**HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG CƠ SỞ**

**THÀNH PHỐ HỒ CHÍ MINH**

**KHOA CÔNG NGHỆ THÔNG TIN 2**





**BÁO CÁO**

**MÔN HỌC: NHẬP MÔN CÔNG NGHỆ PHẦN MỀM**

**BÀI TẬP CHƯƠNG 1: PHẠM VI CÔNG NGHỆ PHẦN MỀM**

|  |  |
| --- | --- |
| **Giáo viên hướng dẫn:** | **Gv. Châu Văn Vân** |
| **Nhóm sinh viên thực hiện:** | **15** |
| **Lớp:** | **D22CQAT01** |
| **Thành viên nhóm:** | **Phạm Đình Nam - N21DCCN054**  **Vũ Huy Hùng – N20DCCN020** |
|  | **Hồ Xuân Thiện – N22DCAT051**  **Lê Anh Dũng – N22DCAT011**  **Nguyễn Duyên Duy – N22DCAT013** |

***TP.HCM, 2/2025***

# LỜI CẢM ƠN

Em chân thành cảm ơn Thầy Châu Văn Vân, người đã tận tình hướng dẫn và giúp đỡ em trong quá trình học môn Nhập Môn Công Nghệ Phần Mềm. Kiến thức và kỹ năng thu được từ môn học này cũng sẽ hỗ trợ em trong việc làm việc nhóm và quản lý dự án, xây dựng phần mềm trong tương lai.

Em cũng muốn bày tỏ lòng biết ơn đến Ban lãnh đạo và các giảng viên tại Học Viện Công Nghệ Bưu Chính Viễn Thông, đã tạo điều kiện và cung cấp cơ sở vật chất để chúng em có môi trường học tập và rèn luyện.

Dù em nhận thức rằng đề tài của mình có những hạn chế và thiếu sót do kiến thức và kinh nghiệm còn hạn chế, và thời gian nghiên cứu có hạn, nhưng em hy vọng nhận được sự đóng góp ý kiến của thầy để có thêm kiến thức và kinh nghiệm, từ đó hoàn thiện đề tài một cách tốt nhất.

Em xin trân trọng cảm ơn!

*Thành phố Hồ Chí Minh, tháng 2 năm 2025*

# BẢNG PHÂN CÔNG CÔNG VIỆC

SINH VIÊN : Nguyễn Duyên Duy

|  |  |  |  |
| --- | --- | --- | --- |
| **STT** | **Họ tên** | **Phụ trách công việc** | **Thời gian** |
| 1 | Phạm Đình Nam  (Nhóm trưởng) | Phân công nhiệm vụ.  Thực hiện:   * Câu 1.1, 1.2 * Câu 2.1, 2.2 * Câu 3 * Câu 4.1, 4.2 | Phân công nhiệm vụ:  Ngày 22/2/2025.  Thực hiện:  Từ ngày 23/2.  Đến ngày 26/2.  Tổng hợp câu 3:  Ngày 27/2/2025  \*Note: Câu 3 là câu thảo luận nhóm nên cần tổng hợp lại. |
| 2 | Vũ Huy Hùng | Thực hiện:   * Câu 1.3, 1.4 * Câu 2.3, 2.4 * Câu 3 * Câu 4.3, 4.4 |
| 3 | Hồ Xuân Thiện | Thực hiện:   * Câu 1.5, 1.6 * Câu 2.5, 2.6 * Câu 3 * Câu 4.5, 4.6 |
| 4 | Lê Anh Dũng | Thực hiện:   * Câu 1.7, 1.8 * Câu 2.7, 2.8 * Câu 3 * Câu 4.7, 4.8 |
| 5 | Nguyễn Duyên Duy | Thực hiện:   * Câu 1.9, 1.10 * Câu 2.9, 2.10 * Câu 3 * Câu 4.9, 4.10 |

**MỤC LỤC**

[LỜI CẢM ƠN 2](#_heading=h.gjdgxs)

[BẢNG PHÂN CÔNG CÔNG VIỆC 3](#_heading=h.30j0zll)

[PHẦN 1: CÂU HỎI TRẮC NGHIỆM 6](#_heading=h.1fob9te)

[Câu 1.1: Phần mềm bao gồm các loại nào dưới đây? 7](#_heading=h.3znysh7)

[Câu 1.2: Công nghệ phần mềm là gì? 7](#_heading=h.2et92p0)

[Câu 1.3: Quy trình phát triển phần mềm gồm mấy giai đoạn chính? 7](#_heading=h.tyjcwt)

[Câu 1.4: Hoạt động nào dưới đây thuộc quy trình bảo trì phần mềm? 7](#_heading=h.3dy6vkm)

[Câu 1.5: Chi phí bảo trì phần mềm chiếm bao nhiêu phần trăm tổng chi phí vòng đời phần mềm? 7](#_heading=h.1t3h5sf)

[Câu 1.6:Nguyên nhân chính gây ra việc vượt chi phí khi phát triển phần mềm là gì? 7](#_heading=h.4d34og8)

[Câu 1.7:Yêu cầu nào dưới đây không phải là yêu cầu phi chức năng? 7](#_heading=h.ifbdls9kfnx5)

[Câu 1.8: Khi nào phần mềm được coi là hoàn thành? 8](#_heading=h.urq8c3q735vi)

[Câu 1.9:Vấn đề phổ biến nào thường gặp khi phát triển phần mềm? 8](#_heading=h.cbjbuj3385f1)

[Câu 1.10:Phần mềm có thể được chia thành bao nhiêu loại chính? 8](#_heading=h.26in1rg)

[PHẦN 2: CÂU TRẢ LỜI NGẮN 8](#_heading=h.lnxbz9)

[Câu 2.1: Phần mềm là gì? 8](#_heading=h.35nkun2)

[Câu 2.2: Công nghệ phần mềm là gì? 9](#_heading=h.1ksv4uv)

[Câu 2.3: Các loại phần mềm chính là gì? 9](#_heading=h.44sinio)

[Câu 2.4: Tại sao công nghệ phần mềm lại quan trọng? 9](#_heading=h.2jxsxqh)

[Câu 2.5: Quy trình phát triển phần mềm gồm những giai đoạn nào? 9](#_heading=h.z337ya)

[Câu 2.6:Khía cạnh kinh tế của công nghệ phần mềm là gì? 11](#_heading=h.3j2qqm3)

[Câu 2.7. Khía cạnh công nghệ của công nghệ phần mềm là gì? 12](#_heading=h.1y810tw)

[Câu 2.8:Khía cạnh bảo trì của công nghệ phần mềm là gì? 14](#_heading=h.4i7ojhp)

[Câu 2.9: Các nguyên nhân chính gây trễ thời hạn khi phát triển phần mềm là gì? 15](#_heading=h.2xcytpi)

[Câu 2.10: Bảo trì phần mềm bao gồm những hoạt động nào? 16](#_heading=h.1ci93xb)

[PHẦN 3: CÂU HỎI THẢO LUẬN NHÓM 16](#_heading=h.3whwml4)

[Câu 3.1: Phân biệt phần mềm hệ thống và phần mềm ứng dụng. 16](#_heading=h.2bn6wsx)

[Câu 3.2: Thảo luận về vai trò của công nghệ phần mềm trong lĩnh vực tài chính. 17](#_heading=h.qsh70q)

[Câu 3.3: Nêu các thách thức thường gặp trong bảo trì phần mềm. 17](#_heading=h.3as4poj)

[Câu 3.4: Vì sao phần mềm thương mại điện tử cần được bảo trì thường xuyên? 18](#_heading=h.1pxezwc)

[Câu 3.5:Phân tích những vấn đề khi yêu cầu khách hàng liên tục thay đổi trong quá trình phát triển phần mềm. 19](#_heading=h.49x2ik5)

[Câu 3.6: So sánh chi phí phát triển và chi phí bảo trì phần mềm. 21](#_heading=h.2p2csry)

[Câu 3.7: Phân biệt các loại yêu cầu trong phát triển phần mềm (chức năng và phi chức năng). 23](#_heading=h.pzw3h4teoxip)

[Câu 3.8:Thảo luận về các mô hình quy trình phát triển phần mềm phổ biến. 23](#_heading=h.3o7alnk)

[Câu 3.9: Đề xuất giải pháp giảm thiểu lỗi phần mềm sau khi bàn giao. 24](#_heading=h.23ckvvd)

[PHẦN 4: CÂU HỎI TÌNH HUỐNG 25](#_heading=h.32hioqz)

[Câu 4.1: 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. 25](#_heading=h.1hmsyys)

[Câu 4.2: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?. 26](#_heading=h.41mghml)

[Câu 4.3: 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? 27](#_heading=h.2grqrue)

[Câu 4.4: 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? 27](#_heading=h.vx1227)

[Câu 4.5: 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? 28](#_heading=h.3fwokq0)

[Câu 4.6: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? 30](#_heading=h.kubh4cqw0r2m)

[Câu 4.7: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? 31](#_heading=h.kqk9o1985btz)

[Câu 4.8: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? 32](#_heading=h.2u6wntf)

[Câu 4.9: 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? 33](#_heading=h.19c6y18)

[Câu 4.10: 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](#_heading=h.3tbugp1) [cầu này mà không làm ảnh hưởng đến hoạt động sản xuất của khách hàng?](#_heading=h.611zzem7g1e1) 35

# PHẦN 1: CÂU HỎI TRẮC NGHIỆM

## Câu 1.1: 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**

**Đáp án : D**

## Câu 1.2: 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

**Đáp án: A**

## Câu 1.3: Quy trình phát triển phần mềm gồm mấy giai đoạn chính?

A. 3

B. 4

C. 5

**D. 6**

**Đáp án: D.** Giải thích: 6 giai đoạn chính (Lấy yêu cầu, Thiết kế, Lập trình, Kiểm thử, Triển khai, Bảo trì)

## Câu 1.4: 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

**Đáp án: C.**

## Câu 1.5: 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%

**Đáp án: C.**

## 

## Câu 1.6: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

**Đáp án: B.**

## Câu 1.7: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**

Đáp án: D

## Câu 1.8: 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**

Đáp án: D

## Câu 1.9: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

Đáp án: B.

## Câu 1.10:Phần mềm có thể được chia thành bao nhiêu loại chính?

A. 2

**B. 3**

C. 4

D. 5

Đáp án: B

## 

# PHẦN 2: CÂU TRẢ LỜI NGẮN

## Câu 2.1: Phần mềm là gì?

Phần mềm là tập hợp các chương trình máy tính, các thủ tục, quy tắc và tài liệu liên quan được thiết kế để thực hiện các nhiệm vụ cụ thể trên một hệ thống máy tính. Phần mềm có thể được chia thành ba loại chính:

* Phần mềm hệ thống (System Software):
  + Quản lý và điều khiển hoạt động của máy tính
  + Ví dụ: Hệ điều hành (Windows, Linux, MacOS), trình điều khiển thiết bị (drivers), các tiện ích hệ thống
* Phần mềm công cụ (Tool Software):
  + Hỗ trợ phát triển và bảo trì phần mềm khác
  + Ví dụ: Trình biên dịch, trình gỡ lỗi, công cụ kiểm thử, IDE (môi trường phát triển tích hợp)
* Phần mềm ứng dụng (Application Software):
  + Phục vụ nhu cầu cụ thể của người dùng
  + Ví dụ: Phần mềm văn phòng (Word, Excel), Trình duyệt web, Phần mềm đồ họa, Trò chơi, Phần mềm kế toán

## Câu 2.2: Công nghệ phần mềm là gì?

Công nghệ phần mềm là một ngành kỹ thuật và quản lý áp dụng các phương pháp có hệ thống, có kỷ luật và định lượng cho việc phát triển, vận hành và bảo trì phần mềm. Nó bao gồm:

* Các quy trình, phương pháp và công cụ để phát triển phần mềm
* Các kỹ thuật quản lý dự án phần mềm
* Các phương pháp đảm bảo chất lượng phần mềm
* Các nguyên tắc thiết kế và kiến trúc phần mềm
* Các phương pháp kiểm thử và bảo trì phần mềm

## Câu 2.3: Các loại phần mềm chính là gì?

**Có 3 loại phần mềm chính:**

* 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 máy giặt, điều hòa.

## Câu 2.4: Tại sao công nghệ phần mềm lại quan trọng?

Khía cạnh kinh tế

* 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.

Khía cạnh công nghệ

* 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.

Khía cạnh bảo trì

* 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.

## Câu 2.5: Quy trình phát triển phần mềm gồm những giai đoạn nào?

Quy trình phát triển phần mềm thường tuân theo **Vòng đời phát triển phần mềm (Software Development Life Cycle - SDLC)**, gồm các giai đoạn chính sau:

* **Phân tích yêu cầu (Requirement Analysis)**

📌 **Mục tiêu**: Hiểu rõ yêu cầu của khách hàng, xác định phạm vi và mục tiêu của phần mềm.  
📌 **Hoạt động chính**:

* Thu thập yêu cầu từ khách hàng hoặc người dùng.
* Phân tích tính khả thi về kỹ thuật và tài chính.
* Viết tài liệu đặc tả yêu cầu phần mềm (SRS - Software Requirement Specification).  
  📌 **Lưu ý**: Nếu yêu cầu không rõ ràng, dự án dễ bị chậm tiến độ và vượt chi phí.
* **Thiết kế hệ thống (System Design)**

📌 **Mục tiêu**: Xây dựng kiến trúc phần mềm dựa trên yêu cầu đã phân tích.  
📌 **Hoạt động chính**:

* Thiết kế kiến trúc hệ thống (phân lớp, module).
* Thiết kế giao diện người dùng (UI/UX).
* Thiết kế cơ sở dữ liệu.
* Viết tài liệu thiết kế (Software Design Document - SDD).  
  📌 **Lưu ý**: Một thiết kế tốt giúp phần mềm dễ bảo trì và mở rộng trong tương lai.
* **Lập trình và triển khai (Implementation & Coding)**

📌 **Mục tiêu**: Viết mã nguồn phần mềm dựa trên thiết kế.  
📌 **Hoạt động chính**:

* Lập trình theo module đã thiết kế.
* Tuân thủ các nguyên tắc lập trình tốt (OOP, SOLID...).
* Code Review để đảm bảo chất lượng mã nguồn.  
  📌 **Lưu ý**: Đây là giai đoạn **tốn nhiều thời gian và nhân lực nhất**.
* **Kiểm thử phần mềm (Testing & Quality Assurance)**

📌 **Mục tiêu**: Đảm bảo phần mềm hoạt động đúng yêu cầu và không có lỗi.  
📌 **Hoạt động chính**:

* **Unit Testing** (kiểm thử từng đơn vị nhỏ).
* **Integration Testing** (kiểm thử tích hợp giữa các module).
* **System Testing** (kiểm thử toàn bộ hệ thống).
* **User Acceptance Testing (UAT)** (kiểm thử bởi người dùng).  
  📌 **Lưu ý**: Giai đoạn này giúp phát hiện lỗi trước khi phần mềm được triển khai.
* **Triển khai và vận hành (Deployment & Implementation)**

📌 **Mục tiêu**: Đưa phần mềm vào sử dụng thực tế.  
📌 **Hoạt động chính**:

* Cài đặt phần mềm trên hệ thống của khách hàng hoặc phát hành sản phẩm.
* Hướng dẫn sử dụng phần mềm.
* Theo dõi hiệu suất của hệ thống.  
  📌 **Lưu ý**: Có thể triển khai theo từng giai đoạn (Beta test trước khi phát hành chính thức).
* **Bảo trì và nâng cấp (Maintenance & Support)**

📌 **Mục tiêu**: Đảm bảo phần mềm hoạt động ổn định và cập nhật khi cần.  
📌 **Hoạt động chính**:

* **Sửa lỗi** phát sinh sau khi triển khai.
* **Nâng cấp tính năng** theo yêu cầu mới.
* **Tối ưu hiệu suất** và bảo mật hệ thống.  
  📌 **Lưu ý**: Đây là giai đoạn **tốn nhiều chi phí nhất** (chiếm khoảng **76% tổng chi phí vòng đời phần mềm**).

## Câu 2.6:Khía cạnh kinh tế của công nghệ phần mềm là gì?

Công nghệ phần mềm không chỉ liên quan đến kỹ thuật lập trình mà còn có tác động mạnh mẽ đến kinh tế. Các khía cạnh kinh tế của công nghệ phần mềm bao gồm:

* **Chi phí phát triển phần mềm** 💰

Chi phí phát triển phần mềm bao gồm:

* **Chi phí nhân công**: Lương cho lập trình viên, tester, quản lý dự án, UX/UI designer...
* **Chi phí phần cứng**: Máy chủ, thiết bị phát triển, hệ thống lưu trữ dữ liệu.
* **Chi phí phần mềm**: Công cụ phát triển (IDE, framework, API, bản quyền phần mềm).
* **Chi phí đào tạo**: Đào tạo nhân viên sử dụng công nghệ mới, quy trình làm việc.
* **Chi phí quản lý dự án**: Giám sát tiến độ, kiểm thử, sửa lỗi.

📌 **Lưu ý**: Nếu không kiểm soát tốt, chi phí có thể vượt mức ngân sách, gây ảnh hưởng đến lợi nhuận.

* **Chi phí bảo trì phần mềm 🛠️**
* Bảo trì phần mềm chiếm **76% tổng chi phí vòng đời phần mềm**.
* Bao gồm:
  + **Sửa lỗi** (Corrective Maintenance).
  + **Cập nhật tính năng** theo yêu cầu người dùng.
  + **Tối ưu hóa hiệu suất** (Performance Improvement).
  + **Bảo trì thích nghi** khi công nghệ thay đổi.

📌 **Ví dụ**: Các phần mềm như Windows, Android, iOS phải cập nhật thường xuyên để đảm bảo bảo mật và hiệu suất.

* **Hiệu quả kinh tế của tự động hóa phần mềm** ⚙️
* **Tự động hóa quy trình phát triển** giúp tiết kiệm thời gian và nhân lực.
* **Áp dụng DevOps, CI/CD** giúp tăng tốc độ phát triển và triển khai.
* **AI & Machine Learning** giúp tối ưu hóa việc kiểm thử và bảo trì.

📌 **Ví dụ**: Các công ty sử dụng **AI để phát hiện lỗi code** tự động, giảm chi phí kiểm thử.

* **Lợi nhuận từ phần mềm (Revenue Streams) 💼**

Các mô hình kiếm tiền từ phần mềm:

* **Bán giấy phép (License-based)**: Windows, Microsoft Office.
* **Mô hình SaaS (Software as a Service)**: Google Drive, Adobe Creative Cloud.
* **Freemium + Quảng cáo**: Facebook, YouTube.
* **Mua trong ứng dụng (In-app purchases)**: Game di động, ứng dụng trên App Store/Google Play.

📌 **Lưu ý**: Chọn mô hình kinh doanh phù hợp giúp tối đa hóa lợi nhuận.

* **Rủi ro kinh tế trong công nghệ phần mềm ⚠️**

Các rủi ro kinh tế có thể xảy ra:

* **Dự án thất bại**: Sai phạm trong quản lý dự án, yêu cầu không rõ ràng, vượt ngân sách.
* **Lỗi phần mềm gây tổn thất tài chính**: Bug nghiêm trọng có thể làm mất dữ liệu hoặc gây gián đoạn dịch vụ.
* **Cạnh tranh khốc liệt**: Các công ty phần mềm phải đổi mới liên tục để duy trì vị thế.
* **Vấn đề bảo mật**: Nếu bị tấn công mạng, doanh nghiệp có thể mất hàng triệu USD (ví dụ: vụ hack Equifax năm 2017).

## Câu 2.7. Khía cạnh công nghệ của công nghệ phần mềm là gì?

Khía cạnh công nghệ trong công nghệ phần mềm đề cập đến các công cụ, phương pháp, kỹ thuật và nền tảng hỗ trợ quá trình phát triển phần mềm. Nó bao gồm:

**Ngôn ngữ lập trình**

Sử dụng các ngôn ngữ như Python, Java, C++, JavaScript, v.v., để xây dựng phần mềm.

Lựa chọn ngôn ngữ phù hợp với yêu cầu hệ thống và hiệu suất mong muốn.

**Mô hình phát triển phần mềm**

Các mô hình như **Waterfall, Agile, Spiral, DevOps**, v.v., giúp quản lý quy trình phát triển.

Agile và DevOps ngày càng phổ biến nhờ tính linh hoạt và khả năng cải tiến liên tục.

**Công cụ và môi trường phát triển**

**IDE (Integrated Development Environment)**: VS Code, IntelliJ, Eclipse, PyCharm.

**Hệ thống quản lý mã nguồn**: Git, GitHub, GitLab, Bitbucket.

**Công cụ CI/CD**: Jenkins, GitHub Actions, GitLab CI/CD hỗ trợ tự động hóa triển khai phần mềm.

**Kiến trúc phần mềm**

**Kiến trúc Monolithic**: Hệ thống đơn khối, tất cả các thành phần gắn liền nhau.

**Kiến trúc Microservices**: Hệ thống chia nhỏ thành các dịch vụ độc lập, dễ bảo trì.

**Kiến trúc Serverless**: Chạy trên nền tảng đám mây mà không cần quản lý máy chủ.

**Cơ sở dữ liệu và quản lý dữ liệu**

**Cơ sở dữ liệu quan hệ (SQL)**: MySQL, PostgreSQL, SQL Server.

**Cơ sở dữ liệu NoSQL**: MongoDB, Firebase, Cassandra.

**Big Data & Data Science**: Hadoop, Spark, AI-driven analytics.

**Bảo mật phần mềm**

Mã hóa dữ liệu, bảo vệ thông tin người dùng.

Kiểm thử bảo mật: OWASP, Penetration Testing.

Cơ chế xác thực: OAuth, JWT, SAML.

**Công nghệ điện toán đám mây & DevOps**

AWS, Azure, Google Cloud giúp triển khai phần mềm nhanh chóng.

DevOps kết hợp phát triển (Dev) và vận hành (Ops) để tối ưu hóa quy trình phần mềm.

**Trí tuệ nhân tạo (AI) và Machine Learning (ML) trong phần mềm**

Ứng dụng AI vào phần mềm như chatbot, xử lý ngôn ngữ tự nhiên (NLP), nhận diện hình ảnh.

Các framework phổ biến: TensorFlow, PyTorch, OpenAI API.

## Câu 2.8:Khía cạnh bảo trì của công nghệ phần mềm là gì?

Bảo trì phần mềm là quá trình chỉnh sửa, cập nhật và tối ưu phần mềm sau khi đã triển khai để đảm bảo nó hoạt động ổn định, an toàn và đáp ứng nhu cầu thay đổi của người dùng. Bảo trì thường chiếm **khoảng 76%** tổng chi phí vòng đời phần mềm.

**Các loại bảo trì phần mềm**

Bảo trì phần mềm có thể được chia thành **bốn loại chính**:

**Bảo trì sửa lỗi (Corrective Maintenance)**

Sửa lỗi phát sinh sau khi triển khai (bug, lỗi logic, lỗi hiệu suất, v.v.).

Được thực hiện dựa trên phản hồi của người dùng hoặc phát hiện trong quá trình vận hành.

**Bảo trì thích nghi (Adaptive Maintenance)**

Điều chỉnh phần mềm để tương thích với thay đổi trong môi trường hoạt động (hệ điều hành, phần cứng, công nghệ mới).

Ví dụ: Cập nhật phần mềm để hỗ trợ phiên bản mới của Windows hoặc trình duyệt web.

**Bảo trì hoàn thiện (Perfective Maintenance)**

Cải tiến hiệu suất, tối ưu mã nguồn, nâng cấp giao diện người dùng để tăng trải nghiệm.

Không sửa lỗi mà tập trung vào việc cải thiện chức năng hiện có.

**Bảo trì phòng ngừa (Preventive Maintenance)**

Dự đoán và ngăn ngừa lỗi tiềm ẩn trước khi chúng gây ra sự cố.

Ví dụ: Cải thiện bảo mật, tối ưu cơ sở dữ liệu để tránh quá tải hệ thống.

**Các thách thức trong bảo trì phần mềm**

**Chi phí cao**: Bảo trì thường chiếm phần lớn ngân sách của vòng đời phần mềm.

**Tính phức tạp**: Phần mềm lớn có nhiều module, việc chỉnh sửa một phần có thể ảnh hưởng đến toàn bộ hệ thống.

**Tài liệu không đầy đủ**: Nếu tài liệu thiết kế và mã nguồn không được duy trì tốt, việc bảo trì trở nên khó khăn.

**Thay đổi yêu cầu liên tục**: Người dùng và doanh nghiệp thường yêu cầu cập nhật mới, gây áp lực lên đội ngũ bảo trì. Công cụ và phương pháp hỗ trợ bảo trì

**Công cụ và phương pháp hỗ trợ bảo trì**

**Quản lý mã nguồn**: Git, GitHub, GitLab giúp theo dõi thay đổi và kiểm soát phiên bản.

**Tự động kiểm thử**: Selenium, JUnit, PyTest giúp kiểm thử phần mềm sau mỗi lần cập nhật.

**Hệ thống giám sát lỗi**: Sentry, New Relic, LogRocket giúp phát hiện lỗi và theo dõi hiệu suất phần mềm.

**Mô hình bảo trì**: Agile, DevOps giúp triển khai bảo trì liên tục và linh hoạt hơn.

## Câu 2.9: Các nguyên nhân chính gây trễ thời hạn khi phát triển phần mềm là gì?

* **Yêu cầu thay đổi**: Thay đổi yêu cầu từ phía khách hàng hoặc các bên liên quan trong quá trình phát triển có thể gây ra sự chậm trễ. Điều này đòi hỏi đội ngũ phải điều chỉnh thiết kế và mã nguồn.
* **Kế hoạch không hợp lý**: Thiếu kinh nghiệm trong việc ước lượng thời gian và nguồn lực cần thiết có thể dẫn đến kế hoạch không thực tế và gây ra trễ hạn.
* **Thiếu nguồn lực**: Sự thiếu hụt nhân lực, thiết bị hoặc tài nguyên cần thiết để hoàn thành dự án có thể gây ra sự chậm trễ.
* **Lỗi và kiểm thử**: Việc phát hiện lỗi trong giai đoạn kiểm thử và phải sửa chữa chúng có thể kéo dài thời gian hoàn thành dự án.
* **Giao tiếp kém**: Thiếu sự giao tiếp và phối hợp hiệu quả giữa các thành viên trong đội ngũ phát triển hoặc giữa đội ngũ và các bên liên quan có thể dẫn đến hiểu lầm và sai sót.
* **Thay đổi nhân sự**: Sự thay đổi nhân sự trong quá trình phát triển có thể gây ra gián đoạn và mất thời gian cho việc bàn giao công việc và làm quen với dự án.
* **Quá tải công việc**: Nếu các thành viên trong đội ngũ phải đảm nhận nhiều công việc cùng lúc, họ có thể không có đủ thời gian để tập trung vào dự án, dẫn đến trễ hạn.

## Câu 2.10: Bảo trì phần mềm bao gồm những hoạt động nào?

* **Sửa lỗi**: Khắc phục các lỗi phần mềm (bugs) phát hiện sau khi phần mềm đã được phát hành. Các lỗi này có thể ảnh hưởng đến chức năng và hiệu suất của phần mềm.
* **Cập nhật và nâng cấp**: Thêm các tính năng mới, cải thiện hiệu suất, hoặc thay đổi phần mềm để phù hợp với các yêu cầu mới từ người dùng hoặc thị trường.
* **Tối ưu hóa**: Cải thiện hiệu quả hoạt động của phần mềm bằng cách tối ưu hóa mã nguồn, giảm thiểu sử dụng tài nguyên hoặc tăng tốc độ xử lý.
* **Bảo mật**: Khắc phục các lỗ hổng bảo mật để bảo vệ phần mềm và dữ liệu người dùng khỏi các mối đe dọa và tấn công.
* **Tương thích**: Đảm bảo phần mềm tương thích với các hệ điều hành, thiết bị, và ứng dụng mới hoặc cập nhật.
* **Tài liệu**: Cập nhật tài liệu hướng dẫn và tài liệu kỹ thuật để phản ánh các thay đổi và cải tiến trong phần mềm.
* **Hỗ trợ người dùng**: Cung cấp hỗ trợ kỹ thuật và giải đáp thắc mắc từ người dùng để giúp họ sử dụng phần mềm hiệu quả hơn.

# PHẦN 3: CÂU HỎI THẢO LUẬN NHÓM

## Câu 3.1: Phân biệt phần mềm hệ thống và phần mềm ứng dụng.

Phân biệt phần mềm hệ thống và phần mềm ứng dụng:

* Phần mềm hệ thống:
  + Mục đích: Quản lý và điều khiển hoạt động của máy tính
  + Người dùng: Chủ yếu tương tác với phần cứng và phục vụ các chương trình khác
  + Không thể thiếu để máy tính hoạt động
  + Ví dụ cụ thể:
    - Windows 11 quản lý bộ nhớ, điều khiển quá trình, quản lý file
    - Driver màn hình NVIDIA giúp card đồ họa giao tiếp với hệ điều hành
    - Phần mềm antivirus bảo vệ hệ thống
* Phần mềm ứng dụng:
  + Mục đích: Phục vụ nhu cầu cụ thể của người dùng
  + Người dùng: Tương tác trực tiếp với người dùng cuối
  + Có thể cài đặt hoặc gỡ bỏ tùy nhu cầu
  + Ví dụ cụ thể:
    - Microsoft Word để soạn thảo văn bản
    - Zoom để họp trực tuyến
    - Adobe Photoshop để chỉnh sửa ảnh

## Câu 3.2: Thảo luận về vai trò của công nghệ phần mềm trong lĩnh vực tài chính.

* Giao dịch ngân hàng:
  + Ví dụ: Ứng dụng mobile banking của Vietcombank cho phép:
    - Chuyển tiền 24/7
    - Thanh toán hóa đơn tự động
    - Mở tài khoản tiết kiệm online
* Quản lý đầu tư:
  + Ví dụ: Phần mềm SSI Trading Pro:
    - Theo dõi biến động chứng khoán realtime
    - Phân tích kỹ thuật với các chỉ báo
    - Đặt lệnh tự động theo điều kiện
* Kế toán doanh nghiệp:
  + Ví dụ: Phần mềm MISA SME:
    - Tự động hóa sổ sách kế toán
    - Lập báo cáo tài chính
    - Quản lý hóa đơn điện tử
* Bảo mật tài chính:
  + Ví dụ: Hệ thống phát hiện gian lận của Mastercard:
    - Phát hiện giao dịch bất thường
    - Cảnh báo real-time
    - Tự động khóa thẻ khi phát hiện rủi ro
* Tất cả những ứng dụng này đều đòi hỏi công nghệ phần mềm hiện đại để đảm bảo:
  + Độ chính xác cao
  + Bảo mật thông tin
  + Xử lý realtime
  + Khả năng mở rộng
  + Tích hợp đa nền tảng

## Câu 3.3: Nêu các thách thức thường gặp trong bảo trì phần mềm.

* Hiểu mã nguồn cũ (Legacy Code):
  + Khi phần mềm đã tồn tại từ lâu, mã nguồn có thể trở nên phức tạp, thiếu tài liệu hoặc được viết bởi các lập trình viên không còn làm việc trong dự án. Việc hiểu và sửa đổi mã nguồn này thường tốn nhiều thời gian và dễ gây lỗi.
* Thay đổi yêu cầu:
  + Người dùng hoặc khách hàng có thể liên tục đưa ra các yêu cầu mới hoặc thay đổi yêu cầu cũ, khiến việc bảo trì phải thích nghi nhanh chóng. Điều này đôi khi xung đột với thiết kế ban đầu của phần mềm.
* Tương thích ngược (Backward Compatibility):
  + Khi cập nhật phần mềm, cần đảm bảo nó vẫn hoạt động tốt với các phiên bản cũ hơn, dữ liệu cũ hoặc hệ thống khác. Điều này có thể hạn chế khả năng áp dụng các công nghệ mới.
* Quản lý lỗi (Bug Management):
  + Việc phát hiện và sửa lỗi là một phần không thể tránh khỏi trong bảo trì. Tuy nhiên, một số lỗi có thể khó tái hiện hoặc liên quan đến các phần khác của hệ thống, làm phức tạp quá trình xử lý.
* Tài nguyên hạn chế:
  + Đội ngũ bảo trì thường có ít thời gian, ngân sách hoặc nhân lực hơn so với giai đoạn phát triển ban đầu, dẫn đến áp lực phải làm việc hiệu quả trong điều kiện không lý tưởng.
* Tác động dây chuyền (Ripple Effect):
  + Việc sửa đổi một phần của phần mềm có thể vô tình gây lỗi ở các phần khác, đặc biệt trong các hệ thống lớn và phức tạp.
* Áp dụng công nghệ mới:
  + Khi công nghệ thay đổi, việc tích hợp các công cụ hoặc nền tảng mới vào phần mềm cũ có thể gặp khó khăn, đòi hỏi phải cân bằng giữa cải tiến và duy trì sự ổn định.
* Kiểm thử không đầy đủ:
  + Sau khi bảo trì, việc kiểm thử toàn diện để đảm bảo không có lỗi mới phát sinh đôi khi bị bỏ qua do áp lực thời gian, dẫn đến rủi ro cho chất lượng phần mềm.

## Câu 3.4: Vì sao phần mềm thương mại điện tử cần được bảo trì thường xuyên?

* Bảo mật dữ liệu
  + Hệ thống thương mại điện tử xử lý thông tin nhạy cảm như dữ liệu khách hàng, đơn hàng và thanh toán.
  + Các lỗ hổng bảo mật có thể bị tin tặc khai thác, vì vậy cần cập nhật vá lỗi và nâng cấp bảo mật thường xuyên.
* Hiệu suất và tốc độ
  + Tốc độ tải trang ảnh hưởng trực tiếp đến trải nghiệm người dùng và tỷ lệ chuyển đổi.
  + Bảo trì giúp tối ưu mã nguồn, cơ sở dữ liệu và máy chủ để hệ thống chạy nhanh hơn, giảm downtime.
* Cập nhật tính năng mới
  + Xu hướng mua sắm và nhu cầu của khách hàng luôn thay đổi, cần bổ sung tính năng mới để cạnh tranh.
* Trải nghiệm người dùng (UX/UI)
  + Cần kiểm tra và cải thiện giao diện để đảm bảo tính thân thiện, dễ sử dụng và hỗ trợ tốt trên mọi thiết bị.
  + Phản hồi từ khách hàng cũng giúp doanh nghiệp điều chỉnh chức năng phù hợp.
* Khả năng mở rộng
  + Khi số lượng khách hàng và đơn hàng tăng lên, hệ thống cần được tối ưu và mở rộng để tránh quá tải.
  + Việc bảo trì định kỳ giúp đảm bảo hệ thống có thể phát triển mà không gặp sự cố.
* Khắc phục lỗi (Bug Fixes)
  + Lỗi phần mềm có thể ảnh hưởng đến trải nghiệm khách hàng hoặc gây gián đoạn kinh doanh.
  + Bảo trì giúp phát hiện và sửa lỗi kịp thời, đảm bảo hoạt động liên tục.

## Câu 3.5:Phân tích những vấn đề khi yêu cầu khách hàng liên tục thay đổi trong quá trình phát triển phần mềm.

Trong quá trình phát triển phần mềm, việc khách hàng liên tục thay đổi yêu cầu có thể gây ra nhiều vấn đề ảnh hưởng đến tiến độ, chi phí và chất lượng sản phẩm. Dưới đây là những phân tích chi tiết về các vấn đề này:

* **Ảnh hưởng đến tiến độ và thời gian phát triển ⏳**
* Khi yêu cầu thay đổi liên tục, nhóm phát triển phải cập nhật thiết kế, sửa đổi mã nguồn và kiểm thử lại, dẫn đến **chậm tiến độ**.
* Một số thay đổi có thể ảnh hưởng đến **toàn bộ kiến trúc phần mềm**, khiến việc điều chỉnh trở nên phức tạp.
* Nếu nhóm phát triển phải thường xuyên quay lại các giai đoạn trước đó, **thời gian hoàn thành dự án sẽ bị kéo dài**.

📌 **Ví dụ**: Trong một dự án phần mềm quản lý doanh nghiệp, nếu khách hàng yêu cầu thay đổi giao diện người dùng hoặc quy trình nghiệp vụ nhiều lần, thời gian hoàn thành có thể bị kéo dài gấp đôi so với kế hoạch ban đầu.

* **Gia tăng chi phí phát triển 💰**
* Việc thay đổi yêu cầu thường dẫn đến **phát sinh chi phí** cho việc tái thiết kế, lập trình lại và kiểm thử bổ sung.
* Nếu dự án áp dụng mô hình **Waterfall**, việc thay đổi ở giai đoạn sau có thể gây ra **chi phí sửa đổi rất cao**.
* **Nhân sự phải làm thêm giờ**, kéo dài thời gian thuê tài nguyên (máy chủ, công cụ phát triển), làm tăng tổng chi phí dự án.

📌 **Ví dụ**: Một dự án phần mềm thương mại điện tử đã gần hoàn thành nhưng khách hàng yêu cầu thêm tính năng thanh toán bằng tiền điện tử, nhóm phát triển phải viết lại phần lớn mã nguồn và hệ thống bảo mật, làm chi phí tăng đáng kể.

* **Ảnh hưởng đến chất lượng phần mềm 🛠️**
* Nếu các thay đổi không được kiểm soát tốt, phần mềm có thể trở nên **phức tạp, thiếu ổn định và dễ xảy ra lỗi**.
* Mỗi lần thay đổi đều có nguy cơ tạo ra **bug mới**, đặc biệt là khi không có đủ thời gian kiểm thử kỹ lưỡng.
* Nếu nhóm phát triển cố gắng **đẩy nhanh tiến độ để đáp ứng thay đổi**, họ có thể bỏ qua một số quy trình quan trọng như **kiểm thử và tối ưu hóa**, dẫn đến phần mềm kém chất lượng.

📌 **Ví dụ**: Một phần mềm kế toán được cập nhật liên tục theo yêu cầu khách hàng nhưng không được kiểm thử kỹ, dẫn đến lỗi tính toán sai số liệu tài chính.

* **Khó khăn trong quản lý dự án 📊**
* Khi yêu cầu thay đổi liên tục, **việc lập kế hoạch, phân công công việc và ước tính tài nguyên** trở nên khó khăn.
* Các tài liệu dự án (đặc tả yêu cầu, thiết kế hệ thống) phải cập nhật thường xuyên, gây tốn nhiều thời gian và công sức.
* Có thể xảy ra xung đột giữa **nhóm phát triển và khách hàng**, đặc biệt nếu thay đổi không được đánh giá kỹ lưỡng trước khi thực hiện.

📌 **Ví dụ**: Trong một dự án phần mềm ngân hàng, nếu khách hàng thay đổi quy trình xác thực giao dịch nhiều lần, nhóm phát triển phải liên tục cập nhật hệ thống bảo mật, gây áp lực lớn cho quản lý dự án.

* **Giải pháp để kiểm soát thay đổi yêu cầu ✅**

Để hạn chế tác động tiêu cực của việc thay đổi yêu cầu, có thể áp dụng các biện pháp sau:

🔹 Sử dụng mô hình phát triển linh hoạt (Agile, Scrum)

* **Agile** cho phép khách hàng tham gia vào quá trình phát triển theo từng giai đoạn, giúp kiểm soát tốt hơn các thay đổi.
* **Scrum** chia dự án thành các **Sprint (chu kỳ phát triển ngắn)**, giúp nhóm phát triển thích nghi nhanh với yêu cầu mới.

📌 **Ví dụ**: Một công ty phần mềm sử dụng Scrum để phát triển ứng dụng di động, mỗi 2 tuần họ họp với khách hàng để xem xét tiến độ và điều chỉnh yêu cầu nếu cần.

🔹 Thỏa thuận chặt chẽ về phạm vi yêu cầu ban đầu

* Xác định **rõ ràng yêu cầu phần mềm** ngay từ đầu, ghi chép chi tiết trong tài liệu đặc tả (SRS).
* Áp dụng **hợp đồng linh hoạt**, trong đó quy định cách xử lý yêu cầu thay đổi (chi phí, thời gian, ưu tiên).

📌 **Ví dụ**: Một công ty phần mềm ký hợp đồng với khách hàng, trong đó quy định rằng **chỉ có tối đa 3 lần thay đổi yêu cầu miễn phí**, những thay đổi sau đó sẽ tính phí bổ sung.

🔹 Sử dụng công cụ quản lý yêu cầu (JIRA, Trello, Confluence)

* Các công cụ này giúp theo dõi **các yêu cầu mới, thay đổi và mức độ ưu tiên**.
* Giúp nhóm phát triển dễ dàng phân tích tác động của các thay đổi trước khi thực hiện.

📌 **Ví dụ**: Công ty A sử dụng JIRA để quản lý các yêu cầu thay đổi, khách hàng có thể theo dõi trạng thái yêu cầu mà không cần làm gián đoạn nhóm phát triển.

🔹 Đánh giá tác động trước khi chấp nhận thay đổi

* Trước khi thực hiện thay đổi, nhóm phát triển cần **phân tích ảnh hưởng** đến chi phí, tiến độ và chất lượng phần mềm.
* Nếu thay đổi quá lớn, có thể đề xuất **hoãn sang phiên bản sau** để đảm bảo tiến độ ban đầu.

📌 **Ví dụ**: Khi khách hàng yêu cầu bổ sung tính năng mới vào phần mềm kế toán, nhóm phát triển đánh giá và đề xuất đưa vào phiên bản **v2.0 thay vì làm ngay trong phiên bản hiện tại**.

## Câu 3.6: So sánh chi phí phát triển và chi phí bảo trì phần mềm.

Chi phí phát triển và chi phí bảo trì phần mềm là hai thành phần quan trọng trong vòng đời phần mềm. Dưới đây là sự so sánh giữa hai loại chi phí này:

* **Khái niệm**

🔹 **Chi phí phát triển phần mềm**:

* Là chi phí liên quan đến việc xây dựng một phần mềm từ đầu đến khi hoàn thành sản phẩm đầu tiên.
* Bao gồm các giai đoạn như **phân tích yêu cầu, thiết kế, lập trình, kiểm thử và triển khai**.

🔹 **Chi phí bảo trì phần mềm**:

* Là chi phí dành cho việc duy trì và nâng cấp phần mềm sau khi đã được triển khai.
* Bao gồm **sửa lỗi, cập nhật tính năng mới, tối ưu hiệu suất và hỗ trợ người dùng**.
* **Tỷ lệ chi phí trong vòng đời phần mềm**
* **Chi phí bảo trì phần mềm thường chiếm tỷ lệ cao hơn chi phí phát triển**.
* Theo nghiên cứu, **chi phí bảo trì có thể chiếm từ 50% đến 80% tổng chi phí vòng đời phần mềm**.

📌 **Ví dụ**:

* **Chi phí phát triển** một hệ thống phần mềm doanh nghiệp ban đầu là **100.000 USD**.
* **Chi phí bảo trì** có thể lên đến **500.000 USD** trong 10 năm tiếp theo do cập nhật, sửa lỗi và cải tiến hệ thống.
* **Thành phần chi phí**

|  |  |  |
| --- | --- | --- |
| **Yếu tố** | **Chi phí phát triển** | **Chi phí bảo trì** |
| **Phân tích yêu cầu** | Nghiên cứu, thu thập yêu cầu khách hàng | Điều chỉnh yêu cầu mới từ khách hàng |
| **Thiết kế** | Thiết kế kiến trúc hệ thống, giao diện người dùng | Thay đổi thiết kế do yêu cầu mới hoặc công nghệ thay đổi |
| **Lập trình** | Viết mã nguồn từ đầu | Sửa lỗi, cải tiến mã nguồn hiện có |
| **Kiểm thử** | Kiểm thử toàn diện trước khi phát hành | Kiểm thử lại sau khi sửa lỗi hoặc cập nhật |
| **Triển khai** | Cài đặt, cấu hình ban đầu | Cập nhật, nâng cấp phiên bản phần mềm |
| **Hỗ trợ người dùng** | Đào tạo ban đầu cho khách hàng | Hỗ trợ sử dụng, xử lý lỗi phát sinh |

* **Nguyên nhân chi phí bảo trì cao hơn phát triển**
* **Phần mềm lớn và phức tạp hơn theo thời gian**: Các hệ thống lâu đời cần bảo trì nhiều hơn do công nghệ cũ dần lạc hậu.
* **Khách hàng thay đổi yêu cầu**: Các doanh nghiệp luôn muốn bổ sung tính năng mới, thay đổi theo thị trường.
* **Sửa lỗi và cập nhật bảo mật**: Phần mềm luôn cần vá lỗi bảo mật và cải thiện hiệu suất để tránh rủi ro.
* **Chi phí nhân công**: Lập trình viên phải nghiên cứu lại mã nguồn cũ, dẫn đến thời gian sửa chữa lâu hơn.
* **Cách tối ưu chi phí bảo trì phần mềm**

✅ **Lập kế hoạch phát triển tốt**: Thiết kế phần mềm dễ bảo trì ngay từ đầu giúp giảm chi phí sửa đổi sau này.  
✅ **Áp dụng mô hình phát triển linh hoạt (Agile, DevOps)**: Giúp triển khai và cập nhật dễ dàng hơn.  
✅ **Tự động hóa kiểm thử và giám sát lỗi**: Giúp phát hiện lỗi sớm, giảm chi phí sửa chữa.  
✅ **Viết mã nguồn sạch, dễ đọc**: Giúp lập trình viên dễ dàng bảo trì sau này.

## Câu 3.7: Phân biệt các loại yêu cầu trong phát triển phần mềm (chức năng và phi chức năng).

Trong phát triển phần mềm, yêu cầu được chia thành hai loại chính: **yêu cầu chức năng (Functional Requirements)** và **yêu cầu phi chức năng (Non-functional Requirements)**.

|  |  |  |
| --- | --- | --- |
| **Tiêu chí** | **Yêu cầu chức năng** | **Yêu cầu phi chức năng** |
| **Định nghĩa** | Mô tả các chức năng, tính năng mà phần mềm phải cung cấp. | Mô tả cách hệ thống vận hành hơn là các chức năng cụ thể. |
| **Mục tiêu** | Xác định **hệ thống làm gì**. | Xác định **hệ thống hoạt động như thế nào**. |
| **Ví dụ** | - Đăng nhập/đăng xuất.  - Quản lý người dùng.  - Xử lý thanh toán. | - Tốc độ phản hồi dưới 2 giây.  - Hệ thống có thể xử lý 10.000 người dùng đồng thời.  - Độ bảo mật cao, mã hóa dữ liệu. |
| **Tầm quan trọng** | Nếu không có, hệ thống **không thể hoạt động**. | Nếu không có, hệ thống **vẫn hoạt động nhưng có thể kém hiệu quả**. |
| **Mức độ đo lường** | Dễ đo lường thông qua test case. | Khó đo lường, thường đánh giá bằng tiêu chuẩn hoặc benchmarking. |

## Câu 3.8:Thảo luận về các mô hình quy trình phát triển phần mềm phổ biến.

Mô hình phát triển phần mềm mô tả cách phần mềm được thiết kế, phát triển, kiểm thử và triển khai. Dưới đây là một số mô hình phổ biến:

#### Mô hình Thác nước (Waterfall Model)

✅ **Ưu điểm:**

* Dễ hiểu, dễ quản lý do có quy trình rõ ràng.
* Phù hợp với dự án có yêu cầu cố định, ít thay đổi.

❌ **Nhược điểm:**

* Không linh hoạt, khó thay đổi khi yêu cầu thay đổi.
* Không có sản phẩm chạy thử cho đến giai đoạn cuối.

#### Mô hình Phát triển Linh hoạt (Agile Model)

✅ **Ưu điểm:**

* Linh hoạt, dễ thích nghi với thay đổi yêu cầu.
* Phát triển theo từng vòng lặp (iteration), có sản phẩm chạy thử sớm.

❌ **Nhược điểm:**

* Cần sự phối hợp chặt chẽ giữa khách hàng và nhóm phát triển.
* Khó dự đoán chi phí và thời gian hoàn thành chính xác.

#### Mô hình Spiral (Mô hình xoắn ốc)

✅ **Ưu điểm:**

* Kết hợp ưu điểm của Waterfall và Agile, có khả năng quản lý rủi ro tốt.
* Phù hợp với dự án lớn, phức tạp.

❌ **Nhược điểm:**

* Tốn nhiều thời gian và chi phí để thực hiện.
* Đòi hỏi đội ngũ giàu kinh nghiệm để quản lý rủi ro.

#### Mô hình DevOps

✅ **Ưu điểm:**

* Tích hợp liên tục (CI/CD), triển khai nhanh chóng.
* Giúp phát hiện lỗi sớm, giảm thời gian đưa sản phẩm ra thị trường.

❌ **Nhược điểm:**

* Yêu cầu tự động hóa cao, cần nhiều công cụ hỗ trợ.
* Phù hợp với dự án có quy mô lớn, đòi hỏi sự phối hợp giữa đội phát triển và vận hành.

## Câu 3.9: Đề xuất giải pháp giảm thiểu lỗi phần mềm sau khi bàn giao.

* **Tăng cường kiểm thử:** Áp dụng kiểm thử tự động (Automation Testing) và kiểm thử tích hợp (Integration Testing) trong suốt quá trình phát triển.
* **Quy trình kiểm soát chất lượng:** Thiết lập quy trình quản lý lỗi (bug tracking) hiệu quả, báo cáo và xử lý lỗi kịp thời.
* **Đào tạo đội ngũ:** Đào tạo lập trình viên và nhân viên kiểm thử về tiêu chuẩn coding và các công cụ kiểm thử hiện đại.
* **Thu thập phản hồi:** Thực hiện kiểm thử người dùng (User Acceptance Testing) trước khi bàn giao chính thức để phát hiện sớm các vấn đề.

*Ví dụ:* Sử dụng công cụ Selenium cho kiểm thử tự động kết hợp với quy trình kiểm thử do khách hàng thực hiện (UAT) để đảm bảo sản phẩm không còn lỗi nghiêm trọng.

**Câu 3.10:Vai trò của đội kiểm thử trong quy trình phát triển phần mềm.**

* **Phát hiện lỗi**: Xác định, báo cáo và theo dõi lỗi trong quá trình phát triển, từ kiểm thử đơn vị đến kiểm thử hệ thống.
* **Đảm bảo chất lượng**: Đảm bảo rằng phần mềm đáp ứng đầy đủ yêu cầu chức năng và phi chức năng.
* **Đánh giá hiệu năng**: Kiểm thử hiệu năng, bảo mật, và khả năng mở rộng của phần mềm trước khi bàn giao.
* **Phản hồi liên tục**: Cung cấp phản hồi nhanh cho đội phát triển để cải thiện và tối ưu hóa sản phẩm.

***Ví dụ:*** Trong dự án phát triển hệ thống ngân hàng, đội kiểm thử sẽ kiểm tra kỹ các chức năng giao dịch, bảo mật và hiệu năng để đảm bảo hệ thống hoạt động ổn định và an toàn trước khi đưa vào vận hành.

# PHẦN 4: CÂU HỎI TÌNH HUỐNG

## Câu 4.1: 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.

* Tình huống: Lỗi phần mềm quản lý tài chính
* Các bước giải quyết:
  + Phân tích vấn đề:
    - Tiến hành thu thập chi tiết về các lỗi phát sinh
    - Xác định ngưỡng giá trị giao dịch gây lỗi
    - Kiểm tra logs hệ thống để tìm nguyên nhân gốc rễ
  + Giải pháp khẩn cấp ngắn hạn:
    - Triển khai hotfix để xử lý các lỗi nghiêm trọng nhất
    - Đặt giới hạn tạm thời cho các giao dịch giá trị lớn
    - Cung cấp hướng dẫn giải pháp tạm thời cho khách hàng
  + Giải pháp dài hạn:
    - Nâng cấp hệ thống xử lý giao dịch lớn
    - Tăng cường kiểm thử với các test case đa dạng
    - Cải thiện quy trình giám sát và cảnh báo sớm
  + Phòng ngừa tương lai:
    - Cập nhật quy trình kiểm thử
    - Thêm các stress test cho giao dịch giá trị lớn
    - Thiết lập hệ thống backup và phục hồi dữ liệu

## 

## Câu 4.2: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?.

## 

* Tình huống: Yêu cầu bổ sung tính năng quản lý kho thuốc
* Các bước xử lý:
  + Đánh giá tác động:
    - Phân tích ảnh hưởng đến tiến độ dự án
    - Đánh giá tài nguyên cần thiết
    - Xem xét các phụ thuộc kỹ thuật
* Đề xuất phương án:
  + Phương án 1: Tách thành phase 2
    - Hoàn thành phase 1 theo kế hoạch ban đầu
    - Phát triển tính năng quản lý kho thuốc ở phase 2
    - Ưu điểm: Không ảnh hưởng tiến độ hiện tại
    - Nhược điểm: Tốn thời gian tích hợp sau
  + Phương án 2: Tích hợp vào dự án hiện tại
    - Điều chỉnh kế hoạch và resources
    - Mở rộng thời gian kiểm thử
    - Ưu điểm: Đồng bộ hệ thống
    - Nhược điểm: Kéo dài thời gian dự án
* Quy trình thực hiện:
  + Họp với khách hàng thống nhất phương án
  + Lập kế hoạch chi tiết cho phương án được chọn
  + Điều chỉnh hợp đồng và ngân sách
  + Phân công lại nhiệm vụ cho team
* Kiến nghị quy trình:
  + Cần có quy trình change request chuẩn
  + Đưa ra các điều khoản về thay đổi yêu cầu trong hợp đồng
  + Tăng cường trao đổi với khách hàng từ giai đoạn đầu
* Với mỗi phương án, cần thảo luận kỹ với khách hàng về:
  + Chi phí phát sinh
  + Thời gian thực hiện
  + Ảnh hưởng đến chất lượng
  + Rủi ro có thể gặp phải

## Câu 4.3: 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?

* Họp lại hội ý với các thành viên trong nhóm
  + Tổ chức cuộc họp nội bộ để xác định rõ những khó khăn và điểm chưa hiểu trong yêu cầu khách hàng.
  + Yêu cầu từng thành viên chia sẻ về phần công việc của mình để phát hiện các điểm mâu thuẫn hoặc hiểu sai.
  + Cung cấp tài liệu chi tiết, sơ đồ và ví dụ cụ thể để làm rõ yêu cầu.
* Trao đổi với khách hàng về nhu cầu
  + Liên hệ trực tiếp với khách hàng để xác minh và làm rõ các yêu cầu quan trọng.
  + Sử dụng tài liệu mô tả yêu cầu (Use Case) để xác nhận lại với khách hàng, tránh hiểu sai.
  + Ghi nhận phản hồi từ khách hàng và cập nhật kịp thời vào kế hoạch dự án.
* Tiến hành dự án và đảm bảo tiến độ
  + Điều chỉnh lại kế hoạch dự án dựa trên thông tin đã xác minh từ khách hàng.
  + Ưu tiên hoàn thành các tính năng quan trọng trước, đồng thời theo dõi tiến độ thường xuyên.

## Câu 4.4: 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?

* Thu thập ý kiến của người dùng
  + Tổng hợp lại ý kiến phản hồi của người dùng, xem lại những bất cập ở phần mềm ở đâu.
* Tiến hành phân tích lại nhu cầu của người dùng
  + Tìm hiểu xem người dùng mong muốn giao diện như thế nào để dễ dàng thao tác hơn.
  + Xác định các điểm gây khó khăn, như quy trình phức tạp, bố cục rối mắt, hoặc thiếu hướng dẫn sử dụng.
* Cập nhật cải tiến giao diện phần mềm
  + Thiết kế lại bố cục sao cho trực quan hơn, giúp người dùng dễ sử dụng.
  + Đơn giản hóa các bước thao tác, giảm bớt những quy trình không cần thiết.
  + Thêm hướng dẫn sử dụng hoặc gợi ý thao tác ngay trong ứng dụng để hỗ trợ người dùng mới.
  + Kiểm thử giao diện mới với một nhóm nhỏ người dùng trước khi triển khai rộng rãi.

## Câu 4.5: 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?

Việc một dự án phát triển phần mềm vượt ngân sách do kéo dài thời gian hoàn thành là một vấn đề phổ biến. Để hạn chế tình trạng này trong tương lai, quản lý dự án có thể áp dụng các giải pháp sau:

* **Lập kế hoạch chi tiết và thực tế**

🔹 Xác định phạm vi dự án rõ ràng

* Đảm bảo yêu cầu dự án được xác định chi tiết ngay từ đầu, tránh mở rộng phạm vi (scope creep) trong quá trình thực hiện.
* Sử dụng Tài liệu đặc tả yêu cầu phần mềm (SRS – Software Requirement Specification) để làm căn cứ triển khai.

**🔹 Xây dựng kế hoạch thực tế**

* Dự báo chính xác thời gian và nguồn lực cần thiết bằng cách tham khảo dữ liệu từ các dự án trước.
* Sử dụng phương pháp Ước lượng ba điểm (Three-Point Estimation) để tính toán thời gian hợp lý.

🔹 Thiết lập quỹ dự phòng ngân sách

* Dự phòng khoảng 10 – 20% ngân sách để xử lý các vấn đề phát sinh.
* **Quản lý rủi ro chặt chẽ**

🔹 Xác định rủi ro sớm

* Lập danh sách các rủi ro tiềm ẩn như thay đổi yêu cầu, thiếu nhân lực, vấn đề kỹ thuật,...
* Đánh giá mức độ ảnh hưởng của từng rủi ro và chuẩn bị kế hoạch giảm thiểu.

🔹 Giám sát tiến độ thường xuyên

* Sử dụng các công cụ quản lý dự án như Jira, Trello, Microsoft Project để theo dõi tiến độ theo từng giai đoạn.
* Tổ chức họp daily stand-up hoặc weekly review để đảm bảo nhóm làm việc đúng kế hoạch.

🔹 Kiểm soát thay đổi yêu cầu

* Áp dụng quy trình Change Request để đánh giá tác động chi phí và thời gian trước khi chấp nhận thay đổi.
* **Tối ưu quy trình phát triển phần mềm**

🔹 Áp dụng mô hình phát triển linh hoạt (Agile, Scrum)

* Phát triển theo từng giai đoạn ngắn (Sprint) giúp dễ dàng điều chỉnh nếu có thay đổi yêu cầu.
* Kiểm tra sản phẩm liên tục để phát hiện lỗi sớm, tránh sửa chữa tốn kém về sau.

🔹 Tăng cường kiểm thử tự động (Automation Testing)

* Giúp phát hiện lỗi nhanh hơn, giảm chi phí sửa lỗi trong giai đoạn sau.

🔹 Chuẩn hóa quy trình phát triển

* Áp dụng DevOps để tối ưu hóa việc triển khai và bảo trì phần mềm, giảm thời gian chờ đợi giữa các bộ phận.
* **Quản lý nguồn lực hiệu quả**

🔹 Đánh giá và phân bổ nhân sự hợp lý

* Đảm bảo nhóm có đủ kỹ năng để hoàn thành công việc, tránh tình trạng thiếu nhân lực làm chậm tiến độ.
* Nếu cần thiết, có thể thuê ngoài (outsourcing) để giảm áp lực lên đội ngũ nội bộ.

🔹 Giám sát năng suất làm việc

* Theo dõi tiến độ công việc bằng KPIs (Key Performance Indicators).
* Nếu thấy tiến độ chậm, cần điều chỉnh nhân sự hoặc quy trình làm việc kịp thời.
* **Tăng cường giao tiếp và phối hợp giữa các bên**

🔹 Giao tiếp thường xuyên với khách hàng và đội ngũ phát triển

* Tổ chức họp định kỳ với khách hàng để cập nhật tiến độ, tránh hiểu lầm về yêu cầu.
* Đảm bảo các bộ phận trong nhóm làm việc chặt chẽ, tránh sai sót do thiếu thông tin.

🔹 Tạo môi trường làm việc hiệu quả

* Khuyến khích các thành viên chia sẻ khó khăn để nhanh chóng tìm ra giải pháp.

## 

## 

## Câu 4.6: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?

Việc quyết định có sửa lỗi hay không trong quá trình bảo trì phần mềm quản lý khách sạn cần được đánh giá dựa trên nhiều yếu tố. Dưới đây là cách tiếp cận hợp lý:

* **Phân tích mức độ ảnh hưởng của lỗi**

🔹 Lỗi có thực sự "nhỏ" hay không?

* Lỗi có ảnh hưởng đến tính chính xác của dữ liệu, bảo mật hệ thống, hay trải nghiệm người dùng không?
* Nếu lỗi gây ra hiểu nhầm hoặc sai sót trong vận hành, có thể cần sửa ngay.

🔹 Tác động lâu dài

* Nếu lỗi có thể tích lũy thành vấn đề lớn hơn theo thời gian, cần cân nhắc sửa sớm để tránh chi phí cao hơn về sau.
* **Đánh giá chi phí sửa lỗi**

🔹 Chi phí sửa lỗi có hợp lý so với lợi ích mang lại không?

* Nếu sửa lỗi đòi hỏi thay đổi nhiều trong hệ thống hoặc ảnh hưởng đến các chức năng khác, chi phí sẽ cao.
* So sánh với tác động nếu không sửa: liệu có làm giảm hiệu suất hoặc gây mất khách hàng không?

🔹 Có phương án thay thế không?

* Có thể tìm cách khắc phục tạm thời (workaround) thay vì sửa lỗi hoàn toàn?
* Có thể gộp sửa lỗi này vào lần cập nhật phần mềm tiếp theo để tối ưu chi phí không?
* **Đánh giá mức độ ưu tiên sửa lỗi**

🔸 Nếu lỗi ảnh hưởng đến bảo mật, hiệu suất hoặc trải nghiệm người dùng quan trọng, cần sửa ngay.  
🔸 Nếu lỗi không ảnh hưởng đáng kể và chi phí sửa cao, có thể trì hoãn và sửa trong bản cập nhật tiếp theo.  
🔸 Nếu lỗi có thể gây hậu quả lớn trong tương lai, dù hiện tại nhỏ, nên xem xét sửa sớm để tránh chi phí tăng dần theo thời gian.

## 

## Câu 4.7: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?

**1. Đánh giá tính khả thi của yêu cầu**

* Phân tích tiến độ hiện tại: Xác định còn bao nhiêu công việc chưa hoàn thành và so sánh với thời gian còn lại.
* Xác định rủi ro: Nếu cố gắng đẩy nhanh tiến độ mà không có kế hoạch rõ ràng, chất lượng sản phẩm có thể bị ảnh hưởng.
* Họp với khách hàng: Nếu yêu cầu không khả thi, cần thảo luận lại với khách hàng để tìm giải pháp phù hợp.

**2. Điều chỉnh phạm vi dự án (Scope Adjustment)**

* Xác định tính năng quan trọng: Nếu không thể hoàn thành tất cả các tính năng đúng hạn, có thể thỏa thuận với khách hàng về việc triển khai giai đoạn đầu với các tính năng cốt lõi và bổ sung các tính năng khác sau đó.
* Cắt giảm yêu cầu không cần thiết: Loại bỏ hoặc trì hoãn các tính năng ít quan trọng để đảm bảo tiến độ.

**3. Tăng cường nhân lực và tài nguyên**

* Thuê thêm lập trình viên hợp đồng hoặc freelancer: Giúp tăng tốc độ phát triển mà không làm quá tải nhân sự hiện tại.
* Huy động nhân lực nội bộ: Xem xét khả năng phân bổ lại công việc trong công ty để hỗ trợ dự án.
* Tăng ca có kiểm soát: Nếu cần làm thêm giờ, cần đảm bảo nhân viên không bị kiệt sức, tránh ảnh hưởng đến chất lượng sản phẩm.

**4. Cải thiện hiệu suất làm việc**

* Áp dụng mô hình phát triển nhanh (Agile, Scrum): Rút ngắn chu kỳ phát triển, tổ chức các sprint ngắn hơn để theo dõi tiến độ sát sao hơn.
* Tự động hóa quy trình: Sử dụng các công cụ kiểm thử tự động (Selenium, JUnit) để giảm thời gian kiểm thử thủ công.
* Tăng cường giao tiếp nội bộ: Tổ chức họp hàng ngày (daily stand-up) để nhanh chóng xử lý các vướng mắc.

**5. Thương lượng với khách hàng về phương án thay thế**

Nếu sau khi đánh giá, việc rút ngắn thời gian vẫn không khả thi, cần trao đổi thẳng thắn với khách hàng và đề xuất các phương án:

* Phương án 1: Giai đoạn triển khai sớm hơn nhưng không đầy đủ tính năng (MVP - Minimum Viable Product).
* Phương án 2: Hoàn thành đúng tiến độ nhưng với chất lượng đảm bảo, tránh rủi ro về lỗi phần mềm.
* Phương án 3: Khách hàng hỗ trợ thêm tài nguyên (ngân sách, nhân sự) để tăng tốc phát triển.

## 

## Câu 4.8: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?

**Giải pháp khắc phục tình huống**

Công ty phần mềm nhỏ gặp phải vấn đề do thay đổi công nghệ liên tục, dẫn đến dự án bị kéo dài và chi phí tăng cao. Để khắc phục vấn đề này, có thể áp dụng các giải pháp sau:

**1. Xác định công nghệ phù hợp ngay từ đầu**

* Phân tích yêu cầu kỹ thuật và nghiệp vụ để chọn công nghệ phù hợp với dự án.
* Ưu tiên công nghệ mà nhóm phát triển có kinh nghiệm, tránh học công nghệ mới trong quá trình phát triển trừ khi thực sự cần thiết.
* Tham khảo công nghệ phổ biến với nhiều tài liệu hỗ trợ, cộng đồng lớn như Flutter, React Native (cho ứng dụng di động đa nền tảng) hoặc Swift/Kotlin (cho ứng dụng gốc).

**2. Ổn định chiến lược phát triển, tránh thay đổi không cần thiết**

* Trước khi bắt đầu, xây dựng kế hoạch công nghệ chi tiết và cam kết theo đuổi đến cuối dự án.
* Nếu cần thay đổi, phải có quy trình phê duyệt chặt chẽ, đánh giá rủi ro, thời gian và chi phí trước khi quyết định.

**3. Nâng cao năng lực đội ngũ**

* Nếu đội ngũ thiếu kinh nghiệm, nên đào tạo nội bộ hoặc thuê chuyên gia tư vấn trước khi bắt đầu dự án.
* Khuyến khích học tập công nghệ trước khi áp dụng vào dự án thực tế thông qua các dự án nhỏ hoặc thử nghiệm (PoC – Proof of Concept).

**4. Áp dụng phương pháp phát triển linh hoạt (Agile)**

* Chia nhỏ dự án thành các giai đoạn (Sprint) để kiểm soát tiến độ và chất lượng tốt hơn.
* Định kỳ đánh giá tiến độ, phát hiện sớm vấn đề công nghệ để điều chỉnh kịp thời.

**5. Kiểm soát chi phí và thời gian**

* Xác định ngân sách cố định cho công nghệ để tránh chi tiêu không kiểm soát.
* Sử dụng công nghệ mã nguồn mở (Open-source) nếu phù hợp để tiết kiệm chi phí.
* Giám sát tiến độ thường xuyên bằng các công cụ quản lý dự án như Jira, Trello để tránh kéo dài thời gian không cần thiết.

## 

## Câu 4.9: 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?

**1. Khẩn cấp xác định và đánh giá lỗi:**

* Xác định mức độ nghiêm trọng:  
  • Thu thập log hệ thống, báo cáo lỗi từ đội phát triển và khách hàng.  
  • Phân tích nguyên nhân gốc rễ của lỗi bảo mật và xác định các điểm yếu có thể bị khai thác.  
  • Sử dụng các công cụ quét lỗ hổng (vulnerability scanners) và phân tích mã nguồn để đánh giá mức độ rủi ro.
* Đánh giá tác động:  
  • Xác định dữ liệu nào có thể bị xâm nhập hoặc rò rỉ.  
  • Ước lượng nguy cơ khai thác của hacker và mức độ thiệt hại có thể xảy ra (mất mát dữ liệu, gián đoạn dịch vụ,…).  
  • Đánh giá sự liên quan của lỗi đối với toàn bộ hệ thống và các module liên quan.

**2. Thông báo cho khách hàng:**

* Liên hệ ngay lập tức:  
  • Sử dụng kênh liên lạc khẩn cấp đã được thiết lập trước đó (email, điện thoại, hoặc họp trực tuyến) để thông báo.  
  • Trình bày rõ ràng tình hình lỗi và mức độ nguy cơ ảnh hưởng đến hoạt động của khách hàng.
* Minh bạch và trung thực:  
  • Giải thích nguyên nhân gây ra lỗi (nếu có thể) và các biện pháp khắc phục đang được thực hiện.  
  • Cam kết sẽ cập nhật tiến độ xử lý lỗi và hỗ trợ khách hàng trong suốt quá trình khắc phục.

**3. Tập trung nguồn lực:**

* Huy động đội ngũ chuyên gia:  
  • Phân công rõ ràng các chuyên gia bảo mật, lập trình viên và quản lý dự án để tập trung xử lý lỗi.  
  • Tạo nhóm khẩn cấp chuyên trách lỗi bảo mật với các thành viên có kinh nghiệm.
* Ưu tiên công việc:  
  • Tạm dừng hoặc hoãn các công việc không khẩn cấp để tập trung hoàn toàn vào việc sửa lỗi bảo mật.  
  • Thiết lập kênh giao tiếp nội bộ để cập nhật tiến độ và phối hợp nhanh chóng giữa các bộ phận.

**4. Triển khai bản vá lỗi:**

* Phát triển bản vá:  
  • Lập tức sửa chữa mã nguồn bị lỗi, đồng thời đảm bảo không gây ra lỗi mới trong quá trình vá.  
  • Xây dựng bản vá trong môi trường kiểm thử riêng (staging environment) để đảm bảo tính ổn định.
* Kiểm thử bản vá:  
  • Sử dụng các công cụ kiểm thử bảo mật (penetration testing) để xác minh bản vá đã khắc phục lỗi.  
  • Đảm bảo có kế hoạch dự phòng (rollback plan) nếu bản vá gặp sự cố khi triển khai.
* Triển khai bản vá trên hệ thống sản xuất:  
  • Thực hiện triển khai vào giờ thấp điểm nếu có thể, nhằm giảm thiểu ảnh hưởng đến người dùng.  
  • Giám sát chặt chẽ hệ thống ngay sau khi cập nhật để phát hiện sớm bất kỳ vấn đề phát sinh nào.

**5. Kiểm thử và xác minh:**

* Kiểm thử toàn bộ hệ thống:  
  • Thực hiện kiểm thử lại toàn diện sau khi triển khai bản vá lỗi, bao gồm cả kiểm thử bảo mật tự động và kiểm thử thủ công.  
  • Đánh giá hiệu quả của bản vá bằng cách mô phỏng các cuộc tấn công tương tự.
* Xác minh hoạt động:  
  • Kiểm tra lại các chức năng quan trọng của hệ thống để đảm bảo không có lỗi mới phát sinh.  
  • Sử dụng các chỉ số giám sát bảo mật để đảm bảo hệ thống trở lại trạng thái an toàn.

**6. Báo cáo và tư vấn:**

* Báo cáo chi tiết:  
  • Soạn thảo báo cáo toàn diện cho khách hàng, nêu rõ nguyên nhân, các bước đã thực hiện và kết quả kiểm tra sau khi vá lỗi.  
  • Cung cấp thông tin về tác động của lỗi và các biện pháp khắc phục đã được triển khai.
* Đề xuất cải tiến:  
  • Tư vấn khách hàng về các biện pháp tăng cường bảo mật, chẳng hạn như cập nhật hệ thống giám sát bảo mật, sử dụng các công cụ bảo mật tiên tiến, và thực hiện kiểm thử bảo mật định kỳ.  
  • Đề xuất tổ chức buổi đào tạo bổ sung cho đội ngũ quản lý và phát triển về bảo mật nhằm ngăn ngừa lỗi tương tự trong tương lai.
* Phản hồi và theo dõi:  
  • Lên kế hoạch theo dõi sát sao hệ thống trong một khoảng thời gian nhất định sau khi triển khai bản vá.  
  • Thiết lập các kênh giao tiếp thường xuyên để cập nhật và hỗ trợ khách hàng nếu có yêu cầu bổ sung.

## 

## Câu 4.10: 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?

**1. Phân tích yêu cầu mới:**

* **Liên lạc với khách hàng:** • Tổ chức cuộc họp trực tiếp hoặc họp trực tuyến với khách hàng để hiểu rõ các thay đổi trong quy trình sản xuất và yêu cầu cụ thể cần cập nhật trên phần mềm.
* **Lập tài liệu yêu cầu:** • Ghi chép chi tiết các yêu cầu mới, xác định phạm vi công việc cần thực hiện và làm rõ các điểm cần sửa đổi.
* **Đánh giá tác động:** • Phân tích rủi ro và xác định các module, chức năng nào của phần mềm bị ảnh hưởng bởi những thay đổi này.

**2. Lập kế hoạch và lên lịch:**

* **Lập kế hoạch phát triển:** • Xác định các giai đoạn phát triển, kiểm thử và triển khai cho các thay đổi cần thiết.  
   • Đưa ra kế hoạch chi tiết (roadmap) với các mốc thời gian rõ ràng.
* **Thống nhất lịch trình:** • Hợp tác với khách hàng để chọn thời gian triển khai (chẳng hạn như giờ thấp điểm hoặc trong khung bảo trì đã thống nhất) nhằm đảm bảo không làm gián đoạn hoạt động sản xuất.
* **Lập kế hoạch dự phòng:** • Xác định các kế hoạch dự phòng (rollback plan) để khôi phục hệ thống nếu có sự cố xảy ra trong quá trình triển khai.

**3. Phát triển và kiểm thử:**

* **Phát triển các thay đổi:** • Thực hiện sửa đổi mã nguồn dựa trên yêu cầu mới, đảm bảo tuân thủ các tiêu chuẩn về chất lượng và bảo mật.
* **Kiểm thử toàn diện:** • Triển khai bản vá lỗi trên môi trường kiểm thử (staging environment) để kiểm tra tính ổn định và khả năng tương thích với các phần khác của hệ thống.  
   • Sử dụng các công cụ tự động và kiểm thử thủ công để đảm bảo không có lỗi mới phát sinh.
* **Đảm bảo sự ổn định:** • Xác nhận rằng phần mềm mới hoạt động tốt, đáp ứng đúng yêu cầu mà không làm ảnh hưởng đến các chức năng cũ.

**4. Triển khai dần dần:**

* **Triển khai theo từng bước:** • Thực hiện việc triển khai dần dần (rolling update) hoặc song song (blue-green deployment) để giảm thiểu rủi ro và đảm bảo rằng hệ thống cũ vẫn hoạt động bình thường trong quá trình cập nhật.
* **Hướng dẫn và đào tạo:** • Cung cấp hướng dẫn sử dụng và đào tạo cho người dùng cuối để họ làm quen với các thay đổi trong quy trình mới.
* **Theo dõi phản hồi ngay lập tức:** • Thu thập phản hồi từ người dùng sau mỗi giai đoạn triển khai để kịp thời điều chỉnh nếu cần.

**5. Giám sát và hỗ trợ:**

* **Theo dõi hoạt động:** • Giám sát chặt chẽ hệ thống sau khi triển khai để phát hiện sớm các vấn đề tiềm ẩn.  
   • Sử dụng các công cụ giám sát và báo cáo để kiểm soát tình trạng hệ thống.
* **Cung cấp hỗ trợ kỹ thuật:** • Đảm bảo đội ngũ hỗ trợ kỹ thuật luôn sẵn sàng giải quyết các thắc mắc và sự cố phát sinh từ người dùng.
* **Rút ra bài học kinh nghiệm:** •Sau khi toàn bộ quá trình cập nhật hoàn tất và hệ thống hoạt động ổn định, tổ chức một cuộc họp tổng kết để đánh giá quá trình triển khai. Qua đó, ghi nhận những điểm mạnh, điểm yếu và những khó khăn gặp phải, từ đó cập nhật quy trình kiểm thử và triển khai cho các phiên bản tiếp theo nhằm phòng ngừa sự cố tương tự.