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

|  |  |
| --- | --- |
| **Giáo viên hướng dẫn:** | **Gv. Châu Văn Vân** |
| **Nhóm sinh viên thực hiện:** | **5** |
| **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, 3/2025***

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 3 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 3/3/2025.  Thực hiện:  Từ ngày 3/3.  Đến ngày 5/3.  Tổng hợp câu 3:  Ngày 6/3/2025  \*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 | Nguyễn Duyên Duy | Thực hiện:   * Câu 1.5, 1.6 * Câu 2.5, 2.6 * Câu 3 * Câu 4.5, 4.6 |
| 4 | Hồ Xuân Thiện | Thực hiện:   * Câu 1.7, 1.8 * Câu 2.7, 2.8 * Câu 3 * Câu 4.7, 4.8 |
| 5 | Lê Anh Dũng | Thực hiện:   * Câu 1.9, 1.10 * Câu 2.9, 2.10 * Câu 3 * Câu 4.9, 4.10 |

**MỤC LỤC**

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

[PHẦN 1: CÂU HỎI TRẮC NGHIỆM 5](#_heading=h.1fob9te)

[Câu 1.1: Mục tiêu chính của kiểm thử phần mềm là gì? 6](#_heading=h.3znysh7)

[Câu 1.2: Nhóm nào chịu trách nhiệm đảm bảo chất lượng phần mềm trong dự án? 6](#_heading=h.2et92p0)

[Câu 1.3: Kiểm thử các sản phẩm phi thực thi bao gồm hoạt động nào dưới đây? 6](#_heading=h.tyjcwt)

[Câu 1.4: Lập kế hoạch trong tiến trình phát triển phần mềm nhằm mục đích gì? 6](#_heading=h.3dy6vkm)

[Câu 1.5:Loại kiểm thử nào sau đây tập trung vào việc phát hiện các lỗi chức năng của phần mềm? 6](#_heading=h.1t3h5sf)

[Câu 1.6:Quản lý phiên bản tài liệu có mục tiêu gì? 6](#_heading=h.4d34og8)

[Câu 1.7:Nhóm SQA có vai trò gì trong kiểm thử phần mềm? 6](#_heading=h.2s8eyo1)

[Câu 1.8:Loại kiểm thử nào thường được thực hiện cuối cùng trước khi phần mềm được bàn giao cho khách hàng? 7](#_heading=h.17dp8vu)

[Câu 1.9: Đâu là một trong những công cụ phổ biến dùng để quản lý phiên bản tài liệu? 7](#_heading=h.3rdcrjn)

[Câu 1.10:Hoạt động lập tài liệu trong mỗi pha phát triển phần mềm nhằm mục đích gì? 7](#_heading=h.26in1rg)

[PHẦN 2: CÂU TRẢ LỜI NGẮN 7](#_heading=h.lnxbz9)

[Câu 2.1: Nhóm SQA là gì và vai trò của nhóm này trong phát triển phần mềm? 7](#_heading=h.35nkun2)

[Câu 2.2: Kiểm thử đơn vị là gì? 8](#_heading=h.1ksv4uv)

[Câu 2.3: Mục tiêu chính của kiểm thử chấp nhận là gì? 9](#_heading=h.44sinio)

[Câu 2.4: Các hoạt động chính trong kiểm thử sản phẩm phi thực thi là gì? 9](#_heading=h.2jxsxqh)

[Câu 2.5:Tại sao việc lập tài liệu cho mỗi pha phát triển phần mềm lại quan trọng? 9](#_heading=h.z337ya)

[Câu 2.6:Quản lý phiên bản tài liệu là gì? 10](#_heading=h.3j2qqm3)

[Câu 2.7:Các loại kiểm thử chính trong kiểm thử sản phẩm thực thi là gì? 11](#_heading=h.1y810tw)

[Câu 2.8:Kiểm thử tích hợp là gì? 12](#_heading=h.4i7ojhp)

[Câu 2.9:Hoạt động lập kế hoạch cho các pha phát triển phần mềm bao gồm những gì? 14](#_heading=h.2xcytpi)

[Câu 2.10:Làm tài liệu kiểm thử bao gồm những gì? 15](#_heading=h.1ci93xb)

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

[Câu 3.1: Vai trò của nhóm SQA trong việc đảm bảo chất lượng phần mềm là gì? 16](#_heading=h.2bn6wsx)

[Câu 3.2: Thảo luận về sự khác nhau giữa kiểm thử đơn vị và kiểm thử tích hợp. 16](#_heading=h.qsh70q)

[Câu 3.3: Tại sao việc lập tài liệu kiểm thử lại quan trọng trong mỗi dự án phần mềm? 17](#_heading=h.3as4poj)

[Câu 3.4: Thảo luận về các thách thức khi lập kế hoạch cho các pha phát triển phần mềm. 17](#_heading=h.1pxezwc)

[Câu 3.5:Quản lý phiên bản tài liệu có ảnh hưởng như thế nào đến quá trình bảo trì phần mềm? 17](#_heading=h.49x2ik5)

[Câu 3.6:So sánh giữa kiểm thử sản phẩm phi thực thi và kiểm thử sản phẩm thực thi. 19](#_heading=h.2p2csry)

[Câu 3.7:Thảo luận về cách cải thiện quy trình lập kế hoạch để giảm thiểu rủi ro trong dự án phần mềm. 21](#_heading=h.147n2zr)

[Câu 3.8:Đề xuất các công cụ hỗ trợ việc lập tài liệu kiểm thử và quản lý phiên bản. 23](#_heading=h.3o7alnk)

[Câu 3.9:Tại sao kiểm thử chấp nhận lại là một giai đoạn quan trọng trong phát triển phần mềm? 26](#_heading=h.23ckvvd)

[Câu 3.10:Thảo luận về các phương pháp hiệu quả để quản lý chất lượng phần mềm trong các dự án lớn. 27](#_heading=h.ihv636)

[PHẦN 4: CÂU HỎI TÌNH HUỐNG 28](#_heading=h.32hioqz)

[Câu 4.1: Một dự án phần mềm đã hoàn thành và sắp bàn giao cho khách hàng, nhưng khách hàng yêu cầu kiểm tra lại toàn bộ tài liệu yêu cầu và thiết kế. Đội phát triển nên xử lý thế nào? 28](#_heading=h.1hmsyys)

[Câu 4.2: Trong quá trình kiểm thử hệ thống, nhóm phát triển phát hiện một lỗi nghiêm trọng nhưng thời hạn bàn giao đang đến gần. Bạn sẽ xử lý tình huống này như thế nào? 29](#_heading=h.41mghml)

[Câu 4.3: Một nhóm phát triển gặp khó khăn trong việc quản lý phiên bản tài liệu do tài liệu liên tục thay đổi. Hãy đề xuất giải pháp. 29](#_heading=h.2grqrue)

[Câu 4.4: Dự án phát triển phần mềm gặp vấn đề khi khách hàng yêu cầu thay đổi lớn trong pha cài đặt. Đội phát triển nên xử lý thế nào? 30](#_heading=h.vx1227)

[Câu 4.5:Nhóm kiểm thử phát hiện nhiều lỗi chức năng trong phần mềm. Tuy nhiên, nhóm phát triển lại cho rằng đây không phải lỗi mà là tính năng. Là trưởng dự án, bạn sẽ làm gì? 30](#_heading=h.3fwokq0)

[Câu 4.6:Khách hàng yêu cầu bổ sung một tính năng mới khi phần mềm đã hoàn thành pha kiểm thử tích hợp. Đội phát triển nên làm gì? 31](#_heading=h.1v1yuxt)

[Câu 4.7:Một công ty phát triển phần mềm nhỏ muốn xây dựng nhóm SQA nhưng gặp khó khăn về ngân sách. Hãy đề xuất giải pháp. 32](#_heading=h.4f1mdlm)

[Câu 4.8:Trong quá trình làm tài liệu kiểm thử, nhóm phát triển không thống nhất được về nội dung cần đưa vào tài liệu. Là trưởng nhóm, bạn sẽ giải quyết vấn đề này như thế nào? 35](#_heading=h.2u6wntf)

[Câu 4.9:Dự án phát triển phần mềm cho một ngân hàng yêu cầu bảo mật cao. Đề xuất cách lập kế hoạch kiểm thử cho dự án này. 36](#_heading=h.19c6y18)

[Câu 4.10:Sau khi triển khai phần mềm, khách hàng phát hiện ra một số lỗi bảo mật nghiêm trọng. Đội phát triển cần xử lý ra sao để khắc phục vấn đề và lấy lại niềm tin từ khách hàng? 39](#_heading=h.3tbugp1)

# PHẦN 1: CÂU HỎI TRẮC NGHIỆM

## Câu 1.1: Mục tiêu chính của kiểm thử phần mềm là gì?

A. Đảm bảo phần mềm không chứa lỗi

B. Đảm bảo phần mềm đáp ứng đúng yêu cầu

C. Hoàn thành dự án đúng thời gian

D. Phát hiện tất cả các lỗi bảo mật

Đáp án A

## Câu 1.2: Nhóm nào chịu trách nhiệm đảm bảo chất lượng phần mềm trong dự án?

A. Nhóm kiểm thử tự động

B. Nhóm bảo trì

C. Nhóm SQA

D. Nhóm thiết kế

Đáp án C

## Câu 1.3: Kiểm thử các sản phẩm phi thực thi bao gồm hoạt động nào dưới đây?

A. Kiểm thử đơn vị

**B. Walkthrough và review tài liệu**

C. Kiểm thử hệ thống

D. Kiểm thử hiệu suất

## Câu 1.4: Lập kế hoạch trong tiến trình phát triển phần mềm nhằm mục đích gì?

**A. Xác định phạm vi công việc và ước tính thời gian hoàn thành**

B. Giảm thiểu chi phí phát triển

C. Phát hiện lỗi sớm trong quá trình phát triển

D. Xác định các công cụ kiểm thử tự động

## Câu 1.5:Loại kiểm thử nào sau đây tập trung vào việc phát hiện các lỗi chức năng của phần mềm?

A. Kiểm thử hiệu suất

**B. Kiểm thử chức năng**

C. Kiểm thử bảo mật

D. Kiểm thử tích hợp

Đáp án: B

## Câu 1.6:Quản lý phiên bản tài liệu có mục tiêu gì?

**A. Đảm bảo tài liệu luôn được cập nhật và có thể truy xuất phiên bản cũ khi cần**

B. Giảm thiểu số lượng tài liệu cần làm

C. Tăng tốc độ phát triển phần mềm

D. Tự động hóa việc kiểm thử tài liệu

Đáp án:

## Câu 1.7:Nhóm SQA có vai trò gì trong kiểm thử phần mềm?

A. Viết mã nguồn cho phần mềm

**B. Đánh giá và đảm bảo quy trình phát triển phần mềm tuân thủ tiêu chuẩn chất lượn**g

C. Triển khai phần mềm lên môi trường thực tế

D. Giám sát hoạt động của hệ thống sau khi triển khai

## 

## Câu 1.8:Loại kiểm thử nào thường được thực hiện cuối cùng trước khi phần mềm được bàn giao cho khách hàng?

A. Kiểm thử đơn vị

B. Kiểm thử hệ thống

C. Kiểm thử tích hợp

**D. Kiểm thử chấp nhận**

## 

## Câu 1.9: Đâu là một trong những công cụ phổ biến dùng để quản lý phiên bản tài liệu?

A. Selenium

**B. Git**

C. Postman

D. JIRA

Đáp Án : B

## Câu 1.10:Hoạt động lập tài liệu trong mỗi pha phát triển phần mềm nhằm mục đích gì?

A. Giảm thời gian phát triển phần mềm

**B. Hỗ trợ quá trình bảo trì và nâng cấp phần mềm sau khi triển khai**

C. Tăng cường bảo mật cho phần mềm

D. Tự động hóa việc phát triển phần mềm

Đáp Án : B

## 

# PHẦN 2: CÂU TRẢ LỜI NGẮN

## Câu 2.1: Nhóm SQA là gì và vai trò của nhóm này trong phát triển phần mềm?

SQA là nhóm chuyên trách đảm bảo chất lượng phần mềm, được thành lập để giám sát, đánh giá và cải thiện quy trình phát triển phần mềm toàn diện.

Vai Trò Chính

1. Quản Lý Chất Lượng

* Xây dựng và duy trì các tiêu chuẩn chất lượng
* Định nghĩa quy trình phát triển tiêu chuẩn (Standard Development Process)
* Đảm bảo tuân thủ các quy trình được thiết lập

2. Kiểm Tra và Đánh Giá

* Thực hiện kiểm tra độc lập
* Đánh giá sản phẩm và quy trình phát triển
* Phát hiện và báo cáo các vấn đề tiềm ẩn

3. Phòng Ngừa Rủi Ro

* Phân tích và quản lý rủi ro
* Đề xuất các biện pháp cải thiện
* Ngăn chặn các lỗi tiềm ẩn trước khi chúng phát sinh

4. Hỗ Trợ và Đào Tạo

* Huấn luyện nhân viên về các tiêu chuẩn chất lượng
* Tư vấn cho các nhóm phát triển
* Chia sẻ kiến thức và thực tiễn tốt nhất

Hoạt Động Cụ Thể

* Kiểm tra tài liệu yêu cầu
* Đánh giá thiết kế hệ thống
* Giám sát quy trình phát triển
* Thực hiện kiểm thử chấp nhận
* Theo dõi và báo cáo chất lượng sản phẩm

## Câu 2.2: Kiểm thử đơn vị là gì?

Kiểm thử đơn vị là quá trình kiểm tra các thành phần nhỏ nhất của phần mềm (thường là các hàm hoặc phương thức) để đảm bảo chúng hoạt động đúng theo yêu cầu.

Đặc Điểm Chính

* Thực hiện bởi các lập trình viên
* Kiểm tra từng module/thành phần độc lập
* Diễn ra ngay trong quá trình phát triển
* Sử dụng các framework kiểm thử chuyên dụng

Mục Tiêu

* Xác minh logic của từng chức năng
* Phát hiện và sửa lỗi ở mức code
* Đảm bảo độ tin cậy của từng thành phần
* Megồn dễ dàng cho các kiểm thử sau này

Phương Pháp Thực Hiện

1. Viết kịch bản kiểm thử
2. Chuẩn bị dữ liệu kiểm thử
3. Thực thi các test case
4. So sánh kết quả thực tế với kết quả mong đợi
5. Ghi nhận và sửa lỗi

## Câu 2.3: Mục tiêu chính của kiểm thử chấp nhận là gì?

Mục đích của Kiểm thử chấp nhận của người dùng (UAT) là xác nhận và đảm bảo rằng ứng dụng phần mềm đáp ứng các nhu cầu và yêu cầu cụ thể của người dùng dự kiến ​​trước khi phát hành chính thức, đảm bảo chức năng, khả năng sử dụng và phù hợp với các tình huống thực tế ở mức cao.

## Câu 2.4: Các hoạt động chính trong kiểm thử sản phẩm phi thực thi là gì?

* Walkthrough: Các thành viên nhóm phát triển cùng nhau xem xét tài liệu để phát hiện lỗi.
* Inspection: Một nhóm chuyên gia độc lập thực hiện đánh giá tài liệu.
* Review: Người quản lý tổ chức buổi họp để thảo luận và rà soát các tài liệu đã hoàn thành.

## Câu 2.5:Tại sao việc lập tài liệu cho mỗi pha phát triển phần mềm lại quan trọng?

Việc lập tài liệu cho mỗi pha phát triển phần mềm là rất quan trọng vì những lý do sau đây:

1. Tính minh bạch và truyền thông hiệu quả

* Tài liệu giúp tất cả các bên liên quan (nhà phát triển, quản lý dự án, khách hàng, v.v.) hiểu rõ về yêu cầu, tiến độ và kết quả của từng pha.
* Nó đảm bảo rằng mọi người đều có cùng một nguồn thông tin chính xác, giảm thiểu sự hiểu lầm hoặc mâu thuẫn.

2. Theo dõi tiến độ và quản lý dự án

* Tài liệu giúp theo dõi tiến độ của dự án, đảm bảo rằng mỗi pha được hoàn thành đúng thời hạn và đáp ứng các yêu cầu đã định.
* Nó cung cấp cơ sở để đánh giá rủi ro, quản lý thay đổi và điều chỉnh kế hoạch khi cần thiết.

3. Đảm bảo chất lượng

* Tài liệu giúp xác định và kiểm tra các tiêu chí chất lượng ở mỗi pha, đảm bảo rằng sản phẩm cuối cùng đáp ứng các yêu cầu kỹ thuật và nghiệp vụ.
* Nó cũng hỗ trợ việc kiểm thử và xác minh rằng các chức năng hoạt động đúng như mong đợi.

4. Hỗ trợ bảo trì và nâng cấp

* Tài liệu chi tiết giúp các nhà phát triển trong tương lai hiểu được cách thức hoạt động của hệ thống, từ đó dễ dàng bảo trì, sửa chữa hoặc nâng cấp phần mềm.
* Nó cũng giúp giảm thời gian và chi phí khi cần thay đổi hoặc cải tiến hệ thống.

5. Tuân thủ các tiêu chuẩn và quy định

* Trong nhiều ngành công nghiệp, việc lập tài liệu là bắt buộc để tuân thủ các tiêu chuẩn quản lý chất lượng (như ISO) hoặc các quy định pháp lý.
* Tài liệu cũng có thể được sử dụng làm bằng chứng trong trường hợp có tranh chấp hoặc kiểm toán.

6. Lưu trữ kiến thức và kinh nghiệm

* Tài liệu giúp lưu trữ kiến thức và kinh nghiệm từ dự án, tạo nền tảng cho các dự án tương lai.
* Nó giúp tránh việc mất thông tin quan trọng khi có sự thay đổi nhân sự hoặc kết thúc dự án.

7. Hỗ trợ đào tạo và chuyển giao công nghệ

* Tài liệu chi tiết giúp đào tạo nhân viên mới hoặc chuyển giao công nghệ cho các bên khác một cách hiệu quả.
* Nó cung cấp một hướng dẫn rõ ràng về cách thức hoạt động và quy trình phát triển phần mềm.

## Câu 2.6:Quản lý phiên bản tài liệu là gì?

**Quản lý phiên bản tài liệu (Document Version Control)** là quy trình quản lý và theo dõi các thay đổi của tài liệu theo thời gian. Nó giúp đảm bảo rằng các phiên bản khác nhau của tài liệu được lưu trữ một cách có hệ thống, cho phép người dùng dễ dàng truy cập, so sánh và khôi phục các phiên bản cũ khi cần thiết.

**Mục tiêu chính của quản lý phiên bản tài liệu**:

1. **Theo dõi thay đổi**: Ghi lại lịch sử các thay đổi trong tài liệu, bao gồm ai đã thay đổi, thay đổi gì và khi nào.
2. **Đảm bảo tính nhất quán**: Tránh tình trạng nhiều phiên bản khác nhau của cùng một tài liệu tồn tại song song, gây nhầm lẫn.
3. **Khôi phục phiên bản cũ**: Cho phép người dùng quay lại các phiên bản trước đó nếu có sai sót hoặc cần tham khảo thông tin cũ.
4. **Hỗ trợ cộng tác**: Giúp nhiều người cùng làm việc trên một tài liệu mà không gây xung đột hoặc mất dữ liệu.

**Các thành phần chính của quản lý phiên bản tài liệu:**

1. **Phiên bản (Version)**: Mỗi lần tài liệu được chỉnh sửa và lưu lại sẽ tạo ra một phiên bản mới. Các phiên bản thường được đánh số (ví dụ: v1.0, v1.1, v2.0) hoặc gắn nhãn thời gian.
2. **Lịch sử thay đổi (Change History)**: Ghi lại chi tiết các thay đổi trong từng phiên bản, bao gồm người thực hiện, ngày giờ và mô tả thay đổi.
3. **Kiểm soát truy cập (Access Control)**: Quản lý quyền truy cập để đảm bảo chỉ những người có thẩm quyền mới có thể chỉnh sửa hoặc phê duyệt tài liệu.
4. **Hệ thống lưu trữ (Repository)**: Nơi lưu trữ tập trung các phiên bản tài liệu, thường được hỗ trợ bởi các công cụ quản lý phiên bản như Git, SVN, hoặc các hệ thống quản lý tài liệu chuyên dụng.

**Lợi ích của quản lý phiên bản tài liệu:**

* **Tăng tính minh bạch**: Mọi thay đổi đều được ghi lại và có thể truy xuất.
* **Giảm thiểu rủi ro**: Tránh mất dữ liệu hoặc nhầm lẫn giữa các phiên bản.
* **Hỗ trợ làm việc nhóm**: Giúp nhiều người cùng làm việc trên một tài liệu mà không gây xung đột.
* **Tuân thủ quy định**: Đáp ứng các yêu cầu về kiểm toán và quản lý chất lượng.

**Công cụ hỗ trợ quản lý phiên bản tài liệu:**

* **Git**: Phổ biến cho quản lý phiên bản mã nguồn, nhưng cũng có thể áp dụng cho tài liệu.
* **SVN (Subversion)**: Một hệ thống quản lý phiên bản tập trung.
* **Google Docs, Microsoft SharePoint**: Hỗ trợ quản lý phiên bản tài liệu trực tuyến.
* **Các hệ thống quản lý tài liệu chuyên dụng**: Như Confluence, Documentum, hoặc Alfresco.

## Câu 2.7:Các loại kiểm thử chính trong kiểm thử sản phẩm thực thi là gì?

Các loại kiểm thử chính trong kiểm thử sản phẩm thực thi

Kiểm thử sản phẩm thực thi (Product Execution Testing) tập trung vào việc **đánh giá cách phần mềm hoạt động trong môi trường thực tế**. Dưới đây là các loại kiểm thử chính:

1. Kiểm thử chức năng (Functional Testing)

* Mục đích: Đảm bảo phần mềm hoạt động đúng theo **yêu cầu chức năng**.
* Phương pháp: Kiểm tra các tính năng như đăng nhập, xử lý dữ liệu, tính toán, báo cáo,...
* Ví dụ: Kiểm tra xem hệ thống có lưu thông tin khách hàng đúng không.

2. Kiểm thử hiệu năng (Performance Testing)

* Mục đích: Đánh giá **tốc độ, độ ổn định và khả năng mở rộng** của hệ thống.
* Các loại kiểm thử hiệu năng:
  + **Load Testing:** Kiểm tra hệ thống hoạt động khi có nhiều người dùng truy cập.
  + **Stress Testing:** Đẩy hệ thống đến giới hạn để xem có bị sập không.
  + **Scalability Testing:** Kiểm tra khả năng mở rộng của hệ thống khi tăng tải.
* Ví dụ: Xem thử website có thể xử lý **10.000 người dùng đồng thời** không.

3. Kiểm thử bảo mật (Security Testing)

* Mục đích: Đánh giá mức độ **an toàn** của phần mềm trước các mối đe dọa.
* Các kỹ thuật phổ biến:
  + **Penetration Testing:** Giả lập hacker tấn công để tìm lỗ hổng.
  + **Authentication Testing:** Kiểm tra cơ chế đăng nhập, phân quyền.
  + **Data Encryption Testing:** Đảm bảo dữ liệu được mã hóa đúng.
* Ví dụ: Kiểm tra xem hệ thống có bị tấn công **SQL Injection** không.

4. Kiểm thử khả dụng (Usability Testing)

* Mục đích: Đánh giá **trải nghiệm người dùng (UX/UI)**.
* Các yếu tố kiểm tra:
  + Dễ sử dụng, điều hướng thuận tiện.
  + Giao diện trực quan, màu sắc, nút bấm hợp lý.
* Ví dụ: Kiểm tra xem người dùng có dễ dàng **đặt hàng trên trang web** không.

5. Kiểm thử tương thích (Compatibility Testing)

* Mục đích: Đảm bảo phần mềm chạy trên nhiều **thiết bị, hệ điều hành, trình duyệt khác nhau**.
* Các loại kiểm thử tương thích:
  + **Cross-browser Testing:** Kiểm tra trên Chrome, Firefox, Edge, Safari.
  + **Cross-platform Testing:** Kiểm tra trên Windows, macOS, Linux.
  + **Device Compatibility Testing:** Kiểm tra trên điện thoại, tablet.
* Ví dụ: Kiểm tra xem ứng dụng có chạy tốt trên **Android và iOS** không.

6. Kiểm thử hồi quy (Regression Testing)

* Mục đích: Đảm bảo các tính năng cũ **không bị lỗi** khi cập nhật phần mềm mới.
* Ví dụ: Sau khi thêm tính năng thanh toán **QR Code**, cần kiểm tra xem **các phương thức thanh toán cũ (Visa, Mastercard) có hoạt động bình thường không**.

## Câu 2.8:Kiểm thử tích hợp là gì?

Kiểm thử tích hợp (Integration Testing) là gì?

**Kiểm thử tích hợp (Integration Testing)** là giai đoạn kiểm thử trong đó **các module hoặc thành phần riêng lẻ** của phần mềm được kết hợp và kiểm tra **cách chúng hoạt động cùng nhau**.

Mục đích của kiểm thử tích hợp

🔹 Xác minh rằng **các module liên kết đúng với nhau**.  
🔹 Kiểm tra **giao tiếp giữa các thành phần** để phát hiện lỗi truyền dữ liệu.  
🔹 Đảm bảo phần mềm hoạt động trơn tru **trong môi trường thực tế**.  
🔹 Phát hiện sớm lỗi về **giao tiếp API, gọi hàm, truyền dữ liệu** giữa các module.

Các chiến lược kiểm thử tích hợp phổ biến

🚀 Có 4 chiến lược chính khi thực hiện kiểm thử tích hợp:

1. Kiểm thử từ trên xuống (Top-Down Testing)

* Bắt đầu kiểm thử từ các **module cấp cao** trước, sau đó dần dần tích hợp các **module cấp thấp**.
* Thường sử dụng **stubs (mô phỏng các module chưa có)** để thay thế các module cấp dưới chưa hoàn thành.
* **Ưu điểm:** Dễ xác định lỗi ở cấp cao, giúp kiểm tra nhanh luồng nghiệp vụ chính.
* **Nhược điểm:** Các module cấp thấp có thể chưa được kiểm thử kỹ ngay từ đầu.

2. Kiểm thử từ dưới lên (Bottom-Up Testing)

* Kiểm thử **các module cấp thấp trước**, sau đó kết hợp dần với các module cấp cao.
* Sử dụng **drivers (mô phỏng module cấp trên)** để thay thế phần chưa có.
* **Ưu điểm:** Kiểm thử tốt các module nền tảng trước khi tích hợp vào hệ thống lớn.
* **Nhược điểm:** Có thể mất thời gian để kiểm tra luồng nghiệp vụ tổng thể.

3. Kiểm thử tích hợp sandwich (Hybrid / Sandwich Testing)

* Kết hợp cả **Top-Down** và **Bottom-Up**, kiểm thử cả **module cấp cao lẫn cấp thấp song song**.
* Thích hợp cho các hệ thống lớn có nhiều tầng (layer).
* **Ưu điểm:** Cân bằng giữa kiểm thử luồng nghiệp vụ tổng thể và từng module nhỏ.
* **Nhược điểm:** Khó tổ chức kiểm thử, cần nhiều tài nguyên.

4. Kiểm thử Big Bang (Big Bang Integration Testing)

* Tích hợp **tất cả module** cùng một lúc và kiểm tra toàn bộ hệ thống.
* **Ưu điểm:** Tiết kiệm thời gian chuẩn bị.
* **Nhược điểm:** Nếu có lỗi, rất khó tìm ra nguyên nhân vì tất cả module được tích hợp cùng lúc.

Ví dụ về kiểm thử tích hợp

👉 **Hệ thống website thương mại điện tử** gồm các module:

1. **Đăng nhập**
2. **Quản lý giỏ hàng**
3. **Thanh toán**
4. **Gửi email xác nhận đơn hàng**

🔹 **Nếu chỉ kiểm thử từng module riêng lẻ (Unit Test), ta không biết chúng có hoạt động tốt khi kết hợp hay không.**🔹 **Kiểm thử tích hợp sẽ kiểm tra luồng sau:**✔️ Người dùng đăng nhập → Thêm sản phẩm vào giỏ hàng → Thanh toán → Nhận email xác nhận.

**⛔ Nếu dữ liệu giỏ hàng không được truyền đúng đến module thanh toán, hệ thống sẽ bị lỗi.**

Khi nào nên thực hiện kiểm thử tích hợp?

✅ Sau khi **các module đơn lẻ đã được kiểm thử (Unit Test)**.  
✅ Khi các thành phần **cần giao tiếp với nhau**, đặc biệt là API hoặc database.  
✅ Khi muốn đảm bảo **dữ liệu truyền giữa các module là chính xác**.

## Câu 2.9:Hoạt động lập kế hoạch cho các pha phát triển phần mềm bao gồm những gì?

**Hoạt động lập kế hoạch cho các pha phát triển phần mềm bao gồm:**

1. **Xác định phạm vi và mục tiêu**
   * Xác định yêu cầu của phần mềm.
   * Xác định mục tiêu chính và kết quả mong muốn.
2. **Lập kế hoạch tiến độ**
   * Chia nhỏ dự án thành các giai đoạn/phases.
   * Thiết lập mốc thời gian (timeline) cho từng pha.
3. **Phân công nhiệm vụ**
   * Xác định vai trò của các thành viên trong nhóm.
   * Giao nhiệm vụ cụ thể cho từng cá nhân/nhóm.
4. **Lựa chọn công nghệ và công cụ**
   * Xác định ngôn ngữ lập trình, framework, công cụ hỗ trợ phát triển.
5. **Quản lý rủi ro**
   * Dự đoán các vấn đề có thể xảy ra.
   * Đề xuất phương án xử lý rủi ro.
6. **Dự toán nguồn lực**
   * Xác định nhân lực, thời gian, ngân sách cần thiết.
7. **Lập kế hoạch kiểm thử và đánh giá**
   * Xác định chiến lược kiểm thử phần mềm.
   * Định nghĩa tiêu chí đánh giá chất lượng.

Việc lập kế hoạch cẩn thận giúp đảm bảo dự án phát triển phần mềm diễn ra suôn sẻ và đạt được mục tiêu đề ra.

## Câu 2.10:Làm tài liệu kiểm thử bao gồm những gì?

Tài liệu kiểm thử phần mềm bao gồm các thành phần sau:

1. **Kế hoạch kiểm thử (Test Plan)**
   * Mục tiêu kiểm thử
   * Phạm vi kiểm thử
   * Chiến lược kiểm thử
   * Công cụ kiểm thử được sử dụng
   * Rủi ro và cách giảm thiểu
2. **Kịch bản kiểm thử (Test Scenario)**
   * Các tình huống kiểm thử chung dựa trên yêu cầu phần mềm
3. **Trường hợp kiểm thử (Test Case)**
   * ID của Test Case
   * Mô tả Test Case
   * Bước thực hiện
   * Dữ liệu đầu vào
   * Kết quả mong đợi
   * Trạng thái (Pass/Fail)
4. **Báo cáo kiểm thử (Test Report)**
   * Tổng hợp kết quả kiểm thử
   * Các lỗi phát hiện được
   * Đề xuất hướng khắc phục
5. **Tài liệu lỗi (Bug Report)**
   * Mô tả lỗi
   * Mức độ nghiêm trọng (Severity)
   * Bước tái hiện lỗi
   * Trạng thái xử lý lỗi

Tài liệu kiểm thử giúp đảm bảo chất lượng phần mềm và hỗ trợ quá trình bảo trì, nâng cấp sau này.

# PHẦN 3: CÂU HỎI THẢO LUẬN NHÓM

## Câu 3.1: Vai trò của nhóm SQA trong việc đảm bảo chất lượng phần mềm là gì?

Các Vai Trò Chính Của SQA

1. Xây Dựng Tiêu Chuẩn Chất Lượng

* Định nghĩa các tiêu chí chất lượng chi tiết
* Thiết lập quy trình phát triển chuẩn
* Xác định các điều kiện nghiệm thu sản phẩm

2. Giám Sát Quy Trình Phát Triển

* Theo dõi tuân thủ quy trình phát triển
* Kiểm tra tính nhất quán trong các giai đoạn
* Đảm bảo mọi hoạt động đáp ứng các tiêu chuẩn đã định

3. Phân Tích và Quản Lý Rủi Ro

* Nhận diện các rủi ro tiềm ẩn trong dự án
* Đánh giá tác động của các rủi ro
* Đề xuất biện pháp phòng ngừa và giảm thiểu rủi ro

4. Đảm Bảo Tuân Thủ

* Kiểm tra việc tuân thủ các yêu cầu kỹ thuật
* Xác minh sự phù hợp với tiêu chuẩn ngành
* Đảm bảo tính pháp lý và an toàn của sản phẩm

5. Cải Tiến Liên Tục

* Thu thập và phân tích dữ liệu về chất lượng
* Đề xuất các biện pháp cải thiện quy trình
* Hỗ trợ đào tạo và nâng cao năng lực đội ngũ

## Câu 3.2: Thảo luận về sự khác nhau giữa kiểm thử đơn vị và kiểm thử tích hợp.

|  |  |  |
| --- | --- | --- |
| **Tiêu Chí** | **Kiểm Thử Đơn Vị** | **Kiểm Thử Tích Hợp** |
| Phạm Vi | Từng thành phần nhỏ | Toàn bộ hệ thống |
| Người Thực Hiện | Lập trình viên | Nhóm kiểm thử |
| Mục Tiêu | Kiểm tra logic | Kiểm tra tương tác |
| Thời Điểm | Trong quá trình phát triển | Sau khi hoàn thiện module |
| Độ Phức Tạp | Đơn giản | Phức tạp |

## Câu 3.3: Tại sao việc lập tài liệu kiểm thử lại quan trọng trong mỗi dự án phần mềm?

Việc lập tài liệu kiểm thử (test documentation) quan trọng trong mỗi dự án phần mềm vì:

* Đảm bảo chất lượng: Giúp xác định rõ các trường hợp kiểm thử (test cases), đảm bảo phần mềm đáp ứng yêu cầu và hoạt động đúng.
* Minh bạch: Cung cấp bằng chứng về quá trình kiểm tra, dễ theo dõi và báo cáo cho các bên liên quan.
* Hỗ trợ bảo trì: Tài liệu là tham chiếu để kiểm tra lại khi có cập nhật hoặc lỗi phát sinh sau này.
* Giảm rủi ro: Phát hiện sớm vấn đề, tránh lỗi nghiêm trọng khi triển khai.
* Hợp tác: Tạo cơ sở chung để đội phát triển, kiểm thử và khách hàng hiểu rõ phạm vi kiểm tra.

## Câu 3.4: Thảo luận về các thách thức khi lập kế hoạch cho các pha phát triển phần mềm.

Lập kế hoạch cho các pha phát triển phần mềm đối mặt với nhiều thách thức sau:

* Yêu cầu thay đổi: Khách hàng thường thay đổi yêu cầu giữa chừng, làm xáo trộn phạm vi và thời gian của các pha.
* Ước lượng không chính xác: Khó dự đoán thời gian và nguồn lực cần thiết do phức tạp kỹ thuật hoặc thiếu thông tin ban đầu.
* Phụ thuộc lẫn nhau: Các pha (thiết kế, phát triển, kiểm thử) thường phụ thuộc vào nhau, nếu một pha chậm trễ sẽ ảnh hưởng toàn bộ kế hoạch.
* Thiếu giao tiếp: Sự không rõ ràng giữa đội phát triển, khách hàng và các bên liên quan dẫn đến hiểu sai mục tiêu.
* Rủi ro kỹ thuật: Công nghệ mới hoặc lỗi không lường trước có thể làm gián đoạn tiến độ.
* Hạn chế tài nguyên: Thiếu nhân lực, ngân sách hoặc công cụ phù hợp làm khó phân bổ hiệu quả.

## Câu 3.5:Quản lý phiên bản tài liệu có ảnh hưởng như thế nào đến quá trình bảo trì phần mềm?

Quản lý phiên bản tài liệu có ảnh hưởng đáng kể đến quá trình bảo trì phần mềm, cụ thể như sau:

**1. Dễ dàng truy xuất thông tin lịch sử**

* Quản lý phiên bản tài liệu giúp lưu trữ và theo dõi tất cả các thay đổi trong tài liệu, bao gồm các yêu cầu, thiết kế, và hướng dẫn kỹ thuật. Điều này giúp nhóm bảo trì dễ dàng truy xuất lại thông tin lịch sử để hiểu rõ lý do và bối cảnh của các thay đổi trước đó.
* Ví dụ: Khi gặp lỗi, nhóm bảo trì có thể xem lại các phiên bản cũ để xác định khi nào lỗi xuất hiện và nguyên nhân gây ra lỗi.

**2. Giảm thiểu thời gian và công sức**

* Việc quản lý phiên bản hiệu quả giúp nhóm bảo trì nhanh chóng tìm thấy các tài liệu liên quan mà không cần mất thời gian tìm kiếm hoặc tái tạo lại thông tin đã mất.
* Điều này đặc biệt hữu ích khi nhóm bảo trì không phải là nhóm phát triển ban đầu, vì họ có thể dựa vào tài liệu để hiểu hệ thống một cách nhanh chóng.

**3. Đảm bảo tính nhất quán**

* Quản lý phiên bản giúp đảm bảo rằng tất cả các thành viên trong nhóm bảo trì đều làm việc trên cùng một phiên bản mới nhất của tài liệu, tránh tình trạng sử dụng thông tin lỗi thời hoặc không chính xác.
* Điều này giúp tránh các sai sót trong quá trình bảo trì, chẳng hạn như sửa đổi nhầm các phần không cần thiết hoặc bỏ qua các yêu cầu quan trọng.

**4. Hỗ trợ khôi phục và phục hồi**

* Trong trường hợp cần khôi phục lại một phiên bản cũ của phần mềm do lỗi nghiêm trọng hoặc yêu cầu từ khách hàng, quản lý phiên bản tài liệu giúp nhóm bảo trì dễ dàng tìm lại các tài liệu tương ứng với phiên bản đó.
* Điều này đảm bảo rằng việc khôi phục được thực hiện một cách chính xác và hiệu quả.

**5. Cải thiện chất lượng bảo trì**

* Với tài liệu được quản lý chặt chẽ, nhóm bảo trì có thể hiểu rõ hơn về kiến trúc hệ thống, các quyết định thiết kế và các yêu cầu nghiệp vụ. Điều này giúp họ đưa ra các giải pháp bảo trì chất lượng cao, đảm bảo rằng các thay đổi không gây ra lỗi mới hoặc ảnh hưởng tiêu cực đến hệ thống.

**6. Hỗ trợ đào tạo và chuyển giao kiến thức**

* Quản lý phiên bản tài liệu giúp lưu trữ kiến thức và kinh nghiệm từ nhóm phát triển ban đầu, tạo điều kiện thuận lợi cho việc đào tạo nhân viên mới hoặc chuyển giao công việc bảo trì cho một nhóm khác.
* Điều này đặc biệt quan trọng trong các dự án dài hạn hoặc khi có sự thay đổi nhân sự.

**7. Tuân thủ và kiểm toán**

* Trong các ngành công nghiệp yêu cầu tuân thủ nghiêm ngặt (như y tế, tài chính), quản lý phiên bản tài liệu giúp đảm bảo rằng mọi thay đổi trong quá trình bảo trì đều được ghi lại và có thể kiểm tra lại khi cần.
* Điều này giúp đáp ứng các yêu cầu pháp lý và tiêu chuẩn chất lượng.

## Câu 3.6:So sánh giữa kiểm thử sản phẩm phi thực thi và kiểm thử sản phẩm thực thi.

|  |  |  |
| --- | --- | --- |
| **Tiêu chí** | **Kiểm thử sản phẩm phi thực thi** | **Kiểm thử sản phẩm thực thi** |
| **Định nghĩa** | Kiểm thử các sản phẩm không thể chạy được, như tài liệu yêu cầu, thiết kế, kế hoạch dự án. | Kiểm thử các sản phẩm có thể chạy được, như mã nguồn, phần mềm, ứng dụng. |
| **Mục tiêu** | Đảm bảo tính chính xác, đầy đủ và nhất quán của tài liệu. | Đảm bảo phần mềm hoạt động đúng theo yêu cầu và không có lỗi. |
| **Đối tượng kiểm thử** | Tài liệu yêu cầu, tài liệu thiết kế, kế hoạch dự án, tài liệu kiểm thử. | Mã nguồn, phần mềm, ứng dụng, hệ thống. |
| **Phương pháp kiểm thử** | - Đánh giá tài liệu (Document Review)  - Kiểm tra tính nhất quán (Consistency Check)  - Kiểm tra tính đầy đủ (Completeness Check) | - Kiểm thử đơn vị (Unit Testing)  - Kiểm thử tích hợp (Integration Testing)  - Kiểm thử hệ thống (System Testing)  - Kiểm thử chấp nhận (Acceptance Testing) |
| **Thời điểm thực hiện** | Thực hiện trong giai đoạn đầu của quy trình phát triển, trước khi viết mã nguồn. | Thực hiện sau khi có mã nguồn hoặc phần mềm có thể chạy được. |
| **Công cụ hỗ trợ** | - Công cụ quản lý tài liệu (Document Management Tools)  - Công cụ đánh giá tài liệu (Review Tools) | - Công cụ kiểm thử tự động (Automated Testing Tools)  - Công cụ gỡ lỗi (Debugging Tools) |
| **Kết quả mong đợi** | - Tài liệu chính xác, đầy đủ và nhất quán.  - Giảm thiểu rủi ro do sai sót trong tài liệu. | - Phần mềm hoạt động đúng theo yêu cầu.  - Phát hiện và sửa chữa lỗi trước khi triển khai. |
| **Ví dụ** | - Kiểm tra tài liệu yêu cầu để đảm bảo không có mâu thuẫn.  - Kiểm tra tính đầy đủ của thiết kế hệ thống. | - Kiểm thử chức năng đăng nhập.  - Kiểm thử hiệu năng của hệ thống. |

## Câu 3.7:Thảo luận về cách cải thiện quy trình lập kế hoạch để giảm thiểu rủi ro trong dự án phần mềm.

Lập kế hoạch hiệu quả là chìa khóa để giảm thiểu rủi ro trong các dự án phần mềm. Một kế hoạch tốt không chỉ giúp nhóm phát triển có lộ trình rõ ràng mà còn giảm thiểu các vấn đề tiềm ẩn như trễ tiến độ, vượt ngân sách, hoặc lỗi phần mềm nghiêm trọng.

Dưới đây là các cách cải thiện quy trình lập kế hoạch nhằm giảm thiểu rủi ro trong dự án phần mềm:

1. Xác định rủi ro sớm bằng phân tích rủi ro

📌 Cách thực hiện:

* Sử dụng ma trận rủi ro (Risk Matrix) để đánh giá mức độ ảnh hưởng và xác suất xảy ra của từng rủi ro.
* Xác định các yếu tố rủi ro tiềm ẩn, ví dụ:
  + Rủi ro kỹ thuật (công nghệ mới, lỗi bảo mật).
  + Rủi ro về nhân lực (thiếu lập trình viên có kỹ năng, nghỉ việc giữa chừng).
  + Rủi ro tài chính (vượt ngân sách).

✅ Lợi ích: Giúp nhóm phát triển chuẩn bị phương án xử lý trước khi rủi ro xảy ra.

2. Sử dụng mô hình phát triển phù hợp

📌 Cách thực hiện:

* Lựa chọn mô hình phát triển phần mềm phù hợp với tính chất dự án:
  + Mô hình Agile: Nếu dự án có yêu cầu thay đổi thường xuyên, cần linh hoạt.
  + Mô hình xoắn ốc: Nếu cần quản lý rủi ro tốt với nhiều giai đoạn kiểm tra.
  + Mô hình Waterfall: Nếu yêu cầu rõ ràng, ít thay đổi.

✅ Lợi ích: Tránh việc chọn mô hình sai, gây tốn thời gian và tài nguyên.

3. Lập kế hoạch thời gian và nguồn lực chi tiết

📌 Cách thực hiện:

* Phân chia công việc thành các giai đoạn nhỏ và đặt mốc hoàn thành (Milestone).
* Sử dụng công cụ quản lý dự án như Jira, Trello, Microsoft Project để theo dõi tiến độ.
* Lập kế hoạch dự phòng (Contingency Plan) để xử lý trường hợp trễ tiến độ hoặc thiếu nhân lực.

✅ Lợi ích: Giúp nhóm tránh tình trạng nước rút (crunch time) vào cuối dự án, gây áp lực lớn.

4. Xây dựng chiến lược kiểm thử sớm và liên tục

📌 Cách thực hiện:

* Áp dụng kiểm thử tự động (Automated Testing) để phát hiện lỗi sớm.
* Kiểm thử tích hợp liên tục (Continuous Integration & Continuous Deployment - CI/CD) để phát hiện lỗi ngay khi code mới được thêm vào.
* Thực hiện kiểm thử bảo mật sớm để tránh lỗ hổng bảo mật nghiêm trọng.

✅ Lợi ích: Giảm thiểu rủi ro về lỗi nghiêm trọng ở giai đoạn cuối, giúp tiết kiệm thời gian sửa lỗi.

5. Quản lý phạm vi dự án để tránh "Scope Creep"

📌 Cách thực hiện:

* Xác định rõ phạm vi dự án ngay từ đầu, tránh bổ sung quá nhiều yêu cầu ngoài kế hoạch ban đầu.
* Sử dụng phương pháp MoSCoW (Must-have, Should-have, Could-have, Won’t-have) để ưu tiên tính năng quan trọng.
* Yêu cầu khách hàng phê duyệt mỗi thay đổi lớn để tránh mở rộng phạm vi quá mức.

✅ Lợi ích: Giữ dự án trong kiểm soát ngân sách và thời gian, tránh việc mở rộng tính năng không cần thiết.

6. Đảm bảo giao tiếp và cộng tác hiệu quả

📌 Cách thực hiện:

* Thiết lập các cuộc họp định kỳ để cập nhật tiến độ và phát hiện vấn đề kịp thời.
* Sử dụng tài liệu rõ ràng và hệ thống quản lý tri thức (Confluence, Google Docs) để đảm bảo mọi thành viên hiểu rõ yêu cầu.
* Tạo môi trường làm việc cởi mở, khuyến khích lập trình viên báo cáo sớm nếu có vấn đề.

✅ Lợi ích: Tránh hiểu sai yêu cầu và giúp cả nhóm làm việc hiệu quả hơn.

7. Đánh giá và điều chỉnh kế hoạch liên tục

📌 Cách thực hiện:

* Sau mỗi giai đoạn, nhóm cần đánh giá lại tiến độ và điều chỉnh kế hoạch nếu cần.
* Sử dụng phương pháp Agile (Sprint Review, Retrospective) để cải tiến liên tục.
* Nếu phát hiện rủi ro mới, cập nhật kế hoạch xử lý rủi ro ngay lập tức.

✅ Lợi ích: Giữ dự án linh hoạt và thích ứng tốt với thay đổi, giảm nguy cơ thất bại.

## 

## Câu 3.8:Đề xuất các công cụ hỗ trợ việc lập tài liệu kiểm thử và quản lý phiên bản.

Trong quy trình phát triển phần mềm, **lập tài liệu kiểm thử** giúp đảm bảo tính chính xác và đầy đủ của các kịch bản kiểm thử, trong khi **quản lý phiên bản** giúp kiểm soát thay đổi mã nguồn, tài liệu và các tệp liên quan. Dưới đây là một số công cụ hữu ích cho từng mục đích.

1️⃣ Công cụ lập tài liệu kiểm thử

Các công cụ này giúp nhóm QA/Testers quản lý test case, test plan, test execution và theo dõi lỗi.

📌 a) TestRail

🔹 **Tính năng chính:**✅ Quản lý test case, test plan, test run trực quan.  
✅ Tích hợp với Jira, Selenium, Jenkins.  
✅ Báo cáo chi tiết về kết quả kiểm thử.

🔹 **Ưu điểm:**✔ Giao diện thân thiện, dễ dùng.  
✔ Hỗ trợ cộng tác giữa QA và Dev.

🔹 **Nhược điểm:**❌ Là công cụ trả phí.

🔹 **Link:** https://www.gurock.com/testrail/

📌 b) Zephyr (Plugin của Jira)

🔹 **Tính năng chính:**✅ Quản lý test case ngay trong Jira.  
✅ Kết nối trực tiếp với bug tracking.  
✅ Hỗ trợ kiểm thử tự động với Selenium, JUnit.

🔹 **Ưu điểm:**✔ Phù hợp cho nhóm đã dùng Jira.  
✔ Dễ tích hợp với quy trình Agile.

🔹 **Nhược điểm:**❌ Phụ thuộc vào Jira, không phải công cụ độc lập.

🔹 **Link:**<https://www.getzephyr.com/>

📌 c) TestLink (Open Source)

🔹 **Tính năng chính:**✅ Lập kế hoạch kiểm thử, tạo test case.  
✅ Theo dõi kết quả kiểm thử.  
✅ Hỗ trợ tích hợp với bug tracking tools như Mantis, Jira.

🔹 **Ưu điểm:**✔ Miễn phí, mã nguồn mở.  
✔ Có thể tùy chỉnh theo nhu cầu.

🔹 **Nhược điểm:**❌ Giao diện cũ, hơi khó dùng cho người mới.

🔹 **Link:**<http://testlink.org/>

2️⃣ Công cụ quản lý phiên bản

Các công cụ này giúp kiểm soát thay đổi mã nguồn và tài liệu trong suốt vòng đời phát triển phần mềm.

📌 a) Git + GitHub/GitLab/Bitbucket

🔹 **Tính năng chính:**✅ Quản lý phiên bản mã nguồn phân tán.  
✅ Hỗ trợ nhiều nhánh (branch) giúp làm việc song song.  
✅ Cho phép theo dõi thay đổi và quay lại phiên bản cũ khi cần.

🔹 **Ưu điểm:**✔ Phổ biến, mạnh mẽ, dễ tích hợp CI/CD.  
✔ GitHub/GitLab có giao diện web tiện lợi.

🔹 **Nhược điểm:**❌ Cần thời gian học Git nếu chưa quen.

🔹 **Link:**

* GitHub:<https://github.com/>
* GitLab:<https://about.gitlab.com/>
* Bitbucket:<https://bitbucket.org/>

📌 b) Apache Subversion (SVN)

🔹 **Tính năng chính:**✅ Hỗ trợ quản lý phiên bản tập trung (Centralized Version Control).  
✅ Quản lý tốt các tập tin lớn (binary files, tài liệu).  
✅ Hỗ trợ quyền truy cập chi tiết hơn so với Git.

🔹 **Ưu điểm:**✔ Phù hợp với dự án yêu cầu kiểm soát quyền truy cập chặt chẽ.  
✔ Dễ dùng hơn Git với người mới.

🔹 **Nhược điểm:**❌ Không hỗ trợ làm việc phân tán như Git.

🔹 **Link:**<https://subversion.apache.org/>

📌 c) Perforce Helix Core

🔹 **Tính năng chính:**✅ Hỗ trợ quản lý phiên bản tập trung, phù hợp cho nhóm lớn.  
✅ Quản lý tốt cả mã nguồn và tài nguyên lớn (video, hình ảnh, tài liệu).  
✅ Tích hợp tốt với Unreal Engine, Unity (phù hợp với game development).

🔹 **Ưu điểm:**✔ Hiệu suất cao, phù hợp với dự án lớn.  
✔ Hỗ trợ làm việc nhóm tốt.

🔹 **Nhược điểm:**❌ Không phổ biến bằng Git.  
❌ Chi phí cao.

🔹 **Link:** https://www.perforce.com/products/helix-core

## Câu 3.9:Tại sao kiểm thử chấp nhận lại là một giai đoạn quan trọng trong phát triển phần mềm?

**Kiểm thử chấp nhận (Acceptance Testing) là một giai đoạn quan trọng trong phát triển phần mềm vì:**

1. **Xác nhận phần mềm đáp ứng yêu cầu**
   * Đảm bảo phần mềm hoạt động đúng theo yêu cầu của khách hàng hoặc người dùng cuối.
   * Kiểm tra tính đầy đủ và chính xác của các tính năng.
2. **Đảm bảo trải nghiệm người dùng**
   * Đánh giá phần mềm từ góc độ người dùng thực tế.
   * Kiểm tra giao diện, hiệu suất và mức độ dễ sử dụng.
3. **Phát hiện lỗi trước khi triển khai**
   * Phát hiện các lỗi chưa được tìm thấy trong các giai đoạn kiểm thử trước đó.
   * Giảm thiểu rủi ro khi triển khai hệ thống.
4. **Hỗ trợ quyết định triển khai**
   * Nếu phần mềm vượt qua kiểm thử chấp nhận, nó sẽ được phê duyệt để triển khai chính thức.
   * Nếu không đạt yêu cầu, cần điều chỉnh trước khi phát hành.
5. **Giảm chi phí bảo trì và sửa lỗi**
   * Phát hiện lỗi sớm giúp giảm chi phí sửa chữa so với khi phần mềm đã đi vào sử dụng.

Kiểm thử chấp nhận giúp đảm bảo phần mềm sẵn sàng triển khai, đáp ứng nhu cầu thực tế và mang lại trải nghiệm tốt nhất cho người dùng.

## Câu 3.10:Thảo luận về các phương pháp hiệu quả để quản lý chất lượng phần mềm trong các dự án lớn.

**Quản lý chất lượng phần mềm trong các dự án lớn** là một thách thức quan trọng nhằm đảm bảo sản phẩm đáp ứng yêu cầu, ổn định và đáng tin cậy. Dưới đây là một số **phương pháp hiệu quả** để quản lý chất lượng phần mềm:

**Áp dụng Mô Hình & Quy Trình Phát Triển Phần Mềm Chuẩn**

* **Agile, Scrum, Kanban**: Giúp phát triển linh hoạt, phản hồi nhanh chóng với yêu cầu thay đổi.
* **Waterfall**: Phù hợp với các dự án lớn có yêu cầu cố định.
* **DevOps**: Kết hợp phát triển (Development) và vận hành (Operations) để đảm bảo triển khai nhanh, ổn định.

**Thực hiện Kiểm Thử Phần Mềm Toàn Diện**

* **Kiểm thử đơn vị (Unit Testing)**: Đảm bảo từng thành phần nhỏ hoạt động chính xác.
* **Kiểm thử tích hợp (Integration Testing)**: Đánh giá sự tương tác giữa các module.
* **Kiểm thử hồi quy (Regression Testing)**: Đảm bảo thay đổi mới không gây lỗi cho hệ thống cũ.
* **Kiểm thử hiệu năng (Performance Testing)**: Đo lường tốc độ, độ ổn định của phần mềm khi tải lớn.
* **Kiểm thử bảo mật (Security Testing)**: Xác định các lỗ hổng bảo mật trước khi triển khai.

**Sử dụng Công Cụ Hỗ Trợ Kiểm Thử & Quản Lý Chất Lượng**

* **Quản lý mã nguồn**: Git, GitHub, GitLab giúp theo dõi thay đổi mã nguồn.
* **Tự động hóa kiểm thử**: Selenium, JUnit, TestNG giúp tăng tốc độ và độ chính xác.
* **Quản lý lỗi**: JIRA, Bugzilla giúp theo dõi và xử lý lỗi hiệu quả.
* **CI/CD**: Jenkins, GitHub Actions để triển khai liên tục, phát hiện lỗi sớm.

**Thực hiện Code Review & Tuân Thủ Coding Standards**

* **Review Code định kỳ**: Peer Review giúp phát hiện lỗi logic và tối ưu code.
* **Sử dụng quy ước mã hóa**: Theo các tiêu chuẩn như PEP 8 (Python), Google Java Style Guide.
* **Áp dụng phân tích tĩnh (Static Code Analysis)**: Sử dụng SonarQube, ESLint để kiểm tra chất lượng mã nguồn tự động.

**Kiểm Soát Rủi Ro & Đảm Bảo Tài Liệu Đầy Đ**ủ

* **Xác định rủi ro sớm**: Dự đoán các vấn đề tiềm ẩn và có kế hoạch xử lý.
* **Ghi chép tài liệu rõ ràng**: Bao gồm tài liệu thiết kế, hướng dẫn sử dụng, tài liệu kiểm thử.

**Đào Tạo & Nâng Cao Nhận Thức Về Chất Lượng**

* **Đào tạo nhân viên định kỳ** về coding best practices, kiểm thử, bảo mật.
* **Xây dựng văn hóa chất lượng**, khuyến khích phản hồi và cải tiến liên tục

# PHẦN 4: CÂU HỎI TÌNH HUỐNG

## Câu 4.1: Một dự án phần mềm đã hoàn thành và sắp bàn giao cho khách hàng, nhưng khách hàng yêu cầu kiểm tra lại toàn bộ tài liệu yêu cầu và thiết kế. Đội phát triển nên xử lý thế nào?

Nguyên Tắc Xử Lý

1. **Đánh Giá Yêu Cầu Chi Tiết**

* Thành lập nhóm chuyên trách rà soát tài liệu
* Phân tích kỹ từng yêu cầu và thiết kế
* So sánh với tài liệu ban đầu và các thỏa thuận

1. **Quản Lý Giao Tiếp**

* Lập biên bản làm việc với khách hàng
* Giải thích chi tiết từng phần đã được thực hiện
* Minh bạch về quá trình phát triển

1. **Phương Án Xử Lý**

* Nếu phát hiện sai sót: Lập kế hoạch khắc phục
* Nếu không có vấn đề: Cung cấp bằng chứng chi tiết
* Sẵn sàng điều chỉnh nếu cần thiết

Tài Liệu Cần Kiểm Tra

* Tài liệu yêu cầu ban đầu
* Biên bản thỏa thuận
* Tài liệu thiết kế chi tiết
* Báo cáo tiến độ
* Các phiên bản thay đổi (nếu có)

## Câu 4.2: Trong quá trình kiểm thử hệ thống, nhóm phát triển phát hiện một lỗi nghiêm trọng nhưng thời hạn bàn giao đang đến gần. Bạn sẽ xử lý tình huống này như thế nào?

Nguyên Tắc Xử Lý

1. **Đánh Giá Mức Độ Lỗi**

* Phân tích chi tiết nguyên nhân lỗi
* Đánh giá tác động đến hệ thống
* Xác định mức độ nghiêm trọng

1. **Các Phương Án Khắc Phục**

* Phương án 1: Sửa lỗi gấp rút
  + Tập trung toàn bộ nguồn lực
  + Ưu tiên khắc phục triệt để
  + Kiểm thử lại toàn diện
* Phương Án 2: Thương Lượng Với Khách Hàng
  + Báo cáo minh bạch về lỗi
  + Đề xuất kế hoạch khắc phục
  + Thống nhất phương án chung

1. **Quản Lý Rủi Ro**

* Chuẩn bị các kịch bản xử lý
* Dự phòng nguồn lực
* Linh hoạt trong giải quyết

Nguyên Tắc Giao TIẾP

* Trung thực về tình trạng hệ thống
* Chủ động trong giải quyết
* Cam kết đảm bảo chất lượng

## Câu 4.3: Một nhóm phát triển gặp khó khăn trong việc quản lý phiên bản tài liệu do tài liệu liên tục thay đổi. Hãy đề xuất giải pháp.

Để quản lý phiên bản tài liệu hiệu quả khi tài liệu liên tục thay đổi, nhóm phát triển có thể áp dụng các giải pháp sau:

* Sử dụng hệ thống quản lý phiên bản (VCS):
  + Áp dụng công cụ như Git (với GitHub, GitLab) hoặc SVN để lưu trữ, theo dõi và quản lý mọi thay đổi của tài liệu. Mỗi phiên bản được gắn nhãn rõ ràng.
* Quy định quy trình cập nhật:
  + Đặt quy tắc cụ thể: ai có quyền chỉnh sửa, khi nào cập nhật, và yêu cầu ghi chú (commit message) mô tả thay đổi để dễ tra cứu.
* Sử dụng công cụ cộng tác trực tuyến:
  + Dùng Google Docs, Confluence, hoặc Notion để chỉnh sửa thời gian thực, tự động lưu lịch sử phiên bản, tránh xung đột.
* Tự động hóa sao lưu:
  + Thiết lập backup định kỳ trên cloud (Google Drive, OneDrive) để đảm bảo không mất dữ liệu.

## Câu 4.4: Dự án phát triển phần mềm gặp vấn đề khi khách hàng yêu cầu thay đổi lớn trong pha cài đặt. Đội phát triển nên xử lý thế nào?

Khi khách hàng yêu cầu thay đổi lớn trong pha cài đặt, đội phát triển nên xử lý:

* Đánh giá tác động:
  + Phân tích ảnh hưởng của thay đổi lên hệ thống hiện tại (chức năng, thời gian, chi phí) và thảo luận với khách hàng.
* Thương lượng ưu tiên:
  + Đề xuất giữ nguyên hệ thống đã cài đặt, đưa thay đổi vào phiên bản tiếp theo để tránh gián đoạn.
* Áp dụng Agile:
  + Thêm yêu cầu vào Product Backlog, lập kế hoạch Sprint mới để xử lý thay đổi, đảm bảo kiểm thử đầy đủ trước khi tích hợp.
* Giao tiếp rõ ràng:
  + Thông báo cho khách hàng về rủi ro, chi phí và thời gian bổ sung, đạt được sự đồng thuận trước khi thực hiện.

## Câu 4.5:Nhóm kiểm thử phát hiện nhiều lỗi chức năng trong phần mềm. Tuy nhiên, nhóm phát triển lại cho rằng đây không phải lỗi mà là tính năng. Là trưởng dự án, bạn sẽ làm gì?

Là trưởng dự án, tôi sẽ xử lý tình huống này theo cách chuyên nghiệp và hợp tác để đảm bảo chất lượng phần mềm, đồng thời tránh xung đột giữa nhóm kiểm thử và nhóm phát triển. Cụ thể, tôi sẽ thực hiện các bước sau:

1. **Xác định và phân loại vấn đề**
   * Yêu cầu nhóm kiểm thử cung cấp tài liệu chi tiết về các lỗi phát hiện được, bao gồm mô tả, bước tái hiện, kết quả mong đợi và kết quả thực tế.
   * Xem xét tài liệu yêu cầu (SRS - Software Requirement Specification) để xác định xem những vấn đề này có vi phạm yêu cầu của khách hàng hay không.
2. **Tổ chức cuộc họp giữa các bên liên quan**
   * Gồm nhóm phát triển, nhóm kiểm thử và (nếu cần) đại diện khách hàng.
   * Mục tiêu là thảo luận về từng lỗi, làm rõ liệu đó là lỗi hay tính năng được thiết kế.
3. **Đối chiếu với yêu cầu ban đầu**
   * Nếu vấn đề vi phạm yêu cầu đã thống nhất ban đầu, đó là lỗi và nhóm phát triển phải sửa.
   * Nếu không vi phạm nhưng ảnh hưởng đến trải nghiệm người dùng, có thể cần thay đổi yêu cầu hoặc cải tiến.
4. **Thống nhất hướng giải quyết**
   * Nếu thực sự là tính năng nhưng gây hiểu nhầm, có thể cần cập nhật tài liệu hướng dẫn hoặc giao diện để tránh nhầm lẫn.
   * Nếu là lỗi, nhóm phát triển cần lên kế hoạch sửa chữa theo mức độ ưu tiên.
5. **Báo cáo và truyền đạt quyết định**
   * Tổng hợp kết quả cuộc họp và quyết định cuối cùng.
   * Đảm bảo tất cả các bên đều hiểu rõ lý do và cách xử lý để tránh tranh cãi về sau.

## Câu 4.6:Khách hàng yêu cầu bổ sung một tính năng mới khi phần mềm đã hoàn thành pha kiểm thử tích hợp. Đội phát triển nên làm gì?

Khi khách hàng yêu cầu bổ sung một tính năng mới sau khi phần mềm đã hoàn thành pha kiểm thử tích hợp, đội phát triển cần thực hiện các bước sau để đảm bảo rằng việc thêm tính năng mới không ảnh hưởng đến hệ thống hiện tại và đáp ứng yêu cầu của khách hàng:

1. Đánh giá yêu cầu mới

* Xác định rõ ràng tính năng mới mà khách hàng yêu cầu và kiểm tra tính khả thi của nó.
* Phân tích tác động của tính năng mới đến hệ thống hiện tại, bao gồm các module liên quan, giao diện người dùng, và cơ sở dữ liệu.
* Xác định xem yêu cầu này có nằm trong phạm vi dự án ban đầu hay cần thỏa thuận bổ sung.

2. Ước lượng tài nguyên và thời gian

* Ước tính thời gian và nguồn lực cần thiết để phát triển và kiểm thử tính năng mới.
* Xác định các rủi ro và thách thức có thể gặp phải khi thêm tính năng mới.
* Đánh giá chi phí thực hiện và trao đổi với khách hàng về ngân sách (nếu cần).

3. Lập kế hoạch triển khai

* Xây dựng kế hoạch chi tiết cho việc phát triển tính năng mới, bao gồm các giai đoạn: thiết kế, lập trình, kiểm thử và tích hợp lại.
* Đảm bảo rằng kế hoạch được sự đồng thuận của tất cả các bên liên quan, bao gồm khách hàng, đội phát triển và đội kiểm thử.

4. Phát triển và kiểm thử tính năng mới

* Tiến hành phát triển tính năng mới theo kế hoạch đã lập.
* Kiểm thử đơn vị (Unit Testing) và kiểm thử độc lập tính năng mới trước khi tích hợp vào hệ thống.

5. Tích hợp lại và kiểm thử toàn diện

* Tích hợp tính năng mới vào hệ thống hiện tại và thực hiện kiểm thử tích hợp lại để đảm bảo không có xung đột.
* Thực hiện kiểm thử hồi quy (Regression Testing) để đảm bảo tính năng cũ không bị ảnh hưởng.
* Đánh giá hiệu suất của hệ thống sau khi tích hợp.

6. Xác nhận với khách hàng

* Gửi phiên bản phần mềm mới đã tích hợp tính năng mới cho khách hàng để kiểm tra và xác nhận.
* Thu thập phản hồi từ khách hàng và điều chỉnh nếu cần.

7. Cập nhật tài liệu và đào tạo người dùng

* Cập nhật tài liệu hệ thống để phản ánh sự thay đổi và bổ sung tính năng mới.
* Đào tạo người dùng (nếu cần) để họ có thể sử dụng tính năng mới một cách hiệu quả.

**Lưu ý:**

* Nếu tính năng mới có tác động lớn, có thể cần một bản cập nhật phần mềm (Patch/Version Update).
* Trong trường hợp tính năng mới không được lên kế hoạch từ đầu, cần thương thảo với khách hàng về chi phí và thời gian phát triển.

## Câu 4.7:Một công ty phát triển phần mềm nhỏ muốn xây dựng nhóm SQA nhưng gặp khó khăn về ngân sách. Hãy đề xuất giải pháp.

Nhóm **Đảm bảo chất lượng phần mềm (SQA - Software Quality Assurance)** đóng vai trò quan trọng trong việc đảm bảo sản phẩm phần mềm đáp ứng yêu cầu và tiêu chuẩn chất lượng. Tuy nhiên, với ngân sách hạn chế, công ty có thể áp dụng các giải pháp tối ưu hóa chi phí như sau:

1️⃣ Tận dụng kiểm thử tự động với công cụ mã nguồn mở

**🔹 Giải pháp:** Thay vì sử dụng các công cụ kiểm thử thương mại đắt đỏ, công ty có thể áp dụng **công cụ mã nguồn mở** để kiểm thử tự động. Điều này giúp giảm chi phí nhân sự cho kiểm thử thủ công mà vẫn đảm bảo chất lượng phần mềm.

**🔹 Công cụ đề xuất:**

* **Selenium** (Web Testing) –<https://www.selenium.dev/>
* **JUnit/TestNG** (Kiểm thử đơn vị cho Java)
* **Cypress** (Kiểm thử giao diện web hiện đại) –<https://www.cypress.io/>
* **Appium** (Kiểm thử ứng dụng di động) –<https://appium.io/>

**🔹 Lợi ích:**✔ Giảm chi phí nhân sự cho kiểm thử thủ công.  
✔ Tăng tốc độ kiểm thử và giảm lỗi do con người.  
✔ Dễ dàng tích hợp với quy trình DevOps.

2️⃣ Xây dựng quy trình SQA tinh gọn (Lean SQA)

**🔹 Giải pháp:** Áp dụng quy trình kiểm thử tinh gọn, tập trung vào **những kiểm thử quan trọng nhất** thay vì kiểm tra tất cả mọi thứ.

**🔹 Cách thực hiện:**

* Xác định các **yếu tố cốt lõi cần kiểm thử** (chức năng quan trọng, hiệu suất, bảo mật).
* Sử dụng **kiểm thử dựa trên rủi ro (Risk-Based Testing - RBT)** để ưu tiên kiểm tra những phần dễ xảy ra lỗi nhất.
* Áp dụng **kiểm thử hồi quy tự động** để tiết kiệm công sức.

**🔹 Lợi ích:**✔ Giảm thời gian kiểm thử mà vẫn đảm bảo chất lượng.  
✔ Tập trung vào các lỗi quan trọng thay vì kiểm thử dàn trải.

3️⃣ Thuê kiểm thử viên theo dự án (Outsourcing & Freelancer)

**🔹 Giải pháp:** Nếu không đủ ngân sách để thuê kiểm thử viên toàn thời gian, công ty có thể **thuê freelancer hoặc công ty kiểm thử bên ngoài** theo từng dự án.

**🔹 Các nền tảng tìm tester freelance:**

* **Upwork** –<https://www.upwork.com/>
* **Freelancer** –<https://www.freelancer.com/>
* **Toptal** (Tester chất lượng cao) –<https://www.toptal.com/>

**🔹 Lợi ích:**✔ Tiết kiệm chi phí so với thuê nhân viên toàn thời gian.  
✔ Linh hoạt mở rộng hoặc thu hẹp nhóm kiểm thử khi cần.

4️⃣ Đào tạo nội bộ thay vì thuê chuyên gia đắt đỏ

**🔹 Giải pháp:** Thay vì tuyển dụng chuyên gia SQA, công ty có thể đào tạo nội bộ bằng cách:

* **Huấn luyện lập trình viên** thực hiện kiểm thử đơn vị và kiểm thử tích hợp.
* **Giao nhiệm vụ kiểm thử cho QA kiêm nhiệm** thay vì có một nhóm riêng biệt.
* **Tận dụng tài nguyên miễn phí để đào tạo**:

📌 **Các khóa học miễn phí về kiểm thử phần mềm:**

* **Udemy** (Một số khóa miễn phí) –<https://www.udemy.com/>
* **Coursera** (Có hỗ trợ tài chính) –<https://www.coursera.org/>
* **Test Automation University** – https://testautomationu.applitools.com/

**🔹 Lợi ích:**✔ Tiết kiệm chi phí thuê chuyên gia SQA.  
✔ Nâng cao năng lực kiểm thử trong nhóm hiện có.

5️⃣ Áp dụng kiểm thử cộng đồng (Crowdsourced Testing)

**🔹 Giải pháp:** Công ty có thể tận dụng các nền tảng **kiểm thử cộng đồng** để có nhiều tester kiểm thử sản phẩm với chi phí thấp.

**🔹 Các nền tảng kiểm thử cộng đồng:**

* **Test IO** –<https://test.io/>
* **uTest** –<https://www.utest.com/>
* **Bugcrowd** (Chuyên về bảo mật) –<https://www.bugcrowd.com/>

**🔹 Lợi ích:**✔ Kiểm thử trên nhiều thiết bị thực tế khác nhau.  
✔ Chi phí thấp hơn so với thuê tester nội bộ.

## Câu 4.8:Trong quá trình làm tài liệu kiểm thử, nhóm phát triển không thống nhất được về nội dung cần đưa vào tài liệu. Là trưởng nhóm, bạn sẽ giải quyết vấn đề này như thế nào?

Là trưởng nhóm, để giải quyết vấn đề không thống nhất về nội dung tài liệu kiểm thử, tôi sẽ thực hiện các bước sau để đảm bảo sự đồng thuận giữa các thành viên trong nhóm và duy trì chất lượng tài liệu kiểm thử.

1️⃣ Xác định nguyên nhân gốc rễ của sự không thống nhất

🔹 Tôi sẽ tổ chức một cuộc họp nhóm để hiểu rõ lý do tại sao có sự bất đồng. Một số nguyên nhân có thể là:

* Không có tiêu chuẩn chung về tài liệu kiểm thử.
* Quan điểm khác nhau về mức độ chi tiết cần có.
* Thiếu thông tin hoặc hiểu nhầm về yêu cầu phần mềm.
* Thiếu quy trình thống nhất khi lập tài liệu kiểm thử.

📌 Giải pháp: Sử dụng các câu hỏi mở để làm rõ vấn đề, ví dụ:

* Tại sao bạn nghĩ rằng mục này không cần có trong tài liệu?
* Mức độ chi tiết nào là phù hợp cho tài liệu này?
* Có chuẩn kiểm thử nào mà chúng ta nên tuân theo không?

2️⃣ Xây dựng một tiêu chuẩn chung cho tài liệu kiểm thử

🔹 Tôi sẽ đề xuất một bộ khung chuẩn để tài liệu kiểm thử tuân theo. Bộ khung này có thể dựa trên ISTQB hoặc một chuẩn phù hợp với dự án.

📌 Ví dụ về cấu trúc tài liệu kiểm thử:  
✅ Mục tiêu kiểm thử (Testing Objectives)  
✅ Phạm vi kiểm thử (Scope of Testing)  
✅ Danh sách các trường hợp kiểm thử (Test Cases)  
✅ Tiêu chí chấp nhận (Acceptance Criteria)  
✅ Kế hoạch kiểm thử (Test Plan)

🔹 Công cụ hỗ trợ tiêu chuẩn hóa tài liệu kiểm thử:

* IEEE 829 - Tiêu chuẩn về tài liệu kiểm thử.
* TestRail, Jira, hoặc Zephyr để quản lý kiểm thử.

3️⃣ Sử dụng dữ liệu và nguyên tắc để ra quyết định

🔹 Nếu vẫn có bất đồng, tôi sẽ dùng dữ liệu để hỗ trợ quyết định.

* Kiểm tra yêu cầu phần mềm: Xác minh tài liệu yêu cầu (SRS) để xác định nội dung nào là cần thiết.
* Tham khảo tài liệu kiểm thử trước đó: Nếu công ty đã có dự án tương tự, tôi sẽ lấy tài liệu cũ làm hướng dẫn.
* Đánh giá tác động: Nếu một nội dung bị loại bỏ, liệu có ảnh hưởng đến chất lượng kiểm thử không?

4️⃣ Thống nhất bằng cách thử nghiệm thực tế

🔹 Tôi sẽ yêu cầu nhóm tạo một phiên bản thử nghiệm của tài liệu kiểm thử với các nội dung tranh cãi.

* Sau đó, nhóm thực hiện một vòng kiểm thử với tài liệu này.
* Nếu nội dung nào không cần thiết hoặc thiếu, chúng ta sẽ điều chỉnh dựa trên phản hồi thực tế.

📌 Lợi ích:  
✔ Tránh tranh luận lý thuyết mà không có bằng chứng thực tế.  
✔ Đưa ra quyết định dựa trên hiệu quả kiểm thử thực tế.

5️⃣ Đưa ra quyết định cuối cùng và tài liệu hóa quy trình

🔹 Sau khi nhóm thống nhất hoặc sau khi thử nghiệm thực tế, tôi sẽ tóm tắt quyết định cuối cùng và tài liệu hóa quy trình kiểm thử để tránh tranh luận trong tương lai.  
🔹 Tôi cũng có thể phân công một người chịu trách nhiệm kiểm tra tài liệu kiểm thử trước khi hoàn thiện để đảm bảo tính nhất quán.

📌 Công cụ hỗ trợ tài liệu hóa quyết định:

* Google Docs/Confluence – Ghi lại quy tắc chuẩn về tài liệu kiểm thử.
* Jira – Ghi chú quyết định vào hệ thống quản lý dự án.

## 

## Câu 4.9:Dự án phát triển phần mềm cho một ngân hàng yêu cầu bảo mật cao. Đề xuất cách lập kế hoạch kiểm thử cho dự án này.

Kế Hoạch Kiểm Thử Cho Dự Án Phần Mềm Ngân Hàng

Dự án phần mềm ngân hàng yêu cầu **bảo mật cao, độ tin cậy lớn và khả năng hoạt động ổn định**. Do đó, kế hoạch kiểm thử cần được xây dựng chặt chẽ với nhiều lớp kiểm thử khác nhau.

1. Xác Định Mục Tiêu Kiểm Thử

* Đảm bảo **tính chính xác** của các giao dịch tài chính.
* Đảm bảo **bảo mật** dữ liệu khách hàng và hệ thống.
* Đảm bảo **khả năng chịu tải** của hệ thống khi có nhiều giao dịch đồng thời.
* Đảm bảo **tính tuân thủ** với các quy định tài chính và bảo mật (PCI DSS, GDPR).

2. Phạm Vi Kiểm Thử

Kiểm thử tập trung vào các tính năng quan trọng:  
✅ **Xác thực & Ủy quyền**: Đăng nhập, đăng xuất, quản lý OTP, xác thực hai yếu tố (2FA).  
✅ **Giao dịch tài chính**: Chuyển tiền, thanh toán hóa đơn, kiểm tra số dư.  
✅ **Bảo mật dữ liệu**: Mã hóa dữ liệu, bảo vệ thông tin cá nhân.  
✅ **Khả năng chịu tải**: Hệ thống vẫn hoạt động khi có hàng nghìn giao dịch đồng thời.  
✅ **Giao diện người dùng**: Đảm bảo UI/UX thân thiện, dễ sử dụng.

3. Chiến Lược Kiểm Thử

3.1. Kiểm Thử Chức Năng (Functional Testing)

Mục tiêu: Đảm bảo các tính năng hoạt động đúng như mong đợi.

* Kiểm thử đơn vị (Unit Testing): Kiểm tra từng module nhỏ.
* Kiểm thử tích hợp (Integration Testing): Đánh giá sự kết nối giữa các module (ví dụ: giữa hệ thống ngân hàng và cổng thanh toán).
* Kiểm thử hệ thống (System Testing): Kiểm tra toàn bộ hệ thống theo luồng thực tế.
* Kiểm thử chấp nhận (UAT - User Acceptance Testing): Khách hàng kiểm tra trước khi triển khai chính thức.

3.2. Kiểm Thử Bảo Mật (Security Testing)

Mục tiêu: Xác định và khắc phục các lỗ hổng bảo mật.

* **Kiểm thử xác thực**: Đảm bảo chỉ có người dùng hợp lệ truy cập được hệ thống.
* **Kiểm thử tấn công giả lập (Penetration Testing)**: Mô phỏng tấn công để phát hiện lỗ hổng.
* **Kiểm thử mã hóa dữ liệu**: Đảm bảo dữ liệu nhạy cảm (số thẻ, mật khẩu) được mã hóa đúng cách.
* **Kiểm thử SQL Injection, XSS, CSRF**: Ngăn chặn các cuộc tấn công phổ biến.

3.3. Kiểm Thử Hiệu Năng (Performance Testing)

Mục tiêu: Đảm bảo hệ thống hoạt động mượt mà ngay cả khi có lượng người dùng lớn.

* **Kiểm thử tải (Load Testing)**: Kiểm tra hệ thống khi có số lượng giao dịch lớn.
* **Kiểm thử áp lực (Stress Testing)**: Mô phỏng trường hợp hệ thống bị quá tải.
* **Kiểm thử ổn định (Stability Testing)**: Kiểm tra hệ thống hoạt động liên tục trong thời gian dài.

3.4. Kiểm Thử Khả Năng Phục Hồi (Recovery Testing)

Mục tiêu: Đảm bảo hệ thống có thể phục hồi sau sự cố.

* **Mô phỏng sự cố server**: Kiểm tra khả năng khôi phục dữ liệu khi mất kết nối hoặc lỗi hệ thống.
* **Kiểm thử sao lưu và khôi phục (Backup & Restore Testing)**: Đảm bảo dữ liệu không bị mất khi gặp sự cố.

4. Công Cụ Hỗ Trợ Kiểm Thử

* **Kiểm thử tự động**: Selenium, JMeter, Appium (cho ứng dụng di động).
* **Kiểm thử bảo mật**: OWASP ZAP, Burp Suite, Metasploit.
* **Kiểm thử hiệu năng**: Apache JMeter, LoadRunner.
* **Quản lý lỗi**: JIRA, Bugzilla.

5. Lập Kế Hoạch Thực Hiện Kiểm Thử

|  |  |  |
| --- | --- | --- |
| **Giai đoạn** | **Hoạt động kiểm thử** | **Thời gian** |
| **Giai đoạn 1** | Lập kế hoạch kiểm thử | 1 tuần |
| **Giai đoạn 2** | Viết test case & chuẩn bị dữ liệu kiểm thử | 2 tuần |
| **Giai đoạn 3** | Thực hiện kiểm thử chức năng | 3 tuần |
| **Giai đoạn 4** | Kiểm thử bảo mật | 2 tuần |
| **Giai đoạn 5** | Kiểm thử hiệu năng | 2 tuần |
| **Giai đoạn 6** | UAT & kiểm thử cuối cùng | 2 tuần |

## Câu 4.10:Sau khi triển khai phần mềm, khách hàng phát hiện ra một số lỗi bảo mật nghiêm trọng. Đội phát triển cần xử lý ra sao để khắc phục vấn đề và lấy lại niềm tin từ khách hàng?

## 

Cách Xử Lý Lỗi Bảo Mật Sau Khi Triển Khai Phần Mềm

Khi khách hàng phát hiện lỗi bảo mật nghiêm trọng sau khi triển khai phần mềm, đội phát triển cần hành động **nhanh chóng, minh bạch và có kế hoạch khắc phục rõ ràng** để bảo vệ dữ liệu và lấy lại niềm tin từ khách hàng. Dưới đây là quy trình xử lý hiệu quả:

1. Phản Hồi Ngay Lập Tức & Đánh Giá Mức Độ Nguy Hiểm

🔹 **Xác nhận lỗi với khách hàng**: Ghi nhận phản hồi một cách chuyên nghiệp, tránh đổ lỗi.  
🔹 **Xác định mức độ nghiêm trọng**:

* **Lỗi nhỏ**: Ảnh hưởng một số chức năng, chưa gây rủi ro lớn.
* **Lỗi nghiêm trọng**: Ảnh hưởng dữ liệu, rủi ro mất thông tin người dùng.
* **Lỗi khẩn cấp**: Hệ thống bị tấn công, lộ dữ liệu nhạy cảm.

2. Ngăn Chặn & Giảm Thiểu Thiệt Hại

🔹 **Tạm thời vô hiệu hóa tính năng bị lỗi** (nếu cần).  
🔹 **Cô lập lỗi** để tránh ảnh hưởng đến toàn bộ hệ thống.  
🔹 **Kiểm tra hệ thống giám sát bảo mật** để phát hiện các cuộc tấn công liên quan.  
🔹 **Thông báo nội bộ & thiết lập nhóm xử lý khẩn cấp** để tập trung khắc phục.

3. Điều Tra Nguyên Nhân & Sửa Lỗi

🔹 **Kiểm tra nhật ký hệ thống (Logs)** để xác định cách thức lỗi xảy ra.  
🔹 **Thực hiện phân tích bảo mật** để tìm các lỗ hổng liên quan.  
🔹 **Sửa lỗi & vá lỗ hổng**:

* Nếu lỗi do **cấu hình sai**, cần điều chỉnh và kiểm tra lại.
* Nếu lỗi do **lỗ hổng trong mã nguồn**, cần sửa code, kiểm thử lại.
* Nếu lỗi do **tấn công mạng**, cần tăng cường tường lửa, kiểm tra truy cập trái phép.  
  🔹 **Kiểm thử bảo mật lại toàn bộ hệ thống** để đảm bảo lỗi đã được khắc phục triệt để.

4. Thông Báo & Cam Kết Với Khách Hàng

🔹 **Thông báo minh bạch**: Giải thích lỗi một cách chuyên nghiệp, tránh gây hoang mang.  
🔹 **Cung cấp giải pháp khắc phục**:

* Đã sửa lỗi & cập nhật bản vá (hotfix).
* Đề xuất các biện pháp để bảo vệ dữ liệu người dùng.  
  🔹 **Công khai cam kết** về việc tăng cường bảo mật để tránh lặp lại sự cố.

5. Cải Thiện Quy Trình Bảo Mật & Phòng Ngừa

🔹 **Cập nhật chính sách bảo mật** để ngăn chặn lỗi tương tự xảy ra.  
🔹 **Tăng cường kiểm thử bảo mật** trong các bản cập nhật sau:

* Kiểm thử thâm nhập (Penetration Testing).
* Kiểm thử mã hóa dữ liệu & xác thực.  
  🔹 **Theo dõi & giám sát bảo mật liên tục** để phát hiện sớm các mối đe dọa.