**Static Test Document**

**Lotte Cinema**

# Giới thiệu

Static test document được tạo ra để hướng dẫn và mô tả các hoạt động static test trong dự án phần mềm. Tài liệu này giúp đảm bảo rằng mã nguồn và thiết kế của hệ thống tuân thủ các tiêu chuẩn, quy tắc và quy trình static test đã được thiết lập.

Trong tài liệu này, các tiêu chuẩn kiểm thử tĩnh áp dụng cho phần mềm được xác định, bao gồm quy chuẩn mã nguồn và quy chuẩn thiết kế. Các hoạt động static test bao gồm xem xét mã nguồn, kiểm tra cú pháp và kiểm tra thiết kế. Chúng tôi sẽ mô tả quy trình và phương pháp thực hiện các hoạt động này, cùng với các công cụ hỗ trợ và kế hoạch static test.

Tài liệu static test này sẽ giúp tăng tính ổn định, đáng tin cậy và hiệu quả của phần mềm, đồng thời đảm bảo rằng mã nguồn và thiết kế đáp ứng các tiêu chuẩn và quy tắc chung.

# Tiêu chuẩn

Tiêu chuẩn được áp dụng cho phần mềm sẽ tuân thủ các tiêu chí sau:

* Quy chuẩn mã nguồn:
  + Đặt tên biến, hàm và lớp: Mã nguồn phải tuân thủ quy tắc đặt tên biến, hàm và lớp có ý nghĩa, dễ đọc, và phù hợp với ngữ nghĩa của chúng.
  + Quy tắc lập trình: Mã nguồn phải tuân thủ các quy tắc lập trình như quy tắc đóng gói, quy tắc tránh sao chép mã nguồn, và quy tắc kiểm soát luồng.
  + Bảo mật: Mã nguồn phải được xây dựng với sự chú ý đến bảo mật và phòng chống các lỗ hổng bảo mật.
  + Hiệu suất: Mã nguồn phải được tối ưu hóa để đảm bảo hiệu suất cao và thời gian phản hồi nhanh của phần mềm.
* Quy chuẩn thiết kế:
  + Cấu trúc module: Thiết kế hệ thống phải tuân thủ cấu trúc module hợp lý, tách biệt và rõ ràng để dễ dàng quản lý và bảo trì.
  + Cấu trúc lớp: Thiết kế lớp phải được xây dựng theo quy tắc đóng gói, kế thừa, đa hình và trừu tượng để tạo ra một hệ thống linh hoạt và dễ mở rộng.
  + Quy tắc tương tác giữa các thành phần: Thiết kế phải đảm bảo sự tương tác chính xác và hiệu quả giữa các thành phần của hệ thống, bao gồm các module, lớp và giao diện người dùng.

# 3. Static test

## 3.1. Xem xét mã nguồn

- Mục đích: Xác định các lỗi và khuyết điểm trong mã nguồn.

- Quy trình:

a) Xác định các đoạn mã cần được xem xét.

b) Xác định các tiêu chí xem xét, bao gồm quy tắc lập trình, bảo mật và hiệu suất.

c) Thực hiện xem xét mã nguồn và ghi lại các lỗi và khuyết điểm tìm thấy.

d) Đề xuất các cải tiến và chỉnh sửa cần thiết.

## 3.2. Kiểm tra cú pháp

- Mục đích: Kiểm tra cú pháp của mã nguồn để đảm bảo rằng nó tuân thủ quy tắc ngôn ngữ lập trình.

- Quy trình:

a) Xác định ngôn ngữ lập trình được sử dụng trong dự án.

b) Thực hiện kiểm tra cú pháp để kiểm tra mã nguồn.

c) Ghi lại các lỗi cú pháp tìm thấy và đề xuất sửa chữa.

## 3.3. Kiểm tra thiết kế

- Mục đích: Kiểm tra thiết kế hệ thống để đảm bảo rằng nó đáp ứng yêu cầu và tuân thủ tiêu chuẩn thiết kế.

- Quy trình:

a) Xem xét thiết kế hệ thống, bao gồm cấu trúc module và cấu trúc lớp.

b) Xác định các lỗi và khuyết điểm trong thiết kế.

c) Ghi lại các lỗi và đề xuất cải tiến cho thiết kế.

# 4. Tiến hành

|  |  |
| --- | --- |
| **Description** | **Status** |
| Lỗi không đóng kết nối khi truy cập database | PASSED |
| Lỗi cơ bản trong chính tả, cú pháp | PASSED |
| Kiểm tra lỗi về logic, kiểu dữ liệu. | PASSED |
| Tính đúng đắn của các thông tin hiển thị trên giao diện | PASSED |
| Tính ràng buộc dữ liệu | PASSED |
| Việc phân quyền được thực hiện đúng cách | PASSED |
| Code không hiển thị thông tin bí mật của người dùng | PASSED |
| Tính khả dụng của ứng dụng (thời gian phản hồi, tốc độ xử lý, ...) | PASSED |
| Các biến, hàm trong ứng dụng có được đặt tên đúng theo đúng vai trò | PASSED |
| Các biến dư thừa | PASSED |
| Thư viện được sử dụng một cách hiệu quả | PASSED |
| Tính tái sử dụng của code | PASSED |
| Xác minh xem cấu trúc lớp đã được xây dựng theo quy tắc đóng gói, kế thừa, đa hình và trừu tượng hay chưa. | PASSED |
| Kiểm tra tính rõ ràng và linh hoạt của quan hệ kế thừa, sự phụ thuộc và giao diện người dùng trong cấu trúc lớp. | PASSED |