TRƯỜNG ĐẠI HỌC GIAO THÔNG VẬN TẢI (Bold, size 14)

PHÂN HIỆU TẠI TP. HỒ CHÍ MINH (Bold, size 14)

BỘ MÔN CÔNG NGHỆ THÔNG TIN (Bold, size 14)

BÁO CÁO ĐỒ ÁN TỐT NGHIỆP

(Bold, size 15)

ĐỀ TÀI: NGHIÊN CỨU SO SÁNH HIỆU SUẤT VÀ CHẤT LƯỢNG TESTCASE GIỮA PHƯƠNG PHÁP THỦ CÔNG VÀ AI AGENT SỬ DỤNG N8N TRÊN HỆ THỐNG ORANGE HRM VỚI CÔNG CỤ PLAYWRIGHT

(Bold, size 16)

Giảng viên hướng dẫn: TRẦN PHONG NHÃ (size 14, chữ hoa)

Sinh viên thực hiện: NGUYỄN NGỌC TIỂU THƯ (size 14, chữ in hoa)

Lớp : CQ.62. CNTT (size 14, chữ in hoa)

Khoá : K62 (size 14, chữ in hoa)

Tp. Hồ Chí Minh, năm 2025 (size 14)

TRƯỜNG ĐẠI HỌC GIAO THÔNG VẬN TẢI (Bold, size 14)

PHÂN HIỆU TẠI TP. HỒ CHÍ MINH (Bold, size 14)

BỘ MÔN CÔNG NGHỆ THÔNG TIN (Bold, size 14)

BÁO CÁO ĐỒ ÁN TỐT NGHIỆP

(Bold, size 15)

ĐỀ TÀI: NGHIÊN CỨU SO SÁNH HIỆU SUẤT VÀ CHẤT LƯỢNG TESTCASE GIỮA PHƯƠNG PHÁP THỦ CÔNG VÀ AI AGENT SỬ DỤNG N8N TRÊN HỆ THỐNG ORANGE HRM VỚI CÔNG CỤ PLAYWRIGHT

(Bold, size 16)

Giảng viên hướng dẫn: TRẦN PHONG NHÃ (size 14, chữ hoa)

Sinh viên thực hiện: NGUYỄN NGỌC TIỂU THƯ (size 14, chữ in hoa)

Lớp : CQ.62. CNTT (size 14, chữ in hoa)

Khoá : K62 (size 14, chữ in hoa)

Tp. Hồ Chí Minh, năm 2025 (size 14)

TRƯỜNG ĐẠI HỌC GIAO THÔNG VẬN TẢI
PHÂN HIỆU TẠI THÀNH PHỐ HỒ CHÍ MINH

CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM
Độc lập – Tự do – Hạnh phúc

NHIỆM VỤ THIẾT KẾ TỐT NGHIỆP

BỘ MÔN: **CÔNG NGHỆ THÔNG TIN**

-------***-------

Mã sinh viên: 6251071101 **Họ tên SV: Nguyễn Ngọc Tiểu Thư**

Khóa: 62 **Lớp: CNTT**

1. Tên đề tài

- Nghiên cứu so sánh hiệu suất và chất lượng test case giữa phương pháp thủ công và AI Agent sử dụng n8n trên hệ thống Orange HRM với công cụ Playwright

2. Mục đích, yêu cầu

Mục đích của đề tài là nghiên cứu và đánh giá sự khác biệt giữa kiểm thử phần mềm theo phương pháp thủ công truyền thống và phương pháp tự động hóa sử dụng AI Agent. Trong bối cảnh phát triển phần mềm hiện đại, việc ứng dụng trí tuệ nhân tạo để tối ưu quy trình kiểm thử là hướng đi tiềm năng nhằm nâng cao hiệu suất và tiết kiệm chi phí.

Đề tài đặt ra yêu cầu xây dựng mô hình kiểm thử tương ứng cho cả hai phương pháp, tiến hành so sánh định lượng dựa trên các tiêu chí cụ thể như thời gian thực thi, độ bao phủ, độ chính xác và khả năng tái sử dụng. Qua đó, đề tài hướng đến việc đánh giá tính khả thi của việc ứng dụng AI vào hoạt động kiểm thử trong thực tiễn doanh nghiệp.

3. Nội dung và phạm vi đề tài

Đề tài tập trung nghiên cứu, triển khai và kiểm thử hệ thống mã nguồn mở Orange HRM. Tiến hành xây dựng một tập hợp test case thủ công cho các chức năng tiêu biểu của hệ thống, đồng thời thiết lập quy trình tạo test case tự động bằng AI Agent trên nền tảng n8n. Các test case này sẽ được thực thi bằng công cụ kiểm thử tự động Playwright. Phạm vi của đề tài bao gồm việc lựa chọn các chức năng tiêu biểu trong mô-đun "Quản lý hồ sơ nhân viên (PIM)" để kiểm thử và đánh giá hiệu quả giữa hai phương pháp. Đề tài không đi sâu vào khía cạnh phát triển AI Agent mà chủ yếu tập trung vào khả năng tích hợp và áp dụng vào bài toán kiểm thử phần mềm.

Ngoài việc xây dựng và so sánh test case giữa phương pháp thủ công và AI Agent, đề tài còn tích hợp hệ thống quản lý dự án JIRA để thực hiện luồng tự động lấy dữ liệu từ các task hoặc issue liên quan đến kiểm thử. Cụ thể, các mô tả chức năng từ JIRA sẽ được trích xuất tự động và chuyển đổi thành test case thông qua AI Agent triển khai trên nền tảng n8n. Sau khi thực thi, kết quả kiểm thử cũng được trả ngược lại về JIRA dưới dạng comment, attachment hoặc cập nhật trạng thái. Việc tích hợp này không chỉ phục vụ cho việc tự động hóa quy trình kiểm thử mà còn giúp nâng cao khả năng quản lý, truy vết và phản hồi lỗi trong thực tiễn phát triển phần mềm.

4. Công nghệ, công cụ và ngôn ngữ lập trình

Trong quá trình triển khai đề tài, sử dụng hệ thống OrangeHRM làm nền tảng thử nghiệm.

- Triển khai AI Agent: công cụ n8n
- Triển khai Automation Testing: công cụ Playwright
- Ngông ngữ lập trình chính: JavaScript kết hợp với Node.js
- Quản lí mã nguồn: Git/Github
- Một số công cụ hỗ trợ khác bao gồm: Postman để test API, Docker để chạy Orange HRM localhost, Draw.io để vẽ các mô hình hệ thống, Google Sheet để lưu trữ và quản lí testcase.

Bên cạnh các công cụ chính như OrangeHRM, n8n và Playwright, đề tài còn sử dụng JIRA (Atlassian) như một hệ thống trung tâm để quản lý yêu cầu và lưu trữ kết quả kiểm thử. Các thao tác tương tác giữa JIRA và n8n sẽ được thực hiện thông qua API REST, đảm bảo tính linh hoạt và có thể tùy chỉnh theo từng loại issue. Việc kết nối giữa các hệ thống sẽ sử dụng Oauth2 hoặc Basic Auth tùy vào thiết lập, đảm bảo an toàn và phù hợp với môi trường thực tế. Ngoài ra, JSON sẽ là định dạng chính để trao đổi dữ liệu giữa các thành phần.

5. Các kết quả chính dự kiến sẽ đạt được và ứng dụng

Trước hết, đề tài sẽ xây dựng được hai bộ test case song song: một được thiết kế thủ công theo quy trình truyền thống và một được tạo tự động thông qua AI Agent tích hợp trên nền tảng n8n. Cả hai bộ kiểm thử đều được áp dụng trên cùng một tập chức năng của hệ thống OrangeHRM, đảm bảo tính đối sánh và nhất quán trong quá trình so sánh. Các test case sẽ bao gồm đầy đủ các loại tình huống như dữ liệu hợp lệ, dữ liệu biên, dữ liệu không hợp lệ và các trường hợp ngoại lệ.

Cung cấp một báo cáo phân tích định lượng về sự khác biệt giữa hai phương pháp kiểm thử, với các tiêu chí đánh giá cụ thể như: thời gian tạo và thực thi test case, tỷ lệ phát hiện lỗi, độ bao phủ chức năng, mức độ tái sử dụng cũng như khả năng thích ứng với thay đổi. Những phân tích này sẽ được trình bày rõ ràng bằng số liệu, biểu đồ và minh chứng cụ thể.

Một kết quả quan trọng khác là quy trình tích hợp toàn diện giữa ba công cụ: JIRA, n8n. Thông qua luồng tự động hóa được thiết kế, dữ liệu từ các nhiệm vụ kiểm thử trên JIRA sẽ được trích xuất, xử lý và sinh test case bằng AI Agent, sau đó được kết quả sẽ được cập nhật ngược trở lại vào JIRA để hỗ trợ việc quản lý chất lượng phần mềm.

Về mặt ứng dụng thực tiễn, kết quả của đề tài có thể triển khai trực tiếp trong các dự án phần mềm ở quy mô vừa và nhỏ, đặc biệt là trong các doanh nghiệp đang hướng đến tự động hóa quy trình kiểm thử hoặc áp dụng kiểm thử liên tục (Continuous Testing). Với sự hỗ trợ của JIRA – công cụ quản lý dự án phổ biến hiện

nay và khả năng tích hợp với AI Agent thông qua n8n, đề tài mang lại một hướng tiếp cận linh hoạt, tiết kiệm chi phí và phù hợp với thực tế triển khai.

Xét về tiềm năng mở rộng, đề tài có thể đóng vai trò là tiền đề cho các nghiên cứu sâu hơn liên quan đến tối ưu hóa việc sinh test case tự động bằng AI, ứng dụng machine learning trong phân tích hành vi người dùng để dự đoán lỗi, hoặc khai thác dữ liệu kiểm thử phục vụ phân tích rủi ro.

6. Giáo viên và cán bộ hướng dẫn

Họ tên: Trần Phong Nhã

Đơn vị công tác: Bộ môn Công nghệ Thông tin

Điện thoại: Email:

Ngày tháng 03 năm 2025	**Đã giao nhiệm vụ TKTN**
Trưởng BM Công nghệ Thông tin	**Giáo viên hướng dẫn**

ThS. Trần Phong Nhã

Đã nhận nhiệm vụ TKTN

Sinh viên: Nguyễn Ngọc Tiểu Thư Ký tên: Thư

Điện thoại:0935367598 Email: 6251071101@gmail.com

LỜI CẢM ƠN (size 15, bold)

(Cách 1 tab, Time newRoman, 20)

Size 13

NHẬN XÉT CỦA GIÁO VIÊN HƯỚNG DẪN (size 15, bold)

(Cách 1 tab, Time newRoman, 20)

Size 13

Tp. Hồ Chí Minh, ngày …… tháng ……. năm …….

Giáo viên hướng dẫn

Trần Phong Nhã

MỤC LỤC (size 15, bold)

(Cách 1 tab, Time newRoman, 20)

BẮT BUỘC DÙNG MỤC LỤC TỰ ĐỘNG

DANH MỤC CHỮ VIẾT TẮT

STT	Mô tả	Ý nghĩa	Ghi chú

BẢNG BIỂU, SƠ ĐỒ, HÌNH VẼ (size 15, bold)

(Cách 1 tab, Time newRoman, 20, mỗi nội dung trình bày bắt đầu từ 1 trang mới)

Bảng 1.1: (size 13)…….....

Bảng 1.2: .....

Sơ đồ 1.1:……..

Hình 1.1: ……..

Ghi chú:

- Xếp sau trang Mục lục
- Chữ số thứ nhất chỉ tên chương
- Chữ số thứ hai chỉ thứ tự bảng biểu, sơ đồ, hình,…trong mỗi chương
- Ở cuối mỗi bảng biểu, sơ đồ, hình,…trong mỗi chương phải có ghi chú, giải thích, nêu rõ nguồn trích hoặc sao chụp,…

CHƯƠNG 1. GIỚI THIỆU ĐỀ TÀI

1.1 Tổng quan dề tài

Trong kỷ nguyên số hóa, phần mềm đóng vai trò ngày càng quan trọng trong mọi lĩnh vực của đời sống. Tuy nhiên, sự phức tạp ngày càng tăng của hệ thống phần mềm đặt ra thách thức lớn trong việc đảm bảo chất lượng. Kiểm thử phần mềm (Software Testing) là một giai đoạn không thể thiếu trong chu trình phát triển phần mềm (Software Development Life Cycle - SDLC), nhằm phát hiện và khắc phục các lỗi (defects) trước khi sản phẩm đến tay người dùng [1]. Tầm quan trọng của kiểm thử không chỉ dừng lại ở việc tìm lỗi, mà còn góp phần nâng cao độ tin cậy, hiệu năng và trải nghiệm người dùng của ứng dụng [2]. Ngoài việc đảm bảo chất lượn phần mềm thì kiểm thử còn giúp phần mềm đáp ứng nhu cầu khách hàng - người dùng.

Theo báo cáo World Quality Report 2023–2024, chi phí sửa lỗi phần mềm ở giai đoạn triển khai có thể cao gấp nhiều lần so với việc phát hiện và sửa lỗi ở giai đoạn đầu [3]. Điều này càng nhấn mạnh tầm quan trọng của một quy trình kiểm thử hiệu quả. Tuy nhiên, với chu kỳ phát hành phần mềm ngày càng ngắn và quy mô ứng dụng ngày càng lớn, kiểm thử thủ công bộc lộ nhiều hạn chế về thời gian, chi phí và tính lặp lại. Điều này thúc đẩy sự phát triển của kiểm thử tự động (Test Automation) [4].

Trong những năm gần đây, sự phát triển của Trí tuệ Nhân tạo (AI), đặc biệt là các mô hình ngôn ngữ lớn (Large Language Models - LLMs), đã mở ra hướng đi mới cho kiểm thử phần mềm. AI được nghiên cứu nhằm hỗ trợ sinh bước kiểm thử, tạo dữ liệu kiểm thử, phân tích kết quả kiểm thử, tối ưu bộ test hoặc hỗ trợ sửa lỗi khi quá trình thực thi thất bại [5], [6]. Một số mô hình AI như ChatGPT, Gemini có thể hỗ trợ sinh các test case tự động từ yêu cầu đầu vào bằng ngôn ngữ tự nhiên, góp phần rút ngắn thời gian và nâng cao tính bao phủ kiểm thử.

Việc tích hợp AI vào kiểm thử đòi hỏi phải phối hợp nhiều thành phần: công cụ tự động hóa (như Playwright), mô hình AI (qua API), công cụ quản lý dữ liệu và báo cáo. Các nền tảng điều phối như n8n cho phép xây dựng quy trình kiểm thử phức tạp mà không cần viết quá nhiều mã nguồn [7]. Công cụ n8n hỗ trợ kết nối nhiều dịch vụ dưới dạng workflow, từ đó có thể xây dựng các "AI Agent" điều phối gọi API AI, xử lý phản hồi, và tự động kích

hoạt các công việc tiếp theo như chạy test case hoặc sinh test case mới dựa trên thông số đầu vào [8].

Trong đề tài này, AI Agent đóng vai trò hỗ trợ tự động sinh test case từ các yêu cầu đầu vào thay vì viết thủ công. Cụ thể, AI Agent nhận thông tin chức năng hệ thống (ví dụ như mô tả nghiệp vụ, đầu vào/đầu ra kỳ vọng, đặc tả chức năng), sau đó sinh ra các test case phù hợp theo cấu trúc chuẩn. Quy trình được thực hiện qua workflow trên n8n, nơi AI được kích hoạt qua API để tạo ra nội dung kiểm thử. Các test case này sau đó được dùng như đầu vào cho công cụ kiểm thử giao diện người dùng (Playwright) để tiến hành kiểm thử tự động.

1.2 Lí do chọn đề tài

Đề tài "Nghiên cứu so sánh hiệu suất và chất lượng test case giữa phương pháp thủ công và AI Agent sử dụng n8n trên hệ thống Orange HRM với công cụ Playwright" được lựa chọn dựa trên các cơ sở sau:

- Tính thực tiễn cao: Orange HRM là một hệ thống quản lý nhân sự mã nguồn mở phổ biến, có nhiều module chức năng đa dạng, phù hợp làm hệ thống thử nghiệm để đánh giá các phương pháp kiểm thử.
- Xu hướng công nghệ: Ứng dụng AI và tự động hóa kiểm thử đang ngày càng phổ biến trong ngành công nghiệp phần mềm. n8n là nền tảng no-code hiện đại, giúp dễ dàng xây dựng các workflow tích hợp AI Agent và các công cụ khác.
- Đóng góp khoa học và thực tiễn: So sánh trực tiếp giữa test case thủ công và AI-generated test case trên cùng một hệ thống thực tế sẽ cung cấp dữ liệu khách quan, giúp doanh nghiệp và cộng đồng phát triển phần mềm có thêm căn cứ lựa chọn phương pháp kiểm thử phù hợp.
- Khả năng mở rộng: Việc tích hợp Jira để tự động tạo task và nhận test case từ AI Agent thông qua n8n là một điểm mới, giúp tự động hóa quy trình kiểm thử và tăng hiệu quả quản lý và traceability

1.3 Mục tiêu nghiên cứu

Các vấn đề được đặt ra:

- Việc sử dụng AI Agent kết hợp n8n và Playwright có cải thiện hiệu suất kiểm thử không?

- Chất lượng test case do AI tự động tạo ra có tương đương hoặc tốt hơn test case thủ công không?
- Việc triển khai hệ thống tích hợp này có những ưu nhược điểm gì trong thực tiễn?

Để trả lời, đề tài sẽ thực hiện một nghiên cứu so sánh thực nghiệm trên hệ thống Orange HRM – một nền tảng mã nguồn mở có kiến trúc module rõ ràng.

- Tìm hiểu lý thuyết và thực tiễn kiểm thử phần mềm (thủ công và tự động), công cụ Playwright, nền tảng n8n và AI Agent.
- Xây dựng quy trình tích hợp AI Agent (sử dụng API mô hình AI) với n8n để hỗ trợ sinh test case tự động.
- So sánh hiệu suất tạo test case: Đánh giá thời gian và công sức cần thiết để tạo test case thủ công so với việc sử dụng AI Agent thông qua n8n.
- Đánh giá chất lượng test case: So sánh độ bao phủ kiểm thử, khả năng phát hiện lỗi, và độ chính xác của test case giữa hai phương pháp.
- Triển khai và đánh giá hệ thống tự động: Xây dựng workflow tích hợp n8n với Jira và Playwright để tự động nhận task, sinh test case và chạy kiểm thử, từ đó phân tích hiệu quả vận hành thực tế.
- Đề xuất giải pháp tối ưu: Dựa trên kết quả nghiên cứu, đề xuất hướng phát triển và áp dụng phù hợp cho các tổ chức phát triển phần mềm.
- Phân tích ưu nhược điểm của phương pháp kiểm thử có AI hỗ trợ trong bối cảnh thực tế.

1.4 Phạm vi

- Hệ thống kiểm thử: Orange HRM phiên bản 5.7, kiểm thử tập trung vào một số chức năng trong module PIM như: quản lí nhân viên (PIM), quản lí nghỉ phép (Leave)
- Loại kiểm thử: Tập trung vào kiểm thử chức năng (Functional Testing) ở mức UI. Không bao gồm kiểm thử hiệu năng, bảo mật hoặc kiểm thử API.
- Công cụ sử dụng:
    - Playwright (UI Automation) [7]
    - n8n (Orchestration Platform) [8]
- AI Agent: API từ OpenAI (hỗ trợ sinh bước kiểm thử, dữ liệu test, đánh giá kết quả và tự động sinh test case).

- Ngôn ngữ lập trình Playwright: JavaScript.

1.5 Ý nghĩa khoa học và thực tiễn

- Về khoa học: Đề tài góp phần làm rõ hiệu quả ứng dụng AI trong tự động hóa kiểm thử phần mềm, đặc biệt là khả năng sinh test case tự động và tích hợp với hệ thống quản lý dự án. Kết quả nghiên cứu sẽ là tài liệu tham khảo hữu ích cho các nhà nghiên cứu và phát triển trong lĩnh vực kiểm thử phần mềm và trí tuệ nhân tạo , .
- Về thực tiễn: Giúp các doanh nghiệp phần mềm hiểu rõ hơn về lợi ích và hạn chế của việc áp dụng AI Agent trong quy trình kiểm thử, từ đó có thể lựa chọn công cụ và phương pháp phù hợp để nâng cao hiệu quả kiểm thử, giảm chi phí và thời gian phát triển

1.6 Cấu trúc báo cáo thực tập tốt nghiệp

CHƯƠNG 2. CƠ SỞ LÝ THUYẾT VÀ TỔNG QUAN TÀI LIỆU

2.1 Tổng quan kiểm thử phần mềm

Kiểm thử phần mềm là một giai đoạn không thể thiếu trong quá trình phát triển phần mềm nhằm đảm bảo chất lượng sản phẩm trước khi triển khai thực tế. Thông qua việc kiểm tra, đánh giá và phát hiện lỗi, kiểm thử góp phần giảm thiểu rủi ro vận hành và chi phí sửa lỗi trong tương lai. Theo Sommerville [1], kiểm thử không chỉ giúp xác định sự tuân thủ của hệ thống đối với các yêu cầu đã định mà còn tăng cường mức độ tin cậy của phần mềm trong mắt người sử dụng.

Trong vòng đời phát triển phần mềm (Software Development Life Cycle – SDLC), kiểm thử có thể được thực hiện theo nhiều chiến lược khác nhau. Với các phương pháp phát triển truyền thống như Waterfall, kiểm thử thường xuất hiện sau giai đoạn lập trình. Tuy nhiên, trong các mô hình linh hoạt như Agile hoặc DevOps, kiểm thử được tích hợp xuyên suốt vòng đời phát triển, đảm bảo phát hiện lỗi sớm và cải tiến liên tục [2]. Madakam và cộng sự [2] cũng nhấn mạnh rằng kiểm thử trong DevOps là thành phần không thể thiếu để đạt được hiệu quả cao trong chu trình tích hợp và triển khai liên tục.

2.1.1 Các khái niệm cơ bản trong kiểm thử phần mềm

Mục tiêu của kiểm thử phần mềm không chỉ là xác minh hệ thống hoạt động đúng theo yêu cầu, mà còn nhằm tìm ra các tình huống phần mềm có thể hoạt động sai lệch so với kỳ

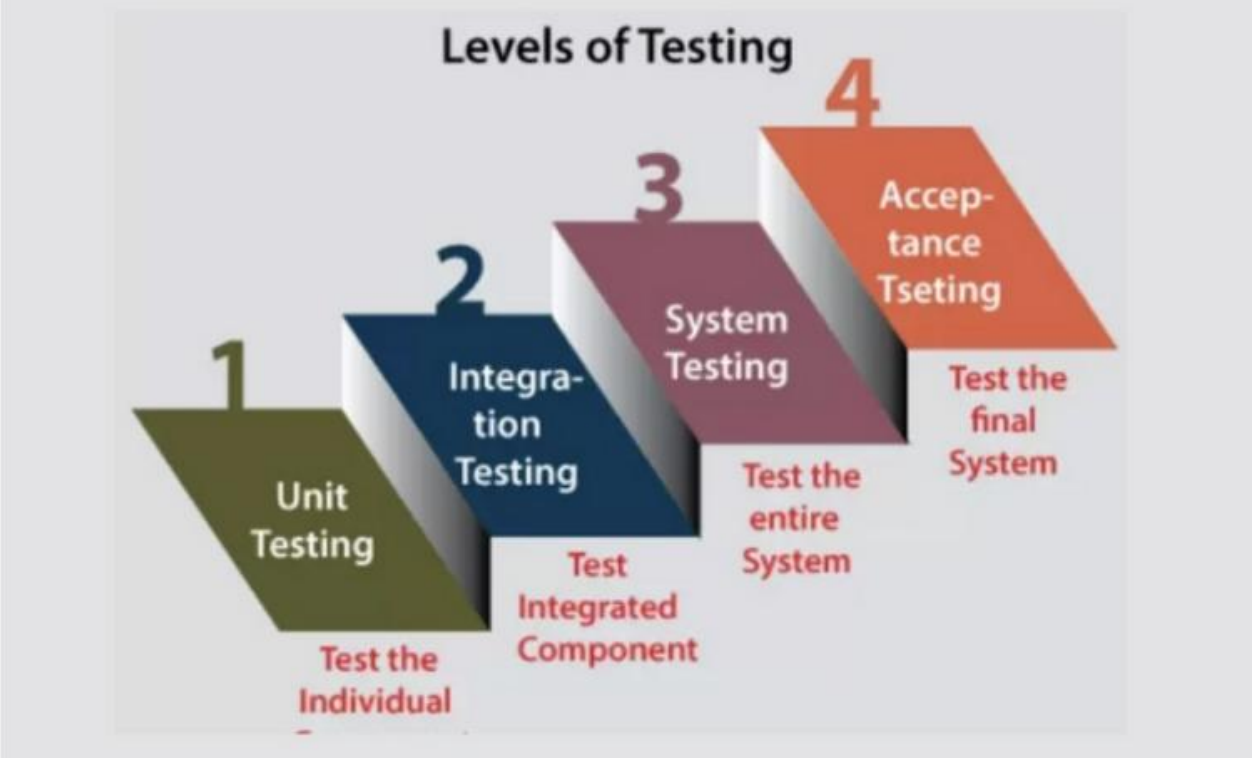
vọng. Nhờ đó, đội phát triển có thể khắc phục kịp thời và tránh phát sinh lỗi nghiêm trọng trong môi trường thực tế [1][2].

Quy trình kiểm thử thông thường bao gồm bốn giai đoạn chính:

- Lập kế hoạch kiểm thử (Test Planning): Xác định chiến lược kiểm thử, phạm vi, tài nguyên, thời gian và rủi ro [2].
- Phân tích và thiết kế test case (Test Design): Dựa trên yêu cầu và thiết kế hệ thống để xác định điều kiện kiểm thử và mô tả chi tiết từng bước kiểm thử [2].
- Thực thi kiểm thử (Test Execution): Tiến hành thực hiện test case, ghi nhận kết quả thực tế, so sánh với kết quả mong đợi và ghi nhận lỗi (nếu có).
- Đóng kiểm thử (Test Closure): Đánh giá toàn bộ quá trình kiểm thử, tổng hợp báo cáo và lưu trữ tài liệu.

Ngoài ra, có nhiều loại kiểm thử khác nhau, mỗi loại phục vụ một mục đích cụ thể trong đảm bảo chất lượng phần mềm:

- Kiểm thử chức năng: Tập trung kiểm tra đầu ra của hệ thống dựa trên các yêu cầu đầu vào. Đề tài này chủ yếu áp dụng kiểm thử chức năng cho giao diện người dùng của OrangeHRM.
- Kiểm thử hồi quy: Thực hiện lại các kiểm thử sau khi phần mềm có thay đổi, nhằm đảm bảo các chức năng cũ vẫn hoạt động ổn định.
- Kiểm thử tích hợp: Xác minh sự tương tác đúng đắn giữa các module trong hệ thống.
- Kiểm thử chấp nhận người dùng (UAT): Được thực hiện bởi người dùng cuối nhằm đảm bảo hệ thống đáp ứng đúng các yêu cầu nghiệp vụ [2].



2.1.2 Kỹ thuật thiết kế test case

Một bộ test case hiệu quả sẽ tối ưu số lượng kiểm thử trong khi vẫn đảm bảo phát hiện lỗi ở mức cao. Các kỹ thuật thiết kế test case ra đời nhằm hỗ trợ người kiểm thử lựa chọn các điều kiện và dữ liệu đầu vào một cách có hệ thống và có căn cứ. Một số kỹ thuật phổ biến bao gồm:

- Phân vùng tương đương (Equivalence Partitioning - EP): Chia dữ liệu đầu vào thành các nhóm có hành vi giống nhau. Với mỗi nhóm, chỉ cần kiểm thử một đại diện là đủ để bao quát hành vi của cả nhóm [4].
- Phân tích giá trị biên (Boundary Value Analysis - BVA): Tập trung vào các giá trị ở ranh giới giữa các phân vùng, vì lỗi thường xảy ra ở đây nhiều hơn là bên trong [4].
- Bảng quyết định (Decision Table): Áp dụng trong các tình huống có nhiều điều kiện và luật rẽ nhánh. Mỗi cột trong bảng là một tổ hợp điều kiện tương ứng với một test case cụ thể [4].
- Trạng thái chuyển tiếp (State Transition): Mô hình hóa hệ thống dưới dạng các trạng thái và các sự kiện dẫn tới chuyển trạng thái, từ đó xây dựng các test case cho từng luồng chuyển tiếp [4].

Việc kết hợp linh hoạt các kỹ thuật trên giúp tạo ra bộ test case vừa đầy đủ, vừa tiết kiệm chi phí kiểm thử.

2.1.3 Các chỉ số đánh giá test case

Bên cạnh việc thiết kế test case, việc đánh giá chất lượng của chúng cũng rất quan trọng. Một số chỉ số phổ biến giúp lượng hóa chất lượng test case bao gồm:

- Độ bao phủ kiểm thử (Test Coverage): Xác định mức độ mà test case đã bao phủ các yêu cầu, chức năng, hoặc các thành phần mã nguồn. Trong đề tài, độ phủ chức năng và độ phủ yêu cầu là hai tiêu chí được ưu tiên [5].
- Tỷ lệ phát hiện lỗi (Defect Detection Rate): Tính toán số lỗi phát hiện được trên mỗi test case hoặc trên mỗi lần kiểm thử. Chỉ số này phản ánh hiệu quả phát hiện lỗi của bộ test case [5].
- Độ rõ ràng và khả năng hiểu (Clarity and Readability): Test case nên được viết ngắn gọn, dễ hiểu để cả người kiểm thử thủ công và công cụ tự động đều có thể thực hiện chính xác.
- Tính trùng lặp (Redundancy): Các test case không nên trùng lặp về nội dung và chức năng được kiểm thử, nhằm tối ưu hóa thời gian và tài nguyên kiểm thử [5].

2.2 AI Agent trong sinh test case và nền tảng n8n

2.2.1 Tổng quan về AI Agent trong kiểm thử phần mềm

2.2.2 Phân loại các phương pháp sinh test case bằng AI

2.2.3 Nền tảng n8n và khả năng ứng dụng trong workflow AI testing

2.2.4 Tổng quan các nghiên cứu liên quan về AI-driven test case generation

2.2.5 Tích hợp hệ thống quản lý yêu cầu (JIRA) trong quy trình kiểm thử hiện đại

2.3 Công cụ kiểm thử và hệ thống thử nghiệm

2.3.1 Giới thiệu về Playwright

2.3.2 Giới thiệu hệ thống mã nguồn mở Orange HRM

CHƯƠNG 3. PHƯƠNG PHÁP NGHIÊN CỨU

CHƯƠNG 4: THIẾT KẾ VÀ TRIỂN KHAI HỆ THỐNG

CHƯƠNG 5: THỰC NGHIỆM VÀ KẾT QUẢ

CHƯƠNG 6: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

PHỤ LỤC

Phụ lục 1: hướng dẫn cài đặt

Phụ lục 2: hướng dẫn sử dụng

TÀI LIỆU THAM KHẢO

[1] I. Sommerville, Software Engineering, 10th ed. Boston, MA: Pearson, 2015.

[2] G. Meszaros, xUnit Test Patterns: Refactoring Test Code. Boston, MA: Addison-Wesley, 2007.

[3] Capgemini and Sogeti, "World Quality Report 2023–2024," 2023. [Online]. Available: https://www.capgemini.com/research/world-quality-report-2023-24/.

[4] A. Z. Abdou and L. Jianhua, "Automated software testing: A survey," in Proc. IEEE Int. Conf. on Software Engineering and Service Science (ICSESS), Beijing, China, 2018, pp. 1–6. doi: 10.1109/ICSESS.2018.8663861.

[5] M. Zhang, M. Kuehlwein, and M. Pradel, "ChatGPT for test case generation: How far are we?" *arXiv preprint arXiv:2304.01115*, 2023. [Online]. Available: https://arxiv.org/abs/2304.01115

[6] L. Luo, Z. Xu, and X. Zheng, "AI-powered software testing: A systematic literature review," Future Generation Computer Systems, vol. 136, pp. 379–397, 2022. doi: 10.1016/j.future.2022.06.014.

[7] Microsoft, "Playwright Documentation," 2024. [Online].
https://playwright.dev/

[8] n8n.io, "n8n Documentation," 2024. [Online]. Available: https://docs.n8n.io/

[9] P. N. Jorge and R. M. Braga, "Metrics for assessing the quality of test cases," in Proc. 2011 IEEE Int. Conf. on Software Testing, Verification and Validation Workshops, Berlin, Germany, 2011, pp. 183–188. doi: 10.1109/ICSTW.2011.48.

[10] A. J. Offutt, Introduction to Software Testing, 2nd ed., Cambridge University Press, 2016.

[11] T. Y. Chen, F. Kuo, R. Merkel, and T. Tse, "Adaptive random testing: The ART of test case diversity," Journal of Systems and Software, vol. 83, no. 1, pp. 60–66, 2019.

[12] M. U. Iftikhar, S. Khan, and M. A. Babar, "Test coverage and adequacy measurement techniques: A systematic review," Journal of Systems and Software, vol. 184, 2022.