Empowering In-House Development with Cursor: Letting AI Write the Boilerplate so Developers Can Focus on Core Logic

Management Information Technology and System Office

Supported by LM

1. Background

As the University' s digital transformation continues to advance, the in-house development team (four developers in total) is responsible for a wide range of development tasks, including but not limited to:

- **Customized business system development;**
- **Script and component development for no-code / low-code platforms;**
- **Script development for the HiAgent intelligent agent platform;**
- **Python script development for RPA workflows.**

In the face of high-frequency and fragmented requirements, the team has long encountered three major challenges:

1. **A high volume of requests with limited manpower**, leading to serious backlogs of routine tasks;
2. **Small-scale requirements often involve heterogeneous technology stacks** (e.g., temporary use of Node.js, Python, or Java), which differ from the team’ s primary stack (React + Java), resulting in high development and debugging costs;
3. **Extension capabilities of no-code and low-code platforms rely heavily on scripting**, but writing repetitive logic is inefficient and time-consuming.

Traditional development models struggle to balance speed, flexibility, and maintainability under these constraints.

2. Solutions

Starting in March 2025, the development team introduced **Cursor** as a core productivity tool, establishing a collaborative development model of “AI-assisted generation + human-led control”:

- **Accelerating routine development**: Using natural language to rapidly generate repetitive code such as CRUD APIs and frontend forms, significantly reducing time spent on boilerplate coding;
- **Supporting small projects with heterogeneous technology stacks**: For non-primary languages or temporary tasks (e.g., writing Python

scripts to batch-process image files), Cursor can directly generate complete, runnable examples, eliminating time spent searching documentation and configuring environments;

- **Context-aware intelligent completion**: Cursor understands the existing project structure and automatically adapts generated code to current naming conventions, dependency libraries, and error-handling patterns, thereby reducing integration costs.

At present, all four developers use Cursor on demand. It requires no changes to existing development workflows and can be adopted on a plug-and-play basis.

3. Outcomes and Benefits

- **Overall development efficiency increased by approximately 30–40%**: Delivery cycles for routine functions have been significantly shortened;
- **Heterogeneous technology stack tasks are no longer a burden**: Multiple temporary projects have been completed successfully (e.g., Python scripts that read Excel files and batch-generate images). In the development of the *llead* mini program, existing *lbss* project code was adapted—Cursor quickly helped developers understand

the original logic and make precise modifications, greatly reducing onboarding and iteration time;

- **Greater flexibility in extending low-code platforms**: Plugin extension requests from administrative colleagues can be rapidly prototyped and embedded using Cursor-generated validation scripts, significantly improving response speed;
- **Improved developer focus**: Team members are able to devote more time to high-value work such as system architecture optimization, cross-system integration, and complex business logic;
- **Extremely low cost**: Only four individual subscriptions are required (approximately RMB 7,500 per year in total), with no additional infrastructure investment.

4. Replicability and Promotion Value

- Applicable to all departments involved in software development;
- Particularly well-suited to IT scenarios characterized by "small but diverse requirements, heterogeneous technology stacks, and limited manpower";
- Does not rely on any specific platform or private model—developers only need basic programming skills to get started quickly;

- Can serve as a pilot example for enhancing the University's "AI + Development" capabilities and provide evidence for future institution-wide tool selection.

5. Next Steps

- **Further exploration of advanced Cursor capabilities**: Investigate its support for the Model Context Protocol (MCP) and explore upgrading Cursor from a "code generator" to an "intelligent development agent," such as automatically calling internal APIs, querying database context, and generating test cases;

Horizontal comparison with other AI editors: Continue evaluating tools such as Cdoex, Trae, and Continue (open source), focusing on aspects such as multi-language support, local model compatibility, and Chinese language understanding, in order to identify alternatives or complementary solutions better suited to the University's technical ecosystem.