

FBA-HN23 Learning Notes

Nguyễn Thái Sơn, BAC Vietnam

Mục lục

FBA01 – Business Analysis Profession	2
#1. Business Analysis là gì?	2
#2. BA Roles trong các mô hình công ty	2
#3. Các loại requirements	3
#4. Các loại stakeholders	5
#5. BA theo Core Concept Model của IIBA	6
FBA02 – BA Approach and Planning	6
#1. Quy trình phát triển & quản lý yêu cầu	6
#2. Phân tích Stakeholder	7
FBA03 – Requirement Elicitation	7
#1. Các kỹ thuật khai thác yêu cầu	7
#2. Tìm hiểu scope và quản lý scope yêu cầu	9
FBA04 – Requirement Analysis	10
#1. Actor Table	10
#2. Activity Diagram	11
#3. Cross-functional Diagram / Swimlane	11
#4. Business Policies	11
#5. Business Rules	11
#6. Status Transition Diagram	12
#7. Use Case Diagram	12
#8. Decision Table	12
#9. Concept Mapping Diagram	13
FBA05 – Documentation	13
#1. Business Requirements Document	13
#2. User Requirements Document	14
#3. Software Requirements Document	14
FBA06 – UX Part 1: UX Analysis Overview	15
#1. Nhiệm vụ của UX	15
#2. Thế nào là “dễ dùng” trên web application?	15
#3. Thế nào là “dễ dùng” trên mobile application?	16
FBA07 – UX Part 2: Wireframing	16
#1. Các phương pháp làm Information Architecture	16
#2. Vai trò của Wireframe	17
FBA08 – Requirement Validation & Management	17
#1. Phân biệt Validate và Verify Requirements	17

#2. Thế nào là một bản requirements xin?	17
#3. Các cách quản lý yêu cầu	21
FBA09 – Agile Perspective	21
#1. Ưu thế của Agile (so với Waterfall)	21
#2. Cách xây dựng User Story	21
#3. Cách làm Feature Roadmap (dựa vào Prioritization trong Agile)	22
FBA10 – BA Career Development	22
#1. Tổng hợp Phân tích nghiệp vụ (Tips & Tricks)	22
#2. Các hướng tiếp cận làm Requirements	23
#3. Vai trò BA trong Agile	23
#4. Định hướng nghề BA	24
#5. Phân biệt Job vs. Career	25
Tổng kết	26

FBA01 – Business Analysis Profession

#1. Business Analysis là gì?



Một cách ngắn gọn nhất có thể, Phân tích Nghiệp vụ gồm:

- **Business** (kiến thức chuyên môn, kiến thức nghiệp vụ): là một trong những yêu cầu cơ bản cho người BA phải đáp ứng trong quá trình làm dự án. Mỗi một ngành nghề, bộ phận lại có những yêu cầu về chuyên môn nghiệp vụ riêng để phục vụ đúng cho ngành nghề hoặc bộ phận đó.






Ví dụ: cùng là bài toán đặt phòng (*room booking*), nhưng đặt phòng (khách sạn, homestay) khi đi du lịch khác với đặt phòng (hợp nhóm, làm việc riêng) trong bài toán coworking space. Khác nhau về quy trình, thông tin phòng đặt, cách xuất hoá đơn, và làm thanh toán.

- **Analysis** (kỹ năng & tư duy phân tích, tổng hợp): là khả năng hình dung, phân loại, sắp xếp, suy luận, làm rõ, khái niệm hóa các vấn đề từ phức tạp đến đơn giản bằng cách vận dụng tư duy logic, đưa ra các mô hình, lựa chọn các hướng đi, ra quyết định hợp lý dựa trên các thông tin hiện có.

Nhiệm vụ của BA là xây dựng được requirements phù hợp và chính xác.

#2. BA Roles trong các mô hình công ty

Cũng cần định vị vai trò của BA trong mô hình của công ty phần mềm, ví dụ outsourcing sẽ khác với làm product trực tiếp cho end-customer.

	Individual or Company (End Customer) (E.g., B2C or B2B)	An Enterprise (A) (E.g., client company signed contract with SEA-Solutions)	Outsourcing Company (B) (E.g., SEA-Solutions - as software solution company)
Roles	 End Customer	 Product Owner (as Customer Representatives)  Other Stakeholders (Sponsor, Managers, Other Departments)	 Business Analyst  Development Team
Methodology		Product Development <ul style="list-style-type: none"> End-to-end (for end customers), from product discovery to development, and operations Provide solutions and bring benefits to end-customer's needs Product thinking / mindset 	Software Development <ul style="list-style-type: none"> Follow the contract / agreement between A & B, normally not end-to-end development Provide solutions as agreed to A in the contract Software thinking. Product mindset is a plus
Approach		Customer-centric requires	Product-centric requires, Customer-centric is a plus

Theo đó có 4 loại vai trò BA chính:

- (1) **Outsourcing BA**: là BA trong công ty outsourcing. BA (công ty B) làm việc nhiều với các đầu mối bên A, thường có người đại diện cho end-customer là Product Owner. Công việc của BA tập trung **chủ yếu vào xây dựng software requirements**, một số dự án có thể phải làm cả user requirements. Outsourcing BA đòi hỏi nhiều về kỹ năng chuyên môn, hơn là kiến thức nghiệp vụ (vì cần phục vụ các khách hàng A khác nhau, theo các domain khác nhau). **Lý tưởng: có kỹ năng chuyên môn của outsourcing BA nhưng với product mindset của product BA.**
- (2) **Product BA**: là BA trong công ty làm sản phẩm (*product-led*). Có thể chia tiếp thành 2 loại nhỏ (tùy theo quy mô từng công ty): BA IT / Technical và BA Nghiệp vụ. Product BA yêu cầu hiểu biết nhiều về nghiệp vụ, về data-driven, về customer-centric. Trong nhiều công ty sản phẩm, Product BA này được gọi là Product Owner. **Lý tưởng: có kỹ năng chuyên môn của product BA và kinh nghiệm của technical / outsourcing BA.**
- (3) **Solution BA**: là BA trong các công ty làm triển khai giải pháp đóng gói (ví dụ: SAP, ERP, CRM). Người BA này thường yêu cầu phải có kiến thức nghiệp vụ (theo ngành) và hiểu biết về giải pháp sẽ triển khai. Kỹ năng phân tích requirements sử dụng nhiều nhất là *fit-gap analysis*¹.
- (4) **In-house BA**: là BA trong các development team chuyên xây dựng giải pháp nội bộ cho công ty. Vai trò này có thể theo hướng generalist BA², không cần nắm vững domain cụ thể nhưng cần biết áp dụng các kỹ thuật phân tích một cách linh hoạt vào từng bài toán.

#3. Các loại requirements

Có 3 loại chính

¹ Fit Gap Analysis: <https://www.linkedin.com/pulse/fit-gap-analysis-james-hu/>

² Generalist BA: <https://www.linkedin.com/pulse/generalist-specialist-path-business-analyst-jamie-champagne/>

Business Requirements	User Requirements	Software Requirements
Yêu cầu nghiệp vụ là cơ sở lập luận, là căn cứ chính đáng cho việc hình thành dự án. Cũng là nhu cầu khái quát của khách hàng. Yêu cầu này bao gồm tầm nhìn của sản phẩm phần mềm, khái quát chung về mục đích kinh doanh cần hướng tới, và chiến lược kinh doanh của công ty.	Yêu cầu người dùng là yêu cầu cho phần mềm từ góc độ người dùng cuối. Yêu cầu mô tả 1- các tác vụ mà người dùng cần hoàn thành khi sử dụng phần mềm, và 2- các tiêu chí chất lượng (quality characteristics) của hệ thống phần mềm cần đáp ứng để thoả mãn người dùng. Yêu cầu người dùng cần phục vụ yêu cầu nghiệp vụ.	Yêu cầu Phần mềm mô tả chi tiết tất cả chức năng và phi chức năng mà phần mềm cần đáp ứng để thoả mãn nhu cầu người dùng, trong khi vẫn phải đảm bảo các ràng buộc kỹ thuật thực thi và công nghệ thiết kế. Yêu cầu Phần mềm cũng giống như một hợp đồng về các chức năng sản phẩm cần đáp ứng giữa bên khách hàng với bên phát triển.

Một số lưu ý khi viết business requirements:

- Luôn đứng từ quan điểm của công ty khách hàng
- Nội dung đủ khái quát, chưa cần quá chi tiết
- Không phải mục tiêu của công ty, mà hỗ trợ công ty đạt được mục tiêu

Trong đó cho các BA nói chung tập trung vào User & Software Requirements. Nhấn mạnh: đối với mọi BA, dứt khoát phải phân biệt được 2 loại yêu cầu này (khác nhau là gì, tập trung vào đối tượng nào, chức năng & mục đích mỗi loại).

Đăng nhập (Login)

Xem xét ví dụ sau cho chức năng Đăng nhập (Login) của người dùng (customer) vào hệ thống mua hàng online.

Business Requirements

BR-01: Dữ liệu khách hàng và thông tin mua hàng của họ phải được giữ bảo mật, đảm bảo tính an toàn thông tin, tránh bị truy cập bởi người dùng khác.

User Requirements

UR-01: Authentication (chứng thực người dùng)

Khách hàng có thể đăng nhập ứng dụng mua hàng qua các kênh khác nhau, bao gồm:

- 1) Một form login trên web, sử dụng Email và Mật khẩu
- 2) Login qua mobile app, sử dụng Email và Mật khẩu. Thông tin này có thể được ghi nhận lại để không phải nhập trong những lần truy cập sau.
- 3) Login qua mobile app sử dụng vân tay.

UR-02: Authorisation (phân quyền)

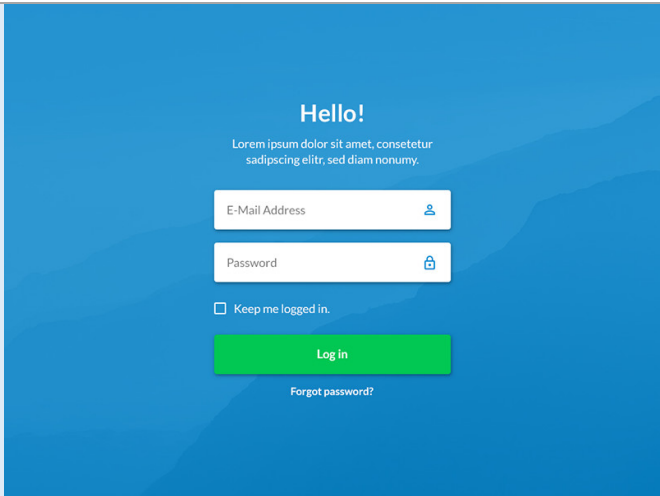
Để phục vụ mục đích mua hàng, khách hàng được phép truy cập các trang / chức năng: tìm kiếm và xem danh sách hàng hoá, chọn hàng hoá cụ thể, xem chi tiết một món hàng, xếp vào giỏ hàng, xem giỏ hàng, nhập thông tin thanh toán, và mua hàng.

Landing Page: Sau khi login thành công, trang xem danh sách hàng hoá mặc định sẽ xuất hiện trên màn hình của người dùng.

Software Requirements

Từ yêu cầu người dùng UR-01.1 (phần cho web login) phát triển thành các yêu cầu phần mềm như sau. Hệ thống cho phép người dùng đăng nhập qua Email và Mật khẩu trên màn hình Đăng nhập.

[minh hoạ wireframe hoặc mockup màn hình Đăng nhập]



SR-01: Hệ thống đối chiếu Email và giải mã Mật khẩu nhập vào có đúng với thông tin người dùng hiện có trong hệ thống hay không. Nếu chính xác, hệ thống sẽ hiển thị landing page của người dùng kèm nội dung menu các chức năng cho phép truy cập
[Xem thêm yêu cầu phần mềm cho **Authorisation** để biết cụ thể danh sách chức năng được phép].

Ngược lại, hệ thống hiển thị thông báo lỗi MS-01.1 [Bổ sung phần Danh sách Thông báo và cảnh báo lỗi].

SR-02: Cho mỗi người dùng, hệ thống cần lưu trữ Email và 5 Mật khẩu gần nhất, các mật khẩu đều phải mã hoá sử dụng một trong ba phương thức SHA-1, SHA-2, hoặc MD5.
[Đội kỹ thuật lựa chọn một phương thức cụ thể, và sử dụng xuyên suốt toàn hệ thống].
Hệ thống không được phép lưu trữ Mật khẩu mà không mã hoá.

SR-03: Nếu người dùng nhập Mật khẩu và bị hệ thống báo sai quá 2 lần trong 1 ngày, thì người dùng sẽ nhận được thông tin cảnh báo rằng chỉ còn duy nhất 1 cơ hội đăng nhập trước khi tài khoản bị khoá. Nếu lần thứ 3 vẫn bị sai mật khẩu, thì hệ thống sẽ khoá tài khoản người dùng.

Xem yêu cầu SR-xx cho chức năng Mở khoá và Đặt lại Mật khẩu.
Chú thích: làm tương tự cho chức năng “Keep me logged in”

Cách biểu diễn các loại requirements theo hình pyramid cũng liên tưởng đến “shape” / scope mà một BA phải nắm được và giúp quản lý scope tốt hơn.

- Nếu KH đưa ra requirements trong khi dự án đang chạy, BA đánh giá và clarify lại với KH, để đưa requirements về “hình” ban đầu nhưng lớn hơn => Như thế gọi là “nắm được” requirements mới trong bối cảnh các requirements cũ đang có.
- Nếu trong khi dự án đang chạy, BA hoặc team phát hiện ra một tính năng / logics nào đó mà scope ban đầu không có, BA cần xác nhận với KH để chắc chắn issue đó là new requirement (then, new shape) hay out-of-scope (keep as-is shape). => Như thế gọi là “nắm được” các requirements đang có.

#4. Các loại stakeholders

Có 4 loại biểu diễn theo sơ đồ Củ hành, đặc biệt quan tâm 2 đối tượng (từ trong ra ngoài):

- Vòng 2 (*business group*): những người trực tiếp sử dụng giải pháp <= BA phải đóng vai những người này để hình dung nhu cầu và hành vi của họ.
- Vòng 3 (*organization group*): những người có trách nhiệm với giải pháp <= BA thường xuyên làm việc với những người này để khai thác & phân tích thông tin, đưa vào requirements.

#5. BA theo Core Concept Model của IIBA

Sử dụng mô hình Con rùa không chỉ để nắm được các khái niệm và công việc của BA một cách khái quát nhất, mà còn sử dụng theo dạng “checklist” đảm bảo đã phân tích đủ 6 góc cạnh của bài toán.

Ví dụ cho chức năng Login ở trên:

- Change (trước và sau khi khách hàng có chức năng Login): ví dụ trước đây không cần login KH vẫn mua hàng được => dẫn đến lỗ hổng trong quản lý thông tin tài khoản, và KH cũng không thể quản lý các orders của họ, không tracking được.
- Needs (nhu cầu Login dùng username & password): để giải quyết issue trên
- Stakeholders: khách hàng
- Solution: sử dụng security matrix đơn giản, áp dụng cho đối tượng người dùng là KH bằng một màn hình username & password.
- Context: ví dụ đặt trong bối cảnh đang có nhiều rủi ro về an ninh thông tin khách hàng.
- Value: tăng tính bảo mật thông tin cho KH, giảm được rủi ro an ninh thông tin => giảm risk costs (chi phí khắc phục rủi ro).

FBA02 – BA Approach and Planning

#1. Quy trình phát triển & quản lý yêu cầu

Gồm 5 công đoạn: Elicitation – Analysis – Documentation – Validation – Management

Mục tiêu của phát triển yêu cầu là:

- Nắm được nhu cầu của các bên liên quan, và đạt được sự đồng thuận
- Hiểu được và hệ thống hoá bằng tài liệu các yêu cầu cũng như mong mỏi của người dùng.
- Quản lý yêu cầu nhằm giảm thiểu rủi ro của việc không đáp ứng yêu cầu người dùng (bởi nhu cầu các bên liên quan luôn thay đổi theo thời gian).

Quá trình phát triển gồm 2 phần chính:

- xây dựng yêu cầu (development), bao gồm từ Elicitation đến Validation
- quản lý yêu cầu (management)

Trong đó Xây dựng yêu cầu liên quan tới 4 công đoạn:

- **Khai thác (Elicitation)**: nhận biết stakeholder, xác định các nguồn thông tin cho requirements, và khai thác yêu cầu từ các nguồn này. Kết quả của elicitation là scope thống nhất được với các bên (một cách tốt nhất) hoặc ít nhất cũng là scope đề xuất cho các bên. Trong dự án, giai đoạn này thường gọi là *Discovery*.
- **Phân tích (Analysis)**: đề xuất và xây dựng yêu cầu, xác định tương tác người dùng với hệ thống, phát triển các mô hình yêu cầu người dùng, qua đó trao đổi với khách hàng và đội phát triển. Kiểm tra & rà soát yêu cầu theo các tiêu chí chất lượng. Sắp xếp độ ưu tiên để loại bỏ các yêu cầu không cần thiết, cũng để phục vụ việc ra quyết định.
- **Mô tả (Documentation)**: mô tả bằng cách viết **tài liệu**³ yêu cầu (chức năng, phi chức năng, các ràng buộc thiết kế). Tùy theo đối tượng quan tâm và mục đích sử dụng mà có các dạng tài liệu khác nhau. Kết hợp sử dụng các khuôn mẫu có sẵn (nếu công ty bạn cung cấp) hoặc tự xây dựng cấu trúc tài liệu. Cần tuân thủ các nguyên tắc về chất lượng yêu cầu. Xem thêm [Kỹ năng Viết](#) nói chung.
- **Xác minh (Validation)**: rà soát & kiểm tra yêu cầu đảm bảo đáp ứng nhu cầu khách hàng / người dùng.

³ **Documentation**: có thể viết tài liệu trên file văn bản (ví dụ Word, Excel), hoặc trên các phần mềm soạn tài liệu (ví dụ Confluence, Jira)

Lưu ý

Quy trình phát triển yêu cầu:

- có thể áp dụng cho bất cứ cấp độ yêu cầu nào, phổ biến nhất là cho yêu cầu người dùng và yêu cầu phần mềm.
- bắt đầu từ yêu cầu / ý tưởng khái quát, và bổ sung chi tiết dần dần cho tới khi thống nhất được phạm vi và mức độ chi tiết.
- có thể làm tuần tự lũy tiến tỉ mỉ dần, kết hợp qua lại các công đoạn. Ví dụ: trong workshop (một hoạt động thuộc công đoạn elicitation) có thể kết hợp phân tích biểu diễn bằng sơ đồ để xác minh cách hiểu đúng đắn. Sau đó cuối workshop viết lại vắn tắt một số yêu cầu chính để tổng hợp những gì thu hoạch được qua workshop.

Quản lý (Management) liên quan tới:

Thiết lập baseline, kiểm soát thay đổi (change control), và truy vết (traceability)

- Baseline trong dự án theo mô hình Waterfall có thể sử dụng version tài liệu.
- Baseline trong dự án theo mô hình Agile/Scrum thường không rõ ràng, mà căn cứ vào hoạt động backlog refinement trước khi làm sprint planning.

#2. Phân tích Stakeholder

Xác định và phân tích hành vi, mối quan tâm của stakeholder về yêu cầu nghiệp vụ và giải pháp.

Sử dụng stakeholder map (hoặc matrix) để quan sát, phân loại, và quản lý stakeholders trong suốt hành trình dự án, qua đó có những hành động phù hợp nhằm hướng đến khai thác & thống nhất yêu cầu. Tập trung vào các stakeholder có vai trò decision making.

Cách gây dựng độ tin cậy trong công việc nói chung, và với KH nói riêng

- Trust – Transparent Matrix: dựa vào sự minh bạch trong công việc, trong cách làm việc với KH
- Trust Equation: dựa vào sự hiểu biết / tính cam kết / sự thân thiện

FBA03 – Requirement Elicitation

#1. Các kỹ thuật khai thác yêu cầu

Technique	Description	Pros	Cons
Workshop	<ul style="list-style-type: none"> – Sử dụng workshop để khai thác và thống nhất requirements từ các stakeholders liên quan theo các chủ đề của dự án. – Stakeholder involved thường là những người quan trọng. 	<ul style="list-style-type: none"> – “Ba mặt một lời” để tránh việc phải luân chuyển thông tin qua nhiều bên, dễ gây thất thoát và sinh ra cách hiểu khác nhau – Hướng đến sự đồng thuận giữa các stakeholders – Phù hợp khi số lượng key stakeholders không quá lớn (vd <10 người) và họ sắp xếp được thời gian 	<ul style="list-style-type: none"> – Tốn kém: khi phải involve nhiều bên không thuộc cùng một vùng địa lý – Không phải lúc nào cũng làm workshop được nếu các bên không hợp tác. – Không hiểu hết nội dung workshop do BA chưa có kiến thức nền tảng về business domain hoặc chưa nắm được thuật ngữ chuyên ngành (glossary) => BA cần trang bị trước khi tham gia workshop.

Interview	<ul style="list-style-type: none"> – Thường dưới dạng 1-1 interview, giữa BA và 1-2 người đại diện cho một bên nào đó. – Có 2 kiểu planned (các câu hỏi gửi trước, có agenda) và unplanned (ngược lại) 	<ul style="list-style-type: none"> – Không tốn kém như workshop, và có thể làm nhiều vòng với các stakeholder. – BA có nhiều thời gian chuẩn bị câu hỏi hơn – Less informal hơn so với workshop => bớt căng thẳng hơn (nhất là đối với BA mới) 	<ul style="list-style-type: none"> – Thông tin có thể “mang tính chủ quan” của stakeholder đó => BA cần cross check khéo léo với stakeholder khác.
Document Analysis	<ul style="list-style-type: none"> – Phân tích tài liệu, hệ thống có sẵn (thường do KH hoặc đối tác cung cấp) 	<ul style="list-style-type: none"> – BA có nhiều thời gian đọc hiểu, nghiên cứu tài liệu. Hoặc được “test” qua hệ thống để hiểu sơ bộ những gì đang có. – BA có thể phối hợp với Team để cùng nghiên cứu 	<ul style="list-style-type: none"> – Nếu có vướng mắc, không được giải thích ngay => Có thể kết hợp với Interview để hiểu đầy đủ document / system mà KH cung cấp.
Brainstorming	<ul style="list-style-type: none"> – BA và team cùng brainstorming về một vấn đề nào đó của KH – 2 bên (bên đối tác / KH và team phát triển) cùng brainstorm về một vấn đề chung nào đó – Nên khoanh vùng 1-2 vấn đề nổi cộm để thảo luận 	<ul style="list-style-type: none"> – Giúp đưa ra nhiều ý tưởng giải pháp. – Dễ đạt được sự đồng thuận 	<ul style="list-style-type: none"> – Có thể bị lan man / mất thời gian => cần người điều tiết để đưa về chủ đề chính.
Observation	<ul style="list-style-type: none"> – Quan sát thực tế quy trình / dây chuyền vận hành của KH / Đối tác để nắm được context và thu thập insights – Tên gọi khác: Shadow – Có thể áp dụng vào job shadow 	<ul style="list-style-type: none"> – Hiểu được context và có thêm insights cho quá trình phân tích – Có thêm connection với business users / end users 	<ul style="list-style-type: none"> – Cần có sự phối hợp và được phép (authority) của KH / Đối tác khi vào khu vực sản xuất của họ. Nên không phải lúc nào cũng thực hiện được. – BA cần có kỹ năng quan sát tốt.
Survey	<ul style="list-style-type: none"> – Làm khảo sát để có thêm insights từ các stakeholders hoặc end-users – Số lượng câu hỏi không quá nhiều, vd <=10 câu. 	<ul style="list-style-type: none"> – Có được feedback khi không xác định được key stakeholders, hoặc số lượng stakeholders / end-users quá nhiều 	<ul style="list-style-type: none"> – BA cần có kiến thức nghiệp vụ đủ sâu và sử dụng ngôn ngữ nghiệp vụ để stakeholders / end-users có thể hiểu đúng câu hỏi

Various Diagrams	<ul style="list-style-type: none"> Sử dụng các loại diagrams để làm rõ yêu cầu Cần chọn các loại diagram đơn giản / dễ hiểu cho stakeholders 	<ul style="list-style-type: none"> Giúp tổng hợp thông tin, hoặc thể hiện được các logics trực quan hơn => làm rõ yêu cầu nhanh hơn 	<ul style="list-style-type: none"> BA cần có hiểu biết nhất định về các loại sơ đồ cơ bản, và biết cách sử dụng tools vẽ các sơ đồ đó.
Interface Analysis	<ul style="list-style-type: none"> Phân tích kết nối / tương tác giữa 2 hệ thống 	<ul style="list-style-type: none"> Giúp minh bạch thông tin trao đổi giữa 2 hệ thống Biết được các chiều của thông, vd từ hệ thống A => B, hay A <= B. 	<ul style="list-style-type: none"> BA cần có kiến thức nhất định về giao thức trao đổi các hệ thống, vd API Và phương thức trao đổi (vd POST, GET), phân biệt được API và Webhook
Prototype / Wireframe	<ul style="list-style-type: none"> Sử dụng wireframe / prototype để làm rõ yêu cầu về cách sắp xếp thông tin trên các màn hình và UX cho người dùng Phân tích UX đóng vai trò quan trọng 	<ul style="list-style-type: none"> Tăng trực quan hóa về UX => giúp làm rõ yêu cầu nhanh và chính xác hơn 	<ul style="list-style-type: none"> BA cần có hiểu biết nhất định về information architecture, interaction design, và kỹ năng sử dụng tools làm wireframe / prototype.

#2. Tìm hiểu scope và quản lý scope yêu cầu

Requirements scope là phạm vi yêu cầu (chức năng và phi chức năng) những gì đội dự án cần phải làm để đảm bảo các chức năng cần thiết cho người dùng và cần bàn giao trong khuôn khổ hợp đồng giữa bên khách hàng (công ty A) với đội phát triển (công ty B). Và BA là người chịu trách nhiệm cho requirements scope đó.

Chú thích:

- Cách viết khác “product scope”, “feature scope”, “business analysis scope”
- Không phải Scope of Work⁴ của Project Manager.

Nếu scope không được xác định và thống nhất rõ ràng, có thể dẫn đến:

- Xây dựng Timeline (kế hoạch dự án) bị thiếu hụt
- Xây dựng giải pháp kiến trúc / thiết kế kỹ thuật bị thiếu tính năng, thiếu khả năng cần đáp ứng
- Thiếu nguồn lực kỹ thuật cần thiết (do nguồn lực developers dựa vào kiến trúc tổng thể). Ví dụ scope không nói rõ cần phát triển cả chức năng trên desktop web và mobile app => dẫn đến việc chuẩn bị nguồn lực có thể bị thiếu người có skills làm mobile app.
- Xác định Cost (chi phí nguồn lực) không đầy đủ.
- Khi KH bổ sung yêu cầu hoặc nhiều khi chỉ là đưa thêm một ý nào đó vào yêu cầu hiện tại, BA không kiểm soát được logics, không phân biệt được yêu cầu đang “bị” mở rộng (ảnh hưởng đến scope) và tự ý bàn giao cho đội phát triển, trong khi không kịp điều chỉnh timeline & cost kịp thời => Gây ra hiện tượng **scope creep** => Dự án phải làm việc over time, công ty tốn kém thêm chi phí mà không được bù đắp.

BA cần sử dụng các phương pháp sau để làm rõ và xác minh được scope với công ty khách hàng / đối tác:

- Work Breakdown Structure** (dạng sơ đồ hoặc mô tả): mô tả các người dùng và chức năng in-scope. Nhấn mạnh những thứ không mô tả là out-of-scope.

⁴ **Scope of Work:** (hoặc Work Order) là một dạng [hợp đồng dự án](#) giữa hai bên.

- **Context diagram** (tên gọi khác “Scope diagram”): bằng một sơ đồ thể hiện được “ranh giới” giữa hệ thống phải làm (in-scope) với các human users (in-scope) và các hệ thống bên ngoài (out-of-scope). Có thể mở rộng context diagram bằng xây dựng **relationship map** để nhìn đầy đủ toàn bộ các hệ thống cũng như ranh giới giữa các hệ thống đó.
- **Use Case diagram**: tương tự mục đích của context diagram, nhưng chi tiết hơn với việc thể hiện các use case / chức năng bên trong. Lưu ý: nên thể hiện use cases ở mức user requirements, không nên chia use cases quá chi tiết.
- **Feature List**: tương tự use case diagram, nhưng thể hiện dạng bảng / danh sách các chức năng. Ưu tiên các chức năng quan trọng lên trên. Cũng nên tránh rơi vào bẫy “feature factory”⁵, nhất là đối với việc phát triển sản phẩm (trong các công ty làm sản phẩm).

Chú thích: Relationship Map được sử dụng nhiều hơn trong việc mô hình hóa các bài toán về xây dựng năng lực hệ thống platform / core systems theo platform business model⁶. Ví dụ: một sản phẩm unsecured lending⁷ cho khách hàng cá nhân trong fintech có thể có các core systems sau:

- Digital eKYC Solution: cung cấp các dịch vụ eKYC từ cơ bản đến nâng cao.
- Loan Origination System: hệ thống tập trung vào các dịch vụ thu thập hồ sơ gốc phục vụ nghiệp vụ cho vay. Có thể bao gồm các dịch vụ: pre-qualification (hoặc pre-approval), loan application, underwriting process, loan decision, electronic signing.
- Risk Decisioning Engine: hỗ trợ ra quyết định dựa trên khẩu vị rủi ro (credit risk), bao gồm: risk assessment dựa trên các bộ rules, risk grading, và risk-based pricing.
- Loan Management System: quản lý khoản vay từ lúc hoàn thành origination đến khi write-off, có thể bao gồm các cấu phần: product settings, limit management, loan accounts, interest rate, fee & penalty, disbursement process, balancing & overdue, account settlement, repayment schedule, và closure process.
- Collection Management: quản lý thu hồi nợ, có thể bao gồm: collection workflow, EMI collections, fees collections.

Quản lý scope tốt giống như việc thể hiện ngắn gọn requirements theo một (hoặc hai) trong bốn dạng ở trên, và người BA có thể nói được, trình bày được cụ thể hơn các requirements bên trong.

FBA04 – Requirement Analysis

Sử dụng các sơ đồ để phân tích chi tiết yêu cầu. Các BA cần tự luyện tập cách vẽ và trải nghiệm với các tools vẽ khác nhau để tìm ra phương pháp vẽ và tools phù hợp với bản thân.

#1. Actor Table

Tên gọi khác: Actor Catalog, Actor Description, User Role Model

Giúp phát hiện đối tượng người dùng bị thiếu trong phân tích, cũng như giúp các stakeholders làm rõ vai trò / trách nhiệm của họ trong hệ thống.

- Ưu điểm: xác định WHO – người dùng cho tính năng nào ngay từ đầu giúp khoanh vùng phạm vi nhanh hơn. Phân biệt human actors với system actors bằng cách sử dụng các bảng khác nhau.
- Nhược điểm: các BA mới dễ bị nhầm lẫn giữa title và role của người dùng <= cần xem xét kỹ tên gọi và chức năng của từng vai trò một và kết hợp với **Context Diagram** đảm bảo tính nhất quán.

Lưu ý: chỉ xây dựng actor table cho các đối tượng người dùng hoặc hệ thống trong phạm vi bài toán.

⁵ **Feature Factory**: <https://hackernoon.com/10-ways-how-to-not-make-your-product-a-feature-factory-cefe7a7759c6>

⁶ **Platform business model**: <https://innovationtactics.com/platform-business-model-complete-guide/>

⁷ **Unsecured Loan**: <https://www.investopedia.com/terms/u/unsecuredloan.asp>

#2. Activity Diagram

Sử dụng các ký hiệu đơn giản để thiết kế luồng hoạt động. Tham khảo [cách vẽ của LucidChart](#)

- Ưu điểm: dễ dùng, rất nhiều tools vẽ hỗ trợ sơ đồ này, và thể hiện được các logics bên trong hệ thống. Sử dụng activity diagram khi không quá quan tâm đến vai trò nào thực hiện chức năng đó.
- Nhược điểm: không thể hiện rõ được một số tình huống phức tạp, ví dụ: khi có nhiều đường thông tin đi vào không biết cái nào trước / sau hay đồng thời

#3. Cross-functional Diagram / Swimlane

Tên gọi khác: swimlane, flowchart, process map, hoặc đôi khi dùng chung với activity diagram. Cross-functional diagram là một trong những sơ đồ được BA sử dụng nhiều nhất.

Mở rộng của Activity Diagram bằng cách thêm:

- (Đối với user requirements) Các cột đại diện cho các end-users tương tác lẫn nhau
- (Đối với software requirements) Các cột đại diện cho end-users và system(s) thể hiện sự tương tác giữa người dùng và chức năng của hệ thống. Có thể sử dụng mỗi sơ đồ cho một luồng chức năng.
- Cần chỉ rõ input & output của một sơ đồ.

Ưu điểm:

- Nhìn rõ vai trò và trách nhiệm của từng bên tham gia
- Sử dụng các ký hiệu đơn giản (tương tự activity diagram)

Nhược điểm:

- Không thể hiện được các logics phức tạp (tương tự activity diagram) <= Có thể kết hợp sử dụng thêm chú thích / comments bên cạnh. Hoặc minh họa thêm logics bằng việc sử dụng ví dụ để làm rõ nghĩa hơn.

#4. Business Policies

Chính sách nghiệp vụ (hay chính sách sản phẩm) là các hướng dẫn (*guidelines*), tiêu chuẩn của ngành (*industry standards*) hoặc các quy định (*regulations*) để ràng buộc việc tuân thủ nghiệp vụ. Các chính sách này là cơ sở để ra quyết định và được xây dựng trong các hệ thống phần mềm cũng như các quy trình vận hành.

Ví dụ: đối với các khách hàng trong một hệ thống online shopping, có thể có một số chính sách sau (tùy theo tình hình kinh doanh):

- KH giao dịch nhiều sẽ được giảm giá
- KH giao dịch nhiều trong ngày sẽ được thăng hạng.
- Các đơn hàng hủy trước khi giao hàng sẽ được hoàn tiền.
- Các khách hàng không thỏa mãn với tình trạng hàng được giao sẽ được hoàn tiền hoặc miễn phí cho đơn hàng giá trị tương đương tiếp theo.

#5. Business Rules

Là mệnh đề mô tả quy tắc nghiệp vụ cụ thể hóa từ một Business Policy. Business Rule cần cụ thể hóa / lượng hóa đến mức “ $1 + 1 = 2$ ”, chứ không thể chung chung như Policy.

Ví dụ, đối với business policy “KH giao dịch nhiều sẽ được giảm giá”, thì business rules phải chỉ ra được:

- Loại khách hàng nào? Ví dụ: KH hạng “đồng chí” (giả sử có 3 levels của một KH: nhôm nhựa, đồng chí, vàng bạc)
- Giao dịch nhiều là bao nhiêu? Ví dụ đối với KH hạng “đồng chí”, giao dịch ≥ 50 tr/ngày được coi là “nhiều”
- Giảm giá thế nào? Ví dụ phải cụ thể giảm 10% tổng giá trị đơn hàng
- Khi nào áp dụng giảm giá? Ví dụ cho đơn hàng tiếp theo.

- Việc giảm giá có thời hạn không? Ví dụ trong vòng 90 ngày kể từ ngày thỏa mãn yêu cầu.

Vậy viết lại business rules như sau:

Rule 1: Nếu một KH hạng “đồng chí” có tổng giá trị mua hàng $\geq 50\text{tr/ngày}$ trong ngày T (tính cho các đơn hàng confirmed), thì khách hàng sẽ được chiết khấu 10% vào một đơn hàng tiếp theo. Chính sách này áp dụng cho một đơn hàng bất kỳ từ ngày T+1 đến T+90.

#6. Status Transition Diagram

Tên gọi khác: State Diagram, State Transition Diagram, State Machine.

Là sơ đồ biểu diễn vòng đời của một đối tượng (business object). Status diagram giúp thể hiện được các logics bên trong hệ thống.

Một số lưu ý:

- Áp dụng các status cho một đối tượng. Ví dụ đối tượng Order trong ví dụ trên, có thể có một loạt status khác nhau, bao gồm:
 - Open: hàng đã xếp vào giỏ, đơn hàng chưa thanh toán
 - Confirmed: đơn hàng đã thanh toán thành công
 - Shipped: đơn hàng đã rời kho (là kho đầu tiên nếu có nhiều kho trung chuyển) và đang trên đường tới địa chỉ nhận
 - Delivered: KH đã nhận hàng thành công
 - Cancelled: đơn hàng đã hủy
- Số lượng các status càng ít mà vẫn thể hiện được đầy đủ nghiệp vụ yêu cầu càng tốt
- Đặt tên các status cần đồng nhất về cách thức và ngắn gọn & xúc tích càng tốt.

#7. Use Case Diagram

Là sơ đồ mô tả cách thức các human actors sử dụng hệ thống để hoàn thành mục tiêu của người dùng. Mỗi use case đại diện cho một chức năng và mục đích của người dùng.

Một số lưu ý:

- Mỗi use case là một luồng “end-to-end” của người dùng cho đến khi đạt được mục đích. Trong hệ thống có thể phải kết nối sang các hệ thống khác.
- Phân biệt use case với các bước (step) trong một use case. Tránh việc hiểu mỗi step là một use case.
- Đặt tên use case dùng ngôn ngữ của business. Ví dụ: không dùng “Create Customer” nếu người dùng là Khách hàng thực hiện đăng ký mở tài khoản trong hệ thống online shopping. Nên là “Register Customer Account”. Hoặc, đối với khách hàng, dùng “Place Order” thay vì “Create Order”.
- Kết hợp use case diagram với phân tích / mô tả theo hướng use case. Sử dụng cấu trúc của use case document. Tham khảo [Bridging the gap](#).
- Nên có mô tả ngắn cho từng use case trên diagram, thay vì chỉ vẽ mỗi diagram.

#8. Decision Table

Một cách biểu diễn có cấu trúc dạng bảng cho một tập business rules bao gồm 2 nhóm chính: tập các điều kiện đầu vào (condition) và tập các kết quả đầu ra (outcome / action / decision). Tham khảo [Visual-diagram](#).

Một số lưu ý:

- Condition có các giá trị hữu hạn, càng đơn giản càng dễ hiểu mà vẫn đủ minh họa các business rules là được, ví dụ mô tả condition đưa về dạng Yes / No
- Cho từng rule, outcome là kết quả của việc thỏa mãn đồng thời các điều kiện tham gia.
- Các rule có tính loại trừ cao được sắp xếp trước.
- Sử dụng ngôn ngữ nghiệp vụ diễn đạt trên decision table.

#9. Concept Mapping Diagram

Sơ đồ biểu diễn mối quan hệ giữa các business concept được sắp xếp cấu trúc phân cấp và kết nối với nhau theo các đường kẻ hoặc mũi tên. Các đường kết nối này được gắn nhãn thể hiện mối quan hệ giữa các concept. Tên gọi khác là conceptual diagram⁸.

Một số lưu ý:

- BA cần tự trang bị danh sách thuật ngữ nghiệp vụ (business concepts). Có thể áp dụng kỹ thuật Glossary.
- Sử dụng ngôn ngữ tự nhiên, ngắn gọn & xúc tích
- Biểu diễn sơ đồ theo luồng từ trên xuống dưới.

FBA05 – Documentation

Cần phân biệt được các loại tài liệu yêu cầu theo từng mô hình dự án Waterfall hoặc Agile.

#1. Business Requirements Document

Business Requirements mô tả nhu cầu chung của sản phẩm làm thoả mãn sứ mệnh như tăng doanh số, giảm chi phí vận hành, tăng chất lượng dịch vụ chăm sóc khách hàng, hoặc đáp ứng điều lệ, quy định mới trong ngành. Tầm nhìn sản phẩm chỉ ra cách nhìn toàn diện và dài hạn mà sản phẩm cuối cùng cần đạt được, có thể bao gồm phạm vi sản phẩm để làm sáng tỏ dần những khả năng mà sản phẩm cung cấp.

ParcelKiosk (PK, www.parcelkiosk.com) đang muốn phát triển một hệ thống web để trang bị tốt hơn cho dịch vụ giao hàng của họ (parcel delivery services) đến tay người mua hàng. PK tiếp cận một trong những đối tác của họ là công ty phần mềm. Nhiệm vụ đầu tiên của bên phần mềm phải làm là phân tích nhu cầu nghiệp vụ. Vậy theo bạn, yêu cầu nghiệp vụ bao gồm những gì?

Có người nói yêu cầu nghiệp vụ nên là vấn đề “*bảo mật*” (security).

Đúng là bảo mật là nhân tố rất quan trọng, nhưng nó không phải yêu cầu nghiệp vụ. Bạn không thể xây dựng một ứng dụng mà không có tính bảo mật, nhưng cung cấp dịch vụ chỉ để giải quyết vấn đề bảo mật KHÔNG phải mục tiêu kinh doanh của PK.

Cũng có người cho rằng yêu cầu nghiệp vụ nên có phần “*Kết nối các dịch vụ vận chuyển với khách hàng*”.

So với yêu cầu bảo mật, tính năng này đúng là yêu cầu nghiệp vụ. Nhưng liệu đây có phải lí do để xây dựng hệ thống web, liệu nó thực sự cần thiết cho dịch vụ giao nhận của doanh nghiệp?

Một số ví dụ khác dưới đây “gần” với thực tế yêu cầu nghiệp vụ hơn:

- Cung cấp giải pháp thông minh hơn để thu thập, đo lường, và vận chuyển hàng
- Quy trình nghiệp vụ giao hàng
- Trang bị khả năng theo dõi vị trí hàng vận chuyển, và quản lý dịch vụ.
- Đảm bảo giao hàng kịp thời và ghi nhận phản hồi khách hàng.

Nếu cần một tài liệu BRD riêng rẽ để mô tả yêu cầu nghiệp vụ thì thường có các phần sau:

- Tầm nhìn và mục tiêu của dự án
- Ngữ cảnh ra đời dự án
- Phạm vi chức năng, phạm vi dự án
- Xác định các bên liên quan: để biết bên nào sẽ chịu ảnh hưởng từ giải pháp phát triển hoặc có tác động tới việc phát triển giải pháp. Từ đó chuẩn bị cho bước tiếp theo - xây dựng mối quan hệ với các bên liên quan trong quá trình phân tích User Requirements
- Yêu cầu Nghiệp vụ

⁸ **Concept Mapping / Conceptual Diagram:** <https://www.lucidchart.com/pages/concept-map>

- Các yếu tố ràng buộc (về thời gian, chi phí, và tài nguyên)

#2. User Requirements Document

Một trong những thách thức khi xây dựng yêu cầu người dùng là phải xác định được người dùng cuối thực sự cần hệ thống làm gì để phục vụ việc cho họ. Nguyên nhân là do người dùng (hoặc đại diện người dùng) rất khó diễn đạt hết được nhu cầu và mong muốn của họ. Chưa kể thông tin cung cấp có thể không đầy đủ, không chính xác, thậm chí thông tin mâu thuẫn lẫn nhau giữa các bên liên quan, và không đạt được sự đồng thuận chung.

Yêu cầu người dùng thường được mô tả trong các tài liệu:

- Yêu cầu người dùng (*URD - user requirements document*): gồm các yêu cầu chức năng và phi chức năng. Một số hệ thống lớn có thể tách mỗi loại yêu cầu này thành các tài liệu riêng rẽ.
- Cho dự án Agile thì có *feature list*, Yêu cầu Sản phẩm (*PRD - product requirements document*), Danh sách Yêu cầu & Việc cần làm (*product backlog*)

Ví dụ: cùng là chức năng check in chuyến bay của hành khách trước giờ khởi hành, trên web hoặc tại quầy.

- Nếu viết dưới dạng yêu cầu chức năng: “Check in chuyến bay” (“Check in for a flight”)
- Nếu viết dưới dạng user story: “Là một hành khách, tôi muốn check-in chuyến bay dựa vào thông tin trên vé máy bay của tôi, để tôi có thể lên máy bay.”

#3. Software Requirements Document

Yêu cầu phần mềm thường được mô tả trong các tài liệu sau:

- Đặc tả phần mềm (*SRS - software requirements specifications*): gồm yêu cầu chức năng (thường viết dưới dạng use case), yêu cầu phi chức năng, các ràng buộc kỹ thuật (*design & implementation constraints*⁹), yêu cầu cần tích hợp (*external interfaces*¹⁰), và các loại yêu cầu khác (ví dụ: data migration, transition requirements)
- Yêu cầu chức năng (*functional requirements*). Cũng tương tự tài liệu yêu cầu người dùng, cho các hệ thống lớn, có thể tách chức năng và phi chức năng thành các tài liệu riêng rẽ.
- Yêu cầu phi chức năng (*non-functional requirements*), hoặc yêu cầu hệ thống (*system requirements*)
- Tài liệu use case: tài liệu phân tích use case chi tiết.

Có nhiều loại yêu cầu phi chức năng, điển hình là security (bảo mật), performance (hiệu suất), usability (tính dễ dùng), reliability (độ tin cậy), scalability (quy mô), maintainability (khả năng duy trì). Yêu cầu phi chức năng khi mô tả cần được lượng hoá cụ thể để có thể test được.

Một số Ví dụ

- [*Response time for loading*] Quá trình lấy thông tin khách hàng từ hệ thống back-end và hiển thị lên màn hình của người dùng không được quá 6 giây tính từ lúc người dùng gửi yêu cầu.
- [*User concurrency*] Trong mùa nghỉ lễ cao điểm, từ 1/11 đến 5/1, tính năng tìm kiếm trong kho cần cho phép lên đến 500 người dùng đồng thời thực hiện tìm kiếm.
- [*System availability*] Hệ thống cần đảm bảo chức năng lập lịch luôn sẵn sàng vào các ngày trong tuần, từ Thứ 2 đến Thứ 6, từ 7a.m (giờ PST) đến 7p.m (giờ PST).

⁹ **Design & Implementation Constraints** là các ràng buộc kỹ thuật (thường do khách hàng yêu cầu) mà giải pháp phần mềm cần tuân thủ. Một số ví dụ:

- Hỗ trợ đa ngôn ngữ: giải pháp cần đáp ứng hiển thị cả tiếng Anh và tiếng Việt. Ưu tiên tiếng Việt trước tại thời điểm go-live, nhưng thiết kế giải pháp cần đảm bảo cả tiếng Anh để đưa vào sử dụng sau này.
- Microsoft Power BI (<https://powerbi.microsoft.com/en-us/>): sử dụng Power BI cho các báo cáo tăng tính trực quan.

¹⁰ **External Interfaces:** là yêu cầu về giao diện tích hợp với các hệ thống bên ngoài tương tác với hệ thống đang phân tích.

FBA06 – UX Part 1: UX Analysis Overview

#1. Nhiệm vụ của UX

Tập trung vào việc am hiểu người dùng, biết được họ cần gì, giá trị thế nào, khả năng cho phép người dùng được làm, cũng như các hạn chế nếu có.

UX bao gồm 4 cấp độ:

- **Functionality:** Tính năng người dùng
- **Usability:** Tính dễ dùng của người dùng khi sử dụng các tính năng đó
- **Desirability:** Sự hứng thú khi dùng tính năng
- **Brand Experience:** Tính “gây nghiện” => thích một sản phẩm kéo theo thích luôn thương hiệu công ty sở hữu sản phẩm đó.

Ngoài việc xây dựng functionality, BA có thể phân tích và giúp cải thiện usability của sản phẩm.

#2. Thế nào là “dễ dùng” nói chung?

10 nguyên tắc (theo [Jakob Nielsen](#)¹¹)

#	Usability Principle	Descriptions
1	Visibility of system status	Hệ thống cần thông báo cho người dùng biết về những gì đang diễn ra một cách thích hợp và đúng lúc, trước khi họ phải đưa ra phán đoán.
2	Match between system and the real world	Người dùng càng ít phải phán đoán càng tốt. Hệ thống nên hiển thị ngôn ngữ của người dùng (cụm từ, concept quen thuộc với người dùng) để giao tiếp với họ hơn là sử dụng những thuật ngữ đặc biệt của hệ thống.
3	User control and freedom	Giúp người dùng xóa bỏ lo ngại khi họ biết rằng những lỗi đó có thể Undo ngay lập tức, nhờ vậy khuyến khích họ khám phá những tính năng khác mà họ chưa từng sử dụng.
4	Consistency and standards	Nên sử dụng thống nhất tất cả các yếu tố / ký hiệu hiển thị để tránh gây bối rối cho người dùng.
5	Error prevention	Không một ai thích gặp phải lỗi trong quá trình sử dụng, tệ hơn nữa là khi chúng ta có cảm giác rằng mình đã làm sai điều gì đó. Do vậy, hệ thống cần ngăn ngừa những lỗi dễ xảy ra cho người dùng, hoặc kiểm tra và thông báo giúp họ trước khi họ xác nhận hành động tiếp theo.
6	Recognition rather than recall	Nhận biết một thứ gì đó sẽ dễ dàng hơn việc phải ghi nhớ nó. Hạn chế việc ghi nhớ của người dùng bằng cách thiết kế vật thể, hành động, các sự lựa chọn có tính nhận biết cao. Kết hợp sử dụng các chỉ dẫn.
7	Flexibility and efficiency of use	Cho phép người dùng chi phối và cá nhân hóa những thao tác thường xuyên trên ứng dụng/sản phẩm.
8	Aesthetic and minimalist design	Thiết kế tối giản => Tối giản hóa giao diện bằng cách loại bỏ những yếu tố/nội dung không cần thiết hoặc không có vai trò giúp đỡ người dùng.

¹¹ **Jakob Nielsen**, là chuyên gia web usability, nhà nghiên cứu human-computer interaction, đồng sáng lập của Tập đoàn Nielsen Norman.

9	Help users recognise, diagnose, and recover from errors	Giúp người dùng nhận biết, chẩn đoán và khôi phục các lỗi Những thông báo lỗi nên được thể hiện bằng ngôn ngữ đơn giản, dễ hiểu (không nên sử dụng các thuật ngữ của hệ thống để thông báo cho người dùng), biểu thị chính xác lỗi mà người dùng gặp phải và đưa ra gợi ý từng bước một để người dùng xử lý vấn đề.
10	Help and documentation	Mặc dù sẽ tốt hơn nếu giao diện được người dùng sử dụng một cách dễ dàng mà không cần đến tài liệu hướng dẫn, đây vẫn là một điều cần thiết để cung cấp cho người dùng sự giúp đỡ bất cứ khi nào họ cảm thấy bối rối. Các tài liệu hướng dẫn nên được thiết kế tối ưu cho việc tiếp cận, tra cứu, tập trung vào những thao tác của người dùng với danh sách các bước cụ thể và đương nhiên, chúng không nên quá dài.

#3. Thế nào là “dễ dùng” trên mobile application?

Ngoài các nguyên tắc trên, bổ sung các nguyên tắc riêng cho mobile app/web development.

#	Usability Principle	Descriptions
1	Know users	Cần am hiểu đối tượng người dùng. Tránh việc cho rằng mọi người sử dụng điện thoại theo cách thức giống nhau. Kết hợp dùng analytics hoặc survey để có thêm insights về hành vi người dùng.
2	Understand context of use	Nắm được bối cảnh / hoàn cảnh, không gian, thời điểm người dùng sử dụng mobile app trên thiết bị của họ.
3	Hand position controls	Biết được các khu vực trên màn hình điện thoại mà ngón tay cái thường xuyên hoạt động (dễ dàng hay phải dướn), cũng như việc dùng điện thoại trên một tay hoặc hai tay. Để từ đó đặt các nút bấm ở vị trí thuận tiện nhất cho người dùng.
4	Put content first	Thiết kế content trước, navigation sau. (Ngược với thiết kế desktop web)
5	Touchscreen target size	Do sử dụng ngón tay để thao tác trên màn hình, kích thước các nút bấm / icon mà người dùng có thể bấm được tối thiểu 44x44 pixels.
6	Minimize data input	Hạn chế tối đa việc người dùng phải nhập dữ liệu mà vẫn đảm bảo thực hiện được chức năng cần thiết.

FBA07 – UX Part 2: Wireframing

#1. Các phương pháp làm Information Architecture

Liên quan đến sắp xếp thông tin theo hành trình người dùng, tương tự sơ đồ tìm đường, Information Architecture giúp cấu trúc hóa thông tin theo cách người dùng có thể hiểu được, dựa vào:

- Vị trí hiện tại của họ ◀ bối cảnh (không gian, địa điểm, trạng thái) tại vị trí đó
- Những thông tin họ thấy được từ vị trí đó
- Các thông tin xung quanh
- Vị trí họ mong muốn tiến tới

Các kỹ thuật / phương pháp cấu trúc thông tin:

- Card sorting: liệt kê => phân loại theo nhóm => sắp xếp theo trình tự logic
- Sitemap: xây dựng cấu trúc trang hình cây

- Labeling: đặt tên, gán nhãn thông tin một cách ngắn gọn & xúc tích nhưng vẫn đảm bảo ý nghĩa đầy đủ đằng sau mỗi label đó.
- Taxonomy: phân loại thông tin theo mục đích sử dụng của người dùng (thay vì theo tính chất vật lý / kỹ thuật của thông tin)
- Data Model: sử dụng sơ đồ khái niệm (conceptual data model) để biểu diễn sơ đồ đường đi của thông tin. Tham khảo #9. Concept Mapping Diagram
- Navigation: xây dựng cách thức để người dùng duyệt tới & lui thông tin trên một hệ thống.
- Searching: cho phép người dùng tìm kiếm thông tin (nhất là đối với thông tin nằm rất sâu trong sơ đồ Card Sorting).

#2. Vai trò của Wireframe

Một số giá trị của wireframe / prototype:

- Giúp người dùng hoặc đại diện người dùng xác định được nhu cầu thực sự của họ, mà đội sản phẩm hoặc BA không mất quá nhiều thời gian thiết kế (so với xây dựng phần mềm)
- Giúp xác minh cách hiểu đúng đắn về UX khi trao đổi với các stakeholders / Khách hàng
- Giúp người dùng trực quan hóa thông tin và ra quyết định nhanh hơn

Để làm wireframe / prototype, BA cần có hiểu biết nhất định về và theo trình tự sau:

- Conceptual data model: đường đi của thông tin
- Information architecture: cấu trúc thông tin
- Interaction design: cách tương tác người dùng với hệ thống
- Visual design: một chút thôi

FBA08 – Requirement Validation & Management

#1. Phân biệt Validate và Verify Requirements

Validate Requirements (“build the right things”): xác minh yêu cầu với các stakeholders / khách hàng => đảm bảo BA hiểu đúng nhu cầu khách hàng, tránh “tam sao thất bản”.

Verify requirements (“build the things right”): xác nhận yêu cầu với các đầu mối trong đội phát triển => đảm bảo yêu cầu được phát triển đáp ứng các chuẩn mực của quy trình phần mềm, cũng như đảm bảo đội phát triển hiểu đúng yêu cầu do BA xây dựng.

Hai kỹ thuật này cần BA áp dụng song hành cùng nhau, nghĩa là:

- Khi khai thác một yêu cầu của KH:
 - o cần verify user requirements đó với đại diện đội phát triển để đảm bảo tính khả thi về giải pháp,
 - o sau đó phát triển sâu hơn và làm rõ yêu cầu chi tiết, và
 - o cuối cùng validate detailed requirements với KH để đảm bảo hiểu đúng yêu cầu ban đầu của họ.
- Khi phát hiện / nghi ngờ một logic nào đó không có trong yêu cầu ban đầu của KH:
 - o cần verify với đội phát triển đảm bảo BA hiểu đúng
 - o sau đó phân tích các yêu cầu liên quan đến logic đó (impact analysis)
 - o và cuối cùng, validate với yêu cầu KH ban đầu để giúp KH bổ sung logic đó vào yêu cầu của họ

Để validate requirements với stakeholders có thể sử dụng kết hợp 3 phương pháp: walk-through, prototyping, và user scenario (theo user journey).

#2. Thế nào là một bản requirements xịn?

Đảm bảo KHÔNG có các vấn đề sau:

Issue	Explanation
Ambiguous (mơ hồ)	<p>Yêu cầu viết ra khiến cho người đọc hiểu theo nhiều cách khác nhau. Sử dụng từ ngữ mơ hồ, cảm tính, không có tính định lượng. Ví dụ: dùng những từ sau trong tài liệu requirements:</p> <ul style="list-style-type: none"> – “better”, “faster”, “ideally”, “normally”, “etc”, “so on” – “Charge numbers should be validated online against the master corporate charge number list, if possible” <p>Cách khắc phục</p> <ul style="list-style-type: none"> – Trong quá trình khai thác & phân tích yêu cầu, BA cần vận dụng tối đa kỹ năng nghe hiểu (active listening) để làm rõ / hỏi lại, và để khẳng định BA hiểu đúng yêu cầu KH. – Đọc đi đọc lại, kết hợp trình bày lại cho người khác, đảm bảo thống nhất cách hiểu chung. – Dùng thể chủ định (active) hơn là thể phủ định – Dùng các câu ngắn, mệnh đề ngắn & rõ ràng, nêu bật một ý cụ thể. Thay vì câu dài & nhiều ý. <p>Tham khảo thêm một số lời khuyên của Karl Wiegers¹² khi viết requirements để tránh các cách hiểu dễ gây nhầm lẫn cho người đọc.</p>
Incomplete (không đầy đủ)	<p>Yêu cầu (hoặc tài liệu yêu cầu) không đề cập đầy đủ các chức năng / logic / thông tin như đã thống nhất trước đó.</p> <p>Cách khắc phục</p> <ul style="list-style-type: none"> – Ghi chép đầy đủ nội dung các buổi họp / trao đổi với khách hàng và đối tác. Nhấn mạnh vào các nội dung quan trọng. – Kết hợp sử dụng tư duy logic, suy diễn / lập luận có cơ sở. Từ đó làm rõ & xác minh các lập luận đó với KH. Tránh việc ngẫm hiểu và tự ý quyết định, nhất là đối với các chức năng quan trọng. – Sử dụng các cách biểu diễn khác nhau (từ presenting – drawing – prototyping – documenting) để giảm thiểu việc thiếu sót requirement – Các cấp độ requirements khác nhau (từ business => user => software requirements) sẽ cần các bên liên quan khác nhau để review & đánh giá. Cần tham vấn đại diện các bên liên quan để biết được yêu cầu đã đầy đủ hay chưa. – Thường xuyên rà soát, kết hợp double check với các đầu mối.
Irrelevant (không liên quan)	<p>Yêu cầu (hoặc tài liệu yêu cầu) đề cập đến các chức năng / logic không có sự kết nối một cách logic đến các yêu cầu khái quát trước đó. Ví dụ: một yêu cầu viết ra trong tài liệu SRS không có sự liên hệ với yêu cầu tương ứng của user requirements. Nói cách khác, BA đang “bịa” ra yêu cầu đó!!</p> <p>Cách khắc phục</p> <ul style="list-style-type: none"> – Bổ sung các mục giúp tham chiếu thông tin, ví dụ “Referenced Documents / Sources” đến các tài liệu hoặc các nguồn thông tin chính thống do khách hàng / đối tác cung cấp. – Sử dụng Requirement ID¹³ để quản lý yêu cầu, giúp liên kết các cấp độ yêu cầu

¹² **Tips for Writing Unambiguous Requirements:** <https://medium.com/analysts-corner/six-tips-for-writing-unambiguous-requirements-70bad5422427>

¹³ **Requirement Attributes:** <https://www.businessanalystlearnings.com/blog/2017/2/26/10-key-attributes-for-describing-software-requirements>

Incorrect (sai)	<p>Yêu cầu (hoặc tài liệu yêu cầu) đề cập đến các chức năng / logic không thuộc scope đã thống nhất trước đó.</p> <p>Cách khắc phục</p> <ul style="list-style-type: none"> – Xây dựng và làm rõ scope sản phẩm ngay từ đầu – Sử dụng sơ đồ, mô tả rõ ràng để biểu diễn scope khái quát trong tài liệu yêu cầu – Kết hợp phân tích top-down (từ khái quát đến chi tiết) và bottom-up (từ chi tiết đến khái quát hóa)
Inconsistent (thiếu đồng nhất)	<p>Yêu cầu (hoặc tài liệu yêu cầu) gây mâu thuẫn lẫn nhau. Sự thiếu đồng nhất có thể xuất hiện trong:</p> <ul style="list-style-type: none"> – Thuật ngữ sử dụng – Trình tự thực hiện các tính năng – Logic tính toán – Thông tin cần xử lý <p>Cách khắc phục</p> <ul style="list-style-type: none"> – Thường xuyên rà soát các nội dung trong tài liệu – Vận dụng các loại sơ đồ tăng tính trực quan, giúp nhận biết rõ hơn & nhanh hơn – Minh họa bằng các ví dụ cho các logic phức tạp
Out-of-date (thiếu cập nhật)	<p>Nội dung tài liệu có các yêu cầu bị outdated, thiếu tính cập nhật.</p> <p>Cách khắc phục</p> <ul style="list-style-type: none"> – Tránh mô tả (và tránh đưa vào scope) các yêu cầu khi vẫn tiềm ẩn thay đổi thường xuyên, chưa đạt được sự đồng thuận từ các bên liên quan. – Bám sát, thúc đẩy nhằm đạt được phê duyệt cho các yêu cầu đó. Kết hợp đưa ra các lập luận phân tích và giải pháp giải quyết khúc mắc cho các bên. – Thường xuyên rà soát tài liệu.
Too technical / solution-bound (quá kỹ thuật)	<p>Sử dụng thuật ngữ “quá” kỹ thuật / công nghệ trong tài liệu yêu cầu (thay vì sử dụng ngôn ngữ nghiệp vụ của người dùng cuối).</p> <p>Cách khắc phục</p> <ul style="list-style-type: none"> – Xây dựng & trang bị kiến thức nghiệp vụ thông qua các business concepts theo domain¹⁴ tương ứng. – Tự đặt vào vị trí của end-user để hình dung về journey¹⁵ & chức năng cho người dùng cuối.
Infeasible / Non- viable (không khả thi)	<p>Tài liệu yêu cầu đưa ra các tính năng mà đội phát triển không thể làm được.</p> <p>Cách khắc phục</p> <ul style="list-style-type: none"> – Cần có tư vấn từ Technical Owner / Technical Architect của development team để biết năng lực kỹ thuật có làm được không. – Cũng như tư vấn từ Project Manager để biết có khả thi trong khuôn khổ timeline và budget cho phép không.

¹⁴ **Business Domain:** <https://www.bridging-the-gap.com/new-business-domain/>

¹⁵ **Learn a new domain:** [https://medium.com/analysts-corner/as-a-it-business-analyst-how-do-you-learn-a-new-domain-](https://medium.com/analysts-corner/as-a-it-business-analyst-how-do-you-learn-a-new-domain-78b7b1a37780)

Not prioritized (không sắp xếp ưu tiên)	<p>Một yêu cầu (đại diện bởi một requirement ID) mô tả chức năng / logic không rõ ràng về mức độ ưu tiên.</p> <p>Cách khắc phục</p> <ul style="list-style-type: none"> – Thống nhất một phương án đánh giá độ ưu tiên (vd MoSCoW, RICE) xuyên suốt toàn bộ tài liệu yêu cầu – Áp dụng cho từng yêu cầu cụ thể.
Untraceable	<p>Liên quan đến quản lý requirements, không “lần theo dấu vết” và không chỉ ra được một requirement (đại diện bởi requirement ID):</p> <ul style="list-style-type: none"> – Đang mapping với cấu phần nào trong thiết kế giao diện (wireframe/prototype), thiết kế kỹ thuật (technical design), thực thi (coding / implementation), kiểm thử (test case). – Nếu không chỉ ra được tên cấu phần, thì ít nhất cũng cần có trạng thái / tình trạng của cấu phần đó. <p>Cách khắc phục</p> <ul style="list-style-type: none"> – Sử dụng Requirement Traceability Matrix (RTM)¹⁶ biểu diễn dưới dạng spreadsheet hoặc các tools có khả năng quản lý yêu cầu (Jira, Trello) – Có thể kết hợp mapping theo các loại yêu cầu khác nhau (theo business – user – software requirements)
Unverifiable / Untestable (không test được)	<p>Yêu cầu không được mô tả một cách cụ thể đến mức có thể định lượng được. Nói cách khác, yêu cầu được gọi là testable¹⁷ nếu đưa vào phần mềm được.</p> <p>Ví dụ, một số yêu cầu sau không thể test được:</p> <ul style="list-style-type: none"> – “The system shall not fail during the first 5 years of normal operation” – “The product shall be easy to use” – “Verify that the system is stable when having a lot of users” – “Verify that color of button is good for users” – “Verify the login process is easy to follow” <p>Cách khắc phục</p> <ul style="list-style-type: none"> – Cần cụ thể hóa / lượng hóa toàn bộ logic, thông tin sử dụng, ví dụ xây dựng danh sách các giá trị cho trước để người dùng lựa chọn (drop-down values, alphanumeric input) – Cụ thể hóa quy tắc nghiệp vụ thành các công thức tính toán (đến mức “a + b = c”, xác định rõ các yếu tố a-b-c đó) – Không sử dụng các từ ngữ mơ hồ, trừu tượng, cảm tính trong tài liệu yêu cầu.
Gold plating (“mạ vàng”)	<p>Xuất hiện và (ngầm) đưa vào phát triển các chức năng / logic mà không có trong scope thống nhất ban đầu, không thuộc phạm vi khách hàng yêu cầu. Nói cách khác, làm nhiều hơn mức độ cần thiết nhưng theo hướng không mong đợi¹⁸.</p> <p>Cách khắc phục</p> <ul style="list-style-type: none"> – Kiểm soát scope (chức năng & phi chức năng) đã thống nhất với KH cũng như việc thực thi yêu cầu hoàn thiện scope. – Nếu phát hiện hiện tượng gold-plating trong phần mềm, cần phối hợp với Project Manager để giải quyết.

¹⁶ **Requirement Traceability:** https://en.wikipedia.org/wiki/Requirements_traceability

¹⁷ **Testable in Agile:** <https://www.pluralsight.com/guides/ensuring-requirements-are-testable-in-agile>

¹⁸ **Gold Plating** in PMI: <https://www.brainbok.com/guide/pmp/study-notes/gold-plating/>

<p>Scope creep (hiện tượng “nở” scope)</p>	<p>Xuất hiện và (ngầm) đưa vào phát triển các chức năng / logic (tăng scope!) mà không có sự điều chỉnh tương ứng về timeline, chi phí, chất lượng, khẩu vị rủi ro. Dẫn đến việc không hoàn thành theo timeline đã thống nhất, hoặc phát sinh chi phí không thể kiểm soát, hoặc nảy sinh hàng loạt rủi ro về chất lượng, về tính tuân thủ, về quy trình.</p> <p>Cách khắc phục</p> <ul style="list-style-type: none"> – Chủ động đánh giá các phát sinh và thống nhất với khách hàng và các bên liên quan – Điều chỉnh mức độ ưu tiên (nếu cần) – Đảm bảo tính minh bạch thông suốt đối với KH và đội phát triển – Phân tích sâu & đánh giá mức độ ảnh hưởng khi có phát sinh / thay đổi yêu cầu, hoặc nghi ngờ có thay đổi yêu cầu. – Thường xuyên bao quát các vấn đề, và sâu sát theo quy trình phần mềm để nắm được các thay đổi ngay từ khi mới bắt đầu.
--	---

#3. Các cách quản lý yêu cầu

Quản lý yêu cầu giúp:

- Duy trì tài liệu yêu cầu mới nhất, phản ánh rõ nét nhất hiện trạng thực tế. Nhưng vẫn đảm bảo các baseline cần thiết (versioning) cung cấp cho đội phát triển.
- Truy vết nguồn gốc (backward traceability) cũng như các tài liệu liên quan trong dự án (forward traceability) khi có vấn đề phát sinh
- Kiểm soát được thay đổi phát sinh (change control) trong quá trình phát triển dự án

FBA09 – Agile Perspective

#1. Ưu thế của Agile (so với Waterfall)

Một số ưu thế nổi trội:

- Nhìn được sản phẩm nhanh hơn, do rút ngắn các công đoạn theo hướng bàn giao càng sớm sản phẩm đến thay người dùng để “test & learn” càng tốt, rồi tiếp tục cải tiến & nâng cấp
- Giảm rủi ro và chi phí phát triển phần mềm, do thích ứng tốt hơn với các thay đổi liên tục của stakeholders.

#2. Cách xây dựng User Story

Một số **ưu điểm**:

- Phát triển requirements theo kiểu nhanh & ngắn gọn, rồi bổ sung dần dần, nhưng cũng cố gắng đủ ý nhất có thể (kết hợp sử dụng *user story splitting*). Thay vì phải hoàn thiện requirements đầy đủ và chuẩn chỉnh rồi mới đưa vào development.
- Cấu trúc gần với ngôn ngữ người dùng
- Phù hợp với các tiêu chí của dự án theo mô hình Agile/Scrum.
- Dễ thích ứng với cách tiếp cận user-centric / customer-centric.

Tham khảo [user stories và các phương pháp chia tách](#) trên Cộng đồng Business Analyst Việt Nam.

Nhược điểm:

- Khó xác định liệu có missing critical requirements hay không <= áp dụng #1. Tổng hợp Phân tích nghiệp vụ (Tips & Tricks)
- Khó quản lý requirements scope <= cần kết hợp với Product Backlog.
- Khó nhìn được ranh giới giữa các user stories <= cần rà soát các user stories trong backlog để tránh việc 2 user stories phát triển cùng một logics hoặc bị overlapped / mâu thuẫn lẫn nhau.

Cần áp dụng phương pháp INVEST vào phân tích user stories:

- I (independent): đảm bảo các user stories hoàn toàn tách bạch nhau. Không có 2 stories nào bị chồng chéo lên nhau.
- N (negotiable): các AC của user story dựa trên trao đổi & bàn bạc giữa PO/BA và các thành viên trong team
- V (value): mỗi user story đều nói rõ giá trị, thể hiện việc tại sao cần phải làm user story đó. Tránh việc đưa user story vào sprint mà không hiểu rõ mục đích / giá trị mang lại.
- E (estimable): các user stories đều được đội phát triển estimate (theo point, hour, hoặc day)
- S (small): các user stories và AC tương ứng đã được chia nhỏ tới mức làm trong một sprint là xong.
- T (testable): các user stories cần test được.

#3. Cách làm Feature Roadmap (dựa vào Prioritization trong Agile)

Đánh giá mức độ ưu tiên của một user story (hoặc cao hơn là feature, hơn nữa là epic), cần dựa vào đồng thời các yếu tố sau:

Business Priority	<ul style="list-style-type: none"> – Business Value: có giá trị nổi trội hơn không. Giá trị có thể về mặt kinh doanh, chi phí, về trải nghiệm khách hàng,... – Urgency: có cấp thiết không. Để giải quyết pain points của người dùng, hoặc cạnh tranh với đối thủ trên thị trường. – Opportunity: có cơ hội gì rõ ràng không. Để chiếm lĩnh thị trường, để sớm cải thiện mặt nào đó của sản phẩm / dịch vụ hiện tại.
Technical Feasibility / Development Size	Khả thi về mặt kỹ thuật và năng lực đội phát triển. Có thể estimate được thì càng tốt.

Một số phương pháp prioritization hay sử dụng:

- **RICE** (Reach – Impact – Confidence – Effort)¹⁹: khi đánh giá các tính năng đưa vào phát triển sản phẩm, giúp lượng hóa theo từng tiêu chí.
- **MoSCoW** (Must – Should – Could – Won't)²⁰: đánh giá theo nhận định của các key stakeholders.

Có thể gắn Product/Feature Roadmap với timeline dự kiến để có cách nhìn tường minh hơn về kế hoạch phát triển sản phẩm.

FBA10 – BA Career Development

#1. Tổng hợp Phân tích nghiệp vụ (Tips & Tricks)

Làm thế nào để đảm bảo không bỏ sót các yêu cầu quan trọng?

#	Practice	Descriptions
1	Phân tích và biểu diễn requirements theo các cách thức khác nhau	<p>Bao gồm:</p> <p>Presentation: trình bày requirements cho người khác, cho stakeholders</p> <ul style="list-style-type: none"> – Ưu điểm: nhanh, dễ thực hiện – Nhược điểm: “lời nói gió bay” ← giữ thói quen làm meeting notes trong hoặc sau buổi họp. <p>Diagrams: thể hiện requirements qua các sơ đồ / hình vẽ</p> <ul style="list-style-type: none"> – Ưu điểm: trực quan, phơi bày được các logics ẩn trong hệ thống. Có thể kết hợp presentation và diagrams giúp stakeholders hiểu logics / ý đồ phân tích nhanh hơn.

¹⁹ **RICE**: <https://roadmunk.com/guides/rice-score-prioritization-framework-product-management/>

²⁰ **MoSCoW**: <https://www.productplan.com/glossary/moscow-prioritization/>

		<ul style="list-style-type: none"> Nhược điểm: sử dụng không đúng ký hiệu trong các loại sơ đồ có thể gây khó hiểu ← cần luyện tập kỹ năng vẽ sơ đồ và sử dụng tools. <p>Prototype / Wireframe: thể hiện cách sắp xếp thông tin trên màn hình</p> <ul style="list-style-type: none"> Ưu điểm: trực quan, giúp xác minh UX nhanh. Có thể kết hợp presentation và prototype giúp stakeholders hiểu ý đồ thiết kế nhanh hơn. Nhược điểm: không tuân thủ các nguyên tắc về usability và information architecture có thể gây khó hiểu cho stakeholders ← cần luyện tập kỹ năng thiết kế wireframe / prototype và sử dụng tools. <p>Documentation: trình bày requirements bằng tài liệu, notes, artifacts.</p> <ul style="list-style-type: none"> Ưu điểm: “giấy trắng mực đen”, đảm bảo các thông tin được viết ra đầy đủ & rõ ràng. Thể hiện chi tiết yêu cầu. Nhược điểm: phải viết chi tiết, dẫn đến mất thời gian ← Là kỹ thuật sử dụng sau khi phân tích & áp dụng 3 cách trên. Tài liệu có đầy đủ diagrams và mockup / screenshots.
2	Sử dụng scenario / kịch bản	<p>Phương pháp:</p> <ul style="list-style-type: none"> Đặt vào vị trí end-users để phân tích và trình bày Đi theo từng luồng chức năng trên user journey (hành trình người dùng, ví dụ CJM – customer journey mapping) Kết hợp sử dụng 5Why để đảm bảo các bước trên hành trình có ý nghĩa và cần thiết.
3	CRUD (Create – Read – Update – Delete)	<p>Áp dụng tính toàn vẹn cho:</p> <ul style="list-style-type: none"> Từng trường dữ liệu trên một màn hình: Dữ liệu ở đâu mà ra, từ hệ thống nào cung cấp hay do người dùng nhập vào. Dữ liệu thay đổi thế nào, và có cơ chế xóa / dọn dẹp dữ liệu không. Từng tính năng của người dùng.

#2. Các hướng tiếp cận làm Requirements

Nổi bật 2 hướng tiếp cận chính:

Predictive (mang tính xác định trước)

- Ưu điểm: cấu trúc tài liệu rõ ràng (thường 2 bên thống nhất templates từ đầu dự án), quy trình đầy đủ, xác định trước nhiều thông tin / cấu phần trong dự án.
- Nhược điểm: khó thay đổi hoặc nếu phải thay đổi thì phải rework rất nhiều.

Ví dụ điển hình là tiếp cận phát triển yêu cầu bằng phương pháp Use Case ← tài liệu rất nặng nề!

Adaptive (mang tính thích nghi theo tình huống)

- Ưu điểm: thay đổi linh hoạt, cần tới đâu phát triển requirements tới đó, có thể theo template hoặc không. Kết hợp sử dụng các tools tăng tính tương tác & phối hợp lẫn nhau (ví dụ phối hợp Confluence và Jira của Atlassian) => càng linh hoạt và tăng hiệu quả.
- Nhược điểm: khi sự thay đổi nhanh và liên tục quá => có thể dẫn đến hiện tượng “quay xe” (the roller coaster ²¹) => giảm hiệu quả công việc, giảm tinh thần chiến đấu.

#3. Vai trò BA trong Agile

BA có thể có các vai trò sau trong dự án Agile/Scrum:

- BA trong Scrum team: giúp điều phối công việc phân tích và làm rõ user stories của Product Owner (PO) cho đội phát triển
- Proxy PO: đại diện cho PO của công ty khách hàng, có thể thay PO đưa ra một số quyết định trong phạm vi user stories và product backlog.

²¹ Roller Coaster: <https://www.linkedin.com/pulse/top-10-dysfunctions-product-management-rajesh-nerlikar/>

- BA Lead: lead các BA trong các dự án lớn, siêu dự án thuộc cùng một domain hoặc một program của công ty phát triển phần mềm.
- PO của công ty khách hàng: giúp xây dựng user stories, product backlog cho công ty. Từ đó làm việc & phối hợp với đội phát triển phần mềm của đối tác

#4. Định hướng nghề BA

Lộ trình phát triển sự nghiệp là sơ đồ định hướng phát triển nghề có cấu trúc logic và khả thi, giúp người BA nắm được các hướng đi công việc dẫn tới vai trò cao nhất có thể đạt được trong sự nghiệp. Khi được áp dụng đúng cách, lộ trình sự nghiệp không chỉ giúp BA nhìn thấy tiềm năng phát triển nghề của họ, mà còn đặt ra các cột mốc rõ ràng để tạo động lực phấn đấu khi bước chân trên con đường ấy.

BA Fresher / Junior => (standard) BA => Senior BA

Một Junior BA (nhập môn) cần có:

- nỗ lực hoàn thành tốt công việc được giao đúng hạn (commitment)
- chủ động xin thêm việc, việc tới đâu hoàn thành tới đó (go extra miles)
- không ngừng học hỏi (non-stop learning)

Một BA chuẩn (thuần thực) cần có:

- không những hoàn thành công việc (đương nhiên, việc cũng khó hơn) mà còn tạo dựng được sự tin cậy vững chắc từ cấp trên (hoặc quản lý dự án và đồng nghiệp)
- tạo tầm ảnh hưởng tốt cho những người thường xuyên làm việc với bạn.
- tiếp tục nỗ lực học hỏi (growth mindset²²).

Một Senior BA (cấp cao) cần có:

- khả năng dự đoán
- lập kế hoạch và xác định phạm vi
- tập trung vào nghiệp vụ
- tư duy hệ thống
- khả năng học một lĩnh vực mới

Một số ví dụ về thang đánh giá kỹ năng của 3 loại BA (nhập môn, thuần thực, và cấp cao)

Kỹ năng	Junior BA (nhập môn)	BA (thuần thực)	Senior BA (cấp cao)
Phân tích Rủi ro	Có thể nhận ra rủi ro nhưng không chắc chắn, họ thường chờ đợi xem điều gì sẽ xảy đến	Họ nhận ra rủi ro và chủ động thông báo cho PM.	Họ nhận ra rủi ro, báo cho đội dự án, đưa ra đề xuất để giảm thiểu rủi ro, và tiếp tục giám sát.
Giải quyết Vấn đề	Thường xuyên không nhận ra vấn đề, và làm theo ý các stakeholder mách bảo.	Nhận ra vấn đề, mô tả vấn đề, nhưng không chắc làm sao để giải quyết.	Nhận ra vấn đề, mô tả vấn đề, và cung cấp ít nhất một giải pháp khả thi.
Điều phối	Chủ yếu lắng nghe trong các buổi thảo luận, và hiếm khi đặt câu hỏi.	Tham gia chủ động các buổi thảo luận / trao đổi. Đặt câu hỏi thẳng vào chủ đề đang bàn bạc.	Chủ động dẫn dắt buổi thảo luận và đưa ra những câu hỏi hỗ trợ đảm bảo đạt được mục tiêu buổi họp.

²² Fixed Mindset vs. Growth Mindset, theo Carol S. Dweck trong cuốn sách "Mindset - How you can fulfil your potential".

#5. Phân biệt Job vs. Career

Tiêu chí	Job	Career
Type	9-6: đủ giờ làm	6-9: đủ giờ làm, và dành thêm thời gian để hiểu kỹ công việc hơn
Motivation	<ul style="list-style-type: none"> – Làm vì đồng lương hiện tại (vì công việc theo mô tả trong job description), hướng đến xong việc hiện tại. – Làm kiểu tích lũy kinh nghiệm (what-if) 	<ul style="list-style-type: none"> – Làm vì đồng lương tương lai (vì sự nghiệp bản thân) hướng đến chuyên nghiệp hóa và nâng cao chất lượng công việc hiện tại. – Làm kiểu tích lũy tri thức (know-how)
Role & Responsibility	<ul style="list-style-type: none"> – Làm theo vai trò và trách nhiệm được giao – Không thường xuyên tự học 	<ul style="list-style-type: none"> – Làm theo tinh thần trách nhiệm cao (ownership) và tính chủ động / có thể dẫn dắt nếu cần (leadership) – Thường xuyên tự học

Tổng kết

Mọi con tàu đều có một thuyền trưởng và người dẫn đường (hoa tiêu), các dự án phần mềm cũng giống một con tàu vậy. Thiếu hai vị trí này, con tàu sẽ đi lạc hướng. Trong hầu hết các tổ chức, người thuyền trưởng thường là quản lý dự án (project manager), quán xuyến chung mọi thứ trên tàu. Từ đó nâng tầm quan trọng của vai trò dẫn đường, bởi con tàu cần di chuyển theo lộ trình cần trọng và hợp lý nhất có thể để tới đích an toàn. BA chính là người dẫn đường quan trọng cho dự án, vì rất nhiều yếu tố có thể làm dự án bị chệch hướng.

Một trong những thách thức khi viết tài liệu yêu cầu cho một hệ thống phần mềm là nó phải dễ hiểu cho hai nhóm đối tượng chính rất khác nhau về tư duy, văn hoá, cách thức giao tiếp: một là bên yêu cầu phần mềm (*stakeholders*), và hai là bên phát triển phần mềm (*development team*). Nếu một trong hai hoặc cả hai bên đều không hiểu yêu cầu bạn viết, qua đó không xác nhận và phê chuẩn hoặc chỉ đơn giản là không sử dụng, thì coi như việc phát triển yêu cầu hoàn toàn thất bại.

Để có thể phát triển phần mềm thành công, trước hết phải phân tích, phát triển, và mô tả yêu cầu chính xác. Việc hiểu đúng yêu cầu giúp bạn "*bắt đầu từ mục tiêu đã định trước*" ²³, và lấy đó làm cơ sở để xác định thành công cho phần mềm phát triển. Sau cùng thì mục đích của phần mềm cũng để thoả mãn nhu cầu người dùng, mà nhu cầu này chính là yêu cầu mà BA xây dựng.

Theo Richards J. Heuer, Jr, một cựu thành viên xuất sắc của CIA Mỹ cho rằng "Phân tích cũng tương tự làm một chiếc xe, đều có thể được đào tạo và học hỏi. Và cũng như đạp xe, việc học lý thuyết hay nghe người khác nói sẽ không đủ, thậm chí không có tác dụng. **Người phân tích học qua thực hành.** Muốn cải thiện tốt kỹ năng phân tích và tư duy logic, cần phải nỗ lực và chăm chỉ qua nhiều quá trình thực hành và trải nghiệm thực tế" ²⁴.

²³ "*begin with the end in mind*", Stephen R. Covey, in the book "7 Habits of Highly Effective People".

²⁴ "Psychology of Intelligence Analysis" by Richards J. Heuer, Jr. <https://www.cia.gov/library/center-for-the-study-of-intelligence/csi-publications/books-and-monographs/psychology-of-intelligence-analysis/PsychofIntelNew.pdf>