A software development team is building an e-commerce platform with several modules like:

1. **User Authentication**
2. **Product Search**
3. **Shopping Cart**
4. **Payment Gateway**
5. **Order History**

**Defect Clustering in Action:**

During testing, the team discovers that most of the bugs and defects are concentrated in the **Payment Gateway** and **Shopping Cart** modules, while the **User Authentication** and **Order History** modules have very few defects.

* The **Payment Gateway** module, responsible for handling transactions, is complex with multiple integrations (third-party APIs, payment providers, etc.). It’s frequently modified to accommodate various currencies and payment methods, increasing the likelihood of introducing defects.
* Similarly, the **Shopping Cart** module has many business rules (like calculating taxes, discounts, and offers) and interacts with multiple other modules, increasing its complexity.

Even though the project has several other modules, **most of the defects (e.g., 60-80%) are found in just 20% of the modules**, which is typically due to:

* High complexity
* Frequent changes
* Interactions with other systems or third-party services

Vì Các test case lặp đi lặp lại cũng chỉ tìm thấy những lỗi mà chúng đã được thiết kế để phát hiện, trong khi bỏ qua các lỗi mới hoặc đã thay đổi.

#### Ví dụ: Tình huống:

### Ví dụ chi tiết:

Nếu bạn có một test case kiểm tra tính năng đăng nhập của hệ thống với các tổ hợp tên người dùng và mật khẩu hợp lệ và không hợp lệ, thì test case đó sẽ tiếp tục tìm được các lỗi liên quan đến **việc nhập sai tên đăng nhập hoặc mật khẩu**. Tuy nhiên, nếu có một tính năng mới được bổ sung vào hệ thống, chẳng hạn như **xác thực hai yếu tố (2FA)**, test case cũ không được thiết kế để kiểm tra tính năng này. Do đó, các lỗi tiềm ẩn trong tính năng xác thực 2FA sẽ bị bỏ qua.

Tóm lại cần nói thêm về nguyên nhân , ví dụ => cần làm gì và lợi ích của nó