ĐẠI HỌC BÁCH KHOA HÀ NỘI

TRƯỜNG CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

BÁO CÁO HỌC PHẦN IT6481

KIỂM CHỨNG VÀ THẨM ĐỊNH PHẦN MỀM

Đề tài: Tìm hiểu phương pháp Event-B và ứng dụng vào bài toán đăng ký phòng họp

Giảng viên hướng dẫn: TS. Trần Nhật Hoá

Học viên: Đinh Hoàng Nam - 20222047M

Hà Nội, 2023

Mục lục

Chương 1: Giới thiệu đề tài.....4

1.1 Đặt vấn đề..... 4

1.2 Mục tiêu của đề tài..... 5

1.3 Phạm vi của đề tài..... 5

Chương 2: Cơ sở lý thuyết 7

2.1 Tổng quan về Phương pháp hình thức (Formal methods)..... 7

2.2 Tổng quan về Chứng minh định lý (Theorem proving) 8

2.3 Tổng quan về Event-B 9

2.4 Một số khái niệm quan trọng trong Event-B:..... 10

2.4.1 Máy (Machines) 10

2.4.2 Ngữ cảnh (Contexts) 11

Chương 3: Ứng dụng thực tế 13

3.1 Tổng quan bài toán Đăng ký phòng họp..... 13

3.2 Mô tả các chức năng hệ thống..... 13

3.2.1 Biểu đồ use case tổng quan chức năng hệ thống..... 13

3.2.2 Mô tả các chức năng hệ thống – mức cơ bản 14

3.2.3 Mô tả các chức năng hệ thống – cải tiến lần 1..... 17

3.2.4 Cây phân cấp máy (machines) cho bài toán đăng ký đặt phòng..... 19

3.2.5 Bảng các ràng buộc về trạng thái của hệ thống 20

3.3 Mô hình hoá hệ thống sử dụng Event-B với công cụ Rodin 20

3.3.1 Định nghĩa context ctx0..... 20

3.3.2 Đĩnh nghĩa machine m0..... 21

3.3.3 Định nghĩa context cxt1..... 28

3.3.4 Định nghĩa machine m1..... 28

3.4 Xây dựng chương trình..... 30

Chương 4: Kết luận *31*

4.1 Kết quả đạt được **31**

4.2 Hướng phát triển trong tương lai **32**

Chương 5: Tài liệu tham khảo *33*

Danh mục hình vẽ

Hình 1: Tổng quan các chức năng hệ thống 14
Hình 2: Định nghĩa context ctx0.....21
Hình 3: Xử lý cấu trúc dữ liệu bản ghi (hướng 1) 22
Hình 4: Hạn chế của hướng xử lý 1 23
Hình 5: Xử lý cấu trúc dữ liệu bản ghi (hướng 2) 24
Hình 6: Các invariants của máy m0.....24
Hình 7: Các invariants bổ sung của máy m1 28

Chương 1: Giới thiệu đề tài

1.1 Đặt vấn đề

Các sản phẩm của ngành công nghiệp phát triển phần cứng và phần mềm trong lĩnh vực công nghệ thông tin (CNTT) đóng một vai trò quan trọng trong nhiều lĩnh vực khác nhau của cuộc sống, và là một trong những yếu tố quan trọng trong việc phát triển kinh tế, xã hội. Từ các lĩnh vực như giải trí, giáo dục cho tới các lĩnh vực như y tế, nghiên cứu khoa học đều có sự hiện diện của các sản phẩm công nghệ thông tin. Chính sự phổ biến và áp dụng rộng rãi này cho thấy tầm quan trọng của việc hoạt động một cách đúng đắn và đáng tin cậy của các hệ thống phần cứng và phần mềm. Điều này càng được đặc biệt quan tâm trong các hệ thống đặc thù có yêu cầu về tính chính xác và an toàn cao như trong y tế, hàng không, sản xuất chip bán dẫn .v.v, vì một sai sót dù rất nhỏ cũng có thể gây ra thiệt hại cực kỳ to lớn về con người và/hoặc kinh tế. Cách phổ biến và cơ bản nhất để phát hiện và kiểm tra lỗi trong hệ thống CNTT là sử dụng kiểm thử, mô phỏng và đánh giá của con người. Tuy nhiên, những kỹ thuật này không thể bao quát được hết các cả các trường hợp và không thể đảm bảo chắc chắn rằng hệ thống đang kiểm thử không tồn tại lỗi. Chính điều này đã làm nảy sinh một nhu cầu cấp thiết cho một phương pháp khác giúp phát triển các hệ thống một cách chính xác, an toàn và đáng tin cậy hơn. Đây chính là động lực cho việc nghiên cứu và phát triển các giải pháp có tên gọi chung là "Phương pháp hình thức" (Formal methods).

Các phương pháp hình thức ưu việt hơn các phương pháp kiểm thử thông thường ở khả năng cho phép kiểm tra toàn diện và phát hiện ra những lỗi mà phương pháp kiểm thử nghiệm có thể bỏ sót. Các phương pháp hình thức này là các kỹ thuật nghiêm ngặt về mặt toán học để đặc tả, phân tích, phát triển và kiểm chứng các hệ thống phần mềm và phần cứng. Việc sử dụng các phương pháp hình thức cho thiết kế phần mềm và phần cứng được thúc đẩy bởi kỳ vọng rằng việc thực hiện phân tích toán học thích hợp có thể góp phần vào độ tin cậy và đúng đắn của thiết kế và giảm thiểu

tối đa các sai sót có thể xảy ra. Hai kỹ thuật phổ biến của phương pháp hình thức này là Kiểm tra mô hình (Model checking) và Chứng minh định lý (Theorem proving). Mỗi kỹ thuật này đều có nhưng ưu và nhược điểm riêng, nhưng điểm khác biệt lớn nhất là kỹ thuật Model checking thực hiện kiểm tra vét cạn các trạng thái (hữu hạn) của một mô hình. Vì vậy, phương pháp này sẽ gặp phải vấn đề khi không gian trạng thái bùng nổ. Trong khi đó, kỹ thuật Theorem proving có khả năng kiểm thử mô hình ở mọi kích cỡ, tuy nhiên công sức bỏ ra để thực hiện phương pháp này là khá đáng kể và thường yêu cầu có sự tham gia thực hiện bởi các chuyên gia.

Những phân tích trên cho thấy ý nghĩa to lớn và tính thiết thực của việc nghiên cứu về phương pháp hình thức. Vì vậy, trong đề tài này, em xin thực hiện tìm hiểu về một phương pháp hình thức, cụ thể là kỹ thuật Theorem proving, sử dụng công nghệ Event B và ứng dụng vào bài toán đăng ký phòng họp.

1.2 Mục tiêu của đề tài

Các mục tiêu chính của đề tài bao gồm:

- Tìm hiểu về các hạn chế của các phương pháp phát triển phần mềm truyền thống
- Tìm hiểu tổng quan về các phương pháp hình thức và hai kỹ thuật phổ biến là Model checking và Theorem proving
- Tìm hiểu và nắm được về phương pháp Event B
- Biết cách sử dụng công cụ Rodin để tạo ra các mô hình theo phương pháp Event-B
- Áp dụng kiến thức và công cụ đã học vào thực hành bài toán đăng ký phòng họp

1.3 Phạm vi của đề tài

Trên thế giới hiện có rất nhiều các nghiên cứu về phương pháp hình thức. Trong số đó Event B - một sự kế thừa và mở rộng của phương pháp B-Method với

nhiều ưu điểm là một phương pháp phổ biến và đánh giá cao. Chính vì vậy, trong khuôn khổ của đề tài, em xin tập trung vào nghiên cứu phương pháp Event B và công cụ hỗ trợ phát triển tương ứng của Event B là Rodin.

Chương 2: Cơ sở lý thuyết

2.1 Tổng quan về Phương pháp hình thức (Formal methods)

Phương pháp hình thức là các kỹ thuật được sử dụng để mô hình hóa các hệ thống phức tạp dưới dạng các thực thể toán học. Bằng cách xây dựng một mô hình toán học chặt chẽ cho một hệ thống phức tạp, chúng ta không chỉ có thể xác minh các thuộc tính của hệ thống một cách kỹ lưỡng hơn (so với cách họ có thể thông qua kiểm thử thông thường) mà còn sử dụng bằng chứng toán học như một phần bổ sung cho kiểm thử hệ thống để đảm bảo các hành vi của hệ thống hoạt động đúng đắn.

Các phương pháp hình thức thường áp dụng cách tiếp cận ba bước để mô hình hóa và đánh giá hệ thống. Trong quá trình đặc tả hình thức về hệ thống, một kỹ sư hoặc nhà thiết kế xác định một cách nghiêm ngặt bằng cách sử dụng ngôn ngữ mô hình hóa—thường bằng cách sử dụng cú pháp toán học và ngữ nghĩa hình thức để loại bỏ sự thiếu chính xác và mơ hồ. Điều này tương tự như việc viết ra các thông số kỹ thuật của hệ thống, mặc dù không phải bằng ngôn ngữ tự nhiên đơn giản. Từ đó, dựa trên đặc tả, các kỹ sư phát triển một bộ định lý về hành vi của một hệ thống. Các định lý này được kiểm chứng sử dụng các bằng chứng toán học—để đảm bảo rằng hành vi của hệ thống là nhất quán về mặt logic và thực sự là hành vi mong muốn. Vì điều này cho phép các nhà thiết kế và kỹ sư phát hiện ra các sai sót ngay cả trước khi thiết kế được triển khai thành mã nguồn, nên nó ngăn ngừa các lỗi tốn kém phát sinh trong các giai đoạn phát triển sau này. Cuối cùng, khi mô hình được chỉ định và xác minh, việc triển khai có thể bắt đầu bằng cách chuyển đổi đặc tả thành mã nguồn.

Các phương pháp hình thức có nhiều ưu điểm: chúng giúp làm rõ các thông số kỹ thuật của hệ thống và làm rõ các giả định ngầm. Chúng cũng giúp bộc lộ những sai sót trong yêu cầu hệ thống và tính chặt chẽ của của phương pháp này giúp hiểu rõ hơn về vấn đề. Vì các phương pháp này sử dụng ngôn ngữ hình thức nên nhiều thành viên trong nhóm phát triển có thể kiểm tra các thông số kỹ thuật một cách độc lập, từ đó giúp giải quyết sớm các lỗi trong quá trình phát triển. Tuy nhiên, các phương pháp

hình thức không thể thay thế hoàn toàn các phương pháp đảm bảo chất lượng. Đây là lý do tại sao chúng ta sử dụng các phương pháp này như một kỹ thuật bổ sung trong quá trình phát triển hệ thống.

2.2 Tổng quan về Chứng minh định lý (Theorem proving)

Kỹ thuật chứng minh định lý, trong bối cảnh của các phương pháp hình thức, là một phương pháp sử dụng suy luận logic và lý luận toán học để xác minh tính chính xác của các hành vi, tính chất và yêu cầu của hệ thống. Phương pháp này đóng vai trò như một biện pháp giúp đảm bảo tính đúng đắn của hệ thống và tránh những hậu quả tai hại có thể xảy ra do lỗi hệ thống.

Kỹ thuật chứng minh định lý liên quan đến việc xây dựng một bằng chứng hình thức, từng bước một, để chứng minh tính đúng đắn của các tính chất, yêu cầu hoặc bất biến cụ thể của hệ thống. Mỗi bước chứng minh đều được chứng minh thông qua việc áp dụng các quy tắc logic, suy luận và nguyên tắc toán học. Những bằng chứng này cung cấp bằng chứng không thể chối cãi rằng hành vi của hệ thống phù hợp với các thông số kỹ thuật hình thức của nó.Các công cụ chứng minh định lý tự động (automated theorem provers), thường được được trang bị các thuật toán thông minh, hỗ trợ xác minh các định lý phức tạp giúp loại bỏ lỗi của con người và đẩy nhanh quá trình xác minh.

Tuy nhiên, kỹ thuật này cũng tồn tại những thách thức nhất định. Các hệ thống phức tạp có thể dẫn đến những bằng chứng phức tạp và việc tìm ra những bằng chứng này có thể đòi hỏi nhiều công sức tính toán. Một số thuộc tính có thể không thể quyết định được, đặt ra những hạn chế về mức độ xác minh có thể đạt được. Tuy nhiên, việc chứng minh định lý, khi được sử dụng một cách thận trọng, có thể mang lại mức độ tin cậy cao nhất về tính đúng đắn và độ tin cậy của hệ thống, đặc biệt là trong các lĩnh vực đòi hỏi tối cao về sự an toàn và đúng đắn của hệ thống.

2.3 Tổng quan về Event-B

Event-B là một phương pháp hình thức và kỹ thuật mô hình hóa được sử dụng trong công nghệ phần mềm và phát triển hệ thống để đặc tả, thiết kế và kiểm thử các hệ thống phức tạp. Nó dựa trên cách tiếp cận chặt chẽ về toán học để phát triển hệ thống và cung cấp một cách có hệ thống để đảm bảo tính chính xác và độ tin cậy của hệ thống phần mềm và phần cứng. Event-B đặc biệt phù hợp cho các ứng dụng mà độ chính xác và tính đúng đúng của các hành vi trong hệ thống được đặt lên độ quan trọng hàng đầu, do đó việc kiểm thử nghiêm ngặt và áp dụng các phương pháp hình thức là cần thiết.

Một số đặc điểm của Event-B bao gồm:

1. Nền tảng toán học: Event-B được thành lập dựa trên logic toán học, lý thuyết tập hợp và các sự hình thức hoá. Nó sử dụng các ký hiệu và cấu trúc toán học để mô tả hành vi, thuộc tính và yêu cầu của hệ thống một cách chính xác.
2. Đặc tả hình thức: Event-B liên quan đến việc tạo ra các đặc tả hình thức cung cấp mô tả rõ ràng, rõ ràng về hành vi dự định của hệ thống. Các thông số kỹ thuật này đóng vai trò là nền tảng cho việc phát triển và kiểm thử hệ thống.
3. Phát triển dựa trên tinh chỉnh: Event-B thúc đẩy cách tiếp cận phát triển dựa trên hoàn thiện từng bước. Các kỹ sư bắt đầu với một mô hình trừu tượng của một hệ thống và dần dần tinh chỉnh nó thành các mô hình chi tiết hơn trong khi vẫn duy trì các thuộc tính chính xác. Quá trình tinh chỉnh có phương pháp này giúp quản lý sự phức tạp của các hệ thống lớn và phức tạp.
4. Kiểm thử: Event-B hỗ trợ kiểm thử hình thức, bao gồm việc chứng minh rằng hành vi của hệ thống tuân thủ các thông số kỹ thuật của nó. Việc kiểm thử này có thể bao gồm tính chính xác, an toàn, và các thuộc tính quan trọng khác.

5. Các ứng dụng quan trọng về sự an toàn và quan trọng về sứ mệnh: Event-B đặc biệt có giá trị trong các lĩnh vực mà tính chính xác và độ tin cậy của hệ thống phần

mềm và phần cứng là vô cùng quan trọng. Nó được sử dụng rộng rãi trong ngành hàng không vũ trụ, ô tô, chăm sóc sức khỏe và các ngành công nghiệp quan trọng về an toàn khác.

6. Hỗ trợ công cụ: Nền tảng Rodin là môi trường phát triển tích hợp phổ biến để lập mô hình và xác minh Event-B. Nó cung cấp các công cụ và tính năng để hỗ trợ các kỹ sư và nhà phát triển áp dụng Event-B một cách hiệu quả.
7. Tài liệu và giao tiếp: Event-B khuyến khích tài liệu có cấu trúc và rõ ràng về các thông số kỹ thuật của hệ thống, hỗ trợ giao tiếp giữa các nhóm phát triển và các bên liên quan. Tài liệu này phục vụ như một tài liệu tham khảo có giá trị trong quá trình phát triển hệ thống và hơn thế nữa.
8. Tích hợp với các phương pháp hình thức khác: Mặc dù Event-B là một phương pháp hình thức mạnh mẽ nhưng nó cũng có thể được tích hợp với các phương pháp hình thức khác, chẳng hạn như kiểm tra mô hình, để cung cấp cách tiếp cận toàn diện cho việc kiểm thử hệ thống.

2.4 Một số khái niệm quan trọng trong Event-B:

2.4.1 Máy (Machines)

Mục đích: Máy trong Event-B mô tả các hành vi động của hệ thống. Chúng thể hiện trạng thái của hệ thống, cách các trạng thái này phát triển theo thời gian và các sự kiện khác nhau kích hoạt quá trình chuyển đổi trạng thái. Máy móc được sử dụng để mô hình hóa chức năng và hành vi của hệ thống.

Các thành phần:

- Biến (Variables): Các biến thể hiện các trạng thái của các thành phần trong hệ thống. Chúng thể hiện các dữ liệu và sự thay đổi của các dữ liệu này dưới tác động của các sự kiện.

- Bất biến (Invariants): Các bất biến là các điều kiện logic mà các trạng thái của máy luôn cần phải đảm bảo tuân thủ. Chúng đóng vai trò như các ràng buộc lên các trạng thái của hệ thống.
- Sự kiện (Events): Các sự kiện mô tả các sự chuyển dịch trong các hành vi của hệ thống. Chúng đại diện cho các hành động có thể làm thay đổi trạng thái của hệ thống. Các sự kiện có các biện pháp bảo vệ (các điều kiện phải được thỏa mãn để sự kiện xảy ra) và các hành động (các đặt tả về cách trạng thái thay đổi khi sự kiện được thực thi).

Tinh chỉnh (Refinement): Sự phát triển của các máy thường được thực hiện thông qua các quá trình. Việc tinh chỉnh bao gồm việc bắt đầu với một máy trừu tượng và dần dần tinh chỉnh nó thành các mô hình chi tiết hơn trong khi vẫn bảo toàn các đặc tính đã được chứng minh ở mỗi cấp độ.

Tính mô-đun (Modules): Trong Event-B, hệ thống có thể được phân tách thành nhiều máy, mỗi máy mô hình hóa các khía cạnh hoặc thành phần khác nhau của hệ thống. Điều này thúc đẩy tính mô-đun và cho phép các nhà phát triển làm việc độc lập trên từng thành phần riêng lẻ.

2.4.2 Ngữ cảnh (Contexts)

Mục đích: Ngữ cảnh cung cấp các yếu tố tĩnh của một hệ thống. Chúng xác định các hằng số, tập hợp, tiên đề và định lý làm nền tảng cho hành vi của hệ thống. Ngữ cảnh thiết lập khuôn khổ toán học trong đó hệ thống vận hành.

Các thành phần:

- Hằng (Constants): Các hằng số biểu thị các giá trị và dữ liệu cố định không thay đổi trong quá trình thực thi hệ thống.
- Tập hợp (Sets): Tập hợp xác định nhóm các phần tử được sử dụng trong hệ thống, giúp cấu trúc và phân loại dữ liệu.
- Tiên đề (Axioms): Tiên đề là các mệnh đề logic thể hiện các thuộc tính và ràng buộc về các hằng số và tập hợp được xác định trong ngữ cảnh.

- Định lý (Theorems): Các định lý được sử dụng để phát biểu và chứng minh các chân lý toán học về các hằng số và tập hợp trong ngữ cảnh. Chúng cung cấp nền tảng cho việc suy luận về hệ thống.

Tính chia sẻ: Các ngữ cảnh có thể được chia sẻ giữa nhiều máy để đảm bảo rằng các phần khác nhau của hệ thống sử dụng cùng các định nghĩa cơ bản. Việc chia sẻ này thúc đẩy tính nhất quán và khả năng tái sử dụng.

Máy và ngữ cảnh trong Event-B phối hợp với nhau để cung cấp cách tiếp cận nghiêm ngặt về mặt toán học và có hệ thống để phát triển hệ thống. Máy nắm giữ các khía cạnh động của hệ thống, xác định cách nó hoạt động theo thời gian, trong khi ngữ cảnh xác định các phần tử tĩnh thiết lập nền tảng của hệ thống. Sự kết hợp giữa hành vi động và các ràng buộc tĩnh này cho phép thể hiện một cách toàn diện và có cấu trúc các hệ thống phức tạp, thúc đẩy tính chính xác, độ tin cậy và khả năng xác minh các thuộc tính thông qua các phương pháp hình thức.

Chương 3: Ứng dụng thực tế

3.1 Tổng quan bài toán Đăng ký phòng họp

Bài toán trong thực tế được mô tả như sau: một trường đại học có nhiều phòng họp/hội trường/giảng đường, và các cán bộ/giảng viên có thể đăng ký mượn các phòng này. Đơn đăng ký sẽ được trưởng phòng Hành chính Tổng hợp (HCTH) xét duyệt về nội dung và mục đích sử dụng (đồng ý hoặc từ chối). Các đơn đã được trưởng phòng HCTH đồng ý sẽ được chuyển qua cho Thư ký. Thư ký sẽ thực hiện kiểm tra lịch trong đơn đăng ký của phòng đó còn trống hay không. Nếu lịch còn trống, thư ký có thể đồng ý cấp phòng. Nếu có nhiều đơn bị trùng lịch, thư ký có thể lựa chọn đồng ý tối đa cho một đơn đăng ký. Đối với các đơn bị trùng còn lại, thư ký có thể liên hệ với người tạo đơn, và trao đổi về việc thay đổi phòng hoặc/và giờ còn trống. Nếu người tạo đơn đồng ý, thư ký tiến hành cập nhật và cấp phòng theo lịch mới. Ngược lại, thư ký từ chối đơn mượn phòng.

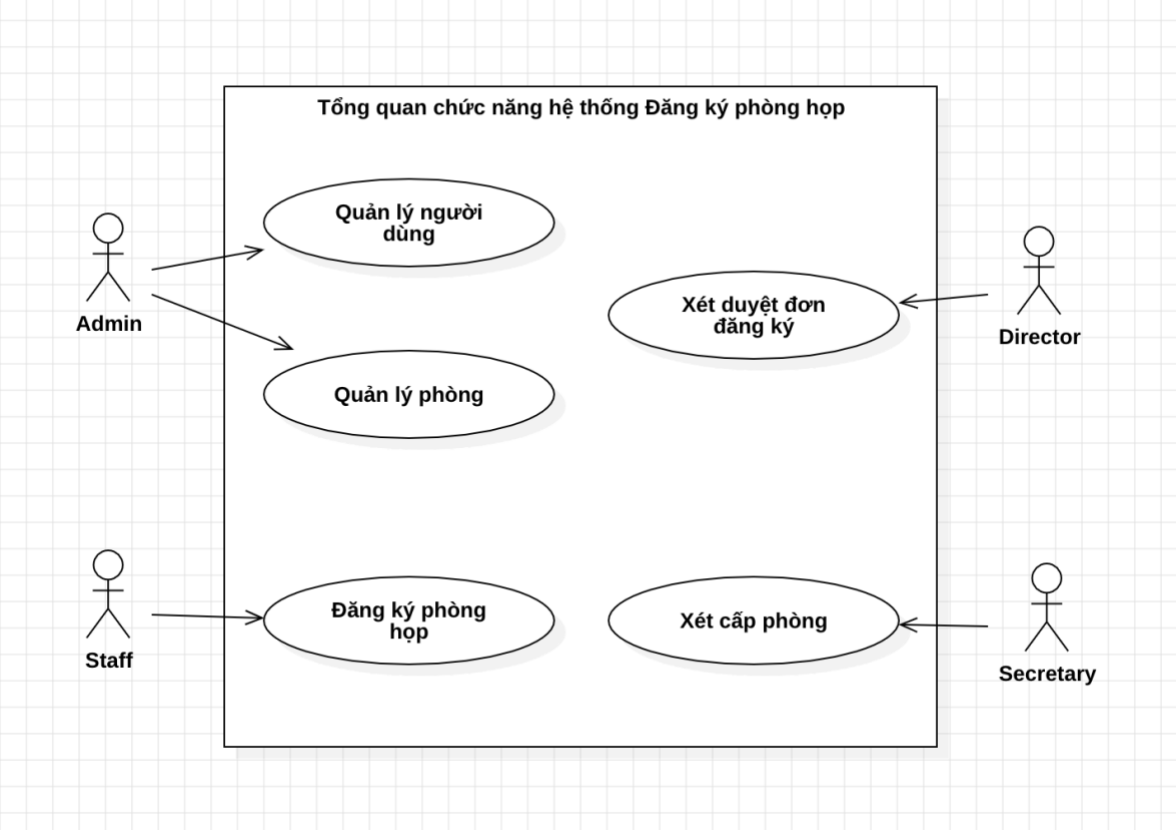
3.2 Mô tả các chức năng hệ thống

Dựa trên mô tả tổng quan của bài toán, dưới đây là mô tả về các chức năng hệ thống mà ta sẽ tiến hành mô hình hoá và xây dựng:

3.2.1 Biểu đồ use case tổng quan chức năng hệ thống

Các vai trò của người dùng trong hệ thống:

- **Admin**: Quản trị hệ thống, có quyền quản lý người và vai trò trong hệ thống. Admin cũng là người có quyền quản lý các phòng.
- **Director**: Trưởng phòng HCTH, có chức năng xét duyệt các đơn yêu cầu
- **Secretary**: Thực hiện xem xét cấp phòng cho các đơn đăng ký đã được đồng ý bởi Director
- **Staff**: Là các cán bộ, giảng viên; có quyền tạo các đơn đăng ký phòng



Hình 1: Tổng quan các chức năng hệ thống

3.2.2 Mô tả các chức năng hệ thống – mức cơ bản

a) Khởi tạo hệ thống

US-00	init
Ý nghĩa	Khởi tạo hệ thống
Mô tả	Điều kiện: Không Xảy ra khi: Bắt đầu hệ thống Thực hiện: Khởi tạo hệ thống với một người dùng duy nhất là root với vai trò Admin. Các dữ liệu khác như phòng, đơn đặt phòng được khởi tạo rỗng.

b) Thêm người dùng

US-01	add_user
Ý nghĩa	Thêm người dùng vào hệ thống
Mô tả	Điều kiện: Thực hiện bởi Admin. Người được thêm chưa tồn tại trong hệ thống. Xảy ra khi: Khi Admin muốn thêm người dùng. Thực hiện: Thêm người dùng với vai trò được chỉ ra vào hệ thống

c) Tạo mới phòng

US-02	add_room
Ý nghĩa	Thêm mới phòng vào hệ thống
Mô tả	Điều kiện: Thực hiện bởi Admin. Phòng chưa tồn tại trong hệ thống. Xảy ra khi: Khi Admin muốn thêm mới phòng họp. Thực hiện: Thêm phòng vào danh sách phòng trong hệ thống.

d) Tạo đơn đăng ký mượn phòng

US-03	create_reservation
Ý nghĩa	Tạo đơn đăng ký mượn phòng

Mô tả	Điều kiện: Thực hiện bởi một người dùng bất kỳ tồn tại trong hệ thống. Phòng muốn mượn phải tồn tại, và thời gian bắt đầu sử dụng phải nhỏ hơn thời gian kết thúc. Xảy ra khi: Người dùng muốn mượn phòng. Thực hiện: Thêm đơn vào danh sách đơn trong hệ thống. Trạng thái của đơn mới tạo sẽ là **Requested**.

e) Đồng ý đơn đăng ký

US-04	accept_reservation
Ý nghĩa	Đồng ý đơn đăng ký mượn phòng
Mô tả	Điều kiện: Thực hiện bởi Director. Đơn ở trạng thái **Requested.** Xảy ra khi: Trưởng phòng xử lý và đồng ý cho đơn đăng ký. Thực hiện: Cập nhật trạng thái của đơn sang **Accepted.**

f) Từ chối đơn đăng ký

US-05	decline_reservation
Ý nghĩa	Từ chối đơn đăng ký mượn phòng
Mô tả	Điều kiện: Thực hiện bởi Director. Đơn ở trạng thái **Requested.** Xảy ra khi: Trưởng phòng xử lý và đồng ý cho đơn đăng ký. Thực hiện: Cập nhật trạng thái của đơn sang **Accepted.**

g) Cấp phòng

US-06	reserve
Ý nghĩa	Cấp phòng cho đơn đăng ký đã được duyệt đồng ý
Mô tả	Điều kiện: Thực hiện bởi Secretary. Đơn ở trạng thái **Accepted.** Không bị trùng lặp thời gian và phòng với bất cứ đơn đã được cấp phòng nào khác. Xảy ra khi: Thư ký xử lý đơn đăng ký. Thực hiện: Cập nhật trạng thái của đơn sang **Reserved.**

h) Từ chối cấp phòng

US-07	deny
Ý nghĩa	Từ chối cấp phòng cho đơn đăng ký đã được duyệt đồng ý
Mô tả	Điều kiện: Thực hiện bởi Secretary. Đơn ở trạng thái **Accepted.** Xảy ra khi: Thư ký xử lý đơn đăng ký, thường do đơn bị trùng với lịch đã xếp cho đơn khác trước. Hoặc cũng có thể vì các lý do khác. Thực hiện: Cập nhật trạng thái của đơn sang **Denied.**

3.2.3 Mô tả các chức năng hệ thống – cải tiến lần 1

Trong lần cải tiến này, hệ thống bổ sung thêm tính năng đánh dấu phòng tạm dừng hoạt động. Khi này, người dùng không thể đăng ký phòng đã tạm dừng. Thư ký không

thể cấp phòng đã dừng hoạt động cho đơn. Lưu ý, phòng chỉ có thể đánh dấu tạm dừng hoạt động khi không có đơn được cấp phòng này đang trong quá trình sử dụng.

Trong phần này ta bổ sung thêm 2 tính năng mới: deactive_room và active_room, đồng thời, một số chức năng khác sẽ cần kiểm tra thêm điều kiện liên quan tới trạng thái hiện tại của phòng đang xét.

i) Đánh dấu phòng tạm dừng hoạt động

US-08	deactive_room
Ý nghĩa	Đánh dấu phòng tạm dừng hoạt động
Mô tả	Điều kiện: Thực hiện bởi Admin. Tất cả các đơn được cấp phòng này đã kết thúc. Xảy ra khi: Admin muốn dừng hoạt động của phòng. Thực hiện: Đưa phòng này vào danh sách phòng dừng hoạt động.

j) Mở hoạt động lại phòng

US-09	active_room
Ý nghĩa	Mở hoạt động lại phòng
Mô tả	Điều kiện: Thực hiện bởi Admin. Phòng đang xét ở trạng thái dừng hoạt động Xảy ra khi: Admin mở hoạt động phòng. Thực hiện: Loại bỏ phòng này khỏi danh sách phòng dừng hoạt động.

k) Tạo đơn đăng ký mượn phòng – tinh chỉnh

US-03-extended	create_room_reservation (extended)
Ý nghĩa	Tạo đơn đăng ký phòng
Mô tả	Điều kiện: Kế thừa các điều kiện ở chức năng gốc. Bổ sung thêm điều kiện kiểm tra trạng thái phòng phải là đang hoạt động. Xảy ra khi: Giống chức năng gốc. Thực hiện: Giống chức năng gốc

l) Cấp phòng – tinh chỉnh

US-06-extended	reserve (extended)
Ý nghĩa	Cấp phòng cho đơn đăng ký
Mô tả	Điều kiện: Kế thừa các điều kiện ở chức năng gốc. Bổ sung thêm điều kiện kiểm tra trạng thái phòng phải là đang hoạt động. Xảy ra khi: Giống chức năng gốc. Thực hiện: Giống chức năng gốc

3.2.4 Cây phân cấp máy (machines) cho bài toán đăng ký đặt phòng

Machine	Observations
m0	Các chức năng cơ bản

m1	Bổ sung thêm tính năng liên quan tới quản lý trạng thái phòng

3.2.5 Bảng các ràng buộc về trạng thái của hệ thống

#	Ràng buộc
req1	Các đơn đăng ký phòng họp được định danh duy nhất trong hệ thống
req2	Không tồn tại đơn đặt phòng nào mà thời gian kết thúc sớm hơn thời gian bắt đầu
req3	Không tồn tại nhiều đơn được cấp phòng bị trùng lịch
req4	Không tồn tại đơn sử dụng phòng đang tạm dừng hoạt động

3.3 Mô hình hoá hệ thống sử dụng Event-B với công cụ Rodin

Với các mô tả trên, trong phần này ta sẽ viết các mô tả hình thức cho hệ thống sử dụng phương pháp Event-B và công cụ Rodin.

3.3.1 Định nghĩa context ctx0

Trong ctx0, ta định nghĩa các sets như USERS – tập nền của người dùng; ROOMS – tập nền của các phòng; ROLES là một tập dạng liệt kê các phần tử mỗi phần tử là một vai trò trong của người dùng; tập STATES là tập các trạng thái của đơn đặt phòng. Ngoài ra, ctx0 cũng định nghĩ một hằng k_root, là đối tượng Admin khi hệ thống được khởi tạo.

```
context ctx0

sets USERS ROLES ROOMS STATES

constants k_root Admin Director Secretary Staff
          Requested Accepted Declined Reserved Denied


axioms
  @axm1 finite(ROOMS)
  @axm2 finite(USERS)

  @axm3 k_root ∈ USERS
  @axm4 partition(ROLES, {Admin}, {Director}, {Secretary}, {Staff})
  @axm5 partition(STATES, {Requested}, {Accepted}, {Declined}, {Reserved}, {Denied})

end
```

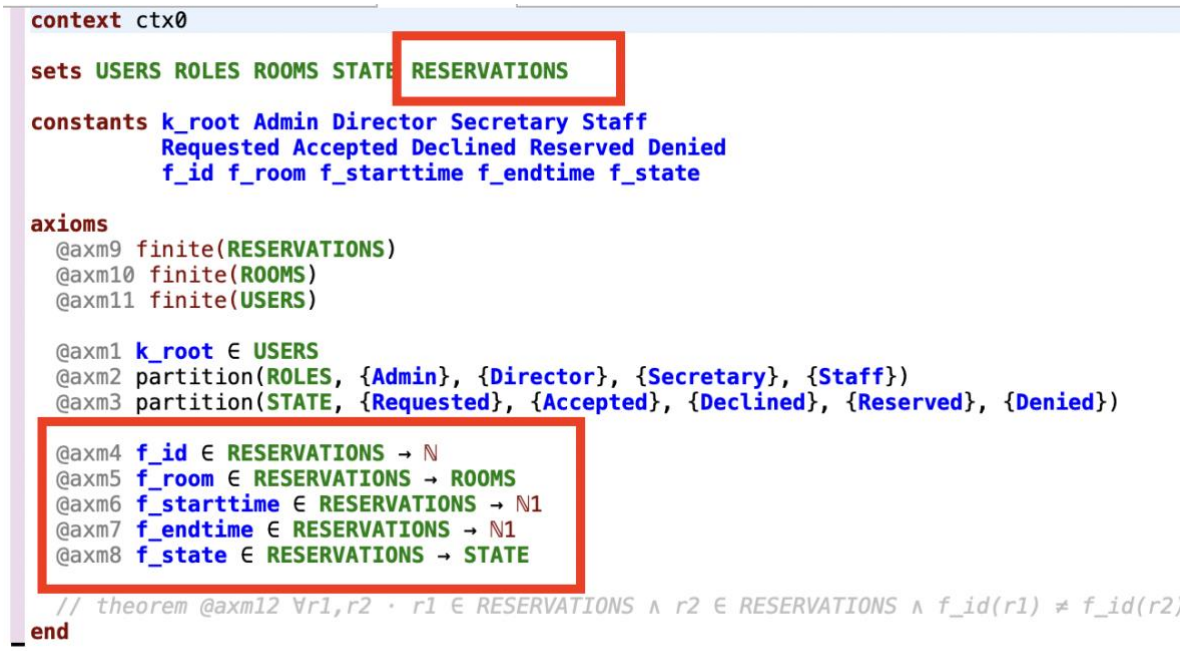
Hình 2: Định nghĩa context ctx0

3.3.2 Đĩnh nghĩa machine m0

3.3.2.1 Cấu trúc dữ liệu bản ghi cho các đơn đặt phòng

Trong phần này, vì mỗi đơn đặt phòng sẽ gồm có: rid - mã định danh; room – phòng đăng ký mượn; starttime – thời gian bắt đầu sử dụng; endtime – thời gian kết thúc; state – trạng thái đơn đăng ký, nên ta cần tìm cách để thể hiện cấu trúc dữ liệu dạng bản ghi này, bởi Event-B không hỗ trợ trực tiếp cấu trúc dữ liệu dạng này.

Trong phiên bản đầu tiên, dưới sự tham khảo từ trang web https://wiki.event-b.org/index.php/Structured_Types, em đã thực hiện tạo một kiểu dữ liệu là Reservation, kèm theo các hàm thuộc tính cho các trường của kiểu dữ liệu này:



Hình 3: Xử lý cấu trúc dữ liệu bản ghi (hướng 1)

Tuy nhiên cách làm này đã dẫn tới sự rườm rà khi xử lý các event, đồng thời em đã gặp khó khăn khi định nghĩa các invariants khi dùng cấu trúc này (công cụ Rodin không thể hỗ trợ chứng minh tự động tính well-defined (WD) của invariant):

```
event create_reservation
  any user new_reservation rid room start_time end_time
  where
    @grd1 user ∈ users
    @grd2 new_reservation ∈ RESERVATIONS
    @grd3 f_id(new_reservation) = rid
    @grd4 f_room(new_reservation) = room
    @grd5 f_starttime(new_reservation) = start_time
    @grd6 f_endtime(new_reservation) = end_time
    @grd7 f_state(new_reservation) = Requested
    @grd8 start_time < end_time
    @grd9 ∀r· r ∈ reservations ⇒  f_id(r) ≠  rid // not allow to same rid
  then
    @act1 reservations ≔ reservations ∪ {new_reservation}
end

event accept_reservation
  any caller reservation new_reservation
  where
    @grd1 caller ∈ users ∧ user_role(caller) = Director
    @grd2 reservation ∈ reservations ∧ f_state(reservation) = Requested
    @grd8 new_reservation ∈ RESERVATIONS
    @grd3 f_id(new_reservation) = f_id(reservation)
    @grd4 f_room(new_reservation) = f_room(reservation)
    @grd5 f_starttime(new_reservation) = f_starttime(reservation)
    @grd6 f_endtime(new_reservation) = f_endtime(reservation)
    @grd7 f_state(new_reservation) = Accepted
  then
    @act1 reservations ≔ (reservations \ {reservation}) ∪ {new_reservation}
end
```

Hình 4: Hạn chế của hướng xử lý 1

Điều này khiến em lựa chọn một cách tiếp cận khác, định nghĩa các hàm toàn phần (total functions) như các cơ chế để biểu diễn các trường dữ liệu của bản ghi như sau:

```
machine m0 sees ctx0

variables users user_role rooms rids f_room f_starttime f_endtime f_state

invariants
  @inv1 users ⊆ USERS
  @inv2 user_role ∈ users → ROLES
  @inv3 rooms ⊆ ROOMS
  @inv4 rids ⊆ ℕ1
  @inv5 f_room ∈ rids → rooms
  @inv6 f_starttime ∈ rids → ℕ1
  @inv7 f_endtime ∈ rids → ℕ1
  @inv8 f_state ∈ rids → STATES
  @inv9 ∀ rid· rid ∈ rids ⇒ f_starttime(rid) < f_endtime(rid)
  @inv10 ∀rid1,rid2 · rid1 ∈ rids ∧ rid2 ∈ rids ∧ rid1 ≠ rid2 ∧ f_state(rid1) = Reserved ∧ f
```

Hình 5: Xử lý cấu trúc dữ liệu bản ghi (hướng 2)

Điều này giúp việc xử lý các event diễn ra đơn giản hơn, và em đã thành công trong việc biểu diễn các invariants của bài toán đảm bảo tính WD.

3.3.2.2 Các invariants của máy m0

```
invariants
  @inv1 users ⊆ USERS
  @inv2 user_role ∈ users → ROLES
  @inv3 rooms ⊆ ROOMS
  @inv4 rids ⊆ ℕ1
  @inv5 f_room ∈ rids → rooms
  @inv6 f_starttime ∈ rids → ℕ1
  @inv7 f_endtime ∈ rids → ℕ1
  @inv8 f_state ∈ rids → STATES
  @inv9 ∀ rid· rid ∈ rids ⇒ f_starttime(rid) < f_endtime(rid)
  @inv10 ∀rid1,rid2 · rid1 ∈ rids ∧ rid2 ∈ rids ∧ rid1 ≠ rid2 ∧ f_state(rid1) = Reserved ∧ f_state(rid2) = Reserved
         ⇒ f_starttime(rid1) > f_endtime(rid2) ∨ f_endtime(rid1) < f_starttime(rid2)
```

Hình 6: Các invariants của máy m0

req1: Việc lưu trữ các định danh của các đơn dưới dạng tập hợp (@inv4), đồng thời các hàm truy cập giá trị các trường của bản ghi đều là các total functions đảm bảo tính duy nhất của các định danh.

req2: được thể hiện bởi @inv9

req3: được thể hiện bởi @inv10

req4: sẽ được thể hiện ở m1 trong bước tiếp theo

3.3.2.3 Các events của máy m0

a) Tính năng thêm người dùng vào hệ thống (US-01):

```
event add_user
  any caller new_user role
  where
    @grd1 caller ∈ users ∧ user_role(caller) = Admin
    @grd2 new_user ∈ USERS \ users
    @grd3 role ∈ ROLES
  then
    @act1 users ≔ users ∪ {new_user}
    @act2 user_role(new_user) ≔ role
end
```

Ở đây, trong bước kiểm ra điều kiện, ta tiến hành kiểm tra vai trò của người gọi, và ràng buộc đó phải là Admin. Ngoài ra, người dùng được thêm vào hệ thống phải là chưa tồn tại, đồng thời người dùng này phải được gán một vai trò hợp lệ trong danh sách vai trò có trong hệ thống. Khi đó, hành động @act1 sẽ thêm người dùng vào danh sách người dùng trong hệ thống, và @act2 sẽ lưu lại thông tin vai trò của người dùng này.

b) Tính năng thêm phòng (US-02):

```
event add_room
  any caller new_room
  where
    @grd1 caller ∈ dom(user_role ▷ {Admin})
    @grd2 new_room ∈ ROOMS \ rooms
  then
    @act1 rooms ≔ rooms ∪ {new_room}
end
```

Nội dung của chức năng này được định nghĩa một cách tương tự như phần trên.

c) Tính năng tạo đơn đăng ký phòng (US-03)

```
event create_reservation
 any caller rid starttime endtime room
 where
   @grd1 caller ∈ users
   @grd2 rid ∈ ℕ1
   @grd3 starttime ∈ ℕ1
   @grd4 endtime ∈ ℕ1
   @grd5 room ∈ rooms
   @grd6 rid ∉ rids
   @grd7 starttime < endtime
   // @grd8 state = Requested
 then
       @act1 rids ≔ rids ∪ {rid}
       @act2 f_room(rid) ≔ room
       @act3 f_starttime(rid) ≔ starttime
       @act4 f_endtime(rid) ≔ endtime
       @act5 f_state(rid) ≔ Requested
end
```

Trong event này, @grd1 kiểm tra người tạo đơn phải là người dùng trong hệ thống. các @grd2 đến @grd5 kiểm tra sự hợp lệ của kiểu dữ liệu. @grd6 đảm bảo rid không trùng lặp với các rid đã tồn tại. @grd7 đảm bảo thời gian bắt đầu phải nhỏ hơn thời gian kết thúc (nhằm thoản @inv9 được bảo toàn). Trong phần "then", ta thực hiện lưu trữ các dữ liệu của đơn mới này các các cấu trúc tương ứng, với trạng thái của đơn được đặt là Requested.

d) Tính năng duyệt đơn đăng ký (US 4-5)

```
event accept_reservation
any caller rid
where
  @grd1 caller ∈ users ∧ user_role(caller) = Director
  @grd2 rid ∈ rids
  @grd3 f_state(rid) = Requested
then
  @act1 f_state(rid) ≔  Accepted
end

event decline_reservation
  any caller rid
  where
    @grd1 caller ∈ users ∧ user_role(caller) = Director
    @grd2 rid ∈ rids
    @grd3 f_state(rid) = Requested
  then
    @act1 f_state(rid) ≔ Declined
end
```

Tính năng duyệt đơn đăng ký (Đồng ý/Từ chối) được định nghĩa như các event ở hình trên đây. Cụ thể, các guards sẽ kiểm tra quyền thực hiện của người gọi, sự tồn tại của đơn, trạng thái của đơn. Nếu đơn hợp lệ, ta chuyển trạng thái sang trạng thái mới tương ứng.

e) Tính năng cấp phòng (US 6)

```
event reserve
  any caller rid
  where
    @grd1 caller ∈ users ∧ user_role(caller) = Secretary
    @grd2 rid ∈ rids
    @grd3 f_state(rid) = Accepted
    @grd4 ∀rid1 · rid1 ∈ rids ∧ f_state(rid1) = Reserved
          ⇒ f_endtime(rid) < f_starttime(rid1) ∨ f_starttime(rid) > f_endtime(rid1)
  then
    @act1 f_state(rid) = Reserved
end
```

Trong phần này, đáng chú ý có @grd4 nhằm đảm bảo không có sự trùng lịch khi cấp phòng nhằm đảm bảo req3.

f) Tính năng từ chối cấp phòng (US 7)

```
event deny
  any caller rid
  where
    @grd1 caller ∈ users ∧ user_role(caller) = Secretary
    @grd2 rid ∈ rids
    @grd3 f_state(rid) = Accepted
  then
    @act1 f_state(rid) ≔ Denied
end
```

Tính năng này được xử lý tương tự như các tính năng duyệt đơn đăng ký đã đề cập.

3.3.3 Định nghĩa context cxt1

Trong phần context này, ta cần định nghĩa thêm hằng số … thể hiện thời gian hiện tại, nhằm mục đích kiểm tra điều kiện không có đơn nào đang sử dụng phòng khi Admin muốn tạm dừng phòng:

```
context ctx1
 constants k_current_timestamp
 axioms
  @axmr1 k_current_timestamp ∈ ℕ1
end
```

3.3.4 Định nghĩa machine m1

```
machine m1 refines m0 sees ctx0 ctx1
variables users user_role rooms rids f_room f_starttime f_endtime f_state inactive_rooms

invariants
  @invr1 inactive_rooms ⊆ rooms
  @invr2 ∀rid · rid∈rids ∧ f_state(rid)=Reserved
         ∧ f_starttime(rid) < k_current_timestamp ∧ f_endtime(rid) > k_current_timestamp
         ⇒ f_room(rid) ∉ inactive_rooms
```

Hình 7: Các invariants bổ sung của máy m1

Trong máy m1 này, ta bổ sung thêm biến inactive_rooms, nhằm lưu trữ thông tin các phòng đang dừng hoạt động. Ngoài ra, ta thêm @invr2 nhằm đáp ứng yêu cầu req4 của bài toán.

3.3.4.1 Các events của máy m1

a) Tính năng tạm dừng hoạt động của phòng (US 8)

```
event deactive_room
  any caller room
  where
    @grd1 caller ∈ dom(user_role ▷ {Admin})
    @grd2 room ∈ rooms
    @grd3 room ∉ inactive_rooms
    @grd4 ∀rid·rid ∈ rids ∧ f_state(rid) = Reserved
          ∧ f_starttime(rid) < k_current_timestamp ∧ f_endtime(rid) > k_current_timestamp
          ⇒ f_room(rid) ≠  room
  then
   @act1 inactive_rooms = inactive_rooms ∪ {room}
end
```

Ở event này, ta cần kiểm tra rằng không có đơn nào đang sử dụng phòng đang muốn tạm dừng thông qua @grd4. Nếu các điều kiện được thoả mãn, ta đưa phòng này vào danh sách phòng đang dừng hoạt động.

b) Tính năng mở hoạt động lại phòng (US 9)

```
event active_room
  any caller room
  where
    @grd1 caller ∈ dom(user_role ▷ {Admin})
    @grd2 room ∈ inactive_rooms
  then
    @act1 inactive_rooms ≔ inactive_rooms ∖ {room}
end
```

Tính năng này được thể hiện qua event như trên. Chú ý phần @act1 sẽ loại bỏ phòng này ra khỏi danh sách phòng đang dừng hoạt động.

Dưới sự tinh chỉnh của m1, một số event ở m0 cũng cần được cập nhật tương ứng như sau:

c) Tính năng tạo đơn đăng ký phòng (bản mở rộng) (US 3 extended)

```
event create_reservation extends create_reservation
  where
    @grdr1 room ∉  inactive_rooms
end
```

Nhận thấy event này là sự mở rộng của event của m0, với thêm điều kiện @grdr1 thể hiện phòng đăng ký không được trong trạng thái dừng hoạt động.

d) Tính năng cấp phòng (bản mở rộng) (US 6 extended)

```
event reserve extends reserve
  where
  @grdr1 f_room(rid) ∉ inactive_rooms
    // @grdr1 f_starttime(rid) < k_current_timestamp ∧ f_endtime(rid)
end
```

Tương tự như trên, event này cũng là sự mở rộng của event thuộc máy m0 nhằm bổ sung điều kiện kiểm tra trạng thái phòng.

3.4 Xây dựng chương trình

Ở phần này, em thực hiện xây dựng một chương trình web apis, với các chức năng và logic hoạt động như đã được mô hình hoá ở bước trên. Hệ thống web REST gồm có các APIs sau:

Method	Endpoint	Mô tả
POST	/api/users/login	login hệ thống
POST	/api/users	tạo user
GET	/api/users	lấy danh sách users
POST	/api/rooms	tạo phòng
GET	/api/rooms	lấy danh sách phòng

PATCH	/api/rooms/deactivate	tạm dừng hoạt động phòng
PATCH	/api/rooms/activate	mở lại hoạt động phòng
POST	/api/reservations	tạo đơn đăng ký phòng
GET	/api/reservations	lấy danh sách đơn đăng ký
PATCH	/api/reservations/accept	duyệt: đồng ý
PATCH	/api/reservations/decline	duyệt: từ chối
PATCH	/api/reservations/reserve	cấp phòng
PATCH	/api/reservations/deny	từ chối cấp phòng

Nhận xét: dựa trên cơ sở mô hình đã xây dựng, việc triển khai lập trình hệ thống diễn ra một cách rõ ràng, mạch lạc và có tính hệ thống hơn, từ đó giúp giảm thiểu tối đa các lỗi lập trình có thể mắc phải trong hệ thống.

Chương 4: Kết luận

4.1 Kết quả đạt được

Thông qua quá trình thực hiện đề tài, em đã tìm tòi và tiếp thu được nhiều kiến thức cũng như kỹ năng mới. Cụ thể, em đã biết đến sự tồn tại của các Phương pháp hình thức (Formal methods) và ý nghĩa của chúng trong việc phát triển các chương trình giúp hạn chế tối đa các lỗi hệ thống. Tiếp đó, em tìm hiểu chuyên sâu về một phương pháp nổi bật là Event-B thông qua các cuốn sách [1] [2] và bài báo [3] cũng như các tài nguyên trên mạng khác. Sau khi tìm hiểu và thực hành với Event-B, em đã tự áp dụng các kiến thức đã học vào giải quyết một bài toán thực tế: Đăng ký đặt lịch phòng. Thông qua việc thực hành xây dựng mô hình này, dù gặp phải nhiều khó khăn và va vấp, do khi xây dựng gặp nhiều vấn đề mà sách lý thuyết chưa đề cập, em đã tìm hiểu thêm các tài nguyên trên mạng để nhằm giải quyết chúng. Tuy nhiên, do thời gian có hạn và năng lực còn nhiều hạn chế, nên mô hình đã thực hiện chỉ đảm bảo các tính năng cơ bản vừa đủ đáp ứng cho các yêu cầu của hệ thống.

4.2 Hướng phát triển trong tương lai

Mô hình hiện tại đã đáp ứng các yêu cầu cơ bản của hệ thống, tuy nhiên, trong tai lai, em đề xuất hướng phát triển bổ sung thêm nhiều tính năng mới như: cho phép Secretary thực hiện điều chỉnh phòng và thời gian của các đơn đặt phòng (nhằm giúp tránh trùng lặp), cho phép người đăng ký có thể huỷ đơn đặt phòng… Việc bổ sung thêm tính năng mới này sẽ giúp nâng cao sự thành thạo sử dụng Event-B. Bên cạnh đó, một hướng phát triển nữa là thực hiện "data refinement" để giúp mô hình gần hơn với việc implement bởi các ngôn ngữ lập trình.

Chương 5: Tài liệu tham khảo

[1] R. Louis, 4 2020. [Online]. Available: https://medium.com/@ruwaid4/rodin-modelling-with-event-b-8fdab6c65003. [Accessed 11 2023].

[2] N. C. Collazos, Java Software Development with Event B, 2020.

[3] J.-R. Abrial, Modeling in Event-B: System and Software Engineering, 2006.

[4] T. S. Hoang, "An Introduction to the Event-B Modelling Method," Springer-Verlag, 2013.