ĐẠI HỌC QUỐC GIA TP. HCM

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

BÀI TẬP VỀ NHÀ BUỔI 3

QUY TRÌNH AGILE

Giáo viên hướng dẫn: ThS. Võ Tấn Khoa

Nhóm thực hiện:

1. **Nguyễn Nhật Trường - 20522087**
2. **Lại Chí Thiện - 20520309**
3. **Lê Thị Phương Vy - 20520355**
4. **Lê Trương Ngọc Hải - 20520481**
5. **Trần Văn Lực - 20521587**
6. **An Trương Tường Vi - 20520856**
7. **Trần Thị Anh Thư - 20520792**
8. **Trương Thị Mai Trinh - 20520825**
9. **Đỗ Thị Ngọc Bích - 19521263**

TP. HỒ CHÍ MINH, 2022

PHÂN TASK CHO TỪNG THÀNH VIÊN

STT	MSSV	HỌ TÊN	TASK
1	20522087	Nguyễn Nhật Trường	Phân task, quản lí tiến độ và hỗ trợ nhóm, làm câu 5
2	20520481	Lê Trương Ngọc Hải	Làm câu 8,10
3	20520355	Lê Thị Phương Vy	Làm câu 6
4	20520309	Lại Chí Thiện	Làm câu 7
5	20521587	Trần Văn Lực	Làm câu 1
6	20520856	An Trương Tường Vi	Làm câu 2
7	20520792	Trần Thị Anh Thư	Làm câu 3
8	20520825	Trương Thị Mai Trinh	Làm câu 4
9	19521263	Đỗ Thị Ngọc Bích	Làm câu 9

BÀI TẬP VỀ NHÀ

1. **(Lực)** - Câu 1: Vào cuối chương trình học, sinh viên của các ngành công nghệ thông tin thường phải hoàn thành một dự án lớn. Giải thích cách phương pháp Agile có thể hữu ích cho sinh viên sử dụng trong trường hợp này.

Với việc phải hoàn thành một dự án lớn thì phương pháp Agile lấy nhóm làm trung tâm rất hữu ích cho sinh viên trong việc hỗ trợ tương tác giữa các thành viên,với phương pháp Agile thì tài liệu thiết kế được giảm thiểu cùng với đó là phương pháp này tập trung chủ yếu vào xây dựng mã code và đây là một trong những thế mạnh của sinh viên các ngành công nghệ thông tin.

Phương pháp Agile sẽ giúp dự án lớn dể quản lý hơn vì phương pháp này làm việc theo từng phần nhỏ giúp sinh viên hiểu sâu hơn vào từng phần cũng như cả quá trình hoàn thành dự án lớn và thường xuyên đánh giá các phiên bản để phát triển giúp đảm bảo chúng hoạt động tốt nhưng nó cũng đòi hỏi sinh viên phải được dạy hoặc tự học những kỹ thuật đó trong quá trình học để thực hiện.

2. **(Vi)** - Câu 2: Giải thích cách các nguyên tắc cơ bản của phương pháp Agile thực hiện dẫn đến việc phát triển và triển khai phần mềm được tăng tốc.

Agile là phương pháp phát triển phần mềm linh hoạt, được ứng dụng trong quy trình phát triển phần mềm với mục tiêu là đưa sản phẩm đến tay người dùng càng nhanh càng tốt. Phương pháp này sẽ thực hiện thay đổi hệ thống theo từng bước nhỏ tùy thuộc theo yêu cầu của khách hàng.

Cách thức để các nguyên tắc cơ bản của phương pháp Agile góp phần làm cho việc phát triển và triển khai phần mềm được tăng tốc có thể giải thích như sau:

1. Sự quan tâm của khách hàng (Customer involvement): Khách hàng tham gia chặt chẽ trong suốt quá trình phát triển để cung cấp các yêu cầu về tính năng mới cho hệ thống, đánh giá trực tiếp và đưa ra phản hồi đúng trọng tâm. Điều này giúp các nhà phát triển tiết kiệm

được khoảng thời gian dùng để tự phân tích, thương lượng và phát triển các yêu cầu để đưa vào hợp đồng hệ thống. Nhờ đó mà các tính năng hữu ích được phát triển và chuyển giao sớm hơn.

2. Phân phối gia tăng (Incremental delivery): Phần mềm được phát triển theo từng bước với việc khách hàng chỉ định các yêu cầu đưa vào từng bước. Việc chia nhỏ dự án như vậy cho phép đội ngũ phát triển có thể tiến hành kiểm tra theo từng phần, xác định và sửa chữa vấn đề nhanh hơn, nhờ đó việc bàn giao công việc sẽ nhất quán và nhanh hơn.

3. Con người quan trọng hơn quy trình (People not process): Các kỹ năng của đội phát triển nên được công nhận và sử dụng. Các thành viên được tự do phát triển theo cách thức làm việc của họ mà không bị buộc tuân theo quy trình, điều này giúp đội ngũ bỏ qua thời gian thích ứng với quy trình chung để có thể tập trung vào việc phát triển và thử nghiệm phần mềm.

4. Nắm lấy thay đổi (Embrace change): Dự đoán các yêu cầu hệ thống thay đổi, từ đó thiết kế hệ thống thích ứng với những thay đổi này. Từng phần nhỏ của hệ thống đều được thiết kế để thích ứng với sự thay đổi sẽ giúp toàn bộ hệ thống linh hoạt hơn, có khả năng đáp ứng nhiều yêu cầu mới của khách hàng hơn nhưng không cần thay đổi quá nhiều mã và cấu trúc hệ thống ban đầu.

5. Duy trì sự đơn giản (Maintain simplicity): Tập trung vào sự đơn giản trong cả phần mềm đang phát triển và quá trình phát triển, đồng thời tích cực loại bỏ sự phức tạp khỏi hệ thống. Chia nhỏ những yêu cầu phức tạp thành những phần nhỏ hơn, đơn giản hóa trong cấu trúc phần mềm, thiết kế, code để dễ dàng xây dựng và bảo trì. Tìm ra những công việc không cần thiết phải làm và đề xuất giải pháp thay thế hoặc loại bỏ. Cách tiếp cận này không hạn chế việc mở rộng tính năng cho phần mềm, mà sẽ tập trung vào việc làm thế nào để mở rộng tính năng đó một cách đơn giản nhất.

3. **(Thư) -** Câu 3: Lập trình cực đoan (extreme programing) thể hiện yêu cầu của người dùng dưới dạng các câu chuyện (user story), với mỗi câu chuyện được viết trên một thẻ (card). Thảo luận về ưu điểm và nhược điểm của cách tiếp cận này đối với mô tả yêu cầu.

Câu chuyện của người dùng (user story) được coi là "phần giữ chỗ cho cuộc trò chuyện" (placeholder for conversation) – nó chỉ nhằm mục đích là mô tả ngắn gọn và súc tích về nhu cầu của người dùng và người ta mong đợi rằng (các) nhà phát triển chịu trách nhiệm triển khai câu chuyện của người dùng sẽ chi tiết hơn nữa dựa trên giao tiếp trực tiếp với người dùng.

Những ưu, nhược điểm của cách tiếp cận này đối với mô tả yêu cầu:

- **Ưu điểm:**

- ***Nó giúp nêu rõ các yêu cầu chính xác của người dùng***: Câu chuyện của người dùng chỉ thành công nếu chúng bao gồm các yếu tố chính như vai trò, khả năng, mong muốn và mục tiêu của người dùng trong một câu duy nhất. Việc có các yếu tố này được khớp nối rõ ràng làm cho chúng chính xác hơn nhiều đối với toàn bộ nhóm sản phẩm, đảm bảo rằng sự phát triển luôn tập trung vào nhu cầu của người dùng cuối.

- ***Có thể giảm thiểu rủi ro của dự án***: Về cốt lõi, câu chuyện của người dùng là một công cụ lấy người dùng làm trung tâm cho nhóm sản phẩm. Điều này giúp đảm bảo rằng các lỗi vượt quá tầm nhìn và phạm vi creep (Scope Creep – dự án trong trường hợp bị những yếu tố không kiểm soát được về mặt thời gian và nguồn lực từ những thay đổi nhỏ rồi thành những thay đổi lớn) được phát hiện sớm.

- **Nhược điểm:**

- ***Ít tập trung vào các yêu cầu phi chức năng***: Câu chuyện của người dùng chủ yếu tập trung vào việc nắm bắt các yêu cầu chức năng (tức là cách người dùng cuối (end-user) tương tác trực tiếp với hệ thống). Các yêu cầu phi chức năng như hiệu suất, khả năng chịu lỗi, khả năng sử dụng, độ bền, khả năng sửa đổi, … có thể không được nắm bắt rõ ràng.

- ***Khó mở rộng quy mô cho các dự án lớn hơn và phúc tạp hơn***: Định dạng của câu chuyện người dùng là các định dạng đơn giản tập trung vào một chức năng duy nhất. Những điều này đôi khi có thể khó mở rộng quy mô vì một dự án trở nên phúc tạp hơn.

- ***Đưa ra các yêu cầu mơ hồ hoặc không đầy đủ***: Vì câu chuyện của người dùng được viết bằng ngôn ngữ thân mật và tự nhiên hơn, các yêu cầu có thể quá mơ hồ để các kỹ sư có thể làm việc.

4. **(Trinh)** - Câu 4: Trong phát triển thử nghiệm đầu tiên, các thử nghiệm được viết trước mã. Giải thích cách bộ thử nghiệm có thể ảnh hưởng đến chất lượng của hệ thống phần mềm đang được phát triển.

Phát triển thử nghiệm đầu tiên (Test-first development): Framework kiểm tra đơn vị tự động (automated unit test) được sử dụng để viết các bài kiểm tra cho một phần chức năng mới trước khi chính chức năng đó được triển khai.

Phát triển thử nghiệm đầu tiên là một cách tiếp cận để phát triển theo đó các bài kiểm tra được viết trước khi mã được kiểm tra. Các thay đổi mã nhỏ được thực hiện và mã được cấu trúc lại cho đến khi tất cả các thử nghiệm thực thi thành công.

Cách bộ thử nghiệm có thể ảnh hưởng đến chất lượng của hệ thống phần mềm đang được phát triển là:

- Bộ thử nghiệm có thể trở nên lỗi thời vào thời điểm mã thực được viết và không phải kiểm tra mọi lỗ hổng mà mã có thể có.

- Lập trình viên có thể đi tắt đón đầu khi phát triển các bài kiểm tra để các bài kiểm tra hệ thống chưa hoàn thiện.

- Nếu các bài kiểm tra không được xem xét và các bài kiểm tra tiếp theo được viết sau khi phát triển, thì các lỗi chưa được phát hiện có thể được phân phối trong bản phát hành hệ thống.

- Rất khó để viết các bài kiểm tra, nó dẫn đến giao diện người dùng phức tạp làm giảm quy trình làm việc của chương trình trong một hệ thống.

- Phát triển thử nghiệm đầu tiên là khó khăn để công nhận tính toàn diện của một tập hợp các thử nghiệm. Trong khi những thử nghiệm này không thể bao phủ toàn bộ các trường hợp của hệ thống.

- Các bộ thử nghiệm cần cập nhật liên tục và tài liệu, nếu không chúng có thể nhanh chóng trở thành một mớ hỗn độn khổng lồ. Nếu bỏ qua việc bảo trì chung, kho lưu trữ thử nghiệm sẽ trở nên khó hiểu. Ngoài ra, các quy trình tài liệu sẽ bị bỏ rơi, có nghĩa là kiến thức thể chế về các bài kiểm tra sẽ bị mất.

- Việc tái cấu trúc thực hiện phá vỡ các kiểm tra hiện có.

- Viết một bài kiểm tra không thành công hoặc thậm chí không biên dịch; Viết mã xấu để làm cho bài kiểm tra vượt qua, mã xấu vi phạm các phương pháp hay nhất.

- Lập trình thử nghiệm đầu tiên với thiết kế mặt trước tối thiểu (Minimal up-front design) làm ảnh hưởng đến chất lượng thiết kế phần mềm bên trong.

5. **(Trường)** - Câu 5: Gợi ý 4 lý do tại sao tỷ lệ năng suất của các lập trình viên làm việc theo cặp có thể cao hơn một nửa so với tỷ lệ năng suất của hai lập trình viên làm riêng lẻ.

Dưới đây là một số lý do tỷ lệ năng suất của các lập trình viên làm việc theo cặp có thể cao hơn một nửa so với tỷ lệ năng suất của hai lập trình viên làm riêng lẻ:

1. ***Hai cái đầu vẫn tốt hơn một cái đầu.*** Nếu lập trình viên gặp vấn đề với code của mình thì sẽ có hai người sẽ giải quyết vấn đề.
2. ***Hiệu quả hơn.*** Suy nghĩ thông thường sẽ làm chậm tiến độ hoàn thành dự án bởi vì bạn đang đặt hiệu suất của hai lập trình viên để phát triển một chương trình duy nhất, thay vì để họ làm việc độc lập trên hai chương trình khác nhau. Các nghiên cứu đã chỉ ra rằng hai lập trình viên làm việc cùng một chương trình chỉ chậm hơn 15% so với làm việc độc lập, chứ không phải chậm hơn 50% như đã giả định trước. Hai lập trình viên có thể giải quyết vấn đề một cách nhanh hơn và hiệu quả hơn.
3. ***Code ít lỗi hơn.*** Bởi vì có một lập trình viên khác đang xem code của mình, do đó code của mình sẽ ít sai sót hơn. Trên thực tế, một nghiên cứu trước đó cho thấy rằng

nó dẫn đến ít lỗi hơn 15% so với code được viết bởi các lập trình viên độc lập. Cả hai lập trình viên phải đều tập trung tối đa vào từng công việc riêng của họ để có thể đạt hiệu quả cao nhất như việc người lập trình viên thứ nhất đang coding thì lập trình viên thứ hai phải tập trung vào xem đồng nghiệp của mình để tránh việc không biết partner mình đã làm tới đâu.

4. ***Một cách hiệu quả để chia sẻ kiến thức.*** Trong khi người cộng sự của mình coding thì mình có thể học hỏi rất nhiều từ người đối tác của mình ví dụ như code như thế nào cho clean hơn, cách tư duy, cách giải quyết vấn đề, ….
5. ***Ít rủi ro hơn cho dự án.*** Sẽ có hơn một người biết chương trình mới hoạt động như thế nào để chẳng may một trong hai người rời khỏi công ty thì nó sẽ không giết chết dự án.
6. ***Phát triển kỹ năng giao tiếp.*** Cộng tác trong một dự án sẽ tăng khả năng giao tiếp và làm việc nhóm cho các lập trình viên.

6. **(Vy)** - Câu 6: So sánh và đối chiếu cách tiếp cận Scrum để quản ý dự án với cách tiếp cận dựa trên kế hoạch thông thường được thảo luận trong chương 23. So sánh của bạn nên dựa trên hiệu quả của từng cách tiếp cận để lập kế hoạch phân bổ người cho các dự án, ước tính chi phí của dự án, duy trì sự gán kết của nhóm và quản lý các thay đổi về thành viên nhóm dự án.

	Cách tiếp cận Scrum	Cách tiếp cận dựa trên kế hoạch thông thường
Lập kế hoạch phân bổ người cho các dự án	Toàn bộ nhóm sẽ tham gia dự án và không có người quản lý. Việc lập kế hoạch có sự tham gia của tất cả các thành viên.	Người quản lý phân chia công việc thành các phần nhỏ sau đó giao cho các thành viên.
Ước tính chi phí dự án	Dự án được phát triển theo	Cần phải ước tính chi phí và

	giai đoạn nên dễ dàng ước tính chi phí và thời gian.	thời gian trước khi phát triển dự án.
Duy trì sự gán kết của nhóm	Các nhóm làm việc cùng nhau nên họ biết rõ tiến độ của công việc, bên cạnh đó họp sẽ được tổ chức thường xuyên để giải quyết các công việc chưa thể giải quyết.	Các nhóm làm việc ở những phần khác nhau nên vì thế các nhóm không biết tiến độ của các nhóm khác.
Thay đổi về thành viên nhóm dự án	Số lượng thành viên tham gia ít vì tất cả các thành viên đều làm việc cùng nhau.	Số lượng thành viên tham gia lớn nhằm đảm bảo các nhóm có đủ thành viên.

7. **(Thiện)** - Câu 7: Để giảm chi phí và tác động đến môi trường của việc đi lại, công ty của bạn quyết định đóng cửa một số văn phòng và hỗ trợ nhân viên làm việc tại nhà. Tuy nhiên, người quản lý cao cấp đưa ra chính sách này không biết rằng phần mềm được phát triển bằng Scrum. Giải thích cách bạn có thể sử dụng công nghệ để hỗ trợ Scrum trong môi trường phân tán để biến điều này thành khả thi. Bạn có thể gặp phải những vấn đề gì khi sử dụng phương pháp này?

Ta có thể ứng dụng công nghệ để hộ trợ Scrum trong môi trường phân tán để biến điều này khả thi:

- Hội họp qua các nền tảng hội họp trực tuyến như Google Meet, Microsoft Teams,... Các cuộc họp được tổ chức thuận tiện, dễ dàng hơn, hỗ trợ đầy đủ những công cụ cần thiết cho việc trình bày, lên kế hoạch.

- Trao đổi trực tiếp bằng các công cụ nhắn tin hoặc video call như Messenger, FaceTime,... Giúp cho việc trao đổi những công việc giữa các nhóm phát triển với nhau được diễn ra thường xuyên và trôi chảy

Những vấn đề có thể gặp phải:

- Việc hội họp thuận tiện bằng các công cụ trực tuyến dẫn đến việc các cuộc họp bất thường có xu hướng diễn ra thường xuyên hơn, ảnh hưởng đến lịch sinh hoạt của những nhân viên khác nhau trong một dự án.
- Chuyển sang hình thức hội họp trực tuyến là một sự đánh đổi sự phụ thuộc vào khoảng cách sang phụ thuộc vào trạng thái mạng của những cá nhân tham gia.
- Các không gian họp không đồng nhất giữa những thành viên, có thể gây ra sự không đồng bộ trong việc nắm bắt thông tin.

8. **(Hải)** - Câu 8: Tại sao cần phải giới thiệu các phương pháp và tài liệu từ hướng tiếp cận dựa trên kế hoạch khi mở rộng quy mô phương pháp Agile cho các dự án lớn đã phát triển bởi các đội phát triển phân tán?

Hướng tiếp cận của phương pháp agile là dựa trên kế hoạch (Plan-based): Đối với flow của SDLC (System development life cycle), phương pháp luận phát triển được sử dụng được gọi là phương pháp luận agile. Cách tiếp cận dựa trên kế hoạch là một phần của phương pháp agile.

Điểm để chỉ ra lý do bao gồm tài liệu và phương pháp cho các dự án lớn được phát triển bằng cách tiếp cận dựa trên kế hoạch trong môi trường phát triển phân tán như được đưa ra dưới đây:

- Nếu tài liệu không được cung cấp, thì các thành viên trong nhóm sẽ rất khó hiểu và khó bắt kịp nhịp độ của toàn bộ dự án.
- Để ghi lại các vấn đề tiến độ đã xác định thu được trong quá trình giám sát chương trình scrum, cần có tài liệu và phương pháp phát triển các dự án lớn.
- Các vấn đề về học tập, lập kế hoạch và ngôn ngữ có thể được xác định một cách dễ dàng.

- Tự động hóa thử nghiệm với sự kết hợp liên tục có thể được theo sau bởi nhóm scrum để phát triển dự án lớn.

Lập kế hoạch dự án:

- Cần thiết khi phát triển phần mềm với các nhóm lớn hơn để đảm bảo có đúng người khi cần thiết.
- Đảm bảo lịch trình giao hàng được điều chỉnh

Phân tích yêu cầu:

- Điều quan trọng là quyết định cách phân phối công việc giữa các nhóm và đảm bảo mỗi nhóm có một số hiểu biết về công việc của các nhóm khác.

Tài liệu thiết kế:

- Nhập để các nhóm phát triển độc lập với quyền truy cập vào phần mềm đang được phát triển.

Quản lý rủi ro:

- Yêu cầu để đảm bảo tất cả các nhóm hiểu được rủi ro phải đối mặt và tổ chức công việc của họ để giảm thiểu những rủi ro này.
- Hữu ích để đối phó với các lịch trình giao hàng khác nhau.

9. **(Bích)** - Câu 9: Giải thích tại sao phương pháp Agile có thể không làm việc/đáp ứng tốt tại các tổ chức có các đội nhóm/con người với nhiều kỹ năng và khả năng khác nhau nhưng tất cả đều làm tốt, đồng thời các quy trình nghiệp vụ cũng đã được thiết lập tốt.

- Tất cả các tổ chức lớn thường có các thủ tục và tiêu chuẩn chất lượng mà tất cả các dự án phải tuân theo và do tính chất quan liêu của chúng, những tiêu chuẩn này có thể không tương thích với các phương pháp Agile.
- ***Yêu cầu quá phức tạp - nhiều dự án rất phức tạp***. Sự phức tạp đó có thể ảnh hưởng trực tiếp đến sự thành công hay thất bại của nó. Với Agile, chìa khóa là chia nhỏ một dự án phức tạp thành các dự án nhỏ hơn và sau đó điều chỉnh

chúng theo thời gian. Các doanh nghiệp lớn với đội ngũ lớn thường gặp khó khăn khi làm điều này, dẫn đến thất bại Agile.

- ***Thiếu kinh nghiệm Với Agile*** - có lẽ lý do lớn nhất khiến các doanh nghiệp và công ty lớn thất bại với Agile là do thiếu kinh nghiệm. Trên thực tế, theo một nghiên cứu từ VersionOne , lý do số một khiến các dự án Agile thất bại (được trích dẫn bởi 44% số người được hỏi) là thiếu kinh nghiệm trong việc triển khai và tích hợp phương pháp Agile. Việc triển khai Agile không thể được thực hiện theo ý thích. Nó cần kinh nghiệm và sự hiểu biết. Nhiều doanh nghiệp lớn không có người có kinh nghiệm về Agile.
- ***Cố gắng giữ vững truyền thống*** - không ai thích thay đổi. Thật khó để từ bỏ vùng an toàn của chúng ta. Một trong những nguyên nhân lớn nhất dẫn đến thất bại Agile là cố gắng bám chặt vào những cách làm truyền thống. Khi bạn chuyển sang Agile, bạn phải cam kết với nó. Bạn không thể cố giữ cách làm cũ. Cố gắng chọn và chọn các quy trình hoặc công cụ kế thừa nhất định và điều chỉnh chúng vào các quy trình Agile sẽ không hiệu quả.

10. **(Hải)** - Câu 10: Một trong những vấn đề của người dùng/khách hàng khi tham gia chặt chẽ với nhóm phát triển là họ bị "đồng hóa" (go native). Có nghĩa là, họ chấp nhận triển vọng của nhóm phát triển và không quan tâm đến nhu cầu của đồng nghiệp là người dùng như họ. Đề xuất ba cách bạn có thể tránh được vấn đề này, và thảo luận về những lợi thế và bất lợi của mỗi cách tiếp cận.

Xác thực các đề xuất do người dùng đưa ra với những người dùng khác nhau: Các đề xuất do người dùng đưa ra phải được xác thực với các đề xuất do bất kỳ người dùng khác đưa ra.

Lợi thế:

- Bằng cách thảo luận về đề xuất của người dùng giúp kiểm tra các đề xuất của người dùng một cách độc lập.

Bất lợi:

- Nếu các đề xuất được thảo luận một cách độc lập thì nó sẽ làm chậm quá trình của phần mềm.
- Nếu dự án bị trì hoãn thì sẽ dẫn đến chi phí tăng thêm cho dự án.

***Nhận đề xuất từ nhiều người dùng*:** Nhiều người dùng phải tham gia vào nhóm phát triển để nhận được đề xuất.

Thuận lợi:

- Bằng nhiều đề xuất, có thể thu được nhiều góc nhìn từ những người dùng khác nhau cho vấn đề cụ thể.
- Xác suất có bất kỳ người dùng điển hình nào sẽ giảm xuống.

Bất lợi:

- Có thể có khả năng tương quan về các đề xuất từ những người dùng khác nhau.
- Nếu mối tương quan tăng lên, thì việc hoàn thành dự án sẽ tăng thêm chi phí.

***Thay đổi thành viên từ nhóm*:** Nếu bất kỳ đề xuất nào của người dùng tạo ra vấn đề trong nhóm. Sau đó, thay đổi người dùng từ nhóm.

Thuận lợi:

- Bằng nhiều gợi ý, có thể thu được nhiều góc nhìn từ những người dùng khác nhau.
- Bằng cách thảo luận về đề xuất của người dùng giúp kiểm tra các đề xuất của người dùng một cách độc lập.

Bất lợi:

- Có thể người dùng mới không biết về hệ thống. Vì vậy, anh ấy / cô ấy sẽ mất thời gian để hiểu hệ thống để đưa ra gợi ý.
- Kiểm tra các đề xuất của người dùng một cách độc lập dẫn đến tăng thời gian năng suất của dự án.
- Nếu dự án bị trì hoãn thì sẽ dẫn đến chi phí tăng thêm cho dự án.

----------**Hết**----------