© 2022 – SE Department – FPT University | HCMC

Sóng bắt đầu từ gió
Viết App bắt đầu từ
Requirements…

© 2022 – SE Department – FPT University | HCMC

Sự tinh tế
là khởi nguồn nên
điều vĩ đại...

© 2022 – SE Department – FPT University | HCMC

Technical Documentation in Software Development

SRS - Common Mistakes & Best Practices

ThS. Nguyễn Thế Hoàng | ThS. Lâm Hữu Khánh Phương

Version 22.04

© 2022 – SE Department – FPT University | HCMC

Agenda

0. SRS+ in Overall
I. Project Introduction
II. Project Management Plan
III. Software Requirements Specification
IV. Q&A

Cấu trúc chung SRS+

- **Hình kế bên là dàn mục lục cho phần SRS**, là nửa đầu (½) cuốn document (report) của Khoá luận tốt nghiệp (Capstone Project) mà sinh viên ngành SE Đại học FPT cần hoàn tất đi kèm với App tự viết để ra trường đi cày code kiếm cơm.
- Nhớ sửa mục III thành Software Requirements Specification giùm tui!!!

© 2022 – SE Department – FPT University | HCMC

Acknowledgements

- **Lời cảm ơn luôn ở đầu cuốn tài liệu.**
- Dù "ghét" chúng tôi đến mấy nhưng theo chuẩn giao tiếp và chuẩn "xã giao" thì phần cảm ơn là điều MUST HAVE. Không tin hãy lật giở trang trong của các cuốn sách của nước ngoài, và sách Việt Nam cũng có mà!
- Nhớ cảm ơn đủ các stakeholders nhé bạn, **với tất cả tấm lòng!**
- Dành trang trọng 1 trang riêng.

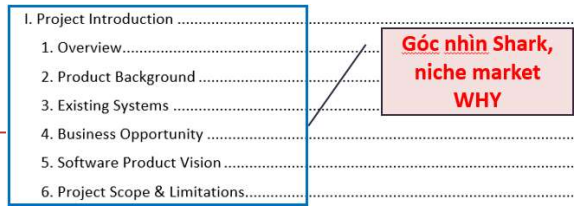
© 2022 – SE Department – FPT University | HCMC

Definition and Acronyms

- **Định nghĩa/giải nghĩa các thuật ngữ, các từ viết tắt sử dụng trong tài liệu**
- Đừng cẩu thả, lơ là phần này, nhiều trò thậm chí giữ nguyên phần ví dụ của template – là những thuật ngữ quá quen của ngành IT!
- Đồ án nào cũng có những thuật ngữ chuyên môn riêng (domain knowledge, term/jargon) ngoài những thuật ngữ IT hay gặp như UI/UX, API…
- Hãy đưa những thuật ngữ riêng của dự án vào đây để chứng minh bạn muốn người đọc hiểu và đánh giá cao thành quả của bạn.
- Ví dụ:

Bill	Bills are issued for cash transactions that are completed in one go. Bill serves as proof of transaction.
Invoice	Invoices are usually used for credit transactions with specific due dates for payment. It is a legal document used for financial reporting.

Source: https://cleartax.in/s/invoice-vs-bill



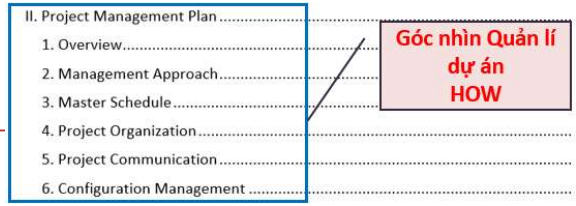
2. Product Background

- **Product Background – Hiện trạng hiện nay – As is System – Legacy System:** Trước khi có App của chúng ta, users/customers đang làm công việc của họ thế nào? Họ có nỗi đau gì – **Customers' Pain Points**?
- **Existing Systems – Các "đối thủ" trên thị trường:** Nên mô tả ít nhất 3 sản phẩm có sẵn trên thị trường. Template gợi ý:

  Tên | URL | Actors/Users chính | Các tính năng chính mỗi Actors/Users có thể sử dụng | Pros& Cons – Ưu điểm & Nhược điểm.

  Phải khảo sát thị trường hay nội bộ để thấy cơ hội và thị trường ngách cho App của bạn. App của bạn cần phải >= các App đang có ở góc độ giải quyết được các vấn đề của As is System.

2. Management Approach



- **Vui lòng tham khảo tài liệu *Seminar Notes* đính kèm, có giải thích cụ thể…**

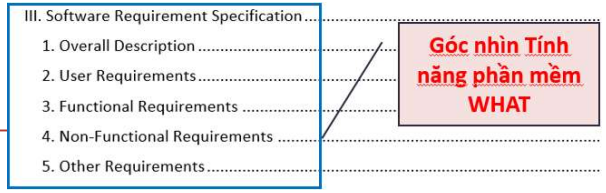
Mục II. Project Management Plan của Capstone Project

- **Quản lí dự án:** Phương pháp luận phát triển phần mềm – Quy trình làm phần mềm – Software Development Methodology/Method/Model/Process, có các trường phái chính:
  - ✓ **Traditional**: Waterfall, V-Model, Sashimi…
  - ✓ **Agile**: Scrum, XP, Kanban, Lean, Lean Startup…
- **CI/CD/DevOps**: Liên quan đến quản lí các tài nguyên dự án; quản lí chất lượng code – chất lượng sản phẩm; và quản lí sản phẩm đầu cuối; quản lí mọi thứ từ lúc bắt đầu dự án đến lúc chuyển giao sản phẩm đến tay khách hàng hay sản phẩm go-live.

FPT University HCMC — Page 2 of 6

SRS – Common Mistakes & Best Practices Seminar — Software Engineering Department

- Trong document của Capstone Project, khi viết về việc lựa chọn Quy trình phát triển phần mềm, cần có những câu lí luận, giải trình cụ thể để làm rõ tại sao bạn lại chọn phương pháp này mà không phải phương pháp kia, kiểu như:
  - ✓ Phải có câu chữ nói lên ý nghĩa đặc trưng của dự án đang làm.
  - ✓ Dự án đang làm có những đặc điểm gì để phù hợp với quy trình phát triển phần mềm được lựa chọn.
  - ✓ Waterfall: thường là Requirements đã ổn định, rõ ràng, khách hàng nói rõ được mong đợi của họ, định nghĩa được ngay những thứ họ cần.
  - ✓ Agile: Requirements còn mơ hồ, chưa rõ ràng, cần có prototype/pilot version để "dò" ý khách hàng.



1. SRS: Overall Description

"Khẩu quyết" tìm Requirements

Sử dụng các **Requirements Elicitation Techniques** đã học trong môn học **Software Requirements** để:

- **Liệt kê** danh sách các User Types/Roles/Actors.
- **Liệt kê** danh sách các tính năng của App mà mỗi loại User có thể sử dụng cho công việc của họ.

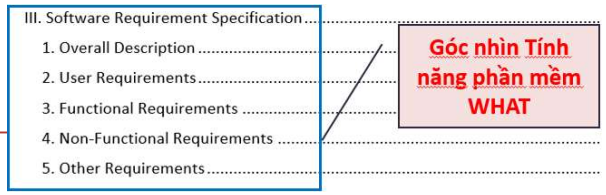
Source: Giáo trình Software Requirements, 3rd, Microsoft Press, 2013

Identify and characterize the different user classes for your product early in the project so you can elicit requirements from representatives of each important class by asking the project sponsor who he expects to use the system...

1. App viết cho **AI** dùng?
2. **AI** làm được **GÌ** với App – App cung cấp tính năng **GÌ** cho **AI**?

Usage-centric or product-centric?

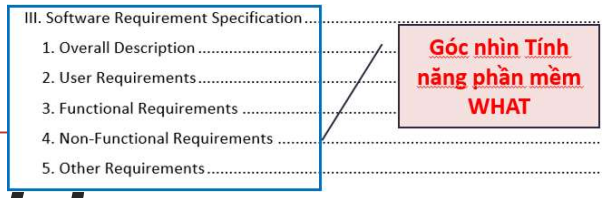
Requirements elicitation typically takes either a usage-centric or a product-centric approach, although other strategies also are possible. The usage-centric strategy emphasizes understanding and exploring user goals to derive the necessary system functionality. The product-centric approach focuses on defining features that you expect will lead to marketplace or business success. A risk with product-centric strategies is that you might implement features that don't get used much, even if they seemed like a good idea at the time. We recommend understanding business objectives and user goals first, then using that insight to determine the appropriate product features and characteristics.



2. SRS: User Requirements

Sơ đồ Use Case Diagram (UCD) – Mô hình tình huống sử dụng App

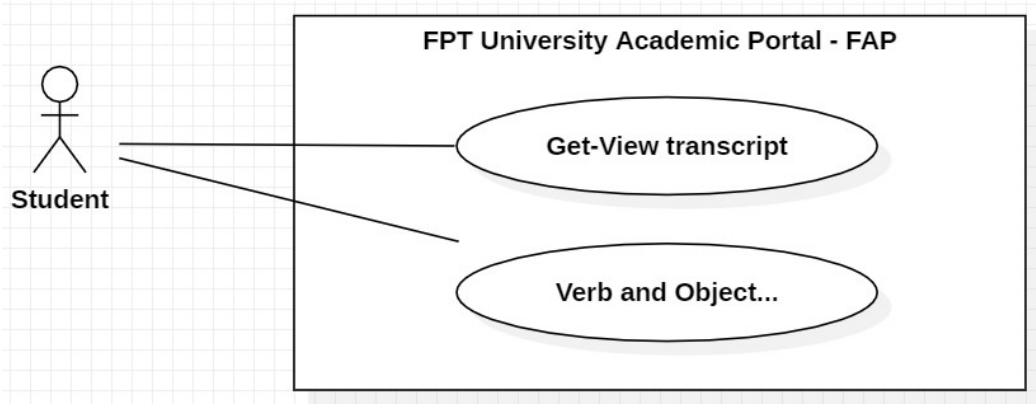
- Là 1 sơ đồ liệt kê danh sách các loại-user mà App hỗ trợ.
- Là 1 sơ đồ liệt kê các tính năng của App.
- Là 1 sơ đồ liệt kê "tên-các-màn-hình, tên-các-chức-năng" của App.
- Là 1 sơ đồ liệt kê “ai/user“ sử dụng tính năng gì của App.
- Là 1 sơ đồ liệt kê các tính năng của App mà mỗi user được quyền dùng để phục vụ cho công việc của họ theo bối cảnh doanh nghiệp nơi App được sử dụng.
- Là sơ đồ mô tả **góc nhìn** tính năng của App **từ phía** người dùng.
- Mỗi tính năng – ứng với một tình huống sử dụng App của người dùng, được gọi là **Use Case (UC)**
- **UCD KHÔNG** là sơ đồ mô tả luồng/flow chạy/luồng liên kết các màn để hình thành một chuỗi các thao tác!!! **NO NO NO!!!**

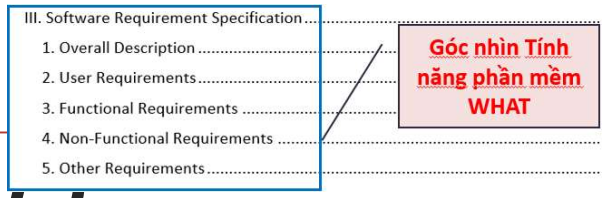


2. SRS: User Requirements *(cont.)*

UCD Checklist in overall

- ✓ Nếu hệ thống có sự tách biệt đủ lớn giữa các khối chức năng, ví dụ Mobile, Web, Customer, Merchant,… có thể tách các khối chức năng này thành các sơ đồ UC riêng cho dễ theo dõi, dễ quản lí các Requirements.
- ✓ **Phải có** System Boundary và tên App để biểu thị **Scope**.

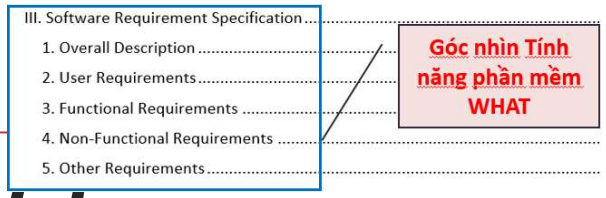




2. SRS: User Requirements *(cont.)*

Actor Checklist

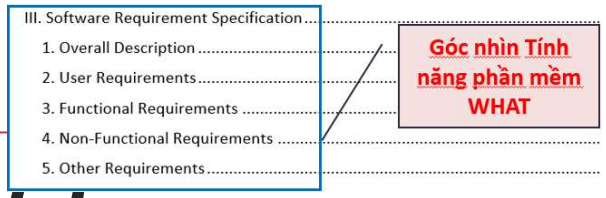
- ✓ Actor: Tên của nhóm user-role/user-class/nhóm người dùng đã được phân quyền sẽ dùng app sau này hoặc những "thứ bên ngoài" có tương tác gửi/nhận thông tin với App chúng ta.
- ✓ Quy ước đặt tên: **Danh từ-Noun**, ví dụ: **Staff**, **Student**, **Admin**, **Cashier**...
- ✓ Các Actor đặc biệt:
  - ✓ **Guest/Visitor**: Kẻ dùng App mà không để lại danh tính, kẻ chưa/không login.
  - ✓ **System Handler:** App có tính năng dính dáng đến "tự động hóa" theo lịch biểu, theo trạng thái data kiểu scheduling, reminder, suggestion, push notification. Hậu trường làm App chính là **đoạn code tự động thực thi** theo 1 thuật toán và trạng thái data nào đó.
  - ✓ **MoMo**, **PayPal**, **Gmail**, **FB**,…: Khi App chúng ta có giao tiếp với Hệ thống bên ngoài (external systems) thì các hệ thống này cũng xem như là actor.



2. SRS: User Requirements *(cont.)*

Actor Checklist

- ✓ **KHÔNG BAO GIỜ** đặt tên 1 actor là: **User**, và chỉ có từ User đứng một mình; vì User rất rộng và bao quát về ý nghĩa.
- ✓ Ai đứng trước App, cho dù **có login hay không**, mà dùng được App ở vài chức năng nào đó, thì **đều là** User cả mà.
- ✓ Đúng nghĩa phải là XYZ User, ví dụ Student User, Customer User, Visitor Users, Un-authenticated User, nhưng ta nên **lược bớt** từ User đi cho đỡ rườm rà mà ai đọc cũng vẫn hiểu đúng ngữ nghĩa.
- ✓ **Hãy** gọi tên, đặt tên **các user** theo **Role** mà họ đảm nhận trong App sau này: Guest, Student, Buyer, Supplier,…



2. SRS: User Requirements *(cont.)*

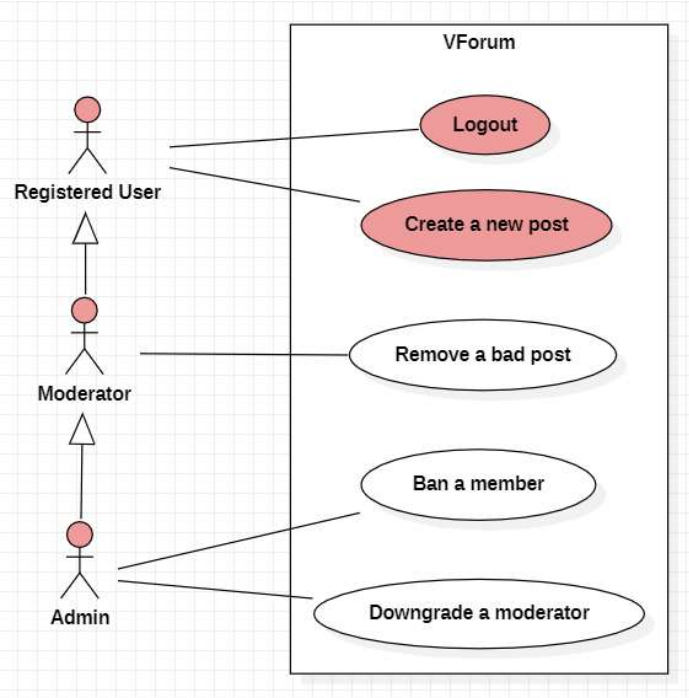
Sự kế thừa chức năng của các actors

✓ **Lập ma trận phân bố tính năng theo Actor-Use Case, tìm điểm chung**

	Actor 1	Actor 2	Actor 3	Actor 4	Actor 5
UC 1	X	X	X	X	X
UC 2		X	X	X	X
UC 3		X	X	X	X
UC 4	X			X	X
UC 5	X			X	X
…					

2. SRS: User Require...

Sự kế thừa chức năng của các actors



Quy Định Của Diễn Đàn - Diễn đàn Tin học Công nghệ - Vforum.vn

Bạn phải đồng ý với những điều dưới đây để tiếp tục đăng ký thành viên!

A Nội quy dành cho member

I. Nội quy thành viên

- Đăng kí thành viên:

II. Những quy định chung

1.Lập chủ đề

2. Về nội dung bài viết :

- Không đăng ký ID bằng các ký tự đặc biệt như: [* ¤ ° ´ ¯)]

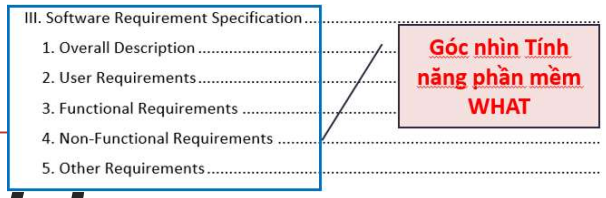
B. Phần dành cho BQT (Admin, Smod, Mod)

1.Admin

- Cấp quyền mod, Smod, admin để điều hành diến đàn.
- Mở các box mới theo yêu cầu biểu quyết của đa số mem
- Tổ chức các cuộc thi theo yêu cầu của mem hoặc tự tổ chức
- Theo dõi, xử lý các sự cố liên quan đến kỹ thuật của diễn đàn.
- Nghiên cứu, cài đặt các mods/hacks/plugins cho diễn đàn.
- Đăng bài quảng cáo, quảng bá trên các diễn đàn, website khác.
- Quản lý Nhân sự
- Ban/unban thành viên.
- Huấn luyện kỹ thuật, phát triển kỹ năng viết cho Smod.
- Theo dõi, biên tập, xử lý và chịu trách nhiệm cuối cùng cho tất cả
- Đề xuất hủy/cấp quyền Mod.
- Đề xuất mở/đóng các boxes, sub-boxes

3. Moderators

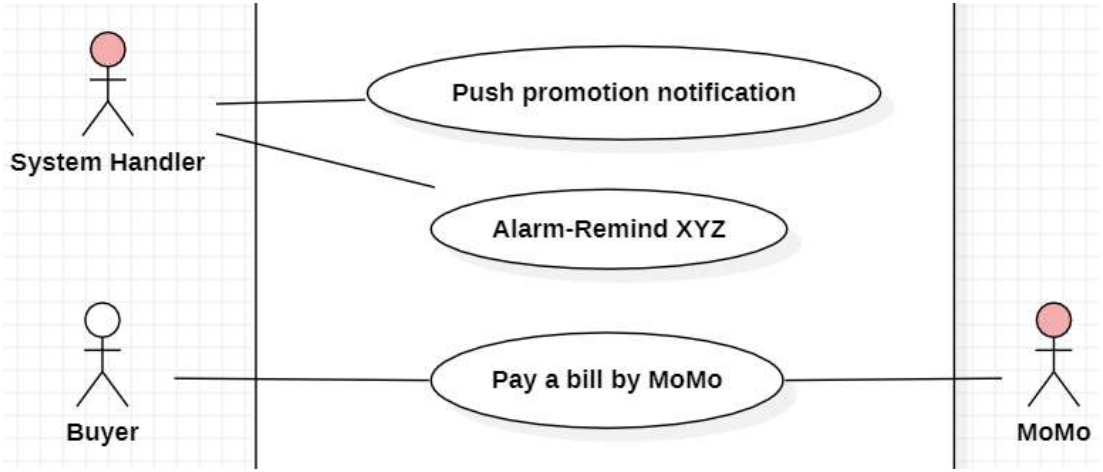
- Là những người giúp Ban Quản Trị điều hành diễn đàn.
- Có quyền quản lý bài viết trong các box được bổ nhiệm
- Ngoài ra những người này đều như thành viên bình thường ở các box khác.
- Phải vào forum thường xuyên
- Thông thạo về lãnh vực của box mà mình quản lý.
- Sửa/xóa ngay những bài đi lạc với chủ đề hay có nội dung không nhằm phát triển dung bài vi phạm

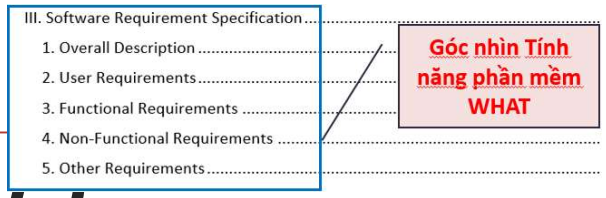


2. SRS: User Requirements *(cont.)*

Actor Checklist

- ✓ App của bạn có tính năng nhắc nhở tự động người dùng điều gì không? Có hả **System Handler** đó!
- ✓ App của bạn có tương tác với App khác không? App khác, App bên ngoài này sẽ là Actor!
- ✓ **Có thể lược bỏ** Actor MoMo cũng không khiến mất đi ý nghĩa của sơ đồ.

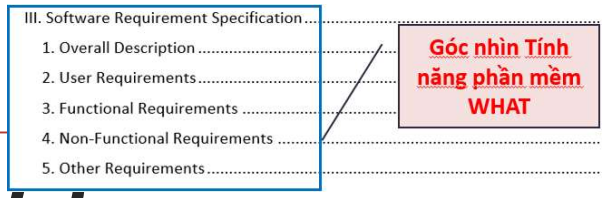




2. SRS: User Requirements *(cont.)*

Use Case Checklist

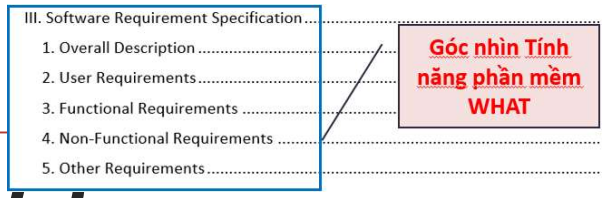
- ✓ Use Case (UC): Là một chức năng, tính năng của App.
- ✓ Ngầm hiểu là 1-hoặc-vài màn hình có ô nhập và nút nhấn để xử lí thông tin nhập vào, tạo ra **một kết quả** có ý nghĩa nào đó cho người dùng – kết quả này cần cho công việc hàng ngày của user.
- ✓ Tên gọi khác: user-story, feature, functional requirement, function, what-goal-target, đích đến.
- ✓ Mỗi UC là, phải là **1 xử lí**, **1 đích đến**, **1 câu chuyện** của các thao tác, **1 tình huống** làm việc qua App để App giúp khách hàng/user làm được điều gì đó.
- ✓ Quy ước đặt tên: **Verb + Object**, ví dụ: **Create an order**, **Book a tour**.



2. SRS: User Requirements *(cont.)*

Use Case Checklist

- ✓ Các UC **KHÔNG** được trùng tên nhau. Nếu trùng tên có nghĩa là các user cùng dùng chung tính năng/UC đó. Tớ, cậu, thầy-cô, sếp, nhân viên, khách hàng, ba-má đều được sửa Profile, vậy chỉ có 1 UC **Edit Profile** dùng chung cho cả đám.
- ✓ **KHÔNG** nhầm lẫn UC với steps để hoàn tất UC đó.
- ✓ Step **không trả về** một kết quả có ý nghĩa với user. UC là 1 xử lí và **trả về** một kết quả có ý nghĩa với user. **UC là đích đến, Steps là hành trình trên đường!**
- ✓ **(Input username), (Input password), (Hit login button)** chỉ là các bước thực thi hay hành trình để tạo ra kết quả có ý nghĩa cuối cùng là ai đó được xác thực.
- ✓ Do đó **(Login)** sẽ là UC, **(~~Input username…~~) chỉ** là step của UC **(Login)**.
- ✓ **(~~Input username~~)** không là UC vì nó không xử lí gì cả để trả về thứ user cần.



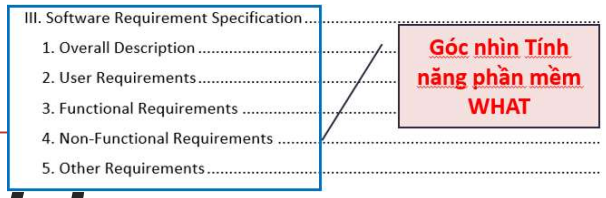
2. SRS: User Requirements *(cont.)*

Use Case Checklist

- ✓ UC **Remove X** và UC **Update X** là 2 UC khác nhau.
- ✓ Ở góc nhìn người dùng (**UCD là góc nhìn người dùng**) hành động **Xoá** và **Chỉnh sửa** là 2 hành động khác nhau. **Xoá** là **mất** khi **F5** (load lại trang). **Update** là **xuất hiện** value **mới** khi **F5**.
- ✓ Ở góc nhìn **viết code**, thì **Xoá** cái gì đó **không hẳn** là xoá, mà **Xoá là update** cột status trong table tương ứng thành trạng thái "bị-xoá"…

  Lệnh SQL ở hậu trường: `Update X set status = 'trạng-thái-xoá'`
- ✓ Ở góc nhìn **viết code**, thì **Update** cái gì đó cũng dùng chung lệnh SQL Update, nhưng ý nghĩa đã khác, có thể **thay đổi nhiều value** khác, nhiều cột khác trong cùng table.

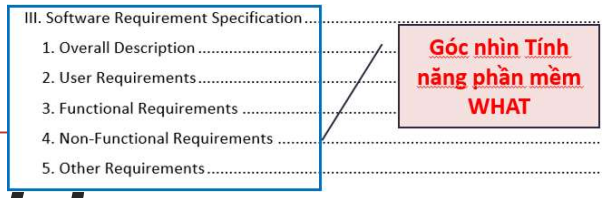
  Lệnh SQL ở hậu trường: `Update X set cột-A = 'value-mới', set cột-B = 'value-mới'…`
- ✓ Dù sao thì người dùng **không quan tâm** cùng lệnh SQL hay không; họ chỉ biết là 2 hành động/2 UC này khi thực thi sẽ ra 2 kết quả khác nhau. **Do đó cần tồn tại 2 UC Remove và Update**.



2. SRS: User Requirements *(cont.)*

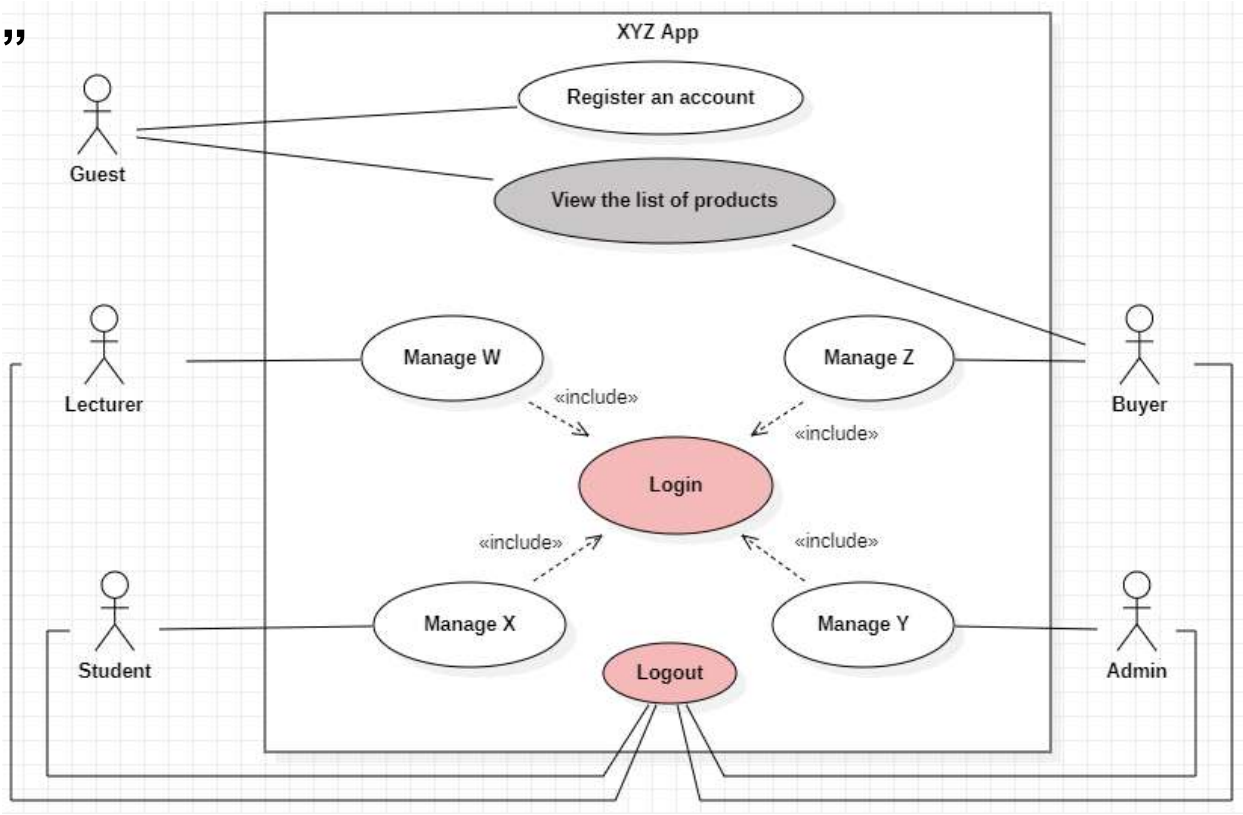
Use Case Checklist

- ✓ **KHÔNG ĐƯỢC VẼ GỘP UC**: UC **CRUD** là không ổn vì mỗi UC là một câu chuyện riêng biệt của thao tác người dùng.
- ✓ Mỗi UC **phải** trả về một kết quả riêng biệt theo mục đích sử dụng của user.
- ✓ **CHỈ ĐƯỢC** vẽ UC gộp nếu bạn vẽ UCD theo nhiều **tầng layer**, từ layer tổng quát chung chung gom "cụm" các chức năng đi xuống đến UCD chi tiết như đang đề cập.
- ✓ UC gộp **CHỈ** dành cho bài toán rất rất to và bạn muốn vẽ gộp để nhìn được bức tranh tổng thể các **"cụm"** chức năng của cả App to mà chưa vội đi vào chi tiết!
- ✓ Nhưng **cuối cùng** bạn vẫn **cần** phải vẽ UC tách riêng để Dev team hiểu rõ từng chức năng cụ thể mới biết đường mà cài đặt code.
- ✓ Khái niệm vẽ UCD từ tổng quát xuống chi tiết gọi là kĩ thuật **Decomposition** – Phân rã một sơ đồ UC. Quy mô Capstone Project của bạn **chưa cần** dùng đến món này!!!



2. SRS: User Requirements *(cont.)*

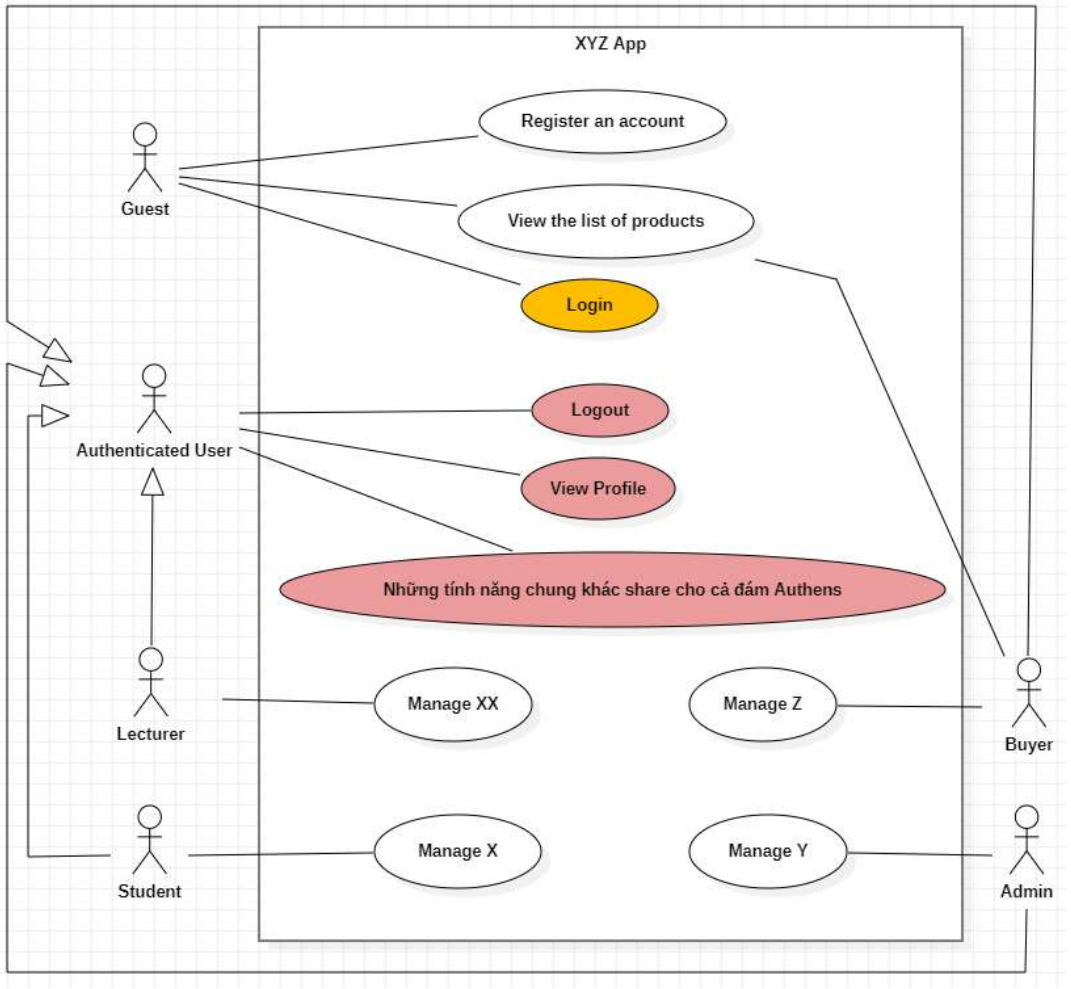
Use Case "Bad Smells"

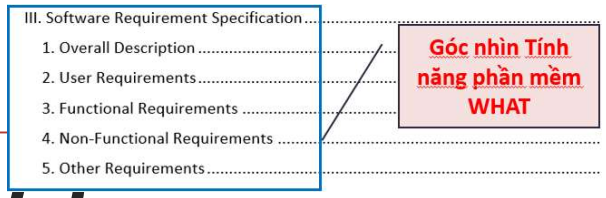


2. SRS...

Use Case "Mlem"??? NO!!!

- ✓ Đứng trước App mà chưa login thì ai cũng như ai, làm được vài thứ cơ bản với App – gọi chung là Guest.
- ✓ **Guest cần login** để thành ROLE nào đó trong App.
- ✓ Student, Lecturer, Buyer… chính là **ROLE**, bản chất là nói về trạng thái **đã login** và **được phân quyền**.





2. SRS: User Requirements *(cont.)*

Câu chuyện UC "Login Dilemma"

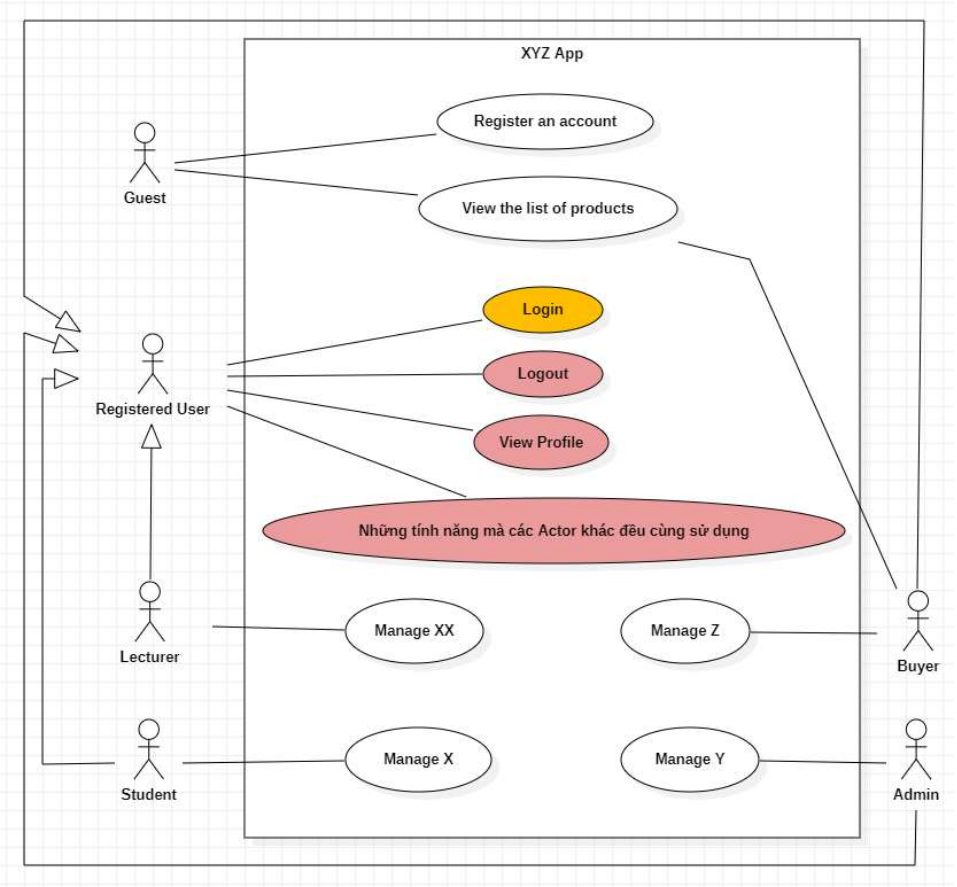
- ✓ **Có nên** xem Login như là **Precondition**, điều kiện tiên quyết để các User-được-phân-quyền làm những việc họ được quyền làm trong App và khi đó **bỏ luôn** UC Login này, mà **chỉ ghi** Login trong UC Specification – UC Description là đủ?
- ✓ **Hay chơi trò "Bad-Smells"** hiện đang minh họa tốt cho tình huống Login với ngữ cảnh <<include>>? Login đủ phức tạp mà – nếu gọi thêm API ngoài!!! Nhưng nhìn sơ đồ sẽ **rất rối**!
- ✓ **Hay dùng "Mlem"** cho **Guest Login** để **"đổi vai"** thành các Roles, dùng Authenticated User như là Abstract User để gom cụm tính năng chung share giữa mọi User được xác thực? Sơ đồ **giảm** được nhiều nét giao cắt – **NHƯNG Guest nối Login thấy Kì Kì at first glance!!!** Và sơ đồ UC lúc này có một **hàm ý thứ tự thực thi** từ **Guest** thành **Ai-đó**? **UCD không** hàm ý thứ tự thực thi mà chỉ là liệt kê!!! **Haiz, nan giải!!!**

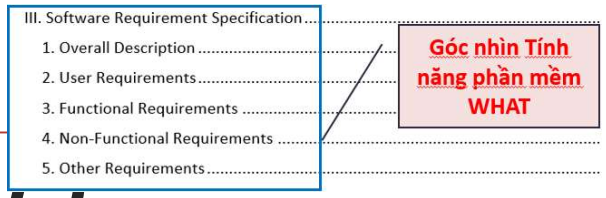
ĐÁP ÁN BEST PRACTICE Ở TRANG SAU!!!

2. SRS...

Use Case "Mlem chốt deal"

- ✓ **Actor** là tên gọi cho chủ thể có tương tác với App với ngữ nghĩa tự thân theo phân loại, không nói về trạng thái lúc vận hành.
- ✓ **Guest**: Những user không có account trong hệ thống. User này sẽ dùng được những tính năng không đòi hỏi xác thực.
- ✓ **Registered User**: Những user có danh tính, account, "người quen" trong hệ thống. User này sẽ dùng những tính năng đòi hỏi xác thực.
- ✓ **Use Case** được thiết kế ra để ai đó có thể vận hành được ở **"main success scenario"**. Ví dụ: Kẻ có account thì login mới có ý nghĩa!!!
- ✓ Login được xem là **"pre-condition"** để thực thi một UC đòi hỏi xác thực.

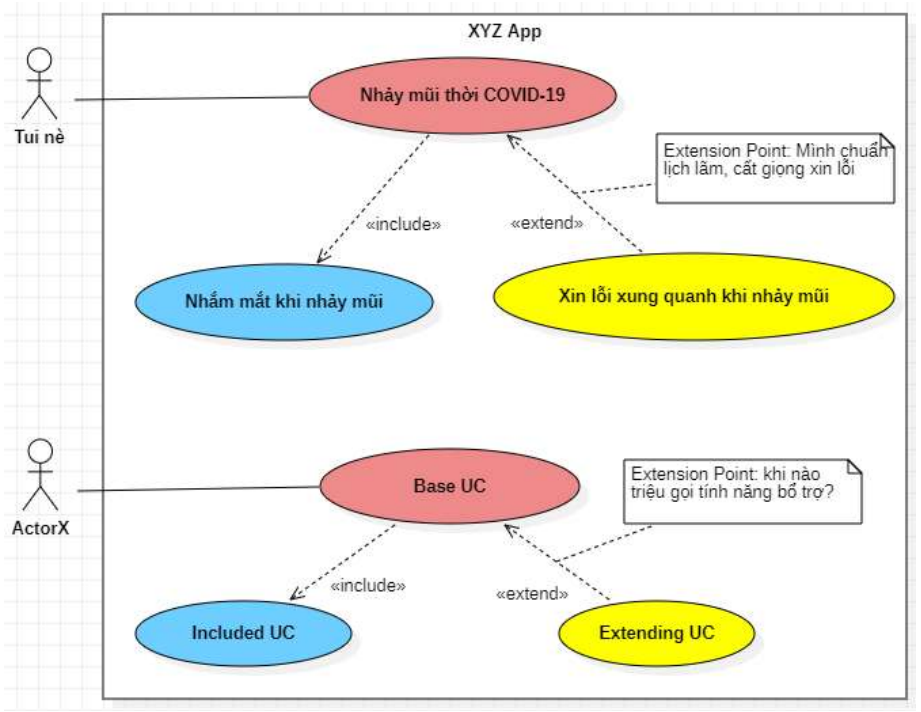




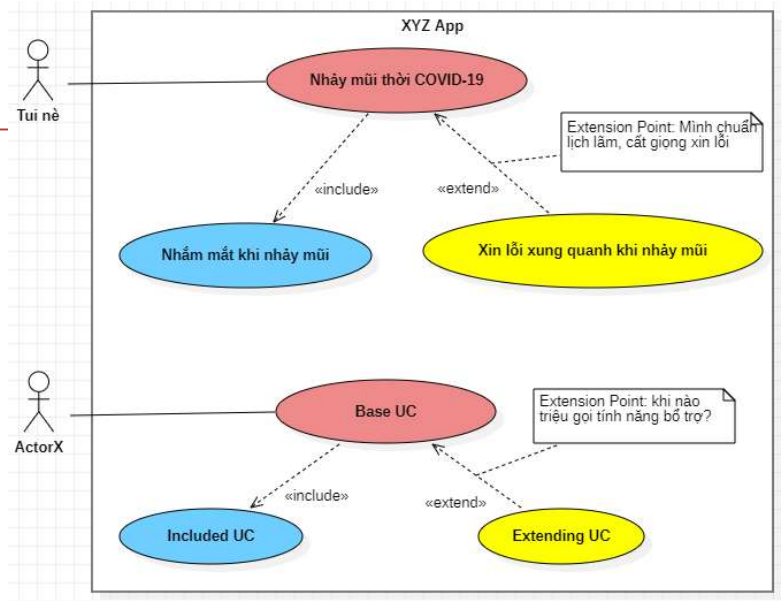
2. SRS: User Requirements *(cont.)*

Cách đọc <<include>>

- **<<include>>** đọc từ **gốc** mũi tên (**Base UC**).
  - ✓ **Nhảy mũi muốn** thành công **bắt buộc** cần **Nhắm mắt**.
- **Include** giống `#include, import, using` trong lập trình.
- Dùng khi có cụm hay phân đoạn xử lí được dùng chung cho các UC khác nhau. Mang ý nghĩa tách xử lí chung thành thư viện. Ví dụ: Login cần dùng cho nhiều tính năng khác nhau!!!
- Phụ thuộc **cực kì nghiêm ngặt**. **Base UC** muốn hoàn tất **bắt buộc** phải hoàn tất **Included UC.** Ví dụ: Muốn rút được tiền phải login thành công.



2. SRS: User Requirements



Cách đọc <<extend>>

- **<<extend>>** đọc từ **ngon** mũi tên (**Base UC**).
  - ✓ **Nhảy mũi đã** thành công, **tuỳ hứng Xin lỗi** hay không!
- **Extend** giống món gọi thêm, cơm thêm, thích thì gọi, không thích thì thôi.
- **Extend** phụ thuộc **cực kì lỏng lẻo, optional**. **Base UC** hoàn toàn độc lập với **Extending UC**. Và mỗi đứa có một vai trò độc lập riêng. **Base UC** hoàn tất mà **chẳng cần đoán hoài, chẳng care Extending UC**.
- Dùng để **liên kết** các màn hình cho user tiện sử dụng, nhưng **không nên lạm dụng** vẽ thành chuỗi flow màn hình, mất chất UCD.
- **Ví dụ:** Search **xong rồi** ta hay nhấn thêm View để xem details, không nhấn chẳng sao.
- Sinh viên **hay nhầm** với Kế thừa trong Java. **Không phải, chẳng liên quan gì cả!!!**. Keyword kế thừa trong Java là "**extends**" nhen, chữ **s** to lắm đó!!!

2. SRS...

Lựa chọn style khi vẽ cụm tính năng

- ✓ Style <<extend>> nhắm đến việc **liên kết màn hình**.

  HAY/OR?

- ✓ Style generalization – kế thừa nhắm đến **gom cụm tính năng, gom menu quản trị**.

III. Software Requirement Specification
1. Overall Description
2. User Requirements
3. Functional Requirements
4. Non-Functional Requirements
5. Other Requirements

Góc nhìn Tính năng phần mềm WHAT

2. SRS: User Requirements *(cont.)*

Use Case Specification – Use Case Description

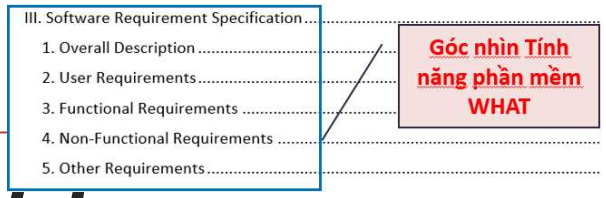
- **Vui lòng tham khảo tài liệu *Seminar Notes* đính kèm, có giải thích chi tiết…**

- **Ví dụ 3: Minh họa về UC Description viết theo style 2 cột bắn qua bắn lại – "Mlem":**

Use Case Specification			
Use Case No.:	<Mã số UC>		
Use Case Name:	<Tên UC>		
Created By:	<Ai viết UC Spec này>		
Date:	<Ngày viết>	**Priority:**	<Mức độ ưu tiên cần hiện thực tính năng này, ví dụ: Must Have | High>
Actors:	<Những user nào sử dụng tính năng này>		
Summary:	<Mô tả ngắn gọn mục đích của UC này>		
Trigger:	<Khi nào UC này được gọi, *ví dụ: Admin clicks button "Update product"*>		
Preconditions:	<Điều kiện tiên quyết cần có trước đó để UC này có thể chạy, *ví dụ: data/thiết bị sẵn dùng là...; user cần phải login trước khi sử dụng*>		
Post–conditions:	<Sau khi UC thực thi xong và thành công, hiện trạng hệ thống là gì, user đạt được điều gì; có thể liệt kê thêm kết quả của tình huống UC thực thi thất bại, *ví dụ: Đơn hàng được ghi nhận và lưu trữ*>		

Main Success Scenario/Main Flow/Normal Flow/Main Path:

Step.	Actor Action	System Response
1	<Luồng xử lí chính, trường hợp Happy Case, người dùng hay làm những điều này để đạt được mục đích UC như đã thiết kế> <Bước 1 người dùng làm gì/nhấn gì/nhập gì>	<Hệ thống xử lí/lưu trữ/hồi đáp lại người dùng cái gì>
2	<Bước 2 người dùng làm gì/nhấn gì/nhập gì tiếp> **[Alternative 1]** Có thể người dùng chọn rẽ nhánh khác để cũng đạt được mục đích UC. Các [Alternative X] được đánh số thứ tự tăng dần	<Hệ thống xử lí/lưu trữ/hồi đáp lại người dùng cái gì> **[Exception 1]** Có thể có tình huống ngoại lệ xảy ra ở bước này Các [Exception X] được đánh số thứ tự tăng dần
3	...	...

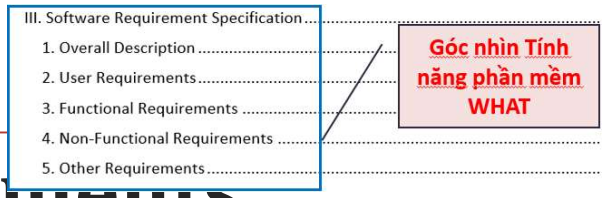


2. SRS: User Requirements *(cont.)*

Use Case Checklist

- ✓ **ĐỪNG QUÊN** các UC liên quan đến tính năng quản trị data phía hậu trường để tạo ra mọi dữ liệu sẵn dùng cho người dùng cuối. Module **Admin** để **CRUD**.
- ✓ Hãy hỏi câu hỏi: làm sao có các thông tin sản phẩm ở trên trang thegioididong.com cho chúng ta (**End-User**) duyệt và mua?
- ✓ Hãy hỏi câu hỏi: làm sao có nhiều món đồ của nhiều shop bán đồ trên Lazada, Shopee, Tiki, Sendo?
- ✓ Chúng ta (**End-User**) cực kì đông đảo nhưng ít quyền trên App – nhưng cực kì quan trọng vì đem lại doanh thu – ta cần Module **End-User UI/UX xịn sò**.
- ✓ Phía quản trị data ít người nhưng nhiều quyền trên App. **UI cơ bản để CRUD!!!**
- ✓ Chỉ bàn về quyền trên App thôi nhen! Vì Admin cũng **chỉ là** nhân viên mà thoy!!!

4. SRS: Non-Functional Requirements



Ví dụ về Non-Functional Requirements

- **Vui lòng tham khảo tài liệu *"Cheat Sheet - Tóm lược kiến thức môn SWE102"***

  URL: *https://github.com/doit-now/cheat-sheet*

Nonfunctional Requirement	Description	Examples
Operational	The physical and technical environments in which the system will operate	■ The system can run on handheld devices. ■ The system should be able to integrate with the existing inventory system. ■ The system should be able to work on any Web browser.
Performance	The speed, capacity, and reliability of the system	■ Any interaction between the user and the system should not exceed 2 seconds. ■ The system downloads new status parameters within 5 minutes of a change. ■ The system should be available for use 24 hours per day, 365 days per year. ■ The system supports 300 simultaneous users from 9–11 A.M.; 150 simultaneous users at all other times.
Security	Who has authorized access to the system under what circumstances	■ Only direct managers can see personnel records of staff. ■ Customers can see their order history only during business hours. ■ The system includes all available safeguards from viruses, worms, Trojan horses, etc.
Cultural and Political	Cultural and political factors and legal requirements that affect the system	■ The system should be able to distinguish between U.S. currency and currency from other nations. ■ Company policy is to buy computers only from Dell. ■ Country managers are permitted to authorize custom user interfaces within their units. ■ Personal information is protected in compliance with the Data Protection Act.

Source: The Atlantic Systems Guild, http://www.systemsguild.com

Metrics for specifying nonfunctional requirements

Property	Measure
Speed	Processed transactions/second User/event response time Screen refresh time
Size	Mbytes Number of ROM chips
Ease of use	Training time Number of help frames
Reliability	Mean time to failure Probability of unavailability Rate of failure occurrence Availability
Robustness	Time to restart after failure Percentage of events causing failure Probability of data corruption on failure
Portability	Percentage of target dependent statements Number of target systems

© 2022 – SE Department – FPT University | HCMC

THANK YOU

Thanks to the Internet for the slide template and images