SWT

SoftWare Testing

Maven:

Rest-api: hàm gọi qua internet, gọi qua url (phần mền Postman)

ci-Maven (Contineus Intervation):

Jmeter: giả lập 10 ngàn người vô web

BigData: một khối data siêu lớn để thu thập dữ liệu người dùng

Docker: công nghệ ảo hóa siêu tốn ram

Kafka: một công nghệ khác

Jira/Notion/Trello: quản lý các requirements/User Story

Slot 2:

Bug Pixel nút gửi mess của Apple

Tomcat 10: dùng jakarta. (tên thủ đô Indo), vẫn alf một app vẫn có nguy cơ lỗi như bình thường. ChangeLog: mô tả và ghi lai các bug của phần mền.

Bugzilla: tool dùng để lưu trữ lại toàn bộx bug

Unix: code/Keined and GUI(tách biệt còn win thì gộp)

MacOS kế thừa từ cái này

Linux: hosting nhanh và rẻ hơn

Windows

Youtrack.jetbrain

ChangeLog / Release Notes

IntelliJ: xịn nhất trong các tool viết Java

Bug từ đâu ra? Bug là gì?

I. Thuật ngữ chung (Terminology):

* Defect, bug, error, failure, mistake
* Bug: sai sót, khiếm khuyết, trục trặc tron phần mềm, và đc tìm thấy bởi dân kiểm thử phần mềm (Tester, QC (Quality Control))

II. ĐỊNH NGHĨA VỀ KIỂM THỬ PHẦN MỀM

Kiểm thử phần mềm là gì??

* 1. So sánh (Compare) giữa cái kì vòng (Expected) và cái thực tế (Actual) coi 2 thứ có khớp/giống nhau hay không, if expected == actual
  + VD: tính năng quản lý giỏ hàng:
    - cập nhật số lượng mua: nếu cập nhật số lượng mua = cách duyệt sản phẩm, nhấn [add to cart] thì giỏ hàng chưa có món này thì tự động add vào giỏ và số lượng là một còn nếu giở hàng đã có sẵn món này rồi thì món không được thêm vào nhưng lại tự tăng số lượng có sẵn lên 1 = số lượng +1
      * Expected: đã có 1 món trong giỏ, add thêm 1 nhát nữa thì số lượng phải bằng 2
      * App chạy thức tế để test, add 2 lần, -> số lượng bên giỏ đang là mấy – “là mấy” gọi là: Actual
      * Expected == Actual thì tính năng đã đúng còn ko thì có bug rồi
* 2. Tiến trình tìm ra những sai sót của app/ tìm bug. Làm thế nào để tìm ra đc bug, sai sót?? -> Mục 5 trong cây bản đồ tư duy
* 3. So sánh xem app có cài đặt đúng như thiết kế, đúng Requirements hay không? Requirements có đủ tốt hay không
  + LIÊN QUAN ĐẾN REQUIREMENTS, có 3 loại tài liệu ta cần chú ý:
    - SRS: Sortware Reqs Specification – bản đặc tả yêu cầu phần mềm (1/2 cuốn CP)
      * SWR ra đc cuốn này
    - FRS/FRD – Functional Reqs Specification/Document – bản đặc tả yêu cầu chứng năng
      * UI + mô tả các ô nhập + mô ta xử lí nút nhấn
    - Backlog (Agile): Nơi tập hợp các USER STORIES
      * As a …, I want to … so that …
      * AC: Acceptance Criteria – tiêu chí nghiệm thu tính năg
      * DoD: Definition of Done – định nghĩa chi tiết/ check list để nói rằng 1 được implement cần \_ phải có nhứng thứ sau
      * Lưu ở Excel, JIRA, Azure DevOps (MS) cạnh tranh với Amazon WS
* 4. Kiểm thử phần mềm thêm cả việc đánh giá chất lượng phần mềm – góc nhìn trải nghiệm UI/UX, góc nhìn NON-FUNCTIONAL REQUIREMENTS
  + Non-Func Res là gì? -> tham khảo cheat sheet SWE
    - Là những yêu cầu/ câu phát biết về chất lượng của app, những yêu cầu về trải nghiệm/ cảm xúc khi xài toàn bộ app
    - “Nhanh/chậm/Đẹp/Xấu/Dễ dùng/Đa dụng…”
      * Nhanh: đo độ phản hồi của một tính năng từ lúc request cho đến lúc response trong bao nhiêu, lưu lượng xử lí, chịu đựng được cùng lúc bao nhiêu nười dùng thao tác.
  + Test app chạy nhanh/chậm
  + Tool: Jmeter (giả lập nhiều người dùng)
* ---------------------------------------------------------------

III. Ai sẽ tham gia vào công việc đảm bảo chất lượng kiểm thử phàn mềm

* 1. Developer/Programmer/Coder
  + Developer trước khi chuyển giao app cho TESTER/QC ở dưới tìm bug, thì developer phải test kĩ code của mình!! Code phải chạy đúng trong từng đoen vị code -> Bức tường tạo nên từ những viên gạch, viên gạch là đơn vị cơ bản của tường -> Hàm/class là đơn vị đo lường co bản của code!!!
  + HÀM VÀ CLASS phải được test kỹ càng trước ki lắp ráp với CLASS khác, với UI, với BACK-END -> TEST kĩ hàm, CLASS được gòi là “Unit Test” (Folder junit)
  + UNIT TEST LÀ CÔNG VIỆC TEST CODE CHẠY ĐÚNG/SAI -> Việc của Developer!!!
  + Làm sao để em có thể test code của chính em???
    - Rất nhiều kĩ thuật, trong đó có 1 kĩ thuật siêu hay đó là TEST AUTOMATION
* 2.QA-Quality Assurance/QC-Quanliti Control, Tester
  + QA khác với 2 thằng kìa: phòng ngừa
    - Định ra các quy tắc làm việc, cách thức làm việc để giảm bug.
    - VD: requirements phải đưẹoc review bởi bên thứ 3 để đảm bảo chất lượng
    - VD: sau 1 đơẹt release, phải họp tổng kết kinh nghiệm, chia sẽ kinh nghiệm rút ra
    - Xem và yêu cầu cả team tuân thủ, yêu cầu xem biên bản của các cuộc họp, không xem app, chỉ xem những phản hồi của khách hàng
  + QC/Tester: tìm bug 2 vị trí là làm những việc đc mô tả ở đ/n kiểm thử
    - 4 việc làm với app để tìm bug:
    - mở app kiểm tra expected và actual
    - Coi app viết có đúng requirements
    - Tìm bug
    - Test non-functional reqs
  + VD: QA phối hợp với PM (Project Manager) chọn Scrum là phương pháp làm app cho dự án X nào đó nhóm sẽ cần thực hiện những điều sau
    - Mỗi sáng, nhóm phải họp nhanh 15p, gọi là daily stand-up meeting. Mỗi member tự nói với các thành viên đúng 3 câu:
      * Hôm qua mình đã làm được những gì cho dự án
      * Hôm nay tui sẽ làm gì tiếp
      * Tui có đang gặp vấn đề gì không?
    - Cuối mỗi SPRINT (thường là 2 tuần) thì họp demo mới khách hàng và ghi nhận Feedback
    - Cuối mỗi SPRINT (2w) nhóm họp riếng, ko khách hàng và chia sẽ kinh nghiệm, bài học mà mỗi thành viên tự rút ra 🡪 Phiên họp chia sẽ RETROSPECTIVE 🡪 phải có bien bản
    - Hàng tháng, hoặc nửa tháng, QA hỏi PM/Team: đã họp chưa, biên bản đâu?
    - 🡪 QA giám sát việc tuân thủ các quy định của các team, các phòng ban, họ cùng các phòng ban đưa ra các quy trình làm việc và giám sát quy trình đó!! Họ không đi vào chi tiết công việc mỗi nhóm
    - 🡪 QC: mở app lên, tìm bug
    - QA sẽ báo cáo/Report với các PM/sếp/Board of manager – ban lãnh đạo về sự tuân thủ/ đảm bảo quy trình. QA có quyền làm việc lấy report của các phòng ban khác, ngoài dev team
    - QC chỉ report bug tới PM
    - QA: Học về quy trình chất lượng: ISO, CMMI
    - QC: trọng tâm của môn học này, tìm bug trong app
* 3. QC LEAD/ QC MANAGER
  + Sếp của nhóm QC/TESTER
  + Sếp QC không cần sờ vào app để tìm bug, việc đó của QC/Tester
  + Sếp làm việc/phối hợp cùng PM(Project Manager) – viết tắc mặc định, và có thể phối hợp với PM (Product Manager – không can thiệp vào code, nhìn vào thị trường/ đối thủ, tự nghĩ xem product của mình phải làm gì ngoài ra còn có PO (Product Owner) là lính của PM) để biết tiến độ làm app, tiến độ realease sản phẩm ra thị trường để lên kế hoạch TEST cho kịp, lên kế hoạch phân bổ tài nghuyên/con người vào các dự án cần kiểm thử/tìm bug
    - Ví dụ: cần bao nhiêu QC để test vào thời điểm X nào đó
    - Cần mua thêm/mượn thiết bị (ví dụ máy in/Barcode/Camera/Sever/…) để test thử app
  + Đọc các báo cáo/Report về bug và ra quyết định
* 4. END-USER/USER/Người dùng/Người dùng cuối
  + Người thực sự cần app cho công việc của họ, sẽ là người tham gia vào việc kiểm thử app, khi họ xài thử app/họ đánh giá xem app có đáp ứng được công việc họ cần hay không. Việc đánh giá 1 app có phù hợp để sử dụng trong công việc của người dùng được gọi là UAT – USER ACCEPTANCE TESTING – KIỂM THỬ CHẤP NHẬN/NGHIỆM THU SẢN PHẨM QUAN TRONG NHẤTTTTT
* ------------------------------
* Ngoại truyện:
* Tui, bạn --- gõ URL ----- Browser ----- Request/response ----- Web Server ------ CODE ------ DB
* Thông tin đi -----------------------------------------------------------------------------------------------------|End
* END|<------------------------------------------------------------------------------
* End-User| Front-end Back-End
* 🡪 Full-stack
* . Front-end
* . Back-End
* ------------------------------
  + 2 LOẠI APP KHÁC NHAU
    - GENERIC APP: app viết cho bá tánh xài, tool đa dụng, nhiều người cùng có nhu cầu dùng
      * Ví dụ: Browser, office, video editor, image processing, IDE, Download tool, xm video
      * Đưa app lên internet cho download, tro trial, lấy feedback, lấy report bug
        + Nightly Build
        + Alpha/Beta/Early Access
        + Preview
        + Insider Review
        + RTM (OS) release to Manufacture
        + RC – Release Candidate (Gần hoàn hảo – của unikey)/FINAL build/LTS – longterm support!!!
      * Ai thích xài thì download và xài và cho feedback/cmt
      * App tự động gửi bug về khi bị treo/ lỗi…
    - CUSTOMIZED APP, BESPOKE APP: app làm theo đặt hàng của ai đó/cty/tổ chức nào đó
      * Ví dụ: site thegioididong != fptshop != didongviet !=…
      * App internet banking của TPBank khác ACB khá BIDV khác VCB …
      * Không có cho download, vì nó được viết riêng cho ai đó/cho tổ chức nào đó
      * Công ty phần mềm sẽ cài vào các máy trong công ty khách hàng để cho khách hàng nhân viên xài feedback
        + VD: Fsoft sẽ cài phần mềm quản lí BV Chợ Rẫy vào máy của các bác sĩ ở BVCR để cho bác sĩ, y tá, tiếp tân, phòng xét nghiệm xài, cho ý kiến !!!
* IV. 7 viên ngọc rồng – 7 nguyên lý kiểm thử phần mềm
  + OOP: 4 + 5
  + . A, E, I, P + S O L I D (Tác giả robert c.Martin)
  + Agile: 4 điều răn + 12 nguyên lý “https://agilemanifesto.org/principles.html”
    - Individuals and interactions over processes and tools
    - Working software over comprehensive documentation
    - Customer collaboration over contract negotiation
    - Responding to change over following a plan
  + Software Testing: 7 nguyên lý – 7 principles of software Testing
    - 7 nguyên lý dùng để định hướng công việc của dân kiểm thử, lý giải cách thức dân kiểm thử làm các công việc tìm bug, định hướng dân kiểm thử khi nào thì nên dừng test và chấp nhận rằng app đã ổn tron chừng mực nào đó.
    - Nguyên lý 1: kiểm thử phần mềm là tìm bug tiềm ẩn, chứ không phải là để khẳng định rằn app hết bug rồi
      * Không bao giờ kết luận rằng app tui hết bug rồi!!! BUG luôn tồn tại trong app dù test kỹ cỡ nào!!!
      * Ý nghĩa: hãy cố gắng tìm ra càng nhiều bug càng tốt, đừng để bug nghiêm trọng còn tồn tại trong app trong khả năng!!!
    - Nguyên lý 2: KHÔNG THỂ TEST HÊT CÁC TỔ HỢP SỬ DỤNG TÍNH NĂNG CỦA APP/

KHÔNG THỂ MÔ PHỎNG HẾT CÁC TÌNH HUỐNG XÀI APP CỦA USER!

* + - * Dân tester/QC là người xài app trước tất cả user, tìm bug, nhờ dev fix, người dùng sau này xài thật, ko gặp bug (lý thuyết là vậy)
      * EXHAUSTIVE TESING IS NOT POSSIPLE
      * Ví dụ cho app caulator ta ko thể mô phỏng hết các tình huống xài tính năng + vó có thể là phép cộng của:
        + 2 số dương, 2 số dương rất rất to, 2 số dương to và nhỏ, 2 số âm, 2 số dương âm,…
        + Vậy không thể test hết các tình huống xài phép + Exhauted testing is not possible làm sao có thể dám kết luận rằng xeif phép cộng đi/ nó đúng đấy/ổn đấy!!!
      * Có nhiều kỹ thuật ví dụ:
        + Chia miền giá trị để test đúng trong biên của miền, đúng cho tất cá trong miền gọi là EQUIVALANCE PARTITIONING – phân vùng tương đương
        + Kĩ thuật quy nạp của toán học
        + -> Mục số 5 cây bản đồ tư duy
      * Góc độ QC, ta chia việc dùng app của user thành các tình huống
        + Nhập data tử tế, app phải xử lí ra kết quả như kỳ vọng
        + Nhập cà chớn, cố tình nhập sai, thì app phải chửi (Validation) thay vì xử lý trên cái sai!!

Nhập ngày tháng, thì ta đưa format nhập: \_\_/\_\_/\_\_\_\_ (DD/MM/YYYY) hoặc chon cuốn lịch

* + - * + Check các tình huống ngoại lệ khác, ví dụ mạng lag, mất kết nối, thiết bị chập chờn/ sai cấu hình
    - Nguyên lí 3: EARLY TESTING: Kiểm thủ càng sớm càng tốt, kiểm thử ngay từ khi bắt đầu làm app, thậm chí khi chưa viết code đã nghĩ đến việc kiểm thử, kiểm thư ngay lúc viết document/SRS/FRS
      * Dân QC tham gia vào ngay công đoạn lấy yêu cầu pm, tham gia đọc/review Requirement document, design, để tìm các sai sót nếu có!!! Phí tổn sửa sai sót lúc này rẻ hơn khi đã code mà sửa lại db, reqs
      * V-Model của môn SWE nhắm yếu tốt QC ngay trên tyuwngf công đoạn của quá trình làm phần mềm
      * Agile thì cứ 2 tuần là có app (Dĩ nhiên QC cũng đã test song song trong 2 tuần) và khi demo cho khách hàng ở cuoois mỗi Sprint, cũng đã Lấy feedback từ sớm đẻ chỉnh sửa trong next Sprint
    - Nguyên lí 4: DEFECT CLUSTERING (differ different difference): Bug sẽ phân bố/ tập trung/ dồn vào một số chỗ nào đó có những chỗ trong app sẽ tìm thấy nhiều bug hơn chỗ khác
      * Liên quan đến 1 nguyên lý về sự phân bố và tính hiệu quả trong hoạt động nguyên lý Pareto – nguyên lý 80/20
      * Có những module/tính năng bug rất nhiều (80% bug)
        + Ví dụ:

module nào có kết nối các thiết bị/sensor

Module/chức năng kết nối với app của bên thứ 3 (3rd party) ví dụ momo, các ví khác, API thứ 3, AWS, FireBase, timeout, mạng rớt, lag

* + - * Nguyên lý này lý giải rằng bug xh nhiều ở 1 chỗ nào đó trong app (Đặc biệt khi app giao tiếp với bên ngoài các hệ thống thiết bị khác) thì QC manager phải có chiến lược điều phối nhân lực, thời gian vừa đủ cho các tính năng này !!!
      * Tính năng CRUD user, edit profile mấy khi có bug, bug nhẹ, dễ tìm
    - Nguyên lý 5: nghịch lý thuốc trừ sâu!!! PESTICIDE PARADOX
      * Hiện tượng kháng thuốc, lờn thuốc, đưa thuốc vào mà lại không trị được bệnh
      * dân kiểm thử để sót nhiều bug, từ đơn giản đến nghiệm trọng
      * khi dân kiểm thử làm 1 công việc lặp đi lặp lại, test mãi 1 cái app, 1 chức năng (do vòng đời phát triền của app – nhiều version) sinh ra chủ quân, tyuwj tin quá mức, cẩu thả, nhàm chán, nên nghĩ rằng đã test rồi, không cần test phần đó nữa, …
      * khi làm một việc gì đó quá nhiều lần, bị “mòn, lỳ” cái tư duy, ko nghĩ đc điều gì mới thêm mà chỉ làm theo thói quen
      * Niềm tin vào các bạn dev đã làm việc qua nhiều mùa!!!
      * -> tất cả các cái “niềm tin, sự chủ quan này” có thể GÂY RA BỎ SÓT BUG!!!
      * MƯỚN BẠN ĐỂ CHẶN BÙ, BẠN LẠI ĐỂ LẠI NHIỀU BUG
      * Câu chuyện cục quản lý hàng không Hoa Kỳ khi verify cấp phép lưu hành cho dòng máy bay Boeing 737 MAX
      * QC manager là đừng làm cho QC chủ quan, nhàm chán -> nên hoán đổi công việc/ dự án test của QC, hoán đổi loại app/ môi trường app chạy cho QC, vì có thay đổi tín chất công việc, có sự tò mò mới, sáng tạo mới -> bug sẽ tìm thấy hiệu quả hơn, nhiều hơn.
        + Test mãi web app thì chuyển sang mobile app, test app ecomerce quen thì chuyển sang test app IoT
    - Nguyên lý 6: CONTEXT DEPENDENT: KIỂM THỬ PHỤ THUỘC VÀO NGỮ CẢNH
      * App khác nhau/ phân loại app khác nhau, môi trường chạy app khác nhau thì có cách thức test khac nhau
      * Ví dụ
        + App về AI/nhận dạng hình ảnh, thì phải chuẩn bị bộ data/ ảnh đủ đa dạng
        + App bãi giữ xe thôn minh, có pần nhận diện ảnh chụp biển số -> biển số rõ, mới, biển số cũ, biển số có bao kiếng, bao kiếng ố vàng, bị hắt ánh sáng xe phía sau, …
        + APP iot thì tính ổn diidnhj đương truyền, truyền data từ sensor, mất this hiệu từ sensor/cảm biến, nhiễu sóng,…
        + Cùng 1 app nhưng 2 phiên bản mobile/Web, web chạy trên mobile -> Responsive chạy đa nền có phụ thuộc OS, vì đòi hỏi cấp quyền truy xuất trên máy là khác nhau!!!
        + App kết noois thiết bị ngoài: camera, barcode scanner, máy in… việc tương thích cấu hình, cấu hình tối thiểu phải chú ý!!!
    - Nguyên lý 7: ANSENCE OF ERRORS FALLACY - ẢO TƯỞNG VỀ VIỆC APP KO CÓ BUG
      * Đừng gáy rằng app em tuyệt với, không có bug, cực kì hiếm bug
      * Gáy rằng app ko có bug, hiếm bug là vô nghĩa, vì làm app ít bug/ít bug nghiêm trọng là HIỂN NHIÊN
      * Hãy gáy về tiện nghi sử dụng, UX, trải nghiệm người dùng, hãy gáy về sự tăng trưởng số lượng người dùng, hãy gáy về app rất được ưa chuộng sử dụng, hãy gáy về app phù hợp nhu cầu sử dụng của user
      * Hãy gáy về UAT, UAT là tối thượng, app phù hợp nhu cầu người dùng, người dùng tin dùng APP, không thể sống thiếu app đó mới là mục tiêu của việc làm app (dĩ nhiên ít bug là default ko thèm đề cập)
  + V. các kĩ thuật tiến hành kiểm thử/đi vào cụ thể các hoạt động kiểm thử:
    - 1. TESTING LEVELS