SWT-Software Testing

▼ SLOT 1: 07/09

tài liệu trong doit-now

text book — cheat sheet: mind map

ISTQB:

- 01/2025-04/2025 kỳ 9
- đồ án xem ở doitnow

```
1. Requirements analysis: SWR
2. Design               : FER/PRN
3. Implementation       :
4. Coding and Testing
5. Maintenance
```

- test case ytb giáo
- USE CASE ytb - PE SWR
- SOLID (dùng interface cho viết code Dependency Injection)
- A E I P
- ASF Bugzilla

▼ SLOT 2: 11/09

- tập nói
- trả bài - rẽ nhánh - tổng quát đến chi tiết
- docker là gì ? ⇒ **Dockerlà một nền tảng để cung cấp cách để building, deploying và running ứng dụng dễ dàng hơn bằng cách sử dụng các**

containers (trên nền tảng ảo hóa). Ban đầu viết bằng Python, hiện tại đã chuyển sang Golang.

- devops
- API : viết cho môn SWT ⇒ WebAPI ⇒ RestAPI ⇒ GraphQL ⇒ OData
- kafka ⇒ ? Kafka là một hệ thống message pub/sub phân tán (distributed messaging system) được sử dụng để truyền một lượng lớn message theo thời gian thực. Bên public dữ liệu được gọi là producer, bên subscribe nhận dữ liệu theo topic được gọi là consumer.
- oauth2 ⇒ ? **Open với Authentication** hoặc Authorization. OAuth ra đời nhằm giải quyết vấn đề trên và xa hơn nữa, đây là một phương thức chứng thực giúp các ứng dụng có thể chia sẻ tài nguyên với nhau mà không cần chia sẻ thông tin username và password.
- **dashboard wep app ⇒ số liệu thống kê biểu đồ**

(SWP hiển thị biểu đồ với dữ liệu: bao nhiêu người đăng ký trong tháng này)

(đếm để tính kpi tính lương)

- bug render front-end ⇒ hiển thị thông tin sai, sai sót phần mềm
- bug win 11
- móng tay
- nhập vai ⇒ tròn vai

ST kiểm thử - TÌM SAI SÓT TRỤC TRẶC PHẦN MỀM

▼ I, Thuật ngữ

bug, defect, error, mistake

▼ II, WHAT is software testing

a, so sánh giá trị kỳ vọng và thực tế

- thao tác so sánh giữa 2 con số, 2 giá trị khi chạy app
- nếu expected value == actual value → tính năng ngon
- nếu expected value != actual value → bug!!!!!!!!!!

vd: bill 1tr giảm 10% ⇒ bill in ra 800k ⇒ ngon

⇒ giả sử mua 1tr phải in ra ⇒ 900k : expected value

b, so sánh trải nghiệm non funcs tasting

hiệu năng

tốc độ

trải nghiệm : kỳ vọng >< thực tế ⇒ bug

kỳ vọng == thực tế ⇒ app ổn

c, đo lường, kiểm tra, so sánh, thiết kế giống đã hứa hay không, đủ chức năng không

cuốn SRS - bản đặc tả yêu cầu phần mềm

soft ware reqs

OUTPUT của môn này

Chốt Hạ : kiểm thử là so sánh 2 thứ

app làm được gì - expected value

thực tế làm gì - actual

nếu expected value == actual → ngon

expected value ≠ actual → bug

▼ SLOT 3 : 14/09

mt

duy trì được không ?

[out story]

- FUNCTIONAL TESTING >> test chức năng app >> trải nghiệm app ≠ FUNC REQS >> test reqs viết ra có ổn không, không liên quan tới test app
- NON - FUNC TESTING ≠ NON FUNC REQS >> vớ vẩn

FUNC TESTING khi test thì tất cả FUNC REQS đều ok

III, WHO JOIN ST - ai tham gia vào quá trình kiểm thử >> đảm bảo chất lượng phần mềm

▼ 1. DEVELOPER >> viết code và test code

- không chỉ viết code làm app >> đảm bảo code viết ra >> chạy thật ổn >> test code trước khi TESTER động tay
- đảm bảo từng đơn vị code >> qua kiểm thử
- unit of code >> đơn vị code nhỏ nhất là func[methods in OOP]và class (không phải dòng lệnh)
- dev hàng ngày viết hàm/func/method và class >> đảm bảo chạy đúng

▼ **TEST FUNC, CLASS**

C1: In ra kết quả xử lí, trả về hàm ra màn hình (sout, cw)

C2: Ghi ra kết quả ở log file

C3: Dùng thư viện/framework tự động hóa quá trình testing

HIỆU QUẢ: Giúp việc **kiểm thử** và **sau kiểm thử** trở thành quy trình tự động hóa

junit-jupiter, pom.xml >> test code dân lập trình

key: QUY TRÌNH CI/CD/DevOps

unit test framework >> jest, mocha, Junit, TestNG, xUnit , MS Test,

▼ 2. TESTER >> test app /QC-quality control

test automation >> toidicodedao

- xài trực tiếp app dân devs làm ra để tìm BUG
- **compare >> expected value == actual value ? >> đo performance/trải nghiệm app** >> check xem app có làm đúng như đã hứa trong SRS?

▼ **HOW COMPARE**

- có app trong tay, có data, tính trước trả về (expected)>> xem app xử lí (acual)
- so sánh ….

- tester chuẩn bị bộ data để thử >> app đổi hệ cơ số >>cần có TEST CASE để so sánh

QA : thanh tra

250 ⇒ FA ⇒ #case1

15 ⇒ F ⇒ #case2

16 ⇒ 10 ⇒ #case3

key: đề PE yêu cầu khoảng 20 test case

- **QC/Tester** dùng sức test >> **MANUAL TESTING** >> bằng tay >> mở app >> nhập test case xem kết quả

key: selenium >> tool tự động hóa

- **QC/Tester** >> tool >> tự động hóa các bước input click select checkbox >>

▼ **AUTOMATION TESTING**

RECORD & PLAYBACK (REPLAY) >> **KATALON**, AKA AT TEST, TEST COMPLATE, RANOEX, TELERIK…

- tự code phần điều khiển >> SELENIUM, APPIUM, CYPRESS >> làm tool cày view, làm bot crawler, cào data,

keyTOOl Cào data ở github

▼ 3. QC MANAGER

quality control

▼ 4. USER/END-USER

chặng cuối của thông tin

end : dừng lại ở người dùng sau quá trình install phần mềm trên mạng về

▼ K

- CHỐT HẠ: CHO QC/TESTER:
- QC/TESTER LÀM VIỆC VỚI APP, KHÔNG FOCUS VÀO CODE
- QC: THIẾT KẾ/DEVELO CÁC TEST CASE (DATA GÌ ĐƯA VÀO APP, EXPECTED VALUE LÀ GÌ, STEP ĐỂ TEST TÍNH NĂNG) – ĐỀ THI PE CẦN LÀM 20 TEST CASE
  TEST RUN, CÓ TEST CASE RỒI, LÔI APP RA CHẠY THEO TEST CASE ĐỂ TÌM BUG HOẶC CHỨNG MINH APP NGON
  NẾU LÀM TEST AUTOMATION THÌ PHẢI VIẾT CODE ĐỂ TỰ ĐỘNG HÓA VIỆC KIỂM THỬ LÀM CHO APP CHẠY NHƯ MA NHẬP, ĐỂ VERIFY APP ĐÚNG SAI
  GHI NHẬN LẠI TOÀN BỘ BG NẾU CÓ BÙ VÀO DATABASE
  [NGOẠI TRUYỆN]: THÔNG TIN TUYỂN DỤNG THƯỜNG GHI: TUYỂN QA/QC (QUALITY ASSURANCE) QA LÀ GÌ?
- gã này ko trực tiếp sờ phần mềm, chạy phần mềm để tìm bug, vì đó là việc của QC
- gã này cùng sếp, ban lãnh đạo, cùng vơi sPM, cùng với SCRUM MASTER định nghĩa ra, phác thảo ra, soạn các quy tắc về chất lượng trong công ty trong nhóm dự án, sau đó đi kiểm tra xem, thanh tra xem các phòng ban, các dự án có làm theo cam kết hay không
- đóng vai trò thanh tra kiểm tra anh em có tuân thủ quy trình hay không sau đó đi méc sếp, méc lãnh đạo
- gã này ko gắn với từng project cụ thể, mà có thể kiểm tra đôn đốc các phòng ban luôn (QC/TESTER được phân công dự án nào thì gắn với dự án đó, không ngó ngang lungtung) ~ giống 1 chút giám thị hành lang (không phải giám thị phòng thi)
- ví dụ:
- nhóm dự án SWP/ ngoài đời là dự án làm app cho ngân hang thống nhất họn SCRUM làm phương pháp quản lý dự án
- SCRUM yêu cầu vài thứ như sau
  -> dự án chia khoảng thời gian release, demo sản phẩm, làm sản phẩm thành những đoạn thời gian đều nhau, trung bình 2 tuần – SPRINT
  -> REQUIREMENTS PHẢI LƯU Ở 1 NƠI – TOOL- JIRA (HOẶC TOOL

TƯƠNG ĐƯƠNG) GỌI LÀ BACKLOG, VÀ SẼ ĐƯỢC PHÂN BỔ VÀO TỪNG SPRINT 2W
-> cuối mỗi 2 tuần- chiều t6 tuần 2, hẹn gặp khách hang demo, ghi nhận feedback vào trong 1 biên bản
-> gặp demo với khách hang xong, thì team có 1 phiên họp gọi là RETROSPECTIVE – họp chia sẻ và rút kinh nghiệm, chia sẻ cảm xúc luôn – ghi biên bản
->Mỗi sáng trong tuần làm việc, team họp đứng, lần lượt mỗi người nói 3 câu: hôm qua tui làm gì, hôm nay tui sẽ làm gì, tui có đang gặp vấn đề gì trong dự án hay không?
-> gọi là DAILY STANDUP MEETING
SCRUM MASTER LÀ GÃ RUN/ QUẢN LÝ NHÓM NÀY THEO ĐÚNG NHỮNG THỨ ĐÃ NÊU Ở TRÊN
QC/TESTER SẼ CHỜ APP CÓ Ở MỖI TUẦN ĐỂ TEST, TEST SONG SONG KHI DEV ĐANG CODE
QA: LÂU LÂU GHÉ VÀO TEAM, HỎI XEM TÌNH HÌNH NHÓM CÓ VẬN HÀNH NHƯ CAM KẾT HAY KHÔNG
PROJECT RUN ĐƯỢC 4 SPRINT, TƯƠNG ĐƯƠNG 2 THÁNG, VẬY QA VẦN NHÌN THẤY 4 CÁI BIÊN BẢN CUỘC HỌP VỚI KHÁCH HÀNG, KHÔNG XEM CHI TIẾT BÊN TRONG

QA KIỂM TRA SỰ TUÂN THỦ VỚI NGUYÊN LÝ ẨN PHÍA SAU: CTY TA ĐÃ CÓ QUY TRÌNH TỐT VÀ MỌI NGƯỜI TUÂN THỦ TỐT, HY VỌNG CHẤT LƯỢNG SẢN PHẨM SẼ TỐT THEO
QA SẼ LÀM QUEN VỚI CÁC TIÊU CHUẨN CHẤT LƯỢNG MÀ THẾ GIỚI HAY DÙNG: ISO, CMMI
QA phối hợp với cá sếp để đtặ ra quy trình, thủ tục cuộc chơi mỗi phòng ban/ project vận hành bên trong mỗi phòng ban/ project thì để mỗi bên tự chơi
3. QC MANAGER -SẾP CỦA QC/TESTER, LƯƠNG CAO HƠN DEV
*CÔNG VIỆC SÚA SẾP QC:

- báo cáo tiến độ kiểm thử với PM/sếp trên
- phối hợp với P/M để lên kế hoạch kiểm thử app, hình thức kiểm thử app
- phân bổ nguồn lực, đội ngũ QC vào các dự án: Mình có trong tay 5 QC mà 3 app cần test

- lên kế hoạch mua sắn thiết bị, cấu hình thiết bị dung cho kiểm thử (vd: test app bán hang thì phải có máy đọc bardcode, máy in, server, máy cài app tính tiền, test app bãi giữ xe thông minh, thì phải mua camera độ phân giải cao để chụp được biển số rõ rang)
- thiết lập quy trình kiểm thử: test case, ai làm test run, bug để ở đâu, report tới ai
- thường sẽ là không cần mở app để tìm bug như QC – tùy quy mô công ty to nhỏ
- LÀM SAO ĐỂ TRỞ THÀNH QC MANAGER
- SOFT SKILLS
- NÊN ĐI LÊN TỪ LÍNH QC/TESTER
- HỌC CÁC CHỨNG CHỈ QUẢN LÝ(PMP/PMI), CHỨNG CHỈ KIỂM THỬ (ISTQB)
  PMP BOK

1. USER – END USER – NGƯỜI DÙNG/ NGƯỜI DÙNG CUỐI CŨNG/ CẦN THAM GIA VÀO QUY TRÌNH KIỂM THỬ PHẦN MỀM
   NGƯỜI DÙNG LÀ GÃ CUỐI CUGNF TRONG CHUỖI TẬN HƯỞNG THÔNG TIN ÀM CHẠY TỪ XA XA TRÊN INTERNET BAXK- END
   Phía sau trình duyệt đẩy về đến server là phái hậu trường back-end
   Trước mặt end – user là trình duyệt, ui, được render
   Frond - end

nhào vào làm sẽ giỏi

sinh hoạt khoa học

▼ SLOT 5 : 21/09

key : SOLUTION ARCHITECT : edwardthienhoang.wordpress.com

▼ Phân loại phần mềm theo platform - nền tảng (phần cứng + OS + Môi trường app phụ trợ khác để giúp app chạy : LINUX, WINDOWS, ANDROID, IOS.

▼ Phân loại phần mềm theo đối tượng sử dụng, mục đích sử dụng.

▼ **Generic app**: đa dụng dành cho đám đông, app phổ thông, người người nhà nhà dùng.

- VD: MS Office, Photoshop, Browser, Tool công cụ: Dowload, Anti Virus, Cắt dựng phim, Game,…
- Đặc trưng: có nút dowload, **hướng đến đám đông.**
- User: thị trường, cộng đồng lớn ⇒ cty làm app tung ra ⇒ nhờ thị trường trải nghiệm, FB, Report Bug,… public các phiên bản phần mềm ⇒ chiến lược **GENERIC APP.**
- Phiên bản: Phiên bản **NIGHTLY BUILD** (bản thử nghiệm dev đưa ra) **: alpha, beta, insider review, preview, RC (release candidate), stable, LTS,** (app tốt và tiếp tục được hỗ trợ ở phiên bản sau mới hơn) ⇒ **trial, pro, enterprise, community(free ít tính năng), open-src (free cả app và src code).**

▼ **Customized, bespoke app**: sử dụng cho mục đích riêng biệt tập user thu hẹp hơn, App chuyên biệt

- VD: app tính tiền coopmart, tocotoco, 7-eleven,…, app quản lí kho Hòa Phát, app ,…
- Đặc trưng : không thấy nút dowload, **hoạt động chuyên biệt.**
- User: Nhân viên, Khách hàng công ty Viết thep đơn đặt hàng, nhu cầu ⇒ user gói gọn ,nội bộ : là nhân viên , bác sĩ, thủ kho, thu ngân, bồi bàn,… ⇒ được cài app vào máy ⇒ sử dụng, đánh giá ⇒ nếu không ổn đề nghị dev sửa lại ⇒ kiểm thử nghiệm thu (user acceptance testing) **UAT**

>>có app là sự kết hợp giữa 2 kiểu này.

>>app đi từ: QC/Tester ⇒ user ⇒ cộng đồng đưa lên internet, doanh nghiệp cài vào máy ⇒ UAT

▼ IV, 7 Principles of ST - 7vienngocrong

[out story]:

▼ **OOP**: 4 + 5

▼ **DB**:

3 dạng chuẩn

2 cách vẽ erd

3 dạng SQL **(DDL-create/drop/alter, DML-select,update,delete, DCL-grant/revoke)**

5 join **(Left, Right, Inner, Full, Cartesian Product)**

▼ **AGILE: 4**

AGILE MANIFESTO - Tuyên ngôn

- Individuals and interactions over processes and tools
- Working software over comprehensive documentation
- Customer collaboration over contract negotiation
- Responding to change over following a plan

Kent Beck Unit Test

Martin Fowler UML

▼ **ARCHITECTURAL VIEW**: góc nhìn cấu trúc bên trong : 4+1 VIEW MODEL

▼ Testing: 7 nguyên lí ⇒ định hướng qc/tester biết nên làm gì với app về mặt chất lượng phần mềm, khi nào kiểm thử, kiểm thử đến mức độ nào thì dừng, làm sao để tìm được nhiều bug, hoặc tránh để sót nhiều bug, giúp chỉ ra cách thức làm nghề kiểm thử 1 cách hiệu quả đúng đắn ⇒ giống oop 4 định hướng object ⇒ làm ra app

- 1, Nguyên lí đầu tiền về kiểm thử phần mềm- testing shows presense of defects

▼ NGUYÊN LÍ CHỈ RA HƯỚNG ĐI CÁCH LÀM…

1. NGUYÊN LÝ ĐẦU TIÊN VỀ KIỂM THỬ: TESTING SHOWS PRESENCE OF DEFECTS

- Kiểm thử phần mềm đơn giản chỉ là đi tìm bug, tìm sai sót, khiếm khuyết còn tồn tại, còn tiềm ẩn trong app
- Kiểm thử phần mềm cố gắng tìm được nhiều bug càng tốt
- Kiểm thử phần mềm là tìm bug, chứ không phải đi chứng minh rằng app em hết bug, “FREE OF ERROR”- KHÔNG ĐƯỢC CHỨNG MINH APP KHÔNG CÓ BUG, BUG HẾT RỒI VÀ ĐIỀU NÀY LÀ BẤT KHẢ
- App luôn có bug dù test kĩ cỡ nào (nói vui bug là một tính năng vì app kiểu gì cũng còn có bug)
- App luôn có bug nhưng không đồng nghĩa với việc test cẩu thả, đại khái, qua loa. Vì nhiệm vụ của dân QC là đảm bảo app dùng ổn, cty mình phải giữ uy tín với khách về phần mềm có chất lượng
- Nhiệm vụ của QC là ngăn ngừa, tìm ra càng nhiều bug càng tốt, không nên để tồn tại những bug nghiêm trọng ảnh hưởng đến việc sử dụng app của user, khách hang

1. EXHAUSTIVE TESTING IS NOT POSIBLE/ IS IMPOSSIBLE

- Vắt kiệt, ép xung, đẩy tới hạn, hết sức bình sinh
- Không thể vét cạn được tất cả, không thể test hết được các tình huống xài app
- Việc mô phỏng lại tất cả các tình huống xài app của user trong tương lai, việc chạy hết, thử nghiệm hết, run hết các test case là bất khả, không thể làm được

Vét cạn hết các tình huống kiểm thử là không thể

1 app viết ra, 1 màn hình của app thì sẽ có vô số cách dùng, cách nhập data, vô số bộ data được đưa vào màn hình, app

Vô số data đưa vào-> vô số test case, vô số các tình huống data đưa vào màn hình

“LOGIC HỢP LÍ: MUỐN KHẲNG ĐỊNH APP NGON THÌ PHẢI TEST HẾT TẤT CẢ CÁC TÌNH HUỐNG XÀI APP”

VÍ DỤ: kiểm thử tính năng cộng 2 con số của app calculator

>chứng minh app ổn để sử dụng

>ta cần chuẩn bị các test case là các bộ data đưa vào + expected output

> sau đó ta run thử app với các bộ data này (test run) coi app có trả về có như kỳ vọng tính toán trước đó hat không. ổn hoặc bug!!!

> test case: 2 số dương cộng nhau

2 số dượng cộng nhau, coi có tràn miền giá trị

2 số âm cộng nhau

2 số âm cộng nhau, coi có tràn miền giá trị

Số dương + số âm

Phép cộng trên các loại số khác nhau: nguyên, thực,…

Sau này mở rộng +, -, *, / trong biểu thức

Vô số test case tồn tại, vậy nếu muốn test hết các case để chắc cú, khẳng định ngon, đó là điều không thể. Thời gian, sức lực, luôn bị chặn bởi ngân sách, nhân lực, thời gian

Vậy không test được hết, không vét cạn được hết các case, EXHAUSTIVE thì làm sao, có cách nào dám kết luận app ổn?

- ĐÁP ÁN: SỬ DỤNG CÁC KĨ THUẬT, CÁCH THỨC THIẾT KẾ TEST CASE (TEST IN TECHNIQUES)

CHỈ DÙNG MỘT LƯỢNG TEST CASE NHẤT ĐỊNH MÀ ĐÃ DÁM KẾT LUẬN

>>>>cheatsheet đã có keyword ở mục 5.testing ttechniques

3.EARLY TESTING – KIỂM THỬ CÀNG SỚM CÀNG TỐT, THẬM CHÍ KIỂM THỬ XẢY RA NGAY KHI BẮT ĐẦU DỰ ÁN, NGAY Ở GIAI ĐOẠN LẤY REQUIREMENTS, DESIGN

- dân QC không chần chừ, vào ngay từ giai đoạn REQUIREMENTS và các giai đoạn kế tiếp, dĩ nhiên việc này phải được lên kế hoạch/ duyệt bởi QC MANAGER và PM

- Càng kiểm thử sớm, càng tiết kiệm được công sức, phí tổn sau này

Code đã viết, phát hiện thiếu tính năng, thiếu table, them table, them relationship, thay đổi câu query (56-82%)

- QC nhảy vào review document được viết bởi BA, PO, để giúp phát hiện sai sót từ sớm

Trong việc hiểu app, hiểu các tính năng của app thì tester hiểu app nhất, BA nhất hoặc nhì

- QC đọc cuốn document được viết bởi BA/designer, technical để thiết kế các test case sẽ dùng cho test run

Cuốn document ở giai đoạn Requirements này gọi là SRS- SOFTWARE REQUIREMENTS SPECIFICATION – BẢN ĐẶC TẢ YÊU CẦU PHẦN MỀM (cuốn document của khóa luận tốt nghiệp chiếm ½ số trang)

1. DEFECT CLUSTERING – SỰ PHÂN BỔ CỦA BUG – bug ở chỗ nào nhiều, chỗ nào ít

- NGUYÊN LÝ 80-20 (PARETO) ÍT MÀ LẠI NHIỀU
- có những khu vực trong app tiềm ẩn cực kỳ nhiều bug, chỗ lại ít bug
- nếu hiểu như vậy nên tập trung nguồn lực vào chỗ nhiều bug, tập trung đúng chỗ thấy được nhiều bug, hiệu quả của việc chọn đúng chỗ
- Chỗ nào nhiều bù, ít bug trong app???

Màn hình CRUD truyền thống thì ít bug: CRUD user, CRUD category, CRUD sane phẩm, CRUD blog,.. những tính năng này tương đối độc lập, xài 1-2 bảng, chỉ là câu query đơn, không join bảng nên mấy khi có bug

Màn hinhg dính đến nhiều table phía hậu trường, gài rang buộc data

Tính năng có dùng them thiết bị ngoại vi: máy quét, máy scan barcode, camera, cảm biến -> đính đến độ ổn định, tương thích song wifi, sóng radio, nhiễu -> test case nhiều

Tính năng có kết nối với API/App bên ngoài: app mình móc với MOMO, GHN, GHTK,…

Có sự bất đồng bộ, tương thích, kết nối do hệ thống phức tạp -> focus vào những chỗ này vì tiềm ẩn nhiều bug

PESTICIDE PARADOX – NGHỊCH LÝ THUỐC TRỪ SÂU- HIỆN TƯỢNG KHÁNG THUỐC

- nhờ QC gác cổng check var về chất lượng sản phẩm, nhưng mày lại để tồn tại quá nhiều bug, bug nghiêm trọng
- Dân QC đôi khi chủ quan, hời hợt, hay làm quen, test tới test lui các app này nửa năm, thậm chí tính năng, fix bug, test lại, ôi dào lần trước test rồi, chắc không sao đâu,… Chắc code không sử dụng module này, thì không cần test module này…

Làm mãi sẽ nhàm, sẽ mất tập trung, chép miệng, ignore -> Bug tiềm ẩn đã không được tìm thấy do quy trình kiểm thử đã bị vi phạm, bỏ qua vài bước, module không được check

- FIX:
- Hoán đổi dự án kiểm thử, module kiểm thử: vd test về theo dõi đơn hangf mãi thì chuyển sang test về theo dõi công nợ
- Hoán đỏi môi trường kiểm thử, loại app kiểm thử. Vd: test quen web app rồi thì mình chuyển sang test app mobile, khác môi trường, loại app cần test, kích thích sự tò mò, khám phá vùng đất mới đôi mắt mới, thấy được vấn đề mơi
- Ứng dụng điều này vào trong việc quản lý thời gian và công việc một cách hiệu quả, luôn duy trì được sự tò mò khám phá

TESTING IN CONTEXT DEPENDENT – KIỂM THỬ PHỤ THUỘC VÀO BỐI CẢNH – BỐI CẢNH CÂU CHUYỆN VỀ APP KHÁC NHAU, LOẠI APP KHÁC NHAU, MÔI TRƯỜNG CHẠY APP KHÁC NHAU, MỤC ĐÍCH APP KHÁC NHAU, CÁCH KIỂM THỬ KHÁC NHAU

Test case được design cho các loại app/ môi trường khác nhau thì cần phải design khác nhau

Test mobile app sẽ có khác so với web app

Responsive trên màn hình nhỏ, dọc màn hình rộng ngang

Có bị xoay màn hình không thường xoay

Mức độ bảo mật app khi cài, cũng có vấn đề bảo mật

khi sử dụng khác nha khác nhau như OS, trình duyệt

- App có thêm liên kết với bên ngoài (momo) với các thiết bị ngoại vi (cảm biến) thì cách tư duy về test và test case cũng cần khác

docker ảo hóa

SLOT 7: 28/09

▼ 7. ABSENSE OF ERRORS FALLACY (sai lầm, ảo tưởng khi suy nghĩ về app hết bug)

- app hết bug → sai: + app luôn có bug, vấn đề là chưa tìm thấy
- thành công của app đo bằng lượng người dùng, sài app ít, bug là hiển nhiên
- dù có bug nhưng phải đem đến khách hàng những mặt có lợi của app: tiện dụng, dễ dùng, user đông, người dùng đông
- góc nhìn người sài app
  - generic
  - customized/bespoke

  ⇒ đều phải được user ưa thích thì sự tồn tại của app mới có ý nghĩa

  UAT là quan trọng nhất liên quan đến lí do tồn tại của app ko phải app nhiều hay ít bug

▼ Chốt 7ball: 7 principles dẫn lối, chỉ dẫn QC/Tester hành nghề testing

- ko cố gắng test hết case vì bất khả
- hoán đổi vị trí công việc QC tạo sự mới mẻ → tránh nghịch lý thuốc trừ sâu
- bug xuất hiện ở 1 số chỗ, tùy app → phân bổ đúng nguồn nhân lực
- kiểm thử nên làm càng sớm càng tốt → hiểu hệ thống, trải nghiệm,
- UAT cực quan trọng → bug not the face

▼ V, TESTING LEVELS - mức độ kiểm thử

> Bàn về kiểm thử app - not document - early testing

> app bắt đầu đến từ code, xét dòng thời gian làm app -tính từ khi viết code

REQS

DESIGN

IMPLEMENTATION → dân dev

TESTING → dân dev

DEPLOYMENT

MAINTERNANCE/ENHANCEMENT

▼ IMPLEMENTATION - TESTING (dân dev viết code và làm app)

→ dev code - test code của mình

- unit test: hàm, class

⇒ Unit testing

⇒ Integration testing level

⇒ System testing level

⇒ Đưa app cho user: site, cài máy cty, deploy server, UAT, UAT level

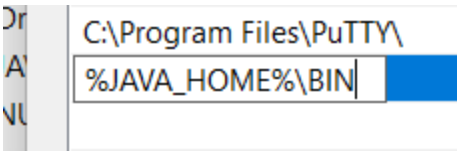
⇒ QC tác động vì đã có ui, tương tác với app ở mức độ user với vài tính năng

⇒ chắc chắn có QC app đã hình thành tương đối

⇒ app xong chắc chắn có QC→ design test case → Manual & Automation

▼ 1. Unit testing level - dev

- Cách 1: In kết quả xử lí ra màn hình
- Cách 2: Ghi ra log file
- Cánh 3: Dùng đồ chơi, thư viện phụ trợ bên ngoài - Unit testing framework ⇒ khuyến khích vì quy trình của **CI/CD/DevOps**



- biến môi trường → đứng ở đâu gõ cũng được → cơ chế hoạt động của hệ điều hành
- tại sao bắt đặt tên com.mucorerarvy → liên quan đến maven

THIẾT KẾ TEST CASE

key: SCM, VCS -version cotrol system

GIT new Github

IDE gồm tất cả tool tự động và cả code editor

▼ 19/10

technical skills

Unit Test, JUnit, TDD, DDT, Continuous Intergration, Github, Maven

- TDD: Test Driven Development >> viết code và test đồng thời cùng lúc, //
- DDT: Data Driven Testing

▼ 23/10 - Testing

▼ 26/10 - Selenium IDE extention

- record and playback

▼ 30/10 - API code

- SAP1701 tat ca hoac khong la gi ca
- Q1: GAP NHAU CHANG PHAI CO HOI
- Q2: CHANG GI LA MAI MAI
- Q3: XA ANH EM PHAI HANH PHUC