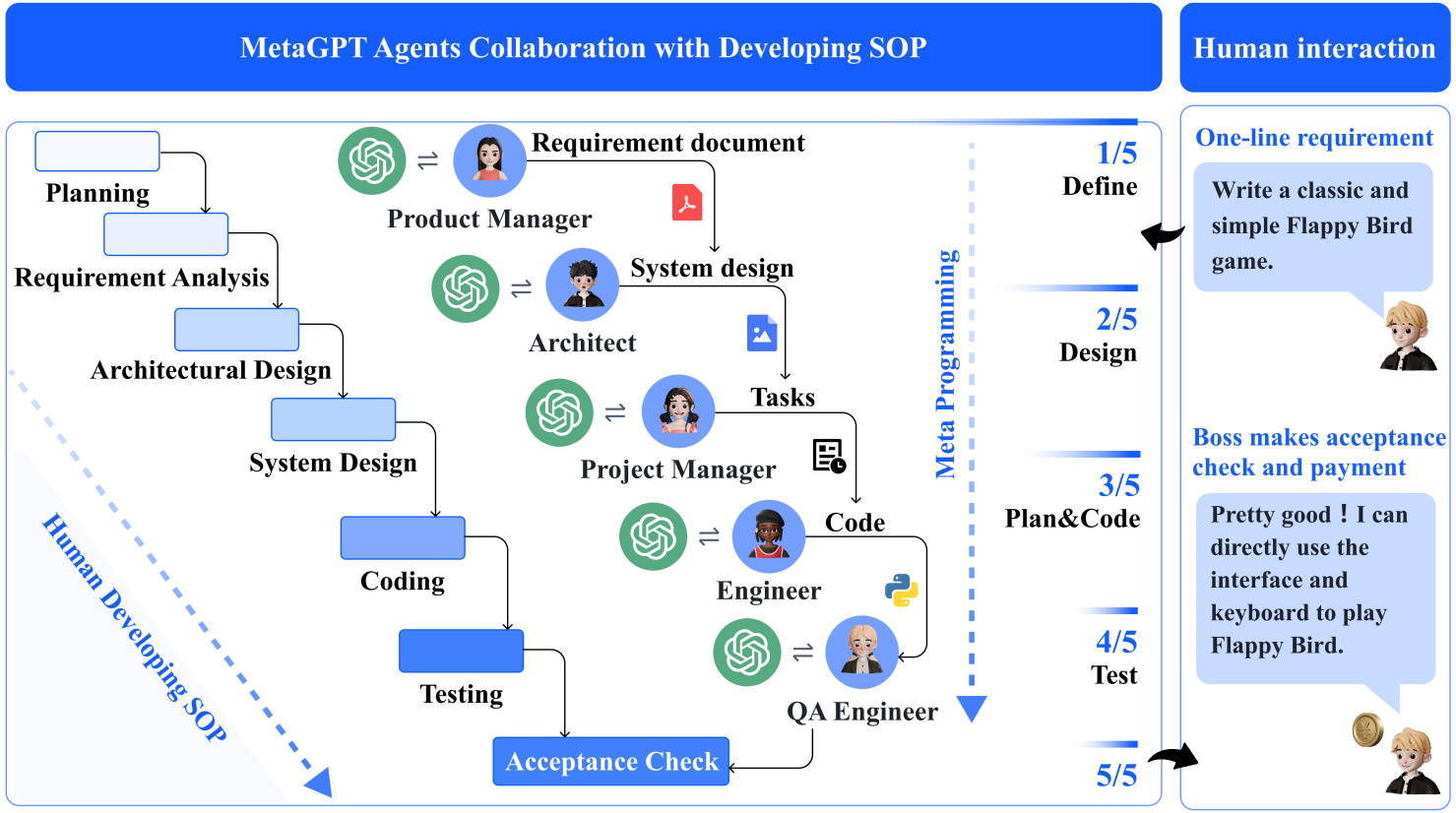
2. Challenges

Hallucinations and Inconsistency,

especially generations with long context.

3. Standard Operating Procedure (SOP)

4. SOP in MetaGPT

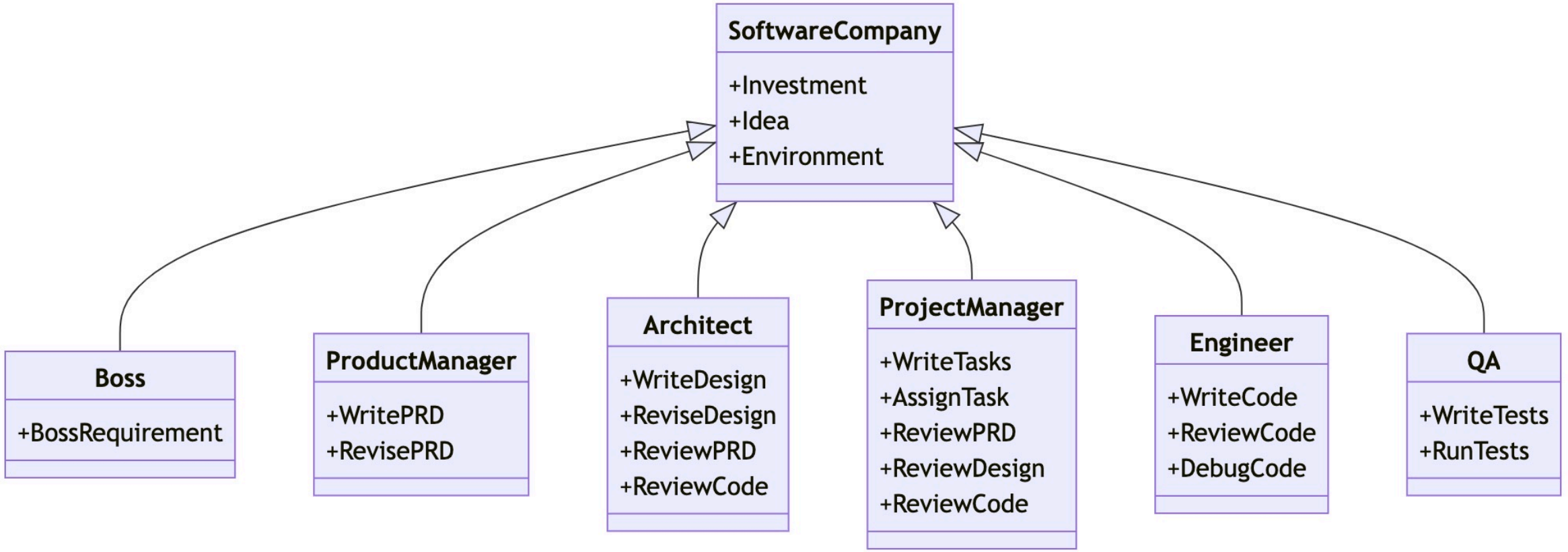


Inspired by this, SOP has been implemented in MetaGPT to improve collaboration. Each agent, from Product Managers to QA Engineers, plays a specific role, contributing distinct elements to the project. This approach allows MetaGPT to deconstruct complex tasks into simpler actions, promoting a smooth and collaborative workflow across all stages in the development.

5. Use Case

```
"""Run a startup. Be a boss."""
from metagpt.roles import ProductManager, Architect, ProjectManager, Engineer, QaEngineer
from metagpt.team import Team

company = Team()
company.hire([ProductManager(), Architect(), ProjectManager()])

if implement or code_review:
    company.hire([Engineer(n_borg=5, use_code_review=code_review)])

if run_tests:
    company.hire([QaEngineer])

company.invest(investment)
company.run_project(idea, project_name=project_name)
asyncio.run(company.run(n_round=n_round))
```

6. Agent Collaboration in Software Development (a)



7. Agent Collaboration in Software Development (b)

Specialization of Roles

❖ Role Playing Mechanism

- Each agent in MetaGPT is designed with a specific role and set of responsibilities, allowing for a division of labor that mirrors real-world software development teams.
- Additionally, each agent is initialized with specific context and skills, such as web search, diagram design, etc.

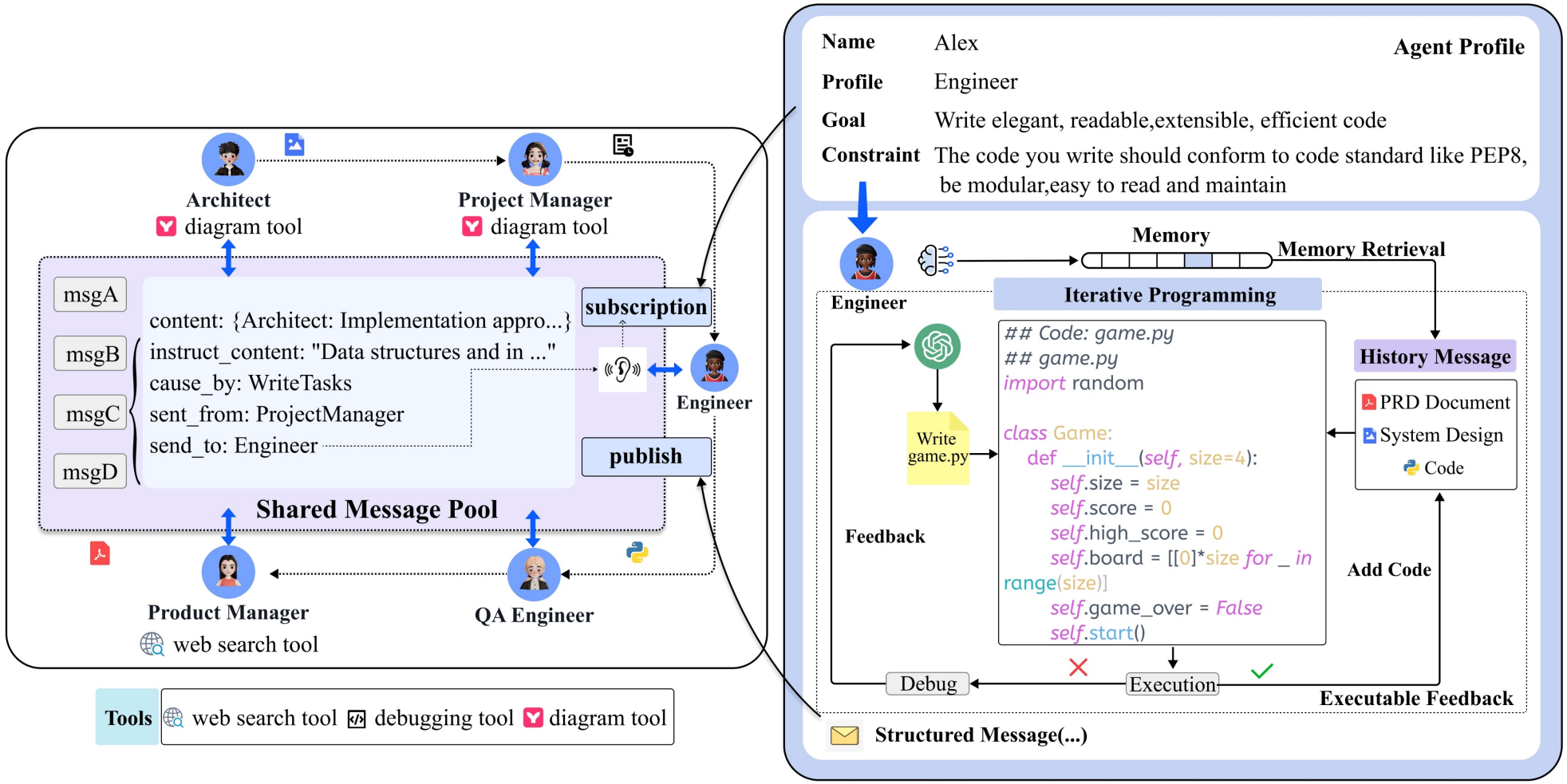
❖ React-Style Mechanism

- Each agent follows react-style behaviors. We extend their observations by providing environment feedback, and they adhere to the think-act-react procedure.
- Each agent has both independent and shared memory, enabling them to be efficient and reliable in task completion.

Workflow across Agents

- Agents collaborate by sharing messages in the same workspace, which either directly trigger actions or actively identify an upcoming task.

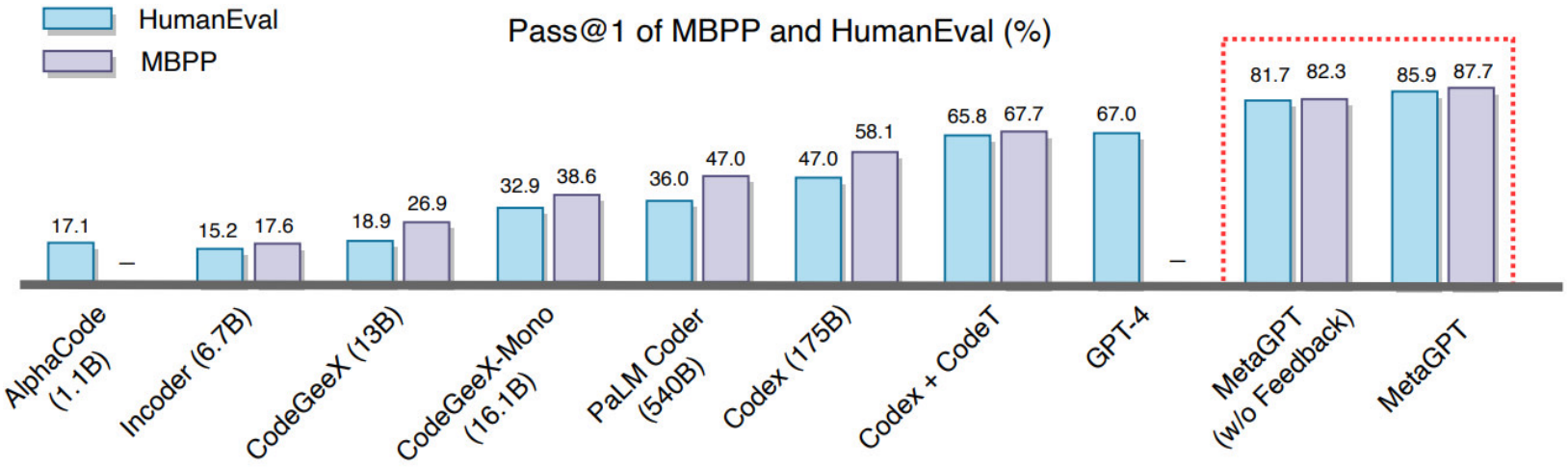
8. Agent Collaboration in Software Development (c)



(1) Structured Communication Interfaces (2) Publish-Subscribe Mechanism

(3) Executable Feedback (4) Iterative Programming

9. Experiments (a)



MetaGPT achieves **85.9%** and **87.7%** in HumanEval and MBPP respectively

MetaGPT achieves a **28.2%** relative improvement on HumanEval compared to GPT-4. Executable feedback led to a **4.2%** and **5.4%** improvement on HumanEval and MBPP respectively.

MetaGPT takes only **503** seconds to finish a task on average.

MetaGPT demonstrates high productivity in code, which needs only **124.3** tokens for one line of code.

MetaGPT achieves a high executability score of **3.75,** very close to 4 (flawless).

Table 1: The statistical analysis on SoftwareDev.

Statistical Index	ChatDev	MetaGPT w/o Feedback	MetaGPT
(A) Executability	2.25	3.67	**3.75**
(B) Cost#1: Running Times (s)	762	**503**	541
(B) Cost#2: Token Usage	**19,292**	24,613	31,255
(C) Code Statistic#1: Code Files	1.9	4.6	**5.1**
(C) Code Statistic#2: Lines of Code per File	40.8	42.3	**49.3**
(C) Code Statistic#3: Total Code Lines	77.5	194.6	**251.4**
(D) Productivity	248.9	126.5	**124.3**
(E) Human Revision Cost	2.5	2.25	**0.83**

10. Experiments (b)

Table 2: **Comparison of capabilities for MetaGPT and other approaches.** '✓' indicates the presence of a specific feature in the corresponding framework, '✗' its absence.

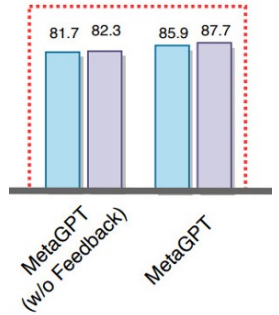
Framework Capabiliy	AutoGPT	LangChain	AgentVerse	ChatDev	MetaGPT
PRD generation	✗	✗	✗	✗	✓
Tenical design genenration	✗	✗	✗	✗	✓
API interface generation	✗	✗	✗	✗	✓
Code generation	✓	✓	✓	✓	✓
Precompilation execution	✗	✗	✗	✗	✓
Role-based task management	✗	✗	✗	✓	✓
Code review	✗	✗	✓	✓	✓

Table 3: **Ablation study on roles.** '#' denotes 'The number of', 'Product' denotes 'Product manager', and 'Project' denotes 'Project manager'. '✓' indicates the addition of a specific role. 'Revisions' refers to 'Human Revision Cost'.

Engineer	Product	Architect	Project	#Agents	#Lines	Expense	Revisions	Executability
✓	✗	✗	✗	1	83.0	**$ 0.915**	10	1.0
✓	✓	✗	✗	2	112.0	$ 1.059	6.5	2.0
✓	✓	✓	✗	3	143.0	$ 1.204	4.0	2.5
✓	✓	✗	✓	3	205.0	$ 1.251	3.5	2.0
✓	✓	✓	✓	**4**	**191.0**	$ 1.385	**2.5**	**4.0**

Table 4: **Executability comparison.** The executability scores are on a grading system ranging from '1' to '4'. A score of '1' signifies complete failure, '2' denotes executable code, '3' represents largely satisfying expected workflow, and '4' indicates a perfect match with expectations.

Task	AutoGPT	LangChain	AgentVerse	ChatDev	MetaGPT
Flappy bird	1	1	1	2	**3**
Tank battle game	1	1	1	2	**4**
2048 game	1	1	1	1	**4**
Snake game	1	1	1	3	**4**
Brick breaker game	1	1	1	1	**4**
Excel data process	1	1	1	**4**	**4**
CRUD manage	1	1	1	2	**4**
Average score	1.0	1.0	1.0	2.1	**3.9**



ABLATION STUDY

The Effectiveness of Roles

- ✓ Adding roles like product manager, architect and project manager consistently improves executability and reduce the labor of revisions.

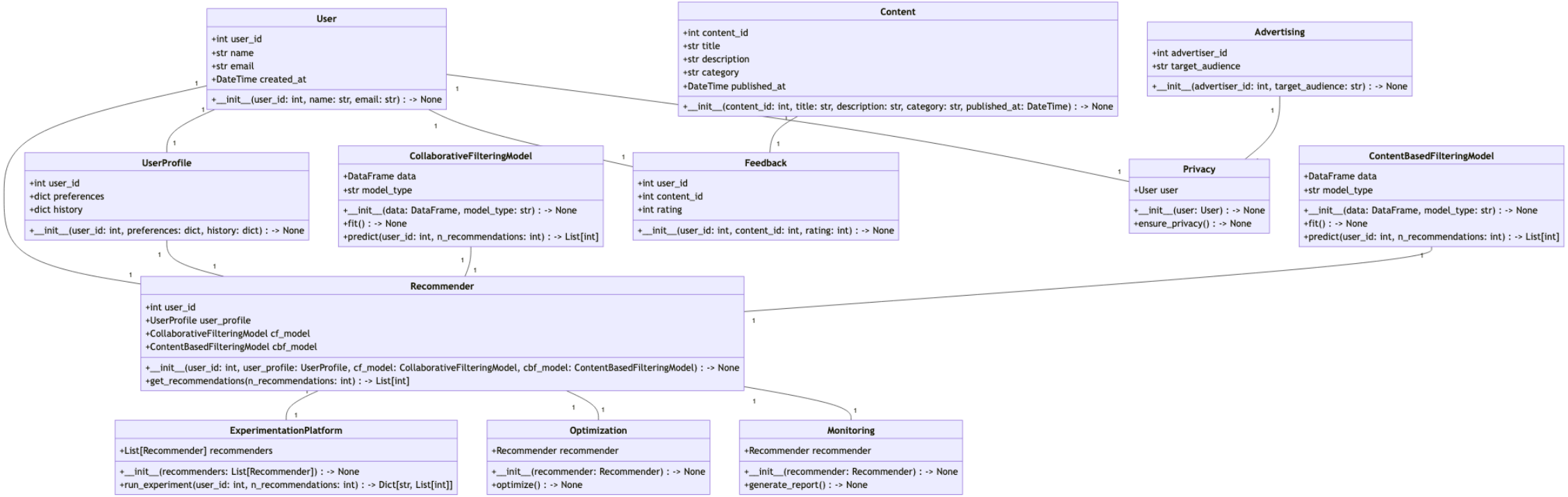
The Effectiveness of Executable Feedback Mechanism

- ✓ The executable feedback of MetaGPT leads to a significant improvement of 4.2% and 5.4% in Pass @1 on HumanEval and MBPP.
- ✓ The feedback mechanism improves functionality and executability, increasing the score from 3.67 to 3.75, while reducing the cost of human revisions significantly, from 2.25 to 0.83.

10. Demos: Architect Design by MetaGPT

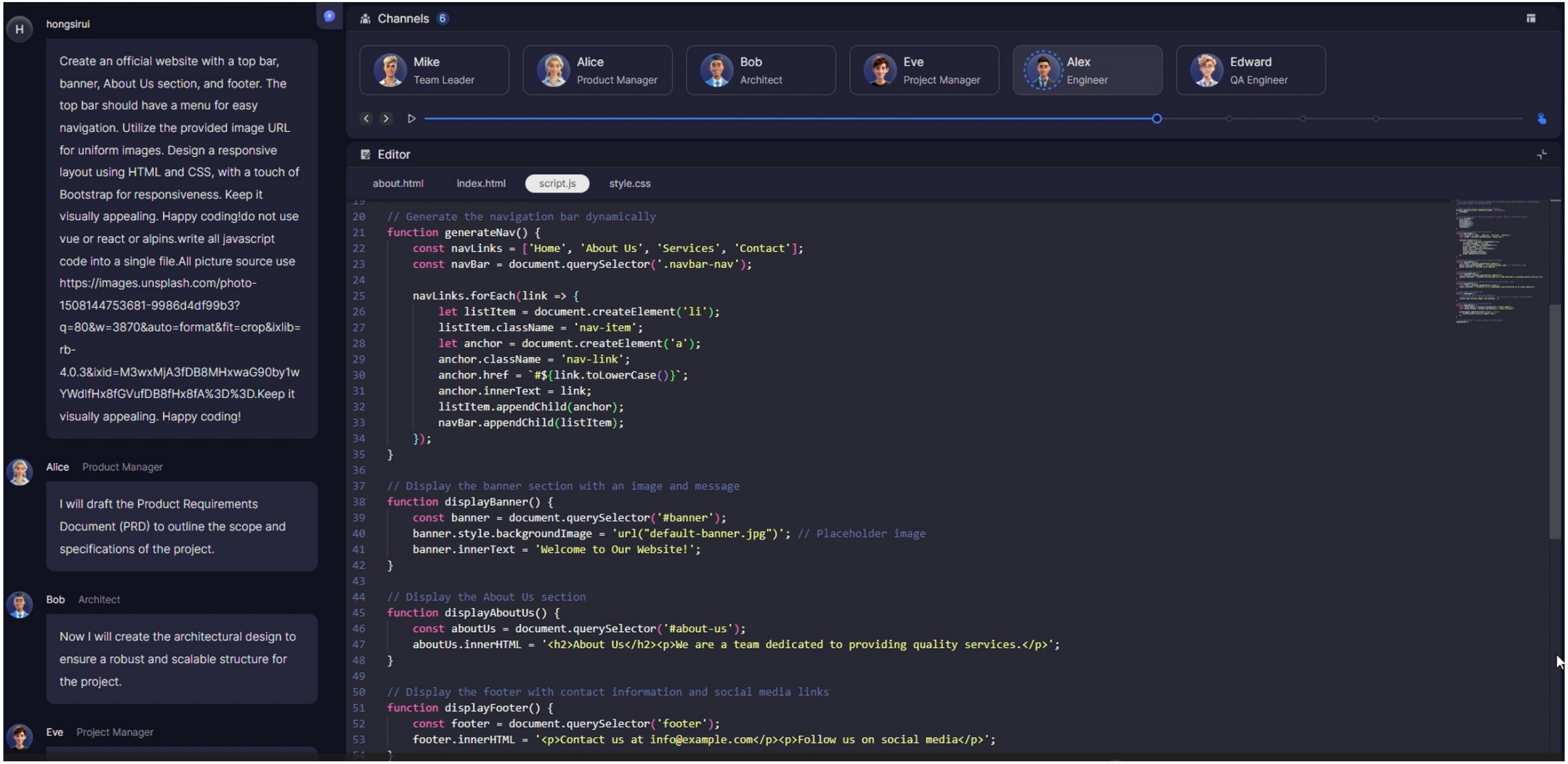
Examples (fully generated by GPT-4)

For example, if you type `metagpt "Design a RecSys like Toutiao"`, you would get many outputs, one of them is data & api design

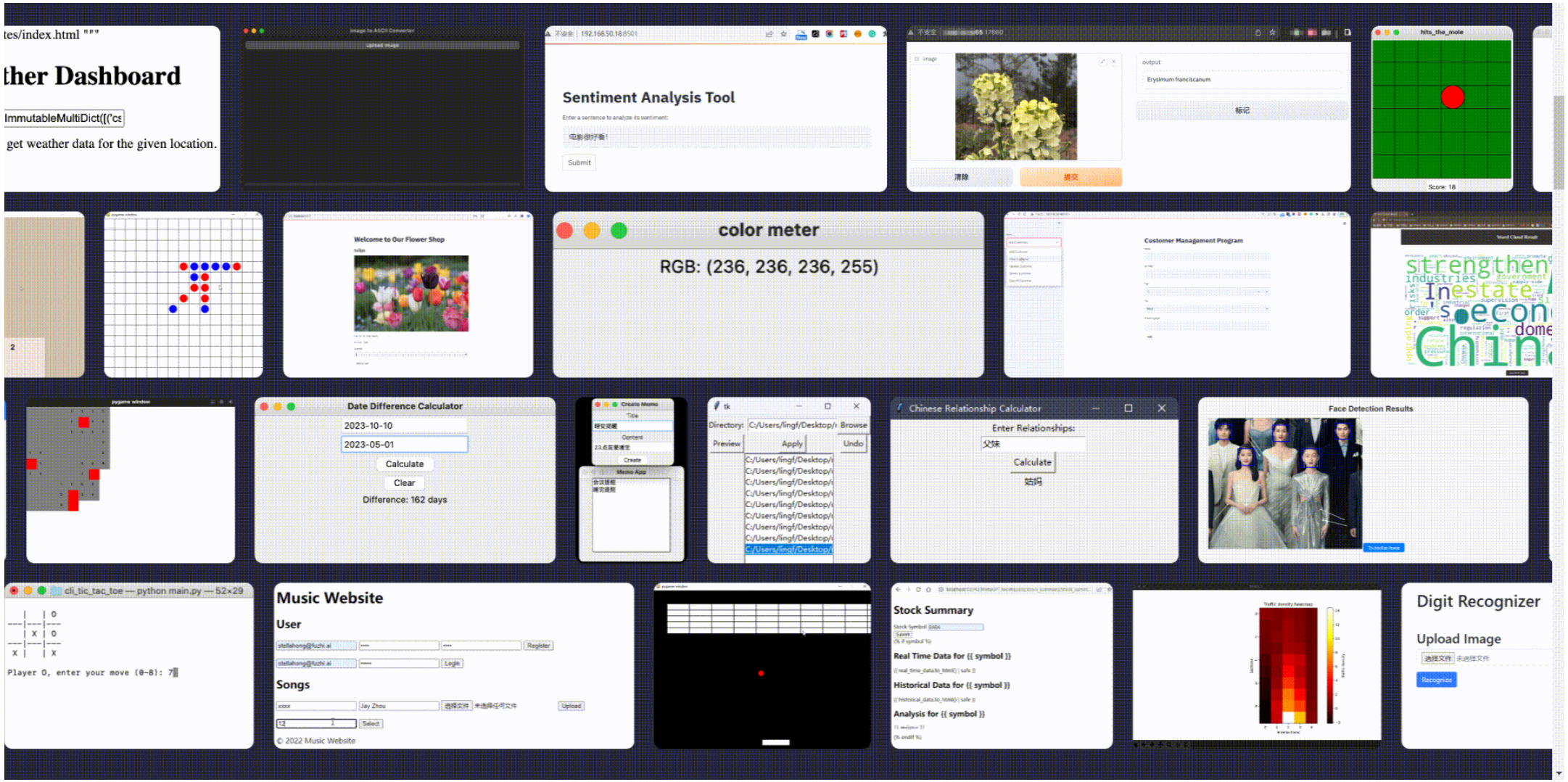


It costs approximately **$0.2** (in GPT-4 API fees) to generate one example with analysis and design, and around **$2.0** for a full project.

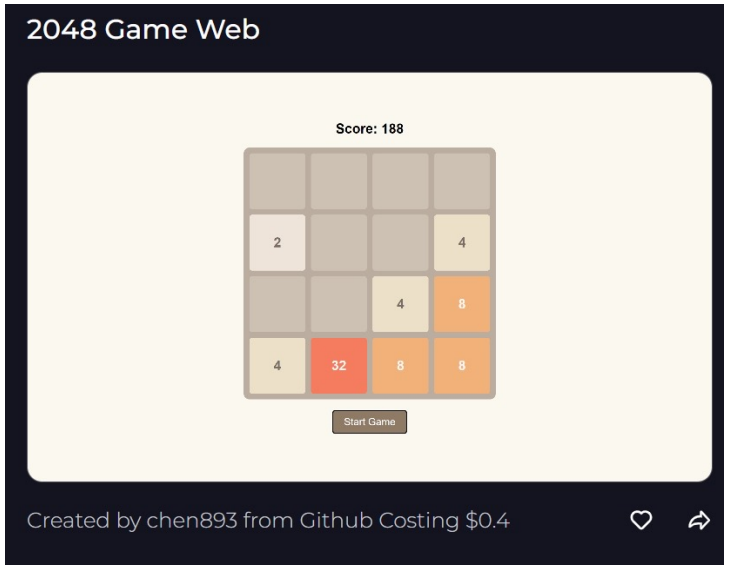
11. Demos: Development Procedures in MetaGPT

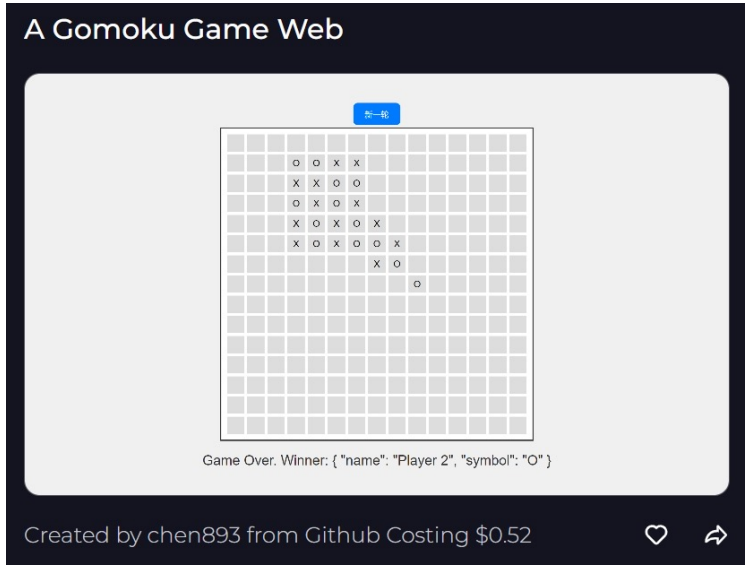


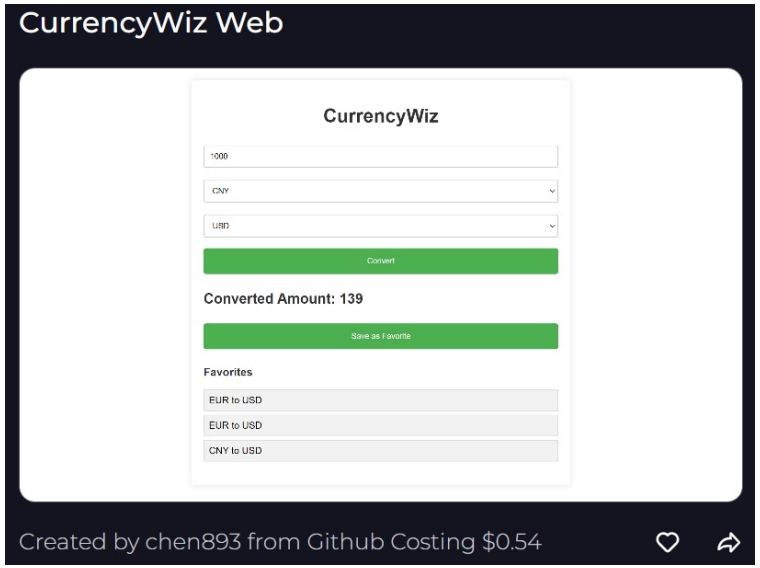
12. Demos: Software generated by MetaGPT

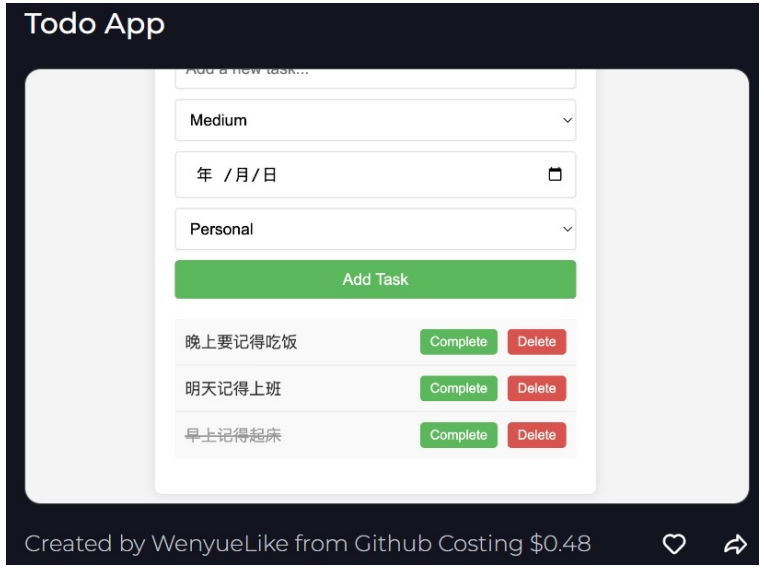


13. Demos: Executions of Generated Software









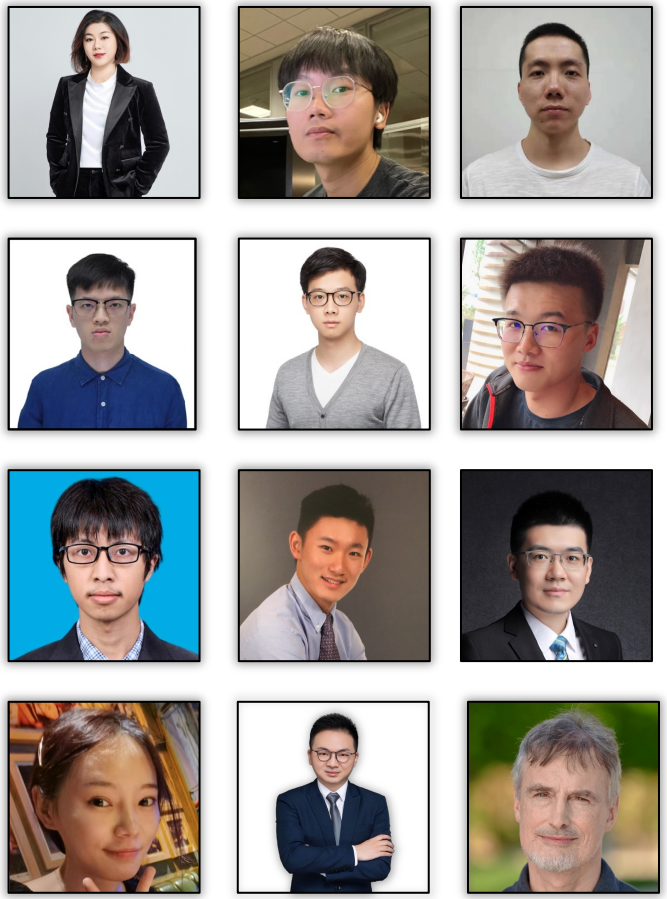
14. Future Directions

Generate more complex software

Understand and interpret data

Recursive (self) improvement

Automatic agent orchestration



DeepWisdom

جامعة الملك عبدالله للعلوم والتقنية
King Abdullah University of Science and Technology

Penn
University of Pennsylvania

Nanjing University 1902

Universitas Amoiensis

Berkeley
University of California

香港中文大學（深圳）
The Chinese University of Hong Kong, Shenzhen

THANKS!

Poster: Halle B #131

Sirui Hong, Mingchen Zhuge*, (equal contributions)*
Jonathan Chen, Xiawu Zheng, Yuheng Cheng, Ceyao Zhang, Jinlin Wang, Zili Wang,
Steven Ka Shing Yau, Zijuan Lin, Liyang Zhou, Chenyu Ran, Lingfeng Xiao,
Chenglin Wu†, Jürgen Schmidhuber (email: alexanderwu@deepwisdom.ai)