

## Mục Lục

CHƯƠNG 1: TỔNG QUAN VỀ KIỂM THỬ PHẦN MỀM .....	5
1.1. Các khái niệm cơ bản về kiểm thử phần mềm .....	5
1.1.1. Kiểm thử phần mềm là gì? .....	5
1.1.2. Các phương pháp kiểm thử .....	5
1.1.3. Các chiến lược kiểm thử .....	6
1.1.4. Các cấp độ kiểm thử phần mềm .....	8
1.1.5. Các phương pháp kiểm thử con người .....	14
1.2. Nguyên tắc kiểm thử phần mềm .....	15
CHƯƠNG 2: THIẾT KẾ TEST – CASE.....	17
2.1. Khái niệm.....	17
2.2. Vai trò của thiết kế test – case .....	17
2.3. Quy trình của thiết kế test – case .....	17
2.3.1. Kiểm thử hộp đen.....	18
2.3.2. Kiểm thử hộp trắng - Kiểm thử bao phủ logic .....	27
2.3.3. Chiến lược .....	27
2.3.4. Kiểm thử API .....	28
CHƯƠNG 3: ÁP DỤNG.....	29
3.1. WBS .....	29
3.2. Đặc tả .....	30
3.3. Kiểm thử hộp đen.....	31
3.3.1. Vẽ đồ thị nguyên nhân – kết quả.....	31
3.4. Kiểm thử chức năng.....	43
3.4.1. Chức năng mua vé.....	44
3.5. Kiểm thử cơ sở dữ liệu.....	74
3.6. Kiểm thử API.....	82
3.6.1. Giới thiệu công cụ Postman .....	82
3.6.2. Sử dụng Postman test chức năng đặt vé xe trong Phần mềm quản lý ứng dụng bán vé trực tuyến của công ty vận tải xe khách Thành Bưởi.....	82

## Danh mục hình

<i>Hình 2.1 Các ký hiệu đồ thị nguyên nhân – kết quả cơ bản</i> .....	23
<i>Hình 2.2 Các ký hiệu ràng buộc</i> .....	24
<i>Hình 2.3 Những xem xét được sử dụng khi dò theo đồ thị</i> .....	25
<i>Hình 3.1 Bảng phân công công việc WBS</i> .....	29
<i>Hình 3.2 Đồ thị nguyên nhân – kết quả</i> .....	33
<i>Hình 3.3 Màn hình quản lý tuyến xe</i> .....	44
<i>Hình 3.4 Màn hình chọn tuyến</i> .....	61
<i>Hình 3.5 Màn hình chọn ghế</i> .....	65
<i>Hình 3.6 Màn hình thông tin khách</i> .....	67
<i>Hình 3.7 Giao diện của Postman</i> .....	82
<i>Hình 3.8 Giao diện chuẩn bị check API Load_DiemDi</i> .....	84
<i>Hình 3.9 Kết quả kiểm tra API Load-DiemDi</i> .....	85
<i>Hình 3.10 Kết quả danh sách giờ trả về của case 1</i> .....	87
<i>Hình 3.11 Kết quả giờ trả về rỗng của case 2</i> .....	89
<i>Hình 3.12 Kết quả test trả về danh sách thông tin vé ghế của chuyến xe đã tồn tại</i> .....	94
<i>Hình 3.13 Kết quả test trả về danh sách thông tin vé ghế của chuyến xe chưa tồn tại</i> ....	95
<i>Hình 3.14 Kết quả trả về danh sách bảng giá của tuyến xe</i> .....	98
<i>Hình 3.15 Kết quả thông tin chuyến xe trả về</i> .....	100
<i>Hình 3.16 Kết quả kiểm tra hành khách cũ</i> .....	105

## Danh mục bảng

<i>Bảng 2.1 Phương pháp thiết kế ca kiểm thử.....</i>	<i>18</i>
<i>Bảng 2.2 Mẫu liệt kê các lớp tương đương .....</i>	<i>19</i>
<i>Bảng 3.1 Bảng quyết định.....</i>	<i>34</i>
<i>Bảng 3.2 Danh sách testcase từ đồ thị .....</i>	<i>39</i>
<i>Bảng 3.3 Test case phân vùng tương đương kiểm tra ngày đi .....</i>	<i>41</i>
<i>Bảng 3.4 Test case phân vùng tương đương kiểm tra số điện thoại .....</i>	<i>42</i>
<i>Bảng 3.5 Test case phân tích giá trị biên kiểm tra ngày đi .....</i>	<i>42</i>
<i>Bảng 3.6 Test case phân tích giá trị biên kiểm tra số điện thoại .....</i>	<i>43</i>
<i>Bảng 3.7 Test plan .....</i>	<i>43</i>
<i>Bảng 3.8 Test case quản lý tuyến xe .....</i>	<i>60</i>
<i>Bảng 3.9 Test case màn hình chọn tuyến .....</i>	<i>64</i>
<i>Bảng 3.10 Test case màn hình chọn ghế .....</i>	<i>67</i>
<i>Bảng 3.11 Test case màn hình thông tin khách .....</i>	<i>73</i>
<i>Bảng 3.12 Test case kiểm tra cơ sở dữ liệu tuyến xe .....</i>	<i>78</i>
<i>Bảng 3.13 Test case kiểm tra cơ sở dữ liệu cho chức năng mua vé .....</i>	<i>81</i>
<i>Bảng 3.14 Test case kiểm tra API Load-DiemDi .....</i>	<i>84</i>
<i>Bảng 3.15 Test case kiểm tra danh sách giờ trả về.....</i>	<i>88</i>
<i>Bảng 3.16 Test case danh sách thông tin vé ghế của chuyến xe .....</i>	<i>95</i>
<i>Bảng 3.17 Test case kiểm tra danh sách bảng giá của tuyến xe .....</i>	<i>97</i>
<i>Bảng 3.18 Test case kiểm tra thông tin chuyến xe .....</i>	<i>100</i>
<i>Bảng 3.19 Test case kiểm tra hành khách .....</i>	<i>107</i>

## LỜI CẢM ƠN

Chúng em cũng xin gửi lời cảm ơn thầy Trần Văn Thọ, giảng viên hướng dẫn môn Kiểm định chất lượng phần mềm, người đã tận tình giảng dạy cho chúng em những kiến thức cơ bản để thực hiện đề tài này

Dù đã cố gắng hoàn thành thật tốt đề tài nhưng chắc chắn sẽ có những thiếu sót không thể tránh khỏi. Chúng em mong sẽ nhận được sự thông cảm và sự tận tình chỉ bảo của thầy, các bạn.

Nhóm chúng em xin chân thành cảm ơn!

# CHƯƠNG 1: TỔNG QUAN VỀ KIỂM THỬ PHẦN MỀM

## 1.1. Các khái niệm cơ bản về kiểm thử phần mềm

### 1.1.1. Kiểm thử phần mềm là gì?

Kiểm thử phần mềm là quá trình khảo sát một hệ thống hay thành phần dưới những điều kiện xác định, quan sát và ghi lại các kết quả, và đánh giá một khía cạnh nào đó của hệ thống hay thành phần đó. (Theo *Bảng chú giải thuật ngữ chuẩn IEEE của Thuật ngữ kỹ nghệ phần mềm- IEEE Standard Glossary of Software Engineering Terminology*).

Kiểm thử phần mềm là quá trình thực thi một chương trình với mục đích tìm lỗi. (Theo *“The Art of Software Testing” – Nghệ thuật kiểm thử phần mềm*).

Kiểm thử phần mềm là hoạt động khảo sát thực tiễn sản phẩm hay dịch vụ phần mềm trong đúng môi trường chúng dự định sẽ được triển khai nhằm cung cấp cho người có lợi ích liên quan những thông tin về chất lượng của sản phẩm hay dịch vụ phần mềm ấy. Mục đích của kiểm thử phần mềm là tìm ra các lỗi hay khiếm khuyết phần mềm nhằm đảm bảo hiệu quả hoạt động tối ưu của phần mềm trong nhiều ngành khác nhau. (Theo *Bách khoa toàn thư mở Wikipedia*).

Có thể định nghĩa một cách dễ hiểu như sau: Kiểm thử phần mềm là một tiến trình hay một tập hợp các tiến trình được thiết kế để đảm bảo mã hóa máy tính thực hiện theo cái mà chúng đã được thiết kế để làm, và không thực hiện bất cứ thứ gì không mong muốn. Đây là một pha quan trọng trong quá trình phát triển hệ thống, giúp cho người xây dựng hệ thống và khách hàng thấy được hệ thống mới đã đáp ứng yêu cầu đặt ra hay chưa?

### 1.1.2. Các phương pháp kiểm thử

Có 2 phương pháp kiểm thử chính là: Kiểm thử tĩnh và Kiểm thử động.

#### 1.1.2.1. Kiểm thử tĩnh – Static testing

Là phương pháp thử phần mềm đòi hỏi phải duyệt lại các yêu cầu và các đặc tả bằng tay, thông qua việc sử dụng giấy, bút để kiểm tra logic, lần từng chi tiết mà không cần chạy chương trình. Kiểu kiểm thử này thường được sử dụng bởi chuyên viên thiết kế người mà viết mã lệnh một mình.

Kiểm thử tĩnh cũng có thể được tự động hóa. Nó sẽ thực hiện kiểm tra toàn bộ bao gồm các chương trình được phân tích bởi một trình thông dịch hoặc biên dịch mà xác nhận tính hợp lệ về cú pháp của chương trình.

#### **1.1.2.2. Kiểm thử động – Dynamic testing**

Là phương pháp thử phần mềm thông qua việc dùng máy chạy chương trình để điều tra trạng thái tác động của chương trình. Đó là kiểm thử dựa trên các ca kiểm thử xác định bằng sự thực hiện của đối tượng kiểm thử hay chạy các chương trình. Kiểm thử động kiểm tra cách thức hoạt động của mã lệnh, tức là kiểm tra sự phản ứng vật lý từ hệ thống tới các biến luôn thay đổi theo thời gian. Trong kiểm thử động, phần mềm phải thực sự được biên dịch và chạy. Kiểm thử động thực sự bao gồm làm việc với phần mềm, nhập các giá trị đầu vào và kiểm tra xem liệu đầu ra có như mong muốn hay không. Các phương pháp kiểm thử động gồm có kiểm thử Unit – Unit Tests, Kiểm thử tích hợp – Intergration Tests, Kiểm thử hệ thống – System Tests, và Kiểm thử chấp nhận sản phẩm – Acceptance Tests

#### **1.1.3. Các chiến lược kiểm thử**

Ba trong số những chiến lược kiểm thử thông dụng nhất bao gồm: Kiểm thử hộp đen, Kiểm thử hộp trắng, và Kiểm thử hộp xám.

##### **1.1.3.1. Kiểm thử hộp đen – Black box testing**

Một trong những chiến lược kiểm thử quan trọng là kiểm thử hộp đen, hướng dữ liệu, hay hướng vào/ra. Kiểm thử hộp đen xem chương trình như là một “hộp đen”. Mục đích của bạn là hoàn toàn không quan tâm về cách cư xử và cấu trúc bên trong của chương trình. Thay vào đó, tập trung vào tìm các trường hợp mà chương trình không thực hiện theo các đặc tả của nó.

Theo hướng tiếp cận này, dữ liệu kiểm tra được lấy chỉ từ các đặc tả.

##### **Các phương pháp kiểm thử hộp đen:**

- Phân lớp tương đương – Equivalence partitioning.
- Phân tích giá trị biên – Boundary value analysis.
- Kiểm thử mọi cặp – All-pairs testing.
- Kiểm thử fuzz – Fuzz testing.

- Kiểm thử dựa trên mô hình – Model-based testing.
- Ma trận dấu vết – Traceability matrix.
- Kiểm thử thăm dò – Exploratory testing.
- Kiểm thử dựa trên đặc tả – Specification-base testing.

Kiểm thử dựa trên đặc tả tập trung vào kiểm tra tính thiết thực của phần mềm theo những yêu cầu thích hợp. Do đó, kiểm thử viên nhập dữ liệu vào, và chỉ thấy dữ liệu ra từ đối tượng kiểm thử. Mức kiểm thử này thường yêu cầu các ca kiểm thử triệt để được cung cấp cho kiểm thử viên mà khi đó có thể xác minh là đối với dữ liệu đầu vào đã cho, giá trị đầu ra (hay cách thức hoạt động) có giống với giá trị mong muốn đã được xác định trong ca kiểm thử đó hay không. Kiểm thử dựa trên đặc tả là cần thiết, nhưng không đủ để ngăn chặn những rủi ro chắc chắn.

#### **Ưu, nhược điểm:**

Kiểm thử hộp đen không có mối liên quan nào tới mã lệnh, và kiểm thử viên chỉ rất đơn giản tâm niệm là: một mã lệnh phải có lỗi. Sử dụng nguyên tắc “Hãy đòi hỏi và bạn sẽ được nhận”, những kiểm thử viên hộp đen tìm ra lỗi mà những lập trình viên đã không tìm ra. Nhưng mặt khác, người ta cũng nói kiểm thử hộp đen “giống như là đi trong bóng tối mà không có đèn vậy”, bởi vì kiểm thử viên không biết các phần mềm được kiểm tra thực sự được xây dựng như thế nào. Đó là lý do mà có nhiều trường hợp mà một kiểm thử viên hộp đen viết rất nhiều ca kiểm thử để kiểm tra một thứ gì đó mà đáng lẽ có thể chỉ cần kiểm tra bằng 1 ca kiểm thử duy nhất, và/hoặc một số phần của chương trình không được kiểm tra chút nào.

Do vậy, kiểm thử hộp đen có ưu điểm của “một sự đánh giá khách quan”, mặt khác nó lại có nhược điểm của “thăm dò mù”.

#### **1.1.3.2. Kiểm thử hộp trắng – White box testing**

Là một chiến lược kiểm thử khác, trái ngược hoàn toàn với kiểm thử hộp đen, kiểm thử hộp trắng hay kiểm thử hướng logic cho phép bạn khảo sát cấu trúc bên trong của chương trình. Chiến lược này xuất phát từ dữ liệu kiểm thử bằng sự kiểm thử tính logic của chương trình. Kiểm thử viên sẽ truy cập vào cấu trúc dữ liệu và giải thuật bên trong chương trình (và cả mã lệnh thực hiện chúng).

#### **Các phương pháp kiểm thử hộp trắng**

- Kiểm thử giao diện lập trình ứng dụng - API testing (application programming interface): là phương pháp kiểm thử của ứng dụng sử dụng các API công khai và riêng tư.
- Bao phủ mã lệnh – Code coverage: tạo các kiểm tra để đáp ứng một số tiêu chuẩn về bao phủ mã lệnh.
- Các phương pháp gán lỗi – Fault injection.
- Các phương pháp kiểm thử hoán chuyển – Mutation testing methods.
- Kiểm thử tĩnh – Static testing: kiểm thử hộp trắng bao gồm mọi kiểm thử tĩnh.

Phương pháp kiểm thử hộp trắng cũng có thể được sử dụng để đánh giá sự hoàn thành của một bộ kiểm thử mà được tạo cùng với các phương pháp kiểm thử hộp đen. Điều này cho phép các nhóm phần mềm khảo sát các phần của 1 hệ thống ít khi được kiểm tra và đảm bảo rằng những điểm chức năng quan trọng nhất đã được kiểm tra.

#### ***1.1.3.3. Kiểm thử hộp xám – Gray box testing***

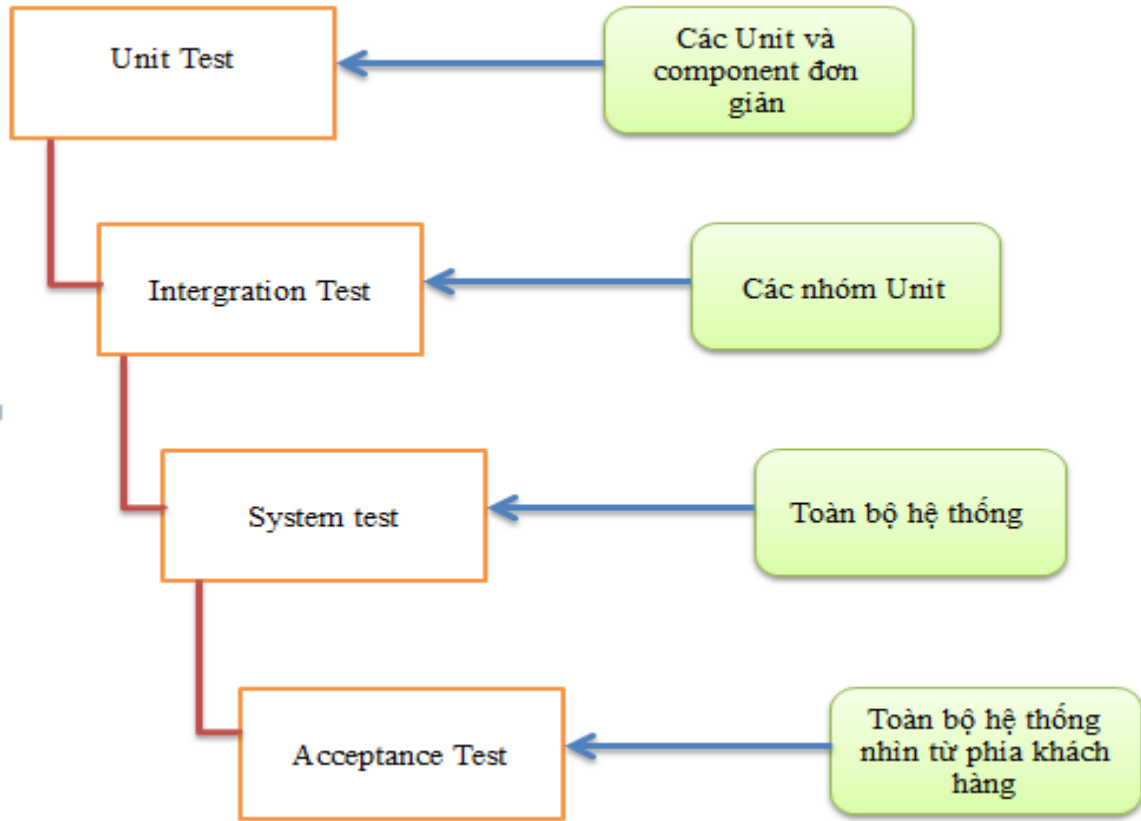
Kiểm thử hộp xám đòi hỏi phải có sự truy cập tới cấu trúc dữ liệu và giải thuật bên trong cho những mục đích thiết kế các ca kiểm thử, nhưng là kiểm thử ở mức người sử dụng hay mức hộp đen. Việc thao tác tới dữ liệu đầu vào và định dạng dữ liệu đầu ra là không rõ ràng, giống như một chiếc “hộp xám”, bởi vì đầu vào và đầu ra rõ ràng là ở bên ngoài “hộp đen” mà chúng ta vẫn gọi về hệ thống được kiểm tra. Sự khác biệt này đặc biệt quan trọng khi quản lý kiểm thử tích hợp – Intergartion testing giữa 2 modul mã lệnh được viết bởi hai chuyên viên thiết kế khác nhau, trong đó chỉ giao diện là được đưa ra để kiểm thử. Kiểm thử hộp xám có thể cũng bao gồm cả thiết kế đối chiếu để quyết định, ví dụ: giá trị biên hay thông báo lỗi.

#### **1.1.4. Các cấp độ kiểm thử phần mềm**

Kiểm thử phần mềm gồm có các cấp độ: Kiểm thử đơn vị, Kiểm thử tích hợp, Kiểm thử hệ thống và Kiểm thử chấp nhận sản phẩm.



Sơ đồ cấp độ kiểm thử:



#### 1.1.4.1. Kiểm thử đơn vị - Unit Test

Một đơn vị là một thành phần phần mềm nhỏ nhất mà ta có thể kiểm thử được. Ví dụ: các hàm (Function), thủ tục (Procedure), lớp (Class) hay phương thức (Method) đều có thể được xem là Unit.

Vì Unit được chọn để kiểm tra thường có kích thước nhỏ và chức năng hoạt động đơn giản, chúng ta không khó khăn gì trong việc tổ chức kiểm thử, ghi nhận và phân tích kết quả kiểm thử. Nếu phát hiện lỗi, việc xác định nguyên nhân và khắc phục cũng tương đối dễ dàng vì chỉ khoanh vùng trong một đơn thể Unit đang kiểm tra. Một nguyên lý đúc kết từ thực tiễn: thời gian tốn cho Unit Test sẽ được đền bù bằng việc tiết kiệm rất nhiều thời gian và chi phí cho việc kiểm thử và sửa lỗi ở các mức kiểm thử sau đó.

Unit Test thường do lập trình viên thực hiện. Công đoạn này cần được thực hiện càng sớm càng tốt trong giai đoạn viết code và xuyên suốt chu kỳ phát triển phần mềm. Thông thường, Unit Test đòi hỏi kiểm thử viên có kiến thức về thiết kế và code của

chương trình. Mục đích của Unit Test là bảo đảm thông tin được xử lý và xuất (khỏi Unit) là chính xác, trong mối tương quan với dữ liệu nhập và chức năng của Unit. Điều này thường đòi hỏi tất cả các nhánh bên trong Unit đều phải được kiểm tra để phát hiện nhánh phát sinh lỗi. Một nhánh thường là một chuỗi các lệnh được thực thi trong một Unit. Ví dụ: chuỗi các lệnh sau điều kiện If và nằm giữa then ... else là một nhánh. Thực tế việc chọn lựa các nhánh để đơn giản hóa việc kiểm thử và quét hết Unit đòi hỏi phải có kỹ thuật, đôi khi phải dùng thuật toán để chọn lựa.

Cùng với các mục kiểm thử khác, Unit Test cũng đòi hỏi phải chuẩn bị trước các ca kiểm thử (Test case) hoặc kịch bản kiểm thử (Test script), trong đó chỉ định rõ dữ liệu đầu vào, các bước thực hiện và dữ liệu đầu ra mong muốn. Các Test case và Test script này nên được giữ lại để tái sử dụng.

#### ***1.1.4.2. Kiểm thử tích hợp – Integration Test***

Integration test kết hợp các thành phần của một ứng dụng và kiểm thử như một ứng dụng đã hoàn thành. Trong khi Unit Test kiểm tra các thành phần và Unit riêng lẻ thì Integration Test kết hợp chúng lại với nhau và kiểm tra sự giao tiếp giữa chúng.

Hai mục tiêu chính của Integration Test:

- Phát hiện lỗi giao tiếp xảy ra giữa các Unit.
- Tích hợp các Unit đơn lẻ thành các hệ thống nhỏ (Subsystem) và cuối cùng là nguyên hệ thống hoàn chỉnh (System) chuẩn bị cho kiểm thử ở mức hệ thống (System Test).

Trong Unit Test, lập trình viên cố gắng phát hiện lỗi liên quan đến chức năng và cấu trúc nội tại của Unit. Có một số phép kiểm thử đơn giản trên giao tiếp giữa Unit với các thành phần liên quan khác, tuy nhiên mọi giao tiếp liên quan đến Unit chỉ thật sự được kiểm tra đầy đủ khi các Unit tích hợp với nhau trong khi thực hiện Integration Test.

Trừ một số ít ngoại lệ, Integration Test chỉ nên thực hiện trên những Unit đã được kiểm tra cẩn thận trước đó bằng Unit Test, và tất cả các lỗi mức Unit đã được sửa chữa. Một số người hiểu sai rằng Unit một khi đã qua giai đoạn Unit Test với các giao tiếp giả lập thì không cần phải thực hiện Integration Test nữa. Thực tế việc tích hợp giữa các Unit dẫn đến những tình huống hoàn toàn khác. Một chiến lược cần quan tâm trong Integration Test là nên tích hợp dần từng Unit. Một Unit tại một thời điểm được tích hợp vào một nhóm các Unit khác đã tích hợp trước đó và đã hoàn tất các đợt Integration Test trước đó.

Lúc này, ta chỉ cần kiểm thử giao tiếp của Unit mới thêm vào với hệ thống các Unit đã tích hợp trước đó, điều này sẽ làm cho số lượng ca kiểm thử giảm đi rất nhiều, và sai sót sẽ giảm đáng kể.

Có 4 loại kiểm thử trong Integration Test:

- **Kiểm thử cấu trúc (Structure Test):** Tương tự White Box Test, kiểm thử cấu trúc nhằm bảo đảm các thành phần bên trong của một chương trình chạy đúng và chú trọng đến hoạt động của các thành phần cấu trúc nội tại của chương trình chẳng hạn các câu lệnh và nhánh bên trong.
- **Kiểm thử chức năng (Functional Test):** Tương tự Black Box Test, kiểm thử chức năng chỉ chú trọng đến chức năng của chương trình, mà không quan tâm đến cấu trúc bên trong, chỉ khảo sát chức năng của chương trình theo yêu cầu kỹ thuật.
- **Kiểm thử hiệu năng (Performance Test):** Kiểm thử việc vận hành của hệ thống.
- **Kiểm thử khả năng chịu tải (Stress Test):** Kiểm thử các giới hạn của hệ thống.

#### ***1.1.4.3. Kiểm thử hệ thống – System Test***

Mục đích System Test là kiểm thử thiết kế và toàn bộ hệ thống (sau khi tích hợp) có thỏa mãn yêu cầu đặt ra hay không.

System Test bắt đầu khi tất cả các bộ phận của phần mềm đã được tích hợp thành công. Thông thường loại kiểm thử này tốn rất nhiều công sức và thời gian. Trong nhiều trường hợp, việc kiểm thử đòi hỏi một số thiết bị phụ trợ, phần mềm hoặc phần cứng đặc thù, đặc biệt là các ứng dụng thời gian thực, hệ thống phân bố, hoặc hệ thống nhúng. Ở mức độ hệ thống, người kiểm thử cũng 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 của toàn hệ thống.

Điểm khác nhau then chốt giữa Integration Test và System Test là System Test chú trọng các hành vi và lỗi trên toàn hệ thống, còn Integration Test chú trọng sự giao tiếp giữa các đơn thể hoặc đối tượng khi chúng làm việc cùng nhau. Thông thường ta

phải thực hiện Unit Test và Integration Test để bảo đảm mọi Unit và sự tương tác giữa chúng hoạt động chính xác trước khi thực hiện System Test.

Sau khi hoàn thành Integration Test, một hệ thống phần mềm đã được hình thành cùng với các thành phần đã được kiểm tra đầy đủ. Tại thời điểm này, lập trình viên hoặc kiểm thử viên bắt đầu kiểm thử phần mềm như một hệ thống hoàn chỉnh. Việc lập kế hoạch cho System Test nên bắt đầu từ giai đoạn hình thành và phân tích các yêu cầu.

System Test kiểm thử cả các hành vi chức năng của phần mềm lẫn các yêu cầu về chất lượng như độ tin cậy, tính tiện lợi khi sử dụng, hiệu năng và bảo mật. Mức kiểm thử này đặc biệt thích hợp cho việc phát hiện lỗi giao tiếp với phần mềm hoặc phần cứng bên ngoài, chẳng hạn các lỗi "tắc nghẽn" (deadlock) hoặc chiếm dụng bộ nhớ. Sau giai đoạn System Test, phần mềm thường đã sẵn sàng cho khách hàng hoặc người dùng cuối cùng kiểm thử chấp nhận sản phẩm (Acceptance Test) hoặc dùng thử (Alpha/Beta Test).

Đòi hỏi nhiều công sức, thời gian và tính chính xác, khách quan, System Test thường được thực hiện bởi một nhóm kiểm thử viên hoàn toàn độc lập với nhóm phát triển dự án. Bản thân System Test lại gồm nhiều loại kiểm thử khác nhau, phổ biến nhất gồm:

- **Kiểm thử chức năng (Functional Test):** Bảo đảm các hành vi của hệ thống thỏa mãn đúng yêu cầu thiết kế.
- **Kiểm thử hiệu năng (Performance Test):** Bảo đảm tối ưu việc phân bổ tài nguyên hệ thống (ví dụ bộ nhớ) nhằm đạt các chỉ tiêu như thời gian xử lý hay đáp ứng câu truy vấn, ...
- **Kiểm thử khả năng chịu tải (Stress Test hay Load Test):** Bảo đảm hệ thống vận hành đúng dưới áp lực cao (ví dụ nhiều người truy xuất cùng lúc). Stress Test tập trung vào các trạng thái tới hạn, các "điểm chết", các tình huống bất thường như đang giao dịch thì ngắt kết nối (xuất hiện nhiều trong kiểm tra thiết bị như POS, ATM...)...
- **Kiểm thử cấu hình (Configuration Test).**
- **Kiểm thử bảo mật (Security Test):** Bảo đảm tính toàn vẹn, bảo mật của dữ liệu và của hệ thống.

- **Kiểm thử khả năng phục hồi (Recovery Test):** Bảo đảm hệ thống có khả năng khôi phục trạng thái ổn định trước đó trong tình huống mất tài nguyên hoặc dữ liệu; đặc biệt quan trọng đối với các hệ thống giao dịch như ngân hàng trực tuyến...

Nhìn từ quan điểm người dùng, các cấp độ kiểm thử trên rất quan trọng: Chúng bảo đảm hệ thống đủ khả năng làm việc trong môi trường thực.

Lưu ý là không nhất thiết phải thực hiện tất cả các loại kiểm thử nêu trên. Tùy yêu cầu và đặc trưng của từng hệ thống, tùy khả năng và thời gian cho phép của dự án, khi lập kế hoạch, người Quản lý dự án sẽ quyết định áp dụng những loại kiểm thử nào.

#### ***1.1.4.4. Kiểm thử chấp nhận sản phẩm – Acceptance Test***

Thông thường, sau giai đoạn System Test là Acceptance Test, được khách hàng thực hiện (hoặc ủy quyền cho một nhóm thứ ba thực hiện). Mục đích của Acceptance Test là để chứng minh phần mềm thỏa mãn tất cả yêu cầu của khách hàng và khách hàng chấp nhận sản phẩm (và trả tiền thanh toán hợp đồng).

Acceptance Test có ý nghĩa hết sức quan trọng, mặc dù trong hầu hết mọi trường hợp, các phép kiểm thử của System Test và Acceptance Test gần như tương tự, nhưng bản chất và cách thức thực hiện lại rất khác biệt.

Đối với những sản phẩm dành bán rộng rãi trên thị trường cho nhiều người sử dụng, thông thường sẽ thông qua hai loại kiểm thử gọi là kiểm thử Alpha – Alpha Test và kiểm thử Beta – Beta Test. Với Alpha Test, người dùng kiểm thử phần mềm ngay tại nơi phát triển phần mềm, lập trình viên sẽ ghi nhận các lỗi hoặc phản hồi, và lên kế hoạch sửa chữa. Với Beta Test, phần mềm sẽ được gửi tới cho người dùng để kiểm thử ngay trong môi trường thực, lỗi hoặc phản hồi cũng sẽ gửi ngược lại cho lập trình viên để sửa chữa.

Thực tế cho thấy, nếu khách hàng không quan tâm và không tham gia vào quá trình phát triển phần mềm thì kết quả Acceptance Test sẽ sai lệch rất lớn, mặc dù phần mềm đã trải qua tất cả các kiểm thử trước đó. Sự sai lệch này liên quan đến việc hiểu sai yêu cầu cũng như sự mong chờ của khách hàng. Ví dụ đôi khi một phần mềm xuất sắc vượt qua các phép kiểm thử về chức năng thực hiện bởi nhóm thực hiện dự án, nhưng khách hàng khi kiểm thử sau cùng vẫn thất vọng vì bố cục màn hình nghèo nàn, thao tác không tự nhiên, không theo tập quán sử dụng của khách hàng v.v...

Gắn liền với giai đoạn Acceptance Test thường là một nhóm những dịch vụ và tài liệu đi kèm, phổ biến như hướng dẫn cài đặt, sử dụng v.v... Tất cả tài liệu đi kèm phải được cập nhật và kiểm thử chặt chẽ.

#### ***1.1.4.5. Một số cấp độ kiểm thử khác***

Ngoài các cấp độ trên, còn một số cấp độ kiểm thử khác như:

#### **Kiểm thử hồi quy – Regression Testing:**

Theo chuẩn IEEE610.12-90, kiểm thử hồi quy là “sự kiểm tra lại có lựa chọn của một hệ thống hay thành phần để xác minh là những sự thay đổi không gây ra những hậu quả không mong muốn...”. Trên thực tế, quan niệm này là chỉ ra rằng phần mềm mà đã qua được các kiểm tra trước đó vẫn có thể được kiểm tra lại. Beizer định nghĩa đó là sự lặp lại các kiểm tra để chỉ ra rằng cách hoạt động của phần mềm không bị thay đổi, ngoại trừ tới mức như yêu cầu. Hiển nhiên là sự thỏa hiệp phải được thực hiện giữa sự đảm bảo được đưa ra bởi kiểm thử hồi quy mỗi lần thực hiện một sự thay đổi và những tài nguyên được yêu cầu thực hiện điều đó.

#### **Kiểm thử tính đúng – Correctness testing:**

Tính đúng đắn là yêu cầu tối thiểu của phần mềm, là mục đích chủ yếu của kiểm thử. Kiểm thử tính đúng sẽ cần một kiểu người đáng tin nào đó, để chỉ ra cách hoạt động đúng đắn từ cách hoạt động sai lầm. Kiểm thử viên có thể biết hoặc không biết các chi tiết bên trong của các modul phần mềm được kiểm thử, ví dụ luồng điều khiển, luồng dữ liệu, v.v .... Do đó, hoặc là quan điểm hộp trắng, hoặc là quan điểm hộp đen có thể được thực hiện trong kiểm thử phần mềm.

#### **1.1.5. Các phương pháp kiểm thử con người**

Hai phương pháp kiểm thử con người chủ yếu là Code Inspections và Walkthroughs. Hai phương pháp này bao gồm một nhóm người đọc và kiểm tra theo mã lệnh của chương trình. Mục tiêu của chúng là để tìm ra lỗi mà không gõ lỗi.

Một Inspection hay Walkthrough là 1 sự cải tiến của phương pháp kiểm tra mà lập trình viên đọc chương trình của họ trước khi kiểm thử nó. Inspections và Walkthroughs hiệu quả hơn là bởi vì những người khác sẽ kiểm thử chương trình tốt hơn chính tác giả của chương trình đó.

Inspections/Walkthroughs và kiểm thử bằng máy tính bổ sung cho nhau. Hiệu quả tìm lỗi sẽ kém đi nếu thiếu đi 1 trong 2 phương pháp. Và đối với việc sửa đổi chương trình cũng nên sử dụng các phương pháp kiểm thử này cũng như các kỹ thuật kiểm thử hồi quy

#### ***1.1.5.1. Tổng duyệt – Walkthrough***

Walkthrough là một thuật ngữ mô tả sự xem xét kỹ lưỡng của một quá trình ở mức trừu tượng trong đó nhà thiết kế hay lập trình viên lãnh đạo các thành viên trong nhóm và những người có quan tâm khác thông qua một sản phẩm phần mềm, và những người tham gia đặt câu hỏi, và ghi chú những lỗi có thể có, sự vi phạm các chuẩn phát triển và các vấn đề khác. Walkthrough mã lệnh là 1 tập các thủ tục và các công nghệ dò lỗi cho việc đọc nhóm mã lệnh. Trong một Walkthrough, nhóm các nhà phát triển – với 3 hoặc 4 thành viên là tốt nhất – thực hiện xét duyệt lại. Chỉ 1 trong các thành viên là tác giả của chương trình.

Một ưu điểm khác của walkthroughs, hiệu quả trong chi phí gỡ lỗi, là 1 thực tế mà khi một lỗi được tìm thấy, nó thường được định vị chính xác trong mã lệnh. Thêm vào đó, phương pháp này thường tìm ra 1 tập các lỗi, cho phép sau đó các lỗi đó được sửa tất cả với nhau. Mặt khác, kiểm thử dựa trên máy tính, chỉ tìm ra triệu chứng của lỗi (chương trình không kết thúc hoặc đưa ra kết quả vô nghĩa), và các lỗi thường được tìm ra và sửa lần lượt từng lỗi một.

#### ***1.1.5.2. Thanh tra mã nguồn – Code Inspection***

Thanh tra mã nguồn là 1 tập hợp các thủ tục và các kỹ thuật dò lỗi cho việc đọc các nhóm mã lệnh. Một nhóm kiểm duyệt thường gồm 4 người. Một trong số đó đóng vai trò là người điều tiết – một lập trình viên lão luyện và không được là tác giả của chương trình và phải không quen với các chi tiết của chương trình. Người điều tiết có nhiệm vụ: phân phối nguyên liệu và lập lịch cho các buổi kiểm duyệt, chỉ đạo phiên làm việc, ghi lại tất cả các lỗi được tìm thấy và đảm bảo là các lỗi sau đó được sửa. Thành viên thứ hai là một lập trình viên. Các thành viên còn lại trong nhóm thường là nhà thiết kế của chương trình (nếu nhà thiết kế khác lập trình viên) và một chuyên viên kiểm thử.

### **1.2. Nguyên tắc kiểm thử phần mềm**

Để kiểm thử đạt hiệu quả thì khi tiến hành kiểm thử phần mềm cần phải tuân thủ một số quy tắc sau:

**Quy tắc 1:** Một phần quan trọng của 1 ca kiểm thử là định nghĩa của đầu ra hay kết quả mong muốn.

**Quy tắc 2:** Lập trình viên nên tránh tự kiểm tra chương trình của mình.

**Quy tắc 3:** Nhóm lập trình không nên kiểm thử chương trình của chính họ.

**Quy tắc 4:** Kiểm tra thấu đáo mọi kết quả của mỗi kiểm tra.

**Quy tắc 5:** Các ca kiểm thử phải được viết cho các trạng thái đầu vào không hợp lệ và không mong muốn, cũng như cho các đầu vào hợp lệ và mong muốn.

**Quy tắc 6:** Khảo sát 1 chương trình để xem liệu chương trình có thực hiện cái mà nó cần thực hiện chỉ là 1 phần, phần còn lại là xem liệu chương trình có thực hiện cái mà nó không cần phải thực hiện hay không.

**Quy tắc 7:** Tránh các ca kiểm thử băng quơ trừ khi chương trình thực sự là 1 chương trình băng quơ.

**Quy tắc 8:** Không dự kiến kết quả của kiểm thử theo giả thiết ngầm là không tìm thấy lỗi.

**Quy tắc 9:** Xác suất tồn tại lỗi trong 1 đoạn chương trình là tương ứng với số lỗi đã tìm thấy trong đoạn đó.

**Quy tắc 10:** Kiểm thử là 1 nhiệm vụ cực kỳ sáng tạo và có tính thử thách trí tuệ.



## **CHƯƠNG 2: THIẾT KẾ TEST – CASE**

### **2.1. Khái niệm**

Thiết kế test – case trong kiểm thử phần mềm là quá trình xây dựng các phương pháp kiểm thử có thể phát hiện lỗi, sai sót, khuyết điểm của phần mềm để xây dựng phần mềm đạt tiêu chuẩn.

### **2.2. Vai trò của thiết kế test – case**

- Tạo ra các ca kiểm thử tốt nhất có khả năng phát hiện ra lỗi, sai sót của phần mềm một cách nhiều nhất.
- Tạo ra các ca kiểm thử có chi phí rẻ nhất, đồng thời tốn ít thời gian và công sức nhất.

### **2.3. Quy trình của thiết kế test – case**

Một trong những lý do quan trọng nhất trong kiểm thử chương trình là thiết kế và tạo ra các ca kiểm thử - các Test case có hiệu quả. Với những ràng buộc về thời gian và chi phí đã cho, thì vấn đề then chốt của kiểm thử trở thành: Tập con nào của tất cả ca kiểm thử có thể có khả năng tìm ra nhiều lỗi nhất?

Thông thường, phương pháp kém hiệu quả nhất là kiểm tra tất cả đầu vào ngẫu nhiên – quá trình kiểm thử một chương trình bằng việc chọn ngẫu nhiên một tập con các giá trị đầu vào có thể. Về mặt khả năng tìm ra nhiều lỗi nhất, tập hợp các ca kiểm thử được chọn ngẫu nhiên có rất ít cơ hội là tập hợp tối ưu hay gần tối ưu. Sau đây là một số phương pháp để chọn ra một tập dữ liệu kiểm thử một cách thông minh.

Để kiểm thử hộp đen và kiểm thử hộp trắng một cách thấu đáo là không thể. Do đó, một chiến lược kiểm thử hợp lý là chiến lược có thể kết hợp sức mạnh của cả hai phương pháp trên: Phát triển 1 cuộc kiểm thử nghiêm ngặt vừa bằng việc sử dụng các phương pháp thiết kế ca kiểm thử hướng hộp đen nào đó và sau đó bổ sung thêm những ca kiểm thử này bằng việc khảo sát tính logic của chương trình, sử dụng phương pháp hộp trắng.

Những chiến lược kết hợp đó bao gồm:

Hộp đen	Hộp trắng
– Phân lớp tương đương	– Bao phủ câu lệnh
– Phân tích giá trị biên	– Bao phủ quyết định
– Đồ thị nguyên nhân kết quả	– Bao phủ điều kiện
– Đoán lỗi	– Bao phủ điều kiện - quyết định
	– Bao phủ đa điều kiện

*Bảng 2.1 Phương pháp thiết kế ca kiểm thử*

Mỗi phương pháp có những ưu điểm cũng như khuyết điểm riêng, do đó để có được tập các ca kiểm thử tối ưu, chúng ta cần kết hợp hầu hết các phương pháp. Quy trình thiết kế các ca kiểm thử sẽ bắt đầu bằng việc phát triển các ca kiểm thử sử dụng phương pháp hộp đen và sau đó phát triển bổ sung các ca kiểm thử cần thiết với phương pháp hộp trắng.

### **2.3.1. Kiểm thử hộp đen**

#### **2.3.1.1. Phân lớp tương đương – Equivalence Partitioning**

Phân lớp tương đương là một phương pháp kiểm thử hộp đen chia miền đầu vào của một chương trình thành các lớp dữ liệu, từ đó suy dẫn ra các ca kiểm thử. Phương pháp này cố gắng xác định ra một ca kiểm thử mà làm lộ ra một lớp lỗi, do đó làm giảm tổng số các trường hợp kiểm thử phải được xây dựng.

Thiết kế ca kiểm thử cho phân lớp tương đương dựa trên sự đánh giá về các lớp tương đương với một điều kiện vào. Lớp tương đương biểu thị cho tập các trạng thái hợp lệ hay không hợp lệ đối với điều kiện vào.

Một cách xác định tập con này là để nhận ra rằng 1 ca kiểm thử được lựa chọn tốt cũng nên có 2 đặc tính khác:

1. Giảm thiểu số lượng các ca kiểm thử khác mà phải được phát triển để hoàn thành mục tiêu đã định của kiểm thử “hợp lý”.

2. Bao phủ một tập rất lớn các ca kiểm thử có thể khác. Tức là, nó nói cho chúng ta một thứ gì đó về sự có mặt hay vắng mặt của những lỗi qua tập giá trị đầu vào cụ thể.

Thiết kế Test-case bằng phân lớp tương đương tiến hành theo 2 bước:

- (1). Xác định các lớp tương đương
- (2). Xác định các ca kiểm thử.

### **Xác định các lớp tương đương:**

Các lớp tương đương được xác định bằng cách lấy mỗi trạng thái đầu vào (thường là 1 câu hay 1 cụm từ trong đặc tả) và phân chia nó thành 2 hay nhiều nhóm (có thể sử dụng bảng 2.2 để liệt kê các lớp tương đương).

<b>Điều kiện bên ngoài</b>	<b>Các lớp tương đương hợp lệ</b>	<b>Các lớp tương đương không hợp lệ</b>

*Bảng 2.2 Mẫu liệt kê các lớp tương đương*

Chú ý là hai kiểu lớp tương đương được xác định: lớp tương đương hợp lệ mô tả các đầu vào hợp lệ của chương trình, và lớp tương đương không hợp lệ mô tả tất cả các trạng thái có thể khác của điều kiện (Ví dụ: các giá trị đầu vào không đúng). Với 1 đầu vào hay điều kiện bên ngoài đã cho, việc xác định các lớp tương đương hầu như là 1 quy trình mang tính kinh nghiệm. Để xác định các lớp tương đương có thể áp dụng tập các nguyên tắc dưới đây:

- Nếu 1 trạng thái đầu vào định rõ giới hạn của các giá trị, xác định 1 lớp tương đương hợp lệ và 2 lớp tương đương không hợp lệ.
- Nếu 1 trạng thái đầu vào xác định số giá trị, xác định 1 lớp tương đương hợp lệ và 2 lớp tương đương bất hợp lệ.
- Nếu 1 trạng thái đầu vào chỉ định tập các giá trị đầu vào và chương trình sử dụng mỗi giá trị là khác nhau, xác định 1 lớp tương đương hợp lệ cho mỗi loại và 1 lớp tương đương không hợp lệ.

- Nếu 1 trạng thái đầu vào chỉ định một tình huống “chắc chắn – must be”, xác định 1 lớp tương đương hợp lệ và 1 lớp tương đương không hợp lệ.

Nếu có bất kỳ lý do nào để tin rằng chương trình không xử lý các phần tử trong cùng 1 lớp là như nhau, thì hãy chia lớp tương đương đó thành các lớp tương đương nhỏ hơn.

### **Xác định các ca kiểm thử:**

Với các lớp tương đương xác định được ở bước trên, bước thứ hai là sử dụng các lớp tương đương đó để xác định các ca kiểm thử. Quá trình này như sau:

- Gán 1 số duy nhất cho mỗi lớp tương đương.
- Cho đến khi tất cả các lớp tương đương hợp lệ được bao phủ bởi (hợp nhất thành) các ca kiểm thử, viết 1 ca kiểm thử mới bao phủ càng nhiều các lớp tương đương đó chưa được bao phủ càng tốt.
- Cho đến khi các ca kiểm thử của bạn đã bao phủ tất cả các lớp tương đương không hợp lệ, viết 1 ca kiểm thử mà bao phủ một và chỉ một trong các lớp tương đương không hợp lệ chưa được bao phủ.
- Lý do mà mỗi ca kiểm thử riêng bao phủ các trường hợp không hợp lệ là vì các kiểm tra đầu vào không đúng nào đó che giấu hoặc thay thế các kiểm tra đầu vào không đúng khác.

Mặc dù việc phân lớp tương đương là rất tốt khi lựa chọn ngẫu nhiên các ca kiểm thử, nhưng nó vẫn có những thiếu sót. Ví dụ: nó bỏ qua các kiểu test – case có lợi nào đó. Hai phương pháp tiếp theo, phân tích giá trị biên và đồ thị nguyên nhân – kết quả, bao phủ được nhiều những thiếu sót này.

#### **2.3.1.2. Phân tích giá trị biên – Boundary Value Analysis**

Kinh nghiệm cho thấy các ca kiểm thử mà khảo sát tỷ mỷ các điều kiện biên có tỷ lệ phần trăm cao hơn các ca kiểm thử khác. Các điều kiện biên là những điều kiện mà các tình huống ngay tại, trên và dưới các cạnh của các lớp tương đương đầu vào và các lớp tương đương đầu ra. Phân tích các giá trị biên là phương pháp thiết kế ca kiểm thử bổ sung thêm cho phân lớp tương đương, nhưng khác với phân lớp tương đương ở 2 khía cạnh:

- Phân tích giá trị biên không lựa chọn phần tử bất kỳ nào trong 1 lớp tương đương là điển hình, mà nó yêu cầu là 1 hay nhiều phần tử được lựa chọn như vậy mà mỗi cạnh của lớp tương đương đó chính là đối tượng kiểm tra.
- Ngoài việc chỉ tập trung chú ý vào các trạng thái đầu vào (không gian đầu vào), các ca kiểm thử cũng nhận được bằng việc xem xét không gian kết quả (các lớp tương đương đầu ra).

Phân tích giá trị biên yêu cầu óc sáng tạo và lượng chuyên môn hóa nhất định và nó là một quá trình mang tính kinh nghiệm rất cao. Tuy nhiên, có một số quy tắc chung như sau:

1. Nếu 1 trạng thái đầu vào định rõ giới hạn của các giá trị, hãy viết các ca kiểm thử cho các giá trị cuối của giới hạn, và các ca kiểm thử đầu vào không hợp lệ cho các trường hợp vừa ra ngoài phạm vi.
2. Nếu 1 trạng thái đầu vào định rõ số lượng giá trị, hãy viết các ca kiểm thử cho con số lớn nhất và nhỏ nhất của các giá trị và một giá trị trên, một giá trị dưới những giá trị này
3. Sử dụng quy tắc 1 cho mỗi trạng thái đầu vào. Ví dụ: nếu 1 trang web mua vé ngày khởi hành được quy định mức tối thiểu được chọn ngày hiện tại là ngày 19/05/2019 và ngày tối đa có thể chọn là ngày cuối cùng trong tháng là ngày 31/05/2019, hãy viết các ca kiểm thử mà ngày khởi hành là ngày 19/05/2019 và ngày 31/05/2019, ngày khởi hành trước ngày 19/05/2019 ví dụ ngày 18/05/2019 và ngày khởi hành sau ngày 31/05/2019 ví dụ ngày 01/06/2019 . Chú ý là việc xem xét giới hạn của không gian kết quả là quan trọng vì không phải lúc nào các biên của miền đầu vào cũng mô tả cùng một tập sự kiện như biên của giới hạn đầu ra (Ví dụ: xét chương trình con tính SIN). Ngoài ra, không phải lúc nào cũng có thể tạo ra 1 kết quả bên ngoài giới hạn đầu ra, nhưng tuy nhiên rất đáng để xem xét tiềm ẩn đó.
4. Sử dụng nguyên tắc 2 cho mỗi trạng thái đầu ra.
5. Nếu đầu vào hay đầu ra của 1 chương trình là tập được sắp thứ tự (Ví dụ: 1 file tuần tự hay 1 danh sách định tuyến hay 1 bảng) tập trung chú ý vào các phần tử đầu tiên và cuối cùng của tập hợp.
6. Sử dụng sự khéo léo của bạn để tìm các điều kiện biên.

### **2.3.1.3. Sử dụng đồ thị nguyên nhân - kết quả - Cause & Effect Graphing**

Một yếu điểm của phân tích giá trị biên và phân lớp tương đương là chúng không khảo sát sự kết hợp của các trường hợp đầu vào. Việc kiểm tra sự kết hợp đầu vào không phải là một nhiệm vụ đơn giản bởi vì nếu bạn phân lớp tương đương các trạng thái đầu vào, thì số lượng sự kết hợp thường là rất lớn. Nếu bạn không có cách lựa chọn có hệ thống một tập con các trạng thái đầu vào, có lẽ bạn sẽ chọn ra một tập tùy hứng các điều kiện, điều này có thể dẫn tới việc kiểm thử không có hiệu quả.

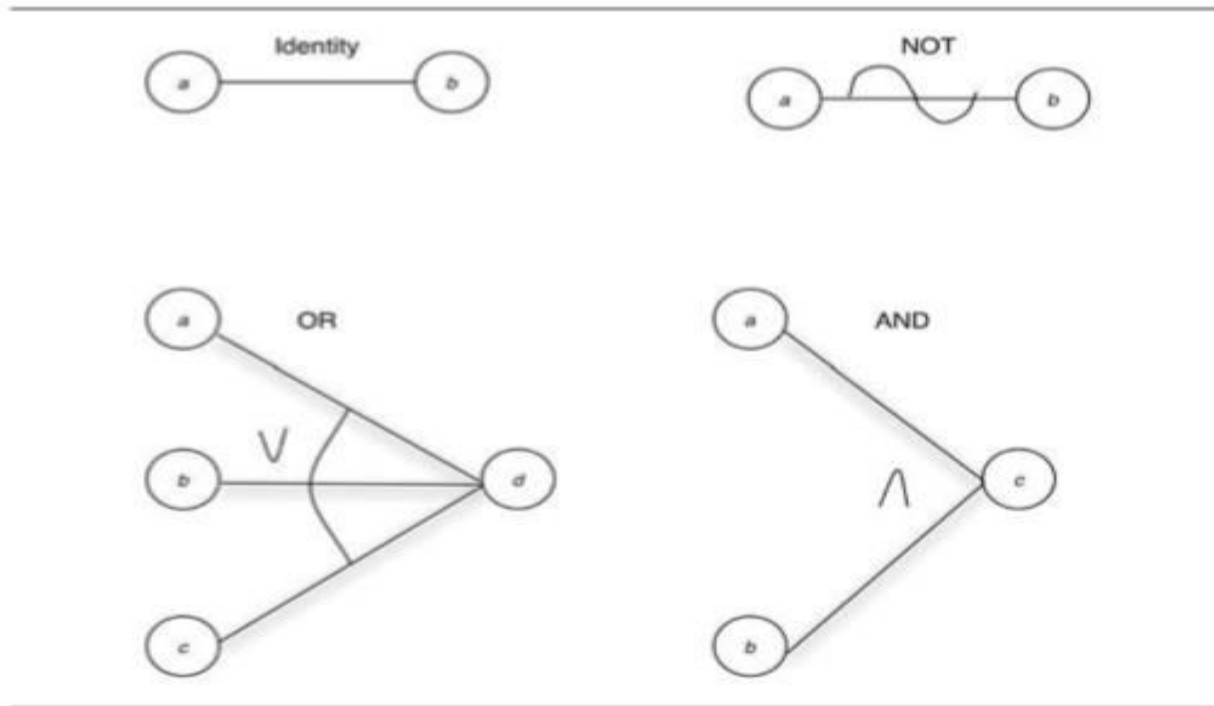
Đồ thị nguyên nhân – kết quả hỗ trợ trong việc lựa chọn một cách có hệ thống tập các ca kiểm thử có hiệu quả cao. Nó có tác động có lợi ảnh hưởng tới việc chỉ ra tình trạng chưa đầy đủ và nhập nhằng trong đặc tả. Nó cung cấp cả cách biểu diễn chính xác cho các điều kiện logic và hành động tương ứng.

Quá trình dưới đây được sử dụng để xây dựng được các test – case:

1. Đặc tả được chia thành các phần có thể thực hiện được. Điều này là cần thiết bởi vì đồ thị nguyên nhân – kết quả trở nên khó sử dụng khi được sử dụng trên những đặc tả lớn.
2. Nguyên nhân và kết quả trong các đặc tả được nhận biết. Một nguyên nhân là một trạng thái đầu vào nhất định hay một lớp tương đương của các trạng thái đầu vào. Một kết quả là một trạng thái đầu ra hay 1 sự biến đổi hệ thống (kết quả còn lại mà 1 đầu vào có trạng thái của 1 chương trình hay hệ thống). Bạn nhận biết nguyên nhân và kết quả bằng việc đọc từng từ của đặc tả và gạch chân các từ hoặc cụm từ mô tả nguyên nhân và kết quả. Khi được nhận biết, mỗi nguyên nhân và kết quả được gán cho 1 số duy nhất.
3. Xây dựng đồ thị nguyên nhân – kết quả bằng cách phát triển và biến đổi nội dung ngữ nghĩa của đặc tả thành đồ thị Boolean nối giữa nguyên nhân và kết quả.
4. Đồ thị được diễn giải với các ràng buộc mô tả những sự kết hợp của nguyên nhân và/hoặc kết quả là không thể vì các ràng buộc ngữ nghĩa và môi trường.
5. Bằng việc dò theo các điều kiện trạng thái trong đồ thị một cách cẩn thận, bạn chuyển đổi đồ thị thành một bảng quyết định mục vào giới hạn. Mỗi cột trong bảng mô tả một ca kiểm thử.

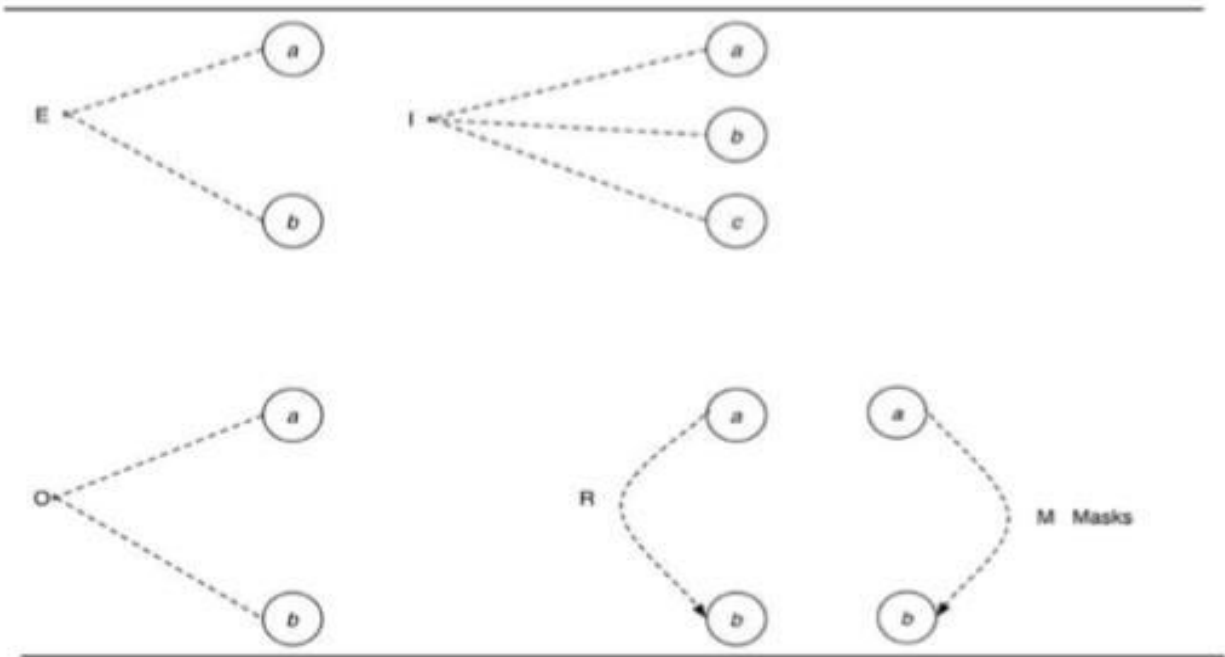
6. Các cột trong bảng quyết định được chuyển thành các ca kiểm thử.

Ký hiệu cơ bản cho đồ thị được chỉ ra trong hình 2.1. Tưởng tượng mỗi nút có giá trị là 0 hoặc 1; 0 mô tả trạng thái vắng mặt và 1 mô tả trạng thái có mặt. Hàm đồng nhất nói là nếu a là 1 thì b là 1; ngược lại, b là 0. Hàm not là nói nếu a là 1 thì b là 0; ngược lại thì b là 1. Hàm or khẳng định rằng nếu a hoặc b hoặc c là 1, thì d là 1; ngược lại d là 0. Hàm and khẳng định nếu cả a và b là 1 thì c là 1; ngược lại c là 0.



Hình 2.1 Các ký hiệu đồ thị nguyên nhân – kết quả cơ bản

Trong hầu hết các chương trình, sự kết hợp nào đó của một số nguyên nhân là không thể bởi vì lý do ngữ nghĩa và môi trường (Ví dụ: một ký tự không thể đồng thời vừa là “A” vừa là “B”). Khi đó, ta sử dụng ký hiệu trong Hình 2.2. Ràng buộc E (Exclude – loại trừ) khẳng định rằng tối đa, chỉ có hoặc a hoặc b có thể là 1 (a và b không thể đồng thời là 1). Ràng buộc I (Include – bao hàm) khẳng định ít nhất một trong a, b hoặc c phải luôn luôn là 1 (a, b hoặc c không thể đồng thời là 0). Ràng buộc O (Only – chỉ một) khẳng định một và chỉ một hoặc a hoặc b phải là 1. Ràng buộc R (Request – yêu cầu) khẳng định rằng khi a là 1, thì b phải là 1 (Ví dụ: không thể có trường hợp a là 1, còn b là 0). Ràng buộc M (Mask – mặt nạ) khẳng định là nếu kết quả a là 1, kết quả b sẽ bắt buộc phải là 0.



*Hình 2.2 Các ký hiệu ràng buộc*

Bước tiếp theo là tạo bảng quyết định mục vào giới hạn – limited-entry decision table. Tương tự với các bảng quyết định, thì nguyên nhân chính là các điều kiện và kết quả chính là các hành động. Quy trình được sử dụng là như sau:

1. Chọn một kết quả để là trạng thái có mặt (1).
2. Lần ngược trở lại đồ thị, tìm tất cả những sự kết hợp của các nguyên nhân (đối tượng cho các ràng buộc) mà sẽ thiết lập kết quả này thành 1.
3. Tạo một cột trong bảng quyết định cho mỗi sự kết hợp nguyên nhân.
4. Với mỗi sự kết hợp, hãy quy định trạng thái của tất cả các kết quả khác và đặt chúng vào mỗi cột.

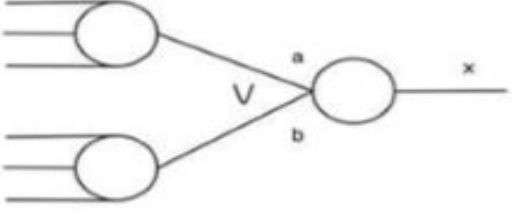
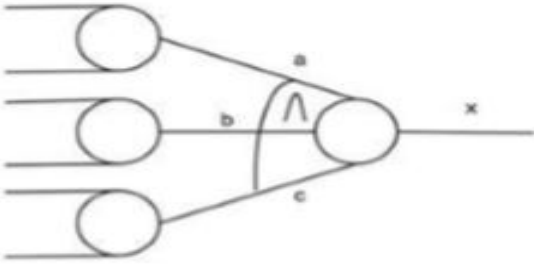
Trong khi biểu diễn bước 2, cần quan tâm các vấn đề sau:

1. Khi lần ngược trở lại qua một nút or mà đầu ra của nó là 1, không bao giờ thiết lập nhiều hơn 1 đầu vào cho nút or là 1 một cách đồng thời. Điều này được gọi là path sensitizing – làm nhạy đường đi. Mục tiêu của nó là để ngăn chặn dò lỗi thất bại vì một nguyên nhân che đi một nguyên nhân khác.
2. Khi lần ngược trở lại qua một nút and mà đầu ra của nó là 0, dĩ nhiên, phải liệt kê tất cả các sự kết hợp đầu vào dẫn tới đầu ra 0. Tuy nhiên, nếu bạn



đang khảo sát trạng thái mà 1 đầu ra là 0 và một hay nhiều đầu ra khác là 1, thì không nhất thiết phải liệt kê tất cả các điều kiện mà dưới điều kiện đó các đầu vào khác có thể là 1.

3. Khi lần ngược trở lại qua một nút and mà đầu ra của nó là 0, chỉ cần liệt kê 1 điều kiện trong đó tất cả đầu vào bằng 0. (Nếu nút and ở chính giữa của đồ thị như vậy thì tất cả các đầu vào của nó xuất phát từ các nút trung gian khác, có thể có quá nhiều trạng thái mà trong trạng thái đó tất cả các đầu vào của nó bằng 0.)

	<ul style="list-style-type: none"> <li>• Nếu <math>x=1</math>, không quan tâm về trường hợp <math>a=b=1</math> (sự xem xét thứ 1)</li> <li>• Nếu <math>x=0</math>, liệt kê tất cả các trường hợp trong đó <math>a=b=0</math>.</li> </ul>
	<ul style="list-style-type: none"> <li>• Nếu <math>x=1</math>, liệt kê tất cả các trường hợp trong đó <math>a=b=c=1</math>.</li> <li>• Nếu <math>x=0</math>, bao gồm chỉ 1 trường hợp mà <math>a=b=c=0</math> (sự xem xét 3). Đối với các trạng thái mà abc là 001, 010, 100, 011, 101 và 110, bao gồm chỉ 1 trường hợp mỗi trạng thái (sự xem xét 2).</li> </ul>

*Hình 2.3 Những xem xét được sử dụng khi dò theo đồ thị*

Những sự xem xét này có thể xuất hiện thất thường, nhưng chúng có một mục đích rất quan trọng: để giảm bớt các kết quả được kết hợp của đồ thị. Chúng liệt kê các trường hợp mà hướng về các ca kiểm thử ít có lợi. Nếu các ca kiểm thử ít có lợi không được liệt kê, một đồ thị nguyên nhân – kết quả lớn sẽ tạo ra một số lượng ca kiểm thử cực kỳ lớn. Nếu số lượng các ca kiểm thử trên thực tế là quá lớn, bạn sẽ chọn ra 1 tập con nào đó, nhưng không đảm bảo là các ca kiểm thử ít có lợi sẽ là những ca kiểm thử được liệt kê. Do đó, tốt hơn hết là liệt kê chúng trong suốt quá trình phân tích của đồ thị.

## ***NHẬN XÉT***

Vẽ đồ thị nguyên nhân – kết quả là phương pháp tạo các ca kiểm thử có hệ thống mô tả sự kết hợp của các điều kiện. Sự thay đổi sẽ là 1 sự lựa chọn kết hợp không thể dự tính trước, nhưng khi thực hiện như vậy, có vẻ như bạn sẽ bỏ sót nhiều ca kiểm thử “thú vị” được xác định bằng đồ thị nguyên nhân – kết quả .

Vì vẽ đồ thị nguyên nhân – kết quả yêu cầu chuyển một đặc tả thành một mạng logic Boolean, nó cung cấp một triển vọng khác và sự hiểu biết sâu sắc hơn nữa về đặc tả. Trên thực tế, sự phát triển của 1 đồ thị nguyên nhân – kết quả là cách hay để khám phá sự mơ hồ và chưa đầy đủ trong các đặc tả.

Mặc dù việc vẽ đồ thị nguyên nhân – kết quả tạo ra tập các ca kiểm thử hữu dụng, nhưng thông thường nó không tạo ra tất cả các ca kiểm thử hữu dụng mà có thể được nhận biết. Ngoài ra, đồ thị nguyên nhân – kết quả không khảo sát thỏa đáng các điều kiện giới hạn. Dĩ nhiên, bạn có thể cố gắng bao phủ các điều kiện giới hạn trong suốt quá trình.

Tuy nhiên, vấn đề trong việc thực hiện điều này là nó làm cho đồ thị rất phức tạp và dẫn tới số lượng rất lớn các ca kiểm thử. Vì thế, tốt nhất là xét 1 sự phân tích giá trị giới hạn tách rời nhau.

Vì đồ thị nguyên nhân – kết quả làm chúng ta mất thời gian trong việc chọn các giá trị cụ thể cho các toán hạng, nên các điều kiện giới hạn có thể bị pha trộn thành các ca kiểm thử xuất phát từ đồ thị nguyên nhân – kết quả. Vì vậy, chúng ta đạt được một tập các ca kiểm thử nhỏ nhưng hiệu quả mà thỏa mãn cả 2 mục tiêu.

Khía cạnh khó nhất của kỹ thuật này là quá trình chuyển đổi đồ thị thành bảng quyết định. Quá trình này có tính thuật toán, tức là bạn có thể tự động hóa nó bằng việc viết 1 chương trình. Trên thị trường đã có một vài chương trình thương mại tồn tại giúp cho quá trình chuyển đổi này.

### ***2.3.1.4. Đoán lỗi – Error Guessing***

Một kỹ thuật thiết kế test-case khác là error guessing – đoán lỗi. Tester được đưa cho 1 chương trình đặc biệt, họ phỏng đoán, cả bằng trực giác và kinh nghiệm, các loại lỗi có thể và sau đó viết các ca kiểm thử để đưa ra các lỗi đó.

Thật khó để đưa ra một quy trình cho kỹ thuật đoán lỗi vì nó là một quy trình có tính trực giác cao và không thể dự đoán trước. Ý tưởng cơ bản là liệt kê một danh sách các lỗi có thể hay các trường hợp dễ xảy ra lỗi và sau đó viết các ca kiểm thử dựa trên danh sách đó. Một ý tưởng khác để xác định các ca kiểm thử có liên đới với các giả định mà lập trình viên có thể đã thực hiện khi đọc đặc tả (tức là, những thứ bị bỏ sót khỏi đặc tả, hoặc là do tình cờ, hoặc là vì người viết có cảm giác những đặc tả đó là rõ ràng). Nói cách khác, bạn liệt kê những trường hợp đặc biệt đó mà có thể đã bị bỏ sót khi chương trình được thiết kế.

### **2.3.2. Kiểm thử hộp trắng - Kiểm thử bao phủ logic**

### **2.3.3. Chiến lược**

Các phương pháp thiết kế test-case đã được thảo luận có thể được kết hợp thành một chiến lược toàn diện. Vì mỗi phương pháp có thể đóng góp 1 tập riêng các ca kiểm thử hữu dụng, nhưng không cái nào trong số chúng tự nó đóng góp một tập trọn vẹn các ca kiểm thử. Chiến lược hợp lý như sau:

1. Nếu đặc tả có chứa sự kết hợp của các điều kiện đầu vào, hãy bắt đầu với việc vẽ đồ thị nguyên nhân – kết quả.
2. Trong trường hợp bất kỳ, sử dụng phương pháp phân tích giá trị biên. Hãy nhớ rằng đây là một sự phân tích của các biên đầu vào và đầu ra. Phương pháp phân tích giá trị biên mang lại một tập các điều kiện kiểm tra bổ sung, và rất nhiều hay toàn bộ các điều kiện này có thể được hợp nhất thành các kiểm thử nguyên nhân – kết quả.
3. Xác định các lớp tương đương hợp lệ và không hợp lệ cho đầu vào và đầu ra, và bổ sung các ca kiểm thử được xác định trên nếu cần thiết.
4. Sử dụng kỹ thuật đoán lỗi để thêm các ca kiểm thử thêm vào.
5. Khảo sát tính logic của chương trình liên quan đến tập các ca kiểm thử. Sử dụng tiêu chuẩn bao phủ quyết định, bao phủ điều kiện, bao phủ quyết định/điều kiện, hay bao phủ đa điều kiện ( trong đó bao phủ đa điều kiện là được sử dụng nhiều nhất ). Nếu tiêu chuẩn bao phủ không đạt được bởi các ca kiểm thử được xác định trong bốn bước trước, và nếu việc đạt được tiêu chuẩn là không thể ( tức là, những sự kết hợp chắc chắn của các điều kiện

có thể là không thể tạo vì bản chất của chương trình), hãy thêm vào các ca kiểm thử có khả năng làm cho thỏa mãn tiêu chuẩn.

Tuy việc sử dụng chiến lược này sẽ không đảm bảo rằng tất cả các lỗi sẽ được tìm thấy, nhưng nó đã được xác minh là đại diện cho một sự thỏa thuận hợp lý.

#### **2.3.4. Kiểm thử API**

##### **2.3.4.1. Định nghĩa**

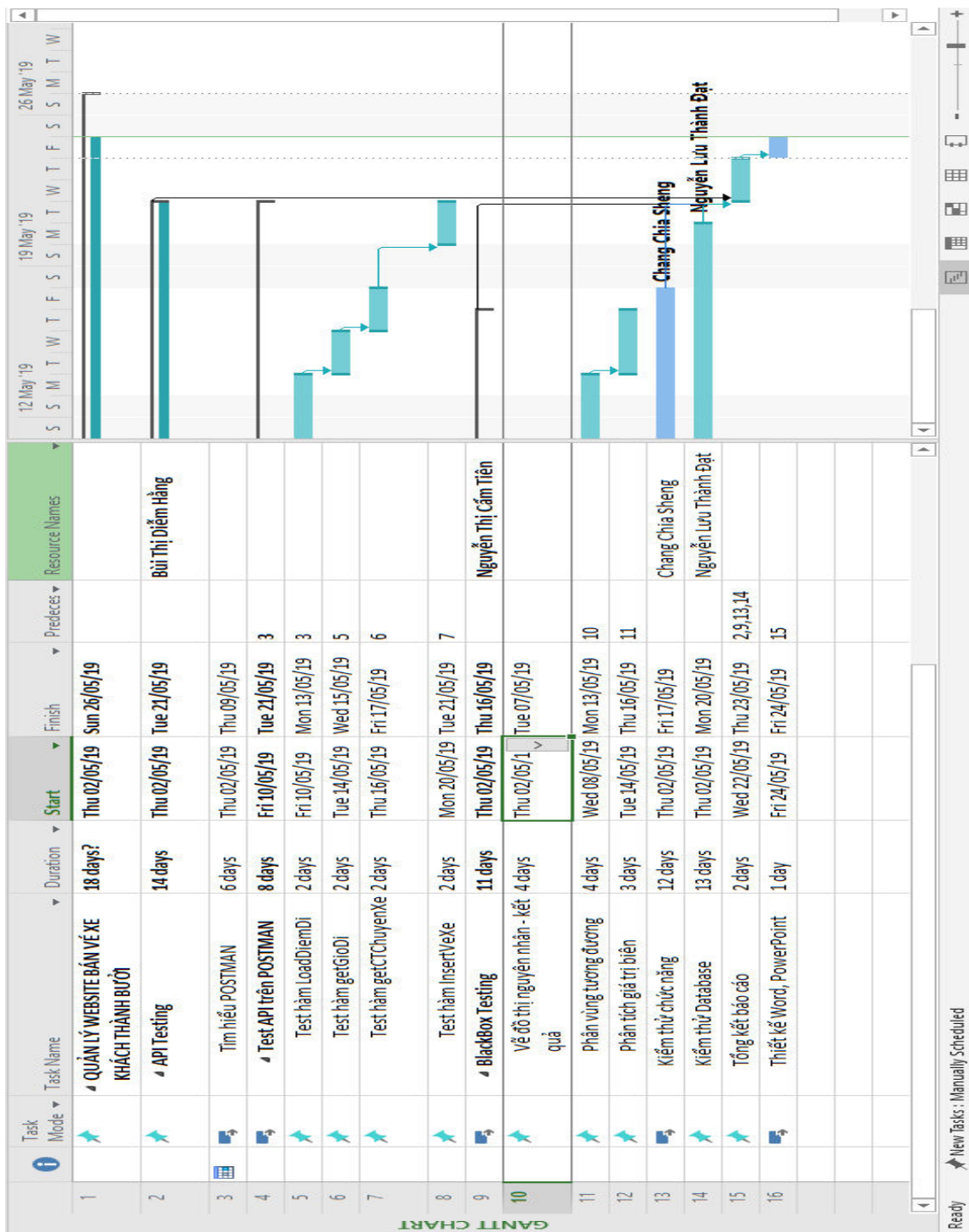
API testing là một loại kiểm thử phần mềm liên quan đến việc kiểm thử các giao diện lập trình ứng dụng (APIs) một cách trực tiếp và là một phần của kiểm thử tích hợp để xác định xem hệ thống có đáp ứng các yêu cầu về tính năng, độ tin cậy, hiệu suất và bảo mật. Vì các API không có GUI nên kiểm thử API được thực hiện ở tầng nghiệp vụ (business layer) . Trong quá trình kiểm thử API, dữ liệu được trao đổi từ XML hoặc JSON thông qua các yêu cầu và phản hồi HTTP ( HTTP requests and responses). Đây là những công nghệ độc lập và sẽ làm việc với nhiều ngôn ngữ lập trình và công nghệ khác nhau.

##### **2.3.4.2. Lý do kiểm thử API**

Kiểm thử ứng dụng sớm và không cần giao diện người dùng: Khi bạn tìm thấy lỗi càng muộn thì càng mất nhiều thời gian, công sức để fix nó. API testing đưa người kiểm thử tham gia sớm vào vòng đời phát triển sản phẩm. Với API testing, bạn có thể bắt đầu kiểm thử ứng dụng sớm ngay cả khi không có giao diện người dùng. Điều này giúp xác định và khắc phục sớm các vấn đề trong vòng đời phát triển, nếu không thì sẽ tốn kém để khắc phục khi được xác định trong quá trình kiểm thử GUI. Ưu điểm của việc kiểm thử API là rất nhiều logic có thể được kiểm tra mà không bị phụ thuộc vào UI.

# CHƯƠNG 3: ÁP DỤNG

## 3.1. WBS



Hình 3.1 Bảng phân công công việc WBS

### 3.2. Đặc tả

Các chức năng chính của phần mềm:

- Quản lý tài xế, nhân viên.
- Quản lý xe, bảo trì xe.
- Quản lý tuyến/chuyến/chi tiết mỗi chuyến.
- Phân công lịch trình xe – tài xế.
- Quản lý đặt/bán vé.
- Thống kê doanh thu tuyến, chuyến, vé.

Website mua vé trực tuyến cho phép mua vé khi khách hàng điền và chọn đầy đủ thông tin bao gồm:

- Chọn điểm đi.
- Chọn điểm đến.
- Chọn ngày khởi hành.
- Chọn giờ khởi hành.
- Chọn ghế.
- Nhập thông tin khách hàng: họ tên, email, ngày sinh, di động.

Sau khi hoàn thành mua vé hệ thống sẽ gửi email về cho khách hàng thông qua email trong thông tin của khách. Khách hàng khi đến lấy vé tại quầy vé sẽ đưa nội dung email cho nhân viên xác nhận và lấy vé.

Những thông tin cần để mua vé trực tuyến phải thỏa mãn:

- Điểm đi và điểm đến không trùng nhau.
- Điểm đến phụ thuộc vào điểm đi. Một điểm đi sẽ có điểm đến cụ thể.
- Ngày khởi hành không được nhỏ hơn ngày hiện tại và không được chọn quá ngày cuối cùng của tháng hiện tại.
- Khi chọn ghế những ghế đã được mua thì không cho chọn, chọn ghế trống và không chọn quá 10 ghế và bắt buộc chọn ít nhất 1 ghế.

- Thông tin khách hàng không được để trống họ tên, email, di động. Định dạng email phải hợp lệ có chứa đuôi @gmail. Số di động phải chứa đủ 10 ký tự số không cho phép nhập chữ và ký tự đặc biệt, đầu số phải bắt đầu bằng số 0.

### **3.3. Kiểm thử hộp đen**

#### **3.3.1. Vẽ đồ thị nguyên nhân – kết quả**

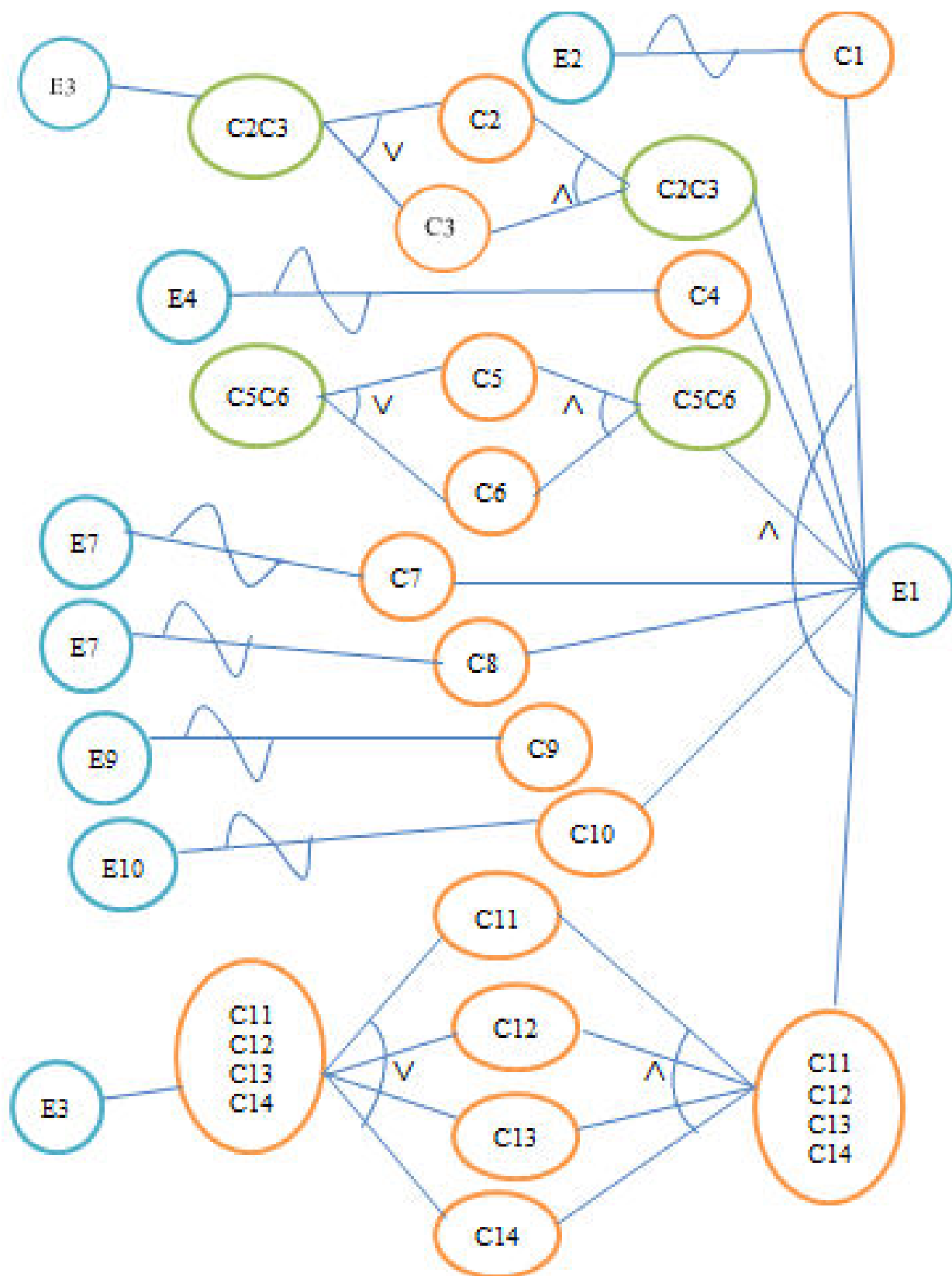
**Nguyên nhân là:**

- C1. Điểm đi và điểm đến không trùng nhau.
- C2. Ngày khởi hành lớn hơn ngày hiện tại.
- C3. Ngày khởi hành nhỏ hơn ngày hiện tại.
- C4. Không chọn ghế đã có người.
- C5. Không chọn nhiều hơn 10 ghế.
- C6. Không chọn nhỏ hơn 1 ghế.
- C7. Không để trống họ tên.
- C8. Không được để trống email.
- C9. Không được để trống số di động.
- C10. Nhập định dạng email hợp lệ.
- C11. Nhập số di động nhỏ hơn 10 số.
- C12. Nhập số di động lớn hơn 10 số.
- C13. Nhập số di động không có số 0 đầu.
- C14. Nhập số di động khác ký tự số.

**Kết quả là:**

- E1. Thông báo mua vé thành công và gửi email về địa chỉ email của khách hàng.
- E2. Thông báo điểm đi phải khác điểm đến.
- E3. Thông báo ngày khởi hành phải thuộc tháng hiện tại và không nhỏ hơn ngày hiện tại.
- E4. Thông báo ghế đã có người chọn.
- E5. Thông báo số ghế chọn không nhỏ hơn 1 và không lớn hơn 10.
- E6. Thông báo yêu cầu nhập họ tên.
- E7. Thông báo yêu cầu nhập email.
- E8. Thông báo yêu cầu nhập số di động.
- E9. Thông báo yêu cầu nhập email hợp lệ
- E10. Thông báo yêu cầu nhập số di động hợp lệ.





Hình 3.2 Đồ thị nguyên nhân – kết quả

Action	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	T11	T12	T13	T14	T14
C1	1	0													
C2	0		1												
C3	0			1											
C4	1				0										
C5	1					0									
C6	1						0								
C7	1							0							
C8	1								0						
C9	1									0					
C10	1										0				
C11	0											1			
C12	0												1		
C13	0													1	
C14	0														1
E1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
E2	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
E3	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0
E4	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
E5	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0
E6	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
E7	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
E8	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
E9	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
E10	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1

*Bảng 3.1 Bảng quyết định*

STT	Tên TestCase	Bước thực hiện	Kết quả mong đợi	Kết quả thực tế	Tình trạng
1	Kiểm tra mua vé thành công khi điền và chọn đầy đủ thông tin như: điểm đi, điểm đến, ngày khởi hành, giờ khởi hành, chọn ghế, điền thông tin khách hàng	1. Truy cập vào website mua vé 2. Chọn điểm đi là: “Tp.Hồ Chí Minh” 3. Chọn điểm đến là “Cần Thơ” 4. Chọn giờ khởi hành là: “7:30” 5. Chọn vị trí ghế là: “1” 6. Điền thông tin cá nhân : họ tên là :”Nguyễn Văn A”, email là: “ <a href="mailto:anguyenvan@gmail.com">anguyenvan@gmail.com</a> ”, di động là: “0903123456”, ngày sinh là:”20/01/1987”	Mua vé thành công và hệ thống gửi email xác nhận về địa chỉ email: “ <a href="mailto:anguyenvan@gmail.com">anguyenvan@gmail.com</a> ”	Mua vé thành công và hệ thống gửi email xác nhận về địa chỉ email: “ <a href="mailto:anguyenvan@gmail.com">anguyenvan@gmail.com</a> ”	PASS
2	Kiểm tra khi điểm đi trùng với điểm đến	1.Truy cập vào website mua vé 2.Chọn điểm đi là:”An Giang” 3. Chọn điểm đến là:”An Giang”	Hiện thị thông báo ”Điểm đi phải khác điểm đến!”	Hiện thị thông báo ”Điểm đi phải khác điểm đến!”	PASS
3	Kiểm tra khi chọn ngày khởi hành nhỏ hơn ngày hiện tại	1.Truy cập vào website mua vé 2.Giả sử ngày hiện tại là “19/05/2019”. Chọn ngày khởi hành nhỏ hơn ngày hiện tại là ngày: “18/05/2019”	Hiện thị thông báo “Ngày khởi hành không được lớn hơn ngày hiện tại!”	Hiện thị thông báo “Ngày khởi hành không được lớn hơn ngày hiện tại!”	PASS

4	Kiểm tra khi chọn ngày khởi hành lớn hơn ngày trong tháng hiện tại	1.Truy cập vào website mua vé 2.Giả sử ngày hiện tại là “19/05/2019”. Chọn ngày khởi hành nhỏ hơn ngày hiện tại là ngày: “20/06/2019” hơn ngày hiện tại	Hiện thị thông báo “Ngày khởi hành không được lớn hơn tháng hiện tại!”	Hiện thị thông báo “Ngày khởi hành không được lớn hơn tháng hiện tại!”	PASS
5	Kiểm tra khi chọn hơn 10 ghế	1.Truy cập vào website mua vé 2. Chọn thông tin tuyến xe 3. Chọn 11 ghế	Khi chọn ghế thứ 11 hiện thị thông báo “Chọn ít nhất 1 ghế và không chọn quá 10 ghế!”	Khi chọn ghế thứ 11 hiện thị thông báo “Chọn ít nhất 1 ghế và không chọn quá 10 ghế!”	PASS
6	Kiểm tra khi không chọn ghế	1.Truy cập vào website mua vé 2. Chọn thông tin tuyến xe 3. Không chọn ghế và chọn button tiếp tục	Hiện thị thông báo “Chọn ít nhất 1 ghế và không chọn quá 10 ghế”	Hiện thị thông báo “Chọn ít nhất 1 ghế và không chọn quá 10 ghế”	PASS
7	Kiểm tra khi nhập thông tin khách hàng và để trống họ tên	1.Truy cập vào website mua vé 2. Chọn thông tin tuyến, chuyến 3. Để trống họ tên, nhập email là: “ <a href="mailto:anguyenvan@gmail.com">anguyenvan@gmail.com</a> ”, di động là: “0903123456”, ngày sinh là: “01/08/1994” và chọn button tiếp tục	Hiện thị thông báo “Vui long nhập họ tên”	Hiện thị thông báo “Vui long nhập họ tên”	PASS

8	Kiểm tra khi nhập thông tin khách hàng và để trống email	1.Truy cập vào website mua vé 2. Chọn thông tin tuyến, chuyển 3. Để trống email, nhập họ tên là: “Nguyễn Văn A”, di động là: “0903123456”, ngày sinh là: “01/08/1994” và chọn button tiếp tục	Hiện thị thông báo “Vui lòng nhập email!”	Hiện thị thông báo “Vui lòng nhập email!”	PASS
9	Kiểm tra khi nhập thông tin khách hàng và để trống di động	1.Truy cập vào website mua vé 2. Chọn thông tin tuyến, chuyển 3. Để trống di động, nhập họ tên là: “Nguyễn Văn A”, nhập email là: “ <a href="mailto:anguyenvan@gmail.com">anguyenvan@gmail.com</a> ”, ngày sinh là: “01/08/1994” và chọn tiếp tục	Hiện thị thông báo “Vui lòng nhập số di động!”	Hiện thị thông báo “Vui lòng nhập số di động!”	PASS

10	Kiểm tra email không đúng định dạng khi nhập	1. Truy cập vào website mua vé 2. Chọn thông tin tuyến, chuyến 3. Nhập họ tên là: “Nguyễn Văn A”, nhập email là: “ <a href="mailto:anguyenvan@gmail.com">anguyenvan@gmail.com</a> ”, di động là “0903123456”, ngày sinh là: “01/08/1994” và chọn button tiếp tục	Hiển thị thông báo “Vui lòng nhập địa chỉ email đúng định dạng!”	Hiển thị thông báo “Vui lòng nhập địa chỉ email đúng định dạng!”	PASS
11	Kiểm tra khi nhập số di động nhỏ hơn 10 ký tự số	1. Truy cập vào website mua vé 2. Chọn thông tin tuyến, chuyến 3. Nhập họ tên là: “Nguyễn Văn A”, nhập email là: “ <a href="mailto:anguyenvan@gmail.com">anguyenvan@gmail.com</a> ”, di động chứa 9 ký tự số là “090312456”, ngày sinh là: “01/08/1994” và chọn button tiếp tục	Hiển thị thông báo “Số di phải chứa 10 ký tự số và bắt đầu bằng số 0!”	Hiển thị thông báo “Số di phải chứa 10 ký tự số và bắt đầu bằng số 0!”	PASS
12	Kiểm tra khi nhập số di động lớn hơn 10 ký tự số	1. Truy cập vào website mua vé 2. Chọn thông tin tuyến, chuyến 3. Nhập họ tên là: “Nguyễn Văn A”, nhập email là:	Hiển thị thông báo “Số di phải chứa 10 ký tự số và bắt đầu bằng số 0!”	Hiển thị thông báo “Số di phải chứa 10 ký tự số và bắt đầu bằng số 0!”	PASS

		“ <a href="mailto:anguyenvan.gmail.com">anguyenvan.gmail.com</a> ”, di động chứa 11 ký tự số là “09031245611”, ngày sinh là: “01/08/1994” và chọn button tiếp tục			
13	Kiểm tra khi nhập số di động không chứa số 0 đầu dãy số	1. Truy cập vào website mua vé 2. Chọn thông tin tuyến, chuyến 3. Nhập họ tên là: “Nguyễn Văn A”, nhập email là: “ <a href="mailto:anguyenvan.gmail.com">anguyenvan.gmail.com</a> ”, di động chứa 11 ký tự số là “1031245611”, ngày sinh là: “01/08/1994” và chọn button tiếp tục	Hiển thị thông báo “Số di phải chứa 10 ký tự số và bắt đầu bằng số 0!”	Hiển thị thông báo “Số di phải chứa 10 ký tự số và bắt đầu bằng số 0!”	PASS

*Bảng 3.2 Danh sách testcase từ đồ thị*

### **3.3.1.1. Phân lớp tương đương**

#### **– Xác định các lớp tương đương**

#### **Nguyên tắc để xác định lớp tương đương:**

**NT1 :** Điều kiện đầu vào định rõ giới hạn của một mảng thì chia vùng tương đương thành 3 tình huống:

- Xác định 1 lớp tương đương hợp lệ.
- Xác định 2 lớp tương đương không hợp lệ.

**Ví dụ:** Khi chọn ngày khởi hành thì phải chọn ngày hiện tại và không chọn ngày vượt quá tháng hiện tại. Giả sử hiện tại là ngày 10/05/2019 vậy sẽ có 3 lớp tương đương là :

- Lớp tương đương hợp lệ: Ngày khởi hành tính từ ngày 10/05/2019 đến ngày 31/05/2019
- Ngày khởi hành nhỏ hơn ngày 10/05/2019
- Ngày khởi hành lớn hơn ngày 31/05/2019

**NT2:** Điều kiện đầu vào là một giá trị xác định thì chia vùng tương đương thành 3 tình huống:

- Xác định 1 lớp tương đương hợp lệ.
- Xác định 2 lớp tương đương không hợp lệ.

**Ví dụ:** Khi nhập thông tin số điện thoại phải nhập đủ 10 số vậy sẽ có 3 lớp tương đương l2:

- Lớp tương đương hợp lệ: Nhập số di động đủ 10 số.
- Lớp tương đương không hợp lệ: Nhập số di động nhỏ hơn 10 số.
- Lớp tương đương không hợp lệ: Nhập số di động lớn hơn 10 số.

**NT3:** Điều kiện đầu vào chỉ định là một tập giá trị thì ta sẽ xác định mỗi giá trị trong tập đó là một lớp tương đương hợp lệ và chia lớp tương đương thành 2 tình huống như sau:

- Xác định 1 lớp tương đương hợp lệ



- Xác định 1 lớp tương đương không hợp lệ

**Ví dụ:** Khi chọn giờ đi của tuyến An Giang => Cần Thơ sẽ có những giờ đi như: 4:00, 6:00, 8:00, 10:00, 12:00, 14:00, 16:00, 18:00.

8 lớp tương đương hợp lệ tương ứng với 8 giờ đi: 4:00, 6:00, 8:00, 10:00, 12:00, 14:00, 16:00, 18:00.

- 1 lớp tương đương không hợp lệ: một giờ đi khác các giờ đi nêu trên ví dụ như : 5:00.

**NT4:** Nếu điều kiện đầu vào xác định là một kiểu đúng sai thì chia vùng tương đương thành 2 tình huống:

- Xác định 1 lớp tương đương hợp lệ
- Xác định 1 lớp tương đương không hợp lệ

**Ví dụ:** Khi nhập số di động ký tự số đầu tiên phải là số 0.

- 1 lớp tương đương hợp lệ: ký tự số đầu tiên phải là số 0.
- 1 lớp tương đương hợp lệ: ký tự số đầu tiên không phải là số 0

#### – *Xác định các ca kiểm thử*

**Ví dụ:** Khi chọn ngày khởi hành thì phải chọn ngày hiện tại và không chọn ngày vượt quá tháng hiện tại. Giả sử hiện tại là ngày 10/05/2019.

- Vùng hợp lệ : 10/05/2019 – 31/05/2019
- Vùng không hợp lệ: trước ngày 10/05/2019 và sau 31/05/2019

STT	Mô tả kịch bản thử nghiệm	Kết quả mong muốn
1	Chọn ngày đi từ ngày 10/05/2019 đến ngày 31/05/2019 ví dụ chọn ngày 20/05/2019	Hệ thống chấp nhận
2	Chọn ngày đi từ 20/04/2019 đến 09/05/2019 ví dụ chọn ngày 02/05/2019	Hệ thống không chấp nhận
3	Chọn ngày đi từ 01/06/2019 đến 30/06/2019 ví dụ chọn ngày 07/06/2019	Hệ thống không chấp nhận

*Bảng 3.3 Test case phân vùng tương đương kiểm tra ngày đi*

**Ví dụ:** Khi nhập thông tin số điện thoại phải nhập đủ 10 :

- Vùng hợp lệ: Nhập số di động đủ 10 số.
- Vùng không hợp lệ: Nhập số di động nhỏ hơn 10 số và nhập số di động lớn hơn 10 số.

STT	Mô tả kịch bản thử nghiệm	Kết quả mong muốn
1	Nhập số điện thoại là 10 số	Hệ thống chấp nhận
2	Nhập số điện thoại từ 0-9 số	Hệ thống không chấp nhận
3	Nhập số điện thoại từ 11- 20 số	Hệ thống không chấp nhận

*Bảng 3.4 Test case phân vùng tương đương kiểm tra số điện thoại*

### **3.3.1.2. Phân tích giá trị biên**

Phân tích giá trị biên bao gồm:

- Giá trị nhỏ nhất - 1.
- Giá trị nhỏ nhất.
- Giá trị nhỏ nhất + 1.
- Giá trị bình thường.
- Giá trị lớn nhất.
- Giá trị lớn nhất - 1.
- Giá trị lớn nhất.
- Giá trị lớn nhất + 1

**Ví dụ:** Khi chọn ngày khởi hành thì phải chọn ngày hiện tại và không chọn ngày vượt quá tháng hiện tại. Giả sử hiện tại là ngày 10/05/2019.

STT	Mô tả kịch bản thử nghiệm	Kết quả mong muốn
1	Chọn ngày đi là 10/05/2019	Hệ thống chấp nhận
2	Chọn ngày đi là 11/05/2019	Hệ thống chấp nhận
3	Chọn ngày đi là 09/05/2019	Hệ thống không chấp nhận
4	Chọn ngày đi là 31/05/2019	Hệ thống chấp nhận
5	Chọn ngày đi là 30/05/2019	Hệ thống chấp nhận
6	Chọn ngày đi là 01/06/2019	Hệ thống không chấp nhận

*Bảng 3.5 Test case phân tích giá trị biên kiểm tra ngày đi*

**Ví dụ:** Khi nhập thông tin số điện thoại phải nhập đủ 10 số

STT	Mô tả kịch bản thử nghiệm	Kết quả mong muốn
1	Nhập số điện thoại 10 số	Hệ thống chấp nhận
2	Nhập số điện thoại 9 số	Hệ thống không chấp nhận
3	Nhập số điện thoại 11 số	Hệ thống không chấp nhận

*Bảng 3.6 Test case phân tích giá trị biên kiểm tra số điện thoại*

### 3.4. Kiểm thử chức năng

TEST PLAN	
Tên project:	Quản lý ứng dụng bán vé trực tuyến của công ty vận tải xe khách Thành Bưởi
Ngày bắt đầu:	02/05/2019
Ngày kết thúc:	26/05/2019
Danh sách mô đun cần test:	Mua vé, tuyến xe
Danh sách test case	Mua vé: Testcase màn hình chọn tuyến, testcase màn hình chọn ghế, testcase màn hình thông tin khách Tuyến xe: Testcase chức năng thêm, testcase chức năng sửa, testcase chức năng xóa

*Bảng 3.7 Test plan*

### 3.4.1. Chức năng mua vé

- Kiểm tra màn hình quản lý tuyến xe

**THÊM TUYẾN XE**

**Điểm đi**  
--Chọn điểm đi--

**Điểm đến**  
--Chọn điểm đến--

**Giá vé**  
Nhập giá vé

**Khoảng cách**  
Nhập khoảng cách

**Thời gian**  
Nhập thời gian

Thêm

**Danh sách tuyến xe**

STT	Mã Tuyến	Điểm đi	Điểm đến	Tên Tuyến	Bảng giá	Khoảng cách	Thời gian	Hành động	
1	MT01	TP.Hồ Chí Minh	Cần Thơ	TP.HCM-Cần Thơ	110000 VNĐ	166		<a href="#" style="background-color: #007bff; color: white; padding: 2px 5px;">Sửa</a>	<a href="#" style="background-color: #28a745; color: white; padding: 2px 5px;">Xóa</a>
2	MT02	An Giang	TP.Hồ Chí Minh	An Giang-TP.HCM	170000 VNĐ	450		<a href="#" style="background-color: #007bff; color: white; padding: 2px 5px;">Sửa</a>	<a href="#" style="background-color: #28a745; color: white; padding: 2px 5px;">Xóa</a>
3	MT03	Bến Tre	TP.Hồ Chí Minh	Bến Tre-TP.HCM	70000 VNĐ	75		<a href="#" style="background-color: #007bff; color: white; padding: 2px 5px;">Sửa</a>	<a href="#" style="background-color: #28a745; color: white; padding: 2px 5px;">Xóa</a>
4	MT04	Trà Vinh	Cần Thơ	Trà Vinh-Cần Thơ	100000 VNĐ	75		<a href="#" style="background-color: #007bff; color: white; padding: 2px 5px;">Sửa</a>	<a href="#" style="background-color: #28a745; color: white; padding: 2px 5px;">Xóa</a>
5	MT05	Châu Đốc	TP.Hồ Chí Minh	Châu Đốc-TP.HCM	140000 VNĐ	186		<a href="#" style="background-color: #007bff; color: white; padding: 2px 5px;">Sửa</a>	<a href="#" style="background-color: #28a745; color: white; padding: 2px 5px;">Xóa</a>
6	MT06	Bạc Liêu	TP.Hồ Chí Minh	Bạc Liêu-TP.HCM	175000 VNĐ	313		<a href="#" style="background-color: #007bff; color: white; padding: 2px 5px;">Sửa</a>	<a href="#" style="background-color: #28a745; color: white; padding: 2px 5px;">Xóa</a>
7	MT07	Cà Mau	TP.Hồ Chí Minh	Cà Mau-TP.HCM	105000 VNĐ	130		<a href="#" style="background-color: #007bff; color: white; padding: 2px 5px;">Sửa</a>	<a href="#" style="background-color: #28a745; color: white; padding: 2px 5px;">Xóa</a>
8	MT08	Cần Thơ	TP.Hồ Chí Minh	Cần Thơ-TP.HCM	155000 VNĐ	310	6	<a href="#" style="background-color: #007bff; color: white; padding: 2px 5px;">Sửa</a>	<a href="#" style="background-color: #28a745; color: white; padding: 2px 5px;">Xóa</a>
9	MT09	Long Xuyên	Cần Thơ	Long Xuyên-Cần Thơ	110000 VNĐ	166	4	<a href="#" style="background-color: #007bff; color: white; padding: 2px 5px;">Sửa</a>	<a href="#" style="background-color: #28a745; color: white; padding: 2px 5px;">Xóa</a>

Hình 3.3 Màn hình quản lý tuyến xe

ID	TITLE	STEP BY STEP	EXPECTED RESULT	ACTUAL RESULT	STATUS
<b>TESTCASE CHỨC NĂNG TÌM KIẾM</b>					
1	Kiểm tra tìm kiếm khi không nhập ký tự	1.Truy cập vào màn hình quản lý tuyến xe  2.Tại textbox tìm kiếm không nhập từ khóa  3.Nhấn button Tìm	Không trả về dữ liệu ở lưới danh sách	Không trả về dữ liệu ở lưới danh sách	PASS

2	Kiểm tra tìm kiếm khi nhập chuỗi có ký tự hoa, ký tự thường	1.Truy cập vào màn hình quản lý tuyến xe 2.Tại textbox tìm kiếm nhập từ khóa là: “An Giang” 3.Nhấn button Tìm	Hiển thị dữ liệu có chứa từ khóa cần tìm	Hiển thị dữ liệu có chứa từ khóa cần tìm	PASS
3	Kiểm tra tìm kiếm khi nhập chuỗi có dấu	1.Truy cập vào màn hình quản lý tuyến xe 2.Tại textbox tìm kiếm nhập từ khóa là: “Cà Mau” 3.Nhấn button Tìm	Hiển thị dữ liệu có chứa từ khóa cần tìm	Hiển thị dữ liệu có chứa từ khóa cần tìm	PASS
4	Kiểm tra tìm kiếm khi nhập ký tự đặc biệt	1.Truy cập vào màn hình quản lý tuyến xe 2.Tại textbox tìm kiếm nhập từ khóa là: “@@@” 3.Nhấn button Tìm	Hiển thị dữ liệu nếu có	Hiển thị dữ liệu nếu có	PASS

<b>TESTCASE CHỨC NĂNG THÊM MỚI</b>					
1	Kiểm tra load dữ liệu trên combobox điểm đi	1.Truy cập vào màn hình quản lý tuyến xe  2.Kiểm tra dữ liệu của combobox điểm đi	Load đúng dữ liệu điểm đi	Load đúng dữ liệu điểm đi	PASS
2	Kiểm tra load dữ liệu trên combobox điểm đến	1.Truy cập vào màn hình quản lý tuyến xe  2.Kiểm tra dữ liệu của combobox điểm đến	Load đúng dữ liệu điểm đến	Load đúng dữ liệu điểm đến	PASS
3	Kiểm tra khi không chọn điểm đi	1.Truy cập vào màn hình quản lý tuyến xe  2. Không chọn điểm đi  3.Chọn và điền các thông tin còn lại hợp lệ  4.Nhấn button Thêm	Hiển thị thông báo “Vui lòng chọn điểm đi!”	Hiển thị thông báo “Vui lòng chọn điểm đi!”	PASS

4	Kiểm tra khi không chọn điểm đến	1.Truy cập vào màn hình quản lý tuyến xe 2. Không chọn điểm đến 3.Chọn và điền các thông tin còn lại hợp lệ 4.Nhấn button Thêm	Hiển thị thông báo “Vui lòng chọn điểm đến!”	Hiển thị thông báo “Vui lòng chọn điểm đến!”	PASS
5	Kiểm tra khi không nhập giá vé	1.Truy cập vào màn hình quản lý tuyến xe 2. Không nhập giá vé 3.Chọn và điền các thông tin còn lại hợp lệ 4.Nhấn button Thêm	Hiển thị thông báo “Vui lòng nhập giá vé!”	Hiển thị thông báo “Vui lòng nhập giá vé!”	PASS
6	Kiểm tra khi không nhập khoảng cách	1.Truy cập vào màn hình quản lý tuyến xe 2. Không nhập khoảng cách 3.Chọn và điền các thông tin còn lại hợp lệ	Hiển thị thông báo “Vui lòng nhập khoảng cách!”	Hiển thị thông báo “Vui lòng nhập khoảng cách!”	PASS

		4.Nhấn button Thêm			
7	Kiểm tra khi không nhập thời gian	1.Truy cập vào màn hình quản lý tuyến xe 2. Không nhập thời gian 3.Chọn và điền các thông tin còn lại hợp lệ 4.Nhấn button Thêm	Hiển thị thông báo “Vui lòng nhập thời gian!”	Hiển thị thông báo “Vui lòng nhập thời gian!”	PASS
8	Kiểm tra khi nhập giá vé là chữ	1.Truy cập vào màn hình quản lý tuyến xe 2. Nhập giá vé là :”Hai trăm ngàn” 3.Chọn và điền các thông tin còn lại hợp lệ 4.Nhấn button Thêm	Không cho nhập chữ	Không cho nhập chữ	PASS
9	Kiểm tra khi nhập giá vé là ký tự đặc biệt	1.Truy cập vào màn hình quản lý tuyến xe 2. Nhập giá vé là :”\$\$\$\$”	Không cho nhập ký tự đặc biệt	Không cho nhập ký tự đặc biệt	PASS



		3.Chọn và điền các thông tin còn lại hợp lệ  4.Nhấn button Thêm			
10	Kiểm tra khi nhập khoảng cách là chữ	1.Truy cập vào màn hình quản lý tuyến xe  2. Nhập khoảng cách là :”Hai trăm KM”  3.Chọn và điền các thông tin còn lại hợp lệ  4.Nhấn button Thêm	Không cho nhập chữ	Không cho nhập chữ	PASS
11	Kiểm tra khi nhập khoảng cách là ký tự đặc biệt	1.Truy cập vào màn hình quản lý tuyến xe  2. Nhập khoảng cách là :”\$\$\$\$”  3.Chọn và điền các thông tin còn lại hợp lệ  4.Nhấn button Thêm	Không cho nhập ký tự đặc biệt	Không cho nhập ký tự đặc biệt	PASS

12	Kiểm tra khi nhập khoảng cách là 1000				
13	Kiểm tra khi nhập thời gian là chữ	1.Truy cập vào màn hình quản lý tuyến xe 2. Nhập thời gian là :”Hai giờ” 3.Chọn và điền các thông tin còn lại hợp lệ 4.Nhấn button Thêm	Không cho nhập chữ	Không cho nhập chữ	PASS
14	Kiểm tra khi nhập giá vé là ký tự đặc biệt	1.Truy cập vào màn hình quản lý tuyến xe 2. Nhập thời gian là :”\$\$\$\$” 3.Chọn và điền các thông tin còn lại hợp lệ 4.Nhấn button Thêm	Không cho nhập ký tự đặc biệt	Không cho nhập ký tự đặc biệt	PASS

15	Kiểm tra nhấn button thêm	1.Truy cập vào màn hình quản lý tuyến xe 2. Nhấn button Thêm	Hiển thị thông báo “Vui lòng nhập thông tin tuyến cần thêm!”	Hiển thị thông báo “Vui lòng nhập thông tin tuyến cần thêm!”	PASS
16	Kiểm tra khi chọn và nhập đầy đủ dữ liệu hợp lệ	1.Truy cập vào màn hình quản lý tuyến xe 2. Chọn điểm đi : “TP.Hồ Chí Minh” 3.Chọn điểm đến: “Sóc trăng” 4. Nhập giá vé:”135000” 5. Nhập thời gian:”5” 6. Nhập khoảng cách:”220” 7..Nhấn button Thêm	Hiển thị thông báo:”Thêm thành công và cập nhật dữ liệu xuống DB”	Hiển thị thông báo:”Thêm thành công và cập nhật dữ liệu xuống DB”	PASS
17	Kiểm tra lưới dữ liệu có được cập nhật sau khi thêm	1.Truy cập vào màn hình quản lý tuyến xe 2. Chọn điểm đi : “TP.Hồ Chí Minh” 3.Chọn điểm đến:	Lưới dữ liệu cập nhật lại có thêm dòng dữ liệu chứa thông tin tuyến vừa thêm	Lưới dữ liệu cập nhật lại có thêm dòng dữ liệu chứa thông tin tuyến	PASS

		“Sóc trắng” 4. Nhập giá vé:”135000” 5. Nhập thời gian:”5” 6. Nhập khoảng cách:”220” 7..Nhấn button Thêm		vừa thêm	
18	Kiểm tra cột “STT” trong lưới dữ liệu có đúng thứ tự	1.Truy cập vào màn hình quản lý tuyến xe 2. Kiểm tra STT dưới lưới dữ liệu	STT trong lưới dữ liệu sắp xếp đúng thứ tự	STT trong lưới dữ liệu sắp xếp đúng thứ tự	PASS
<b>TESTCASE CHỨC NĂNG SỬA</b>					
1	Kiểm tra khi click button sửa	1.Truy cập vào màn hình quản lý tuyến xe 2. Click button sửa dòng dữ liệu : điểm đi : “TP.Hồ Chí Minh”, điểm đến: “Sóc Trăng”, giá vé:”135000”, thời gian:”5”, khoảng cách:”220”	Hiển thị màn hình sửa với các dữ liệu được load lên các control tương ứng	Hiển thị màn hình sửa với các dữ liệu được load lên các control tương ứng	PASS

2	Kiểm tra khi thông tin sửa không thay đổi	<p>1.Truy cập vào màn hình quản lý tuyến xe</p> <p>2. Click button sửa dòng dữ liệu : điểm đi : “TP.Hồ Chí Minh”, điểm đến: “Sóc Trăng”, giá vé:”135000”, thời gian:”5”, khoảng cách:”220”</p> <p>3. Giữ nguyên dữ liệu ban đầu</p> <p>4. Click button Lưu lại</p>	Lưu dữ liệu xuống DB	Lưu dữ liệu xuống DB	PASS
3	Kiểm tra sửa khi điểm đi trống	<p>1.Truy cập vào màn hình quản lý tuyến xe</p> <p>2. Click button sửa dòng dữ liệu : điểm đi : “”, điểm đến: “Sóc Trăng”, giá vé:”135000”, thời gian:”5”, khoảng cách:”220”</p> <p>3. Click button Lưu lại</p>	Hiển thị thông báo “Vui lòng chọn điểm đi cần sửa!”	Hiển thị thông báo “Vui lòng chọn điểm đi cần sửa!”	PASS

4	Kiểm tra sửa khi điểm đến trống	<p>1.Truy cập vào màn hình quản lý tuyến xe</p> <p>2. Click button sửa dòng dữ liệu : điểm đi : “TP.Hồ Chí Minh”, điểm đến: “”, giá vé:”135000”, thời gian:”5”, khoảng cách:”220”</p> <p>3. Click button Lưu lại</p>	Hiển thị thông báo “Vui lòng chọn điểm đến cần sửa!”	Hiển thị thông báo “Vui lòng chọn điểm đến cần sửa!”	PASS
5	Kiểm tra sửa khi nhập giá vé là chữ	<p>1.Truy cập vào màn hình quản lý tuyến xe</p> <p>2. Click button sửa dòng dữ liệu : điểm đi : “TP.Hồ Chí Minh”, điểm đến: “Sóc Trăng”, giá vé:”ba trăm ngàn”, thời gian:”5”, khoảng cách:”220”</p> <p>3. Click button Lưu lại</p>	Không cho phép nhập chữ	Không cho phép nhập chữ	PASS
6	Kiểm tra sửa khi nhập giá vé là ký tự đặc biệt	<p>1.Truy cập vào màn hình quản lý tuyến xe</p>	Không cho phép nhập ký tự đặc biệt	Không cho phép nhập ký tự đặc	PASS

		<p>2. Click button sửa dòng dữ liệu : điểm đi : “TP.Hồ Chí Minh”, điểm đến: “Sóc Trăng”, giá vé:”#\$\$”, thời gian:”5”, khoảng cách:”220”</p> <p>3. Click button Lưu lại</p>		biệt	
7	Kiểm tra sửa khi khoảng cách là chữ	<p>1.Truy cập vào màn hình quản lý tuyến xe</p> <p>2. Click button sửa dòng dữ liệu : điểm đi : “TP.Hồ Chí Minh”, điểm đến: “Sóc Trăng”, giá vé:”220”, thời gian:”5”, khoảng cách:”năm mươi km”</p> <p>3. Giữ nguyên dữ liệu ban đầu</p> <p>4. Click button Lưu lại</p>	Không cho phép nhập chữ	Không cho phép nhập chữ	PASS

8	Kiểm tra sửa khi khoảng cách là ký tự đặc biệt	<p>1.Truy cập vào màn hình quản lý tuyến xe</p> <p>2. Click button sửa dòng dữ liệu : điểm đi : “TP.Hồ Chí Minh”, điểm đến: “Sóc Trăng”, giá vé:”220”, thời gian:”5”, khoảng cách:”#@#”</p> <p>3. Giữ nguyên dữ liệu ban đầu</p> <p>4. Click button Lưu lại</p>	Không cho phép nhập ký tự đặc biệt	Không cho phép nhập ký tự đặc biệt	PASS
9	Kiểm tra sửa khi thời gian là chữ	<p>1.Truy cập vào màn hình quản lý tuyến xe</p> <p>2. Click button sửa dòng dữ liệu : điểm đi : “Tp.Hồ Chí Minh”, điểm đến: “Sóc Trăng”, giá vé:”135000”, thời gian:”sáu giờ”, khoảng cách:”220”</p> <p>3. Click button</p>	Không cho phép nhập chữ	Không cho phép nhập chữ	PASS



		Lưu lại			
10	Kiểm tra sửa khi thời gian là ký tự đặc biệt	<p>1.Truy cập vào màn hình quản lý tuyến xe</p> <p>2. Click button sửa dòng dữ liệu : điể đi : “Tp.Hồ Chí Minh”, điể đến: “Sóc trăng”, giá vé:”135000”, thời gian:”#@\$\$”, khoảng cách:”220”</p> <p>3. Click button Lưu lại</p>	Không cho phép nhập ký tự đặc biệt	Không cho phép nhập ký tự đặc biệt	PASS
11	Kiểm tra button Lưu lại	<p>1.Truy cập vào màn hình quản lý tuyến xe</p> <p>2. Click button sửa dòng dữ liệu : điể đi : “TP.Hồ Chí Minh”, điể đến: “Sóc Trăng”, giá vé:”200000”, thời gian:”7”, khoảng cách:”220”</p> <p>3. Click button Lưu lại</p>	Hiện thị thông báo”Sửa thành công và cập nhật dữ liệu xuống DB”	Hiện thị thông báo”Sửa thành công và cập nhật dữ liệu xuống DB”	PASS

12	Kiểm tra lưới dữ liệu có hiển thị đúng dữ liệu sau khi sửa	<p>1.Truy cập vào màn hình quản lý tuyến xe</p> <p>2. Click button sửa dòng dữ liệu : điểm đi : “TP.Hồ Chí Minh”, điểm đến: “Sóc Trăng”, giá vé:”200000”, thời gian:”7”, khoảng cách:”220”</p> <p>3. Click button Lưu lại</p> <p>4.Kiểm tra lưới dữ liệu</p>	Hiển thị thông tin đúng với thông tin đã sửa	Hiển thị thông tin đúng với thông tin đã sửa	PASS
----	--	--	--	--	------

#### ***TESTCASE CHỨC NĂNG XÓA***

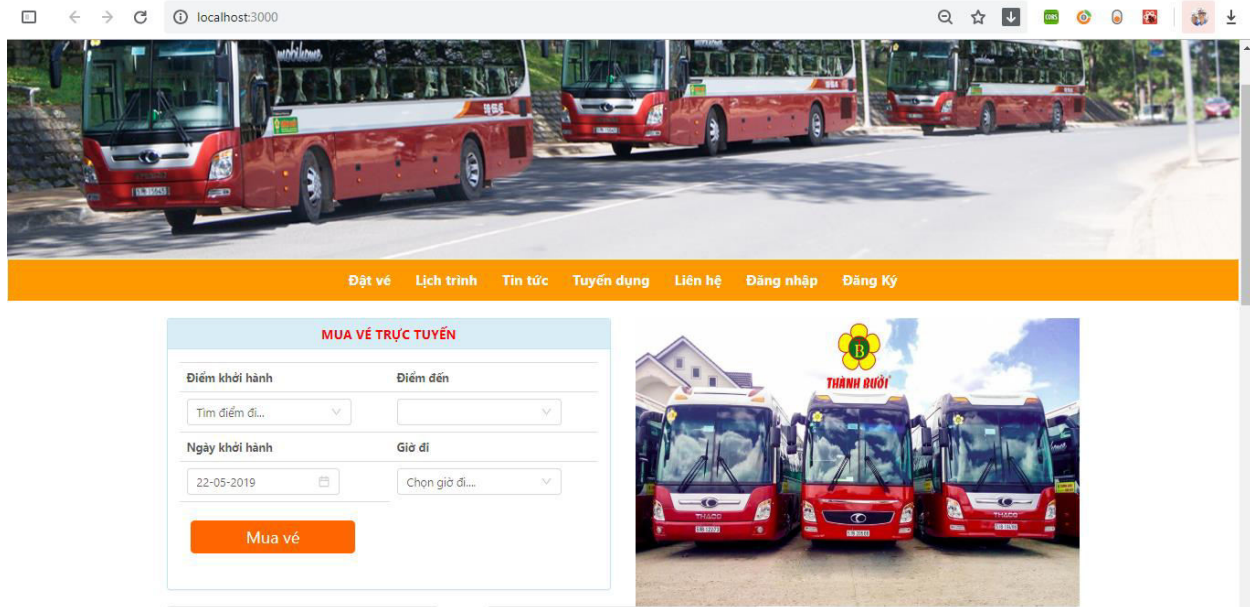
1	Kiểm tra xóa tuyến xe chưa có ràng buộc	<p>1.Truy cập vào màn hình quản lý tuyến xe</p> <p>2.Chọn xóa tuyến xe vừa thêm</p> <p>3. Hiển thị thông báo xác nhận “Bạn có chắc chắn muốn xóa?”</p> <p>4.Chọn “Có”</p>	Dữ liệu bị xóa và cập nhật lại dưới DB	Dữ liệu bị xóa và cập nhật lại dưới DB	PASS
---	---	---	--	--	------

2	Kiểm tra thông báo khi xóa	.Truy cập vào màn hình quản lý tuyến xe 2.Chọn xóa tuyến xe vừa thêm	Hiển thị thông báo “Bạn có chắc chắn muốn xóa?”	Hiển thị thông báo “Bạn có chắc chắn muốn xóa?”	PASS
3	Kiểm tra xóa tuyến xe đã có ràng buộc	1.Truy cập vào màn hình quản lý tuyến xe 2.Chọn xóa tuyến xe đã có chi tiết chuyến	Hiển thị thông báo xác nhận “Không thể xóa tuyến xe này!”	Hiển thị thông báo xác nhận “Không thể xóa tuyến xe này!”	PASS
4	Kiểm tra khi hủy bỏ thao tác xóa	1.Truy cập vào màn hình quản lý tuyến xe 2.Chọn xóa tuyến xe vừa thêm 3. Hiển thị thông báo xác nhận “Bạn có chắc chắn muốn xóa?” 4.Chọn “Không”	Quay lại màn hình quản lý tuyến xe	Quay lại màn hình quản lý tuyến xe	PASS

5	Kiểm tra lưới dữ liệu có hiển thị đúng dữ liệu sau khi xóa	1.Truy cập vào màn hình quản lý tuyến xe 2.Chọn xóa tuyến xe vừa thêm 3. Hiển thị thông báo xác nhận “Bạn có chắc chắn muốn xóa?” 4.Chọn “Có” 5.Kiểm tra trên lưới dữ liệu	Lưới dữ liệu không hiển thị tuyến xe vừa xóa	Lưới dữ liệu không hiển thị tuyến xe vừa xóa	PASS
6	Kiểm tra cột “STT” có sắp xếp đúng sau khi xóa	1.Truy cập vào màn hình quản lý tuyến xe 2.Chọn xóa tuyến xe vừa thêm có STT là”5” 3. Hiển thị thông báo xác nhận “Bạn có chắc chắn muốn xóa?” 4.Chọn “Có” 5. Kiểm tra STT	Số thứ tự được sắp xếp lại	Số thứ tự được sắp xếp lại	PASS

*Bảng 3.8 Test case quản lý tuyến xe*

- Kiểm tra màn hình mua vé
- Kiểm tra màn hình Chọn tuyến



Hình 3.4 Màn hình chọn tuyến

ID	TITLE	STEP BY STEP	EXPECTED RESULT	ACTUAL RESULT	STATUS
<b>TESTCASE MÀN HÌNH CHỌN TUYẾN</b>					
1	Kiểm tra khi không chọn điểm đi	1.Truy cập vào website mua vé 2. Không chọn điểm đi 3.Chọn đầy đủ các thông tin còn lại 4. Click button mua vé	Hiển thị thông báo “Vui lòng chọn điểm đi!”	Hiển thị thông báo “Vui lòng chọn điểm đi!”	PASS

2	Kiểm tra khi không chọn điểm đến	1.Truy cập vào website mua vé 2. Không chọn điểm đến	Điểm đến load theo điểm đi. Nếu không chọn mặc định lấy điểm đến đầu tiên trong combobox điểm đến	Điểm đến load theo điểm đi. Nếu không chọn mặc định lấy điểm đến đầu tiên trong combobox điểm đến	PASS
3	Kiểm tra khi không chọn ngày đi	1.Truy cập vào website mua vé 2. Không chọn ngày đi 3.Chọn đầy đủ các thông tin còn lại	Lấy mặc định ngày hiện tại	Lấy mặc định ngày hiện tại	PASS
4	Kiểm tra khi không chọn giờ đi	1.Truy cập vào website mua vé 2. Không chọn giờ đi 3.Chọn đầy đủ các thông tin còn lại 4. Click button mua vé	Hiển thị thông báo “Vui lòng chọn giờ đi!”	Hiển thị thông báo “Vui lòng chọn giờ đi!”	PASS
5	Kiểm tra khi chọn ngày đi lớn hơn ngày cuối cùng trong tháng hiện tại	1.Truy cập vào website mua vé Ngày hiện tại	Không cho chọn	Không cho chọn	PASS

		22/05/2019 2. Chọn điểm đi:”Tp.Hồ Chí Minh”, Chọn điểm đến: “An Giang”, Chọn ngày đi : “01/06/2019”			
6	Kiểm tra khi chọn ngày đi nhỏ hơn ngày hiện tại	1.Truy cập vào website mua vé Ngày hiện tại 22/05/2019 2. Chọn điểm đi:”Tp.Hồ Chí Minh”, Chọn điểm đến: “An Giang”, Chọn ngày đi : “01/05/2019”	Không cho chọn	Không cho chọn	PASS
7	Kiểm tra load dữ liệu trên combobox điểm đi	1.Truy cập vào website mua vé 2.Kiểm tra dữ liệu của combobox điểm đi	Hiển thị đúng dữ liệu danh sách điểm đi	Hiển thị đúng dữ liệu danh sách điểm đi	PASS
8	Kiểm tra load dữ liệu trên combobox điểm đến	1.Truy cập vào website mua vé 2.Kiểm tra dữ liệu	Hiển thị đúng dữ liệu danh sách điểm đến	Hiển thị đúng dữ liệu danh	PASS

		của combobox điểm đến		sách điểm đến	
9	Kiểm tra load dữ liệu trên combobox giờ đi	1.Truy cập vào website mua vé 2.Kiểm tra dữ liệu của combobox giờ đi	Hiển thị đúng dữ liệu danh sách giờ đi	Hiển thị đúng dữ liệu danh sách giờ đi	PASS
10	Kiểm tra khi thực hiện click button mua vé mà không điền dữ liệu	1.Truy cập vào website mua vé 2.Click button mua vé	Hiển thị thông báo “Vui lòng chọn thông tin tuyến!”	Chưa hiển thị thông báo	FAIL
11	Kiểm tra khi thực hiện click button mua vé khi đã có đầy đủ dữ liệu hợp lệ	1.Truy cập vào website mua vé Ngày hiện tại 22/05/2019 2. Chọn điểm đi:”Tp.Hồ Chí Minh”, Chọn điểm đến: “An Giang”, Chọn ngày đi : “25/05/2019” 3.Click button Mua vé	Chuyển hướng đến màn hình chọn ghế	Chuyển hướng đến màn hình chọn ghế	PASS

*Bảng 3.9 Test case màn hình chọn tuyến*



– Kiểm tra màn hình Chọn ghế

**Thông tin chọn tuyến**

Chọn tuyến xe  
TP.HCM-Cần Thơ

Chọn giờ khởi hành  
06:00

Chọn điểm lên xe  
Điểm lên xe 1

**Quay lại** **Tiếp tục**

**THÔNG TIN CHUYẾN**

Xuất phát từ: Châu Đốc  
ngày đi: 16/04/2019  
6:00

4 giờ

Đến lúc: Cần Thơ  
16/04/2019  
10:00

Số ghế: 17 13 7

Tổng tiền: (330000)<sup>d</sup>

**SƠ ĐỒ GHẾ**

Tải xế

Cửa ra vào

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16
17	18	19	20
21	22	23	24
25	26	27	28
29			

Đang chọn Đã đặt Còn trống

Hình 3.5 Màn hình chọn ghế

ID	TITLE	STEP BY STEP	EXPECTED RESULT	ACTUAL RESULT	STATUS
<b>TESTCASE MÀN HÌNH CHỌN GHẾ</b>					
1	Kiểm tra thông tin tuyến xe	1.Truy cập vào website mua vé 2. Kiểm tra thông tin tuyến xe	Hiển thị đúng thông tin tuyến xe đã chọn	Hiển thị đúng thông tin tuyến xe đã chọn	PASS
2	Kiểm tra thông tin giờ đi	1.Truy cập vào website mua vé 2. Kiểm tra thông tin giờ đi	Hiển thị đúng thông tin giờ đi đã chọn	Hiển thị đúng thông tin giờ đi đã chọn	PASS

3	Kiểm tra thông tin chuyến	1.Truy cập vào website mua vé 2. Kiểm tra thông tin chuyến	Hiển thị đúng thông tin ngày đi, giờ đi, điểm đi, điểm đến đã chọn	Hiển thị đúng thông tin ngày đi, giờ đi, điểm đi, điểm đến đã chọn	PASS
4	Kiểm tra khi không chọn ghế	1.Truy cập vào website mua vé 2. Kiểm tra khi không chọn ghế	Hiển thị thông báo “Vui lòng chọn ghế!”	Chưa thông báo	FAIL
5	Kiểm tra khi chọn hơn 10 ghế	1.Truy cập vào website mua vé 2. Kiểm tra khi chọn 11 ghế	Không cho chọn quá 10 ghế	Không cho chọn quá 10 ghế	PASS
6	Kiểm tra label số ghế đã chọn	1.Truy cập vào website mua vé 2. Kiểm tra label số ghế đã chọn	Hiển thị đúng số ghế chọn	Hiển thị đúng số ghế chọn	PASS
7	Kiểm tra tổng tiền	1.Truy cập vào website mua vé 2. Kiểm tra tổng tiền	Tổng tiền được tính tương ứng với số ghế chọn và tính theo giá của tuyến	Tổng tiền được tính tương ứng với số ghế chọn và tính theo giá của tuyến	PASS
8	Kiểm tra số thứ tự của ghế	1.Truy cập vào website mua vé	Số thứ tự được sắp xếp đúng	Số thứ tự được sắp	PASS

		2. Kiểm tra số thứ tự của ghế		xếp đúng	
9	Kiểm tra button quay lại	1.Truy cập vào website mua vé 2. Click button quay lại	Quay về màn hình chọn tuyến	Chưa quay lại màn hình chọn tuyến	FAIL
10	Kiểm tra button tiếp tục	1.Truy cập vào website mua vé 2. Chọn ghế 15 3.Click Tiếp tục	Chuyển hướng đến màn hình Thông tin khách hàng	Chuyển hướng đến màn hình Thông tin khách hàng	PASS
11	Kiểm tra chọn ghế đã đặt	1.Truy cập vào website mua vé 2. Chọn ghế đã đặt	Không cho chọn	Không cho chọn	PASS

*Bảng 3.10 Test case màn hình chọn ghế*

– Kiểm tra màn hình thông tin khách hàng

The screenshot displays a web interface for booking a train ticket. At the top, there is a navigation bar with links: Đặt vé, Lịch trình, Tin tức, Tuyển dụng, Liên hệ, Đăng nhập, and Đăng Ký. The main content area is divided into two columns. The left column, titled 'Thông tin chọn tuyến', contains three dropdown menus: 'Chọn tuyến xe' (currently showing 'TP.HCM-Cần Thơ'), 'Chọn giờ khởi hành' (currently showing '06:00'), and 'Chọn điểm lên xe' (currently showing 'Điểm lên xe 1'). Below these are two buttons: 'Quay lại' (orange) and 'Tiếp tục' (green). The right column, titled 'THÔNG TIN HÀNH KHÁCH', contains several input fields: 'Email (\*)' (with value 'datnooop01@gmail.com'), 'Ngày sinh' (with value '22-05-2019'), 'Họ tên (\*)' (with value 'Thành Đạt'), 'Di động (\*)' (with value '0903085543'), 'Di động 2' (with value 'Số di động 2'), 'Tỉnh/TP (\*)' (with value 'Chon tinh/tp'), and 'Quận/Huyện (\*)' (with value 'Chon quận/huyen'). There is also a checkbox for 'Chấp nhận điều khoản đặt vé' and a green 'Thanh toán' button. Below the form, there is a section for 'THÔNG TIN CHUYẾN' with fields for 'Xuất phát từ' (Châu Đốc) and 'ngày đi' (16/04/2019). At the bottom, there is a section for 'ĐIỀU KHOẢN VÀ LƯU Ý'.

*Hình 3.6 Màn hình thông tin khách*

ID	TITLE	STEP BY STEP	EXPECTED RESULT	ACTUAL RESULT	STATUS
<b>TESTCASE MÀN HÌNH THÔNG TIN KHÁCH HÀNG</b>					
1	Kiểm tra khi không nhập email	1.Truy cập vào website mua vé 2. Không nhập Email 3. Nhập đầy đủ các thông tin còn lại hợp lệ 4.Click button Thanh toán	Hiển thị thông báo”Vui lòng nhập email!”	Hiển thị thông báo”Vui lòng nhập email!”	PASS
2	Kiểm tra định dạng email	1.Truy cập vào website mua vé 2. Nhập email : “aaagmail.com” 3. Nhập đầy đủ các thông tin còn lại hợp lệ 4.Click button Thanh toán	Hiển thị thông báo: “Email không đúng định dạng!”	Hiển thị thông báo: “Email không đúng định dạng!”	PASS
3	Kiểm tra không nhập họ tên	1.Truy cập vào website mua vé 2.Không nhập họ tên 3. Nhập đầy đủ các thông tin còn	Hiển thị thông báo “Vui lòng nhập họ tên!”	Hiển thị thông báo “Vui lòng nhập họ tên!”	PASS



6	Kiểm tra khi không chọn Tỉnh/TP	1.Truy cập vào website mua vé 2. Không chọn Tỉnh/TP 3. Nhập đầy đủ các thông tin còn lại hợp lệ 4.Click button Thanh toán	Hiển thị thông báo “Vui lòng chọn tỉnh/tp!”	Hiển thị thông báo “Vui lòng chọn tỉnh/tp!”	PASS
7	Kiểm tra khi không chọn quận/ huyện	1.Truy cập vào website mua vé 2. Không chọn Quận/Huyện 3. Nhập đầy đủ các thông tin còn lại hợp lệ 4.Click button Thanh toán	Hiển thị thông báo “Vui lòng chọn quận/huyện!”	Hiển thị thông báo “Vui lòng chọn quận/huyện !”	PASS
8	Kiểm tra khi không tick chọn checkbox chấp nhận điều khoản	1.Truy cập vào website mua vé 2. Không tick chọn checkbox chấp nhận 3. Nhập đầy đủ các thông tin còn lại hợp lệ	Button Thanh toán không sáng lên và không hoạt động	Chưa kiểm tra checkbox	FAIL

9	Kiểm tra button thanh toán	1.Truy cập vào website mua vé 2. Nhập đầy đủ các thông tin hợp lệ 3.Click button Thanh toán	Hệ thống gửi email về đại chỉ email khách hàng	Hệ thống gửi email về đại chỉ email khách hàng	PASS
10	Kiểm tra khi nhập số di động 9 số	1.Truy cập vào website mua vé 2. Nhập email: <a href="mailto:nguyenhoang@gmail.com">nguyenhoang@gmail.com</a> , họ tên: “Nguyễn Hoàng”, di động: “012345678”, tỉnh/tp: “Tp.Hồ Chí Minh”, quận/huyện:”Tân Phú”, tick checkbox chấp nhận 3.Click Thanh toán	Chỉ cho nhập 10 số	Chỉ cho nhập 10 số	PASS
11	Kiểm tra khi nhập số di động 11 số	1.Truy cập vào website mua vé 2. Nhập email: <a href="mailto:nguyenhoang@gmail.com">nguyenhoang@gmail.com</a> , họ tên: “Nguyễn Hoàng”, di động:	Chỉ cho nhập 10 số	Chỉ cho nhập 10 số	PASS

		<p>“01234567891”, tỉnh/tp: “Tp.Hồ Chí Minh”, quận/huyện:”Tân Phú”, tick checkbox chấp nhận</p> <p>3.Click Thanh toán</p>			
12	Kiểm tra khi nhập số di động không bắt đầu bằng số 0	<p>1.Truy cập vào website mua vé</p> <p>2. Nhập email:<a href="mailto:nguyenhoang@gmail.com">nguyenhoang@gmail.com</a>, họ tên: “Nguyễn Hoàng”, di động: “123456789”, tỉnh/tp: “Tp.Hồ Chí Minh”, quận/huyện:”Tân Phú”, tick checkbox chấp nhận</p> <p>3.Click Thanh toán</p>	Hiển thị thông báo “Số điện thoại phải chứa 10 số và bắt đầu bằng số 0”	Chưa kiểm tra đầu số là 0	FAIL
13	Kiểm tra khi nhập số di động là chữ	<p>1.Truy cập vào website mua vé</p> <p>2. Nhập email:<a href="mailto:nguyenhoang@gmail.com">nguyenhoang@gmail.com</a>, họ</p>	Không cho nhập chữ	Không cho nhập chữ	PASS



		<p>tên: “Nguyễn Hoàng”, di động: “aadffadf”,  tỉnh/tp: “Tp.Hồ Chí Minh”,  quận/huyện:”Tân Phú”, tick checkbox chấp nhận</p> <p>3.Click Thanh toán</p>			
14	Kiểm tra khi nhập số di động là ký tự đặc biệt	<p>1.Truy cập vào website mua vé</p> <p>2. Nhập email:<a href="mailto:nguyenhoang@gmail.com">nguyenhoang@gmail.com</a>, họ tên: “Nguyễn Hoàng”, di động: “aadffadf”,  tỉnh/tp: “Tp.Hồ Chí Minh”,  quận/huyện:”Tân Phú”, tick checkbox chấp nhận</p> <p>3.Click Thanh toán</p>	Không cho nhập ký tự đặc biệt	Không cho nhập ký tự đặc biệt	PASS

*Bảng 3.11 Test case màn hình thông tin khách*

### 3.5. Kiểm thử cơ sở dữ liệu

- Kiểm tra màn hình quản lý tuyến xe

ID	TITLE	STEP BY STEP	EXPECTED RESULT	ACTUAL RESULT	STATUS
1	Kiểm tra dữ liệu của combobox điểm đi	1.Vào cơ sở dữ liệu 2. Viết câu lệnh kiểm tra danh sách điểm đi từ bảng điểm đi: “select tendiemdi from DIEMDI where isdelete= 0 ”	Combobox điểm đi load đúng dữ liệu dưới bảng DIEMDI trong cơ sở dữ liệu	Combobox điểm đi load đúng dữ liệu dưới bảng DIEMDI trong cơ sở dữ liệu	PASS
2	Kiểm tra dữ liệu của combobox điểm đến	1.Vào cơ sở dữ liệu 2. Viết câu lệnh kiểm tra danh sách điểm đến từ bảng điểm đến: “select tendiemden from DIEMDEN where isdelete= 0 ”	Combobox điểm đến load đúng dữ liệu dưới bảng DIEMDEN trong cơ sở dữ liệu	Combobox điểm đến load đúng dữ liệu dưới bảng DIEMDEN trong cơ sở dữ liệu	PASS
3	Kiểm tra khi nhập bảng giá	1.Vào cơ sở dữ liệu 2. Kiểm tra dữ liệu được thêm từ giao diện có trùng	Dữ liệu được thêm từ giao diện khớp với kiểu dữ liệu dưới DB	Dữ liệu được thêm từ giao diện khớp với kiểu dữ liệu	PASS

		với kiểu dữ liệu dưới DB		liệu dưới DB	
4	Kiểm tra nhập khoảng cách	1.Vào cơ sở dữ liệu 2. Kiểm tra dữ liệu được thêm từ giao diện có trùng với kiểu dữ liệu dưới DB	Dữ liệu được thêm từ giao diện khớp với kiểu dữ liệu dưới DB	Dữ liệu được thêm từ giao diện khớp với kiểu dữ liệu dưới DB	PASS
5	Kiểm tra nhập thời gian	1.Vào cơ sở dữ liệu 2. Kiểm tra dữ liệu được thêm từ giao diện có trùng với kiểu dữ liệu dưới DB	1.Vào cơ sở dữ liệu 2. Kiểm tra dữ liệu được thêm từ giao diện có trùng với kiểu dữ liệu dưới DB	Dữ liệu được thêm từ giao diện khớp với kiểu dữ liệu dưới DB	PASS
6	Kiểm tra sau khi thêm tuyến xe	1.Vào cơ sở dữ liệu 2. Viết câu lệnh tìm kiếm tuyến xe vừa thêm: “select * from TUYENXE where tentx= =N’TP.HCM-Cần Thơ”	Cơ sở dữ liệu hiển thị dòng dữ liệu mới vừa thêm	Cơ sở dữ liệu hiển thị dòng dữ liệu mới vừa thêm	PASS

7	Kiểm tra mã tuyến có tăng tự động sau khi thêm	1.Vào cơ sở dữ liệu 2.Viết câu lệnh lấy danh sách tuyến xe và kiểm tra mã có tăng tự động	Mã tuyến tăng tự động sau khi thêm mới	Mã tuyến tăng tự động sau khi thêm mới	PASS
8	Kiểm tra các trường sau khi thêm có đúng với các trường dưới cơ sở dữ liệu	1.Vào cơ sở dữ liệu 2.Viết câu lệnh tìm kiếm tuyến xe vừa thêm	Dữ liệu thêm từ giao diện khớp với dữ liệu dưới DB	Dữ liệu thêm từ giao diện khớp với dữ liệu dưới DB	PASS
9	Kiểm tra lưới danh sách tuyến chỉ hiển thị các dòng isdelete = =0	1.Vào cơ sở dữ liệu 2.Viết câu lệnh lấy danh sách các tuyến có isdelete= =0 3. Kiểm tra trên giao diện	Lưới danh sách tuyến chỉ hiển thị các dòng isdelete = =0	Lưới danh sách tuyến chỉ hiển thị các dòng isdelete = =0	PASS
10	Kiểm tra cơ sở dữ liệu sau khi sửa 1 tuyến	1.Vào cơ sở dữ liệu 2.Viết câu lệnh lấy danh sách các tuyến 3. Kiểm tra dữ liệu thêm trên giao diện có thêm	Dữ liệu thêm trên giao diện được cập nhật vào DB	Dữ liệu thêm trên giao diện được cập nhật vào DB	PASS

		vào DB			
11	Kiểm tra cơ sở dữ liệu sau khi xóa 1 tuyển	1.Vào cơ sở dữ liệu 2.Viết câu lệnh lấy danh sách các tuyển 3. Kiểm tra dữ liệu xóa trên giao diện có cập nhật isdeleted==1 vào DB	Cột isdeleted= =1	Cột isdeleted= =1	PASS
12	Kiểm tra sau khi thêm có trùng mã tuyển	1.Vào cơ sở dữ liệu 2. Viết câu lệnh lấy danh sách các tuyển 3.Kiểm tra có trùng mã tuyển	Mã tuyển tăng tự động và không trùng	Mã tuyển tăng tự động và không trùng	PASS
13	Kiểm tra các ràng buộc khóa ngoại	1.Vào cơ sở dữ liệu 2.Kiểm tra các ràng buộc khóa ngoại	Các bảng được tạo ràng buộc cần thiết	Các bảng được tạo ràng buộc cần thiết	PASS

14	Kiểm tra các trường bắt buộc	1.Vào cơ sở dữ liệu 2.Kiểm tra các ràng buộc khóa ngoại	Các trường bắt buộc là không cho null	Các trường bắt buộc là không cho null	PASS
----	------------------------------	--	---------------------------------------	---------------------------------------	------

*Bảng 3.12 Test case kiểm tra cơ sở dữ liệu tuyến xe*

– Kiểm tra màn hình mua vé

ID	TITLE	STEP BY STEP	EXPECTED RESULT	ACTUAL RESULT	STATUS
1	Kiểm tra các trường dữ liệu của bảng TUYENXE	1.Vào cơ sở dữ liệu 2. Kiểm tra các trường trong bảng 3.Kiểm tra MADIEMDI,MA DIEMDEN có tồn tại	Trường MATX không cho null Tất cả dữ liệu phải đúng định dạng Lấy danh sách cột IsDelete=0	Trường MATX không cho null Tất cả dữ liệu phải đúng định dạng Lấy danh sách cột IsDelete=0	PASS
2	Kiểm tra các trường dữ liệu của bảng CHUYENXE	1.Kiểm tra các trường dữ liệu của bảng 2. Kiểm tra ở trường isdeleted	Trường MACX không cho null Tất cả dữ liệu truyền vào phải đúng kiểu dữ liệu Không được trùng giờ đi Lấy danh sách	Trường MACX không cho null Tất cả dữ liệu truyền vào phải đúng kiểu dữ liệu Không được trùng giờ đi Lấy danh sách	PASS

			cột IsDelete=0	cột IsDelete=0	
3	Kiểm tra các trường dữ liệu của bảng CHITIETCHUYEN	1.Kiểm tra các trường dữ liệu của bảng 2. Kiểm tra ở trường isdeleted	Trường MACHITIETC HUYEN không cho null Tất cả dữ liệu truyền vào phải đúng kiểu dữ liệu Khi thêm tự động mã phải tự động tăng theo số dòng và không bị trùng Lấy danh sách cột IsDelete=0	Trường MACHITIET CHUYEN không cho null Tất cả dữ liệu truyền vào phải đúng kiểu dữ liệu Khi thêm tự động mã phải tự động tăng theo số dòng và không bị trùng Lấy danh sách cột IsDelete=0	PASS
4	Kiểm tra các trường dữ liệu của bảng DIEMDI	1.Kiểm tra các trường dữ liệu của bảng 2. Kiểm tra ở trường isdeleted	Trường MADIEMDI không cho null Tất cả dữ liệu truyền vào phải đúng kiểu dữ liệu Lấy danh sách cột IsDelete=0	Trường MADIEMDI không cho null Tất cả dữ liệu truyền vào phải đúng kiểu dữ liệu Lấy danh sách cột IsDelete=0	PASS

5	Kiểm tra các trường dữ liệu của bảng DIEMDEN	<p>1.Kiểm tra các trường dữ liệu của bảng</p> <p>2. Kiểm tra ở trường isdeleted</p>	<p>Trường MADIEMDEN không cho null</p> <p>Tất cả dữ liệu truyền vào phải đúng kiểu dữ liệu</p> <p>Lấy danh sách cột IsDelete=0</p>	<p>Trường MADIEMDEN không cho null</p> <p>Tất cả dữ liệu truyền vào phải đúng kiểu dữ liệu</p> <p>Lấy danh sách cột IsDelete=0</p>	PASS
6	Kiểm tra các trường dữ liệu của bảng VE-GHE	<p>1.Kiểm tra cột isdeleted</p> <p>2.Kiểm tra cột trạng thái</p> <p>3. Kiểm tra các trường dữ liệu còn lại có đúng với dữ liệu trên giao diện</p>	<p>Trường MAVEGHE không cho null</p> <p>Tất cả dữ liệu truyền vào phải đúng kiểu dữ liệu</p> <p>Khi thêm tự động mã phải tự động tăng theo</p> <p>MACTCHUYEN truyền vào</p> <p>Lấy danh sách cột IsDelete=0</p>	<p>Trường MAVEGHE không cho null</p> <p>Tất cả dữ liệu truyền vào phải đúng kiểu dữ liệu</p> <p>Khi thêm tự động mã phải tự động tăng theo</p> <p>MACTCHUYEN truyền vào</p> <p>Lấy danh sách cột IsDelete=0</p>	PASS
7	Kiểm tra các trường dữ liệu của bảng VEXE	1.Kiểm tra MAKH và MAVEGHE	<p>Trường MAVE không cho null</p> <p>Tất cả dữ liệu</p>	<p>Trường MAVE không cho null</p>	PASS



		<p>2.Kiểm tra MAVEXE</p> <p>3. Kiểm tra các trường còn lại có đúng với dữ liệu rên giao diện</p>	<p>truyền vào phải đúng kiểu dữ liệu</p> <p>Khi thêm tự động mã phải tự động tăng theo số dòng và không bị trùng</p> <p>Lấy danh sách cột IsDelete=0</p>	<p>Tất cả dữ liệu truyền vào phải đúng kiểu dữ liệu</p> <p>Khi thêm tự động mã phải tự động tăng theo số dòng và không bị trùng</p> <p>Lấy danh sách cột IsDelete=0</p>	
8	<p>Kiểm tra các trường dữ liệu của bảng KHACHHANG</p>	<p>1.Kiểm tra đã có dữ liệu khách hàng trong bảng chưa</p> <p>2.Kiểm tra các trường dữ liệu có đúng với dữ liệu trên giao diện</p> <p>3. Kiểm tra các trường bắt buộc có null không</p> <p>4.Kiểm tra độ dài, kiểu dữ liệu,của các trường dữ liệu</p>	<p>Trường MAKH không cho null</p> <p>Tất cả dữ liệu truyền vào phải đúng kiểu dữ liệu</p> <p>Khi thêm tự động mã phải tự động tăng theo số dòng và không bị trùng</p> <p>Lấy danh sách cột IsDelete=0</p>	<p>Trường MAKH không cho null</p> <p>Tất cả dữ liệu truyền vào phải đúng kiểu dữ liệu</p> <p>Khi thêm tự động mã phải tự động tăng theo số dòng và không bị trùng</p> <p>Lấy danh sách cột IsDelete=0</p>	PASS

*Bảng 3.13 Test case kiểm tra cơ sở dữ liệu cho chức năng mua vé*

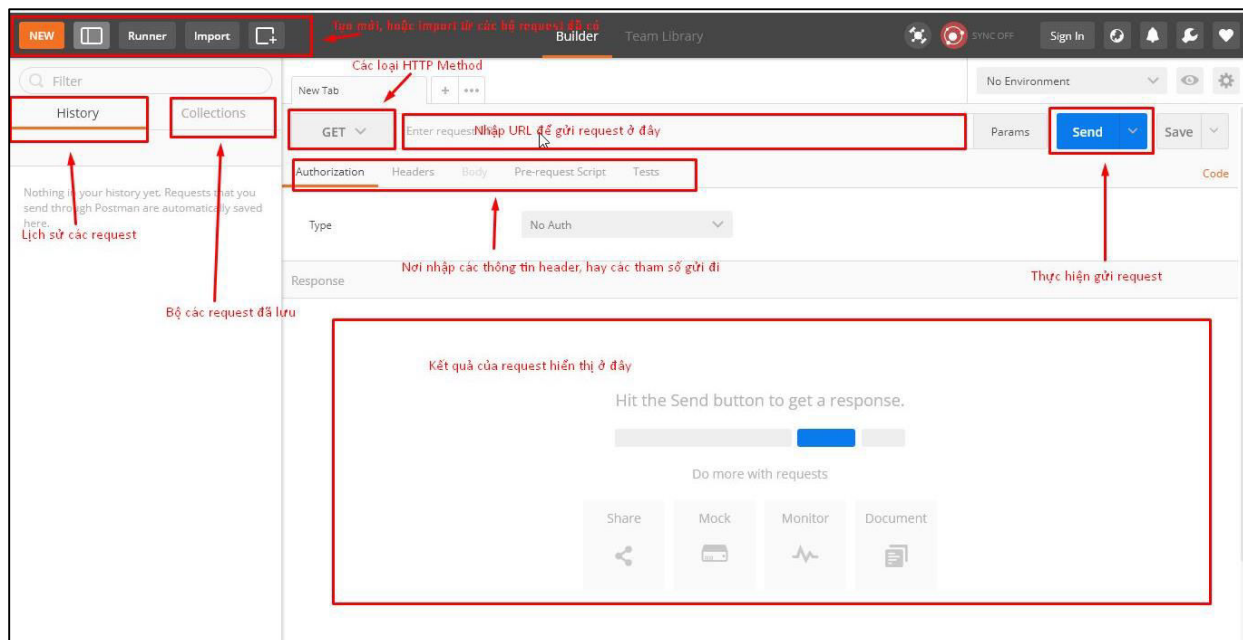
## 3.6. Kiểm thử API

### 3.6.1. Giới thiệu công cụ Postman

Khái niệm: Postman là 1 ứng dụng REST Client, dùng để thực hiện test, gửi các request, API mà không cần sử dụng browser.

Cài đặt Postman: Các bạn có thể download postman tại

<https://www.getpostman.com/>



Hình 3.7 Giao diện của Postman

### 3.6.2. Sử dụng Postman test chức năng đặt vé xe trong Phần mềm quản lý ứng dụng bán vé trực tuyến của công ty vận tải xe khách Thành Bưởi

#### 3.6.2.1. Trên controller DiemDi

- Đặc tả: Lấy tất cả các điểm đi có trong table TUYENXE. Dựa vào mỗi điểm đi sẽ lấy được các điểm đến tương ứng với điểm đi đó.
- Mã lệnh của chương trình:

```
[HttpGet]
[Route("api/DiemDi/Load_DiemDi")]
public HttpResponseMessage Load_DiemDi()
{
    IEnumerable<TUYENXE> tx = new List<TUYENXE>();
```

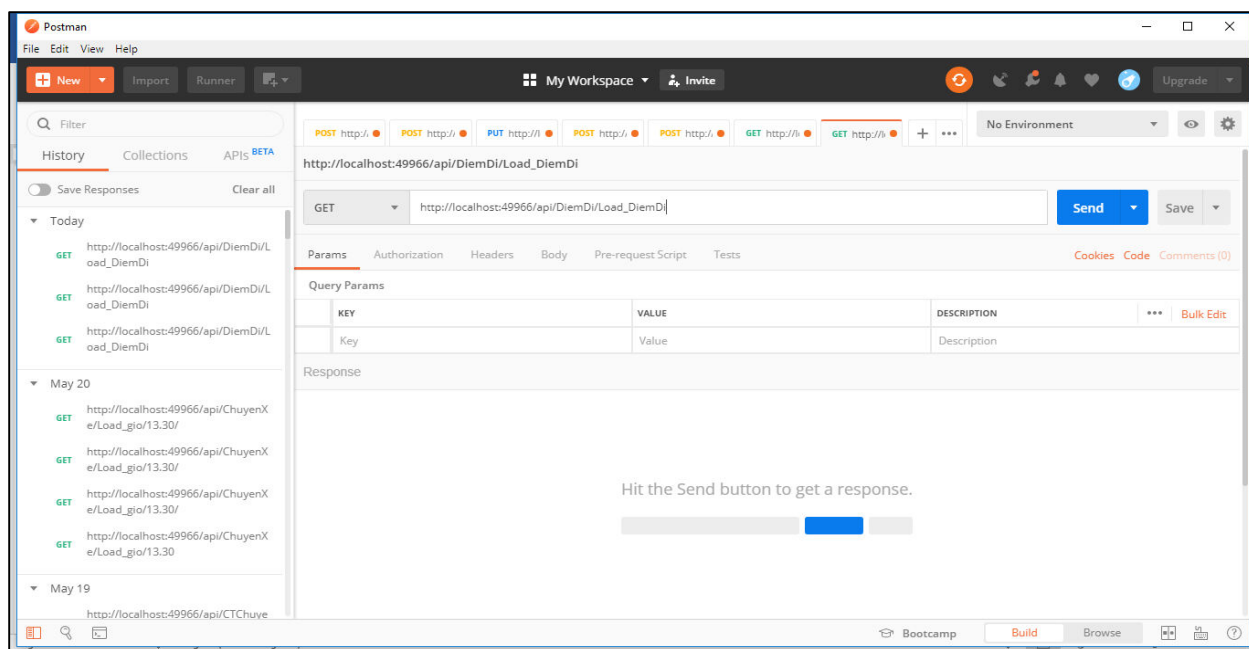
```
tx = (from e in ent.TUYENXEs select new { diemdi = e.DIEMDI
}).ToList().Distinct().Select(x => new TUYENXE { DIEMDI = x.diemdi
}).Distinct();
```

```
    List<DIEMDI> dIEMDIIs = new List<DIEMDI>();
    foreach(TUYENXE ddi in tx )
    {
        DIEMDI iEMDI = new DIEMDI();
        iEMDI = (from es in ent.DIEMDIIs where es.MADIEMDI == ddi.DIEMDI
select es).ToList<DIEMDI>().FirstOrDefault();
        iEMDI.lstDIEMDEN = Load_DiemDen(iEMDI.MADIEMDI); // Thêm
thuộc tính lstDIEMDEN vào đối tượng DIEMDI
        dIEMDIIs.Add(iEMDI);
    }
```

```
IEnumerable<DIEMDI> lstDiemdi = dIEMDIIs.Select(x => new DIEMDI {
MADIEMDI = x.MADIEMDI, TENDIEMDI = x.TENDIEMDI,
lstDIEMDEN = x.lstDIEMDEN }).ToList();
```

```
    return Request.CreateResponse(HttpStatusCode.OK, lstDiemdi);
}
```

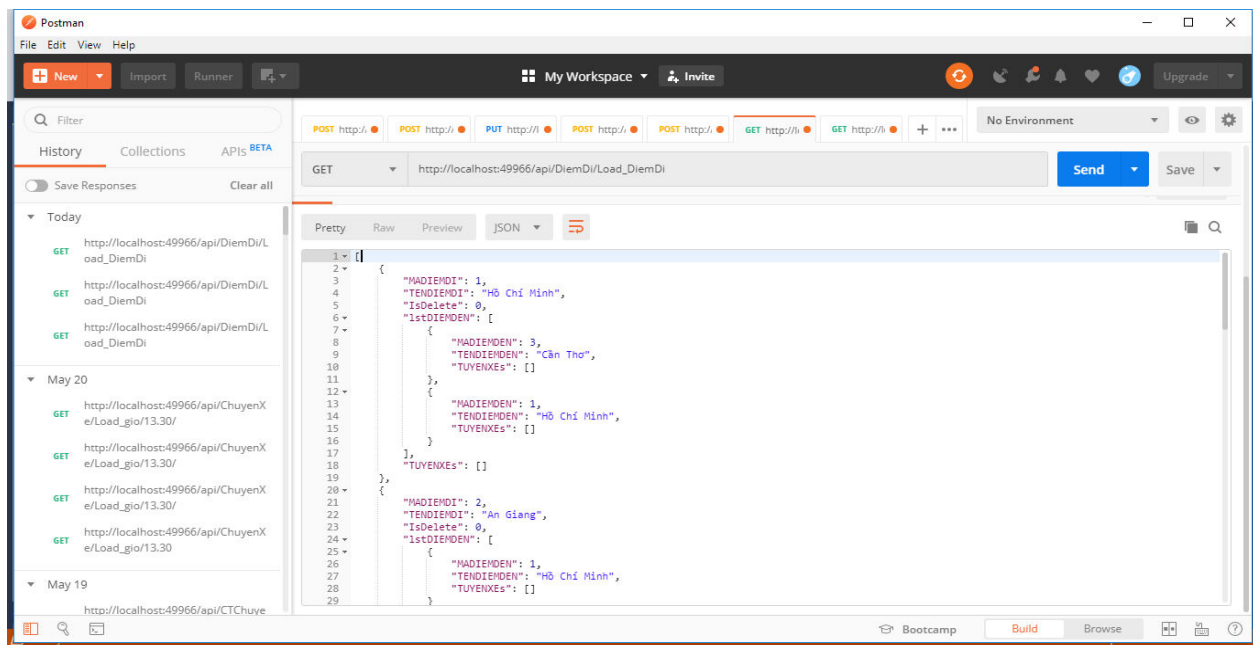
- Tên hàm API để lấy được tất cả các điểm đi có trong tuyến xe, dựa vào điểm đi lấy được các điểm đến trong CSDL: Load\_DiemDi. Phương thức của API là GET.
- Link check gồm 3 phần: link API trên máy + tên Controller + tên hàm.  
[http://localhost:49966/api/DiemDi/Load\\_DiemDi](http://localhost:49966/api/DiemDi/Load_DiemDi)



Hình 3.8 Giao diện chuẩn bị check API Load\_DiemDi

TITLE	STEP BY STEP	EXPECTED RESULT	ACTUAL RESULT	STATUS
Load được tất cả điểm điểm đi trong bảng TUYENXE	<p>1: Mở source code chứa Controller DiemDi.</p> <p>2: Khởi động Postman. Chọn phương thức GET.</p> <p>3: Nhập link check: <a href="http://localhost:4996/api/DiemDi/Load_DiemDi">http://localhost:4996/api/DiemDi/Load_DiemDi</a></p> <p>4: Click Send. Và xem kết quả.</p>	Kết quả trả về thông tin của tất cả điểm đi và điểm đến tương ứng trong Table TUYENXE	Kết quả trả về thông tin của tất cả điểm đi và điểm đến tương ứng trong Table TUYENXE	PASS

Bảng 3.14 Test case kiểm tra API Load-DiemDi



*Hình 3.9 Kết quả kiểm tra API Load-DiemDi*

### 3.6.2.2. Trên controller ChuyenXe

- Đặc tả: Sau khi có được điểm đi và điểm đến của từng tuyến xe. Dùng hàm API Select\_ChuyenXe với dữ liệu vào là điểm đi và điểm đến của từng tuyến xe và dữ liệu ra đó là danh sách giờ đi của chuyến xe thuộc tuyến xe đó.
- Mã lệnh của chương trình:

```

[HttpGet]
[Route("api/ChuyenXe/Select_ChuyenXe/{DiemDi}/{DiemDen}")]
public HttpResponseMessage Select_ChuyenXe(int DiemDi,int DiemDen)
{
    string MaTuyen = string.Empty;
    TUYENXE tUYENXE = (from e in ent.TUYENXEs where e.DIEMDI ==
DiemDi && e.DIEMDEN == DiemDen select
e).ToList<TUYENXE>().FirstOrDefault();
    if (tUYENXE != null)
    {
        MaTuyen = tUYENXE.MATUYEN.Trim();
    }

    IEnumerable<CHUYENXE> cHUYENXEs = new List<CHUYENXE>();

```

```

        cHUYENXEs = (from e in ent.CHUYENXEs where e.MATX ==
        MaTuyen select new { MaCX = e.MACX.Trim(), gioDi =
        e.GIODI.Trim(),MaTX=e.MATX.Trim(),LoaiChuyen=e.LOAICHUYEN.T
        rim() })
        .ToList().Select(x => new CHUYENXE { MACX = x.MaCX, GIODI =
        x.gioDi,MATX=x.MaTX,LOAICHUYEN=x.LoaiChuyen });
        if(cHUYENXEs!= null)
        return Request.CreateResponse(HttpStatusCode.OK, cHUYENXEs);
        return Request.CreateResponse(HttpStatusCode.NotFound,new object {
        });
    }
}

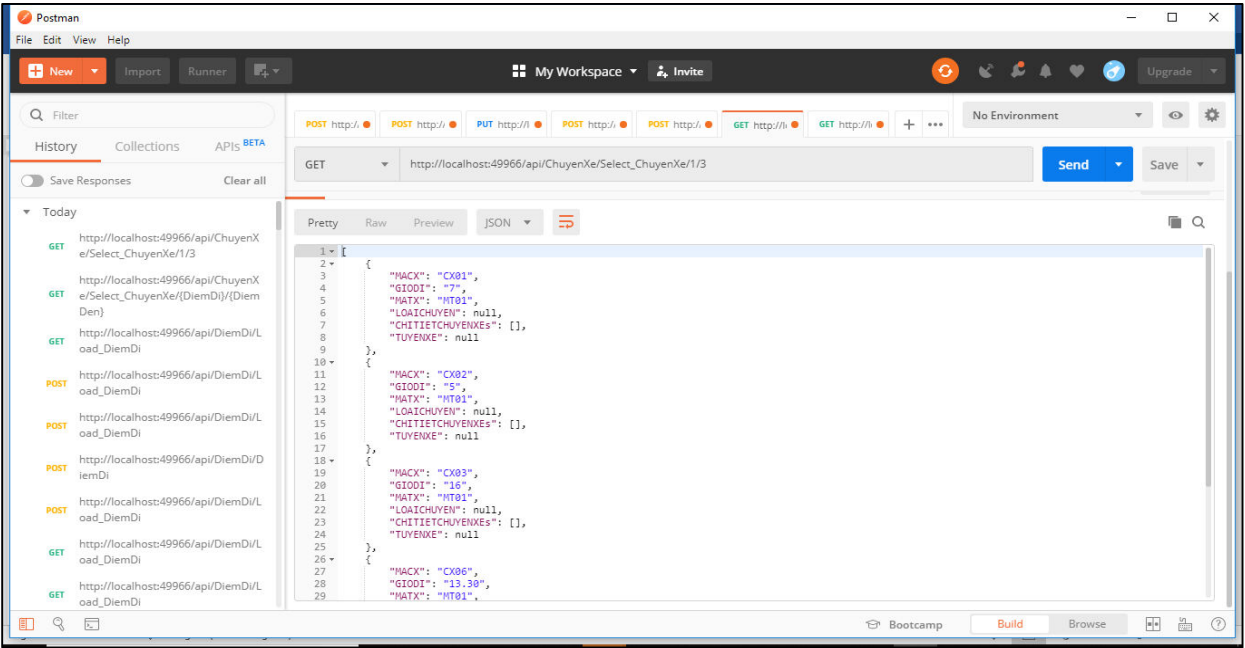
```

- Tên hàm API: Select\_ChuyenXe. Phương thức của API là GET.
- Link check API:

[http://localhost:49966/api/ChuyenXe/Select\\_ChuyenXe/{DiemDi}/{DiemDen}](http://localhost:49966/api/ChuyenXe/Select_ChuyenXe/{DiemDi}/{DiemDen})

ID	TITLE	STEP BY STEP	EXPECTED RESULT	ACTUAL RESULT	STATUS
1	Kết quả trả về danh sách giờ.	1: Mở source code chứa Controller ChuyenXe 2: Khởi động Postman. Chọn phương thức GET.. 3: Nhập link: Chọn dữ liệu vào DiemDi là 1, dữ liệu vào DiemDen là 3. <a href="http://localhost:49966/api/ChuyenXe/Select_ChuyenXe/1/3">http://localhost:49966/api/ChuyenXe/Select_ChuyenXe/1/3</a>	Kết quả trả về danh sách giờ của các chuyến thuộc tuyến xe có MaDiemDi = 1 và MaDiemDen = 3.	Kết quả trả về danh sách giờ của các chuyến thuộc tuyến xe có MaDiemDi = 1 và MaDiemDen = 3 => PASS.	PASS

		<a href="#">Xe/1/3</a>  4: Click Send. Đổi chiều với CSDL trong table CHUYENXE.			
--	--	---	--	--	--



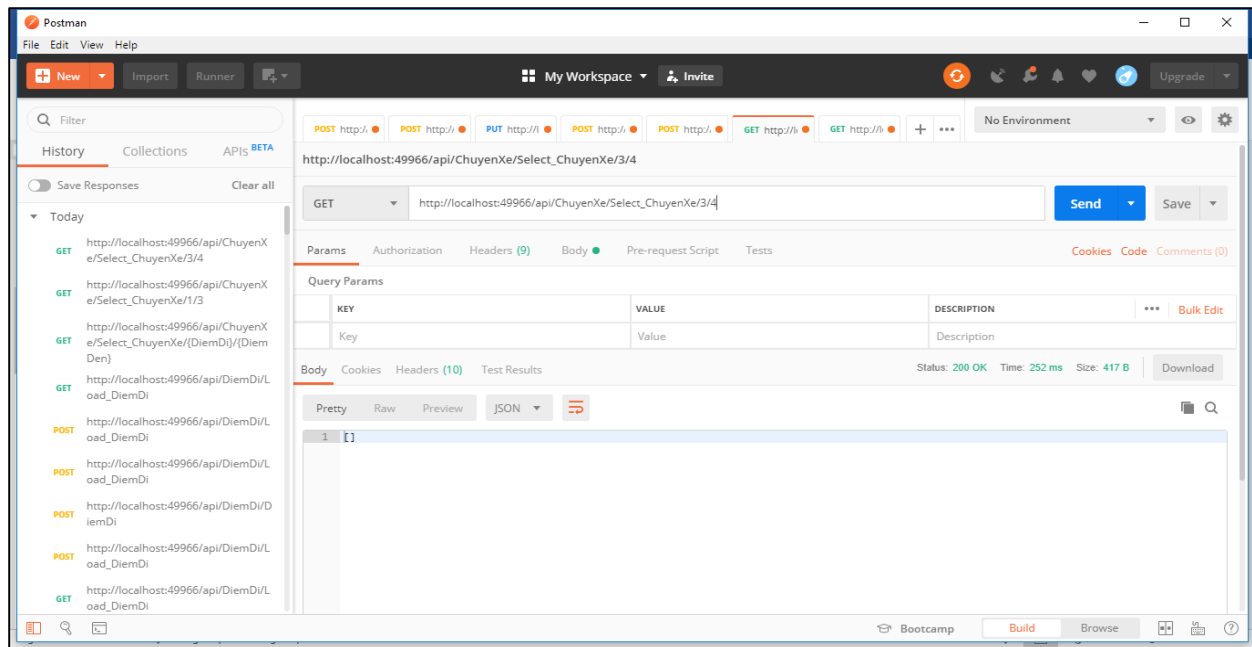
Hình 3.10 Kết quả danh sách giờ trả về của case 1

ID	TITLE	STEP BY STEP	EXPECTED RESULT	ACTUAL RESULT	STATUS
2	Kết quả trả về danh sách giờ rỗng	1: Mở Source code chứa Controller ChuyenXe.  2: Khởi động Postman. Chọn phương thức GET.	Kết quả trả về danh sách rỗng vì dữ liệu vào MaDiemDi và MaDiemDen không thuộc tuyến xe nào.	Kết quả trả về danh sách rỗng	PASS

		<p>3: Nhập dữ liệu vào DiemDi là 3, dữ liệu vào DiemDen là 4 vào link check.</p> <p><a href="http://localhost:49966/api/ChuyenXe/Select_ChuyenXe/3/4">http://localhost:49966/api/ChuyenXe/Select_ChuyenXe/3/4</a></p> <p>4: Click Send. Đối chiếu với CSDL trong table CHUYENXE.</p>			
--	--	--	--	--	--

*Bảng 3.15 Test case kiểm tra danh sách giờ trả về*





*Hình 3.11 Kết quả giờ trả về rỗng của case 2*

### 3.6.2.3. Trên Controller CTChuyenXe

- Đặc tả: Hàm API Select\_CTChuyenXe với dữ liệu vào gồm: mã chuyến xe và ngày đi. Kết quả trả về thông vé ghế của chi tiết chuyến xe thuộc chuyến xe và ngày đi đó.
- Mã lệnh của chương trình:

```
[HttpGet]
[Route("api/CTChuyenXe/Select_CTChuyenXe/{MaChuyen}/{NgayDi}")]
public HttpResponseMessage Select_CTChuyenXe(string MaChuyen, string NgayDi)
{
    //string datetime = NgayDi.ToString("MM/dd/yyyy").Replace('/', '-');
    DateTime date = DateTime.Parse(NgayDi);
    string mactchuyen = string.Empty;
    int a = ktctchuyen(MaChuyen, date);
    string loaichuyen = string.Empty;
    loaichuyen = getLoaiChuyen(MaChuyen);
    if (ktctchuyen(MaChuyen, date) == 1)
    {
        DataTable dt = null;
        dt = getctchuyen(date, MaChuyen);
    }
}
```

```

if (dt.Rows.Count < 1)
    mactchuyen = "";
else
    mactchuyen = dt.Rows[0].ItemArray[0].ToString().Trim();
IEnumerable<VE_GHE> lstghe = new List<VE_GHE>();
lstghe = (from e in ent.VE_GHE.Where(t =>
t.MAVEGHE.Contains(mactchuyen.Trim())) select new { MAVEGHE =
e.MAVEGHE, SOGHE = e.SOGHE, TRANGTHAI = e.TRANGTHAI
}).ToList().Distinct().Select(x => new VE_GHE { MAVEGHE =
x.MAVEGHE, SOGHE = x.SOGHE, TRANGTHAI = x.TRANGTHAI
}).Distinct();
//lstghe = (from e in ent.VE_GHE select new { MAVEGHE =
e.MAVEGHE , SOGHE = e.SOGHE , TRANGTHAI =
e.TRANGTHAI}).ToList().Distinct().Select(x => new VE_GHE {
MAVEGHE = x.MAVEGHE , SOGHE = x.SOGHE , TRANGTHAI =
x.TRANGTHAI }).Distinct();
List<VE_GHE> Ve_Ghe = new List<VE_GHE>();
foreach (VE_GHE vg in lstghe)
{
    VE_GHE v_g = new VE_GHE();
    v_g = (from es in ent.VE_GHE where es.MAVEGHE == vg.MAVEGHE
select es).ToList<VE_GHE>().FirstOrDefault();
    Ve_Ghe.Add(v_g);
}
IEnumerable<VE_GHE> lstveghe = Ve_Ghe.Select(x => new VE_GHE {
MAVEGHE = x.MAVEGHE.Trim(), SOGHE = x.SOGHE, TRANGTHAI
= x.TRANGTHAI.Trim() }).OrderByDescending(x =>
x.SOGHE).ToList();
if (lstveghe != null)
{
    return Request.CreateResponse(HttpStatusCode.OK, lstveghe);
}
return Request.CreateResponse(HttpStatusCode.InternalServerError,
false);
}
else
{
    int dem = countctchuyen() + 1;
    String mv1;
    if (dem >= 10)
    {
        mactchuyen = "MACTC" + dem.ToString();
    }
}

```

```

else
    mactchuyen = "MACTC0" + dem.ToString();
int? khoangcach = 0;
int slGhe = 0;
string masdGhe = string.Empty;
khoangcach = getKhoangCach(MaChuyen);
if (loaichuyen == "Đường ngắn")
{
    slGhe = 29;
    masdGhe = "SDG01";
}
else
{
    masdGhe = "SDG02";
    slGhe = 44;
}
CHITIETCHUYENXE ctcx = new CHITIETCHUYENXE();
ctcx.MACHITIETCHUYEN = mactchuyen;
ctcx.MACHUYEN = MaChuyen;
ctcx.NGAYDI = date;
ctcx.TAIXE1 = "TX01";
if (InsertAuto_CTCX(ctcx) == true)
{
    for(int i=1;i<= slGhe;i++)
    {
        VE_GHE ve = new VE_GHE();
        ve.MAVEGHE = mactchuyen + i;
        ve.MASDGHE = masdGhe;
        ve.SOGHE = i;
        ve.TRANGTHAI = "0";
        if(Ghe.Insert_VeGhe(ve) == true)
        {
            string sl = "thêm ghế số: " + i + "mã vé: " + ve.MAVEGHE;
        }
    }
    else
    {
        return Request.CreateResponse(HttpStatusCode.InternalServerError,
false);
    }
}
IEnumerable<VE_GHE> lstghe = new List<VE_GHE>();
lstghe = (from e in ent.VE_GHE.Where(t =>
t.MAVEGHE.Contains(mactchuyen.Trim())) select new { MAVEGHE =

```

```

e.MAVEGHE, SOGHE = e.SOGHE, TRANGTHAI = e.TRANGTHAI
}).ToList().Distinct().Select(x => new VE_GHE { MAVEGHE =
x.MAVEGHE, SOGHE = x.SOGHE, TRANGTHAI = x.TRANGTHAI
}).Distinct();
//lstghe = (from e in ent.VE_GHE select new { MAVEGHE =
e.MAVEGHE , SOGHE = e.SOGHE , TRANGTHAI =
e.TRANGTHAI}).ToList().Distinct().Select(x => new VE_GHE {
MAVEGHE = x.MAVEGHE , SOGHE = x.SOGHE , TRANGTHAI =
x.TRANGTHAI }).Distinct();
List<VE_GHE> Ve_Ghe = new List<VE_GHE>();
foreach (VE_GHE vg in lstghe)
{
    VE_GHE v_g = new VE_GHE();
    v_g = (from es in ent.VE_GHE where es.MAVEGHE == vg.MAVEGHE
select es).ToList<VE_GHE>().FirstOrDefault();
    Ve_Ghe.Add(v_g);
}

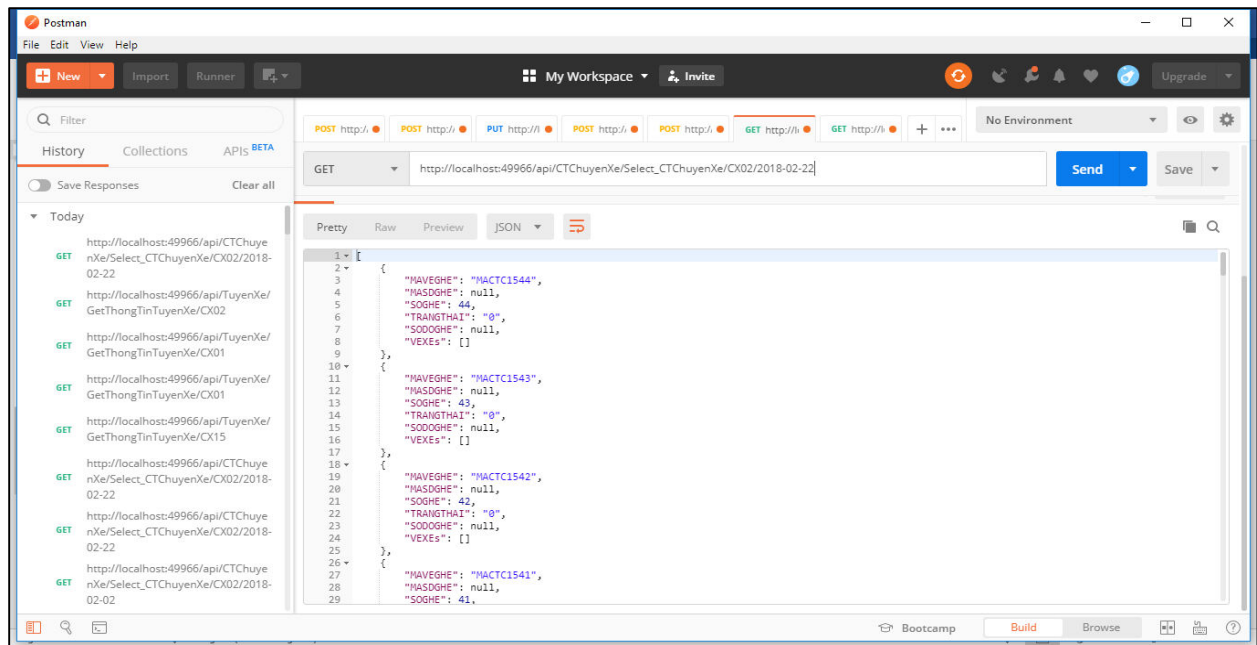
IEnumerable<VE_GHE> lstveghe = Ve_Ghe.Select(x => new VE_GHE {
MAVEGHE = x.MAVEGHE.Trim(), SOGHE = x.SOGHE, TRANGTHAI
= x.TRANGTHAI.Trim() }).ToList();
if (lstveghe != null)
{
    return Request.CreateResponse(HttpStatusCode.OK, lstveghe);
}
Return
Request.CreateResponse(HttpStatusCode.InternalServerError, false);
}
else
    return Request.CreateResponse(HttpStatusCode.InternalServerError,
false);
}
}

```

- Tên hàm API: Select\_CTChuyenXe. Phương thức của API là GET.
- Link check API:

[http://localhost:49966/api/CTChuyenXe/Select\\_CTChuyenXe/{MaChuyen}/{NgayDi}](http://localhost:49966/api/CTChuyenXe/Select_CTChuyenXe/{MaChuyen}/{NgayDi})

ID	TITLE	STEP BY STEP	EXPECTED RESULT	ACTUAL RESULT	STATUS
1	Đã tồn tại chi tiết chuyến xe có mã chuyến xe là CX02 và ngày đi là ngày 22/02/2018. Kết quả trả về danh sách thông tin vé ghế của chi tiết chuyến xe đó.	<p>1: Mở Source code chứa Controller CTChuyenXe.</p> <p>2: Khởi động Postman. Chọn phương thức GET.</p> <p>3: Nhập dữ liệu vào MaCX là CX02 và NgayDi là 22/02/2018 vào link check.</p> <p><a href="http://localhost:49966/api/CTChuyenXe/Select_CTChuyenXe/CX02/2018-02-22">http://localhost:49966/api/CTChuyenXe/Select_CTChuyenXe/CX02/2018-02-22</a></p> <p>4: Click Send. Đối chiếu với CSDL trong table VE-GHE.</p>	Kết quả trả về danh sách thông tin vé ghế của chi tiết chuyến xe CX02 trong ngày 22/02/2018.	Kết quả trả về danh sách thông tin vé ghế của chi tiết chuyến xe CX02 trong ngày 22/02/2018	PASS

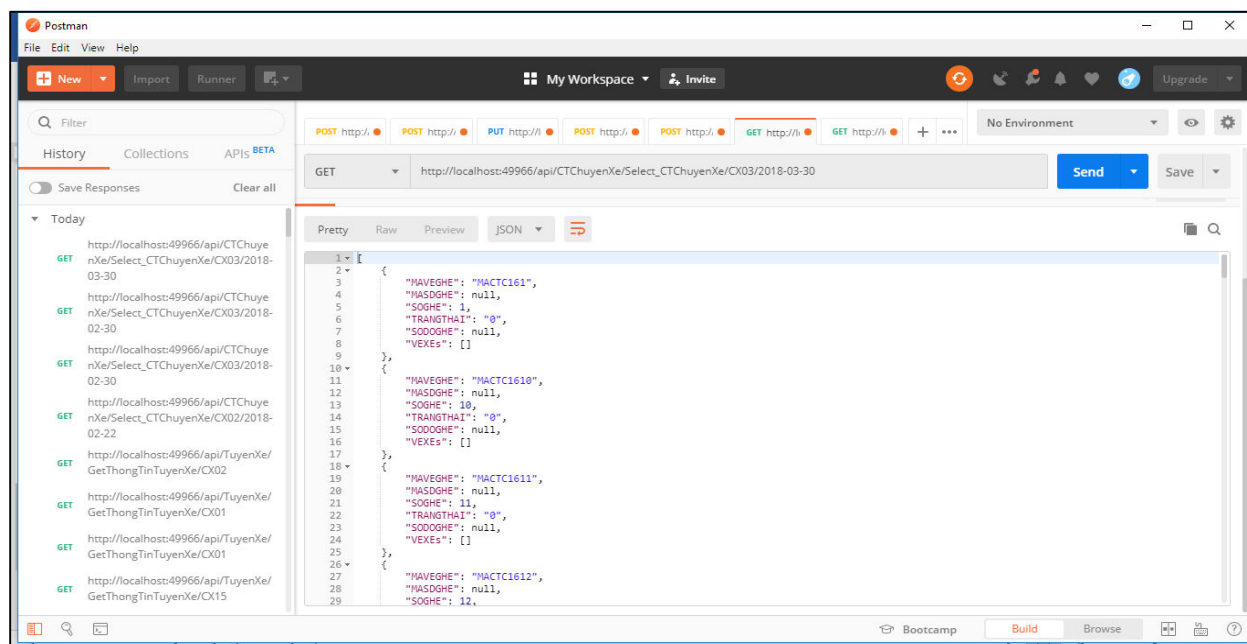


Hình 3.12 Kết quả test trả về danh sách thông tin vé ghế của chuyến xe đã tồn tại

ID	TITLE	STEP BY STEP	EXPECTED RESULT	ACTUAL RESULT	STATUS
2	Chưa tồn tại chi tiết chuyến xe có mã chuyến xe là CX03 và ngày đi là ngày 30/03/2018. Kết quả trả về danh sách thông tin vé ghế của chi tiết chuyến xe đó	<p>1: Mở Source code chứa Controller CTChuyenXe.</p> <p>2: Khởi động Postman. Chọn phương thức GET.</p> <p>3: Nhập dữ liệu vào MaCX là CX03 và NgayDi là 30/3/2018 vào</p>	Trong table CTChuyenXe sẽ tự động thêm 1 dòng dữ liệu chi tiết chuyến xe mới và tự động sinh danh sách vé ghế trong table VE-GHE.	Trong table CTChuyenXe sẽ tự động thêm 1 dòng dữ liệu chi tiết chuyến xe mới và tự động sinh danh sách vé ghế trong table VE-GHE	PASS

		<p>link check.</p> <p><a href="http://localhost:49966/api/CTChuyenXe/Select_CTChuyenXe/CX03/2018-03-30">http://localhost:49966/api/CTChuyenXe/Select_CTChuyenXe/CX03/2018-03-30</a></p> <p>4: Click Send.</p> <p>Đối chiếu với CSDL trong table CTChuyenXe và table VE-GHE.</p>			
--	--	---	--	--	--

*Bảng 3.16 Test case danh sách thông tin vé ghế của chuyến xe*



*Hình 3.13 Kết quả test trả về danh sách thông tin vé ghế của chuyến xe chưa tồn tại*

#### 3.6.2.4. Trên Controller ChuyenXe

- Đặc tả: khi thực thi hàm API getBangGia với dữ liệu vào là mã chuyển xe và mỗi chuyển xe sẽ thuộc vào một tuyến xe. Kết quả trả về là thông tin bảng giá của tuyến xe mà chuyển xe đó đi.
- Mã lệnh của chương trình

```
[HttpGet]
[Route("api/ChuyenXe/getBangGia/{MaCX}")]
public HttpResponseMessage getBangGia(string MACX)
{
    string MaTuyen = string.Empty;
    TUYENXE tuyenXE = (from e in ent.TUYENXEs
    join cx in ent.CHUYENXEs on e.MATUYEN equals cx.MATX
    where cx.MACX == MACX select e)
    .ToList<TUYENXE>().FirstOrDefault();
    if (tuyenXE != null)
    {
        MaTuyen = tuyenXE.MATUYEN.Trim();
    }
    IEnumerable<TUYENXE> tUYENXEs = new List<TUYENXE>();
    tUYENXEs = (from e in ent.TUYENXEs where e.MATUYEN ==
    MaTuyen select new { MaTX = e.MATUYEN.Trim(), BangGia =
    e.BANGGIA })
    .ToList().Select(x => new TUYENXE { MATUYEN = x.MaTX,
    BANGGIA = x.BangGia });
    if (tUYENXEs != null)
        return Request.CreateResponse(HttpStatusCode.OK, tUYENXEs);
    return Request.CreateResponse(HttpStatusCode.InternalServerError,
    false);
}
```

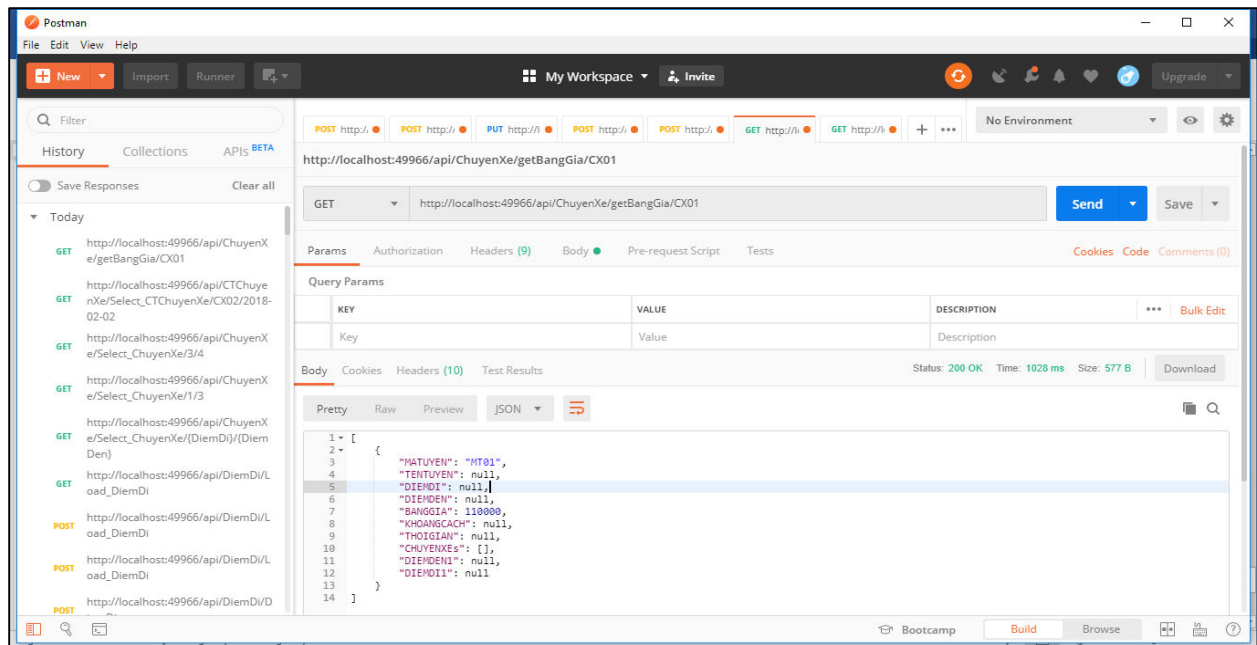
- Tên hàm API: getBangGia. Phương thức của API là GET.
- Link check API:

<http://localhost:49966/api/ChuyenXe/getBangGia/{MaCX}>



ID	TITLE	STEP BY STEP	EXPECTED RESULT	ACTUAL RESULT	STATUS
1	Trả về thông tin bảng giá của mã chuyến xe đã chọn	<p>1: Mở Source code chứa Controller ChuyenXe.</p> <p>2: Khởi động Postman. Chọn phương thức GET.</p> <p>3: Nhập dữ liệu vào MaCX là CX01 vào link check.</p> <p><a href="http://localhost:49966/api/ChuyenXe/getBangGia/CX01">http://localhost:49966/api/ChuyenXe/getBangGia/CX01</a></p> <p>4: Click Send. Đối chiếu với CSDL trong table TUYENXE.</p>	Kết quả trả về thông tin bảng giá của tuyến xe mà chuyển xe đó đi.	Kết quả trả về thông tin bảng giá của tuyến xe mà chuyển xe đó đi	PASS

*Bảng 3.17 Test case kiểm tra danh sách bảng giá của tuyến xe*



Hình 3.14 Kết quả trả về danh sách bảng giá của tuyến xe

### 3.6.2.5. Trên Controller TuyenXe

- Đặc tả: Trên hàm API GetThongTinTuyenXe với dữ liệu đầu vào là mã chuyển xe. Kết quả trả về là thông tin đầy đủ của 1 tuyến xe của chuyển xe đó đi.
- Mã lệnh của chương trình:

```

[HttpGet]
[Route("api/TuyenXe/GetThongTinTuyenXe/{maCX}")]
public HttpResponseMessage GetThongTinTuyenXe(string maCX)
{
    CHUYENXE cHUYENXE = (from e in ent.CHUYENXEs where
    e.MACX == maCX select e).ToList<CHUYENXE>().FirstOrDefault();
    TUYENXE tUYENXE = (from e in ent.TUYENXEs where e.MATUYEN
    == cHUYENXE.MATX.Trim() select
    e).ToList<TUYENXE>().FirstOrDefault();
    if (tUYENXE != null)
    {
        TUYENXE tuyenxe = new TUYENXE();
        tuyenxe.MATUYEN = tUYENXE.MATUYEN.Trim();
        tuyenxe.TENTUYEN = tUYENXE.TENTUYEN;
        tuyenxe.DIEMDI = tUYENXE.DIEMDI;
    }
}
  
```

```

tuyenxe.DIEMDEN = tUYENXE.DIEMDEN;
tuyenxe.BANGGIA = tUYENXE.BANGGIA;
tuyenxe.KHOANGCACH = tUYENXE.KHOANGCACH;
tuyenxe.THUOIGIAN = tUYENXE.THUOIGIAN;
return Request.CreateResponse(HttpStatusCode.OK, tuyenxe);
}
return null;
}

```

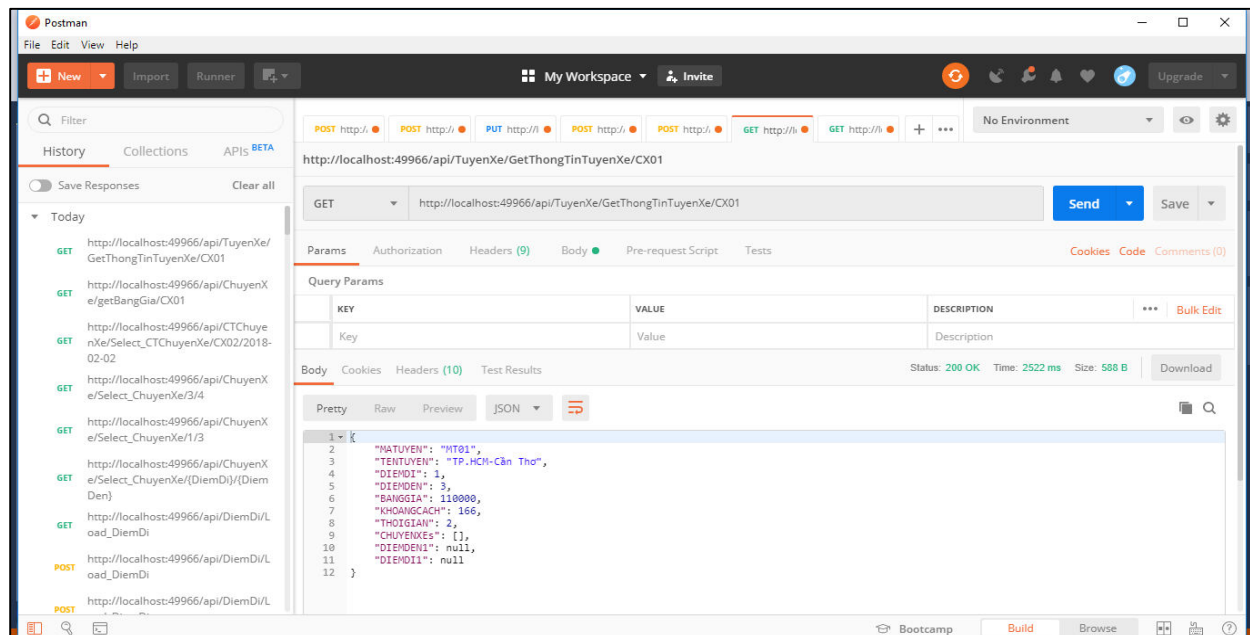
- Tên hàm API: GetThongTinTuyenXe. Phương thức của API là GET.
- Link check API:

<http://localhost:49966/api/TuyenXe/GetThongTinTuyenXe/{maCX}>

ID	TITLE	STEP BY STEP	EXPECTED RESULT	ACTUAL RESULT	STATUS
1	Kết quả trả về thông tin tuyến xe của mã chuyến xe CX01.	<p>1: Mở Source code chứa Controller TuyenXe.</p> <p>2: Khởi động Postman. Chọn phương thức GET.</p> <p>3: Nhập dữ liệu vào MaCX là CX01 vào link check.</p> <p><a href="http://localhost:49966/api/ChuyenXe/getBangGia/CX01">http://localhost:49966/api/ChuyenXe/getBangGia/CX01</a></p> <p>4: Click Send. Đối chiếu với CSDL</p>	Kết quả trả về thông tin của chuyến xe có mã CX01.	Kết quả trả về thông tin của chuyến xe có mã CX01	PASS

		trong table TUYENXE.			
--	--	-------------------------	--	--	--

*Bảng 3.18 Test case kiểm tra thông tin chuyến xe*



*Hình 3.15 Kết quả thông tin chuyến xe trả về*

### 3.6.2.6. Trên Controller KháchHang

- Đặc tả: Thực thi hàm API InsertTicked dữ liệu đầu vào là thông tin của 1 đối tượng là KHACHHANG. Kết quả trả về thêm mới 1 khách hàng chưa có trong table KHACHHANG, thêm thông tin của 1 vé xe mới trong table VEXE và cập nhật lại trạng thái ghế trong table VE-GHE
- Mã lệnh của chương trình:

```

[HttpPost]
[Route("api/KhachHang/InsertTicked")]
public HttpResponseMessage InsertTicked([FromBody]KHACHHANG
khACHHHANG)
{
    try

```

```

{
string maKH = string.Empty;
KHACHHANG khachhang = new KHACHHANG();
// Thêm thuộc tính lstGhe vào đối tượng KHACHHANG
KHACHHANG khang = (from kh in ent.KHACHHANGs where
kh.DIENTHOAIKHACH == kHACHHANG.DIENTHOAIKHACH select
kh).ToList<KHACHHANG>().FirstOrDefault();
if (khang == null) //Nếu khách hàng đó chưa có trong CSDL => Tạo mới
KH , thêm Vé xe, update lại trạng thái ghế
{
int dem = countDSKH() + 1;
String mv1;
if (dem >= 10)
{
maKH = "KH" + dem.ToString();
}
else
maKH = "KH0" + dem.ToString();
if (!string.IsNullOrEmpty(maKH))
{
khachhang.MAKH = maKH;
khachhang.DIENTHOAIKHACH =
kHACHHANG.DIENTHOAIKHACH;
khachhang.TENKH = kHACHHANG.TENKH;
ent.KHACHHANGs.Add(khachhang);
ent.SaveChanges();
foreach(VE_GHE vg in kHACHHANG.lstGhe)
{
if(updateVe(vg.MAVEGHE) == true)
{
string mactchuyen = string.Empty;
//int dem = 0;
int l = countVEXE() + 1;
String mv=string.Empty;
if (l >= 10)
{
mv = "MV" + l.ToString();
}
else
mv = "MV0" + l.ToString();

DataTable dt = null;

```

```

        dt = ct.gettctchuyen(DateTime.Parse(kHACHHANG.NGAYDI),
kHACHHANG.MACX);
        if (dt.Rows.Count < 1)
            mactchuyen = "";
        else
            mactchuyen = dt.Rows[0].ItemArray[0].ToString().Trim();
        VEXE v = new VEXE();
        v.MAVE = mv;
        v.MACTCHUYEN = mactchuyen;
        v.MANV = "NV01";
        v.MAKH = maKH;
        v.MAVEGHE = vg.MAVEGHE;
        ent.VEXEs.Add(v);
        ent.SaveChanges();
    }
    else
    {
        return Request.CreateResponse(HttpStatusCode.InternalServerError,
false);
    }
}
return Request.CreateResponse(HttpStatusCode.OK, true);
}
}
else //nếu KH có rồi thêm vé xe của KH đó , update trạng thái ghế
{
    if (!string.IsNullOrEmpty(khang.MAKH))
    {
        foreach (VE_GHE vg in kHACHHANG.lstGhe)
        {
            if (updateVe(vg.MAVEGHE) == true)
            {
                string mactchuyen = string.Empty;
                //int dem = 0;
                int l = countVEXE() + 1;
                String mv;
                if (l >= 10)
                {
                    mv = "MV" + l.ToString();
                }
                else
                    mv = "MV0" + l.ToString();
            }
        }
    }
}

```

```

        DataTable dt = null;
        dt = ct.getctchuyen(DateTime.Parse(kHACHHANG.NGAYDI),
        kHACHHANG.MACX);
        if (dt.Rows.Count < 1)
            mactchuyen = "";
        else
            mactchuyen = dt.Rows[0].ItemArray[0].ToString().Trim();
        VEXE v = new VEXE();
        v.MAVE = mv;
        v.MACTCHUYEN = mactchuyen;
        v.MANV = "NV01";
        v.MAKH = khang.MAKH.Trim();
        v.MAVEGHE = vg.MAVEGHE;
        ent.VEXEs.Add(v);
        ent.SaveChanges();
    }
    else
    {
        return Request.CreateResponse(HttpStatusCode.InternalServerError,
        false);
    }
    }
    return Request.CreateResponse(HttpStatusCode.OK, true);
    }
    }
    return Request.CreateResponse(HttpStatusCode.InternalServerError,
    false);

    }
    catch (Exception ex)
    {
        return Request.CreateResponse(HttpStatusCode.InternalServerError,
        false);
    }
    }

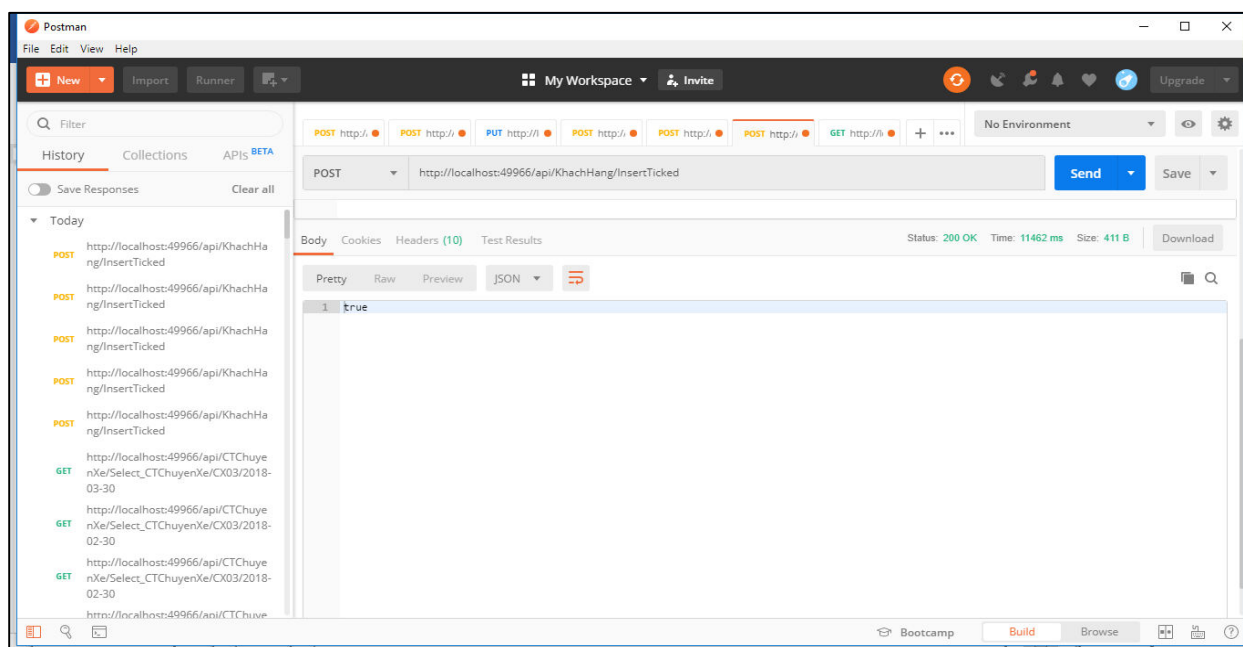
```

ID	TITLE	STEP BY STEP	EXPECTED RESULT	ACTUAL RESULT	STAT US
----	-------	--------------	--------------------	------------------	------------

1	Kiểm tra Khách hàng cũ	<p>1: Mở Source code chứa Controller KháchHang.</p> <p>2: Khởi động Postman. Chọn phương thức POST.</p> <p>3: Nhập link check API:  <a href="http://localhost:49966/api/KhachHang/InsertTicked">http://localhost:49966/api/KhachHang/InsertTicked</a></p> <p>4: Nhập dữ liệu đầu vào vào phần Body của Postman:</p> <pre>{</pre> <p>DIENTHOAIKHACH:</p> <p>'0932123456',</p> <p>MaCX: 'CX03 ',</p> <p>NGAYDI: '2018-03-30',</p> <p>lstGhe:[{</p> <p>MAVEGHE:'MACTC1639',</p> <p>MASDGHE:'SDG</p>	Kết quả trả về true. Trong table VEXE sẽ tự động thêm 1 vé mới cho khách hàng trên và trong table VE-GHE cột trạng thái sẽ cập nhật lại bằng 1.	Kết quả trả về true. Trong table VEXE sẽ tự động thêm 1 vé mới cho khách hàng trên và trong table VE-GHE cột trạng thái sẽ cập nhật lại bằng 1.	PASS
---	------------------------	--	---	---	------



		02', SOGHE:'39', TRANGTHAI:'0'} }] 5: Click Send. Xem thay đổi trong CSDL table KHACHHANG, VEXE và VE- GHE.			
--	--	--	--	--	--



*Hình 3.16 Kết quả kiểm tra hành khách cũ*

ID	TITLE	STEP BY STEP	EXPECTED RESULT	ACTUAL RESULT	STATUS
2	Kiểm tra khách hàng mới	1: Mở Source code chứa	Kết quả trả về true. Trong	Kết quả trả về true. Trong	PASS

		<p>Controller KhachHang.</p> <p>2: Khởi động Postman. Chọn phương thức POST</p> <p>3: Nhập link check API: <a href="http://localhost:49966/api/KhachHang/InsertTicked">http://localhost:49966/api/KhachHang/InsertTicked</a></p> <p>4: Nhập dữ liệu đầu vào vào phần Body của Postman:</p> <pre>{</pre> <p>DIENTHOAIKH ACH: '0933333333',</p> <p>MaCX: 'CX03 ',</p> <p>NGAYDI: '2018- 03-30',</p> <p>TENKH: 'Nguyễn Văn A',</p> <p>lstGhe:[{</p> <p>MAVEGHE:'MA CTC1637',</p> <p>MASDGHE:'SDG</p>	<p>table KHACHHAN</p> <p>G sẽ tự động thêm 1 khách hàng mới.</p> <p>Trong table VEXE sẽ thêm 1 vé mới cho khách hàng trên và trong table VE-GHE cột trạng thái sẽ cập nhật lại bằng 1.</p>	<p>table KHACHHAN</p> <p>G sẽ tự động thêm 1 khách hàng mới.</p> <p>Trong table VEXE sẽ thêm 1 vé mới cho khách hàng trên và trong table VE-GHE cột trạng thái sẽ cập nhật lại bằng 1.</p>	
--	--	---	--	--	--

		02', SOGHE:'37', TRANGTHAI:'0'} }] 5: Click Send. Xem thay đổi trong CSDL table KHACHHANG, VEXE và VE- GHE.			
--	--	--	--	--	--

*Bảng 3.19 Test case kiểm tra hành khách*