CÂU HỎI TRẮC NGHIỆM

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

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

Câu hỏi 3: Quy trình phát triển phần mềm gồm mấy giai đoạn chính?

A. 3
B. 4
C. 5
D. 6

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

Câu hỏi 5: Chi phí bảo trì phần mềm chiếm bao nhiêu phần trăm tổng chi phí vòng đời phần mềm?

A. 46%
B. 56%
C. 76%
D. 86%

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

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

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

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

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

Câu 1: Phần mềm là gì?

Phần mềm là tập hợp các chương trình, chỉ dẫn và dữ liệu được viết bằng ngôn ngữ lập trình để điều khiển hoạt động của máy tính và các thiết bị điện tử, giúp thiết bị hiểu và thực hiện các chức năng mà người dùng yêu cầu.

Câu 2: Công nghệ phần mềm là gì?

Công nghệ phần mềm là một lĩnh vực chuyên nghiên cứu, xây dựng, kiểm thử và triển khai các ứng dụng, hệ thống phần mềm để nâng cao hiệu quả hoạt động và chất lượng cuộc sống.

Câu 3: Các loại phần mềm chính là gì?

Các loại phần mềm chính bao gồm phần mềm hệ thống (như hệ điều hành) quản lý tài nguyên máy tính và phần mềm ứng dụng (như trình duyệt web).

Câu 4: Tại sao công nghệ phần mềm lại quan trọng?

Công nghệ phần mềm quan trọng vì nó là nền tảng cho hoạt động của mọi thiết bị điện tử, thúc đẩy tự động hóa, nâng cao năng suất, và cải thiện chất lượng cuộc sống.

Câu 5: Quy trình phát triển phần mềm gồm những giai đoạn nào?

- Khảo sát và phân tích yêu cầu: thu thập, phân tích và xác định rõ ràng các yêu cầu của người dùng và mục tiêu của phần mềm.
- Thiết kế hệ thống và kiến trúc phần mềm: Dựa trên các yêu cầu đã phân tích, giai đoạn này sẽ thiết kế kiến trúc hệ thống, giao diện người dùng (UI/UX) và cơ sở dữ liệu của phần mềm.
- Lập trình (coding): xây dựng mã nguồn và các thành phần của phần mềm dựa trên thiết kế đã được duyệt.
- Kiểm thử (testing): xác định các lỗi, vấn đề và sự không phù hợp với yêu cầu ban đầu và báo cáo lỗi.
- Triển khai (deployment): cài đặt và đưa phần mềm vào hoạt động thực tế, cung cấp cho người dùng cuối.
- Bảo trì và nâng cấp (maintenance):

Câu 6: Khía cạnh kinh tế của công nghệ phần mềm là gì?

- Tối ưu chi phí phát triển và vận hành phần mềm.
- Tạo việc làm, nâng cao hiệu quả sử dụng nguồn lực (thời gian, nhân lực, công cụ).
- Giúp doanh nghiệp giảm chi phí quản lý, tăng năng suất và lợi nhuận, tạo lợi thế cạnh tranh
- Đổi mới và sáng tạo, là nền tảng cho sự đổi mới trong nhiều lĩnh vực, tạo ra các giải pháp mới và cải thiện chất lượng cuộc sống.

Câu 7: Khía cạnh công nghệ của công nghệ phần mềm là gì?

- Áp dụng quy trình, phương pháp phát triển, công cụ và kỹ thuật một cách có hệ thống để phát triển, sử dụng và bảo trì phần mềm.
- Ứng dụng các ngôn ngữ lập trình, công cụ phát triển, nền tảng, framework.
- Áp dụng các biện pháp kỹ thuật để bảo vệ dữ liệu và hệ thống khỏi các mối đe dọa an ninh.
- Đảm bảo tính tương thích, khả năng mở rộng, tính bảo mật và chất lượng của phần mềm.

Câu 8: Khía cạnh bảo trì của công nghệ phần mềm là gì?

- **Sửa lỗi (Corrective maintenance):** Sửa các lỗi phát sinh sau khi phần mềm đã được triển khai.
- **Nâng cấp/tối ưu hiệu suất (Perfective maintenance):** Cải tiến, bổ sung các tính năng mới để tối ưu hóa công dụng của phần mềm theo nhu cầu của người dùng.
- **Thích nghi khi môi trường thay đổi (Adaptive maintenance):** Cập nhật phần mềm để nó có thể hoạt động hiệu quả trong môi trường công nghệ mới, hệ điều hành mới hoặc các quy định thay đổi.
- **Phòng ngừa rủi ro, nâng cao độ ổn định (Preventive maintenance):** Kiểm tra, theo dõi định kỳ để phát hiện sớm các vấn đề tiềm ẩn, ngăn ngừa sự cố xảy ra.

Câu 9: Các nguyên nhân chính gây trễ thời hạn khi phát triển phần mềm là gì?

- Xác định yêu cầu không rõ ràng, thay đổi yêu cầu liên tục, kế hoạch không thực tế.
- Thiếu kinh nghiệm, năng lực của đội ngũ phát triển.
- Quản lý dự án kém, thiếu tài nguyên hoặc phân bổ sai.
- Khó khăn trong kiểm thử, phát hiện lỗi muộn, áp dụng các giải pháp tạm thời, không tối ưu để hoàn thành nhanh chóng, tạo ra "nợ kỹ thuật".
- Quản lý dự án yếu kém, làm việc thiếu hệ thống, không bám sát chặt chẽ tiến độ..
- Tích hợp hệ thống phức tạp, sử dụng công cụ phát triển không hiệu quả, lỗi thời.

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

- Phân tích và đánh giá phần mềm sau khi triển khai.
- Xử lý sự cố và khắc phục lỗi.
- Cập nhật, nâng cấp tính năng mới.
- Cải thiện hiệu năng, bảo mật.
- Điều chỉnh theo yêu cầu môi trường (OS, phần cứng, nền tảng).

CÂU HỎI THẢO LUẬN NHÓM

Câu 1: Phân biệt phần mềm hệ thống và phần mềm ứng dụng.

Phần mềm hệ thống quản lý, điều khiển phần cứng và chạy nền tảng, cung cấp môi trường cho các ứng dụng khác hoạt động.

- Mục đích: Cung cấp một môi trường hoạt động và quản lý tài nguyên cho máy tính và các phần mềm khác.

- Chức năng: Điều khiển phần cứng, quản lý các ứng dụng, và cung cấp các dịch vụ cơ bản để hệ thống hoạt động trơn tru.
- Tương tác: Ít tương tác trực tiếp với người dùng cuối, hoạt động ở chế độ nền.
- Thời điểm hoạt động: Khởi chạy ngay khi bật máy và hoạt động liên tục trong quá trình sử dụng.
- Tính bắt buộc: Là nền tảng cốt lõi, không thể thiếu để máy tính hoạt động.
- Ví dụ: hệ điều hành như Windows, macOS, Linux

Phần mềm ứng dụng là công cụ phục vụ nhu cầu cụ thể của người dùng.

- Mục đích: Giúp người dùng thực hiện các tác vụ hoặc chức năng cụ thể như soạn thảo văn bản, duyệt web, hoặc chơi game và phụ thuộc vào phần mềm hệ thống để hoạt động
- Chức năng: Chuyên biệt hóa cho các nhu cầu cụ thể, như tạo tài liệu, xử lý dữ liệu, giải trí.
- Tương tác: Tương tác trực tiếp với người dùng để thực hiện các thao tác theo yêu cầu.
- Thời điểm hoạt động: Chỉ hoạt động khi người dùng khởi động và yêu cầu.
- Tính tùy chọn: Chỉ được cài đặt và sử dụng khi người dùng có nhu cầu.
- Ví dụ: Word, Excel, Zalo, Facebook

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

- **Quản lý và tự động hóa giao dịch**: Phần mềm giúp xử lý giao dịch nhanh chóng, chính xác, an toàn, đồng thời tự động hóa các tác vụ thủ công, giảm sai sót và nâng cao hiệu suất.
- **Phân tích dữ liệu lớn và dự báo thị trường**: AI và Big Data hỗ trợ phân tích khối lượng dữ liệu tài chính khổng lồ, phát hiện xu hướng, dự báo rủi ro, thị trường và cá nhân hóa dịch vụ đầu tư.
- **Tăng cường bảo mật và quản lý rủi ro**: Ứng dụng công nghệ Blockchain và các giải pháp bảo mật tiên tiến giúp minh bạch, an toàn dữ liệu và hạn chế gian lận trong hoạt động tài chính.
- **Thông tin theo thời gian thực**: Các hệ thống đám mây cho phép truy cập, giám sát và phân tích thông tin tài chính tức thời, hỗ trợ ra quyết định chính xác, kịp thời.
- **Ứng dụng trong dịch vụ tài chính mới**: Nền tảng cho ngân hàng số, ví điện tử, thanh toán trực tuyến, fintech, blockchain… mở ra nhiều sản phẩm và dịch vụ tài chính hiện đại.
- **Nâng cao trải nghiệm khách hàng**: Cung cấp dịch vụ cá nhân hóa, hỗ trợ 24/7 và mang lại sự tiện lợi, linh hoạt trong việc sử dụng dịch vụ tài chính.

Câu 3: Các thách thức thường gặp trong bảo trì phần mềm

1. Kỹ thuật

- **Hiểu hệ thống cũ**: Nhiều phần mềm được viết lâu đời, thiếu tài liệurời đ khiến việc đọc và hiểu mã nguồn khó khăn.

  Ví dụ: Một phần mềm quản lý nhân sự viết bằng **Visual Basic 6** từ năm 2005, tài liệu gốc thất lạc→ người bảo trì phải đọc lại toàn bộ mã spaghetti, rất tốn thời gian.

- **Độ phức tạp gia tăng**: Phần mềm càng lâu đời thì càng dễ tích tụ "nợ kỹ thuật", mã khó mở rộng, khó chỉnh sửa mà không gây lỗi.

- **Tương thích hệ thống**: Cần bảo đảm phần mềm hoạt động tốt với phần cứng, hệ điều hành, thư viện hoặc môi trường công nghệ mới.

  Ví dụ: Phần mềm kế toán chạy tốt trên **Windows 7**, nhưng khi công ty nâng cấp lên **Windows 11**, một số hàm API cũ không còn hỗ trợ → phải chỉnh sửa lại code hoặc viết lớp tương thích.

- **Quản lý lỗi và kiểm thử**: Mỗi lần chỉnh sửa có nguy cơ phát sinh lỗi mới (regression). Kiểm thử hồi quy (regression testing) tốn kém và phức tạp.

2. Tài liệu và kiến thức

- **Thiếu tài liệu**: Nhiều hệ thống không có tài liệu đầy đủ, hoặc tài liệu không được cập nhật theo thay đổi mã nguồn.

  Ví dụ: Hệ thống thương mại điện tử được phát triển nhanh để “chạy thử thị trường”, không kịp viết tài liệu API. Sau này khi muốn tích hợp với app di động, đội dev mới không biết API cũ hoạt động thế nào.

- **Truyền đạt kiến thức**: Người mới khó tiếp nhận kiến thức từ đội ngũ cũ nếu không có quá trình chuyển giao tốt.

  Ví dụ: Ở một công ty, chỉ có 1 lập trình viên duy nhất hiểu hệ thống **ERP** nội bộ. Khi người này nghỉ, đội ngũ kế thừa mất nhiều tháng để tìm hiểu lại từ đầu.

3. Quản lý và tổ chức

- **Chi phí cao**: Bảo trì thường chiếm 60–80% tổng chi phí vòng đời phần mềm.

  Ví dụ: Hệ thống ngân hàng cũ cần bảo trì liên tục. Ngân hàng phải chi hàng triệu USD mỗi năm chỉ để giữ cho phần mềm core banking chạy ổn định, trong khi chi phí phát triển ban đầu thấp hơn

- **Ưu tiên công việc**: Cần cân bằng giữa sửa lỗi, thêm tính năng mới và tối ưu hệ thống.
  Ví dụ: Người dùng yêu cầu thêm tính năng "báo cáo doanh thu theo khu vực" nhưng cùng lúc hệ thống có nhiều lỗi bảo mật cần vá gấp. Đội bảo trì phải cân nhắc ưu tiên vá lỗi trước hay phát triển tính năng mới
- **Phối hợp đội ngũ**: Khi nhiều lập trình viên cùng tham gia bảo trì, dễ phát sinh xung đột mã hoặc thay đổi không đồng bộ.

4. Người dùng và yêu cầu

- **Yêu cầu thay đổi liên tục**: Người dùng thường muốn cập nhật thêm tính năng mới, gây áp lực cho đội bảo trì.

Ví dụ: Ứng dụng giao đồ ăn ban đầu chỉ hỗ trợ thanh toán tiền mặt. Sau này khách hàng muốn tích hợp **Momo, ZaloPay, thẻ tín dụng** → đội bảo trì phải viết lại phần thanh toán phức tạp hơn.

- **Khó khăn khi triển khai**: Một số thay đổi nhỏ có thể tác động lớn đến quy trình làm việc của người dùng.

  Ví dụ: Một công ty cập nhật phần mềm quản lý kho để tối ưu quy trình, nhưng nhân viên đã quen cách cũ. Sau khi nâng cấp, họ phản đối vì "khó sử dụng hơn", khiến việc triển khai bị trì hoãn.

5. Bảo mật

- **Lỗ hổng bảo mật**: Hệ thống cũ dễ gặp rủi ro bảo mật, cần vá lỗi thường xuyên.
  Ví dụ: Một website bán hàng dùng phiên bản **PHP 5.6** (hết hỗ trợ từ 2018). Hacker khai thác lỗ hổng upload file để chiếm quyền server. Đội bảo trì buộc phải nâng cấp toàn bộ lên PHP 8.x.
- **Chuẩn mới**: Phần mềm cần tuân thủ các tiêu chuẩn an toàn, bảo mật mới theo thời gian.

  Ví dụ: Hệ thống thanh toán cũ không đạt chuẩn **PCI DSS** (chuẩn bảo mật thẻ tín dụng) mới → nếu không nâng cấp, công ty bị đối tác ngân hàng từ chối hợp tác.

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

1. Đảm bảo hiệu suất và tốc độ

- Người dùng ngày càng kỳ vọng vào tốc độ tải trang nhanh và trải nghiệm mượt mà.
- Bảo trì giúp tối ưu hóa mã nguồn, cơ sở dữ liệu và hạ tầng để tránh tình trạng chậm trễ hoặc gián đoạn.

  Ví dụ: Một lỗ hổng bảo mật trong plugin thanh toán của "ThanhToanPro" không được cập nhật kịp thời đã bị hacker khai thác, dẫn đến rò rỉ thông tin thẻ tín dụng của hơn 1.000 khách hàng. Việc bảo trì định kỳ có thể đã ngăn chặn sự cố này.

2. Cập nhật bảo mật

- Các mối đe dọa an ninh mạng luôn thay đổi, từ malware đến tấn công DDoS.
- Việc bảo trì định kỳ giúp vá lỗi bảo mật, cập nhật phần mềm và bảo vệ dữ liệu khách hàng.

  Ví dụ: Một lỗ hổng bảo mật trong plugin thanh toán của "ThanhToanPro" không được cập nhật kịp thời đã bị hacker khai thác, dẫn đến rò rỉ thông tin thẻ tín dụng của hơn 1.000 khách hàng. Việc bảo trì định kỳ có thể đã ngăn chặn sự cố này.

3. Tương thích với công nghệ mới

- Trình duyệt, hệ điều hành, thiết bị di động... đều liên tục được cập nhật.

- Phần mềm cần được điều chỉnh để hoạt động tốt trên các nền tảng mới, tránh lỗi hiển thị hoặc mất chức năng.

  Ví dụ: "GiayXinh.vn" nhận thấy nhiều khách hàng bỏ giỏ hàng giữa chừng. Sau khi bảo trì và cải tiến giao diện giỏ hàng (thêm hình ảnh sản phẩm, nút thanh toán dễ thấy), tỷ lệ hoàn tất đơn hàng tăng 25%.

4. Cải thiện trải nghiệm người dùng (UX)

- Phân tích hành vi người dùng có thể chỉ ra những điểm chưa tối ưu.
- Bảo trì giúp cập nhật giao diện, thêm tính năng mới hoặc loại bỏ những phần không cần thiết.

  Ví dụ: "GiayXinh.vn" nhận thấy nhiều khách hàng bỏ giỏ hàng giữa chừng. Sau khi bảo trì và cải tiến giao diện giỏ hàng (thêm hình ảnh sản phẩm, nút thanh toán dễ thấy), tỷ lệ hoàn tất đơn hàng tăng 25%.

5. Hỗ trợ kinh doanh và tăng doanh thu

- Hệ thống ổn định giúp khách hàng tin tưởng hơn khi mua sắm.
- Tránh mất đơn hàng do lỗi kỹ thuật hoặc gián đoạn dịch vụ.

  Ví dụ: "DienMayOnline" từng bị lỗi không hiển thị sản phẩm khuyến mãi đúng cách. Sau khi bảo trì và sửa lỗi hiển thị, doanh thu trong tuần khuyến mãi tăng gấp đôi so với tuần trước.

6. Dễ dàng mở rộng quy mô

- Khi doanh nghiệp phát triển, phần mềm cần được nâng cấp để xử lý nhiều đơn hàng hơn, tích hợp thêm dịch vụ, hoặc mở rộng sang thị trường mới.

  Ví dụ: "ThucPhamSach.vn" muốn mở rộng sang thị trường nước ngoài. Nhờ đã bảo trì và xây dựng hệ thống đa ngôn ngữ từ trước, họ dễ dàng triển khai phiên bản tiếng Anh và tích hợp thanh toán quốc tế mà không cần viết lại toàn bộ hệ thống.

Câu 5: **Những vấn đề phát sinh khi yêu cầu thay đổi liên tục**

1. Phá vỡ kế hoạch và tiến độ

- **Phân tích:** Mỗi thay đổi đều ảnh hưởng đến lịch trình đã định. Nhóm phát triển phải điều chỉnh lại các giai đoạn thiết kế, lập trình, kiểm thử, gây trễ deadline.

  Ví dụ: Dự án xây dựng hệ thống quản lý bán hàng cho "Công ty ABC" dự kiến hoàn thành trong 3 tháng. Sau 6 tuần, khách hàng yêu cầu thêm chức năng quản lý kho theo thời gian thực. Việc này khiến nhóm phải viết lại phần lớn kiến trúc hệ thống, làm trễ tiến độ thêm 1 tháng.

2. Tăng chi phí phát triển

- **Phân tích:** Thay đổi liên tục đồng nghĩa với việc cần thêm nhân lực, thời gian, và tài nguyên. Nếu không có hợp đồng rõ ràng, chi phí có thể vượt ngân sách.

  Ví dụ: "Startup XYZ" thuê một nhóm freelancer phát triển ứng dụng đặt đồ ăn. Ban đầu chỉ yêu cầu chức năng đặt món. Sau đó, họ thêm yêu cầu tích hợp bản đồ, chat với tài xế, và hệ thống điểm thưởng. Chi phí tăng gấp đôi so với dự toán ban đầu.

3. Giảm chất lượng sản phẩm

- **Phân tích:** Khi nhóm phát triển phải liên tục sửa đổi, họ có ít thời gian để kiểm thử, tài liệu hóa, và tối ưu mã nguồn. Điều này dễ dẫn đến lỗi và trải nghiệm người dùng kém.

  Ví dụ: Ứng dụng "HọcOnline" liên tục thay đổi giao diện theo phản hồi người dùng. Do không kiểm thử đầy đủ, phiên bản mới bị lỗi hiển thị trên thiết bị Android, khiến hàng trăm người dùng đánh giá 1 sao.

4. Gây mâu thuẫn nội bộ

- **Phân tích:** Nhóm phát triển có thể mất tinh thần khi phải làm lại nhiều lần, đặc biệt nếu không có sự thống nhất giữa khách hàng và quản lý dự án.

  Ví dụ: Trong một dự án ERP cho doanh nghiệp sản xuất, khách hàng thay đổi yêu cầu báo cáo tài chính 4 lần trong 2 tuần. Điều này khiến lập trình viên và trưởng nhóm mâu thuẫn về cách xử lý, làm giảm hiệu suất làm việc.

5. Thiếu định hướng sản phẩm

- **Phân tích:** Nếu khách hàng không có tầm nhìn rõ ràng, sản phẩm dễ bị "loãng" vì cố gắng đáp ứng quá nhiều yêu cầu nhỏ lẻ, mất đi giá trị cốt lõi.

  Ví dụ: Ứng dụng "DuLịch360" ban đầu tập trung vào đặt tour. Sau nhiều thay đổi, nó tích hợp thêm blog, mạng xã hội, bản đồ, và ví điện tử. Kết quả: người dùng không biết ứng dụng dùng để làm gì, tỷ lệ giữ chân thấp.

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

Tiêu chí	Chi phí phát triển	Chi phí bảo trì
Thời điểm phát sinh	Giai đoạn đầu của dự án	Sau khi phần mềm được triển khai và sử dụng lâu dài
Mục tiêu chính	Xây dựng chức năng, giao diện, kiến trúc phần mềm ban đầu	Duy trì hoạt động ổn định, cập nhật, sửa lỗi, cải tiến UX

Tính chất công việc	Thiết kế, lập trình, kiểm thử, triển khai	Vá lỗi, cập nhật bảo mật, tối ưu hiệu suất, hỗ trợ người dùng
Chi phí thường thấy	Cao trong giai đoạn đầu, tập trung một lần	Phát sinh định kỳ, có thể kéo dài nhiều năm
Tác động nếu bỏ qua	Không có sản phẩm để sử dụng	Sản phẩm xuống cấp, dễ bị lỗi, mất khách hàng

Ví dụ cụ thể: Ứng dụng bán hàng "MobiShop"

Giai đoạn phát triển ban đầu

- Chi phí: 500 triệu VND
- Bao gồm: thiết kế giao diện, xây dựng chức năng giỏ hàng, thanh toán, quản lý sản phẩm, tích hợp API vận chuyển.
- Thời gian: 4 tháng

Giai đoạn bảo trì sau triển khai

- Chi phí năm 1: 120 triệu VND
    - Vá lỗi thanh toán khi có bản cập nhật iOS mới.
    - Tối ưu tốc độ tải trang khi lượng người dùng tăng.
    - Cập nhật bảo mật để chống tấn công SQL injection.
- Chi phí năm 2: 150 triệu VND
    - Thêm chức năng chat với khách hàng.
    - Tích hợp ví điện tử mới theo yêu cầu thị trường.
    - Cập nhật giao diện theo xu hướng thiết kế mới.

Tổng chi phí bảo trì sau 2 năm: 270 triệu VND, chiếm hơn 50% chi phí phát triển ban đầu.

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

1. Yêu cầu chức năng (Functional Requirements)

Định nghĩa: Là những yêu cầu mô tả hệ thống cần làm gì—tức là các chức năng, hành vi, hoặc quy trình mà phần mềm phải thực hiện.

Ví dụ: Người dùng có thể đăng ký tài khoản bằng email và mật khẩu.

- Hệ thống cho phép thêm sản phẩm vào giỏ hàng và thanh toán qua ví điện tử.
- Quản trị viên có thể xuất báo cáo doanh thu theo tháng.

2. Yêu cầu phi chức năng (Non-functional Requirements)

Định nghĩa: Là những yêu cầu mô tả cách hệ thống hoạt động, không liên quan trực tiếp đến chức năng cụ thể, mà liên quan đến chất lượng, hiệu suất, bảo mật, khả năng mở rộng...

Ví dụ:

- Thời gian phản hồi của hệ thống không quá 2 giây khi người dùng tìm kiếm sản phẩm.
- Hệ thống phải hoạt động liên tục 99.9% thời gian trong năm (tính sẵn sàng cao).
- Dữ liệu người dùng phải được mã hóa theo chuẩn AES-256 để đảm bảo bảo mật.
- Giao diện phải tương thích với mọi thiết bị di động và trình duyệt phổ biến.

3. So sánh

Tiêu chí	Yêu cầu chức năng	Yêu cầu phi chức năng
Mục tiêu	Mô tả hành vi, chức năng cụ thể của hệ thống	Mô tả cách hệ thống vận hành và chất lượng dịch vụ
Ví dụ	Đăng nhập, đặt hàng, gửi email	Tốc độ xử lý, bảo mật, khả năng mở rộng
Kiểm thử	Dễ kiểm thử bằng hành động cụ thể	Khó kiểm thử hơn, thường cần đo lường hoặc mô phỏng
Tác động đến người dùng	Trực tiếp	Gián tiếp nhưng ảnh hưởng lớn đến trải nghiệm

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

1. Mô hình thác nước (Waterfall)

Đặc điểm:

- Quy trình tuyến tính, từng bước: Phân tích → Thiết kế → Lập trình → Kiểm thử → Triển khai → Bảo trì.
- Mỗi giai đoạn phải hoàn thành trước khi chuyển sang giai đoạn tiếp theo.

Ưu điểm:

- Dễ quản lý, rõ ràng về tiến độ và tài liệu.
- Phù hợp với dự án có yêu cầu ổn định, ít thay đổi.

Nhược điểm:

- Khó thích ứng nếu yêu cầu thay đổi giữa chừng.
- Phát hiện lỗi muộn, thường là sau khi đã lập trình xong.

Ví dụ: Phù hợp với các dự án phần mềm nội bộ như hệ thống quản lý nhân sự cho doanh nghiệp vừa và nhỏ.

2. Mô hình Agile

Đặc điểm:

- Phát triển theo chu kỳ ngắn (sprint), có phản hồi liên tục từ khách hàng.
- Tập trung vào sự linh hoạt, cộng tác và cải tiến liên tục.

Ưu điểm:

- Dễ thích ứng với thay đổi yêu cầu.
- Tăng tốc độ ra mắt sản phẩm và cải thiện trải nghiệm người dùng.

Nhược điểm:

- Khó quản lý nếu không có đội ngũ giàu kinh nghiệm.
- Cần sự tham gia thường xuyên từ khách hàng.

Ví dụ: Phù hợp với startup phát triển ứng dụng di động như app đặt xe, app học online, nơi yêu cầu thay đổi liên tục theo thị trường.

3. Mô hình V (V-Model)

Đặc điểm:

- Là biến thể của Waterfall, nhưng nhấn mạnh vào kiểm thử song song với phát triển.
- Mỗi giai đoạn phát triển đều có giai đoạn kiểm thử tương ứng.

Ưu điểm:

- Kiểm soát chất lượng tốt hơn.
- Phát hiện lỗi sớm hơn so với Waterfall.

Nhược điểm:

- Ít linh hoạt với thay đổi.
- Tốn thời gian và chi phí kiểm thử.

Ví dụ: Phù hợp với phần mềm y tế, hàng không, hoặc các hệ thống yêu cầu độ tin cậy cao.

4. Mô hình Spiral (Xoắn ốc)

Đặc điểm:

- Kết hợp giữa mô hình lặp và Waterfall, tập trung vào phân tích rủi ro.
- Mỗi vòng xoắn là một giai đoạn phát triển có kiểm thử và đánh giá.

Ưu điểm:

- Quản lý rủi ro tốt.
- Phù hợp với dự án lớn, phức tạp.

Nhược điểm:

- Tốn chi phí và thời gian.
- Khó triển khai với nhóm nhỏ.

Ví dụ: Phù hợp với hệ thống ngân hàng, bảo hiểm, hoặc phần mềm chính phủ.

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

1. Thiết kế và lập tài liệu yêu cầu rõ ràng ngay từ đầu

- **Giải pháp:** Xây dựng tài liệu SRS (Software Requirement Specification) chi tiết, có xác nhận từ khách hàng.
- **Lợi ích:** Giảm hiểu sai yêu cầu, tránh phát triển sai chức năng.

  Ví dụ: Trong dự án phần mềm quản lý kho, việc mô tả rõ "tự động cảnh báo khi tồn kho dưới 10 đơn vị" giúp lập trình viên không nhầm lẫn với "cảnh báo khi hết hàng".

2. Áp dụng kiểm thử toàn diện trước khi bàn giao

- **Giải pháp:** Thực hiện kiểm thử đơn vị (unit test), kiểm thử tích hợp, kiểm thử hệ thống và kiểm thử chấp nhận người dùng (UAT).
- **Lợi ích:** Phát hiện lỗi sớm, đảm bảo phần mềm hoạt động đúng trong môi trường thực tế.

  Ví dụ: Ứng dụng đặt vé máy bay được kiểm thử với nhiều loại trình duyệt, thiết bị và phương thức thanh toán để đảm bảo không lỗi khi triển khai

3. Tự động hóa kiểm thử (Automated Testing)

- **Giải pháp:** Sử dụng các công cụ như Selenium, JUnit, Postman để kiểm thử tự động các chức năng quan trọng.
- **Lợi ích:** Giảm lỗi do thao tác thủ công, kiểm thử nhanh và lặp lại dễ dàng.

  Ví dụ: Website thương mại điện tử tự động kiểm tra giỏ hàng, thanh toán và đăng nhập mỗi khi có bản cập nhật mới.

4. Triển khai môi trường staging trước khi đưa vào production

- **Giải pháp:** Tạo môi trường giả lập giống môi trường thật để kiểm thử cuối cùng.
- **Lợi ích:** Giảm rủi ro khi triển khai, phát hiện lỗi tương thích hoặc hiệu suất.

Ví dụ**:** Phần mềm quản lý bệnh viện được chạy thử trên môi trường staging với dữ liệu giả lập trước khi áp dụng vào hệ thống thật.

5. Đào tạo và hướng dẫn sử dụng cho người dùng cuối

- **Giải pháp:** Cung cấp tài liệu hướng dẫn, video demo, hoặc buổi training.
- **Lợi ích:** Giảm lỗi do người dùng thao tác sai, tăng hiệu quả sử dụng.

Ví dụ: Hệ thống CRM bàn giao cho đội ngũ bán hàng kèm video hướng dẫn cách tạo khách hàng tiềm năng, giúp giảm lỗi nhập liệu.

Câu 10: Vai trò của đội kiểm thử trong quy trình phát triển phần mềm.

1. Phát hiện lỗi sớm

- **Vai trò:** Kiểm thử giúp phát hiện lỗi ngay từ giai đoạn đầu (thiết kế, lập trình).

Ví dụ: Trong ứng dụng ngân hàng, QA phát hiện lỗi logic khi tính lãi suất trước khi hệ thống được triển khai, tránh rủi ro tài chính.

2. Đảm bảo phần mềm đúng yêu cầu

- **Vai trò:** So sánh phần mềm thực tế với tài liệu yêu cầu (SRS).

Ví dụ: QA kiểm tra chức năng “đặt hàng” để đảm bảo đúng quy trình: chọn sản phẩm → nhập địa chỉ → thanh toán.

3. Thực hiện kiểm thử toàn diện

- **Vai trò:** Kiểm thử đơn vị, tích hợp, hệ thống, hiệu năng, bảo mật, khả năng tương thích...

Ví dụ: Kiểm thử ứng dụng thương mại điện tử trên nhiều trình duyệt và thiết bị để đảm bảo không lỗi hiển thị.

4. Đánh giá trải nghiệm người dùng (UX)

- **Vai trò:** Phát hiện điểm chưa hợp lý trong giao diện, luồng thao tác.

Ví dụ: QA đề xuất thay đổi vị trí nút “Thanh toán” vì người dùng thường bỏ sót.

5. Kiểm thử bảo mật

- **Vai trò:** Phát hiện lỗ hổng bảo mật, kiểm tra quyền truy cập, mã hóa dữ liệu.

Ví dụ: QA phát hiện người dùng thường có thể truy cập trang quản trị nếu không kiểm tra session đúng cách.

6. Đảm bảo hiệu suất hệ thống

- **Vai trò:** Kiểm tra tốc độ phản hồi, khả năng chịu tải.

  Ví dụ: QA kiểm thử hệ thống đặt vé máy bay với 1.000 người dùng đồng thời để đảm bảo không sập server.

7. Kiểm thử hồi quy (Regression Testing)

- **Vai trò:** Đảm bảo các chức năng cũ vẫn hoạt động sau khi cập nhật mới.

  Ví dụ: Sau khi thêm tính năng “chat với người bán”, QA kiểm tra lại toàn bộ giỏ hàng, thanh toán để đảm bảo không bị ảnh hưởng.

8. Đảm bảo tiến độ và chất lượng dự án

- **Vai trò:** Làm cầu nối giữa nhóm phát triển và khách hàng để đảm bảo sản phẩm đạt chuẩn trước khi bàn giao.

  Ví dụ: QA xác nhận phần mềm đạt 100% yêu cầu chức năng và 95% yêu cầu phi chức năng trước khi release.