# Requirements and Use Case Diagrams

Building software that users actually need

## From Requirements to Visual Models

- Requirements tell us what the client wants and guide software development.
- They are a key communication tool between clients and engineers.

Think of requirements like a recipe:

- Actor-Goal Format: Who wants what, and why?
- Use Case Diagrams: A visual recipe showing all the roles and steps involved.

## Real Example: Employee Management System

```
Epic requirement:
As an "employer,"
I want to "generate reports"
so that "I can manage my employees."

Sub requirement 1: As an "account manager,"
I want to "generate an accounting report (name + salary),"
so that "I can track monthly payments."

Sub requirement 2: As a "staff manager,"
I want to "generate a staffing report (name + job title),"
so that "I can track my team."

Sub requirement 3: As a "CEO,"
I want to "generate a schedule report (shift times),"
so that "I can track working hours."
```

```
<svg viewBox="0 0 800 600"
xmlns="http://www.w3.org/2000/svg"> <rect x="200" y="50"
width="500" height="500" fill="none" stroke="black" stroke-
width="2" rx="10"/> <text x="430" y="40" text-
anchor="middle" font-family="Arial" font-size="14" font-
weight="bold">Employee Management System</text> <circle
cx="100" cy="150" r="15" fill="none" stroke="black" stroke-
width="2"/> <line x1="100" y1="165" x2="100" y2="200"
stroke="black" stroke-width="2"/> <line x1="85" y1="180"
x2="115" y2="180" stroke="black" stroke-width="2"/> <line
x1="100" y1="200" x2="85" y2="230" stroke="black" stroke-
width="2"/> <line x1="100" y1="200" x2="115" y2="230"
stroke="black" stroke-width="2"/> <text x="100" y="250" text-
anchor="middle" font-family="Arial" font-size="12">Account
Manager</text> <circle cx="100" cy="320" r="15" fill="none"
```

## Why Use Case Diagrams Matter

Visualizing Requirements

- **Improved Communication:**
    - Everyone can see and understand what's needed.
- **Complete Coverage:**
    - Ensures no features are missed.

## Foundation for Architecture & Design

- Defined Boundaries:
    - Clarifies what each module does and how they interact.
- Testing Reference:
    - Acts as a checklist to verify every module works as intended.