TRƯỜNG ĐẠI HỌC ĐÀ LẠT
KHOA CÔNG NGHỆ THÔNG TIN

THỰC HÀNH
CÔNG NGHỆ PHẦN MỀM

(*Lưu hành nội bộ*)

MỤC LỤC

HƯỚNG DẪN CHUNG

Mỗi bài thực hành đều được thực hiện theo 4 công đoạn:

- Lập kế hoạch (Plan): Quyết định những công việc cần làm và liệu những việc đó có cần lặp lại.
- Tổ chức (Organize): Sắp xếp thứ tự công việc (có thể làm tuần tự hoặc song song) và phân công nhiệm vụ cho từng thành viên trong nhóm.
- Thực hiện (Do): Mỗi thành viên thực hiện công việc mình được giao.
- Suy ngẫm (Reflect): Suy nghĩ về những gì đạt được sau khi hoàn thành bài tập.

Đầu mỗi buổi thực hành, mọi thành viên trong nhóm đều phải đọc lướt qua toàn bộ bài tập để nắm được những yêu cầu cần phải thực hiện. Sau đó, mỗi nhóm bỏ ra khoảng 10 phút để lập kế hoạch và tổ chức thực hiện công việc.

LAB 1. MÔ HÌNH HÓA NGHIỆP VỤ

Trước khi bắt đầu xây dựng một hệ thống phần mềm, ta cần phải có những hiểu biết nhất định về nghiệp vụ sẽ sử dụng phần mềm đó. Đối với các nghiệp vụ lớn, ta chỉ có thể mô hình hóa một phần các lĩnh vực của nghiệp vụ đó, mà có thể được thực hiện trên phần mềm.

Sử dụng mô hình Use case, là cách tốt và khá đơn giản để có thể hiểu và ghi lại cách thức một nghiệp vụ vận hành như thế nào. Một mô hình Use case nghiệp vụ bao gồm một danh sách các actors, các use cases và một bảng chú thích.

Trong mô hình nghiệp vụ, cũng như trong bất kỳ giai đoạn lặp đi lặp lại khác, đều không quy định nghiêm ngặt về thứ tự làm việc. Vì vậy, bài lab này (và hầu hết những bài lab sau) được xem như là các bước trong một lần lặp đầu tiên. Đọc kỹ toàn bộ bài lab trước khi bắt đầu để có thể tùy ý sắp xếp lại các bước hoặc thực hiện chúng một cách song song.

Nếu gặp khó khăn về một use case nào đó, hãy thử sử dụng sơ đồ hoạt động hoặc sơ đồ giao tiếp để phác thảo ý tưởng.

Mục tiêu: Vào cuối buổi thực hành, bạn có thể:

- Tạo và duy trì một workbook cho dự án.
- Lựa chọn và phân công công việc cho các Member trong nhóm.
- Bắt đầu một bảng chú thích.
- Hiểu được các yêu cầu nghiệp vụ và hệ thống.
- Làm việc với khách hàng để nắm rõ công việc cụ thể.
- Xác định và mô tả các actor nghiệp vụ.
- Xác định và mô tả các use case nghiệp vụ.
- Thêm các chi tiết các use case.

1.1 Tạo một workbook cho dự án

Hãy bắt đầu bằng việc tạo một file mới và đặt tên nó là tên dự án của các bạn. Nó sẽ chứa tất cả những thông tin, mô tả, mô hình .v.v.. về dự án, do đó bạn có thể tra cứu cũng như cập nhật những chi tiết mới nhất về bất kỳ phần nào của hệ thống.

Trong quá trình làm việc, bạn có thể sao lưu, cập nhật hoặc tạo một phiên bản mới của tài liệu này. Đừng bỏ đi bất cứ tài liệu nào vì nó sẽ hữu ích cho việc bảo trì hệ thống sau này. Tốt nhất nên có một người trong nhóm làm nhiệm vụ “Workbook Keeper” để đảm bảo cho tài liệu được an toàn, và copy cho mỗi Member vào cuối dự án.

1.2 Phân công công việc

Thêm một bảng phân công công việc cho các Member của nhóm.

1.2.1 Xác định vai trò của dự án

Trước khi làm bước tiếp theo, nên xác định một số vai trò cho nhóm. Vì nhóm của phải làm việc với thời gian chặt chẽ, do đó phải tổ chức nhóm để tránh bị rối loạn (và chắc chắn rằng có một giải pháp cho mỗi bài tập).

Một số vai vai trò có thể xem xét là:

- Người quản lý dự án (Project manager) đảm bảo:
  - Lập kế hoạch và lịch trình cho dự án.
  - Biết rõ trách nhiệm của bản thân.
  - Thường xuyên xem xét tiến độ của công việc.
- Người quản lý thời gian (Time Keeper): Chắc chắn các công việc được tiến hành đúng lịch trình.
- Chuyên viên (Guru): Những người chuyên về một lĩnh vự cụ thể của dự án.
- Thư ký (Scribe): Ghi nhận lại các bước tiến hành của từng công việc.
- Người chú thích (Glossary Keeper): Có trách nhiệm cập nhật, bổ sung các bác chú thích.

1.2.2 Phân công việc

Phân công công việc cho các Member của nhóm dự vào các vài trò đã xác định được ở bước 1.2.1, có thể luân chuyển công việc giữa các Member trong nhóm để ai cũng có thể hiểu mỗi vai trò cần phải làm những gì.

1.3 Thêm workbook cho mô hình kinh doanh

Thêm phần ‘mô hình kinh doanh” cho workbook để nắm được kết quả bài tập, ngoại trừ danh sách thuật ngữ.

Khi thêm mục mới vào workbook, trừ lại phần trống cho mỗi mục. Điều này cho phép chỉnh sửa hoặc thêm ghi chú vào mục cá nhân trong khi lặp, mà không tạo phiên bản mới của tài liệu. Ví dụ, để lại nửa trang cho mỗi actor, cả 2 bên cho mỗi use case v.v...

1.4 Duy trì danh sách thuật ngữ

Thêm mục danh sách thuật ngữ cho workbook

1.4.1 Thêm thuật ngữ và từ đồng nghĩa mới

Mỗi khi mẫu thuật ngữ kinh doanh cụ thể xuất hiện, thêm mô tả từ viết tắt trong danh mục thuật ngữ. Tương tự, khi xuất hiện từ đồng nghĩa cũng thêm thuật ngữ "tùy chọn".

1.4.2 Tài liệu tham khảo mô hình kinh doanh

Khi thêm mục vào trong danh sách thuật ngữ có liên quan trực tiếp với mô hình kinh doanh, thực hiện ghi chú cho mục này.

Khách hàng (business actor, business object) bất cứ người nào thuê xe, xem mẫu xe hoặc đặt trước mẫu xe.

1.4.3 Xem xét vấn đề báo cáo

Khách hàng (khoa Khoa học máy tính tại trường đại học Blue sky) đã đưa báo cáo trong mẫu tài liệu được thảo luận một cách hình thức (xem phụ lục A). Đọc tài liệu thảo luận và chắc chắn là hiểu vấn đề.

Như việc xử lý bài tập, nhớ rằng bạn không phải là khách hàng. Do vậy không đưa ra giả định về việc kinh doanh của khách hàng mà thay vào đó nên hỏi họ.

1.5 Quy trình tạo danh sách actor

Ghi xuống tất cả con người hoặc vai trò hệ thống con trong hệ thống đánh giá sinh viên của bộ phận kinh doanh. Nhớ tập trung vào hệ thống kinh doanh hiện có, không phải cách kinh doanh sẽ hoạt động như thế nào khi xây dựng hệ thống AQS mới. Nếu cần thiết, giảm danh sách ứng cử viên actor nhiều nhất có thể (bất kỳ actor nào loại bỏ vẫn đồng nghĩa trong danh sách thuật ngữ). Thêm mô tả ngắn (chỉ 1 câu) cho mỗi actor.

1.6 Quy trình tạo danh sách use case

Liệt kê tất cả các use case kinh doanh cho việc doanh nghiệp đánh giá sinh viên. (mỗi use case nên bắt đầu với 1 actor và cung cấp giá trị cho 1 hoặc nhiều actor). Rút gọn danh sách use case thành tập tối thiểu hoàn chỉnh, loại bỏ những cái không sử dụng. Thêm mô tả ngắn cho mỗi use case để tham khảo nhanh.

1.7 Thêm chi tiết cho use case

Mặc dù các use case có thể được mô tả sử dụng 1 hoặc 2 đoạn bằng ngôn ngữ tự nhiên, nó thường rõ ràng hơn và ít mơ hồ để cung cấp chuỗi các bước thay thế. Cụ thể mỗi use case với danh sách các bước (khoảng 10 bước là đủ).

1.8 Minh họa

1.8.1 Yêu cầu nghiệp vụ

Phần này làm các mô hình yêu cầu nghiệp vụ sẽ thực hiện trong giai đoạn yêu cầu của sự phát triển iCoot, sử dụng các mô hình Use case nghiệp vụ. Mô hình use case nghiệp vụ cũng áp dụng cho hệ thống Coot đầy đủ.

1.8.2 Sứ mệnh của khách hàng

Dưới đây là những nhiệm vụ cung cấp bởi Nowhere Cars khi đầu của dự án Coot:

Vì chúng ta theo dõi ô tô tự động tại các cửa hàng - sử dụng mã vạch, counter-top, các thiết bị đầu cuối và máy đọc laser - chúng tôi đã thấy nhiều lợi ích: năng suất cho thuê của chúng tôi đã tăng lên 20%, xe ít khi bị mất và khách hàng của chúng tôi lien tục tăng thêm (theo nghiên cứu thị trường của chúng tôi, điều này một phần là do nhận thức về tính chuyên nghiệp và hiệu quả được cải thiện).

Nhà quản lý cảm thấy rằng Internet cung cấp các cơ hội thú vị cho việc tăng hiệu quả và giảm bớt chi phí. Ví dụ, thay vì in ấn catalog xe có sẵn, chúng tôi có thể làm catalog online cho phép người dung Internet có thể xem trực tuyến. Đối với khách hàng đặc quyền, chúng tôi có thể cung cấp thêm các dịch vụ, chẳng hạn như đặt chỗ, chỉ với một nút bấm. Việc này giúp chúng tôi tiết kiệm 15% chi phí trong mỗi cửa hàng.

Trong vòng hai năm, sử dụng toàn bộ sức mạnh của thương mại điện tử, chúng tôi hướng đến việc cung cấp tất cả các dịch vụ của chúng tôi thông qua trình duyệt Web, việc giao hang tận nhà của khách hang giúp chi phí giảm đi tối thiểu so với việc tới trực tiếp cửa hàng.

1.8.3 Danh sách actors

- Assistant: Nhân viên tư vấn khách hàng đến thuê xe hoặc đặt mẫu xe.
- Customer: Người trả tiền để mua một trong những dịch vụ tiêu chuẩn.
- Member: Khách hàng thân thiết và được tín nhiệm xứng đáng được hưởng các dịch vụ đặc biệt (chẳng hạn như đặt lịch hẹn qua điện thoại hoặc qua Internet).
- NonMember: Khách hàng có danh tính chưa được xác thực phải ký quỹ để đặt chỗ và cung cấp một bản sao của Giấy phép thuê xe.
- Auk: Các hệ thống hiện có để xử lý các chi tiết khách hàng, đặt phòng, thuê xe và Catalog của các mẫu xe sẵn có.
- DebtDepartment: Bộ phận giao dịch với các khoản chi phí chưa thanh toán.
- LegalDepartment: Bộ phận giao dịch với các vụ tai nạn có liên quan đến việc thuê xe.

1.8.4 Danh sách Use case

- B1: Customer Rents Car (khách hàng thuê xe): khách hàng thuê một chiếc xe mà họ đã lựa chọn từ những mẫu xe có sẵn.
- B2: Member Reserves CarModel: Member yêu cầu được thông báo khi một mẫu xe có mặt.
- B3: NonMember Reserves CarModel: người trả tiền cọc để được thông báo khi một mẫu xe có mặt.
- B4: Customer Cancels Reservation: Khách hàng đích thân hủy một yêu cầu đặt xe trực tiếp hoặc qua điện thoại.
- B5: Customer Returns Car: Khách hàng trả lại xe mà họ đã thuê.
- B6: Customer Told CarModel Is Available: khách hàng được liên lạc bởi một Assistant khi một mẫu xe có tại của hàng.
- B7: Car Reported Missing: Khách hàng hoặc Assistant phát hiện ra rằng một chiếc xe đã bị mất.
- B8: Customer Renews Reservation: Khách hàng gia hạn một lịch thuê xe mà đã vượt quá một tuần.
- B9: Customer Accesses Catalog: khách hàng duyệt các danh mục, trong cửa hàng hoặc ở nhà.
- B10: Customer Fined for Uncollected Reservation: Khách hàng không đến thuê xe mà họ đã đặt.
- B11: Customer Collects Reserved Car: Khách hàng đến nhận xe mà họ đã đặt.
- B12: Customer Becomes a Member: Khách hàng cung cấp thông tin chi tiết Thẻ Tín Dụng và địa chỉ để trở thành một Member.
- B13: Customer Notified Car Is Overdue: Assistant liên lạc theo địa chỉ của khách hàng để cảnh báo họ rằng chiếc xe họ thuê đã quá hạn hơn một tuần.
- B14: Customer Loses Keys: cung cấp lại cho khách khi mất chìa khóa.
- B15: MembershipCard Is Renewed: Assistant liên lạc với Member để đổi mới thẻ Member khi thẻ tín dụng của họ đã hết hạn.
- B16: Car Is Unreturnable: Xe bị đắm hoặc bị phá vỡ.

1.8.5 Sơ đồ giao tiếp của use case

Sơ đồ giao tiếp không được dùng rộng rãi trong suốt mô hình yêu cầu kinh doanh (mặc dù chúng được bao quát trong quá trình thu thập yêu cầu hệ thống). Tuy nhiên, 1 sơ đồ (xem hình B.1) được tạo ra để minh họa actor bên trong và bên ngoài B3: NonMember Reserves CarModel.

1.8.6 Sơ đồ hoạt động của use case

Sơ đồ hoạt động không sử dụng để mở rộng trong suốt mô hình yêu cầu kinh doanh. Tuy nhiên, 1 sơ đồ (xem hình B.2) được tạo ra để minh họa hướng mịn hơn của use case B3: NonMember Reserves CarModel.

1.8.7 Chi tiết use case

B1: Customer Rents Car (khách hàng thuê xe)

1. Khách hàng hỏi người bán hàng (Assistant) về CarModel mà họ muốn thuê.
2. Nếu Auk chỉ rõ không có xe thì đề nghị lựa chọn khác cho khách hàng.
3. Nếu có xe, người bán hàng đánh dấu xe như lấy từ Auk.
4. Người bán hàng hỏi giấy phép của khách hàng để xác nhận danh tính.
5. Đối với mỗi Member, người bán hàng lấy số từ MembershipCard và kiểm tra họ không có lệ phí chưa thanh toán và không bị cấm.
6. Đối với NonMember, người bán hàng kiểm tra liệu họ đã có trong Auk chưa; nếu chưa, người bán hàng sẽ scan 1 bản copy giấy phép tới Auk; lưu lại tên, số điện thoại và số giấy phép của họ.
7. Nếu thông tin của khách hàng thỏa mãn và họ có khả năng trả phí thanh toán thì họ được tính tiền thuê.
8. Nếu việc thanh toán lỗi, xe sẽ bị loại bỏ khỏi Auk.
9. Nếu thanh toán thành công, chìa khóa xe được đưa cho khách hàng và hướng dẫn đến vùng hiển thị.

B2: Member Reserves CarModel

1. Member báo số thành viên cho người bán hàng (qua số điện thoại hoặc thông tin cá nhân).
2. Member báo cho người bán hàng về CarModel muốn đặt chỗ.
3. Nếu Member không bị chặn, thẻ tín dụng của họ không hết hạn và họ không có phí thanh toán, thì Reservation được tạo trên Auk.
4. Nếu Reservation được tạo trên phone, Member có thể trả phí thanh toán bằng cách xác nhận chi tiết thẻ tín dụng của họ phải khớp với chi tiết đã lưu trên Auk và không quá hạn.
5. Member có được số đặt chỗ.

B3: NonMember Reserves CarModel

1. NonMember hỏi người bán hàng về CarModel để đặt chỗ.
2. Người bán hàng tìm CarModel trên Auk
3. Người bán hàng yêu cầu đặt cọc tiền cho việc đặt chỗ.
4. Người bán hàng yêu cầu đăng ký và số điện thoại của NonMember.
5. Người bán hàng kiểm tra lại đăng ký.

6. Nếu đăng ký có giá trị, người bán hàng tạo thông tin đặt chỗ mới, lưu lại số đăng ký, điện thoại và scan đăng ký trên Auk.
7. Người bán hàng đưa cho NonMember phiếu đặt chỗ chứa số ID.

B4: Customer Cancels Reservation (Khách hàng hủy bỏ đặt chỗ)

1. Bất cứ khi nào, khách hàng đều có thể hủy bỏ đặt chỗ.
2. Member có thể làm điều đó qua điện thoại hoặc qua con người bằng cách cung cấp số Member của họ.
3. NonMember có thể hủy bỏ trực tiếp: họ đưa đăng ký cho người bán hàng, người bán hàng sẽ kiểm tra khớp với bản scan trên Auk và trả lại tiền đặt cọc cho họ.
4. Nếu xe thực sự đã xóa khỏi danh sách đặt chỗ và di chuyển trở về danh sách hiển thị.

B5: Customer Returns Car (Khách hàng trả xe)

1. Khi xe được trả về vùng kiểm tra, người bán hàng scan mã để xác nhận trả lại và kiểm tra thùng xăng đầy.
2. Xe được trả về trong danh sách hiện thị bởi người bán hàng.
3. Nếu khách hàng trả xe quá hạn hoặc xe không đầy xăng, khách hàng phải trả phí phù hợp – Member có thể trả phí thông qua thông tin thẻ tín dụng nếu chưa hết hạn.
4. Nếu khách hàng từ chối trả, thông tin của họ sẽ chuyển qua DebtDeparment.

B6: Customer Told Car Model is Available (Thông báo cho khách hàng CarModel có sẵn)

1. Khi xe trả về, Auk yêu cầu người bán hàng kiểm tra liệu nó khớp với bất kỳ đối tượng đặt chỗ nào không.
2. Nếu có, người bán hàng di chuyển xe sang vùng đặt chỗ.
3. Dựa trên nguyên tắc ai đến trước phục vụ trước, người bán hàng sẽ liên hệ với khách hàng khớp qua điện thoại.
4. Nếu khách hàng không hồi đáp lại trong vòng 2 ngày, thì việc đặt chỗ của họ sẽ hủy bỏ và xe được di chuyển từ vùng đặt chỗ sang vùng hiện thị.

B7: Car Reported Missing (Báo mất xe)

1. Nếu xe mà Auk chỉ ra trong vùng hiển thị không được tìm thấy khi cần hoặc trong suốt quá trình kiểm tra tài sản, xe được báo mất cho cảnh sát.
2. Nếu xe được báo mất do khách hàng, nó sẽ được báo mất tới cảnh sát, cùng với chi tiết giấy phép của khách hàng (người cuối cùng giữ xe).
3. Trong cả 2 trường hợp, ngày mất sẽ được lưu lại trên Auk.

B8: Customer Renews Reservation (Khách hàng cập nhật đặt chỗ)

1. Nếu đặt chỗ không được đáp ứng trong vòng 7 ngày thì cần đặt chỗ lại.

2. Người bán hàng liên hệ với khách hàng qua điện thoại trong 2 ngày để xác nhận liệu họ muốn đặt chỗ lại hơn 7 ngày không.
3. Nếu khách hàng không muốn đặt chỗ lại thì thông tin đặt chỗ bị hủy, khách hàng phải trở lại cửa hàng để trình giấy phép đăng ký để nhận lại tiền đặt cọc.

B9: Customer Accesses Catalog (Khách hàng truy cập vào danh mục)

1. Khách hàng có thể tới cửa hàng để xem bản danh mục.
2. Khách hàng có thể đóng phí để lấy bản copy danh mục về nhà.
3. Nếu khách hàng chọn cách gửi mail thì sẽ nhận được bản copy danh mục miễn phí hàng tháng qua mail.

B10: Customer Fined for Uncollected Reservation (Khách hàng bị phạt vì đặt chỗ không thành do lỗi khách hàng)

1. Nếu CarModel tồn tại cho việc đặt chỗ riêng và người bán hàng thông báo với khách hàng qua điện thoại đã có CarModel, khách hàng có 2 ngày để đến lấy.
2. Nếu khách hàng không đến lấy, việc đặt chỗ chấm dứt và người bán hàng sẽ chuyển xe từ vùng đặt chỗ về vùng hiển thị.
3. Đối với NonMember, tiền đặt cọc của họ sẽ bị mất.
4. Đối với Member, tiền phạt bị ghi lại trên Auk và chi tiết của họ được đưa qua DebtDepartment.

B11: Customer Collects Reserved Car (Khách hàng nhận xe đã đặt chỗ)

1. Khách hàng tới cửa hàng để lấy xe từ vùng đặt chỗ.
2. Khách hàng trình giấy phép đăng ký.
3. Nếu giấy phép khớp với chi tiết trên Auk, việc đặt chỗ được đánh dấu kết thúc.
4. Người bán hàng đưa chìa khóa cho khách hàng và hướng dẫn họ tới vùng đặt chỗ.

B12: Customer Becomes Member (Khách hàng trở thành Member)

1. Để trở thành Member, khách hàng phải cung cấp giấy phép đăng ký, địa chỉ và thẻ tín dụng
2. Người bán hàng kiểm tra giấy phép đăng ký và địa chỉ.
3. Người bán hàng kiểm tra thẻ tín dụng với công ty tín dụng.
4. Nếu OK, người bán hàng ghi lại thông tin số đăng ký, địa chỉ, số điện thoại và chi tiết thẻ tín dụng trên Auk.
5. Auk đưa ra thẻ Member mới với số ID.
6. Nếu thẻ tín dụng hết hạn, mọi hoạt động của Member sẽ bị giới hạn trừ khi Member trở lại cửa hàng để đưa ra thẻ tín dụng mới.

B13: Customer Notified Car is Overdue (Khách hàng trả xe quá hạn)

1. Vì tiền thuê trả trước, khách hàng được cảnh báo nếu họ quên trả xe.

2. Nếu xe quá hạn trên 1 tuần, người bán hàng sẽ cố gắng liên hệ với khách hàng qua điện thoại.
3. Nếu không liên hệ được với khách hàng trong 2 tuần, xe được báo mất (B7).

B14: Customer Loses Keys (Khách hàng mất chìa khóa)

1. Nếu khách hàng báo mất chìa khóa với người bán hàng, chìa khóa thay thế sẽ được người giao hàng đưa tới cho khách hàng (nếu cần thiết).
2. Chi phí chìa khóa thay thế được thêm vào chi tiết của khách hàng trên Auk.

B15: MembershipCard is Renewed (Thay mới thẻ thành viên).

1. Auk ghi lại Member có thẻ tín dụng hết hạn ở trạng thái không tốt.
2. Auk báo cho người bán hàng biết thẻ tín dụng của Member hết hạn.
3. Người bán hàng liên hệ với Member qua điện thoại để nói cho họ biết cần phải làm lại thẻ Member của họ.
4. Member trở lại cửa hàng với thẻ tín dụng mới và chi tiết sẽ được đưa vào Auk.
5. Auk ghi lại Member với trạng thái tốt.

B16: Car is Unreturnable (Xe không thể trả lại)

1. Nếu khách hàng báo với người bán hàng xe bị phá vỡ hoặc phá hỏng, người bán hàng sắp xếp phục hồi.
2. Nếu xe bị đắm, chi tiết được đưa vào LegalDepartment.

LAB 2. XÁC ĐỊNH YÊU CẦU HỆ THỐNG

Mô hình hóa Use Case ở dạng đầy đủ của nó là một cách tốt nhất để mô tả các yêu cầu chức năng của một hệ thống phần mềm. Các Use Case cũng cung cấp một vị trí thuận tiện để ghi lại hầu hết các yêu cầu phi chức năng.

Trong bài tập này, chúng ta sẽ xây dựng một mô hình Use Case hoàn chỉnh cho các hệ thống câu hỏi tự động bao gồm các Actor với các miểu tả, Use Case với các mô tả và thông tin chi tiết và một sơ đồ Use Case. Ghi lại các yêu cầu bổ sung không phù hợp với các Use Case cụ thể, đưa ra một số bản phác thảo của các thành phần giao diện người dùng.

Không thực hiện các giả định về những gì khách hang mong muốn ngoài hệ thống. (Điều này áp dụng cho các yêu cầu không có chức năng cũng như các yêu cầu chức năng.)

Để đơn giản, chúng ta có thể giả định rằng các giao diện sau đây sẽ được mô hình hóa:

- Công cụ cho việc tạo và chỉnh sửa các câu hỏi.
- Công cụ để thiết lập một buổi hợp.
- Công cụ để đối chiếu kết quả của buổi hợp.

Nói cách khác, chúng ta chỉ cần mô hình chi tiết các chức năng của các công cụ mà sinh viên nhìn thấy khi thực hiện bài kiểm tra. (Có thể tham khảo các cơ sở của các công cụ khác ở mức độ cao, để thêm các ngữ cảnh - ví dụ, có thể đưa chúng trong danh sách Actor, danh sách Use Case, sơ đồ Use Case và độ ưu tiên Use Case, nhưng không phải trong chi tiết Use Case hoặc phác thảo giao diện người dùng).

Để tiết kiệm thời gian, chúng ta không cần phải thực hiện khảo sát Use Case.

Mục tiêu: Cuối bài tập này, bạn sẽ học được các kiến thức:

- Xác định các Actor hệ thống.
- Xác định các Use Case hệ thống.
- Vẽ sơ đồ Use Case.
- Vẽ Use Case chi tiết hệ thống.
- Ghi các yêu cầu bổ sung.
- Đưa ra các bản phác thảo giao diện người dùng.
- Độ ưu tiên các Use Case.

2.1 Thêm workbook cho yêu cầu hệ thống

Thêm “mô hình Use Case hệ thống” và “Phác thảo giao diện người dùng” cho workbook của bạn

2.2 Xác định các Actor hệ thống

- Tạo danh sách các ứng cử viên cho Actor AQS.
- Viết mô tả ngắn của mỗi Actor AQS

2.3 Xác định các Use Case hệ thống

- Quyết định Use Case AQS phải hỗ trợ.
- Cung cấp mô tả ngắn về từng Use Case

2.4 Biểu diễn hệ thống bằng sơ đồ Use Case

Vẽ sơ đồ hiển thị tất cả các Actor AQS ngoài hệ thống (hiển thị như một hình chữ nhật) và tất cả các Use Case bên trong. Kết nối các Actor với các Use Case mà chúng tham gia.

Chúng ta có thể lưu vết các trường hợp cho các quan hệ 'include', 'extend' hoặc 'specialize' giữa các Use Case của bạn không? Chúng ta có thể xây dựng mối quan hệ bằng cách đưa ra một vài Use Case mới?

Nếu chúng ta có thể nhận thấy trường hợp lý tưởng để đưa ra Use Case mới hoặc các mối quan hệ, hãy thực hiện ngay. Nhưng đừng cố gắng đưa ra quá phức tạp vì lợi ích của việc tái sử dụng.

Chúng ta có thể nghĩ về các quan hệ 'specialize' giữa các Actor (đưa ra những cái mới cho mục đích này)? Đưa ra có chọn lọc các Actor và các mối quan hệ mới.

2.5 Chi tiết hóa Use Case

Đối với mỗi Use Case chuyên biệt (specialize) cho một Use Case khác, ghi rõ ở đầu các Use Case.

- Thêm các tiền điều kiện hoặc hậu điều kiện mà bạn có thể nghĩ đến. Nếu không có tiền điều kiện, chỉ cần viết 'Không'. Vì mỗi Use Case có nghĩa vụ phải cung cấp giá trị cho một hoặc nhiều Actor, luôn luôn phải có hậu điều kiện dọc theo dòng của “giá trị được cung cấp” hoặc không thành công.
- Thêm các bước cá nhân trong việc thực hiện các Use Case, chỉ ra nơi Use Case khác được bao gồm hoặc có thể được sử dụng để mở rộng Use Case hiện tại.
- Thêm các đường dẫn khác vào một danh sách ở cuối các Use Case.
- Bất cứ khi nào một yêu cầu phi chức năng phù hợp với một Use Case, thêm nó vào cuối các Use Case.

2.6 Ghi yêu cầu bổ sung

Trong workbook, thêm các yêu cầu phi chức năng mà không liên quan đến một Use Case cụ thể vào một trang 'Yêu cầu bổ sung' trong phần 'Yêu cầu hệ thống'.

2.7 Tạo giao diện người dùng

Đối với mỗi giao diện người dùng trong hệ thống, phác thảo trong cửa sổ chính. Có một giao diện người dùng cho mỗi Actor lấy thông tin từ hệ thống, thêm thông tin mới hoặc hướng hệ thống làm một nhiệm vụ.

Hầu hết các giao diện người dùng có các cửa sổ phụ xuất hiện khi cần thiết (họ có thể bật lên hoặc họ có thể che phủ các cửa sổ chính). Xác định các cửa sổ phụ cho giao diện người dùng sẽ xuất hiện.

Đối với mỗi giao diện người dùng, minh họa Use Case bằng cách hiển thị thứ tự mà các cửa sổ chính và cửa sổ phụ xuất hiện trong một tương tác thông thường.

2.8 Độ ưu tiên các yêu cầu

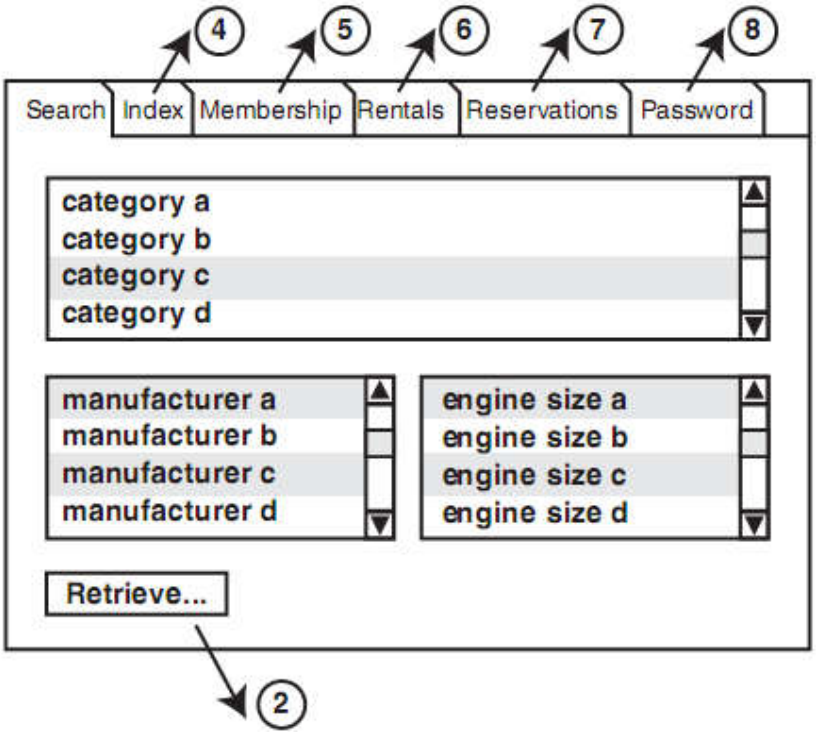
Đặt Use Case trong thứ tự ưu tiên thực hiện. Sử dụng màu xanh lá cây và màu đỏ để đánh dấu mong muốn của mỗi Use Case, và các yêu cầu bổ sung, cho việc phát hành phên bản đầu tiên.

2.9 Thực hành

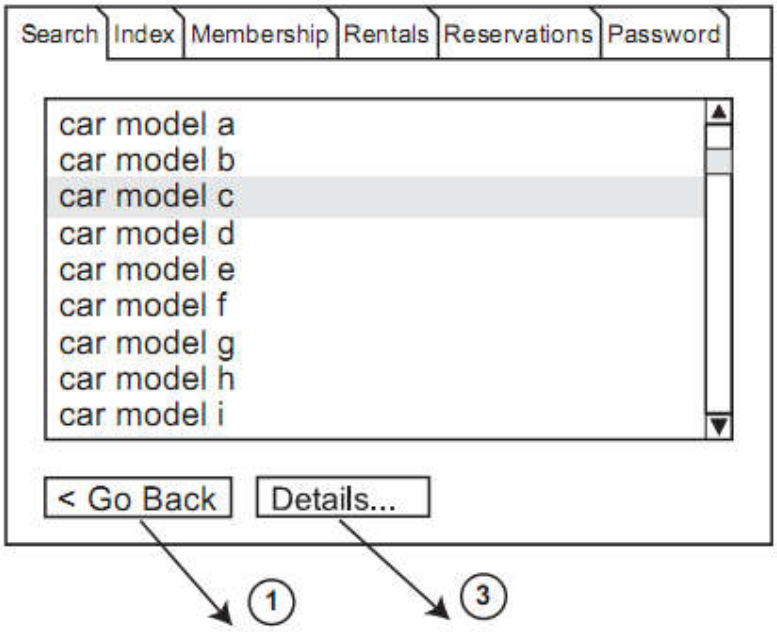
Phần này làm rõ kết quả của mô hình hệ thống trong các yêu cầu giai đoạn của sự phát triển iCoot, về bản phác thảo giao diện người dùng và mô hình Use Case hệ thống.

2.9.1 Giao diện người dùng

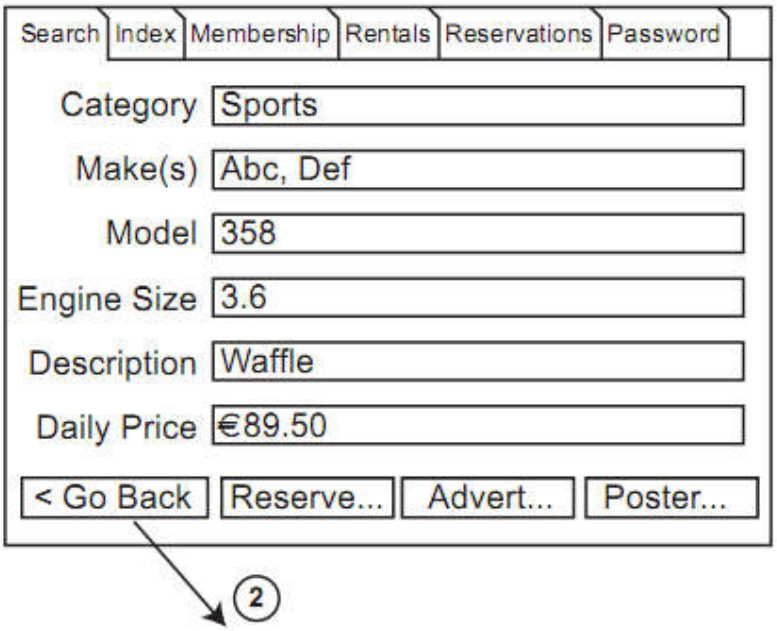
Các phác thảo giao diện người dùng cho iCoot, được đưa ra với sự giúp đỡ của khách hàng, được thể hiện trong hình B.3 đến B.10.



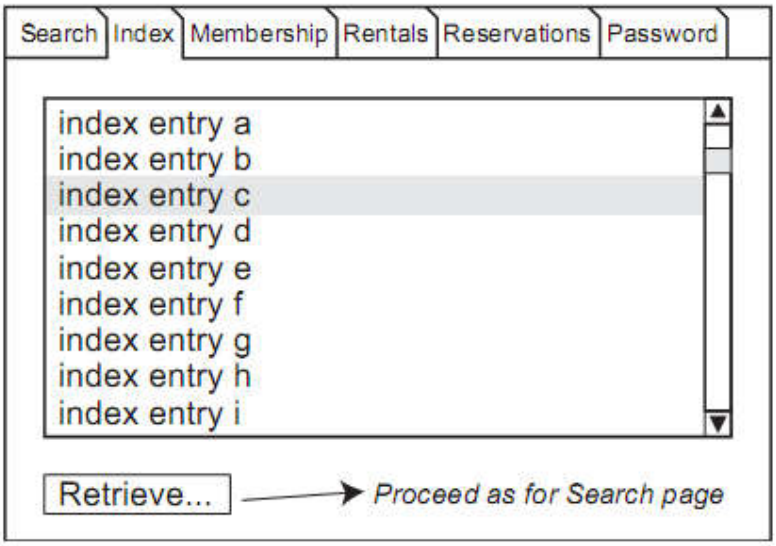
Hình B.3: Giao diện người dùng 1 (Truy vấn)



Hình B.4: Giao diện người dùng 2 (Kết quả tìm kiếm)

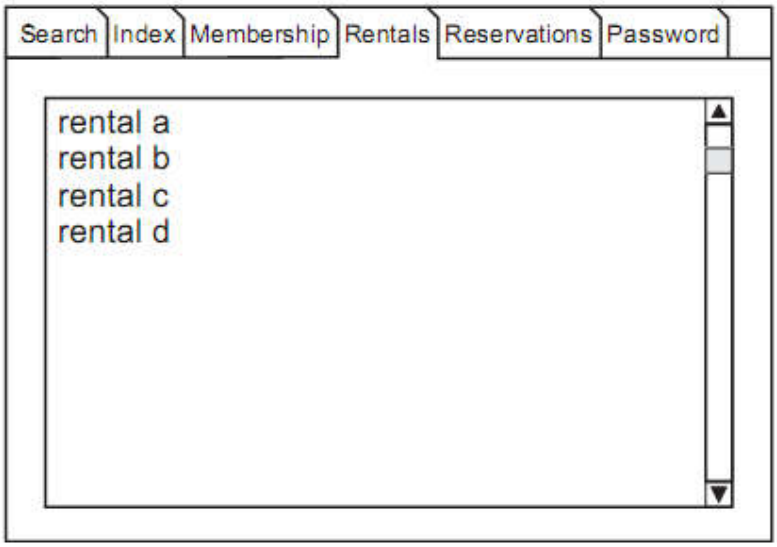


Hình B.5: Giao diện người dùng 3 (Chi tiết mô hình xe)

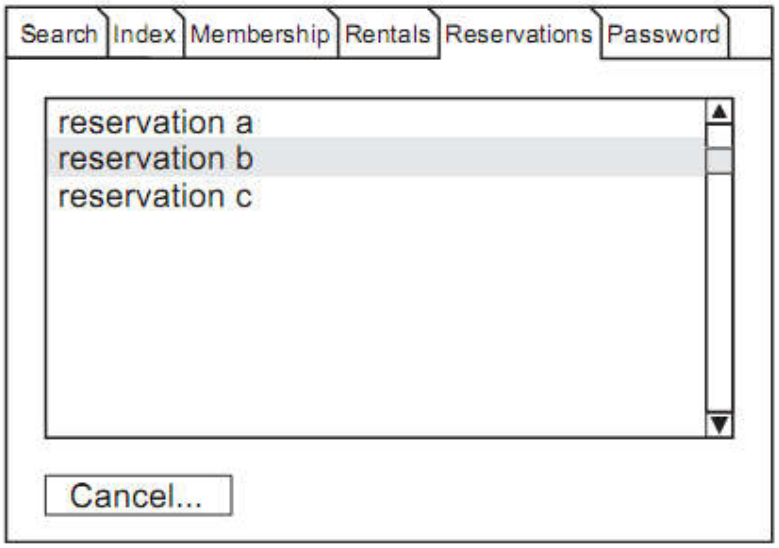


Hình B.6: Giao diện người dùng 4 (Lựa chọn index)

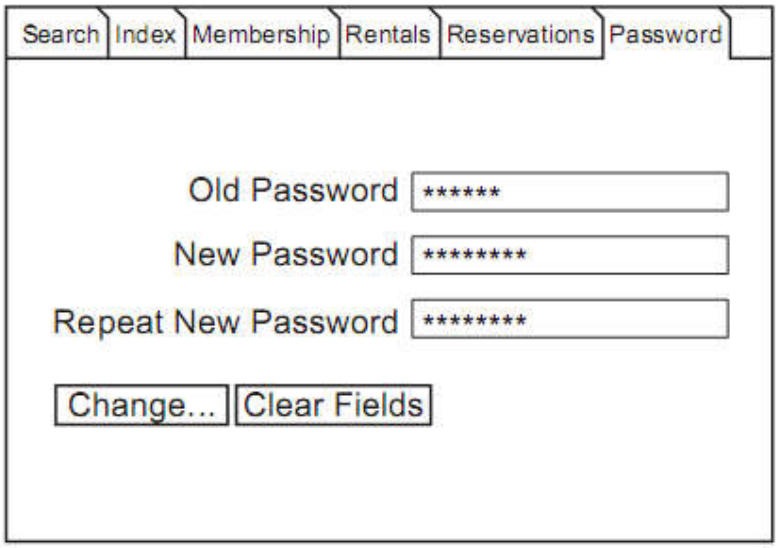
Hình B.7: Giao diện người dùng 5 (Chi tiết thành viên)



Hình B.8: Giao diện người dùng 6 (Thuê xe)



Hình B.9: Giao diện người dùng 7 (Đăng ký)



Hình B.10: Giao diện người dùng 8 (thay đổi mật khẩu)

2.9.2 Danh sách Actor

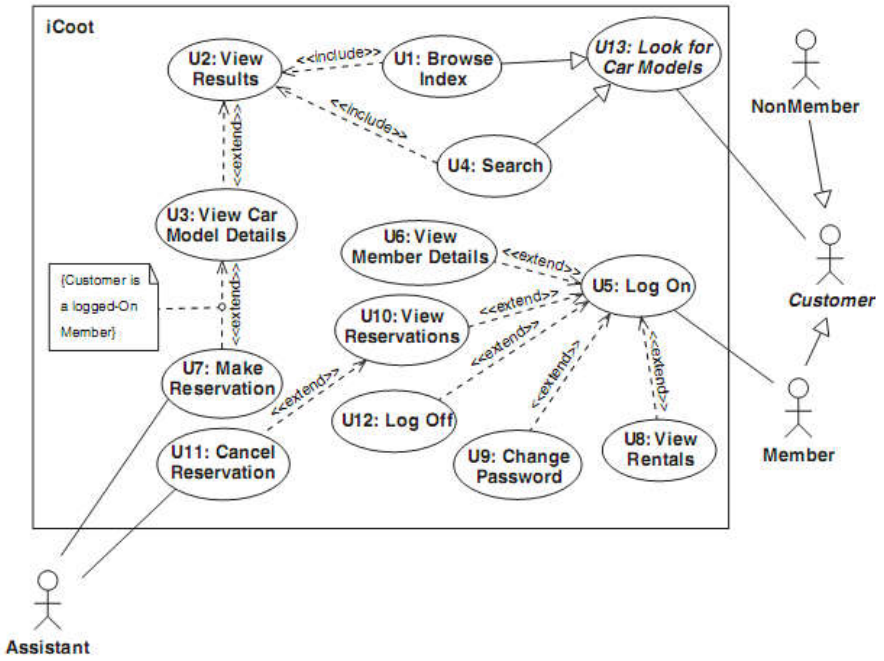
- Customer: Một người sử dụng trình duyệt web để truy cập iCoot.
- Member: Một khách hàng đã được ghi tên, địa chỉ và thẻ tín dụng tại một trong những cửa hàng; mỗi thành viên được tạo một mật khẩu Internet cùng với số thành viên của họ.
- NonMember: Một khách hàng không phải là một thành viên.
- Assistant: Một nhân viên tại một cửa hàng người liên lạc về tiến độ đăng ký.

2.9.3 Danh sách Use Case

- U1: Browse Index: Một khách hàng duyệt các CarModels chỉ mục. (Specializes U13, bao gồm U2.)
- U2: View Results: Một khách hàng được hiển thị các CarModels đã được đăng ký. (bao gồm U1 và U4, mở rộng U3.)
- U3: View CarModel Details: Một khách hàng được hiển thị chi tiết các CarModel, như mô tả và quảng cáo. (Mở rộng U2, mở rộng U7.)
- U4: Search: Tìm kiếm khách hàng cho các CarModel bằng cách xác định hạng mục, cấu tạo và kích cỡ động cơ. (Specializes U13, bao gồm U2.)
- U5: Log On: Thành viên đăng nhập vào iCoot sử dụng số thành viên và mật khẩu hiện tại. (Mở rộng bởi U6, U8, U9, U10 và U12.)
- U6: View Member Details: Một thành viên xem một số chi tiết được lưu trữ bởi iCoot, chẳng hạn như tên, địa chỉ và CreditCard chi tiết. (Mở rộng U5).
- U7: Make Reservation: Một thành viên đăng ký một CarModel khi xem chi tiết của nó. (Mở rộng U3.)
- U8: View Rentals: Một thành viên xem một bản tóm tắt các Cars họ hiện đang thuê. (Mở rộng U5).
- U9: Change Password: Một thành viên thay đổi mật khẩu mà họ sử dụng để đăng nhập. (Mở rộng U5).
- U10: View Reservations: Một Thành viên xem tóm tắt các đăng ký như ngày, giờ và CarModel. (Mở rộng U5, mở rộng thêm bởi U11.)
- U11: Cancel Reservation: Một thành viên hủy một đăng ký. (Mở rộng U10.)
- U12: Log Off: Một thành viên log off từ iCoot. (Mở rộng U5).
- U13: Look for CarModels: Một khách hàng tìm các CarModels từ danh mục. (khái quát hóa bởi U1 và U4.)

2.9.4 Sơ đồ Use Case

Sơ đồ Use Case cho iCoot được hiển thị trong Hình B.11.



Hình B.11: Sơ đồ Use Case cho iCoot

2.9.5 Tổng quan Use Case

Customer có thể tìm kiếm các CarModel trong danh mục, bằng cách duyệt các chỉ số CarModel (U1) hoặc bằng cách tìm kiếm (U4). Trong trường hợp tìm kiếm, các Customer cụ thể hóa các hạng mục, cấu tạo và kích thước động cơ mà họ đang quan tâm. Dù bằng cách nào, sau mỗi lần đạt được, các Customer sẽ được hiển thị các kết quả tập các CarModels phù hợp (U2), cùng với các thông tin cơ bản như tên CarModel. Sau đó, Customer có thể chọn để xem thêm thông tin về các đối tượng CarModel cụ thể như mô tả và quảng cáo (U3).

Customer đã trở thành một thành viên có thể đăng nhập (U5) và đạt được quyền truy cập vào các dịch vụ bổ sung. Các dịch vụ bổ sung thêm là: Đăng ký (U7), Hủy đăng ký (U11), kiểm tra các chi tiết thành viên (U6), Xem đăng ký (U10), thay đổi mật khẩu (U9), xem thuê xe (U8) và Đăng xuất (U12).

Có hai loại Customer, Thành viên và không phải thành viên.

Truy cập chỉ mục và tìm kiếm CarModels hai cách tìm kiếm khác nhau cho CarModels (U13). Để xem chi tiết CarModel, một Customer phải xem kết quả tìm kiếm Model.

Để đặt CarModel, một thành viên xem chi tiết của nó (không phải thành viên không thể đăng ký, ngay cả khi họ đang xem chi tiết).

Để hủy đăng ký, một thành viên phải xem đăng ký họ.

2.9.6 Chi tiết Use Case

U1: Duyệt Index. (Specializes U13, bao gồm U2.)
Điều kiện tiên quyết: Không có.
1. Customer chọn một tiêu đề chỉ mục.
2. Customer lựa chọn để xem CarModels cho tiêu đề chỉ mục đã chọn.
3. Bao gồm U2.
Hậu điều kiện: Không có.

U2: Xem kết quả. (Bao gồm U1 và U4, mở rộng U3.)
Điều kiện tiên quyết: Không có.
1. iCoot trình bày Customer với một bản tóm tắt của mỗi CarModel, bao gồm cả số Model và giá cả.
2. Mở rộng với U3.
Hậu điều kiện: Không

U3: Xem CarModel chi tiết. (Mở rộng U2, mở rộng U7.)
Điều kiện tiên quyết: Không có.
1. Customer chọn một trong những CarModels phù hợp.
2. Customer yêu cầu chi tiết của CarModel chọn.
3. iCoot hiển thị chi tiết cho các mẫu xe được lựa chọn (làm, kích thước động cơ, giá cả, mô tả, quảng cáo và poster).
4. Nếu Customer là thành viên đăng nhập, mở rộng với U7.
Hậu điều kiện: iCoot đã hiển thị chi tiết của CarModels chọn.
Yêu cầu phi chức năng: r1. Adverts sẽ được hiển thị bằng cách sử dụng giao thức truyền tải chứ không phải là yêu cầu tải về.

U4: Tìm kiếm. (Specializes U13, bao gồm U2.)
Điều kiện tiên quyết: Không có.
1. Customer chọn loại (nếu có).
2. Customer chọn cấu tạo (nếu có).
3. Customer chọn yêu cầu kích thước động cơ (nếu có).
4. Customer thực hiện việc tìm kiếm.
5. Bao gồm U2.
Hậu điều kiện: Không có.

U5: Log On. (Mở rộng bởi U6, U8, U9, U10 và U12.)
Điều kiện tiên quyết: Thành viên đã có được mật khẩu từ cửa hàng địa phương.
1. Thành viên nhập vào số lượng thành viên.
2. Thành viên nhập vào mật khẩu.
3. Kể từ iCoot phải thực thi một đăng nhập với một Thành viên, Thành viên có thể lựa chọn để ăn cắp (chỉ hiệu lực và do đó lấy lên từ) một session đang tồn tại.
4. Thành viên bầu chọn để đăng nhập vào.
5. Mở rộng với U6, U8, U9, U10, U12.

Hậu điều kiện: Thành viên đăng nhập.

U6: Xem chi tiết thành viên. (Mở rộng U5).
1. Thành viên lựa chọn xem chi tiết thành viên.
2. Thành viên được hiển thị các chi tiết thành viên (tên, địa chỉ, tình trạng, số tiền thiếu nợ, chi tiết CreditCard).
3. Vì lý do an ninh, iCoot hiển thị chỉ có bốn chữ số cuối của số CreditCard của thành viên.
4. iCoot thông báo thành viên đó để chỉnh sửa chi tiết, họ phải liên hệ với cửa hàng địa phương của họ.
Hậu điều kiện: Thành viên đã được hiển thị với các chi tiết thành viên.

U7: Đăng ký. (Mở rộng U3.)
Điều kiện tiên quyết: Khách hàng là một thành viên đã đăng nhập.
1. Thành viên lựa chọn đăng ký CarModel.
2. iCoot yêu cầu thành viên xác nhận, đưa ra một cảnh báo rằng nếu không thu thập được một CarModel đã đăng ký.
3. Thành viên xác nhận đăng ký.
4. iCoot hiển thị cho thành viên số đăng ký và chỉ ra rằng trợ lý sẽ liên lạc khi có xe.
5. Khi một trợ lý đăng nhập iCoot, trợ lý đưa ra một danh sách đăng ký đòi hỏi hành động.
6. Trợ lý thực hiện các hành động cần thiết để theo dõi Đăng ký.
Hậu điều kiện: Bất kỳ yêu cầu Đăng ký đã được thực hiện.

U8: Xem thuê xe. (Mở rộng U5).
Điều kiện tiên quyết: Không có. Các mối quan hệ: U5.
1. Thành viên bầu ra để xem thuê nhà của họ.
2. iCoot thiệu thành viên với bản tóm tắt của từng xe mà họ đang có nhưng không có cho thuê (bao gồm cả biển số và ngày đến hạn).
Hậu điều kiện: iCoot hiển thị thành viên với các tóm tắt Ô tô đang thuê.

U9: Đổi mật khẩu. (Mở rộng U5).
Điều kiện tiên quyết: Không có.
1. Thành viên lựa chọn thay đổi mật khẩu.
2. Thành viên nhập mật khẩu cũ (được che khuất trên màn hình).
3. Thành viên nhập mật khẩu mới (được ẩn).
4. Thành viên nhập mật khẩu mới một lần nữa.
5. Thành viên chọn thay đổi.
6. iCoot yêu cầu xác nhận (cảnh báo rằng mật khẩu mới phải được ghi nhớ).
7. Nếu thành viên xác nhận, mật khẩu được thay đổi.
Hậu điều kiện: Mật khẩu đã được thay đổi.

U10: Xem đăng ký. (Mở rộng U5, mở rộng thêm bởi U11.)
Điều kiện tiên quyết: Không có.
1. Thành viên lựa chọn xem đăng ký.
2. iCoot hiển thị tóm tắt các đăng ký của thành viên.

3. Mở rộng với U11.
Hậu điều kiện: Thành viên đã được hiển thị với bản tóm tắt.

U11: Hủy đăng ký. (Mở rộng U10.)
Điều kiện tiên quyết: Không có.
1. Thành viên chọn một đăng ký.
2. Thành viên chọn hủy đăng ký.
3. iCoot yêu cầu xác nhận.
4. Thành viên xác nhận rằng họ muốn hủy bỏ việc đăng ký.
5. iCoot đánh dấu hủy đăng ký và cập nhật thông tin cho trợ lý.
Hậu điều kiện: Các đăng ký bị hủy được xác nhận và được đánh dấu.

U12: Đăng xuất.
Điều kiện tiên quyết: Không có.
1. Thành viên lựa chọn đăng xuất.
2. iCoot kết thúc phiên hiện tại.
3. iCoot cập nhật không cho sử dụng chức năng.
Hậu điều kiện: Thành viên đăng xuất.

U13: Tìm CarModel (chuyên môn hóa của U1 và U4.)
Điều kiện tiên quyết: Không có.
Hậu điều kiện: Khách hàng đã được hiển thị tóm lược các CarModel.

7. Các yêu cầu bổ sung
s1. Các chức năng khách hàng phải chạy trong các phiên bản .Net Framework khác nhau.
s2. iCoot phải có khả năng để đáp ứng với một danh mục của 100.000 CarModels.
s3. iCoot phải có khả năng phục vụ 1.000.000 khách hàng đồng thời
8. Độ ưu tiên Use Case
Dưới đây là danh sách các ưu tiên Use Case cho iCoot theo màu sắc:
• Màu xanh:
– U1: Browse Index
– U4: Search
–U2:ViewResults
–U3:View CarModel Details
–U5:LogOn
• Màu hổ phách:
– U12: Logoff.
–U6:View Member Details
–U7: Make a Reservation
– U10: View Reservations
• Màu đỏ:
– U11: Cancel Reservation
–U8:View Rentals
– U9: Change Password

LAB 3. PHÂN TÍCH KHÍA CẠNH TĨNH

Trong quá trình phân tích, chúng ta cố gắng xác định và mô tả tất cả các thực thể nghiệp vụ liên quan tới hệ thống cần xây dựng, cùng với các thuộc tính và mối quan hệ giữa chúng. Đây chính là mục đích của việc phân tích khía cạnh tĩnh và cũng là vấn đề chính của bài thực hành này. Việc phân tích chi tiết và kiểm tra kết quả sẽ được thực hiện trong bài thực hành số 4.

Xuyên suốt bài thực hành này, bạn cần phải cập nhật bảng thuật ngữ của dự án và các yêu cầu phi chức năng (trong cả phần Use Cases và phần Các yêu cầu bổ sung).

Sau khi hoàn thành bài Lab này, bạn có thể:

- Xác định được các lớp (ở mức phân tích) biểu diễn cho các thực thể nghiệp vụ.
- Xác định các mối quan hệ giữa các lớp và bản số của chúng.
- Thể hiện các lớp, mối quan hệ và bản số trên một sơ đồ lớp.
- Tìm và bổ sung các thuộc tính cho các lớp.
- Cập nhật bảng thuật ngữ của dự án và các yêu cầu phi chức năng.

3.1 Xác định các lớp

3.1.1 Tạo danh sách các lớp ứng viên

Đọc qua mô tả về các use cases hệ thống, tìm và liệt kê tất cả các danh từ. Danh sách này sẽ tạo thành tập các lớp ứng viên ban đầu.

Ví dụ: Xem xét mô tả về các use cases U3, U4, U5. Ta đánh dấu các danh từ như sau:

U2:View Results. (Included by U1 and U4, extended by U3.)

Preconditions: None.

1. iCoot presents Customer with a summary of each retrieved car model, including model number and price.
2. Extend with U3.

Postconditions: None.

U3:View CarModel Details. (Extends U2, extended by U7.)

Preconditions: None.

1. Customer selects one of the matching car models.
2. Customer requests details of the selected car model.
3. iCoot displays details for the selected car model (makes, engine size, price, description, advert and poster).

4. If Customer is a logged-on Member, extend with U7.

Postconditions: *iCoot* has displayed details of selected car models.

Non-Functional Requirements: r1. Adverts should be displayed using a streaming protocol rather than requiring a download.

U4:Search. (Specializes U13, includes U2.)

Preconditions: None.

1. Customer selects required categories (if any).
2. Customer selects required makes (if any).
3. Customer selects required engine sizes (if any).
4. Customer initiates the search.
5. Include U2.

Postconditions: None.

Abnormal paths: a1. If Customer specifies no categories, makes or engine sizes, rather than retrieving the entire catalog, iCoot should not allow the search to be initiated.

Liệt kê các danh từ vừa tìm được:

iCoot, customer, car model, model number, price, customer requests details (reservation), makes, engine size, price, description, advert, poster, member, details of car models (car model details), category, the search.

Bổ sung thêm các lớp mà bạn phát hiện được nhờ kinh nghiệm phát triển ứng dụng hoặc những lớp mà bạn nghĩ là có liên quan khi suy nghĩ về hệ thống cần xây dựng.

Ví dụ: có thể bổ sung thêm các danh từ sau: *car, vendor*.

Ở giai đoạn này không cần phải kiểm tra, rà soát lại phần mô tả vấn đề (problem statement) hoặc mô hình nghiệp vụ vì chúng được phát biểu ở dạng tổng quát thay vì các use cases hệ thống trọng tâm và cụ thể.

3.1.2 Sàng lọc danh sách lớp ứng viên

Kiểm tra từng lớp ứng viên trong danh sách và loại bỏ nó nếu rơi vào các trường hợp sau:

- Nói về bản thân hệ thống: vì nó sẽ được tách thành nhiều lớp khác.
- Một actor không có thông tin gì đặc biệt: Ta không cần mô hình hóa các tác nhân (con người hoặc hệ thống khác) trừ khi chúng ta cần xử lý thông tin của chúng.
- Nói về giao diện người dùng hoặc chi tiết xử lý: Vấn đề giao diện người dùng và thao tác xử lý sẽ được thảo luận trong giai đoạn phân tích khía cạnh động của hệ thống nhưng cũng không cần mô hình hóa chi tiết cho tới giai đoạn thiết kế.

- Tầm thường: Là những danh từ có kiểu tầm thường (chẳng hạn như chuỗi ký tự) hoặc có thể trở thành một thuộc tính của một lớp khác (ví dụ như mã số khách hàng).

Ví dụ: Từ danh sách các danh từ ở trên, ta loại bỏ các danh từ sau:

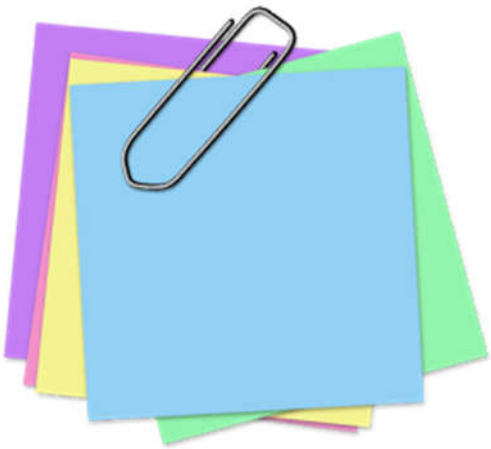
- *iCoot*: bản thân hệ thống.
- *model number, price, engine size, price, description, advert, poster*: tầm thường.
- *the search*: nói về việc xử lý.

Như vậy, các danh từ còn lại gồm có:

customer, car model, customer requests details (reservation), makes, details of car models (car model details), category, car, vendor.

Trong quá trình xem xét lớp nào cần được giữ lại và lớp nào cần phải loại bỏ, bạn có thể phát hiện thêm các lớp mới. Khi đó, thêm các lớp này vào danh sách ứng viên và lại áp dụng các quy tắc sàng lọc như trên.

Index Card



Sticky Notes

Với mỗi lớp được giữ lại sau quá trình sàng lọc, viết tên của nó lên các mảnh giấy (index card hoặc sticky notes). Việc này sẽ giúp bạn dễ dàng hơn trong việc sắp xếp, bố trí các lớp trong sơ đồ lớp ở giai đoạn tiếp theo.

3.2 Tạo quan hệ giữa các lớp

3.2.1 Kết nối các lớp liên quan với nhau

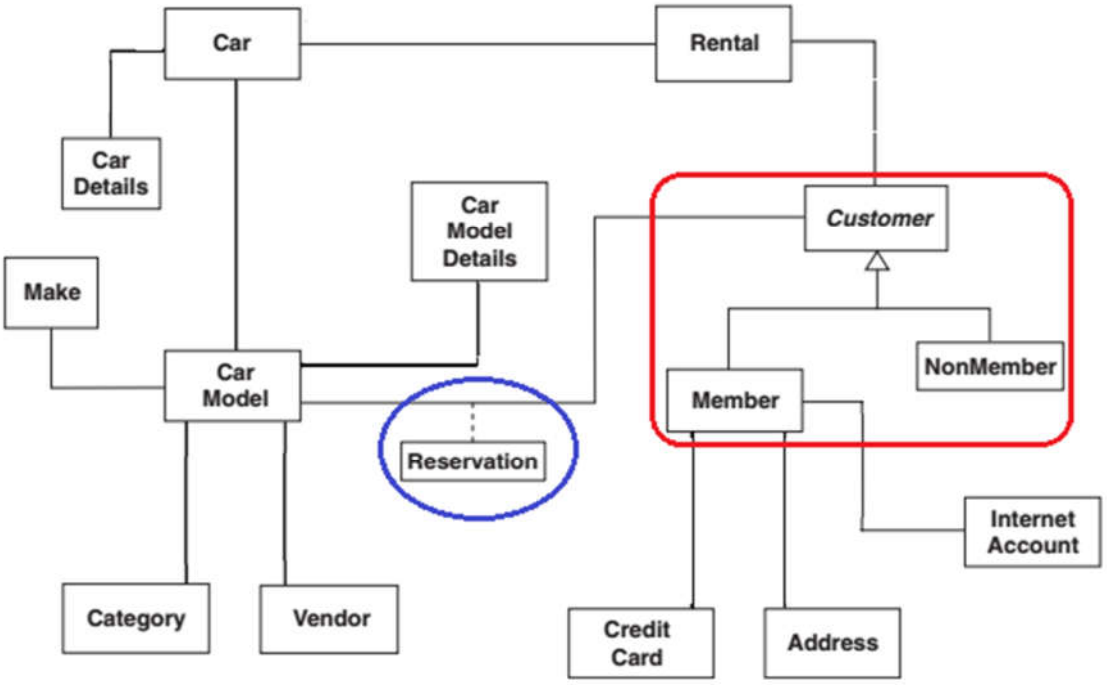
Vẽ mối quan hệ giữa các cặp lớp mà bạn nghĩ chúng có liên quan chặt chẽ với nhau. Ở giai đoạn này, chỉ cần thể hiện mối quan hệ kết hợp (association) và mối quan hệ kế thừa (inheritance). Các mối kết hợp sẽ được tinh chỉnh cho phù hợp trong giai đoạn tiếp theo.

Nếu có quan hệ kế thừa, cần xem xét đến việc bổ sung lớp các trừu tượng (để dễ mở rộng trong tương lai). Khi bạn phân vân và muốn sử dụng mối quan hệ kế thừa, phải đảm bảo nó thỏa mãn các tiêu chí sau:

- Nó làm cho sơ đồ rõ ràng hơn và dễ hiểu hơn, đặc biệt là các khách hàng không có chuyên môn kỹ thuật cũng có thể xem và hiểu được sơ đồ.
- Nó không dùng cho mục đích thể hiện việc cài đặt mã nguồn. Việc chia sẻ mã cài đặt là vấn đề thiết kế, không phải là vấn đề phân tích.

Nếu có một lớp nào đó kết hợp (nối) với một mối quan hệ giữa hai lớp khác, hãy xem xét đến việc tạo ra một lớp kết hợp.

Ví dụ: Hình sau cho thấy mối quan hệ giữa các lớp trong hệ thống iCoot.

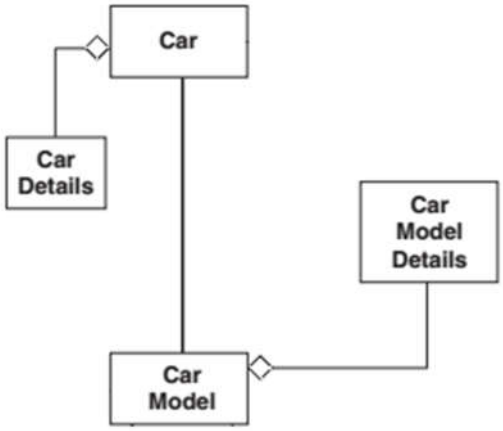


Hình chữ nhật góc tròn màu đỏ cho thấy các lớp có mối quan hệ kế thừa và lớp trừu tượng Customer. Hình ê-lip màu xanh cho thấy lớp kết hợp Reservation được tạo ra từ mối quan hệ nhiều-nhiều giữa hai lớp Customer và CarModel.

3.2.2 Tinh chỉnh mối kết hợp

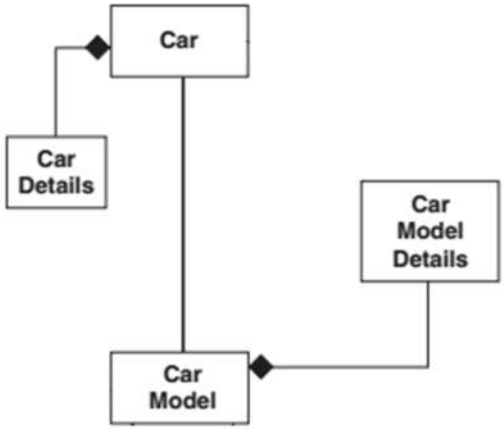
Một vài mối kết hợp có thể được mô hình hóa chính xác hơn bằng cách sử dụng mối quan hệ kết tập (aggregation). Hãy tìm các mối quan hệ như vậy.

Ví dụ: Vì thông tin chi tiết về mẫu mã xe (CarModelDetails) là thật sự cần thiết cho mỗi mẫu xe (CarModel) và mỗi mẫu xe có nhiều thông tin chi tiết về nó nên mối quan hệ giữa hai lớp này được chuyển sang dạng quan hệ kết tập. Tương tự đối với 2 lớp còn lại.



Nếu có các mối quan hệ kết tập, hãy tìm trong số chúng các mối quan hệ hội đủ điều kiện để trở thành mối quan hệ cấu thành (composition).

Ví dụ: Mỗi mẫu mã xe bắt buộc phải có các thông tin mô tả đầy đủ và chi tiết về nó. Nếu loại bỏ chi tiết về mẫu mã xe thì thông tin về mẫu mã xe không còn ý nghĩa. Vì vậy, quan hệ giữa hai lớp CarModel và CarModelDetails được chuyển thành quan hệ cấu thành.

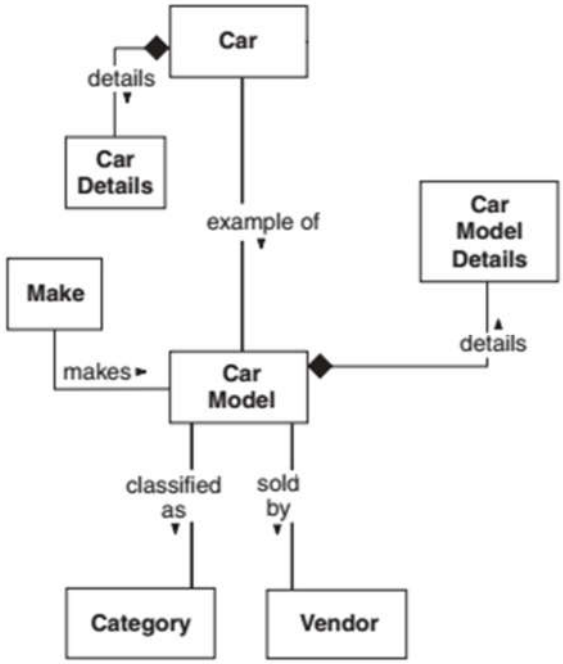


3.2.3 Mô tả các quan hệ

Với mỗi mối quan hệ, ngoại trừ quan hệ kế thừa, hãy bổ sung thêm các mô tả sau:

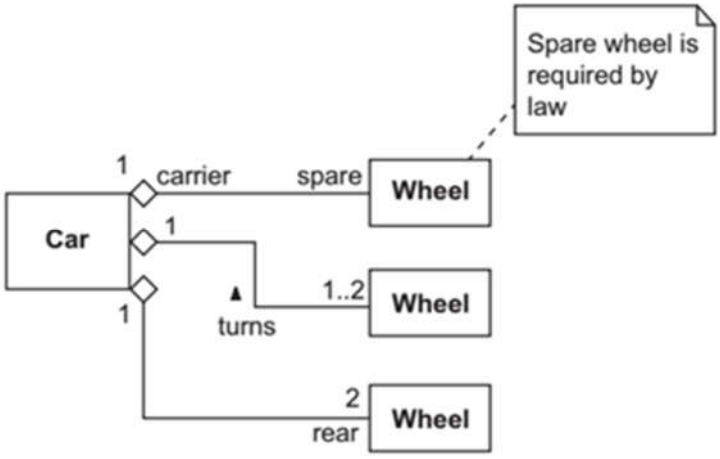
- Tên của mối kết hợp: cho biết mối kết hợp biểu diễn cái gì.
- Vai trò của lớp nguồn: mô tả vai trò của lớp nguồn trong mối kết hợp.
- Vai trò của lớp đích: mô tả vai trò của lớp đích trong mối kết hợp.

Ví dụ: Tên của mối kết hợp giữa các lớp trong iCoot được thể hiện như sau:



Bạn không cần phải mô tả đầy đủ cả 3 thứ trên, có thể chỉ cần tên của mối kết hợp hoặc tên vai trò nếu muốn. Mối quan hệ kết tập hoặc cấu thành thường đã được biểu diễn rõ ràng nên có thể bỏ qua phần mô tả này.

Ví dụ về cách thể hiện vai trò của lớp nguồn và lớp đích:



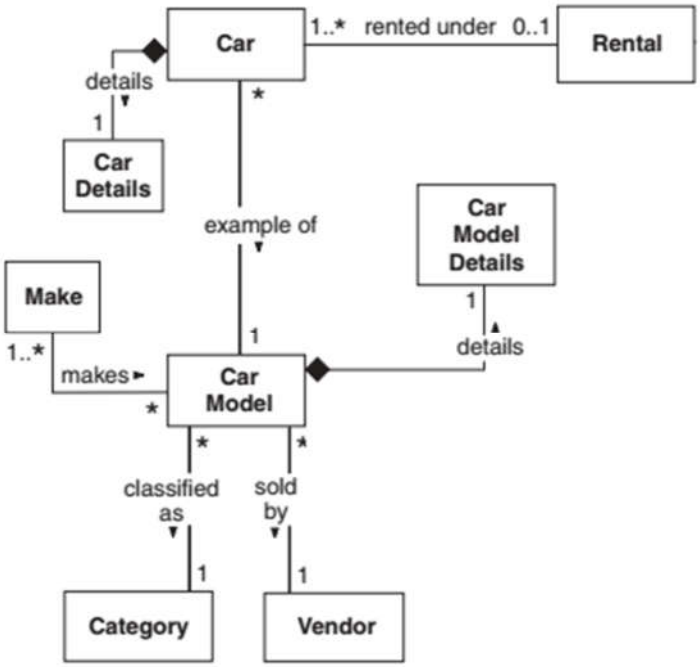
3.2.4 Thêm bản số

Chỉ ra số lượng đối tượng liên quan trong mỗi quan hệ (ngoại trừ quan hệ kế thừa đã được hiểu ngầm là quan hệ 1:1). Sử dụng một trong các cách sau để chỉ ra bản số của các mối kết hợp:

- m: Có chính xác m đối tượng.
- m..n: Có từ m tới n đối tượng.

- m..*: Có ít nhất m đối tượng.
- *: Một số bất kỳ, kể cả 0. Viết tắt cho trường hợp 0..*.

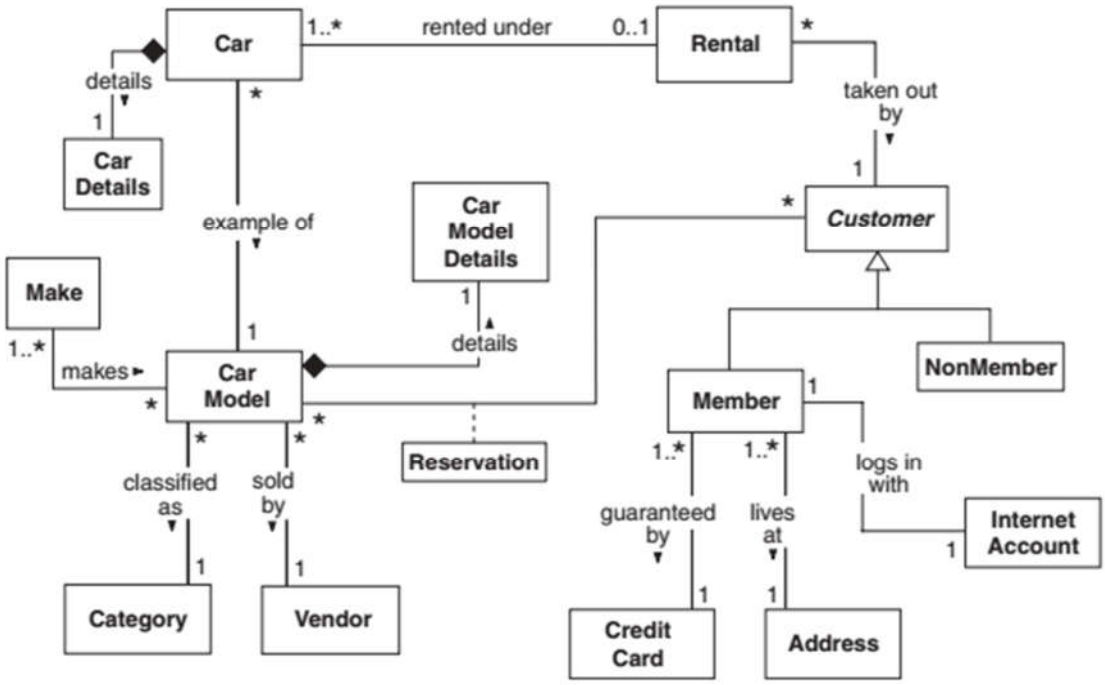
Ví dụ:



3.2.5 Vẽ sơ đồ lớp mức phân tích

Mỗi lớp được biểu diễn bởi một hình chữ nhật, bên trong ghi tên lớp ở dạng in đậm. Nếu là lớp trừu tượng, tên được viết in nghiêng hoặc được đánh dấu bằng từ khóa {abstract} phía trên hoặc bên trái tên lớp thay vì chữ in nghiêng. Mối quan hệ giữa các lớp được thể hiện bằng các đoạn thẳng kèm theo chú thích.

Ví dụ: Sơ đồ lớp mức phân tích cho hệ thống iCoot được vẽ như sau:



3.3 Thêm mô tả các lớp vào bảng thuật ngữ

Một khi sơ đồ lớp cơ bản được chấp nhận, bạn cần thêm một đoạn mô tả ngắn cho mỗi lớp và bổ sung vào bảng thuật ngữ. Một số tên lớp đã tồn tại trong bảng thuật ngữ. Vì vậy, miễn là phần mô tả thuật ngữ vẫn phù hợp thì chỉ cần đánh dấu thuật ngữ đó là một lớp ở mức phân tích.

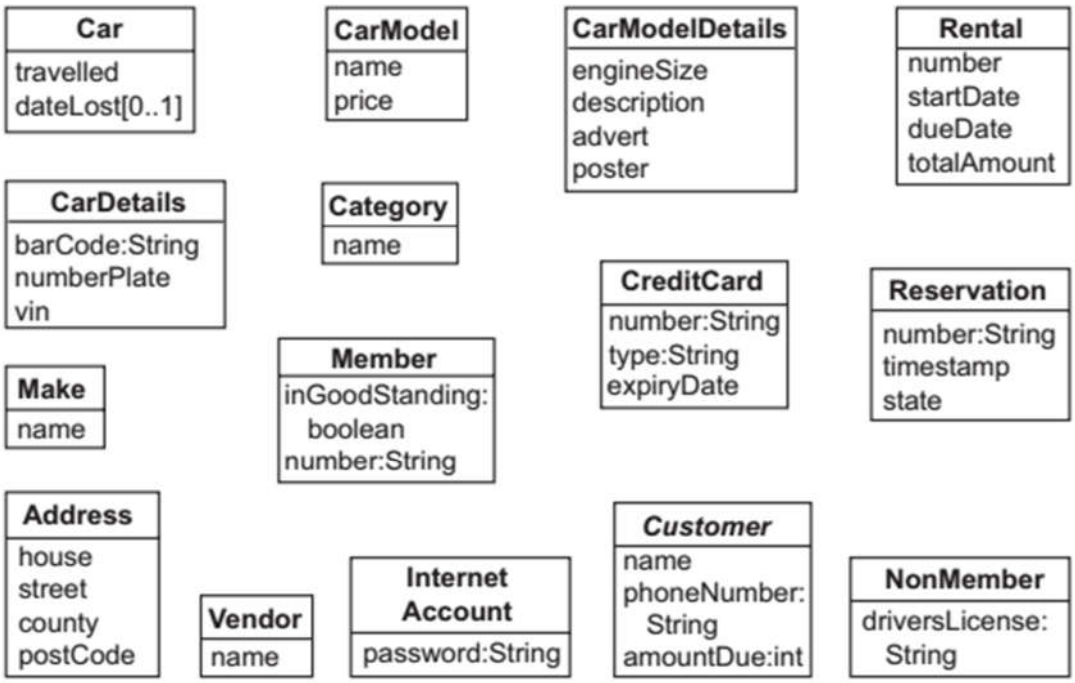
Ví dụ: Các lớp được mô tả trong bảng thuật ngữ như sau:

Thuật ngữ	**Định nghĩa**
Car (business object, system object, analysis object)	Một loại xe cụ thể ứng với CarModel được trưng bày trong cửa hàng để cho thuê, bán.
CarDetails (Analysis object)	Các thông tin chi tiết hơn về Car.
CarModel (business object, system object, analysis object)	Một mẫu mã xe trong danh mục xe, dùng cho việc đặt hàng.
CarModelDetails (Analysis object)	Thông tin chi tiết hơn một CarModel.
Catalog (Business)	Một danh mục, mô tả các mẫu mã xe cho thuê

3.4 Thêm thuộc tính vào các lớp

Cuối cùng là tìm các thuộc tính cho mỗi lớp đối tượng. Ghi lại các thuộc tính của mỗi lớp vào một mảnh giấy riêng. Không cần ghi lại các thuộc tính có thể suy diễn (dẫn xuất) từ các thuộc tính khác. Chẳng hạn, lớp Circle (hình tròn) có 2 thuộc tính bán kính và đường kính nhưng chỉ cần chọn một trong hai và bỏ thuộc tính còn lại.

Ví dụ: Hình dưới đây cho thấy thuộc tính của các lớp trong hệ thống iCoot.



Với mỗi thuộc tính, hãy viết một đoạn mô tả ngắn về nó. Sau đó, chỉ ra kiểu dữ liệu thích hợp cho chúng. Một số trường hợp, bạn có thể chưa biết kiểu hoặc không muốn áp kiểu cho nó. Vì thế, kiểu của chúng sẽ được trong giai đoạn thiết kế.

LAB 4. PHÂN TÍCH KHÍA CẠNH ĐỘNG

Ở phần trước sinh viên đã làm quen với phân tích khía cạnh tĩnh, tại lab này sinh viên cần sử dụng phân tích khía cạnh động để kiểm chứng những thông tin đã được khám phá. Phần này yêu cầu sử dụng lược đồ giao tiếp (nhận diện use case) liệt kê và mô tả các quá trình hoạt động và xây dựng mô hình trạng thái trong làm trắc nghiệm.

Mục tiêu đạt được sau quá trình phân tích khía cạnh động chính là khả năng phát hiện và bỏ lỗi phát sinh trong lược đồ lớp, do đó ta cần phải vừa làm vừa sửa tuy nhiên phải cố gắng để hoàn thiện với ít lỗi nhất có thể. Sinh viên cũng cần sử dụng thành thạo các lớp điều khiển (controller classes) và các lớp biên (boundary classes) từ đó có thể mô tả các lớp này trong danh sách lớp.

Hoàn tất bài lab này sinh viên phải có khả năng:

- Kiểm chứng một mô hình phân tích lớp.
- Nhận dạng các use case.
- Vẽ lược đồ giao tiếp.
- Nhận dạng và mô tả các hoạt động.
- Xây dựng và vẽ mô hình trạng thái.

4.1 NHẬN DIỆN CÁC USE CASE

4.1.1 Chọn một use case

Chọn ra một use case để nhận dạng. Ta nên bắt đầu với use case đầu tiên sinh viên xử lý khi làm một bài trắc nghiệm và từ đó mở rộng theo các luồng chính của các use case khác một khi ta đã hoàn tất xử lý các luồng chính, ta ó thể xử lý các use case thứ cấp.

4.1.2 Vẽ lược đồ giao tiếp

Các use case thường bắt đầu bởi một tác nhân (actor) xác định đối tượng biên (một giao diện người dùng) theo đó đối tượng này làm một việc gì đó nhằm lấy thông tin. Lấy ví dụ, ta có thể dùng một đường thẳng nối giữa tác nhân và biên để chỉ ra đường giao tiếp giữa chúng. Gán nhãn tác nhân và biên với tên và lớp (nếu cần thiết). Tiếp đến, vẽ một thông điểm được gửi từ biên bởi tác nhân. Xử lý thông qua các use case từng bước một, các luồng thông điệp giữa các đối tượng được phát sinh nhằm thỏa mãn các bước use case. Ngoài ra ta cần thêm các điều khiển (controllers), biên, thực thể vào từ điển từ vựng.

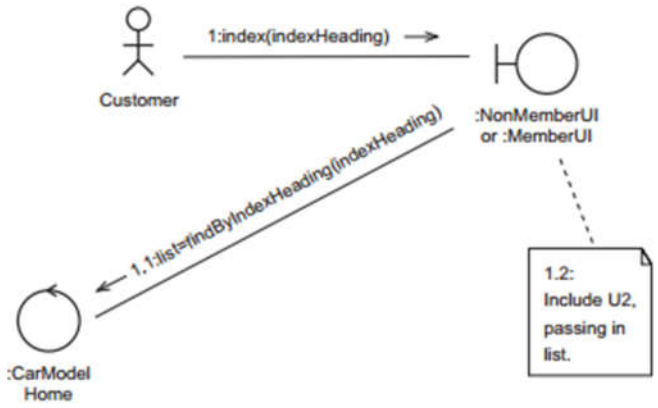
Những mẹo nhỏ giúp vẽ lược đồ:

- Vẽ đối tượng biên với kích thước lớn, bao trùm một số các tác vụ liên quan (sau này các tác vụ có thể bị phân nhỏ trong quá trình thiết kế)

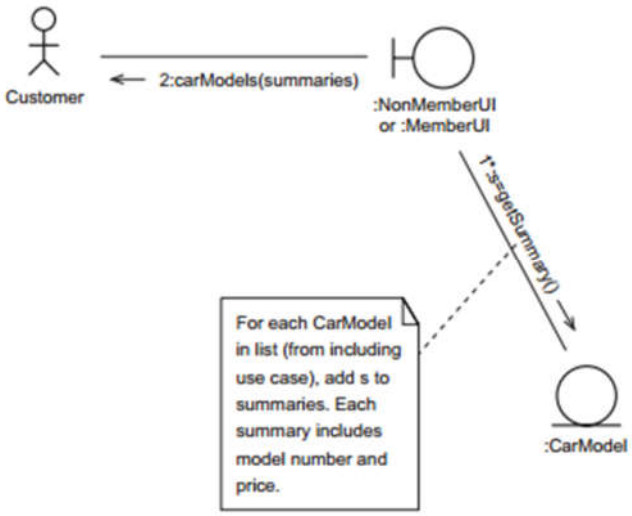
- Đảm bảo rằng các đối tượng thực thể đều được vẽ trong lược đồ lớp.
- Xây dựng thêm điều khiển (controller) ngay khi ta cần tập trung điều khiên của một quy trình nghiệp vụ phức tạp.
- Giới thiệu các nhà (nhà là một dạng điều khiển với mục đích đặc biệt) ngay khi cần tạo ra thực thể hoặc cần lấy thực thể từ cơ sở dữ liệu của các thực thể được tạo ra trước đó.
- Sử dụng từ viết tắt cho các tham số và sử dụng ghi chú để ghi rõ chữ viết tắt đó nghĩa là gì và viết tắt cho gì.
- Chỉ gán giá trị cho giá trị trả về khi các giá trị đó không rõ ràng (ta cần cung cấp thêm thông tin mô tả cho qua trình vận hành)
- Tên chỉ có giá trị khi chúng được dùng trong lược đồ (lấy ví dụ nếu giá trị trả về sau đó được dùng như một tham số)

Ví dụ iCoot:

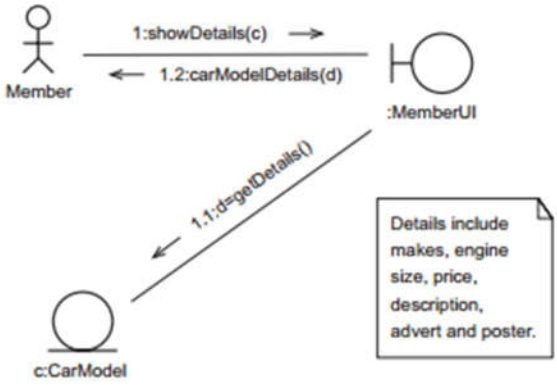
Các lược đồ giao tiếp cho iCoot được mô tả bởi từng hình dưới đây. Lưu ý rằng sử dụng các lính canh (Guards- thể hiện điều kiện trong ngoặc vuông) để chỉ rõ thông điệp có điều kiện và dấu * để chỉ ra thao tác lặp (lính canh lặp có thể được dùng để điều khiển việc lặp, tuy nhiên những thông tin này có thể khiến lược đồ trở nên rất phức tạp)



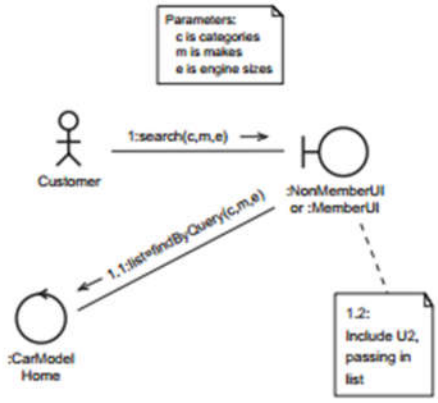
Lược đồ giao tiếp cho U1: Browse Index



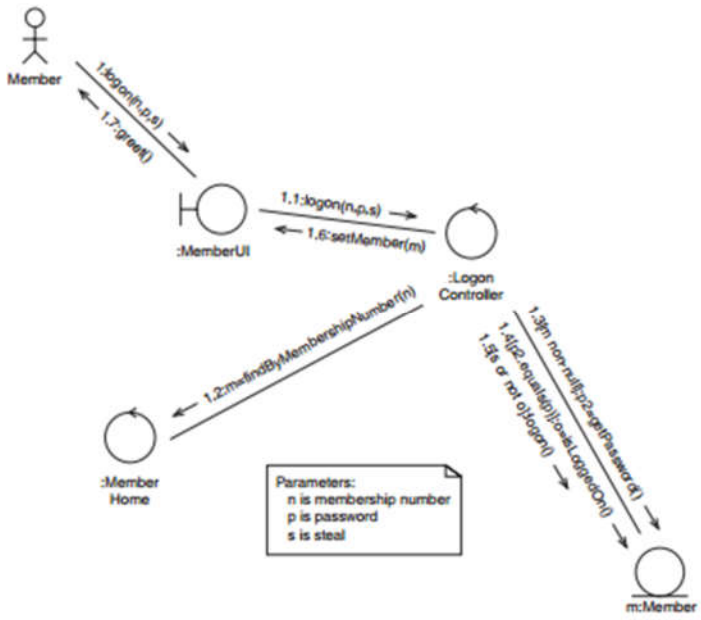
Lược đồ giao tiếp cho U2: View results



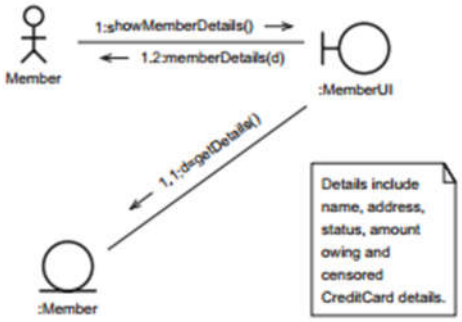
Lược đồ giao tiếp cho U3: View CarModel Details



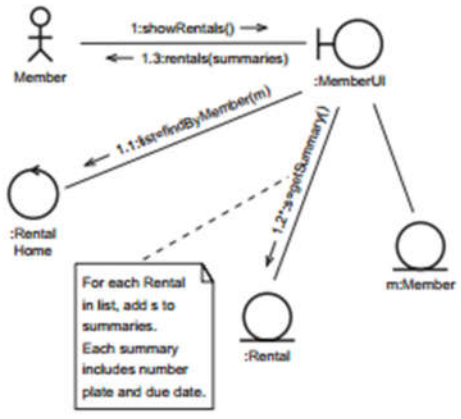
Lược đồ giao tiếp cho U4: Search



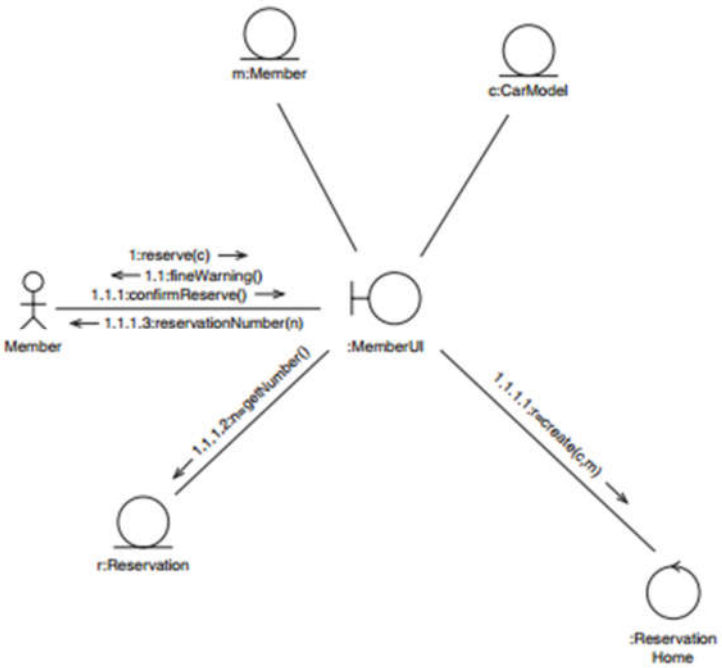
Lược đồ giao tiếp cho U5: Logon



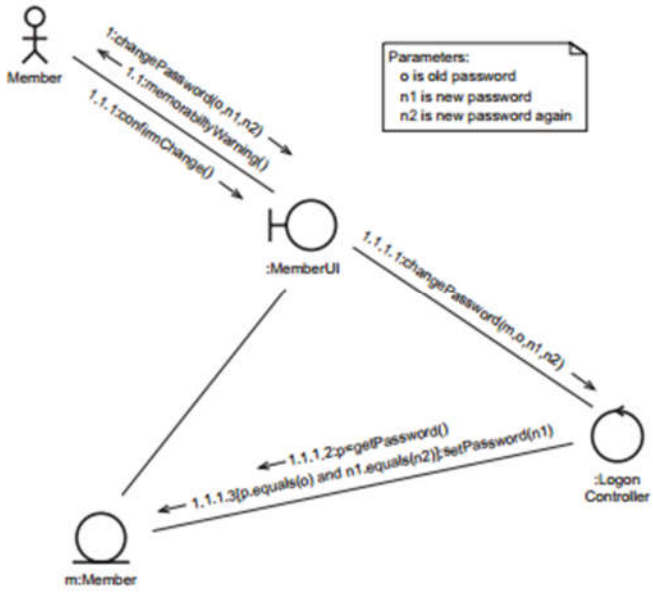
Lược đồ giao tiếp cho U6: View Member Detail



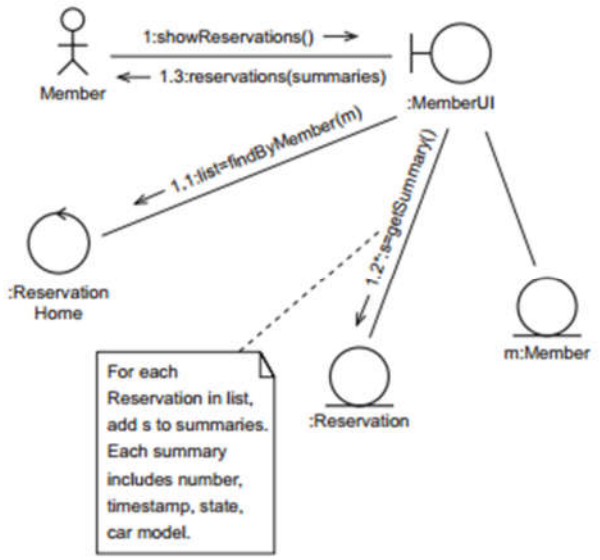
Lược đồ giao tiếp cho U8: View rentals



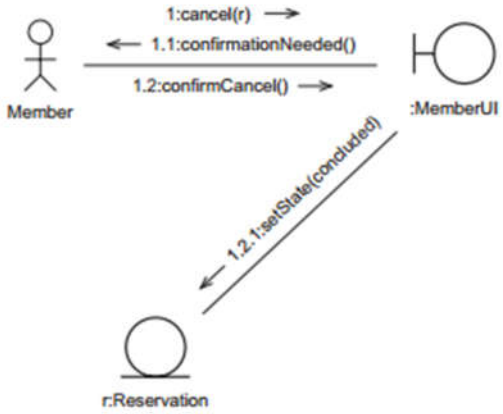
Lược đồ giao tiếp cho U7: Make reservation



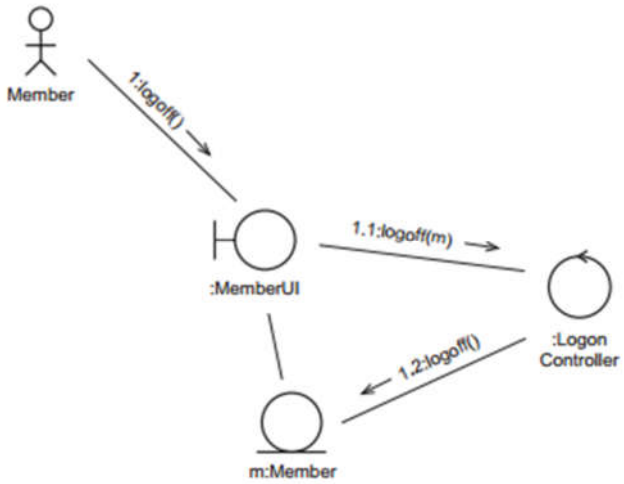
Lược đồ giao tiếp cho U9: Change password



Lược đồ giao tiếp cho U10: View reservation



Lược đồ giao tiếp cho U11: Cancel reservation



Lược đồ giao tiếp cho U12: Log off

4.2 CHI TIẾT HÓA CÁC THAO TÁC

Liệt kê mỗi lớp (gồm các thực thể, giao diện, điều khiển) trên giấy, mỗi lớp một mặt giấy.

Với mỗi lớp, lướt qua các lược đồ giao tiếp và ghi lại những thông điệp được gửi đến một đói tượng của lớp thông qua các thao tác. Trong quá trình thực thi việc này, cần chú ý chỉ rõ ra loại của tham số cũng như kiểu dữ liệu trả về nếu có. Đôi khi ta không biết kiểu dữ liệu tại thời điểm xem xét hoặc ta muốn đợi đến bước thiết kế mới quyết định kiểu dữ liệu, trong những trường hợp như vậy ta có thể để kiểu dữ liệu trắng.

Cần nhớ rằng tham số và kiểu dữ liệu trả về phải phù hợp với kiểu thuộc tính mà ta chọn ở bước phân tích tĩnh- lấy ví dụ nếu ta đã định ra rằng ma số thư viện của sinh viên có kiểu dữ liệu là kiểu chuỗi thì nó phải luôn là kiểu chuỗi khi được dùng là tham số hay giá trị trả về.

Với mỗi thao tác, thêm một hoặc hai câu ngắn để mô tả thao tác này làm gì và các tham số đầu vào cũng như giá trị trả về của nó là gì.

4.3 XÂY DỰNG MÔ HÌNH TRẠNG THÁI

Nghĩ về điều gì sẽ xảy ra khi sinh viên làm bài trắc nghiệm. Lấy ví dụ có thể làm theo cách ban đầu sinh viên không cần đăng nhập, sinh viên vẫn có thể trả lời một số câu hỏi, sau đó có kết quả và cứ thế tiếp tục. Tuy nhiên làm thế nào để ta mô hình hóa chu trình này cho máy tính trạng thái. Ta có thể nghĩ đến sử dụng phác thảo giao diện (GUI) mà ta đã tạo ở các bước trước.

Liệt kê toàn bộ các trạng thái trong chu trình vòng đời của việc làm trắc nghiệm (tuy nhiên ta cần phải giới hạn một sinh viên đang làm mà thôi chứ khoan quan tâm đến tất cả sinh viên cùng lúc). Điều này giúp cho việc hiểu các trạng thái một sinh viên sẽ trải qua từ các góc nhìn của hệ thống. Vẽ mỗi trạng thái trong một hình chữ nhật bo góc.

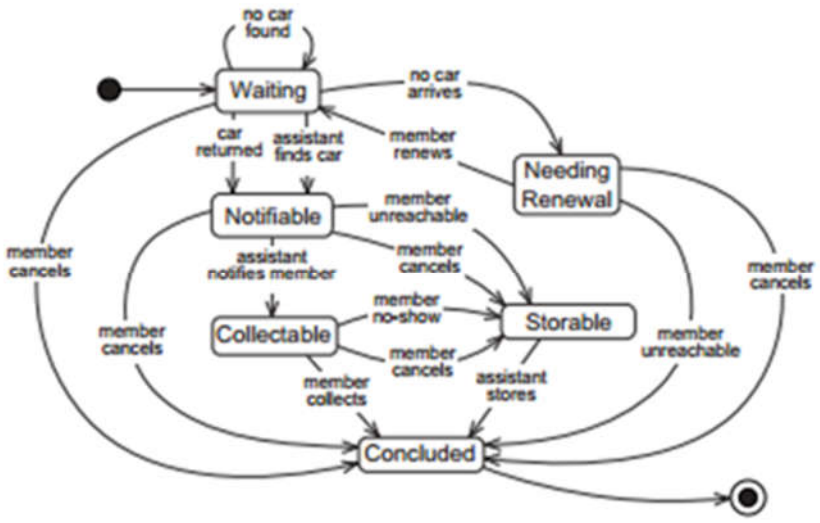
Kiểm tra lại toàn bộ vòng đời này sẽ chỉ ra cách thức một học sinh thay đổi từ một trạng thái này sang trạng thái khách qua những bước chuyển tương ứng với lệnh mà họ đưa cho hệ thống hoặc những câu hỏi mà họ hỏi hệ thống.

Đảm bảo rằng mô hình trạng thái đã hoàn tất đúng đắn bằng cách xem xét một vài ngữ cảnh đặc trưng khi sinh viên làm trắc nghiệm.

Ví dụ iCoot:

Hình sau mô tả lược đồ trạng thái cho Reservation (đặt xe), lược đồ được dùng để mô hình hóa vòng đời phức tạp ủa nó.

Lược đồ trạng thái cho Reservation



Khi một Member đặt trước một CarModel qua internet, Reservation được khởi tạo ở trạng thái chờ (waiting) và cần phải được xử lý bởi một Assistant. Resesrvation nhận giá trị Notifable trong trường hợp Assistant tìm được một Car phù hợp chưa được đặt trong khu vực trưng bày hoặc trong bãi xe, hoặc một Customer nào đó trả xe lại. Trong trường hợp này, Car sẽ được chuyển sang khu vực đặt riêng.

Nếu không có Car nào trong thời điểm đặt Reservation tương ứng trong vòng 1 tuần, Reservation sẽ có trạng thái NeedingReewal: Member sẽ được liên lạc qua điện thoại hoặc gặp trực tiếp theo đó họ có thể hủy Reservation hoặc được tư vấn để đặt mới vào tuần khác. Nếu Member hủy đặt xe hoặc không liên hệ lại trong vòng ba ngày thì Reservation sẽ mang trạng thái Collectable ngược lại nó sẽ mang trạng thái Displayble (Car trước đó được chuyển đến khu vực đặt thì phải chuyển ngược lại về khu vực trưng bày).

Một khi Reservation là Collectable, Member phải lấy Car trong ba ngày: nếu họ lấy xe, Reservation ở trạng thái Concluded; ngược lại, Reservation ở trang thái Displayable.

Một khi Displayable Reservation Car được đặt lại vào khu vực trưng bày, Reservation cũng nhận trạng thái concluded.

Trong mọi thời điểm, Member có thể hủy Reservation qua internet, điện thoại hoặc gặp trực tiếp.

Hệ thống sẽ thông báo cho các Assistants tình trạng hiện thời của Reservation theo đó họ có thể có những thao tác tương ứng.

LAB 5. THIẾT KẾ HỆ THỐNG

Trong bài thực hành này, chúng ta sẽ thiết kế kiến trúc một hệ thống AQS. Điều này liên quan đến các quyết định sử dụng thiết bị máy móc, các subnode chạy trên mỗi máy, các gói kèm theo cần phân chia thành các lớp mang tính logic và các thành phần con nếu tồn tại. Chúng ta cũng cần xem xét các vấn đề xảy ra đồng thời, các công nghệ được sử dụng để thực thi hệ thống và cách tạo ra một hệ thống bảo mật. Việc bảo mật bao gồm việc xây dựng hệ thống an toàn trước sự tấn công của hacker; mang tính riêng tư để đối phó với việc bị lấy trộm và giải mã thông tin trên Internet; và bảo vệ các thiết bị máy tính của người dùng và server khỏi các các thiệt hại do phần mềm của chúng ta gây nên. Hãy luôn nhớ nâng cấp tài liệu thuật ngữ khi tái sử dụng bất cứ thuật ngữ nào hoặc giới thiệu bất cứ những thuật ngữ mới.

Mục tiêu: Nội dung bài thực hành gồm các mục tiêu sau:

- Xây dựng một hệ thống có tổ chức
- Lựa chọn các công nghệ thích hợp
- Lên kế hoạch sử dụng các tiến trình và tiểu trình hợp lý
- Mô hình hóa các tiến trình thành các subnode
- Phân rã một hệ thống thành các tầng
- Gom nhóm các lớp liên quan thành các gói (package)
- Thiết kế bảo mật
- Thiết kế song song
- Vẽ một mô hình triển khai UML

5.1 Lựa chọn hệ thống có tổ chức

5.1.1 Số tầng hệ thống

Phần này chúng ta sẽ lựa chọn số tầng phù hợp cho hệ thống phần mềm, có thể là 1 tầng, 2 tầng, 3 tầng hay nhiều hơn tùy thuộc yêu cầu thực tế của hệ thống phần mềm. Lưu ý, nên chọn ít nhất 2 tầng bởi vì các yêu cầu hệ thống đầu vào đã chỉ ra rằng:

- Câu hỏi cần phải truy vấn dựa theo yêu cầu
- Câu trả lời cần phải được submit ngay khi được đưa ra

Hay nói cách khác, chúng ta không thể thực thi một hệ thống 1 tầng với tất cả câu hỏi và câu trả lời được cung cấp trên CD và câu trả lời thu thập qua email. Trong bài Lab này, chúng ta sẽ thực hiện mô hình 3 tầng.

5.1.2 Thiết bị máy móc

Sau khi định rõ hệ thống phần mềm được xây dựng dựa trên bao nhiêu tầng, chúng ta cũng phải liệt kê các thiết bị máy móc (phần cứng) liên quan để xây dựng và triển khai hệ thống phần mềm. Vì các yêu cầu hệ thống không có giới hạn rõ ràng, vì vậy chúng ta nên tránh đưa ra các hạn chế liên quan đến các dạng phần cứng cần có (ví dụ: PC, trạm làm việc, mainframe). Để đơn giản hóa, hãy bỏ qua các vấn đề phức tạp như cân bằng tải, sự lặp lại, fail-over, sự dư thừa và các backup hệ thống.

5.1.3 Xây dựng mô hình đơn giản

Vẽ một hình mô hình triển khai đơn giản chỉ chưa các node thiết bị – mỗi node thể hiện một máy tính vật lý trong một cấu hình đặc trưng của hệ thống.

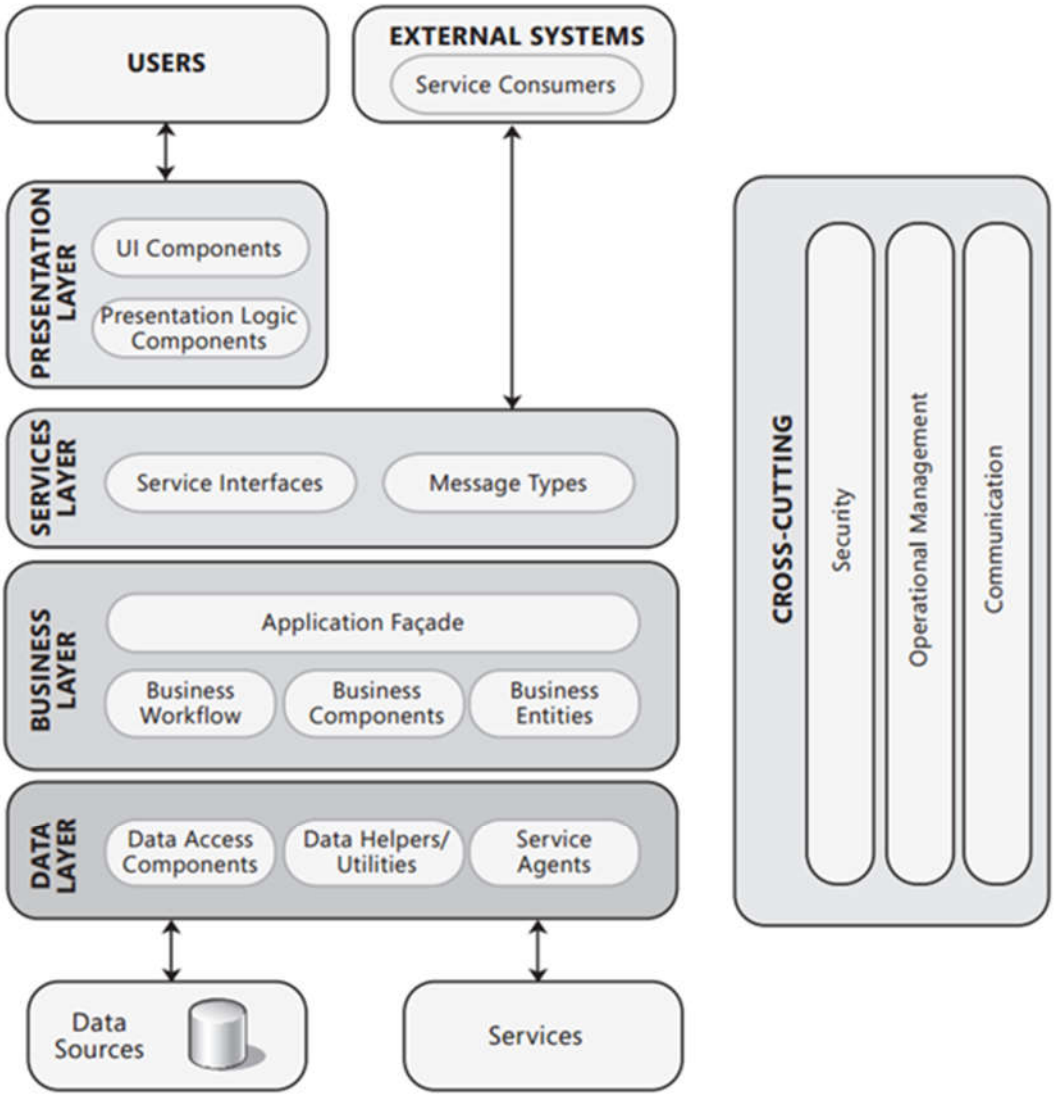
5.2 Lựa chọn công nghệ

Ở bước này, chúng ta cần lựa chọn các công nghệ, công cụ và nền tảng phù hợp với việc phát triển và xây dựng chương trình phần mềm. Lưu ý là không đặt ra các hạn chế bất hợp lý ở máy tính người dùng (ví dụ: sự phức tạp của việc cài đặt hay sự phức tạp của việc kết nối tới tầng tiếp theo). Lưu lại các công nghệ sử dụng, và lý do vì sao lựa chọn các công nghệ đó kèm theo các giao thức giữa các tầng nếu cần thiết.

Trong bài Lab này, chúng ta sẽ xây dựng hệ thống phần mềm chạy trên hệ điều hành Windows 7, sử dụng bộ công cụ Visual Studio .NET 2013, ngôn ngữ lập trình C# và hệ thống cơ sở dữ liệu Microsoft SQL 2008. Các phiên bản trở về sau của các phần mềm, công cụ, hệ điều hành trên đều được chấp nhận.

5.3 Lựa chọn các tầng hệ thống

Với việc sử dụng mô hình 3 tầng, chúng ta có mô hình như sau:



Có 3 tầng chính: Data Access Layer, Business Logic Layer và Presentation Layer.

5.4 Cân nhắc việc sử dụng các tiến trình và tiểu trình

Chúng ta phải lựa chọn các tiến trình cần thiết để thực thi ở mỗi node. Thêm các tiến trình khi chúng ta thiết kế hay phát hiện giống như các subnode trong môi trường thực thi trên mô hình triển khai. Tiếp đến, chúng ta đưa ra các quyết định tiến trình nào cần sử dụng với các tiểu trình nào một cách rõ ràng hoặc, bất cứ tiến trình nào sử dụng tiểu trình có ảnh hưởng phụ với công nghệ nên dùng. Lưu lại ghi nhớ các tiểu trình và các tầng cần được đảm bảo an toàn.

5.5 Đóng gói các lớp

Gom nhóm các lớp thành các package có liên quan với. Chúng ta sẽ không biết chính xác các lớp nào cần gom nhóm với nhau, nhưng chúng ta có thể gom nhóm dựa trên cảm nhận của mình. Đặt mỗi package ở vị trí thích hợp trên mô hình triển khai.

5.6 Hoàn thiện mô hình triển khai

Ở bước này, chúng ta triển khai hệ thống và các thêm các tài nguyên cần thiết vào mô hình triển khai như thư mục web server, các tập tin lưu trữ, … Ở mô hình, hiển thị các sự phụ thuộc giữa các node, subnode, package và các tài nguyên cho thích hợp.

5.7 Thực thi các tác vụ đồng thời

Lưu lại các ghi chép về việc kiểm soát các truy cập đồng thời và cách xử lý các truy cập đồng thời này.

5.8 Bảo mật hệ thống

Ở phần này, chúng ta phải hiểu các bước cần thiết để bảo vệ server trước sự tấn công của hacker. Ví dụ, một hacker có thể muốn đánh cắp các tài sản sở hữu trí tuệ (mã nguồn, hình ảnh, video, database, …) hoặc phá hoại server (làm chậm, nghẽn, hư mã nguồn, hư chương trình phần mềm, …). Do đó, cần ghi lại tất cả phát hiện vào sổ tay. Ghi lại các cách ngăn chặn sự rò rĩ thông tin nhạy cảm trên Internet và các phương pháp an toàn bảo mật cho các máy tính của người dùng, các hệ thống thiết bị trên server, đường truyền Internet và phần mềm ứng dụng của chúng ta.

LAB 6. THIẾT KẾ CHI TIẾT

Hiện tại bạn đã đã định nghĩa các công nghệ AQS và các lớp trong thiết kế hệ thống, bạn có thể chuyển sự chú ý của mình vào bên trong các lớp và mối quan hệ giữa chúng. Bài tập này được chia thành 3 bước:

- Chuyển các lớp phân tích vào trong các lớp thiết kế(business logic) , thêm các trường như bạn đã có; bước này được dẫn chứng sử dụng sơ đồ lớp.
- chuyển các lớp thiết kế và các trường vào một lược đồ cơ sở dữ liệu và miêu tả các bảng sử dụng các ký hiệu minh bạch
- định nghĩa các dịch vụ (kinh doanh), thiết kế các lớp dịch vụ và kiểm tra sự tương tác giữa client, các lớp dịch vụ và business logic. bước này được dẫn chứng bằng việc sử dụng các sơ đồ lớp và các sơ đồ tuần tự. nó cũng giúp bạn tìm ra các thông điệp cho các lớp business logic.

Như vậy, bạn nên cập nhật các thuật ngữ tại các vị trí thích hợp.

Mục đích: Sau khi hoàn thành bài tập này, bạn sẽ có thể:

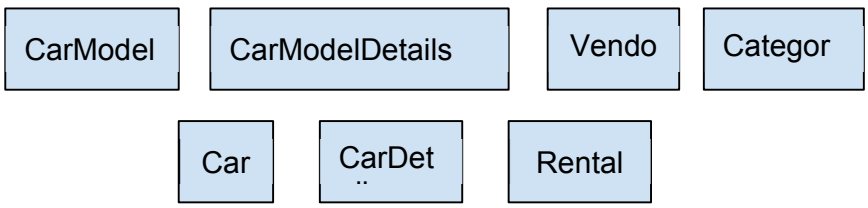
- Chuyển các lớp phần tích vào các lớp thiết kế, kể cả các lớp mở rộng, các lớp dư thừa, các thuộc tính, các thành phần, các sự kết hợp, các lớp kết hợp, kế thừ và tất cả các xác định khác.
- Định nghĩa lược đồ cơ sở dữ liệu cơ bản như một sự mapping trực tiếp của các lớp thiết kế.
- Tìm ra các dịch vụ kinh doanh (business services)
- Phân loại các dịch vụ kinh doanh và giới thiệu các lớp hỗ trợ cần thiết
- Chỉ ra đối tượng và lớp tương tác sử dụng sờ đồ tuần tự

6.1 Thiết kế logic kinh doanh (business logic)

6.1.1 Chọn các lớp và các trường

Sử dụng mô hình các lớp phân tích như một hướng dẫn, chọn các lớp thiết kế bạn cần. Thêm chúng vào một sơ đồ lớp mới.

Ví dụ ***iCoot***: Một số lớp được xác định trong mô hình kinh doanh như sau: ***CarModel, CarModealDetails, Vendor, Category, Car, CarDetails, Rental,...***



Trên sơ đồ lớp của bạn, với mỗi lớp, thêm bất kỳ trường mà bạn cần để lưu giữ các thuộc tính. Nhớ rằng mỗi lớp thực thể nên có ít nhất một xác định toàn cục. Đừng quên xác định tầm nhìn của mỗi trường (sử dụng +, -, ~ hay #).

Ví dụ iCoot: các trường (thuộc tính) được xác định trên các đối tượng như sau:

CarModel:

- -name:String. Là tên duy nhất cho mẫu xe
- -price: int. Giá thuê xe mỗi ngày, tính bằng cents.

CarModelDetails:

- -engineSize: int. Công suất của động cơ, tính bằng centimet khối.
- -description: String. Mô tả về CarModel
- -advert: String. Tên của một quảng cáo cho CarModel
- -poster: String. Tên của poster cho CarModel

Vendor:

- -name: String. Tên của người cho thuê

Category:

- -name: String. Tên của danh mục xe

Car:

- -traveled: int. Khoảng cách, tính bằng kilomet, số kilomet mà xe đã đi.
- -dateLost: Date. Ngày xe được báo cáo là thiếu, null nếu không.

CarDetails:

- -barCode: String. Mã code, được đính kèm trên kính ôto.
- -numberPlate: String. Số giấy phép của xe
- -vin:String. Số xác định xe (số khung), duy nhất cho mỗi xe.

Rental:

- -number: String. Số duy nhất cho mỗi Rental.
- -startDate: Date: Ngày thuê xe
- -dueDate: Date. Ngày kết thúc thuê xe
- -totalAmount:int. Tổng số tiền phải trả

...

Nếu bạn tìm ra một thông tin mà cần cho việc lưu trữ cho một lớp ban đầu của các thực thể, thêm một trường của lớp vào lớp. Các trường lớp được gạch chân để phân biệt chúng với các trường khác.

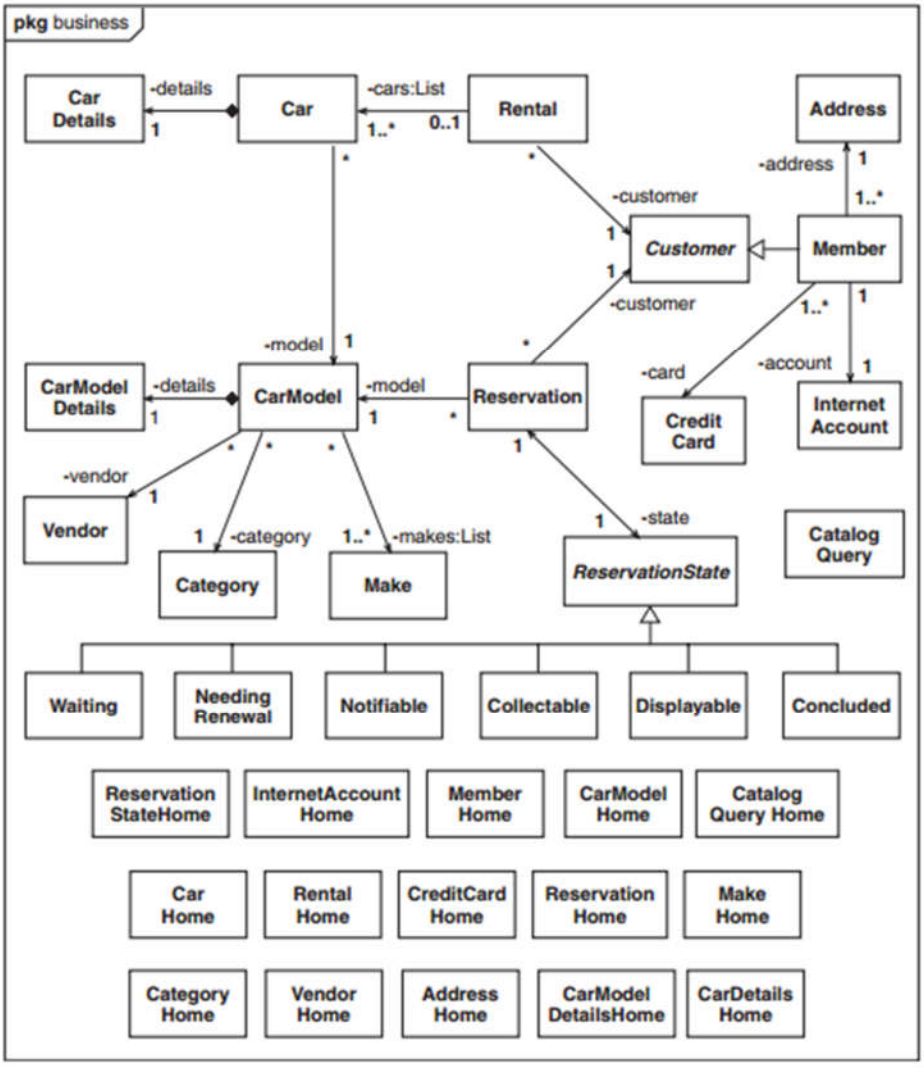
6.1.2 Thêm các mối quan hệ

Thêm các mối quan hệ giữa các lớp (kế thừa, kết hợp, thành phần và quan hệ ngang hàng) và tính phức tạp tới sơ đồ lớp của bạn. Với quan hệ thành phần và kết hợp, bạn có thể giữ lại các ký hiệu hình thoi như tài liệu hướng dẫn thêm.

Mọi mối quan hệ không phải kế thừa nên hiển thị các hướng của quan hệ với một hay 2 mũi tên ở các đầu.

Tên và tầm nhìn của các trường được dùng để ghi lại các mối quan hệ có thể được xác định tới các đầu mũi tên tương ứng; với các mối quan hệ thực thi như bộ sưu tập, bạn nên thêm tên lớp sưu tập là tốt nhất.

Ví dụ iCoot: sơ đồ quan hệ giữa các lớp như sau:



6.1.3 Cập nhật thuật ngữ

Thêm các mô tả của các lớp business logic tới thuật ngữ. Bạn có thể đã có sẵn các mục thuật ngữ cho các lớp, ví dụ như kết quả của phân tích, vì vậy bạn có thể chỉ cần xác định rằng các các mục cũng phù hợp với các lớp thiết kế.

6.2 Thiết kế lược đồ cơ sở dữ liệu

6.2.1 Đồng ý một ký hiệu cho lược đồ của bạn

Đồng ý một ký hiệu trong nhóm của bạn cho việc mô tả một lược đồ. Ví dụ, bạn có thể chỉ ra một bảng dưới dạng text như sau:

CARMODEL
+ID: INTEGER, NAME:VARCHAR(256), >MAKEID: INTEGER

Trong ký hiệu này, + xác định một khóa chính và > xác định một khóa ngoại

6.2.2 Quyết định các kiểu dữ liệu SQL

Chọn một số ít các kiểu dữ liệu SQL để sử dụng cho các cột dữ liệu của bạn. Ví dụ, bạn nên thực hiện như sau:

- INTEGER: một số 32 bit
- DECIMAL: một số thực, với độ chính xác xấp xỉ (chỉ sử dụng decimals khi cần thiết, INTEGER thường được sử dụng nhiều hơn)
- VARCHAR(X): một chuỗi với X ký tự
- BOOLEAN: một 1-bit dữ liệu, 0 cho FALSE, 1 cho TRUE
- DATE: một ngày trong lịch Gregorian
- TIME: một sự kết hợp cụ thể của giờ, phút, giây bên trong ngày Gregorian
- TIMESTAMP: một kết hợp giữa DATE và TIME

Các kiểu tổng quát SQL-92 thay đổi từ một cơ sở dữ liệu thực thi tới một cơ sở dữ liệu khác, nhưng nó là hợp lý để đặt giả thiết tính tương đương giữa các kiểu là có thật.

6.2.3 Giới thiệu thực thể Bảng

Thêm một bảng vào lược đồ của bạn ứng với mỗi thực thể kinh doanh rõ ràng. Sử dụng xác định toàn diện như khóa chính. Thêm các khóa ngoại cho thực thể rõ ràng trong mối quan hệ một-nhiều và một-một.

6.2.4 Giới thiệu liên kết các bảng

Thêm các liên kết bảng cho các thực thể với quan hệ nhiều-nhiều và các lớp liên kết ngang hàng (trực tiếp).

Ví dụ iCoot: Chi tiết các Bảng trong cơ sở dữ liệu như sau:

Address(id:integer,house:varchar(99),street:varchar(99),county:varchar(99),
postcode:varchar(99))
Car(id:integer,travelled:integer,datelost:date,cardetailsid:integer)
Card(id:integer,type:varchar(99),number:varchar(99))
Cardetails(id:integer,barcode:varchar(99),numberplate:varchar(99),vin:varchar(99))
Carmodel(id:integer,name:varchar(99),price:integer,carmodeldetailsid:integer,
categoryid:integer,vendorid:integer)

Carmodeldetails(id:integer,enginesize:varchar(99),description:varchar(256),advert:varchar(99),poster:varchar(99))
Category (id:integer,name:varchar(99))
Collectablereservation(reservationid:integer,datenotified:date)
Concludedreservation(reservationid:integer,reason:varchar(99))
Customer(id:integer,name:varchar(99),phone:varchar(99),amountdue:integer)
Displayablereservation(reservationid:integer,reason:varchar(99))
Internetaccount(id:integer,password:varchar(99),sessionid:integer)
Make(id:integer,name:varchar(99))
Makecarmodel(carmodelid:integer,makeid:integer)
Member(id:integer,number:varchar(99),ingoodstanding:boolean,cardid:integer,addressid:integer)
Needingrenewalreservation(reservationid:integer,daterenewalneeded:date)
Nonmember(id:integer,driverslicense:varchar(99))
Notifiablereservation(reservationid:integer,dateputaside:date)
Rental(id:integer,number:varchar(99),startdate:date,duedate:date,totalamount:integer)
Rentalcar (rentalid:integer,carid:integer)
Reservation(id:integer,number:varchar(99),timestamp:timestamp,customerid:integer,carmodelid:integer)
Vendor(id:integer,name:varchar(99))
Waitingreservation(reservationid:integer,lastrenewedddate:date)

6.2.5 Cập nhật thuật ngữ

Trong thuật ngữ của bạn, thêm tham khảo tới các bảng trong cơ sở dữ liệu

6.3 Thiết kế các dịch vụ kinh doanh (Designing the business services)

6.3.1 Xác định các dịch vụ kinh doanh

Với liên kết tới các phác thảo GUI, xác định các dịch vụ ảo phải được cung cấp cho tầng client bởi tầng middle và danh sách các dịch vụ trong bài tập của dự án. Mỗi dịch vụ kinh doanh trong danh sách có thể đơn giản như "Thuê một chiếc xe"

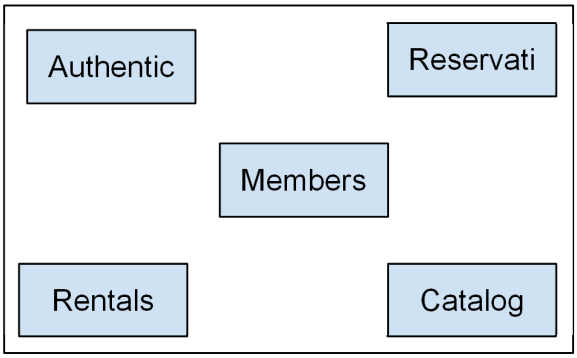
Nhóm các dịch vụ kinh doanh vào các thông điệp trên các lớp dịch vụ cụ thể. Với mỗi thông điệp, bạn sẽ cần xác định các tham số giữa client và các kết quả trả về từ server nếu có.

Bạn hoàn toàn tự do trong việc xác định chính xác các luồng thông tin trước và sau như thế nào. Tuy nhiên, bạn nên chắc chắn rằng:

- Client không chuyển thông tin tới server nếu server đã có thông tin đó (trong cơ sở dữ liệu của nó)
- Client không đưa ra các mẫu thông tin tương tự nhau 2 lần

Các xác định toàn diện có thể giúp bạn ở đây. Bạn có thể thoải mái các luật để xử lý chứng thực các khóa session, vì kịch bản này cón thể dư thừa thông tin.

Ví dụ iCoot, các dịch vụ được cung cấp cho chương trình và sơ đồ như sau:



Các thông điệp được đưa ra trên các lớp như sau:

Authentication Controller:

- **logon:** Từ trang chủ, với mã số thành viên, mật khẩu và các tham số đủ mạnh trên việc đăng nhập thành công, trả về trang thành viên.
- **logoff**: Từ trang thành viên, đăng xuất khỏi thành viên Member.

Reservation Controller:

- **reserve:** Từ trang chi tiết, với tham số là một **CarModel** và trả về trang **confirmReserve**
- **confirmReserve:** Từ trang **confirmReserve**, đặt lại **CarModel** đã được xác định và trả về trang xác nhận
- **ok:** Từ trang xác nhận, trả về trang chi tiết
- **reservations**: Từ trang thành viên, trở về trang đặt (xe), chứa các **Reservations** của thành viên hiện tại
- **cancel:** Từ trang đặt (xe), với tham số là mã của Reservation và trả về trang **confirmCancel**.
- **confirmCancel:** Từ trang **confirmCancel**, hủy các **Reservation** đã đặt trước đó và trở về trang đặt (xe).

Membership Controller:

- **membership:** Từ trang thành viên, trở về trang quyền thành viên (membership page), chứa chi tiết quyền của thành viên hiện tại.
- **password:** Từ trang thành viên, trở về trang mật khẩu
- **changePassword:** Từ trang thành viên, với tham số là một mật khẩu cũ và một mật khẩu mới và trả về trang **confirmChange**.
- **confirmChange:** Từ trang **confirmChange**, thiết lập mật khẩu mới cho thành viên nếu cung cấp đúng mật khẩu cũ.

Rentals Controller:

- **rentals:** Từ trang thành viên, trở về trang cho thuê, chứa các Rentals của thành viên hiện tại.

Catalog Controller:

- **index:** Từ trang thành viên hay không là thành viên, trả về trang chủ chứa các tiêu đề để chọn
- **browse:** Từ trang chủ, với tham số là chỉ số tiêu đề và trả về trang chứa các kết quả phù hợp với CarModels.
- **search:** Từ trang thành viên hay không là thành viên, trả về trang tìm kiếm chứa tất cả **Category**, **Make** và kích thước động cơ để chọn
- **query:** Từ trang tìm kiếm, với tham số là mã Category , mã Make và kích thước động cơ và trả về trang kết quả chứa các CarModels phù hợp.
- **details:** từ trang kết quả, với tham số là mã CarModel và trả về trang chi tiết chứa thông tin chi tiết của CarModel.

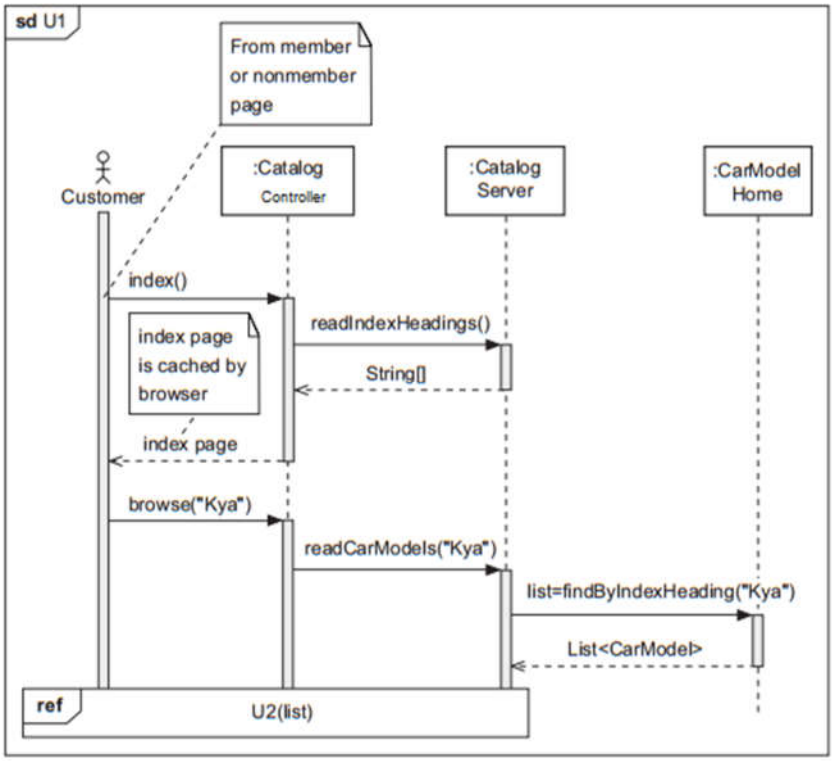
6.3.2 Vẽ các sơ đồ tuần tự

Với mỗi dịch vụ kinh doanh, vẽ một sơ đồ tuần tự để hiển thị luồng thông điệp từ GUI client tới các lớp server và trên các lớp business logic. Điều này sẽ phục vụ để xác định tính khả thi của kiến trúc.

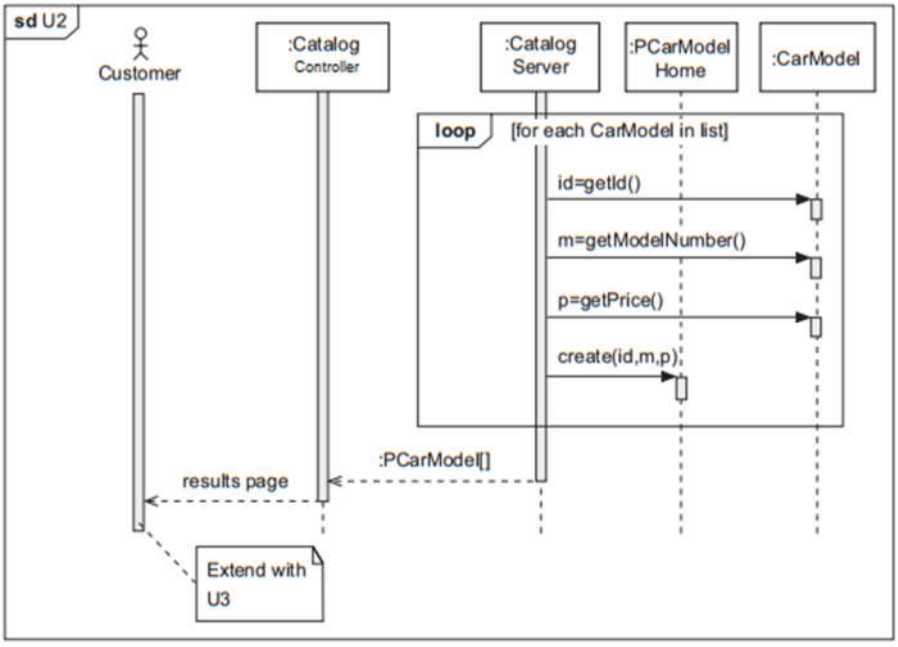
Để đơn giản, bạn nên cố gắng chỉ hiển thị các thông điệp được gởi giữa các đối tượng, hơn là các thông điệp được gởi bởi một đối tượng tới chính nó. Đừng hiển thị bất kỳ chi tiết nào của sự tương tác giữa business logic và các lớp dưới đó, mặc dù sự tương tác giữa các lớp business logic đôi khi là thích hợp.

Nếu bất kỳ sự tương tác là quá phức tạp để hiển thị trên các hình, bạn có thể xem xét thêm một ghi chú ngắn.

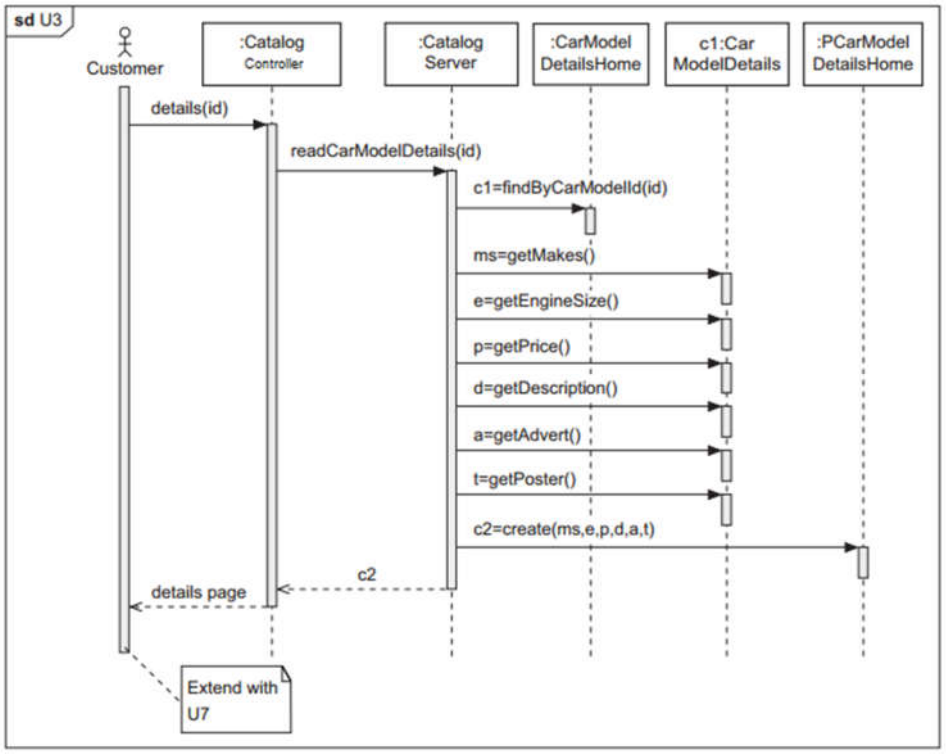
Ví dụ iCoot, các sơ đồ tuần tự như sau:



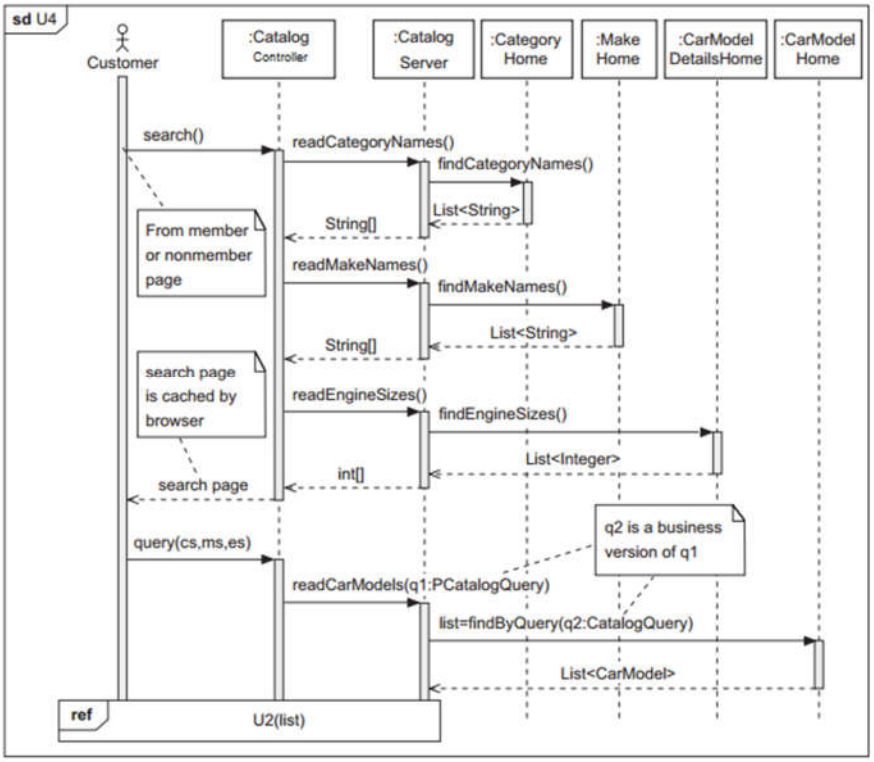
Sơ đồ tuần tự 1: Browse Index



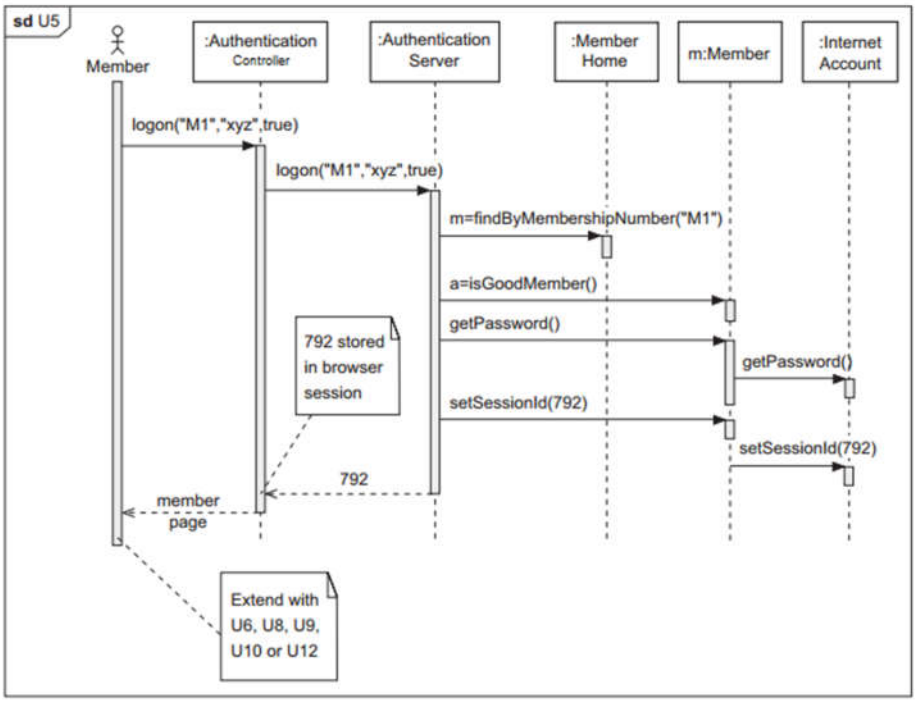
Sơ đồ tuần tự 2: View Results



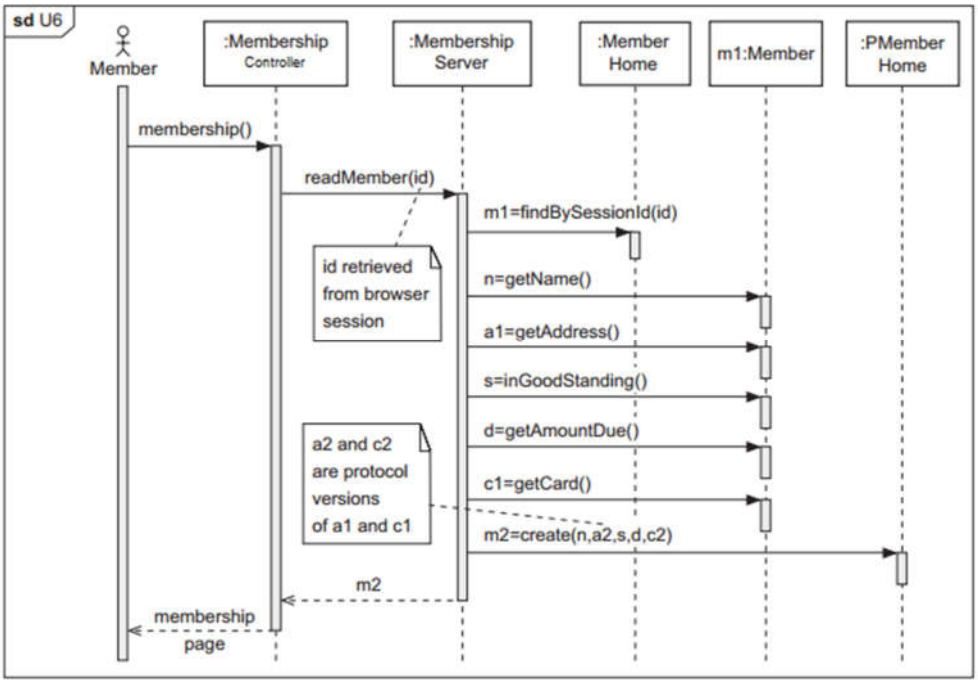
Sơ đồ tuần tự 3: View CarModel Details



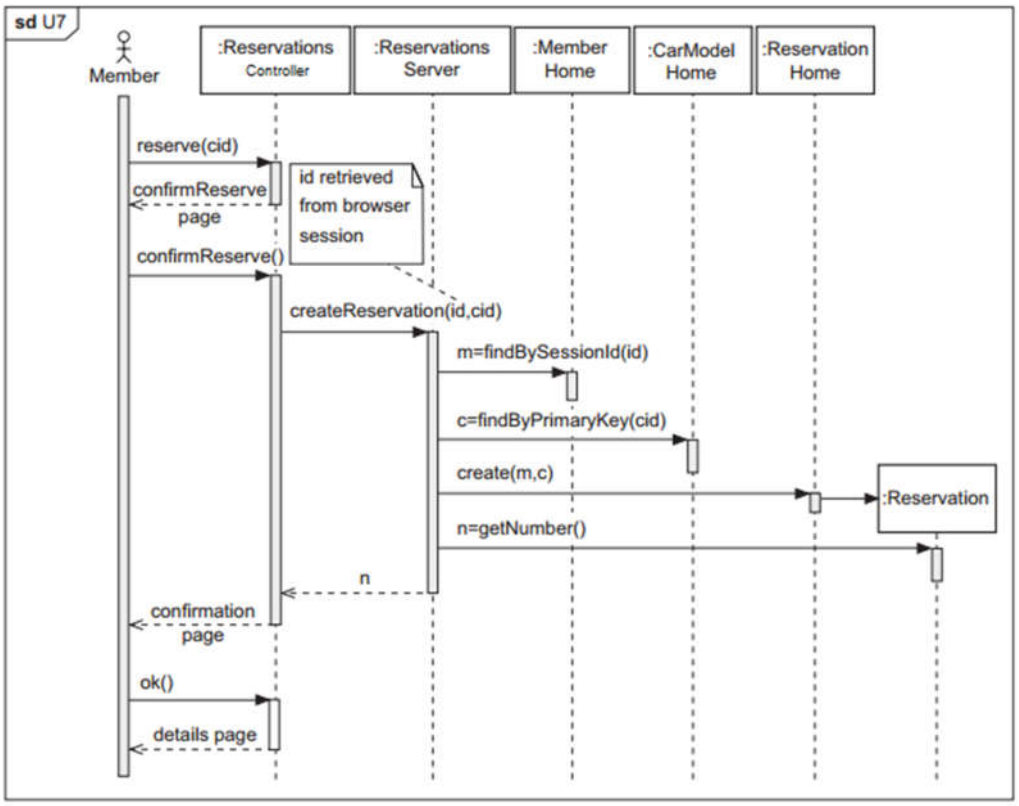
Sơ đồ tuần tự 4: Search



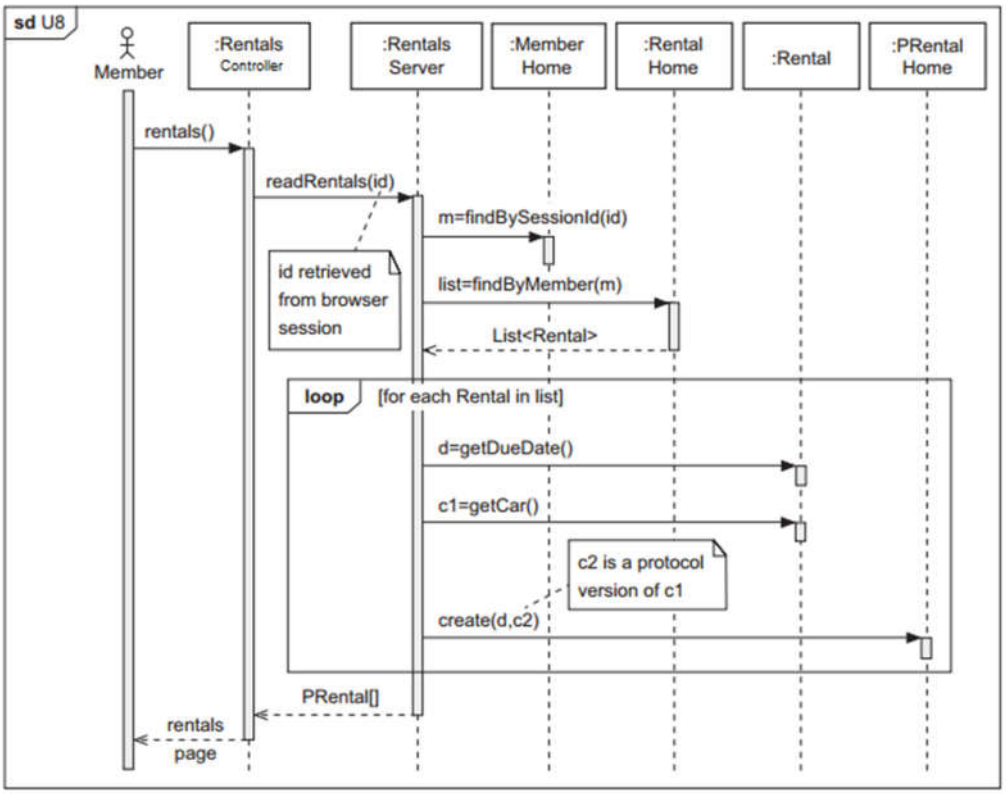
Sơ đồ tuần tự 5: Log On



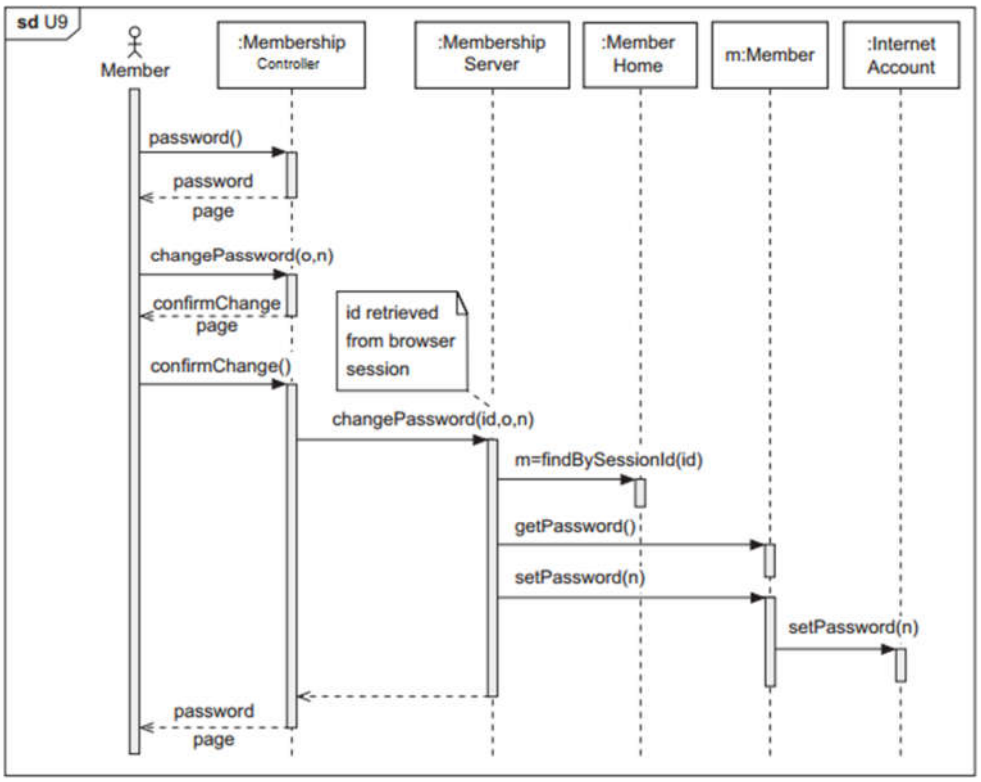
Sơ đồ tuần tự 6: View Member Details



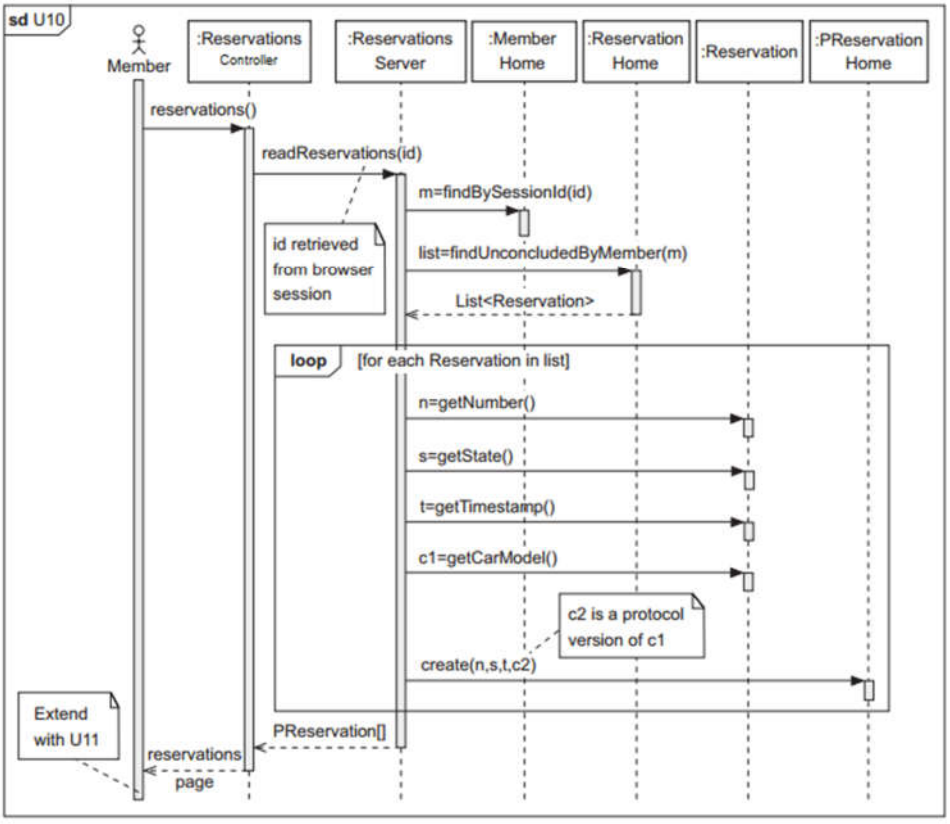
Sơ đồ tuần tự 7: Make Reservation



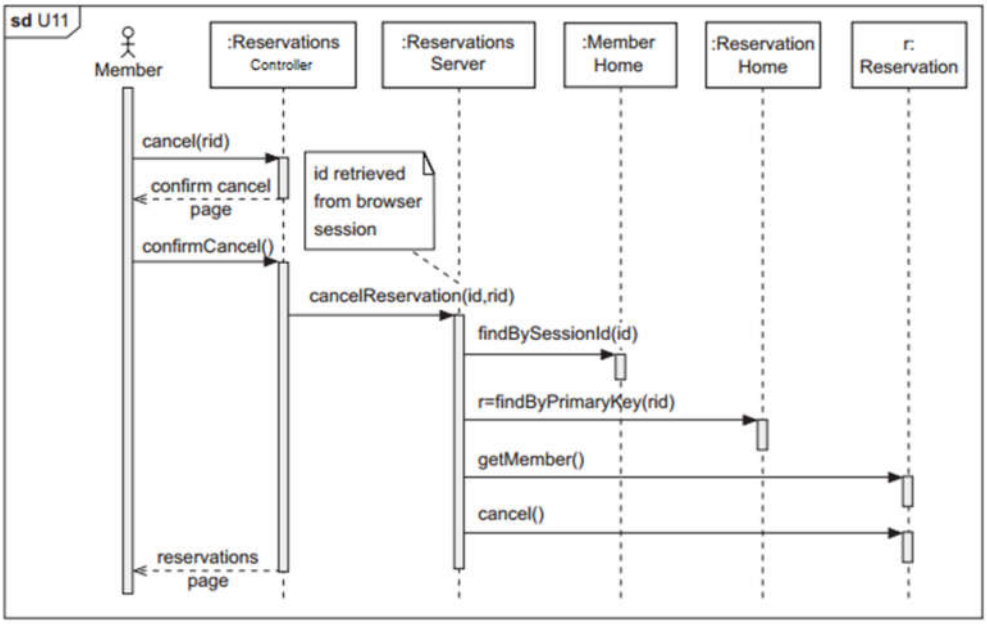
Sơ đồ tuần tự 8: View Rentals



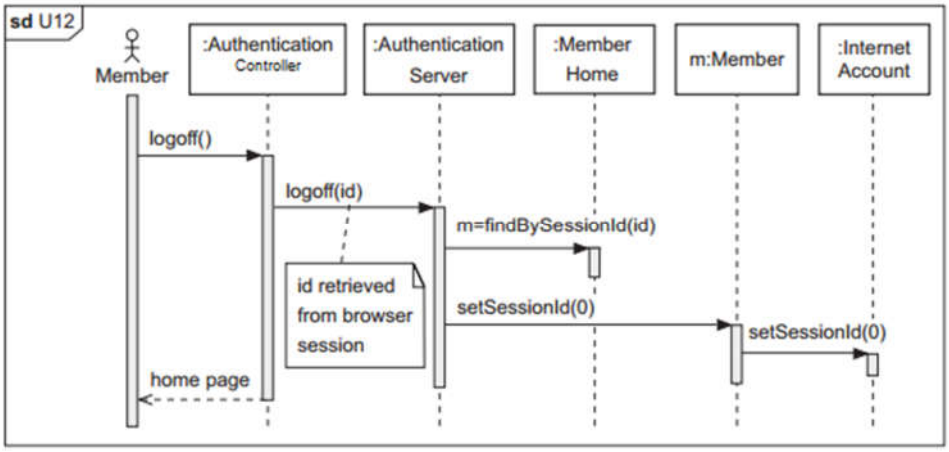
Sơ đồ tuần tự 9: Change Password



Sơ đồ tuần tự 10: View Reservations



Sơ đồ tuần tự 11: Cancel Reservation



Sơ đồ tuần tự 12: Log Off

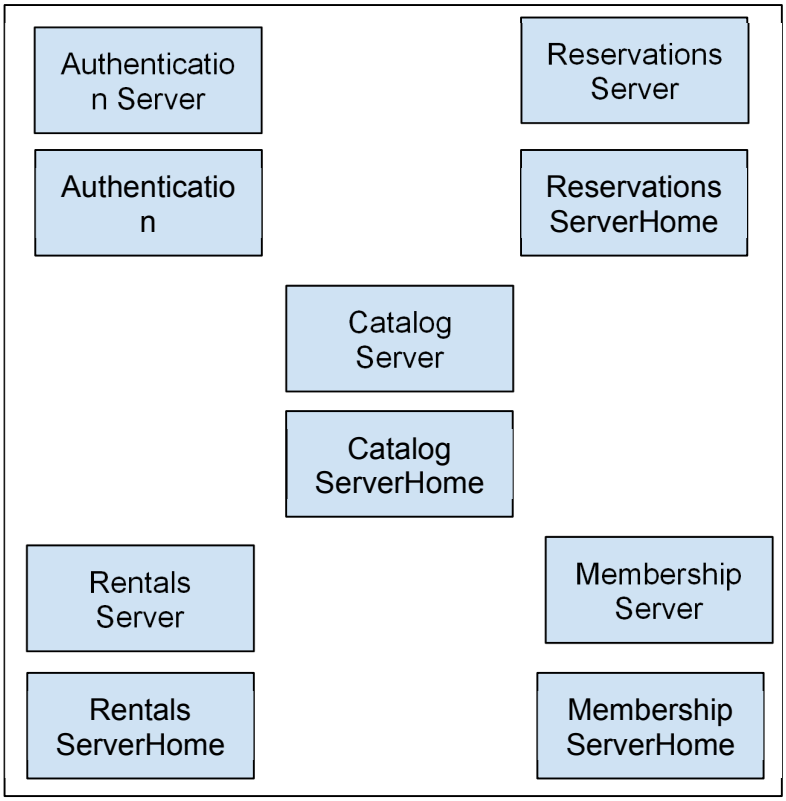
6.3.3 Danh sách Các thông điệp kinh doanh logic (Business Logic)

Trong sách bài tập của dự án, danh sách các tin nhắn mà bạn đã khám phá cho các lớp business logic, hoàn thiện với các tham số và kiểu trả về. Bạn có thể đọc chúng từ các sơ đồ tuần tự hoàn thiện của bạn.

Nếu bạn đã sử dụng công cụ CASE, các thông điệp có thể được thêm trực tiếp tới thiết kế sơ đồ lớp của bạn. Bạn thường không làm việc này trên giấy.

một khi bạn hoàn thành quá trình này, bạn nên có một tập tối thiểu hoàn chỉnh của các lớp server và các lớp business logic với các trường và thông điệp. Các lớp có thể được lấy từ giai đoạn mô tả chức năng trước đó.

Ví dụ **iCoot**, các dịch vụ được xác định như sau:



AuthenticationServer:

- **+logon(n:String, p:String, s:boolean):long**. Đăng nhập thành viên với mã số n, mật khẩu p và không tồn tại bất kỳ phiên làm việc s nào hiện tại.
- **+logoff(i:int)**. Đăng xuất thành viên khỏi phiên làm việc i

CatalogServer

- **+readCategoryNames(): String[].** Đọc tên của tất cả các danh mục(Category)
- **+readMakeNames(): String[].** Đọc tất cả tên **Makes**
- **+readEngineSizes(): int[].** Đọc kích thước động cơ cho tất cả các CarModel
- **+readIndexHeadings(): String[].** Đọc các chỉ số từ tất cả các CarModel và Make
- **+readCarModels(h: String): PCarModel[].** Đọc tất cả CarModels phù hợp với chỉ số h.
- **+readCarModelDetails(i:int): PCarModelDetails.** Đọc chi tiết của CarModel với id i
- **+readCarModels(q: PCatalogQuery): PCarModel[].** Đọc tất cả CarModels phù hợp với truy vấn q.

MembershipServer

- **+readMember(i:int): PMember.** Xác định Member với phiên làm việc i
- **+changePassword(i:int, o: String, n: String).** Thay đổi mật khẩu cho Member với phiên làm việc i, sử dụng mật khẩu cũ o và mật khẩu mới n.

RentalsServer

- **+readRentals(i:int): PRental[].** Đọc tất cả các Rental của Member với phiên làm việc i

ReservationServer

- **+readReservations(i:int): PReservation[].** Đọc tất cả các Reservation của Member với phiên làm việc i.
- **+createReservation(i:int, c:int).** Tạo một Reservation cho Member với phiên làm việc i và CarModel c.
- **+cancelReservation(i:int, r: int).** Hủy một Reservation mã số r của Member với phiên làm việc i, mà Reservation phù hợp với Member.

6.3.4 Cập nhật thuật ngữ

Trong thuật ngữ của bạn, thêm tham khảo tới các lớp server

LAB 7. ĐẶC TẢ CÁC LỚP

Để hoàn thành việc phát triển phần mềm, chúng ta phải bước qua giai đoạn đặc tả kỹ thuật. Công việc trong giai đoạn này không mong chờ sẽ được thực hiện một cách chỉnh chu và đầy đủ, tuy nhiên những điều mà ta bắt buộc phải quan tâm đó là nghĩ về các precondition, postcondition, invariant theo cách nó đươc áp dụng cho các lớp quen thuộc. Ngoài ra, chúng ta cũng nên nghĩ cẩn thận về những biệt lệ (exceptions) đã được kiểm soát hoặc chưa được kiểm soát để báo hiệu các lỗi (nằm trong vùng kiểm soát) đến người dùng phần mềm.

Sau khi kết thúc bài lab này, sinh viên có thể thực hiện:

- Xác định các precondition và postcondition cho các thông điệp
- Xác định các invariant của lớp
- Lựa chọn được cách sử dụng biệt lệ

7.1 Chọn lựa các lớp

Lựa chọn một lớp ở tầng ứng dụng (Presentation/Application Layer - UI Layer). Tốt nhất, ta nên chọn các lớp đủ lớn và phức tạp để thực hiện. Các lớp ở tầng này là một ứng viên tốt cho việc đặc tả bởi vì các lớp cần phải tự bảo vệ chính nó từ Client bằng việc có một thiết kế khế ước rõ ràng và chặt chẽ cho việc thực hiện.

Ngoài ra, chọn một lớp ở tầng Business cũng là một phương án tốt. Các lớp ở tầng Business thường chứa một tập các invariant đa dạng. Một lần nữa, lớp ta chọn nên là lớp phức tạp để có thể có những mô tả gì đó rõ ràng về nó.

Ví dụ trong phần này ta chọn các lớp *ReservationUI* ở tầng ứng dụng với các phương thức cho phép người dùng đặt hủy các đơn đặt hàng. Ở tầng Business, ta xem xét lớp *Member*, nơi thông tin của các thành viên được tạo lưu dưới các thuộc tính của lớp.

7.2 Thêm các invariant

Lấy hai tờ giấy, mỗi tờ cho một lớp đã chọn. Viết tên của mỗi lớp ở phía trên cùng của tờ giấy và sau đó liệt kê các invariants bất kỳ bạn có thể nghĩ ra(nhớ lại rằng các invariants chỉ ra trách nhiệm bắt buộc cho cả bên mua và bên bán)

Khi đã tìm thấy bạn dễ dàng để tạo invariant cho các thuộc tính của một đối tượng (nghĩa là các giá trị có sẵn bên ngoài thông qua các thiết lập get và set)

Bạn có thể viết các invariant của bạn bằng việc sử dụng bất kỳ sự kết hợp nào của ngôn ngữ tự nhiên, OCL hoặc Eiffel - miễn là nó rõ ràng và phù hợp với mục đích của mình

Ví dụ:

Ở lớp *ReservationUI* ta có thể thấy là nó không có các Invariants bởi vì nó là đối tượng không trạng thái

Trong khi đó ở lớp Member ta có thể xác định một loạt các Invariant như id thì không thay đổi sau khi khởi tạo và đương nhiên là khác rỗng, tương tư như vậy cho các thuộc tính *number*, *internetAccount* và *address*. Ngoài chiều dài của number này phải khác không.

```
Invariants:
   /*
    * Values named in invariants, preconditions and postconditions
    * are attributes, with a getter and an optional setter. Each
    * invariant is an extra precondition for the corresponding setter
    * and an extra postcondition for the corresponding getter.
    */
   id is fixed after creation
   id != 0
   number != null
   number.size() != 0
   internetAccount != null
   address != null
```

7.3 Thêm các precondition

Viết xuống các thông điệp của lớp, liệt kê danh sách các precondition. Nhớ rằng một precondition là nghĩa vụ ràng buộc của người mua.

Precondition có thể liên quan đến các tham số, các thuộc tính công khai hoặc kết quả của một phương thức boolean (ví dụ trong *cheksOutOk(aParam)*)

Bạn có thể giả định rằng một invariant tự động là một precondition

Ví dụ: tạo một đặt hàng mới cho một thành viên nào đó với định danh của phiên làm việc là i và mẫu xe có định danh là c. Phương thức tạo đặt hàng *createReservation(int i, int c)* trong lớp *ReservationUI* có các precondition nhằm đảm bảo yêu cầu thành viên là hợp lệ và mẫu xe tồn tại trong cơ sở dữ liệu:

```
Preconditions:
   i != 0
   For mem = MemberHome.getInstance().findBySessionId(i),
      mem != null
      mem.isingoodStanding()
      mem.getAmountDue() == 0
   c != 0
   CarModelHome.getInstance().findByPrimaryKey(i) != null
```

Đối với việc hủy đặt hàng của thành viên có định danh r cùng với phiên làm việc i có precondition trong phương thức *cancelReservation(int i, int r)* nhằm đảm bảo yêu

cầu thông tin thanh viên dùng cho việc hủy đặt hàng sẽ phải nằm trong danh sách những người đã đặt hàng trước đó:

```
Preconditions:
   i != 0
   For mem = MemberHome.getInstance().findBySessionId(i)
      mem != null
   r != 0
   For res = ReservationHome.getInstance().findByPrimaryKey(r)
      res != null
      res.getMember() == mem
```

Xem tất cả các đặt hàng của khách hàng với định danh i với phương thức *readReservations(int i)* có precondition nhằm yêu cầu thành viên phải đăng nhập vào hệ thống tức là tồn tại phiên làm việc của thành viên muốn xem tất cả các đơn đặt hàng của mình:

```
Preconditions:
   i != 0
   MemberHome.getInstance().findBySessionId(i) != null
```

Trong khi đó với việc thiết lập thông tin cho các thuộc tính trong lớp Member, ta có thể thấy rằng không có rõ ràng các precondition nào cần định nghĩa.

7.4 Thêm các postcondition

Thêm postcondition cho các thông điệp. Thông thường sẽ tham chiếu đến các giá trị phản hồi và các thuộc tính của đối tượng

Ví dụ: trong lớp *ReservationUI* ta xem xét thêm vào các postcondition cho các phương thức theo từng ngữ cảnh cụ thể.

Đối với phương thức *createReservation(int i, int c)* postcondition nhằm đảm bảo sau khi thực hiện xong một đặt hàng mới được tạo ra với thông tin của thành viên với định danh i và mẫu xe với định danh c.

Trong khi đó đối với phương thức *cancelReservations(int i, int r)*, postcondition được thêm vào nhằm đảm bảo đúng thành viên với đinh danh r đã thực hiện và lý do thực hiện:

```
Postconditions:
   For res = ReservationHome.getInstance().findByPrimaryKey(r)
      res.isConcluded()
      res.getReason().equals("Canceled by customer");
```

Đối với phương thức xem các đặt hàng *readReservations(int i)* thì postcondition sẽ đảm bảo kết quả trả về là danh sách những đặt hàng của khách hàng với định danh i và kết quả trả trả về phải được tạo trên mảng mới được gửi đến khách hàng:

```
Postconditions:
   result != null
   result contains all unconcluded reservations for Member
      with session identifier i
   result is a new array, exclusive to the client
```

Đối với các lớp Member, ta có thể thấy rằng postcondition được được định nghĩa nhằm đảm bảo các giá trị của các thuộc tính được thiết lập đúng với các giá trị được truyền vào.

Postcondition đối với với thuộc tính number:

```
/*
 * Fetch the receiver's number
 *
 * Preconditions: NONE
 * Postconditions:
 *    result == number
 * Exceptions: NONE
 */
public String getNumber();
```

```
/*
 * Set the receiver's number to n
 *
 * Preconditions: NONE
 * Postconditions:
 *    number == n
 * Exceptions: NONE
 */
public void setNumber(String n);
```

Poscondition đối với thuộc tính internetAccount:

```
/*
 * Fetch the receiver's internetAccount
 *
 * Preconditions: NONE
 * Postconditions:
 *    result == internetAccount
 * Exceptions: NONE
 */
public InternetAccount getInternetAccount();
/*
 * Set the receiver's internetAccount to ia
 *
 * Preconditions: NONE
 * Postconditions:
 *    internetAccount == ia
 * Exceptions: NONE
 */
public void setInternetAccount(InternetAccount ia);
```

Postcondition đối với thược tính address:

```
/*
 * Fetch the receiver's address
 *
 * Preconditions: NONE
 * Postconditions:
 *    result == address
 * Exceptions: NONE
 */
public String getAddress();
```

```
/*
 * Set the receiver's address to a
 *
 * Preconditions: NONE
 * Postconditions:
 *    address == a
 * Exceptions: NONE
 */
public void setAddress(Address a);
```

7.5 Lên kế hoạch quản lý biệt lệ

Nơi nào bạn nghĩ rằng các biệt lệ không được được kiểm soát có thể có ích? (Gợi ý: Một ngoại lệ không được kiểm soát có thể được sử dụng bởi các nhà cung cấp (Supplier) để báo hiệu rằng người mua (Buyer) đã làm một yêu cầu không phù hợp hoặc một lỗi nội tại của hệ thống - bug - đã gặp phải)

Nơi nào bạn có thể sử dụng các biệt lễ được kiểm soát? (Gợi ý: biệt lệ kiểm soát không thể bỏ qua bởi người mua, vì vậy nó là tốt cho việc xây dựng một tường lửa ngăn cách người sử dụng hoặc các hệ thống bên ngoài)

Ví dụ: Trong lớp *ReservationUI*, những biệt lệ có thể xuất hiện trong các phương thức *createReservation(int i, int c), cancelReservation(int i, int r)*, *readReservations(int i)* như lỗi đột xuất xuất phát từ phần cứng, những lỗi từ các tham số truyền vào không hợp lệ, lỗi từ việc kết nối cơ sở dữ liệu ..:

```
Exceptions:
   PServerException (checked) thrown if the server has a problem
   IllegalArgumentException (unchecked) thrown if parameters are
      invalid
```