TRƯỜNG ĐẠI HỌC CMC

KHOA CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG



KIỂM THỬ PHẦN MỀM

**Software Testing Document (STD)**

**ĐỀ TÀI: KIỂM THỬ WEB QUẢN LÝ KÝ TÚC XÁ**

|  |  |
| --- | --- |
| Sinh viên: | Nguyễn Đình Việt ( BIT220169) |
|  | Nguyễn Sơn Tùng (BIT220166) |
|  | Nguyễn Minh Huy (BIT220076) |
|  | Nguyễn Văn Thiện (BIT220234)  Trần Mai Trung (BIT220242) |
| Lớp: | 22SE1.2 |
| Giảng viên: | Vũ Việt Vũ |

Hà Nội, năm 2025

Theo dõi phiên bản tài liệu

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Ngày |  | Phiên bản | Lý do thay đổi | Mô tả thay đổi | Tác giả |
| 1/3/2025 |  | 1.0 |  | Mô tả yêu cầu, nghiệp vụ phần mềm |  |
|  |  |  |  |  |  |

Contents

[1. Giới thiệu 4](#_Toc194317451)

[1.1. Mục tiêu 4](#_Toc194317452)

[1.2. Phạm vi kiểm thử 4](#_Toc194317453)

[2. Chiến lược kiểm thử 4](#_Toc194317454)

[2.1.Các kỹ thuật kiểm thử được áp dụng 4](#_Toc194317455)

[2.1.1. Kiểm thử hộp trắng (White-box Testing) 4](#_Toc194317456)

[2.1.2.Kiểm thử hộp đen (Black-box Testing) 5](#_Toc194317457)

[2.2. Phương pháp kiểm thử 5](#_Toc194317458)

[2.3. Môi trường kiểm thử 5](#_Toc194317459)

[3. Các thành phần kiểm thử 5](#_Toc194317460)

[3.1. Các chức năng chính được kiểm thử 5](#_Toc194317461)

[3.2. Các kịch bản nghiệp vụ (Use Case) 6](#_Toc194317462)

[4. Lịch trình kiểm thử 6](#_Toc194317463)

[5. Tiêu chí chấp nhận 7](#_Toc194317464)

[5.1. Tiêu chí hoàn thành kiểm thử (Exit Criteria) 7](#_Toc194317465)

[5.2. Tiêu chí từ chối kiểm thử (Suspension Criteria) 7](#_Toc194317466)

[6. Rủi ro và biện pháp giảm thiểu 7](#_Toc194317467)

[6.1. Rủi ro 7](#_Toc194317468)

[6.2. Biện pháp giảm thiểu 7](#_Toc194317469)

[7. Báo cáo và theo dõi 8](#_Toc194317470)

MỞ ĐẦU

Với sự phát triển nhanh chóng của công nghệ thông tin nói chung và công nghệ phần mềm nói riêng, việc phát triển phần mềm ngày càng được hỗ trợ bởi nhiều công cụ tiên tiến, giúp cho việc xây dựng phần mềm đỡ mệt nhọc và hiệu quả hơn. Tuy nhiên, vì độ phức tạp của phần mềm và những giới hạn về thời gian và chi phí, cho dù các hoạt động đảm bảo chất lượng phần mềm nói chung và kiểm thử nói riêng ngày càng chặt chẽ và khoa học, vẫn không đảm bảo được rằng các sản phẩm phần mềm đang được ứng dụng không có lỗi. Lỗi vẫn luôn tiềm ẩn trong mọi sản phẩm phần mềm và cũng có thể gây những thiệt hại khôn lường.

Kiểm thử phần mềm là một quá trình liên tục, xuyên suốt mọi giai đoạn phát triển phần mềm để đảm bảo rằng phần mềm thoả mãn các yêu cầu thiết kế và các yêu cầu đó đáp ứng các nhu cầu của người dùng. Các kỹ thuật kiểm thử phần mềm đã và đang được nghiên cứu, và việc kiểm thử phần mềm đã trở thành quy trình bắt buộc trong các dự án phát triển phần mềm trên thế giới. Kiểm thử phần mềm là một hoạt động rất tốn kém, mất thời gian, và khó phát hiện được hết lỗi. Vì vậy, việc kiểm thử phần mềm đòi hỏi phải có chiến lược phù hợp, một kế hoạch hợp lý và việc thực hiện được quản lí chặt chẽ.

Và với những trang web ngày càng được sử dụng với nhiều hơn nhằm đáp ứng như cầu của mỗi người dùng. Từ một trang web được lập trình và thiết kế các chức năng theo yêu cầu của khách hàng. Từ đây lại đặt ra một vấn đề hiển nhiên là kiểm thử các chức năng trên trang web này có đáp ứng được các yêu cầu đề ra ban đầu hay không trước khi trao sản phẩm tới tay khách hàng. Dó là lý do nhóm em chọn đề tài “Kiểm thử website quản lý ký túc xá” làm đề tài báo cáo.

# 1. Giới thiệu

## 1.1. Mục tiêu

* Xác nhận rằng hệ thống Quản Lý Ký Túc Xá hoạt động đúng theo các yêu cầu chức năng và phi chức năng đã được mô tả trong SRS.
* Đảm bảo các tính năng như đăng nhập/đăng xuất, quản lý tài khoản, quản lý sinh viên, hợp đồng, hóa đơn, dịch vụ, phòng và báo cáo được kiểm thử đầy đủ.
* Phát hiện sớm các lỗi nghiêm trọng (Critical, High) để giảm thiểu rủi ro khi triển khai hệ thống.

## 1.2. Phạm vi kiểm thử

* **Kiểm thử chức năng:** Đảm bảo các use case chính (đăng nhập, tạo tài khoản, quản lý sinh viên, hợp đồng, hóa đơn, phòng, dịch vụ, thanh toán, báo cáo) hoạt động đúng.
* **Kiểm thử giao diện:** Đối với giao diện web và ứng dụng (nếu có) nhằm đảm bảo tính thân thiện, dễ sử dụng.
* **Kiểm thử hiệu năng:** Đảm bảo hệ thống xử lý đúng yêu cầu về tốc độ phản hồi và tải của người dùng.
* **Kiểm thử tích hợp:** Đảm bảo các module liên quan (User, Sinh viên, Hợp đồng, Hóa đơn, Phòng, Dịch vụ) tích hợp và giao tiếp với nhau một cách chính xác.

# 2. Chiến lược kiểm thử

## 2.1.Các kỹ thuật kiểm thử được áp dụng

Trong quá trình phát triển phần mềm, kiểm thử đóng vai trò quan trọng để đảm bảo chất lượng và độ tin cậy của hệ thống. Có nhiều phương pháp kiểm thử phần mềm, trong đó hai phương pháp phổ biến nhất là **kiểm thử hộp trắng** và **kiểm thử hộp đen.**

### ****2.1.1. Kiểm thử hộp trắng (White-box Testing)****

Kiểm thử hộp trắng là phương pháp kiểm thử dựa trên cấu trúc bên trong của phần mềm. Người kiểm thử cần hiểu rõ mã nguồn, thuật toán và luồng xử lý của chương trình để thiết kế các trường hợp kiểm thử hiệu quả.

**Đặc điểm của kiểm thử hộp trắng:**

* Dựa trên mã nguồn và logic bên trong của phần mềm.
* Kiểm thử mức độ bao phủ mã (Code Coverage) như: kiểm thử đường dẫn, điều kiện, vòng lặp...
* Được thực hiện bởi các nhà phát triển phần mềm hoặc tester có hiểu biết về lập trình.

Trong dự án này nhóm quyết định kiểm thử hộp trắng với **kỹ thuật phân nhánh**

**Kiểm thử phân nhánh** là một kỹ thuật kiểm thử hộp trắng nhằm kiểm tra tất cả các nhánh điều kiện (if-else, switch-case, vòng lặp) trong chương trình. Mục tiêu chính của phương pháp này là đảm bảo rằng mỗi nhánh có thể xảy ra trong mã nguồn đều được thực thi ít nhất một lần.

Cách thực hiện kiểm thử phân nhánh:

* Xác định các nhánh điều kiện: Phân tích mã nguồn để tìm các lệnh rẽ nhánh như if, else, switch, while, for.
* Viết test case tương ứng: Đảm bảo rằng có đủ test case để kiểm tra từng nhánh. Ví dụ, nếu có một câu lệnh if-else, cần có ít nhất 2 test case: một cho nhánh if và một cho nhánh else.
* Chạy thử và kiểm tra bao phủ mã: Xác minh rằng tất cả các nhánh đều đã được kiểm thử bằng cách sử dụng công cụ đo độ bao phủ mã (Code Coverage).

Ưu điểm:

* Đảm bảo tất cả các nhánh của chương trình đều được kiểm thử.
* Giúp phát hiện lỗi logic trong điều kiện rẽ nhánh.

Nhược điểm:

* Không đảm bảo kiểm thử hết tất cả tổ hợp dữ liệu đầu vào.
* Cần có hiểu biết về mã nguồn để thiết kế test case hiệu quả.

### ****2.1.2.Kiểm thử hộp đen (Black-box Testing)****

Kiểm thử hộp đen là phương pháp kiểm thử dựa trên đầu vào và đầu ra của hệ thống mà không cần quan tâm đến mã nguồn hay logic bên trong.

**Đặc điểm của kiểm thử hộp đen:**

* Dựa vào yêu cầu và chức năng của phần mềm.
* Không yêu cầu kiến thức về mã nguồn.
* Được thực hiện bởi tester hoặc người dùng cuối.

Ở dự án này, nhóm chọn kiểm thử hộp đen bằng phương pháp kiểm thử khám phá và đoán lỗi

**Kiểm thử khám phá (Exploratory Testing)** và **đoán lỗi (Error Guessing)** đều là những kỹ thuật kiểm thử không dựa trên các test case được viết sẵn mà tận dụng kinh nghiệm và trực giác của tester.

**Kiểm thử khám phá (Exploratory Testing)**  
Tester tự do khám phá phần mềm, tìm kiếm lỗi thông qua việc tương tác trực tiếp và quan sát hành vi của hệ thống. Quá trình kiểm thử diễn ra song song với việc thiết kế test case, cho phép điều chỉnh chiến lược kiểm thử theo thời gian thực dựa trên những phát hiện ban đầu.

**Ưu điểm:**

* + Cho phép phát hiện các lỗi khó lường, không được dự đoán trước.
  + Linh hoạt và thích ứng nhanh với các thay đổi của phần mềm.
  + Khai thác được kinh nghiệm và sự nhạy bén của tester.

**Nhược điểm:**

* + Kết quả kiểm thử có thể khó tái tạo do thiếu tính hệ thống.
  + Đánh giá mức độ bao phủ của kiểm thử khó xác định so với kiểm thử có kịch bản cụ thể.

### Đoán lỗi (Error Guessing) Dựa trên kinh nghiệm và kiến thức về phần mềm, tester đưa ra các kịch bản kiểm thử nhằm "đoán" những khu vực có khả năng phát sinh lỗi. Đây là phương pháp dự đoán lỗi mà không có hướng dẫn hay tài liệu kiểm thử cụ thể.

**Ưu điểm:**

* Giúp phát hiện những lỗi có tính chất đặc thù, thường không được phát hiện qua các kỹ thuật kiểm thử tiêu chuẩn.
* Tận dụng được kinh nghiệm của tester trong việc nhận biết các mẫu lỗi phổ biến.

**Nhược điểm:**

* Phụ thuộc nhiều vào kinh nghiệm cá nhân, do đó kết quả có thể không nhất quán giữa các tester khác nhau.
* Không có một phương pháp đo lường cụ thể cho mức độ bao phủ của các kịch bản "đoán" lỗi.

## 2.2. Phương pháp kiểm thử

* **Unit Testing:** Sử dụng JUnit và Mockito để kiểm thử từng module riêng lẻ (ví dụ: UserService, StudentService, ContractService,...).
* **Functional Testing:** Sử dụng Postman hoặc các công cụ tự động (ví dụ: Selenium cho giao diện web) để mô phỏng các kịch bản nghiệp vụ (use case) theo tài liệu SRS.
* **Performance Testing:** Sử dụng JMeter hoặc để kiểm tra khả năng chịu tải của hệ thống.
* **Kiểm thử end-to-end (E2E Testing):** Mô phỏng luồng nghiệp vụ thực tế từ giao diện người dùng cho tới backend, giúp đảm bảo các use case hoàn chỉnh hoạt động như mong đợi.

## 2.3. Môi trường kiểm thử

* **Máy chủ kiểm thử:** Sử dụng môi trường staging cài đặt hệ điều hành Windows với MySQL, Java SpringBoot (server) và React/React Native (client) như đã được mô tả.
* **Môi trường mạng:** Kiểm thử trên mạng nội bộ localhost.
* **Công cụ kiểm thử:** JUnit, Mockito, Spring Boot Test, Postman, JMeter, Selenium.

# 3. Các thành phần kiểm thử

## 3.1. Các chức năng chính được kiểm thử

* **Đăng nhập/Đăng xuất:** Kiểm tra quy trình xác thực và hủy phiên.
* **Quản lý tài khoản:** Tạo, cập nhật, vô hiệu tài khoản.
* **Quản lý thông tin cá nhân:** Cập nhật thông tin, đổi mật khẩu.
* **Quản lý sinh viên:** Xếp phòng, tìm kiếm sinh viên.
* **Quản lý hợp đồng:** Tạo, tìm kiếm, cập nhật hợp đồng.
* **Quản lý hóa đơn:** Tạo, tìm kiếm, in hóa đơn.
* **Quản lý dịch vụ:** Thêm, cập nhật, vô hiệu dịch vụ.
* **Quản lý phòng:** Thêm, cập nhật, xóa phòng, giao phòng cho trực ban.
* **Thanh toán:** Xem, thanh toán hóa đơn.
* **Báo cáo & thống kê:** Xem báo cáo tổng quan, số sinh viên, số tiền nợ,...

## 3.2. Các kịch bản nghiệp vụ (Use Case)

Mỗi Use Case được kiểm thử theo các bước sau:

* **Kịch bản chính (happy path):** Kiểm tra luồng hoạt động chính (ví dụ: đăng nhập thành công, tạo tài khoản với dữ liệu hợp lệ).
* **Kịch bản ngoại lệ:** Kiểm tra các điều kiện lỗi như nhập sai thông tin, dữ liệu không hợp lệ, thiếu quyền truy cập,...
* **Kịch bản mở rộng:** Các nhánh “extend” trong use case (ví dụ: đổi mật khẩu khi mật khẩu cũ không hợp lệ).

# 4. Lịch trình kiểm thử

* **Giai đoạn 1: Chuẩn bị môi trường kiểm thử**

Cài đặt máy chủ, cấu hình cơ sở dữ liệu, triển khai ứng dụng trên môi trường staging.  
Thời gian: 1 tuần.

* **Giai đoạn 2: Viết và thực thi Unit Test, Integration Test**  
  Phát triển test case cho từng module, chạy kiểm thử tự động qua CI/CD.  
  Thời gian: 2 tuần.
* **Giai đoạn 3: Kiểm thử chức năng và giao diện**  
  Sử dụng Postman, Selenium để mô phỏng các use case nghiệp vụ theo SRS.  
  Thời gian: 2 tuần.
* **Giai đoạn 4: Kiểm thử hiệu năng**  
  Sử dụng JMeter, để kiểm tra hệ thống dưới tải.  
  Thời gian: 1 tuần.
* **Giai đoạn 5: Tổng hợp kết quả, báo cáo và sửa lỗi:**  
  Phân tích báo cáo kiểm thử, ưu tiên sửa các lỗi nghiêm trọng và tái kiểm thử.  
  Thời gian: 1 tuần.

# 5. Tiêu chí chấp nhận

## 5.1. Tiêu chí hoàn thành kiểm thử (Exit Criteria)

* **Unit Test Coverage:** Ít nhất 70% (theo yêu cầu của dự án) cho các module Service.
* **Integration Test:** Tất cả các kịch bản nghiệp vụ chính phải PASS.
* **Functional Test:** Tất cả các use case được mô tả trong SRS phải được thực hiện và đạt kết quả mong đợi.
* **Performance:** Hệ thống phải đáp ứng yêu cầu về thời gian phản hồi dưới 2 giây trong điều kiện tải bình thường.
* **Bảo mật:** Không có lỗ hổng bảo mật nghiêm trọng (Critical).

## 5.2. Tiêu chí từ chối kiểm thử (Suspension Criteria)

* Nếu hệ thống gặp lỗi nghiêm trọng (Critical) khiến không thể tiếp tục kiểm thử các chức năng khác.
* Nếu môi trường kiểm thử không ổn định hoặc không thể mô phỏng đúng môi trường triển khai thực tế.

# 6. Rủi ro và biện pháp giảm thiểu

## 6.1. Rủi ro

* **Thay đổi yêu cầu:** Nếu có thay đổi lớn trong SRS giữa chừng, cần cập nhật test plan kịp thời.
* **Thiếu tài nguyên:** Đội ngũ kiểm thử chưa đủ kinh nghiệm hoặc thiếu thời gian.
* **Tích hợp bên thứ ba:** Dịch vụ thanh toán hoặc hệ thống bên ngoài không ổn định ảnh hưởng đến kiểm thử tích hợp.
* **Ứng dụng chưa hoàn thiện:** Một số chức năng của ứng dụng chưa hoàn thiện khiến cho ca test chưa thể triển khai

## 6.2. Biện pháp giảm thiểu

* Cập nhật liên tục Test Plan theo thay đổi của SRS.
* Đào tạo, phối hợp chặt chẽ giữa các nhóm phát triển và kiểm thử.
* Sử dụng môi trường mô phỏng và các công cụ ảo hóa (mock) để kiểm thử độc lập với các bên thứ ba.

# 

# 7. Báo cáo và theo dõi

* **Công cụ báo cáo:** Sử dụng JaCoCo để đo lường Unit Test Coverage
* **Báo cáo bug:** Ghi nhận và phân loại lỗi theo mức độ nghiêm trọng (Critical, High, Medium, Low) trong hệ thống quản lý lỗi Github issue
* **Đánh giá kết quả:** Họp định kỳ (hàng tuần) để xem xét kết quả kiểm thử, phân tích lỗi và đề xuất giải pháp.

# 8. Kết luận

Test Plan này nhằm đảm bảo hệ thống Quản Lý Ký Túc Xá hoạt động ổn định, đáp ứng đầy đủ các yêu cầu nghiệp vụ và chất lượng theo SRS. Các nhóm phát triển, kiểm thử và quản lý dự án cần phối hợp chặt chẽ để cập nhật kế hoạch kiểm thử theo các thay đổi của dự án.