CSC 640 Course Project (HW4) & Documentation (HW5)

- You are `problem solvers` who use software techniques to build and deliver high-quality software products; we call them `software engineers`.
- You are also leading a software engineering team; you need to teach how to use software engineering rules and tools to build high-quality software prodcuts to your team members.

> We have a meeting with a client.

- The clients can be your boss, manager, supervisor, stakeholder, or the real client who pays you to build software.

Client requirements

> The client says

- I need a REST API server
    i. The servers provides student-related information (or any information that you choose to support).
    ii. I should access the server to get information in the JSON format.

- The REST API server should be high-qualtiy
    i. It is designed following software design principles.
    ii. It is easy to read and update the code/documentation.
    iii. It is easy to find and fix bugs.

Software Engineer's Response

- As software engineers, we don't say "Yes" on the spot.
- The answer starts with "I think I can do it, but ..."
    - Let me think more about it ...
    - I need to know more about A and B before I give you my answer ...

Even when you know the solution

- Even when you know you can solve the problem, you should say:
  - It depends on A because of B.
- We need to understand:
  - What technology stack?
  - What programming language?
  - What database?

From the requirements into questions

- We also need to `interpret` the requirements into client questions.
    - What is the problem domain?
    - How do we solve the problem in the most efficient way?
    - Why is this problem worth solving?
    - Who are the stakeholders?

Three client questions

The REST API server should be high-qualtiy (HW4)

1. It is designed following software design principles.

- We need to understand software design principles (HW2).

2. It is easy to read and update the code/documentation.

- We need to use tools such as Marp/Hugo/GitHub/Github.io to publish our work (HW5).

3. It is easy to find and fix bugs.

- We need to use high-level programming languages (HW3)

Software Design (HW2)

- Professional software engineers use these software design principles:
  - OOP APIE & SOLID principles
  - Design Patterns
  - Refactorings

- Professional software engineers do the following to make code easy to read and update:
  - Sense code smell
  - Refactor the code to remove code smell

High-level Programming Language (HW3)

- We don't use low-level programming languages, such as C or assembly language to solve problems.
- We use high-level programming languages:
  - to abstract our thinking
  - to check possible errors by compilers
  - to automate process

Delivery (HW4/5)

- We will use GitHub to deliver the product (REST API server) and document to clients.
- We also will use GitHub for software engineering document for the team members.
- We will use Markdown for documentation.
    - We use Marp for presentation.
    - We use Hugo for web pages.

Think

- `Software Engineers` prove themselves by `solving problems`.
- The proof is from the `building high quality software`.
- The high-quality software must be `adaptable` to any `changes` through `software design`.

How to survive in the AI age as software engineers

> Revolution and dilemma

- Many junior-level software engineering jobs are gone.
- Senior-level software engineers are hard to find.
- Only 10x programmers (with the help of LLM) can and will survive.

- It is said that LLM writes more than 35% of the code in tech giants.
- It is reported that startups use LLM 90 %+ of their code.
- Many believe it will be 99 %+ in five years.
- **No matter what, the revolution is here and now!**

> My experience

- Making toy-level programs (and tests) is almost a miracle.
- Making a professional-level program is a different story, and it is impossible with vibe-coding (yet).

> Fortunately, from the SE perspective, it is a good direction

- Tedious coding/testing labor is gone.
- Instead, we should focus on design.
- We should do the job of senior-level software engineers by using LLMs as junior-level software engineers.

> In this course

- We focus on solving problems to build high-qaulity software.
- We use `software design` and `high-level programming languages` as the guideline.
- As long as `you` solve problems, and if you prove that, you are not wasting your time.

> Your job is

- Understand any users' requirements and translate them into questions.
- Find the solutions, then design, build, test, and deliver Flutter applications.
- Design the software system that can easily (1) find and fix bugs and (2) respond to users' requests.

Good Signals

- "I'm happy I can focus more on software design, as I don't need to write tedious code and tests anymore."
- "It's fun to design applications with LLM, and I can guide LLM in producing my code and tests."
- "It's amazing that LLM finds bugs and helps me fix them."
- "It's fun to discuss my MVVM structure with LLM."

Bad (Dangerous) Signals

- "I don't understand the code/tests that LLM produces, but I finished my homework anyway."
- "I just pushed to github the LLM code produced by LLM, but I just finished my tasks anyway."
- "I have no idea what I'm doing, but I feel like I'm doing something anyway."

OK. Let's start learning how to design and build high quality software using software engineering rules and tools together!