

<my_code>

Works!

(I have no idea why)

Prompt → Code → Works! → Why? → Trust? → Engineering!



Hlasujeme o nejvhodnější emotikon



- My code **works**! I have no idea why 🍺
- My code **works**! I have no idea why 😱
- My code **works**! I have no idea why 🤔

Hlasujeme o nejvhodnější emotikon

👍💪🤔🧐 ... 😏💩

- My code **works**! I have no idea why 🍺
- My code **works**! I have no idea why 😱
- My code **works**! I have no idea why 🤔



Proč? 🤔

My code works! 🎉

- JÁ jsem kód napsal.
- JÁ svému kódu rozumím.
- JÁ jej dokážu vysvětlit.

vs

- Kód byl napsán AI. Funguje, ale ... 🤔

Ještě nedávno byla tahle věta programátorský vtip:
*„programátor něco napsal, ono to začalo fungovat, ale nevěděl
přesně proč.“* Dnes dostává úplně nový význam: *„kód totiž
možná **vůbec** nenapsal.“*

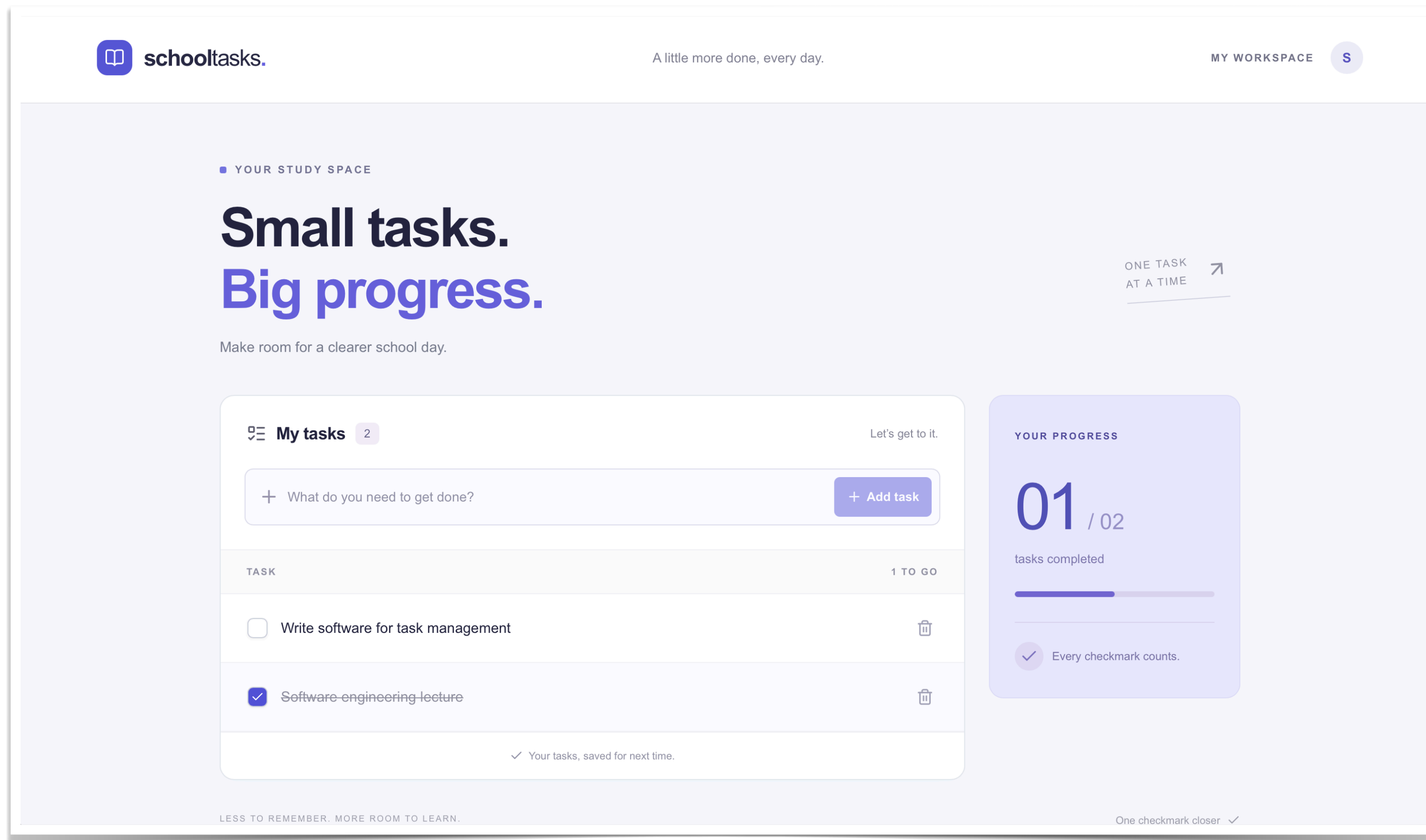
Tehdy a nyní

AI mění roli programátora !

Tehdy	Nyní
Člověk -> Kód -> Počítač	Člověk -> AI -> Kód -> Počítač
Já jsem napsal kód.	AI napsalo kód.
Já jsem kód odladil.	AI opravilo svůj kód.
Já bych měl svému kódu rozumět.	Stále tomu kódu rozumím?

Požádejme AI:

Create a simple web application for managing my school tasks. I want to add, complete and delete tasks.



- Vytvoření aplikace trvalo **10+ minut**
- Celkem **109 zdrojových a konfiguračních souborů**, celkem **1,22 MB**.
- Včetně závislostí, sestavení a pomocných souborů: **52 579 souborů**, celkem **971,19 MB**.

WOW! 🤯

It **works**! Takže máme hotovo?

- Funguje aplikace vždy jak má?
- Je aplikace dostatečně zabezpečená?
- Mohu aplikaci rozšířit případně pozměnit?
- Rozumím tomu, jak aplikace funguje?
- **Mohu aplikaci věřit?**

Prompt → Code → **Works! → Why? → **Trust?****

AI nemusí svému kódu rozumět, já ale ano!

Chat: dotaz -> odpověď



- Chat = AI poradce.
- Já se ptám, AI odpovídá. Já se rozhoduji, co udělám.
- Zeptám se, dostanu odpověď. AI mi může vysvětlit algoritmus, najít chybu nebo napsat funkci. Ale pořád jsem to já, kdo vezme výsledek a něco s ním udělá.
- **AI navrhuje, já konám.**

Může AI konat?

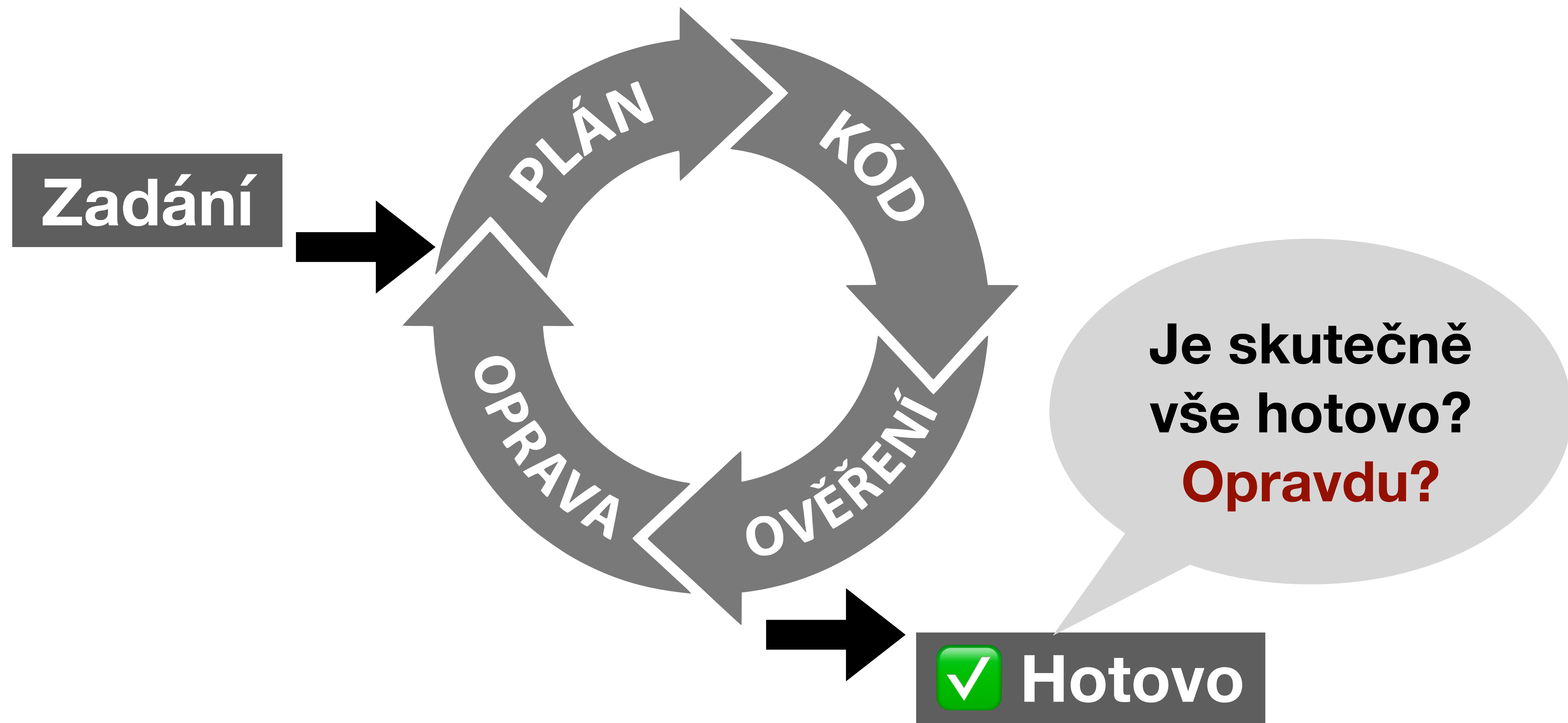
Vítejte ve světě AI agentů 🤖

💬 Komunikuje 📖 Čte soubory ✍️ Píše kód ▶ Spouští programy 🧪 Testuje 🛠️ Opravuje

💬 Chat	🤖 Agent
Odpovídá	Koná
Navhuje kód	Píše kód
Vysvětluje chyby	Opravuje chyby
Navrhuje příkazy	Vykonává příkazy
Čeká na tazatele	Pracuje samostatně

Smyčka agenta

Od zadání k výslednému řešení



Kdo zkontroloval kontrolora? 🤔

Kód byl napsán AI -> otestován AI -> opraven AI -> ✅ AI

- Splnění testů ≠ správný software
- 10/10 splněný testů! 🎉
ALE
 - Otestovali jsme všechny funkce?
 - Nezapomněli jsme na něco?
 - Je specifikace požadavků správná?
- Testování ukáže pouze to, že software všechny testy splnil. To ale nedokazuje, že **BYL vytvořen správný software.**

Clear and Present Danger

AI dokáže vytvořit chyby velmi rychle, ve velkém množství a velmi přesvědčivě. 🤨

- AI je ale **výjimečný** inženýrský nástroj.
- Problém není v používání AI, ale v **nekritické používání AI**.
- **Mění se role programátorů:**
 - **Dříve:** psaní kódu (programování)
 - **Dnes:** definuj -> navrhni -> vygeneruj -> ověř -> vylepši **svůj kód**
- **Less typing, more thinking!**

Co znamená být dobrým vývojářem

Prompt → Code → Works! → Why? → Trust? → **Engineering!**

- **SOFTWARE ENGINEERING**

- Specifikace požadavků
- Analýza a návrh, architektura systému
- Kódování
- Správa verzí
- Verifikace a validace
- Bezpečnost
- Udržitelnost softwarového systému



Is there really a professional programmer who writes code without planning?

AI dělá kódování jednodušším, ale nečiní softwarové inženýrství nepotřebným.

AI Paradox

Čím snadnější je napsat kód pomocí AI, tím důležitější je použít inženýrský přístup!

- **Vibecoding vs Softwarové inženýrství**
 - Vibecoding: **Prompt** → **Generate** → **Run** → **Works!** → **Ship it** 🚀
 - SWE: **Requirements** → **Design** → **Generate** → **Review** → **Test** → **Verify!** → **Deploy**
- **Kdy a co použít závisí na míře rizika**
 - 🎮 Počítačové hry?
 - 🏦 Bankovní systém?
 - 🚗 Autonomní řízení?
 - 🏥 Medicínské aplikace?

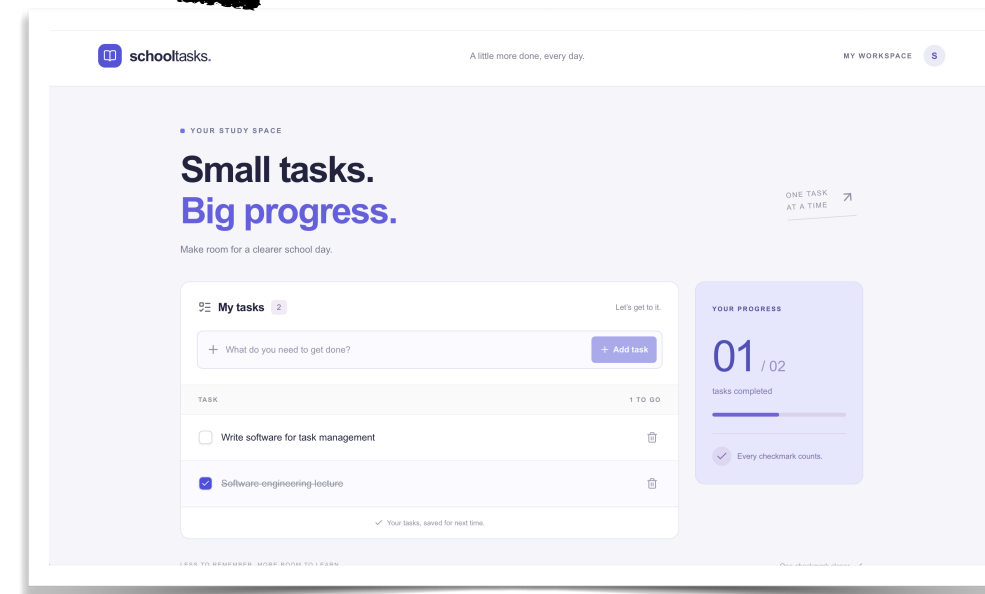
Poletíte letadlem, jehož řídicí software byl vygenerován AI, kterému výrobce úplně nerozumí, ale prošel všemi testy?

Vibecoding - správa úkolů

Prompt → Generate → Run → **Works!** → Ship it 🚀

Create a simple web application for managing my school tasks. I want to add, complete and delete tasks.

Generate



Trust?

Works!

Run



Trust?

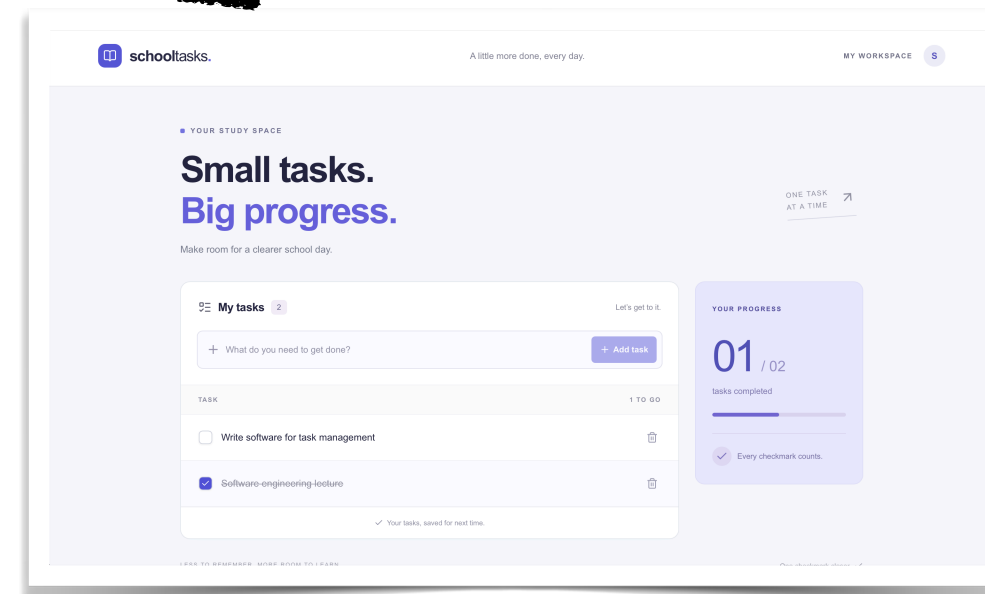
Trust?

Vibecoding - správa úkolů

Prompt → Generate → Run → **Works!** → Ship it 🚀

Create a simple web application for managing my school tasks. I want to add, complete and delete tasks.

Generate



Trust?

Works!

Run



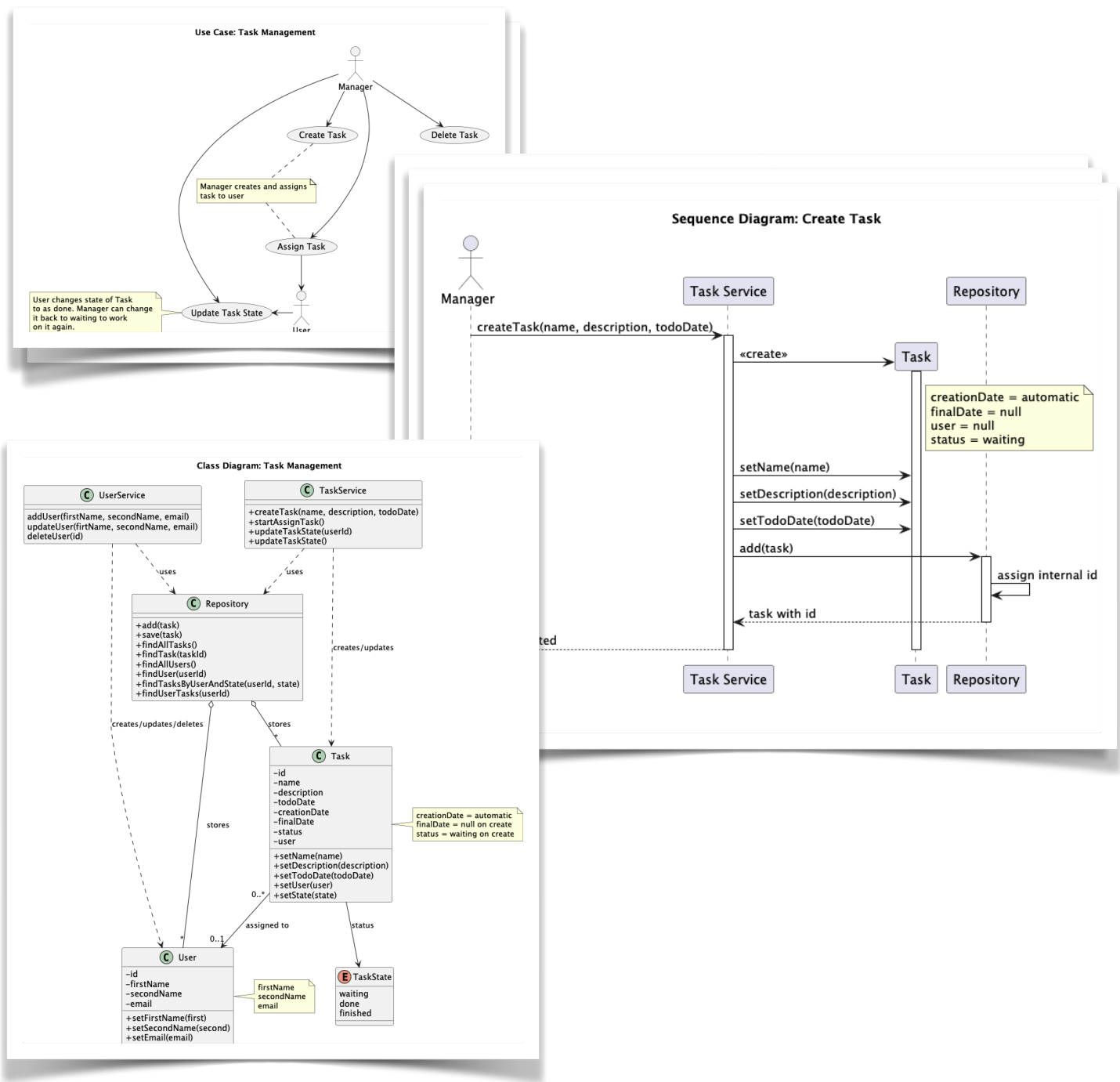
Trust?

Trust?

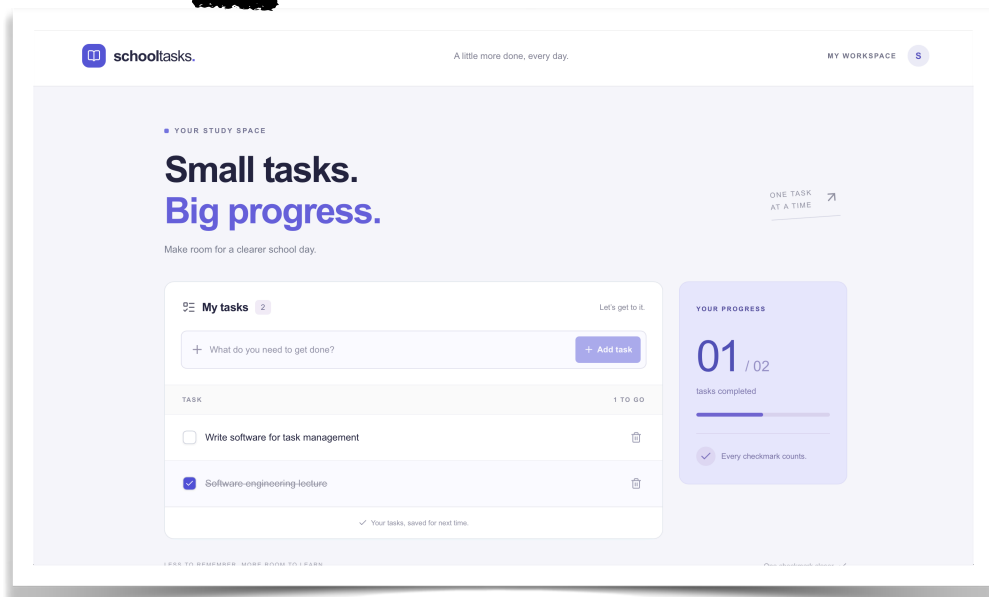


Inženýrský přístup

Requirements → Design → Generate → Review → Test → **Verify!** → Deploy



Generate



Deploy

Trust?

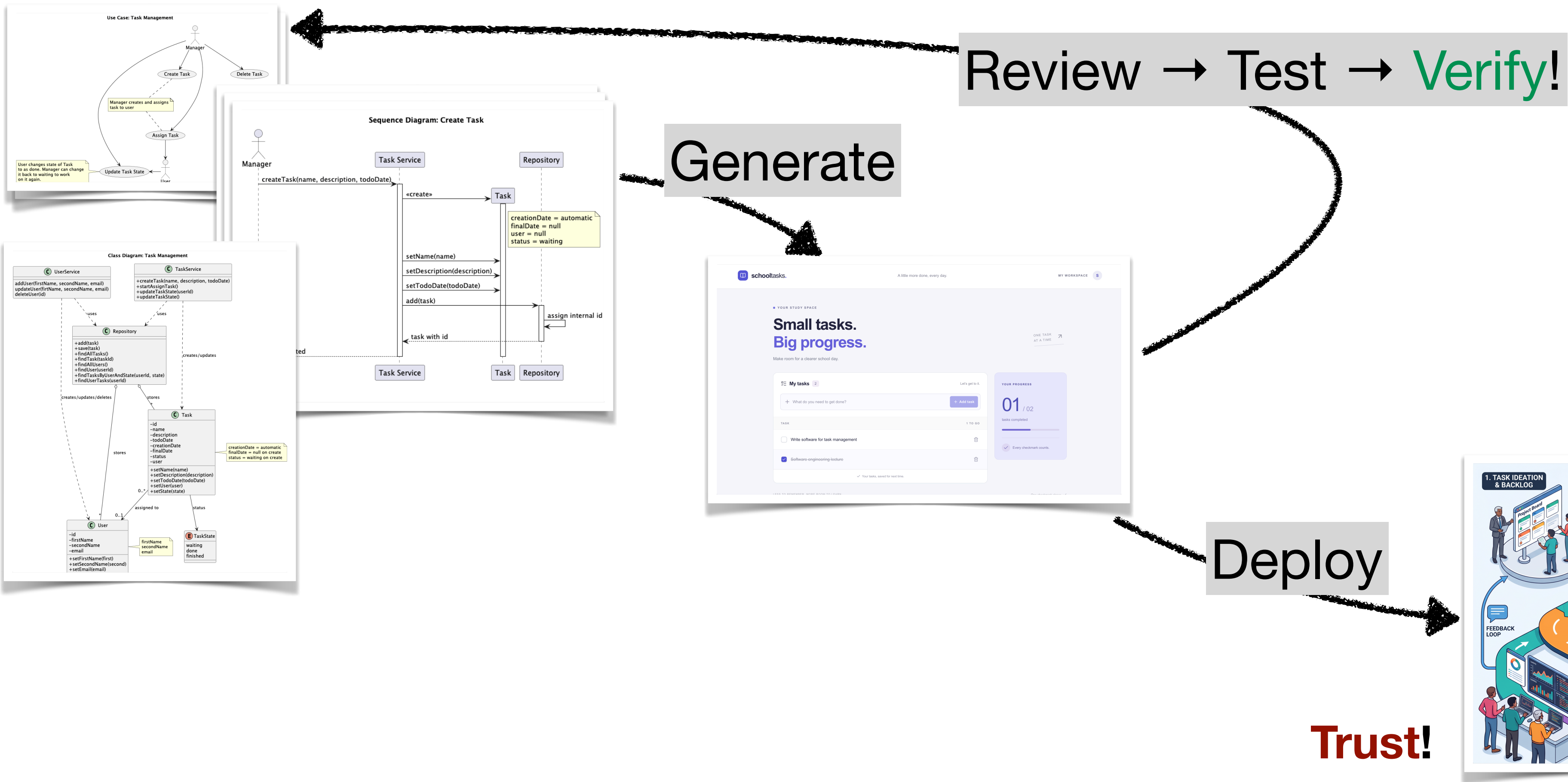
Trust?

Trust?



Inženýrský přístup

Requirements → Design → Generate → Review → Test → **Verify!** → Deploy



Trust!

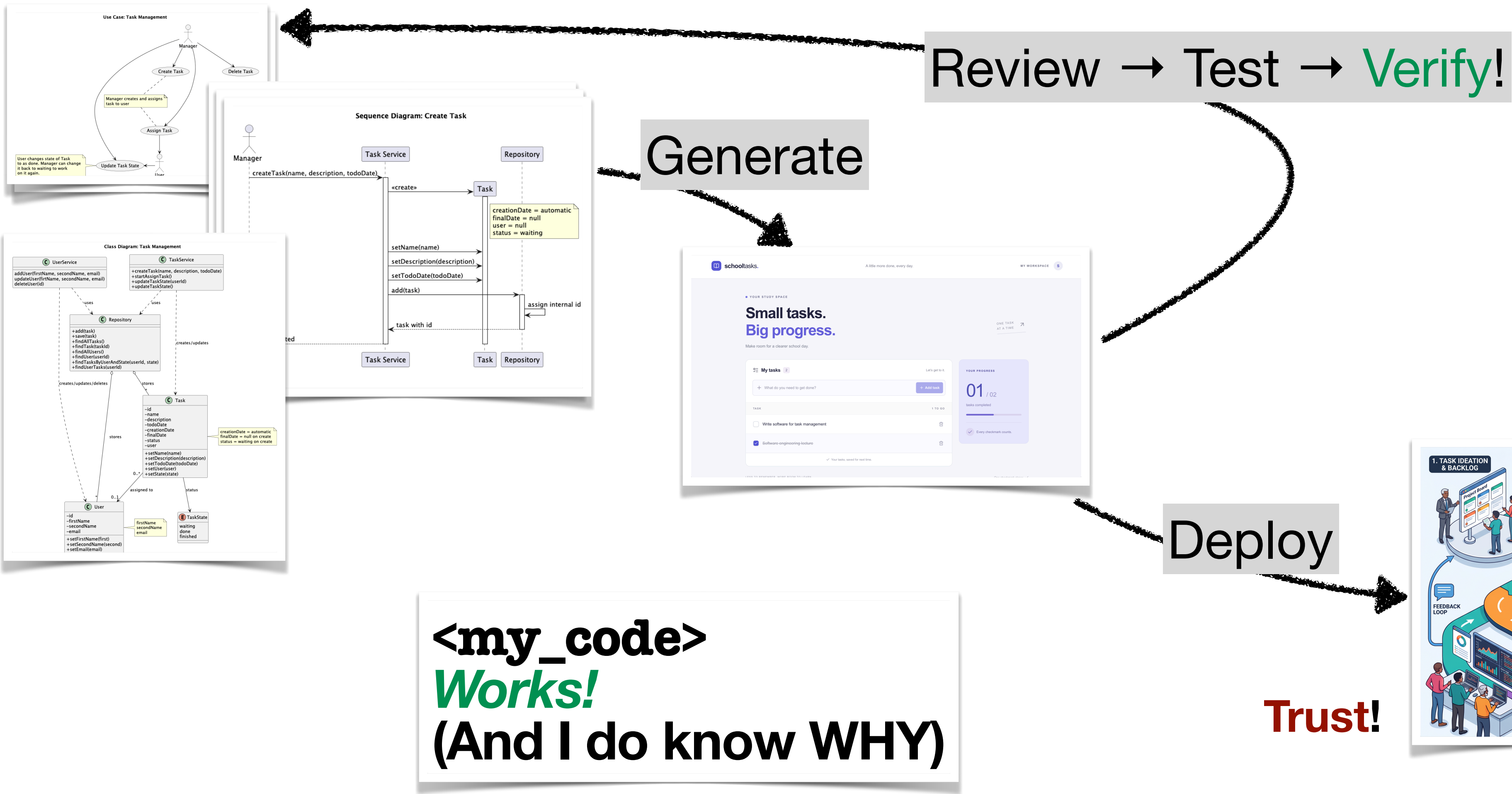
Trust!

Trust!



Inženýrský přístup

Requirements → Design → Generate → Review → Test → **Verify!** → Deploy



Trust!

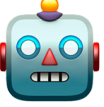
Trust!

Trust!



Závěrem

Co si z toho všeho odnést?

-  **Používej AI.**
Je to mimořádně silný nástroj. Nauč se s ním pracovat.
-  **Přemýšlej.**
Nevěř kódu jen proto, že vypadá dobře a funguje.
-  **Rozuměj.**
Měl bys umět vysvětlit, co tvůj software dělá a proč.
-  **Ověřuj.**
Testuj nejen kód, ale i to, zda řešíš správný problém.
-  **Bud' inženýr.**
Požadavky → Návrh → Implementace → Testování → Ověření

Odpovědnost zůstává na tobě!

Důležité!

To, co platí pro vývoj software,
platí i pro ostatní oblasti našeho života.

AI může navrhovat.

AI může tvořit.

AI může rozhodovat.

Ale člověk musí **rozumět, ověřovat**
a nést odpovědnost.

HUMAN IN THE LOOP!