40 CÂU HỎI PHỎNG VẤN DÀNH CHO FRESHER TESTER

10 câu bản thân + 20 câu lý thuyết + 10 câu tình huống - Kèm đáp án mẫu chi tiết

Tài liệu được tổng hợp dành cho các bạn **Fresher/Junior Tester** đang chuẩn bị bước vào những buổi phỏng vấn đầu tiên trong sự nghiệp QA/Testing. Tài liệu được chia thành 3 phần theo đúng trình tự của một buổi phỏng vấn thực tế: **Câu hỏi về bản thân (HR)**, **Câu hỏi lý thuyết** và **Câu hỏi tình huống**. Mỗi câu đều có đáp án mẫu, ví dụ minh họa và mẹo trả lời.

Biên soạn: Anh Tester

Website: https://anhtester.com

Ngày phát hành: 28/05/2026

MỤC LỤC

Bấm vào tiêu đề câu hỏi để chuyển đến phần nội dung tương ứng.

Câu 25. Bug Life Cycle (Vòng đời của một Bug) gồm những trạng thái nào?

Câu 26. Một Bug Report tốt cần có những thông tin gì?

Câu 27. Boundary Value Analysis (BVA) và Equivalence Partitioning (EP) là gì?

Câu 28. Test Plan và Test Strategy khác nhau như thế nào?

Câu 29. Manual Testing và Automation Testing - khi nào nên dùng cái nào?

Câu 30. Là Fresher Tester, bạn sẽ làm gì khi gặp Dev không thừa nhận đó là bug?

◆ PHẦN III: CÂU HỎI TÌNH HUỐNG (10 câu)

Câu 31. Dự án sắp release trong 2 ngày nhưng còn rất nhiều test case chưa chạy. Bạn xử lý thế nào?

Câu 32. Bạn phát hiện một bug nghiêm trọng (Critical) ngay trước giờ release. Bạn sẽ làm gì?

Câu 33. Yêu cầu (requirement) mơ hồ hoặc không rõ ràng, bạn xử lý thế nào để viết test case?

Câu 34. Khi không có tài liệu requirement, bạn sẽ test phần mềm như thế nào?

Câu 35. Một bug bạn báo lên đã bị Dev đóng với lý do 'Not a Bug' nhưng bạn vẫn tin đó là bug. Bạn làm gì?

Câu 36. Bạn được giao test một module hoàn toàn mới mà bạn chưa từng biết. Bạn bắt đầu từ đâu?

Câu 37. Bạn vừa fix bug và verify Pass, nhưng sau khi release vào production, người dùng vẫn báo lỗi. Bạn xử lý thế nào?

Câu 38. Bạn nhận được build mới rất gấp cần test xong trong 2 tiếng để release. Bạn ưu tiên test gì?

Câu 39. Đồng nghiệp Tester khác báo bug giống y hệt bug bạn đã báo cách đây 1 tuần. Bạn xử lý thế nào?

Câu 40. Bạn không đồng ý với cách Test Lead phân chia công việc/giao test case. Bạn xử lý thế nào?

PHẦN I

CÂU HỎI VỀ BẢN THÂN

(10 câu)

Phần này gồm 10 câu hỏi **về bản thân ứng viên** - thường do HR đặt ra ở đầu buổi phỏng vấn. Đây là phần đánh giá **tính cách, động lực, định hướng nghề nghiệp** và **sự phù hợp văn hóa** của ứng viên với công ty. Trả lời tốt phần này sẽ tạo ấn tượng đầu rất tích cực cho cả buổi phỏng vấn.

Câu 1. Hãy giới thiệu về bản thân bạn?

◆ Gợi ý cách trả lời:

Đây là câu hỏi mở đầu hầu như luôn xuất hiện - hãy chuẩn bị kỹ.

Công thức trả lời: Quá khứ → Hiện tại → Tương lai (thời lượng 1-2 phút).

Quá khứ: Họ tên, học trường nào, chuyên ngành gì, lý do chọn ngành Testing.

Hiện tại: Đã làm/thực tập ở đâu, có kỹ năng gì (manual, automation, tools đã sử dụng), dự án nổi bật đã tham gia.

Tương lai: Mục tiêu nghề nghiệp ngắn hạn và lý do ứng tuyển vào công ty này.

★ Tip phỏng vấn:

► **Tránh:** Đọc lại nguyên văn CV - nhà tuyển dụng đã đọc rồi.

► **Nên:** Nhấn mạnh điểm phù hợp với JD (Job Description) của vị trí ứng tuyển.

► Giữ thái độ tự tin, mỉm cười, giao tiếp bằng mắt.

► Luyện trước ở nhà 3-5 lần để trả lời mượt mà, không vấp.

Câu 2. Tại sao bạn chọn nghề Tester?

◆ Gợi ý cách trả lời:

Câu hỏi đánh giá đam mê và mức độ nghiêm túc với nghề.

KHÔNG NÊN trả lời:

► 'Vì em không giỏi code nên chọn Tester' - đây là câu trả lời tiêu cực và sai lầm.

► 'Vì nghề này dễ vào' - thể hiện thiếu nghiêm túc.

► 'Vì lương cao' - đặt vật chất lên trên đam mê.

NÊN trả lời theo hướng tích cực:

► Yêu thích sự tỉ mỉ, chú trọng chi tiết.

► Có tư duy phân tích, logic, thích đặt câu hỏi 'điều gì sẽ xảy ra nếu...'.

► Thích vai trò 'bảo vệ chất lượng sản phẩm' - người dùng cuối là ưu tiên.

► Yêu thích quá trình 'phá vỡ' hệ thống để giúp sản phẩm tốt hơn.

★ **Tip:** Có thể kể một câu chuyện cá nhân - ví dụ: từng phát hiện bug khi dùng một ứng dụng nào đó và thấy thú vị, hoặc đã tham gia khóa học Testing và say mê từ đó. Câu chuyện thật làm câu trả lời thuyết phục hơn lý thuyết.

Câu 3. Tại sao bạn nghỉ việc ở công ty cũ? (cho người đã có kinh nghiệm)

◆ Gợi ý cách trả lời:

Câu hỏi nhạy cảm - đánh giá thái độ và tính cách của ứng viên.

TUYỆT ĐỐI KHÔNG:

► Nói xấu công ty cũ, sếp cũ, đồng nghiệp cũ.

► Kể chi tiết các xung đột, drama tiêu cực.

► Đổ lỗi cho người khác về việc nghỉ việc.

Nhà tuyển dụng nghĩ gì: Nếu bạn nói xấu công ty cũ, họ sẽ lo rằng tương lai bạn cũng sẽ nói xấu họ với người khác.

NÊN trả lời theo các hướng tích cực:

► **Phát triển bản thân:** 'Em muốn tiếp cận lĩnh vực mới như automation/performance testing mà công ty cũ chưa có cơ hội.'

► **Domain phù hợp hơn:** 'Em muốn làm trong domain fintech/e-commerce phù hợp với định hướng dài hạn.'

► **Lộ trình thăng tiến:** 'Em muốn môi trường có lộ trình phát triển rõ ràng hơn.'

► **Lý do khách quan:** Công ty cũ tái cấu trúc, thu hẹp quy mô, dự án kết thúc.

► **Vị trí địa lý:** Chuyển nhà, gần gia đình hơn.

★ **Tip:** Trung thực nhưng **khéo léo** - kể đủ ý nhưng không đi vào chi tiết tiêu cực.

Câu 4. Bạn đã làm những công việc gì ở công ty cũ? Mô tả về dự án bạn từng tham gia?

◆ Gợi ý cách trả lời:

Đây là câu hỏi kiểm tra kinh nghiệm thực tế - hãy chuẩn bị câu trả lời theo cấu trúc STAR.

Cấu trúc STAR:

► **S - Situation:** Tên/loại dự án, lĩnh vực (banking, e-commerce, healthcare...), quy mô.

► **T - Task:** Vai trò của bạn trong dự án, nhiệm vụ chính.

► **A - Action:** Cụ thể bạn đã làm gì, dùng công cụ/kỹ thuật nào.

► **R - Result:** Kết quả đạt được, đóng góp nổi bật.

Ví dụ câu trả lời mẫu:

'Ở công ty cũ em tham gia dự án XYZ - một hệ thống e-commerce cho khoảng 50,000 người dùng. Team gồm 8 người (5 Dev, 2 Tester, 1 BA). Vai trò của em là Manual Tester cho module Payment.

Công việc cụ thể của em:

► Phân tích requirement từ BA, đặt câu hỏi làm rõ.

► Viết khoảng 200+ test case cho module Payment.

► Thực thi test trên web và mobile app (iOS/Android).

► Log gần 80 bug, trong đó có 15 bug critical liên quan đến luồng thanh toán.

► Tham gia daily standup, sprint review, retrospective theo Scrum.

► Hỗ trợ regression test và UAT.'

★ **Tip:** Càng có **con số cụ thể** càng thuyết phục. Tránh nói chung chung 'em làm Tester thôi'.

Câu 5. Điểm mạnh và điểm yếu của bạn là gì?

◆ Gợi ý cách trả lời:

Câu hỏi 'kinh điển' - cần chuẩn bị câu trả lời thông minh.

PHẦN ĐIỂM MẠNH (chọn 2-3 điểm liên quan đến Tester):

► **Tỉ mỉ, chú trọng chi tiết:** 'Em là người cẩn thận, hay để ý đến những chi tiết nhỏ - phù hợp với việc tìm bug.'

► **Tư duy phân tích, logic:** 'Em thích đặt câu hỏi WHY, WHAT IF khi đối mặt với vấn đề.'

► **Giao tiếp tốt:** 'Em làm việc tốt với cả Dev và BA.'

► **Tinh thần học hỏi:** 'Em chủ động tự học - hiện đang học automation qua Udemy.'

► **Kiên nhẫn:** 'Em không nản khi phải lặp lại test case hay tái hiện bug khó.'

Mỗi điểm mạnh nên kèm 1 ví dụ thực tế chứng minh.

PHẦN ĐIỂM YẾU (quan trọng và khó hơn):

► **TRÁNH:** Câu trả lời sáo rỗng như 'em là người cầu toàn quá mức' - ai cũng dùng, nhà tuyển dụng nghe nhàm.

► **TRÁNH:** Điểm yếu chí mạng cho nghề Tester như 'em không tỉ mỉ' hoặc 'em ngại đọc tài liệu'.

► **NÊN:** Chọn điểm yếu thật nhưng đang cải thiện, không nghiêm trọng:

• 'Em còn yếu về automation testing - hiện đang tự học Selenium qua khóa online và làm dự án cá nhân.'

• 'Tiếng Anh giao tiếp của em chưa tốt - em đang tham gia lớp speaking 3 buổi/tuần.'

• 'Em đôi khi quá chi tiết, mất nhiều thời gian - em đang học cách prioritize tốt hơn.'

★ **Tip:** Điểm yếu PHẢI kèm **hành động cụ thể** đang thực hiện để cải thiện - thể hiện tinh thần cầu tiến.

Câu 6. Mục tiêu nghề nghiệp 3-5 năm tới của bạn là gì?

◆ Gợi ý cách trả lời:

Câu hỏi đánh giá sự nghiêm túc và định hướng dài hạn.

Lộ trình tham khảo theo từng giai đoạn:

► **Ngắn hạn (6 tháng - 1 năm):**

• Thành thạo manual testing trên dự án thực tế.

• Làm quen automation testing với 1 framework (Selenium/Cypress).

• Đóng góp giá trị cho team, hoàn thành xuất sắc các task được giao.

► **Trung hạn (2-3 năm):**

• Lên vị trí Junior/Middle Tester.

• Thành thạo 1-2 framework automation.

• Có thể tự dẫn dắt test cho 1 module/feature lớn.

• Có chứng chỉ ISTQB Foundation Level.

► **Dài hạn (5 năm):**

• Trở thành Senior Tester / Test Lead.

• Hoặc chuyên sâu một mảng (Performance, Security, Automation Architect).

• Mentor cho các Junior Tester.

★ **Tip phỏng vấn:**

► Mục tiêu phải **phù hợp với lộ trình công ty đang tuyển**.

► **TRÁNH** nói 'em muốn chuyển sang Dev' hoặc 'em muốn tự kinh doanh sau 3 năm' - công ty sẽ ngại đầu tư đào tạo.

► **TRÁNH** trả lời mơ hồ 'em muốn phát triển bản thân' - hãy cụ thể.

Câu 7. Tại sao bạn muốn làm việc tại công ty chúng tôi?

◆ Gợi ý cách trả lời:

Câu hỏi đánh giá mức độ tìm hiểu và động lực thực sự của ứng viên.

BƯỚC CHUẨN BỊ TRƯỚC PHỎNG VẤN:

► Đọc kỹ website, fanpage công ty.

► Xem review trên Glassdoor, ITviec, LinkedIn.

► Tìm hiểu sản phẩm/dịch vụ, khách hàng chính, công nghệ sử dụng.

► Đọc kỹ JD để hiểu vị trí cần gì.

CÔNG THỨC TRẢ LỜI:

1. Lý do liên quan đến công ty:

► 'Em ấn tượng với sản phẩm X của công ty vì... (nêu cụ thể).'

► 'Em đọc được công ty đang phát triển mạnh trong mảng Y - đúng định hướng em muốn theo đuổi.'

► 'Em thấy văn hóa công ty (qua reviews, social media) phù hợp với tính cách của em.'

2. Lý do liên quan đến vị trí:

► 'JD đề cập đến X, Y, Z - đúng với những gì em đã chuẩn bị và muốn làm.'

► 'Em muốn được học hỏi từ team senior có kinh nghiệm về Z.'

3. Lý do liên quan đến bản thân:

► 'Em tin với kỹ năng A, B của em và môi trường công ty, em có thể đóng góp giá trị và phát triển.'

★ TUYỆT ĐỐI TRÁNH:

► 'Vì công ty trả lương cao'

► 'Vì công ty gần nhà em'

► 'Vì em đang cần một công việc'

► 'Em không biết, em chỉ apply nhiều nơi'

Câu 8. Mức lương mong muốn của bạn là bao nhiêu?

◆ Gợi ý cách trả lời:

Câu hỏi nhạy cảm - cần chuẩn bị kỹ trước phỏng vấn.

BƯỚC 1 - Research thị trường trước:

► TopCV, VietnamWorks, ITviec - xem mức lương cho vị trí tương đương.

► Hỏi bạn bè, mentor, anh chị đi trước.

► Mức tham khảo cho Fresher Tester tại HCM/HN (cập nhật theo thời điểm):

• Manual Tester: 8-12 triệu/tháng.

• Có biết Automation: 10-15 triệu/tháng.

• Có chứng chỉ ISTQB + dự án thực tế: 12-16 triệu/tháng.

BƯỚC 2 - Cách trả lời thông minh:

Cách 1 - Đưa range:

'Dựa trên research thị trường và những gì em có thể đóng góp, em mong muốn mức lương từ X-Y triệu. Tuy nhiên, em open để thương lượng dựa trên benefit tổng thể và lộ trình phát triển công ty đưa ra.'

Cách 2 - Hỏi ngược (an toàn hơn cho Fresher):

'Là Fresher, em đặt cơ hội học hỏi lên hàng đầu. Em muốn được biết range lương công ty dành cho vị trí này, em tin nó sẽ phù hợp với năng lực và đóng góp của em.'

Cách 3 - Dựa trên offer trước:

'Ở vị trí trước em nhận X triệu. Mong muốn ở vị trí mới em được tăng khoảng 15-20% để phù hợp với scope công việc và năng lực hiện tại.'

★ Tip quan trọng:

► **TRÁNH** nói số quá thấp - tự đánh giá thấp bản thân, dễ bị ép giá lâu dài.

► **TRÁNH** nói số quá cao - thiếu thực tế, bị loại ngay vòng đầu.

► Luôn nhấn mạnh sự linh hoạt và tinh thần học hỏi.

► Đừng nói 'em không quan tâm lương' - không thực tế.

Câu 9. Bạn xử lý áp lực công việc và deadline gấp như thế nào?

◆ Gợi ý cách trả lời:

Câu hỏi đánh giá khả năng quản lý bản thân và làm việc dưới áp lực.

TRÁNH:

► 'Em không bao giờ thấy áp lực' - không thực tế, thiếu thuyết phục.

► 'Em sẽ làm tăng ca đến khi xong' - thể hiện không có phương pháp.

► 'Em sẽ cố hết sức' - mơ hồ.

NÊN: Trình bày phương pháp cụ thể.

► Lập kế hoạch rõ ràng:

• Liệt kê tất cả task cần làm.

• Phân loại theo ma trận Eisenhower (Quan trọng/Khẩn cấp).

• Ước lượng thời gian cho từng task.

► Giao tiếp chủ động:

• Báo cáo sớm với Lead/PM nếu thấy không kịp deadline.

• Đề xuất giải pháp - không chỉ kể vấn đề.

• Xin hỗ trợ khi cần thay vì cố gắng một mình.

► Quản lý thời gian hiệu quả:

• Sử dụng kỹ thuật Pomodoro (25 phút focus + 5 phút nghỉ).

• Tránh đa nhiệm (multitasking) - tập trung một việc tại một thời điểm.

• Tắt thông báo không cần thiết khi làm việc quan trọng.

► Chăm sóc bản thân:

• Ngủ đủ giấc - thiếu ngủ làm giảm năng suất.

• Tập thể dục ít nhất 30 phút/ngày.

• Không bỏ bữa, uống đủ nước.

• Có hobby/sở thích để giải tỏa.

★ **Tip:** Kèm 1 ví dụ thực tế bạn đã từng vượt qua áp lực sẽ rất thuyết phục. Ví dụ: 'Trong đợt làm đồ án tốt nghiệp song song với học lái xe và đi làm thêm, em đã áp dụng cách trên và hoàn thành tất cả đúng hạn.'

Câu 10. Bạn có câu hỏi nào dành cho chúng tôi không?

◆ Gợi ý cách trả lời:

Câu hỏi cuối buổi phỏng vấn - QUAN TRỌNG hơn bạn nghĩ!

TUYỆT ĐỐI ĐỪNG nói 'Em không có câu hỏi gì' - đây là sai lầm phổ biến của Fresher, thể hiện bạn không thực sự quan tâm đến công ty/vị trí.

CHUẨN BỊ TRƯỚC 3-5 CÂU HỎI THÔNG MINH:

► Về công việc cụ thể:

• 'Quy trình test của team đang được tổ chức như thế nào?'

• 'Hiện tỷ lệ manual/automation testing trong team ra sao?'

• 'Một ngày làm việc điển hình của vị trí này diễn ra thế nào?'

• 'Team đang sử dụng những công cụ và framework nào?'

► Về team và môi trường:

• 'Team Testing có bao nhiêu người? Cơ cấu cấp bậc thế nào?'

• 'Văn hóa làm việc của team có điểm gì đặc biệt?'

• 'Team có hoạt động chia sẻ kiến thức nội bộ không?'

► Về phát triển nghề nghiệp:

• 'Công ty có chương trình đào tạo nội bộ cho Tester không?'

• 'Lộ trình thăng tiến của một Tester ở công ty như thế nào?'

• 'Công ty có hỗ trợ chi phí học chứng chỉ như ISTQB không?'

► Về dự án và sản phẩm:

• 'Hiện công ty đang phát triển dự án/sản phẩm gì?'

• 'Tech stack và domain chính là gì?'

• 'Quy mô người dùng/khách hàng của sản phẩm?'

► Về quy trình tiếp theo:

• 'Sau buổi phỏng vấn này, các bước tiếp theo trong quy trình tuyển dụng là gì?'

• 'Khi nào em có thể nhận được phản hồi?'

★ TRÁNH hỏi ở vòng đầu:

► Lương, thưởng, OT, bảo hiểm (nên hỏi sau khi có offer hoặc cuối buổi nếu được mời hỏi).

► Ngày nghỉ, work-from-home.

► Những thông tin đã có sẵn trên website công ty.

★ **Tip vàng:** Mang sẵn 1 cuốn sổ nhỏ và bút - ghi chú câu trả lời của nhà tuyển dụng. Hành động này thể hiện sự chuyên nghiệp và quan tâm thực sự.

PHẦN II

CÂU HỎI LÝ THUYẾT

(20 câu)

Phần này gồm 20 câu hỏi về **kiến thức nền tảng** của Software Testing - khái niệm, định nghĩa, phân biệt thuật ngữ, các kỹ thuật và phương pháp test. Đây là phần kiểm tra **kiến thức cơ bản** mà bất cứ Tester nào cũng cần nắm vững.

Câu 11. Software Testing là gì? Tại sao cần kiểm thử phần mềm?

◆ Đáp án mẫu:

Định nghĩa: Software Testing (Kiểm thử phần mềm) là quá trình thực thi phần mềm nhằm phát hiện lỗi (bug/defect), xác minh phần mềm hoạt động đúng yêu cầu (verification) và đáp ứng nhu cầu người dùng (validation).

Mục đích chính:

► Phát hiện lỗi sớm để giảm chi phí sửa chữa.

► Đảm bảo chất lượng sản phẩm trước khi đến tay người dùng.

► Tăng độ tin cậy, an toàn và uy tín cho sản phẩm/doanh nghiệp.

► Đáp ứng đúng yêu cầu nghiệp vụ (business requirements).

★ **Mẹo trả lời:** Nên nhấn mạnh testing không chỉ là 'tìm bug' mà còn là **phòng ngừa lỗi** (defect prevention) thông qua review tài liệu, requirements ngay từ đầu dự án.

Câu 12. Phân biệt QA, QC và Tester?

◆ Đáp án mẫu:

QA (Quality Assurance) - Đảm bảo chất lượng: Tập trung vào **quy trình**, định nghĩa chuẩn, hướng dẫn, audit để phòng ngừa lỗi. QA mang tính chủ động (proactive).

QC (Quality Control) - Kiểm soát chất lượng: Tập trung vào **sản phẩm**, kiểm tra sản phẩm có đạt chuẩn không, phát hiện lỗi đã có. QC mang tính bị động (reactive).

Tester: Là người thực thi việc kiểm thử (test execution) - viết test case, chạy test, log bug. Tester là một phần của QC.

★ **Tóm gọn:** QA = phòng bệnh, QC = chữa bệnh, Tester = bác sĩ thực hiện chẩn đoán.

Câu 13. SDLC và STLC là gì? Có gì khác nhau?

◆ Đáp án mẫu:

SDLC (Software Development Life Cycle): Vòng đời phát triển phần mềm, gồm các giai đoạn: Requirement → Design → Development → Testing → Deployment → Maintenance.

STLC (Software Testing Life Cycle): Vòng đời kiểm thử phần mềm, gồm: Requirement Analysis → Test Planning → Test Case Design → Test Environment Setup → Test Execution → Test Closure.

Khác biệt cốt lõi: STLC là một **phần** nằm trong SDLC, chuyên về testing. SDLC bao quát toàn bộ quá trình tạo ra sản phẩm phần mềm.

Câu 14. Phân biệt Verification và Validation?

◆ Đáp án mẫu:

Verification (Xác minh): 'Are we building the product right?' - Kiểm tra phần mềm có được xây dựng đúng theo tài liệu/đặc tả không. Là **static testing** (review, walkthrough, inspection - không chạy code).

Validation (Xác nhận): 'Are we building the right product?' - Kiểm tra phần mềm có đáp ứng đúng nhu cầu thực tế của người dùng không. Là **dynamic testing** (chạy phần mềm thật để test).

★ **Ví dụ dễ hiểu:** Xây nhà - Verification là kiểm tra xây có đúng bản vẽ không; Validation là hỏi gia chủ nhà này ở có thoải mái, đáp ứng nhu cầu không.

Câu 15. Test Case là gì? Một Test Case tốt gồm những thành phần nào?

◆ Đáp án mẫu:

Test Case: Là tập hợp các bước thực hiện, dữ liệu đầu vào, điều kiện tiên quyết và kết quả mong đợi nhằm kiểm tra một chức năng cụ thể của phần mềm.

Thành phần cơ bản của một Test Case:

► **Test Case ID:** Mã định danh (vd: TC_LOGIN_001)

► **Title/Description:** Tiêu đề mô tả ngắn gọn

► **Pre-condition:** Điều kiện tiên quyết

► **Test Steps:** Các bước thực hiện

► **Test Data:** Dữ liệu đầu vào

► **Expected Result:** Kết quả mong đợi

► **Actual Result:** Kết quả thực tế

► **Status:** Pass/Fail/Blocked

► **Priority/Severity:** Mức độ ưu tiên/nghiêm trọng

Câu 16. Phân biệt Test Case và Test Scenario?

◆ Đáp án mẫu:

Test Scenario (Kịch bản kiểm thử): Là một ý tưởng tổng quát về **cái gì** cần test (What to test). Ví dụ: 'Kiểm tra chức năng đăng nhập'.

Test Case (Trường hợp kiểm thử): Là tập hợp các bước chi tiết **làm thế nào** để test (How to test). Ví dụ: 'TC1: Đăng nhập với username/password hợp lệ → vào trang Home thành công'.

Quan hệ: Một Test Scenario có thể bao gồm nhiều Test Case. Scenario ở cấp độ cao (high-level), Test Case ở cấp độ chi tiết (low-level).

Câu 17. Phân biệt Bug, Defect, Error, Failure?

◆ Đáp án mẫu:

Error (Lỗi): Sai sót do con người tạo ra trong quá trình code hoặc thiết kế (developer mistake).

Defect/Bug: Là kết quả của Error - sự sai khác giữa kết quả thực tế và kết quả mong đợi, được Tester phát hiện trong giai đoạn testing.

Failure: Khi defect được phát hiện ở môi trường **production** bởi end-user, làm hệ thống không hoạt động đúng.

★ **Lưu ý phỏng vấn:** Bug và Defect thường được dùng thay thế nhau, nhưng về lý thuyết Bug = lỗi không chính thức, Defect = lỗi đã được xác nhận chính thức.

Câu 18. Severity và Priority khác nhau như thế nào?

◆ Đáp án mẫu:

Severity (Mức độ nghiêm trọng): Ảnh hưởng kỹ thuật của bug lên hệ thống. Do Tester quyết định. Mức: Critical / High / Medium / Low.

Priority (Mức độ ưu tiên): Mức độ cần xử lý bug nhanh hay chậm theo nghiệp vụ. Do PM/PO/Dev Lead quyết định. Mức: P1/P2/P3/P4.

Các tổ hợp ví dụ:

► **High Severity + High Priority:** Hệ thống crash khi user login (nghiêm trọng + cần fix gấp).

► **High Severity + Low Priority:** Crash ở tính năng ít dùng, sắp bị loại bỏ.

► **Low Severity + High Priority:** Sai chính tả tên công ty trên trang chủ (ảnh hưởng hình ảnh).

► **Low Severity + Low Priority:** Lỗi nhỏ về màu sắc trong popup phụ.

Câu 19. Các Level of Testing (Cấp độ kiểm thử) gồm những gì?

◆ Đáp án mẫu:

1. Unit Testing: Kiểm thử từng module/đơn vị code nhỏ nhất. Do **Developer** thực hiện.

2. Integration Testing: Kiểm thử khi tích hợp các module với nhau, kiểm tra dữ liệu/giao tiếp giữa chúng.

3. System Testing: Kiểm thử toàn bộ hệ thống end-to-end sau khi tích hợp, do **Tester** thực hiện.

4. UAT (User Acceptance Testing): Kiểm thử bởi khách hàng/end-user để xác nhận sản phẩm đáp ứng yêu cầu thực tế.

★ **Mô hình tham khảo:** V-Model thể hiện rõ mỗi level testing tương ứng với một giai đoạn phát triển.

Câu 20. Black Box, White Box và Gray Box Testing là gì?

◆ Đáp án mẫu:

Black Box Testing (Kiểm thử hộp đen): Tester **không biết** code bên trong, chỉ tập trung vào input/output theo specification. Phù hợp cho Functional, System, UAT testing.

White Box Testing (Kiểm thử hộp trắng): Tester **biết và xem code**, kiểm thử dựa trên cấu trúc nội bộ, luồng logic, điều kiện. Thường do Dev thực hiện (Unit test).

Gray Box Testing (Kiểm thử hộp xám): Kết hợp cả 2 - tester biết một phần cấu trúc nội bộ (vd: database schema, API) nhưng test ở mức chức năng. Phù hợp cho Integration testing.

★ **Gợi nhớ:** Black = không thấy gì, White = thấy hết, Gray = thấy lờ mờ.

Câu 21. Functional Testing và Non-Functional Testing khác nhau ra sao?

◆ Đáp án mẫu:

Functional Testing (Kiểm thử chức năng): Kiểm tra phần mềm **làm gì** (what) - các chức năng có hoạt động đúng requirement không. Ví dụ: Login, Search, Payment...

Non-Functional Testing (Kiểm thử phi chức năng): Kiểm tra phần mềm **hoạt động như thế nào** (how) - về performance, usability, security, compatibility...

Các loại Non-Functional phổ biến:

► **Performance Testing:** Tốc độ, độ ổn định.

► **Load Testing:** Hệ thống chịu được bao nhiêu user đồng thời.

► **Stress Testing:** Đẩy hệ thống đến giới hạn.

► **Security Testing:** Bảo mật.

► **Usability Testing:** Trải nghiệm người dùng.

Câu 22. Smoke Testing và Sanity Testing khác nhau như thế nào?

◆ Đáp án mẫu:

Smoke Testing: Test nhanh các **chức năng chính** để xem build có 'sống' không, có đủ ổn định để test sâu hơn không. Thực hiện sau mỗi build mới. Mang tính **rộng nhưng nông**.

Sanity Testing: Test một **chức năng cụ thể** sau khi vừa có thay đổi nhỏ hoặc fix bug, để xác nhận thay đổi đó hoạt động đúng. Mang tính **hẹp nhưng sâu**.

★ So sánh nhanh:

► Smoke = 'Build có chạy được không?'

► Sanity = 'Fix mới có work không, có ảnh hưởng gì không?'

► Smoke thường được tự động hóa, Sanity thường làm thủ công.

Câu 23. Regression Testing là gì? Khi nào cần thực hiện?

◆ Đáp án mẫu:

Regression Testing (Kiểm thử hồi quy): Kiểm thử lại các chức năng **đã hoạt động ổn định trước đó** sau khi có thay đổi (fix bug, thêm tính năng, refactor code) để đảm bảo những thay đổi không gây ra lỗi mới ở các phần khác.

Khi nào cần thực hiện:

► Sau khi fix bug.

► Sau khi thêm tính năng mới.

► Sau khi tích hợp với hệ thống khác.

► Sau khi nâng cấp môi trường/version.

★ **Best practice:** Vì test lặp đi lặp lại nhiều lần nên **nên tự động hóa** (automation testing) các test case regression để tiết kiệm thời gian.

Câu 24. Re-testing và Regression Testing khác nhau ở điểm nào?

◆ Đáp án mẫu:

Re-testing (Confirmation Testing): Test lại **chính xác bug đã được fix** để xác nhận bug đã được giải quyết. Chỉ test trên các test case đã fail trước đó.

Regression Testing: Test các **chức năng liên quan/đã hoạt động** để xem việc fix có gây lỗi gì khác không.

★ Khác biệt khác:

► Re-testing: dùng cùng data, cùng môi trường. Regression: có thể dùng data khác.

► Re-testing: không thể tự động hóa hoàn toàn (bug mới). Regression: nên automation.

► Re-testing có Priority cao hơn Regression.

Câu 25. Bug Life Cycle (Vòng đời của một Bug) gồm những trạng thái nào?

◆ Đáp án mẫu:

Vòng đời bug cơ bản:

► **New:** Bug vừa được Tester log lên.

► **Assigned:** Bug được gán cho Dev xử lý.

► **Open:** Dev xác nhận và đang xem xét.

► **Fixed:** Dev đã sửa xong và đẩy build mới.

► **Pending Retest:** Chờ Tester test lại.

► **Retest:** Tester đang test lại.

► **Verified:** Bug đã được sửa thành công.

► **Closed:** Bug đóng chính thức.

► **Reopen:** Bug fix chưa thành công, mở lại.

Các trạng thái phụ: Rejected (Dev từ chối), Deferred (hoãn xử lý), Duplicate (trùng), Not a Bug (không phải bug).

Câu 26. Một Bug Report tốt cần có những thông tin gì?

◆ Đáp án mẫu:

Các thành phần thiết yếu của Bug Report:

► **Bug ID:** Mã định danh duy nhất

► **Title/Summary:** Mô tả ngắn gọn (ai cũng hiểu được)

► **Environment:** Browser, OS, Device, Version

► **Steps to Reproduce:** Các bước tái hiện chi tiết

► **Expected Result:** Kết quả mong đợi

► **Actual Result:** Kết quả thực tế

► **Severity và Priority**

► **Screenshots/Video/Log:** Bằng chứng

► **Assignee:** Người được giao xử lý

► **Reporter:** Người báo lỗi

★ **Tip:** Title bug nên theo công thức: *[Module] - Hành động - Hiện tượng*. Ví dụ: '[Login] - Đăng nhập sai password 5 lần - Hệ thống không khóa tài khoản'.

Câu 27. Boundary Value Analysis (BVA) và Equivalence Partitioning (EP) là gì?

◆ Đáp án mẫu:

Equivalence Partitioning (EP) - Phân vùng tương đương: Chia input thành các nhóm có hành vi tương tự, mỗi nhóm chỉ cần chọn 1 giá trị đại diện để test.

Boundary Value Analysis (BVA) - Phân tích giá trị biên: Test các giá trị ở biên (min, min-1, min+1, max, max-1, max+1) vì lỗi thường xảy ra ở biên.

★ **Ví dụ minh họa:** Trường nhập tuổi hợp lệ từ 18-60:

► **EP:** 3 vùng - dưới 18 (vd: 10), trong khoảng 18-60 (vd: 30), trên 60 (vd: 70).

► **BVA:** Test các giá trị 17, 18, 19, 59, 60, 61.

★ **Best practice:** Kết hợp cả 2 kỹ thuật để có bộ test case tối ưu - phủ rộng (EP) và phủ sâu ở điểm rủi ro (BVA).

Câu 28. Test Plan và Test Strategy khác nhau như thế nào?

◆ Đáp án mẫu:

Test Plan: Tài liệu chi tiết cho một **dự án/release cụ thể**: phạm vi, tài nguyên, lịch trình, deliverables, môi trường test, tiêu chí entry/exit. Do **Test Lead/Manager** viết.

Test Strategy: Tài liệu ở **cấp tổ chức/công ty**, định hướng cách tiếp cận testing chung: chuẩn mực, công cụ, quy trình. Áp dụng cho nhiều dự án. Do **Quản lý cấp cao** viết.

★ So sánh:

► Test Plan = cụ thể, cho 1 dự án, có thể thay đổi.

► Test Strategy = tổng quát, cho cả tổ chức, ít thay đổi.

► Test Strategy thường là một phần (subset) hoặc tài liệu tham chiếu của Test Plan.

Câu 29. Manual Testing và Automation Testing - khi nào nên dùng cái nào?

◆ Đáp án mẫu:

Manual Testing - Nên dùng khi:

► Test UI/UX, trải nghiệm người dùng.

► Exploratory Testing (kiểm thử thăm dò).

► Ad-hoc Testing.

► Dự án ngắn hạn, requirements thay đổi thường xuyên.

► Test case chạy ít lần, khó tự động hóa.

Automation Testing - Nên dùng khi:

► Regression Testing (chạy lặp lại nhiều lần).

► Smoke Testing trên mỗi build mới.

► Performance, Load, Stress Testing.

► Test case lặp đi lặp lại, ổn định, ít thay đổi.

► Dữ liệu lớn, cần độ chính xác cao.

★ **Kết luận:** Không cái nào thay thế cái nào hoàn toàn. Dự án thực tế thường kết hợp **~70% Automation + 30% Manual** tùy đặc thù.

Câu 30. Là Fresher Tester, bạn sẽ làm gì khi gặp Dev không thừa nhận đó là bug?

◆ Đáp án mẫu:

Đây là câu hỏi tình huống rất phổ biến. Cách trả lời tham khảo:

Bước 1 - Bình tĩnh, không tranh cãi: Tôi sẽ giữ thái độ chuyên nghiệp, không cá nhân hóa vấn đề. Mục tiêu của cả 2 là chất lượng sản phẩm, không phải 'thắng/thua'.

Bước 2 - Kiểm tra lại bằng chứng: Tôi sẽ tái hiện lại bug, chuẩn bị đầy đủ steps to reproduce, screenshot, video, log để chứng minh đây là một bug thực sự.

Bước 3 - Đối chiếu requirement/specification: Mở tài liệu yêu cầu hoặc user story để chỉ rõ hành vi hiện tại sai khác với yêu cầu ở điểm nào.

Bước 4 - Thảo luận trực tiếp: Trao đổi cùng Dev, lắng nghe góc nhìn của họ. Có thể đây là behavior đúng nhưng tôi hiểu sai requirement.

Bước 5 - Leo thang khi cần: Nếu vẫn không thống nhất, mời BA/PO/Test Lead vào để có quyết định cuối cùng dựa trên business logic.

★ **Tip phỏng vấn:** Nhà tuyển dụng đánh giá cao **thái độ chuyên nghiệp, kỹ năng giao tiếp, dựa trên bằng chứng và requirement** - chứ không phải khả năng cãi thắng Dev.

PHẦN III

CÂU HỎI TÌNH HUỐNG

(10 câu)

Phần này gồm 10 câu hỏi **tình huống thực tế** - kiểm tra cách bạn xử lý các vấn đề phát sinh trong dự án: áp lực deadline, giao tiếp với Dev, xử lý bug Critical, quản lý ưu tiên... Đây là phần đánh giá **kỹ năng mềm và tư duy nghề nghiệp** - thường quyết định việc bạn có pass phỏng vấn hay không.

Câu 31. Dự án sắp release trong 2 ngày nhưng còn rất nhiều test case chưa chạy. Bạn xử lý thế nào?

◆ Gợi ý cách trả lời:

Đây là tình huống thực tế thường gặp - kiểm tra kỹ năng quản lý thời gian và ưu tiên công việc của Tester.

Bước 1 - Đánh giá tình hình: Tôi sẽ rà soát danh sách test case còn lại, ước lượng thời gian cần thiết để chạy hết và xác định mức độ rủi ro.

Bước 2 - Báo cáo minh bạch với Team Lead/PM: Không che giấu mà thẳng thắn báo cáo: số test case đã chạy, còn lại bao nhiêu, rủi ro nếu release đúng hạn. Đây là trách nhiệm của Tester.

Bước 3 - Áp dụng Risk-Based Testing: Ưu tiên test các phần quan trọng nhất:

► **P1 - Phải test:** Critical path (luồng nghiệp vụ chính), tính năng mới, phần có nhiều bug trước đó.

► **P2 - Nên test:** Smoke test toàn bộ hệ thống, regression các module liên quan.

► **P3 - Có thể hoãn:** Test case ở module ít người dùng, đã ổn định lâu dài.

Bước 4 - Đề xuất giải pháp: Có thể đề xuất hoãn release, thêm người hỗ trợ test, hoặc release với phạm vi giới hạn (soft launch).

★ **Tip:** Câu trả lời tốt thể hiện được **tư duy ưu tiên (prioritization)** và **tinh thần trách nhiệm** - không chỉ 'cố gắng chạy hết'.

Câu 32. Bạn phát hiện một bug nghiêm trọng (Critical) ngay trước giờ release. Bạn sẽ làm gì?

◆ Gợi ý cách trả lời:

Tình huống áp lực cao - kiểm tra khả năng xử lý khủng hoảng và giao tiếp.

Bước 1 - Xác nhận lại bug: Tái hiện bug ít nhất 2-3 lần để chắc chắn, kiểm tra trên môi trường giống production nhất. Tránh báo bug giả gây hoảng loạn không cần thiết.

Bước 2 - Thu thập thông tin đầy đủ: Steps to reproduce rõ ràng, screenshot/video, log file, môi trường, dữ liệu test - tất cả phải sẵn sàng để Dev có thể fix ngay.

Bước 3 - Báo cáo NGAY LẬP TỨC: Liên hệ trực tiếp (gọi điện/Slack/Teams) với Team Lead, PM, Dev Lead - không chỉ log Jira rồi ngồi đợi.

Bước 4 - Đề xuất phương án:

► Hoãn release để fix bug.

► Release với workaround tạm thời (nếu có).

► Rollback nếu đã release.

► Disable tính năng có bug, release các phần còn lại.

Bước 5 - Hỗ trợ verify sau khi fix: Sẵn sàng test lại ngay khi Dev fix xong, làm regression nhanh để đảm bảo không ảnh hưởng các phần khác.

★ **Điểm cộng:** Thể hiện **tinh thần chủ động (proactive)**, không né tránh trách nhiệm dù phát hiện bug ở phút chót.

Câu 33. Yêu cầu (requirement) mơ hồ hoặc không rõ ràng, bạn xử lý thế nào để viết test case?

◆ Gợi ý cách trả lời:

Đây là tình huống RẤT phổ biến với Fresher - kiểm tra khả năng giao tiếp và phân tích.

KHÔNG NÊN làm: Đoán mò requirement rồi viết test case dựa trên giả định cá nhân → dẫn đến test sai, miss case.

NÊN làm:

Bước 1 - Liệt kê các điểm chưa rõ: Đọc kỹ requirement, ghi chú lại từng điểm mơ hồ, đặt câu hỏi cụ thể (vd: 'Khi user nhập password sai 5 lần thì khóa bao lâu? Có gửi mail thông báo không?').

Bước 2 - Hỏi BA/PO/Khách hàng: Đưa câu hỏi rõ ràng, có thể đề xuất phương án để họ confirm. Tránh hỏi chung chung 'Cái này làm sao?'.

Bước 3 - Tham khảo nguồn khác:

► Tài liệu thiết kế UX/UI (mockup, prototype).

► Behavior của các sản phẩm tương tự trên thị trường.

► Hỏi Dev xem họ đã code theo logic nào.

Bước 4 - Document lại quyết định: Sau khi có câu trả lời, ghi chú vào test case hoặc Confluence để cả team thống nhất, tránh tranh cãi về sau.

Bước 5 - Áp dụng kỹ thuật Exploratory Testing: Trong lúc chờ confirm, tự khám phá hệ thống để hiểu rõ hơn, từ đó đặt câu hỏi chính xác hơn.

★ **Tip:** Câu trả lời thể hiện bạn **chủ động giao tiếp** chứ không thụ động chờ đợi - đây là kỹ năng quan trọng của Tester.

Câu 34. Khi không có tài liệu requirement, bạn sẽ test phần mềm như thế nào?

◆ Gợi ý cách trả lời:

Tình huống thực tế ở nhiều dự án nhỏ/startup - kiểm tra tư duy linh hoạt.

Bước 1 - Tìm kiếm các nguồn thông tin thay thế:

► **User Story/Jira tickets:** Mô tả ngắn về tính năng.

► **Mockup/Wireframe:** Thiết kế UI giúp hiểu luồng.

► **Email/Slack chat:** Trao đổi giữa BA-Dev-PM.

► **Source code/Code commits:** Đọc commit message, comment.

► **Database schema:** Hiểu cấu trúc dữ liệu.

Bước 2 - Trao đổi trực tiếp với stakeholders: Phỏng vấn nhanh PO, Dev, end-user để hiểu mục đích nghiệp vụ. Ghi lại bằng văn bản.

Bước 3 - Áp dụng các kỹ thuật testing không cần requirement:

► **Exploratory Testing:** Khám phá hệ thống, học và test cùng lúc.

► **Ad-hoc Testing:** Test ngẫu nhiên dựa trên kinh nghiệm.

► **Error Guessing:** Đoán các lỗi thường gặp dựa trên kinh nghiệm.

► **Tham khảo benchmark:** So sánh với sản phẩm cùng lĩnh vực.

Bước 4 - Tự xây dựng requirement từ thực tế: Vừa test vừa document lại behavior hiện tại để làm tài liệu tham khảo cho team.

★ **Tip:** Đề xuất với team thiết lập quy trình tài liệu hóa tối thiểu cho các tính năng quan trọng - thể hiện tư duy **cải tiến quy trình**.

Câu 35. Một bug bạn báo lên đã bị Dev đóng với lý do 'Not a Bug' nhưng bạn vẫn tin đó là bug. Bạn làm gì?

◆ Gợi ý cách trả lời:

Tình huống kiểm tra kỹ năng giao tiếp, kiên định nhưng không cứng nhắc.

Bước 1 - Xem lại comment của Dev: Đọc kỹ lý do Dev đóng bug. Đôi khi Dev có thông tin mà bạn chưa biết (vd: behavior này đã được PO confirm).

Bước 2 - Tự kiểm tra lại: Test lại theo các steps, kiểm tra trên môi trường khác nhau. Có thể bug đã được fix mà bạn chưa cập nhật build mới.

Bước 3 - Đối chiếu với tài liệu: Mở requirement, user story, mockup để xác nhận bạn đúng hay sai. Đây là 'bằng chứng' khách quan nhất.

Bước 4 - Reopen bug với bằng chứng đầy đủ: Nếu chắc chắn mình đúng, reopen bug và **thêm thông tin chi tiết**:

► Trích dẫn cụ thể từ requirement (số dòng, mục).

► Video tái hiện bug rõ ràng.

► Giải thích tại sao đây là bug từ góc nhìn user.

Bước 5 - Trao đổi trực tiếp: Không chỉ comment qua lại trên Jira mà chat/họp ngắn với Dev để hiểu góc nhìn của họ. Đôi khi 1 cuộc nói chuyện 5 phút giải quyết được tranh luận kéo dài cả tuần.

Bước 6 - Triệu tập bên thứ 3: Mời BA/PO/Test Lead vào quyết định cuối cùng dựa trên business value.

★ **Tip phỏng vấn:** Nhấn mạnh thái độ **'không cá nhân hóa - dựa trên bằng chứng - hướng đến chất lượng sản phẩm'**.

Câu 36. Bạn được giao test một module hoàn toàn mới mà bạn chưa từng biết. Bạn bắt đầu từ đâu?

◆ Gợi ý cách trả lời:

Tình huống đánh giá khả năng tự học và phương pháp tiếp cận công việc.

Bước 1 - Tìm hiểu nghiệp vụ (Business Domain):

► Đọc tài liệu mô tả module, user story.

► Hỏi BA/PO về mục đích và đối tượng người dùng.

► Tìm hiểu các thuật ngữ chuyên ngành.

Bước 2 - Tìm hiểu sản phẩm cùng lĩnh vực: Trải nghiệm các sản phẩm tương tự trên thị trường để hiểu luồng người dùng kỳ vọng.

Bước 3 - Sử dụng thử module: Đăng nhập, thực hiện các thao tác như một end-user. Đây là cách nhanh nhất để hiểu module.

Bước 4 - Trao đổi với Dev:

► Hỏi về kiến trúc tổng quan, các API, database liên quan.

► Hỏi về các edge case, các giả định kỹ thuật.

► Hỏi về các phần Dev nghĩ 'dễ phát sinh bug'.

Bước 5 - Xây dựng Test Plan dần:

► Vẽ Mind Map các tính năng cần test.

► Liệt kê Test Scenario tổng quan trước, chi tiết hóa sau.

► Bắt đầu từ Happy Path → Edge Case → Negative Case.

Bước 6 - Học từ regression suite cũ: Nếu có module tương tự đã test trước đó, đọc test case cũ để học pattern.

★ **Tip:** Thể hiện bạn là người **tự học, có phương pháp, biết khi nào cần hỏi** - không phải kiểu 'không hiểu thì hỏi từng cái nhỏ nhặt'.

Câu 37. Bạn vừa fix bug và verify Pass, nhưng sau khi release vào production, người dùng vẫn báo lỗi. Bạn xử lý thế nào?

◆ Gợi ý cách trả lời:

Tình huống áp lực cao - kiểm tra cách xử lý sự cố và tinh thần học hỏi.

Bước 1 - Không hoảng loạn, không đổ lỗi: Tập trung giải quyết vấn đề trước, phân tích nguyên nhân sau. Thái độ chuyên nghiệp.

Bước 2 - Thu thập thông tin từ production:

► Liên hệ Customer Support để lấy chi tiết: user nào, thao tác gì, lúc nào, browser/device gì.

► Lấy log production, screenshot, video từ user.

► Xác định scope ảnh hưởng: bao nhiêu user, có ảnh hưởng dữ liệu không.

Bước 3 - Tái hiện bug trên production-like environment: Cố gắng tái hiện chính xác để Dev có thể fix.

Bước 4 - Phối hợp xử lý:

► Báo cáo Team Lead/PM ngay lập tức.

► Hỗ trợ Dev fix nhanh nhất có thể.

► Thông báo Customer Support phương án tạm thời cho user.

Bước 5 - Phân tích nguyên nhân gốc (Root Cause Analysis):

► Tại sao test trên Staging Pass nhưng Prod Fail? (Khác data, khác config, khác môi trường?)

► Có miss test case nào không?

► Quy trình test có vấn đề gì cần cải tiến?

Bước 6 - Học hỏi và cải tiến:

► Thêm test case cho trường hợp này vào regression suite.

► Đề xuất cải tiến quy trình (test trên Prod-like data, tăng coverage, automation...).

► Document lại bài học vào lessons learned.

★ **Tip:** Thể hiện **tinh thần trách nhiệm** và **tư duy cầu tiến** - quan trọng hơn việc bào chữa.

Câu 38. Bạn nhận được build mới rất gấp cần test xong trong 2 tiếng để release. Bạn ưu tiên test gì?

◆ Gợi ý cách trả lời:

Tình huống thực tế trong hotfix - kiểm tra tư duy ưu tiên dưới áp lực thời gian.

Bước 1 - Hỏi rõ scope thay đổi: Trao đổi nhanh với Dev:

► Build này fix/sửa cái gì cụ thể?

► Code change ảnh hưởng đến module nào?

► Có config/database change không?

Bước 2 - Lên thứ tự ưu tiên test (60-90 phút đầu):

► **Smoke Test (15 phút):** Đảm bảo build chạy được, login, các luồng chính không bị crash.

► **Test trực tiếp bug đã fix (30 phút):** Verify fix có hoạt động đúng, test cả các biến thể của bug.

► **Regression có chọn lọc (30-45 phút):** Test các tính năng có khả năng bị ảnh hưởng bởi thay đổi (Impact Analysis).

Bước 3 - Sử dụng tài nguyên có sẵn:

► Chạy automation regression suite (nếu có) song song với manual test.

► Nhờ đồng nghiệp hỗ trợ test các module quan trọng.

Bước 4 - Báo cáo kết quả rõ ràng (15 phút cuối):

► Liệt kê những gì đã test, kết quả Pass/Fail.

► Nêu rõ những phần CHƯA test do giới hạn thời gian.

► Đánh giá mức độ rủi ro nếu release - đề xuất Go/No-Go.

★ **Tip:** Đừng nói 'em sẽ cố test hết' - hãy nói **'em sẽ test có ưu tiên và báo cáo minh bạch về rủi ro'**. Đây là tư duy của Tester chuyên nghiệp.

Câu 39. Đồng nghiệp Tester khác báo bug giống y hệt bug bạn đã báo cách đây 1 tuần. Bạn xử lý thế nào?

◆ Gợi ý cách trả lời:

Tình huống kiểm tra kỹ năng làm việc nhóm và xử lý tinh tế.

KHÔNG NÊN làm:

► Phản ứng gay gắt, chỉ trích đồng nghiệp 'không check trước khi báo'.

► Im lặng để bug duplicate tồn tại trên hệ thống tracking.

NÊN làm:

Bước 1 - Xác nhận bug có thực sự trùng không: Đọc kỹ cả 2 bug report. Có thể chỉ giống nhau bề ngoài nhưng nguyên nhân/môi trường khác nhau.

Bước 2 - Nếu là duplicate thật:

► Mark bug mới là 'Duplicate' và link đến bug gốc của bạn.

► Comment lịch sự thông báo cho đồng nghiệp biết và link đến bug gốc.

► Có thể chuyển thông tin bổ sung (nếu có) từ bug duplicate sang bug gốc.

Bước 3 - Trao đổi nhẹ nhàng với đồng nghiệp:

► Không trách móc, chia sẻ tips để check duplicate trước khi log bug (search keyword, filter theo module).

► Hỏi xem có cần hỗ trợ làm quen với hệ thống tracking không.

Bước 4 - Phản ánh vấn đề hệ thống (nếu lặp lại nhiều lần):

► Đề xuất team có quy trình rõ ràng về việc search bug trước khi log.

► Cải tiến cách đặt title bug để dễ search.

► Phân chia module test rõ ràng giữa các Tester.

★ **Tip:** Câu trả lời tốt thể hiện **tinh thần đồng đội, sự tinh tế trong giao tiếp** - không cạnh tranh hay đổ lỗi. Tester giỏi là người làm cho cả team tốt hơn.

Câu 40. Bạn không đồng ý với cách Test Lead phân chia công việc/giao test case. Bạn xử lý thế nào?

◆ Gợi ý cách trả lời:

Tình huống kiểm tra cách bạn xử lý mâu thuẫn với cấp trên - rất khó.

Nguyên tắc: Tôn trọng quyết định của Lead nhưng có quyền và trách nhiệm phản hồi mang tính xây dựng.

Bước 1 - Xác định rõ lý do mình không đồng ý:

► Lý do khách quan: khối lượng không hợp lý, năng lực chưa đủ, deadline không khả thi?

► Lý do chủ quan: thích/không thích module, ngại khó?

Nếu chỉ là lý do chủ quan, nên cố gắng thực hiện và học hỏi.

Bước 2 - Chọn thời điểm phù hợp: Không phản hồi ngay trong meeting đông người. Hẹn 1-1 trao đổi riêng với Test Lead để giữ thể diện cho cả 2 bên.

Bước 3 - Trao đổi mang tính xây dựng:

► Bắt đầu bằng việc thể hiện hiểu quyết định của Lead.

► Trình bày cụ thể vấn đề: 'Em thấy với khối lượng X test case và deadline Y, em sợ không đảm bảo chất lượng vì lý do Z.'

► **Đề xuất giải pháp thay thế:** 'Em đề xuất chia thành 2 đợt' hoặc 'Em xin được hỗ trợ phần này từ anh A'.

Bước 4 - Tôn trọng quyết định cuối cùng: Sau khi đã trình bày, nếu Lead vẫn giữ quyết định, hãy thực hiện hết khả năng. Lead có thể có thông tin/bối cảnh mà bạn chưa biết.

Bước 5 - Học hỏi từ trải nghiệm: Sau khi hoàn thành, đánh giá lại - quyết định của Lead có hợp lý không? Mình có thể làm tốt hơn lần sau ở điểm nào?

★ **Tip phỏng vấn:** Tránh nói 'em sẽ làm theo Lead bảo' (thụ động) hoặc 'em sẽ tranh luận đến cùng' (cứng đầu). Thái độ đúng là **'tôn trọng nhưng dám có ý kiến mang tính xây dựng'**.

LỜI KẾT

Cảm ơn bạn đã đồng hành cùng tài liệu này! Hy vọng **40 câu hỏi & đáp án mẫu** trên đây sẽ giúp bạn tự tin hơn trong buổi phỏng vấn vị trí Fresher/Junior Tester.

Lưu ý quan trọng: Đáp án mẫu chỉ mang tính tham khảo. Hãy diễn đạt theo cách của riêng bạn, gắn với trải nghiệm thực tế và những dự án bạn đã từng làm (kể cả dự án học tập, đồ án) để câu trả lời thêm sinh động và thuyết phục.

Đối với câu hỏi về bản thân, hãy luyện tập trước ở nhà để trả lời tự nhiên, không vấp. Với câu tình huống, nhà tuyển dụng quan tâm đến **quá trình tư duy** nhiều hơn là một đáp án hoàn hảo - hãy thể hiện cách bạn phân tích vấn đề, đưa ra giải pháp và sẵn sàng học hỏi.

✦ Chúc bạn phỏng vấn thành công!

Theo dõi Anh Tester tại: https://anhtester.com