ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN
KHOA CÔNG NGHỆ PHẦN MỀM

Báo cáo đề tài

ĐẶC TẢ PHẦN MỀM

Môn: Serminar các vấn đề hiện đại của CNPM

GVHD: Thầy Vũ Thanh Nguyên

Sinh viên thực hiện:

1. Văn Vũ Tuấn 12520487
2. Phạm Ngọc Linh 12520228
3. Huỳnh Đức Đăng Khoa 12520204

Thành phố Hồ Chí Minh – 12/12016

NHẬN XÉT CỦA GIẢNG VIÊN

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Nội dung

I. Giới thiệu đề tài

1.1 Tổng quan

Trong quá trình thực hiện bất cứ dự án phần mềm nào, xử lý các yêu cầu mà khách hàng đưa ra bao giờ cũng là một vấn đề vô cùng quan trọng. Các yêu cầu nếu được xử lý đúng đắn sẽ giúp cho các nhà phát triển tham gia vào dự án có được cái nhìn tổng thể về dự án, các nhà quản lý có cơ sở để hoạch định các kế hoạch quản lý của mình. Như vậy việc đặc tả phần mềm đóng vai trò cốt lõi trong viêc tạo ra một sản phẩm phần mềm có giá trị. Trong nội dung bài báo này, chúng em xin trình bày một cách tổng quan và cô đọng nhất các vấn đề quan trọng trong quá trình đặc tả một phần mềm.

1.2 Phạm vi báo cáo

Báo cáo này chúng em xin tập trung nói về đặc tả phần mềm trong quy trình phát triển phần mềm. Nội dung bao gồm giới thiệu sơ lược về Kỹ nghệ phần mềm, Kỹ thuật yêu cầu và Đặc tả yêu cầu phần mềm. Phần Đặc tả yêu cầu phần mềm sẽ là phần trọng tâm của báo csaso quy trình thực hiện, các tài liệu cuối cùng, và các kĩ thuật thường sử dụng trong quá trình đặc tả yêu cầu.

1.3 Nội dung báo cáo

Trong phạm vi báo cáo này, chúng em xin trình bày theo cấu trúc như sau:

- Chương 1: Giới thiệu tổng quan về đề tài Đặc tả Phần mềm.
- Chương 2: Giới thiệu tổng quan về Kỹ nghệ Phần mềm.
- Chương 3: Giới thiệu tổng quan về một kỹ thuật trong Kỹ nghệ Phần mềm là Kỹ thuật Yêu cầu, kỹ thuật được sử dụng chủ yếu trong quá trình xử lý yêu cầu phần mềm.
- Chương 4: Trình bày về Đặc tả yêu cầu Phần mềm, đây là một quá trình quan trọng trong Kỹ nghệ phần mềm.
- Chương 5: Trình bày về các kỹ thuật được sử dụng thường xuyên trong quá trình Đặc tả yêu cầu phần mềm.
- Cuối cùng là phần tài liệu đặc tả cho dự án demo.

II. Kỹ nghệ Phần mềm

2.1 Khái niệm

Muốn hiểu rõ khái niệm Kỹ nghệ Phần mềm, ta cần hiểu rõ hai khái niệm *Kỹ nghệ* (*Engineering)* và *Phần mềm (Software).*

Kỹ nghệ là một danh từ chỉ việc chúng ta áp dụng toán học và các kiến thức về kinh tế, xã hội, thực tiễn vào việc phát minh, sáng tạo, thiết kế, xây dựng, bảo trì, nghiên cứu và cải thiện các kiến trúc, các máy móc, các công cụ, các hệ thống, các thành phần, các nguyên liệu, các quy trình, các giải pháp và kể cả các tổ chức….

Phần mềm là một phần của hệ thống máy tính, tập hợp các câu lệnh và các chỉ thị được viết bằng một hay nhiều ngôn ngữ lập trình theo một trật tự nhất định nhằm tự động thực hiện một số nhiệm vụ hay chức năng hoặc giải quyết một vấn đề nào đó.

Vậy, Kỹ nghệ Phần mềm là một cách tiếp cận có hệ thống trong quá trình sản xuất phần mềm, ta sử dụng nó để phân tích, thiết kế, đánh giá, hiện thực, kiểm thử, bảo trì và tái cấu trúc phần mềm, hay nói cách khác là áp dụng kỹ nghệ sản xuất vào trong quá trình tạo ra phần mềm.

Trong cách tiếp cận kỹ nghệ phần mềm, ta sẽ định nghĩa một số mô hình cho vòng đời sống của phần mềm, mỗi mô hình bao gồm nhiều pha khác nhau. Để đánh giá các pha nói chung và các mô hình nói riêng, ta sẽ cần sử dụng nhiều phương pháp luận khác nhau.

Kỹ nghệ phần mềm sử dụng kiến thức của nhiều lĩnh vực khác nhau như kỹ thuật máy tính, khoa học máy tính, quản lý, toán học, quản lý dự án, quản lý chất lượng, công thái học phần mềm (software ergonomics), và kỹ nghệ hệ thống (systems engineering).

2.2 Lịch sử ra đời

Lịch sử của ngành kỹ nghệ phần mềm gắn liền cùng vơi sự phát triển của máy tính cá nhân (personal computer). Từ những năm 1950, phát triển phần mềm và kỹ nghệ phần mềm đã có những bước đi đầu tiên, tuy nhiên nhiều người tin rằng phải đến năm 1968, kỹ nghệ phần mềm mới chính thức trở thành khái niệm được công nhận bởi một hội nghị do NATO

tài trợ. Hội nghị này chính là động lực thúc đẩy các nghiên cứu sau này về kỹ nghệ phần mềm.

Nhiều người trong ngành tự gọi mình là "kỹ sư phần mềm" từ năm 1989. Đến cuối những năm 1980, đầu những năm 1990, các trường đại học bắt đầu giảng dạy về kỹ nghệ phần mềm.

2.3 Sự phát triển của Kỹ nghệ Phần mềm

Sự phát triển của kỹ nghệ phần mềm đi liên với sự phát triển của máy tính cá nhân (personal computer), từ các chương trình trên bìa đục lỗ hết sức đơn giản dành cho các máy tính mainframe khổng lồ có kích cỡ ngang một tòa nhà cho tới các ứng dụng chạy trên các thiết bị di động có kích thước chỉ bằng bàn tay. Ta có thể tóm tắt sự phát triển của Kỹ nghệ phần mềm qua các giai đoạn chính sau:

- *Những năm đầu từ 1950 đến 1960*

Giai đoạn này phần cứng thay đổi liên tục, số lượng máy tính rất ít và phần lớn mỗi máy đều được đặt hàng chuyên dụng cho một ứng dụng đặc biệt. Phương thức chính là xử lý theo lô (batch), tức là "gói" các chương trình có sử dụng kết quả của nhau lại thành một khối dể tăng tốc độ thực hiện.

Thời kỳ này việc lập trình các chương trình máy tính được coi là "theo bản năng", không hề có một quy trình hay một phương pháp nào có hệ thống. Việc phát triển phần mềm chưa được quản lý.

Môi trường lập trình có tính chất cá nhân; thiết kế, tiến trình phần mềm không tường minh, thường không có tài liệu. Sản xuất có tính đơn chiếc, theo đơn đặt hàng. Người lập trình thường là người sử dụng và kiêm cả việc bảo trì và sửa lỗi.

- *Thời kỳ mở rộng từ những năm 1960 đến giữa những năm 1970*

Các hệ thống đa nhiệm, đa người sử dụng (ví dụ: Multics, Unix,...) xuất hiện dẫn đến khái niệm mới về tương tác người máy. Kỹ thuật này mở ra thế giới mới cho các ứng dụng và đòi hỏi mức độ tinh vi hơn cho cả phần mềm và phần cứng.

Nhiều hệ thống thời gian thực với các đặc trưng thu thập, phân tích và biến đổi dữ liệu từ nhiều nguồn khác nhau và phản ứng (xử lý, tạo output) trong một khoảng thời gian nhất định xuất hiện. Các tiến bộ trong việc lưu trữ trực tuyến làm xuất hiện các hệ quản trị CSDL thế hệ đầu tiên.

Số lượng các hệ thống dựa trên máy tính phát triển, nhu cầu phân phối mở rộng, thư viện phần mềm phát triển, quy mô phần mềm ngày càng lớn làm nẩy sinh nhu cầu sửa chữa khi gặp lỗi, cần sửa đổi khi người dùng có yêu cầu hay phải thích nghi với những thay đổi của môi trường phần mềm (phần cứng, hệ điều hành, chương trình dịch mới). Công việc bảo trì phần mềm dần dần tiêu tốn nhiều công sức và tài nguyên đến mức báo động.

- *Thời kỳ từ giữa những năm 1970 đến đầu những năm 1990*

Hệ thống phân tán (bao gồm nhiều máy tính, mỗi máy thực hiện một chức năng và liên lạc với các máy khác) xuất hiện làm tăng quy mô và độ phức tạp của phần mềm ứng dụng trên chúng.

Mạng toàn cục và cục bộ, liên lạc số giải thông cao phát triển mạnh làm tăng nhu cầu thâm nhập dữ liệu trực tuyến, nảy sinh yêu cầu lớn phát triển phần mềm quản lý dữ liệu.

Công nghệ chế tạo các bộ vi xử lý tiến bộ nhanh khiến cho máy tính cá nhân, máy trạm để bàn, và các thiết bị nhúng (dùng cho điều khiển trong robot, ô tô, thiết bị y tế, đồ điện gia dụng,...) phát triển mạnh khiến cho nhu cầu về phần mềm tăng nhanh.

Thị trường phần cứng đi vào ổn định, chi phí cho phần mềm tăng nhanh và có khuynh hướng vượt chi phí mua phần cứng.

Các phương pháp quản lý mới được nghiên cứu và lần đầu đưa vào sử dụng trong quá trình phát triển phần mềm.

- *Thời kỳ sau 1990*

Kỹ nghệ hướng đối tượng là cách tiếp cận mới đang nhanh chóng thay thế nhiều cách tiếp cận phát triển phần mềm truyền thống trong các lĩnh vực ứng dụng.

Sự phát triển của Internet làm cho người dùng máy tính tăng lên nhanh chóng, nhu cầu phần mềm ngày càng lớn, quy mô và độ phức tạp của những hệ thống phần mềm mới cũng tăng đáng kể.

Phần mềm trí tuệ nhân tạo ứng dụng các thuật toán phi số như hệ chuyên gia, mạng nơ ron nhân tạo được chuyển từ phòng thí nghiệm ra ứng dụng thực tế mở ra khả năng xử lý thông tin và nhận dạng kiểu con người.

Càng ngày càng có nhiều các phương pháp luận, các công cụ, các nền tảng, các framework, các ngôn ngữ, các kiến trúc mới được phát triển, kỹ nghệ phần mềm đạt được một sự đa dạng chưa từng có kể từ những năm đầu sơ khai.

2.4 Các thách thức hiện tại

Phát triển nhanh chóng nhưng kỹ nghệ phần mềm vẫn chưa khắc phục hết các vấn đề mà bản thân nó còn tồn đọng. Ví dụ như về khía cạnh yêu cầu sản phẩm, khách hàng mong muốn một sản phẩm có thể đáp ứng các nhu cầu của họ, trong suy nghĩ của họ việc xây dựng một ứng dụng như thế là phù hợp, tuy nhiên các nhà phát triển khó có thể tạo ra một sản phẩm hoàn toàn phù hợp do hạn chế về kỹ thuật và thời gian.

Thứ hai là phạm vi dự án. Nếu dự án có quy mô nhỏ, các yêu cầu thường đơn giản và dễ dàng phát triển, đôi khi chỉ mất ½ tháng để hoàn tất. Nếu dự án có quy mô lớn thì khó khăn hơn nhiều, đôi khi là rất khó để tạo ra các sản phẩm sạch lỗi và đáp ứng được yêu cầu.

Thứ ba là sự khác biệt về kiến thức chuyên ngành. Các khách hàng có kiến thức chủ yếu là kiến thức nghiệp vụ (business) trong lĩnh vực của họ. Còn các nhà phát triển có kiến thức chuyên môn về công nghệ, kỹ thuật. Do đó cần có một sự chuyển giao kiến thức trong quá trình phát triển phần mềm từ khách hàng sang các nhà phát triển. Nếu quá trình này diễn ra không thành công thì xác suất thất bại của dự án là khá cao.

Cuối cùng, có một vấn đề thường xuyên xảy ra là khách hàng thay đổi yêu cầu của họ, sự thay đổi này có thể diễn ra không đúng lúc, sau khi yêu cầu đó đã được hiện thực hóa, khi đó muốn thay đổi lại phần đã thực hiện thì tốn nhiều chi phí, thời gian và công sức.

III. Kỹ thuật Yêu Cầu

Trước khi đi vào phần đặc tả yêu cầu phần mềm, chúng em xin giới thiệu sơ qua về Kỹ thuật yêu cầu: các khái niệm, đặc điểm, mục tiêu, vai trò và các hoạt động chính trong Kỹ thuật Yêu cầu. Sau đó giới thiệu các hoạt động chính.

Một vấn đề quan trọng trong Kỹ thuật Yêu cầu là Mô hình hóa Yêu cầu. Nó sẽ liên quan đến nhiều kỹ thuật khác nhau để mô tả các yêu cầu theo đúng như mong đợi của khách hàng. Các kỹ thuật này cáo thể ở mức trừu tượng cao như các đặc tả có dạng mã giả, các logic hình thức và các biểu diễn đồ họa. Bất cứ kỹ thuật nào được dùng, các kỹ sư cũng cần phải cố gắng để thu thập các đặc tả yêu cầu hệ thống một cách đầy đủ, chính xác và chi tiết.

3.1 Tổng quan

Khái niệm:

Ta có thể coi *Kỹ thuật yêu cầu* là một phạm vi con (subdiscipline) của Kỹ nghệ Phần mềm mà ở đó các công việc chủ yếu liên quan đến việc xác định các mục tiêu, các chức năng, các ràng buộc của hệ thống phần mềm. Kỹ thuật Yêu cầu cũng bao gồm các yếu tố kỹ thuật cũng như các đặc tả chính xác về hành vi của phần mềm và sự tiến hóa của nó qua thời gian.

Kỹ thuật yêu cầu gồm các giai đoạn tìm kiếm, ghi chép, phân tích, xác thực và quản lý các yêu cầu. Có nhiều cách tiếp cận kỹ thuật yêu cầu, mỗi cách tiếp cận có mức độ hoàn thiện khác nhau, tuy nhiên chúng đều có phương pháp luận đầy đủ và đều đảm bảo mỗi giai đoạn đều có tài liệu riêng.

Khi nào bắt đầu:

Theo lý thuyết, kỹ thuật yêu cầu sẽ bắt đầu với một hoạt động nghiên cứu tính khả thi (feasibility study) để sinh ra tài liệu về tính khả thi của dự án (feasibility report). Tùy vào kết quả của tài liệu này mà ta sẽ quyết định có tiếp tục thực hiện dự án hay không. Nếu

nghiên cứu tính khả thi đề nghị tiếp tục phát triển sản phẩm thì giai đoạn phân tích yêu cầu sẽ được tiến hành.

Mục tiêu:

Các đặc tả phần mềm chính xác cung cấp các thông tin cơ bản cho các pha phân tích thiết kết sau này, xác thực những mong muốn của stakeholder, định nghĩa những gì mà các kỹ sư cần xây dựng, và bằng chứng để chứng mình những gì họ làm là chính xác.

Vai trò:

Đặc tả yêu cầu phần mềm cho phép chúng ta biết được động lực để phát triển hệ thống phần mềm. Các yêu cầu phần mềm được đặc tả một cách kỹ càng giúp các nhà quản lý hoạch định kế hoạch tương lai và nắm bắt được sự phát triển của phần mềm qua thời gian. Cách tiếp cận này phản ánh thực tế của một thế giới luôn thay đổi và nhu cầu sử dụng lại một phần đặc tả đã được dùng trong một hệ thống phần mềm khác.

Các hoạt động chính:

Có 5 hoạt động chính trong quá trình đặc tả yêu cầu phần mềm:

- Tìm kiếm yêu cầu – **Requirements Elicitation.**
- Mô hình hóa yêu cầu – **Requirements Modeling.**
- Phân tích yêu cầu – **Requirements Analysis.**
- Quản lý các yêu cầu – **Requirements Documentation.**
- Xác thực các yêu cầu – **Requirements Varifiaction.**

Tìm kiếm các yêu cầu có thể coi là hoạt động khó khăn nhất trong trong kỹ thuật yêu cầu. Bởi vì các yêu cầu không phải lúc nào cũng rõ ràng, chúng thường được stakeholders diễn tả không chính xác, vì đôi khi stakeholders cũng không hiểu rõ các yêu cầu của chính mình, nhiều yêu cầu có thể rất phức tạp hoặc chúng có thể mâu thuẫn với nhau. Các kỹ sư cần phải tự mình tìm tòi và suy ra các yêu cầu chính xác từ những gì mà khách hàng cung cấp cho họ.

Mô hình hóa yêu cầu liên quan đến việc biểu diễn các yêu cầu dưới các dạng văn bản, đồ họa, hình ảnh, các công thức toán học,.. Nhờ vậy ta có thể hiểu được các yêu cầu một cách trực quan và rõ ràng.

Phân tích yêu cầu liên quan đến việc xác định xem yêu cầu đó có chính xác hay không. Ngoài ra còn một số yếu tố mà ta cần quan tâm đến là sự chặt chẽ, tính hoàn thiện, độ chi tiết, v.v… Mô hình của các yêu cầu và các thuộc tính của chúng cần phải được liên hệ với các bên liên quan.

Xác thực yêu cầu là hoạt động mà các kỹ sư cần làm việc với các bên liên quan để kiểm chứng độ đúng đắn của các mô hình yêu cầu mà họ đã lập nên. Bất kì thay đổi nào trong quá trình cũng này đều dẫn tới việc quay trở lại thực hiện các hoạt động bên trên.

Quản lý các yêu cầu có trách nhiệm đảm bảo các yêu cầu sau khi được tạo ra sẽ được lưu trữ cẩn thận, bất kì thay đổi này theo thời gian cũng cần được cập nhật kịp thời và đồng bộ.

3.2 Các hoạt động chính

Tìm kiếm các yêu cầu

Là các công việc liên quan đến việc làm việc với khách hàng để xác định lĩnh vực ứng dụng, các dịch vụ mà hệ thống cần cung cấp cũng như các ràng buộc của hệ thống.

Vấn đề đặt ra là không phải lúc nào khách hàng cũng hiểu rõ cái mà họ cần, hoặc các yêu cầu thường không rõ ràng, hoặc các bên liên quan (stackholder) diễn tả không chính xác. Do đó, các kỹ sư cần phải cảm nhận được yêu cầu của khách hàng và nhắc nhở họ bằng kinh nghiệm của bản thân, sau đó thông qua giao tiếp với khách hàng bằng các cuộ gặp gỡ, trao điển để tìm ra các yêu cầu của họ là gì.

Những nhân tố tham gia vào quá trình này bao gồm: Các bên liên quan (stakeholder): người dùng cuối, quản lý, các kỹ sư, các chuyên gia có lên quan đến lĩnh vực của ứng dụng.

Có 3 cách cách tiếp cận chủ yếu để giúp cho các phân tích viên có thể tìm kiếm và nắm bắt được các yêu cầu:

- Thiết kế ứng dụng chung (Joint Application Design – JAD)
- Triển khai chức năng chất lượng (Quality Function Deployment – QFD)
- Thiết kế như người học việc (Desginer as apprentice)

Thiết kế ứng dụng chung (Joint Application Design)

Được đánh giá cao trong việc tổ chức và thực hiện các cuộc họp nhằm lấy yêu cầu hệ thống**.** Mỗi cuộc họp có sự tham gia của người dùng hệ thống, người sử hữu hệ thống và các phân tích viên**.** Quy trình này giúp các kỹ sư phần mềm trong việc tìm ra các yêu cầu cho việc đặc tả yêu cầu phần mềm, thiết kế và mô tả thiết kế hệ thống, code, kiểm thử, viết tài liệu hướng dẫn người dùng.

Quy trình thực hiện JAD:

- Chọn người tham gia
- Chuẩn bị tài liệu (agenda)
- Chọn địa điểm tiến hành.
- Thực hiện hội thảo

Triển khai chức năng chất lượng (Quality Function Deployment)

Được giới thiệu vào năm 1966 bởi Yoji Akao sử dụng trong sản xuất, công nghiệp nặng và hệ thống kỹ thuật. Là kỹ thuật để xác định yêu cầu của khách hàng và xác định được các điểm mà chất lượng sản phẩm được đảm bảo trong toàn bộ quá trình phát triển. Cung cấp cấu trúc cho việc lắng nghe cái khách hàng muốn và cần. Sau đó chuyển chúng (tiếng nói của khách hàng) thành tài liệu kỹ thuật. Giúp cải thiện việc tham gia của người dùng và người quản lý. Rút ngắn vòng đời phát triển sản phẩm. Cung cấp công cụ dự phòng trong việc tránh mất mát thông tin trong mỗi giai đoạn phát triển sản phẩm

Thiết kế như người học việc (Designer as apprentice)

Người thiết kế phải học tất cả các công việc của khách hàng đã, đang và sẽ làm. Phương pháp này mang lại nhiều khó khăn cho người thiết kế, bởi vì đôi khi không phải khách hàng nào cũng có khả năng truyền đạt toàn bộ những gì họ hiểu hoặc có thể không hiểu hoàn

toàn những gì họ đang làm. Do đó cần có sự hợp tác chặc chẽ của khách hàng trong suốt quá trình "học tập" của người thiết kế.

Thuận lợi mang lại cũng rất đáng để lưu ý: người phân tích sẽ hiểu sâu hơn về cái mình sẽ làm, đồng lời tích lũy thêm kinh nghiệm, học hỏi cái mới. Khách hàng nhận được sản phẩm sát với yêu cầu nhất và thời gian phát triển sản phẩm sẽ được rút ngắn.

Mô hình hóa các yêu cầu

Có rất nhiều phương pháp khác nhau để mô hình hóa các yêu cầu phần mềm. Bởi vì tính chất của ngôn ngữ tự nhiên, nó ít khi được dùng trong các tài liệu đặc tả yêu cầu phần mềm, hoặc nếu được dùng thì cần phải đi kèm các cấu trúc, ràng buộc ngữ nghĩa để đảm bảo tính đúng đắn và mạch lạc của mô hình.

Các giải pháp thay thế cho ngôn ngữ tự nhiên:

- Ngôn ngữ đặc tả có cấu trúc (structured language specification)
- Ngôn ngữ thiết kế phần mềm (Program Design Language – PDL)
- Kí hiệu đồ họa
- Use case/User Story
- Formal method (ngôn ngữ Z, VDM)
- Informal/Semifomal method (DFD, UML)

Phân tích các yêu cầu

Trong công đoạn này, các nhà phát triển sẽ bắt tay vào xem liệu các yêu cầu được mô hình hóa có rõ ràng, minh bạch, hoàn chỉnh, v.v… hay không. Sau đó giải quyết các vấn đề còn tồn đọng trong mô hình bằng cách làm việc với khách hàng. Đồng thời họ sẽ xem xét các nhu cầu/điều kiện đã có để quyết định xem nên làm một sản phẩm mới hay chỉnh sửa một sản phẩm có sẵn.

Các yêu cầu sau khi đã qua bước phân tích cần đáp ứng các tiêu chí: có thể tiến hành, có thể đo lường được, có thể kiểm tra được, liên quan đến các nhu cầu kinh doanh, và được định nghĩa ở mức độ chi tiết đủ để thiết kế hệ thống

Lập tài liệu các yêu cầu

Các yêu cầu sau khi đã mô hình hóa và được phân tích kỹ càng sẽ được tập hợp lại trong một tài liệu đặc tả yêu cầu phần mềm. Tài liệu này bao gồm các định nghĩa và các đặc tả của các yêu cầu, tuy nhiên nó không phải là tài liệu thiết kế. Nó cho chúng ta biết hệ thống của chúng ta sẽ làm được cái gì chứ không phải làm thế nào mà chúng được thực hiện.

Những người sẽ sử dụng SRS:

- Khách hàng: cho biết làm thế nào để thỏa mãn yêu cầu của họ; tiếp tục quá trình này trong suốt quá trình sản xuất.
- Người quản lý: lập kế hoạch cho dự án và quản lý quá trình sản xuất phần mềm
- Nhà phát triển : tạo ra các thiết kế phù hợp với các yêu cầu.
- Kỹ sư kiểm thử: là căn cứ để xác định hệ thống có phù hợp hay không.
- Kỹ sư bảo trì: hiểu được hệ thống sẽ làm gì và nó trở nên như thế nào qua thời gian.

IV. Đặc tả Yêu cầu Phần mềm

4.1 Tổng quan

Khái niệm

Đặc tả yêu cầu phần mềm là một tập hợp các hoạt động được thiết kế nhằm nắm bắt các khía hạn chức năng và phi chức năng của hệ thống, sau đó đặc tả chúng trong các tài liệu SRS (tài liệu đặc tả yêu cầu phần mềm). Mục tiêu chính của quá trình này là nhằm cung cấp một mô tả đầy đủ về hành vi của hệ thống (cái gì mà hệ thống cần làm) mà không cần phải mô tả chi tiết cấu trúc bên trong (các yêu cầu đó được hệ thống thực hiện như thế nào, ra sao).

Phân loại

Đặc tả yêu cầu phần mềm thường được phân loại dựa theo mức độ trừu tượng của các yêu cầu. Có 3 loại đặc tả chính:

- Đặc tả yêu cầu người dùng tương ứng với yêu cầu của người dùng,.
- Đặc tả yêu cầu hệ thống tương ứng với yêu cầu của hệ thống.
- Đặc tả thiết kế phần mềm

Đặc tả yêu cầu người dùng

Thường được sử dụng để thu hút các nhà đầu tư, thường có hình thức một bản yêu cầu đề xuất (RFP). Yêu cầu người dùng thường chứa các phát biểu trừu tượng bằng ngôn ngữ tự nhiên (chẳng hạn, Tiếng Anh), có kèm theo các sơ đồ chính thức, kể cả các bản vẽ chi tiết về các logic cho nghiệp vụ của khách hàng. Yêu cầu của người dùng mô tả các yêu cầu chức năng và phi chức năng được trình bày dưới dạng trực quan về dễ hiểu cho phía khách hàng và người dùng hệ thống.

Đặc tả yêu cầu hệ thống

Đây là phần chúng ta đã đề cập ở trên, là các mô tả chi tiết về các dịch vụ và ràng buộc của hệ thống. Đặc tả yêu cầu hệ thống có nguồn gốc từ việc phân tích yêu cầu người dùng.

Các yêu cầu hệ thống cần có cấu trúc và rõ ràng bởi vì chúng như là hợp đồng giữa client và contractors (các bên tham gia phát triển). Đặc tả này gần như là hoàn chỉnh, chỉ chứa các sơ đặc tả và bỏ qua các phần không cần thiết.

Đặc tả thiết kế phần mềm

Là đặc tả chi tiết nhất được viết ra và được sử dụng bởi các kỹ sư phần mềm như là một bản thiết kế cơ bản cho kiến trúc hệ thống.

4.2 Quy trình

Các yêu cầu có được sau khi thực hiện qúa trình thu thập sẽ được đưa vào quá trình xử lý để có được đặc tả tương ứng.

Đặc điểm:

- Các hoạt động này không phải tuyến tính, chúng lặp đi lặp lại và song song với nhau
- Trong quá trình đặc tả một yêu cầu cũng có thể giúp ta phân tích được yêu cầu đó.
- Thẩm định có thể cho thấy điểm không hợp lý trong quá trình đặc tả, có thể dẫn tới quá trình phân tích sâu hơn.
- "Chia để trị" là chiến lược cơ bản, ta chia các phần lớn thành các phần nhỏ hơn, hiểu được mỗi phần làm gì và mối quan hệ giữa các phần.
- Các quá trình này sẽ sinh ra một lượng lớn tài liệu có liên quan.
- Cách quản lý tốt các tài liệu này cũng là một thách thức khác.
- Áp dụng nhiều kỹ thuật khác nhau trong toàn bộ quá trình (DFD, UML, ...).

Quy trình đặc tả yêu cầu sẽ bao gồm các bước sau:

- Phân tích yêu cầu
- Đặc tả yêu cầu
- Thẩm định yêu cầu

Phân tích yêu cầu

Để hiểu được yêu cầu và các ràng buộc của hệ thống, tạo tiền đề cho việc đặc tả chúng.

Để có thể hiểu được yêu cầu và tiến phân tích, nhà phân tích sẽ tiến hành các hoạt động sau:

- Phỏng vấn khách hàng.
- Đọc các hướng dẫn (manual) nghiệp vụ.
- Học tập hệ thống (mà khách hàng đang sử dụng) hiện tại.
- Giúp khách hàng hiểu được các khả năng có thể xảy ra.
- Tư vấn khách hàng, các yêu cầu có thể không phù hợp hoặc không rõ ràng, nhà phân tích phải hiểu được công việc của tổ chức, khách hàng, người sử dụng để giúp họ đưa ra yêu cầu đúng.

Vì đa số các kiến thức về yêu cầu có được là nhờ vào quá trình giao tiếp với khách hàng nên kỹ năng giao tiếp của người phân tích là cực kỳ quan trọng. Trong quá trình phân tích ta có thể cần phải sử dụng nhiều kỹ thuật khác nhau để nắm bắt toàn bộ khía cạnh của yêu cầu.

Đặc tả yêu cầu

Các yêu cầu sau khi được đặc tả và xác thực sẽ là tiền đề để tạo nên tài liệu SRS

Không dễ dàng gì để chuyển đổi các yêu cầu trực tiếp từ quá trình phân tích thành các đặc tả. Các kiến thức được yêu cầu về hệ thống trong quá trình phân tích được sử dụng lại trong quá trình đặc tả.

Lựa chọn phương pháp đặc tả phù hợp với đặc điểm của các yêu cầu.

- Ngôn ngữ tự nhiên nếu được sử dụng cần phải hỗ trợ các đặc điểm mong muốn của SRS và phải đi kèm theo các cấu trúc nhất định; thường được dùng để đặc tả các yêu cầu có tình chất đơn giản.
- Ngôn ngữ đặc tả chuyên biệt(formal language) có độ chính xác cao và rõ ràng nhưng khó để nắm bắt và sử dụng; chúng thường được dùng để đặc tả các tính năng đặc biệt và cực kỳ quan trọng của hệ thống.

Một tài liệu SRS cần làm ghi rõ các yêu cầu về:

- Functionality - Tính năng
- Performance – Hiệu suất
- Design constraints – Ràng buộc thiết kế
- External interfaces – Các giao tiếp bên ngoài

Yêu cầu chức năng

Trái tim của tài liệu SRS; phần lớn các đặc tả kỹ thuật là do chúng tạo nên. Các yêu cầu chức năng đặc tả tất cả các chức năng mà hệ thống phải hỗ trợ. Mỗi chức năng cần có đầu ra tương ứng với đầu vào và mối quan hệ giữa chúng đồng thời phải xác định hành vi của hệ thống đối với các đầu vào không hợp lệ.

Yêu cầu hiệu suất

Mô tả tất cả các ràng buộc về mặt hiệu suất của hệ thống phần mềm. Các yêu cầu này thông thường là về thời gian đáp ứng, thông lượng, v.v… (động – có thể thay đổi), dung lượng (tĩnh – cố định), nói chung là các khái niệm ta có thể đo lường được.

Ví dụ: Thời gian phản hồi nên ít hơn xx giây, dung lượng ổ đĩa chiếm dụng ít hơn 1 TB.

Ràng buộc về thiết kế

Các yếu tố trong môi trường mà khách hàng sử dụng hệ thống sẽ hạn chế các lựa chọn thiết kế hệ thống. Một vài ràng buộc có thể có:

- Tuân thủ tiêu chuẩn đã có nhằm tương thích với hệ thống khác.
- Hạn chế về mặt phần cứng.
- Độ tin cậy, khả năng chịu lỗi, yêu cầu sao lưu.
- Độ bảo mật.

Giao tiếp bên ngoài

Tất cả các tương tác của phần mềm với con người, các hệ phần mềm khác có trong môi trường sử dụng, các phần cứng có sẵn của doanh nghiệp (như máy in, …). Sau khi thu thập cũng cần phải kiểm chứng cẩn thận để đảm bảo sau khi sản phẩm hoàn thành nó có thể hoạt động hoàn hảo với các yếu tố bên trên.

Cấu trúc tài liệu SRS theo tiêu chuẩn được đề ra bởi IEEE:

- Giới thiệu
- Mục đích, mục tiêu cơ bản của hệ thống,
- Phạm vi của hệ thống: làm những gì, không làm những gì.
- Các định nghĩa và từ viết tắt.
- Các liên kết (references)
- Tổng quan

- Mô tả chung

- Quan điểm về sản phẩm
- Chức năng sản phẩm
- Đặc điểm người sử dụng
- Các giả định và sự phụ thuộc (Assumptions and dependencies)
- Các ràng buộc

- Các yêu cầu cụ thể

- Giao tiếp bên ngoài
- Yêu cầu chức năng
- Yêu cầu hiệu suất
- Ràng buộc về thiết kế

- Phụ lục

- Mục lục

Thẩm định yêu cầu

Thực tế là chưa chắc chúng ta đã hiểu đúng các yêu cầu. Yêu cầu sau khi được đặc tả chưa chắc đúng với thực tế, việc chúng có lỗi là hiển nhiên, nếu ta không có biện pháp để phát hiện và sửa chữa ngay dễ dẫn tới các hậu quả nghiêm trọng sau này.

Tuy nhiên ta cũng không thể đảm bảo là có thể loại bỏ hoàn toàn các lỗi có trong các đặc tả. Ta chỉ có thể cố gắng loại bỏ các lỗi trong SRS càng nhiều càng tốt.

Các lỗi thường gặp khi thẩm định lại các yêu cầu đã được đặc tả:

- Thiếu sót - 30%
- Không chặt chẽ - 10-30%
- Sự thật không chính xác - 10-30%
- Mơ hồ - 5 -20%

V. Các phương pháp đặc tả yêu cầu

5.1 Phân loại

Tùy theo các tiêu chuẩn mà người ta phân loại các phương pháp đặc tả. Có hai hướng để phân loại: theo trạng thái ngôn ngữ và theo mức độ hình thức.

Theo trạng thái ngôn ngữ:

- Ngôn ngữ tự nhiên có cấu trúc
- Ngôn ngữ mô tả thiết kế
- Ký hiệu đồ họa
- Đặc tả toán học

Theo mức độ hình thức:

- Hình thức – formal
- Không hình thức – informal
- Bán hình thức – semiformal

5.2 Ngôn ngữ tự nhiên có cấu trúc

Là một hình thức giới hạn của ngôn ngữ tự nhiên được dùng để đặc tả các yêu cầu.

Chúng sử dụng một tập từ vựng và ngữ pháp nhất định để thể hiện các yêu cầu dưới một sự kiểm soát chặt chẽ để có thể được phân tích (parse)

Ưu điểm: Giúp ta loại bỏ các vấn đề phát sinh từ sự mơ hồ và tăng mức độ đồng bộ trên tài liệu đặc tả.

Nhược điểm Đòi hỏi các bên có liên quan (stakeholder) phải được đào tạo ở một trình độ nhất định để có thể sử dụng.

Cách tiếp cận chủ yếu là ta sử dụng các template dạng biểu mẫu (form-based template). Trong ví dụ mình họa là một dạng biểu mẫu có thể được sử dụng.

Chức năng	
Mô tả	
Các đầu vào	
Các đầu ra	
Mục đích	
Yêu cầu	
Điều kiện trước	
Điều kiện sau	
Side-effects	

5.3 Ngôn ngữ thiết kế chương trình – PDL

Sử dụng các ngôn ngữ tự nhiên giống như là ngôn ngữ lập trình nhưng sự biểu đạt linh hoạt hơn, không bị gò bó như ngôn ngữ lập trình. Về mặt hình thức thì PDL gần giống với mã giả (pseudo-code) nhưng nó không phải là mã giả.

Ta có thể sử dụng PDL trong 2 trường hợp:

- Khi một hoạt động được mô tả như là một choỗi các hành động và thứ tự giữa chúng rất quan trọng
- Khi giao tiếp (interface) phần cứng và phần mềm cần phải được quy định.

Ưu điểm của phương pháp này là nó có cấu trúc gần giống như ngôn ngữ lập trình nên sẽ dễ hiểu hơn đối với các lập trình viên, đồng thời nó cũng có ưu thế từ vốn từ vựng đa dạng của ngôn ngữ tự nhiên hơn là tập hợp các keyword của ngôn ngữ lập trình.

Nhược điểm là PDL có thể chưa đủ khả năng để biểu diễn các khái niệm trong miền ứng dụng (domain). Đặc tả được tạo ra bởi PDL chỉ được dùng cho việc thiết kế hơn là một đặc tả tuần túy. Và cũng giống như mã giả, nó chỉ có thể hiểu được bởi những người có kiến thức về ngôn ngữ lập trình, do đó không phù hợp để trao đổi với các khách hàng.

Trong hình là ví dụ về PDL và mã giả. Ta có thể dễ dàng nhận ra sự tương đồng và sự khác biệt giữa hai ngôn ngữ.

Basic style pseudo code

```
Sub fizzbuzz()
  For i = 1 to 100
    print_number = True
    If i is divisible by 3 Then
      Print "Fizz"
      print_number = False
    End If
    If i is divisible by 5 Then
      Print "Buzz"
      print_number = False
    End If
    If print_number = True Then print i
    Print a newline
  Next i
End Sub
```

```
BEGIN NOTIFY PROC

    INPUT
        FILE OR SAVE A MARKED DOCUMENT IN LIBRARY
        USING A NOTIFY ACTION MARKER REQUEST OR
        ACTIVATE NOTIFY PROCESS SUBSEQUENT TO
        DOCUMENT BEING FILED.
    PROCESS
        CREATE A TEMPORARY RECIPIENT TABLE (TRT)
        DO FOR EACH ACTION MARKER IN THE DOCUMENT
          FOR THIS RESPONDING END USER (EUR)
            IF THIS IS A FIRST MARKER IN THE
              DOCUMENT FOR THIS EUR, THEN
                CREATE AN ENTRY FOR THIS EUR IN
                THE TRT
            ENDIF
            PRESENT A DEFAULT MESSAGE AND COMMAND
            SPECIFICATIONS
            GET THE MESSAGE AND THE COMMAND
```

5.4 Use case

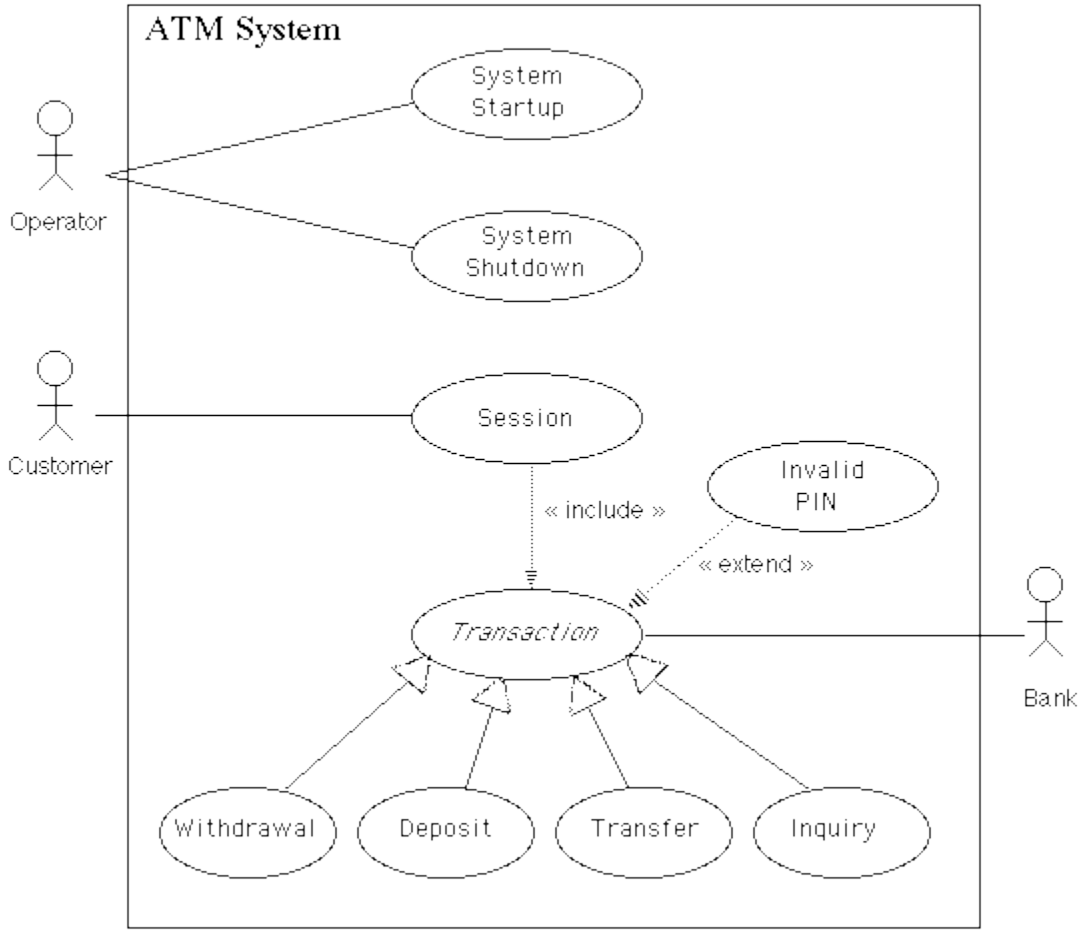
Use case mô tả sự tương tác đặc trưng giữa người dùng bên ngoài (actor) và hệ thống. Nó thể hiện cách ứng xử của hệ thống đối với các tác nhân bên ngoài, trong một hoàn cảnh nhất định, xét từ quan điểm của người sử dụng.

Nó mô tả các yêu cầu mà actor mong muốn thực hiện đối với hệ thống, có nghĩa là sơ đồ Use case chỉ mô tả hệ thống cần phải làm những gì chứ không nói chi tiết hệ thống phải

thực hiện điều đó như thế nào. Mỗi use case mô tả cách thức actor tương tác với hệ thống để đạt được mục tiêu nào đó.

Một hoặc nhiều kịch bản (scenario) có thể được tạo ra từ mỗi use case, tương ứng với ngữ cảnh mà người dùng tương tác với hệ thống theo cách thức nào đó để đạt được mục đích nhất định. Khi mô tả Use case, người ta thường tránh dùng thuật ngữ kỹ thuật, thay vào đó họ sử dụng ngôn ngữ của người dùng cuối hoặc ngôn ngữ chuyên ngành của lĩnh vực đó.

Trong ví dụ minh họa là sơ đồ Use case của một máy ATM. Ta có thể thấy được các Actor tương tác với hệ thống, các Use case (trong hình oval) mô tả những gì mà hệ thống có thể thực hiện được.



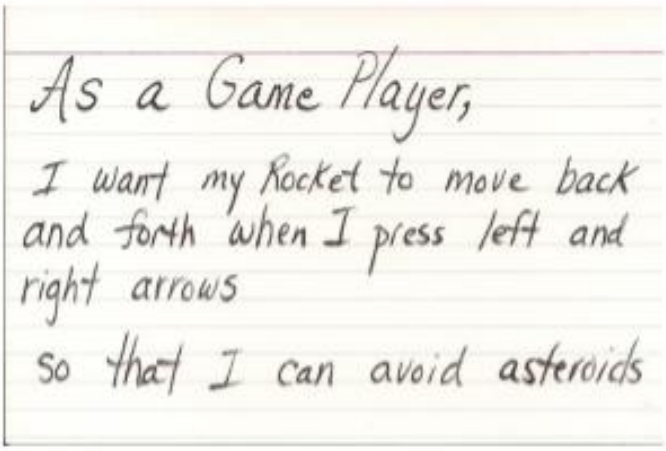
5.5 User story

Là một tài liệu sơ giản về yêu cầu sản phẩm với góc nhìn người dùng. Đây là phương pháp đặc tả thường được sử dụng trong các phương pháp luận Agile như Scrum, …

Thông thường, User Story do khách hàng, hoặc đại điện của khách hàng viết dưới sự hướng dẫn của các nhà phát triển, nhờ có sự cộng tác này nên nhóm phát triển và khách hàng sẽ có sự chia sẻ hiểu biết về sản phẩm tốt hơn.

Mục đích chính của nó là mô tả hệ thống dưới cái nhìn của người dùng, nói cách khác là hệ thống cần đáp ứng cho họ những gì.

Thường có cú pháp: "Là <*người dùng cụ thể\vai trò*> , tôi muốn <*làm gì đó*> để <*phục vụ mục đích nào đó*>"



5.6 Formal method

Phương pháp hình thức được sử dụng để cải thiện việc xây dựng các yêu cầu bằng cách áp dụng toán học và logic. Phương pháp hình thức là kết hợp từ các kiến thức toán học được kết hợp lại từ các thành tố dưới đây:

- Tính toán vị ngữ - predicate calculus (first order logic).
- Lý thuyết hàm đệ quy - recursive function theory.
- Tính toán Lambda - lambda calculus.
- Ngữ nghĩa ngôn ngữ lập trình - programming language semantics.

- Toán rời rạc - discrete mathematics.
- Lý thuyết số - number theory.
- Đại số trừu tượng - abstract algebra.

Cách tiếp cận này hấp dẫn bởi vì cung cấp một phương pháp khoa học cho việc đặc tả yêu cầu. Các yêu cầu được mô hình hóa bởi phương pháp này thường dễ dàng phát hiện lỗi ngay từ đầu, và các lỗi có thể được xử lý một cách nhanh chóng và đơn giản, ít tốn chi phí.

Các ngôn ngữ thường dùng: Ngôn ngữ Z, Ngôn ngữ VDM, CSP (cating sequential processes).

Các phương pháp hình thức được sử dụng trong các trường hợp sau:

- Kiểm tra tính chặt chẽ: các yêu cầu hành vi hệ thống được mô tả dựa trên các ký hiệu toán học.
- Kiểm tra mô hình: các trạng thái máy (state machine) được sử dụng để xác thực nếu một thuộc tính được cho là thỏa mãn mọi điều kiện
- Chứng minh định lý: tiên đề của một hành vi hệ thống được sử dụng để chứng minh rằng hệ thống sẽ hoạt động theo cách định sẵn.

5.6.1 Ngôn ngữ Z

Ngôn ngữ Z là một phương pháp đặc tả hình thức dựa trên lý thuyết tập hợp và các tính toán vị ngữ được giới thiệu vào năm 1982. Đây là một ngôn ngữ toán học chặt chẽ, được sử dụng chủ yếu trong đặc tả các yêu cầu chức năng của một hệ thống, đặc biệt là hệ thống phần mềm. Nó không được thiết kế để mô tả các yêu cầu phi chức năng (liên quan đến công cụng, hiệu suất, kích thước, v.v…), cũng không phù hợp với các đặc tả theo thời gian hoặc xử lý song song. Muốn làm được điều này, ta phải kết hợp Z và các công cụ khác.

Ví dụ về một đặc tả bằng ngôn ngữ Z cho chức năng của thang máy.

$$SWITCH ::= on \mid off$$

$$MOVE ::= up \mid down$$

$$FLOORS : \mathbb{N}$$
$$FLOORS > 0$$

IntButtons
$$IntReq : 1 \,..\, FLOORS \rightarrow SWITCH$$

FloorButtons
$$ExtReq : 1 \,..\, FLOORS \rightarrow \mathbb{P}\, MOVE$$
$$down \notin ExtReq(1)$$
$$up \notin ExtReq(FLOORS)$$

Scheduler
$$NextFloorToServe : 0 \,..\, FLOORS$$

Elevator
$$CurFloor : 1 \,..\, FLOORS$$
$$CurDirection : MOVE$$

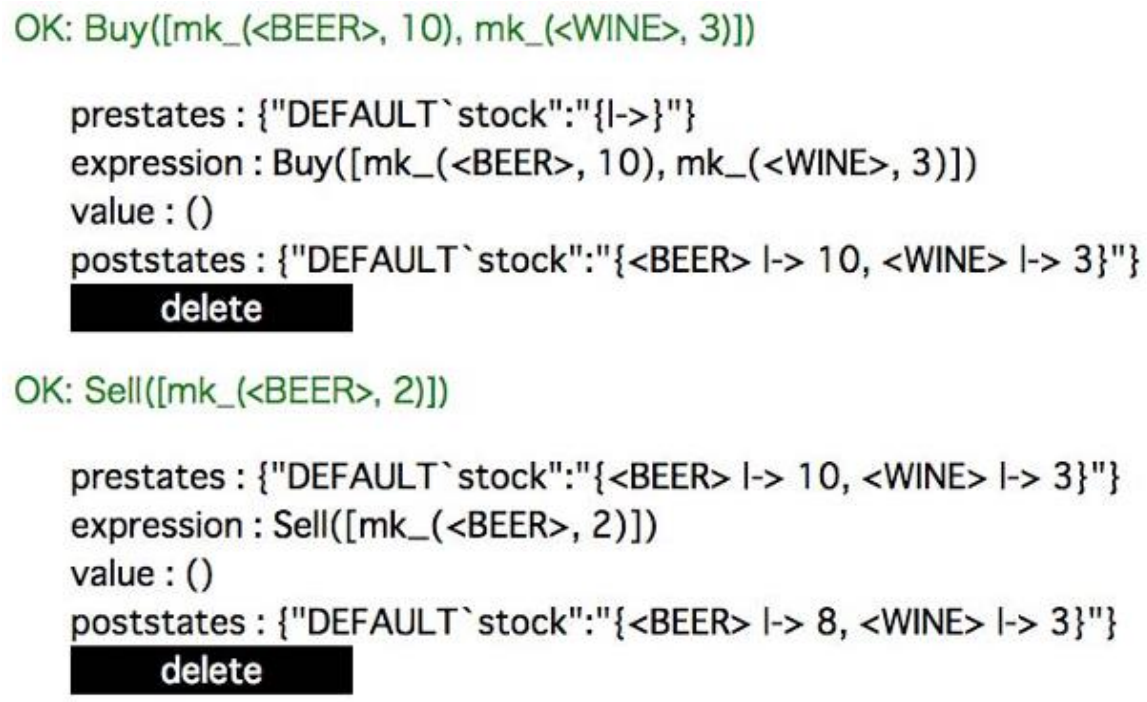
5.6.1 Ngôn ngữ VDM

Ngôn ngữ VDM là là một trong những phương pháp lâu đời nhất cho việc phát triển các hệ thống dựa trên máy tính. Có nguồn gốc từ phòng thí nghiệm IBM Viên (Áo) và được ra đời năm 1970, nó đã phát triển thành thành một công cụ bao gồm nhiều kỹ thuật và công cụ khác nhau dựa trên ngôn ngữ đặc tả hình thức – VDM-SL. Bản mở rộng VDM++ hỗ trợ các mô hình hướng đối tượng và các hệ thống song song.

Các nguồn hỗ trợ cho VDM bao gồm các các công cụ thương mại và giáo dục cho việc phân tích các mô hình, kiểm thử, chứng minh các thuộc tính của mô hình và tạo ra mã nguồn từ các mô hình VDM đã được xác thực.

VDM có một đóng góp quan trọng trong ngành công nghiệp, bản thân nó và các công cụ đi kèm có vai trò thúc đẩy sự phát triể của nghiên cứu hình thức, đóng góp đáng kể vào

các kỹ thuật của nhiều hệ thống quan trọng, các trình biên dịch, các hệ thống song song và logic trong khoa học máy tính.



5.7 Semi/Informal method

Các cách tiếp cận khác nếu không phải phải là phương pháp hình thức thì ta có thể coi chúng là không hình thức (như các biểu đồ dòng chảy – flow-charting) hoặc là bán hình thức (như UML).

Cách tiếp cận bán hình thức có nghĩa là chúng không xuất hiện dưới dạng các kí hiệu toán học, tuy nhiên ta có thể dùng các công cụ tạo ra chúng để chuyển đổi một phần hoặc toàn bộ các sơ đồ, logic thành các biểu diễn toán học khác nhau.

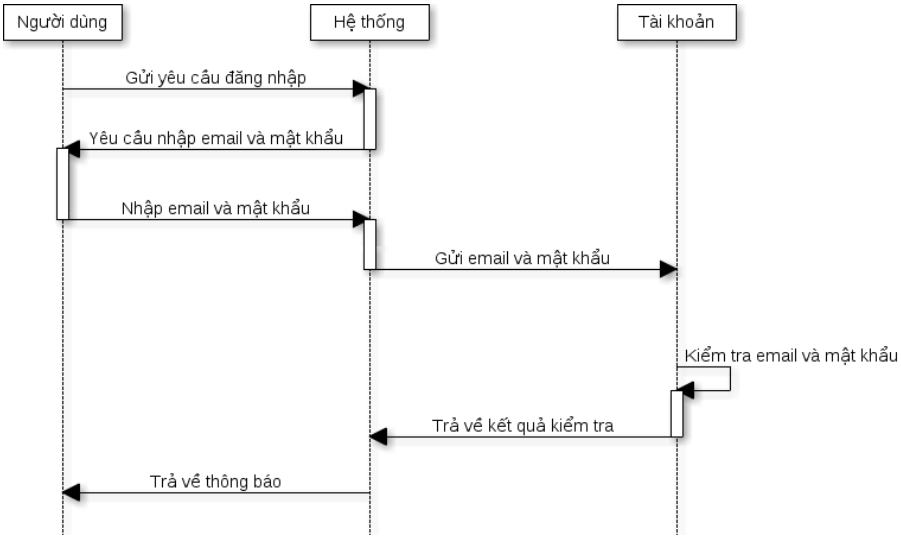
5.7.1 Ngôn ngữ mô hình hóa thống nhất - UML

UML là một cách tiếp cận bán hình thức, được sử dụng trong hầu hết các giai đoạn của quá trình phát triển phần mềm.

Trong quá trình đặc tả yêu cầu phần mềm người ta thường sử dụng các sơ đồ tuần tự và sơ đồ hoạt động

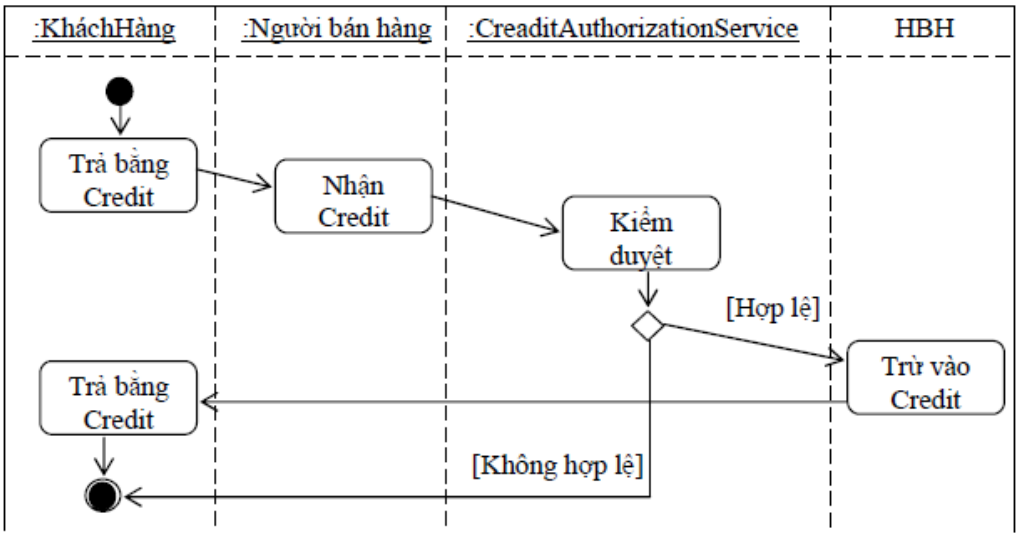
Sơ đồ tuần tự:

Là bản vẽ mô tả sự tương tác của các đối tượng để tạo nên các chức năng của hệ thống. Bản vẽ này mô tả sự tương tác theo thời gian nên rất phù hợp với việc sử dụng để thiết kế và cài đặt chức năng cho hệ thống phần mềm.



Sơ đồ hoạt động:

Là bản vẽ tập trung vào mô tả các hoạt động, luồng xử lý bên trong hệ thống. Nó có thể được sử dụng để mô tả các qui trình nghiệp vụ trong hệ thống, các luồng của một chức năng hoặc các hoạt động của một đối tượng.



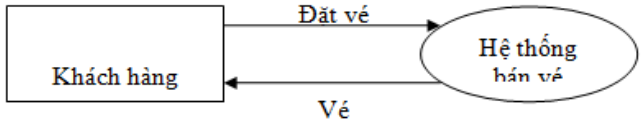
5.7.2 Sơ đồ dòng chảy dữ liệu - DFD

Là một cách tiếp cận không hình thức, được sử dụng rộng rãi, tập trung vào các yêu cầu chức năng sẽ được thực hiện trong hệ thống.

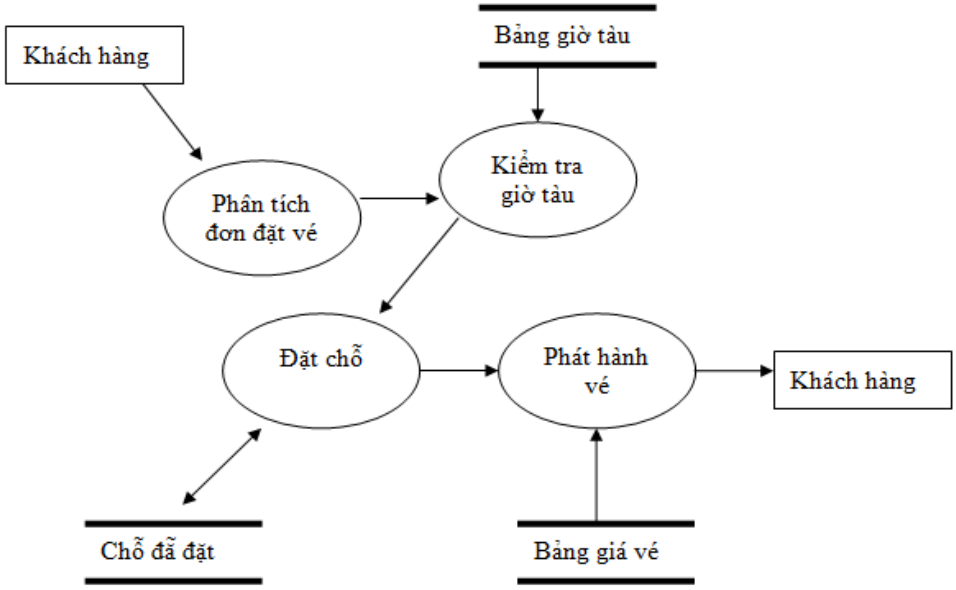
Phương pháp này xem hệ thống như một mạng lưới các dữ liệu chuyển đổi thông qua các luồng dữ liệu,

Thường được sử dụng bằng cách chia nhỏ các chức năng cần mô hình hóa và áp dụng DFD cho các phần được chia.

Phương pháp Phân tích và Thiết kế hệ thống (SSAD) sử dụng DFD để tổ chức thông tin và hướng dẫn phân tích



DFD mức 0



DFD mức 1

5.8 Petri net

Là một phương pháp hình thức (formal) được dùng để xác định các hoạt động sẽ được thực hiện trong môi trường đa xử lý hoặc môi trường đa nhiệm.

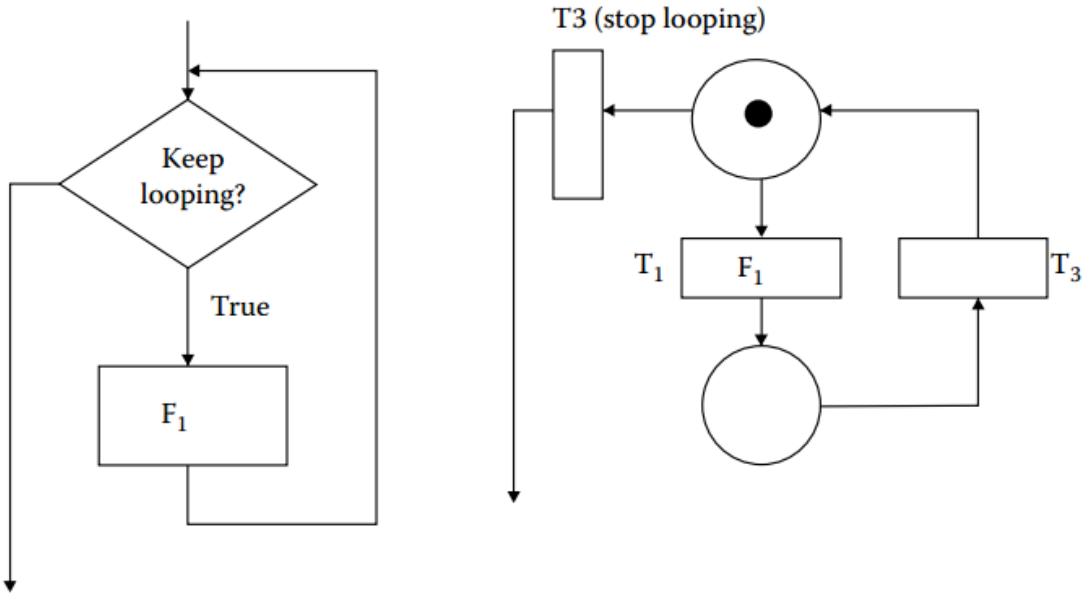
Phương pháp này dùn các kí hiệu đồ họa bao gồm:

- Các bong bóng tròn được gọi là "place" dùng để đại diện cho kho dữ liệu hoặc các xử lý.
- Các hộp chữ nhật đại diện cho các quá trình chuyển đổi hoặc các hoạt động.
- Các quy trình và chuyển tiếp (transition) được đánh số và được nối bằng các đường vòng cung

Petri net được xử dụng cho việc thể hiện các hệ thống đa xử lý và đa tác vụ, đặc biệt là khi các hàm thực hiện có mức độ phức tạp thấp.

Lợi ích khi sử dụng pertri net là bởi vì chúng thuần túy toán học, cho nên các kỹ thuật được dùng cho việc tối ưu và hình thức hóa chương trình có thể được sử dụng cho các đặc tả sử dụng phương pháp này.

Trong ví dụ là một minh họa về vòng lặp while sử dụng petri net.



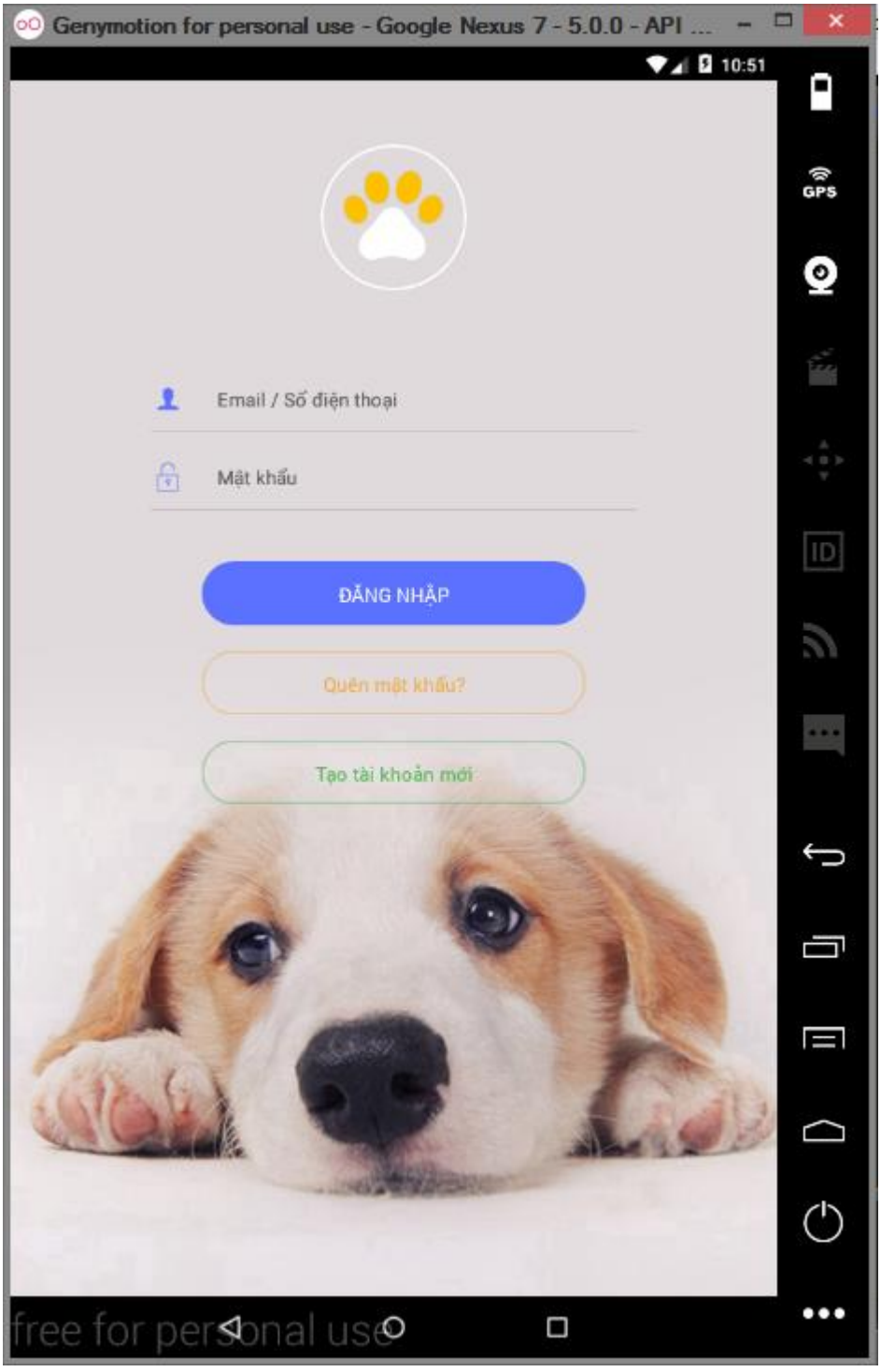
VI. Tài liệu đặc tả cho dự án demo

Trong phần này, nhóm em xin trình tài liệu đặc tả yêu cầu phần mềm cho dự án demo "Mạng xã hội bán PetOnline".

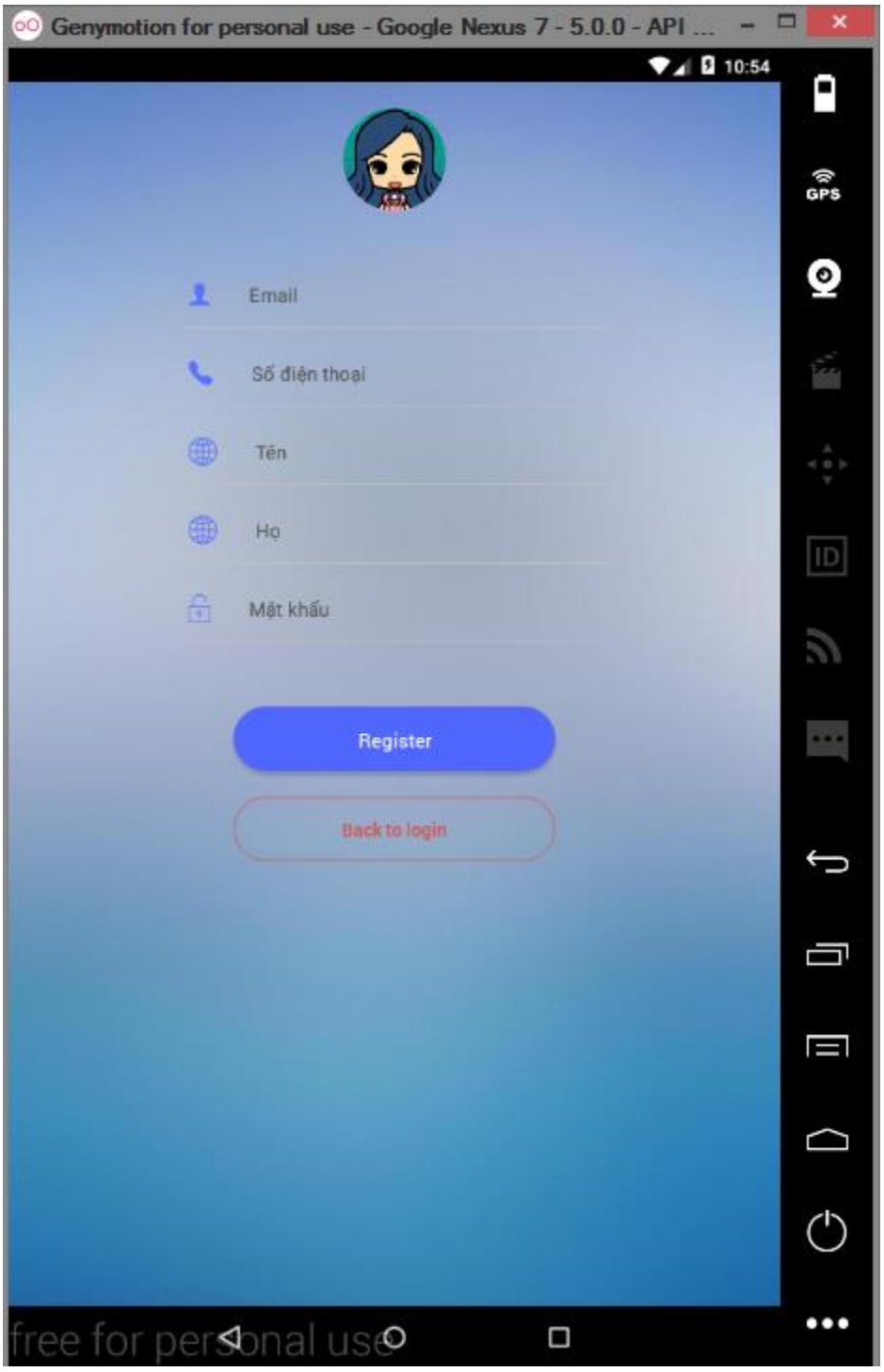
Mời thầy xem các tài liệu nằm trong thư mục Demo/Specifications/ bao gồm 2 tài liệu là "Đặc tả PetOnline.docx" và "Pets_APIv1.docx" nằm trong 2 thư mục con lần lượt API_Specification và PetOnline_Specification.

Kết quả khi chạy chương trình demo:

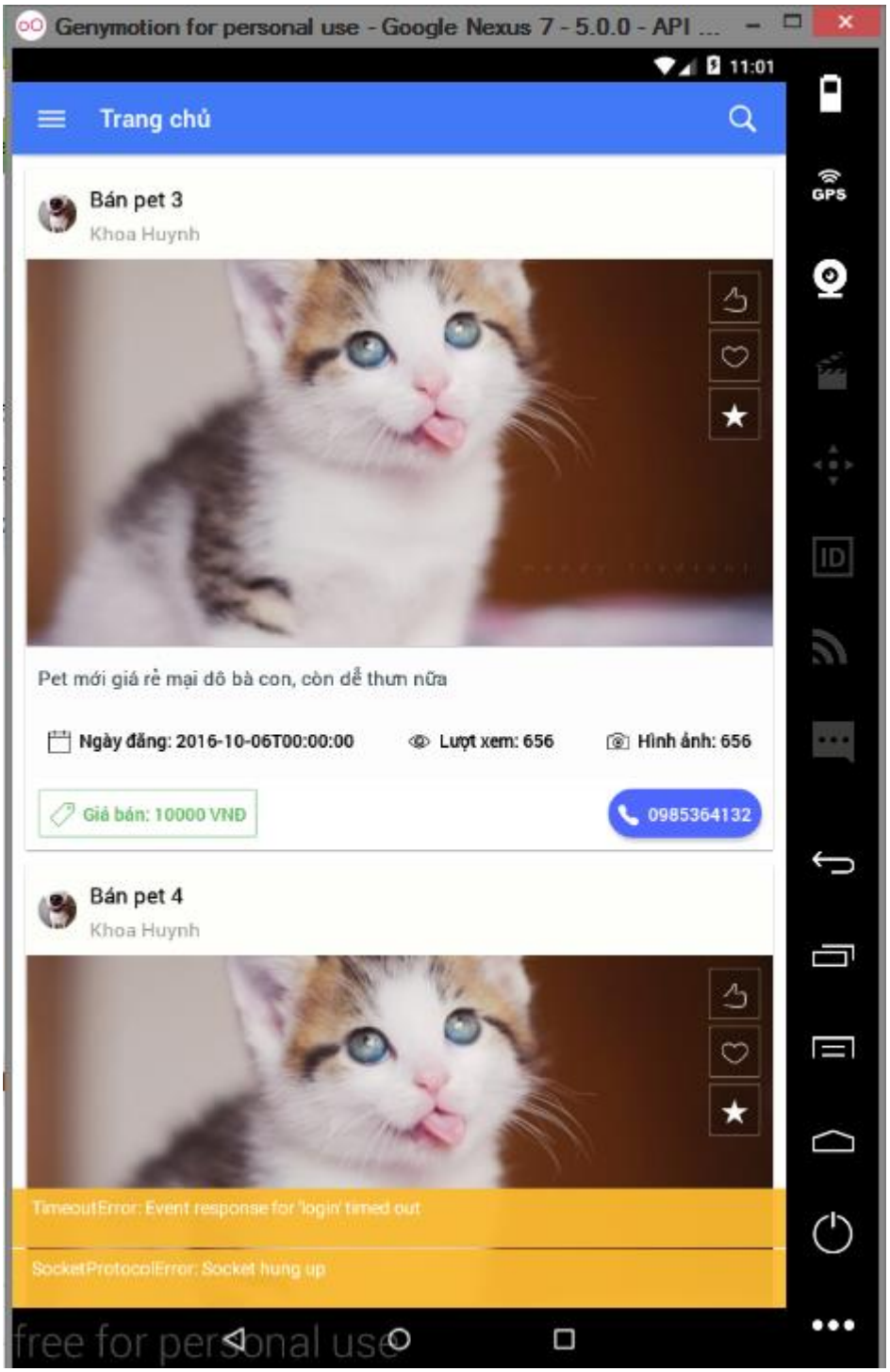
Giao diện đăng nhập



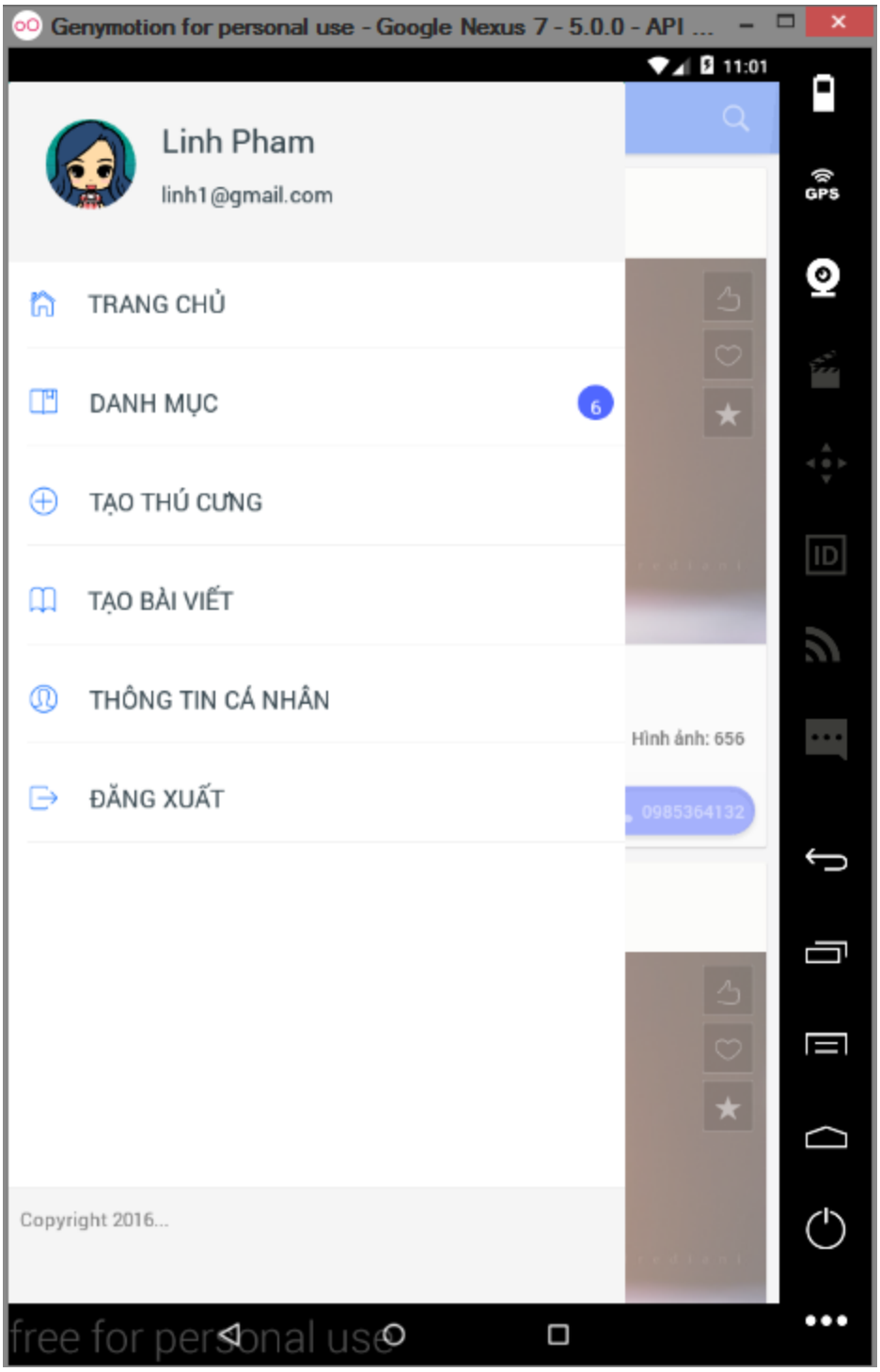
Giao diện đăng kí



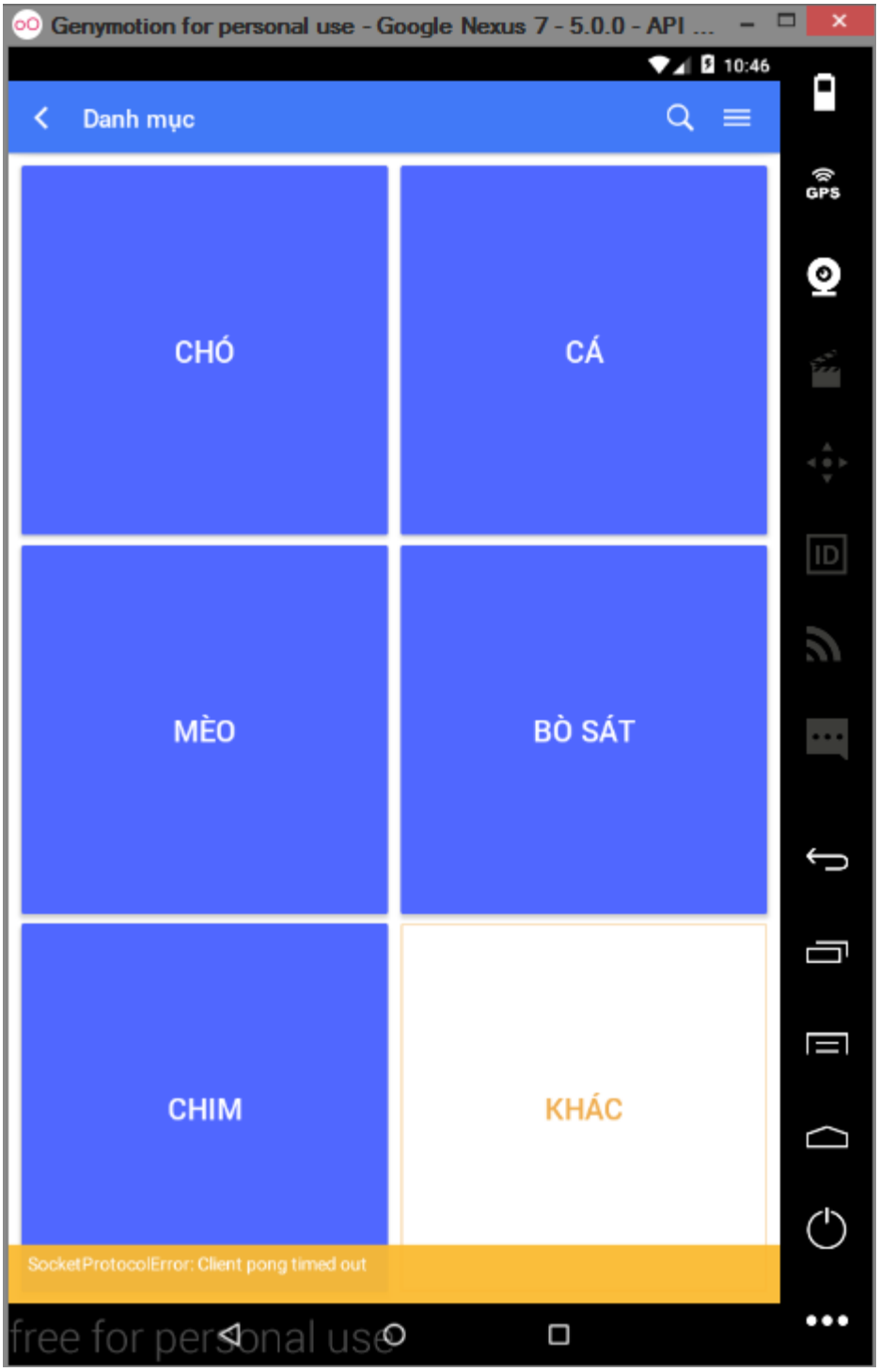
Giao diện trang chủ



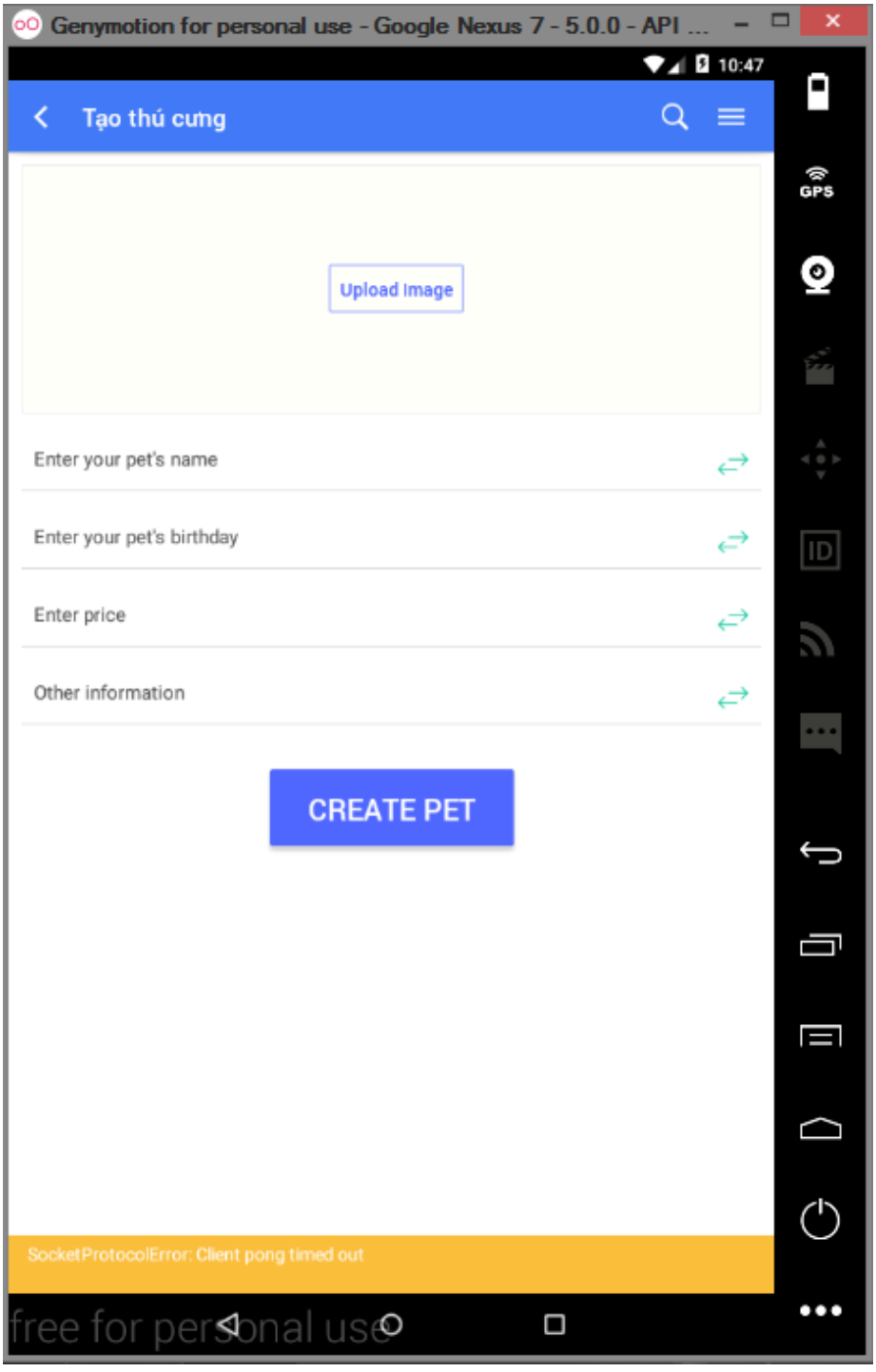
Giao diện panel



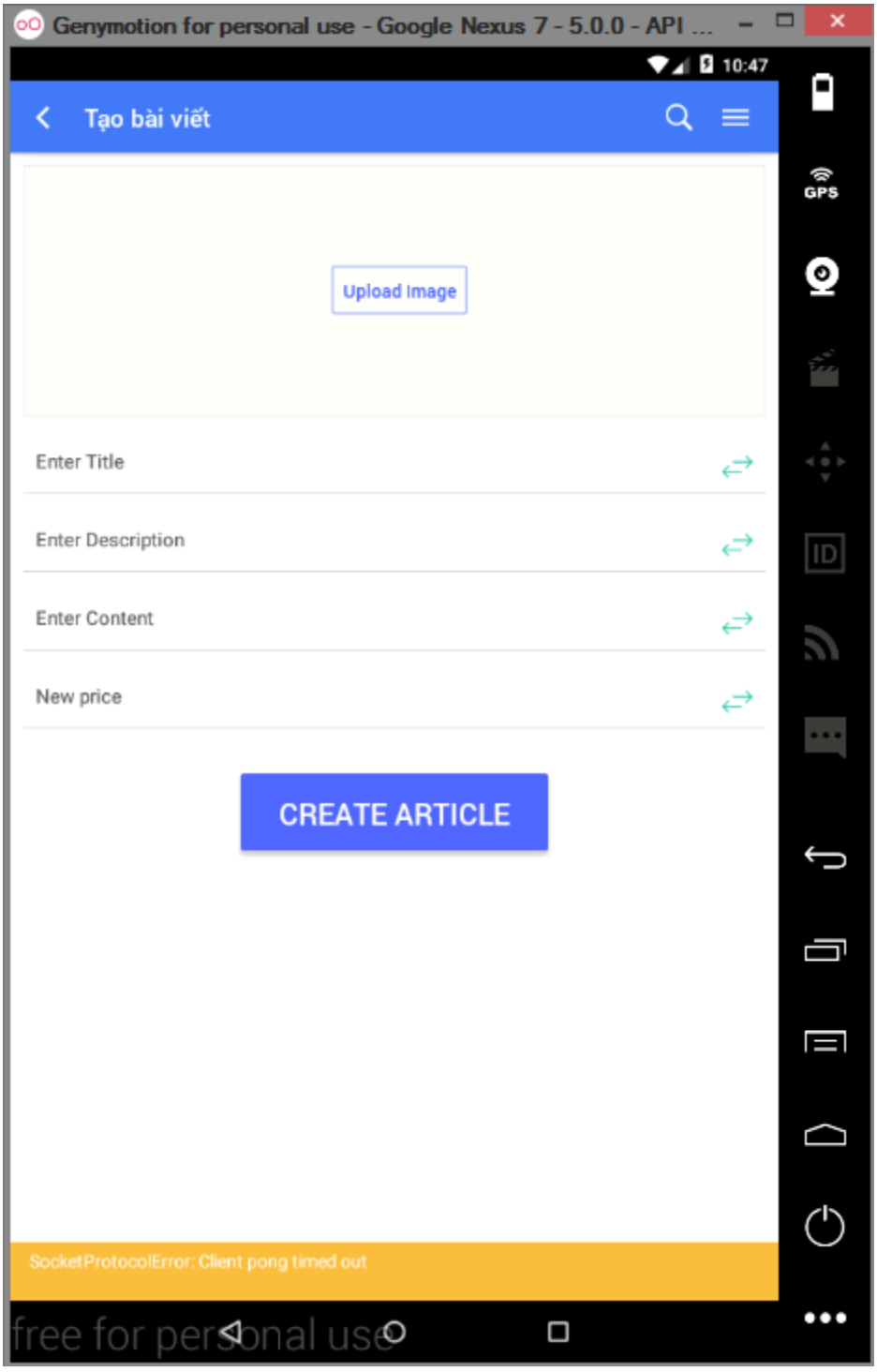
Giao diện xem danh mục Pet



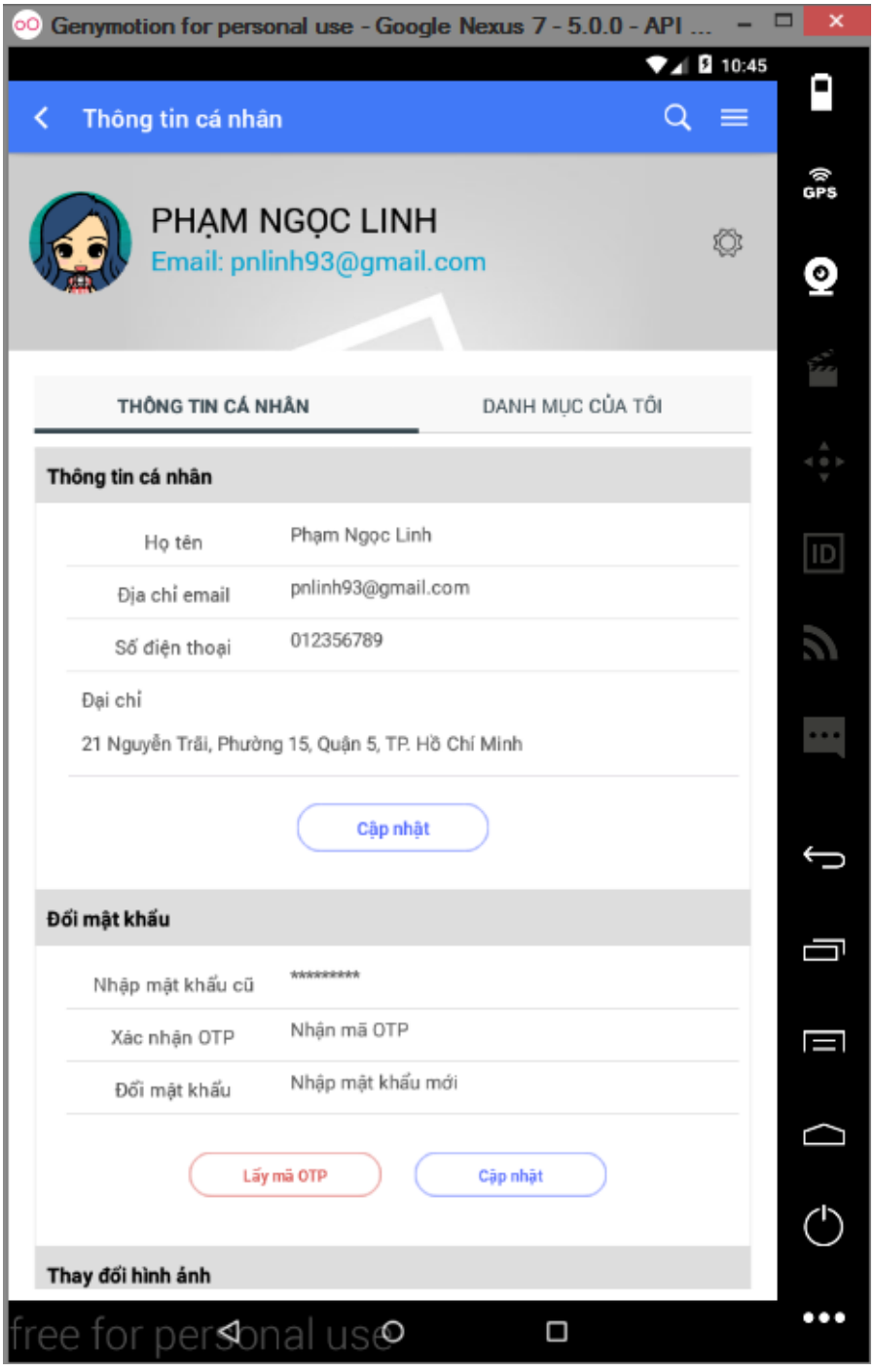
Giao diện khi cho yêu cầu tạo pet:



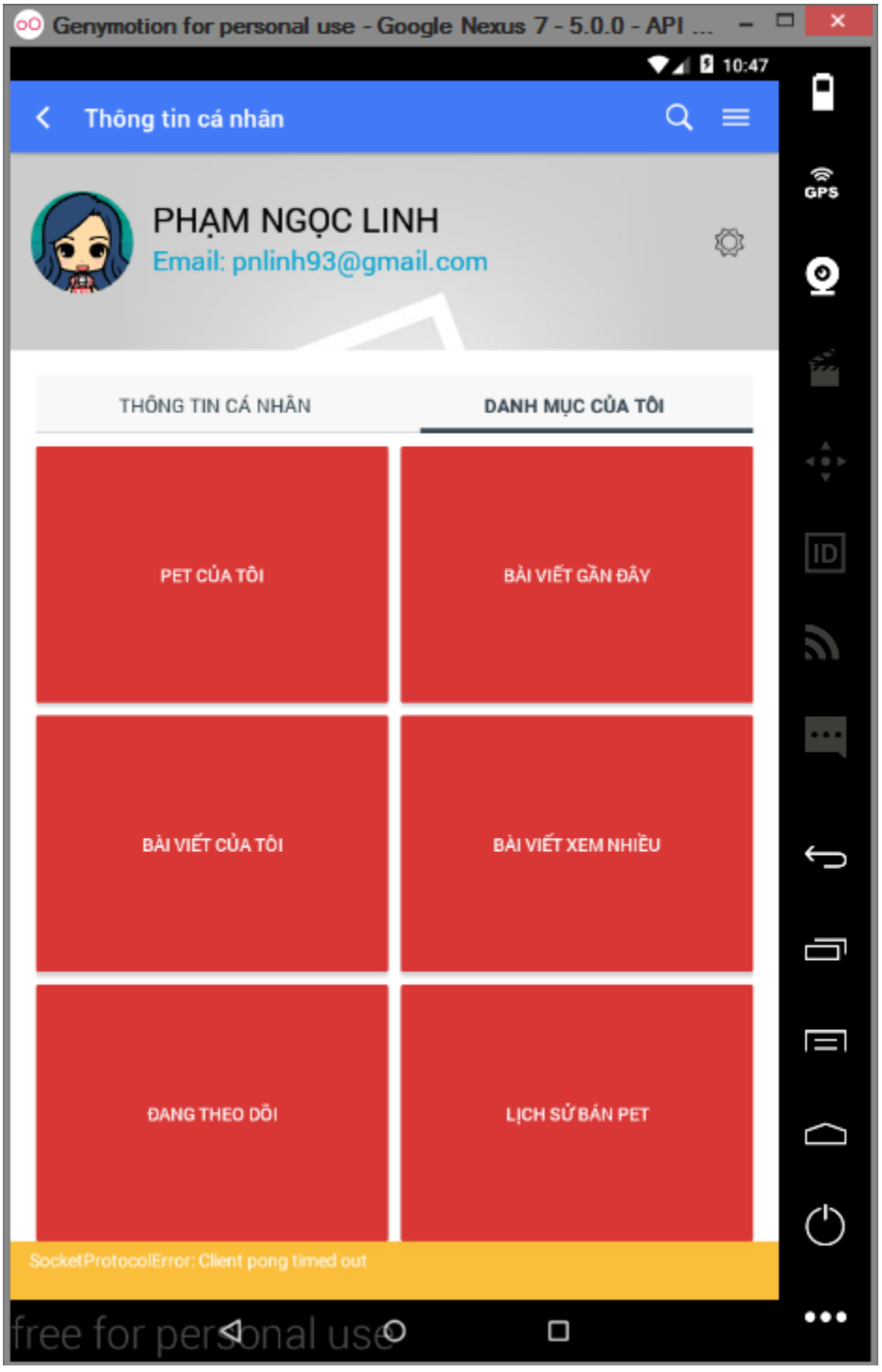
Giao diện cho yêu cầu tạo bài viết



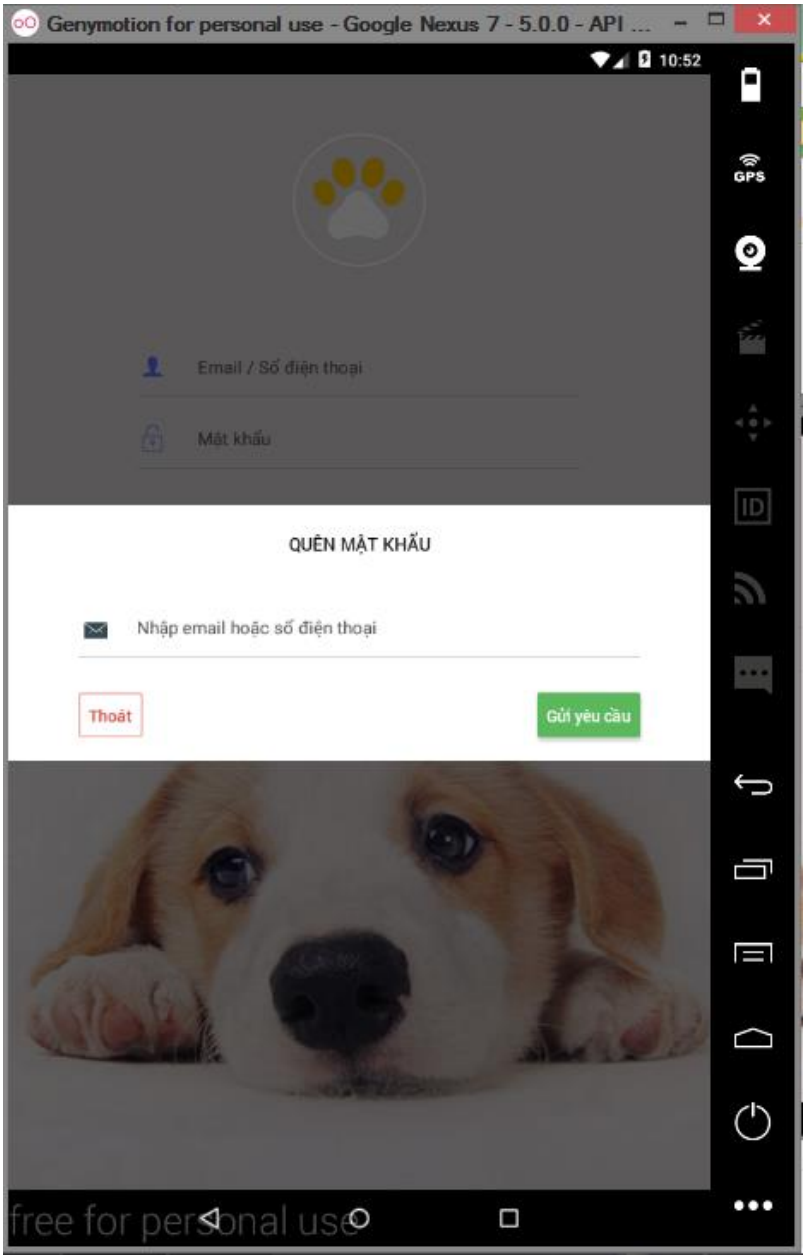
Giao diện thông tin người dùng



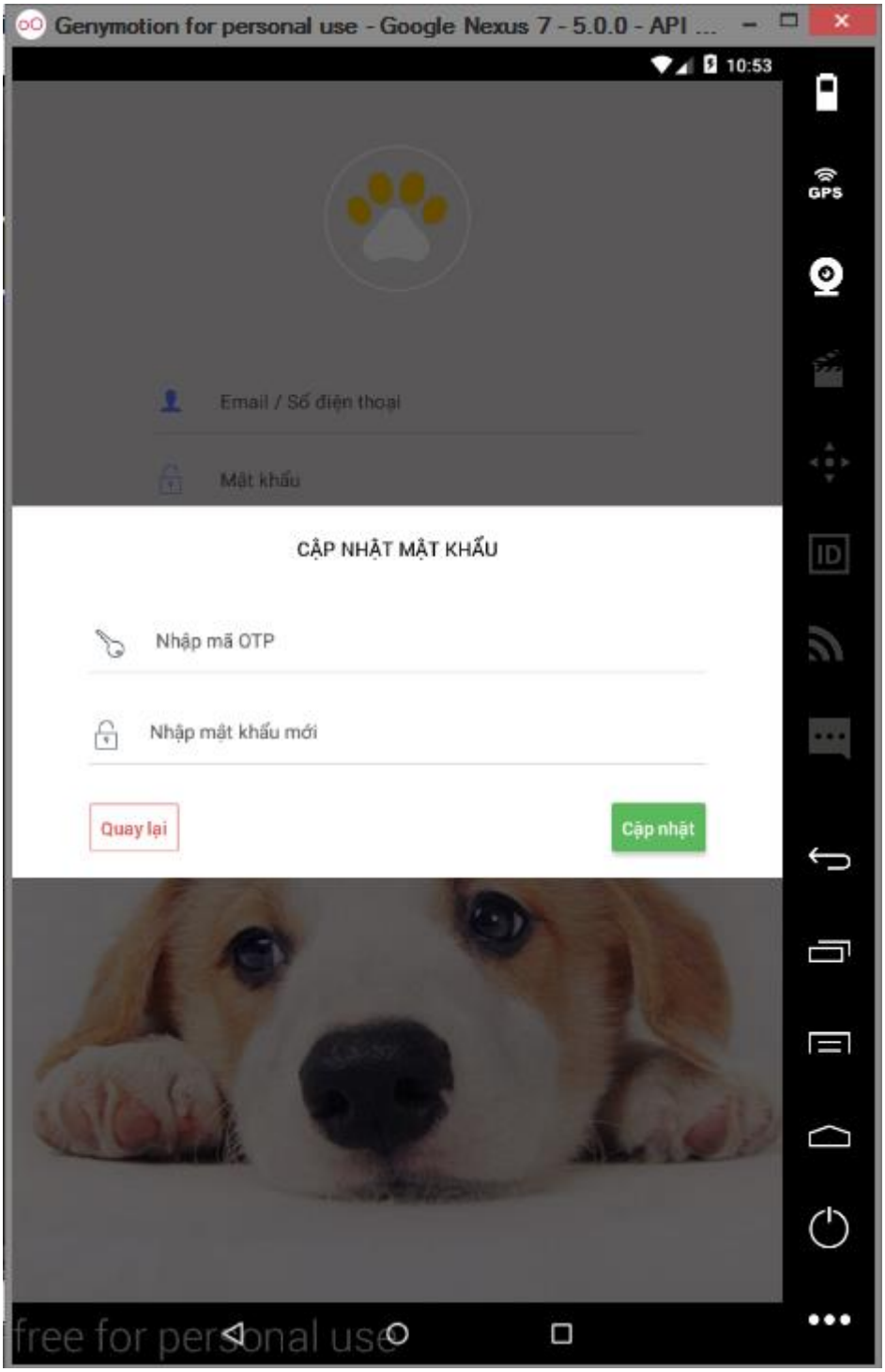
Giao diện trang Danh mục của tôi



Giao diện đổi mật khẩu (1)



Giao diện đổi mật khẩu (2)



VII. Các tài liệu tham khảo

Sách:

- What everything engineer should know.
- Advanced Software Engineering - Formal Methods.
- Informal, Fomal, Semiformal Method for Software Requirement Specification.