**SWE201c**

1. **Development Methodology**

1. **Waterfall**

1. **Agile**

* Agile is well-suited for complex projects with multiple requirements.

*(Phương pháp Agile thích hợp cho các dự án phức tạp có nhiều yêu cầu.)*

* Agile works best in a cross-functional team of 5 to 9 developers working on a project of medium to large scale (ranging from 4 months to several years).

*(Agile hoạt động tốt nhất trong một nhóm đa chức năng gồm từ 5 đến 9 nhà phát triển làm việc trên một dự án có quy mô từ vừa đến lớn (trong khoảng từ 4 tháng đến vài năm).*

**Advantages:**

* Individuals can perform multiple roles, for example, a developer can perform testing.

*(Các cá nhân có thể thực hiện nhiều vai trò, ví dụ như một nhà phát triển có thể thực hiện kiểm thử.)*

* Early error detection.

*(Phát hiện lỗi sớm.)*

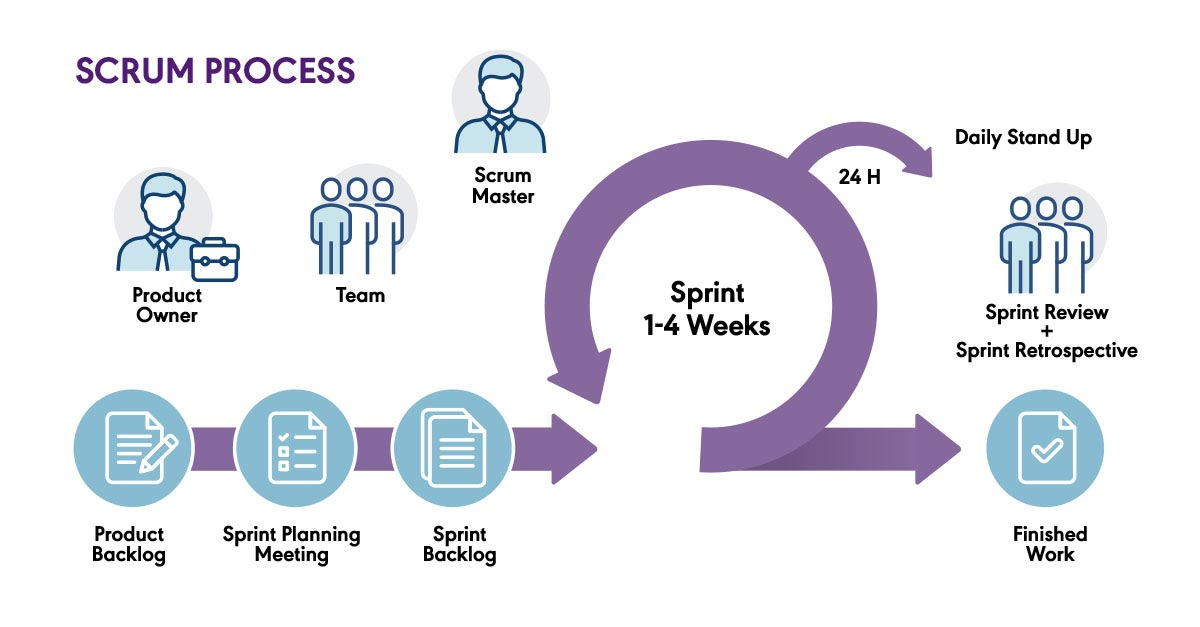
* Applicable to projects with unclear initial customer requirements.

*(Áp dụng được cho các dự án có yêu cầu từ khách hàng không rõ ràng ban đầu.)*

* Customers quickly see the product, thereby providing early feedback.

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

* **Scrum:**

****

Product Owner plays a crucial role in gathering input from customers, users, stakeholders, and other sources to shape and prioritise the Product Backlog.   
*(Product Owner đóng một vai trò quan trọng trong việc thu thập thông tin từ khách hàng, người dùng, các bên liên quan và các nguồn thông tin khác để hình thành và ưu tiên hóa Product Backlog.)*

From there, a Sprint Planning Meeting is established. In this meeting, the development team and Product Owner collaborate to identify specific goals for the upcoming sprint, select items from the Product Backlog to work on, and set detailed plans for the development activities during the sprint.

*(Từ đó, một Cuộc họp Lập kế hoạch Sprint được tạo ra. Trong cuộc họp này, nhóm phát triển và Product Owner cùng nhau làm việc để xác định những mục tiêu cụ thể cho sprint tới, chọn các mục từ Product Backlog để thực hiện, và thiết lập kế hoạch chi tiết cho các hoạt động phát triển trong sprint. )*

And then that starts their execution and that starts the sprint where everybody is working to implement the software. And during that sprint, the whole team gets together for a daily stand-up in which everybody talks about what they did yesterday, what they are going to do today and if there are any roadblocks.

*(Và sau đó, điều này bắt đầu quá trình thực hiện và khởi đầu cho chu kỳ phát triển, nơi mọi người đang làm việc để triển khai phần mềm. Trong suốt chu kỳ phát triển đó, toàn bộ nhóm tụ họp hàng ngày để có một cuộc họp đứng, trong đó mọi người chia sẻ về những gì họ đã làm hôm qua, họ sẽ làm gì hôm nay và nếu có bất kỳ trở ngại nào.)*

The second meeting that happens at the end of the sprint is called a Sprint Retrospective, where they talk about the process and not about the product.

*(Cuộc họp thứ hai diễn ra vào cuối chu kỳ phát triển được gọi là "Sprint Retrospective", trong đó họ thảo luận về quy trình thực hiện mà không liên quan đến sản phẩm.)*

To keep track of the sprint and assess whether we will achieve our sprint goal, the team utilises a project management tool such as Jira or Trello, which displays the remaining amount of work or the remaining days in the sprint.

*(Để theo dõi tiến trình của chu kỳ phát triển và đánh giá xem liệu chúng ta có đạt được mục tiêu của chu kỳ hay không, nhóm sử dụng một công cụ quản lý dự án như Jira hoặc Trello, mà hiển thị số lượng công việc còn lại hoặc số ngày còn lại trong chu kỳ phát triển.)*

*Examples:*

***Q1 Sp23 Retake:***

***Solution:***

*As a scrum master, I have decided to use the agile development methodology because it offers several advantages:*

* *Agile is well-suited for complex projects with multiple requirements.*
* *Agile works best in a cross-functional team of 6 developers and 2 QA working on a project of medium scale*

*(Theo bài ra ta có thể suy đoán đây là một dự án vừa phải, team có 6 devs và 2 QAs nên thật sự thích hợp với ý trên)*

* *Individuals can perform multiple roles, for example, a developer can perform testing.*
* *Early error detection.*
* *Applicable to projects with unclear initial customer requirements.*

*(Ta có thể gạch ý trên bởi vì đây là một dự án khá clear về yêu cầu, tránh viết thừa)*

* *Customers quickly see the product, thereby providing early feedback.*

The model that best fits the description I provided above is Scrum.

Here's how I put it into action:

* To begin, I collect all the necessary requirements from employees to establish a product backlog.
* Subsequently, my team and I convene for a sprint planning meeting to define the specific goals and scope of the upcoming sprint, select and prioritise items from the product backlog to work on, and establish a detailed plan for the development activities to be carried out during the sprint.
* I will break down the project into 3 sprints, with each sprint spanning a duration of one to two weeks.
* **Sprint 1:** *Training course, course catalogue, course schedule,  course delivery (Từ đề bài mà ra)*
* **Sprint 2:** Course assessments, training record, performance tracing *(Cũng từ đề bài mà ra)*
* **Sprint 3:** Feedback and surveys, skill assessments, support resources *(Cũng vậy)*
* My team will concurrently manage both the developing and testing aspects within each Sprint, with six developers focusing on developing and two Quality Assurance (QA) professionals responsible for testing. We use ReactJS for frontend development and .NET for backend development. *(Tận dụng tốt tài nguyên nhân sự bài cho, cho nhiều dùng nhiều, cho ít dùng ít, không cho bốc phét. Câu này cho 6 devs và 2 QAs thì để 6 thằng devs đi phát triển còn 2 thằng QAs đi test là hợp lý zồi, công nghệ thì bốc phét frontend dùng ReactJs, AngularJs,..., backend thì .NET, Java Spring Boot, NodeJS,...).*
* I have chosen to utilise Jira as the tool for project management. *(Xài Jira hay Trello để quản lý đều được, nó ngon như nhau).*

***Solution (Extended Version - Văn vở hơn, điểm cao hơn, cũng giống như ý trên):***

  Bài 9d HCM, có thể tham khảo ở đây

Về phần này chủ yếu là văn  vở hơn ý trên, là trả lời từng câu một. Thường là các bạn sẽ trả lời các ý

* Requirements characteristics là như thế nào?
* Có đáng tin cậy không? (Có - cả Waterfall lẫn Agile đều được, Không - only Agile , chú ý là requirements đáng tin cậy là phải trả lời được 5 câu hỏi - có Consistency không, có  Unambiguous không, có Complete không, có Verifiable không, có Stable không)
* Số lượng requirements là bao nhiêu? Nhiều hay ít? (Ít - cả Waterfall lẫn Agile đều được, Nhiều - Only Agile )
* Requirements có thay đổi thường xuyên hay không? (Không - cả Waterfall lẫn Agile đều được, Có - Only Agile )
* Requirements can be defined clearly in the early stage? (Có - cả Waterfall lẫn Agile đều được. Không - Only Agile )
* Development team
* Team bé (2-4 em), team vừa (4-10 em), team khủng là lớn hơn (Team bé - vừa Waterfall lẫn Agile đều được, lớn thì nên Agile )
* Level of understanding of user requirements by the developers (Trình thấp, dốt - cả Agile lẫn Waterfall đều được, trình cao, tiếp thu nhanh thì Agile )
* User involvement (Liên quan đến người dùng, để lấy feedback, vì Waterfall là không có feedback)
* Liên lạc tốt =>  Waterfall or Agile , Liên lạc không được => Waterfall.

Làm như thế này yêu cầu kĩ năng đọc hiểu phải tốt, ví dụ câu 1

***Solution:***

*As a scrum, I have decided to use the agile development methodology because:*

* Requirements characteristics
* Reliability? Certainly, the requirements exhibit a sufficient level of reliability.
* Consistency : Yes, the requirements are consistency (Chém gió, thấy requirements dễ hiểu nên auto consistency)
* Unambiguous : Yes, the requirements are not ambiguous, since they are quite well defined. (Chém gió, thấy requirements dễ hiểu nên auto unambiguous )
* Complete: Yes, the requirements can be easily understood completely. (Chém gió, thấy requirements dễ hiểu nên auto complete)
* Verifiable: Indeed, the requirements are capable of being verified. (Chém gió, thấy requirements dễ hiểu nên auto verifiable)
* Stable: Yes, the requirements are stable (Chém gió, thấy requirements dễ hiểu nên auto stable)
* Types and number of requirements?
* There may be more than 10 functional requirements defined in the context. (Đọc đề là thấy rõ nó nhiều hơn 10, mỗi cái gạch đầu dòng là đã có nhiều hơn 1 requirements rồi)
* The requirements are defined clearly enough to understand. (Chém gió, thấy requirements dễ hiểu nên cứ ghi)
* Can requirements change occasionally? No, the requirements may change occassionaly (Chém gió, vì không có dữ kiện nào đề cập tới việc này => ghi may change để có lợi cho Agile)
* Requirements is defined clearly at an early stage? Indeed, the requirements are established with clarity in the initial phase, but several requirements are not. (Giống mấy cái trên, thêm cái several requirements are not để có lợi cho Agile)
* Development team
* Numbers of team members? There are 6 members in the team, 4 developers and 2 QAs, so that the team is considered to be a medium team. (Từ bài ra)
* Level of understanding of user requirements by the developers? The team may understand user requirements well (Không có thông tin => ghi may understand để có lợi cho Agile)
* The organisation can provide additional resources and information when needed (Dòng cuối từ đề)
* User involvement
* May contact well with employees to get quality feedbacks (Không có thông tin => ghi may contact để có lợi cho Agile)

The model that best fits the description I provided above is Scrum.

Here's how I put it into action:

* To begin, I collect all the necessary requirements from employees to establish a product backlog.
* Subsequently, my team and I convene for a sprint planning meeting to define the specific goals and scope of the upcoming sprint, select and prioritise items from the product backlog to work on, and establish a detailed plan for the development activities to be carried out during the sprint.
* I will break down the project into 3 sprints, with each sprint spanning a duration of one to two weeks.
* **Sprint 1:** *Training course, course catalogue, course schedule,  course delivery (Từ đề bài mà ra)*
* **Sprint 2:** Course assessments, training record, performance tracing *(Cũng từ đề bài mà ra)*
* **Sprint 3:** Feedback and surveys, skill assessments, support resources *(Cũng vậy)*
* My team will concurrently manage both the developing and testing aspects within each Sprint, with six developers focusing on developing and two Quality Assurance (QA) professionals responsible for testing. We use ReactJS for frontend development and .NET for backend development. *(Tận dụng tốt tài nguyên nhân sự bài cho, cho nhiều dùng nhiều, cho ít dùng ít, không cho bốc phét. Câu này cho 6 devs và 2 QAs thì để 6 thằng devs đi phát triển còn 2 thằng QAs đi test là hợp lý zồi, công nghệ thì bốc phét frontend dùng ReactJs, AngularJs,..., backend thì .NET, Java Spring Boot, NodeJS,...).*
* I have chosen to utilise Jira as the tool for project management. *(Xài Jira hay Trello để quản lý đều được, nó ngon như nhau).*

1. **Requirements**
2. **Functional Requirements**

* Là những yêu cầu chức năng, bám sát đề để ghi, thường là paraphrase từ những yêu cầu có sẵn.

1. **Non Functional Requirements**

* Là những yêu cầu phi chức năng, mà đã phi chức năng tức là không có trong đề. Gặp kiểu này cứ lấy vài cái trong đống dưới rồi paraphrase lại là được.

1. Performance:

* **Response Time:** The system should respond to user interactions within 2 seconds.
* **Throughput:** The system should support a minimum of 1000 concurrent users.
* **Latency:** The system's communication latency between components should not exceed 50 milliseconds.

2. Reliability:

* **Availability:** The system should be available 99.9% of the time.
* **Fault Tolerance:** The system should continue functioning properly even in the presence of hardware or software failures.
* **Recoverability:** The system should be able to recover data and resume normal operation within 1 hour after a failure.

3. Security:

* **Authentication:** Users must be authenticated using two-factor authentication before accessing sensitive data.
* **Data Encryption:** All sensitive data should be encrypted both in transit and at rest.
* **Access Control:** Different user roles should have appropriate access privileges to system features and data.

4. Usability:

* **User Interface Consistency:** The user interface should follow the company's branding guidelines and maintain a consistent design.
* **Accessibility:** The system should comply with accessibility standards (e.g., WCAG) to ensure usability for users with disabilities.
* **User Training:** The system should be intuitive and require minimal user training to operate.

5. Scalability:

* **Vertical Scalability:** The system should be able to handle a 50% increase in load by adding more resources (e.g., CPU, memory).
* **Horizontal Scalability:** The system should support adding additional nodes to the cluster to handle increased load.

6. Maintainability:

* **Modularity:** The system's components should be organised into separate modules to facilitate easy maintenance and updates.
* **Code Documentation:** All code should be well-documented with comments and clear explanations of functionality.
* **Extensibility:** The system's architecture should allow for easy integration of new features and functionalities.

7. Compatibility:

* **Browser Compatibility:** The system should be compatible with the latest versions of popular web browsers (e.g., Chrome, Firefox, Safari).
* **Platform Compatibility:** The system should run on Windows, macOS, and Linux operating systems.

8. Performance Efficiency:

* **Resource Utilisation:** The system should use no more than 50% of available system resources under typical load conditions.
* **Energy Efficiency:** The system should minimise energy consumption to reduce environmental impact.

*Examples:*

***Q2 Sp23 Retake:***

***Solution:***

***Functional Requirements:***

* The system should offer a library of training courses that employees can take to improve their skills and knowledge.
* The system should provide a catalogue of available training courses with detailed descriptions, prerequisites, and recommended audiences.
* The system should allow employees to enrol in upcoming training courses and provide information about course dates, times, and locations.
* The system should offer multiple modes of course delivery, including online, classroom, and self-paced.

*(4 cái này là 4 cái ý đầu tiên trong đề thi và cũng là functional requirements, chưa hề được  paraphrase, đi thi làm ơn nhớ hãy paraphrase dùm).*

***Non Functional Requirements:***

+ The system should respond to user interactions within 2 seconds.

+ All sensitive data should be encrypted both in transit and at rest.

*(Lấy 2 ý trong đống trên, cũng chưa được paraphrase, đi thi làm ơn hãy paraphrase để tránh đạo văn).*

1. **User Story**

Cú pháp để viết User Story:

AS A …, I WANT TO …, SO THAT …

Viết User Stories dựa trên những functional requirements của đề, nhưng mang tính MỞ RỘNG cái requirements đó để làm nó rõ ràng hơn. User Stories sẽ dựa trên góc nhìn của người SỬ DỤNG ứng dụng chứ không phải là người PHÁT TRIỂN ỨNG DỤNG

*Examples:*

***Q3 Sp23 Retake:***

*Solution:*

* As an employee, I want to have a search function for finding the necessary courses so that I can quickly acquire skills and knowledge.

* As an employee, I want specific resources, such as downloadable documents and source code, so that I can access them offline.

(Người trực tiếp sử dụng ứng dụng là employee nên chúng ta phải khai thác vào employee, nên vì vậy cả 2 ý đều là As an employee. 2 ý trên chính là 2 mở rộng cho 2 chức năng trong đề, cụ thể là chức năng đầu tiên và chức năng cuối cùng. Đi thi tập trung toàn bộ chất xám để có thể nghĩ ra những cái User Stories legit nhất có thể.)

1. **Storymap**

Để vẽ story map ta cần tối thiểu là 2 tầng, tối đa là 3 tầng (nếu được hãy vẽ 3 tầng).

*Examples:*

***Q4 Sp23 Retake:***

Vẽ màn hình cho manager

* Note:
* Blue : Activity tasks
* Green: Steps
* Yellow: Details, release 1
* Orange: Details, Step 2

(1 hoặc 2 cái màu xanh chính là 1 dấu tích trong đề)

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Training courses | | | | | Performance tracing | Skill assessment | Feedback and survey | | Support resources |
| Manager libraries | Manage courses | Manage schedules | Manage deliveries | Manage assessments | Manage performance tracing | Manage skill assessment | Manage feedback | Manage survey | Manage resources |
| Add a library | Add a course | Start an enrollment | Add a mode (online, class,...) to an existed course | View evaluations | View tracing result | Add assessment | View feedback list | View survey result | Add resources |
| Delete a library | Delete a course | Close an enrollment | Remove a mode (online, class,...) from an existed course | Analyse evaluations | Analyse tracing result | Delete assessment | Analyse feedback list | Analyse survey result | Delete resource |
| Update a library | Update a course | Update an enrollment |  |  | Start tracing | Update assessment | Reply a feedback | Start a survey | Update resources |
| Search for a library | Search for a course | Add employee to a course by mail |  |  | End tracing | Delete assessment |  | End a survey | Search for an resources |
| Sort libraries | Sort  courses |  |  | Export evaluations (excel, XLS, …) | Tracing automatically | Provide an existed assessment to an employee |  |  | Validate a resource |
| Hide a library | Hide a course |  |  |  |  |  |  |  |  |

* Mỗi ô màu xanh là tập hợp các ý.
* Mỗi ô màu lục là mỗi ý trong đề.
* Mỗi ô màu vàng hoặc cam là details của những cái màu lục. Màu vàng là release 1, màu cam là release là 2. Release 1 tức là những cái quan trọng, release 2 là những cái ít quan trọng hơn.

1. **Risks**

**Các high impact, high probabilities (Ảnh hưởng lớn, dễ xảy ra) dễ xảy ra**

* Rủi ro đồng bộ giữa các thiết bị (thiết bị này được, thiết bị kia không được)
* Rủi ro đường truyền (mạng lag, cá mập cắn cáp,... )
* Rủi ro về hiệu suất hệ thống: Nếu hệ thống không đáp ứng được một lượng lớn người dùng hoặc gặp sự cố về tải, điều này có thể làm suy yếu trải nghiệm của người dùng và gây mất cơ hội kinh doanh.
* Rủi ro về khả năng mở rộng: Nếu hệ thống không được thiết kế để dễ dàng mở rộng để đáp ứng nhu cầu tăng cao, bạn có thể gặp khó khăn trong việc tăng cường quy mô hoặc thêm các tính năng mới.
* Rủi ro về sự phụ thuộc vào nguồn cung cấp ngoại vi: Nếu hệ thống phụ thuộc quá nhiều vào các dịch vụ hoặc tài nguyên từ bên ngoài (ví dụ: API của bên thứ ba), sự cố hoặc thay đổi từ phía bên ngoài có thể ảnh hưởng đến hoạt động của hệ thống.
* Rủi ro về hỗ trợ và đào tạo: Nếu không có đủ tài nguyên hoặc kế hoạch hỗ trợ và đào tạo cho người dùng, họ có thể gặp khó khăn trong việc sử dụng và tận dụng hệ thống một cách hiệu quả.
* Rủi ro về thay đổi yêu cầu từ khách hàng: Khách hàng có thể yêu cầu thay đổi hoặc bổ sung các tính năng sau khi sản phẩm đã hoàn thành, gây ảnh hưởng đến tiến trình phát triển và thời gian hoàn thành.
* Rủi ro về tích hợp hệ thống: Khi tích hợp với các hệ thống hoặc dịch vụ khác, sự cố có thể phát sinh nếu không có sự tương thích hoặc giao tiếp không đúng cách.

**Các high impact, low probabilities (Ảnh hưởng lớn, khó xảy ra) dễ xảy ra**

* Ứng dụng sau khi xuất bản vẫn bug: bug là toang cmmr, nhưng mà khó xảy ra tại check kĩ rồi
* Rủi ro về bảo mật thông tin: Sự cố bảo mật có thể dẫn đến rò rỉ dữ liệu nhạy cảm hoặc việc truy cập trái phép vào hệ thống. Điều này có thể ảnh hưởng lớn đến uy tín của sản phẩm và công ty. Nhưng mà khó xảy ra.

**Các low impact, high probabilities (Ảnh hưởng bé, dễ xảy ra) dễ xảy ra**

* Sai kiến thức trong tài liệu, nội dung truyền tải: Nếu thông tin trong tài liệu hoặc nội dung truyền tải bị sai lệch, có thể dẫn đến hiểu lầm, sự nhầm lẫn, hoặc việc ra quyết định sai. Tuy nhiên, xác suất xảy ra thấp vì thường có quá trình kiểm tra và xác minh thông tin trước khi công bố hoặc truyền tải.
* Rủi ro về thu thập những số liệu và thống kê sai sự thật: Số liệu và thống kê không chính xác có thể dẫn đến quyết định sai lệch hoặc đánh giá không đúng về tình hình. Tuy nhiên, xác suất xảy ra thấp do thường có quy trình kiểm tra và xác minh số liệu.

**Low impact if wrong, low probability of being wrong (Ảnh hưởng bé, khó xảy ra):**

* Sự thay đổi nhỏ trong giao diện người dùng: Một số thay đổi nhỏ trong giao diện người dùng có thể không gây ảnh hưởng đáng kể đến trải nghiệm người dùng hoặc chức năng hệ thống. Xác suất xảy ra thấp do việc kiểm tra và xác minh thông thường được thực hiện trước khi triển khai.
* Rủi ro về việc định dạng văn bản hoặc hiển thị không đúng: Sự cố liên quan đến định dạng văn bản, lỗi hiển thị hay chức năng nhỏ trong giao diện có thể gây mất cân đối nhưng thường không có ảnh hưởng lớn. Xác suất xảy ra thấp do việc kiểm tra và thử nghiệm trước triển khai.

*Examples:*

***Q5 Sp23 Retake:***

Low Impact, Low Probability

**Employee Mistakenly Register for Wrong Course**

Reason: Occasional employee registration errors, stemming from factors like course name confusion or technical issues, carry low probability. The impact remains minimal, as employees can easily rectify the situation by re-registering for the correct course.

Low Impact, Low Probability

**Employee Mistakenly Register for Wrong Course**

Reason: Occasional employee registration errors, stemming from factors like course name confusion or technical issues, carry low probability. The impact remains minimal, as employees can easily rectify the situation by re-registering for the correct course.

Title: High Impact, Low Probability Situation

**Poor Course Content Quality**

Reason: Rare instances of course content misunderstandings by creators or ineffective knowledge dissemination, with low probability of occurrence. The significant impact arises from the potential for employee misunderstanding and incorrect knowledge absorption.

Title: High Impact, High Probability

**Delayed Course Schedule**

Reason: Frequent delays of 10-15 minutes in course schedule due to preparation, resulting in a high probability of occurrence. The considerable impact stems from wasted time for participants and potential conflicts with other appointments.

**Nói chung phần này đòi hỏi chất xám phải cao + tư duy tốt nên nếu không nghĩ ra được gì thì mới chơi văn mẫu như trên.**

1. **Testing**

**Kiểu Test**

**Developer:**

* Blackbox Testing: Developers có thể tham gia vào kiểm thử blackbox bằng cách đảm bảo rằng các tính năng và chức năng đã được phát triển hoạt động chính xác theo yêu cầu và không gây ra lỗi hoặc xung đột trong giao diện người dùng.
* Whitebox Testing: Developers có thể thực hiện kiểm thử whitebox bằng cách kiểm tra mã nguồn và đảm bảo rằng các điều kiện và nhánh mã đã được thực hiện đúng cách và không có lỗ hổng bảo mật.

**QA (Quality Assurance):**

* Blackbox Testing: QA có thể thực hiện kiểm thử blackbox như kiểm tra tính năng, giao diện người dùng và các kịch bản kiểm thử không biết về cấu trúc nội bộ của mã.
* Whitebox Testing: QA có thể kiểm tra các khía cạnh nội bộ của mã và đảm bảo tích hợp giữa các thành phần.

**QC (Quality Control):**

* Blackbox Testing: QC có thể thực hiện kiểm thử blackbox bằng cách kiểm tra tính năng từ góc độ người dùng cuối.
* Whitebox Testing: Tùy thuộc vào khả năng của từng QC, họ có thể tham gia vào kiểm thử whitebox để đảm bảo mã nguồn tuân theo các quy tắc và tiêu chuẩn lập trình.

**Tester:**

* Blackbox Testing: Tester thường tham gia vào kiểm thử blackbox bằng cách kiểm tra tính năng, giao diện người dùng và kiểm tra xem sản phẩm hoạt động như mong đợi từ góc độ người dùng cuối.
* Whitebox Testing: Tester có thể tham gia vào kiểm thử whitebox nếu họ có kiến thức về lập trình và cấu trúc mã.

**Tên các bài test**

**Developer (Blackbox Testing):**

Functionality Verification Test: Kiểm tra tính năng để đảm bảo rằng chức năng hoạt động đúng theo yêu cầu.

User Interface Testing: Kiểm tra giao diện người dùng để đảm bảo tích hợp hợp lý và dễ sử dụng.

Scenario-based Testing: Kiểm thử dựa trên kịch bản để xác minh các tình huống thực tế.

Compatibility Testing: Kiểm tra khả năng tương thích với các môi trường và thiết bị khác nhau.

Usability Testing: Kiểm tra tính khả dụng và trải nghiệm người dùng.

**Developer (Whitebox Testing):**

Code Review and Unit Testing: Xem xét mã nguồn và kiểm tra đơn vị để đảm bảo chất lượng mã.

Code Coverage Analysis: Phân tích phạm vi mã nguồn để đảm bảo rằng các phần quan trọng được kiểm tra.

Security Vulnerability Assessment: Đánh giá lỗ hổng bảo mật để bảo vệ khỏi các lỗ hổng tiềm năng.

Performance Benchmarking: Đánh giá hiệu suất để đảm bảo hiệu suất ổn định.

Integration Testing: Kiểm tra tích hợp giữa các thành phần để đảm bảo hoạt động đồng bộ.

**QA (Blackbox Testing):**

User Acceptance Testing: Kiểm tra chấp nhận từ người dùng để đảm bảo rằng sản phẩm đáp ứng yêu cầu của họ.

Regression Testing: Kiểm tra lặp lại để đảm bảo rằng các tính năng đã kiểm tra vẫn hoạt động sau các thay đổi.

Exploratory Testing: Kiểm thử thám hiểm để khám phá các kịch bản thử nghiệm mới.

Boundary Testing: Kiểm tra giới hạn để xác minh phản ứng của sản phẩm trong các tình huống cận biên.

Error Handling Testing: Kiểm tra xử lý lỗi để đảm bảo rằng sản phẩm xử lý lỗi một cách chính xác.

**QA (Whitebox Testing):**

Code Integration Testing: Kiểm tra tích hợp mã nguồn để đảm bảo tích hợp đúng.

API Testing: Kiểm tra các giao diện lập trình ứng dụng để đảm bảo tích hợp dữ liệu chính xác.

Data Integrity Testing: Kiểm tra tích hợp dữ liệu để đảm bảo tích hợp và lưu trữ chính xác.

Database Testing: Kiểm tra cơ sở dữ liệu để đảm bảo tích hợp và truy vấn dữ liệu chính xác.

End-to-End Testing: Kiểm tra toàn diện từ đầu đến cuối để đảm bảo toàn bộ hệ thống hoạt động một cách liền mạch.

**QC (Blackbox Testing):**

User Experience Testing: Kiểm tra trải nghiệm người dùng để đảm bảo tính khả dụng và tương tác suôn sẻ.

Functionality Validation: Xác nhận tính năng để đảm bảo rằng chúng hoạt động theo mong đợi từ góc độ người dùng cuối.

Scenario-based Testing: Thực hiện kiểm thử dựa trên các kịch bản để xác minh hoạt động thực tế.

Accessibility Testing: Đảm bảo tính truy cập của sản phẩm cho người dùng với các khả năng khác nhau.

Localization Testing: Kiểm tra tích hợp đa ngôn ngữ và vùng địa lý để đảm bảo sự phù hợp với các quy định về ngôn ngữ và văn hóa.

**QC (Whitebox Testing):**

Code Quality Assessment: Đánh giá chất lượng mã nguồn để đảm bảo tuân thủ tiêu chuẩn lập trình.

Code Integration Testing: Kiểm tra tích hợp mã nguồn để đảm bảo tích hợp đúng và hiệu quả.

Data Consistency Testing: Kiểm tra tích hợp dữ liệu để đảm bảo tính nhất quán và đúng đắn.

Database Integrity Testing: Kiểm tra tích hợp cơ sở dữ liệu để đảm bảo tính toàn vẹn và bảo mật dữ liệu.

API Testing: Kiểm tra giao diện lập trình ứng dụng để đảm bảo tích hợp và trao đổi dữ liệu chính xác.

**Tester (Blackbox Testing):**

End-to-End Testing: Kiểm tra toàn diện hệ thống từ đầu đến cuối để đảm bảo tính hoàn chỉnh và tương tác giữa các thành phần.

Usability Testing: Kiểm tra khả dụng của sản phẩm bằng cách đánh giá trải nghiệm người dùng.

Scenario-based Testing: Thực hiện kiểm thử dựa trên kịch bản sử dụng thực tế để xác minh hoạt động chính xác.

Regression Testing: Kiểm tra lặp lại sau các thay đổi để đảm bảo tính hoạt động liền mạch của sản phẩm.

Error Handling Testing: Kiểm tra xử lý lỗi để đảm bảo tính toàn vẹn và thông báo lỗi chính xác.

**Tester (Whitebox Testing):**

Code Review and Unit Testing: Xem xét mã nguồn và kiểm tra đơn vị để đảm bảo chất lượng mã.

Code Coverage Analysis: Phân tích phạm vi mã nguồn để đảm bảo kiểm tra đầy đủ.

Security Testing: Kiểm tra bảo mật để đảm bảo tích hợp an toàn và không có lỗ hổng bảo mật.

Performance Testing: Kiểm tra hiệu suất để đảm bảo tính ổn định và khả năng chịu tải.

Integration Testing: Kiểm tra tích hợp giữa các thành phần để đảm bảo sự liên kết và hoạt động chính xác.

Có văn mẫu ở trên rồi cứ vậy mà áp dụng.

*Examples:*

***Q6 Sp23 Retake:***

ĐỌC PHÁT LÀ BIẾT SẼ CHỌN 1 TRONG CÁC Ý TRÊN ĐỂ TRIỂN KHAI KO LÓI NHÌU!

Team có 6 devs + 2 QAs như vậy sẽ tập trung khai thác vào 2 role này.

(White là test code, Blackbox là test output)

I have opted for a blend of black box and whitebox testing, assigning whitebox testing to our developers and blackbox testing to our Quality Assurance (QA) team. Developers will handle Unit Testing and Integration Testing, guaranteeing the seamless functionality and absence of conflicts in the code. Meanwhile, for QA, I've chosen User Experience Testing to ensure a delightful user experience when interacting with the application.

(Đoạn văn trên cũng là ý từ đống đá ở trên ra chỉ cần biết chơi đúng cách thì sẽ ra đoạn này)

**Solution bài Spring**

[**De**](https://drive.google.com/drive/folders/1P5sUYmKQe0qYHnQficXIHzuIQZ5L4n9Q)

**1,** *As a scrum, I have decided to use the agile development methodology because:*

* Requirements characteristics
* Reliability? Certainly, the requirements are reliable enough.
* Types and number of requirements?
* There may be more than 5 functional requirements defined in the context.
* The requirements are not defined clearly enough to understand. (Cảm giác đề này nó không có rõ ràng lắm, việc ghi không rõ ràng nó sẽ tăng sức mạnh cho quan điểm xài agile)
* Can requirements change occasionally? Yes, the requirements may change occassionaly
* Requirements is defined clearly at an early stage? Indeed, the requirements is not defined clearly. (Cảm thayass những requirements bạn chưa thể xác định từ đầu được=> cho khảo thí biết là những requirements chưa rõ ràng, và như vậy thì việc xài agile nó ok hơn)
* Development team
* We don't know the exact number of team members, but we may have enough members to form an average or large team
* Level of understanding of user requirements by the developers? The team may understand user requirements well
* User involvement
* May contact well with users to get quality feedbacks

The model that best fits the description I provided above is Scrum.

Here's how I put it into action:

* To begin, I collect all the necessary requirements from employees to establish a product backlog.
* Subsequently, my team and I convene for a sprint planning meeting to define the specific goals and scope of the upcoming sprint, select and prioritise items from the product backlog to work on, and establish a detailed plan for the development activities to be carried out during the sprint.
* I will break down the project into 3 sprints, with each sprint spanning a duration of one to two weeks.
* **Sprint 1:** *Comprehensive Management, Promotion Management*
* **Sprint 2:** Listing Management, Analytics
* **Sprint 3:** Manage Discount & Promotion
* My team will concurrently manage both the developing and testing aspects within each Sprint. We use ReactJS for frontend development and .NET for backend development and Flutter for Mobile Development.
* I have chosen to utilise Jira as the tool for project management

2,

Func (đọc đề, viết càng chi tiết càng tốt)

* Business owners are provided with a user interface enabling them to oversee and manage all aspects of their home service.
* Admin can create substantial discounts and bargains on the on-demain service market.
* Admin can view the panel that displays all popular services, newly enrolled consumers and orders that have been fulfilled.
* Admin can interact with a panel to create marketing campaigns.

Non-Func

* The system should respond to user interactions within 2 seconds.
* The system should be compatible with the latest versions of popular web browsers (e.g., Chrome, Firefox, Safari) and mobile devices (Iphone, Ipad).

3,

As an admin, I want to have a search field for services so that I can easily find what service I need.

As an admin, I want to have AI-based suggestions for marketing campaigns so that I can decide which campaign is suitable.

4,

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Comprehensive Management | Promotion Management | Listing Management | | Analytics | | |
| Manage Home Services | Manage Discounts | Manage service providers | Manage provider’s services | Analyse popular services | Analyse enrolled customer | Analyse Orders |
| Add a home service | Add discount to an existed home service | Add a provider | Approve a new service request | View popular service | View enrol customer | View order |
| Update a home service | Update discount | Update a provider | Delete a service for a provider | Visualise data in chart | Visualise data in chart | Visualise data in chart |
| Delete a home service | Remove discount from an existed home service | Remove a provider |  |  |  |  |
| Search for a home service | View Active Discounts | Search for a provider |  |  |  |  |
|  |  | View a provider |  |  |  |  |
| Sort home services | Set Discount Criteria | Ban a provider |  | Export data as EXCEL | Export data as EXCEL | Export data as EXCEL |
| Hide a home service |  | Unban a provider |  |  |  |  |

5,

Tôi không thích Sprial bởi vì những lý do sau

* **The Spiral model is more complex than Scrum (dòng 1 ở dưới)**
* **The Spiral model requires a significant amount of resources than Scrum**
* **….**

**Tao đề xuất sự kết hợp cả 2 whitebox lẫn blackbox cho đội hình này. Với developer tao cho nó chơi whitebox lẫn blackbox bằng 2 pp Unit Test để nó test code và pp là Integration test để nó test các component liệu chúng có work well hay không (những cái đặc điểm của các bài test => ghi rõ ở trên), còn QA tao cho chơi blakbox cụ thể là user experinence testing để nó gom dc feedbck xibj từ người dùng về c ho bố.**

**CAC MO HINH KHAC VS SCRUM**

**Spiral Vs Scrum**

**Spiral:**

Model: The Spiral model is a risk-driven approach to software development that combines iterative development with elements of Waterfall methodology.

Iterations: The project is divided into a series of smaller cycles, known as spirals. Each spiral includes planning, risk assessment, engineering, and evaluation phases.

Risk Management: Emphasizes risk assessment and management throughout the development process, with each spiral addressing identified risks.

Flexibility: Allows for flexibility in accommodating changes and adjustments as the project progresses.

Complex Projects: Suited for large, complex projects where risks need to be carefully managed.

Documentation: Emphasizes comprehensive documentation, which can lead to detailed project records.

Scrum:

Framework: Scrum is an agile framework that follows an iterative and incremental approach to software development.

Roles: Employs specific roles such as Product Owner, Scrum Master, and Development Team.

Iterations: Work is organized into fixed-length time periods called sprints (typically 2-4 weeks). Each sprint results in a potentially shippable product increment.

User Stories: Work is organized into user stories, which represent user requirements or features.

Daily Standups: Daily standup meetings are held to discuss progress, challenges, and plans.

Visibility and Collaboration: Prioritizes visibility, collaboration, and adaptability. Scrum emphasizes frequent inspection and adaptation.

Suitability: Well-suited for projects where requirements are likely to change or evolve, and where continuous feedback and adaptation are necessary.

Less Documentation: While documentation is important, Scrum tends to have less emphasis on comprehensive documentation compared to some traditional methods.

**The Spiral model and Scrum are both software development methodologies, each with its own strengths and weaknesses. Here are some weaknesses of the Spiral model when compared to Scrum:**

* **Complexity: The Spiral model is more complex and may require more detailed planning and documentation due to its iterative nature and the involvement of risk analysis and mitigation at each phase. This complexity can make it harder to manage and implement, especially for smaller projects or teams.**
* **Resource Intensive: The Spiral model requires a significant amount of resources, including time and personnel, to carry out risk analysis, prototyping, and other iterative activities. This can make it more costly and time-consuming compared to Scrum, which emphasizes smaller, more frequent iterations.**
* **Adaptability: While the Spiral model does involve iterative cycles, it might not be as adaptable as Scrum when it comes to changing requirements. Scrum's shorter iterations and regular reviews make it easier to pivot and accommodate changing priorities.**
* **Documentation Overhead: The Spiral model places emphasis on documentation, particularly for risk analysis and mitigation strategies. This documentation can become cumbersome and time-consuming, especially for smaller projects or projects with rapidly changing requirements.**
* **Team Collaboration: The Spiral model may not inherently encourage the same level of collaboration and communication as Scrum. Scrum's daily stand-ups, frequent reviews, and emphasis on cross-functional teams promote better communication and coordination.**
* **Lack of Fixed Schedule: The Spiral model's flexible and iterative nature might lead to a lack of fixed timelines, making it challenging for stakeholders and clients to predict when the project will be completed. Scrum's fixed-length iterations (sprints) and regular release cycles offer more predictability.**
* **Less Emphasis on Deliverables: The Spiral model's focus on risk analysis and exploration might sometimes result in less emphasis on delivering tangible, working software in each iteration. Scrum's commitment to delivering potentially shippable increments after each sprint ensures a steady stream of deliverables.**

**It's important to note that the appropriateness of a methodology depends on the specific project, team, and organizational context. While Scrum might address some of the weaknesses of the Spiral model, it's not a one-size-fits-all solution, and careful consideration should be given to the unique requirements of each project.**

**V-Model Vs Scrum**

**V-Model:**

Approach: The V-Model is a sequential and plan-driven approach that emphasizes validation and verification at each step of development.

Phases: It follows a strict sequence of phases, where each development phase corresponds to a testing phase on the opposite side of the "V."

Testing Emphasis: Testing is a significant focus, with the goal of ensuring that each development phase meets its corresponding testing phase's requirements.

Documentation: Comprehensive documentation is produced at each phase, aiding in the validation and verification process.

Predictability: The V-Model is more predictable and suitable for projects with stable and well-defined requirements.

Suitability: Best suited for projects where requirements are well understood and unlikely to change significantly.

Scrum:

Approach: Scrum is an agile framework that emphasizes iterative and incremental development, collaboration, and flexibility.

Iterations: Work is organized into time-boxed iterations called sprints, typically lasting 2-4 weeks. Each sprint results in a potentially shippable product increment.

User Stories: Work is organized around user stories, representing end-user requirements or features.

Adaptability: Scrum embraces change and allows for adjustments based on feedback. It encourages continuous learning and adaptation.

Collaboration: Collaboration among team members and stakeholders is a core principle. Daily standup meetings and frequent reviews promote communication.

Visibility: Transparency and visibility are emphasized, with product backlogs, sprint backlogs, and burn-down charts providing clear insights into progress.

Suitability: Well-suited for projects where requirements are likely to change or evolve, and where continuous feedback and customer involvement are crucial.

In summary, the V-Model is a structured, predictive approach that emphasizes thorough testing and documentation, while Scrum is an agile framework that prioritizes flexibility, collaboration, and iterative development. The choice between V-Model and Scrum depends on the nature of the project, the level of uncertainty in requirements, and the preferred development philosophy.

**The V-Model and Scrum are two different software development methodologies, each with its own strengths and weaknesses. Here are some weaknesses of the V-Model when compared to Scrum:**

* **Rigidity: The V-Model can be more rigid and less adaptable to changing requirements compared to Scrum. Once the requirements are defined and the corresponding design and tests are created, any changes or deviations can be challenging to accommodate without impacting the entire development process.**
* **Late Testing: The V-Model's testing phase typically occurs later in the development lifecycle, which can lead to the discovery of defects only after significant work has been completed. This can result in higher costs and efforts to rectify issues that could have been identified earlier.**
* **Limited Customer Involvement: The V-Model may not inherently encourage as much customer or stakeholder involvement throughout the development process as Scrum does. Scrum's frequent reviews and iterations provide more opportunities for customer feedback and collaboration.**
* **Documentation Overhead: The V-Model places a strong emphasis on documentation, including detailed requirement specifications, design documents, and test plans. This documentation can become cumbersome and time-consuming, potentially slowing down the development process.**
* **Project Delays: If any phase of the V-Model takes longer than planned or encounters issues, it can lead to project delays. The sequential nature of the V-Model can make it difficult to recover from such delays without affecting subsequent phases.**
* **Limited Flexibility: The V-Model's sequential approach may limit the ability to incorporate changes or new insights as the project progresses. Scrum's iterative nature allows for more flexibility and easier adaptation to evolving requirements.**
* **Risk Management: While the V-Model includes a validation and verification process, it might not address risks as comprehensively as Scrum's iterative cycles, which include regular risk assessment and mitigation.**

**It's worth noting that both methodologies have their own advantages and disadvantages, and the choice between them should be based on the specific project requirements, team dynamics, and organizational context. Scrum's flexibility and focus on collaboration and customer feedback might address some of the weaknesses associated with the V-Model, but it's important to carefully evaluate which approach aligns better with the project's goals and constraints.**

**Iterative Development:**

Approach: Iterative development is a software development methodology that involves breaking a project into smaller parts, called iterations, and developing them incrementally.

Phases: Each iteration includes phases such as planning, design, implementation, testing, and deployment.

Feedback: Iterations allow for frequent feedback and the incorporation of changes or improvements based on that feedback.

Risk Management: The iterative approach allows for better risk management as potential issues can be identified and addressed in early iterations.

Documentation: Documentation is produced throughout each iteration, providing a comprehensive record of the development process.

Predictability: It offers a degree of predictability as the project progresses incrementally, but changes can still be accommodated.

Suitability: Well-suited for projects with evolving or unclear requirements, as well as projects that benefit from early user involvement and feedback.

Scrum:

Approach: Scrum is an agile framework that emphasizes collaboration, adaptability, and iterative development.

Iterations: Work is organized into time-boxed iterations called sprints, typically lasting 2-4 weeks. Each sprint results in a potentially shippable product increment.

Roles: Scrum roles include Product Owner, Scrum Master, and Development Team, promoting clear responsibilities and accountability.

User Stories: Work is organized around user stories, representing user requirements or features.

Collaboration: Scrum encourages close collaboration among team members and stakeholders, with daily standup meetings and regular reviews.

Adaptability: Scrum embraces change and allows for adjustments based on feedback. Continuous learning and improvement are key principles.

Visibility: Transparency is promoted through visual tools like product backlogs, sprint backlogs, and burn-down charts.

Suitability: Suited for projects where requirements are likely to change or evolve, and where collaboration, flexibility, and customer involvement are essential.

Weaknesses of a Traditional Iterative Approach vs. Scrum:

* Lack of Framework: A traditional iterative approach might lack a structured framework like Scrum, which provides specific roles, ceremonies, and artifacts for teams to follow. This can lead to inconsistency in how iterations are planned, executed, and reviewed.
* Incomplete Deliverables: In some traditional iterative approaches, there may be a tendency to focus on technical iterations without ensuring that each iteration produces a fully functional and potentially shippable product increment. Scrum, in contrast, emphasizes delivering valuable working software at the end of each sprint.
* Limited Customer Involvement: Without clear practices for customer collaboration and regular feedback, a traditional iterative approach might result in less involvement from stakeholders or customers, leading to potential misalignments between development and business needs.
* Risk Management: Some traditional iterative approaches may not include built-in mechanisms for regular risk assessment and adaptation, which are core components of Scrum's iterative cycles.
* Adaptability: A traditional iterative approach might lack the level of adaptability and responsiveness that Scrum provides through its short, time-boxed iterations (sprints) and frequent review and planning meetings.
* Documentation Overhead: Similar to other iterative methodologies, a traditional iterative approach might also carry the risk of excessive documentation, which can slow down the development process and reduce flexibility.

It's important to note that Scrum itself is a specific implementation of an iterative approach to software development. Scrum addresses some of the weaknesses associated with more traditional iterative approaches by providing a well-defined framework that promotes collaboration, regular feedback, adaptability, and the delivery of valuable product increments.

When evaluating different iterative approaches, including Scrum, it's crucial to consider the specific context, project requirements, team dynamics, and organizational goals to determine the most suitable approach for a given situation.

**Incremental vs Scrum**

Incremental Development:

Approach: Incremental development is a software development methodology that divides a project into smaller, manageable segments called increments.

Phases: Each increment involves a subset of the complete system's functionality and includes phases like planning, design, implementation, testing, and deployment.

Iterative Nature: Each increment is developed using an iterative approach, meaning that it goes through multiple cycles of development and refinement.

Partial System: After each increment is completed, a partial system with new functionality is delivered and can be used by stakeholders.

Feedback and Evolution: Incremental development allows for early user feedback, and subsequent increments can evolve based on this feedback.

Risk Management: Risks are addressed incrementally, allowing for early identification and mitigation.

Suitability: Suited for projects where delivering functional subsets of the system is valuable, and where user needs may evolve over time.

Scrum:

Approach: Scrum is an agile framework that emphasizes collaboration, adaptability, and iterative development.

Iterations: Work is organized into time-boxed iterations called sprints, typically lasting 2-4 weeks. Each sprint results in a potentially shippable product increment.

Roles: Scrum roles include Product Owner, Scrum Master, and Development Team, promoting clear responsibilities and accountability.

User Stories: Work is organized around user stories, representing user requirements or features.

Collaboration: Scrum encourages close collaboration among team members and stakeholders, with daily standup meetings and regular reviews.

Adaptability: Scrum embraces change and allows for adjustments based on feedback. Continuous learning and improvement are key principles.

Visibility: Transparency is promoted through visual tools like product backlogs, sprint backlogs, and burn-down charts.

Suitability: Suited for projects where requirements are likely to change or evolve, and where collaboration, flexibility, and customer involvement are essential.

Weaknesses of Incremental Development:

* Integration Challenges: With each increment adding new functionality, integration of these increments can become complex and challenging. Ensuring that all increments work seamlessly together can require careful planning and testing.
* System Architecture: Incremental development might lead to evolving the system architecture over time, which could result in a less optimal or well-designed overall structure if not managed properly.
* Incomplete Features: Prioritization of features might lead to certain critical features being deferred to later increments. This could mean that early increments provide a partially functional product, which might not meet users' immediate needs.
* User Feedback Delays: Since functionality is delivered incrementally, users might have to wait for multiple iterations before they can start providing feedback on the complete system. This delay could potentially lead to misunderstandings or misalignments between the development team and users.
* Increased Complexity: As increments are added, the complexity of the project can accumulate. This could result in challenges related to maintainability, troubleshooting, and understanding the overall system.
* Documentation and Communication: Managing documentation and communication about the evolving system can become more challenging, especially when changes are frequent and incremental.
* Unforeseen Dependencies: The introduction of new features in later increments might reveal unforeseen dependencies on previous increments, potentially requiring rework or adjustments.
* Project Management: Incremental development requires effective project management to ensure that increments are well-defined, properly prioritized, and delivered on time. Without proper management, the project might become disorganized or lose sight of its goals.
* Scope Changes: Frequent incremental deliveries could lead to changing requirements or scope adjustments between increments, potentially affecting project timelines and budgets.

While incremental development has its weaknesses, many of these challenges can be mitigated through careful planning, effective project management, and close collaboration between the development team and stakeholders. It's important to weigh the benefits and weaknesses of incremental development against the specific needs and context of your project to determine whether it's the most suitable approach.

**Kanban vs Scrum**

Kanban:

Approach: Kanban is a visual workflow management method that focuses on optimizing the flow of work and improving efficiency.

Process: Work is organized on a Kanban board, which visualizes the workflow stages (e.g., to-do, in progress, done). Tasks move through these stages as they are worked on.

Iterations: Kanban does not require predefined iterations. Work is continuously pulled from the backlog as capacity allows.

Flexibility: Kanban allows for changes and adjustments in real-time. There are no time-boxed iterations, and new work can be added or reprioritized at any time.

Limiting Work in Progress (WIP): Kanban focuses on managing WIP to prevent overloading the team and maintain a smooth flow of work.

Roles: Kanban typically has fewer predefined roles than Scrum. Team members have more flexibility in their responsibilities.

Continuous Improvement: Kanban encourages gradual process improvements based on data and feedback, promoting an evolutionary change approach.

Scrum:

Approach: Scrum is a structured framework with roles, ceremonies, and artifacts. It aims to deliver incremental value through fixed-length time-boxed iterations (sprints).

Iterations: Work is organized into time-boxed sprints (usually 2-4 weeks), and each sprint produces a potentially shippable product increment.

Roles: Scrum has defined roles including Product Owner, Scrum Master, and Development Team, each with specific responsibilities.

Artifacts: Scrum uses artifacts like the product backlog (list of work) and sprint backlog (work selected for the current sprint) to manage and prioritize tasks.

Ceremonies: Scrum includes ceremonies such as sprint planning, daily standups, sprint review, and sprint retrospective to facilitate collaboration and communication.

Predictability: Scrum provides a predictable cadence with fixed iterations, making it easier to plan and manage work.

Adaptability: Scrum allows changes between sprints, but changes are generally minimized during a sprint to maintain focus.

Suitability:

Kanban: Well-suited for teams that require flexibility, have variable workloads, and want to optimize existing processes. It's particularly beneficial for support and maintenance projects.

Scrum: Suited for teams that benefit from fixed iterations, prioritize collaboration and regular feedback, and are looking for a structured framework.

In summary, Kanban focuses on visualizing and optimizing workflow with a flexible and evolutionary approach, while Scrum provides a more structured framework with fixed iterations and specific roles. The choice between Kanban and Scrum depends on the team's preferences, project characteristics, and desired level of structure.

Weaknesses of Kanban Compared to Scrum:

* Less Structured Process: Kanban is often considered less prescriptive and structured compared to Scrum. While this flexibility can be an advantage in some cases, it might lead to ambiguity or lack of clear guidelines for teams, especially those new to agile practices.
* Lack of Time-Boxed Iterations: Kanban does not have fixed time-boxed iterations like Scrum's sprints. This might lead to a lack of urgency or accountability for timely delivery, which could potentially result in delays.
* Limited Predictability: Kanban's focus on continuous flow and variability in work items might make it more challenging to predict when specific features or tasks will be completed, which could affect release planning.
* Dependency Management: Kanban might require more proactive effort to manage dependencies between tasks or work items, as it doesn't have dedicated ceremonies like Scrum's Sprint Planning to address these dependencies upfront.
* Risk Management: Kanban's emphasis on incremental improvements and gradual change might make it less suitable for projects requiring more structured risk assessment and mitigation, compared to Scrum's iterative approach.
* Team Collaboration: While Kanban promotes collaboration, it might not provide the same level of structured collaboration opportunities as Scrum, such as Scrum's Daily Standup and Sprint Review meetings.
* Lack of Time for Reflection: Scrum includes regular retrospective meetings where teams reflect on their processes and make improvements. Kanban might not emphasize this aspect as strongly, potentially hindering continuous process improvement.
* Limited Focus on Roles: Scrum defines specific roles (Product Owner, Scrum Master, Development Team) that provide clarity on responsibilities. Kanban doesn't prescribe roles as explicitly, which could lead to confusion or overlapping responsibilities.
* Less Defined Feedback Loops: Scrum's time-boxed iterations create distinct feedback loops that help teams assess progress. In Kanban, the absence of time-boxed iterations might result in less structured feedback opportunities.

It's important to note that both Kanban and Scrum have their own strengths and weaknesses, and the choice between them should be based on the specific context, project requirements, team dynamics, and organizational goals. Kanban's flexibility might be more suitable for certain environments, while Scrum's structured approach might be better in others.

**XP vs Scrum**

Extreme Programming (XP):

Approach: XP is an agile software development methodology that emphasizes technical practices and high-quality code.

Practices: XP includes practices such as test-driven development (TDD), continuous integration, pair programming, and frequent releases.

Customer Involvement: XP promotes strong customer involvement through practices like on-site customer and user story creation.

Iterations: XP uses iterations, similar to Scrum, to deliver incremental value. However, XP iterations are typically shorter and measured in weeks or even days.

Feedback: Frequent feedback loops are integral to XP. Testing, collaboration, and customer input ensure that the product meets user needs.

Roles: XP has defined roles like Programmer, Customer, and Tracker. The roles emphasize close collaboration and clear responsibilities.

Continuous Improvement: XP promotes continuous improvement through regular retrospectives and adapting practices to enhance development.

Scrum:

Approach: Scrum is an agile framework that emphasizes collaboration, adaptability, and iterative development.

Iterations: Work is organized into time-boxed iterations called sprints, typically lasting 2-4 weeks. Each sprint results in a potentially shippable product increment.

Roles: Scrum roles include Product Owner, Scrum Master, and Development Team, promoting clear responsibilities and accountability.

User Stories: Work is organized around user stories, representing user requirements or features.

Collaboration: Scrum encourages close collaboration among team members and stakeholders, with daily standup meetings and regular reviews.

Adaptability: Scrum embraces change and allows for adjustments based on feedback. Continuous learning and improvement are key principles.

Visibility: Transparency is promoted through visual tools like product backlogs, sprint backlogs, and burn-down charts.

Suitability:

Extreme Programming (XP): Suited for teams that prioritize technical excellence, code quality, and engineering practices. Well-suited for projects where software craftsmanship is a priority.

Scrum: Suited for teams that benefit from fixed iterations, prioritize collaboration and regular feedback, and are looking for a structured framework.

In summary, XP places a strong emphasis on technical practices and customer involvement, while Scrum is a broader framework that focuses on collaboration, adaptability, and delivering value. The choice between XP and Scrum depends on the team's preferences, project characteristics, and desired level of emphasis on technical practices.

Weaknesses of Extreme Programming (XP) Compared to Scrum:

* Documentation Emphasis: XP places a strong emphasis on documentation, including user stories, acceptance criteria, and technical documentation. While documentation is important, an excessive focus on it can slow down development and potentially lead to less time spent on actual coding and testing.
* Formal Roles: XP's emphasis on specific roles like Pair Programmers, Tracker, and Coach could be seen as too rigid for some teams. Scrum's roles are more generalized, allowing for greater flexibility and cross-functional collaboration.
* Small Team Requirement: XP suggests smaller teams, often ranging from 2 to 12 members. While this can foster close collaboration, it might not be as suitable for larger projects or organizations with more complex needs.
* Pair Programming Dependency: XP strongly encourages pair programming, which might not be universally applicable or efficient for all tasks or team dynamics. It can slow down development for some types of work or for individual team members who work better independently.
* Customer Availability: XP requires a high level of customer involvement and availability, which might not always be feasible or practical, particularly in projects with remote customers or time zone differences.
* Continuous Integration: While continuous integration is a core practice of XP, it might not provide as comprehensive an integration and testing framework as Scrum's regular sprints, which allow for more structured testing and validation.
* Rapid Changes: XP's emphasis on embracing changes could lead to instability if not managed well. Frequent changes in requirements might impact the stability and predictability of the project.
* Lack of Formal Time-Boxing: Unlike Scrum's fixed-length sprints, XP doesn't prescribe specific time-boxed iterations. This might lead to less predictable release schedules and potentially slower overall progress.
* Requirement Prioritization: While XP involves customer collaboration, it might not provide as formalized a process for prioritizing requirements as Scrum's Product Backlog refinement and Sprint Planning sessions.
* Adoption Challenges: Extreme Programming introduces a significant shift in team culture, practices, and mindset. Organizations that are new to Agile practices might find it more challenging to adopt XP compared to Scrum, which provides a more gradual transition.

It's important to remember that both Extreme Programming and Scrum have their own strengths and weaknesses, and the choice between them should be based on the specific project requirements, team dynamics, and organizational context. What works best will depend on factors such as team size, project complexity, customer availability, and the organization's overall Agile maturity.

**Waterfall Model:**

Approach: The Waterfall model is a sequential and plan-driven methodology.

Phases: The development process is divided into distinct phases, including requirements, design, implementation, testing, deployment, and maintenance. Each phase is completed before the next one begins.

Documentation: Comprehensive documentation is a key focus, with detailed specifications and plans created at each stage.

Predictability: The Waterfall model provides a clear and predictable project timeline, making it easier to plan and estimate.

Change Management: Change is difficult to accommodate once a phase is completed, which can lead to challenges if requirements evolve.

Suitability: Well-suited for projects with well-defined and stable requirements, where changes are minimal or known in advance.

Scrum:

Approach: Scrum is an agile framework that emphasizes collaboration, adaptability, and incremental development.

Iterations: Work is organized into time-boxed iterations called sprints, typically lasting 2-4 weeks. Each sprint results in a potentially shippable product increment.

Roles: Scrum roles include Product Owner, Scrum Master, and Development Team, promoting clear responsibilities and accountability.

User Stories: Work is organized around user stories, representing user requirements or features.

Collaboration: Scrum encourages close collaboration among team members and stakeholders, with daily standup meetings and regular reviews.

Adaptability: Scrum embraces change and allows for adjustments based on feedback. Continuous learning and improvement are key principles.

Visibility: Transparency is promoted through visual tools like product backlogs, sprint backlogs, and burn-down charts.

Suitability: Suited for projects where requirements are likely to change or evolve, and where collaboration, flexibility, and customer involvement are essential.

Comparison:

Approach: Waterfall is a sequential approach, while Scrum is iterative and incremental.

Documentation: Waterfall emphasizes documentation, while Scrum values working software over comprehensive documentation.

Change Management: Waterfall makes changes challenging once a phase is completed, whereas Scrum accommodates changes between sprints.

Predictability: Waterfall offers predictability but might struggle with evolving requirements, while Scrum is adaptable to changes.

Roles and Collaboration: Scrum emphasizes collaboration and roles like Product Owner and Scrum Master, promoting teamwork and accountability.

The choice between Waterfall and Scrum depends on project requirements, team dynamics, and the desired level of flexibility and collaboration. Waterfall is suitable for stable and well-defined projects, while Scrum is more adaptive to changing requirements and offers ongoing customer involvement.

Weaknesses of Scrum:

* Lack of Detailed Documentation: Scrum's emphasis on working software over comprehensive documentation might lead to gaps in detailed documentation, which could be a concern for projects with strict regulatory requirements.
* Dependency Management: Scrum's focus on delivering increments in each sprint can sometimes overlook dependencies between different user stories, potentially leading to integration challenges later on.
* Predictability for Long-Term Planning: Scrum's iterative and adaptive approach can make it more challenging to predict long-term project outcomes, making it potentially less suitable for projects requiring strict predictability.
* Resource Allocation: Scrum's team-based approach might face resource allocation challenges when team members are shared across multiple projects, affecting the velocity and stability of sprint deliverables.
* Continuous Availability: Scrum requires continuous involvement and availability of the Product Owner and stakeholders, which might be difficult to achieve in some project environments.
* Initial Learning Curve: Transitioning to Scrum requires a mindset shift and learning curve for team members and stakeholders who are not familiar with Agile practices.
* Underestimating Work: Agile estimates can sometimes be overly optimistic, leading to underestimation of work effort and potentially affecting sprint planning and delivery.

Both methodologies have their own strengths and weaknesses, and the choice between them should be based on the specific project requirements, team dynamics, and organizational context.

**ĐỀ FE**

***Solution:***

*As a product owner, I have decided to use the agile development methodology because:*

* Requirements characteristics
* Reliability? Absolutely, the requirements are dependable.
* Types and number of requirements?
* There may be more than7 functional requirements defined in the context. (Đọc đề là thấy rõ nó nhiều hơn 7 mỗi cái gạch đầu dòng là đã có nhiều hơn 1 requirements rồi)
* The requirements are not defined clearly enough to understand. (Chém gió, thấy requirements dễ hiểu nên cứ ghi, quan điểm, cảm thấy như nào thì nó là như vậy, dễ hiểu tức là đọc phát hiểu luôn, còn chưa rõ là hơi confuse)
* Can requirements change occasionally? No, the requirements may change occassionaly (Chém gió, vì không có dữ kiện nào đề cập tới việc này => ghi may change để có lợi cho Agile)
* Requirements is defined clearly at an early stage? Indeed, the requirements are established with clarity in the initial phase, but several requirements are not. (Giống mấy cái trên, thêm cái several requirements are not để có lợi cho Agile)
* Development team
* Numbers of team members? There are 6 members in the team, 4 developers and 2 QAs, so that the team is considered to be a medium team. (Từ bài ra)
* Level of understanding of user requirements by the developers? The team may understand user requirements well (Không có thông tin => ghi may understand để có lợi cho Agile)
* The organisation can provide additional resources and information when needed (Dòng cuối từ đề)
* User involvement

> 5 ý: dễ hiểu, <=5: khó hiểu

* May contact well with employees to get quality feedbacks (Không có thông tin => ghi may contact để có lợi cho Agile)

The model that best fits the description I provided above is Scrum.

Here's how I put it into action:

* To begin, I collect all the necessary requirements from employees to establish a product backlog.
* Subsequently, my team and I convene for a sprint planning meeting to define the specific goals and scope of the upcoming sprint, select and prioritise items from the product backlog to work on, and establish a detailed plan for the development activities to be carried out during the sprint.
* I will break down the project into 3 sprints, with each sprint spanning a duration of one to two weeks.
* **Sprint 1:** *Training course, course catalogue, course schedule,  course delivery (Từ đề bài mà ra)*
* **Sprint 2:** Course assessments, training record, performance tracing *(Cũng từ đề bài mà ra)*
* **Sprint 3:** Feedback and surveys, skill assessments, support resources *(Cũng vậy)*
* My team will concurrently manage both the developing and testing aspects within each Sprint, with six developers focusing on developing and two Quality Assurance (QA) professionals responsible for testing. We use Flutter for mobile development and .NET for backend development. *(Tận dụng tốt tài nguyên nhân sự bài cho, cho nhiều dùng nhiều, cho ít dùng ít, không cho bốc phét. Câu này cho 6 devs và 2 QAs thì để 6 thằng devs đi phát triển còn 2 thằng QAs đi test là hợp lý zồi, công nghệ thì bốc phét frontend dùng ReactJs, AngularJs,..., backend thì .NET, Java Spring Boot, NodeJS,...).*
* I have chosen to utilise Jira as the tool for project management. *(Xài Jira hay Trello để quản lý đều được, nó ngon như nhau).*

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| *Goal Settings* | | *Feedback* | *Competency Assessment* |  |  |  |  |
| *Manage Performance Goals* | *Manage track process* | *Manage received feedbacks* | *Manage assessment* |  |  |  |  |
| *Set performance goal for a task* | *View a task progress* | *View feedbacks* | *View assessments* |  |  |  |  |
| *Edit performance goal for a task* | *Update task progress* | *Note feedback* | *Offer assmentss* |  |  |  |  |
| *Delete performance goal from a task* | *Set complete flag to a task progress* | *Delete feedback (hơi liều)* | *…* |  |  |  |  |
|  |  | *Reply a feedback* |  |  |  |  |  |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| *Virtually Fitting Room* | | *C* |  |  |  |  |  |
| *Try on clothes virtually* | *Manage photos* | *Manage Users* |  |  |  |  |  |
| *Try on clothes using AR* | *Add* |  |  |  |  |  |  |
| *Try on clothes using VR* | *Update* |  |  |  |  |  |  |
|  | *Delete* |  |  |  |  |  |  |
|  | *View* |  |  |  |  |  |  |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| *Size Guides and Recommendations* | *360-Degree Product Views* | *High-Quality Product Images* | |  |  |  |  |  |  |
| *Choose the right size* | *See a garment from different angles* | *Show items in another qualier* | *Zoom the items* |  |  |  |  |  |  |
| *Use an saved size* | *View in fixed angle (frotnr,back, left, right,...)* | *Display item in difffenrent color* | *Zoom out* |  |  |  |  |  |  |
| *Choose an existed size(XL,L,XXL,...)* | *Zoom the items* | *Dispaly item in an other fablic* | *Zoom in* |  |  |  |  |  |  |
| *Choose an manual size* | *View in custom angle* | *Display item in better quality (HD,4k,2k, FHD,...)* |  |  |  |  |  |  |  |
| *Save a size to wishlist (chả quan trong lắm nếu so sánh với đàn anh ở)* |  | *Print the item* | *Manage anti-alising (khử răng cưa)* |  |  |  |  |  |  |

*Use can zoom the item in larger view (2x, 4x,....)*

*Use can Chose clothes in an existed size(XL,L,XXL,...)*

*ð BẢO VỆ QUAN ĐIỂM LÀ SỬ AGILE (Scrum. Kanban,XP,…)*

*As a scrum, I have decided to use the agile development methodology because:*

*·         Requirements characteristics*

*·        Reliability? Certainly, the requirements exhibit a sufficient level of reliability.*

*+ There are 5 features is described in the context, and it is quite will description.*

*·          Types and number of requirements?*

*·        There may be many 50 requirements, since there are 5 featrues is described in the context, and each feature can provides much 10 requirements.*

*·        The requirements are defined clearly enough to understand.  The requiremetns are quite hard to understand, since in the first feature, I do not know exactly how online retailer works.*

*·        Can requirements change occasionally? No, the requirements may change occassionaly (Chém gió, vì không có dữ kiện nào đề cập tới việc này => ghi may change để có lợi cho Agile)*

*·        Requirements is defined clearly at an early stage? No, the requiremetns are not defined clearly at early since the the 5 feature are describie generally, there are too short for defind the requirements in early starge*

* *Development team*

*·        Numbers of team members? There are 7 members in the team, 4 developers and 2 QAs and me = PO, so that the team is considered to be a medium team. (Từ bài ra) 5-10 medium, > 10 large, < 5 team small (khó ra small)*

*·        Level of understanding of user requirements by the developers? The team may understand user requirements well (Không có thông tin => ghi may understand để có lợi cho Agile)*

*·        The organisation can provide additional resources and information when needed*

* *User involvement*

*·        May contact well with customer to get quality* ***feedbacks*** *(Không có thông tin => ghi may contact để có lợi cho Agile)*

*The model that best fits the description I provided above is Scrum.*

*Here's how I put it into action:*

*·        To begin, I collect all the necessary requirements from employees to establish a product backlog.*

*·        Subsequently, my team and I convene for a sprint planning meeting to define the specific goals and scope of the upcoming sprint, select and prioritise items from the product backlog to work on, and establish a detailed plan for the development activities to be carried out during the sprint.*

*·        I will break down the project into 3 sprints, with each sprint spanning a duration of one to two weeks.*

*·* ***Sprint 1:*** *Training course, course catalogue, course schedule,  course delivery (Từ đề bài mà ra)*

*·* ***Sprint 2:*** *Course assessments, training record, performance tracing (Cũng từ đề bài mà ra)*

*·* ***Sprint 3:*** *Feedback and surveys, skill assessments, support resources (Cũng vậy)*

*·        My team will concurrently manage both the developing and testing aspects within each Sprint, with six developers focusing on developing and two Quality Assurance (QA) professionals responsible for testing. We use ReactJS for frontend development and .NET for backend development. (Tận dụng tốt tài nguyên nhân sự bài cho, cho nhiều dùng nhiều, cho ít dùng ít, không cho bốc phét. Câu này cho 6 devs và 2 QAs thì để 6 thằng devs đi phát triển còn 2 thằng QAs đi test là hợp lý zồi, công nghệ thì bốc phét frontend dùng ReactJs, AngularJs,..., backend thì .NET, Java Spring Boot, NodeJS,...).*

*·        I have chosen to utilise Jira as the tool for project management. (Xài Jira hay Trello để quản lý đều được, nó ngon như nhau).*

*As a customer I want to have zoom function for clothes so that I can view the cloth in better quality..*

*AS A … I WANT …. SO THAT*

*HIGH IMPACT, HIGH PROBABILITY: User cannot return cloth if purchased broke.*

*Reason (Lý do này chính là lý do vì sao nó là hig-gih): Người có thể vô tình làm hỏng trang phục bởi vì làm dơ, làm rách, làm giãn,... và việc này rất dễ xảy ra nên nnos là high probablilty.  If customer cannot return clothes, they may waste a lot of moneys in unusable cloth, so that it is high impact.*