HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

NHẬP MÔN CNPM

Bài tập:

Chương 1-5

Người hướng dẫn **ThS. Châu Văn Vân**

Sinh viên thực hiện **NGÔ QUANG MINH**
N22DCCN053

Lớp **D22CQCN01-N**

TP.HCM – 2025

CHƯƠNG 1: PHẠM VI CÔNG NGHỆ PHẦN MỀM

CÂU HỎI TRẮC NGHIỆM

1. Câu hỏi 1: Phần mềm bao gồm các loại nào dưới đây?
A. Phần mềm hệ thống
B. Phần mềm ứng dụng
C. Phần mềm nhúng
D. Cả A, B và C
2. Câu hỏi 2: Công nghệ phần mềm là gì?
A. Việc viết mã nguồn cho phần mềm
B. Phát triển phần mềm mà không có lỗi
C. Ứng dụng các phương pháp khoa học để phát triển phần mềm
D. Chỉ bảo trì phần mềm
3. Câu hỏi 3: Quy trình phát triển phần mềm gồm mấy giai đoạn chính?
A. 3
B. 4
C. 5
D. 6
4. Câu hỏi 4: Hoạt động nào dưới đây thuộc quy trình bảo trì phần mềm?
A. Lập kế hoạch
B. Triển khai phần mềm
C. Cập nhật phần mềm để phù hợp với thay đổi môi trường
D. Phân tích yêu cầu
5. Câu hỏi 5: Chi phí bảo trì phần mềm chiếm bao nhiêu phần trăm tổng chi phí vòng đời phần mềm?
A. 10%
B. 30%
C. 60%
D. 90%
6. Câu hỏi 6: Nguyên nhân chính gây ra việc vượt chi phí khi phát triển phần mềm là gì?
A. Thiếu nhân lực
B. Không xác định rõ yêu cầu
C. Thay đổi công nghệ
D. Cả A và C
7. Câu hỏi 7: Yêu cầu nào dưới đây không phải là yêu cầu phi chức năng?
A. Hiệu suất xử lý
B. Tính bảo mật
C. Khả năng mở rộng
D. Chức năng đăng nhập
8. Câu hỏi 8: Khi nào phần mềm được coi là hoàn thành?
A. Khi hoàn thành việc viết mã nguồn
B. Khi được bàn giao cho khách hàng và không còn lỗi
C. Khi được triển khai trên hệ thống của khách hàng
D. Khi được khách hàng chấp nhận và đưa vào sử dụng
9. Câu hỏi 9: Vấn đề phổ biến nào thường gặp khi phát triển phần mềm?
A. Thiếu công cụ hỗ trợ

B. Vượt chi phí, trễ thời hạn và lỗi sau khi bàn giao
C. Không có đội kiểm thử
D. Tất cả đều đúng
10. Câu hỏi 10: Phần mềm có thể được chia thành bao nhiêu loại chính?
A. 2
B. 3
C. 4
D. 5

CÂU HỎI NGẮN

1. Phần mềm là gì?
Phần mềm là tập hợp các hướng dẫn, chương trình được viết để máy tính thực thi nhằm thực hiện các chức năng hoặc nhiệm vụ cụ thể.
2. Công nghệ phần mềm là gì?
Công nghệ phần mềm là lĩnh vực nghiên cứu, phát triển và áp dụng các phương pháp có hệ thống để xây dựng phần mềm chất lượng cao.
3. Các loại phần mềm chính là gì?
Phần mềm hệ thống: Hệ điều hành, trình điều khiển thiết bị.
Phần mềm ứng dụng: Các chương trình phục vụ công việc hoặc nhu cầu của người dùng cuối như Microsoft Office, trình duyệt web.
Phần mềm nhúng: Phần mềm điều khiển các thiết bị phần cứng như máygiặt, điều hòa.
4. Tại sao công nghệ phần mềm lại quan trọng?
Tạo ra phần mềm đúng yêu cầu khách hàng.
Đảm bảo phần mềm có thể bảo trì và nâng cấp dễ dàng.
Giảm thiểu thời gian và chi phí phát triển.
5. Quy trình phát triển phần mềm gồm những giai đoạn nào?
Lấy yêu cầu: Thu thập và phân tích các yêu cầu từ khách hàng.
Thiết kế: Lên kế hoạch và cấu trúc hệ thống phần mềm.
Lập trình: Chuyển đổi thiết kế thành mã nguồn thực thi.
Kiểm thử: Đảm bảo phần mềm hoạt động đúng chức năng.
Triển khai: Cài đặt và bàn giao phần mềm cho khách hàng.
Bảo trì: Khắc phục lỗi và nâng cấp phần mềm sau khi triển khai.
6. Khía cạnh kinh tế của công nghệ phần mềm là gì?
Phần mềm là yếu tố cốt lõi trong nhiều ngành công nghiệp như tài chính, y tế, giáo dục. Sự phát triển của phần mềm giúp tăng năng suất lao động và tối ưu hóa chi phí vận hành doanh nghiệp. Các dự án phần mềm lớn thường có chi phí rất cao, do đó việc áp dụng công nghệ phần mềm giúp kiểm soát chi phí tốt hơn
7. Khía cạnh công nghệ của công nghệ phần mềm là gì?
Sự phát triển nhanh chóng của công nghệ yêu cầu phần mềm phải được cải tiến liên tục để đáp ứng nhu cầu mới. Phần mềm giúp kết nối các hệ thống phức tạp và xử lý khối lượng dữ liệu lớn.
8. Khía cạnh bảo trì của công nghệ phần mềm là gì?
Sau khi phần mềm được triển khai, bảo trì là hoạt động cần thiết để đảm bảo phần mềm hoạt động ổn định và đáp ứng các yêu cầu thay đổi của người dùng. Bảo trì phần mềm chiếm khoảng 60% tổng chi phí vòng đời của một hệ thống phần mềm.
9. Các nguyên nhân chính gây trễ thời hạn khi phát triển phần mềm là gì?

Phần mềm thường bị trễ tiến độ so với kế hoạch ban đầu do việc ước lượng thời gian không chính xác. Nguyên nhân: Yêu cầu thay đổi liên tục từ phía khách hàng. Thiếu nhân lực hoặc sự phối hợp kém giữa các thành viên trong nhóm phát triển.

10.Bảo trì phần mềm bao gồm những hoạt động nào?

Sau khi phần mềm được triển khai, bảo trì là hoạt động cần thiết để đảm bảo phần mềm hoạt động ổn định và đáp ứng các yêu cầu thay đổi của người dùng. Bảo trì phần mềm chiếm khoảng 60% tổng chi phí vòng đời của một hệ thống phần mềm.

CÂU HỎI THẢO LUẬN NHÓM

1. **Phân biệt phần mềm hệ thống và phần mềm ứng dụng.**

Đặc điểm	Phần mềm hệ thống	Phần mềm ứng dụng
Mục đích	Quản lý và điều khiển phần cứng và phần mềm máy tính	Thực hiện nhiệm vụ cụ thể của người dùng
Đối tượng sử dụng	Máy tính, phần mềm ứng dụng	Người dùng cuối
Tương tác	Gián tiếp qua phần mềm ứng dụng	Trực tiếp

2. **Thảo luận về vai trò của công nghệ phần mềm trong lĩnh vực tài chính**.

Tự động hóa và tối ưu quy trình:Phần mềm giúp tự động hóa quy trình thủ công , giảm thiểu sai sót và tăng hiệu quả hoạt động

Nâng cao trải nghiệm khách hàng: Thông qua internet dịch vụ được triển khai tới người dùng một cách nhanh chóng và tiện lợi

Quản lý rủi ro và bảo mật: Phần mềm giúp phát hiện và ngăn chặn các hành vi gian lận, công nghệ mã hóa và xác thực giúp bảo vệ dữ liệu

Phần mềm giúp phân tích dữ liệu, dự đoán xu hướng từ đó đưa ra hướng phát triển

3. **Nêu các thách thức thường gặp trong bảo trì phần mềm**.

Thay đổi yêu cầu: Yêu cầu của người dùng có thể thay đổi theo thời gian, đòi hỏi phải cập nhật và sửa đổi

Lỗi và sự cố bất ngờ: Phần mềm có thể gặp lỗi hoặc sự cố buộc phải đưa ra hướng giải quyết nhanh chóng và phù hợp, tránh ảnh hưởng tới trải nghiệm của người dùng cuối

Nguy cơ bảo mật: khi triển khai phần mềm sẽ có rủi ro phát sinh lỗ hổng bảo mật, buộc phải có bản vá kịp thời cho lỗ hổng đó tránh ảnh hưởng tới tính bảo mật của dữ liệu

Khả năng tương thích: Phần mềm cần phải đảm bảo khả năng tương thích với các nền tảng, hệ điều hành, phần cứng khác nhau

Nhân lực: Phải luôn tính toán nguồn nhân lực hợp lý đối với từng giai đoạn từ phát triển đến bảo trì

4. **Vì sao phần mềm thương mại điện tử cần được bảo trì thường xuyên**?

Đảm bảo phần mềm hoạt động ổn định và liên tục: Phần mềm thương mại điện tử là nền tảng cho mọi giao dịch, vì vậy sự ổn định là yếu tố then chốt. Bảo trì định kỳ giúp phát hiện và khắc

phục sớm các lỗi kỹ thuật, tránh tình trạng gián đoạn hoạt động, ảnh hưởng đến doanh thu và uy tín của doanh nghiệp.
Tăng cường bảo mật: Các trang web thương mại điện tử chứa nhiều thông tin nhạy cảm của khách hàng, như thông tin cá nhân, thông tin thanh toán
Cải thiện hiệu suất và tốc độ: Hiệu suất và tốc độ của trang web ảnh hưởng trực tiếp đến trải nghiệm người dùng. Bảo trì giúp tối ưu hóa mã nguồn, cơ sở dữ liệu, cải thiện tốc độ tải trang, tăng tỷ lệ chuyển đổi.
Cập nhật tính năng và công nghệ: Thị trường thương mại điện tử thay đổi liên tục, đòi hỏi phần mềm phải được cập nhật để đáp ứng nhu cầu mới của khách hàng và cạnh tranh với đối thủ.
Khắc phục lỗi và vấn đề phát sinh: Trong quá trình vận hành, phần mềm có thể phát sinh các lỗi hoặc vấn đề không mong muốn.Bảo trì giúp phát hiện và khắc phục sớm các lỗi này, tránh ảnh hưởng đến hoạt động kinh doanh và trải nghiệm người dùng.
Tăng cường khả năng tương thích: Người dùng truy cập trang web thương mại điện tử từ nhiều thiết bị và trình duyệt khác nhau.Bảo trì giúp đảm bảo phần mềm tương thích với các thiết bị và trình duyệt mới nhất, mang lại trải nghiệm tốt nhất cho người dùng.

5. **Phân tích những vấn đề khi yêu cầu khách hàng liên tục thay đổi trong quá trình phát triển phần mềm.**

Tăng chi phí và kéo dài thời gian:Mỗi thay đổi yêu cầu đều đòi hỏi đội phát triển phải điều chỉnh lại thiết kế, mã nguồn và kiểm thử. Điều này dẫn đến tăng chi phí nhân công, vật tư và thời gian thực hiện.Việc thay đổi liên tục khiến dự án khó hoàn thành đúng tiến độ, gây chậm trễ cho việc ra mắt sản phẩm và ảnh hưởng đến kế hoạch kinh doanh của khách hàng.
Giảm chất lượng phần mềm:Thay đổi liên tục có thể dẫn đến việc thiếu kiểm thử kỹ lưỡng, gây ra lỗi và giảm độ ổn định của phần mềm.Việc thay đổi gấp gáp có thể làm mất tính nhất quán trong thiết kế và kiến trúc phần mềm, ảnh hưởng đến khả năng bảo trì và nâng cấp sau này.
Gây mất động lực cho đội phát triển:Việc liên tục thay đổi yêu cầu khiến đội phát triển cảm thấy mệt mỏi, chán nản và mất động lực.
Gây khó khăn trong việc quản lý dự án:Việc thay đổi yêu cầu liên tục khiến việc lập kế hoạch, theo dõi tiến độ và quản lý rủi ro trở nên khó khăn hơn.Người quản lý dự án phải dành nhiều thời gian để xử lý các yêu cầu thay đổi, ảnh hưởng đến các nhiệm vụ quan trọng khác.
Gây mâu thuẫn giữa khách hàng và đội phát triển:Khách hàng có thể cảm thấy không hài lòng khi dự án bị chậm trễ hoặc vượt quá ngân sách.Đội phát triển có thể cảm thấy không được tôn trọng khi các yêu cầu của họ bị thay đổi liên tục.

6. **So sánh chi phí phát triển và chi phí bảo trì phần mềm.**

Chi phí phát triển phần mềm:
Đây là chi phí phát sinh trong giai đoạn đầu tiên của vòng đời phần mềm, từ việc lập kế hoạch và thiết kế đến việc triển khai.
Các yếu tố ảnh hưởng:
Độ phức tạp của phần mềm.
Quy mô dự án.
Công nghệ được sử dụng.
Kỹ năng và kinh nghiệm của đội ngũ phát triển.
Thời gian phát triển.
Đặc điểm:
Thường là chi phí đầu tư một lần hoặc chi phí trả trước.
Có thể dự đoán và lập ngân sách tương đối chính xác.

Chi phí bảo trì phần mềm:
Đây là chi phí phát sinh sau khi phần mềm đã được triển khai, bao gồm việc sửa lỗi, cập nhật, nâng cấp và hỗ trợ.
Các yếu tố ảnh hưởng:
Độ phức tạp của phần mềm.
Tần suất và mức độ nghiêm trọng của các lỗi.
Yêu cầu thay đổi và nâng cấp từ người dùng.
Sự thay đổi của công nghệ và môi trường hoạt động.
Mức độ hỗ trợ kỹ thuật cần thiết.
Đặc điểm:
Thường là chi phí phát sinh liên tục trong suốt vòng đời phần mềm.
Khó dự đoán và lập ngân sách chính xác do tính chất không lường trước của các vấn đề phát sinh.
Thường được tính theo phần trăm của chi phí phát triển ban đầu.

So sánh:
Thông thường, chi phí bảo trì phần mềm có thể vượt quá chi phí phát triển ban đầu trong suốt vòng đời của phần mềm.
Chi phí bảo trì thường chiếm một phần đáng kể trong tổng chi phí vòng đời của phần mềm, đôi khi lên đến 70-80%.
Việc lập kế hoạch và quản lý chi phí bảo trì hiệu quả là rất quan trọng để đảm bảo tính bền vững của phần mềm.

7. **Phân biệt các loại yêu cầu trong phát triển phần mềm (chức năng và phi chức năng).**

Đặc điểm	Yêu cầu chức năng	Yêu cầu phi chức năng
Mô tả	Cái gì phần mềm phải làm	Phần mềm hoạt động như thế nào
Tính chất	Hành động cụ thể	Thuộc tính chất lượng
Kiểm chứng	Dễ	Khó
Liên quan	Tính năng phần mềm	Trải nghiệm và chất lượng

8. **Thảo luận về các mô hình quy trình phát triển phần mềm phổ biến.**

	Mô tả	Ưu điểm	Nhược điểm
Mô hình thác nước	Mô hình này tuân theo một quy trình tuyến tính, tuần tự. Mỗi giai đoạn phải được hoàn thành trước khi giai đoạn tiếp theo bắt đầu.	Đơn giản và dễ hiểu. Phù hợp cho các dự án có yêu cầu rõ ràng và ổn định.	Khó thích ứng với sự thay đổi yêu cầu. Không phù hợp cho các dự án phức tạp hoặc rủi ro cao.

	Các giai đoạn bao gồm: thu thập yêu cầu, thiết kế, triển khai, kiểm thử và bảo trì.		
Mô hình chữ V	Mô hình này mở rộng mô hình thác nước bằng cách thêm các giai đoạn kiểm thử tương ứng với mỗi giai đoạn phát triển. Nhấn mạnh vào việc xác minh và xác nhận chất lượng phần mềm.	Tăng cường kiểm soát chất lượng. Phù hợp cho các dự án yêu cầu độ tin cậy cao.	Khó thích ứng với sự thay đổi yêu cầu. Đòi hỏi nhiều tài liệu và quy trình.
Mô hình xoắn ốc	Mô hình này kết hợp các yếu tố của mô hình thác nước và mô hình lặp. Mỗi vòng xoắn ốc đại diện cho một giai đoạn phát triển, bao gồm lập kế hoạch, phân tích rủi ro, triển khai và đánh giá.	Thích ứng tốt với sự thay đổi yêu cầu. Giảm thiểu rủi ro thông qua việc phân tích và đánh giá liên tục.	Phức tạp và khó quản lý. Đòi hỏi kỹ năng phân tích rủi ro cao.
Mô hình Agile	Mô hình này tập trung vào sự linh hoạt, hợp tác và phản hồi nhanh chóng. Chia dự án thành các chu kỳ ngắn (sprint), mỗi chu kỳ tạo ra một phiên bản phần mềm có thể sử dụng được.	Thích ứng tốt với sự thay đổi yêu cầu. Tăng cường sự hợp tác giữa khách hàng và đội phát triển. Cung cấp phần mềm có giá trị sớm và thường xuyên.	Đòi hỏi sự tham gia tích cực của khách hàng. Khó lập kế hoạch và ước tính chi phí chính xác.
Mô hình tăng cường	Mô hình tăng trưởng phân chia quá trình phát triển thành các giai đoạn nhỏ hơn, dễ	Giảm thiểu rủi ro bằng cách chia nhỏ dự án. Cho phép khách hàng	Đòi hỏi lập kế hoạch và thiết kế kỹ lưỡng. Có thể gặp khó khăn trong việc tích hợp

	quản lý hơn. Mỗi giai đoạn tạo ra một phần chức năng của phần mềm, sau đó được tích hợp vào các phần đã có.	sử dụng các phần chức năng sớm. Dễ dàng thích ứng với thay đổi yêu cầu.	các phần chức năng.

9. **Đề xuất giải pháp giảm thiểu lỗi phần mềm sau khi bàn giao.**

- **Tăng cường giai đoạn kiểm thử:**

- **Kiểm thử toàn diện**:
  - Đảm bảo rằng phần mềm được kiểm thử kỹ lưỡng trên nhiều môi trường và thiết bị khác nhau.
  - Sử dụng các phương pháp kiểm thử khác nhau, bao gồm kiểm thử chức năng, kiểm thử hiệu suất, kiểm thử bảo mật và kiểm thử khả năng sử dụng.
- **Kiểm thử tự động**:
  - Sử dụng các công cụ kiểm thử tự động để tăng hiệu quả và giảm thiểu lỗi do con người gây ra.
  - Tự động hóa các trường hợp kiểm thử lặp đi lặp lại để tiết kiệm thời gian và công sức.
- **Kiểm thử người dùng (UAT)**:
  - Cho phép người dùng cuối kiểm thử phần mềm trong môi trường thực tế trước khi bàn giao.
  - Thu thập phản hồi từ người dùng để xác định và khắc phục các lỗi và vấn đề còn tồn tại.

- **Cải thiện quy trình phát triển phần mềm:**

- **Áp dụng phương pháp Agile**:
  - Sử dụng các phương pháp Agile như Scrum hoặc Kanban để tăng cường sự linh hoạt và khả năng thích ứng với thay đổi.
  - Chia dự án thành các giai đoạn ngắn và thường xuyên đánh giá tiến độ và chất lượng.
- **Thực hiện đánh giá mã nguồn**:
  - Thực hiện đánh giá mã nguồn thường xuyên để phát hiện và khắc phục các lỗi và vấn đề tiềm ẩn.
  - Sử dụng các công cụ phân tích mã nguồn tự động để tăng hiệu quả.
- **Tăng cường giao tiếp và hợp tác**:
  - Khuyến khích sự giao tiếp và hợp tác chặt chẽ giữa các thành viên trong nhóm phát triển.
  - Tổ chức các cuộc họp thường xuyên để trao đổi thông tin và giải quyết vấn đề.

- **Tăng cường tài liệu hóa:**

- **Tài liệu hóa chi tiết**:
    - Tạo ra các tài liệu chi tiết về thiết kế, chức năng và hướng dẫn sử dụng của phần mềm.
    - Đảm bảo rằng tài liệu được cập nhật thường xuyên và dễ hiểu.
- **Tạo video hướng dẫn**:
    - Tạo các video hướng dẫn chi tiết cho người dùng.
    - Lưu trữ video ở những nơi dễ dàng cho khách hàng có thể tìm thấy.

- **Cung cấp hỗ trợ sau bàn giao:**

- **Thiết lập hệ thống hỗ trợ**:
    - Thiết lập một hệ thống hỗ trợ hiệu quả để giải quyết các vấn đề và yêu cầu của khách hàng sau khi bàn giao.
    - Cung cấp nhiều kênh hỗ trợ khác nhau, bao gồm email, điện thoại và trò chuyện trực tuyến.
- **Cung cấp đào tạo cho người dùng**:
    - Cung cấp đào tạo cho người dùng về cách sử dụng phần mềm một cách hiệu quả.
    - Tạo ra các tài liệu hướng dẫn và video đào tạo.

- **Theo dõi và đánh giá:**

- **Thu thập phản hồi từ khách hàng**:
    - Thu thập phản hồi từ khách hàng về chất lượng và hiệu suất của phần mềm.
    - Sử dụng phản hồi để cải thiện phần mềm và quy trình phát triển.
- **Theo dõi và phân tích lỗi**:
    - Theo dõi và phân tích các lỗi phát sinh sau khi bàn giao để xác định nguyên nhân và đưa ra biện pháp khắc phục.
    - Sử dụng các công cụ theo dõi lỗi để tự động hóa quá trình này.

- **10.Vai trò của đội kiểm thử trong quy trình phát triển phần mềm.**
- **Phát hiện và báo cáo lỗi:**

- Đây là vai trò cốt lõi của đội kiểm thử. Họ tiến hành các bài kiểm tra khác nhau để tìm ra các lỗi (bugs) trong phần mềm.
- Họ ghi lại chi tiết các lỗi, bao gồm các bước tái hiện, ảnh chụp màn hình và nhật ký lỗi, để giúp đội phát triển dễ dàng sửa chữa.

- **Đảm bảo chất lượng phần mềm:**

- Đội kiểm thử xác minh rằng phần mềm đáp ứng các yêu cầu chức năng và phi chức năng đã được xác định.
- Họ đảm bảo rằng phần mềm hoạt động ổn định, hiệu quả và an toàn.

- **Đánh giá khả năng sử dụng (usability):**

- Đội kiểm thử đánh giá xem phần mềm có dễ sử dụng và thân thiện với người dùng hay không.
- Họ đưa ra phản hồi về giao diện người dùng, luồng công việc và trải nghiệm người dùng tổng thể.

- **Đóng góp vào việc cải tiến quy trình phát triển:**

- Thông qua việc phân tích các lỗi và vấn đề, đội kiểm thử giúp xác định các điểm yếu trong quy trình phát triển.
- Họ đề xuất các cải tiến để ngăn ngừa các lỗi tương tự xảy ra trong tương lai.

- **Đảm bảo sự hài lòng của khách hàng:**

- Bằng cách đảm bảo chất lượng phần mềm, đội kiểm thử giúp tăng cường sự hài lòng của khách hàng.
- Điều này góp phần xây dựng uy tín và danh tiếng của công ty.

- **Các công việc cụ thể đội kiểm thử cần thực hiện:**

- Lập kế hoạch kiểm thử: Xác định phạm vi, phương pháp và nguồn lực cần thiết cho việc kiểm thử.
- Thiết kế các trường hợp kiểm thử (test cases): Tạo ra các kịch bản kiểm thử chi tiết để kiểm tra các chức năng và tính năng của phần mềm.
- Thực hiện kiểm thử: Tiến hành các bài kiểm tra theo kế hoạch và ghi lại kết quả.
- Báo cáo lỗi: Ghi lại chi tiết các lỗi và vấn đề phát hiện được.
- Kiểm thử hồi quy (regression testing): Kiểm tra lại phần mềm sau khi sửa lỗi để đảm bảo không có lỗi mới phát sinh.
- Kiểm thử hiệu năng (performance testing): Kiểm tra về tốc độ, khả năng chịu tải và sự ổn định của phần mềm.
- Kiểm thử bảo mật (security testing): Kiểm tra các lỗ hổng bảo mật của phần mềm.

CÂU HỎI TÌNH HUỐNG

Tình huống 1:Một công ty phát triển phần mềm quản lý tài chính đã hoàn thành dự án và bàn giao cho khách hàng. Tuy nhiên, sau 2 tháng sử dụng, khách hàng phát hiện ra nhiều lỗi phát sinh khi phần mềm xử lý các giao dịch có giá trị lớn. Hãy đề xuất giải pháp xử lý tình huống này.

Tiếp nhận thông tin
Phân tích lỗi
Tiến hành sửa lỗi và kiểm thử

Tình huống 2: Trong quá trình phát triển phần mềm quản lý bệnh viện, khách hàng yêu cầu bổ sung thêm tính năng quản lý kho thuốc khi dự án đã đi vào giai đoạn kiểm thử. Là trưởng nhóm phát triển, bạn sẽ xử lý yêu cầu này như thế nào?.

Tiếp nhận thông tin
Tiến hành phân tích yêu cầu

Triển khai và kiểm thử
Bàn giao

Tình huống 3: Một nhóm phát triển phần mềm gặp phải vấn đề trễ tiến độ do nhiều thành viên không hiểu rõ yêu cầu của khách hàng.Là trưởng dự án, bạn sẽ làm gì để giải quyết vấn đề này và đảm bảo tiến độ dự án?

Trao đổi lại với khách hàng
Phân tích yêu cầu
Tiến hành triển khai và kiểm thử
Bàn giao

Tình huống 4: Sau khi triển khai phần mềm quản lý thư viện, người dùng phản hồi rằng giao diện khó sử dụng và không thân thiện.Đội phát triển cần làm gì để cải thiện trải nghiệm người dùng?

Tiếp nhận ý kiến
Phân tích yêu cầu
Tiến hành triển khai và kiểm thử
Bàn giao

Tình huống 5: Một dự án phát triển phần mềm đã vượt quá ngân sách dự kiến do thời gian hoàn thành lâu hơn kế hoạch. Là quản lý dự án, bạn sẽ đề xuất những giải pháp nào để hạn chế việc vượt ngân sách trong tương lai?

Lập kế hoạch chi tiết và thực tế
Quản lý tiến độ dự án chặt chẽ
Quản lý thay đổi hiệu quả
Quản lý nguồn lực hợp lý
Kiểm soát chi phí chặt chẽ
Học hỏi kinh nghiệm
Áp dụng các phương pháp quản lý dự án hiện đại

Tình huống 6: Trong quá trình bảo trì phần mềm quản lý khách sạn, một nhân viên phát hiện ra một lỗi nhỏ không ảnh hưởng lớn đến hoạt động. Tuy nhiên, chi phí để sửa lỗi này khá cao. Bạn sẽ quyết định sửa lỗi hay không? Vì sao? Đánh giá mức độ ảnh hưởng của lỗi, nếu lỗi ảnh hưởng đến hệ thống, có rủi ro về bảo mật dữ liệu thì sẽ xem xét sửa lỗi, nếu chỉ ảnh hưởng đến trải nghiệm người dùng sẽ không quyết định chỉnh sửa.

Nếu lỗi không ảnh hưởng đến hệ thống, không có rủi ro về bảo mật dữ liệu, chỉ ảnh hưởng nhỏ đến trải nghiệm người dùng và chi phí sửa lỗi cao thì tôi sẽ không quyết định sửa lỗi.
Lý do:

- Chi phí sửa lỗi cao trong khi lỗi không gây ảnh hưởng lớn đến hoạt động của phần mềm.
- Việc sửa lỗi có thể tiềm ẩn rủi ro phát sinh lỗi mới.
- Thay vì tập trung sửa lỗi nhỏ, tôi sẽ ưu tiên nguồn lực cho các công việc quan trọng hơn như phát triển tính năng mới, cải thiện hiệu suất, nâng cao bảo mật.

Tuy nhiên, tôi sẽ ghi nhận lại lỗi này và theo dõi sát sao. Nếu trong tương lai, lỗi này trở nên nghiêm trọng hơn hoặc chi phí sửa lỗi giảm xuống, tôi sẽ xem xét lại quyết định của mình.

Tình huống 7: Khách hàng yêu cầu đội phát triển phải hoàn thành dự án sớm hơn 1 tháng so với kế hoạch ban đầu. Đội phát triển đang gặp khó khăn về nhân lực và tài nguyên. Bạn sẽ xử lý yêu cầu này như thế nào?

Đánh giá lại kế hoạch dự án và nguồn lực hiện tại.
Điều chỉnh lại kế hoạch sao cho phù hợp với tình hình nguồn lực và tài nguyên hiện tại.

Nếu có thể thì tăng thêm chi phí sản xuất để hoàn thành tiến độ

Tình huống 8: Một công ty phần mềm nhỏ nhận được dự án phát triển ứng dụng di động. Do hạn chế về nguồn lực và kinh nghiệm, công ty đã liên tục thay đổi công nghệ sử dụng trong dự án. Điều này khiến dự án bị kéo dài và chi phí tăng cao. Bạn sẽ đưa ra giải pháp gì để khắc phục?

Xem xét lại yêu cầu của khách hàng

Lập kế hoạch, xem xét công nghệ cần sử dụng

Tăng cường nhân sự đối với công nghệ phù hợp

Triển khai dự án dựa trên kế hoạch đã lập

Tình huống 9: Sau khi bàn giao phần mềm cho khách hàng, đội phát triển phát hiện ra một lỗi bảo mật nghiêm trọng có thể bị hacker khai thác. Là người phụ trách dự án, bạn sẽ giải quyết tình huống này như thế nào?

Đánh giá mức độ nghiêm trọng và phạm vi ảnh hưởng

Báo lại với khách hàng và hoãn thời gian bàn giao

Tiến hành khắc phúc lỗi

Tình huống 10: Dự án phát triển hệ thống quản lý sản xuất đã được triển khai thành công tại nhà máy. Tuy nhiên, do thay đổi quy trình sản xuất, khách hàng yêu cầu sửa đổi phần mềm để phù hợp với quy trình mới. Đội phát triển cần làm gì để đáp ứng yêu cầu này mà không làm ảnh hưởng đến hoạt động sản xuất của khách hàng?

Đánh giá lại yêu cầu khách hàng

Lập kế hoạch theo hướng thay đổi mà khách hàng yêu cầu

Tăng cường nguồn lực để chỉnh sửa theo yêu cầu của khách hàng

CHƯƠNG 2: TIẾN TRÌNH PHẦN MỀM

CÂU HỎI TRẮC NGHIỆM

1. Câu hỏi 1: Workflow nào trong tiến trình phát triển phần mềm chịu trách nhiệm thu thập yêu cầu từ khách hàng?
A. Workflow thiết kế
B. Workflow lấy yêu cầu
C. Workflow kiểm thử
D. Workflow triển khai

2. Câu hỏi 2: Pha nào trong tiến trình thống nhất (Unified Process) tập trung vào việc phân tích rủi ro và xây dựng kiến trúc ban đầu?
A. Pha khởi đầu
B. Pha làm rõ
C. Pha xây dựng
D. Pha chuyển giao

3. Câu hỏi 3: Mô hình CMM mức nào yêu cầu quy trình phát triển phần mềm phải được quản lý định lượng?
A. Mức 2
B. Mức 3
C. Mức 4
D. Mức 5

4. Câu hỏi 4: Các pha trong tiến trình thống nhất bao gồm:
A. Lấy yêu cầu, phân tích, thiết kế, kiểm thử
B. Khởi đầu, làm rõ, xây dựng, chuyển giao
C. Lập kế hoạch, thiết kế, phát triển, bảo trì
D. Phân tích, kiểm thử, triển khai, bảo trì

5. Câu hỏi 5: Trong tiến trình thống nhất, workflow nào thực hiện sau cùng?
A. Workflow phân tích
B. Workflow thiết kế
C. Workflow cài đặt
D. Workflow kiểm thử

6. Câu hỏi 6: Mô hình CMM mức 1 có đặc điểm gì?
A. Quy trình được định nghĩa rõ ràng
B. Quy trình được kiểm soát và đo lường
C. Quy trình không ổn định, phụ thuộc vào cá nhân
D. Quy trình liên tục được tối ưu hóa

7. Câu hỏi 7: Tiến trình thống nhất là một ví dụ của mô hình nào?
A. Mô hình vòng đời thác nước
B. Mô hình lặp và tăng trưởng
C. Mô hình mã nguồn mở
D. Mô hình Agile

8. Câu hỏi 8: Trong mô hình CMM mức 5, quy trình phát triển phần mềm có đặc điểm gì?
A. Quy trình được cải tiến liên tục
B. Quy trình chỉ định nghĩa cơ bản
C. Quy trình chưa được quản lý

D. Quy trình chỉ tập trung vào bảo trì

9. Câu hỏi 9: Workflow thiết kế bao gồm việc thực hiện hoạt động nào?

A. Thu thập yêu cầu

B. Lập kế hoạch dự án

C. Thiết kế kiến trúc và chi tiết hệ thống

D. Kiểm thử tích hợp

10. Câu hỏi 10: CMM viết tắt của cụm từ nào?

A. Configuration Management Model

B. Capability Maturity Model

C. Continuous Maintenance Model

D. Complex Management Model

CÂU HỎI NGẮN

1. Pha khởi đầu trong tiến trình thống nhất là gì?

Đây là giai đoạn ban đầu trong đó phạm vi, mục tiêu và tính khả thi của dự án được xác định.
Các hoạt động chính trong giai đoạn này bao gồm xác định các bên liên quan, xác định các yêu cầu ban đầu, phác thảo kế hoạch dự án và đánh giá rủi ro
Mục tiêu chính là hiểu rõ các yêu cầu cơ bản và đánh giá tính khả thi của dự án.

2. Mục tiêu của workflow lấy yêu cầu là gì?

Mục tiêu: Xác định và ghi nhận tất cả các yêu cầu từ phía khách
hàng.

3. Tiến trình thống nhất gồm bao nhiêu pha chính?

Bao gồm 4 pha chính:
Pha khởi đầu
Pha làm rõ
Pha xây dựng
Pha chuyển giao

4. Sự khác nhau giữa CMM mức 2 và mức 3 là gì?

CMM mức 2:
Quy trình được quản lý ở mức cơ bản.
Các hoạt động như lập kế hoạch, quản lý rủi ro được thực hiện.
Tập trung vào việc thiết lập các chính sách quản lý dự án cơ bản.
Kinh nghiệm với các dự án trước đó được sử dụng để quản lý các dự án mới tương tự.
CMM mức 3:
Quy trình được định nghĩa rõ ràng và nhất quán trong toàn tổ chức.
Các tiêu chuẩn quy trình được xây dựng và áp dụng.
Tài liệu về các hướng dẫn và thủ tục tiêu chuẩn được đưa ra

5. Workflow kiểm thử có nhiệm vụ gì?

Mục tiêu: Đảm bảo rằng phần mềm hoạt động đúng như mong đợi.

6. Mô hình CMM có bao nhiêu mức?

Bao gồm 5 mức khác nhau:
Initial (Ban đầu)
Managed (Quản lý)
Defined (Định nghĩa)
Quantitatively Managed (Quản lý định lượng)
Optimizing (Tối ưu hóa)

7. Khác biệt giữa mô hình thác nước và mô hình lặp là gì?
Mô hình thác nước:
Theo quy trình từng bước nghiêm ngặt: Yêu cầu → Thiết kế → Thực hiện → Kiểm tra → Triển khai → Bảo trì.
Không có sự chồng chéo của các giai đoạn.
Những thay đổi trong các giai đoạn sau là tốn kém và khó thực hiện.
Phù hợp nhất cho các dự án với các yêu cầu được xác định rõ ràng và ổn định.
Mô hình lặp:
Phát triển được thực hiện trong các chu kỳ lặp đi lặp lại (lặp lại).
Mỗi lần lặp lại tinh chỉnh và cải thiện cái trước dựa trên phản hồi.
Hệ thống phát triển theo thời gian và phản hồi được kết hợp liên tục.
Thích hợp cho các dự án trong đó các yêu cầu không rõ ràng hoặc dự kiến sẽ thay đổi.

8. Tiến trình thống nhất có phải là mô hình lặp không?
Tiến trình thống nhất đi theo cách tiếp cận lặp đi lặp lại vì sự phát triển được chia thành nhiều chu kỳ (lặp lại), trong đó mỗi lần lặp lại sửa đổi và tinh chỉnh hệ thống.
Mỗi lần lặp lại dẫn đến một phiên bản hoàn chỉnh hơn của hệ thống, cải thiện chức năng, hiệu suất và thiết kế.
Các nhà phát triển có thể xem lại và cải thiện công việc trước đây, cho phép linh hoạt trong các yêu cầu và điều chỉnh thiết kế.

9. Mục đích của workflow thiết kế là gì?

Mục tiêu: Thiết kế chi tiết các thành phần phần mềm dựa trên kết quả
phân tích.

10.CMM mức 5 tập trung vào điều gì?

Quy trình liên tục được cải tiến dựa trên phản hồi và dữ liệu.
Mục tiêu là đạt được sự hoàn hảo trong phát triển phần mềm.

CÂU HỎI THẢO LUẬN NHÓM

1. Thảo luận về vai trò của từng workflow trong tiến trình phát triển phần mềm.
Workflow lấy yêu cầu: Xác định và ghi lại các yêu cầu của người dùng và các bên liên quan.
Workflow phân tích và thiết kế: Chuyển đổi các yêu cầu thành một thiết kế chi tiết, bao gồm kiến trúc hệ thống, cơ sở dữ liệu và giao diện người dùng.
Workflow lập trình: Hiện thực hóa thiết kế thành mã nguồn.

Workflow kiểm thử: Đảm bảo chất lượng của phần mềm bằng cách tìm kiếm và sửa lỗi.
Workflow triển khai: Đưa phần mềm vào sử dụng.
Workflow bảo trì: Sửa lỗi và cải tiến phần mềm sau khi triển khai.

2. Phân biệt mô hình vòng đời thác nước và tiến trình thống nhất.

Đặc điểm	**Mô hình vòng đời thác nước**	**Tiến trình thống nhất**
Các giai đoạn	Tuần tự, mỗi giai đoạn chỉ được thực hiện một lần	Lặp đi lặp lại, các giai đoạn có thể được thực hiện song song
Thay đổi yêu cầu	Khó thay đổi sau khi giai đoạn lấy yêu cầu kết thúc	Dễ dàng thay đổi hơn
Rủi ro	Rủi ro cao nếu có lỗi ở giai đoạn đầu	Rủi ro thấp hơn
Phù hợp	Dự án nhỏ, yêu cầu rõ ràng	Dự án lớn, phức tạp, yêu cầu có thể thay đổi

3. Thảo luận về các ưu và nhược điểm của mô hình lặp và tăng trưởng.
Ưu điểm:
- Giảm thiểu rủi ro
- Dễ dàng thích ứng với thay đổi
- Khách hàng có thể tham gia vào quá trình phát triển

Nhược điểm:
- Đòi hỏi nhiều thời gian và nguồn lực
- Khó quản lý nếu dự án quá phức tạp

4. Vì sao mô hình CMM được sử dụng rộng rãi trong quản lý chất lượng phần mềm?
Mô hình CMM (Capability Maturity Model) được sử dụng rộng rãi vì nó cung cấp một khuôn khổ để đánh giá và cải tiến quy trình phát triển phần mềm. CMM giúp các tổ chức:
- Nâng cao chất lượng sản phẩm
- Giảm chi phí
- Tăng năng suất

5. Thảo luận về các khó khăn khi áp dụng mô hình CMM trong thực tế
- Đòi hỏi sự cam kết của lãnh đạo
- Tốn kém thời gian và nguồn lực
- Khó đo lường hiệu quả

6. Đề xuất các giải pháp để cải tiến quy trình phát triển phần mềm
- Sử dụng các công cụ và công nghệ hỗ trợ
- Áp dụng các phương pháp luận Agile
- Tăng cường đào tạo cho nhân viên
- Thiết lập hệ thống quản lý chất lượng

7. Phân tích ưu điểm của việc áp dụng tiến trình thống nhất trong các dự án lớn
- Quản lý rủi ro tốt hơn
- Dễ dàng thích ứng với thay đổi

- Tăng cường sự tham gia của khách hàng

8. Thảo luận về sự cần thiết của việc kiểm thử trong từng pha của tiến trình thống nhất

Kiểm thử là cần thiết trong từng pha của tiến trình thống nhất để:

- Phát hiện lỗi sớm
- Đảm bảo chất lượng sản phẩm
- Giảm chi phí sửa lỗi

9. So sánh giữa mô hình CMM mức 4 và mức 5

Đặc điểm	CMM mức 4 (Quản lý)	CMM mức 5 (Tối ưu hóa)
Trọng tâm	Quản lý quy trình	Tối ưu hóa quy trình
Đo lường	Sử dụng các số liệu để đo lường hiệu quả	Sử dụng các số liệu để cải tiến quy trình
Cải tiến	Cải tiến dựa trên phân tích dữ liệu	Cải tiến liên tục

Xuất sang Trang tính

10. Đề xuất cách tổ chức hoạt động nhóm trong workflow lấy yêu cầu

- Xác định rõ vai trò và trách nhiệm của từng thành viên
- Sử dụng các công cụ hỗ trợ giao tiếp và cộng tác
- Tổ chức các buổi họp để thu thập và phân tích yêu cầu
- Ưu tiên các yêu cầu quan trọng nhất

CÂU HỎI TÌNH HUỐNG

1. Một công ty phát triển phần mềm gặp khó khăn khi yêu cầu của khách hàng liên tục thay đổi trong pha xây dựng. Đội phát triển nên làm gì để giải quyết vấn đề này?

Xây dựng quy trình quản lý thay đổi yêu cầu

Thiết lập một quy trình tiếp nhận, đánh giá và phê duyệt thay đổi yêu cầu.

Xác định rõ ràng những thay đổi nào có thể thực hiện ngay, thay đổi nào cần xem xét sau.

Ghi lại tất cả các thay đổi để tránh tình trạng lặp lại yêu cầu hoặc hiểu sai.

Thỏa thuận về phạm vi dự án ngay từ đầu

Ký kết hợp đồng phát triển phần mềm có quy định về giới hạn phạm vi (Scope).

Nếu khách hàng yêu cầu thay đổi quá lớn, có thể đàm phán về chi phí và thời gian bổ sung.

Giao tiếp hiệu quả với khách hàng

Duy trì kênh liên lạc thường xuyên để khách hàng hiểu rõ tác động của việc thay đổi yêu cầu.

Trình bày các hậu quả của thay đổi như kéo dài thời gian, tăng chi phí hoặc ảnh hưởng đến các tính năng khác.

2. Trong pha chuyển giao của tiến trình thống nhất, khách hàng yêu cầu bổ sung thêm tính năng mới. Đội phát triển nên xử lý ra sao?

- **Đánh giá tác động của yêu cầu**
  - Kiểm tra xem yêu cầu có ảnh hưởng đến hệ thống hiện tại không.
  - Đánh giá mức độ phức tạp: Liệu tính năng mới có thể triển khai nhanh chóng hay đòi hỏi thay đổi lớn?
  - Xem xét tác động đến thời gian bàn giao, chi phí và chất lượng phần mềm.
- **Trao đổi với khách hàng về lựa chọn xử lý**
  - Nếu tính năng đơn giản và có thể thêm ngay mà không ảnh hưởng lớn, có thể thực hiện ngay trong phiên bản hiện tại.
  - Nếu tính năng phức tạp, đề xuất đưa vào một phiên bản cập nhật sau khi triển khai chính thức.
  - Xác định rõ các chi phí bổ sung và điều chỉnh hợp đồng nếu cần.
- **Ưu tiên tính ổn định của hệ thống**
  - Trong pha chuyển giao, mục tiêu chính là đảm bảo phần mềm hoạt động ổn định, không có lỗi lớn.
  - Việc thêm tính năng mới có thể gây lỗi hoặc làm chậm tiến trình kiểm thử và triển khai.
  - Do đó, nếu tính năng mới có thể gây rủi ro, đội phát triển nên ưu tiên bàn giao phiên bản hiện tại trước, sau đó phát triển thêm trong một đợt cập nhật riêng.

3. Dự án phát triển phần mềm bị trễ tiến độ do lỗi phát sinh liên tục trong quá trình kiểm thử. Là trưởng dự án, bạn sẽ làm gì?

- **Phân tích nguyên nhân gốc rễ (Root Cause Analysis)**
  - Thiết kế phần mềm có vấn đề? → Kiểm tra kiến trúc, logic code.
  - Yêu cầu không rõ ràng? → So sánh code với tài liệu yêu cầu.
  - Chất lượng code kém? → Kiểm tra cách lập trình viên viết code, có tuân thủ coding standard không.
  - Kiểm thử chưa hiệu quả? → Xem lại quy trình test, có thiếu test case nào không
- **Ưu tiên sửa lỗi nghiêm trọng trước**
  - Xác định lỗi nào ảnh hưởng lớn đến hệ thống (Critical, High) và tập trung sửa trước.
  - Đặt mức độ ưu tiên theo mức ảnh hưởng đến người dùng.
  - Nếu có lỗi nhỏ nhưng không ảnh hưởng nhiều, có thể trì hoãn và sửa trong bản cập nhật sau.
- **Cải thiện quy trình kiểm thử**
  - Tự động hóa kiểm thử (Test Automation): Nếu chỉ dùng kiểm thử thủ công, hãy bổ sung test tự động với các công cụ như Selenium, JUnit, PyTest.
  - Áp dụng kiểm thử hồi quy (Regression Testing): Để tránh lỗi cũ tái xuất hiện khi sửa lỗi mới.
  - Bổ sung kiểm thử đơn vị (Unit Test): Nếu thiếu Unit Test, hãy yêu cầu lập trình viên bổ sung để phát hiện lỗi sớm hơn.

4. Trong workflow thiết kế, kiến trúc sư phần mềm muốn thay đổi thiết kế ban đầu để cải thiện hiệu suất. Đội phát triển nên xử lý thế nào?

- **Đánh giá tác động của thay đổi**
  - Hiệu suất hiện tại có thực sự là vấn đề lớn không?
  - Thay đổi có ảnh hưởng đến các module khác hay không?
  - Có rủi ro phát sinh lỗi mới khi thay đổi không?
  - Thay đổi có làm trễ tiến độ không?
- **Thảo luận với các bên liên quan**
  - Lập trình viên: Xem xét tính khả thi của thay đổi.
  - Tester: Đánh giá ảnh hưởng của thay đổi đến quá trình kiểm thử.
  - Project Manager: Xác định xem thay đổi có làm trễ tiến độ không.
  - Khách hàng (nếu cần): Nếu thay đổi lớn, cần đảm bảo rằng khách hàng đồng ý với sự điều chỉnh này.
- **Lập kế hoạch triển khai thay đổi**
  - Tạo thiết kế mới, đảm bảo rằng nó giải quyết được vấn đề hiệu suất.
  - Kiểm tra lại các yêu cầu phần mềm để đảm bảo thay đổi không vi phạm yêu cầu ban đầu.
  - Cập nhật tài liệu thiết kế để tránh nhầm lẫn về sau.
  - Lập kế hoạch thử nghiệm để xác nhận rằng thay đổi thực sự cải thiện hiệu suất mà không gây lỗi mới.
- **Thử nghiệm thay đổi trước khi áp dụng rộng rãi**
  - Xây dựng nguyên mẫu (Prototype) hoặc thử nghiệm trên môi trường giả lập để kiểm tra hiệu quả của thay đổi.
  - Chạy Benchmark để so sánh hiệu suất trước và sau thay đổi.
  - Thực hiện kiểm thử hồi quy (Regression Testing) để đảm bảo không làm hỏng các chức năng hiện có.
- **Tiến hành triển khai thay đổi**
  - Nếu mọi thứ ổn, áp dụng thay đổi vào dự án chính.
  - Nếu có rủi ro cao, có thể triển khai dần dần (Incremental Changes) thay vì thay đổi toàn bộ một lúc.

5. **Khách hàng yêu cầu rút ngắn thời gian phát triển dự án mà không thay đổi yêu cầu. Đội phát triển nên phản ứng ra sao?**

- **Đánh giá tính khả thi của yêu cầu**
  - Dự án hiện tại đang ở giai đoạn nào?
  - Rút ngắn bao nhiêu thời gian? (Vài tuần có thể khả thi, nhưng rút ngắn vài tháng thì khó)
  - Có đủ nhân lực không?
  - Giảm thời gian có ảnh hưởng đến chất lượng không?
- **Trao đổi với khách hàng về các lựa chọn**
  - Nếu khách hàng muốn nhanh hơn, có thể yêu cầu họ cung cấp thêm ngân sách để thuê thêm nhân lực.

- Nếu không thể tăng ngân sách, có thể đề xuất cắt giảm một số phần không quan trọng (ví dụ: giao diện phức tạp, tính năng phụ) để đảm bảo thời gian.
- Đưa ra báo cáo rủi ro: Nếu cố rút ngắn mà không có sự điều chỉnh hợp lý, chất lượng có thể giảm.

6. Một công ty nhỏ muốn áp dụng mô hình CMM nhưng gặp khó khăn do thiếu nguồn lực. Hãy đề xuất giải pháp.Bắt đầu với các mức CMM phù hợp

- **Bắt đầu với các mức CMM phù hợp**
  - Tập trung vào Mức 2 (Managed) trước, vì đây là mức quan trọng giúp chuẩn hóa quy trình cơ bản.
  - Khi có đủ nguồn lực, dần nâng cấp lên Mức 3 (Defined) để cải thiện tính nhất quán trong quy trình.
  - Không cần vội đạt đến Mức 4 (Quantitatively Managed) hay Mức 5 (Optimizing) ngay lập tức, vì chúng đòi hỏi đầu tư lớn.
- **Sử dụng công cụ hỗ trợ thay vì đầu tư lớn vào tài nguyên mới**
  - **Quản lý dự án**: Trello, Jira (miễn phí cho nhóm nhỏ).
  - **Quản lý mã nguồn**: GitHub, GitLab, Bitbucket.
  - **Tự động hóa kiểm thử**: Selenium, JUnit, PyTest.
  - **Quản lý tài liệu và quy trình**: Confluence, Notion.

7. Trong workflow lấy yêu cầu, khách hàng cung cấp thông tin không rõ ràng. Đội phát triển cần làm gì?

- **Xác định những điểm chưa rõ và yêu cầu làm rõ**
  - Xem lại tài liệu yêu cầu, xác định các phần mơ hồ, thiếu thông tin hoặc mâu thuẫn.
  - Gửi danh sách các câu hỏi cụ thể để khách hàng làm rõ.
- **Tổ chức buổi họp với khách hàng**
  - Sắp xếp buổi họp trực tiếp hoặc online để giải thích những điểm chưa rõ.
  - Sử dụng phương pháp hỏi đáp (Q&A) để thu thập thêm thông tin.
  - Ghi lại biên bản họp (meeting notes) để tránh nhầm lẫn sau này.
- **Sử dụng mô hình trực quan (Wireframe, Prototype, User Story)**
  - Nếu yêu cầu quá chung chung, hãy tạo wireframe (bản phác thảo UI) hoặc prototype (mô hình thử nghiệm) để khách hàng dễ hình dung.
  - Viết User Story (kịch bản người dùng) để làm rõ tình huống sử dụng.

8. Một dự án gặp rủi ro cao trong pha khởi đầu do thiếu tài liệu yêu cầu rõ ràng. Đội phát triển nên làm gì?

- **Xác định và thu thập yêu cầu từ khách hàng**
  - Tổ chức các buổi họp với khách hàng để làm rõ mục tiêu dự án.
  - Sử dụng phương pháp phỏng vấn, khảo sát hoặc workshop để thu thập thông tin.
  - Nếu khách hàng chưa xác định rõ yêu cầu, đề xuất các giải pháp khả thi để họ lựa chọn.
- **Viết tài liệu yêu cầu phần mềm (SRS)**
  - Ghi lại yêu cầu dưới dạng tài liệu có cấu trúc, bao gồm:
    - Mô tả tổng quan về dự án
    - Chức năng chính của hệ thống
    - Yêu cầu phi chức năng (hiệu suất, bảo mật, khả năng mở rộng)
    - Các ràng buộc kỹ thuật và pháp lý
  - Định nghĩa rõ ràng tiêu chí hoàn thành cho từng yêu cầu để tránh hiểu sai.
- **Sử dụng mô hình trực quan để làm rõ yêu cầu**
  - Dùng sơ đồ UML như Use Case Diagram, Sequence Diagram để minh họa luồng hệ thống.
  - Xây dựng wireframe hoặc prototype để giúp khách hàng hình dung giao diện phần mềm.
  - Viết User Story nếu nhóm áp dụng Agile để mô tả từng tình huống sử dụng của người dùng.
- **Áp dụng phương pháp quản lý rủi ro**
  - Xác định các rủi ro chính liên quan đến yêu cầu không rõ ràng.
  - Đánh giá mức độ ảnh hưởng và xác suất xảy ra của từng rủi ro.
  - Lập kế hoạch giảm thiểu rủi ro bằng cách kiểm soát phạm vi dự án và có quy trình phê duyệt thay đổi yêu cầu.
- **Thiết lập quy trình kiểm soát thay đổi yêu cầu**
  - Nếu yêu cầu chưa rõ, đưa vào danh sách backlog và làm rõ trong các giai đoạn tiếp theo.
  - Định nghĩa quy trình tiếp nhận và phê duyệt thay đổi yêu cầu để tránh ảnh hưởng đến tiến độ.
  - Sử dụng Change Request Form để ghi nhận thay đổi và đánh giá tác động trước khi thực hiện.
- **Xác nhận yêu cầu với khách hàng trước khi bắt đầu phát triển**
  - Gửi tài liệu yêu cầu và các mô hình trực quan để khách hàng xem xét và phản hồi.
  - Thống nhất phạm vi dự án bằng cách yêu cầu khách hàng ký xác nhận tài liệu.
  - Nếu sử dụng phương pháp Agile, lập Product Backlog và ưu tiên các tính năng cốt lõi trước.

9. Dự án phần mềm lớn có nhiều nhóm phát triển ở các địa điểm khác nhau. Làm thế nào để đảm bảo các nhóm phối hợp hiệu

- **Thiết lập quy trình làm việc rõ ràng**

    - Chuẩn hóa quy trình phát triển phần mềm để đảm bảo tất cả các nhóm tuân theo một mô hình thống nhất như Agile, Scrum hoặc SAFe (Scaled Agile Framework).
    - Xác định rõ trách nhiệm của từng nhóm, cách giao tiếp và cơ chế báo cáo tiến độ.
    - Sử dụng công cụ quản lý công việc như Jira, Trello, hoặc Azure DevOps để theo dõi trạng thái công việc theo thời gian thực.
- **Sử dụng công cụ hỗ trợ làm việc từ xa**
    - Họp trực tuyến định kỳ: Tổ chức các cuộc họp qua Zoom, Google Meet hoặc Microsoft Teams để cập nhật tiến độ, thảo luận vấn đề và giải quyết khó khăn.
    - Chia sẻ tài liệu tập trung: Dùng Google Drive, Confluence hoặc Notion để lưu trữ tài liệu kỹ thuật, tài liệu yêu cầu và hướng dẫn triển khai.
    - Công cụ giao tiếp nhanh: Slack, Microsoft Teams hoặc Mattermost giúp các nhóm trao đổi thông tin nhanh chóng và hiệu quả.
- **Quản lý mã nguồn và kiểm soát phiên bản chặt chẽ**
    - Sử dụng Git (GitHub, GitLab, Bitbucket) với chiến lược quản lý nhánh (branching strategy) rõ ràng như Git Flow hoặc Trunk-based Development.
    - Áp dụng Code Review thông qua Pull Request (PR) để đảm bảo chất lượng mã nguồn và phát hiện lỗi sớm.
    - Sử dụng CI/CD (Continuous Integration/Continuous Deployment) để tự động hóa kiểm thử và triển khai, giúp giảm xung đột giữa các nhóm khi tích hợp sản phẩm.
- **Định nghĩa rõ ràng API và kiến trúc hệ thống**
    - Nếu các nhóm làm việc trên các module khác nhau, cần xây dựng tài liệu API chi tiết để đảm bảo khả năng tương tác giữa các hệ thống.
    - Áp dụng Microservices Architecture hoặc Modular Monolith để phân chia hệ thống thành các phần độc lập, giảm sự phụ thuộc giữa các nhóm.
    - Sử dụng OpenAPI (Swagger) để mô tả API một cách rõ ràng, giúp các nhóm dễ dàng hiểu và tích hợp.
- **Xây dựng văn hóa giao tiếp và hợp tác giữa các nhóm**
    - Họp daily/weekly sync: Tạo điều kiện cho các nhóm cập nhật công việc và trao đổi khó khăn.
    - Xây dựng quy tắc giao tiếp: Đảm bảo thông tin quan trọng được truyền đạt đầy đủ, tránh hiểu sai do khác biệt múi giờ hoặc văn hóa làm việc.
    - Định kỳ tổ chức workshop hoặc retrospective meeting để đánh giá hiệu quả làm việc nhóm và cải thiện quy trình.
- **Kiểm soát tiến độ và chất lượng sản phẩm**
    - Sử dụng OKR (Objectives and Key Results) hoặc KPI (Key Performance Indicators) để đo lường hiệu suất của từng nhóm.
    - Áp dụng Test Automation để đảm bảo chất lượng phần mềm đồng nhất giữa các nhóm.
    - Thực hiện kiểm thử tích hợp thường xuyên để đảm bảo các module hoạt động tốt khi ghép nối.

10. Một công ty phát triển phần mềm gặp khó khăn trong việc quản lý quy trình do không có chuẩn hóa. Hãy đề xuất giải pháp.

- **Xác định mô hình phát triển phù hợp**
  - **Agile (Scrum, Kanban)**: Phù hợp với các dự án cần sự linh hoạt, có yêu cầu thay đổi thường xuyên.
  - **Waterfall**: Phù hợp với các dự án có yêu cầu cố định ngay từ đầu.
  - **Hybrid Model**: Kết hợp giữa Agile và Waterfall để tận dụng điểm mạnh của cả hai mô hình.
- **Xây dựng quy trình phát triển phần mềm chuẩn (SDLC - Software Development Life Cycle)**
  - **Lấy yêu cầu**: Xác định yêu cầu từ khách hàng, viết tài liệu SRS (Software Requirement Specification).
  - **Thiết kế**: Tạo kiến trúc hệ thống, thiết kế cơ sở dữ liệu, lập kế hoạch phát triển.
  - **Lập trình**: Mã hóa theo tiêu chuẩn, tuân thủ quy ước đặt tên và cấu trúc dự án.
  - **Kiểm thử**: Áp dụng kiểm thử tự động và kiểm thử thủ công để đảm bảo chất lượng.
  - **Triển khai**: Sử dụng CI/CD để tự động hóa quy trình triển khai phần mềm.
  - **Bảo trì & cải tiến**: Theo dõi hiệu suất và cập nhật phiên bản mới.
- **Chuẩn hóa quy trình quản lý công việc**
  - Áp dụng **công cụ quản lý dự án** như Jira, Trello, hoặc ClickUp để theo dõi công việc.
  - Sử dụng **mô hình quản lý công việc** như Scrum (Sprint, Backlog, Daily Standup) hoặc Kanban (Task Board).
  - Xây dựng **báo cáo tiến độ định kỳ**, đánh giá hiệu suất nhóm để tối ưu quy trình làm việc.
- **Thiết lập tiêu chuẩn chất lượng phần mềm**
  - Áp dụng **quy tắc coding** theo chuẩn như **Clean Code**, **SOLID**, **Design Patterns**.
  - Thực hiện **Code Review** để đảm bảo chất lượng mã nguồn trước khi tích hợp.
  - Sử dụng **kiểm thử tự động** (unit test, integration test, end-to-end test) để phát hiện lỗi sớm.

    - Áp dụng **CI/CD** để tự động hóa build, kiểm thử và triển khai phần mềm.
- **Xây dựng tài liệu quy trình và hướng dẫn nội bộ**
    - Viết tài liệu hướng dẫn về quy trình phát triển, coding convention, quy tắc review code.
    - Sử dụng **Confluence, Notion hoặc Google Docs** để lưu trữ tài liệu một cách tập trung.
    - Định kỳ tổ chức **workshop nội bộ** để đào tạo nhân viên về quy trình mới.
- **Cải tiến liên tục và quản lý thay đổi**
    - **Lấy phản hồi** từ các nhóm phát triển để điều chỉnh quy trình cho phù hợp.
    - Áp dụng **Retrospective Meeting** để đánh giá hiệu quả sau mỗi chu kỳ phát triển.
    - Thực hiện **đánh giá định kỳ** và cập nhật quy trình để cải thiện hiệu suất.

CHƯƠNG 3: MỘT SỐ MÔ HÌNH VÒNG ĐỜI PHÁT TRIỂN

CÂU HỎI TRẮC NGHIỆM

1. Câu hỏi 1: Pha nào trong mô hình lý thuyết vòng đời phát triển phần mềm chịu trách nhiệm chuyển đổi yêu
cầu thành đặc tả kỹ thuật?
A. Pha thiết kế
B. Pha lấy yêu cầu
C. Pha phân tích
D. Pha cài đặt
2. Câu hỏi 2: Mô hình vòng đời nào phát triển phần mềm bằng cách tạo các phiên bản nhỏ và tăng dần tính
năng?
A. Mô hình thác nước
B. Mô hình lặp và tăng trưởng
C. Mô hình bản mẫu nhanh
D. Mô hình tiến trình linh hoạt
3. Câu hỏi 3: Pha bảo trì trong vòng đời phát triển phần mềm bao gồm hoạt động nào?
A. Viết mã nguồn
B. Sửa lỗi và cập nhật tính năng mới
C. Gỡ bỏ phần mềm
D. Phân tích yêu cầu
4. Câu hỏi 4: Mô hình thác nước phù hợp nhất với loại dự án nào?
A. Dự án nhỏ, đơn giản
B. Dự án có yêu cầu rõ ràng và ít thay đổi
C. Dự án yêu cầu linh hoạt cao
D. Dự án mã nguồn mở
5. Câu hỏi 5: Trong mô hình xoắn ốc, mỗi vòng xoắn tương ứng với:
A. Một pha kiểm thử
B. Một chu kỳ lặp của toàn bộ quy trình phát triển
C. Một phiên bản phần mềm nhỏ
D. Một lần phân tích rủi ro
6. Câu hỏi 6: Điểm yếu lớn nhất của mô hình xây và sửa là gì?
A. Khó phát triển nhanh
B. Tốn nhiều chi phí bảo trì
C. Khó kiểm soát chất lượng
D. Không phù hợp với dự án nhỏ
7. Câu hỏi 7: Mô hình nào tập trung vào việc tạo các nguyên mẫu nhanh để thu thập phản hồi từ khách hàng?
A. Mô hình lặp và tăng trưởng
B. Mô hình bản mẫu nhanh
C. Mô hình xoắn ốc
D. Mô hình tiến trình linh hoạt
8. Câu hỏi 8: Pha nào kết thúc vòng đời phát triển phần mềm?

A. Pha bảo trì
B. Pha cài đặt
C. Pha giải thể
D. Pha thiết kế
9. Câu hỏi 9: Điểm khác biệt chính giữa mô hình lặp và tăng trưởng với mô hình thác nước là gì?
A. Mô hình thác nước lặp lại nhiều lần
B. Mô hình lặp và tăng trưởng phát triển theo từng đợt nhỏ
C. Mô hình lặp và tăng trưởng không có giai đoạn kiểm thử
D. Mô hình thác nước không có giai đoạn bảo trì
10. Câu hỏi 10: Mô hình nào có khả năng thích nghi tốt nhất với sự thay đổi của yêu cầu khách hàng?
A. Mô hình thác nước
B. Mô hình xoắn ốc
C. Mô hình xây và sửa
D. Mô hình tiến trình linh hoạt

CÂU HỎI NGẮN

1.Pha lấy yêu cầu là gì và có vai trò gì trong vòng đời phát triển phần mềm?
Thu thập và phân tích các yêu cầu từ khách hàng để hiểu rõ những gì phần mềm cần thực hiện.
Vai trò:
Tổ chức phỏng vấn, khảo sát khách hàng.
Xác định các yêu cầu chức năng và phi chức năng.
Lập tài liệu yêu cầu phần mềm (SRS).
2.Mô hình thác nước hoạt động như thế nào?
Đặc điểm: Phát triển tuyến tính, từng pha hoàn thành trước khi chuyển sang pha tiếp theo.
Ưu điểm: Dễ quản lý và theo dõi tiến độ.
Nhược điểm: Khó thích nghi khi yêu cầu thay đổi
3.Mô hình lặp và tăng trưởng khác gì so với mô hình thác nước?
Mô hình lặp và tăng trưởng:
Phát triển phần mềm theo từng đợt lặp lại, mỗi đợt cải tiến dần phần mềm.
Mô hình thác nước:
Phát triển tuyến tính, từng pha hoàn thành trước khi chuyển sang pha tiếp theo. Để giai đoạn tiếp theo thực hiện, bắt buộc giai đoạn trước đó phải được hoàn thiện.
4.Mục tiêu của pha bảo trì là gì?
Khắc phục lỗi và nâng cấp phần mềm sau khi triển khai.
5.Mô hình xây và sửa có nhược điểm gì?
Phát triển nhanh, không có quy trình rõ ràng, dễ phát sinh lỗi khi mở rộng.
6.Mô hình bản mẫu nhanh là gì?
Tạo các nguyên mẫu nhanh chóng để thu thập phản hồi từ khách hàng

7. Pha giải thể là gì?

Pha giải thể trong chu kỳ phát triển phần mềm là giai đoạn mà một hệ thống phần mềm bị ngừng hoạt động và thay thế bằng một hệ thống mới hoặc các thành phần của nó bị tái sử dụng cho các mục đích khác. Quá trình giải thể thường bao gồm việc loại bỏ dữ liệu nhạy cảm, ngừng hỗ trợ kỹ thuật và dừng hoàn toàn mọi hoạt động liên quan đến hệ thống cũ.

8. Mô hình xoắn ốc là gì?

Mô hình xoắn ốc là một mô hình phát triển phần mềm trong đó dự án được chia thành các giai đoạn lặp lại, mỗi giai đoạn gồm bốn hoạt động chính: lập kế hoạch, phân tích rủi ro, triển khai, và đánh giá. Mô hình này kết hợp các yếu tố của mô hình thác nước và mô hình lặp lại, cho phép nhóm phát triển linh hoạt điều chỉnh và cải thiện sản phẩm thông qua mỗi vòng lặp.

9. Tại sao mô hình tiến trình linh hoạt được đánh giá cao?

Mô hình tiến trình linh hoạt (Agile) được đánh giá cao vì nó tập trung vào việc phát triển phần mềm nhanh chóng và đáp ứng nhanh chóng với sự thay đổi yêu cầu từ khách hàng. Một số lợi ích chính của mô hình này bao gồm:

- Tăng khả năng đáp ứng: Đội phát triển có thể nhanh chóng điều chỉnh để thích ứng với các thay đổi.
- Tương tác liên tục với khách hàng: Khách hàng có thể liên tục đưa ra phản hồi và tham gia vào quá trình phát triển.
- Phát hành sản phẩm thường xuyên: Các phiên bản phần mềm được phát hành thường xuyên giúp người dùng có trải nghiệm sớm và góp ý.

10. Điểm khác biệt chính giữa mô hình mã nguồn mở và các mô hình khác là gì?

Mô hình mã nguồn mở (Open Source) có một số điểm khác biệt chính so với các mô hình phát triển phần mềm khác:

- Sự minh bạch: Mã nguồn của phần mềm được công khai và bất kỳ ai cũng có thể truy cập, sử dụng, sửa đổi và phân phối lại.
- Sự cộng tác: Mô hình này khuyến khích sự tham gia của cộng đồng phát triển toàn cầu, giúp cải tiến và phát triển phần mềm nhanh chóng.
- Chi phí thấp hơn: Phần mềm mã nguồn mở thường không yêu cầu phí giấy phép sử dụng, giúp giảm chi phí cho người dùng và doanh nghiệp.

CÂU HỎI THẢO LUẬN NHÓM

So sánh ưu và nhược điểm của mô hình thác nước và mô hình xoắn ốc:

- **Mô hình thác nước:**
  - *Ưu điểm*: Quy trình đơn giản, rõ ràng và dễ hiểu; dễ quản lý và kiểm soát.
  - *Nhược điểm*: Khó điều chỉnh khi yêu cầu thay đổi; không linh hoạt; rủi ro lỗi lớn ở cuối giai đoạn.
- **Mô hình xoắn ốc:**
  - *Ưu điểm*: Tích hợp quản lý rủi ro; cho phép lặp lại và cải tiến liên tục; dễ dàng thay đổi yêu cầu.

    - *Nhược điểm*: Phức tạp hơn trong việc quản lý; có thể đòi hỏi chi phí và thời gian phát triển cao hơn.

2. Thảo luận về tình huống thực tế có thể áp dụng mô hình lặp và tăng trưởng:

- **Mô hình lặp và tăng trưởng** thích hợp với các dự án có yêu cầu không rõ ràng hoặc dự kiến sẽ thay đổi. Một ví dụ thực tế là phát triển các ứng dụng di động, nơi người dùng thường xuyên yêu cầu các tính năng mới hoặc cải tiến dựa trên phản hồi.

3. Tại sao mô hình xây và sửa không phù hợp với các dự án lớn?

- Mô hình xây và sửa thường không phù hợp với các dự án lớn vì nó thiếu quy trình rõ ràng, dễ dẫn đến việc làm lại nhiều lần và khó quản lý. Điều này có thể làm tăng chi phí và thời gian phát triển, cùng với nguy cơ sản phẩm cuối cùng không đạt yêu cầu.

4. So sánh giữa mô hình bản mẫu nhanh và mô hình tiến trình linh hoạt:

- **Mô hình bản mẫu nhanh**:
    - *Ưu điểm*: Cho phép thử nghiệm và phản hồi sớm; giảm rủi ro và chi phí phát triển.
    - *Nhược điểm*: Không tập trung vào tài liệu; dễ dẫn đến việc bỏ qua các yêu cầu kỹ thuật chi tiết.
- **Mô hình tiến trình linh hoạt**:
    - *Ưu điểm*: Linh hoạt, dễ điều chỉnh theo yêu cầu khách hàng; tương tác liên tục với khách hàng.
    - *Nhược điểm*: Cần đội ngũ có kỹ năng cao; khó quản lý trong các dự án lớn và phức tạp.

5. Phân tích vai trò của quản lý rủi ro trong mô hình xoắn ốc:

- Quản lý rủi ro là một phần quan trọng của mô hình xoắn ốc. Từng giai đoạn lặp lại của mô hình xoắn ốc đều bắt đầu bằng việc phân tích rủi ro, từ đó giúp xác định và giải quyết các vấn đề tiềm ẩn sớm. Điều này làm giảm nguy cơ thất bại và tăng khả năng thành công của dự án.

6. Khi nào nên sử dụng mô hình thác nước thay vì mô hình tiến trình linh hoạt?

- Mô hình thác nước nên được sử dụng khi yêu cầu dự án rõ ràng, cố định và không dự kiến thay đổi. Nó cũng phù hợp với các dự án có quy trình và tài liệu chi tiết, như trong lĩnh vực xây dựng phần mềm hệ thống hoặc các dự án có quy định nghiêm ngặt.

7. Thảo luận về những khó khăn khi áp dụng mô hình mã nguồn mở:

- Các khó khăn khi áp dụng mô hình mã nguồn mở bao gồm:
    - *An ninh và bảo mật*: Rủi ro về bảo mật cao hơn do mã nguồn công khai.
    - *Quản lý dự án*: Khó khăn trong việc điều phối và kiểm soát đóng góp từ cộng đồng toàn cầu.
    - *Khả năng tương thích*: Cần đảm bảo phần mềm tương thích với nhiều hệ thống và cấu hình khác nhau.

8. Phân tích cách mô hình tiến trình linh hoạt giúp cải thiện chất lượng phần mềm:

- Mô hình tiến trình linh hoạt (Agile) giúp cải thiện chất lượng phần mềm thông qua việc phát triển theo từng phần nhỏ, kiểm tra liên tục và nhận phản hồi nhanh từ khách hàng. Điều này đảm bảo rằng các lỗi được phát hiện và sửa chữa sớm, và phần mềm đáp ứng tốt hơn các yêu cầu của người dùng.

9. Vai trò của pha bảo trì trong vòng đời phát triển phần mềm

Pha bảo trì là giai đoạn cuối cùng trong vòng đời phát triển phần mềm và đóng vai trò cực kỳ quan trọng vì những lý do sau:

- **Khắc phục lỗi**: Bất kỳ phần mềm nào cũng có thể gặp lỗi trong quá trình sử dụng. Pha bảo trì giúp phát hiện và sửa chữa những lỗi này, đảm bảo phần mềm hoạt động ổn định và chính xác.
- **Cập nhật và nâng cấp**: Công nghệ luôn thay đổi, và phần mềm cần được cập nhật để đáp ứng yêu cầu mới hoặc tích hợp với các công nghệ mới. Pha bảo trì đảm bảo phần mềm luôn được cải tiến và nâng cấp theo thời gian.
- **Tối ưu hiệu suất**: Pha bảo trì giúp cải thiện hiệu suất của phần mềm, làm cho nó chạy nhanh hơn, mượt mà hơn và tiêu tốn ít tài nguyên hơn.
- **Bảo mật**: Bảo mật là một trong những yếu tố quan trọng nhất trong phần mềm. Pha bảo trì đảm bảo các lỗ hổng bảo mật được phát hiện và khắc phục, bảo vệ phần mềm và dữ liệu người dùng khỏi các cuộc tấn công mạng.
- **Đảm bảo tính khả dụng**: Phần mềm phải hoạt động liên tục và đáp ứng các yêu cầu của người dùng. Pha bảo trì đảm bảo phần mềm luôn sẵn sàng và hoạt động đúng cách.

10. Đề xuất mô hình vòng đời phù hợp cho dự án phát triển phần mềm ngân hàng và giải thích lý do

Đối với một dự án phát triển phần mềm ngân hàng, mô hình vòng đời phát triển phần mềm Agile là lựa chọn phù hợp nhất. Lý do cụ thể bao gồm:

- **Linh hoạt và thích nghi**: Ngành ngân hàng thường gặp phải những thay đổi nhanh chóng trong quy định, công nghệ và yêu cầu của khách hàng. Mô hình Agile với các sprint ngắn và khả năng thay đổi ưu tiên sẽ giúp đội phát triển nhanh chóng thích nghi với các thay đổi này.
- **Tương tác thường xuyên với khách hàng**: Agile khuyến khích sự hợp tác liên tục giữa đội phát triển và khách hàng, giúp hiểu rõ yêu cầu và phản hồi của khách hàng, từ đó tạo ra phần mềm đáp ứng đúng nhu cầu.
- **Phát hành thường xuyên**: Agile cho phép phát hành phiên bản phần mềm mới một cách thường xuyên, giúp ngân hàng nhanh chóng cung cấp các tính năng mới cho khách hàng và điều chỉnh theo phản hồi.
- **Chất lượng cao**: Agile tập trung vào kiểm thử liên tục và tích hợp liên tục, giúp phát hiện và khắc phục lỗi sớm, đảm bảo chất lượng phần mềm tốt nhất.
- **Minh bạch và kiểm soát**: Agile sử dụng các công cụ và phương pháp như Scrum, Kanban để quản lý công việc và tiến độ, giúp đội phát triển và quản lý dự án có cái nhìn rõ ràng về tiến trình và công việc cần làm.

CÂU HỎI TÌNH HUỐNG

1. Xử lý yêu cầu thay đổi sau khi hoàn thành pha thiết kế trong mô hình thác nước

- Đánh giá tác động của thay đổi đối với toàn bộ dự án.
- Thảo luận với khách hàng về chi phí và thời gian cần thiết.
- Cập nhật tài liệu thiết kế và tiến hành lại các pha bị ảnh hưởng.

2. Giải quyết trễ tiến độ do thiếu nhân lực trong mô hình lặp và tăng trưởng liên tục

- Tìm kiếm và tuyển dụng thêm nhân lực.
- Điều chỉnh lại phạm vi và ưu tiên của dự án.
- Tăng cường đào tạo và phân công lại công việc hiện có.

3. Xử lý tình trạng khách hàng không đưa ra phản hồi kịp thời trong mô hình tiến trình linh hoạt

- Thiết lập kênh liên lạc rõ ràng và thường xuyên với khách hàng.
- Đặt ra các mốc thời gian cụ thể cho phản hồi.

- Tạo ra các tài liệu chi tiết và dễ hiểu để khách hàng dễ dàng đưa ra phản hồi.

4. Xử lý yêu cầu bổ sung tính năng mới khi phần mềm đã bước vào pha cài đặt

- Áp dụng mô hình **Agile** để linh hoạt đáp ứng các yêu cầu thay đổi.
- Lên kế hoạch phát hành tính năng mới trong các sprint tiếp theo.

5. Giải pháp cho công ty nhỏ gặp khó khăn khi áp dụng mô hình bản mẫu nhanh

- Sử dụng công cụ phần mềm tự động hoá để giảm khối lượng công việc.
- Tập trung vào việc phát triển các bản mẫu nhỏ gọn và dễ điều chỉnh.
- Hợp tác với các đơn vị bên ngoài để bổ sung nhân lực khi cần thiết.

6. Mô hình phù hợp khi khách hàng liên tục yêu cầu thay đổi giao diện trong dự án thương mại điện tử

- Mô hình **Agile** hoặc **Iterative** sẽ phù hợp để linh hoạt đáp ứng các yêu cầu thay đổi giao diện.

7. Áp dụng mô hình cho dự án phát triển phần mềm lớn với nhiều nhóm phát triển ở các quốc gia khác nhau

- Mô hình **Scrum of Scrums** hoặc **Scaled Agile Framework (SAFe)** sẽ phù hợp để quản lý nhiều nhóm phát triển phân tán.

8. Cách giảm thiểu rủi ro khi áp dụng mô hình xoắn ốc

- Thực hiện đánh giá rủi ro thường xuyên và cập nhật kế hoạch giảm thiểu rủi ro.
- Tăng cường kiểm thử và đánh giá định kỳ.
- Đảm bảo sự tham gia của các bên liên quan trong quá trình phát triển.

9. Mô hình phù hợp cho dự án phát triển phần mềm ngân hàng cần yêu cầu bảo mật cao

- Mô hình **DevSecOps** để tích hợp bảo mật vào từng giai đoạn phát triển phần mềm.

10. Khi nào nên kết thúc vòng đời phần mềm và thực hiện pha giải thể?

- Khi phần mềm không còn đáp ứng được yêu cầu kinh doanh hoặc kỹ thuật.
- Khi chi phí bảo trì vượt quá lợi ích mang lại.
- Khi có sản phẩm mới thay thế phần mềm hiện tại.

CHƯƠNG 4: VẤN ĐỀ KIỂM THỬ, LẬP KẾ HOẠCH VÀ TÀI LIỆU

CÂU HỎI TRẮC NGHIỆM

Câu hỏi 1: Mục tiêu chính của kiểm thử phần mềm là gì?
A. Đảm bảo phần mềm không chứa lỗi
B. Đảm bảo phần mềm đáp ứng đúng yêu cầu
C. Hoàn thành dự án đúng thời gian D. Phát hiện tất cả các lỗi bảo mật
2. Câu hỏi 2: Nhóm nào chịu trách nhiệm đảm bảo chất lượng phần mềm trong dự án? A. Nhóm kiểm thử tự động
B. Nhóm bảo trì
C. Nhóm SQA
D. Nhóm thiết kế
3. Câu hỏi 3: Kiểm thử các sản phẩm phi thực thi bao gồm hoạt động nào dưới đây?
A. Kiểm thử đơn vị
B. Walkthrough và review tài liệu
C. Kiểm thử hệ thống
D. Kiểm thử hiệu suất
4. Câu hỏi 4: Lập kế hoạch trong tiến trình phát triển phần mềm nhằm mục đích gì?
A. Xác định phạm vi công việc và ước tính thời gian hoàn thành
B. Giảm thiểu chi phí phát triển
C. Phát hiện lỗi sớm trong quá trình phát triển
D. Xác định các công cụ kiểm thử tự động
5. Câu hỏi 5: Loại kiểm thử nào sau đây tập trung vào việc phát hiện các lỗi chức năng của phần mềm?
A. Kiểm thử hiệu suất
B. Kiểm thử chức năng
C. Kiểm thử bảo mật
D. Kiểm thử tích hợp
Câu hỏi 6: Quản lý phiên bản tài liệu có mục tiêu gì?
A. Đảm bảo tài liệu luôn được cập nhật và có thể truy xuất phiên bản cũ khi cần
B. Giảm thiểu số lượng tài liệu cần làm
C. Tăng tốc độ phát triển phần mềm
D. Tự động hóa việc kiểm thử tài liệu
7. Câu hỏi 7: Nhóm SQA có vai trò gì trong kiểm thử phần mềm?
A. Viết mã nguồn cho phần mềm
B. Đánh giá và đảm bảo quy trình phát triển phần mềm tuân thủ tiêu chuẩn chất lượng
C. Triển khai phần mềm lên môi trường thực tế
D. Giám sát hoạt động của hệ thống sau khi triển khai
8. Câu hỏi 8: Loại kiểm thử nào thường được thực hiện cuối cùng trước khi phần mềm được bàn giao cho khách hàng?
A. Kiểm thử đơn vị
B. Kiểm thử hệ thống
C. Kiểm thử tích hợp
D. Kiểm thử chấp nhận

9. Câu hỏi 9: Đâu là một trong những công cụ phổ biến dùng để quản lý phiên bản tài liệu?
A. Selenium
B. Git
C. Postman
D. JIRA
10.Câu hỏi 10: Hoạt động lập tài liệu trong mỗi pha phát triển phần mềm nhằm mục đích gì?
A. Giảm thời gian phát triển phần mềm
B. Hỗ trợ quá trình bảo trì và nâng cấp phần mềm sau khi triển khai
C. Tăng cường bảo mật cho phần mềm
D. Tự động hóa việc phát triển phần mềm 8

Câu hỏi ngắn:

1. **Nhóm SQA là gì và vai trò của nhóm này trong phát triển phần mềm?**

Nhóm SQA (Software Quality Assurance) là nhóm chịu trách nhiệm đảm bảo chất lượng phần mềm bằng cách thực hiện các quy trình và quy chuẩn kiểm tra chất lượng. Vai trò của nhóm này bao gồm việc thiết lập các chuẩn chất lượng, giám sát quá trình phát triển, thực hiện các kiểm thử, và đảm bảo phần mềm đạt được các yêu cầu về chất lượng.

2. **Kiểm thử đơn vị là gì?**

Kiểm thử đơn vị (Unit Testing) là quá trình kiểm tra từng đơn vị nhỏ nhất của mã nguồn phần mềm một cách độc lập để đảm bảo rằng từng phần tử hoạt động đúng như mong đợi.

3. **Mục tiêu chính của kiểm thử chấp nhận là gì?**

Mục tiêu chính của kiểm thử chấp nhận (Acceptance Testing) là xác minh rằng hệ thống hoặc phần mềm đáp ứng đầy đủ các yêu cầu và tiêu chuẩn của người dùng cuối hoặc khách hàng, và sẵn sàng để triển khai.

4. **Các hoạt động chính trong kiểm thử sản phẩm phi thực thi là gì?**

Kiểm thử sản phẩm phi thực thi bao gồm việc kiểm tra các tài liệu liên quan như tài liệu yêu cầu, thiết kế, và hướng dẫn sử dụng. Các hoạt động chính gồm: đọc tài liệu, phân tích, đánh giá và xem xét lại các tài liệu.

5. **Tại sao việc lập tài liệu cho mỗi pha phát triển phần mềm lại quan trọng?**

Việc lập tài liệu giúp ghi lại các yêu cầu, thiết kế, quyết định và thay đổi trong suốt quá trình phát triển. Điều này giúp đảm bảo tính minh bạch, khả năng truy xuất và bảo trì dễ dàng, đồng thời hỗ trợ giao tiếp giữa các thành viên trong nhóm.

6. **Quản lý phiên bản tài liệu là gì?**

Quản lý phiên bản tài liệu (Document Version Control) là quy trình quản lý và theo dõi các thay đổi của tài liệu trong suốt quá trình phát triển, đảm bảo rằng mọi thành viên trong nhóm đều làm việc với phiên bản tài liệu mới nhất.

7. **Các loại kiểm thử chính trong kiểm thử sản phẩm thực thi là gì?**

Các loại kiểm thử chính bao gồm: kiểm thử chức năng, kiểm thử phi chức năng, kiểm thử hồi quy, kiểm thử hệ thống, kiểm thử hiệu suất, kiểm thử bảo mật và kiểm thử chấp nhận.

8. **Kiểm thử tích hợp là gì?**

Kiểm thử tích hợp (Integration Testing) là quá trình kiểm tra sự tương tác giữa các đơn vị hoặc mô-đun phần mềm để đảm bảo rằng chúng hoạt động cùng nhau một cách chính xác.

9. **Hoạt động lập kế hoạch cho các pha phát triển phần mềm bao gồm những gì?**

Hoạt động lập kế hoạch bao gồm: xác định phạm vi công việc, phân bổ nguồn lực, thiết lập lịch trình, định rõ mục tiêu và tiêu chí đánh giá, và quản lý rủi ro.

10. **Làm tài liệu kiểm thử bao gồm những gì?**

Làm tài liệu kiểm thử bao gồm: mô tả mục tiêu kiểm thử, kế hoạch kiểm thử, các kịch bản kiểm thử, kết quả kiểm thử, và báo cáo lỗi.

Câu hỏi thảo luận nhóm:

1. **Vai trò của nhóm SQA trong việc đảm bảo chất lượng phần mềm là gì?**
    - Nhóm SQA (Software Quality Assurance) đảm bảo rằng phần mềm phát triển đáp ứng các yêu cầu đã định và không có lỗi. Họ thực hiện các quy trình kiểm tra, xem xét mã nguồn, lập kế hoạch kiểm thử, và theo dõi các vấn đề để đảm bảo chất lượng phần mềm từ giai đoạn thiết kế đến triển khai.
2. **Thảo luận về sự khác nhau giữa kiểm thử đơn vị và kiểm thử tích hợp.**
    - Kiểm thử đơn vị (Unit Testing) tập trung vào kiểm thử từng đơn vị nhỏ nhất của phần mềm, chẳng hạn như các hàm hoặc mô-đun, để đảm bảo chúng hoạt động đúng. Trong khi đó, kiểm thử tích hợp (Integration Testing) kiểm thử các đơn vị đã được tích hợp lại với nhau để đảm bảo rằng chúng hoạt động hợp lý khi kết hợp.
3. **Tại sao việc lập tài liệu kiểm thử lại quan trọng trong mỗi dự án phần mềm?**
    - Tài liệu kiểm thử là quan trọng để ghi nhận chi tiết các kịch bản kiểm thử, các bước thực hiện, kết quả mong đợi và kết quả thực tế. Nó giúp đảm bảo rằng quá trình kiểm thử có hệ thống và có thể tái lập lại, đồng thời cung cấp bằng chứng cho quá trình kiểm thử và giúp đội ngũ dễ dàng phát hiện và sửa lỗi.
4. **Thảo luận về các thách thức khi lập kế hoạch cho các pha phát triển phần mềm.**
    - Lập kế hoạch phát triển phần mềm có nhiều thách thức như xác định yêu cầu chính xác, ước lượng thời gian và nguồn lực, đối phó với các thay đổi yêu cầu, và duy trì sự liên lạc liên tục giữa các thành viên trong nhóm. Những yếu tố này có thể ảnh hưởng đến tiến độ và chất lượng dự án.
5. **Quản lý phiên bản tài liệu có ảnh hưởng như thế nào đến quá trình bảo trì phần mềm?**
    - Quản lý phiên bản tài liệu đảm bảo rằng tất cả các thay đổi và cập nhật được theo dõi chính xác và có thể phục hồi lại khi cần thiết. Điều này rất quan trọng trong việc bảo trì phần mềm vì nó giúp đội ngũ xác định các thay đổi đã thực hiện và đảm bảo rằng phần mềm luôn cập nhật với phiên bản mới nhất.
6. **So sánh giữa kiểm thử sản phẩm phi thực thi và kiểm thử sản phẩm thực thi.**
    - Kiểm thử sản phẩm phi thực thi (Non-Executable Testing) liên quan đến việc kiểm thử các tài liệu như tài liệu yêu cầu, thiết kế, và tài liệu người dùng. Trong khi đó, kiểm thử sản phẩm thực thi (Executable Testing) liên quan đến việc chạy phần mềm để kiểm tra các chức năng và hiệu suất của nó.
7. **Thảo luận về cách cải thiện quy trình lập kế hoạch để giảm thiểu rủi ro trong dự án phần mềm.**

- Để cải thiện quy trình lập kế hoạch và giảm thiểu rủi ro, đội ngũ có thể thực hiện các bước như phân tích rủi ro, lập kế hoạch dự phòng, theo dõi tiến độ thường xuyên, và sử dụng các phương pháp phát triển linh hoạt (Agile) để nhanh chóng phản ứng với các thay đổi.

8. **Đề xuất các công cụ hỗ trợ việc lập tài liệu kiểm thử và quản lý phiên bản.**
   - Một số công cụ hỗ trợ việc lập tài liệu kiểm thử và quản lý phiên bản bao gồm JIRA, Confluence, TestRail, Git, và GitHub. Những công cụ này giúp đội ngũ dễ dàng quản lý các tài liệu kiểm thử, theo dõi các lỗi và cập nhật phiên bản phần mềm.
9. **Tại sao kiểm thử chấp nhận lại là một giai đoạn quan trọng trong phát triển phần mềm?**
   - Kiểm thử chấp nhận (Acceptance Testing) là giai đoạn cuối cùng trước khi phần mềm được triển khai. Nó xác nhận rằng phần mềm đã hoàn thành đúng các yêu cầu và đáp ứng mong đợi của người dùng. Đây là bước quan trọng để đảm bảo rằng sản phẩm cuối cùng thực sự đáp ứng được nhu cầu của khách hàng.
10. **Thảo luận về các phương pháp hiệu quả để quản lý chất lượng phần mềm trong các dự án lớn.**
    - Các phương pháp hiệu quả để quản lý chất lượng phần mềm trong các dự án lớn bao gồm sử dụng các phương pháp phát triển theo hướng kiểm thử (Test-Driven Development), lập kế hoạch kiểm thử chi tiết, thực hiện kiểm thử tự động, và duy trì một quy trình kiểm soát chất lượng liên tục. Điều quan trọng là có sự cam kết từ tất cả các bên liên quan để đảm bảo chất lượng được duy trì xuyên suốt dự án.

Câu hỏi tình huống:

1. Một dự án phần mềm đã hoàn thành và sắp bàn giao cho khách hàng, nhưng khách hàng yêu cầu kiểm tra lại toàn bộ tài liệu yêu cầu và thiết kế. Đội phát triển nên xử lý thế nào?

Phương án:

- **Tiếp nhận yêu cầu một cách chuyên nghiệp:** Thể hiện sự sẵn sàng hợp tác và tôn trọng yêu cầu của khách hàng.
- **Xác định phạm vi kiểm tra:** Thảo luận với khách hàng để hiểu rõ mục đích và phạm vi của việc kiểm tra.
- **Lập kế hoạch kiểm tra:** Xác định thời gian, nguồn lực và phương pháp kiểm tra phù hợp.
- **Thực hiện kiểm tra:** Tiến hành kiểm tra kỹ lưỡng và ghi lại kết quả.
- **Báo cáo kết quả:** Báo cáo kết quả kiểm tra cho khách hàng một cách rõ ràng và chi tiết.
- **Thống nhất giải pháp:** Thảo luận với khách hàng để thống nhất về các vấn đề cần điều chỉnh và cách thức thực hiện.

2. Trong quá trình kiểm thử hệ thống, nhóm phát triển phát hiện một lỗi nghiêm trọng nhưng thời hạn bàn giao đang đến gần. Bạn sẽ xử lý tình huống này như thế nào?

Phương án:

- **Tiếp nhận yêu cầu một cách chuyên nghiệp:** Thể hiện sự sẵn sàng hợp tác và tôn trọng yêu cầu của khách hàng.

- **Xác định phạm vi kiểm tra:** Thảo luận với khách hàng để hiểu rõ mục đích và phạm vi của việc kiểm tra.
- **Lập kế hoạch kiểm tra:** Xác định thời gian, nguồn lực và phương pháp kiểm tra phù hợp.
- **Thực hiện kiểm tra:** Tiến hành kiểm tra kỹ lưỡng và ghi lại kết quả.
- **Báo cáo kết quả:** Báo cáo kết quả kiểm tra cho khách hàng một cách rõ ràng và chi tiết.
- **Thống nhất giải pháp:** Thảo luận với khách hàng để thống nhất về các vấn đề cần điều chỉnh và cách thức thực hiện.

3. Một nhóm phát triển gặp khó khăn trong việc quản lý phiên bản tài liệu do tài liệu liên tục thay đổi. Hãy đề xuất giải pháp.
Phương án:

- **Sử dụng công cụ quản lý phiên bản:** Sử dụng các công cụ như Git, SVN để quản lý phiên bản tài liệu một cách hiệu quả.
- **Thiết lập quy trình quản lý phiên bản:** Xây dựng quy trình rõ ràng về việc tạo, sửa đổi và lưu trữ tài liệu.
- **Đào tạo nhân viên:** Đào tạo nhân viên về cách sử dụng công cụ và tuân thủ quy trình.
- **Kiểm soát truy cập:** Phân quyền truy cập tài liệu để đảm bảo tính bảo mật và tránh xung đột.

4. Dự án phát triển phần mềm gặp vấn đề khi khách hàng yêu cầu thay đổi lớn trong pha cài đặt. Đội phát triển nên xử lý thế nào?
Phương án:

- **Đánh giá tác động:** Đánh giá tác động của thay đổi đến tiến độ, chi phí và phạm vi của dự án.
- **Thương lượng với khách hàng:** Thảo luận với khách hàng về những tác động này và tìm kiếm giải pháp phù hợp.
- **Lập kế hoạch thay đổi:** Nếu quyết định thực hiện thay đổi, cần lập kế hoạch chi tiết và điều chỉnh lịch trình dự án.
- **Quản lý thay đổi:** Quản lý thay đổi một cách chặt chẽ để đảm bảo dự án vẫn được hoàn thành đúng hạn và ngân sách.

5. Nhóm kiểm thử phát hiện nhiều lỗi chức năng trong phần mềm. Tuy nhiên, nhóm phát triển lại cho rằng đây không phải lỗi mà là tính năng. Là trưởng dự án, bạn sẽ làm gì?
Phương án:

- **Phân tích yêu cầu:** Xem xét lại tài liệu yêu cầu để xác định rõ ràng đâu là lỗi, đâu là tính năng.
- **Thống nhất định nghĩa:** Thống nhất với nhóm phát triển về định nghĩa lỗi và tính năng.
- **Giải quyết tranh chấp:** Nếu có tranh chấp, cần có người thứ ba (ví dụ: trưởng dự án) đứng ra giải quyết.
- **Ghi lại kết quả:** Ghi lại kết quả phân tích và giải thích rõ ràng lý do.

6. Khách hàng yêu cầu bổ sung một tính năng mới khi phần mềm đã hoàn thành pha kiểm thử tích hợp. Đội phát triển nên làm gì?
Phương án:

- **Đánh giá tác động:** Đánh giá tác động của tính năng mới đến hệ thống và lịch trình dự án.
- **Thương lượng với khách hàng:** Thảo luận với khách hàng về chi phí và thời gian cần thiết để bổ sung tính năng.
- **Lập kế hoạch bổ sung:** Nếu khách hàng đồng ý, cần lập kế hoạch chi tiết và điều chỉnh lịch trình dự án.
- **Kiểm tra lại:** Sau khi bổ sung tính năng, cần kiểm tra lại kỹ lưỡng để đảm bảo tính năng hoạt động đúng và không gây ra lỗi.

7. Một công ty phát triển phần mềm nhỏ muốn xây dựng nhóm SQA nhưng gặp khó khăn về ngân sách. Hãy đề xuất giải pháp.
Phương án:

- **Ưu tiên các hoạt động quan trọng:** Tập trung vào các hoạt động SQA quan trọng nhất, ví dụ như kiểm thử chức năng, kiểm thử hiệu năng, kiểm thử bảo mật.
- **Sử dụng công cụ miễn phí:** Sử dụng các công cụ kiểm thử miễn phí hoặc mã nguồn mở.
- **Đào tạo nội bộ:** Đào tạo nhân viên hiện có để thực hiện các công việc SQA.
- **Hợp tác với bên ngoài:** Thuê ngoài một số công việc SQA nếu cần thiết.

8. Trong quá trình làm tài liệu kiểm thử, nhóm phát triển không thống nhất được về nội dung cần đưa vào tài liệu. Là trưởng nhóm, bạn sẽ giải quyết vấn đề này như thế nào?
Phương án:

- **Thống nhất mục tiêu:** Thống nhất với nhóm kiểm thử về mục tiêu của tài liệu kiểm thử.
- **Xác định đối tượng:** Xác định đối tượng sử dụng tài liệu kiểm thử.
- **Lựa chọn nội dung phù hợp:** Lựa chọn nội dung phù hợp với mục tiêu và đối tượng.
- **Thảo luận và thống nhất:** Tổ chức buổi họp để nhóm thảo luận và thống nhất về nội dung tài liệu.

9. Dự án phát triển phần mềm cho một ngân hàng yêu cầu bảo mật cao. Đề xuất cách lập kế hoạch kiểm thử cho dự án này.
Phương án:

- **Xác định rủi ro:** Xác định các rủi ro bảo mật có thể xảy ra.
- **Lựa chọn phương pháp kiểm thử:** Lựa chọn các phương pháp kiểm thử phù hợp, ví dụ như kiểm thử xâm nhập, kiểm thử bảo mật ứng dụng.
- **Xây dựng kịch bản kiểm thử:** Xây dựng các kịch bản kiểm thử chi tiết để kiểm tra các lỗ hổng bảo mật.
- **Sử dụng công cụ kiểm thử:** Sử dụng các công cụ kiểm thử bảo mật chuyên dụng.
- **Phân tích kết quả:** Phân tích kết quả kiểm thử để xác định các vấn đề bảo mật.

10.Sau khi triển khai phần mềm, khách hàng phát hiện ra một số lỗi bảo mật nghiêm trọng. Đội phát triển cần xử lý ra sao để khắc phục vấn đề và lấy lại niềm tin từ khách hàng?

Phương án:

- **Khắc phục lỗi:** Khắc phục lỗi bảo mật một cách nhanh chóng và triệt để.
- **Thông báo cho khách hàng:** Thông báo cho khách hàng về lỗi và biện pháp khắc phục.

- **Điều tra nguyên nhân:** Điều tra nguyên nhân gây ra lỗi để tránh tái diễn.
- **Cải tiến quy trình:** Cải tiến quy trình phát triển để đảm bảo an ninh bảo mật.
- **Xây dựng lại niềm tin:** Thực hiện các biện pháp để xây dựng lại niềm tin từ khách hàng, ví dụ như cung cấp cam kết về bảo mật, hỗ trợ khách hàng kịp thời.

CHƯƠNG 5: LẤY YÊU CẦU

CÂU HỎI TRẮC NGHIỆM

1. Câu hỏi 1: Kỹ thuật nào sau đây thường được sử dụng để thu thập yêu cầu khách hàng?
A. Lập trình theo cặp
B. Phỏng vấn khách hàng
C. Kiểm thử đơn vị
D. Triển khai hệ thống
2. Câu hỏi 2: Trong bước tổng hợp kết quả yêu cầu, việc nào sau đây là cần thiết?
A. Xây dựng sơ đồ lớp
B. Phân loại yêu cầu và loại bỏ thông tin trùng lặp
C. Viết mã nguồn
D. Thiết kế giao diện
3. Câu hỏi 3: Use case mô tả:
A. Cách kiểm thử phần mềm
B. Cách người dùng tương tác với hệ thống
C. Cách quản lý tài liệu
D. Cách bảo trì phần mềm
4. Câu hỏi 4: Tại sao cần xây dựng danh sách từ khóa chuyên môn khi tìm hiểu lĩnh vực ứng dụng?
A. Để giảm thời gian lập trình
B. Để tăng tốc độ kiểm thử
C. Để đảm bảo đội phát triển và khách hàng hiểu rõ các thuật ngữ
D. Để giảm chi phí phát triển
5. Câu hỏi 5: Quan hệ nào sau đây trong use case mô tả một use case có thể mở rộng thêm chức năng trong một số điều kiện nhất định?
A. Include
B. Extend
C. Generalization
D. Aggregation 10
6. Câu hỏi 6: Khi mô tả yêu cầu bằng ngôn ngữ tự nhiên, điều quan trọng nhất là gì?
A. Sử dụng nhiều thuật ngữ kỹ thuật
B. Viết thật ngắn gọn và không cần kiểm tra lại với khách hàng
C. Viết rõ ràng, dễ hiểu và tránh từ đa nghĩa
D. Chỉ tập trung vào các yêu cầu phi chức năng
7. Câu hỏi 7: Mục tiêu chính của việc trích các use case là gì?
A. Thiết kế giao diện người dùng
B. Mô tả các chức năng mà hệ thống cung cấp cho người dùng
C. Viết mã nguồn cho hệ thống
D. Lập kế hoạch kiểm thử
8. Câu hỏi 8: Quan hệ nào giữa các use case thể hiện việc một use case phải gọi một use case khác để thực hiện chức năng đầy đủ?
A. Include
B. Extend
C. Generalization

D. Dependency
9. Câu hỏi 9: Hoạt động nào sau đây là bước đầu tiên trong việc xây dựng mô hình nghiệp vụ?
A. Thiết kế giao diện người dùng
B. Trích các use case
C. Viết mã nguồn
D. Thực hiện kiểm thử hệ thống
10. Câu hỏi 10: Một yêu cầu chức năng là gì?
A. Một yêu cầu mô tả cách hệ thống xử lý dữ liệu
B. Một yêu cầu về tốc độ xử lý của hệ thống
C. Một yêu cầu về khả năng bảo trì phần mềm
D. Một yêu cầu về giao diện người dùng

CÂU HỎI TRẢ LỜI NGẮN

1. Kỹ thuật phỏng vấn khách hàng là gì?

Kỹ thuật phỏng vấn khách hàng là một phương pháp thu thập thông tin trực tiếp từ khách hàng hoặc người dùng liên quan đến dự án phần mềm. Trong quá trình phỏng vấn, người phân tích nghiệp vụ sẽ đặt câu hỏi, lắng nghe ý kiến và ghi nhận lại các yêu cầu, mong muốn của khách hàng về phần mềm.

2. Tại sao cần tổng hợp kết quả yêu cầu?

Việc tổng hợp kết quả yêu cầu là cần thiết vì:

- **Đảm bảo tính đầy đủ**: Giúp thu thập đầy đủ các yêu cầu từ nhiều nguồn khác nhau (khách hàng, người dùng, tài liệu, ...).
- **Loại bỏ trùng lặp**: Phát hiện và loại bỏ các yêu cầu trùng lặp, giúp danh sách yêu cầu rõ ràng và mạch lạc hơn.
- **Phân loại và sắp xếp**: Phân loại các yêu cầu theo nhóm (chức năng, phi chức năng, ...) và sắp xếp chúng theo mức độ quan trọng.
- **Xác định xung đột**: Phát hiện các yêu cầu mâu thuẫn hoặc xung đột lẫn nhau để có biện pháp xử lý phù hợp.

3. Use case là gì?

Use case (trường hợp sử dụng) là một kỹ thuật mô tả cách người dùng tương tác với hệ thống để thực hiện một chức năng cụ thể nào đó. Mỗi use case đại diện cho một hành động hoặc mục tiêu của người dùng khi sử dụng hệ thống.

4. Mục tiêu của việc xây dựng danh sách từ khóa chuyên môn là gì?

Mục tiêu của việc xây dựng danh sách từ khóa chuyên môn là:

- **Thống nhất thuật ngữ**: Đảm bảo rằng tất cả các bên liên quan (đội phát triển, khách hàng, người dùng) hiểu và sử dụng cùng một thuật ngữ khi nói về dự án.
- **Tránh hiểu lầm**: Giảm thiểu nguy cơ hiểu lầm do sử dụng các từ ngữ không rõ ràng hoặc đa nghĩa.
- **Giao tiếp hiệu quả**: Tạo điều kiện giao tiếp hiệu quả giữa các bên liên quan, giúp dự án diễn ra suôn sẻ hơn.

5. Quan hệ Include giữa các use case là gì?

Quan hệ Include (bao gồm) giữa các use case thể hiện rằng một use case phải gọi một use case khác để thực hiện chức năng đầy đủ. Use case được gọi là một phần không thể thiếu của use case gọi.

6. Quan hệ Extend giữa các use case là gì?

Quan hệ Extend (mở rộng) giữa các use case thể hiện rằng một use case có thể mở rộng thêm chức năng cho một use case khác trong một số điều kiện nhất định. Use case mở rộng không bắt buộc phải được thực hiện, nó chỉ được kích hoạt khi có điều kiện đặc biệt.

7. Yêu cầu phi chức năng là gì?

Yêu cầu phi chức năng là các yêu cầu mô tả các đặc tính của hệ thống, chẳng hạn như:

- Hiệu năng: Tốc độ xử lý, thời gian phản hồi.
- Bảo mật: Khả năng bảo vệ dữ liệu khỏi truy cập trái phép.
- Độ tin cậy: Khả năng hoạt động ổn định và không bị lỗi.
- Khả năng sử dụng: Giao diện thân thiện, dễ sử dụng.

8. Mô hình nghiệp vụ là gì?

Mô hình nghiệp vụ là một biểu đồ trực quan mô tả các quy trình, hoạt động và các yếu tố liên quan trong một lĩnh vực kinh doanh cụ thể. Nó giúp các bên liên quan hiểu rõ hơn về cách thức hoạt động của doanh nghiệp và xác định các vấn đề cần giải quyết.

9. Điều kiện trước và điều kiện sau của use case là gì?

- **Điều kiện trước**: Là các điều kiện cần phải được đáp ứng trước khi use case có thể được thực hiện.
- **Điều kiện sau**: Là các kết quả hoặc trạng thái của hệ thống sau khi use case đã được thực hiện thành công.

10. Một số lưu ý khi trích các use case là gì?

Một số lưu ý khi trích các use case:

- **Tập trung vào người dùng**: Use case nên được mô tả từ góc độ của người dùng, tập trung vào mục tiêu và hành động của họ.

- **Mô tả rõ ràng**: Mỗi use case nên được mô tả ngắn gọn, rõ ràng và dễ hiểu, tránh sử dụng các thuật ngữ quá chuyên môn.
- **Đầy đủ thông tin**: Mỗi use case nên bao gồm đầy đủ thông tin về điều kiện trước, điều kiện sau, luồng xử lý chính và các luồng xử lý ngoại lệ.
- **Kiểm tra lại**: Sau khi trích các use case, cần kiểm tra lại với khách hàng để đảm bảo rằng chúng phản ánh đúng nhu cầu và mong muốn của họ.

CÂU HỎI THẢO LUẬN NHÓM

1. Khách hàng không hiểu rõ các thuật ngữ kỹ thuật trong tài liệu yêu cầu.

- **Giải pháp:**
    - **Giải thích thuật ngữ:** Trưởng nhóm phát triển cần dành thời gian giải thích các thuật ngữ kỹ thuật một cách dễ hiểu, sử dụng ví dụ minh họa hoặc so sánh với các khái niệm quen thuộc.
    - **Sử dụng ngôn ngữ đơn giản:** Chuyển đổi các thuật ngữ phức tạp thành ngôn ngữ đơn giản, gần gũi với khách hàng.
    - **Tạo bảng chú giải:** Lập một bảng chú giải các thuật ngữ kỹ thuật, kèm theo giải thích ngắn gọn và dễ hiểu.
    - **Trực quan hóa:** Sử dụng sơ đồ, hình ảnh để minh họa các khái niệm kỹ thuật, giúp khách hàng dễ hình dung hơn.

2. Trong quá trình lấy yêu cầu, khách hàng liên tục thay đổi ý kiến về chức năng cần thiết.

- **Giải pháp:**
    - **Xác định nguyên nhân:** Tìm hiểu lý do khách hàng thay đổi ý kiến, có thể do chưa hiểu rõ yêu cầu, hoặc do có thêm thông tin mới.
    - **Linh hoạt:** Đội phát triển cần linh hoạt, sẵn sàng điều chỉnh yêu cầu khi có thay đổi từ khách hàng.
    - **Ghi lại thay đổi:** Ghi lại tất cả các thay đổi về yêu cầu, bao gồm cả lý do thay đổi và các ảnh hưởng đến dự án.
    - **Ưu tiên:** Thống nhất với khách hàng về mức độ ưu tiên của các yêu cầu, để có thể tập trung vào các yêu cầu quan trọng nhất.
    - **Kiểm soát thay đổi:** Áp dụng quy trình kiểm soát thay đổi để quản lý các thay đổi một cách hiệu quả, tránh ảnh hưởng tiêu cực đến tiến độ và ngân sách dự án.

3. Dự án phần mềm quản lý bán hàng gặp vấn đề khi các yêu cầu được mô tả quá chung chung, khó thực hiện.

- **Giải pháp:**
    - **Phân tích chi tiết:** Đội phát triển cần phân tích chi tiết các yêu cầu chung chung, chia nhỏ chúng thành các yêu cầu cụ thể, rõ ràng và có thể đo lường được.

    - **Sử dụng Use Case:** Sử dụng Use Case để mô tả chi tiết các tương tác giữa người dùng và hệ thống, giúp làm rõ các yêu cầu.
    - **Xác định tiêu chí:** Xác định các tiêu chí đánh giá mức độ thành công của từng yêu cầu, để đảm bảo rằng chúng được thực hiện đúng cách.

4. Khách hàng đưa ra yêu cầu không rõ ràng và chỉ có thể mô tả một cách sơ lược.

- **Giải pháp:**
    - **Phỏng vấn sâu:** Thực hiện phỏng vấn sâu với khách hàng, đặt câu hỏi mở để khách hàng chia sẻ thêm thông tin về yêu cầu của mình.
    - **Sử dụng Prototype:** Xây dựng một bản mẫu (prototype) đơn giản để khách hàng hình dung rõ hơn về chức năng mong muốn.
    - **Lắng nghe:** Lắng nghe cẩn thận những gì khách hàng nói, ngay cả những chi tiết nhỏ nhất.
    - **Ghi chép:** Ghi chép lại tất cả các thông tin thu thập được từ khách hàng.
    - **Xác nhận:** Xác nhận lại với khách hàng về những gì đã hiểu để đảm bảo rằng không có sự hiểu lầm.

5. Trong quá trình xây dựng mô hình nghiệp vụ, một số use case bị trùng lặp về chức năng.

- **Giải pháp:**
    - **Rà soát:** Rà soát kỹ lưỡng tất cả các use case để phát hiện các trường hợp trùng lặp.
    - **Hợp nhất:** Hợp nhất các use case trùng lặp thành một use case duy nhất, đảm bảo chức năng được mô tả đầy đủ và không bị thiếu sót.
    - **Loại bỏ:** Loại bỏ các use case không cần thiết hoặc không phù hợp với mục tiêu của dự án.

6. Đội phát triển và khách hàng có ý kiến khác nhau về ý nghĩa của một số từ khóa chuyên môn.

- **Giải pháp:**
    - **Thống nhất:** Tổ chức một buổi họp giữa đội phát triển và khách hàng để thống nhất về ý nghĩa của các từ khóa chuyên môn.
    - **Giải thích:** Giải thích rõ ràng ý nghĩa của từng từ khóa, sử dụng ví dụ minh họa để khách hàng dễ hiểu.
    - **Tài liệu:** Lập một tài liệu giải thích ý nghĩa của các từ khóa chuyên môn, để tất cả các bên liên quan đều có thể tham khảo.

7. Sau khi hoàn thành việc trích các use case, đội phát triển nhận ra rằng một số chức năng quan trọng bị bỏ sót.

- **Giải pháp:**
    - **Bổ sung:** Bổ sung các use case bị bỏ sót vào mô hình nghiệp vụ.

- **Đánh giá:** Đánh giá mức độ ảnh hưởng của việc bổ sung use case đến tiến độ và ngân sách dự án.
- **Điều chỉnh:** Điều chỉnh kế hoạch dự án nếu cần thiết để đảm bảo rằng các chức năng mới được thực hiện đúng cách.

8. Trong quá trình xây dựng mô hình nghiệp vụ, khách hàng yêu cầu bổ sung thêm một số chức năng mới.

- **Giải pháp:**
  - **Đánh giá:** Đánh giá mức độ ảnh hưởng của việc bổ sung chức năng mới đến tiến độ và ngân sách dự án.
  - **Ưu tiên:** Thống nhất với khách hàng về mức độ ưu tiên của các chức năng mới.
  - **Bổ sung:** Bổ sung các use case mới vào mô hình nghiệp vụ.
  - **Điều chỉnh:** Điều chỉnh kế hoạch dự án nếu cần thiết để đảm bảo rằng các chức năng mới được thực hiện đúng cách.

9. Một nhóm phát triển gặp khó khăn trong việc mô tả chi tiết các use case vì chưa hiểu rõ quy trình nghiệp vụ.

- **Giải pháp:**
  - **Tìm hiểu:** Dành thời gian tìm hiểu kỹ lưỡng quy trình nghiệp vụ của khách hàng.
  - **Phỏng vấn:** Phỏng vấn các nhân viên của khách hàng để hiểu rõ hơn về cách họ thực hiện công việc.
  - **Quan sát:** Quan sát trực tiếp cách nhân viên của khách hàng làm việc để hiểu rõ hơn về quy trình nghiệp vụ.
  - **Tài liệu:** Tham khảo các tài liệu về quy trình nghiệp vụ của khách hàng.

10. Khách hàng yêu cầu thêm một chức năng mới khi hệ thống đã bước vào giai đoạn thiết kế chi tiết.

- **Giải pháp:**
  - **Đánh giá:** Đánh giá mức độ ảnh hưởng của việc bổ sung chức năng mới đến tiến độ và ngân sách dự án.
  - **Ưu tiên:** Thống nhất với khách hàng về mức độ ưu tiên của các chức năng mới.
  - **Thay đổi:** Thực hiện thay đổi đối với thiết kế hệ thống để đáp ứng yêu cầu mới của khách hàng.
  - **Kiểm tra:** Kiểm tra kỹ lưỡng hệ thống sau khi thực hiện thay đổi để đảm bảo rằng không có lỗi xảy ra.

CÂU HỎI TÌNH HUỐNG (1-2)

1.Tình huống 1: Xác định phạm vi dự án

- Thu thập yêu cầu chi tiết: Tổ chức các buổi họp với khách hàng để thu thập thông tin chi tiết về cả hệ thống quản lý khách hàng (CRM) và hệ thống quản lý kho (Inventory Management System). Tìm hiểu rõ mục tiêu, chức năng mong muốn của từng hệ thống.
- Xác định ranh giới: Phân tích và xác định rõ ràng những chức năng nào thuộc phạm vi của dự án ban đầu, những chức năng nào có thể được xem xét trong giai đoạn sau.
- Ưu tiên: Thảo luận với khách hàng để ưu tiên các chức năng cần thiết cho giai đoạn đầu của dự án.
- Lập tài liệu: Ghi lại phạm vi dự án đã được thống nhất, bao gồm các chức năng chính, mục tiêu và giới hạn của dự án.
- Thỏa thuận: Đảm bảo khách hàng hiểu và đồng ý với phạm vi dự án đã được xác định.

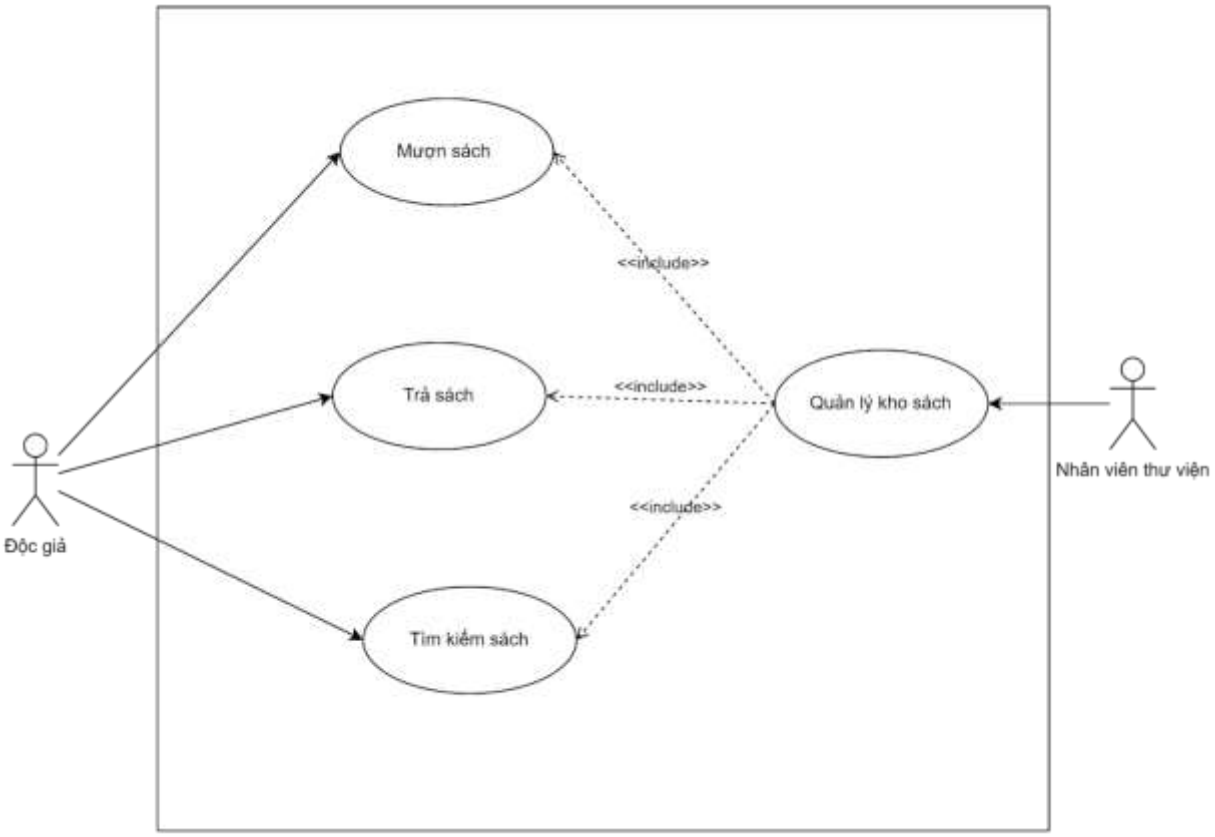
2.Tình huống 2: Phỏng vấn khách hàng

- "Hệ thống 'dễ sử dụng' theo ý anh/chị là như thế nào? Ví dụ, giao diện trực quan, thao tác đơn giản, hay có hướng dẫn chi tiết?"
- "Hệ thống 'nhanh chóng' ở đây được hiểu là gì? Thời gian phản hồi cho mỗi thao tác là bao lâu? Có yêu cầu cụ thể về tốc độ xử lý không?"
- "Anh/chị có thể cho tôi xem một số hệ thống đặt vé trực tuyến mà anh/chị thấy dễ sử dụng và nhanh chóng được không?"
- "Những tính năng cụ thể nào mà anh/chị muốn có trong hệ thống đặt vé trực tuyến này?"
- "Đối tượng người dùng chính của hệ thống này là ai? Họ có những yêu cầu đặc biệt nào về giao diện và chức năng?"

3.Tình huống 3: Viết tài liệu SRS

- Mục tiêu và phạm vi: Mô tả mục tiêu của hệ thống, các chức năng chính và phạm vi của dự án.
- Mô tả chức năng: Liệt kê chi tiết các chức năng của hệ thống, bao gồm cả chức năng người dùng và chức năng quản trị.
- Yêu cầu người dùng: Mô tả các yêu cầu về giao diện, trải nghiệm người dùng, và các tương tác của người dùng với hệ thống.
- Yêu cầu phi chức năng: Mô tả các yêu cầu về hiệu năng, bảo mật, độ tin cậy, khả năng mở rộng, và các yêu cầu khác không liên quan trực tiếp đến chức năng.
- Các ràng buộc: Liệt kê các ràng buộc về kỹ thuật, ngân sách, thời gian, và các yếu tố khác có thể ảnh hưởng đến dự án.
- Định nghĩa thuật ngữ: Giải thích các thuật ngữ chuyên môn được sử dụng trong tài liệu.

4.Tình huống 4:



5.Tình huống 5: Quản lý thay đổi yêu cầu

- Đánh giá ảnh hưởng: Đánh giá mức độ ảnh hưởng của tính năng đặt bàn trực tuyến đến tiến độ, ngân sách và các chức năng khác của dự án.
- Thảo luận với khách hàng: Thảo luận với khách hàng về chi phí và thời gian cần thiết để thực hiện thay đổi này.
- Ưu tiên: Xác định mức độ ưu tiên của tính năng mới so với các tính năng khác trong dự án.
- Lập kế hoạch: Lập kế hoạch cụ thể để thực hiện thay đổi này, bao gồm cả việc điều chỉnh lịch trình và phân bổ nguồn lực.
- Thỏa thuận: Đảm bảo khách hàng hiểu và đồng ý với các thay đổi đã được thống nhất.

6.Tình huống 6: Phát hiện rủi ro

Những rủi ro có thể xảy ra và giải pháp giảm thiểu:

- Rủi ro: Không thể đồng bộ dữ liệu giữa ứng dụng di động và máy chủ.
- Giải pháp:

    - Nghiên cứu và lựa chọn công nghệ đồng bộ dữ liệu phù hợp (ví dụ: REST API, WebSockets).
    - Xây dựng cơ chế xử lý lỗi và khôi phục dữ liệu khi có sự cố.
    - Kiểm tra và thử nghiệm kỹ lưỡng quá trình đồng bộ dữ liệu.
- Rủi ro: Ứng dụng di động không tương thích với nhiều loại thiết bị.
- Giải pháp:
    - Thiết kế ứng dụng theo hướng responsive design để tương thích với nhiều kích cỡ màn hình.
    - Kiểm tra ứng dụng trên nhiều thiết bị khác nhau.
- Rủi ro: Vấn đề bảo mật dữ liệu người dùng.
- Giải pháp:
    - Áp dụng các biện pháp bảo mật như mã hóa dữ liệu, xác thực người dùng, và kiểm soát truy cập.
    - Tuân thủ các tiêu chuẩn bảo mật và quy định về bảo vệ dữ liệu cá nhân.

7.Tình huống 7: Viết test case

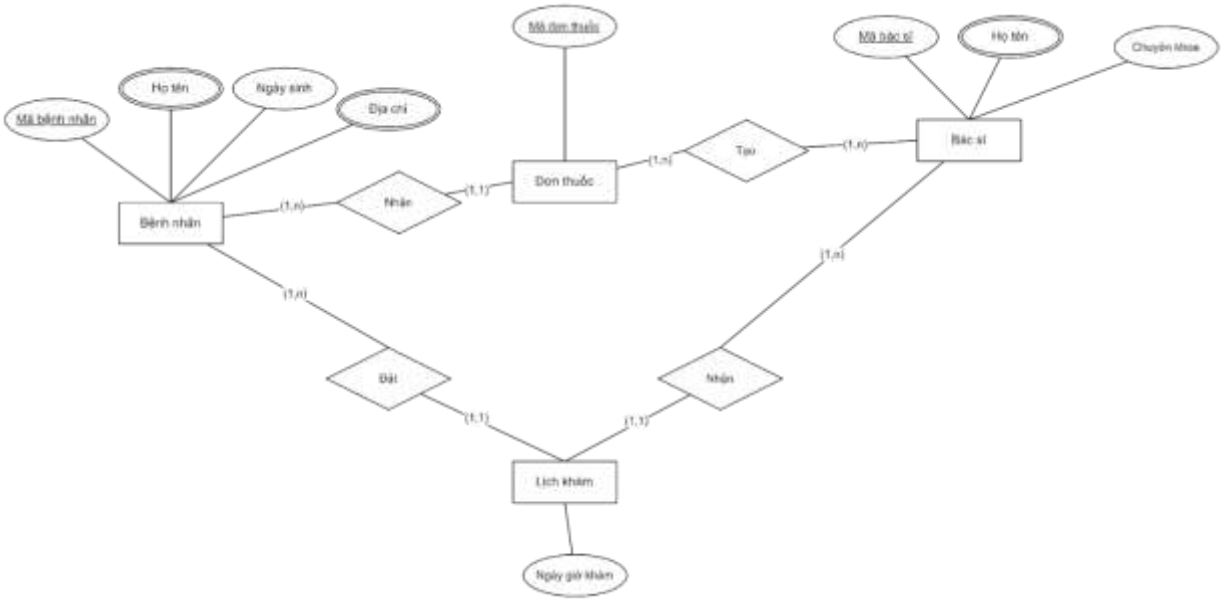
- Test case 1: Đặt vé thành công với đầy đủ thông tin và thanh toán bằng thẻ tín dụng.
- Test case 2: Đặt vé không thành công do thiếu thông tin (ví dụ: thiếu số điện thoại).
- Test case 3: Đặt vé không thành công do thanh toán không thành công (ví dụ: thẻ tín dụng hết hạn).
- Test case 4: Đặt vé thành công với mã giảm giá hợp lệ.
- Test case 5: Đặt vé không thành công với mã giảm giá không hợp lệ.
- Test case 6: Kiểm tra chức năng hủy vé.

8.Tình huống 8: Nghiệm thu phần mềm

- Thu thập phản hồi chi tiết: Tổ chức các buổi gặp mặt với nhân viên để thu thập thông tin chi tiết về những khó khăn họ gặp phải khi sử dụng hệ thống.
- Phân tích phản hồi: Phân tích các phản hồi để xác định những vấn đề cụ thể cần được giải quyết.
- Đề xuất giải pháp: Đề xuất các giải pháp để cải thiện tính dễ sử dụng của hệ thống, ví dụ như đơn giản hóa giao diện, cung cấp hướng dẫn sử dụng chi tiết, hoặc tổ chức đào tạo cho nhân viên.
- Thực hiện cải tiến: Thực hiện các cải tiến dựa trên các giải pháp đã được đề xuất.
- Kiểm tra và đánh giá: Sau khi cải tiến, kiểm tra và đánh giá lại hệ thống để đảm bảo rằng nó đã đáp ứng được yêu cầu của người dùng.

- Nghiệm thu: Tổ chức nghiệm thu chính thức với sự tham gia của đại diện người dùng và các bên liên quan.

10.Tình huống 10: Phân tích nghiệp vụ với ERD

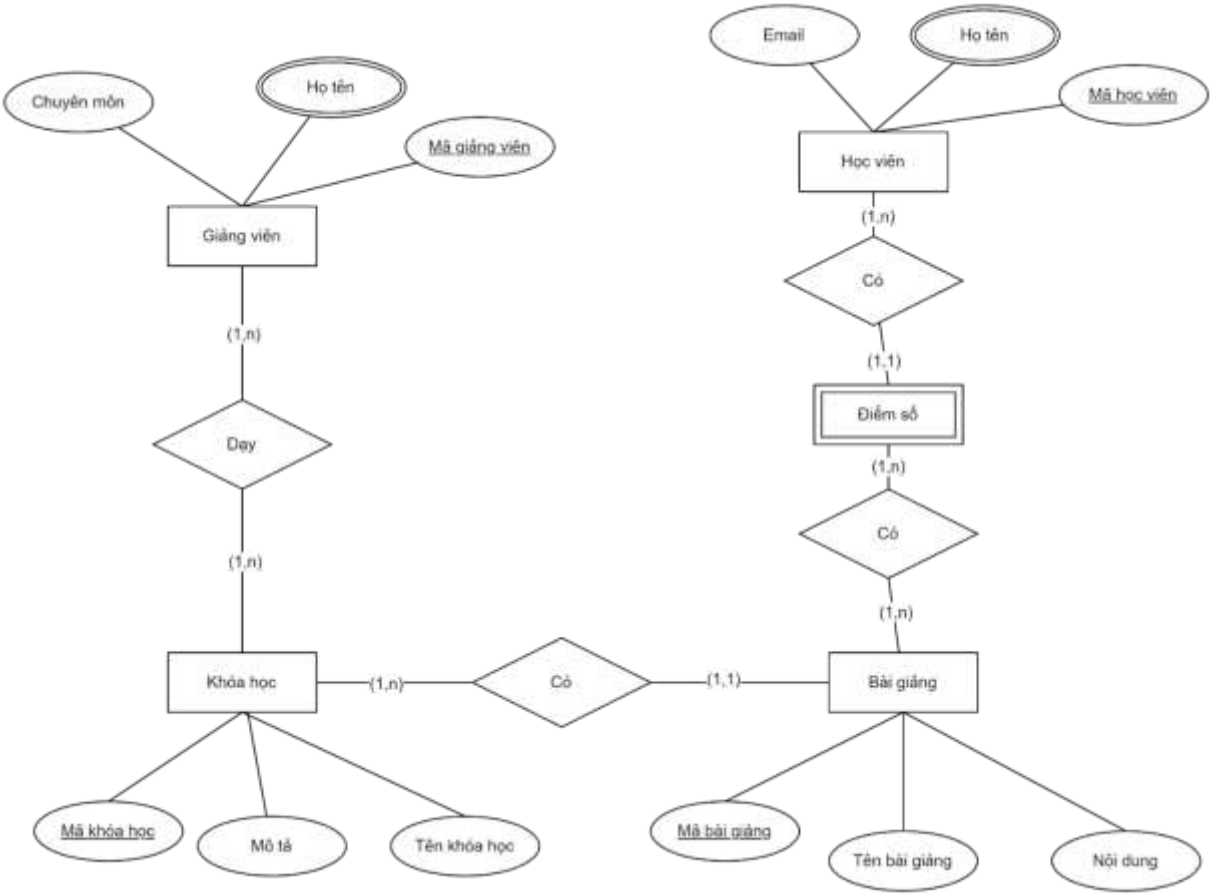


11.Tình huống 11: Lựa chọn mô hình kiến trúc

- Ưu điểm của Microservices:
    - Khả năng mở rộng: Dễ dàng mở rộng từng dịch vụ riêng lẻ khi cần thiết, không ảnh hưởng đến toàn bộ hệ thống.
    - Tính ổn định: Nếu một dịch vụ gặp sự cố, các dịch vụ khác vẫn hoạt động bình thường.
    - Linh hoạt trong công nghệ: Cho phép sử dụng các công nghệ khác nhau cho từng dịch vụ, phù hợp với yêu cầu cụ thể.
    - Phát triển và triển khai độc lập: Các nhóm phát triển có thể làm việc độc lập trên từng dịch vụ, giúp tăng tốc độ phát triển và triển khai.
    - Dễ dàng bảo trì và nâng cấp: Việc bảo trì và nâng cấp một dịch vụ sẽ ít ảnh hưởng đến các dịch vụ khác.
- Tại sao không chọn các mô hình khác:

- Monolithic: Khó mở rộng, cập nhật và bảo trì khi hệ thống lớn mạnh.
- Client-Server: Chỉ tập trung vào việc chia sẻ tài nguyên giữa client và server, không đề cập đến cấu trúc bên trong của hệ thống.
- MVC: Tách biệt các thành phần của ứng dụng (Model, View, Controller), nhưng vẫn có thể được triển khai dưới dạng monolithic hoặc microservices.

12. Tình huống 12:



13.Tình huống 13: Phân tích chi phí - lợi ích

- Ước tính chi phí:
  - Chi phí phát triển ứng dụng (500,000 USD).
  - Chi phí bảo trì và nâng cấp ứng dụng.
  - Chi phí marketing và quảng bá ứng dụng.
  - Chi phí hỗ trợ khách hàng.
  - Các chi phí khác (nếu có).
- Ước tính lợi ích:

    - Tăng doanh thu từ việc đặt dịch vụ trực tuyến.
    - Tiết kiệm chi phí vận hành (ví dụ: giảm chi phí thuê nhân viên).
    - Nâng cao trải nghiệm khách hàng và tăng sự hài lòng.
    - Tăng cường khả năng cạnh tranh.
- So sánh chi phí và lợi ích:
    - Tính toán giá trị hiện tại ròng (NPV) của dự án.
    - Tính toán tỷ suất hoàn vốn nội bộ (IRR) của dự án.
    - So sánh các chỉ số này với các tiêu chuẩn của doanh nghiệp.
- Đánh giá rủi ro:
    - Xác định các rủi ro có thể ảnh hưởng đến dự án (ví dụ: rủi ro công nghệ, rủi ro thị trường).
    - Đánh giá mức độ ảnh hưởng của từng rủi ro.
    - Đề xuất các biện pháp giảm thiểu rủi ro.
- Đưa ra khuyến nghị:
    - Dựa trên kết quả phân tích chi phí - lợi ích và đánh giá rủi ro, đưa ra khuyến nghị cho doanh nghiệp về việc có nên phát triển ứng dụng di động hay không.

14. Tình huống 14: Đánh giá rủi ro của dự án

- Đào tạo và nâng cao năng lực:
    - Tổ chức các khóa đào tạo chuyên sâu về tích hợp API tài chính cho đội ngũ phát triển.
    - Thuê chuyên gia tư vấn hoặc cố vấn có kinh nghiệm trong lĩnh vực này.
- Tìm hiểu và lựa chọn API phù hợp:
    - Nghiên cứu và đánh giá các API tài chính khác nhau để lựa chọn API phù hợp nhất với yêu cầu của dự án.
    - Đảm bảo API được lựa chọn có tài liệu rõ ràng, hỗ trợ tốt và đã được kiểm chứng.
- Xây dựng môi trường thử nghiệm:

- Thiết lập một môi trường thử nghiệm riêng biệt để đội ngũ phát triển có thể thực hành và làm quen với việc tích hợp API.
- Sử dụng dữ liệu thử nghiệm để tránh ảnh hưởng đến dữ liệu thật.

- Kiểm thử kỹ lưỡng:
  - Thực hiện kiểm thử tích hợp kỹ lưỡng để đảm bảo rằng hệ thống hoạt động đúng như mong đợi sau khi tích hợp API.
  - Kiểm tra các trường hợp lỗi và xử lý ngoại lệ.
- Lập kế hoạch dự phòng:
  - Lập kế hoạch dự phòng cho các tình huống xấu nhất có thể xảy ra (ví dụ: API bị lỗi, hệ thống không ổn định).
  - Chuẩn bị các phương án thay thế để đảm bảo dự án không bị gián đoạn.

15. Tình huống 15: Viết kịch bản kiểm thử (Test Case)

- **Test case 1: Tìm kiếm khách sạn theo địa điểm và ngày đến/đi.**
- **Test case 2: Lọc kết quả tìm kiếm theo giá, loại phòng, tiện nghi.**
- **Test case 3: Xem chi tiết khách sạn (hình ảnh, mô tả, tiện nghi, đánh giá).**
- **Test case 4: Chọn loại phòng và số lượng phòng.**
- **Test case 5: Nhập thông tin khách hàng (họ tên, email, số điện thoại).**
- **Test case 6: Chọn phương thức thanh toán (thẻ tín dụng, chuyển khoản).**
- **Test case 7: Xác nhận đặt phòng.**
- **Test case 8: Hủy đặt phòng.**
- **Test case 9: Kiểm tra email xác nhận đặt phòng.**
- **Test case 10: Đặt phòng với mã giảm giá.**

16. Tình huống 16: Kiểm thử hiệu năng (Performance Testing)

- Kiểm thử tải (Load Testing):
  - Mô phỏng 10,000 người dùng truy cập và thực hiện giao dịch đồng thời để xem hệ thống có thể xử lý được không.
  - Đo các chỉ số hiệu năng như thời gian phản hồi, số lượng giao dịch thành công, và mức sử dụng tài nguyên (CPU, RAM).
- Kiểm thử chịu tải (Stress Testing):

    - Tăng dần số lượng người dùng và giao dịch để tìm ra giới hạn chịu tải của hệ thống.
    - Xác định điểm mà hệ thống bắt đầu gặp sự cố hoặc ngừng hoạt động.
- Kiểm thử độ bền (Endurance Testing):
    - Cho hệ thống hoạt động liên tục trong một thời gian dài (ví dụ: vài giờ hoặc vài ngày) để xem có vấn đề gì xảy ra không.
    - Kiểm tra xem hệ thống có ổn định và có thể duy trì hiệu năng tốt trong thời gian dài hay không.
- Kiểm thử tăng tải (Spike Testing):
    - Tăng đột ngột số lượng người dùng và giao dịch để xem hệ thống có thể xử lý được không.
    - Đánh giá khả năng phục hồi của hệ thống sau khi có sự cố.
- 17. Tình huống 17: Chuyển đổi từ Waterfall sang Agile
- Đánh giá hiện trạng:
    - Phân tích các vấn đề mà mô hình Waterfall đang gặp phải.
    - Đánh giá mức độ sẵn sàng của nhóm cho việc chuyển đổi sang Agile.
- Lựa chọn phương pháp Agile phù hợp:
    - Nghiên cứu và lựa chọn phương pháp Agile phù hợp với đặc thù của công ty (ví dụ: Scrum, Kanban).
- Đào tạo và huấn luyện:
    - Tổ chức các buổi đào tạo về Agile và phương pháp đã chọn cho tất cả thành viên trong nhóm.
    - Cung cấp huấn luyện và hỗ trợ trong quá trình chuyển đổi.
- Thay đổi dần dần:
    - Không nên chuyển đổi toàn bộ dự án sang Agile ngay lập tức.
    - Bắt đầu với một dự án nhỏ hoặc một phần của dự án lớn để làm quen với Agile.
- Thiết lập vai trò và trách nhiệm:
    - Xác định rõ vai trò và trách nhiệm của từng thành viên trong nhóm theo mô hình Agile.
    - Ví dụ: Product Owner, Scrum Master, Development Team.
- Xây dựng văn hóa Agile:
    - Khuyến khích sự hợp tác, giao tiếp cởi mở và tinh thần tự giác trong nhóm.

- Tạo môi trường làm việc linh hoạt và hỗ trợ lẫn nhau.

- Liên tục cải tiến:
  - Thường xuyên đánh giá và cải tiến quy trình làm việc để đạt hiệu quả cao nhất.
  - Học hỏi từ kinh nghiệm và điều chỉnh phương pháp Agile cho phù hợp.

18. Tình huống 18: Lập kế hoạch Sprint trong Scrum

1. Họp Sprint Planning:
   - Mời Product Owner và Development Team tham gia.
   - Product Owner trình bày Product Backlog và ưu tiên các User Story cho Sprint.
   - Development Team thảo luận và ước tính Story Points cho từng User Story.
   - Thống nhất Sprint Goal (mục tiêu của Sprint).
2. Chọn User Stories:
   - Development Team chọn các User Story từ Product Backlog vào Sprint Backlog dựa trên năng suất của nhóm (40 Story Points).
   - Ưu tiên các User Story có giá trị cao nhất và phù hợp với Sprint Goal.
3. Lập kế hoạch chi tiết:
   - Development Team chia nhỏ các User Story thành các Task nhỏ hơn.
   - Ước tính thời gian thực hiện cho từng Task.
   - Phân công Task cho từng thành viên.
4. Xem xét các yếu tố:
   - Năng suất của nhóm (40 Story Points/Sprint).
   - Sprint Goal.
   - Độ phức tạp của các User Story.
   - Thời gian có sẵn của các thành viên.
   - Các ràng buộc khác (ví dụ: deadline, ngân sách).
5. Kết thúc Sprint Planning:
   - Sprint Backlog được hoàn thành và mọi người đều hiểu rõ kế hoạch Sprint.

19. Tình huống 19: Đảm bảo bảo mật dữ liệu người dùng

- Thu thập dữ liệu tối thiểu:
  - Chỉ thu thập dữ liệu cần thiết cho mục đích sử dụng của ứng dụng.
  - Không thu thập dữ liệu nhạy cảm nếu không thực sự cần thiết.

- Mã hóa dữ liệu:
    - Mã hóa dữ liệu người dùng khi lưu trữ và truyền tải.
    - Sử dụng các thuật toán mã hóa mạnh mẽ.
- Kiểm soát truy cập:
    - Phân quyền truy cập dữ liệu chặt chẽ.
    - Chỉ cho phép những người có thẩm quyền truy cập vào dữ liệu.
- Bảo vệ dữ liệu khỏi truy cập trái phép:
    - Sử dụng tường lửa, hệ thống phát hiện xâm nhập và các biện pháp bảo mật khác để bảo vệ hệ thống.
- Tuân thủ các quy định của GDPR:
    - Đảm bảo ứng dụng tuân thủ các quy định về bảo vệ dữ liệu cá nhân của GDPR.
    - Ví dụ: cung cấp thông tin rõ ràng về việc thu thập và sử dụng dữ liệu, cho phép người dùng truy cập, chỉnh sửa và xóa dữ liệu của họ.
- Xây dựng chính sách bảo mật:
    - Xây dựng chính sách bảo mật rõ ràng và minh bạch.
    - Công bố chính sách bảo mật trên ứng dụng để người dùng hiểu rõ về cách thức bảo vệ dữ liệu của họ.

20. Tình huống 20: Xử lý tấn công bảo mật

- Ngừng hệ thống:
    - Ngay lập tức ngừng hệ thống để ngăn chặn hacker tiếp tục truy cập và đánh cắp dữ liệu.
- Điều tra:
    - Tìm hiểu nguyên nhân và cách thức tấn công.
    - Xác định mức độ thiệt hại và dữ liệu bị lộ.
- Khắc phục lỗ hổng:
    - Sửa chữa lỗ hổng SQL Injection bằng cách sử dụng Prepared Statements hoặc Parameterized Queries.
    - Kiểm tra và vá các lỗ hổng bảo mật khác (nếu có).
- Khôi phục dữ liệu:
    - Khôi phục dữ liệu bị đánh cắp (nếu có thể).
    - Thông báo cho khách hàng về việc dữ liệu của họ bị lộ.

- Tăng cường bảo mật:
    - Áp dụng các biện pháp bảo mật mạnh mẽ hơn để ngăn chặn các cuộc tấn công tương tự trong tương lai.
    - Ví dụ: sử dụng tường lửa, hệ thống phát hiện xâm nhập, kiểm tra bảo mật thường xuyên.
- Phân tích nguyên nhân gốc rễ:
    - Tìm hiểu nguyên nhân gốc rễ dẫn đến lỗ hổng SQL Injection.
    - Đảm bảo rằng các biện pháp phòng ngừa được áp dụng cho tất cả các ứng dụng và hệ thống khác.
- Học hỏi kinh nghiệm:
    - Rút ra bài học kinh nghiệm từ vụ tấn công này để cải thiện quy trình phát triển và bảo mật phần mềm.