Câu hỏi trắc nghiệm

Câu 1: B. Pha lấy yêu cầu

Câu 2: B. Mô hình lặp và tăng trưởng

Câu 3: B. Sửa lỗi và cập nhật tính năng mới

Câu 4: B. Dự án có yêu cầu rõ ràng và ít thay đổi

Câu 5: C. Một phiên bản phần mềm nhỏ

Câu 6: C. Khó kiểm soát chất lượng

Câu 7**:** B. Mô hình bản mẫu nhanh

Câu 8**:** A. Pha bảo trì

Câu 9**:** B. Mô hình lặp và tăng trưởng phát triển theo từng đợt nhỏ

Câu 10**:** D. Mô hình tiến trình linh hoạt

Câu trả lời 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.
-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?

Mô hình thác nước là một mô hình phát triển phần mềm tuyến tính, trong đó quá trình phát triển diễn ra theo các giai đoạn tuần tự và không có sự quay lại. Các giai đoạn chính bao gồm:

Thu thập yêu cầu: Xác định yêu cầu hệ thống từ khách hàng.

Thiết kế: Thiết kế hệ thống dựa trên yêu cầu đã xác định.

Lập trình: Phát triển phần mềm theo thiết kế.

Kiểm thử: Kiểm tra phần mềm để phát hiện lỗi.

Triển khai: Cài đặt phần mềm cho người dùng.

Bảo trì: Cập nhật và sửa lỗi phần mềm sau khi triển khai.

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 qua nhiều vòng lặp, mỗi vòng bổ sung tính năng mới và có thể thay đổi yêu cầu trong suốt quá trình.

Mô hình thác nước là tuyến tính, phát triển qua các giai đoạn cố định, và không cho phép thay đổi yêu cầu khi đã qua mỗi giai đoạn.

4.Mục tiêu của pha bảo trì là gì?

Đảm bảo phần mềm tiếp tục hoạt động ổn định và đáp ứng các yêu cầu mới.

5.Mô hình xây và sửa có nhược điểm gì?

Nhược điểm của mô hình xây và sửa (Build and Fix Model):

- **Thiếu cấu trúc và quy trình**: Không có thiết kế hay kế hoạch rõ ràng, gây khó khăn trong quản lý dự án.
- **Khó kiểm soát chất lượng**: Sửa lỗi sau khi viết mã dễ bỏ sót vấn đề, ảnh hưởng đến chất lượng.
- **Khó bảo trì và mở rộng**: Thiếu tài liệu, việc nâng cấp và bảo trì trở nên phức tạp.

6.Mô hình bản mẫu nhanh là gì?
Mô hình bản mẫu nhanh là phương pháp phát triển phần mềm trong đó một bản mẫu được xây dựng nhanh chóng để khách hàng hoặc người dùng thử nghiệm và cung cấp phản hồi. Bản mẫu này sẽ được cải tiến qua các vòng lặp cho đến khi đáp ứng đầy đủ yêu cầu của khách hàng.
7.Pha giải thể là gì?
Mô hình bản mẫu nhanh là phương pháp phát triển phần mềm trong đó một bản mẫu của hệ thống được xây dựng nhanh chóng để người dùng hoặc khách hàng có thể thử nghiệm và cung cấp phản hồi. Sau đó, bản mẫu này sẽ được cải tiến và điều chỉnh qua nhiều vòng lặp cho đến khi đáp ứng đủ yêu cầu của người dùng.
8.Mô hình xoắn ốc là gì?
Mô hình xoắn ốc là một phương pháp phát triển phần mềm kết hợp giữa các yếu tố của mô hình thác nước và mô hình lặp và tăng trưởng. Mô hình này tập trung vào việc quản lý rủi ro và tiến hành phát triển qua các vòng lặp, mỗi vòng lặp bao gồm các giai đoạn như lập kế hoạch, phân tích rủi ro, thiết kế, xây dựng, kiểm thử và đánh giá. Mỗi

vòng lặp cải tiến sản phẩm và có thể điều chỉnh kế hoạch dựa trên phản hồi và rủi ro phát sinh.
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 khắc phục được nhiều nhược điểm của các mô hình phát triển phần mềm truyền thống, như mô hình thác nước (Waterfall). Những lý do chính bao gồm:

- **Linh hoạt và thích ứng nhanh với thay đổi:** Agile cho phép nhóm phát triển phản hồi nhanh với các thay đổi từ khách hàng hoặc thị trường mà không làm gián đoạn quy trình làm việc.
- **Phát triển theo từng giai đoạn nhỏ (Iterative Development):** Thay vì triển khai toàn bộ sản phẩm một lần, Agile chia dự án thành các phần nhỏ (sprint hoặc iteration) giúp kiểm soát tốt hơn tiến độ và chất lượng.
- **Cải thiện chất lượng sản phẩm:** Việc kiểm thử liên tục và phản hồi thường xuyên giúp phát hiện lỗi sớm, đảm bảo phần mềm luôn đáp ứng yêu cầu.
- **Cộng tác tốt hơn giữa các bên liên quan:** Agile khuyến khích giao tiếp thường xuyên giữa khách hàng, nhóm phát triển và các bên liên quan để đảm bảo sản phẩm đáp ứng đúng nhu cầu thực tế.
- **Tăng hiệu suất làm việc của nhóm:** Agile đề cao tinh thần làm việc nhóm, tự quản lý và khuyến khích sáng tạo, giúp nâng cao năng suất và động lực làm việc.
- **Giảm rủi ro:** Vì sản phẩm được phát triển và kiểm thử liên tục, các lỗi hoặc sai sót được phát hiện sớm, giúp giảm nguy cơ thất bại khi triển khai.

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

Mô hình mã nguồn mở có sự khác biệt rõ ràng so với các mô hình phần mềm khác, đặc biệt là phần mềm độc quyền (proprietary software). Một số điểm khác biệt chính bao gồm:

- **Quyền truy cập mã nguồn:**
    - Mã nguồn mở (Open Source): Mọi người đều có thể xem, chỉnh sửa và phân phối lại mã nguồn.
    - Mã nguồn đóng (Closed Source/Proprietary): Chỉ nhà phát triển hoặc tổ chức sở hữu mới có quyền truy cập và chỉnh sửa.
- **Giấy phép sử dụng:**
    - Mã nguồn mở: Thường được phân phối theo các giấy phép như GNU GPL, MIT, Apache, cho phép sử dụng tự do, miễn phí hoặc có điều kiện nhất định.

    - Mã nguồn đóng: Người dùng phải mua hoặc có giấy phép sử dụng, bị hạn chế quyền can thiệp vào mã nguồn.
- **Cộng đồng phát triển:**
    - Mã nguồn mở: Được duy trì và phát triển bởi một cộng đồng lập trình viên toàn cầu. Mọi người có thể đóng góp, sửa lỗi và cải tiến sản phẩm.
    - Mã nguồn đóng: Chỉ được phát triển và bảo trì bởi công ty hoặc tổ chức sở hữu phần mềm.
- **Tính bảo mật và ổn định:**
    - Mã nguồn mở: Do có nhiều người tham gia kiểm tra và sửa lỗi, các lỗ hổng bảo mật có thể được phát hiện và khắc phục nhanh hơn.
    - Mã nguồn đóng: Việc sửa lỗi và cập nhật phụ thuộc vào nhà phát triển, có thể mất nhiều thời gian.
- **Chi phí sử dụng:**
    - Mã nguồn mở: Thường miễn phí hoặc có phí hỗ trợ kỹ thuật.
    - Mã nguồn đóng: Phải trả tiền để mua bản quyền hoặc thuê dịch vụ.

Nhìn chung, mã nguồn mở thúc đẩy sự sáng tạo, chia sẻ kiến thức và giảm chi phí, trong khi mã nguồn đóng đảm bảo tính thương mại hóa và kiểm soát chặt chẽ hơn đối với sản phẩm.

Câu thảo luận nhóm

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

Mô hình thác nước:

Ưu điểm:

- **Quy trình rõ ràng:** Các giai đoạn phát triển được xác định rõ ràng, dễ theo dõi.
- **Dễ quản lý:** Phù hợp với dự án có yêu cầu và phạm vi ổn định, dễ quản lý tiến độ.
- **Thực hiện đơn giản:** Quá trình thực hiện dễ dàng khi các yêu cầu và thiết kế không thay đổi nhiều.

Nhược điểm:

- **Khó thay đổi yêu cầu:** Không linh hoạt khi phải thay đổi yêu cầu trong suốt quá trình phát triển.
- **Rủi ro phát hiện lỗi muộn:** Lỗi và thiếu sót chỉ được phát hiện sau khi hoàn thành các giai đoạn trước, khó khắc phục sớm.
- **Không phù hợp với dự án lớn, phức tạp:** Nếu yêu cầu thay đổi thường xuyên, mô hình này có thể gặp khó khăn.

Mô hình xoắn ốc:

Ưu điểm:

- **Linh hoạt:** Có thể điều chỉnh và thay đổi yêu cầu qua mỗi vòng xoắn.
- **Quản lý rủi ro tốt:** Giúp nhận diện và xử lý rủi ro từ sớm, do dự án được chia thành các vòng nhỏ.
- **Phù hợp với dự án phức tạp:** Thích hợp cho dự án lớn, phức tạp với yêu cầu thay đổi liên tục.

Nhược điểm:

- **Chi phí cao:** Quá trình phân tích và lập kế hoạch nhiều vòng xoắn có thể tốn kém và thời gian.
- **Phức tạp:** Quản lý và triển khai có thể phức tạp do cần liên tục tái đánh giá và điều chỉnh.
- **Khó xác định phạm vi hoàn thiện:** Đôi khi khó xác định được thời gian hoàn thiện dự án do tính chất linh hoạt và lặp lại.

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

Tình huống: Phát triển ứng dụng di động cho khách hàng

Một công ty phát triển phần mềm nhận hợp đồng phát triển một ứng dụng di động cho một khách hàng trong lĩnh vực dịch vụ du lịch. Khách hàng muốn có một ứng dụng cho phép người dùng tìm kiếm các chuyến du lịch, đặt tour, thanh toán và đánh giá dịch vụ.

Tuy nhiên, khách hàng không hoàn toàn chắc chắn về các tính năng cụ thể sẽ được tích hợp trong ứng dụng, và yêu cầu có thể thay đổi tùy thuộc vào phản hồi từ người dùng trong quá trình thử nghiệm.

Áp dụng mô hình lặp và tăng trưởng:

Giai đoạn 1: Phiên bản đầu tiên (vòng lặp 1)

- Phát triển các tính năng cơ bản như tìm kiếm tour du lịch, đăng ký và đăng nhập người dùng.
- Triển khai phiên bản đầu tiên cho khách hàng và người dùng thử nghiệm.

Giai đoạn 2: Cải tiến và bổ sung tính năng (vòng lặp 2)

- Dựa trên phản hồi từ người dùng và khách hàng, tiếp tục phát triển thêm tính năng như đặt tour, thanh toán trực tuyến, và đánh giá dịch vụ.

Giai đoạn 3: Tinh chỉnh và tối ưu

- Sau mỗi vòng lặp, nhóm phát triển cải tiến tính năng, khắc phục lỗi, và bổ sung các yêu cầu mới dựa trên phản hồi của người dùng.

Lý do áp dụng mô hình lặp và tăng trưởng:

- **Linh hoạt thay đổi yêu cầu:** Khách hàng có thể thay đổi yêu cầu sau mỗi vòng lặp, giúp đội ngũ phát triển dễ dàng điều chỉnh mà không phải làm lại từ đầu.
- **Phát triển nhanh chóng:** Các tính năng cơ bản được phát triển và cung cấp sớm, cho phép người dùng thử nghiệm và phản hồi nhanh chóng.
- **Giảm rủi ro:** Việc phát triển theo từng vòng giúp phát hiện lỗi và vấn đề sớm, giảm thiểu rủi ro trong suốt quá trình phát triển.

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

Mô hình xây và sửa không phù hợp với các dự án lớn vì:

Khó kiểm soát: Khi thay đổi liên tục trong quá trình phát triển, việc quản lý các thay đổi và cập nhật trở nên phức tạp, đặc biệt khi dự án có quy mô lớn.

Tốn thời gian và chi phí: Việc xây dựng lại và sửa chữa nhiều lần dẫn đến lãng phí tài nguyên, thời gian và chi phí.

Thiếu cấu trúc: Mô hình này thiếu kế hoạch rõ ràng, dễ dẫn đến việc phát triển mất phương hướng và không đạt được mục tiêu ban đầu của dự án.

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

Tiêu chí	**Bản mẫu nhanh (Rapid Prototyping)**	**Tiến trình linh hoạt (Agile Process)**
Mục tiêu chính	Xây dựng nhanh bản mẫu để xác nhận yêu cầu khách hàng.	Phát triển phần mềm theo từng giai đoạn ngắn, thích ứng với thay đổi.

Cách tiếp cận	Phát triển một nguyên mẫu (prototype), nhận phản hồi, rồi chỉnh sửa tiếp.	Chia dự án thành nhiều vòng lặp nhỏ (sprint) với sản phẩm tăng dần.
Tốc độ phát triển	Nhanh trong giai đoạn đầu, nhưng có thể tốn thời gian sửa đổi.	Tiến độ ổn định, mỗi sprint đều có kết quả cụ thể.
Xử lý thay đổi	Dễ dàng sửa đổi trong giai đoạn đầu nhưng khó khi đã triển khai.	Linh hoạt với thay đổi trong suốt quá trình phát triển.
Bảo trì & mở rộng	Không được tối ưu, cần tái cấu trúc lại khi phát triển bản chính thức.	Dễ mở rộng nhờ code được thiết kế có tổ chức ngay từ đầu.
Phạm vi dự án phù hợp	Dự án nhỏ, cần xác định rõ yêu cầu trước khi phát triển chính thức.	Dự án lớn, yêu cầu thay đổi liên tục, cần phản hồi nhanh.
Nhược điểm	- Có thể lãng phí thời gian nếu bản mẫu không được sử dụng. - Không tập trung vào kiến trúc phần mềm.	- Cần sự hợp tác chặt chẽ từ khách hàng. - Yêu cầu kỷ luật cao trong quản lý tiến trình.

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

1. Xác định và đánh giá rủi ro:

- Mỗi vòng lặp của mô hình xoắn ốc đều bắt đầu bằng việc xác định và đánh giá các rủi ro tiềm ẩn có thể ảnh hưởng đến dự án.
- Điều này bao gồm việc xác định các rủi ro về kỹ thuật, tài chính, tiến độ, và các rủi ro khác.

2. Lập kế hoạch giảm thiểu rủi ro:

- Sau khi xác định và đánh giá rủi ro, mô hình xoắn ốc tập trung vào việc lập kế hoạch giảm thiểu rủi ro.
- Điều này bao gồm việc đưa ra các chiến lược và biện pháp để giảm thiểu tác động tiêu cực của các rủi ro đã được xác định.

3. Theo dõi và kiểm soát rủi ro:

- Trong suốt quá trình phát triển, rủi ro được theo dõi và kiểm soát liên tục.
- Điều này giúp đảm bảo rằng các biện pháp giảm thiểu rủi ro đang được thực hiện hiệu quả và các rủi ro mới được phát hiện kịp thời.
- Việc theo dõi và kiểm soát rủi ro giúp dự án luôn đi đúng hướng và giảm thiểu nguy cơ thất bại.

4. Ra quyết định dựa trên rủi ro:

- Mô hình xoắn ốc cho phép đưa ra các quyết định dự án dựa trên đánh giá rủi ro.
- Việc ra quyết định dựa trên rủi ro giúp đảm bảo rằng dự án được thực hiện một cách an toàn và hiệu quả.

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 thay vì mô hình tiến trình linh hoạt trong các trường hợp sau:

- **Yêu cầu ổn định và rõ ràng**: Khi yêu cầu của khách hàng đã được xác định rõ ràng ngay từ đầu và không có khả năng thay đổi trong suốt quá trình phát triển.
- **Dự án quy mô nhỏ hoặc đơn giản**: Các dự án không quá phức tạp, có phạm vi và yêu cầu đơn giản, không cần thay đổi hay điều chỉnh liên tục.
- **Khách hàng có ít sự tham gia**: Khi khách hàng không có khả năng tham gia thường xuyên vào quá trình phát triển hoặc không cần phản hồi thường xuyên trong từng giai đoạn.
- **Tiến độ và kế hoạch rõ ràng**: Khi dự án yêu cầu một lịch trình cố định, các giai đoạn phát triển không bị gián đoạn và có thể thực hiện theo kế hoạch đã xác định.
- **Dự án có yêu cầu pháp lý hoặc chuẩn mực cao**: Các dự án yêu cầu tính chính xác, tuân thủ các quy định nghiêm ngặt và dễ dàng kiểm tra được từng giai đoạn.

Mô hình thác nước là lựa chọn tốt khi mọi thứ đều rõ ràng từ đầu và không cần thay đổi hoặc điều chỉnh thường xuyên, còn mô hình tiến trình linh hoạt phù hợp hơn với các dự án yêu cầu sự thay đổi linh hoạt trong quá trình phát triển.

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

- Không có cam kết hỗ trợ từ nhà phát triển chính thức như phần mềm thương mại.
- Phụ thuộc vào cộng đồng để sửa lỗi, cập nhật, có thể mất thời gian.

- Mã nguồn mở có thể bị tin tặc lợi dụng để tìm lỗ hổng bảo mật.
- Nếu không có đội ngũ kiểm tra chặt chẽ, phần mềm có thể dễ bị tấn công.
- Nhiều dự án mã nguồn mở có tài liệu không đầy đủ, khó sử dụng.
- Việc tích hợp vào hệ thống doanh nghiệp có thể gặp khó khăn do thiếu tài liệu hướng dẫn chi tiết.
- Do nhiều người đóng góp, chất lượng mã nguồn không đồng đều.
- Không có tiêu chuẩn chung trong phát triển có thể gây khó khăn khi bảo trì.
- Nhiều phần mềm mã nguồn mở có thể không tương thích với hệ thống hiện có.
- Thiếu dịch vụ hỗ trợ chính thức làm giảm khả năng triển khai ở quy mô lớn.
- Có nhiều loại giấy phép mã nguồn mở khác nhau (GPL, MIT, Apache…), cần hiểu rõ trước khi sử dụng.
- Nguy cơ vi phạm bản quyền nếu không tuân thủ giấy phép khi kết hợp với phần mềm thương mại.

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 bằng cách áp dụng các nguyên tắc phát triển lặp lại, kiểm thử liên tục và tích hợp phản hồi nhanh chóng. Một trong những yếu tố quan trọng là **kiểm thử liên tục (Continuous Testing)**, giúp phát hiện và sửa lỗi ngay trong từng vòng lặp phát triển, thay vì chờ đến giai đoạn cuối như mô hình Thác nước. Điều này giúp giảm chi phí sửa lỗi và nâng cao độ tin cậy của phần mềm. Bên cạnh đó, **tích hợp liên tục (CI) và triển khai liên tục (CD)** giúp đảm bảo mã nguồn luôn được cập nhật và kiểm thử tự động, giảm thiểu lỗi khi tích hợp các tính năng mới. Agile cũng tập trung vào **khách hàng**, cho phép điều chỉnh phần mềm theo phản hồi thực tế, giúp sản phẩm cuối cùng đáp ứng đúng nhu cầu. Cuối cùng, Agile khuyến khích **cải tiến liên tục** thông qua các buổi Sprint Retrospective, giúp nhóm phát triển học hỏi từ sai sót và tối ưu hóa quy trình làm việc, từ đó nâng cao chất lượng sản phẩm một cách bền vững.

9.Thảo luận về vai trò của pha bảo trì trong vòng đời phát triển phần mềm.

Pha bảo trì là một giai đoạn quan trọng trong **vòng đời phát triển phần mềm (SDLC - Software Development Life Cycle)**, giúp phần mềm hoạt động ổn định, an toàn và phù hợp với nhu cầu thay đổi của người dùng. Vai trò chính của pha bảo trì bao gồm:

1. Đảm bảo tính ổn định và liên tục của phần mềm

- Sau khi triển khai, phần mềm có thể gặp lỗi chưa được phát hiện trong quá trình kiểm thử. Pha bảo trì giúp sửa lỗi và đảm bảo hệ thống hoạt động ổn định.

2. Cải thiện và nâng cấp phần mềm

- Phần mềm cần được nâng cấp để đáp ứng các nhu cầu mới của người dùng, tối ưu hiệu suất hoặc bổ sung tính năng mới.

3. Thích nghi với thay đổi môi trường

- Phần mềm phải tương thích với các thay đổi về phần cứng, hệ điều hành hoặc các yêu cầu pháp lý, bảo mật mới.

4. Bảo đảm an toàn và bảo mật

- Liên tục cập nhật và vá lỗi bảo mật giúp ngăn chặn các lỗ hổng có thể bị hacker khai thác, đảm bảo dữ liệu và hệ thống không bị xâm nhập.

5. Giảm chi phí về lâu dài

- Bảo trì định kỳ giúp ngăn chặn các sự cố lớn, giảm chi phí sửa chữa và duy trì tuổi thọ phần mềm.

Tóm lại, pha bảo trì là một phần không thể thiếu trong vòng đời phần mềm, giúp hệ thống hoạt động bền vững và phù hợp với sự phát triển của doanh nghiệp.

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

Mô hình vòng đời phù hợp cho dự án phát triển phần mềm ngân hàng và lý do

Mô hình phù hợp: Mô hình V (V-Model) hoặc Thác nước cải tiến

Phát triển phần mềm ngân hàng yêu cầu **độ chính xác cao, bảo mật chặt chẽ và ít thay đổi sau khi triển khai**, do đó, **mô hình V hoặc Thác nước cải tiến** là lựa chọn phù hợp.

Lý do chọn mô hình V-Model hoặc Thác nước cải tiến:

1. **Yêu cầu rõ ràng, ít thay đổi:**
    - Các hệ thống ngân hàng phải tuân theo quy trình chặt chẽ, có yêu cầu nghiêm ngặt ngay từ đầu, nên không cần thay đổi nhiều trong quá trình phát triển.
2. **Kiểm thử song song với phát triển, đảm bảo chất lượng cao:**
    - Mô hình V yêu cầu mỗi giai đoạn phát triển đều có giai đoạn kiểm thử tương ứng, giúp phát hiện lỗi sớm và đảm bảo chất lượng cao.
    - Điều này rất quan trọng đối với phần mềm ngân hàng, nơi sai sót có thể gây tổn thất tài chính nghiêm trọng.

3. **Tính bảo mật và tuân thủ pháp lý:**
    - Ngành ngân hàng yêu cầu tiêu chuẩn bảo mật cao (ví dụ: PCI DSS, ISO 27001). Mô hình V có kiểm thử bảo mật nghiêm ngặt ngay từ đầu, giúp đáp ứng các yêu cầu này.
4. **Độ tin cậy cao, triển khai ổn định:**
    - Phần mềm ngân hàng cần hoạt động ổn định, tránh lỗi nghiêm trọng khi triển khai. Mô hình V giúp đảm bảo hệ thống được kiểm thử kỹ lưỡng trước khi đưa vào sử dụng.
5. **Dễ bảo trì và nâng cấp:**
    - Mô hình Thác nước cải tiến cho phép cập nhật và bảo trì dễ dàng, phù hợp với hệ thống ngân hàng cần hoạt động liên tục.

Tại sao không chọn mô hình Agile?

- Agile phù hợp với dự án có yêu cầu thay đổi liên tục, nhưng ngân hàng cần **sự ổn định và tính tuân thủ cao**, nên không thể thay đổi quá nhiều trong quá trình phát triển.

Kết luận:
Với phần mềm ngân hàng, **mô hình V hoặc Thác nước cải tiến** là lựa chọn tốt nhất vì đảm bảo chất lượng, bảo mật và tuân thủ quy định pháp lý, giúp hệ thống hoạt động ổn định và an toàn.

Câu tình huống

1. Một công ty phát triển phần mềm theo mô hình thác nước gặp vấn đề khi khách hàng yêu cầu thay đổi sau khi hoàn thành pha thiết kế. Đội phát triển nên xử lý như thế nào?

Khi khách hàng yêu cầu thay đổi sau khi pha thiết kế đã hoàn thành trong mô hình thác nước, đội phát triển nên xử lý như sau:

Đánh giá ảnh hưởng của thay đổi: Xác định phạm vi và mức độ thay đổi yêu cầu, xem xét tác động của thay đổi đối với tiến độ, ngân sách và các giai đoạn tiếp theo.

Thảo luận với khách hàng: Liên hệ với khách hàng để làm rõ yêu cầu thay đổi, làm rõ lý do và mức độ quan trọng của thay đổi đối với dự án.

Điều chỉnh thiết kế và kế hoạch: Nếu thay đổi là cần thiết và khả thi, đội phát triển cần cập nhật thiết kế, điều chỉnh kế hoạch và tiến độ để phù hợp với yêu cầu mới.

Cập nhật tài liệu và kiểm tra lại: Điều chỉnh các tài liệu yêu cầu, thiết kế và tiến hành kiểm tra lại các giai đoạn đã hoàn thành để đảm bảo tính tương thích.

Thông báo cho các bên liên quan: Đảm bảo rằng tất cả các bên liên quan, bao gồm nhóm phát triển, khách hàng và các bộ phận khác, đều nắm rõ thay đổi và ảnh hưởng đến tiến độ dự án.

2. Dự án phát triển phần mềm theo mô hình lặp và tăng trưởng liên tục bị trễ tiến độ do thiếu nhân lực. Là quản lý dự án, bạn sẽ làm gì?

- Đánh giá lại tiến độ và phân công lại công việc:

Kiểm tra các vòng lặp còn lại và đánh giá mức độ quan trọng của các tính năng. Ưu tiên các tính năng quan trọng và cắt giảm những phần không cần thiết để giảm tải công việc.

- Tăng cường nhân lực:

Tuyển dụng thêm nhân viên tạm thời hoặc thuê ngoài (outsourcing) để bổ sung nguồn lực. Nếu có thể, tái phân bổ nhân sự từ các dự án khác hoặc từ các bộ phận trong công ty.

- Tái phân bổ nguồn lực nội bộ:

Chuyển một số nhân sự từ các nhiệm vụ ít quan trọng sang hỗ trợ dự án chính. Đồng thời, có thể cải tiến quy trình làm việc để tiết kiệm thời gian.

- Cải thiện quy trình làm việc:

Tinh giản quy trình phát triển và kiểm thử để giảm thời gian chờ đợi, tăng hiệu quả công việc. Áp dụng phương pháp làm việc hiệu quả hơn như Agile hoặc DevOps nếu cần.

- Cập nhật tiến độ với khách hàng:

Thông báo cho khách hàng về tình hình thực tế của dự án, điều chỉnh kỳ vọng và có thể thương thảo lại về thời gian giao hàng.

- Theo dõi tiến độ sát sao:

Giám sát chặt chẽ tiến độ của từng vòng lặp, đảm bảo mọi người đều tập trung vào các nhiệm vụ quan trọng nhất và hoàn thành đúng hạn.

3. Trong quá trình phát triển phần mềm theo mô hình tiến trình linh hoạt, khách hàng không đưa ra phản hồi kịp thời. Đội phát triển nên xử lý ra sao?

Khi khách hàng không đưa ra phản hồi kịp thời trong quá trình phát triển phần mềm theo mô hình tiến trình linh hoạt (Agile), đội phát triển có thể xử lý như sau:

Liên lạc với khách hàng:

Chủ động liên hệ với khách hàng để nhắc nhở và yêu cầu phản hồi, làm rõ lý do sự chậm trễ và đưa ra giải pháp cụ thể để đẩy nhanh tiến độ.

Tăng cường giao tiếp:

Thiết lập các cuộc họp định kỳ (ví dụ: cuộc họp trực tuyến, họp nhanh) để cập nhật tiến độ và yêu cầu phản hồi. Đảm bảo khách hàng hiểu được tầm quan trọng của việc phản hồi kịp thời.

Linh hoạt điều chỉnh công việc:

Nếu không thể có phản hồi, đội phát triển có thể tập trung vào các nhiệm vụ khác không cần sự phản hồi từ khách hàng, như cải tiến mã nguồn, tăng cường chất lượng hoặc chuẩn bị cho các tính năng tiếp theo.

Đưa ra quyết định dựa trên giả định hợp lý:

Nếu cần thiết, đội phát triển có thể đưa ra quyết định tạm thời dựa trên những giả định hợp lý, nhưng cần đảm bảo rằng khi có phản hồi từ khách hàng, có thể điều chỉnh ngay lập tức.

Tài liệu và minh bạch:

Tạo ra tài liệu ghi nhận tất cả các quyết định, vấn đề và các thay đổi tạm thời để khách hàng có thể dễ dàng theo dõi và cung cấp phản hồi khi cần thiết.

4. Khách hàng yêu cầu bổ sung một số tính năng mới khi phần mềm đã bước vào pha cài đặt. Nên áp dụng mô hình nào để xử lý tốt nhất yêu cầu này?

Mô hình lặp và tăng trưởng:

- **Ưu điểm:**
    - Tăng trưởng từng bước: Cho phép tích hợp các tính năng mới một cách có hệ thống.
    - Phản hồi liên tục: Nhận phản hồi từ khách hàng sau mỗi lần phát hành.
- **Áp dụng:**
    - Xác định các tính năng mới và chia thành các phiên bản phát hành.
    - Phát hành các phiên bản tăng dần với các tính năng mới được tích hợp.

- Cần có sự kiểm soát về những thay đổi để tránh gây ra những lỗi nghiêm trọng.

5. Một công ty nhỏ muốn áp dụng mô hình bản mẫu nhanh nhưng gặp khó khăn do thiếu nguồn lực. Hãy đề xuất giải pháp.

Sử dụng công cụ prototyping miễn phí hoặc chi phí thấp

Chọn các công cụ như Figma, Balsamiq, hoặc Adobe XD để tạo bản mẫu mà không tốn nhiều chi phí.

Ưu tiên bản mẫu đơn giản, tập trung vào tính năng cốt lõi

Không cần xây dựng bản mẫu hoàn chỉnh ngay từ đầu, chỉ cần mô phỏng những chức năng quan trọng.

Thuê ngoài hoặc hợp tác với freelancer

Thuê freelancer trên các nền tảng như Upwork, Fiverr hoặc hợp tác với sinh viên CNTT để tiết kiệm chi phí.

Tận dụng phương pháp MVP (Minimum Viable Product)

Xây dựng một phiên bản tối giản nhưng khả dụng, giúp công ty nhanh chóng thu thập phản hồi khách hàng.

Làm việc theo từng giai đoạn nhỏ

Chia quá trình phát triển bản mẫu thành nhiều bước nhỏ, giúp kiểm soát nguồn lực tốt hơn.

6. Trong dự án phần mềm thương mại điện tử, khách hàng liên tục yêu cầu thay đổi giao diện. Mô hình nào sẽ phù hợp nhất?

Trong dự án thương mại điện tử, giao diện người dùng là yếu tố quan trọng và có thể thay đổi liên tục theo phản hồi của khách hàng.
Mô hình Agile, đặc biệt là Scrum hoặc Kanban, cho phép linh hoạt điều chỉnh yêu cầu, cập nhật giao diện sau từng vòng lặp (iteration/sprint), giúp phản hồi nhanh với thay đổi của khách hàng.

7. Dự án phát triển phần mềm lớn với nhiều nhóm phát triển ở các quốc gia khác nhau nên áp dụng mô hình nào?

- Với dự án lớn có nhiều nhóm làm việc ở các quốc gia khác nhau, cần mô hình hỗ trợ sự phối hợp và giao tiếp hiệu quả.
- **Agile với Scaled Agile Framework (SAFe)** hoặc **DevOps** có thể được áp dụng để đảm bảo tính linh hoạt, giao tiếp liên tục và tích hợp liên tục giữa các nhóm.
- Các công cụ như Jira, Confluence, Slack, GitHub giúp hỗ trợ cộng tác và quản lý tiến độ hiệu quả.

8. Phân tích rủi ro là hoạt động chính trong mô hình xoắn ốc. Đề xuất cách giảm thiểu rủi ro khi áp dụng mô hình này.

- **Xác định rủi ro sớm**: Sử dụng phân tích SWOT, FMEA để nhận diện rủi ro ngay từ đầu.
- **Ưu tiên xử lý rủi ro cao**: Xây dựng chiến lược giảm thiểu tác động của rủi ro quan trọng trước khi phát triển tiếp.
- **Lặp lại quá trình đánh giá**: Mỗi vòng lặp (iteration) phải có đánh giá rủi ro để điều chỉnh kế hoạch.
- **Tạo nguyên mẫu (prototype)**: Giúp khách hàng kiểm tra và điều chỉnh yêu cầu sớm, tránh sai sót lớn.
- **Quản lý thay đổi hiệu quả**: Áp dụng kỹ thuật như MoSCoW để phân loại mức độ ưu tiên của yêu cầu thay đổi.
- **Tăng cường giao tiếp**: Duy trì các cuộc họp định kỳ giữa các bên liên quan để cập nhật tình hình và điều chỉnh kịp thời.

9. Một dự án phát triển phần mềm ngân hàng cần yêu cầu bảo mật cao. Nên áp dụng mô hình nào để đảm bảo yêu cầu này?

Đối với phần mềm ngân hàng, **mô hình V (V-Model) hoặc mô hình Thác nước cải tiến** là lựa chọn phù hợp để đảm bảo yêu cầu bảo mật cao.

Lý do chọn mô hình V-Model hoặc Thác nước cải tiến:

1. **Bảo mật chặt chẽ ngay từ đầu:**
    - Các yêu cầu bảo mật được xác định rõ ràng ngay từ giai đoạn phân tích, giúp thiết kế hệ thống an toàn ngay từ đầu.
2. **Kiểm thử song song với phát triển:**
    - Mô hình V yêu cầu kiểm thử nghiêm ngặt ở mỗi giai đoạn, bao gồm kiểm thử bảo mật, giúp phát hiện sớm các lỗ hổng trước khi triển khai.
3. **Tuân thủ các tiêu chuẩn bảo mật:**
    - Ngành ngân hàng phải tuân thủ các tiêu chuẩn bảo mật như **PCI DSS, ISO 27001, OWASP Top 10**, nên cần một mô hình kiểm soát chặt chẽ từng giai đoạn phát triển.
4. **Hạn chế thay đổi giúp giảm rủi ro bảo mật:**

    - Không giống như Agile, nơi yêu cầu có thể thay đổi liên tục, mô hình V giữ yêu cầu ổn định, giúp kiểm soát rủi ro bảo mật tốt hơn.
5. **Triển khai an toàn và ổn định:**
    - Mô hình này đảm bảo phần mềm được kiểm thử đầy đủ trước khi triển khai, giúp giảm thiểu rủi ro bảo mật trong môi trường thực tế.

Tại sao không chọn Agile?

- Agile linh hoạt nhưng khó kiểm soát bảo mật chặt chẽ vì yêu cầu thay đổi liên tục. Trong ngân hàng, **tính ổn định, an toàn và tuân thủ quan trọng hơn tốc độ phát triển**, nên không phù hợp bằng mô hình V hoặc Thác nước cải tiến.

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

Vòng đời phần mềm kết thúc và thực hiện pha giải thể khi:

1. **Phần mềm không còn đáp ứng nhu cầu kinh doanh:**
    - Công nghệ hoặc mô hình kinh doanh thay đổi, phần mềm trở nên lỗi thời và không còn giá trị sử dụng.
2. **Chi phí bảo trì cao hơn giá trị mang lại:**
    - Khi chi phí bảo trì vượt quá lợi ích phần mềm mang lại, doanh nghiệp có thể quyết định thay thế bằng hệ thống mới.
3. **Không còn khả năng mở rộng hoặc nâng cấp:**
    - Phần mềm không còn tương thích với nền tảng hiện tại hoặc không thể tích hợp với hệ thống mới.
4. **Rủi ro bảo mật quá cao:**
    - Nếu phần mềm không thể đáp ứng các tiêu chuẩn bảo mật mới hoặc có nhiều lỗ hổng không thể khắc phục, cần thay thế bằng hệ thống an toàn hơn.
5. **Hết vòng đời hỗ trợ từ nhà cung cấp:**
    - Nếu phần mềm dựa trên công nghệ đã lỗi thời và không còn được nhà cung cấp hỗ trợ (ví dụ: hệ điều hành, cơ sở dữ liệu cũ), doanh nghiệp buộc phải dừng sử dụng.

Các bước thực hiện pha giải thể:

- **Thông báo cho người dùng và lên kế hoạch thay thế.**
- **Chuyển dữ liệu sang hệ thống mới nếu cần thiết.**
- **Gỡ bỏ phần mềm một cách an toàn để tránh rủi ro bảo mật.**
- **Cập nhật tài liệu và lưu trữ dữ liệu quan trọng để tuân thủ quy định.**

Pha giải thể giúp đảm bảo hệ thống cũ không gây ảnh hưởng tiêu cực đến doanh nghiệp, đồng thời tối ưu hóa tài nguyên và bảo mật.