[LỜI MỞ ĐẦU 4](#_30j0zll)

[DANH MỤC HÌNH ẢNH 5](#_1fob9te)

[DANH MỤC BẢNG 6](#_3znysh7)

[DANH MỤC TỪ VIẾT TẮT 6](#_2et92p0)

[PHẦN I. MỞ ĐẦU 7](#_tyjcwt)

[1.](#_3dy6vkm) Tên đề tài 7

[2.](#_1t3h5sf) Lý do chọn đề tài 7

[3.](#_2s8eyo1) Mục đích nghiên cứu 7

[4.](#_17dp8vu) Bố cục đề tài 7

[5.](#_3rdcrjn) Phương pháp nghiên cứu 8

[PHẦN II. NỘI DUNG 9](#_26in1rg)

[CHƯƠNG I. TỔNG QUAN VỀ PHẦN MỀM 9](#_lnxbz9)

[I.1 Phần mềm và chất lượng phần mềm 9](#_35nkun2)

[I.2 Lỗi phần mềm 10](#_z337ya)

[I.3 Định nghĩa chất lượng phần mềm và đảm bảo chất lượng phần mềm 13](#_1ci93xb)

[I.4 Vòng đời phần mềm 14](#_qsh70q)

[I.5 Tổng kết chương I. 16](#_3as4poj)

[CHƯƠNG II. TỔNG QUAN VỀ KIỂM THỬ PHẦN MỀM 17](#_1pxezwc)

[II.1 Định nghĩa về kiểm thử phần mềm 17](#_49x2ik5)

[II.2 Mục đích của kiểm thử: 18](#_2p2csry)

[II.3 Các thuật ngữ và định nghĩa cơ bản về kiểm thử 18](#_147n2zr)

[II.4 Các kỹ thuật kiểm thử phần mềm 21](#_3o7alnk)

[II.5 Các mức kiểm thử 22](#_1hmsyys)

[II.6 Quy trình kiểm thử 23](#_41mghml)

[II.7 Nguyên tắc kiểm thử phần mềm 25](#_2grqrue)

[CHƯƠNG III KIỂM THỬ TỰ ĐỘNG VÀ CÔNG CỤ HỖ TRỢ 28](#_vx1227)

[III.1 Tổng quan về kiểm thử tự động 28](#_3fwokq0)

[III.2 Kiến trúc của một bộ kiểm thử tự động 28](#_1v1yuxt)

[III.3 Ưu điểm, nhược điểm của kiểm thử tự động 30](#_2u6wntf)

[III.4 Một số công cụ kiểm thử tự động 31](#_19c6y18)

[III.5 Tổng kết kiểm thử tự đông 32](#_3tbugp1)

[CHƯƠNG IV. CÔNG CỤ KIỂM THỬ PHẦN MỀM TỰ ĐỘNG CYPRESS VÀ ỨNG DỤNG 33](#_28h4qwu)

[IV.1 Tìm hiểu công cụ kiểm thử tự động Cypress 33](#_nmf14n)

[IV.2 Các trình duyệt hỗ trợ 35](#_37m2jsg)

[IV.3 Đặc điểm 36](#_46r0co2)

[IV.4 Ngôn ngữ sử dụng viết Script 36](#_2lwamvv)

[IV.5 Cài đặt và sử dụng Cypress 37](#_111kx3o)

[IV.6 Ứng dụng 42](#_206ipza)

[IV.6.1 Bài toán 42](#_4k668n3)

[IV.6.2 Lập kế hoạch kiểm thử 43](#_3ygebqi)

[IV.6.3 Kiểm thử đơn vị: 57](#_3mzq4wv)

[IV.6.4 Kiểm thử hệ thống 71](#_2250f4o)

[PHẦN III. KẾT LUẬN 90](#_40ew0vw)

[TÀI LIỆU THAM KHẢO 91](#_2fk6b3p)

# LỜI MỞ ĐẦU

Xã hội ngày nay đang phát triển một cách nhanh chóng, đặc biệt dường như công nghệ thông tin đã và đang chiếm một phần quan trọng trong cuộc sống mỗi chúng ta. Nếu như trước đây phần mềm máy tính chỉ được sử dụng để tính toán khoa học kỹ thuật và xử lý dữ liệu thì ngày nay nó đã được ứng dụng vào mọi mặt của của đời sống hàng ngày của con người. Vì thế con người ngày càng phụ thuộc chặt chẽ vào các sản phẩm phần mềm và do vậy đòi hỏi về chất lượng của các sản phẩm phần mềm ngày càng cao, tức là các phần mềm phải được sản xuất với giá thành thấp hơn, dễ dàng sử dụng, an toàn và tin cậy được. Vì vậy kiểm thử có phương pháp là một hoạt động không thể thiếu trong quy trình sản xuất phần mềm để đảm bảo các yếu tố chất lượng nêu trên của các sản phẩm phần mềm.

Trên thực tế cho thấy, việc kiểm thử phần mềm là giai đoạn chiếm đến hơn 40% thời gian, kinh phí và nhân lực trong quá trình phát triển các dự án phần mềm. Vậy làm thế nào để có thể kiểm tra dự án phần mềm của ta đang chạy ổn định, đạt được tính hiệu quả cao, nhưng lại tiết kiệm được thời gian cũng như kinh phí trong quá trình kiểm thử là một điều thiết yếu đối với các nhà kiểm thử. Tăng năng suất kiểm thử là một nhu cầu thiết yếu để tăng chất lượng phần mềm. Vì thế nghiên cứu để phát triển các kỹ thuật, công cụ kiểm thử hữu hiệu và đào tạo đội ngũ kiểm thử có kỹ năng và kinh nghiệm là các đóng góp thiết thực nhất để tăng chất lượng của các sản phẩm phần mềm.

Để hoàn thiện được báo cáo đồ án tốt nghiệp này, em xin gửi lời cám ơn chân thành đến Trường Đại học Công Nghệ- Đại học Quốc Gia Hà Nội, Khoa Công nghệ thông tin đã tạo điều kiện cho em trong suốt quá trình học tập, và đặc biệt em xin chân thành cảm ơn đến **PGS-TS Trương Anh Hoàng**, đã tận tình trực tiếp hướng dẫn em hoàn thành báo cáo này.

Mặc dù đã cố gắng để hoàn thiện đề tài tốt nhất có thể, nhưng chắc chắn không thể tránh khỏi những thiếu sót. Em rất mong nhận được sự góp ý của các thầy cô và các bạn để báo cáo đồ án tôt nghiệp này của em được hoàn thiện hơn.

*Em xin chân thành cảm ơn!*

# DANH MỤC HÌNH ẢNH

# DANH MỤC BẢNG

# DANH MỤC TỪ VIẾT TẮT

|  |  |  |
| --- | --- | --- |
| Thuật ngữ/ Từ viết tắt | Thuật ngữ Tiếng Anh | Ý nghĩa |
| **IEEE** | Institute of Electrical and Electronics Engineers | Viện kỹ nghệ Điện và Điện tử |
| **UI** | User Interface | Giao diện người dùng |
| **PM** | Project Manger | Quản lý dự án |
| **TL** | Test Lead | Trưởng nhóm kiểm thử |

# PHẦN I. MỞ ĐẦU

### Tên đề tài

***Tự động hóa các ca kiểm thử cơ bản của ứng dụng web bằng cypress***

### Lý do chọn đề tài

Ngày nay, việc tự động hóa đang được nghiên cứu và ứng dụng đưa vào nhiều lĩnh vực, trong đó công nghệ phần mềm nói chung và kiểm thử phần mềm nói riêng. Khi mà kiểm thử phần mềm vẫn tiêu tốn một lượng lớn thời gian, kinh phí và nhân lực trong một dự án phần mềm thì song song với kiểm thử truyền thống là kiểm thử thủ công, sự ra đời của các công cụ hỗ trợ kiểm thử tự động như Selenium, Quick Test Professional, Nunit, Junit, Load Runner (thường dùng trong kiểm thử hiệu năng) là tất yếu. Trong các công cụ kiểm thử tự động,Cypress là một công cụ kiểm thử tự động mới. Cypress được biết đến là một công cụ kiểm thử tự động các ứng dụng Web, sử dụng ngôn ngữ JavaScript hiện đại và đặc biệt công cụ này là một bộ mã nguồn mở, do đó các tổ chức không cần tốn kinh phí mua bản quyền.

Với mong muốn có cái nhìn chính xác, rõ ràng hơn về quy trình kiểm thử phần mềm, đảm bảo chất lượng phần mềm và tiếp cận với công cụ kiểm thử tự động Cypress, đồng thời rèn kỹ năng làm việc tạo tiền đề định hướng cho tương lai sau khi ra trường nên em chọn đề tài “***Tự động hóa các ca kiểm thử cơ bản của ứng dụng web.***”

### Mục đích nghiên cứu

Đề tài tìm hiểu cơ sở lý thuyết về kiểm thử, kiểm thử tự động và cách sử dụng công cụ Cypress.io để đảm bảo chất lượng phần mềm.

### Bố cục đề tài

Những nội dung và kết quả tìm hiểu của đề tài được trình bày trong 3 chương như sau:

- Chương I: Tổng quan về phần mềm: Chương này trình bày về các khái niệm, vấn đề cơ bản về phần mềm, vòng đời,…

- Chương II: Tổng quan về kiểm thử phần mềm: Chương này trình bày được những định nghĩa và những vấn đề cơ bản về kiểm thử phần mềm, lỗi phần mềm.

- Chương III: Kiểm thử tự động và công cụ hỗ trợ: Chương này trình bày các vấn đề cơ bản của kiểm thử tự động và một số công cụ hỗ trợ kiểm thử phần mềm.

- Chương IV: Công cụ kiểm thử tự động Cypress và ứng dụng: Chương này trình bày các thành phần của công cụ Cypress, nắm được cách sử dụng và vận hành của nó trên một ứng dụng cụ thể.

### Phương pháp nghiên cứu

***Phương pháp nghiên cứu lý thuyết:***

* Tiến hành thu thập và nghiên cứu các tài liệu có liên quan đến đề tài.
* Tổng hợp các tài liệu.
* Chọn lọc các tài liệu nghiên cứu để báo cáo đề tài khóa luận tốt nghiệp.

***Phương pháp nghiên cứu thực nghiệm:***

* Tìm hiểu cách thức, tiêu chí kiểm thử chức năng.
* Ứng dụng công cụ Cypress vào kiểm thử phần mềm.
* Kiểm tra và đánh giá kết quả đạt được.

# PHẦN II. NỘI DUNG

# CHƯƠNG I. TỔNG QUAN VỀ PHẦN MỀM

### I.1 Phần mềm và chất lượng phần mềm

#### Khái niệm phần mềm

* **Theo định nghĩa IEEE:** Phần mềm là các chương trình máy tính, các thủ tục, các tài liệu có liên quan và các dữ liệu để vận hành của một hệ thống máy tính.
* **Theo định nghĩa ISO**, phần mềm gồm 4 thành phần:
  + - * **Chương trình máy tính (mã nguồn):** Thành phần này giúp cho máy tính thực thi các ứng dụng được yêu cầu.
      * **Các thủ tục:** Các thủ tục được yêu cầu, để định nghĩa thứ tự và lịch trình mà chương trình sẽ thực thi, các hàm triển khai, người thực thi các hành động cần thiết cho ứng dụng phần mềm.
      * **Các tài liệu:** Có rất nhiều những tài liệu cần thiết với nhân viên phát triển, người sử dụng và nhân viên bảo trì như: tài liệu thiết kế, tài liệu phân tích, tài liệu hướng dẫn sử dụng, tài liệu hướng dẫn bảo trì.
      * **Dữ liệu cần cho việc vận hành hệ thống:** Dữ liệu bao gồm các biến, mã nguồn, danh sách tên thích ứng phần mềm với yêu cầu xác định của khách hàng để vận hành phần mềm.

#### I.1.1 Định nghĩa công nghệ phần mềm

**Công nghệ phần mềm** là việc áp dụng các công cụ, kỹ thuật một cách hệ thống trong việc phát triển các ứng dụng dựa trên máy tính. Nói cách khác đó là việc áp dụng các quan điểm, các tiến trình có kỷ luật và có bài bản vào để phát triển, vận hành và bảo trì phần mềm.

* **Theo Fritz Bauer**: Công nghệ phần mềm là quá trình sử dụng các nguyên tắc kỹ thuật có cơ sở để xây dựng các phần mềm 1 cách kinh tế để hoạt động tin cậy và chạy hiệu quả trên hệ thống máy thật.
* **Theo IEEE**: Việc áp dụng cách nghiên cứu và tiếp cận có hệ thống, quy tắc, có thể xác định số lượng để phát triển, vận hành và bảo trì phần mềm, nghĩa là áp dụng kỹ thuật vào phần mềm.

Mục tiêu của công nghệ phần mềm là tạo ra những phần mềm tốt, giảm đến mức tối thiểu sai sót xảy ra trong quá trình vận hành, cũng như tạo điều kiện thuận lợi trong việc bảo trì và nâng cấp phần mềm.

#### I.1.2 Các đặc trưng của phần mềm

* Phần mềm được kỷ nghệ, không được chế tạo theo nghĩa cổ điển:
  + - * Phần mềm được thiết kế, chế tạo như các loại sản phẩm công nghiệp khác, nhưng không được định hình trước.
      * Quá trình phát triển phần mềm quyết định giá thành và chất lượng của nó.
      * Các phần mềm chỉ thực sự được tìm ra lỗi trong các pha phát triển.
* Có tính phức tạp cao và luôn thay đổi:
  + - * Phần mềm là 1 hệ thống logic với nhiều khái niệm và các mối liên hệ logic khác nhau. Với mỗi vòng lặp với 1 giá trị khác nhau là cơ hội để tìm ra lỗi phần mềm.
      * Thay đổi theo nhu cầu người dùng.
      * Thay đổi để đáp ứng môi trường vận hành.
* Phần mềm không nhìn thấy được:
  + - * Phần mềm không nhìn thấy được mà chỉ có thể nhận biết qua sự mô tả từ những khía cạnh khác nhau (sơ đồ điều khiển, mô hình luồng dữ liệu, mô hình tương tác…)
* Do đặc trưng này nên khả năng tìm lỗi 1 cách nhanh chóng là không thể.

### I.2 Lỗi phần mềm

#### Định nghĩa lỗi phần mềm và phân loại lỗi phần mềm

* Định nghĩa lỗi phần mềm: Có rất nhiều định nghĩa về lỗi phần mềm nhưng có thể hiểu và phát biểu một cách đơn giản thì "Lỗi phần mềm là sự không khớp giữa chương trình và đặc tả của nó".
* Dựa vào định nghĩa, ta có thể phân loại lỗi phần mềm thành 3 dạng:
  + - * Lỗi sai: Sản phẩm phần mềm được xây dựng khác với đặc tả.
      * Lỗi thiếu: Các yêu cầu của sản phẩm phần mềm đã có trong đặc tả nhưng lại không có trong sản phẩm thực tế.
      * Lỗi thừa: Sản phẩm thực tế có những tính năng không có trong tài liệu đặc tả.

#### Các nguyên nhân gây lỗi phần mềm

Lỗi phần mềm có thể đến từ nhiều nguyên nhân khác nhau, trong đó có cả các nguyên nhân chủ quan và các nguyên nhân khách quan. Dưới đây là chín nguyên nhân chủ yếu gây ra lỗi phần mềm:

* **Định nghĩa sai yêu cầu khách hàng**: Những lỗi trong việc xác định yêu cầu thường nằm ở phía khách hàng. Một số lỗi thường gặp là: định nghĩa sai yêu cầu, lỗi không hoàn chỉnh, thiếu các yêu cầu quan trọng hoặc là quá chú trọng các yêu cầu không thật sự cần.
* **Thất bại trong việc giao tiếp giữa khách hàng và nhà phát triển**: Hiểu lầm trong giao tiếp giữa khách hàng và nhà phát triển cũng là nguyên nhân gây lỗi. Những lỗi này thường xuất hiện trong giai đoạn đầu của dự án. Một số lỗi hay gặp phải: hiểu sai chỉ dẫn trong tài liệu yêu cầu, hiểu sai thay đổi khi khách hàng trình bày bằng lời nói và tài liệu, hiểu sai về phản hồi và thiếu quan tâm đến những đề cập của khách hàng.

Giải pháp khắc phục: Cần có ủy ban liên kết giữa khách hàng và nhà cung cấp để tránh lỗi trong giao tiếp. Ủy ban do quản trị dự án đứng đầu và khách hàng phải giới thiệu những người hiểu về mặt nghiệp vụ vào ủy ban đó.

* **Sai lệch có chủ ý với các yêu cầu phần mềm**: Trong một số trường hợp các nhà phát triển cố tình làm sai lệnh các yêu cầu trong tài liệu đặc tả. Nguyên nhân của việc này đến từ các áp lực thời gian, ngân sách, hay cố tình sử dụng lại các module từ các dự án trước mà chưa phân tích đầy đủ những thay đổi để thích nghi với các yêu cầu mới.

Giải pháp khắc phục: Dựa trên những thống kê để quyết định xem giải pháp như thế nào, sắp xếp ưu tiên xem bỏ được yêu cầu nào hay sử dụng lại được module nào.

* **Các lỗi thiết kế logic**: Lỗi phần mềm xảy ra trong quá trình các chuyên gia thiết kế hệ thống, các kiến trúc sư hệ thống, kỹ sư phần mềm, các nhà phân tích xây dựng phần mềm theo yêu cầu. Các lỗi điển hình bao gồm:
  + - * Định nghĩa các yêu cầu phần mềm bằng các thuật toán sai.
      * Quy trình định nghĩa có chứa trình tự lỗi.
      * Sai sót trong các định nghĩa biên như > 3 hay ≥ 3.
* **Các lỗi lập trình**: Có rất nhiều lý do dẫn đến việc các lập trình viên gây ra các lỗi lập trình. Những lý do này bao gồm: sự hiểu sai các tài liệu thiết kế, ngôn ngữ; sai sót trong ngôn ngữ lập trình; sai sót trong việc áp dụng các công cụ phát triển; sai sót trong lựa chọn dữ liệu...
* **Không tuân thủ theo các tài liệu hướng dẫn và tiêu chuẩn lập trình**: Các lỗi phần mềm có thể đến từ việc không tuân thủ các tài liệu và tiêu chuẩn lập trình của các tổ chức phát triển phần mềm.
* **Thiếu sót trong quá trình kiểm thử**: Lỗi phần mềm có thể đến từ chính quá trình kiểm thử khi mà người kiểm thử để lọt lỗi.

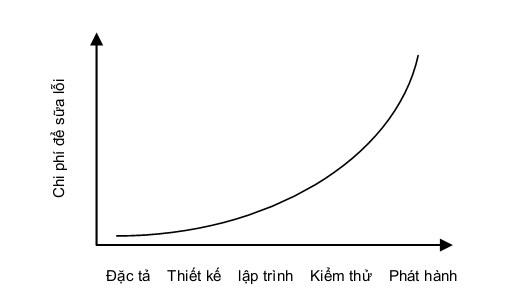
Những lỗi này đến từ các nguyên nhân sau đây:

* Kế hoạch kiểm thử chưa hoàn chỉnh, để sót yêu cầu cần kiểm thử.
* Lỗi trong tài liệu và báo cáo kiểm thử.
* Việc sửa chữa các lỗi được phát hiện không hoàn chỉnh do áp lực thời gian hay do thiếu cẩn thận.
* **Các lỗi thủ tục**: Các thủ tục hướng dẫn cho người sử dụng tại từng bước của tiến trình. Chúng có tầm quan trọng đặc biệt trong các hệ thống phần mềm phức tạp mà các tiến trình được thực bằng nhiều bước, mỗi bước có thể có nhiều kiểu dữ liệu và cho phép kiểm tra các kết quả trung gian. Các lỗi có thể đến từ việc viết các thủ tục.
* **Các lỗi về tài liệu**: Các lỗi về tài liệu là vấn đề của các đội phát triển và bảo trì khi có những sai sót trong các tài liệu liên quan. Những lỗi này có thể là nguyên nhân gây ra lỗi trong giai đoạn phát triển kế tiếp và giai đoạn bảo trì.

#### Chi phí cho việc sửa lỗi phần mềm

Việc kiểm thử và sửa lỗi phần mềm có thể thực hiện trong bất cứ giai đoạn nào của vòng đời phần mềm. Tuy nhiên công việc này nên được thực hiện càng sớm càng tốt vì càng về giai đoạn sau của vòng đời phần mềm, chi phí cho việc tìm và sửa lỗi càng tăng, đặc biệt là đến giai đoạn đã triển khai phần mềm thì chi phí cho sửa lỗi sẽ trở nên rất lớn và ảnh hưởng trực tiếp đến uy tín của tổ chức phát triển phần mềm.

Theo tài liệu của Boehm, chi phí cho việc tìm và sửa lỗi phần mềm sẽ tăng theo hàm mũ trong biểu đồ sau:



*Hình 1.1 Chi phí cho việc sửa lỗi phần mềm*

### I.3 Định nghĩa chất lượng phần mềm và đảm bảo chất lượng phần mềm

#### Định nghĩa chất lượng phần mềm

Có rất nhiều định nghĩa về chất lượng phần mềm được đưa ra bởi các tổ chức, cá nhân khác nhau. Trong phạm vi của đồ án này trình bày một số định nghĩa tiêu biểu:

- **Định nghĩa theo IEEE (1991):**

* **Định nghĩa 1:** Chất lượng phần mềm là một mức độ mà một hệ thống, thành phần hệ thống hay tiến trình đáp ứng được yêu cầu đã được đặc tả.
* **Định nghĩa 2:** Chất lượng phần mềm là mức độ mà một hệ thống, thành phần hệ thống hay tiến trình đáp ứng được yêu cầu và sự mong đợi của khách hàng hay người sử dụng.

- **Phân tích hai quan điểm của IEEE:**

* Theo quan điểm thứ nhất của IEEE: Nếu theo quan điểm này, chúng ta sẽ bị phụ thuộc quá nhiều vào tài liệu đặc tả yêu cầu, dẫn đến nếu việc xác định yêu cầu bị sai, thiếu thì một phần mềm được làm đúng với đặc tả chưa chắc đã là một phần mềm có chất lượng.
* Theo quan điểm thứ hai của IEEE: Khách hàng đôi khi không có nhiều kiến thức về công nghệ, họ có thể đưa ra những mong muốn hết sức vô lý và có thể thay đổi yêu cầu với phần mềm nhiều lần thậm chí thay đổi ngay trong giai đoạn cuối. Điều này gây rất nhiều khó khăn cho việc phát triển phần mềm.
* **Định nghĩa theo Pressman**: Chất lượng phần mềm là sự phù hợp của các yêu cầu cụ thể về hiệu năng và chức năng, các tiêu chuẩn phát triển phần mềm được ghi lại rõ ràng bằng tài liệu với các đặc tính ngầm định của tất cả các phần mềm được phát triển chuyên nghiệp.
* Định nghĩa của Pressman đề xuất ba yêu cầu với chất lượng phần mềm phải được đáp ứng khi phát triển phần mềm:
* Các yêu cầu chức năng rõ ràng là nhân tố chính quyết định chất lượng đầu ra của phần mềm.
* Các tiêu chuẩn chất lượng phần mềm sẽ được nói đến trong hợp đồng.

Các đặc tính ngầm định cần được đáp ứng trong quá trình phát triển cho dù không được nói đến rõ ràng trong hợp đồng.

#### Định nghĩa đảm bảo chất lượng phần mềm

* **Định nghĩa theo Daniel Galin**: Đảm bảo chất lượng phần mềm (Software Quality Assure) là một tập hợp các hành động cần thiết được lên kế hoạch một cách hệ thống để cung cấp đầy đủ niềm tin rằng quá trình phát triển phần mềm phù hợp để thành lập các yêu cầu chức năng kỹ thuật cũng như các yêu cầu quản lý theo lịch trình và hoạt động trong giới hạn ngân sách.

### I.4 Vòng đời phần mềm

Vòng đời phần mềm là một loạt các pha mà phần mềm phải trải qua từ việc khám phá các khái niệm đến khi phần mềm bị loại bỏ hoàn toàn. Mô hình vòng đời phần mềm gồm 7 pha:

* **Pha yêu cầu** (requirements phase): Là pha đầu tiên trong quá trình xây dựng phần mềm, còn gọi là pha tìm hiểu khái niệm (concept exploration). Ở pha này, đại diện nhóm phát triển và khách hàng gặp nhau, khách hàng nêu ra những yêu cầu mà phần mềm phải có, đại diện nhóm phát triển ghi chép lại. Nếu dịch theo tiếng Anh "requirements phase" là pha yêu cầu. Tuy nhiên cách gọi này có thể hơi khó hiểu, do đó người ta thường gọi là pha xác định yêu cầu.
* **Pha đặc tả** (specification phase): Pha này phân tích các yêu cầu của khách hàng. Mô tả các kết quả phân tích dưới dạng "tài liệu đặc tả". Cuối giai đoạn này, kế hoạch quản lý dự án phần mềm được đưa ra, mô tả chi tiết quá trình phát triển phần mềm. Câu hỏi mà pha này cần trả lời là: "Phần mềm sẽ làm gì?".
* **Pha thiết kế** (design phase): Là pha tiếp theo pha đặc tả. Căn cứ vào tài liệu đặc tả, pha này mô tả cách thức mà phần mềm thực hiện các công việc cụ thể. Pha này trả lời câu hỏi "phần mềm sẽ được làm như thế nào?". Pha thiết kế thường có hai phần: thiết kế kiến trúc (architecture design) và thiết kế chi tiết (detail design).Thiết kế kiến trúc phân chia phần mềm thành các thành phần gọi là module. Thiết kế chi tiết là thiết kế từng module một cách chi tiết.
* **Pha cài đặt** (implementation phase): Ở pha này, người ta thực hiện viết chương trình bằng một ngôn ngữ lập trình cụ thể.
* **Pha tích hợp** (integration phase): Kết nối các module đã viết của chương trình thành một phần mềm thống nhất và chạy thử, chỉnh sửa cho đến khi phần mềm chạy tốt.
* **Pha bảo trì** (maintenance phase): Khi chương trình đã được cài đặt trên máy của khách hàng vẫn có thể có lỗi phát sinh trong phần mềm cần loại trừ và điều chỉnh lại theo yêu cầu khách hàng. Công việc bảo trì được chia thành hai loại: bảo trì sửa lỗi (software repair) và bảo trì cập nhật(software update). Bảo trì sửa lỗi là sửa các lỗi có thể vẫn còn xuất hiện trong khi chạy chương trình. Bảo trì cập nhật lại được chia làm hai loại: bảo trì hoàn thiện (perfective maintenance) và bảo trì thích nghi (adaptive maintenance). Bảo trì hoàn thiện là sửa đổi phần mềm theo ý khách hàng để nâng cao hiệu quả. Bảo trì thích nghi là sửa đổi để phần mềm thích nghi với môi trường mới. Người ta tính toán rằng bảo trì sửa lỗi và thích nghi chiếm thời gian gần bằng nhau và bằng khoảng 20% thời gian bảo trì, còn bảo trì hoàn thiện chiếm thời gian khoảng gấp ba lần mỗi loại bảo trì kia (khoảng 60%).

**Pha loại bỏ** (retirement phase): Đến một thời điển nào đó, do sự thay đổi của môi trường làm việc và một số yếu tố khác, chi phí bảo trì phần mềm sẽ lớn hơn rất nhiều so với chi phí phát triển một phần mềm mới. Đó là lúc việc loại bỏ phần mềm được thực hiện.

### I.5 Tổng kết chương I.

Chương 1 của đồ án đã trình bày được những định nghĩa và những vấn đề cơ bản xung quanh phần mềm, công nghệ phần mềm, lỗi phần mềm và xử lý lỗi phần mềm. Các vấn đề cụ thể bao gồm:

* Các định nghĩa về phần mềm, công nghệ phần mềm, chất lượng phần mềm, bảo đảm chất lượng phần mềm, lỗi phần mềm.

Các vấn đề liên quan đến lỗi phần mềm và qui trình xử lý lỗi phần mềm.

# CHƯƠNG II. TỔNG QUAN VỀ KIỂM THỬ PHẦN MỀM

Kiểm thử nhằm đánh giá chất lượng hoặc tính chấp nhận được của sản phẩm. Kiểm thử cũng nhằm phát hiện lỗi hoặc bất cứ vấn đề gì về sản phẩm. Chúng ta cần kiểm thử vì biết rằng con người luôn có thể mắc sai lầm. Điều này đặc biệt đúng trong lĩnh vực phát triển phần mềm và các hệ thống điều khiển bởi phần mềm. Chương này nhằm phác họa một bức tranh tổng thể về kiểm thử phần mềm.

## II.1 Định nghĩa về kiểm thử phần mềm

Kiểm thử phần mềm có nhiều định nghĩa khác nhau đề xuất bởi nhiều tổ chức hay cá nhân khác nhau. Sau đây là một số định nghĩa về kiểm thử phần mềm.

**- Theo GlenFord Myers: “**Kiểm thử là quá trình vận hành một chương trình để tìm ra lỗi”. Theo như định nghĩa này, quá trình kiểm thử bao gồm tất cả các hoạt động từ kiểm tra mã nguồn được thực hiện bởi trưởng nhóm phát triển, đến việc chạy thử chương trình được tiến hành bởi các đồng nghiệp khác. Tất cả các hoạt động trên đều được coi là các hoạt động kiểm thử.

- **Theo IEEE, Kiểm thử là:**

**Định nghĩa 1:** Quá trình vận hành một hệ thống hoặc thành phần của hệ thống dưới những điều kiện xác định, quan sát hoặc ghi nhận kết quả và đưa ra đánh giá về hệ thống hoặc thành phần hệ thống.

**Định nghĩa 2:** Là quá trình phân tích phần mềm để tìm ra sự khác biệt giữa điều kiện thực tế và điều kiện yêu cầu và dựa vào điểm khác biệt đó để đánh giá tính năng phần mềm.

**- Định nghĩa của Daniel Galin**: "Kiểm thử phần mềm là một quá trình được tiến hành bởi một nhóm chuyên viên kiểm thử, trong đó một đơn vị phần mềm, một nhóm các đơn vị được tích hợp, hoặc cả gói phần mềm được kiểm tra bằng cách chạy các chương trình trên máy tính. Tất cả các bước kiểm tra được tiến hành theo các thủ tục kiểm thử và các trường hợp kiểm thử đã được thông qua".

Định nghĩa của Daniel Galin là một định nghĩa khá hoàn thiện về kiểm thử phần mềm. Một số thuật ngữ có trong định nghĩa của Daniel Galin:

* Nhóm chuyên viên kiểm thử: Một nhóm độc lập hoặc nhóm tư vấn từ bên ngoài, những người chuyên kiểm thử được chỉ định để thực hiện các nhiệm vụ chủ yếu là để phát hiện và loại bỏ sai lệch và để đảm bảo kiểm thử hiệu quả bởi các chuyên gia kiểm thử được đào tạo.
* Các thủ tục kiểm thử đã được thông qua: Quá trình kiểm thử được thực hiện theo kế hoạch kiểm thử và các thủ tục kiểm thử được thông qua phù hợp với các thủ tục đảm bảo chất lượng phần mềm được thông qua bởi tổ chức phát triển phần mềm.

Các trường hợp kiểm thử được thông qua: Các trường hợp kiểm thử được định nghĩa đầy đủ trong kế hoạch kiểm thử. Không có sự thiếu sót hoặc bổ sung nào được mong đợi xảy ra trong suốt quá trình thực thi kiểm thử.

## II.2 Mục đích của kiểm thử:

Kiểm thử không chỉ giới hạn ở việc thực hiện một chương trình hoặc ứng dụng với mục đích đi tìm các lỗi phần mềm (bao gồm các lỗi và các thiếu sót) mà còn là một quá trình phê chuẩn và xác minh một chương trình máy tính / ứng dụng / sản phẩm nhằm:

-Tìm ra được càng nhiều lỗi càng tốt trong điều kiện về thời gian đã định và nguồn lực sẵn có.

- Chứng minh rằng sản phẩm phần mềm phù hợp với các đặc tả của nó.

- Xác thực chất lượng kiểm thử phần mềm đã dùng chi phí và nỗ lực tối thiểu.

- Thiết kế tài liệu kiểm thử một cách có hệ thống và thực hiện nó sao cho có hiệu quả, tiết kiệm được thời gian, công sức.

## II.3 Các thuật ngữ và định nghĩa cơ bản về kiểm thử

Kỹ nghệ kiểm thử đã phát triển và tiến hoá hàng mấy chục năm nên các thuật ngữ trong các tài liệu khác nhau thường không thống nhất và thiếu tương thích. Các thuật ngữ được trình bày trong cuốn sách này dựa vào các thuật ngữ chuẩn được phát triển bởi IEEE (Viện Kỹ nghệ điện và điện tử) với việc chọn lọc cẩn thận các thuật ngữ tiếng Việt tương ứng.

**Lỗi phần mềm( Software Error):** Lỗi là những vấn đề mà con người mắc phải trong quá trình phát triển các sản phẩm phần mềm. Trong thực tế, con người luôn có thể phạm lỗi. Khi lập trình viên phạm lỗi trong lập trình, ta gọi các lỗi đó là bug (con bọ). Lỗi có thể phát tán. Chẳng hạn, một lỗi về xác định yêu cầu có thể dẫn đến sai lầm về thiết kế và càng sai khi lập trình theo thiết kế này. Lỗi phần mềm có thể là lỗi cú pháp hoặc lỗi logic. Lỗi là nguyên nhân dẫn đến sai sót.

**Sai sót của phần mềm (Software Fault):** Sai là kết quả của lỗi, hay nói khác đi, lỗi sẽ dẫn đến sai. Cũng có thể nói sai là một biểu diễn của lỗi dưới dạng một biểu thức, chẳng hạn chương trình, văn bản, sơ đồ dòng dữ liệu, biểu đồ lớp,.... Sai lầm có thể khó bị phát hiện. Khi nhà thiết kế mắc lỗi bỏ sót trong quá trình thiết kế, sai kết quả từ lỗi đó là thiếu mất cái gì đó mà lẽ ra cần phải có. Sai về nhiệm vụ xuất hiện khi vào sai thông tin, còn sai về bỏ quên xuất hiện khi không vào đủ thông tin. Loại sai thứ hai khó phát hiện và khó sửa hơn loại sai thứ nhất.

Sai sót của phần mềm không phải lúc nào cũng do lỗi phần mềm. Có thể có sai sót do dư thừa hoặc bỏ sót yêu cầu phần mềm (từ khâu khảo sát, phân tích, đưa ra yêu cầu phần mềm bị thừa hoặc bị bỏ so với yêu cầu của khách hàng).

**Hỏng hóc của phần mềm (Software Failure):** Thất bại xuất hiện khi một lỗi được thực thi. Có hai điều cần lưu ý ở đây. Một là thất bại chỉ xuất hiện dưới dạng có thể chạy được mà thông thường là mã nguồn. Hai là các thất bại chỉ liên kết với các lỗi về nhiệm vụ. Còn các thất bại tương ứng với các lỗi về bỏ quên thì xử lý thế nào? Những cái lỗi không bao giờ được tiến hành, hoặc không được tiến hành trong khoảng thời gian dài cần được xử lý thế nào? Virus Michaelangelo là một ví dụ về lỗi loại này. Nó chỉ được tiến hành vào ngày sinh của Michaelangelo, tức ngày 6/3 mà thôi. Việc khảo sát có thể ngăn chặn nhiều thất bại bằng cách tìm ra các lỗi thuộc cả hai loại.

**Sự cố (Incident):** Khi thất bại xuất hiện, nó có thể hiển thị hoặc không, tức là rõ ràng hoặc không rõ ràng đối với người dùng hoặc người kiểm thử. Sự cố là triệu chứng liên kết với một thất bại và thể hiện cho người dùng hoặc người kiểm thử về sự xuất hiện của thất bại này.

**Yêu cầu của khách hàng và đặc tả của phần mềm:** Phần mềm được viết để thực hiện các nhu cầu của khách hàng. Các nhu cầu của khách hàng được thu thập, phân tích và khảo cứu và là cơ sở để quyết định chính xác các đặc trưng cần thiết mà sản phẩm phần mềm cần phải có. Dựa trên yêu cầu của khách hàng và các yêu cầu bắt buộc khác, đặc tả được xây dựng để mô tả chính xác các yêu cầu mà sản phẩm phần mềm cần đáp ứng, và có giao diện thế nào. Tài liệu đặc tả là cơ sở để đội ngũ phát triển phần mềm xây dựng sản phẩm phần mềm. Khi nói đến thất bại trên đây là nói đến việc sản phẩm phần mềm không hoạt động đúng như đặc tả. Lỗi một khi được tiến hành có thể dẫn đến thất bại. Do đó, lỗi về bỏ quên được coi là tương ứng với các lỗi khi xây dựng đặc tả.

**Xác minh và thẩm định:** Xác minh (verification) và thẩm định (validation) hay được dùng lẫn lộn, nhưng thực ra chúng có ý nghĩa khác nhau. Xác minh là quá trình để đảm bảo rằng một sản phẩm phần mềm thỏa mãn đặc tả của nó. Còn thẩm định là quá trình để đảm bảo rằng sản phẩm đáp ứng được yêu cầu của người dùng (khách hàng). Trong thực tế, chúng ta cần thực hiện kiểm chứng trước khi thực hiện việc thẩm định sản phẩm phần mềm. Vì vậy, chúng ta có thuật ngữ V&V (Verification & Validation). Lý do của việc này là chúng ta cần đảm bảo sản phẩm đúng với đặc tả trước. Nếu thực hiện việc thẩm định trước, một khi phát hiện ra lỗi, chúng ta không thể xác định được lỗi này do đặc tả sai hay do lập trình sai so với đặc tả.

**Chất lượng và độ tin cậy của phần mềm:** Theo từ điển, chất lượng của một sản phẩm được thể hiện bằng các đặc trưng phù hợp với đặc tả của nó. Theo cách hiểu này, chất lượng của một sản phẩm phần mềm là sự đáp ứng các yêu cầu về chức năng (tức là các hàm cần được tính toán), sự hoàn thiện và các chuẩn đã được đặc tả, cùng các đặc trưng mong chờ từ mọi sản phẩm phần mềm chuyên nghiệp. Chất lượng phần mềm đặc trưng cho “độ tốt, độ tuyệt hảo” của phần mềm, và gồm có các yếu tố về chất lượng như: tính đúng đắn (hành vi đúng như đặc tả), tính hiệu quả (tiết kiệm thời gian và tiền bạc), độ tin cậy, tính khả kiểm thử (kiểm thử được và dễ), dễ học, dễ sử dụng, dễ bảo trì ... Như vậy, độ tin cậy chỉ là một yếu tố để đánh giá chất lượng phầm mềm. Người kiểm thử hay nhầm lẫn độ tin cậy với chất lượng. Khi kiểm thử đạt tới mức phần mềm chạy ổn định, có thể phụ thuộc vào nó, người kiểm thử thường cho rằng phần mềm đã đạt chất lượng cao. Các yếu tố về mặt chất lượng mà liên quan trực tiếp đến việc phát triển phần mềm được gọi là các tiêu chuẩn chất lượng như tính có cấu trúc, tính đơn thể, tính khả kiểm thử, ...

Độ tin cậy của phần mềm là xác suất để phần mềm chạy không có thất bại trong một khoảng thời gian nhất định. Nó được xem là một yếu tố quan trọng của chất lượng phần mềm. Ngoài ra, thời gian trung bình cho việc khắc phục một sự cố cũng là một thông số quan trọng trong việc đánh giá độ tin cậy của sản phẩm phần mềm.

**Kiểm thử**: Rõ ràng việc kiểm thử liên quan đến các khái niệm trên: lỗi, sai, thất bại và sự cố. Có hai mục đích chính của một phép thử: tìm thất bại hoặc chứng tỏ việc tiến hành của phần mềm là đúng đắn. Vai trò của kiểm thử phần mềm: Kiểm thử phần mềm đóng vai trò quan trọng trong việc đánh giá và thu được chất lượng cao của sản phẩm phần mềm trong quá trình phát triển. Thông qua chu trình “ kiểm thử - tìm lỗi - sửa lỗi”, ta hy vọng chất lượng của sản phẩm phần mềm sẽ được cải tiến. Mặt khác, thông qua việc tiến hành kiểm thử mức hệ thống trước khi cho lưu hành sản phẩm, ta biết được sản phẩm của ta tốt ở mức nào. Vì thế, nhiều tác giả đã mô tả việc kiểm thử phần mềm là một quy trình kiểm chứng để đánh giá và tăng cường chất lượng của sản phẩm phần mềm.

Quy trình này gồm hai công việc chính là phân tích tĩnh và phân tích động.

• *Phân tích tĩnh*: Việc phân tích tĩnh được tiến hành dựa trên việc khảo sát các tài liệu được xây dựng trong quá trình phát triển sản phẩm như tài liệu đặc tả nhu cầu người dùng, mô hình phần mềm, hồ sơ thiết kế và mã nguồn phần mềm. Các phương pháp phân tích tĩnh truyền thống bao gồm việc khảo sát đặc tả và mã nguồn cùng các tài liệu thiết kế.. Người ta cũng có thể dùng các kỹ thuật phân tích hình thức như kiểm chứng mô hình (model checking) và chứng minh định lý (theorem proving) để chứng minh tính đúng đắn của thiết kế và mã nguồn. Các kỹ thuật này tương đối phức tạp và nằm ngoài khuôn khổ của cuốn giáo trình này. Công việc này không động đến việc thực thi chương trình mà chỉ duyệt, lý giải về tất cả các hành vi có thể của chương trình khi được thực thi. Tối ưu hóa các chương trình dịch là các ví dụ về phân tích tĩnh.

• *Phân tích động*: Phân tích động liên quan đến việc thực thi chương trình để phát hiện những thất bại có thể có của chương trình, hoặc quan sát các tính chất nào đó về hành vi và hiệu quả (performance). Vì gần như không thể thực thi chương trình trên tất cả các dữ liệu đầu vào có thể, ta chỉ có thể chọn một tập con các dữ liệu đầu vào để thực thi, gọi là các “ca kiểm thử”.

Bằng việc phân tích tĩnh và động, người kiểm thử muốn phát hiện nhiều lỗi nhất có thể được để chúng có thể được sửa ở giai đoạn sớm nhất trong quá trình phát triển phần mềm. Phân tích tĩnh và động là hai kỹ thuật bổ sung cho nhau và cần được làm lặp đi lặp lại nhiều trong quá trình kiểm thử.

## II.4 Các kỹ thuật kiểm thử phần mềm

Có 3 kỹ thuật kiểm thử phần mềm chính là:

* Kiểm thử hộp đen.
* Kiểm thử hộp trắng.
* Kiểm thử hộp xám.

#### II.4.1. Kiểm thử hộp đen

* Kỹ thuật kiểm thử hộp đen xem chương trình như là một “**hộp đen**”, trong đó người kiểm thử không quan tâm đến cấu trúc bên trong của chương trình mà chỉ quan tâm tới dữ liệu đầu vào và đầu ra sau khi được xử lý.
* Ưu, nhược điểm: Kiểm thử hộp đen có ưu điểm là có thể đánh giá phần mềm một cách khách quan, người kiểm thử có thể không hiểu biết về mã lệnh và có thể tìm ra các lỗi mà nhân viên phát triển không tìm ra. Song kiểm thử hộp đen lại có nhược điểm là thăm dò mù, do nhân viên kiểm thử không biết các chương trình thực sự được xây dựng như thế nào, dẫn đến trường hợp nếu kiểm thử hộp đen phải viết rất nhiều trường hợp kiểm thử trong khi chỉ cần viết một ca kiểm thử duy nhất để có thể kiểm tra được.

#### II.4.2. Kiểm thử hộp trắng

Kỹ thuật kiểm thử hộp trắng hay còn gọi là “kiểm thử cấu trúc” là kỹ thuật kiểm thử cho phép khảo sát kiến trúc bên trong của chương trình. Kiểm thử hộp trắng là chiến lược được thực hiện trên ba trong sáu loại kiểm thử cơ bản trong các giai đoạn kiểm thử phần mềm là: kiểm thử đơn vị, kiểm thử tích hợp và kiểm thử hồi quy. Mục tiêu của kiểm thử hộp trắng là kiểm thử bao phủ nhiều nhất các câu lệnh, điểm quyết định và các rẽ nhánh trong mã nguồn nếu có thể.

#### II.4.3. Kiểm thử hộp xám

Kiểm thử hộp xám là kỹ thuật kiểm thử có sự kết hợp giữa kiểm thử hộp đen và kiểm thử hộp trắng. Trong đó ta cũng quan tâm đến dữ liệu đầu vào và đầu ra giống như trong kiểm thử hộp đen, song lại đòi hỏi có sự truy cập đến cấu trúc dữ liệu và giải thuật để thiết kế các trường hợp kiểm thử.

## II.5 Các mức kiểm thử

Một trong các khái niệm then chốt của việc kiểm thử là các mức của việc kiểm thử. Các mức của việc kiểm thử phản ánh mức độ trừu tượng được thấy trong mô hình thác nước của vòng đời của việc phát triển phần mềm.

Các mức kiểm thử có thể được mô tả sơ bộ như sau:

#### II.5.1. Kiểm thử đơn vị (Unit Testing)

Kiểm thử đơn vị là việc kiểm thử các đơn vị chương trình một cách cô lập. Thế nào là một đơn vị chương trình? Câu trả lời phụ thuộc vào ngữ cảnh công việc. Một đơn vị chương trình là một đoạn mã nguồn như hàm hoặc phương thức của một lớp, có thể được gọi từ ngoài, và cũng có thể gọi đến các đơn vị chương trình khác. Đơn vị cũng còn được coi là một đơn thể để kết hợp. Đơn vị chương trình cần được kiểm thử riêng biệt để phát hiện lỗi trong nội tại và khắc phục trước khi được tích hợp với các đơn vị khác. Kiểm thử đơn vị thường được làm bởi chính tác giả của chương trình, và có thể tiến hành theo hai giai đoạn: kiểm thử đơn vị tĩnh và kiểm thử đơn vị động.

#### II.5.2. Kiểm thử tích hợp(Integration Testing)

Mức kế tiếp với kiểm thử đơn vị là kiểm thử tích hợp. Sau khi các đơn vị chương trình để cấu thành hệ thống đã được kiểm thử, chúng cần được kết nối với nhau để tạo thành hệ thống đầy đủ và có thể làm việc. Công việc này không hề đơn giản và có thể có những lỗi về giao diện giữa các đơn vị, và cần phẩi kiểm thử để phát hiện những lỗi này. Công đoạn này gồm hai giai đoạn: giai đoạn kiểm thử tích hợp và giai đoạn kiểm thử hệ thống. Kiểm thử tích hợp nhằm đảm bảo hệ thống làm việc ổn định trong môi trường thí nghiệm để sẵn sàng cho việc đưa vào môi trường thực sự bằng cách đặt các đơn vị với nhau theo phương pháp tăng dần.

#### II.5.3. Kiểm thử hệ thống(System Testing)

Kiểm thử mức này được áp dụng khi đã có một hệ thống đầy đủ sau khi tất cả các thành phần đã được tích hợp. Mục đích của kiểm thử hệ thống là để đảm bảo rằng việc cài đặt tuân thủ đầy đủ các yêu cầu được đặc tả của người dùng. Công việc này tốn nhiều công sức, vì có nhiều khía cạnh về yêu cầu người dùng cần được kiểm thử.

#### II.5.4. Kiểm thử chấp nhận(Acceptance Testing)

Khi nhóm kiểm thử hệ thống đã thỏa mãn với một sản phẩm, sản phẩm đó đã sẵn sàng để đưa vào sử dụng. Khi đó hệ thống cần trải qua giai đoạn kiểm thử chấp nhận. Kiểm thử chấp nhận được thực thi bởi chính các khách hàng nhằm đảm bảo rằng sản phẩm phần mềm làm việc đúng như họ mong đợi. Có hai loại kiểm thử chấp nhận: kiểm thử chấp nhận người dùng, được tiến hành bởi người dùng, và kiẻm thử chấp nhận doanh nghiệp, được tiến hành bởi nhà sản xuất ra sản phẩm phần mềm.

## II.6 Quy trình kiểm thử

Quy trình kiểm thử tổng quát:

* Lập kế hoạch kiểm thử (Test Planning).
* Phân tích và thiết kế các ca kiểm thử (Test analysis and Design).
* Thực thi kiểm thử (Test Executing).
* Báo cáo kiểm thử, đánh giá (Test Report and Evaluation).

#### II.6.1 Lập kế hoạch kiểm thử

Kế hoạch kiểm thử là tài liệu mô tả về phạm vi, cách tiếp cận, các nguồn lực và kế hoạch thực hiện kiểm thử.

+ Mục tiêu của lập kế hoạch kiểm thử:

* Thiết lập được mục tiêu ngắn hạn và dài hạn của việc kiểm thử.
* Nhận biết được các rủi ro có thể xảy ra.
* Xác định được cách tiếp cận và kế hoạch cho việc kiểm thử.

+ Nhiệm vụ chính của hoạt động lập kế hoạch kiểm thử:

* Xác định được phạm vi kiểm thử, các rủi ro có thể xảy ra, xác định được mục tiêu kiểm thử để đảm bảo đo đạc được chất lượng phần mềm.
* Xác định cách tiếp cận việc kiểm thử, các kĩ thuật được sử dụng, độ bao phủ kiểm thử cho phép.
* Thực thi theo chính sách và chiến lược kiểm thử.
* Xác định nguồn lực, chi phí kiểm thử cần thiết.
* Lên kế hoạch cho hoạt động phân tích, thiết kế, thực thi kiểm thử.
* Xác định tiêu chí kết thúc kiểm thử.
* Liệt kê những kết quả, tài kiệu có được sau khi thực hiện kiểm thử.

#### II.6.2 Phân tích và thiết kế ca kiểm thử

Phân tích và thiết kế ca kiểm thử là hoạt động chuyển các mục tiêu của kiểm thử thành các trường hợp kiểm thử cụ thể.

+ Mục tiêu của Phân tích và thiết kế ca kiểm thử:

Xây dựng được bộ khung các tình huống cần kiểm thử mà trong đó:

* Các Testcase cần bao phủ tất cả các khía cạnh kiểm thử cho từng yêu cầu phần mềm.
* Các Testcase cần bao phủ tất cả các yêu cầu trong các chiến lược kiểm thử.
* Nếu cần kiểm thử tự động, Test designer sẽ xây dựng các kịch bản kiểm thử tự động dựa trên các test case/ test procedure.

+ Hoạt động chính của quá trình Phân tích và thiết kế ca kiểm thử:

* Kiểm tra lại tất cả các tài liệu của dự án: bản đặc tả yêu cầu hệ thống, thiết kế kiến trúc, thiết kế chi tiết, nguyên mẫu giao diện hệ thống.
* Phân tích và đánh giá khả năng kiểm thử được của hệ thống dựa trên yêu cầu của khách hàng.
* Xác định và đặt thứ tự ưu tiên cho các điều kiện kiểm thử dựa trên kết quả phân tích các chức năng cần kiểm thử, bản mô tả các chức năng đó.
* Thiết kế và đặt ưu tiên cho các tình huống kiểm thử mức cao.
* Xác định dữ liệu kiểm thử cần thiết cho các điều kiện và trường hợp kiểm thử.
* Thiết kế cho việc thiết lập môi trường kiểm thử và sự thay đổi yêu cầu của khách hàng nếu có.
* Tạo mối liên hệ giữa yêu cầu khách hàng và các trường hợp kiểm thử để kiểm soát được hoạt động kiểm thử và sự thay đổi yêu cầu của khách hàng nếu có.

#### II.6.3 Thực hiện kiểm thử

Thực hiện kiểm thử là quá trình bao gồm:

* Phát triển và đặt thứ tự ưu tiên cho các thủ tục kiểm thử, tạo dữ liệu kiểm thử, các công cụ kiểm thử (nếu có), kịch bản kiểm thử tự động (Test Script), môi trường kiểm thử (Test Environment).
* Chạy thử một phần chức năng hay cả một hệ thống dựa trên các sản phẩm, tài liệu đã chuẩn bị ở bước trên nhằm đưa ra kết quả thực tế.

+ Mục tiêu của hoạt động chuẩn bị và thực hiện kiểm thử:

* Xây dựng được các thủ tục kiểm thử.
* Cài đặt môi trường, chuẩn bị dữ liệu kiểm thử.
* Thực hiện kiểm thử, báo cáo lỗi.

+ Nhiệm vụ của hoạt động Thực hiện kiểm thử:

* Phát triển và xét thự tự ưu tiên cho các tình huống kiểm thử, tạo dữ liệu và chuẩn bị các công cụ kiểm thử, viết các tình huống kiểm thử tự động.
* Xây dựng các bộ kiểm thử từ các tình huống kiểm thử để kiểm thử đạt hiệu quả nhất.
* Cài đặt và kiểm tra môi trường kiểm thử.
* Thực hiện kiểm thử cho một bộ, hoặc theo các Test case.
* Ghi lại kết quả của kiểm thử.
* So sánh kết quả kiểm thử thực tế với kết quả mong đợi.
* Báo cáo sự cố và phân tích nguyên nhân.
* Lặp lại hoạt động kiểm thử cho mỗi sự cố đã được xác định hoặc kiểm tra lại. Để xác nhận lỗi xảy ra không ảnh hưởng đến các thành phần khác của phần mềm.

#### II.6.4 Báo cáo và đánh giá kiểm thử

Báo cáo và đánh giá kiểm thử là hoạt động đánh giá kết quả kiểm thử so với mục tiêu đã đề ra.

+ Mục tiêu chính:

* Để đánh giá việc kiểm thử như thế nào là đủ cho mỗi giai đoạn kiểm thử.

+ Nhiệm vụ chính:

* Kiểm tra các kết quả kiểm thử so với các tiêu chí dừng kiểm thử.
* Đánh giá tình trạng hiện tại để tiến hành kiểm thử thêm nếu cần.
* Viết báo cáo tổng kết các hoạt động kiểm thử cho các bên liên quan.

## II.7 Nguyên tắc kiểm thử phần mềm

*II.7.1 Kiểm thử chứng mình sự hiện diện của lỗi:*

Kiểm thử chỉ có thể chứng minh được rằng sản phẩm có lỗi. [Kiểm thử phần mềm](http://hockiemthu.com/tong-quan-kiem-thu-phan-mem/)không thể chứng mình rằng sản phẩm không còn lỗi. Nghĩa là sản phẩm luôn có lỗi cho dù có kiểm thử nhiều bao nhiêu. Do đó, điều quan trọng là chúng ta phải thiết kế các trường hợp kiểm thử (test case) sao cho có thể tìm được càng nhiều lỗi càng tốt

*II.7.2 Kiểm thử toàn bộ là không khả thi*

Trừ khi sản phẩm được kiểm thử quá đơn giản cũng như không có nhiều giá trị đầu vào thì việc chứng minh sản phẩm không còn bug cho dù có kiểm thử nhiều đến đâu là không khả thi. Hầu hết các sản phẩm ngày nay rất đa dạng và phức tạp do được phát triển trên nhiều nền tảng, công nghệ phong phú cũng như khả năng lưu trữ kết nối dữ liệu lớn, khiến việc kiểm thử trở nên khó khăn và việc kiểm thử toàn bộ là gần như không thể.

Do đó, việc chúng ta có thể làm là chọn thực thi những loại kiểm thử quan trọng nhất dựa trên phân tích rủi ro cũng như tầm quan trọng và độ ưu tiên của việc kiểm thử. Nghĩa là phải lên kế hoạch kiểm thử, thiết kế trường hợp kiểm thử sao cho có độ bao phủ nhiều nhất và giảm thiểu rủi ro sót lỗi khi đến tay người dùng

*II.7.3 Kiểm thử càng sớm càng tốt*

Hoạt động kiểm thử được triển khai sớm chừng nào thì tốt chừng ấy. Chẳng hạn, chúng ta có thể bắt đầu hoạt động kiểm thử ngay từ khi sản phẩm bắt đầu hình thành chẳng hạn như giai đoan lấy yêu cầu khách hàng hay thiết kế tài liệu sản phẩm.

Thông thường, thời gian cho kiểm thử thường bị co lại khi sản phẩm gần ra thị trường. Do đó, việc kiểm thử sớm trong giai đoạn đầu sẽ giúp chúng ta có thời gian để tiến hành kiểm thử trong từng giai đoạn một cách đầy đủ.

Ngoài ra ai làm phần mềm cũng biết được rằng việc phát hiện lỗi càng trể bao nhiêu thì chi phí để sửa lỗi càng cao bấy nhiêu. Tương tự, việc thay đổi yêu cầu không đúng ngay từ đầu thường tốn ít chi phí thay đổi tính năng trong hệ thống.

*II.7.4 Lỗi phân bố tập trung*

Trong quá trình kiểm thử, chúng ta sẽ có thể dễ dàng quan sát thấy đa phần những lỗi tìm được thường chỉ liên quan đến một vài tính năng của hệ thống. Điều này cũng thuận theo [nguyên lý Pareto](https://vi.wikipedia.org/wiki/Nguy%C3%AAn_l%C3%BD_Pareto): 80% số lượng lỗi được tìm thấy trong 20% tính năng của hệ thống.

Điều này cũng có nghĩa là khi tìm lỗi, chúng ta nên tập trung vào những module, thành phần chức năng chính của hệ thống. Một lỗi trên những module chính này có giá trị gấp nhiều lần lỗi tìm được trên những module phụ khác.

*II.7.5 Nghịch lý thuốc trừ sâu (pesticide paradox)*

Trong kiểm thử phần mềm, nếu bạn cứ thực thi lặp đi lặp lại một bộ test case thì có khả năng rất thấp bạn sẽ tìm được lỗi từ những trường hợp kiểm thử này. Nguyên nhân là do khi hệ thống ngày càng hoàn thiện, những lỗi được tìm thấy lúc trước đã được sửa trong khi những trường hợp kiểm thử đã cũ. Do đó, khi một lỗi được sửa hay một tính năng mới được thêm vào, chúng ta nên tiến hành làm regression (kiểm thử hồi qui) nhằm mục đích đảm bảo những thay đổi này không ảnh hưởng đến những vùng khác của sản phẩm. Tuy nhiên, các trường hợp kiểm thử trong regression test cũng cần phải được cập nhật để phản ánh sự thay đổi tương ứng của hệ thống.

*II.7.6 Kiểm thử phần mềm phụ thuộc vào ngữ cảnh*

Tùy thuộc vào loại cũng như bản chất của ứng dụng mà chúng ta sẽ áp dụng những phương thức, kỹ thuật, cũng như loại kiểm thử khác nhau.

Chẳng hạn như phần mềm trong các thiết bị y khoa cần sẽ được kiểm thử kỹ lưỡng hơn một trò chơi điện tử. Quan trọng hơn, việc kiểm thử trên loại thiết bị này đòi hỏi phải dựa trên đánh giá rủi ro, đáp ứng những qui định nghiêm ngặt trong y khoa cũng như một bộ kiểm thử đặc thù. Tương tự, một website “lớn” cần phải được một cách đầy đủ từ hiệu năng đến tính năng nhằm đảm bảo server không bị ảnh hưởng khi có nhiều người truy cập vào hệ thống

*II.7.7 Quan niệm sai lầm về việc “hết lỗi”*

Việc không tìm thấy lỗi trên sản phẩm không đồng nghĩa với việc sản phẩm đã sẵn sàng để tung ra thị trường. Việc không tìm thấy lỗi cũng có thể là do bộ trường hợp kiểm thử được tạo ra chỉ nhằm kiểm tra những tính năng được làm đúng theo yêu cầu thay vì nhằm tìm kiếm lỗi mới.

# CHƯƠNG III KIỂM THỬ TỰ ĐỘNG VÀ CÔNG CỤ HỖ TRỢ

Kiểm thử đang được xem là giải pháp chủ yếu nhằm đảm bảo chất lượng cho các sản phẩm phần mềm. Tuy nhiên, các hoạt động kiểm thử hiện nay chủ yếu được thực hiện một cách thủ công và tiêu tốn khoảng 30-50% tài nguyên (thời gian, nhân lực và chi phí) của quá trình phát triển sản phẩm phần mềm. Hơn nữa, độ phức tạp của các phần mềm ngày càng tăng và trong môi trường cạnh tranh như hiện nay đòi hỏi các công ty phần mềm phải áp dụng các phương pháp và công cụ nhằm tự động hóa các hoạt động kiểm thử. Chương này giới thiệu về kiểm thử tự động và các công cụ hỗ trợ nhằm giải quyết vấn đề này.

## III.1 Tổng quan về kiểm thử tự động

Kiểm thử tự động là quá trình thực hiện một cách tự động các bước trong một kịch bản kiểm thử. Kiểm thử tự động bằng một công cụ nhằm rút ngắn thời gian kiểm thử. Mục đích của kiểm thử tự động là giảm thiểu thời gian, công sức và kinh phí, tăng độ tin cậy, tăng tính hiệu quả và giảm sự nhàm chán cho người kiểm thử trong quá trình kiểm thử sản phẩm phần mềm.

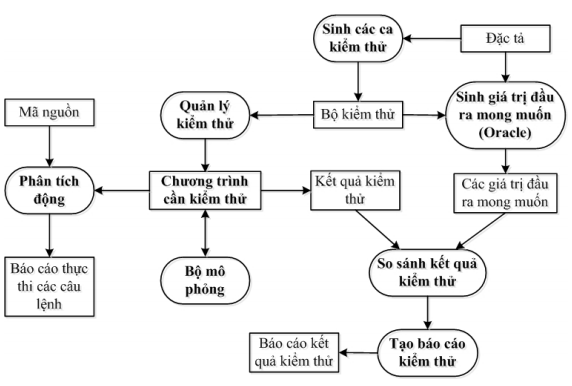
Quy trình kiểm thử tự động gồm 4 bước:

* Bước 1: Viết kịch bản kiểm thử, dùng công cụ kiểm thử để ghi lại các thao tác lên phần mềm cần kiểm tra và tự động sinh ra test script.
* Bước 2: Chỉnh sửa để kịch bản kiểm thử thực hiện kiểm tra theo đúng yêu cầu đặt ra, làm theo trường hợp kiểm thử cần thực hiện.
* Bước 3: Chạy kịch bản kiểm thử, giám sát hoạt động kiểm tra phần mềm của kịch bản kiểm thử.
* Bước 4: Kiểm tra kết quả thông báo sau khi thực hiện kiểm thử tự động. Sau đó bổ sung, chỉnh sửa những sai sót.

Kiểm thử tự động sẽ được sử dụng khi dự án không đủ tài nguyên (thời gian, nhân lực và chi phí), phải thực hiện kiểm thử hồi quy khi sản phẩm được sửa đổi hoặc nâng cấp và cần kiểm thử lại các tính năng đã thực hiện tốt trước đó, kiểm tra khả năng vận hành của sản phẩm trong các môi trường đặc biệt (đo tốc độ xử lý trung bình ứng với mỗi yêu cầu, xác định khả năng chịu tải tối đa, xác định cấu hình tối thiểu để thực thi hệ thống, kiểm tra các cơ chế an ninh và an toàn, ...).

## III.2 Kiến trúc của một bộ kiểm thử tự động

Trong thức tế, có rất nhiều bộ công cụ hỗ trợ kiểm thử tự động được phát triển nhằm góp phần giải quyết các vấn đề khó khăn của quy trình kiểm thử. Hình 2.1 mô tả kiến trúc chung nhất của một bộ kiểm thử tự động [Som10]. Trong kiến trúc này, các công cụ kiểm thử được tích hợp trong một quy trình thống nhất nhằm hỗ trợ đầy đủ các hoạt động kiểm thử trong quy trình kiểm thử các sản phẩm phần mềm.



*Hình 3.1 Kiến trúc chung của một bộ kiểm thử tự động.*

Các công cụ cơ bản trong kiến trúc này bao gồm:

• Quản lý kiểm thử: công cụ này cho phép quản lý việc thực hiện/thực thi các ca kiểm thử. Nó giám sát việc thực hiện từng ca kiểm thử ứng với bộ giá trị đầu vào, giá trị đầu ra mong muốn và giá trị đầu ra thực tế. JUnit là một ví dụ điển hình về công cụ này.

• Sinh các ca kiểm thử: Đây là một trong những công cụ quan trọng nhất của các bộ kiểm thử tự động. Tùy thuộc vào các kỹ thuật kiểm thử được áp dụng, công cụ này sẽ sinh ra tập các ca kiểm thử (chưa gồm giá trị đầu ra mong muốn) cho chương trình/đơn vị chương trình cần kiểm thử. Các ca kiểm thử được sinh ra chỉ chứa giá trị đầu vào để thực hiện nó. Các giá trị này có thể được lựa chọn trong cơ sở dữ liệu hoặc được sinh một cách ngẫu nhiên.

• Sinh giá trị đầu ra mong muốn: Các ca kiểm thử được sinh ra bởi công cụ trên chỉ chứa các giá trị đầu vào. Công cụ này cho phép sinh ra giá trị đầu ra mong muốn ứng với mỗi bộ dữ liệu đầu vào của mỗi ca kiểm thử. Giá trị đầu ra mong muốn này sẽ được so sánh với giá trị đầu ra thực tế khi thực hiện ca kiểm thử này nhằm phát hiện ra các lỗi/khiếm khuyết của sản phẩm.

• So sánh kết quả kiểm thử: Công cụ này so sánh giá trị đầu ra thực tế và giá trị đầu ra mong muốn của mỗi ca kiểm thử khi nó được thực hiện trên chương trình/đơn vị chương trình cần kiểm thử.

• Tạo báo cáo kiểm thử: Một trong những ưu điểm của các bộ công cụ kiểm thử tự động là nó có cơ chế sinh báo cáo kiểm thử một cách chính xác và nhất quán. Dựa vào kết quả của công cụ so sánh kết quả kiểm thử, công cụ này sẽ tự động sinh ra báo cáo kết quả kiểm thử theo định dạng mong muốn của đơn vị phát triển.

• Phân tích động: Công cụ này cung cấp một cơ chế nhằm kiểm tra việc thực hiện của các câu lệnh của chương trình cần kiểm thử nhằm phát hiện ra các lỗi và phát hiện các câu lệnh/đoạn lệnh không được thực hiện bới một tập các ca kiểm thử cho trước. Công cụ này cũng rất hiệu quả trong việc đánh giá tính hiệu quả của một bộ kiểm thử cho trước.

• Bộ mô phỏng: Có nhiều loại mình mô phỏng được cung cấp trong các bộ kiểm thử tự động. Mục đích của các công cụ này là mô phỏng quá trình thực hiện của chương trình cần kiểm thử. Ví dụ, các công cụ mô phỏng giao diện người dùng cho phép thực hiện tự động các tương tác giữa người dùng và sản phẩm. Cypress1 là một ví dụ về một công cụ mô phỏng giao diện người dùng cho các ứng dụng Web.

Trong thực tế, các bộ công cụ kiểm thử tự động có thể có thêm một số công cụ khác như cho phép đặc tả các tính chất của hệ thống cần kiểm thử…. Một số bộ công cụ chỉ hỗ trợ một số công cụ trong các công cụ đã liệt kê ở trên.

## III.3 Ưu điểm, nhược điểm của kiểm thử tự động

+ Ưu điểm:

* Độ tin cậy cao (Reliability): Công cụ kiểm thử tự động có sự ổn định cao hơn sơ với con người, đặc biệt trong trường hợp có nhiều Test case, nên độ tin cậy cao hơn so với kiểm thử thủ công.
* Khả năng lặp (Repeatability): Công cụ kiểm thử tự động ra đời là để giúp cho các kiểm thử viên không phải lặp đi lặp lại các thao tác (ví dụ: nhập dữ liệu, click, check kết quả,…) một cách nhàm chán với độ tin cậy và ổn định cao.
* Khả năng tái sử dụng (Reusability): Với một bộ kiểm thử tự động, người ta có thể sử dụng cho nhiều phiên bản ứng dụng khác nhau, đây chính là tính tái sử dụng.
* Tốc độ cao (High speed): Do thực thi bởi công cụ nên tốc độ của kiểm thử tự động nhanh hơn nhiều so với tốc độ của con người
* Chi phí thấp (Cost Reduction): Nếu áp dụng kiểm thử tự động đúng cách, người ta có thể tiết kiệm được nhiều chi phí, thời gian và nhân lực, do kiểm thử tự động nhanh hơn nhiều so với kiểm thử thủ công, đồng thời nhân lực cần để thực thi và bảo trì không nhiều.

+ Nhược điểm:

* Khó mở rộng, khó bảo trì (Poor scalability and maintainability): Trong cùng một dự án, để mở rộng phạm vi cho kiểm thử tự động khó hơn nhiều so với kiểm thử thủ công vì cập nhật hay chỉnh sửa yêu cầu nhiều công việc như debug, thay đổi dữ liệu đầu vào và cập nhật code mới.
* Khả năng bao phủ thấp (Low coverage): Do khó mở rộng và đòi hỏi nhiều kỹ năng lập trình nên độ bao phủ của kiểm thử tự động thấp xét trên góc nhìn toàn dự án.
* Vấn đề công cụ và nhân lực (Technology and people issues): Hiện nay cũng có nhiều công cụ hỗ trợ kiểm thử tự động khá tốt nhưng chúng vẫn còn nhiều hạn chế. Ngoài ra nhân lực đạt yêu cầu (có thể sử dụng thành thạo các công cụ này) cũng không nhiều.

## III.4 Một số công cụ kiểm thử tự động

*II.4.1 Quick Test Professional (QTP)*

Quick Test Professionallà phần mềm kiểm soát việc kiểm thử tự động các chức năng của các sản phẩm phần mềm cần kiểm thử. Sản phẩm này bao gồm một tập các mô-đun có thể tương tác với nhau nhằm quản lý toàn bộ quy trình kiểm thử phần mềm. Quick Test Professional là một công cụ hỗ trợ kiểm thử hàm (kiểm thử chức năng) và cho phép tiến hành kiểm thử hồi quy một cách tự động.

*II.4.2 Jmeter*

Jmeter là một mã nguồn mở được viết bằng java. Công cụ để đo độ tải và performance của đối tượng, có thể sử dụng để test performance trên cả nguồn tĩnh và nguồn động, có thể kiểm tra độ tải và hiệu năng trên nhiều loại server khác nhau như: Web – HTTP, HTTPS, SOAP, Database thông qua JDBC, LDAP, JMS, Mail – SMTP(S), POP3(S) and IMAP(S)…

Cách thức hoạt động của Jmeter: Giả lập một nhóm người dùng gửi các yêu cầu tới một máy chủ, nhận và xử lý các response từ máy chủ và trình diễn các kết quả đó cho người dùng dưới dạng bảng biểu, đồ thị,cây…

*II.4.3 Load Runner*

Load Runner giả lập một môi trường ảo gồm nhiều người dùng thực hiện các giao dịch cùng một lúc nhằm giám sát các thông số xử lý của phần mềm cần kiểm thử. Kết quả thống kê sẽ được lưu lại và cho phép kiểm thử viên thực hiện phân tích nhằm kiểm thử khả năng chịu tải và các yêu cầu phi chức năng khác của sản phẩm. Trong quá trình kiểm thử, Load Runner tự động tạo ra các kịch bản kiểm thử để lưu lại các thao tác người dùng tương tác lên phần mềm. Mỗi kịch bản này còn được xem là hoạt động của một người dùng ảo mà Load Runner giả lập. Ngoài ra, công cụ này còn cho phép tổ chức, điều chỉnh, quản lý và giám sát hoạt động kiểm tra khả năng chịu tải.

*II.4.4 Selenium*

Selenium là một trong những công cụ kiểm thử phần mềm tự động mã nguồn mở (open source test automation tool) mạnh mẽ nhất hiện nay cho việc kiểm thử ứng dụng Web. Selenium script có thể chạy được trên hầu hết các trình duyệt như IE, Mozilla FireFox, Chrome, Safari, Opera; và hầu hết các hệ điều hành như Windows, Mac, Linux. Selenium hỗ trợ nhiều ngôn ngữ lập trình. Selenium có 4 phần: Selenium IDE, Selenium RC, Selenium Grid và Selenium WebDriver.

*II.4.5 NUnit*

NUnit là một testing framework mã nguồn mở được phát triển bởi một nhóm lập trình viên (Charlie Poole, Rob Prouse and Simone Busoli), tương tác trực tiếp với Visual Studio, đồng thời có thể chạy độc lập mà không phụ thuộc vào Visual Studio

*II.4.6 JUnit*

JUnit là một framwork đơn giản dùng cho việc tạo các unit testing tự động, và chạy các test có thể lặp đi lặp lại. Nó chỉ là một phần của kiến trúc xUnit cho việc tạo các unit testing. Junit là một chuẩn trên thực tế cho unit testing trong Java. JUnit tránh cho người lập trình phải làm đi làm lại những việc kiểm thử nhàm chán bằng cách tách biệt mã kiểm thử ra khỏi mã chương trình, đồng thời tự động hóa việc tổ chức và thi hành các bô số kiểm thử.

*II.4.7 Cypress*

Cypress là công cụ kiểm tra đầu cuối thế hệ tiếp theo được xây dựng cho các trang web hiện đại. Cypress là một công cụ kiểm tra miễn phí, mã nguồn mở, được cài đặt cục bộ và một dịch vụ ghi lại các bài kiểm tra của bạn. Nó sử dụng ngôn ngữ lập trình JavaScript và cho phép người dùng kiểm tra bất cứ thứ gì chạy trên trình duyệt.

## III.5 Tổng kết kiểm thử tự đông

Kiểm thử tự động đang được quan tâm như là một giải pháp hiệu quả và duy nhất nhằm cải thiện tính chính xác và hiệu quả cũng như giảm kinh phí và rút ngắn thời gian trong quá trình kiểm thử các sản phẩm phần mềm. Đã có nhiều công cụ được phát triển hỗ trợ các mục đích trên. Tùy thuộc vào yêu cầu kiểm thử của từng sản phẩm, các công ty sẽ lựa chọn các công cụ phù hợp. Tuy nhiên, rất khó để tìm được một bộ công cụ đáp ứng tất cả các yêu cầu kiểm thử. Trong nhiều trường hợp, các công ty cần chủ động mở rộng và phát triển thêm các công cụ phục vụ các mục đích cụ thể.

# CHƯƠNG IV. CÔNG CỤ KIỂM THỬ PHẦN MỀM TỰ ĐỘNG CYPRESS VÀ ỨNG DỤNG

## IV.1 Tìm hiểu công cụ kiểm thử tự động Cypress

Cypress là một công cụ kiểm thử phần mềm tự động được sáng lập bởi Brian Man và Randall Kent. Đây là công cụ kiểm thử front - end tiên tiến nhất hiện nay. Sau một thời gian phát triển và chỉ tập trung vào một nhóm người dùng nhất định, cypress đã chính thức phát hành bản beta thử nghiệm vào năm 2015 và hoàn toàn là mã nguồn mở. Bất cứ ai muốn dử dụng cypress đều có thể tải về và sử dụng ứng dụng trên máy tính cá nhân mà không cần phải liên hệ trước hay đăng ký bất cứ thứ gì với đội ngũ sản xuất.

Cypress là một mã nguồn mở và là một công cụ kiểm thử phần mềm tự động hóa để thử nghiệm các ứng dụng web. Nó có khả năng hoạt động trên nhiều các trình duyệt và hệ điều hành khác nhau. Cypress không chỉ là một công cụ duy nhất mà là một bộ các công cụ giúp những người kiểm thử tự động hóa các ứng dụng dựa trên web hiệu quả hơn.

## IV.2 Các hệ điều hành và trình duyệt hỗ trợ

1. Các hệ điều hành hỗ trợ ứng dụng

Cypress là một ứng dụng dành cho máy tính cá nhân được cài đặt trực tiếp trên máy tính. Ứng dụng này hiện tại đang hỗ trợ các hệ điều hành

* Mac OS 10.9+ (Mavericks +), chỉ có các gói nhị phân 64 bit được cung cấp cho macOS
* Linux Ubuntu 12.04+, Fedora 21, Debian 8
* Window 7+, chỉ có 32 bit nhị phân được cung cấp cho Windows

2. Các trình duyệt được hỗ trợ

Khi cypress ban đầu được chạy từ trình chạy thử, bạn có thể chọn chạy cypress trong một số trình duyệt bao gồm

* Canary
* Chrome
* Choromium
* Electron

Cypress tự động phát hiện các trình duyệt hiện có trên hệ điều hành của bạn.

Nhiều trình duyệt như Firefox, Safari, Internet Explorer hiện không được hỗ trợ.

Nhà sản xuất cũng nói rằng việc hỗ trợ chạy ứng dụng trên các trình duyệt này là công việc họ cần làm trong tương lai.

## IV.3 Đặc điểm

* Cypress là một công cụ mã nguồn mở: Vì là mã nguồn mở nên chúng ta có thể sử dụng mà không phải lo lắng về phí bản quyền hay thời hạn sử dụng.
* Cypress hỗ trợ chạy trên nhiều OS khác nhau, Cypress triển khai trên nền tảng Windows, Linux và Mac.
* Nó cho phép recording, editting and debugging tests.
* Cypress không sử dụng Selenium: hầu hết các công cụ kiểm tra đầu cuối đều dựa trên Selenium, đó là lý do tại sao tất cả chúng đều có những vấn đề tương tự. Cypress đã tạo ra sự khác biệt bằng cách xây dựng một kiến trúc mới từ dưới lên. Trong khi Selenium thực hiện các lệnh từ xa thông qua mạng, Cypress chạy trong cùng một vòng lặp chạy như ứng dụng của bạn.
* Cypress tập trung vào làm thế nào để kết thúc thử nghiệm một cách tốt nhất: cypress không phải là một khuôn khổ tự động hóa chung, cũng không phải là một khuôn khổ thử nghiệm đơn vị cho các dịch vụ cuối cùng của bạn. Đã có những công cụ tuyệt vời ngoài đó để làm điều đó. Thay vào đó, cypress chuyên về một điều - tạo ra trải nghiệm tuyệt vời trong khi bạn viết bài kiểm tra cuối cùng cho các ứng dụng web của mình.
* Cypress hoạt động trên bất kỳ khuôn khổ front-end hoặc trang web: cypress thử nghiệm bất cứ thứ gì chạy trên một trình duyệt web. Tất cả kiến trúc xung quanh Cypress được xây dựng để xử lý các khuôn khổ JavaScript hiện đại đặc biệt tốt. Cypress cũng hoạt động tốt như nhau trên các trang hoặc các ứng dụng được hiển thị trên máy chủ cũ hơn thậm chí trong các dự án sử dụng khuôn khổ React, Angular, Vue, Elm,...mới nhất.
* Các bài kiểm tra Cypress chỉ được viết bằng JavaScript: mặc dù bạn có thể sử dụng bất kỳ ngôn ngữ nào khác, cuối cùng mã thử nghiệm được thực hiện bên trong trình duyệt.
* Cypress là tất cả trong một: viết bài kiểm tra cuối cùng để kết thúc có rất nhiều công cụ khác nhau để làm việc cùng nhau. Với cypress bạn có nhiều công cụ trong một. Không cần phải cài đặt 10 công cụ riêng biệt và thư viện để cài đặt bộ thử nghiệm của bạn. Cypress đã có một số công cụ tốt nhất mà bạn có thể đã quen thuộc và làm cho tất cả chúng làm việc cùng nhau liền mạch.
* Cypress dành cho các nhà phát triển và kỹ sư QA
* Cypress chạy nhiều, nhanh hơn nhiều: những cải tiến kiến trúc này mở ra khả năng thực hiện TDD với các bài kiểm tra đầu cuối hoàn chỉnh cho lần đầu tiên. Cypress đã được xây dựng để thử nghiệm và phát triển có thể xảy ra đồng thời. Bạn có thể phát triển nhanh hơn trong khi thực hiện toàn bộ quá trình thử nghiệm bằng các bài kiểm tra bởi vì: bạn có thể thấy ứng dụng của bạn; bạn vẫn có quyền truy cập vào các công cụ dành cho nhà phát triển; và những thay đổi được phản ánh trong thời gian thực. Kết quả cuối cùng là bạn sẽ phát triển thêm, mã của bạn sẽ tốt hơn, và nó sẽ được kiểm tra hoàn toàn.

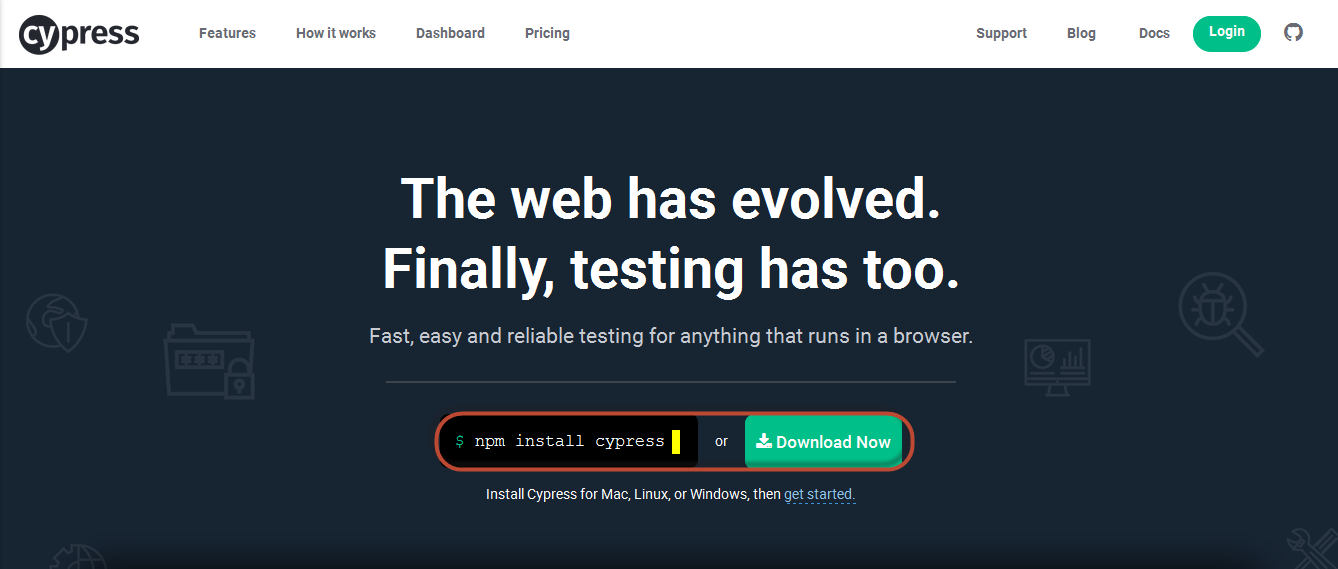
## IV.4 Ngôn ngữ sử dụng viết Script

Cypress sử dụng ngôn ngữ JavaScript để viết test script. **JavaScript**, theo phiên bản hiện hành, là một ngôn ngữ lập trình kịch bản dựa trên đối tượng  được phát triển từ các ý niệm nguyên mẫu. Ngôn ngữ này được dùng rộng rãi cho các trang web, nhưng cũng được dùng để tạo khả năng viết script sử dụng các đối tượng nằm sẵn trong các ứng dụng.

JavaScript là một ngôn ngữ lập trình được nhúng được trong các trang HTML. JavaScript nâng cao tính động và khả năng tương tác cho web-site bằng cách sử dụng các hiệu ứng của nó như thực hiện các phép tính, kiểm tra form, viết các trò chơi, bổ sung các hiệu ứng đặc biệt, tuỳ biến các chọn lựa đồ hoạ, tạo ra các mật khẩu bảo mật và hơn thế nữa.

## IV.5 Cài đặt và sử dụng Cypress.io trên hệ điều hành Window

+ Download cypress trên trang web: https://www.cypress.io/



* Giản nén file đã tải về
* // Bổ sung sau

*IV.5.2 Những lệnh cơ bản*

+ Một số câu lệnh hay được sử dụng trong Cypress:

* cy.visit() // mở một trang web mới trong trình duyệt hiện tại.
* cy.contains () // truy vấn cho một phần tử, tìm phần tử này theo nội dung của nó.
* cy.click () // nhấp chuột vào một phẩn tử
* cy.should () // thực hiện một xác nhận.
* cy.get() // chọn một phần tử dựa trên một lớp CSS
* cy.type() // nhập văn bản với đầu vào đã chọn
* cy.exec() // chạy lệnh hệ thống
* cy.request() // để thực hiện các yêu cầu http
* cy.blur() // làm mờ yếu tố DOM tập trung
* cy.focus() // tập trung vào một phần tử DOM
* cy.clear() // xóa giá trị đầu vào hoặc khu vực nhập dữ liệu
* cy.uncheck() // bỏ chọn các hộp kiểm tra
* cy.select() // chọn một obtion bên trong một select
* cy.dblclick() // nháy đúp chuột vào một phần tử
* cy.clearCookies () // xóa tất cả cookie của trình duyệt.
* cy.its() // tìm một thuộc tính về chủ đề hiện tại
* cy.and() // nối nhiều khẳng định với nhau

*IV.5.3 Tạo một kịch bản Test*

Sau đây là ví dụ về một kịch bản Test chức năng đăng nhập:

+ Tạo 1file js trong thư mục: Cypress.

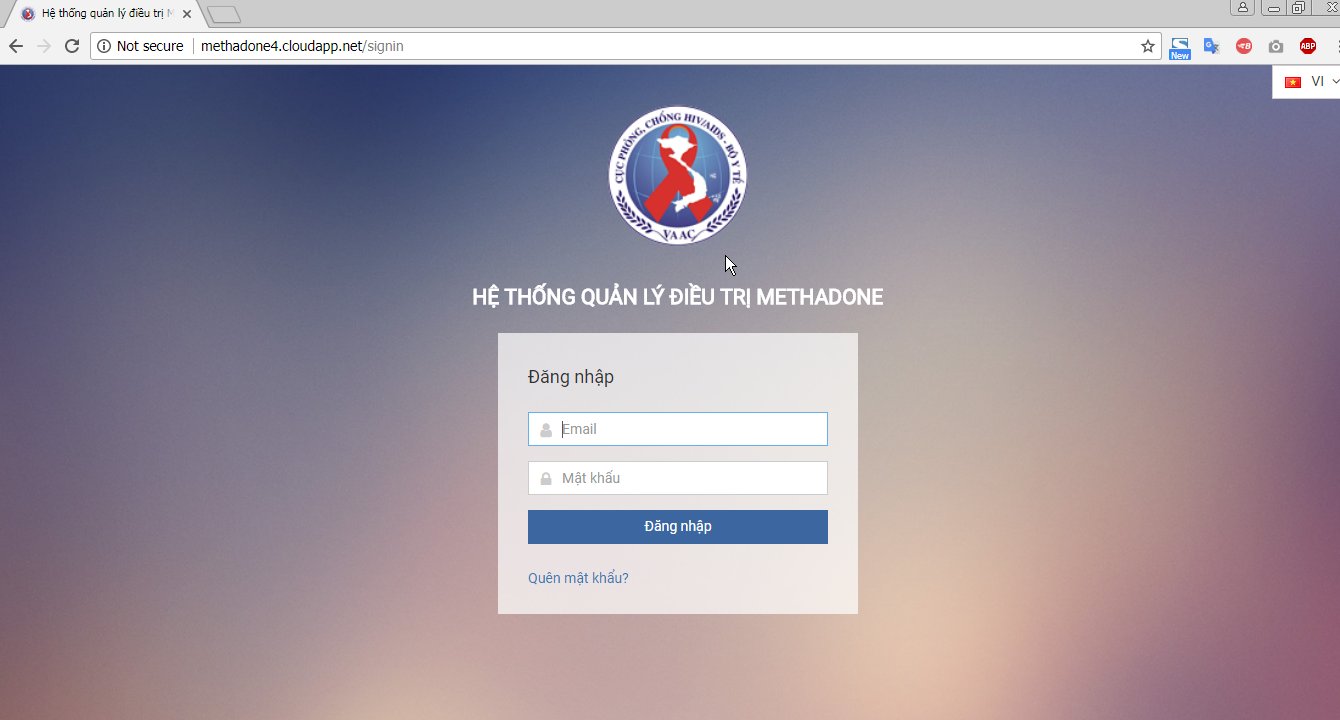
+) Đoạn code tương ứng như sau:

describe(**'Kiểm thử trang đăng nhập'**, **function**() {  
  
 context(**'Form login'**, **function**(){  
 beforeEach(**function**(){  
 cy.visit(**'http://methadone4.cloudapp.net'**)  
 })  
  
 it(**'Không điền email'**, **function**(){  
 cy.get(**'input[name=password]'**).type(**'12345678{enter}'**)  
  
 cy.get(**'.login-form'**)  
 .find(**"div[ng-message]"**)  
 .should(**"contain"**, **'Trường này không được để trống.'**)  
  
 cy.url().should(**'include'**, **'/signin'**)  
 })  
  
 it(**'Không điền mật khẩu'**, **function**(){  
 cy.get(**'input[name=email]'**).type(**'quanly@gmail.com'**)  
 cy.get(**'button'**).click();  
 cy.get(**'.login-form'**)  
 .find(**"div[ng-message]"**)  
 .should(**"contain"**, **'Trường này không được để trống.'**)  
  
 cy.url().should(**'include'**, **'/signin'**)  
 })  
  
 it(**'Không điền mật khẩu vào email'**, **function**(){  
 cy.get(**'button'**).click();  
  
 cy.get(**'.login-form'**)  
 .find(**"div[ng-message]"**).first()  
 .should(**"contain"**, **'Trường này không được để trống.'**)  
  
 cy.get(**'.login-form'**)  
 .find(**"div[ng-message]"**).last()  
 .should(**"contain"**, **'Trường này không được để trống.'**)  
  
 cy.url().should(**'include'**, **'/signin'**)  
 })  
  
 it(**'Email sai định dạng'**, **function**(){  
 cy.get(**'input[name=email]'**).type(**'email'**)  
 cy.get(**'input[name=password]'**).type(**'12345678{enter}'**)  
  
 cy.get(**'.login-form'**)  
 .find(**"div[ng-message]"**)  
 .should(**"contain"**, **'Email không dúng định dạng.'**)  
  
 cy.url().should(**'include'**, **'/signin'**)  
 })  
  
 it(**'Đăng nhập không thành công'**, **function**(){  
 cy.get(**'input[name=email]'**).type(**'quanly@gmail.com'**)  
 cy.get(**'input[name=password]'**).type(**'123{enter}'**)  
  
 cy.url().should(**'include'**, **'/signin'**)  
 })  
  
 it(**'Ðăng nhập thành công'**, **function**(){  
 cy.get(**'input[name=email]'**).type(**'quanly@gmail.com'**)  
 cy.get(**'input[name=password]'**).type(**'12345678{enter}'**)  
  
 cy.url().should(**'include'**, **'/main/doctor/dashboard'**)  
 })  
  
 })  
})

# IV.6 Ứng dụng

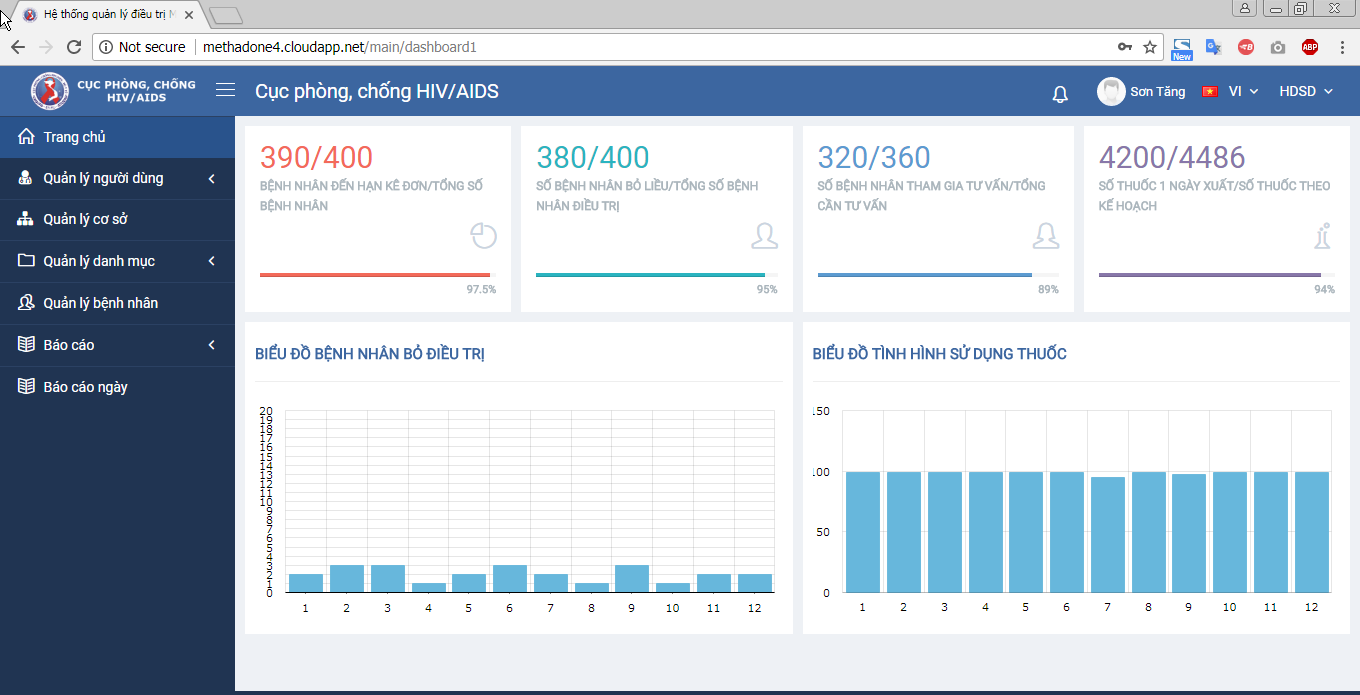
### IV.6.1 Bài toán

*+* Kiểm thử chức năng đăng nhập của ứng dụng Web hệ thống quản lý điều trị thuốc Methadone



*Hình 4.7 Giao diện màn hình đăng nhập*

+ Kiểm thử các common testcase cho các chức năng của hệ thống



*Hình 4.8 Giao diện màn hình quản lý*

### IV.6.2 Lập kế hoạch kiểm thử

###### Giới thiệu

a.1 Mục tiêu

* Bản kế hoạch kiểm thử là tài liệu mô tả về phạm vi, cách tiếp cận, các nguồn lực và kế hoạch thực hiện kiểm thử.
* Thiết lập được mục tiêu ngắn hạn và dài hạn của việc kiểm thử.
* Nhận biết được các rủi ro có thể xảy ra.
* Xác định được cách tiếp cận và kế hoạch cho việc kiểm thử.

a.2 Thông tin chung

Xây dựng dự án Website quản lý điều trị thuốc Methadone nhằm đáp ứng nhu cầu hỗ trợ bệnh nhân điều trị Methadone, giúp họ dễ dàng kết nối và uống thuốc ở các cơ sở điều trị Methadone khác nhau, đồng thời để tăng cường hiệu quả, hiệu lực quản lý trên lĩnh vực này, ứng dụng công nghệ thông tin dựa trên nền tảng trình duyệt Internet là một giải pháp có tính đột phá và mang lại sự thay đổi vượt bậc trong quản lý điều trị Methadone ở nước ta.

Thực hiện tự động hóa các ca kiểm thử cơ bản của ứng dụng

a.3 Nhiệm vụ chính

* Xác định được phạm vi kiểm thử, các rủi ro có thể xảy ra, xác định được mục tiêu kiểm thử để đảm bảo đo đạc được chất lượng phần mềm.

Xác định cách tiếp cận việc kiểm thử, các kĩ thuật được sử dụng, độ bao phủ kiểm thử cho phép.

* Thực thi theo chính sách và chiến lược kiểm thử.
* Xác định nguồn lực, chi phí kiểm thử cần thiết.
* Lên kế hoạch cho hoạt động phân tích, thiết kế, thực thi kiểm thử.
* Xác định tiêu chí kết thúc kiểm thử.
* Liệt kê những kết quả, tài liệu có được sau khi thực hiện kiểm thử.

a.4 Tài liệu liên quan

|  |  |  |  |
| --- | --- | --- | --- |
| STT | Tên tài liệu | Nguồn | Ghi chú |
| 1 | Tài liệu mô tả yêu cầu  Requirements Specification |  |  |
| 2 | Tài liệu mô tả chức năng  Functional Specification |  |  |
| 3 | Tài liệu kế hoạch dự án |  |  |
| 4 | Tài liệu phân tích thiết kế |  |  |
| 5 | Tài liệu hướng dẫn sử dụng |  |  |

a.5 Một số thuật ngữ liên quan

|  |  |  |  |
| --- | --- | --- | --- |
| STT | Tên thuật ngữ | Viết tắt | Ghi chú |
| 1 | Project Manager | PM |  |
| 2 | Test Leader | TL |  |
| 3 | Developer | DV |  |

a.6 Chiến lược kiểm thử

Chiến lược kiểm thử mô tả tất cả các phương pháp tiếp cận tổng thể để kiểm thử. Đối với các chức năng chính hoặc các luồng sự kiện, việc chỉ định phương pháp tiếp cận này sẽ đảm bảo các nhóm chức năng sẽ được kiểm thử đầy đủ và chính xác.

* Chiến lược kiểm thử giới thiệu phương án tiếp cận để test các mục tiêu kiểm thử
* Những vấn đề chính trong chiến lược kiểm thử là các kỹ thuật được áp dụng và điều kiện để biết khi nào việc kiểm thử được hoàn thành.
* Mô tả các kiểu kiểm thử dùng trong dự án.
* Có thể liệt kê với mỗi kiểu kiểm thử tương ứng kiểm thử cho chức năng nào.

b.1 Các kiểu kiểm thử

#### b.1.1 Kiểm thử chức năng (Functional Testing)

- Kiểm thử chức năng (Functional Testing)

|  |  |
| --- | --- |
| **Mục đích test:** | Đảm bảo mục tiêu test đúng đắn của chức năng, bao gồm định hướng, dữ liệu đầu vào, xử lý và dữ liệu nhận được. |
| **Cách thực hiện:** | Thực hiện mỗi UC, chu trình UC hoặc chức năng, sử dụng dữ liệu hợp lệ và không hợp lệ để kiểm tra:  -    Kết quả mong đợi với dữ liệu hợp lệ.  -    Lỗi thích hợp hoặc thông báo hiển thị khi dữ liệu không hợp lệ.  -    Mỗi qui tắc nghiệp vụ đều được áp dụng đúng |
| **Điều kiện hoàn thành:** | -   Toàn bộ kế hoạch test đã được thực hiện.  -     Toàn bộ các lỗi phát hiện ra đã được ghi nhận. |
| **Các vấn đề đặc biệt:** | Xác định hoặc mô tả các vấn đề (nội bộ hoặc bên ngoài) ảnh hưởng đến việc test chức năng |

* Kiểm thử giao diện người sử dụng (User Interface Testing)

|  |  |
| --- | --- |
| **Mục đích test:** | Kiểm tra:  -  Việc sử dụng thông qua mục tiêu test phản ánh đúng các chức năng và yêu cầu nghiệp vụ, bao gồm màn hình đến màn hình, trường đến trường và sử dụng các phương pháp truy cập (phím tabs, di chuột, tổ hợp phím)  -   Các đối tượng và thuộc tính màn hình như menus, size, position, state, và tập tring vào việc tương thích với chuẩn |
| **Cách thực hiện:** | Tạo ra và chỉnh sửa test cho mỗi màn hình để kiểm tra việc sử dụng đúng cách và tình trạng các đối tượng cho mỗi màn hình và đối tượng của ứng dụng |
| **Điều kiện hoàn thành:** | Mỗi màn hình được kiểm tra thành công đúng với phiên bản kiểm tra hoặc phạm vi chấp nhận được |
| **Các vấn đề đặc biệt:** | Không phải toàn bộ các thuộc tính của các đối tượng đều truy cập được |

* Test dữ liệu và tích hợp dữ liệu (Data and Database Integrity Testing)

|  |  |
| --- | --- |
| **Mục đích test:** | Đảm bảo rằng các phương pháp truy cập và chức năng xử lý là đúng và không có sai lệch dữ liệu |
| **Cách thực hiện:** | * Thực hiện từng phương pháp truy cập và xử lý, thử từng trường hợp với dữ liệu hợp lệ và không hợp lệ hoặc các yêu cầu dữ liệu. * Kiểm tra cơ sở dữ liệu để đảm bảo rằng dữ liệu được lưu trữ như mong đợi, toàn bộ các sự kiện với cơ sở dữ liệu xảy ra đều đúng, hợc xem xét các dữ liệu trả về để đảm bảo rằng đã nhận được dữ liệu đúng cho các lý do đúng |
| **Điều kiện hoàn thành:** | Tất cả các phương pháp truy cập và chức năng xử lý đều giống như thiết kế và không có sai lệch dữ liệu |
| **Các vấn đề đặc biệt:** | -    Việc test có thể đòi hỏi phải môi trường phát triển DBMS hoặc driver để truy cập hoặc sửa dữ liệu trực tiếp trong cơ sở dữ liệu.  -   Các xử lý phải được thực hiện bằng tay.  -   Cơ sở dữ liệu có kích thước nhỏ hoặc tối thiểu (giới hạn số bản ghi) phải được dùng để làm rõ thêm các sự kiện không được phép chấp nhận |

* Test chu trình nghiệp vụ (Business Cycle Testing)

|  |  |
| --- | --- |
| **Mục đích test:** | Đảm bảo mục đích của test là đúng đắn và các tiến trình chạy ngầm thực hiện đúng yêu cầu về mô hình nghiệp vụ và lịch trình |
| **Cách thực hiện:** | Việc test sẽ giả lập vài chu trình nghiệp vụ bằng cách thực hiện các công việc sau:  -   Các test dùng cho việc test chức năng sẽ được sửa lại hoặc nâng cấp để tăng số lần mỗi chức năng được thực hiện để giả lập một số người dùng khác nhau trong chu kỳ đã định.  -    Toàn bộ các chức năng theo ngày tháng sẽ được thực hiện với dữ liệu hợp lệ và không hợp lệ hoặc chu kỳ thời gian  -   Toàn bộ các chức năng xảy ra trong lịch trình chu kỳ sẽ được thực hiện vào thời gian thích hợp  -   Việc test sẽ bao gồm cả dữ liệu hợp lệ và không hợp lệ để kiểm tra:     +Kết quả xảy ra khi dữ liệu hợp lệ.        + Lỗi tương tự hoặc cảnh báo hiển thị khi dữ liệu không hợp lệ.  -     Mỗi qui tắc nghiệp vụ đều được áp dụng. |
| **Điều kiện hoàn thành:** | - Toàn bộ kế hoạch test đã được thực hiện.  - Toàn bộ các lỗi phát hiện ra đều được ghi nhận |
| **Các vấn đề đặc biệt:** | -   Ngày và các sự kiện của hệ thống có thể đòi hỏi các hoạt động hỗ trợ đặc biệt  -   Mô hình nghiệp vụ đòi hỏi xác định các yêu cầu và thủ tục test thích hợp |

#### b.1.2 Kiểm thử hiệu suất (Performance testing)

- Performance Profiling

|  |  |
| --- | --- |
| **Mục đích test:** | Kiểm tra các biểu hiện về hiệu suất cho các giao dịch hoặc chức năng nghiệp vụ đã thiết kế theo những điều kiện sau:  -    workload bình thường đã biết trước (normal anticipated workload)  -    workload xấu đã biết trước (anticipated worst case workload) |
| **Cách thực hiện:** | -    Sử dụng các thủ tục test cho test chức năng và chu trình nghiệp vụ  -    Chỉnh sửa file dữ liệu để tăng số lượng các giao dịch hoặc scripts để tăng số tương tác xảy ra trong mỗi giao dịch  -    Scripts phải được chạy trên một máy (trường hợp tốt nhất để đánh giá người dùng đơn lẻ, giao dịch đơn lẻ) và phải lặp lại trên nhiều máy trạm (ảo hoặc thực, xem các vấn đề đặc biệt dưới đây) |
| **Điều kiện hoàn thành:** | -    Giao dịch đơn lẻ hoặc người dùng đơn lẻ: Thực hiện thành công test script không có lỗi và trong phạm vi mong đợi hoặc thời gian phản hồi cho mỗi giao dịch  -    Nhiều giao dịch hoặc nhiều người dùng: Thực hiện thành công test script không có lỗi và trong thời gian chấp nhận được |
| **Các vấn đề đặc biệt:** | Việc test hiệu suất toàn diện bao gồm phải có một workload nền trên máy chủ.  Có một số phương pháp để thực hiện, bao gồm:  -   “Drive transactions” trực tiếp đến máy chủ, thường trong các form gọi SQL.  -    Tạo các người dùng ảo để giả lập nhiều máy trạm, thường là vài trăm. Sử dụng công cụ Remote Terminal Emulation để thực hiện việc load này, kỹ thuật này còn được dùng để load giao dịch trên mạng  -    Sử dụng nhiều người dùng, mỗi người chạy một test script để load lên hệ thống  Test hiệu suất phải được thực hiện trên máy chuyên dụng hoặc thời gian chuyên dùng. Điều đó cho phép việc tính toán được đầy đủ và chính xác.  Cơ sở dữ liệu sử dụng để test hiệu suất phải có kích thước thực tế hoặc đo bằng nhau |

* Load Testing

|  |  |
| --- | --- |
| **Mục tiêu test:** | Kiểm tra hiệu suất về thời gian cho các giao dịch hoặc tình huống nghiệp vụ đã thiết kế với nhiều điều kiện workload |
| **Cách thực hiện:** | -    Sử dụng các test đã xây dựng cho test chức năng và chu trình nghiệp vụ.  -    Sửa lại file dữ liệu để tăng số lượng giao dịch hoặc test nhằm tăng thêm số lần thực hiện mỗi giao dịch |
| **Điều kiện hoàn thành:** | Nhiều giao dịch hoặc nhiều người dùng: Thực hiện thành công việc test không có lỗi và trong thời gian chấp nhận được |
| **Các vấn đề đặc biệt:** | -    Load testing phải được thực hiện trên máy chuyên dụng hoặc vào những giờ chuyên biệt. Nó cho phép đo đạc đầy đủ và chính xác.  -    Cơ sở dữ liệu dùng cho load testing phải có kích thước thực tế hoặc đo bằng nhau |

* Stress Testing

|  |  |
| --- | --- |
| **Mục đích test:** | Kiểm tra các chức năng của mục đích test là đúng đắn và không có lỗi với những điều kiện sau:  -    Có ít hoặc không có bộ nhớ phù hợp trên máy chủ (RAM và DASD)  -    Số lượng máy trạm tối đa trong thực tế hoặc giả lập kết nối vào máy chủ  -    Nhiều người dùng thực hiện cùng một giao dịch với cùng dữ liệu hoặc account  -    Độ lớn các giao dịch xấu hoặc hỗn hợp (xem phần Performance Testing ở trên).  Chú ý: Mục đích của Stress Testing có thể được phát biểu rõ và ghi ra các điều kiện mà hệ thống có thể lỗi, không thể tiếp tục thực hiện các chức năng một cách thích hợp |
| **Cách thực hiện:** | -    Sử dụng các test đã xây dựng để thực hiện Performance Profiling hoặc Load Testing.  -    Để test việc hạn chế tài nguyên, test phải chạy trên máy đơn lẻ và RAM và DASD trên máy chủ phải giảm đi hoặc hạn chế  -    Để thực hiện các stress tests khác phải sử dụng nhiều người dùng cùng chạy một TC hoặc bổ sung các test để thực hiện độ lớn giao dịch xấy hoặc hỗn hợp. |
| **Điều kiện hoàn thành:** | Toàn bộ kế hoạch test được thực hiện và các hạn chế của hệ thống được xác định thỏa mãn các điều kiện tối thiểu đã đặt ra hoặc chỉ sai trong trong hợp các điều kiện không nằm trong điều kiện đã xác định |
| **Các vấn đề đặc biệt:** | -    Việc test Stressing mạng có thể đòi hỏi những công cụ để load mạng với nhiều thông báo hoặc gói dữ liệu.  -    DASD dùng cho hệ thống phải tạm thời giảm xuống để hạn chế khả năng chỗ trống cho tăng trưởng cơ sở dữ liệu.  -    Đồng bộ hóa các máy trạm đồng thời truy cập vào cùng một bản ghi hoặc các account dữ liệu |

* Volume Testing

|  |  |
| --- | --- |
| **Mục đích test:** | Kiểm tra xem mục tiêu test có thực hiện thành công các chức năng theo những điều kiện sau không:  -    Số máy trạm lớn nhất kết nối (thực tế hoặc vật lý – có thể), hoặc giả lập, tất cả đều thực hiện cùng một chức năng nghiệp vụ trong một chu kỳ mở rộng.  -    Kích thước cơ sở dữ liệu lớn nhất có thể (thực tế hoặc đo được) và nhiều query hoặc giao dịch báo cáo được thực hiện đồng thời. |
| **Cách thực hiện:** | -    Sử dụng các test đã xây dựng cho Performance Profiling hoặc Load Testing.  -    Có thể dùng nhiều người dùng, chạy cùng một test hoặc bổ sung các test để thực hiện trường hợp giao dịch volume hoặc hỗn hợp xấu nhất (xem Stress Testing ở trên) trong một chu kỳ mở rộng.  -    Tạo ra cơ sở dữ liệu lớn nhất (thực tế, qui đổi, hoặc lọc các dữ liệu đại diện) và nhiều người dùng chạy các query và giao dịch báo cáo đồng thời trong một chu kỳ mở rộng |
| **Điều kiện hoàn thành:** | -    Toàn bộ kế hoạch test được thực hiện và các giới hạn của hệ thống được xác định là đạt tới hoặc xử lý mà không có lỗi |
| **Các vấn đề đặc biệt:** | Chu kỳ thời gian như thế nào là chấp nhận được cho điều kiện cơ sở dữ liệu lớn, như đã nói ở trên |

#### b.1.3 Kiểm thử Bảo mật và kiểm soát truy cập (Security and Access Control Testing)

|  |  |
| --- | --- |
| **Mục đích test:** | -         Bảo mật mức ứng dụng: Đảm bảo rằng một người dùng chỉ có thể truy cập vào những chức năng hoặc dữ liệu mà nhóm người dùng đó được phép  -         Bảo mật mức hệ thống: Đảm bảo rằng chỉ những người được phép truy cập hệ thống và ứng dụng được phép truy cập chúng |
| **Cách thực hiện:** | -         Bảo mật ứng dụng: Xác định và liệt kê từng nhóm người dùng và các chức năng hoặc dữ liệu mà họ được phép truy cập  -    Tạo test case cho mỗi nhóm người dùng và kiểm tra từng quyền bằng cách tạo các giao dịch xác định cho mỗi nhóm  -    Sửa lại nhóm người dùng và chạy lại tình huống test cho cùng những người dùng. Với mỗi trường hợp, kiểm tra các chức năng thêm vào hoặc dữ liệu có đúng không hay bị từ chối.  -         Truy cập mức hệ thống: tham khảo các điều kiện đặc biệt dưới đây |
| **Điều kiện hoàn thành:** | Với mỗi nhóm người dùng đều có các chức năng hoặc dữ liệu thích hợp, và toàn bộ các chức năng giao dịch đều như dự kiến và chạy trong các test chức năng ứng dụng trước đó |
| **Các vấn đề đặc biệt:** | Truy cập vào hệ thống phải được xem xét hoặc thảo luận với quản trị hệ thống hoặc quản trị mạng, có thể không cần nếu nó là chức năng của quản trị mạng hoặc quản trị hệ thống |

#### b.1.4 Kiểm thử hồi qui (Regression Testing)

|  |  |
| --- | --- |
| **Mục đích test:** | Test hồi qui dùng để kiểm tra các phần được sửa chữa trong phần mềm, để đảm bảo rằng những sự thay đổi đó không gây ra lỗi trong những phần khác |
| **Cách thực hiện:** | -    Tái sử dụng các TC từ những phần test trước để test các module đã được sửa chữa  -   80% các TC được chọn ngẫu nhiên  - Xây dựng một chương trình phân tích sơ sở hạ tầng. Chúng ta dựng một cơ sở hạ tầng có thể mở rộng được để thực hiện và đánh giá chương trình phân tích. Dựa vào kết quả phân tích chúng ta xác định phạm vi cần test hồi qui. |
| **Điều kiện hoàn thành:** | -    Toàn bộ các TC được thực hiện và đạt yêu cầu  -    Toàn bộ các TC được chọn được thực hiện và đạt yêu cầu |

*b.2 Các cấp độ kiểm thử*

b.2.1 Kiểm thử đơn vị (Unit Testing)

- Kiểm thử đơn vị nhằm kiểm tra đơn vị thiết kế nhỏ nhất, một modul phần mềm. Một modul hoạt động thường có trao đổi thông tin với module mức dưới và mức trên nó, do đó phạm vi phát hiện lỗi liên quan chặt chẽ tới module này.

- Người tiến hành kiểm thử đơn vị: lập trình viên cùng nhóm của mình.

- Kỹ thuật kiểm thử đơn vị: chủ yếu là hộp trắng, có thể sử dụng thêm kỹ thuật kiểm thử hộp đen.

b.2.2 Kiểm thử tích hợp (Integration Testing)

**-** Kiểm thử tích hợp nhằm nhận biết được một bộ phận chức năng hay hệ thống hoạt động có tốt hay không.

**-** Là một kỹ thuật có tính hệ thống để xây dựng cấu trúc của chương trình.

**-** Từ các module đã qua kiểm thử đơn vị, xây dựng cấu trúc chương trình đảm bảo tuân theo thiết kế.

**-** Có 2 cách tích hợp:

**+** Tích hợp từng bước: Có 3 chiến lược:

* Tích hợp từ dưới lên (bottom-up testing)
* Tích hợp từ trên xuống (top-down testing )
* Kết hợp 2 chiến lược (sandwich testing)

+ Tích hợp đồng thời: Kiểm thử vụ nổ lớn (big bag testing)

b.2.3 Kiểm thử hệ thống (System Testing)

- Tìm kiếm các lỗi, nhưng trọng tâm là đánh giá về hoạt động, thao tác, sự tin cậy, và các yêu cầu khác liên quan đến chất lượng toàn hệ thống.

- Đặc biệt thích hợp cho việc phát hiện lỗi giao tiếp với phần mềm và phần cứng bên ngoài, chẳng hạn các lỗi “tắc nghẽn” (deadlock) hoặc chiếm dụng bộ nhớ.

- Đòi hỏi nhiều thời gian, công sức, thiết bị.

- Mục đích: Kiểm thử thiết kế và toàn bộ hệ thống(sau khi tích hợp) có thỏa mãn yêu cầu đặt ra hay không.

- Phương pháp kiểm thử: kiểm thử hộp đen.

- Người thực hiện: kiểm thử viên.

- Khi nào có thể thực hiện kiểm thử hệ thống:

+ Hệ thống cần kiểm thử đã hoàn thiện.

+ Kiểm thử tích hợp và đơn vị đã hoàn thành.

+ Sản phm được tích hợp đúng thiết kế.

+ Các tài liệu đặc tả đã là bản cuối cùng.

+ Các tài liệu hỗ trợ kiểm thử Test Plan, Test case đã hoàn thành.

b.2.4 Kiểm thử chấp nhận (Acceptance Testing)

**-** Kiểm thử chấp nhận vận hành hệ thống trong môi trường của người sử dụng.

- Kiểm thử Alpha:

* Người dùng thực hiện với số liệu giả lập.
* Được thực hiện trong môi trường người phát triển.

- Kiểm thử Beta:

* Người dùng thực hiện với số liệu thực.
* Được thực hiện trong môi trường ứng dụng thực.

*b.3 Các công cụ kiểm thử*

Các công cụ sẽ áp dụng cho dự án

|  |  |  |  |
| --- | --- | --- | --- |
| **Mục đích** | **Công cụ** | **Nhà cung cấp/Tự xây dựng** | **Phiên bản** |
| Quản lý Test Case | Microsoft Office Excel | Microsoft | 2010 |
| Quản lý Configuration | Microsoft Office Word | Microsoft | 2010 |
| Defect Tracking | Microsoft Office Excel | Microsoft | 2010 |
| Quản lý tiến độ kiểm thử | Microsoft Project Professional | Microsoft | 2010 |
| Tool test tự động | Cypress |  |  |

*b.4 Môi trường kiểm thử*

Các yếu tố về môi trường:

- Kiểm thử ứng dụng Quản lý bệnh nhân sử dụng Methadone trên trình duyệt Chrome

###### Tài nguyên

+ Nhân lực

|  |  |
| --- | --- |
| Vị trí | Trách nhiệm/ghi chú |
| Quản lý kiểm thử | Quản lý hoạt động kiểm thử.   * + Hướng dẫn kỹ thuật   + Sử dụng và quản lý nguồn lực   + Báo cáo quản lý |
| Nhân viên kiểm thử | Thực hiện việc kiểm thử.  Nhiệm vụ:   * + Tiến hành kiểm thử   + Viết các ghi chú kết quả kiểm thử (Test Logs)   + Viết tài liệu báo cáo kiểm thử |
| Quản trị hệ thống | Đảm bảo môi trường hệ thống để kiểm thử.  Nhiệm vụ:   * + Phối hợp kiểm tra hệ thống môi trường (máy chủ...)   + Cài đặt các ứng dụng cần thiết để kiểm thử hệ thống   + Báo cáo tình trạng hệ thống |

Các mốc kiểm soát các giai đoạn kiểm thử (Test Milestones)

|  |  |  |  |
| --- | --- | --- | --- |
| **Milestone Name** | **Nguồn lực** | **Ngày bắt đầu** | **Ngày kết thúc** |
| Lập kế hoạch kiểm thử | TL, Tester |  |  |
| Cập nhật Kế hoạch kiểm thử | PM, TL, Tester |  |  |
| Thiết kế Test case | Tester |  |  |
| Cập nhật Test case | Tester |  |  |
| Thực thi Test case | Tester |  |  |
| Log lỗi | Tester |  |  |
| Báo cáo | Tester |  |  |

###### Các sản phẩm

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **STT** | **Sản phẩm** | **Ngày bàn giao** | **Người bàn giao** | **Người nhận** |
| 1 | Test Plan |  | TL, Tester | PM |
| 2 | Unit Test case |  | DV | TL |
| 3 | Integration Test Cases |  | Tester | TL |
| 4 | System Test case |  | Tester | TL |
| 5 | Defect log |  | Tester | TL, DV |
| 6 | Test reports |  | Tester | TL, PM |

### IV.6.3 Kiểm thử đơn vị:

###### **a. Đăng nhập**

Đang bổ sung

*Đánh giá kiểm thử*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **TEST REPORT** | | | | | |
|  |  |  |  |  |  |  | |
| **Tên dự án** | *Website quản lý điều trị Methadone* | | **Tester** | | **VuThiThuy** |
| **Mã dự án** |  | | **Trưởng nhóm** | |  |
| **Mã tài liệu** |  | | **Ngày báo cáo** | | **24/03/2018** |
| **Ghi chú** |  | | | | |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
| **No** | **Module code** | **Pass** | **Fail** | **Untested** | **Number of test cases** |  |
| 1 | Đăng nhập | 5 | 0 | 0 | 5 |  |
| 2 | Liên hệ | 8 | 0 | 0 | 8 |  |
|  | **Sub total** | 13 | 0 | 0 | 13 |  |
|  |  |  |  |  |  |  |
|  | **Test coverage** |  | **100** | % |  |  |
|  | **Test successful coverage** | | **100** | % |  |  |

# PHẦN III. KẾT LUẬN

Sau quá trình tìm hiểu về công cụ Cypress thì em nhận ra được những hữu ích thật lớn của công cụ hỗ trợ cho việc kiểm tra, đã giúp cho toàn bộ quá trình kiểm tra trở nên đơn giản, nhanh chóng, tiện lợi hơn. Môi trường Cypress là một môi trường làm việc chuyên nghiệp, những hỗ trợ của nó cũng rất hoàn thiện và càng ngày thì phần mềm Cypress càng cải tiến thêm nhiều tính năng giúp cho những kỹ thuật viên kiểm tra thực hiện công việc dễ dàng và việc làm việc theo nhóm lớn cũng thống nhất hơn, đồng bộ hơn.

Qua đề tài đồ án tốt nghiệp này em đã phần nào hiểu rõ hơn về tổng quan kiểm thử, kiểm thử tự động và tìm hiểu về công cụ kiểm thử tự động Cypress. Do thời gian tìm hiểu và nghiên cứu có hạn và chưa có nhiều kinh nghiệm trong triển khai dự án, nên bài báo cáo không thể tránh khỏi những sai sót mong thầy cô và các bạn có thể đóng góp ý kiến giúp em hoàn thiện báo cáo của mình hơn.

Em xin chân thành cảm ơn thầy giáo ***PGS.TS Trương Anh Hoàng*** đã tận tình giúp đỡ em hoàn thành được báo cáo này.

# TÀI LIỆU THAM KHẢO

Website

[1] <http://www.testingvn.com>

[2] <http://www.guru99.com>

[3] <http://toolsqa.com>

[4] <http://www.>cypress.io