SINGAPORE UNIVERSITY OF TECHNOLOGY AND DESIGN

[RE]IMAGINE EDUCATION

A weekly Design·AI newsletter

MERGEGATE: PROVE IT BEFORE YOU MERGE IT

Charles Foo, Nisal Deelaka Jinadasa, Ang Kar Kin, Lee Kai Boo, Lasch Nicolas Andreas G, and Sagar Pratap Singh, Graduate Students
In Conversation with Ravi Singaram, Graduate student


Lasch Nicolas Andreas G


Sagar Pratap Singh


Charles Foo


Lee Kai Boo


Ang Kar Kin


Nisal Deelaka Jinadasa

As AI-assisted coding becomes increasingly common, software engineers can generate and deploy code faster than ever—but speed does not always guarantee understanding. Designed by Charles Foo, Nisal Deelaka Jinadasa, Ang Kar Kin, Lee Kai Boo, Lasch Nicolas Andreas G, and Sagar Pratap Singh during SUTD's **01.921 Innovating with Design & AI for Leaders** course, MergeGate is a conceptual checkpoint that asks developers to demonstrate that they understand, review, test, and can take responsibility for AI-generated code before it is merged into the main branch. Guided by the principle "Prove it before you merge it," the concept reframes responsible AI use as a matter of critical engagement, accountability, and human judgment. How might MergeGate—or a similar checkpoint—be applied in your organisation, classroom, or development team?

Editor of Newsletter:
Dr. Nachamma Sockalingam, Office of Strategic Planning

Contact us at nachamma@sutd.edu.sg to share your Design·AI stories

AI can accelerate coding through “vibe coding.” But is that advisable?

With the surge of AI, it is all too common for “software engineers” to completely rely on AI for writing code - known as “vibe coding”.

“Vibe coding” is a term coined by computer scientist Andrej Karpathy in February 2025, to describe software creation by prompting AI using natural language instead of actual coding.

Although writing code fast is a valuable skill, doing so using AI without understanding and refining the code may lead to operational issues that they may not be able to troubleshoot. Indeed, in today’s age, an ideal software engineer would be someone who uses AI as a smart tool rather than a workhorse which he/she completely relies on. They should not take what AI generates at face value, but instead, critique and verify every logic block, thereby still retaining deep knowledge of the code. While blind vibe coding may not get the user far, responsible AI-assisted coding could be the way forward.

The question is, “How can we encourage more responsible AI-assisted coding?”

Two coders. Two mindsets. One question: Do you truly understand the code you are about to merge?

So then, how can one ensure that the programmer truly understands AI-generated code?

That's where MergeGate comes in!

Designed by Charles Foo, Nisal Deelaka Jinadasa, Ang Kar Kin, Lee Kai Boo, Lasch Nicolas Andreas G, and Sagar Pratap Singh as part of SUTD's 01.921 Innovating with Design & AI for Leaders course, MergeGate is a conceptual system that explores how developers might prove they understand AI-generated code before it is merged into the main project.

Before the merge is approved, MergeGate checks whether the developer has meaningfully reviewed, tested, and understood the AI-generated code. It does this by analysing indicators such as how much of the code came from AI, how much was revised by the developer, the level of test coverage, the quality of review comments, and the time spent checking the work.

The developer may also be asked questions about the code, such as why a particular approach was chosen, what could go wrong, and how the solution was tested. The goal is not to stop developers from using AI. It is to ensure that they can explain and take responsibility for the code before it becomes part of the main system.

In simple terms, MergeGate asks: **Did the developer merely accept the AI's code, or did they understand, question, and improve it?**

MergeGate Dashboard Simulation

How do developers use MergeGate, and what does it do?

Developers use MergeGate when they are ready to submit AI-assisted code for merging into the main project. The system connects to the coding workflow, reviews how the code was generated and refined, and then displays the results on a physical dashboard.

Its main features include:

Live Pull Request Health: Shows how much code was generated by AI, how much was reviewed by the developer, the level of test coverage, time spent, and the quality of review comments. It can trigger a warning when the code appears to have been accepted with too little human engagement.

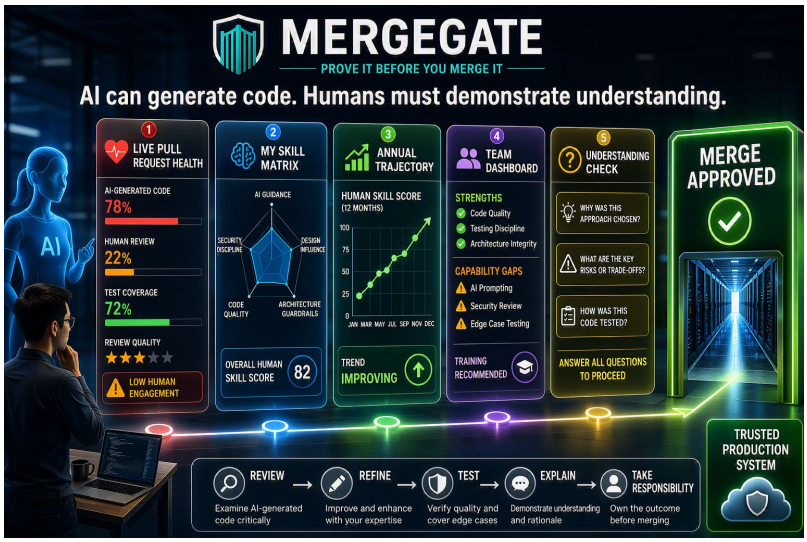
My Skill Matrix: Assesses how well the developer guides the AI, improves its suggestions, protects the system's architecture, maintains code quality, and checks for security risks.

Annual Trajectory: Tracks the developer's aggregated human skill score over 12 months, showing whether their ability to work responsibly with AI is improving.

Team Dashboard: Gives an overview of the team's strengths, capability gaps, and areas that may require training or closer review.

Understanding Check: Before the code is merged, the developer answers questions about the code's logic, design choices, risks, and testing. This helps confirm that they understand and can take responsibility for the final submission.

In practice, MergeGate acts like a final checkpoint: developers can still use AI to write code, but they must show that they have reviewed, tested, and understood it before the merge is approved.



MergeGate Features

Can MergeGate be adapted for younger learners or for self-directed learning?

MergeGate can be adapted for younger learners, with gamification and reward systems.

Instead of checking professional software code, a classroom version could ask students to explain how they used AI, what they changed, and why they accepted or rejected its suggestions.

Before receiving a reward, learners might complete a short challenge, answer questions about their work, or show that they tested the AI's response rather than simply copying it. In this way, the same "prove it before you merge it" principle could help students build habits of reflection, verification, and responsible AI use from an early age.

For self-directed learners, MergeGate could act as a reflective checkpoint before they move on to the next task. Learners could explain what they understood, how they used AI, and how they verified its suggestions. This encourages them to take ownership of their learning rather than simply accepting AI-generated answers.

MergeGate may be only a concept, but the question it raises applies to every AI user: what does responsible use look like when AI can produce convincing work in seconds? Whether we use AI to write, analyse, design, research, or code, we should not mistake polished output for reliable understanding. AI can accelerate the work, but responsibility for the final outcome remains with us. Perhaps every user needs a personal "MergeGate"—a moment to pause, question, verify, refine, and ask: Do I understand this well enough to trust it, use it, and take responsibility for it?