**Question 1:**

In this circumstance, I recommend using Agile methodology for some following reasons:

* *In terms of requirements characteristics:*

+ Reliability: The requirements are defined quite clearly but they are not enough and still have some missing points that need to discover later in the development process. So, when applying Agile, after each sprint, the requirements will be more detailed thanks to the collaboration between customers and development team.

+ How often the requirements can change: As mentioned above, the requirements are not 100% clear, fixed and still missing in some points. It is the reason why we are not sure that the requirements will remain unchanged throughout the development process. With the help of the organization, using Agile is a good option because one of the principles of Agile is "Welcome changes", so we can adapt easily with changes in user requirements.

+ Can the requirements be defined at an early stage: the requirements are defined quite clear, but still have some vague points needed to be discovered by the collaboration between the team and customer.

* *In terms of development team:*

+ Team size: From this situation, our team includes 6 developers and 2 QA. Thus, it is very suitable to apply Agile methods because the number of members for a team working in Agile methods is quite small. And the appropriate number of members for Agile method is between 3 and 9 members, so 8 members in this case (6 developers and 2 QA) will help the team operate in the most effective way.

+  Level of understanding of user requirements by the developers: From this circumstance, our developers are expected to have a quite good understanding of requirements. This makes the development process become easier. And with the help of customer to clarify the requirements and their needs, after each sprint, customer requirements become clearer and more detailed, developers' understanding of the requirements improves.

* *In terms of user involvement in the project:*

+ In this case, the organization is expected to contact with our company to provide additional resources. It is a good condition to choose Agile because in Agile, the customer participation will be throughout the product development process, helping to provide feedbacks as well as additional resources to help to improve the quality of product.

* Because of the reasons mentioned above, I suggest using Agile methodology for the above situation.

**Question 2:**

* The list of 4 functional requirements that I want to apply the given project:

+ Managers could provide feedback on employee performance and employee could give feedback on their performance.

+ Managers and employees could create development plans based on areas for improvement identified through performance monitoring.

+ Managers could assess employee competencies, such as technical skills, communicational skills, or leadership abilities.

+ Managers and employees could set performance goals and track progress.

* The list of 2 non-functional requirements that I want to apply the given project:

+ The system should maintain employee privacy by ensuring that only authorized individuals can access employee performance data.

+ The module should be able to handle a growing number of users and tasks without degrading performance or reliability.

**Question 3:**

* The list of 2 user stories based on my answer in question 2:

+ As a manager, I want to performance goals and track my progress so that I can create a detailed plan and schedule to achieve my goals.

+ As a manager, I want to provide feedback on employee performance so that I can help them to improve the quality of their work.

**Question 5:**

* The list of 3 assumptions regarding the competency assessment feature:

+ High impact if wrong, high probability of providing feedback to employee by manager being wrong. (1)

+ High impact if wrong, low probability of setting performance goals and tracking progress being wrong. (2)

+ High impact if wrong, low probability of creating development plans based on areas for improvement identified through performance monitoring. (3)

Explain:

(1): This is because managers could not to assess the performance of employee exactly 100% remotely, due to the surrounding impact, like the noisy of background. And if it happens, the performance of the work will be impact greatly and even can damage to development cost.

(2): This is because nowadays, the technology is becoming modern than ever before, so tracking progress and setting goals are easy task for computer. But if it happens, the impact of it will be huge because we keep tracks wrongly and so can not create a right plan to work towards.

(3): This is because as mentioned above, the current technology is extremely good, that’s why it enables us to create development plans easily. But if the development plans are created wrongly, it will have a negative impact for the whole development process.

**Question 6:**

As we know, functional testing is a type of testing that aims to check whether the features of the product match the customer's expectations. It involves black box testing and has nothing to do with the source code inside. In contrast, non-functional testing is a type of testing that checks aspects such as performance, reliability, security, usability, and so on. Non-functional testing improves the user experience.

From the characteristics of functional testing and non functional testing, in this case, I suggest using functional testing for the team because of two main reasons:

+ Functional testing helps to check whether the product's features meet the needs of the user or not. And if something goes wrong, they will be fixed quickly to avoid huge damage later.

+ Because defects are detected and fixed early, functional testing not only helps companies reduce the time it takes to fix bugs, but also saves money in the long run.

**Phát triển phần mềm 5 bước: Requirement - Design - Implement - Testing - Deploy/Maintaince**

**2 Models chính: Waterfall - Agile**

**Waterfall:**

**Requirement (BA xử lý) - Design (BA xử lý - Dev support) - Implement - Test - Deploy/Maintaince**

**Chạy trong 1 vòng đời phần mềm**

**Deploy được sản phẩm hoàn chỉnh**

**Nhược điểm: Khó sửa lỗi - Tốn thời gian**

**Ưu điểm:**

**Agile:**

**Chia nhỏ vòng đời phần mềm**

**Deploy từng giai đoạn**

**Ưu điểm: Dễ chỉnh sửa, tiết kiệm thời gian**

**Nhược điểm:**

**1. Nêu cái mô hình muốn chọn:**

**I suggest choosing Agile methodology, because:**

**- REQUIREMENTS CHARATERISTIC (đặc điểm yêu cầu)**

**+ Độ chính xác và rõ ràng của yêu cầu (nếu rõ ràng có thể chọn Waterfall, k thì băm agile)**

**+ Có các loại yêu cầu nào: FUNCTIONAL, NON-FUNCTIONAL...**

**+ Độ tin cậy của các loại yêu cầu (yêu cầu có dễ bị thay đổi hay không: không còn phụ hợp hay j đó..., ít thay đổi -> Waterfall, dễ thay đổi -> Agile)**

**+ Nếu chọn Agile, cố gắng nêu ưu điểm so với Waterfall**

**- DEVELOPMENT TEAM (đội ngũ phát triển)**

**+ Quy mô nhóm phát triển: số lượng, vai trò**

**+ mức độ hiểu biết (kinh nghiệm)**

**+ team nhỏ, hiểu biết ít -> Agile, team chuyên nghiệp, yêu cầu rõ ràng -> waterfall**

**+**

**- USER INVOLVEMENT IN THE PROJECT (sự tham gia của người dùng vào dự án)**

**+ Xác định xem người dùng có tham gia nhiều vào dự án hay không**

**+ Đánh giá kích thước của dự án**

**=> Conclusion:**

**Functional:**

**Chức năng có đụng đến data để thực hiện**

**Non-Functional:**

**Tiêu chí hệ thống đạt được khi thực thi (hiệu năng, thực hiện trên môi trường nào, khả năng bảo trì,...)**

**2. Thiếu thì phải bịa**

**- 4 functional requirements for employee:**

**+**

**+**

**+**

**+**

**- 2 non-functional requirements of system:**

**+**

**+**

**3.**

**2 user stories of employee role:**

**+ As a [role], I want to [action], so that [purpose]**

**E.g: As a employee, I want to see my work schedule, so that I can calculate my salary.**

**As a employee, I want to see my shift by month, by week, and by day, so that I can adjust my work schedule accordingly.**

**5.**

**a.**

**+ Luôn giữ quan điểm phương án mình đã chọn ở câu 1**

**+ I do not clearly agree with this require, I will choose ... method for this situation. Because ...**

**+ Luận điểm giống nhưng biến tấu**

**b. Bạn sẽ chọn kiểu kiểm thử nào cho project này**

**- Black box (functional testing): kiểm tra chức năng, kiểm tra tương tác người dùng, không cần quan tâm đến code và logic**

**- White box (non-functional testing): kiểm tra cả thiết kế project, code, thuật toán, cách thực hiện**

**4.**

**- CUSTOMER: người đặt hàng. Là người đưa ra yêu cầu làm phần mềm một cách tổng quát.**

**- USER: người trực tiếp sử dụng. Là người nắm rõ mong ước về phần mềm một cách chi tiết.**

**\* Làm phần mềm chia làm 5 công đoạn lớn:**

**-> REQUIREMENTS: CUSTOMER/USER muốn gì ?**

**-> DESIGN: UI, DB, kiến trúc ( cách bố trí chia code... ) - MVC.**

**-> IMPLEMENT: viết CODE.**

**-> VALIDATION/TESTING**

**-> DEPLOYEMENT/MAINTERNANCE: triển khai cho khách hàng, bàn giao cài lên server... và bảo trì.**

**5 CÔNG ĐOẠN NÀY PHỐI HỢP NHAU ĐỂ RA ĐƯỢC PHẦN MỀM, CÓ NHIỀU CÁCH PHỐI HỢP, TRỘN 5 BƯỚC TRÊN**

**-> TRADITIONAL (WATERFALL...)**

**-> AGILE (SCRUM/XP/KANBAN/LEAN)**

**\*\*\*\*\* QUY TRÌNH LÀM PHẦN MỀM THEO WATER FALL:**

**+ THE WHOLE ( LUÔN LÀM TRÊN TẤT CẢ - HIỂU HẾT BÀI TOÁN MỚI ĐI TIẾP, DESIGN HẾT MỚI ĐI TIẾP,**

**CODE LUÔN CẢ APP, TEST LUÔN CẢ APP )**

**+ CHIA KHÔNG ĐỀU VỀ GIAI ĐOẠN THỜI GIAN**

**REQS**

**DESIGN**

**CODE**

**TEST**

**WORKING SOFTWARE**

**>>>>> APP RA CHẬM, DO ĐI QUA NHIỀU CÔNG ĐOẠN, ĐÒI HỎI TRÊN TẤT CẢ CÁC MODULE**

**\*\*\*\*\* 2001 -> 17 người sếp cty lớn p/m, giáo sư đại học, kiến trúc sư p/m**

**họp lại với nhau -> TUYÊN NGÔN AGILE**

**>>>>> VIẾT RA THÔNG BÁO ĐẶT TRÊN 1 WEBSITE, HƯỚNG DẪN MỌI NGƯỜI CÁCH LÀM P/M KIỂU MỚI**

**>>>>> Link TUYÊN NGÔN AGILE: https://agilemanifesto.org/**

**>>>>> TUYÊN NGÔN AGILE BAO GỒM 4 GIÁ TRỊ VÀ 12 NGUYÊN TẮC**

**\*\*\*\*\* QUY TRÌNH THỰC DỤNG HƠN, QUY TRÌNH LINH HOẠT, AGILE (INTERATIVE & INCREMENTAL - LẶP & TĂNG TRƯỞNG DẦN DẦN)**

**+ DEVIDE & CONQUERE: chia để trị**

**+ tách module ra làm sớm, hoàn thành sớm, lấy feedback sớm.**

**>>>>> HƯỚNG TỚI SỰ HÀI LÒNG CỦA CUSTOMER/END-USER**

**\*\*\*\*\* AGILE**

**SCRUM**

**XP**

**KANBAN**

**LEAN**

**----------- MỤC TIÊU CHUNG ----------------**

**KHOE SẢN PHẨM SỚM TỚI K/H**

**FEEDBACK SỚM TỪ K/H**

**REQS: NHỮNG MONG ƯỚC CỦA K/H VỀ TÍNH NĂNG P/M, MÀN HÌNH PHẦN MỀM, GIÚP CHO CV CỦA HỌ**

**BÊN AGILE NHỮNG REQS ĐƯỢC GỌI LÀ USER STORY**

**as a... i want to... so that...**

**as a student i want to feedback my own lecturer so that...**

**MẸO LÀM VIỆC K/H:**

**1- XÁC ĐỊNH XEM APP VIẾT CHO AI DÙNG (USER TYPES/ROLES/CLASSIFY) \***

**VD: FAP AI DÙNG ?**

**- Student**

**- Lecturer**

**- Academic Staff**

**- Parent**

**VD: PHẦN MỀM BÁN HÀNG ?**

**- Thu ngân/Cashier**

**- Sales**

**- ...**

**VD: QUẢN LÝ THƯ VIỆN**

**- Librian/Bookeeper**

**- Reader**

**- ...**

**2- VỚI MỖI LOẠI NGƯỜI DÙNG/ROLE TA LIỆT KÊ RA NHỮNG TÍNH NĂNG (CÁC MÀN HÌNH = Ô NHẬP + NÚT NHẤN + XỬ LÝ ĐI KÈM**

**-> RA ĐƯỢC 1 KẾT QUẢ MÀ NGƯỜI DÙNG MONG ĐỢI/KẾT QUẢ HỌ VẪN PHẢI LÀM KHI CHƯA CÓ PHẦN MỀM)**

**WHAT, REQUIREMENT, GOAL, TARGET, OBJECTIVE, USER STORY.**

**>>>>> Các tính năng (màn hình) là những thứ mà các user phải làm hằng ngày khi chưa có phần mềm trợ giúp hoặc nếu đã có phần mềm thì đó là phần mềm cũ/version cũ.**

**VD: PHẦN MỀM QUẢN LÍ BÁN HÀNG**

**-> Cashier**

**- Create an order**

**- Scan thông tin khách hàng thân thiết để tích lũy điểm mua hàng**

**-> Sales (admin) administrator (quản trị - người giữ, quản lí thông tin phía sau app khiến cho thông tin luôn sẵn sàng cho mọi roles khác)**

**- Add a new item**

**- Remove a item**

**- Update a item**

**- View report**

**\*\*\*\*\* CÁCH VIẾT REQS NHƯ Ở TRÊN: CÂU ĐI KÈM 2 THỨ: USER ROLE - WHAT (AI LÀM ĐƯỢC GÌ VỚI APP, APP CUNG CẤP TÍNH NĂNG GÌ CHO AI)**

**-> VIẾT REQUIREMENT THEO KIỂU TRUYỀN THỐNG - WATER FALL**

**\*\*\*\*\* AGILE/SCRUM... VIẾT THEO KIỂU TEMPLATE ("MÀU MÈ")**

**AS A... I WANT TO... SO THAT...**

**VD (Ở GÓC NHÌN NGƯỜI DÙNG KHÔNG PHẢI NGƯỜI ĐẶT HÀNG):**

**- AS A CASHIER, I WANT TO A CREATE AN OTHER, SO THAT I CAN RECORD WHAT OUR CUSTOMER BOUGHT.**

**- AS A BOOKEEPER, I WANT TO ADD A NEW BOOK, SO THAT IT IS AVAILABLE TO ANY BORROWER.**

**REQS CHIA LÀM 2 LOẠI CHÍNH: FUNCTIONAL REQUIREMENTS VÀ NON-FUNCTIONAL REQUIREMENTS**

**+ FUNCTIONAL REQUIREMENTS: là những yêu cầu mô tả NHỮNG GÌ HỆ THỐNG PHẢI LÀM ĐƯỢC. Thông thường thì yêu cầu chức năng sẽ chỉ rõ những hành vi hoặc chức năng mà hệ thống sẽ làm.**

**-> Một số FUNCTIONAL REQS:**

**- Business Rules**

**- Transaction corrections, adjustments and cancellations**

**- Administrative functions**

**- Authentication**

**- Authorization levels**

**- Audit Tracking**

**- External Interfaces**

**- Certification Requirements**

**- Reporting Requirements**

**- Historical Data**

**- Legal or Regulatory Requirements**

**- Quy tắc kinh doanh**

**- Chỉnh sửa, điều chỉnh và hủy giao dịch**

**- Chức năng quản trị**

**- Xác thực**

**- Các cấp độ ủy quyền**

**- Theo dõi kiểm tra**

**- Giao diện bên ngoài**

**- Yêu cầu chứng nhận**

**- Yêu cầu báo cáo**

**- Dữ liệu lịch sử**

**- Yêu cầu pháp lý hoặc quy định**

**+ NON-FUNCTIONAL REQUIREMENTS: là các tiêu chí mà hệ thống cần đạt được khi thực thi.**

**-> Một số NON-FUNCTIONAL REQS:**

**- Performance – for example Response Time, Throughput, Utilization, Static Volumetric**

**- Scalability**

**- Capacity**

**- Availability**

**- Reliability**

**- Recoverability**

**- Maintainability**

**- Serviceability**

**- Security**

**- Regulatory**

**- Manageability**

**- Environmental**

**- Data Integrity**

**- Usability**

**- Interoperability**

**- Hiệu suất - ví dụ như Thời gian đáp ứng, Thông lượng, Sử dụng, Thể tích tĩnh**

**- Khả năng mở rộng**

**- Sức chứa**

**- Khả dụng**

**- Độ tin cậy**

**- Khả năng phục hồi**

**- Khả năng bảo trì**

**- Khả năng phục vụ**

**- Bảo vệ**

**- Quy định**

**- Khả năng quản lý**

**- Thuộc về môi trường**

**- Toàn vẹn dữ liệu**

**- Khả năng sử dụng**

**- Khả năng tương tác**

**VD:**

**\* Mũ bảo hiểm:**

**+ FUNCTIONAL REQS:**

**- Hình tròn**

**- Màu đen**

**+ NON-FUNCTIONAL REQS:**

**- Chịu được lực va đập 3000 NEWTON**

**\* Ly nước:**

**+ FUNCTIONAL REQS:**

**- Ly đựng được nước**

**+ NON\_FUNCTIONAL REQS:**

**- Ly rớt hong bể**

**- Chịu được nhiệt độ nước cao**

**\* Phần mềm ABC của bạn Cải**

**+ FUNCTIONAL REQS:**

**- Quản lí thông tin khách hàng**

**+ NON-FUNCTIONAL REQS:**

**- Có nút HELP hướng dẫn người dùng online ngay trên hệ thống.**

**- Yêu cầu tính bảo mật thông tin người dùng...**

**REQUIREMENTS CHARATERISTICS (đặc điểm yêu cầu):**

**+ Độ chính xác và rõ ràng của yêu cầu**

**+ Có các loại yêu cầu nào: FUNCTIONAL, NON-FUNCTIONAL...**

**+ Độ tin cậy của các loại yêu cầu này**

**+ Các yêu cầu này đã rõ ràng chưa, có dễ bị thay đổi không...**

**+ Các yêu cầu có thể được xác định từ ban đầu hay không**

**DEVELOPMENT TEAM (đội ngũ phát triển):**

**+ Quy mô của nhóm phát triển: số lượng, vai trò...**

**+ Mức độ hiểu biết của nhóm phát triển, kinh nghiệm của nhóm...**

**USER INVOLVEMENT IN THE PROJECT (sự tham gia của người dùng vào dự án)**

**+ Xác định xem người dùng có tham gia vào dự án hay không**

**+ Đánh giá kich thước của dự án: nhỏ, trung bình, lớn...**

**( nếu số lượng nhóm phát triển tương đương với số lượng yêu cầu -> trung bình, 5ng - 5cv => 1ng = 1cv**

**nếu số lượng nhóm phát triển nhiều hơn số lượng yêu cầu -> nhỏ, 10ng - 2cv => 1ng = 0.2cv**

**nếu số lượng nhóm phát triển ít hơn số lượng yêu cầu -> lớn ) 5ng - 20cv => 1ng = 4cv**

**-> Chốt lại vấn đề, đưa ra đề xuất mô hình phù hợp với tình huống hiện tại...**

**\*\*\*\*\* MỘT SỐ LOẠI MÔ HÌNH**

**- WATERFALL (MÔ HÌNH THÁC NƯỚC):**

**\* Điều kiện:**

**+ Ứng dụng cho các dự án nhỏ, ngắn hạn**

**+ Các dự án có ít thay đổi về yêu cầu, và các yêu cầu dự án phải rõ ràng**

**+ Nắm vững được công nghệ phát triển.**

**+ Không có những yêu cầu không rõ ràng.**

**+ Áp dụng Waterfall khi nắm được rõ yêu cầu của dự án một cách tốt nhất, yêu cầu là rõ ràng và có tính ổn định cao.**

**\* Ưu điểm:**

**+ Dễ sử dụng, dễ tiếp cận, dễ quản lý.**

**+ Sản phẩm phát triển theo các giai đoạn được xác định rõ ràng.**

**+ Xác nhận ở từng giai đoạn, đảm bảo phát hiện sớm các lỗi.**

**\* Nhược điểm:**

**+ Ít linh hoạt, phạm vi điều chỉnh hạn chế.**

**+ Rất khó để đo lường sự phát triển trong từng giai đoạn.**

**+ Mô hình không thích hợp với những dự án dài, đang diễn ra, hay những dự án phức tạp, có nhiều thay đổi về yêu cầu trong vòng đời phát triển.**

**+ Khó quay lại khi giai đoạn nào đó đã kết thúc.**

**----------------------------------------------------------------------------------------------------------------------------------------------------------------**

**- INCREMENTAL MODEL (MÔ HÌNH TĂNG TRƯỞNG):**

**\* Điều kiện:**

**+ Áp dụng cho những dự án có yêu cầu đã được mô tả, định nghĩa và hiểu một cách rõ ràng.**

**+ Khách hàng có nhu cầu về sản phẩm sớm.**

**+ Sử dụng mô hình này khi một công nghệ mới được áp dụng.**

**+ Tài nguyên và kỹ năng chuyên môn luôn sẵn sàng.**

**+ Khi có một tính năng hay các mục tiêu có nguy cơ lỗi cao.**

**\* Ưu điểm:**

**+ Phần mềm làm việc một cách nhanh chóng trong suốt vòng đời phát triền**

**+ Mô hình này linh hoạt hơn, ít tốn kém hơn để thay đổi phạm vi và yêu cầu**

**+ Dễ dàng hơn trong việc kiểm tra và sửa lỗi với sự lặp lại nhỏ hơn**

**\* Nhược điểm:**

**+ Cần lập plan và thiết kế tốt**

**+ Cần một định nghĩa rõ ràng và đầy đủ của toàn bộ hệ thống trước khi nó có thể được chia nhỏ và được xây dựng từng bước**

**+ Tổng chi phí là cao hơn so với thác nước.**

**----------------------------------------------------------------------------------------------------------------------------------------------------------------**

**- ITERATIVE MODEL (MÔ HÌNH TIẾP CẬN LẶP):**

**\* Điều kiện:**

**+ Yêu cầu của hề thống đã hoàn chỉnh, được xác định rõ ràng và dễ hiểu**

**+ Yêu cầu chính cần được xác định, và một số chi tiết có thể được đổi mới theo thời gian**

**+ Một công nghệ mới đang được sử dụng và đang được học tập bởi nhóm phát triển trong khi làm việc trong dự án.**

**+ Phù hợp cho các dự án lớn và nhiệm vụ quan trọng.**

**\* Ưu điểm:**

**+ Xây dựng và hoàn thiện các bước sản phẩm theo từng bước.**

**+ Thời gian làm tài liệu sẽ ít hơn so với thời gian thiết kế.**

**+ Một số chức năng làm việc có thể được phát triển nhanh chóng và sớm trong vòng đời.**

**+ Ít tốn kém hơn khi thay đổ phạm vi, yêu cầu.**

**+ Dễ quản lý rủi ro.**

**+ Trong suốt vòng đời, phần mềm được sản xuất sớm để tạo điều kiện cho khách hàng đánh giá và phản hồi.**

**\* Nhược điểm:**

**+ Yếu cầu tài nguyên nhiều.**

**+ Các vấn đề về thiết kế hoặc kiến trúc hệ thống có thể phát sinh bất cứ lúc nào.**

**+ Yêu cầu quản lý phức tạp hơn.**

**+ Tiến độ của dự án phụ thuộc nhiều vào giai đoạn phân tích rủi ro.**

**----------------------------------------------------------------------------------------------------------------------------------------------------------------**

**- SCRUM MODEL (MÔ HÌNH SCRUM):**

**\* Điều kiện:**

**+ Scrum phù hợp với những dự án phức tạp, có nhiều yêu cầu**

**+ Scrum sẽ hoạt động tốt nhất trong một nhóm đa chức năng từ 5 đến 9 nhà phát triển làm việc trong một dự án có quy mô vừa và lớn (từ 4 tháng đến nhiều năm).**

**\* Ưu điểm:**

**+ Một người có thể thực hiện nhiều việc ví dụ như dev có thể test.**

**+ Phát hiện lỗi sớm.**

**+ Có khả năng áp dụng được cho những dự án mà yêu cầu khách hàng không rõ ràng ngay từ đầu.**

**+ Khách hàng nhanh chóng thấy được sản phẩm qua đó đưa ra phản hồi sớm.**

**\* Nhược điểm:**

**+ Trình độ của nhóm là có một kỹ năng nhất định**

**+ Phải có sự hiểu biết về mô hình aglie .**

**+ Khó khăn trong việc xác định ngân sách và thời gian.**

**+ Luôn nghe ý kiến phản hồi từ khách hàng và thay đổi theo nên thời gian sẽ kéo dài khi có quá nhiều yêu cầu thay đổi từ khách hàng.**

**+ Vai trò của PO rất quan trọng, PO là người định hướng sản phẩm. Nếu PO làm không tốt sẽ ảnh hưởng đến kết quả chung**

**----------------------------------------------------------------------------------------------------------------------------------------------------------------**

**- KANBAN MODEL (MÔ HÌNH KANBAN):**

**\* Điều kiện:**

**+ Cần một hệ thống linh hoạt để thêm hoặc xóa các mục khi trong quá trình làm việc mà có thay đổi trong thời gian ngắn.**

**+ Dự án của bạn nhấn mạnh nhiều về quy trình làm việc liên tục hơn là những thời hạn hoàn thành đơn lẻ và quan trọng.**

**+ Không có nhiều áp lực về thời gian hoàn thành.**

**+ Cần cải tiến liên tục trong quá trình làm việc.**

**+ Bạn muốn nhóm có khả năng báo cáo kết quả bất cứ lúc nào.**

**+ Nhóm của bạn thích cải thiện gia tăng các quy trình hiện có hơn là áp đặt một hệ thống mới triệt để.**

**+ Hệ thống hiện tại dễ hiểu để áp dụng Kanban.**

**+ Ưu tiên hàng đầu là đáp ứng nhu cầu của khách hàng.**

**+ Đang gặp phải tồn đọng do công việc bị đình trệ, quy trình cơ bản đã ổn định nhưng hoạt động cần mượt mà và hiệu quả hơn.**

**\* Ưu điểm:**

**+ Độ chính xác giờ giấc**

**+ Độ chính xác sản phẩm…**

**+ Tiết kiệm tối đa vật tư và nguyên liệu, giảm trực tiếp chi phí và tránh lãng phí**

**+ Vòng đời sản phẩm quay nhanh vì khả năng phân tán lao động cao.**

**+ Hệ thống đơn giản, dễ hiểu, nhanh nhạy**

**+ Yêu cầu tập trung vào các nhiệm vụ hiện tại cho đến khi hoàn thành**

**+ Mang lại tính minh bạch cho toàn bộ quá trình làm việc**

**\* Nhược điểm:**

**+ Không có khung thời gian của từng giai đoạn**

**+ Kanban sẽ trở nên rất khó áp dụng nếu có quá nhiều hoạt động hoặc nhiệm vụ liên quan đến nhau trong một hệ thống**

**+ Đầu ra có thể không đảm bảo chất lượng**

**+ Phải cập nhật bảng: Các nhóm phải nhấn mạnh tầm quan trọng của việc cập nhật bảng, nếu không, họ có nguy cơ làm việc với thông tin không chính xác**

**----------------------------------------------------------------------------------------------------------------------------------------------------------------**

**- LEAN DEVELOPMENT (PHÁT TRIỂN PHẦN MỀM TINH GỌN):**

**\* Điều kiện:**

**+ Lean hoạt động tốt cho các dự án nhỏ, ngắn hạn do vòng đời của chúng ngắn.**

**+ Cả nhóm phải làm việc tại một văn phòng để có thể giao tiếp**

**+ Nó bao gồm việc triển khai một sản phẩm với bộ tính năng tối thiểu, đủ để đáp ứng những người dùng ban đầu. Ý tưởng của chiến lược MVP là thu thập và phân tích phản hồi của khách hàng để biết liệu họ có thích sản phẩm này và muốn mua sản phẩm đó hay không.**

**+ Kiến thức về thói quen, thị hiếu và nhu cầu của khách hàng là chìa khóa để sản xuất các sản phẩm thành công về mặt thương mại. Các nhà phát triển sử dụng phản hồi để tạo ra một lộ trình phát triển trong tương lai.**

**\* Ưu điểm:**

**+ Cách tiếp cận hợp lý cho phép nhiều chức năng được phân phối trong thời gian ngắn hơn**

**+ Loại bỏ các hoạt động không cần thiết và kết quả là có thể giảm chi phí**

**+ Trao quyền cho nhóm phát triển để đưa ra quyết định, điều này cũng có thể thúc đẩy tinh thần**

**+ Học hỏi từ nhu cầu và phản hồi trực tiếp từ khách hàng tạo ra sự chuẩn bị cần thiết trước những rủi ro**

**\* Nhược điểm:**

**+ Tùy thuộc nhiều vào nhóm tham gia, làm cho nó không thể mở rộng như các framework khác**

**+ Phụ thuộc vào tài liệu mạnh và việc không làm như vậy có thể dẫn đến sai lầm trong phát triển**

**\*\*\*\*\* Draw a story map**

**Chia làm 3 tầng**

**END-USER: WHO IS THE USER?**

**TẦNG 1: USER ACTIVITIES |ACTIVITY A | ACTIVITY B | ACTIVITY C**

**TẦNG 2: USER TASKS |TASK A TASK B|TASK C TASK D|TASK E TASK F TASK G**

**TẦNG 3: FEATURES | FEATURE A1 FEATURE B1 | FEATURE C1 FEATURE D1 | FEATURE E1 FEATURE F1 FEATURE G1**

**| FEATURE A2 FEATURE B2 | FEATURE D2 | FEATURE G2**

**| FEATURE A3 | FEATURE D3 |**

**END-USER: Người dùng, đối tượng sử dụng sản phẩm**

**USER ACTIVITIES: hoạt động của người dùng (Khởi tạo phiên làm việc, Quản lí, Báo cáo, Kết thúc phiên làm việc...)**

**USER TASKS: các nhiệm vụ của từng hoạt động (Đăng nhập/ đăng ký, Quản lí thông tin khách hàng, Báo cáo doanh thu, Đăng xuất)**

**FEATURES: tính năng của từng nhiệm vụ (Kiểm tra/Tạo tài khoản, Tìm kiếm/Thêm/Sửa/Xóa, Gửi báo cáo/ thống kê doanh thu, Đăng xuất)**

**\*\*\*\* Để thực hiện TASK A người dùng cần hoàn thành các FEATURE A1,A2,A3. Để hoàn tất ACTIVITIES A thì người dùng cần thực hiện TASK A, TASK B.**

**\*\*\* black box: test function - kiểm tra chức năng , test behavior - kiểm tra hành vi người dùng, don't care code and logic - hông quan tâm code và thuật toán**

**\*\*\* white box: kiểm tra thiết kế, thuật toán - cấu trúc giải thuật bên trong, việc thực hiện các công việc**

**\*\*\* Software quality attributes that we care about during software architecture are:**

**A. Performance.**

**B. Reliability.**

**C. Testability.**

**D. Security.**

**E. Usability.**

**-> Các thuộc tính chất lượng phần mềm mà chúng tôi quan tâm trong quá trình kiến ​​trúc phần mềm là:**

**A. Hiệu suất. B. Độ tin cậy. C. Khả năng kiểm tra. D. Bảo mật. E. Khả năng sử dụng.**

**--------------------------------------------------------------------------------**

**Question 4:**

Story map for the employee’s screen in mobile app:

**Set goals and track progress**

**Create development plans**

**Give feedback to their performance**

**RELEASE 3**

**Sort goals by the importance ascendingly / descendingly**

**RELEASE 2**

**RELEASE 1**

**Create sub plans (plan B) for unexpected situations**

**View update history**

**Save in private archive**

**Export to PDF**

**Filter feedback based on day, month, year**

**Assess communicational skills**

**Assess technical skills**

**Visualize the progress using charts**

**View all goals**

**Update development plans**

**Use graphic tools to visualize the plans**

**View feedback history**

**Keep track of the progress**

**Write feedback based on performance monthly, annually**

**Set goals by week/month/year**

**Assess leadership skills**