Question 1:

Đê 9,5:

In this situation, I suggest employing the Agile methodology for several reasons that follow:

* *In terms of requirements characteristics:*

+ Reliability: The specifications are reasonably clear, yet they remain insufficient with some gaps that must be uncovered during the later stages of development. By adopting Agile, after each sprint, the requirements will become more comprehensive due to the collaborative efforts between customers and the development team.

+ How often the requirements can change:   
As stated earlier, the requirements are not entirely clear, fixed, and still exhibit some gaps. This uncertainty raises doubts about the stability of the requirements throughout the development process. Leveraging Agile is a favorable choice with organizational support because one of its principles is to "Welcome changes," enabling easy adaptation to shifts in user requirements.

+ Is it possible to establish the requirements early on? While the requirements are generally clear, there are still certain ambiguous aspects that require discovery through collaboration between the team and the customer.

* *In terms of development team:*

+ Team size: From this situation, our team includes 6 developers and 2 QA. Hence, the application of Agile methods is highly fitting since Agile teams typically consist of a relatively small number of members. The optimal team size for Agile methodologies ranges between 3 and 9 members, and having 8 members in this case (comprising 6 developers and 2 QA) will enhance the team's operational efficiency.

Top of Form

+  Understanding of user requirements by developers: In this situation, it is anticipated that our developers possess a solid comprehension of the requirements, facilitating a smoother development process. Collaborating with the customer to elucidate requirements and needs enhances clarity and detail in customer requirements after each sprint, thereby refining the developers' understanding of the requirements.

*In terms of user involvement in the project:*

+ In this scenario, the organization is anticipated to engage with our company to allocate extra resources. Opting for Agile is advantageous in this context, as it involves continuous customer participation throughout the product development process, enabling feedback and the provision of additional resources to enhance product quality.

* Due to the aforementioned reasons, my recommendation is to implement the Agile methodology in the given situation.

Đề 9:

In this instance, I recommend adopting the Agile methodology for the project, and the rationale for this choice is outlined below:

**Requirements Characteristics:**

* *Reliability:* The requirements for a basic remote working software helper are well-defined, and there is a high demand for the product.
* *Functional and Non-functional Requirements:* The project includes at least 5 functional and 3 non-functional requirements.
* *Future Changes in Requirements:* Given the unprecedented nature of the COVID-19 pandemic, requirements may evolve based on user experiences and needs. Flexibility is crucial to accommodate potential changes.
* *Early Stage Definition of Requirements:* The requirements were initially defined clearly.

**Desirable Time for Product Introduction:**

* The urgency for introducing the product or its features is very high due to the significant impact of the COVID-19 pandemic, with a widespread need for efficient remote work solutions.

**Tools and Technology:**

* The tools and technology used in the project can be subject to change to keep up with advancements and improve user experience. Adapting to newer technologies is essential in a rapidly evolving landscape.

**Development Team:**

* The team comprises 6 developers and 2 QA members.
* The development team has a basic understanding of the project, given the unprecedented nature of the pandemic and the novelty of addressing such a large-scale negative impact.

**User Involvement:**

* The level of user involvement is substantial, focusing on indoor company use and employee engagement.
* The project size is deemed medium.

Considering these characteristics, the Agile methodology is most suitable for this project. Its adaptability to changing requirements, emphasis on user involvement, and ability to accommodate a developing understanding within the development team align well with the project's unique features. The core functionalities of the project include task management, deadline tracking, and progress tracking.

Đề 8 điểm:

**Question 1:**

1. Requirements characteristics
   * Reliability:
     + As previously mentioned, a clearly identified problem necessitates the development of an application to address it.
     + The project requirements are explicitly outlined and feasible.
     + The successful completion of the project renders it executable.

=> The project has a high reliability.

* + Types and number of requirement:
    - The software requirements encompass both functional and non-functional aspects..
    - All requirements are explicitly and straightforwardly articulated..
    - The total number of listed requirements exceeds 7, comprising 5 functional requirements and 2 non-functional requirements as detailed earlier..

=> The project's requirements, both in terms of type and quantity, are specified, and the complexity level is manageable for the team.

* + How often the requirement can change:
    - The previously mentioned requirements represent only a subset of this application's features. .
    - Given the rapidly evolving nature of the Covid-19 situation during the project, certain features may be adjusted and modified to align with customer requirements..

=> The requirements may be changed regularly in the future.

* + Determination of requirements at an early stage
    - While some requirements are thoroughly detailed above, they are insufficient for constructing a fully realized system..
    - The organization retains the flexibility to incorporate or eliminate features throughout the project's progression..

=> It is well-defined but not enough.

* Development team
  + Team size:
    - The situation above mentioned our team has 9 people.
    - It is *6 developers, 2 QA* and a leader team who is me.

1 người - < 1 công viec : team lon

1 người - 1 cv : team trung binh

1 nguoi - >1.5 cv: TEAM NHO

=> It is an average team size and enough to build a project that was not too complex.

* + Level of understanding of user requirements by the developers:
    - All of the requirements defined clearly above and our members can understand.
    - The organization can provide additional resources and information when needed.

=> Our team can easily understand and build an application that meets the requirements.

* User involvement
  + The situation mentioned “*The organization had contracted with a local company to provide additional resources when needed*.”
  + We have a contact with the organization to communicate and give feedback about the project.

=> The user involvement is high.

In conclusion, considering the outlined requirements characteristics, I recommend the utilization of the Agile/Scrum methodology.The system must be delivered earlier for the user to buy quickly in Covid-19 context, therefore they should deliver part of the project earlier to get feedback from the user, then apply necessary change for another part. This user-focused methodology guarantees the alignment of the final product with user requirements. In general, the Agile/Scrum approach is aptly suited for this software development project and is expected to yield a high-quality end product.

Question2: Test

Dề 8 điểm:

The project description lacks information about the tester's knowledge or experience, making black-box testing an appropriate choice. Additionally, this testing method doesn't necessitate specialized expertise from the analyst, as it doesn't require in-depth technical knowledge of the system. Hence, I recommend opting for black-box testing as it facilitates user involvement in identifying and providing feedback, especially regarding the system's usability.

Đề 9 điểm:

In this scenario, I recommend employing a blend of manual testing and automated testing. Automated testing is not comprehensive for certain aspects such as performance and assessing the user-friendliness of the application interface. Manual testing will encompass two types: White box and Black box.

* White box testing is utilized to identify errors in the code and examine the code flow, aiding the development team in efficiently addressing errors during subsequent software maintenance.
* Black box testing is employed to verify whether the application aligns with all functional and non-functional customer requirements. It also assesses user experience-related issues such as the user-friendly interface and overall performance. Automated testing, on the other hand, contributes to cost reduction, time efficiency in the testing process, and increased error-finding effectiveness due to its automated and regular deployment.

Bài 9,5 điểm:

As we are aware, functional testing is a testing method aimed at verifying whether the product's features align with customer expectations. It employs black-box testing and does not involve examination of the source code. On the other hand, non-functional testing assesses aspects such as performance, reliability, security, and usability, ultimately contributing to an enhanced user experience.

Given the characteristics of functional and non-functional testing, I recommend opting for functional testing in this instance for two primary reasons:

* Functional testing ensures that the product's features meet user needs, and any identified issues can be promptly addressed to prevent significant repercussions later on.
* By detecting and addressing defects early in the process, functional testing not only reduces the time required to rectify bugs but also leads to long-term cost savings for the company.

Question4:

Non-funtional:

1. **Performance:**
   * *Response Time:* The system is required to respond to user interactions within a 2-second timeframe.
   * *Throughput:* The system must support a minimum of 1000 concurrent users.
   * *Latency:* The communication latency between the system's components should not exceed 50 milliseconds.
2. **Reliability:**
   * *Availability:* The system should maintain a 99.9% availability rate.
   * *Fault Tolerance:* The system is expected to continue functioning properly despite hardware or software failures.
   * *Recoverability:* The system should recover data and resume normal operation within 1 hour after a failure.
3. **Security:**
   * *Authentication:* Users are required to undergo two-factor authentication before accessing sensitive data.
   * *Data Encryption:* All sensitive data must be encrypted both in transit and at rest.
   * *Access Control:* Different user roles should possess appropriate access privileges to system features and data.
4. **Usability:**
   * *User Interface Consistency:* The user interface should adhere to the company's branding guidelines and maintain a consistent design.
   * *Accessibility:* The system must comply with accessibility standards (e.g., WCAG) to ensure usability for users with disabilities.
   * *User Training:* The system should be intuitive, requiring minimal user training for operation.
5. **Scalability:**
   * *Vertical Scalability:* The system should handle a 50% increase in load by adding more resources (e.g., CPU, memory).
   * *Horizontal Scalability:* Support for adding additional nodes to the cluster to handle increased load.
6. **Maintainability:**
   * *Modularity:* The system's components should be organized 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 must be compatible with the latest versions of popular web browsers (e.g., Chrome, Firefox, Safari).
   * *Platform Compatibility:* The system should run seamlessly on Windows, macOS, and Linux operating systems.
8. **Performance Efficiency:**
   * *Resource Utilization:* The system should not exceed 50% of available system resources under typical load conditions.
   * *Energy Efficiency:* The system should minimize energy consumption to reduce environmental impact.
9. **Hiệu suất:**
   * *Thời gian Phản hồi:* Hệ thống cần phản hồi đối với tương tác của người dùng trong khoảng 2 giây.
   * *Khả năng xử lý:* Hệ thống phải hỗ trợ ít nhất 1000 người dùng đồng thời.
   * *Độ trễ:* Độ trễ truyền thông giữa các thành phần của hệ thống không được vượt quá 50 mili giây.
10. **Độ tin cậy:**
    * *Sẵn có:* Hệ thống phải duy trì tỷ lệ sẵn có là 99,9%.
    * *Khả năng chống lỗi:* Hệ thống dự kiến sẽ tiếp tục hoạt động đúng đắn ngay cả khi có sự cố về phần cứng hoặc phần mềm.
    * *Khả năng khôi phục:* Hệ thống phải có khả năng khôi phục dữ liệu và tiếp tục hoạt động bình thường trong vòng 1 giờ sau một sự cố.
11. **Bảo mật:**
    * *Xác thực:* Người dùng phải trải qua xác thực hai yếu tố trước khi truy cập dữ liệu nhạy cảm.
    * *Mã hóa dữ liệu:* Tất cả dữ liệu nhạy cảm phải được mã hóa cả khi truyền và khi lưu trữ.
    * *Kiểm soát Truy cập:* Các vai trò người dùng khác nhau phải có đặc quyền truy cập phù hợp vào các tính năng và dữ liệu của hệ thống.
12. **Tiện ích sử dụng:**
    * *Tính nhất quán của Giao diện Người dùng:* Giao diện người dùng phải tuân theo hướng dẫn về thương hiệu của công ty và duy trì một thiết kế nhất quán.
    * *Tiện ích:* Hệ thống phải tuân thủ các tiêu chuẩn về tiện ích (ví dụ: WCAG) để đảm bảo tính khả dụng cho người sử dụng có khuyết tật.
    * *Đào tạo Người dùng:* Hệ thống phải dễ sử dụng và đòi hỏi ít đào tạo người dùng nhất có thể.
13. **Khả năng Mở rộng:**
    * *Mở rộng Theo chiều dọc:* Hệ thống phải có khả năng xử lý một tăng 50% trong lượng công việc bằng cách thêm nguồn lực (ví dụ: CPU, bộ nhớ).
    * *Mở rộng Theo chiều ngang:* Hệ thống phải hỗ trợ thêm các nút vào cụm để xử lý tăng cường.
14. **Dễ bảo trì:**
    * *Phân khối:* Các thành phần của hệ thống phải được tổ chức thành các mô-đun riêng biệt để dễ bảo trì và cập nhật.
    * *Tài liệu mã nguồn:* Tất cả mã nguồn phải được tài liệu đầy đủ với các chú thích và giải thích rõ ràng về chức năng.
    * *Khả năng Mở rộng:* Kiến trúc của hệ thống phải cho phép tích hợp dễ dàng các tính năng và chức năng mới.
15. **Tương thích:**
    * *Tương thích với Trình duyệt:* Hệ thống phải tương thích với các phiên bản mới nhất của các trình duyệt web phổ biến (ví dụ: Chrome, Firefox, Safari).
    * *Tương thích với Nền tảng:* Hệ thống phải chạy trơn tru trên các hệ điều hành Windows, macOS và Linux.
16. **Hiệu suất Hiệu quả:**
    * *Sử dụng nguồn lực:* Hệ thống không nên sử dụng quá 50% tài nguyên hệ thống có sẵn dưới điều kiện công việc bình thường.
    * *Hiệu quả Năng lượng:* Hệ thống phải giảm thiểu tiêu thụ năng lượng để giảm thiểu ảnh hưởng môi trường.

Question7:

**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.

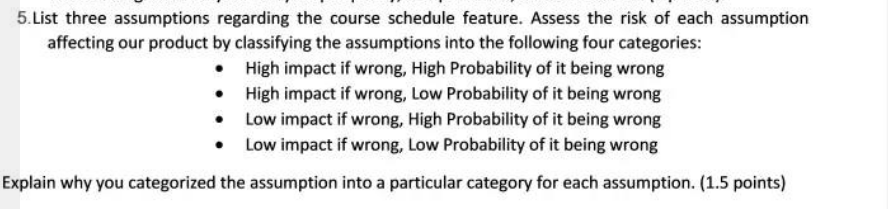
**Low impact if wrong, low probability of being wrong (Ảnh hưởng bé, khó xảy ra):**

* 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.**

Question3: write 4 test cases or a descriptive narrative for what you expect the testing team to use when testing this product.

1. **Test Case: Response Time Verification**
   * *Objective:* To ensure the system adheres to the specified response time requirement.
   * *Steps:*
     1. Perform a series of user interactions within the system.
     2. Measure the time taken for the system to respond to each interaction.
   * *Expected Result:* The response time for all interactions should be within the defined limit of 2 seconds.
2. **Test Case: Concurrent User Throughput**
   * *Objective:* To verify the system's ability to support the specified number of concurrent users.
   * *Steps:*
     1. Simulate a load of 1000 concurrent users accessing the system.
     2. Monitor the system's performance under this load.
   * *Expected Result:* The system should effectively handle the load, and performance metrics should meet or exceed the specified threshold.
3. **Test Case: Authentication and Access Control**
   * *Objective:* To validate the security features related to user authentication and access control.
   * *Steps:*
     1. Attempt to access sensitive data without proper authentication.
     2. Verify that users undergo two-factor authentication before accessing sensitive information.
     3. Test different user roles to ensure appropriate access privileges are enforced.
   * *Expected Result:* Unauthorized access attempts should be denied, and users must pass two-factor authentication. Different roles should have correct access permissions.
4. **Test Case: Compatibility Testing**
   * *Objective:* To confirm the product's compatibility with different browsers and operating systems.
   * *Steps:*
     1. Access the system using various web browsers (e.g., Chrome, Firefox, Safari).
     2. Run the system on different operating systems (e.g., Windows, macOS, Linux).
   * *Expected Result:* The system should function seamlessly on all tested browsers and operating systems without any compatibility issues.

These test cases aim to cover different aspects, including performance, security, and compatibility, to ensure a comprehensive and reliable testing process for the product.