## Mô hình Waterfall là gì?

**Mô hình Thác nước** là một trong những mô hình phát triển phần mềm đầu tiên và được sử dụng rộng rãi. Nó được ví như một dòng thác, các giai đoạn phát triển diễn ra tuần tự, liên tục và không quay trở lại giai đoạn trước.

**Các giai đoạn chính của mô hình Thác nước:**

1. **Phân tích yêu cầu:** Thu thập, phân tích và xác định rõ ràng các yêu cầu của khách hàng đối với sản phẩm phần mềm.
2. **Thiết kế:** Dựa trên các yêu cầu đã thu thập, tiến hành thiết kế kiến trúc, giao diện người dùng, cơ sở dữ liệu và các thành phần khác của hệ thống.
3. **Thực hiện:** Viết mã nguồn để xây dựng sản phẩm dựa trên thiết kế đã hoàn thiện.
4. **Kiểm thử:** Kiểm tra xem sản phẩm đã hoàn thành có đáp ứng các yêu cầu ban đầu và hoạt động đúng không.
5. **Triển khai:** Đưa sản phẩm vào sử dụng và bàn giao cho khách hàng.
6. **Bảo trì:** Tiến hành cập nhật, sửa lỗi và nâng cấp sản phẩm sau khi đã đưa vào sử dụng.

**Ưu điểm của mô hình Thác nước:**

* **Rõ ràng, đơn giản:** Quy trình phát triển được chia thành các giai đoạn rõ ràng, dễ quản lý.
* **Dễ lập kế hoạch:** Thời gian và chi phí cho từng giai đoạn có thể được ước tính một cách tương đối chính xác.
* **Thích hợp cho các dự án quy mô nhỏ, yêu cầu ổn định:** Khi các yêu cầu của khách hàng không thay đổi nhiều.

**Nhược điểm của mô hình Thác nước:**

* **Khó thay đổi:** Nếu có thay đổi yêu cầu trong quá trình phát triển, sẽ rất khó để điều chỉnh và có thể dẫn đến chi phí tăng cao.
* **Khách hàng không được tham gia sớm:** Khách hàng chỉ được xem sản phẩm hoàn chỉnh ở giai đoạn cuối, có thể dẫn đến sự không hài lòng.
* **Rủi ro cao:** Nếu có lỗi phát sinh ở các giai đoạn đầu, sẽ chỉ được phát hiện ở các giai đoạn sau, gây lãng phí thời gian và tài nguyên.

# Giai đoạn yêu cầu là gì?

**Giai đoạn Yêu cầu** là bước đầu tiên và vô cùng quan trọng trong mô hình phát triển phần mềm theo phương pháp Thác nước. Ở giai đoạn này, mục tiêu chính là **hiểu rõ và xác định một cách chính xác những gì khách hàng cần**.

# Giai đoạn thiết kế là gì?

**Giai đoạn thiết kế** là bước tiếp theo sau khi đã xác định rõ ràng các yêu cầu trong giai đoạn phân tích. Đây là giai đoạn mà đội ngũ phát triển sẽ chuyển đổi các yêu cầu đó thành một thiết kế chi tiết cho sản phẩm phần mềm.

# Giai đoạn thực hiện là gì?

**Giai đoạn thực hiện** là giai đoạn mà các nhà phát triển phần mềm chuyển đổi thiết kế chi tiết đã được phê duyệt thành một sản phẩm phần mềm hoàn chỉnh. Đây là giai đoạn mà mã nguồn được viết, các thành phần phần mềm được tích hợp và hệ thống được xây dựng.

# Giai đoạn kiểm thử là gì ?

**Giai đoạn kiểm thử** là giai đoạn quan trọng để đảm bảo rằng phần mềm đã phát triển đáp ứng đầy đủ các yêu cầu đặt ra và hoạt động đúng như mong đợi. Trong giai đoạn này, các nhà phát triển sẽ tiến hành kiểm tra kỹ lưỡng từng chức năng, tính năng của phần mềm để tìm ra và sửa chữa các lỗi, sai sót.

# Giai đoạn triển khai là gì ?

**Giai đoạn triển khai** là giai đoạn cuối cùng trong mô hình phát triển phần mềm theo phương pháp Thác nước. Sau khi phần mềm đã được phát triển, kiểm thử và đảm bảo chất lượng, giai đoạn này sẽ đưa sản phẩm vào môi trường thực tế để người dùng cuối cùng có thể sử dụng.

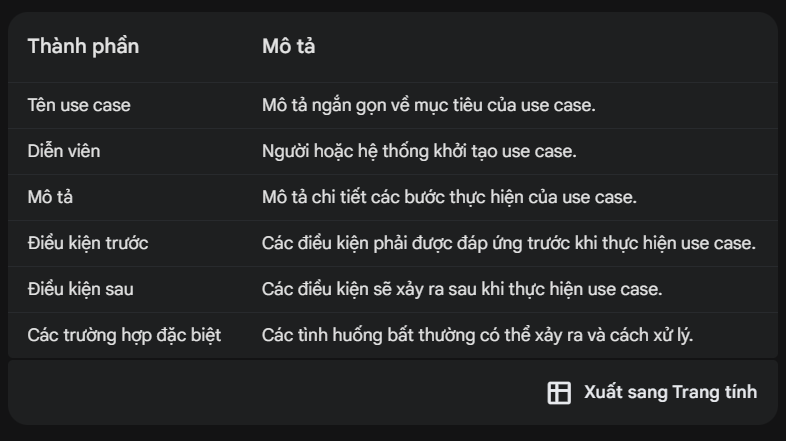
# Giai đoạn bảo trì là gì ?

**Giai đoạn bảo trì** là giai đoạn cuối cùng và liên tục trong mô hình phát triển phần mềm theo phương pháp Thác nước. Sau khi phần mềm đã được triển khai và đưa vào sử dụng, giai đoạn bảo trì sẽ đảm bảo rằng phần mềm luôn hoạt động ổn định, hiệu quả và đáp ứng được nhu cầu thay đổi của người dùng.

# Use case là gì

**Use case** (trường hợp sử dụng) là một mô tả chi tiết về cách một người dùng tương tác với một hệ thống để hoàn thành một mục tiêu cụ thể. Nó là một công cụ hữu ích trong phân tích hệ thống, giúp chúng ta hiểu rõ hơn về những gì người dùng mong đợi từ hệ thống và xác định các chức năng mà hệ thống cần phải cung cấp.

# Các thành phần của một Use case là gì



# Lợi ích của việc sử dụng Use case là gì

**10 Lợi Ích Của Việc Sử Dụng Use Case**

Use case là một công cụ vô cùng hữu ích trong quá trình phát triển phần mềm, giúp đảm bảo rằng sản phẩm cuối cùng đáp ứng đúng nhu cầu của người dùng. Dưới đây là 10 lợi ích chính của việc sử dụng use case:

1. **Hiểu rõ yêu cầu của người dùng:** Use case giúp các nhà phát triển hiểu rõ hơn về cách người dùng thực sự tương tác với hệ thống, từ đó xác định chính xác các yêu cầu chức năng.
2. **Cải thiện giao tiếp:** Use case cung cấp một ngôn ngữ chung dễ hiểu cho cả khách hàng, nhà phát triển và các bên liên quan khác, giúp giảm thiểu hiểu lầm và đảm bảo mọi người cùng chung một mục tiêu.
3. **Xác định phạm vi dự án:** Use case giúp xác định rõ ràng phạm vi của dự án, tránh việc phát triển những tính năng không cần thiết, giúp tiết kiệm thời gian và tài nguyên.
4. **Tạo ra thiết kế tốt hơn:** Use case là cơ sở để thiết kế giao diện người dùng, các luồng làm việc và các quy trình nghiệp vụ trong hệ thống.
5. **Cải thiện chất lượng phần mềm:** Bằng cách xác định rõ các trường hợp sử dụng, các nhà phát triển có thể dễ dàng phát hiện và sửa lỗi sớm trong quá trình phát triển.
6. **Tăng cường khả năng kiểm thử:** Use case là cơ sở để xây dựng các trường hợp kiểm thử, giúp đảm bảo rằng phần mềm hoạt động đúng như mong đợi.
7. **Hỗ trợ tài liệu hóa:** Use case có thể được sử dụng để tạo ra tài liệu hướng dẫn người dùng, giúp họ dễ dàng sử dụng hệ thống.
8. **Tăng tính minh bạch:** Use case giúp làm rõ các quyết định thiết kế và các thay đổi trong quá trình phát triển.
9. **Hỗ trợ quản lý dự án:** Use case có thể được sử dụng để theo dõi tiến độ dự án và đánh giá hiệu quả của các hoạt động phát triển.
10. **Tái sử dụng:** Use case có thể được tái sử dụng trong các dự án khác, giúp tiết kiệm thời gian và công sức.

# Làm thế nào để xác định các Use case

**1. Phỏng vấn người dùng:**

* **Tiếp xúc trực tiếp:** Gặp gỡ và phỏng vấn người dùng để hiểu rõ cách họ làm việc hiện tại và những vấn đề họ gặp phải.
* **Sử dụng câu hỏi mở:** Đặt các câu hỏi mở để khuyến khích người dùng chia sẻ chi tiết về công việc và nhu cầu của họ.
* **Quan sát trực tiếp:** Quan sát người dùng thực hiện công việc để hiểu rõ hơn về quy trình làm việc của họ.

**2. Phân tích tài liệu hiện có:**

* **Tài liệu yêu cầu:** Đọc kỹ các tài liệu yêu cầu để xác định các chức năng mà hệ thống cần phải cung cấp.
* **Sổ tay hướng dẫn người dùng:** Phân tích các sổ tay hướng dẫn người dùng hiện có để hiểu rõ cách người dùng tương tác với hệ thống hiện tại.
* **Các báo cáo:** Phân tích các báo cáo liên quan đến hệ thống để tìm ra các vấn đề và nhu cầu cải tiến.

**3. Sử dụng kỹ thuật CRUD:**

* **CRUD** là viết tắt của Create, Read, Update, Delete (Tạo, Đọc, Cập nhật, Xóa).
* Áp dụng kỹ thuật này để xác định các hoạt động cơ bản mà người dùng cần thực hiện với dữ liệu.
* Ví dụ: Một hệ thống quản lý khách hàng sẽ có các use case như tạo khách hàng mới, xem thông tin khách hàng, cập nhật thông tin khách hàng và xóa khách hàng.

**4. Áp dụng các kỹ thuật khác:**

* **Phân tích Use Case:** Sử dụng các kỹ thuật phân tích để xác định các use case từ các yêu cầu chung.
* **Storming:** Tổ chức các buổi thảo luận nhóm để thu thập ý kiến từ nhiều người.
* **Card Sorting:** Sử dụng các thẻ để phân loại các chức năng của hệ thống.

# ERD là gì

**ERD** là viết tắt của **Entity-Relationship Diagram**, dịch sang tiếng Việt là **Sơ đồ Mối quan hệ Thực thể**. Đây là một công cụ trực quan được sử dụng rộng rãi trong thiết kế cơ sở dữ liệu để mô tả mối quan hệ giữa các thực thể (entity) trong một hệ thống.

# Các thành phần chính của một ERD là gì

**1. Thực thể (Entity)**

* **Định nghĩa:** Một đối tượng hoặc khái niệm có ý nghĩa trong hệ thống mà chúng ta muốn mô hình hóa.
* **Biểu diễn:** Thường được biểu diễn bằng hình chữ nhật.
* **Ví dụ:** Khách hàng, sản phẩm, đơn hàng, nhân viên.

**2. Thuộc tính (Attribute)**

* **Định nghĩa:** Các đặc tính mô tả một thực thể.
* **Biểu diễn:** Được liệt kê bên trong hình chữ nhật biểu diễn thực thể.
* **Ví dụ:** Đối với thực thể "Khách hàng", các thuộc tính có thể là: Mã khách hàng, Tên, Địa chỉ, Số điện thoại.

**3. Mối quan hệ (Relationship)**

* **Định nghĩa:** Mô tả sự liên kết giữa các thực thể.
* **Biểu diễn:** Được biểu diễn bằng đường nối giữa các thực thể.
* **Các loại mối quan hệ:**
  + **Một-một (One-to-one):** Một thực thể chỉ liên kết với một thực thể khác và ngược lại.
  + **Một-nhiều (One-to-many):** Một thực thể có thể liên kết với nhiều thực thể khác, nhưng một thực thể khác chỉ liên kết với một thực thể.
  + **Nhiều-nhiều (Many-to-many):** Một thực thể có thể liên kết với nhiều thực thể khác và ngược lại.

**4. Độ bội (Cardinality)**

* **Định nghĩa:** Chỉ ra số lượng bản ghi của một thực thể có thể liên kết với số lượng bản ghi của một thực thể khác.
* **Biểu diễn:** Thường được biểu diễn bằng các số ở hai đầu đường nối.
* **Ví dụ:** 1:N (một-nhiều), N:M (nhiều-nhiều).

# Lợi ích của việc sử dụng ERD là gì

**1. Hiểu rõ cấu trúc dữ liệu:**

* **Hình ảnh hóa:** ERD cung cấp một cái nhìn tổng quan và trực quan về cách các thực thể (entity) và thuộc tính (attribute) liên kết với nhau trong hệ thống.
* **Dễ hiểu:** Ngay cả những người không chuyên về cơ sở dữ liệu cũng có thể dễ dàng hiểu được cấu trúc dữ liệu thông qua ERD.

**2. Cải thiện giao tiếp:**

* **Ngôn ngữ chung:** ERD trở thành một ngôn ngữ chung giữa các thành viên trong nhóm phát triển, từ nhà thiết kế đến lập trình viên.
* **Giảm thiểu hiểu lầm:** Việc sử dụng ERD giúp giảm thiểu những hiểu lầm về cấu trúc dữ liệu, đảm bảo mọi người cùng chung một hình dung.

**3. Phát hiện lỗi sớm:**

* **Nhận biết bất hợp lý:** ERD giúp phát hiện các bất hợp lý trong cấu trúc dữ liệu ngay từ giai đoạn thiết kế, tránh các lỗi nghiêm trọng khi triển khai.
* **Tiết kiệm thời gian và chi phí:** Việc sửa lỗi ở giai đoạn thiết kế sẽ tiết kiệm được nhiều thời gian và chi phí hơn so với việc sửa lỗi sau khi hệ thống đã được triển khai.

**4. Tối ưu hóa cơ sở dữ liệu:**

* **Thiết kế hiệu quả:** ERD giúp thiết kế các bảng và mối quan hệ trong cơ sở dữ liệu một cách hiệu quả, tránh sự trùng lặp dữ liệu và đảm bảo tính toàn vẹn của dữ liệu.
* **Nâng cao hiệu suất:** Một cơ sở dữ liệu được thiết kế tốt dựa trên ERD sẽ hoạt động nhanh hơn và hiệu quả hơn.

**5. Dễ dàng bảo trì:**

* **Tài liệu tham khảo:** ERD đóng vai trò như một tài liệu tham khảo quan trọng trong quá trình bảo trì và phát triển hệ thống.
* **Dễ dàng cập nhật:** Khi có sự thay đổi trong cấu trúc dữ liệu, ERD có thể được cập nhật một cách dễ dàng.

**6. Hỗ trợ quá trình phát triển phần mềm:**

* **Cơ sở cho lập trình:** ERD cung cấp một cơ sở vững chắc để các lập trình viên viết mã truy vấn và thao tác với dữ liệu.
* **Tích hợp với các công cụ khác:** ERD có thể được tích hợp với các công cụ phát triển phần mềm khác, giúp tự động hóa một số quá trình.