Software Engineering in the Age of AI

Presentation by **Solomon Eshun** — AI/ML & Software Engineer

> *"The future belongs to those who build it."*

What are your expectations?

We will not become expert engineers in next 1h:30mins, but we will *see* how software engineering works…

Today Roadmap

1

Theory

What is Software Engineering & How AI is changing Software Engineering

2

Hands-on Session

Python mini-project ~30 - 40 mins

3

Wrap-up

Q&A, resources, next-steps

Theory I: Software Engineering

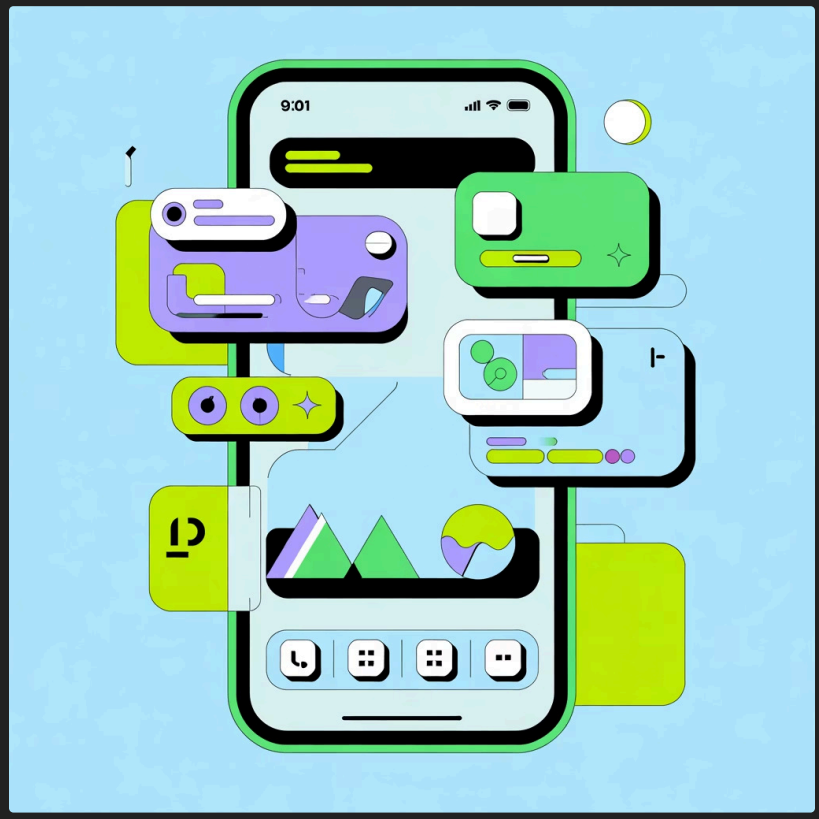
What's your favorite app, and what does it it do?

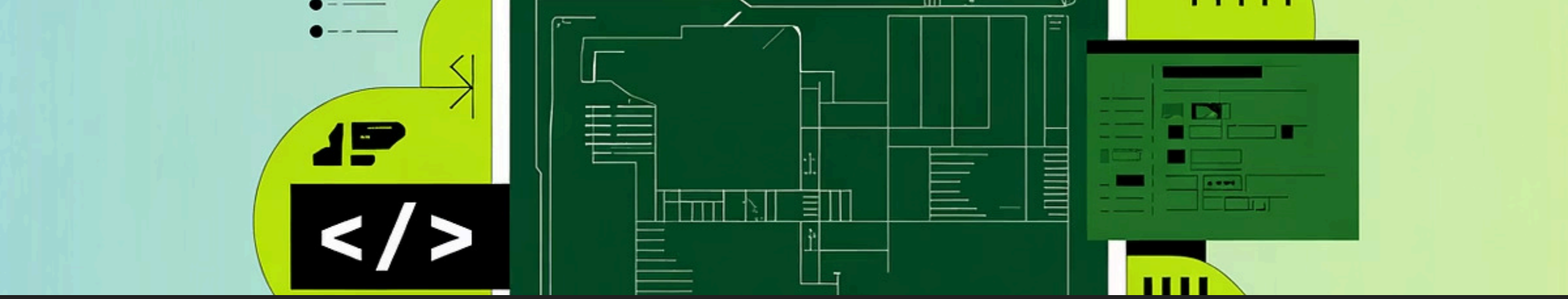
What is Software? 💻

Software is a set of instructions that tell a computer what to do. Think of it like a recipe — but instead of cookies, you're creating experiences, solving problems, and making life easier.

Software runs everything around you:

- The apps on your phone 📱
- Web apps like YouTube and Instagram
- Games that keep you entertained 🎮
- Hospital systems saving lives
- Banking apps managing money securely





What is Engineering? 🔧

Problem Solving

Finding creative solutions to real-world challenges

Science & Logic

Using systematic thinking and proven methods

Creativity

Designing innovative systems that work beautifully

Engineering is the art of solving problems using science, logic, and creativity. Engineers build bridges, design cars, create medical devices — and yes, they build the software systems that power our digital world.

So… What is Software Engineering? 🚀

Software engineering is the disciplined, systematic approach to building, testing, and maintaining reliable software systems.

Reliability

Works correctly every time

Scalability

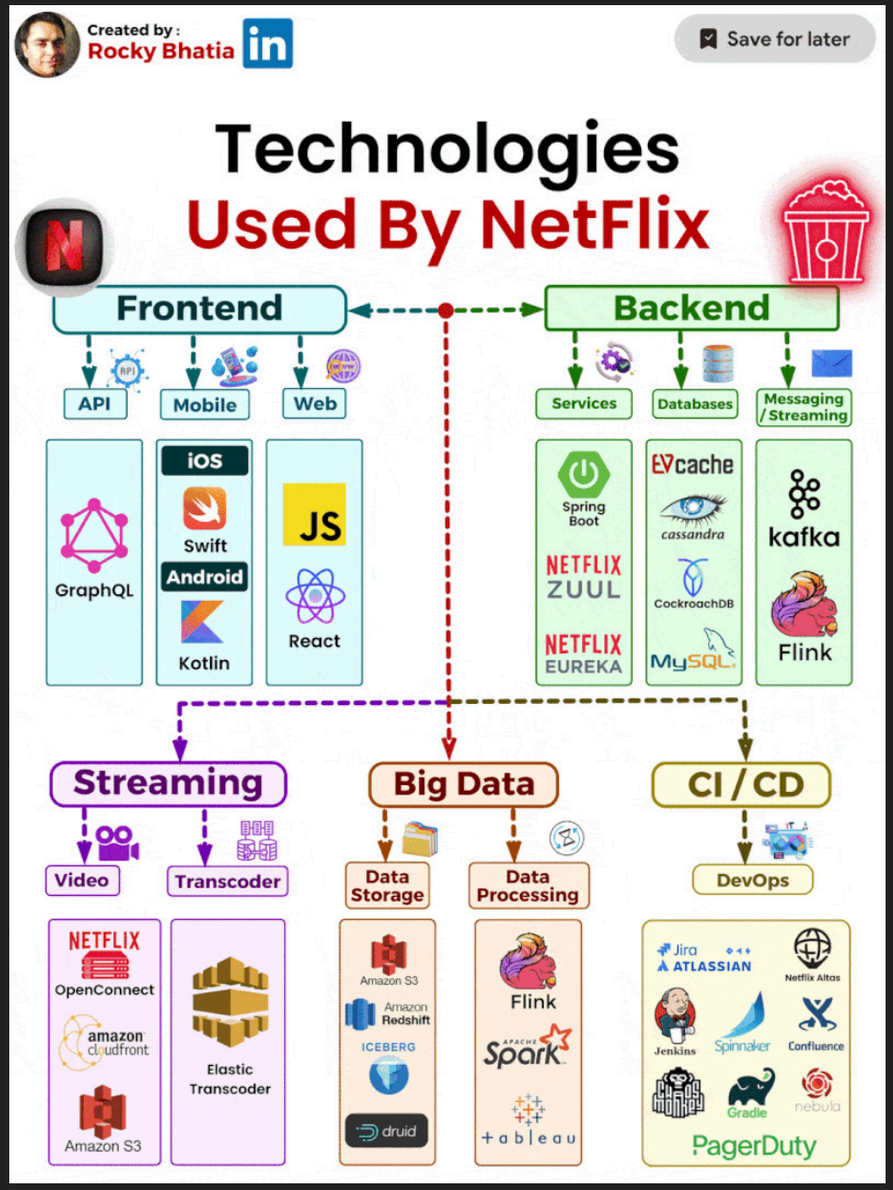
Handles growth from 10 to 10 million users

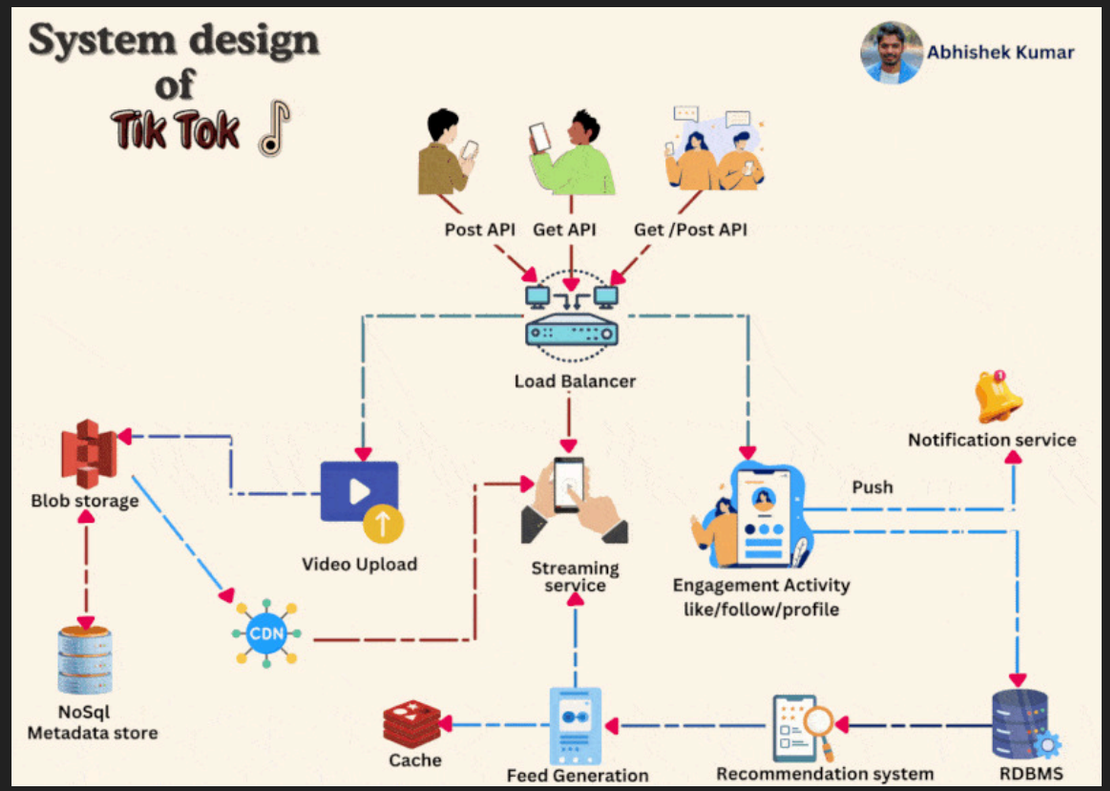
Maintainability

Easy to update and improve over time

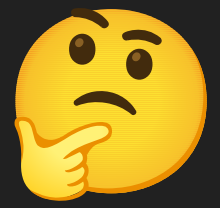
Correctness

Does exactly what it's supposed to do





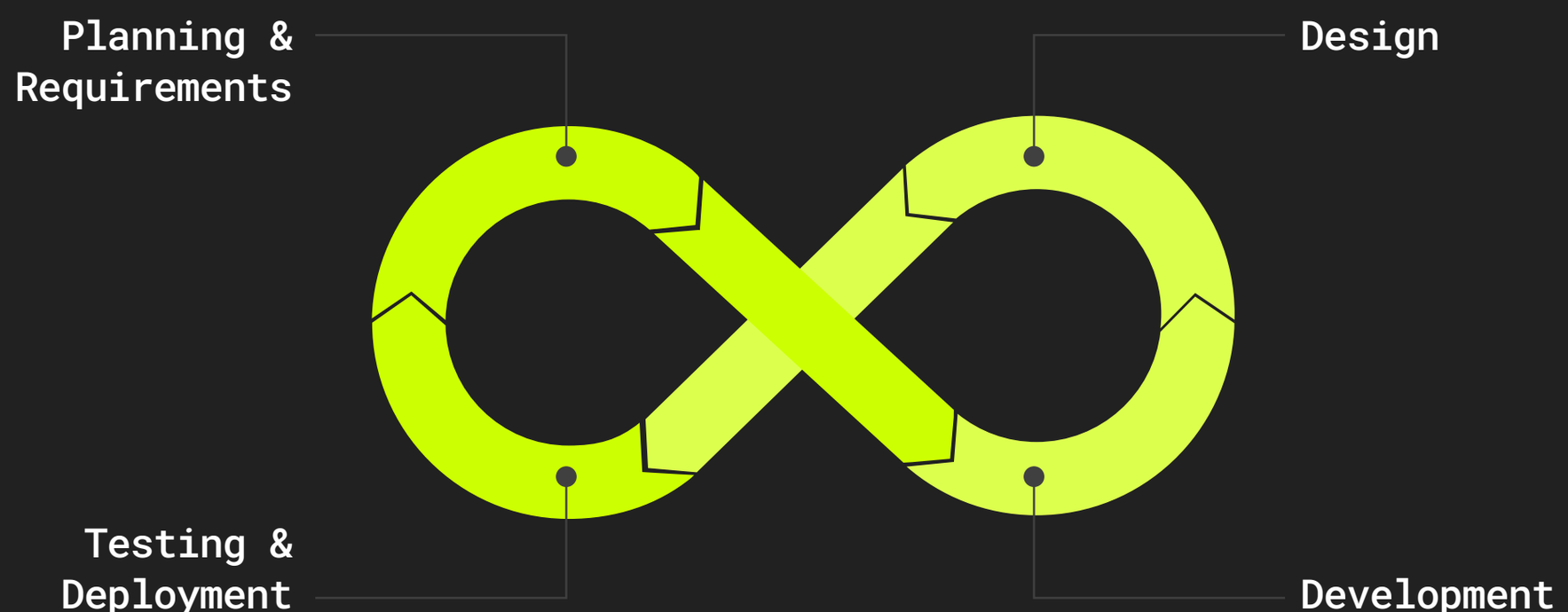
Okay… So now, *how are softwares built?*



Software Development Life Cycle (SDLC) Explained

The **Software Development Life Cycle (SDLC)** is a structured process that guides the development of high-quality software.

> “SDLC is the roadmap that guides how software moves from an idea in your head to an app in your hands.”



It provides a clear framework for building, deploying, and maintaining software, ensuring consistency, efficiency, and quality throughout the project lifecycle.

Popular SDLC Models

Waterfall

Sequential, one step after another. Ideal for simple, predictable projects where requirements are stable.

Agile

Iterative, with fast feedback and strong team collaboration. Best suited for modern startups and AI projects requiring flexibility.

Extreme Programming (XP)

Emphasizes high collaboration, pair programming, and continuous testing. When flexibility and top-notch quality are paramount.

Hybrid (Water-Agile)

Combines structured planning with agile iteration. Great for large teams or corporate projects that need both structure and adaptability.

> “Good engineers don’t just code — they engineer solutions by following structured processes.”

Beyond the Code: Deconstructing the Software Engineering Workflow

> “Software engineering isn’t just about typing code. It’s about thinking, planning, testing, and communicating.”

The 40-20-40 Rule

A classic industry guideline for effort distribution:

- **40% — Planning, Analysis, Design:** Understanding requirements, architecting solutions, and mapping out the project.
- **20% — Actual Coding:** Writing the software instructions and implementing features.
- **40% — Testing, Debugging, Deployment:** Ensuring quality, fixing errors, and getting the software into users' hands.

Meta Group Study Findings

Real-world data further illustrates this balance:

- **Coding:** Only **23%** of the total project effort.
- **The Rest:** The vast majority of time is spent on non-coding activities such as:
  - Meetings and communication
  - Analysis and requirements gathering
  - Design and architecture
  - Documentation
  - Testing and quality assurance

> “Coding is the fun part — but planning and testing are what make it engineering.”

Software Engineering workflow illustration

Generated with Gemini Nano Banana

The Building Blocks of Software Engineering 🏗️

Requirements Engineering

"What needs to be built?" — Understanding user needs and defining goals clearly.

Design

"How will we build it?" — Planning the architecture, modules, and data flow.

Implementation

"Writing the code!" — Turning designs into working software using programming languages.

Testing & QA

"Does it work correctly?" — Finding and fixing bugs before users see them.

Deployment & Maintenance

"Let's launch and keep improving!" — Releasing to users and updating continuously.

Roles in Software Engineering 👥

Building software is a team sport! Each role brings unique skills to create amazing products.

Software Developer 👩🏽‍💻

Writes code to build features and functionality

QA / Test Engineer ✅

Tests software to catch bugs and ensure quality

DevOps Engineer ⚙️

Manages deployment, servers, and automation

Systems Architect 🏗️

Designs the overall system structure and tech choices

UI/UX Designer 🎨

Creates beautiful, user-friendly interfaces

Product Manager 💡

Guides vision, priorities, and user experience

Which role do you think fits your personality?

From Idea to App – Real Example 📱

Let's Build a Simple Study Time Tracker App

01

Write Requirements

Users want to log study sessions, track total hours, and see their progress weekly

02

Design the Screens

Sketch the timer screen, history log, and stats dashboard with clean layouts

03

Code It

Write the logic using Python for backend and React for frontend

04

Test It

Check if timers work correctly, data saves properly, and calculations are accurate

05

Deploy & Share

Launch the app on the web or app store and gather user feedback

Remember: Software engineering is teamwork! Even solo projects benefit from organized thinking.

Theory II: How AI is changing Software Engineering

Software in the era of AI🤖✨

AI tools are revolutionizing how engineers work. They help us write code faster, catch bugs earlier, and optimize performance automatically.

Andrej Karpathy's vision of **"Software 3.0"**: humans write prompts and high-level instructions, while AI generates parts of the code.

Software 1.0

Manual code, line by line

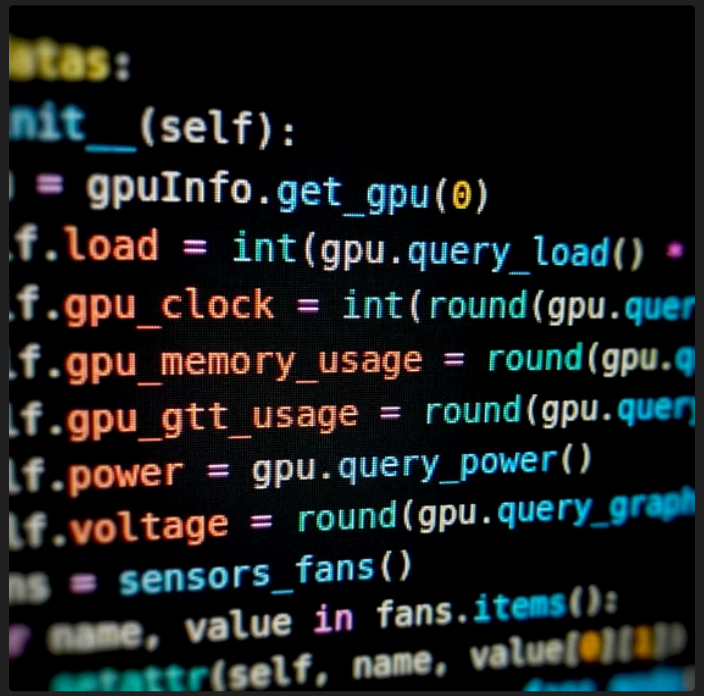
Software 2.0

ML models learn from data

Software 3.0

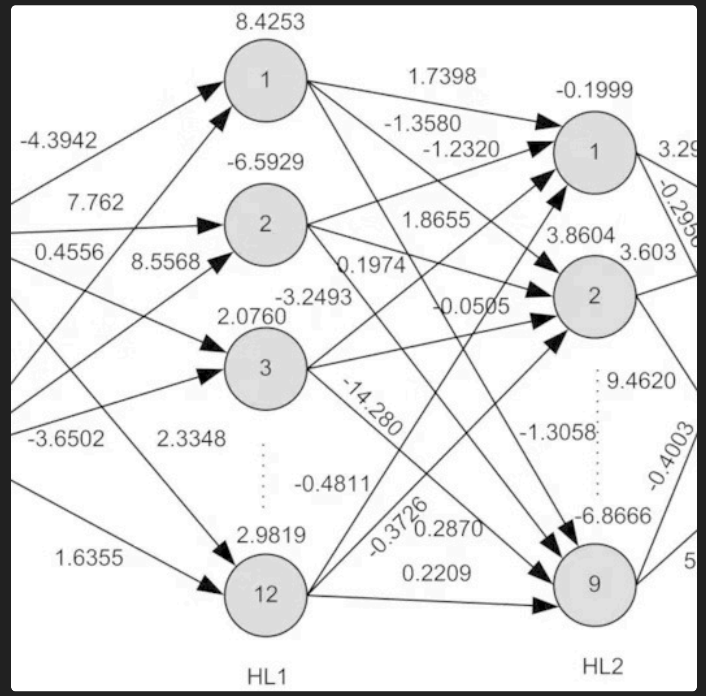
AI-assisted coding & creation

Software is Changing(Again) ~Andrej Karpathy



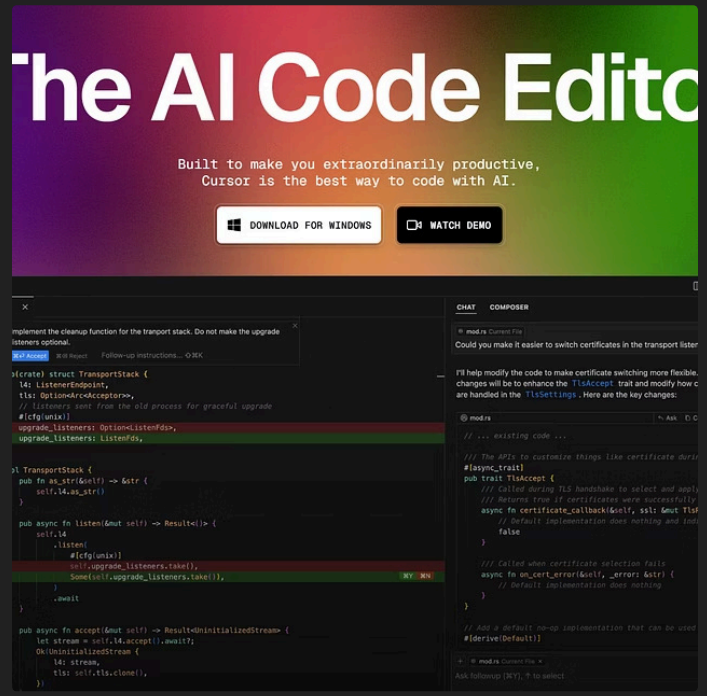
Software 1.0

Manual code



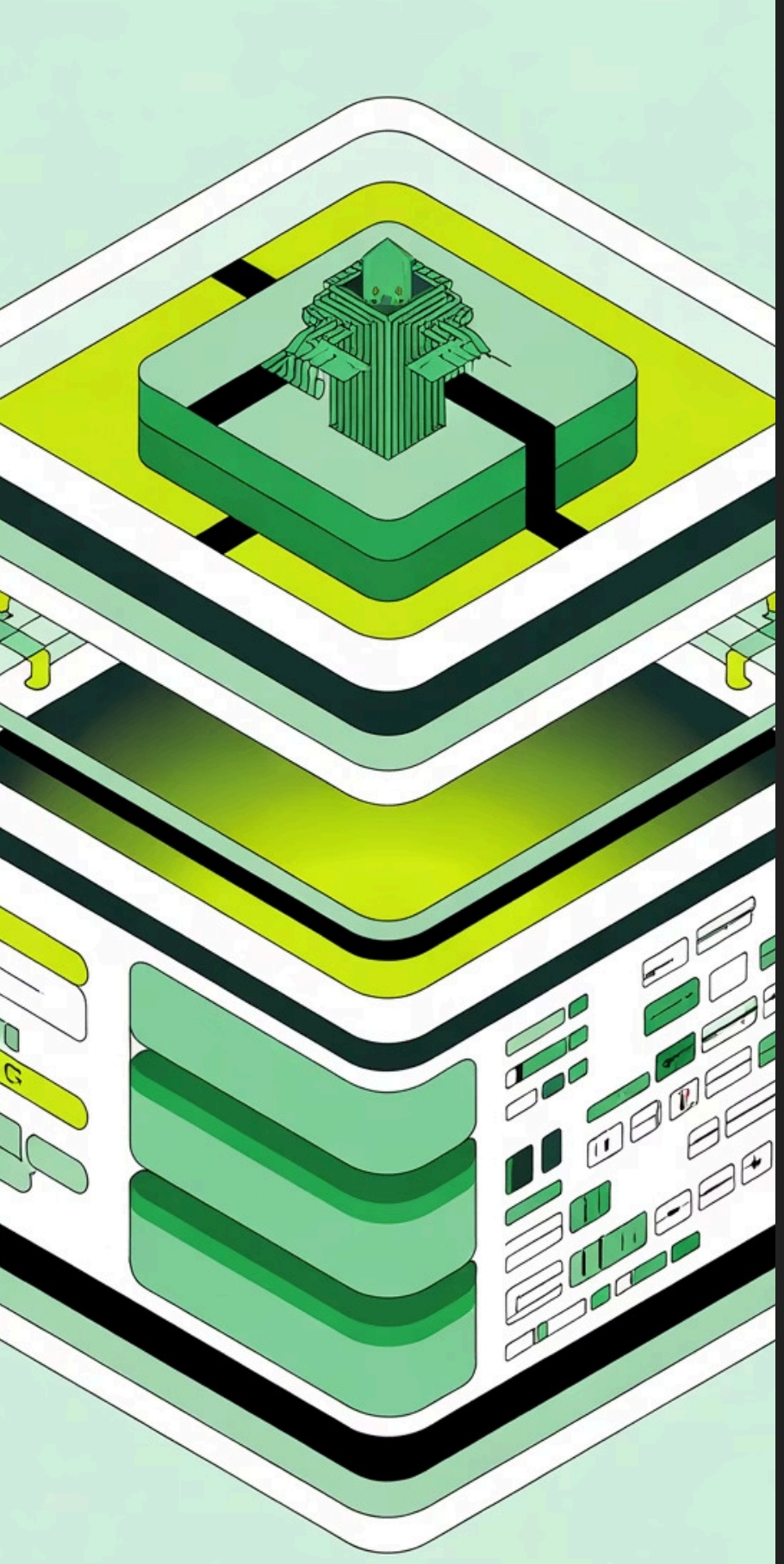
Software 2.0

Machine learning weights(represent what the computer has learnt)



Software 3.0

With LLMs the interface becomes natural language



Why Learning Software Engineering Matters 💪

(Even if You Want to Focus on AI)

Every AI system needs strong software engineering around it. Machine learning models don't work in isolation — they need:

Data Pipelines

Systems to collect, clean, and process data efficiently

APIs & Integration

Connections that let your AI talk to other systems

User Interfaces

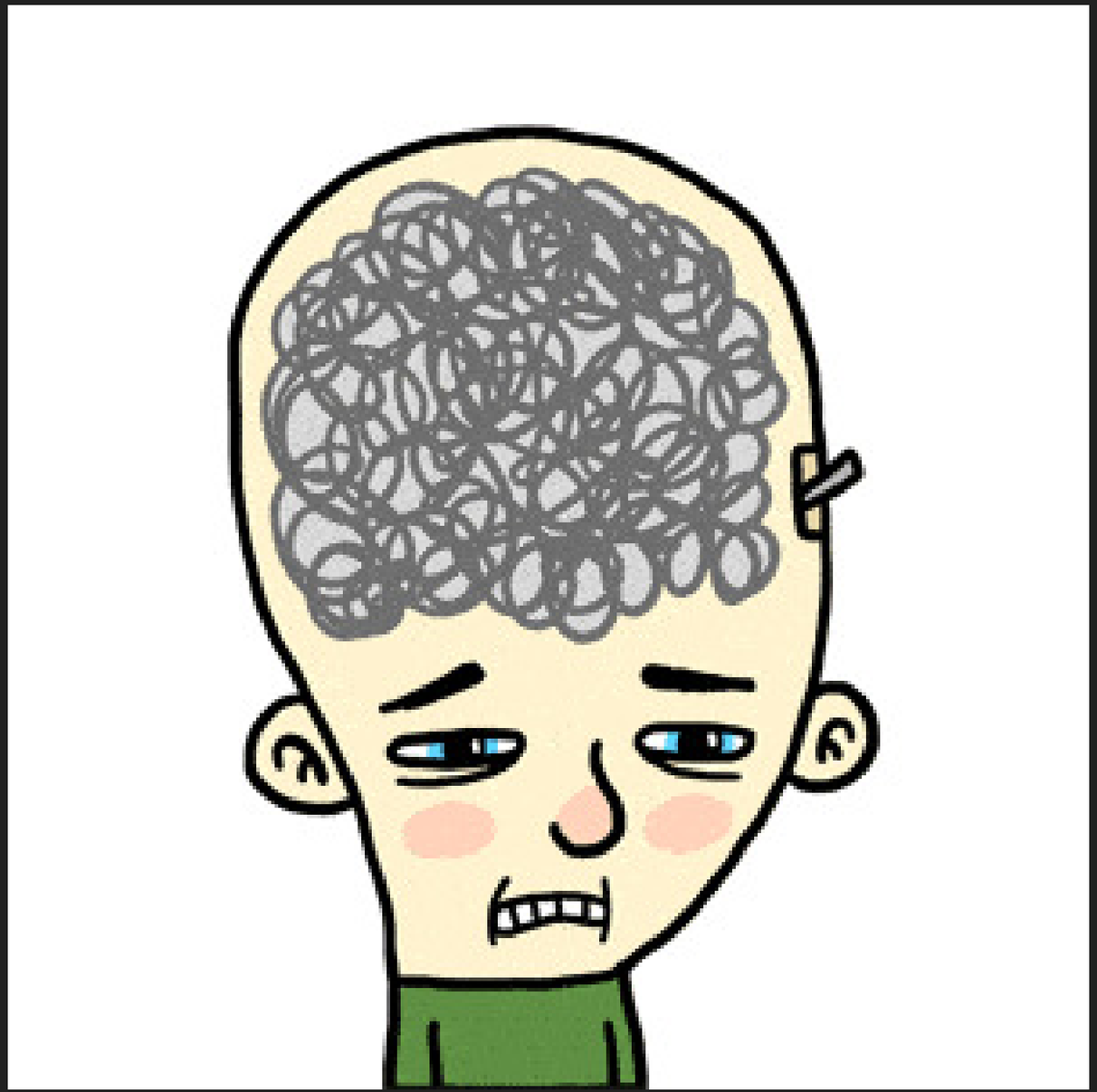
Beautiful apps that make AI accessible to everyone

Reliability Checks

Monitoring, testing, and ensuring systems work 24/7

"Great AI is built by great engineers." — Good software engineering gives you a strong foundation for any tech career.

Let take 5mins Quick break/Stretch



Hands-on Session: Python mini-project ~30-40mins

Mini Project!🎯

Build a Quote Generator Web App

Today, you'll experience real software engineering! We're building a web app that generates inspiring quotes using Python, FastAPI, and HTML.

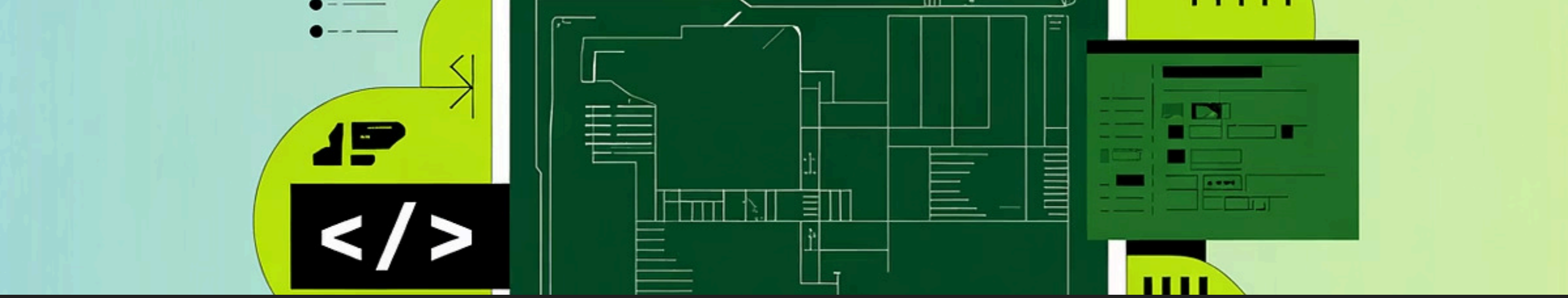
You'll see how these pieces work together:

- **Python** — The programming language for logic
- **FastAPI** — Framework for creating web APIs
- **HTML** — The structure of web pages

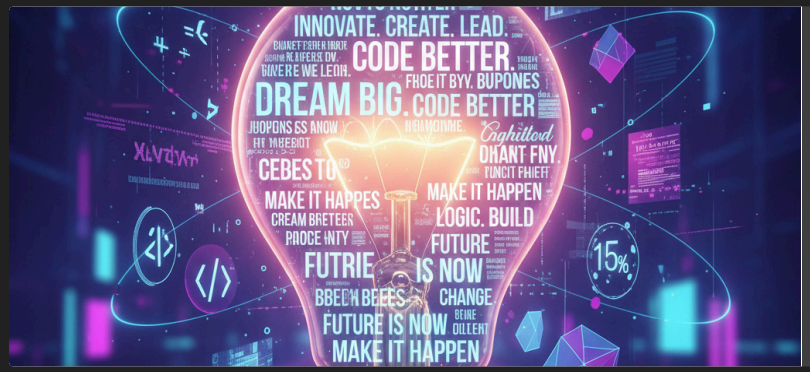
From a simple `print("Hello, world!")` to a working web application in one session!

```
# Your first line of code
print("Hello, world!")

# Soon becomes...
@app.get("/quote")
def get_quote():
    return {"quote": "Keep building!"}
```



Let's visit the code



GitHub

GitHub - soloeinsteinmit/inspireme-fastapi-demo: A beginner-friendly Fas...

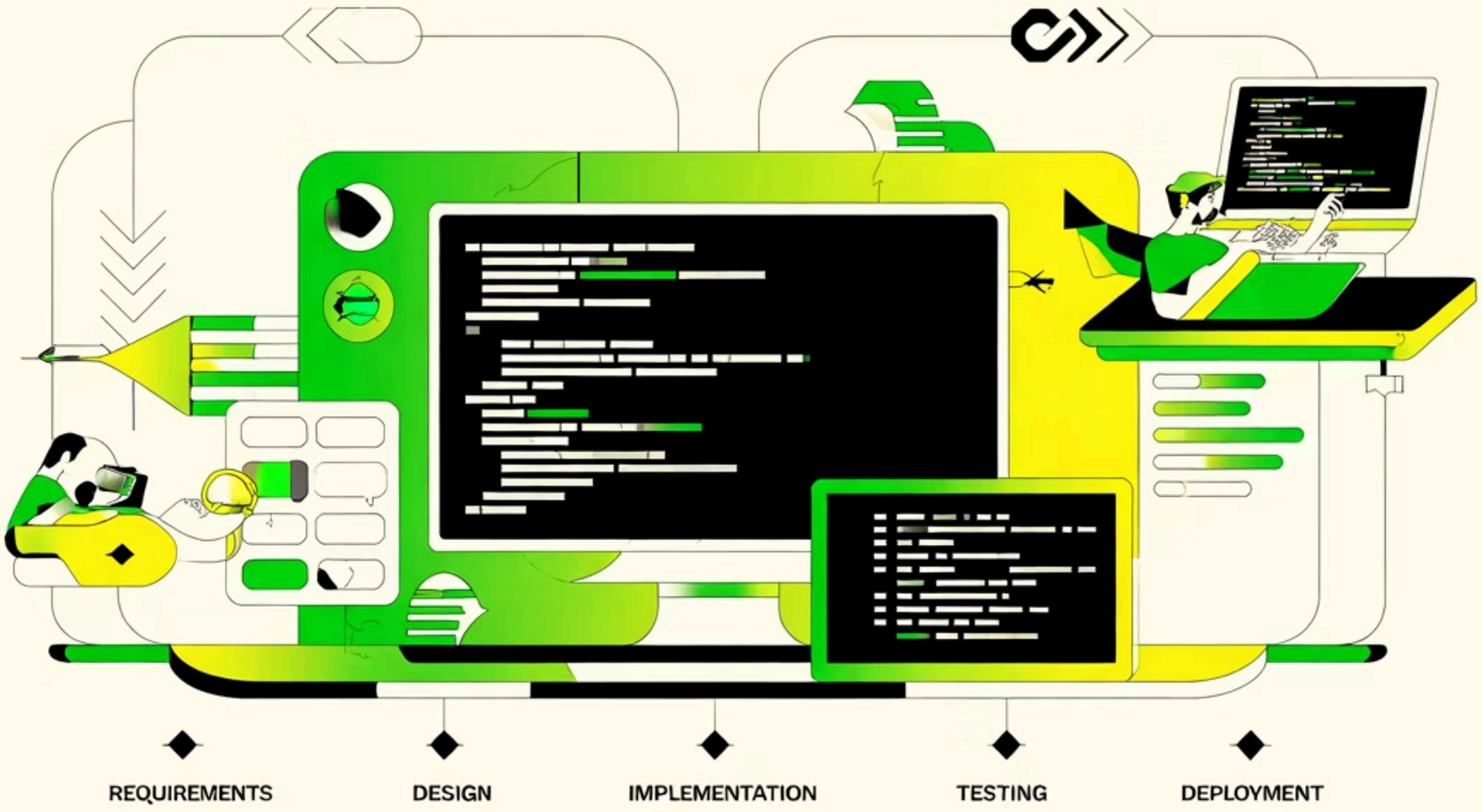
A beginner-friendly FastAPI web app built during the Tech & Beyond Expo Ghana, Academic City University workshop. This demo teaches software engineering...

https://github.com/soloeinsteinmit/inspireme-fastapi-demo.git

Software Engineering Principles Used 🧩

Let's connect the theoretical building blocks of software engineering to the practical mini-project we just explored: the InspireMe Quote Generator.

1️⃣	Requirements	“Build an app that shows random motivational quotes.”
2️⃣	Design	Use FastAPI + HTML + CSS for a simple web interface.
3️⃣	Implementation	Write the Python code for FastAPI, and HTML/CSS for the frontend.
4️⃣	Testing	Run the app locally to ensure quotes are generated, displayed, and the API works correctly.
5️⃣	Deployment	(Optional) Upload to GitHub, or deploy to a cloud service like Render for others to access.



Software 3.0: Keeping AI agent on a leash

"AI-assisted Coding" workflows (very rapidly evolving...)

- describe the single, next concrete, incremental change
- don't ask for code, ask for approaches
  - pick an approach, draft code
  - review / learn: pull up API docs, ask for explanations, ...
  - wind back, try a different approach
- test
- git commit
- ask for suggestions on what could be implemented next
- repeat

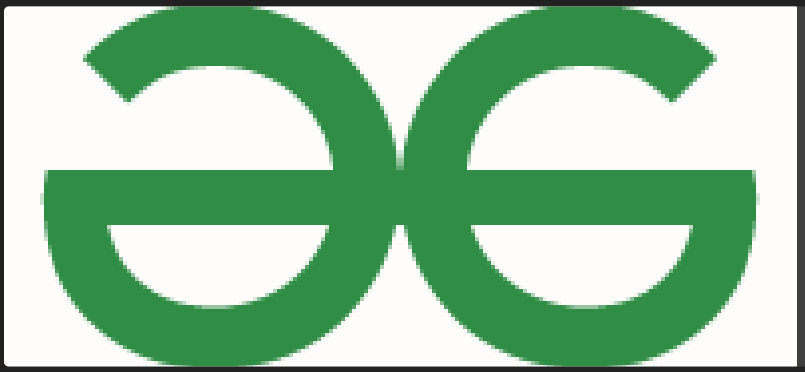
Wrap-up

Resources

www.w3schools.com

W3Schools.com

W3Schools offers free online tutorials, references and exercises in all the major languages of the web. Covering popular subjects like HTML, CSS, JavaScript, Python,...



GeeksforGeeks

GeeksforGeeks

Your All-in-One Learning Portal. It contains well written, well thought and well explained computer science and programming articles, quizzes and practice/competitive...

- YouTube

Next-steps

PRACTICE! PRACTICE!! PRACTICE!!!

PRACTICE MAKES PERFECT...

Inspiration & Take-Home Message🌟

⚡ Andrej Karpathy's Key Insights on Software Engineering and AI

Andrej Karpathy — one of the top AI engineers in the world — said that *we're entering a new era where AI is changing how we build software itself*. Here are his key ideas:

Software: The Ultimate Lever

It's how humans amplify their creativity and problem-solving capabilities.

AI as the New "Engineer"

Instead of manual coding, humans will soon guide AI to build software with them.

Future: AI Software Engineers

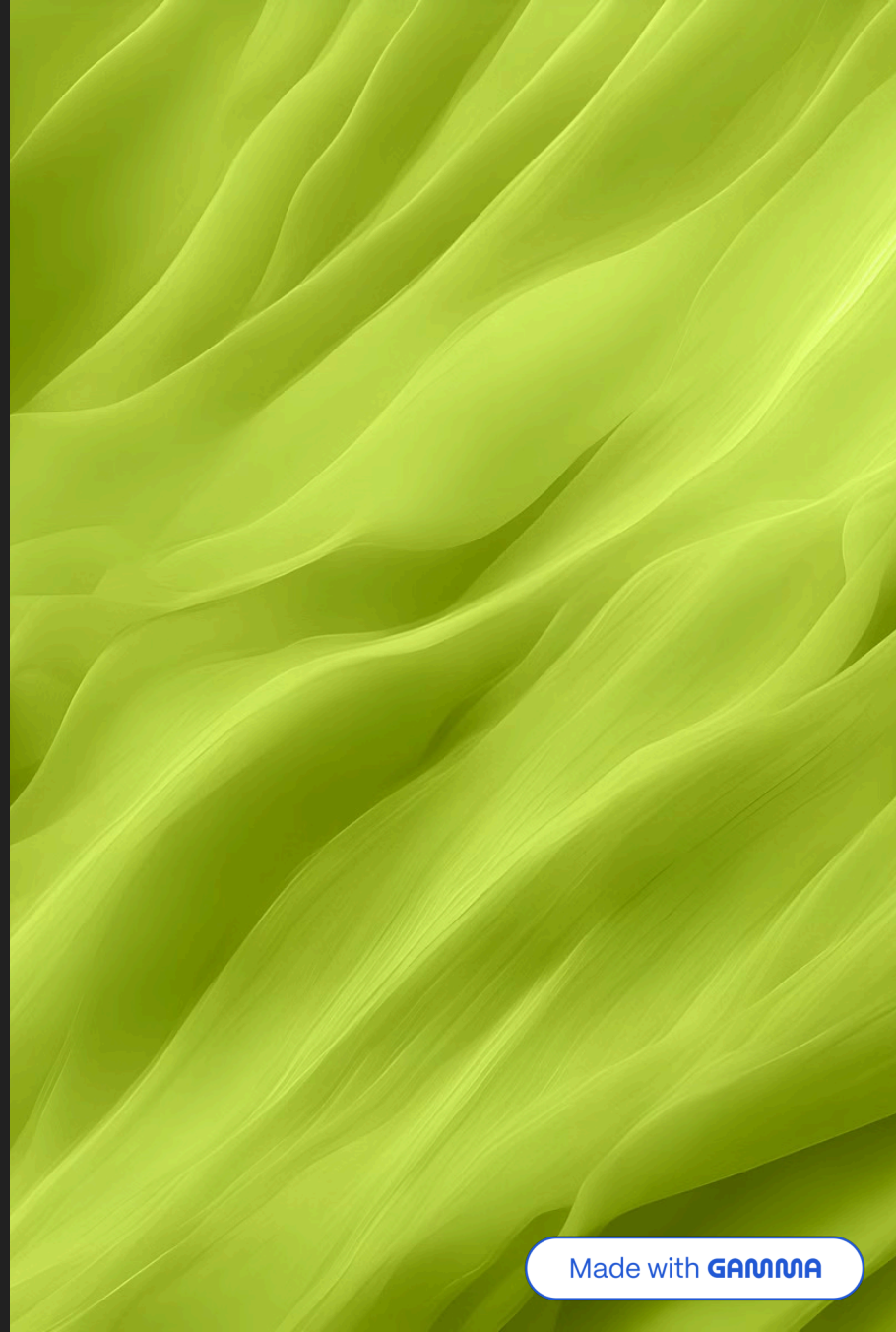
Those who understand both traditional software development and AI integration will thrive.

Master the Foundations

Understanding SDLC and basic coding makes you unstoppable when AI tools emerge.

"If it is to be, it is up to me." — Start small, think big, and keep building. The world runs on software, and you can be part of creating it.

> It's not over until I win.



Thank You / Q&A

Thank you for joining this session! I hope you found it insightful and inspiring.

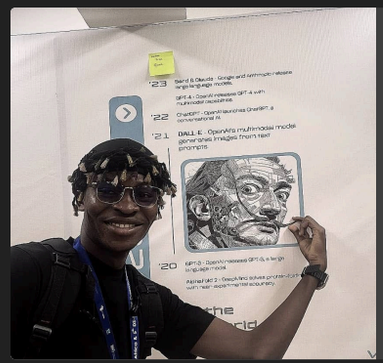
GitHub

GitHub

soloeinsteinmit – Overview

ML/AI Engineer 🧠 | SWE+AI 👩‍💻 | Robotics 🤖 | IoT | FinTech💵(Forex,...

Medium



Medium

Solomon Eshun – Medium

Read writing from Solomon Eshun on Medium. ML/AI Engineer 🧠| LLMs |...

LinkedIn

Solomon Eshun

www.linkedin.com

Email

solomoneshun373@example.com

References

Medium

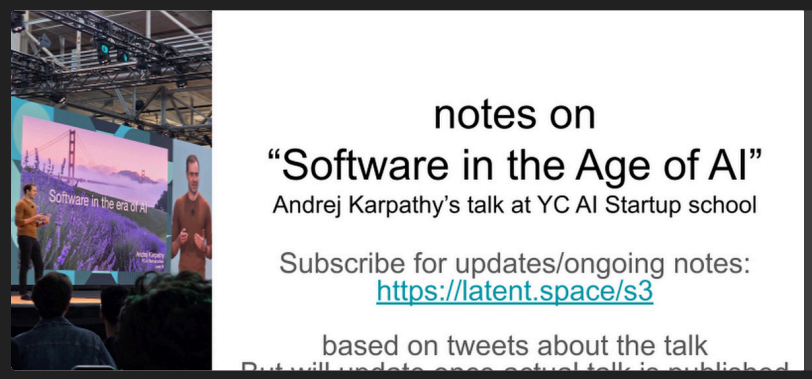
Software 2.0

I sometimes see people refer to neural networks as just "another tool in your machine learning toolbox". They have some pros and cons, they...

YouTube

Andrej Karpathy: Software Is Changing (Again)

Andrej Karpathy's keynote on June 17, 2025 at AI Startup School in San Francisco. Slides provided by Andrej:...



Google Docs

Andrej Karpathy – AI Startup School 2025 – WITH FULL TRANSCRIPT – htt...

notes on "Software in the Age of AI" Andrej Karpathy's talk at YC AI Startup school Subscribe for updates/ongoing notes: https://latent.space/s3 based on tweets about...

www.latent.space

Andrej Karpathy on Software 3.0: Software in the Age of AI

Annotated notes on Andrej's talk at YC AI Startup School 2025