Week One: Managing Complexity in Software Engineering

Date: October 16th

🧠 Software Engineering = Managing Complexity

Software engineering is about designing systems that can handle complexity over time. The most difficult challenges often come from unknown unknowns — problems or requirements that aren't visible until they cause issues. These hidden risks can derail projects if not addressed early.

Quality software anticipates change and uncertainty through thoughtful design, planning, and the use of appropriate tools. The earlier complexity is identified, the better the system can be structured to handle it.

📏 Rules of the Game

To build quality software, follow two essential principles:

- KISS (Keep It Simple, Stupid): Simplicity makes code easier to read, test, and maintain. Complexity increases the risk of bugs and slows down development. Simple systems are more robust and easier to evolve.

📏 Rules of the Game

- No Surprises (especially in teams): Your teammates should never be confused or caught off guard by your code. This means:
    - Clear structure and naming: Code should be self-explanatory and follow agreed conventions.
    - Consistent delivery: Push updates regularly, follow version control best practices, and avoid last-minute changes that disrupt others.
    - Predictable behavior: Code should do what it says it does — no hidden logic or unexpected side effects.
    - Team success = readable, predictable, and consistently delivered code.

🛠️ Tools for Managing Complexity

Software engineers rely on key tools and concepts to tame complexity:

Tool	Purpose
Software Design	Organizes code into modules and interfaces
Process	Guides development through planning, building, testing, and reviewing
High-Level Languages	Simplify development and allow developers to focus on solving problems

🐾 HW4 Example: Animal Shelter API

- **High-Level Language Use**: The API is built entirely in PHP, a high-level language that handles server-side logic and abstracts away low-level operations. JavaScript is used solely on the client side to access and interact with the API. SQL is used for structured data storage and querying.
- **KISS**: The implementation of each endpoint is kept simple and direct — each function does one thing clearly and predictably, making the code easy to read, debug, and extend.