BỘ GIÁO DỤC VÀ ĐÀO TẠO
TRƯỜNG ĐẠI HỌC THĂNG LONG

-----o0o-----

KHÓA LUẬN TỐT NGHIỆP

TÌM HIỂU VỀ PUSH NOTIFICATION XÂY DỰNG ỨNG DỤNG NHẮC LỊCH THI CHO SINH VIÊN THĂNG LONG TRÊN NỀN TẢNG ANDROID

Giáo viên hướng dẫn : Ths. Lê Minh Tuấn
Sinh viên thực thiện : Phạm Trung Kiên – A14538
Hồ Chí Nghĩa – A14982
Chuyên ngành : Công Nghệ Thông Tin

HÀ NỘI - 2014

LỜI NÓI ĐẦU

Trong sự phát triển mạnh mẽ của công nghệ thông tin nói chung và công nghệ di động nói riêng, các ứng dụng di động đang dần đóng vai trò quan trọng trong việc ứng dụng công nghệ thông tin phục vụ trong cuộc sống con người. Công nghệ di động đã dần chiếm lĩnh vị trị trong cuộc sống, chiếm ưu thế về số lượng ứng dụng trên các thiết bị di động như điện thoại thông minh, máy tính bảng, tivi thông minh...Có thể nói nổi bật lên trong đó là nền tảng Android mạnh mẽ do có khả năng tương thích với nhiều thiết bị.

Đi cùng với việc nâng cao chất lượng đào tạo của trường Đại học Thăng Long là việc ứng dụng hiệu quả các thành tựu công nghệ thông tin vào quản lý đào tạo cũng như trực tiếp đào tạo.

Trên cơ sở những kiến thức đã được học trong các môn học tại trường cũng như trong khuôn khổ của một đề tài khóa luận tốt nghiệp, nhóm tác giả đã xây dựng được một phần mềm chạy trên nền tảng di động Android với mục đích cập nhật tự động lịch thi học kỳ cho sinh viên Thăng Long. Cùng với đó là tìm hiểu và mở rộng phạm vi kiến thức của mình.

Khóa luận này được trình bày thành 5 chương lớn:

- **Chương 1**: Giới thiệu dự án. Chương này trình bày lý do nghiên cứu và phương pháp thực hiện khóa luận;
- **Chương 2**: Hệ điều hành Android. Chương này giới thiệu căn bản về hệ điều hành Android, các đặc trưng và kiến trúc của hệ điều hành;
- **Chương 3**: Giới thiệu về công nghệ Push Notification và Service trong Android.
- **Chương 4**: Ứng dụng nhắc lịch thi. Chương này trình bày về quá trình phân tích thiết kế ứng dụng;
- **Chương 5**: Các kỹ thuật xử lý quan trọng. Chương này trình bày chi tiết hơn về các kỹ thuật được sử dụng trong xây dựng ứng dụng, quá trình gửi nhận và xử lý dữ liệu của công nghệ Push Notification.
- Kết luận và hướng phát triển;
- Các tài liệu tham khảo.

Sau khi hoàn thành tài liệu này, nhóm tác giả mong muốn đây cũng sẽ là một tà liệu tham khảo bổ ích cho các sinh viên đang và sẽ tìm hiểu về lập trình Android nó chung, cũng như công nghệ Push Notification nói riêng.

Chúng em xin chân thành cảm ơn thầy giáo Ths. Lê Minh Tuấn đã hướng dẫn và giúp đỡ chúng em thiết kế - xây dựng và hoàn thành khóa luận này.

Chúng em cũng xin gửi lời cảm ơn đến thầy giáo Nguyễn Đức Dân đã giúp đỡ chúng em về mặt kết nối đến dữ liệu lịch thi của nhà trường cùng với các thầy cô bộ môn Tin học đã tâm huyết dạy dỗ và đào tạo chúng em suốt những năm Đại học.

Hà Nội. ngày 12 tháng 04 năm 2014

Nhóm tác giả thực hiện: Phạm Trung Kiên

Hồ Chí Nghĩa

MỤC LỤC

DANH MỤC CÁC HÌNH MINH HOẠ

CHƯƠNG 1. GIỚI THIỆU DỰ ÁN

1.1. Tên đề tài

Tìm hiểu dịch vụ Push Notification, xây dựng ứng dụng nhắc lịch thi cho sinh viên Đại học Thăng Long.

1.2. Lý do nghiên cứu

Hiện nay, việc ứng dụng CNTT vào trong công tác đào tạo cũng như quản lý của trường Đại học Thăng Long đã trở nên phổ biến. Nhiều hệ thống, phần mềm được đưa vào hoạt động nhằm làm tăng hiệu quả công việc như Hệ thống đăng ký học, Hệ thống đào tạo trực tuyến, các phần mềm quản lý điểm, tài chính – kế toán...

Mặc dù đã có những bước phát triển trong suốt quá trình hoạt động, đi kèm với những thay đổi về công tác giảng dạy và quản lý đào tạo, các hệ thống phần mềm mới luôn được nâng cấp, thay thế những phần mềm hệ thống cũ bằng những phần mềm hệ thống mới đã đáp ứng tốt nhu cầu quản lý và đào tạo. Nhưng vẫn còn đâu đó sự thiếu sót và hạn chế của những hệ thống phần mềm hiện tại.

Một minh chứng cho sự hạn chế trên đó chính là hệ thống Đăng ký học trực tuyến của nhà trường. Một vài ví dụ đưa ra sau đây có thể chỉ ra sự hạn chế này:

– Trước khi kỳ thi bắt đầu, nhà trường thông báo về thời điểm công bố lịch thi chính thức, nhưng đến thời điểm đó lại xảy ra một số trục trặc về kỹ thuật khiến máy chủ không thể truy cập;

– Khi gặp sự cố, máy chủ không thể truy cập, sinh viên không theo dõi được lịch thi của mình, buộc ngay khi có được dữ liệu về lịch thi, sinh viên phải tự lưu trữ lại bằng cách ghi chép lại lịch thi hoặc chụp ảnh... gây bất tiện cho sinh viên;

– Khi có một sự thay đổi về lịch thi thì sinh viên không được cảnh báo hay có thông báo từ hệ thống, sinh viên phải tự cập nhật liên tục lịch thi của mình;

– Trước khi vào phòng thi sinh viên không nhớ phòng thi, thậm chí ca thi của mình, sinh viên phải sử dụng các thiết bị kết nối với hệ thống để có thể biết được thông tin, ngay lúc này, nếu hệ thống không hoạt động thì sinh viên sẽ gặp nhiều khó khăn;

– Do trong quá trình ôn thi căng thẳng, sinh viên quên mất lịch thi.

Từ những hạn chế trên đã thúc đẩy chúng tôi phát triển một ứng dụng nhắc lịch thi, đồng thời nghiên cứu phương thức tốt nhất để đáp được sự tiện lợi cho sinh viên trong quá trình học tập tại trường Đại học Thăng Long.

Khi hoàn thành đề tài này, chúng tôi sẽ có được một ứng dụng nhắc lịch thi cho sinh viên chạy được trên các thiết bị di động cầm tay như điện thoại, máy tính bảng...và đảm bảo được các yêu cầu cơ bản như:

– Sinh viên sẽ có được lịch thi ngay khi lịch thi chính thức được ban hành;

– Dữ liệu về lịch thi khi có bất kỳ sự thay đổi sẽ được thông báo tới cho sinh viên, giúp sinh viên chủ động trong việc ôn tập và thi cử;

– Dữ liệu về lịch thi được lưu trữ trên thiết bị và có thể truy xuất bất kỳ lúc nào, bất kỳ nơi đâu.

1.3. Phương pháp thực hiện

Để thực hiện đề tài này, chúng tôi sẽ sử dụng công nghệ Push Notification và phát triển ứng dụng trên nền tảng Android.

Công nghệ Push Notification sẽ giúp đảm bảo được việc dữ liệu về lịch thi mới sẽ luôn được cập nhật và thông báo tới sinh viên.

Nền tảng Android cho phép ứng dụng có thể chạy được trên nhiều thiết bị di động như điện thoại thông minh hay máy tính bảng. Cùng với sự phát triển về nền tảng di động Android, nhiều thiết bị di động khác trong tương lai chạy hệ điều hành này sẽ có thể sử dụng được ứng dụng hữu ích này.

Hệ thống này dựa trên mô hình khách chủ (client/server) bao gồm 1 Web Service đóng vai trò là Server đảm bảo về dịch vụ truy xuất dữ liệu và một ứng dụng chạy trên thiết bị di động đóng vai trò là Client.

CHƯƠNG 2. HỆ ĐIỀU HÀNH ANDROID VÀ PUSH NOTIFICATION

2.1. Giới thiệu hệ điều hành Android

Android là một hệ điều hành di động dựa trên một phiên bản sửa đổi của Linux. Được phát triển vào năm 2005 với một dự án cùng tên "Android". Như một phần chiến lược của mình để lấn sâu vào lĩnh vực di động Google Android đã mua về toàn bộ quá trình phát triển cũng như đội phát triển nó. Đây là con át chủ bài của Google để cạnh tranh thị phần hệ điều hành di động với Apple.

Google Android muốn mở và miễn phí, vì vậy hầu hết các mã Android được được đưa ra dưới dạng mã nguồn mở Apache License, điều này tương đương với việc bất cứ ai muốn sử dụng Android có thể làm như vậy bằng cách tải về mã nguồn Android đầy đủ. Hơn nữa các nhà cung cấp (thường là những nhà phát triển phần cứng) có thể thêm phần mở rộng và tùy biến cho Android để phân biệt sản phẩm của họ với sản phẩm của những người khác. Điều này đơn giản làm cho mô hình phát triển Android rất hấp dẫn và do đó khơi dậy sự quan tâm của nhiều nhà cung cấp. Những nhà sản xuất coi Android như một giải pháp – họ sẽ tiếp tục thiết kế phần cứng của riêng mình và sử dụng Android như một hệ điều hành chính.

Ưu điểm chính của việc áp dụng Android là nó cung cấp một cách tiếp cận thống nhất để phát triển ứng dụng. Các nhà phát triển chỉ cần phát triển cho Android và các ứng dụng của họ có thể chạy trên nhiều thiết bị khác nhau, miễn là các thiết bị được hỗ trợ bằng cách sử dụng Android. Trong thế giới điện thoại thông minh ứng dụng là một phần quan trọng nhất của chuỗi thành công. Do đó các nhà sản xuất thiết bị coi Android như là hy vọng tốt nhất để thách thức sự tấn công của Apple.

2.2. Những đặc trưng của hệ điều hành Android

– **Application framework**: cho phép sử dụng lại và thay thế các thành phần trong lập trình ứng dụng;

– **Dalvik virtual machine**: tối ưu hóa cho thiết bị di động;

– **Intergrated browser**: trình duyệt tích hợp, dựa trên cơ chế WebKit mã nguồn mở;

– **SQLite**: sơ sở dữ liệu trong môi trường di động;

– **Media support**: hỗ trợ các định dạng audio, video và hình ảnh thông dụng;

– **GSM Telephony**: mạng điện thoại di động (phụ thuộc vào phần cứng);

– **Bluetooth, EDGE, 3G, và WiFi**: các chuẩn kết nối dữ liệu (phụ thuộc vào phần cứng);

– **Camera, GPS, la bàn, và gia tốc kế**: (phụ thuộc vào phần cứng);

– **Môi trường phát triển phong phú**: bao gồm thiết bị mô phỏng, công cụ cho việc dò tìm lỗi, bộ nhớ và định hình hiệu năng và một plugin cho Eclipse IDE.

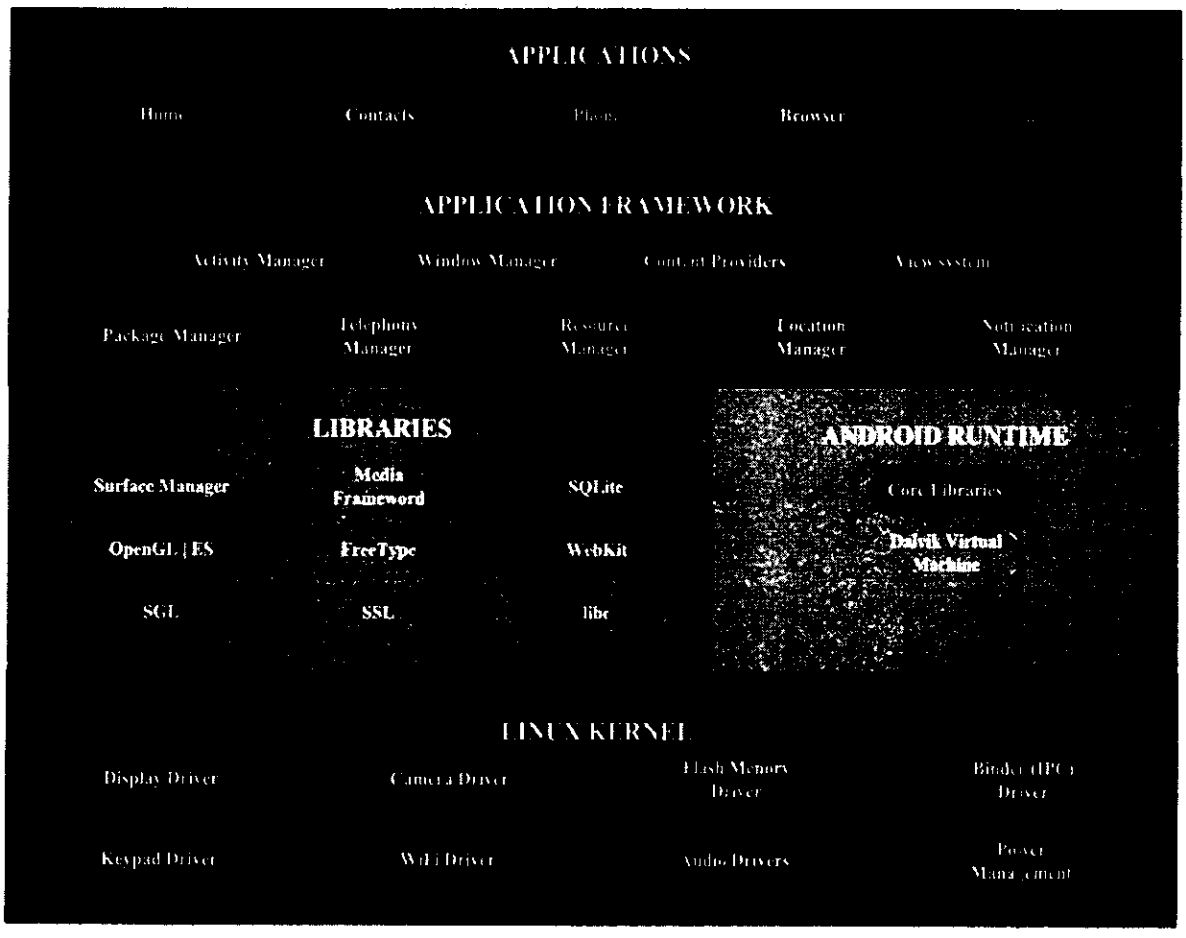
2.3. Các tính năng hỗ trợ sẵn trong hệ điều hành Android

Các tính năng được hỗ trợ tùy thuộc vào cấu hình phần cứng và phần mềm.

- **Storage**: Sử dụng SQLite, một cơ sở dữ liệu quan hệ nhẹ cho việc lưu trữ dữ liệu;
- **Connectivity**: Hỗ trợ GSM/EDGE, IDEN, CDMA, EV-DO, UMTS, Bluetooth (bao gồm AD2P và AVRCP), Wifi, LTE và WiMAX;
- **Messaging**: hỗ trợ cả SMS và MMS;
- **Web browser**: Dựa trên mã nguồn mở Webkit, cùng với công nghệ JavaScript V8 của Chrome;
- **Media support**: Bao gồm hỗ trợ các phương tiện truyền thông sau: H.263, H.264 (Trong 3GP hoặc MP4 container), MPEG-4 SP, AMR, AMR-WB (3GP container), AAC, HE-AAC (MP4 hoặc 3GP container), MP3, MIDI, Ogg Vorbis, WAV, JPEG, PNG, GIF và BMP;
- **Hardware support**: Accelerometer Sensor, Camera, Digital Compass, Proximity Sensor và GPS;
- **Multi-touch**: Hỗ trợ màn hình cảm ứng đa điểm;
- **Multi-tasking**: Hỗ trợ ứng dụng đa tác vụ;
- **Flash support**: Android 2.3 hỗ trợ Flash 10.1;
- **Tethering**: Hỗ trợ kết nối internet không dây/có dây.

2.4. Kiến trúc và các thành phần trong hệ điều hành Android

Mô hình sau thể hiện đầy đủ kiến trúc các thành phần của hệ điều hành Android.



Hình 2.1. Kiến trúc các thành phần của hệ điều hành Android[1]

Linux kernel (nhân Linux)

Kernel Linux hoạt động như một lớp trừu tượng hóa giữa phần cứng và tầng dưới của phần mềm. Lớp này chứa tất cả các thiết bị mức thấp điều khiển các thành phần phần cứng khác nhau của một thiết bị Android.

Libraries

Libraries bao gồm một tập hợp các thư viện lập trình chứa mã lệnh cung cấp những tính năng và thao tác chính trên hệ điều hành. Một số các thư viện cơ bản được liệt kê dưới đây:

– **System C library** – a BSD-derived triển khai các thư viện hệ thống ngôn ngữ C chuẩn, được nhúng vào các thiết bị dựa trên hệ điều hành Linux;

– **Media Libraries** – Dựa trên PacketVideo's OpenCORE; thư viện này hỗ trợ cho việc chơi nhạc, quay phim, chụp hình theo các định dạng file MPEG4, H.264, MP3, AAC, AMR, JPG, and PNG;

[1] Theo http://developer.android.com (trang web chính thức của Google Android dành cho các nhà phát triển)

– **Surface Manager** – Quản lý truy cập đến các hệ thống con hiển thị cũng như các lớp đồ họa 2D, 3D từ tầng ứng dụng;

– **LibWebCore** – Thư viện được dùng để tạo nên thành phần webview trong Android và có thể nhúng được vào nhiều ứng dụng;

– **SGL** – Thư viện hỗ trợ đồ hoạ 2D;

– **3D libraries** – Thư viện đồ họa 3D;

– **FreeType** - bitmap and vector font rendering;

– **SQLite** – Một cơ sở dữ liệu nhỏ được dùng cho các thiết bị cầm tay có bộ nhớ hạn chế. SQLite không có quan hệ như các cơ sở dữ liệu khác.

Android runtime

Tại cùng một tầng với Libraries, Android runtime cung cấp một bộ lõi thư viện cho phép các nhà phát triển viết các ứng dụng Android bằng cách sử dụng ngôn ngữ lập trình java. Android runtime cũng bao gồm các máy ảo Dalvik, cho phép mọi ứng dụng Android chạy trong tiến trình riêng của mình. Dalvik là một máy ảo chuyên dụng được thiết kế đặc biệt cho Android và tối ưu hóa cho các thiết bị điện thoại di động với giới hạn bộ nhớ và CPU.

Application framework 2.4.3 Android runtime

Bằng cách cung cấp một nền tảng phát triển mở, Android cung cấp cho các nhà phát triển khả năng xây dựng các ứng dụng cực kỳ phong phú và sáng tạo. Nhà phát triển được tự do tận dụng các thiết bị phần cứng, thông tin địa điểm truy cập, các dịch vụ chạy nền, thiết lập hệ thống báo động, thêm các thông báo để các thanh trạng thái, và nhiều, nhiều hơn nữa.

Nhà phát triển có thể truy cập vào các API được sử dụng bởi các ứng dụng lõi. Các kiến trúc ứng dụng được thiết kế để đơn giản hóa việc sử dụng lại các API. Đưa ra những khả năng khác nhau của hệ điều hành Android vào ứng dụng để sử dụng chúng trong các ứng dụng của mình.

Cơ bản tất cả các ứng dụng là một bộ các dịch vụ và các hệ thống, bao gồm: các View (là dùng để hiển thị thông tin và để người dùng thao tác), Content Provider để chia sẻ dữ liệu giữa các ứng dụng, Resource Manager truy xuất tài nguyên, Notification Manager hiển thị các thông báo, Activity Manager quản lý chu trình sống của ứng dụng và điều hướng Activity.

Applications

Tại lớp trên cùng sẽ là các ứng dụng cho Android (như điện thoại, danh bạ, trình duyệt,...) cũng như các ứng dụng được tải về và cài đặt từ AndroidMarket hay bất lý ứng dụng nào bạn viết được tại tầng này.

2.5. Các khái niệm cơ bản trong lập trình ứng dụng Android

Activity

Một activity thể hiện một giao diện đồ họa người dùng. Ví dụ một activity có thể biểu diễn một danh sách các menu item để người dùng có thể chọn và có thể hiển thị ảnh cùng với tiêu đề. Một ứng dụng gửi tin nhắn văn bản có thể có một hoạt động là hiển thị một danh sách các liên hệ để gửi tin nhắn tới, hoạt động thứ hai là viết tin nhắn tới liên hệ được chọn, các hoạt động khác nữa là xem lại tin nhắn cũ hay thay đổi cài đặt. Mặc dù chúng làm việc cùng nhau để tạo thành một giao diện người dùng, mỗi activity độc lập với những cái khác.

Mỗi activity là một lớp con của lớp cơ sở Activity. Một ứng dụng có thể gồm chỉ một activity hay nhiều activity. Activity chính phải được hiển thị đầu tiên khi khởi động chương trình. Chuyển từ một activity sang activity khác bằng cách cho activity hiện thời khởi động activity kế tiếp.

Mỗi activity được vẽ vào một cửa sổ trên màn hình, mặc định sẽ lấp đầy màn hình, nhưng nó có thể nhỏ hơn màn hình và nằm trên các cửa sổ khác ví dụ như activity thông báo một thông tin gì đó.

Nội dung trực quan của cửa sổ được cung cấp bởi một cây phân cấp các đối tượng view dẫn xuất từ lớp View. Mỗi view điều khiển một khoảng hình chữ nhật cụ thể bên trong cửa sổ. View cha chứa và tổ chức bố cục các view con. Các view lá vẽ trong hình chữ nhật mà chúng điều khiển và đáp ứng lại các hành động người dùng trực tiếp ở khoảng trống này. Do đó, các view là nơi mà các tương tác của activity với người dùng diễn ra.

Ví dụ một view có thể hiển thị một hình ảnh nhỏ và khởi tạo một hoạt động khi người dùng nhấn vào hình ảnh đó. Android có một số view đã xây dựng sẵn mà bạn có thể sử dụng – gồm có các buttons, text fields, scroll bars, menu items, check boxes... Một cây phân cấp view được đặt trong một cửa sổ của activity bằng phương thức Activity.setContentView(). Content view là đối tượng View ở gốc của cây phân cấp.

Class cơ sở Activity định nghĩa một loạt các sự kiện mà điều chỉnh vòng đời của một hoạt động. Class Activity định nghĩa các sự kiện sau đây:

- onCreate(): Được gọi khi hoạt động được tạo ra lần đầu tiên;
- onStart(): Được gọi khi hoạt động trở nên hữu hình so với người dùng;
- onResume(): Được gọi khi hoạt động bắt đầu tương tác với người sử dụng;

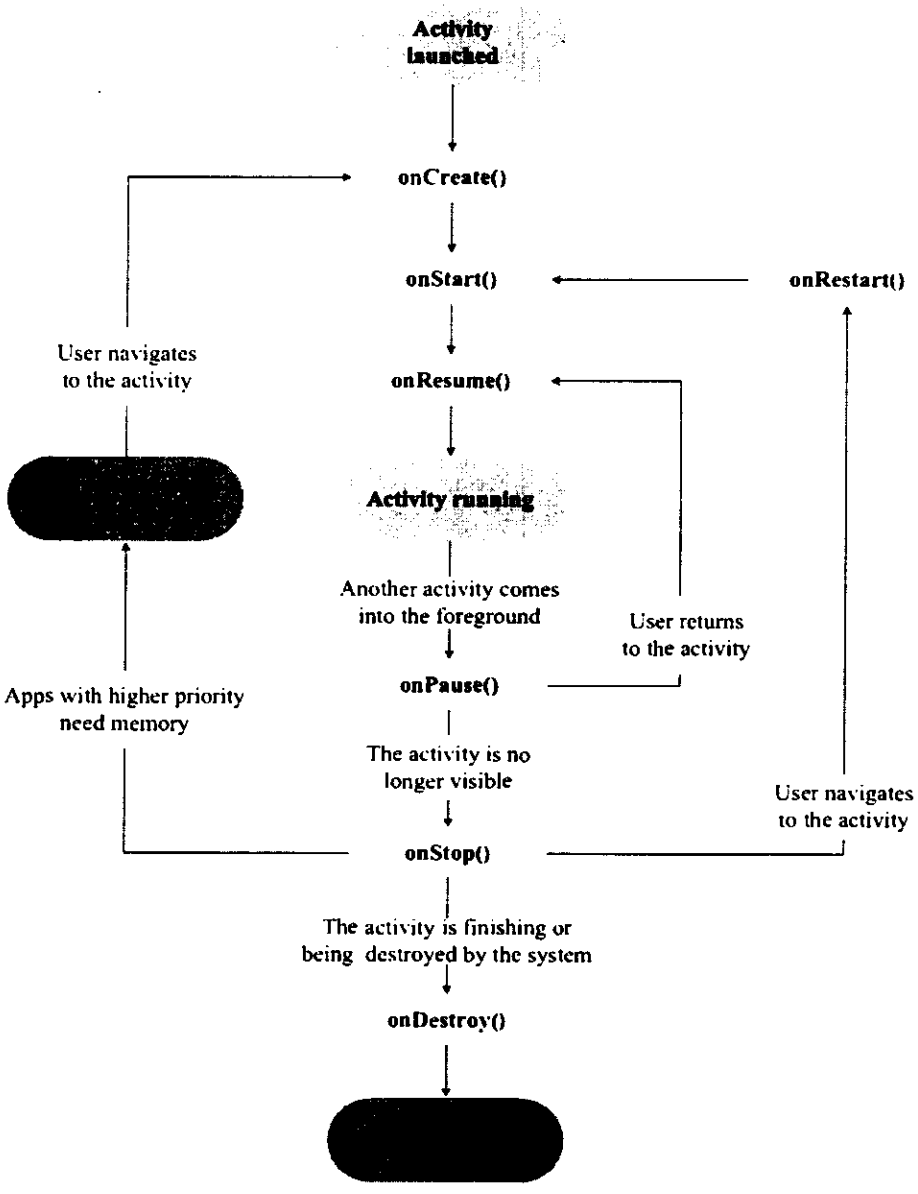
– onPause(): Được gọi để dừng các hoạt động hiện tại và nối lại các hoạt động trước đó;

– onStop(): Được gọi khi hoạt động không còn hiển thị với người dùng;

– onDestroy(): Được gọi trước khi hoạt động bị phá hủy bởi hệ thống (bằng tay hoặc bằng hệ thống để bảo tồn bộ nhớ);

– onRestart(): Được gọi khi hệ thống đã được dừng lại và khởi động lại một lần nữa.

Sau đây là sơ đồ các sự kiện trong vòng đời của một Activity:



Hình 2.2. Vòng đời của một Activity[2]

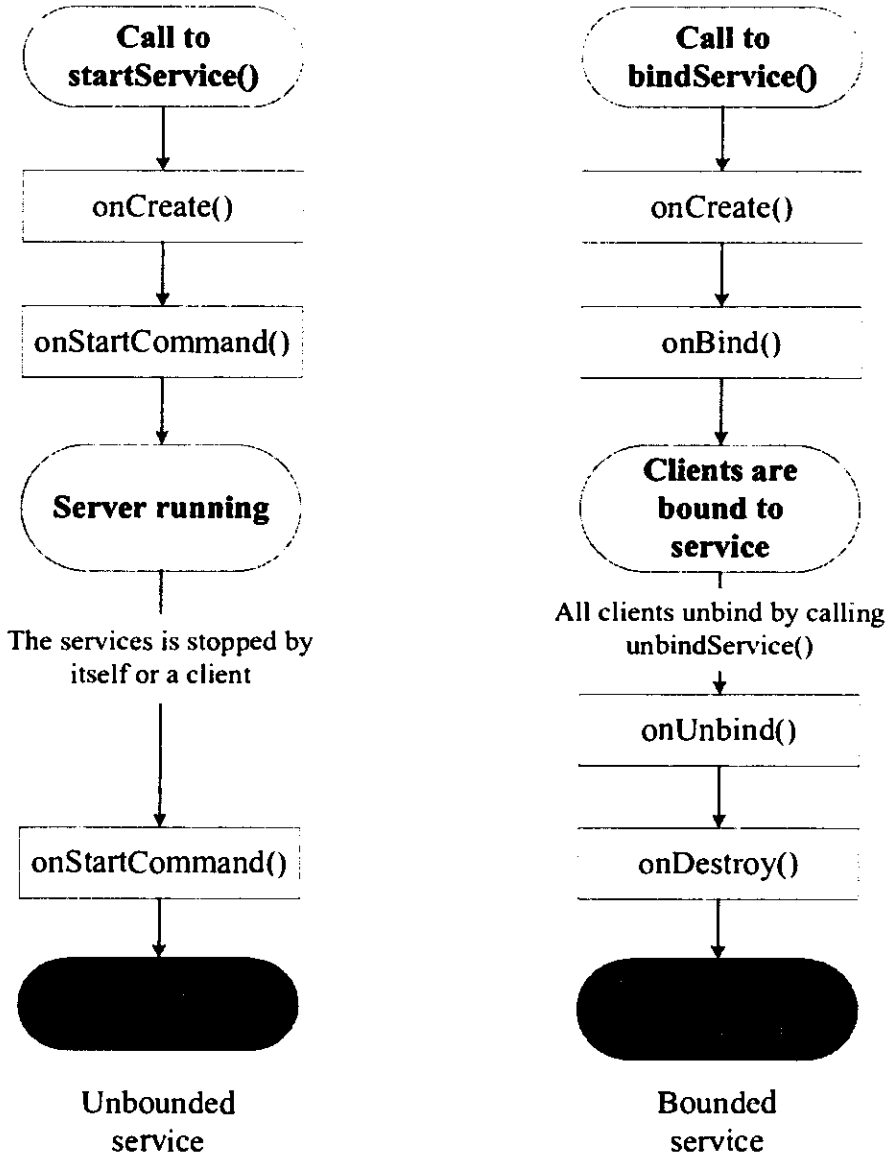
[2] Theo http://developer.android.com (trang web chính thức của Google Android dành cho các nhà phát triển)

Service

Một service không có giao diện trực quan, nó chạy trên nền trong một khoảng thời gian không xác định. Ví dụ một service có thể chơi nhạc nền, hay nó nạp dữ liệu trên mạng hay tính toán cái gì đó và cung cấp kết quả cho activity cần đến nó. Mỗi service mở rộng từ lớp cơ sở Service.

Trong khi kết nối, người sử dụng có thể giao tiếp với service thông qua giao diện mà service đó trưng ra. Ví dụ như trong service chơi nhạc, giao diện này có thể cho phép người dùng pause, rewind, stop và restart lại playback.

Giống như các activity và các thành phần khác khác, service chạy trong thread chính của tiến trình ứng dụng. Vì thế chúng không thể chặn những thành phần khác hay giao diện người dùng, chúng thường tạo ra các thead khác cho các nhiệm vụ hao tốn thời gian.Sơ đồ các sự kiện trong vòng đời của một service:



Hình 2.3. Vòng đời của một Service

Content provider

Một content provider tạo ra một tập cụ thể các dữ liệu của ứng dụng khả dụng cho các ứng dụng khác. Dữ liệu có thể được lưu trữ trong hệ thống file, trong một cơ sở dữ liệu SQLite, hay trong một cách khác nào đó. Content provider mở rộng lớp cơ sở ContentProvider để cài đặt một tập các chuẩn các phương thức cho phép các ứng dụng khác đạt được và lưu trữ dữ liệu của kiểu mà nó điều khiển. Tuy nhiên, các ứng dụng không gọi trực tiếp các phương thức này, chúng sử dụng một đối tượng ContentResolver và gọi các phương thức của nó. Một ContentResolver có thể nói chuyện với bất cứ content provider nào, chúng cộng tác với provider để quản lý giao tiếp liên tiến trình.

Broadcast Receive

Một Broadcast Rreceiver là một thành phần không làm gì ngoài việc nhận và đáp lại các thông báo broadcast. Nhiều broadcast khởi đầu trong mã hệ thống - ví dụ như thông báo múi giờ thay đổi, pin yếu, ảnh đã được chụp, hay người dùng đã thay đổi ngôn ngữ ... Các ứng dụng có thể tạo ra các broadcast, chẳng hạn để ứng dụng khác biết được một số dữ liệu đã được tải về thiết bị và sẵn sàng cho việc sử dụng chúng.

Một ứng dụng có thể có một số Broadcast Receiver để đáp lại bất cứ thông báo nào mà nó cho là quan trọng. Tất cả các receiver mở rộng từ lớp cơ sở BroadcastReceiver.

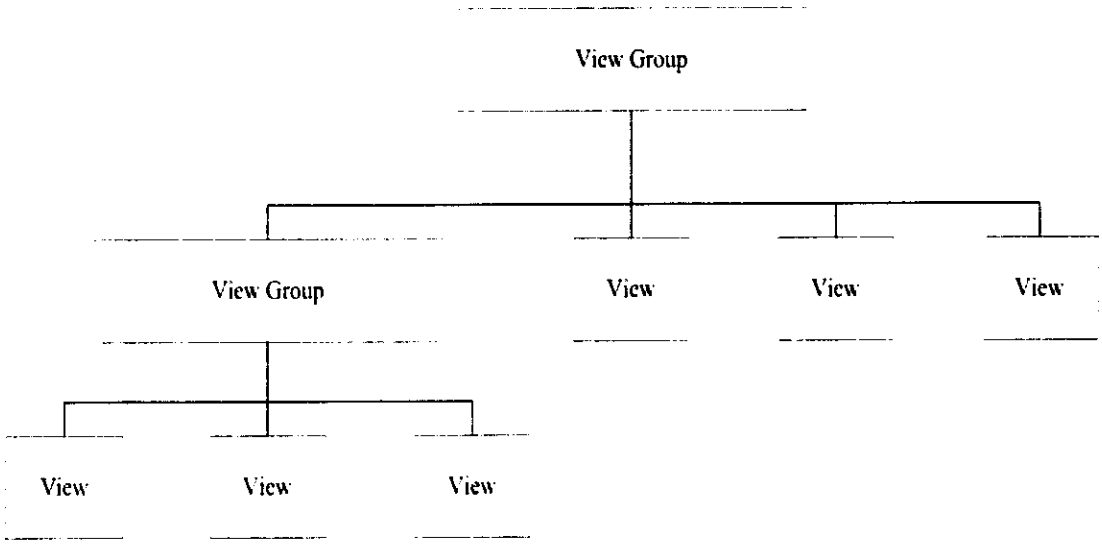
Broadcast Receiver không hiển thị một giao diện người dùng. Tuy nhiên chúng có thể bắt đầu một activity để đáp lại thông tin mà chúng nhận, hay chúng có thể sử dụng NotificationManager để cảnh báo người dùng. Notifications có thể lấy sự chú ý của người dùng bằng nhiều cách, lóe sáng đèn sau, rung, tạo ra âm thanh, vân vân. Chúng thường lấy một biểu tượng bền vững trong thanh trạng thái, cái mà người dùng có thể mở để lấy thông điệp.

View

Trong một ứng dụng Android, giao diện người dùng được xây dựng từ các đối tượng View và ViewGroup. Có nhiều kiểu View và ViewGroup. Mỗi một kiểu là một con của class View và tất cả các kiểu đó được gọi là các Widget.

Tất cả mọi widget đều có chung các thuộc tính cơ bản như là cách trình bày vị trí, background, kích thước, lề ... Tất cả những thuộc tính chung này được thể hiện hết ở trong đối tượng View.

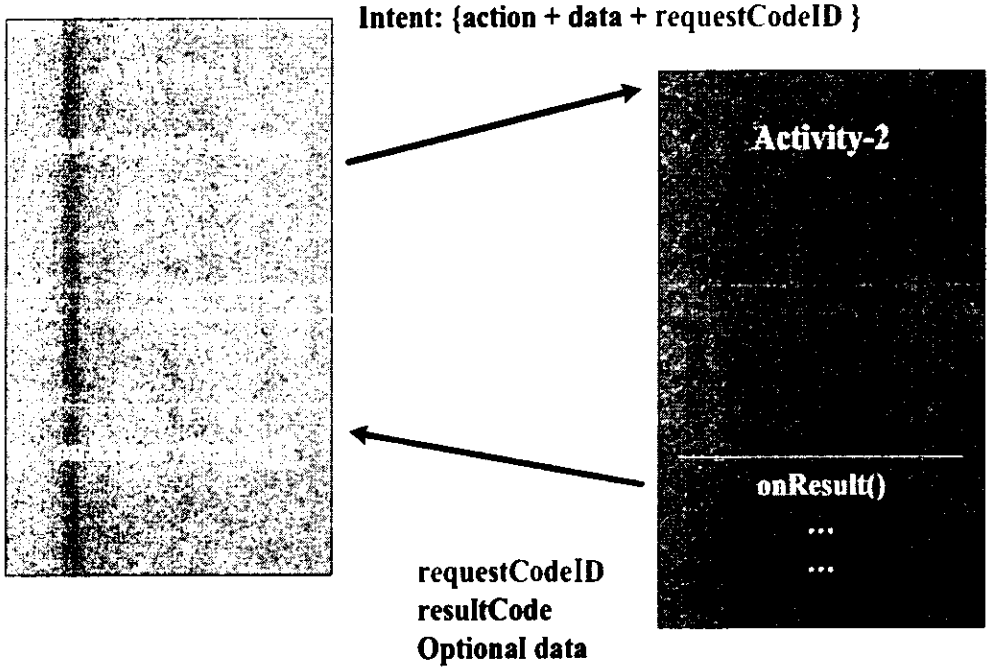
Trong Android Platform, các screen luôn được bố trí theo một kiểu cấu trúc phân cấp như hình dưới. Một màn hình là một tập hợp các Layout và các widget được bố trí có thứ tự. Để thể hiện một màn hình thì trong hàm onCreate của mỗi Activity cần phải được gọi một hàm là setContentView(R.layout.main); hàm này sẽ load giao diện từ file XML lên để phân tích thành mã bytecode.



Hình 2.4. View

Intent

Là cầu nối giữa các Activity: ứng dụng Android thường bao gồm nhiều Activity, mỗi Activity hoạt động độc lập với nhau và thực hiện những công việc khác nhau. Intent chính là người đưa thư, giúp các Activity có thể triệu gọi cũng như truyền các dữ liệu cần thiết tới một Activity khác. Điều này cũng giống như việc di chuyển qua lại giữa các Forms trong lập trình Windows Form.



Hình 2.5. Intent

2.6. Các thành phần trong một project ứng dụng Android

AndroidManifest.xml

Trong bất kì một dự án Android nào khi tạo ra đều có một file AndroidManifest.xml, file này được dùng để định nghĩa các màn hình sử dụng, các quyền cũng như các giao diện cho ứng dụng. Đồng thời nó cũng chứa thông tin về phiên bản SDK cũng như màn hình chính sẽ chạy đầu tiên.

Ví dụ file manifest:

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
  package="com.example.example1"
  android:versionCode="1"
  android:versionName="1.0" >

  <uses-sdk
    android:minSdkVersion="9"
    android:targetSdkVersion="9" />

  <application
    android:allowBackup="true"
    android:icon="@drawable/ic_launcher"
    android:label="@string/app_name"
    android:theme="@style/AppTheme" >
    <activity
      android:name="com.example.example1.MainActivity"
      android:label="@string/app_name" >
      <intent-filter>
        <action android:name="android.intent.action.MAIN" />

        <category android:name="android.intent.category.LAUNCHER" />
      </intent-filter>
    </activity>
  </application>
</manifest>
```

File này được tự động sinh ra khi tạo một dự án Android. Trong file manifest bao giờ cũng có 3 thành phần chính đó là: application, permission và version.

Application: chứa các giá trị định nghĩa cho một ứng dụng Android như icon,tên ứng dụng, chế độ hiển thị giao diện... Ngoài ra cần phải khai báo các Activity và Service có trong chương trình tại đây.

Permission: bao gồm các thuộc tính chỉ định quyền truy xuất và sử dụng tài nguyên của ứng dụng. Khi cần sử dụng một loại tài nguyên nào đó thì trong file manifest của ứng dụng cần phải khai báo các quyền truy xuất tương ứng.

SDK version: Xác định phiên bản SDK nhỏ nhất mà ứng dụng hiện đang sử dụng tương ứng với một phiên bản hệ điều hành Android mà ứng dụng có thể tương thích.

File R.java.

File R.java là một file tự động sinh ra ngay khi tạo ứng dụng, file này được sử dụng để quản lý các thuộc tính được khai báo trong file XML của ứng dụng và các tài nguyên hình ảnh.Mã nguồn của file R.java được tự động sinh khi có bất kì một sự kiện nào xảy xa làm thay đổi các thuộc tính trong ứng dụng.

Có thể nói file R.java hoàn toàn không cần phải đụng chạm gì đến trong cả quá trình xây dựng ứng dụng.

Thư mục src.

Là vị trí chứa gói các class trong ứng dụng. Các class có thể là các một Activity hoặc Service hoặc các lớp chức năng nào đó được viết bằng ngôn ngữ Java dựa trên API được cung cấp sẵn của Android. Cần phải có ít nhất một Activity và khai báo là Activity chính để chương trình có thể chạy được.

Thư mục rec

Thư mục chứa tài nguyên ứng dụng. Thư mục này bao gồm 5 thư mục con là: drawable – hdpi, drawable – mdpi, drawable – ldpi, layout, values.

Drawable – hdpi, drawable – mdpi, drawable – ldpi là ba thư mục dùng để chứa các hình ảnh được sử dụng trong quá trình thiết kế giao diện ứng dụng, bao gồm cả icon của ứng dụng. Ba thư mục tương ứng với hình ảnh sẽ được sử dụng ở ba độ phân giải khác nhau lần lược là: cao, trung bình, thấp. Điều này giúp các nhà lập trình có thể thiết kế giao diện ứng dụng phù hợp với nhiều độ phân giải màn hình tương thích với nhiều loại thiết bị.

Thư mục layout chứa các file xml dùng để khai báo và thiết kế giao diện cho một Activity hay một thành phần điều khiển con trong ứng dụng Android.

Thư mục values gồm các file xml chứa các giá trị chuỗi. mã màu ... Giúp người lập trình có thể dễ dàng thay đổi những giá trị này trong ứng dụng một cách nhanh chóng mà không cần phải sửa trong code của ứng dụng.

CHƯƠNG 3. CÔNG NGHỆ PUSH NOTIFICATION VÀ SERVICE TRONG ANDROID

3.1. Giới thiệu về công nghệ Push Notification

3.1.1. Tổng quan

GCM cho Android là một dịch vụ miễn phí giúp người phát triển phần mềm gửi dữ liệu từ máy chủ đến ứng dụng Android trên thiết bị Android, và thông điệp ngược lại từ thiết bị đến cloud. Đó có thể là thông điệp nhẹ "nói với" ứng dụng là có dữ liệu mới vừa được tải về từ máy chủ, hoặc nó có thể là thông điệp có dung lượng lên tới 4KB. Dịch vụ GCM xử lý tất cả các khía cạnh của hàng đợi tin nhắn và cung cấp đến ứng dụng đích trên một thiết bị đích.

Các đặc điểm chính của GCM:

– Cho phép ứng dụng server của bên thứ 3 gửi thông điệp đến ứng dụng Android của họ;

– Sử dụng GCM Cloud Connection Server, có thể nhận thông điệp ngược lại từ thiết bị của người dùng;

– Ứng dụng Android không cần chạy liên tục để nhận thông điệp. Hệ thống sẽ tự "đánh thức" ứng dụng thông qua Intent broadcast khi thông điệp đến, miễn là ứng dụng được cho phép và thiết lập với broadcast reciver thích hợp;

– GCM không cung cấp bất kỳ một giao diện người dùng hoặc các xử lý khác đối với thông điệp hoặc dữ liệu;

– GCM chỉ đơn giản gửi thẳng dữ liệu thô nhận được đến ứng dụng, ứng dụng sẽ có đầy đủ các khả năng để xử lý nó. Ví dụ, các ứng dụng có thể gửi thông báo, hiển thị một giao diện người dùng, hoặc âm thầm đồng bộ dữ liệu;

– Yêu cầu thiết bị chạy Android 2.2 hoặc cao hơn và đã cài ứng dụng Google Play Store, hoặc một giả lập chạy Android 2.2 với các API của Google. Tuy nhiên, bạn không bị giới hạn việc triển khai ứng dụng của bạn thông qua Google Play Store.

Nó sử dụng một kết nối hiện tại cho các dịch vụ của Google. Cho các thiết bị trước 3.0, điều này đòi hỏi người dùng thiết lập tài khoản Google của họ trên các thiết bị di động của họ. Một tài khoản Google không phải là một yêu cầu trên các thiết bị chạy Android 4.0.4 hoặc cao hơn.

3.1.2. Thuật ngữ và khái niệm liên quan

Bảng sau tóm tắt các thuật ngữ chính và khái niệm liên quan trong GCM. Nó được chia thành các danh mục:

– Components (thành phần) – các đối tượng chính trong GCM;

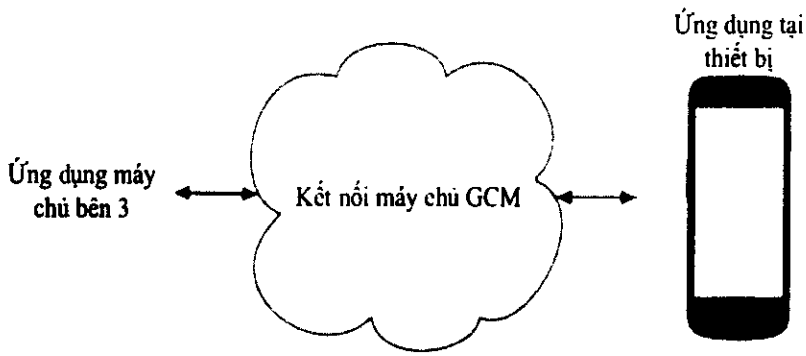
– Credentials - các ID và thẻ được sử dụng trong các giai đoạn khác nhau của GCM để đảm bảo rằng tất cả các bên đã được xác thực, và thông báo đó là đi đến địa điểm chính xác.

Components	
Client App	GCM cho phép ứng dụng Android đang chạy trên một thiết bị. Đây phải là một thiết bị Android 2.2 có Google Play Store được cài đặt, và nó phải có ít nhất một đăng nhập tài khoản Google nếu thiết bị đang chạy mộ t phiên bản thấp hơn so với Android 4.0.4. Ngoài ra, để thử nghiệm, bạn có thể sử dụng một trình giả lập chạy Android 2.2 với các API của Google.
3rd-party Application Server	Một máy chủ ứng dụng mà bạn viết như là một phần của việc thực hiện GCM. Các máy chủ ứng dụng của bên thứ 3 gửi dữ liệu đến một ứng dụng Android trên điện thoại thông qua máy chủ kết nối GCM.
GCM Connection Servers	Google-cung cấp máy chủ có liên quan trong việc thực hiện các tin nhắn từ máy chủ ứng dụng của bên thứ 3 và gửi chúng vào thiết bị.
Credentials	
SenderID	Số của dự án mà bạn có được từ giao diện điều khiển API (khi đăng ký ứng dụng trên Google APIs Console. SenderID được sử dụng trong quá trình đăng ký để xác định một máy chủ ứng dụng bên thứ 3 mà được phép gửi tin nhắn đến thiết bị.
ApplicationID	Ứng dụng Android được đăng ký để nhận tin nhắn. Các ứng dụng Android được xác định bằng tên gói từ mainifest. Điều này đảm bảo rằng các thông điệp được nhắm mục tiêu đến các ứng dụng Android chính xác.
RegistrationID	Một ID do các máy chủ GCM cho các ứng dụng Android cho phép nó nhận tin nhắn. Một khi ứng dụng Android có registrationID, nó

	sẽ gửi nó đến máy chủ ứng dụng của bên thứ 3, GCM sử dụng nó để xác định từng thiết bị đã đăng ký để nhận tin nhắn cho một ứng dụng Android nhất định. Nói cách khác, một registrationID được gắn với một ứng dụng Android đặc biệt chạy trên một thiết bị cụ thể. **Lưu ý**: Nếu sử dụng sao lưu và phục hồi, nên tránh sao lưu ID đăng ký. Khi sao lưu trên một thiết bị, ứng dụng sao lưu chia sẻ Prefs bừa bãi. Nếu bạn không loại trừ một cách rõ ràng GCM registrationID, nó có thể được tái sử dụng trên một thiết bị mới, trong đó sẽ gây ra lỗi chuyển giao thông điệp.
Google User Account	Cho GCM để làm việc, các thiết bị di động phải bao gồm ít nhất một tài khoản Google nếu thiết bị đang chạy một phiên bản thấp hơn so với Android 4.0.4.
Sender Auth Token	Một API key được lưu trên các máy chủ ứng dụng của bên thứ 3 cung cấp cho các máy chủ ứng dụng cho phép truy cập vào các dịch vụ của Google. API key được bao gồm trong tiêu đề của các yêu cầu POST để gửi tin nhắn.

3.1.3. Kiến trúc tổng quan

Việc triển khai GCM bao gồm một máy chủ của Google cung cấp kết nối, máy chủ ứng dụng bên thứ 3 mà tương tác với các máy chủ kết nối, và một ứng dụng client GCM cho phép chạy trên một thiết bị Android:



Hình 3.1. Kiến trúc tổng quan GCM[3]

[3] Theo http://developer.android.com (trang web chính thức của Google Android dành cho các nhà phát triển)

Các thành phần tương tác:

– Google cung cấp máy chủ kết nối GCM nhận tin nhắn từ một máy chủ ứng dụng của bên thứ 3 và gửi các tin nhắn đến một ứng dụng Android GCM cho phép (các "ứng dụng khách hàng") đang chạy trên một thiết bị. Hiện tại, Google cung cấp các máy chủ kết nối cho HTTP và XMPP;

– Các máy chủ ứng dụng của hãng thứ 3 là một thành phần bạn triển khai để làm việc với máy chủ kết nối GCM được lựa chọn. Các máy chủ ứng dụng gửi tin nhắn đến một máy chủ kết nối GCM; các kết nối máy chủ xếp vào hàng đợi và lưu trữ các tin nhắn, và sau đó gửi nó đến thiết bị khi thiết bị đang trực tuyến;

– Ứng dụng khách là một ứng dụng Android GCM cho phép chạy trên một thiết bị. Để nhận tin nhắn GCM, ứng dụng này phải đăng ký với GCM và có được một registrationID. Nếu đang sử dụng (CCS) máy chủ kết nối XMPP, các ứng dụng khách hàng có thể gửi "ngược dòng" tin nhắn lại cho máy chủ kết nối.

3.1.4. Chu trình vòng đời

– Kích hoạt GCM - Một ứng dụng Android chạy trên một thiết bị di động đăng ký để nhận tin nhắn;

– Gửi tin nhắn - Một máy chủ ứng dụng của bên thứ 3 sẽ gửi tin nhắn đến điện thoại;

– Nhận một tin nhắn - Một ứng dụng Android nhận được một tin nhắn từ một máy chủ GCM.

Mô tả chi tiết quá trình:

Kích hoạt GCM:

Lần đầu tiên ứng dụng Android cần sử dụng dịch vụ nhắn tin, nó gọi phương thức GoogleCloudMessaging register(). Phương thức đó trả về một registrationID. Các ứng dụng Android nên lưu trữ ID này để sử dụng về sau (ví dụ, để kiểm tra trong onCreate() nếu nó đã được đăng ký).

Gửi thông điệp:

Đây là trình tự của các sự kiện xảy ra khi một ứng dụng máy chủ gửi một thông điệp:

– Các máy chủ ứng dụng gửi tin nhắn đến các máy chủ GCM;

– Google enqueues và lưu trữ các tin nhắn trong trường hợp thiết bị đang ẩn;

– Khi điện thoại trực tuyến, Google sẽ gửi tin nhắn tới các thiết bị;

– Trên thiết bị, hệ thống các chương trình phát sóng tin nhắn tới các ứng dụng Android được chỉ định thông qua quảng bá Intent với các điều khoản thích hợp, để chỉ những ứng dụng Android nhắm mục tiêu được thông tin. Điều này đánh thức các ứng

dụng Android lên. Các ứng dụng Android không cần phải được chạy trước để nhận được thông báo;

– Các ứng dụng Android xử lý thông báo. Nếu ứng dụng Android đang làm gia công không tầm thường, bạn có thể muốn lấy một *PowerManager.WakeLock* và làm bất cứ xử lý trong một dịch vụ;

– Một ứng dụng Android có thể hủy đăng ký GCM nếu nó không còn muốn nhận tin nhắn.

Nhận thông điệp:

Đây là chuỗi các sự kiện xảy ra khi một ứng dụng Android được cài đặt trên thiết bị di động nhận được tin nhắn:

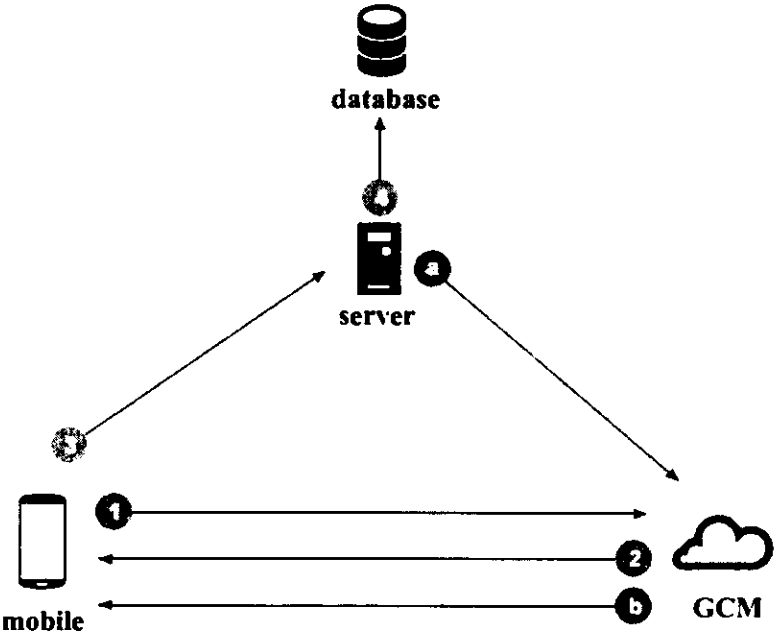
– Hệ thống nhận được các tin nhắn gửi đến và chiết xuất các cặp key / value dữ liệu thô từ tin nhắn nếu cần thiết;

– Hệ thống chuyển các cặp key/value đến ứng dụng Android đích thông qua *Intent com.google.android.c2dm.intent.RECEIVE;*

– Ứng dụng Android chiết xuất dữ liệu thô từ Intent c2dm thông qua các key và xử lý dữ liệu đã được chiết xuất.

3.2. Ứng dụng Push Notification trong ứng dụng Android

3.2.1. Sơ đồ tổng quan các bước trong quá trình gửi nhận tín hiệu



Hình 3.2. Sơ đồ tổng quan các bước trong quá trình gửi nhận tín hiệu[4]

[4] Theo http://developer.android.com (trang web chính thức của Google Android dành cho các nhà phát triển)

– Bước 1: Thiết bị Android gửi Sender ID và Application ID đến máy chủ GCM để đăng ký;

– Bước 2: Thiết bị nhận lại RegistrationID từ GCM server;

– Bước 3: Thiết bị gửi yêu cầu đến máy chủ ứng dụng có RegistrationID kèm theo;

– Bước 4: Máy chủ ứng dụng gửi trả dữ liệu có RegistrationID kèm theo cho máy chủ GCM;

– Bước 5: Thiết bị nhận dữ liệu được gửi về từ máy chủ GCM.

3.2.2. Thực hiện thiết lập Push Notification cho ứng dụng Android

Đăng ký dịch vụ GCM

GCM – Google Cloud Message là một dịch vụ cho phép gửi dữ liệu từ server đến thiết bị Android và cũng có thể nhận được thông điệp từ thiết bị với cùng kiểu kết nối. Để có thể sử dụng dịch vụ này, cần phải đăng ký dịch vụ và thiết lập thông số với Google API.

– Tạo Google API project

+ Truy cập vào trang https://code.google.com/apis/console;

+ Nếu chưa có API project, sẽ có thông báo tạo mới;

+ Google sẽ cung cấp số hiệu project, lưu lại số này vì nó được dùng sau này.

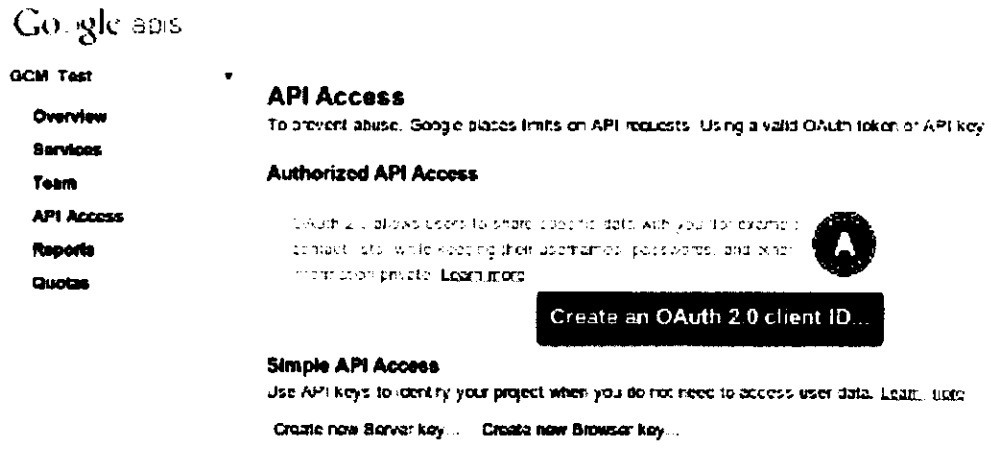
– Bật dịch vụ GCM

+ Ở trang chủ của Google APIs Console, chọn Services;

+ Chuyển Google Cloud Message sang chế độ ON.

– Lấy API key

+ Trong chính trang điều khiển Google API, chọn API truy cập. Bạn sẽ thấy một màn hình tương tự như sau:



+ Nhấp vào Create new Server key và lưu API key nhận được.

Thiết lập trên ứng dụng Android

GoogleCloudMessaging API là một bộ thư viện của Android cho phép cung cấp các chức năng nhận thông điệp từ GCM Server của Google.

Để gửi hoặc nhận tin nhắn, ứng dụng đầu tiên cần phải có được một RegistrationID. RegistrationID xác định các thiết bị và ứng dụng, và cũng có thể xác định máy chủ ứng dụng của bên thứ 3 được phép gửi tin nhắn ứng dụng này.

Để có được một RegistrationID, bạn phải cung cấp một hoặc nhiều SenderID. Một SenderID là một con số dự án mà bạn có được từ giao diện điều khiển API – chính là API key lấy được ở trên. SenderID được sử dụng trong quá trình đăng ký để xác định một máy chủ ứng dụng của bên thứ 3 được phép gửi tin nhắn đến điện thoại. Đoạn code sau đây cho thấy làm thế nào để gọi phương thức register() để lấy được được ResgistrationID:

```
String SENDER_ID = "My-sender-ID";
GoogleCloudMessaging gcm = GoogleCloudMessaging.getInstance(context);
String registrationId = gcm.register(SENDER_ID);
```

Để nhận tin nhắn của GCM, cần phải khai báo một sự cho phép (permission) và một BroadcastReceiver trong file manifest.xml.

Để cho phép các ứng dụng sử dụng GCM, thêm permission vào manifest.xml:

```
<uses-permission android:name="com.google.android.c2dm.permission.RECEIVE" />
```

GCM mang lại tin nhắn như một phát sóng (broadcast). Người nhận phải đăng ký trong tập tin manifest để đánh thức ứng dụng.

```
<receiver android:name=".gcmUtilities.GCMReceiver" android:exported="true"
   android:permission="com.google.android.c2dm.permission.SEND" >
   <intent-filter>
     <action android:name="com.google.android.c2dm.intent.RECEIVE" />
     <category android:name="YOUR_PACKAGE_NAME" />
   </intent-filter>
</receiver>
```

Service trên Android

Như đã trình bày sơ qua ở phía trên về Service trên Android, trong phần này chúng tôi muốn trình bày rõ hơn về Service trên Android – thành phần quan trọng của ứng dụng sẽ xây dựng.

Service là một thành phần của ứng dụng Android dùng để thực thi một phần tác vụ ngầm bên dưới nền và không có giao diện hiển thị nội dung. Service cũng giống như các thành phần khác của ứng dụng (Activity, BroadcastReciver...), nó sẽ chạy trên luồng chính của tiến trình mà ứng dụng đang chạy trên đó. Điều này có nghĩa là nếu ứng dụng cần thực hiện một công việc nào đó tốn nhiều thời gian như chơi nhạc, tải dữ liệu từ trên mạng về thì cần phải đưa công việc đó vào một luồng riêng để thực thi. Việc này sẽ tránh cho các công việc đang thực thi trên luồng chính không bị gián đoạn. Chúng ta cần xác định rõ các đặc trưng của Service:

– Một service không phải một tiến trình tách biệt. Một đối tượng Service không hề chạy trên tiến trình của riêng nó mà nó chạy trên tiến trình của ứng dụng;

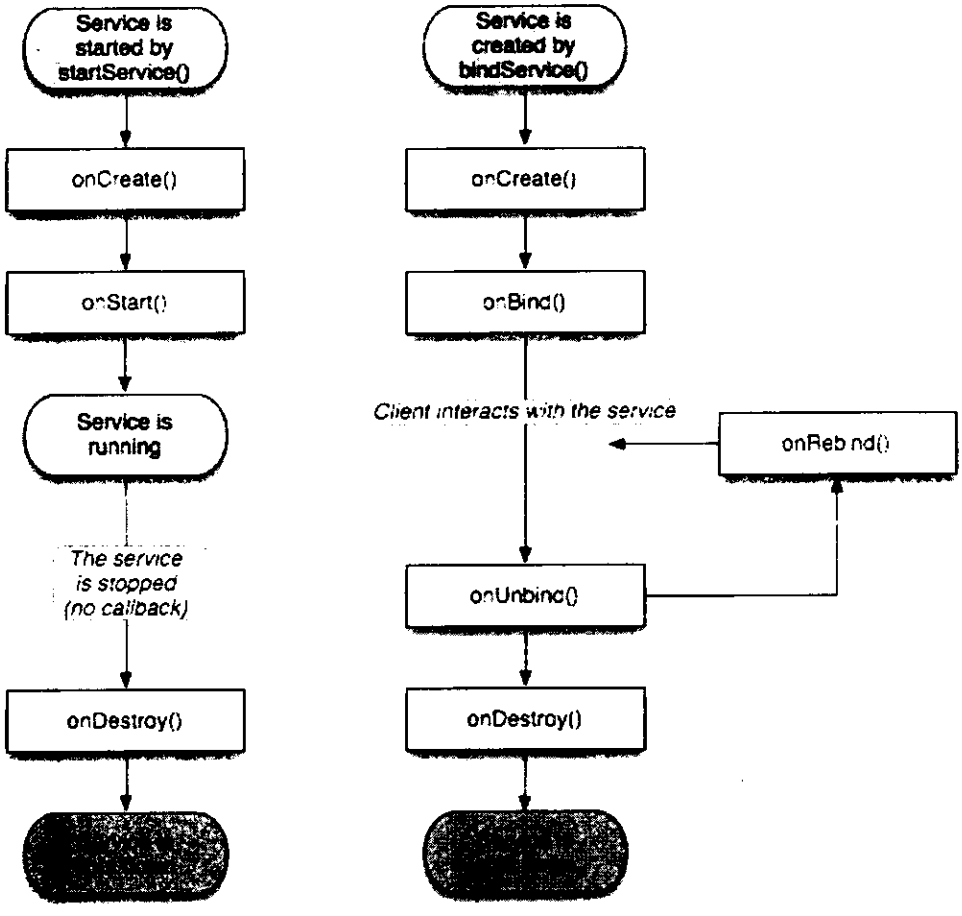
– Một service không phải là một luồng. Điều này có nghĩa là mọi công việc sẽ được luồng chính thực thi. Chính vị vậy một đối tượng Service thường định nghĩa một luồng của riêng nó để thực hiện các công việc nhằm tránh tình trạng gián đoạn các công việc đang thực thi ở luồng chính.

Một Service có thể được sử dụng theo hai cách:

– Một Service có thể được dùng để thực thi một công việc dưới nền mà không cần hiển thị giao diện người dùng. Loại service này được bắt đầu và được cho phép hoạt động cho đến khi một người nào đó dừng nó lại hoặc nó tự ngắt. Ở chế độ này, nó được bắt đầu bằng cách gọi *Context.startService()* và dừng bằng lệnh *Contect.stopService()*. Một Service có thể tự ngắt bằng lệnh *Service.stopSelf()* hoặc *Service.stopResulf()*. Mỗi Service chỉ có một thể hiện duy nhất, do đó chỉ cần một lệnh *stopService()* để ngừng một Service lại cho dù lệnh *startService()* có được gọi ra bao nhiêu lần;

– Một Service còn có thể được sử dụng để cung cấp một tính năng nào đó cho ứng dụng khác kết nối và sử dụng. Một ứng dụng có thể thiết lập một đường truyền tới đối tượng Service và sử dụng đường kết nối đó để điều khiển Service. Kết nối này được thiết lập bằng cách gọi lệnh *Context.bindService()* và được đóng bằng cách gọi lệnh *Context.unbindService()*. Nhiều ứng dụng có thể kết nối tới cùng một đối tượng Service. Nếu Service được một ứng dụng khách kết nối đến vẫn chưa được khởi chạy thì lệnh *bindService()* có thể tùy ý khởi chạy nó.

Hai chế độ này thì không tách biệt toàn bộ. Bạn có thể kết nối với một Service mà nó đã được khởi chạy với lệnh *startService()*.



Hình 3.3. Vòng đời của một Service

Dựa vào lược đồ trên ta có hai vòng lặp quan trọng trong vòng đời của một đối tượng Service cục bộ:

– Vòng đời toàn diện (entire lifetime): bắt đầu từ lúc gọi phương thức *onCreate()* đến lúc gọi phương thức *onDestroy()*. Cũng giống như Activity, đối tượng Service sẽ khởi tạo các giá trị tại phương thức *onCreate()* và dọn dẹp bộ nhớ tại phương thức *onDestroy()*;

– Vòng đời thực thi (activce lifetime): bắt đầu từ lúc gọi phương thức *onStart()*.

Nếu một đối tượng Service cho phép một ứng dụng khác kết nối đến nó thì nó phải cài đặt 03 phương thức sau:

– *Ibinder onBind (Intent intent)*: Khi một đối tượng muốn tạo kết nối đến một đối tượng Service thì nó sẽ gọi phương thức *Context bindService(...)* và gửi đi một đối tượng Intent. Phương thức *onBind()* sẽ được gọi để sử lý yêu cầu kết nối này. Nó sẽ trả về các kênh giao tiếp mà đối tượng cần kết nối có thể sử dụng để tương tác với Service;

– *Boolean onUnbind (Intent intent)*: Phương thức này tương tự như phương thức onBind(). Tuy nhiên nó sẽ được gọi khi có một đối tượng gọi phương thức Context.unBindService() để ngắt

kết nối với Service. Lúc này phương thức onUnbind() sẽ được gọi để xử lý yêu cầu ngắt kết nối đến Service;

– *Void onRebind (Intent intent)*: Phương thức này được gọi khi có một đối tượng khách mới muốn kết nối đến Service.

Để khởi chạy service, có hai chế độ có thể thêm vào quá trình được quyết định khi chạy, tùy thuộc vào giá trị trả về từ hàm *onStartCommand()*: START_STICKY được sử dụng cho các dịch vụ được bắt đầu một cách rõ ràng và dừng lại khi cần thiết, trong khi START_NOT_STICKY hoặc START_REDELIVER_INTENT được sử dụng cho các dịch vụ mà chỉ nên duy trì hoạt động trong khi xử lý các lệnh gửi cho service.

Hằng số được trả về từ *onStartCommand (Intent, int, int)*: nếu quá trình của service bị dừng lại (killed) trong khi nó được bắt đầu (sau khi trả về từ hàm onStartCommand (Intent, int, int), sau đó đưa service về trạng thái đã bắt đầu nhưng không giữ lại những intent đã cung cấp. Sau đó hệ thống sẽ cố gắng để "tái tạo" dịch vụ.

CHƯƠNG 4. XÂY DỰNG ỨNG DỤNG NHẮC LỊCH THI CHO SINH VIÊN THĂNG LONG

4.1. Tổng quan yêu cầu hệ thống

4.1.1. Mô tả

Hiện nay, sinh viên Thăng Long cập nhật lịch thi của mình trên trang web của nhà trường. Mỗi khi có lịch thi, trang chủ website của nhà trường đăng thông báo cho sinh viên toàn trường được biết đã có lịch thi mới. Thế nhưng hiện nay, hệ thống đăng ký học online của nhà trường thường xuyên bị quá tải do lượng sinh viên truy cập quá lớn, cùng với việc thông báo trên website bắt buộc sinh viên phải theo dõi và cập nhật liên tục thông tin qua mạng đã gây nên những khó khăn cho nhiều sinh viên như không thể truy cập internet, máy chủ nhà trường không hoạt động, lịch thi có những thay đổi mà sinh viên không chú ý sẽ không cập nhật kịp thời. Sinh viên phải có nhiều biện pháp để lưu trữ lịch thi như chép tay, lưu vào máy tính, điện thoại dưới dạng tệp ảnh, tệp văn bản.

Do vậy, nhóm tác giả thực hiện đề tài mong muốn phát triển một ứng dụng di động nhằm mục đích quản lý và cập nhật lịch thi của sinh viên sao cho đạt hiệu quả cao nhất, hạn chế rủi ro xảy ra trong khi kì thi diễn ra. Nhắc nhở cho sinh viên biết lịch thi của các môn thi sắp tới giúp sinh viên tránh được tình trạng "quên đi thi".

4.1.2. Hiện trạng tại trường

Hiện tại thì việc xem và cập nhật lịch thi của sinh viên Đại học Thăng Long diễn ra một cách thủ công, vẫn thường xuyên xảy ra các sự cố khiến cho sinh viên không thể theo dõi lịch thi của mình.

4.1.3. Yêu cầu nghiệp vụ

BR1: Cập nhật lịch thi

Lịch thi sẽ được cập nhật đảm bảo lịch thi được trả về là lịch thi học kỳ gần nhất của sinh viên. Bất kỳ một sự thay đổi nào về lịch thi sẽ được thông báo cho sinh viên.

BR2: Hiển thị lịch thi

Hệ thống giúp sinh viên có thể theo dõi lịch thi của mình một cách chi tiết nhất, các môn thi được sắp xếp theo thứ tự thời gian giúp cho sinh viên thu xếp được thời gian học tập.

BR3: Lịch thi được lưu trữ trên CSDL

Khi lịch thi mới nhất được hệ thống cập nhật, lịch thi đó sẽ được lưu trữ lại vào cơ sở dữ liệu của thiết bị. Giúp cho quá trình theo dõi lịch thi của sinh viên không bị gián đoạn nếu hệ thống của nhà trường xảy ra sự cố.

BR4: Nhắc lịch thi

Khi có một môn thi sắp bắt đầu, hệ thống sẽ hiển thị thông báo để sinh viên biết được môn thi gần nhất của mình là gì.

BR5: Tuỳ chọn thời gian cập nhật, nhắc lịch thi

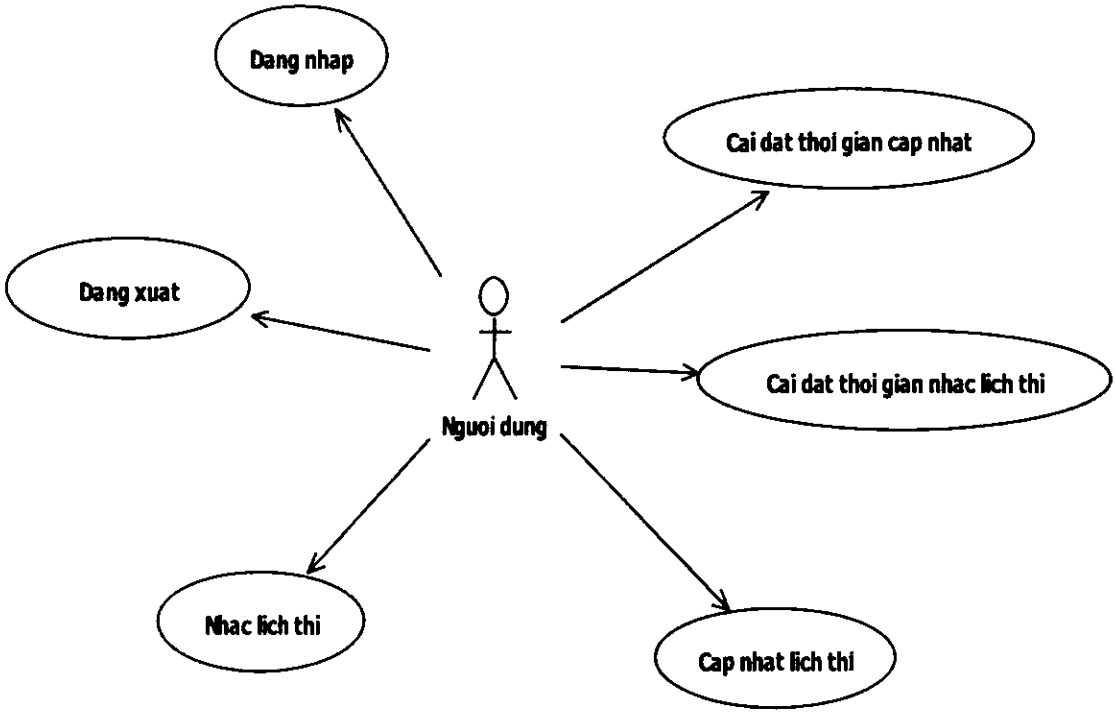
Hệ thống cho phép người dùng có thể tuỳ chọn thời gian cập nhật lịch thi hoặc thời gian nhắc lịch thi.

4.2. Ứng dụng Android

4.2.1. Mô tả

Đây là ứng dụng phía người dùng (client) được phát triển nhằm thông báo, hiển thị, lưu trữ dữ liệu về lịch thi, đồng thời nhắc nhở lịch thi cho sinh viên.

4.2.2. Sơ đồ tổng quan các chức năng chính của ứng dụng



4.2.3. Các tác nhân tham gia

Người dùng: là người sử dụng ứng dụng trên thiết bị Android, người dùng có thể cập nhật lịch thi hoặc xem lịch thi sau khi đăng nhập vào hệ thống.

4.2.4. Các chức năng chính của ứng dụng

Đăng nhập: người dùng cần phải đăng nhập vào ứng dụng sử dụng mã sinh viên được cấp bởi trường Thăng Long. Ứng với mỗi mã sinh viên sẽ có được lịch thi tương ứng của sinh viên đó.

Đăng xuất: người dùng khi không muốn nhận thông báo hoặc cập nhật lịch thi có thể sử dụng chức năng đăng xuất để hủy đăng ký mã sinh viên với máy chủ hệ thống.

Cập nhật lịch thi: chức năng này được sử dụng khi người dùng cài đặt chế độ cập nhật bằng tay, người dùng sẽ chủ động kiểm tra cập nhật bằng việc thao tác trực tiếp với ứng dụng. Nếu người dùng chọn chế độ cập nhật tự động, chức năng này sẽ ẩn đi và được chạy tự động mà không cần thao tác trực tiếp với ứng dụng.

Cài đặt thời gian cập nhật: chức năng này cho phép người dùng tùy chỉnh về khoảng thời gian ứng dụng kiểm tra cập nhật lịch thi với hệ thống máy chủ.

Cài đặt thời gian nhắc lịch thi: chức năng này cho phép người dùng tùy chỉnh về khoảng thời gian nhắc nhở trước ngày thi đối với mỗi môn thi. Đến đúng thời gian được cài đặt, nếu có môn thi sắp diễn ra, ứng dụng sẽ thông báo tới người dùng.

4.2.5. Các thực thể chính

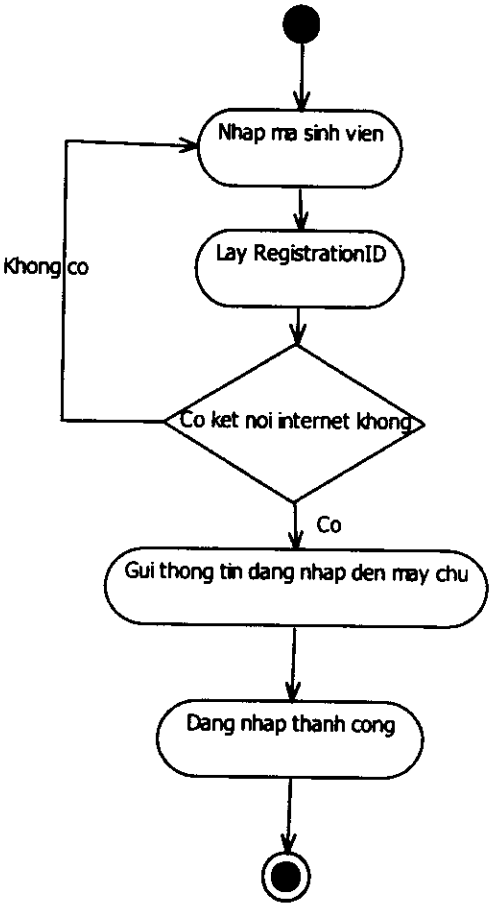
Thực thể môn thi: Mỗi môn thi đều có các thuộc tính: mã môn, tên môn, ngày thi, ca thi, phòng thi, tình trạng.

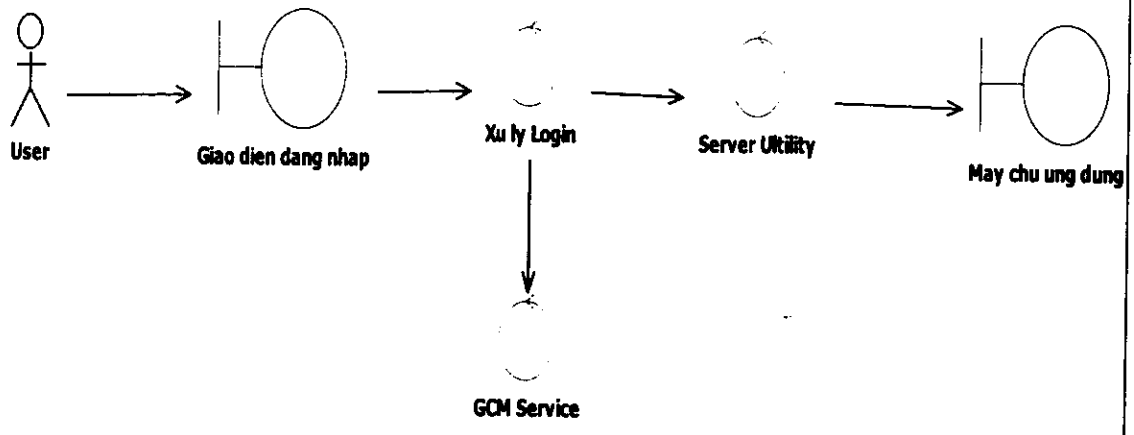
Thực thể lịch thi: Lịch thi gồm nhiều môn thi được sắp xếp tăng dần theo ngày thi.

Thực thể lịch nhắc nhở: Danh sách các thông báo nhắc nhở về môn thi.

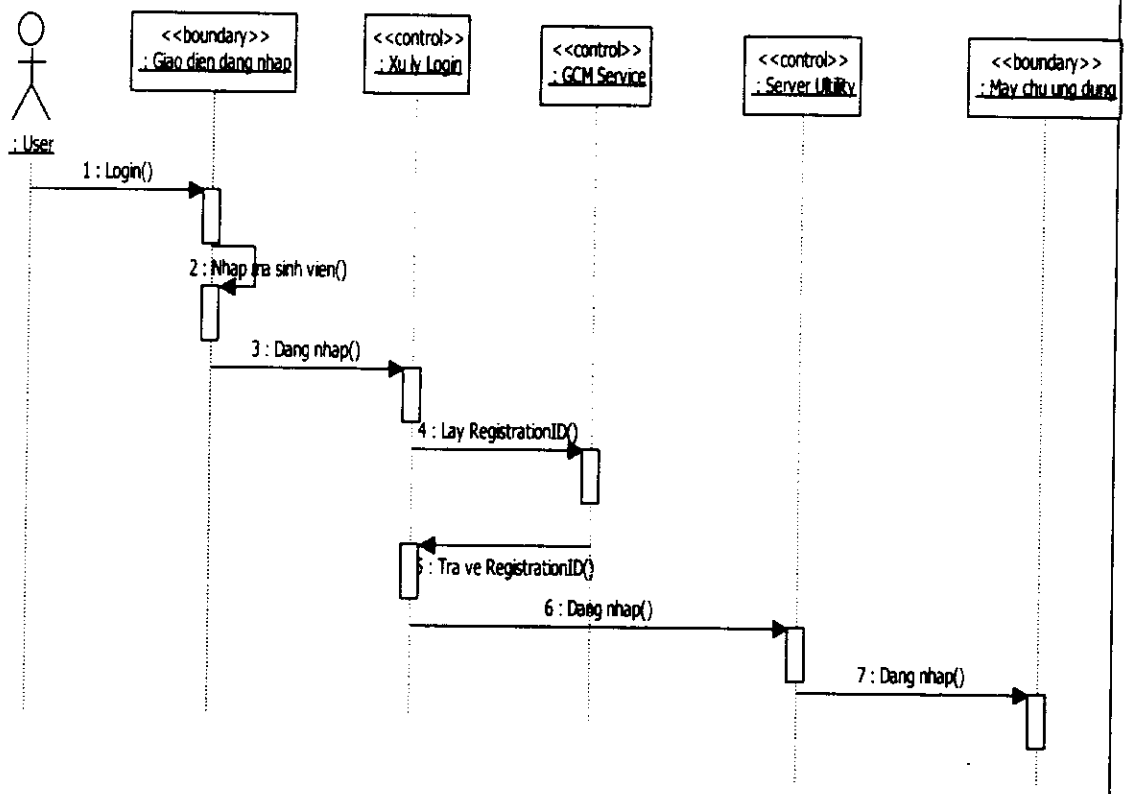
4.2.6. Đặc tả các chức năng của ứng dụng

<table>
<tr><td colspan="2">UC #1</td><td>ĐĂNG NHẬP</td><td>Độ phức tạp: Low</td></tr>
<tr><td colspan="2">Mô tả</td><td colspan="2">Chức năng này cho phép người dùng đăng nhập vào ứng dụng, ứng dụng sẽ đăng ký mã sinh viên của người dùng với hệ thống máy chủ.</td></tr>
<tr><td rowspan="2">Tác nhân</td><td>Chính</td><td colspan="2">Người dùng.</td></tr>
<tr><td>Phụ</td><td colspan="2">Không.</td></tr>
<tr><td colspan="2">Tiền điều kiện</td><td colspan="2">Ứng dụng khởi chạy không có lỗi, thiết bị có kết nối internet, chưa có mã sinh viên nào trước đó đang đăng nhập.</td></tr>
<tr><td rowspan="2">Hậu điều kiện</td><td>Thành công</td><td colspan="2">Người dùng đăng nhập được vào ứng dụng, mã sinh viên và RegistrationID được đăng ký với máy chủ.</td></tr>
<tr><td>Lỗi</td><td colspan="2">Người dùng không đăng nhập được vào ứng dụng, trạng thái ứng dụng không thay đổi.</td></tr>
</table>

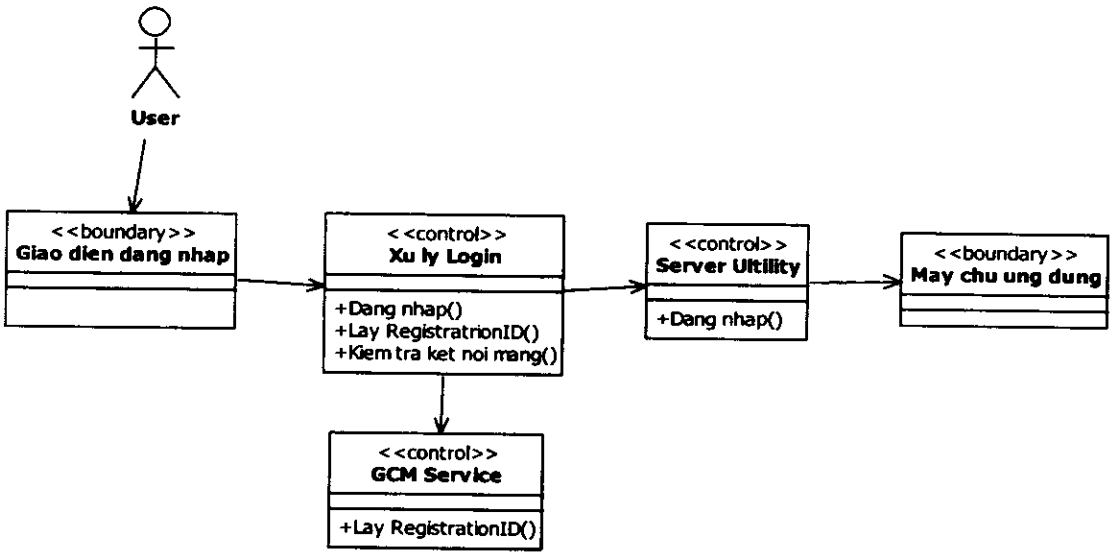
ĐẶC TẢ CHỨC NĂNG
Luồng sự kiện chính / kịch bản chính
Chức năng này bắt đầu khi người dùng mở ứng dụng và chưa có tài khoản nào đang truy cập: 1. Hệ thống yêu cầu người dùng nhập vào mã sinh viên; 2. Người dùng nhập mã sinh viên; 3. Ứng dụng lấy RegistrationID và mã sinh viên người dùng vừa nhập gửi lên hệ thống máy chủ; 4. Ứng dụng chuyển vào giao diện chính.
Luồng sự kiện phát sinh / kịch bản phát sinh
Nếu thiết bị không được kết nối internet, ứng dụng sẽ thông báo lỗi. Người dùng có thể đóng ứng dụng hoặc bật kết nối internet để tiếp tục.
Sơ đồ hành động (Activity diagram)


Các yêu cầu đặc biệt khác
Không có.
Tình trạng trước khi thực hiện use-case
Ứng dụng ở chế độ chờ đăng nhập. Người dùng không sử dụng được chức năng nào khác.
Tình trạng sau khi thực hiện use-case
- Nếu đăng nhập thành công, giao diện chính của ứng dụng được kích hoạt. Thông tin đăng ký với máy chủ được ghi nhận và lưu trữ. - Nếu đăng nhập thất bại, ứng dụng thông báo và không có gì thay đổi.
THIẾT KẾ UML
Sơ đồ lớp phân tích (Analysis class diagram)


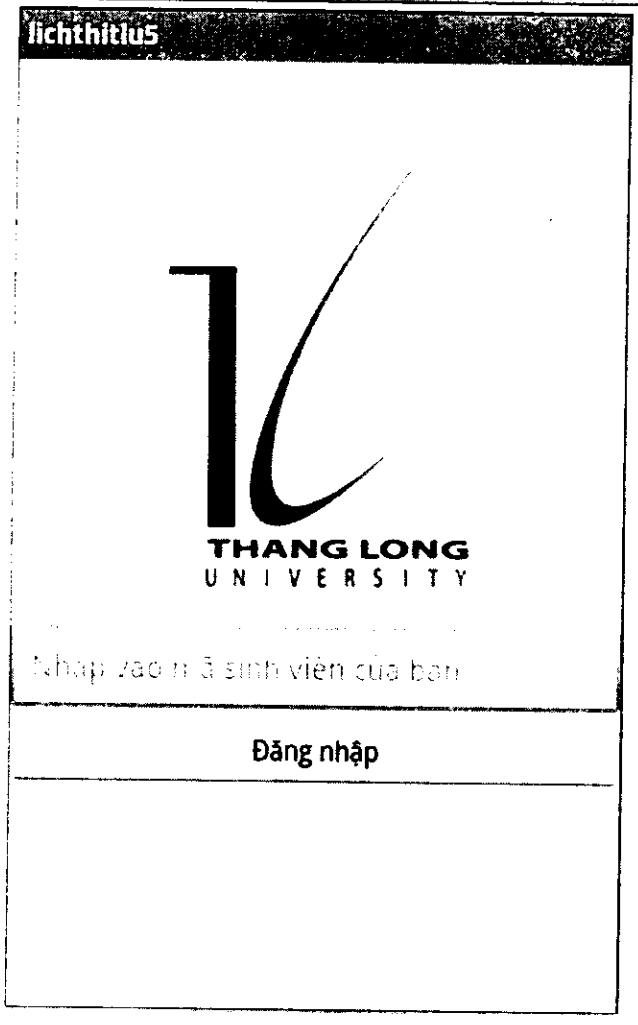
Sơ đồ trình tự (Sequence diagram)



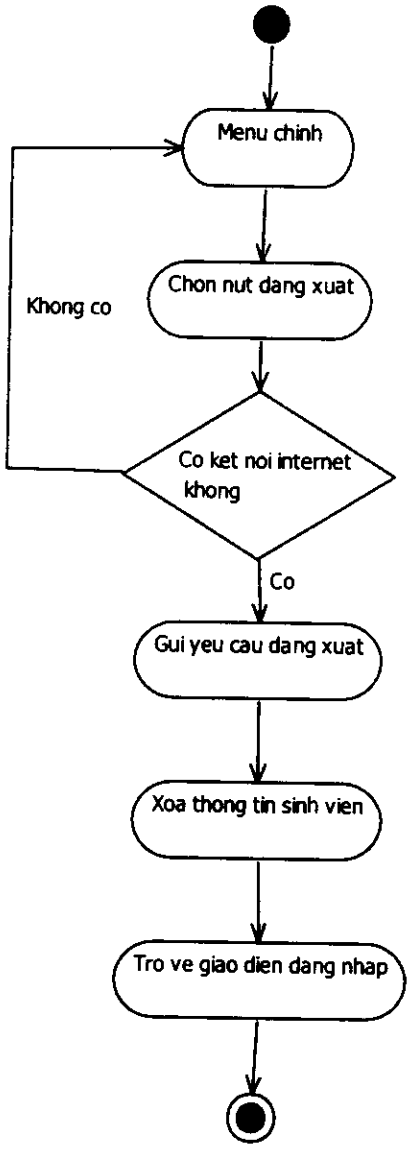
Sơ đồ lớp chi tiết (Class diagram)



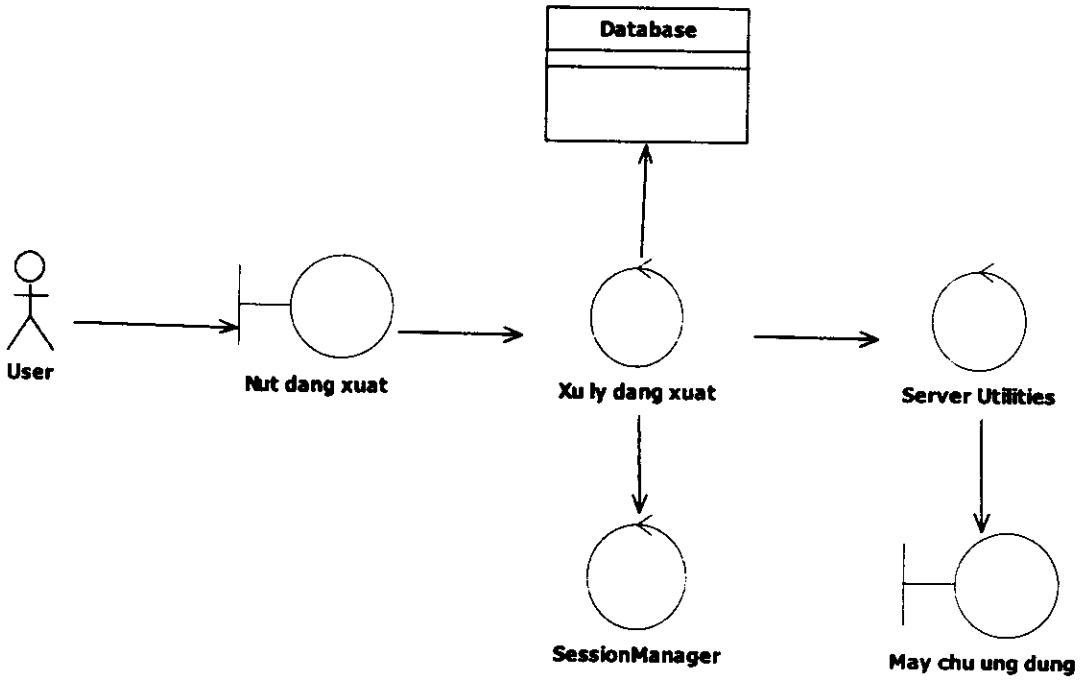
Hình ảnh minh họa



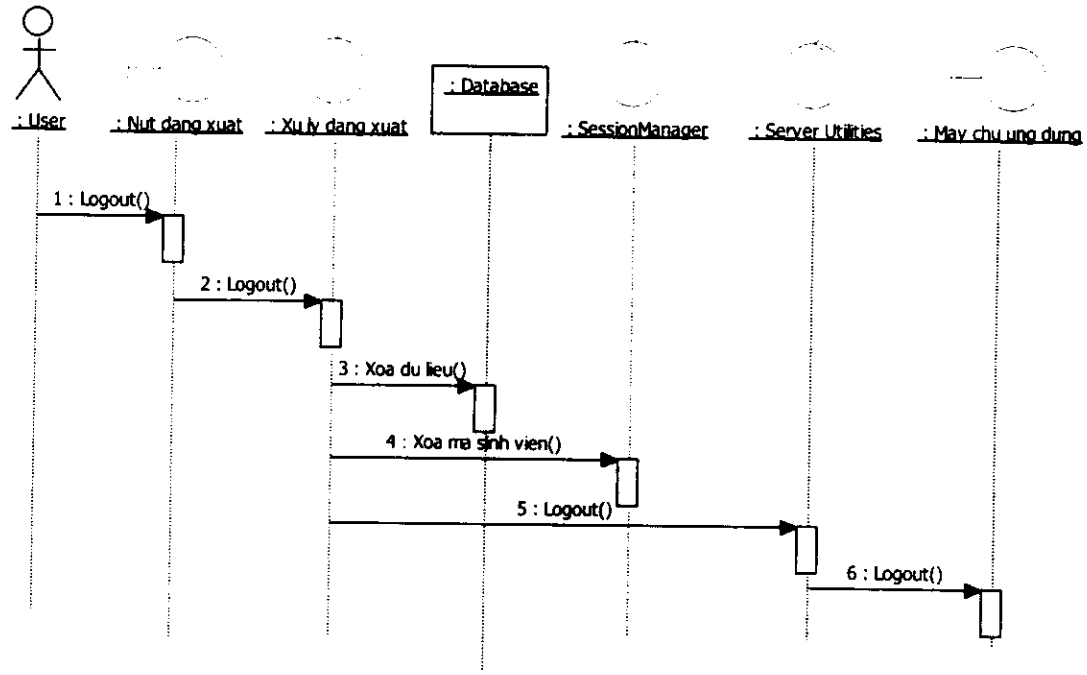
UC #2		ĐĂNG XUẤT	Độ phức tạp: Low
Mô tả		Chức năng cho phép người dùng đăng xuất ra khỏi ứng dụng.	
Tác nhân	Chính	Người dùng.	
	Phụ	Không có.	
Tiền điều kiện		Ứng dụng khởi chạy không có lỗi, đang có tài khoản đăng nhập ứng dụng.	
Hậu điều kiện	Thành công	Ứng dụng trở về màn hình đăng nhập, các thông tin được lưu trữ trên thiết bị của người dùng trước bị xóa bỏ, mã sinh viên trước được xóa khỏi hệ thống máy chủ.	
	Lỗi	Ứng dụng không thay đổi trạng thái.	
ĐẶC TẢ CHỨC NĂNG			
Luồng sự kiện chính / kịch bản chính			
Chức năng này bắt đầu khi người dùng chọn chức năng "đăng xuất": 1. Ứng dụng xóa các thông tin lưu trữ của người dùng; 2. Ứng dụng gửi thông báo đăng xuất đến máy chủ; 3. Ứng dụng trở về chức năng đăng nhập, mã sinh viên được xóa bỏ trên máy chủ.			

Sơ đồ hành động (Activity diagram)

Luồng sự kiện phát sinh / kịch bản phát sinh
Không có.
Các yêu cầu đặc biệt khác
Không có.
Tình trạng trước khi thực hiện use-case
Người dùng đang đăng nhập vào ứng dụng.

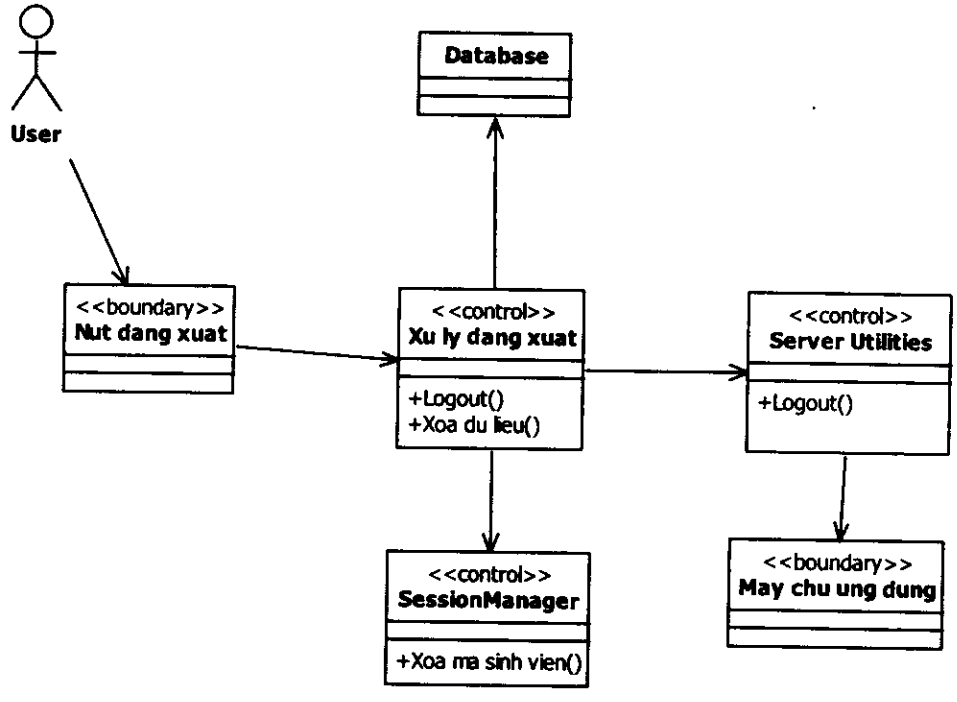
Tình trạng sau khi thực hiện use-case
1. Nếu use-case thực hiện thành công, ứng dụng sẽ trở về màn hình đăng nhập. Mã sinh viên được xoá bỏ khỏi máy chủ. 2. Use-case thực hiện không thành công, trạng thái ứng dụng không thay đổi.
Điểm mở rộng
Không có
Thiết kế UML
Sơ đồ lớp phân tích (Analysis class diagram)



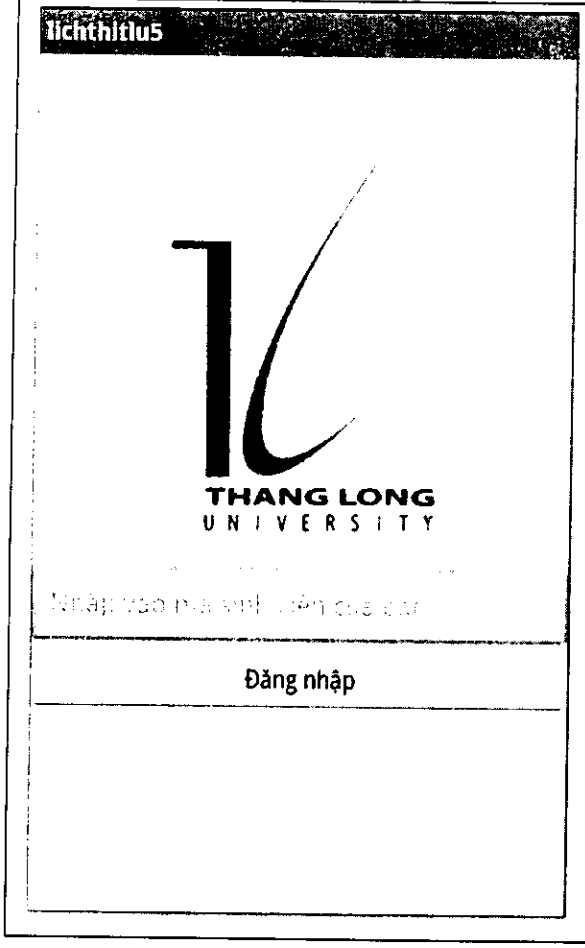
Sơ đồ trình tự (Sequence diagram)

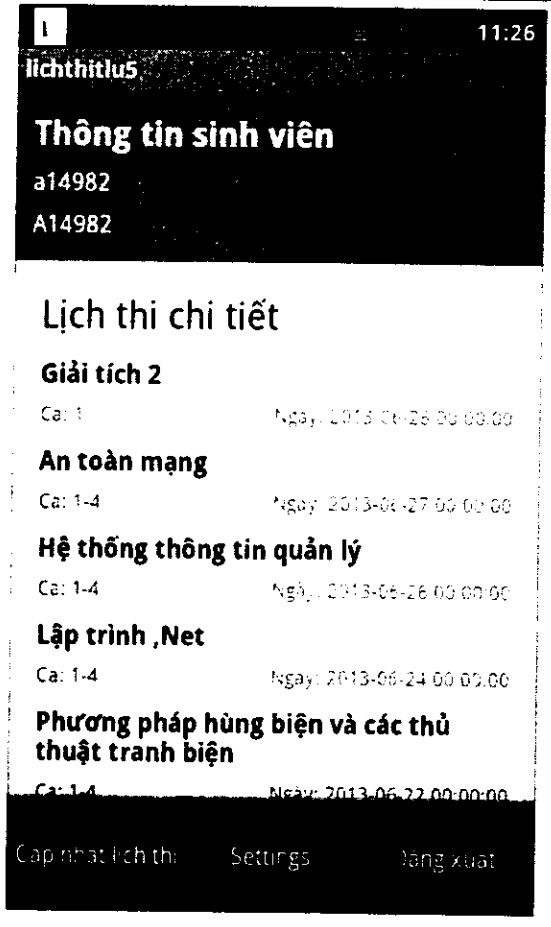


Sơ đồ lớp chi tiết (Class diagram)

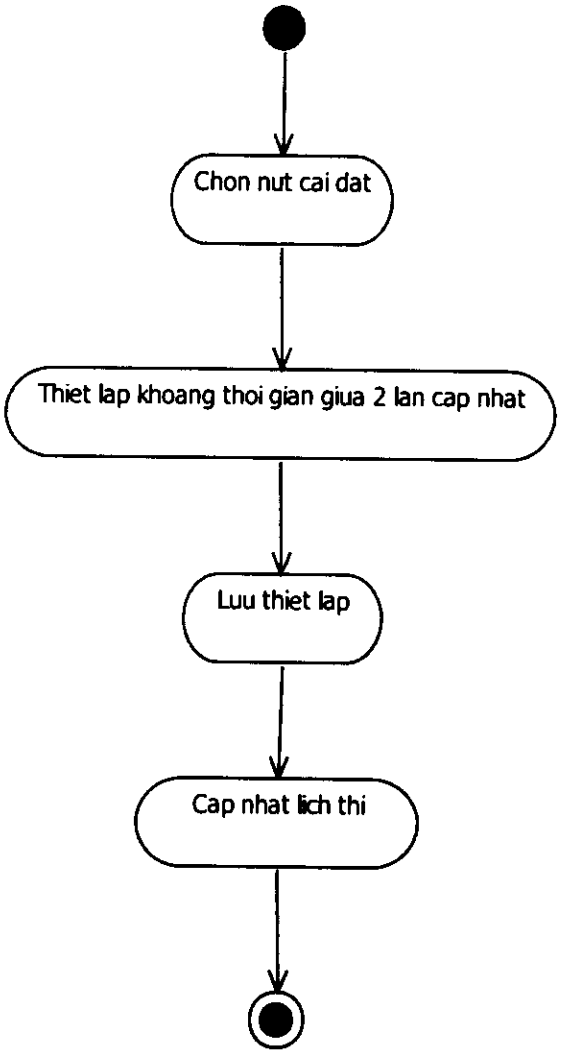


Hình ảnh minh họa

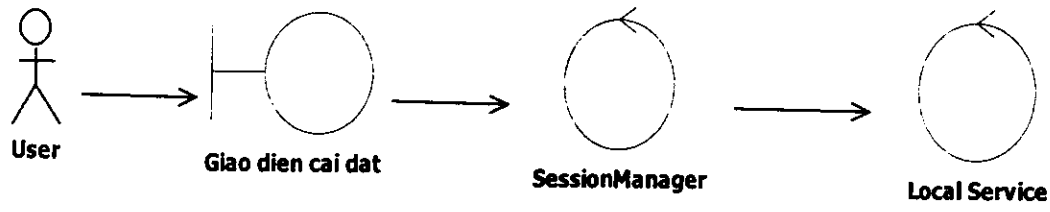




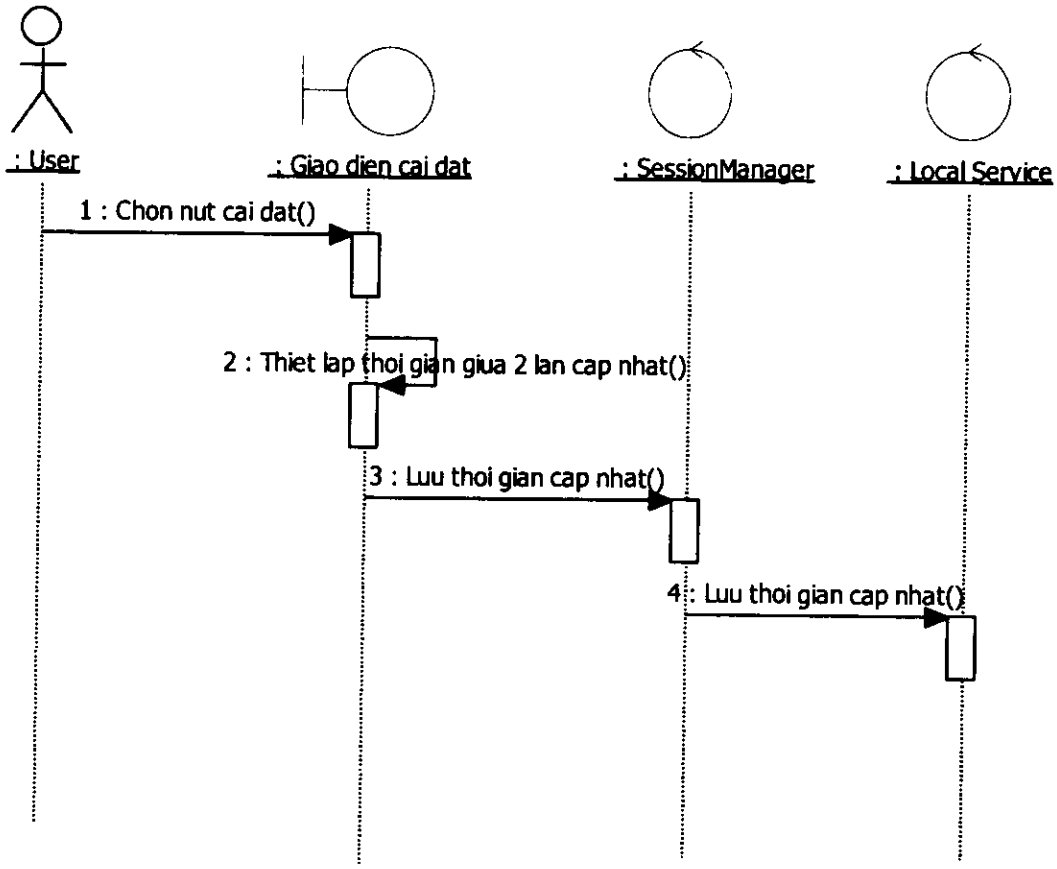
<table>
<tr><td colspan="2">UC #3</td><td>CÀI ĐẶT THỜI GIAN CẬP NHẬT</td><td>Độ phức tạp: Medium</td></tr>
<tr><td colspan="2">Mô tả</td><td colspan="2">Chức năng cho phép người dùng thay đổi khoảng thời gian giữa hai lần cập nhật hoặc chuyển sang chế độ cập nhật thủ công.</td></tr>
<tr><td rowspan="2">Tác nhân</td><td>Chính</td><td colspan="2">Người dùng.</td></tr>
<tr><td>Phụ</td><td colspan="2">Không có.</td></tr>
<tr><td colspan="2">Tiền điều kiện</td><td colspan="2">ứng dụng khởi chạy không có lỗi, đang có tài khoản đăng nhập ứng dụng.</td></tr>
<tr><td rowspan="2">Hậu điều kiện</td><td>Thành công</td><td colspan="2">Thời gian giữa hai lần cập nhật được thay đổi, nếu chế độ cập nhật thủ công được thiết lập, ứng dụng sẽ chỉ cập nhật khi nào người dùng sử dụng chức năng cập nhật lịch thi.</td></tr>
<tr><td>Lỗi</td><td colspan="2">Ứng dụng không có thay đổi.</td></tr>
<tr><td colspan="4">ĐẶC TẢ CHỨC NĂNG</td></tr>
<tr><td colspan="4">Luồng sự kiện chính / kịch bản chính</td></tr>
<tr><td colspan="4">Chức năng bắt đầu khi người dùng chọn chức năng "Cài đặt":
1. Giao diện cài đặt hiển thị lên cho người dùng biết họ đang để thời gian cập nhật là bao nhiêu;
2. Thời gian cập nhật có thể thay đổi giữa các mốc:
• 5 phút;
• 10 phút;
• 15 phút;
• 30 phút;
• Hàng giờ;
• Thủ công.
3. Khi người dùng chọn một mốc thời gian, ứng dụng sẽ lưu lại và tự động cập nhật lịch thi theo mốc thời gian đã được chọn;
4. Nếu người dùng chọn "Thủ công", lịch thi sẽ chỉ được cập nhật khi người dùng chọn chức năng "Cập nhật lịch thi".</td></tr>
</table>

Sơ đồ hành động (Activity diagram)

Luồng sự kiện phát sinh / kịch bản phát sinh
Không có.
Các yêu cầu đặc biệt khác
Không có.
Tình trạng trước khi thực hiện use-case
Ứng dụng khởi chạy không có lỗi, đang có tài khoản đăng nhập ứng dụng.
Tình trạng sau khi thực hiện use-case
1. Sau khi use-case được thực hiện thành công, thiết lập của người dùng được ứng dụng lưu lại. 2. Nếu use-case thất bại thì trạng thái ứng dụng trước đó không thay đổi.

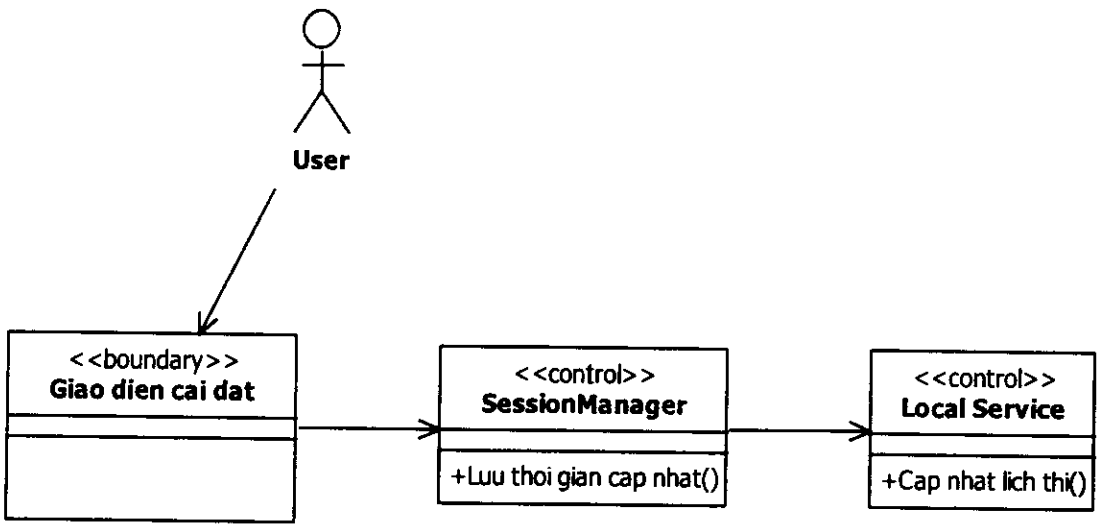
Điểm mở rộng
Không có
Thiết kế UML
Sơ đồ lớp phân tích (Analysis class diagram)



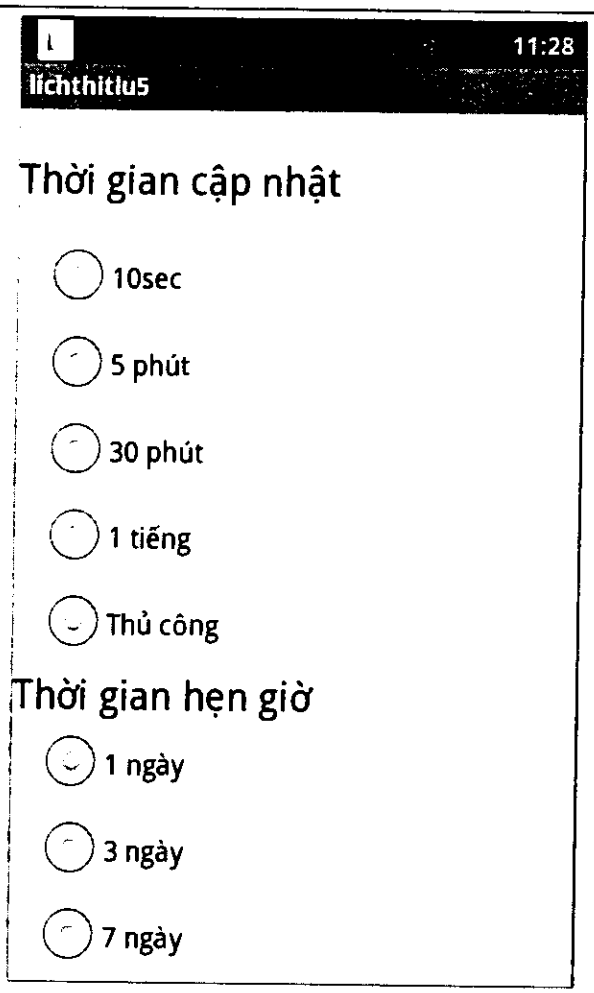
Sơ đồ trình tự (Sequence diagram)



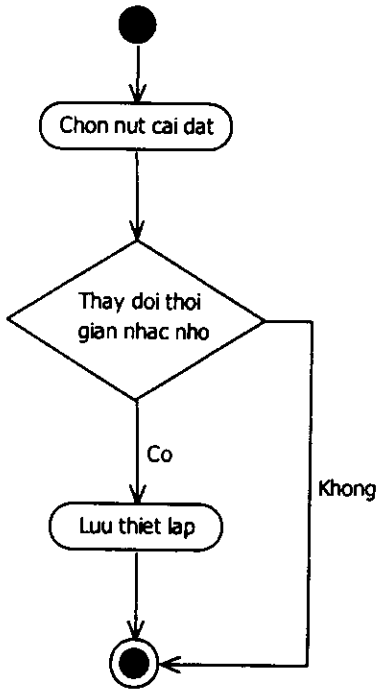
Sơ đồ lớp chi tiết (Class diagram)



Hình ảnh minh họa

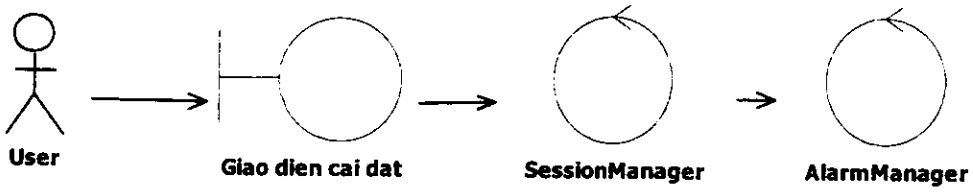


<table>
<tr><td colspan="2">UC #4</td><td>CÀI ĐẶT THỜI GIAN NHẮC NHỞ</td><td>Độ phức tạp: Medium</td></tr>
<tr><td colspan="2">Mô tả</td><td colspan="2">Chức năng cho phép người dùng thay đổi khoảng thời gian báo trước thời điểm diễn ra các môn thi trong lịch thi.</td></tr>
<tr><td rowspan="2">Tác nhân</td><td>Chính</td><td colspan="2">Người dùng.</td></tr>
<tr><td>Phụ</td><td colspan="2">Không có.</td></tr>
<tr><td colspan="2">Tiền điều kiện</td><td colspan="2">Ứng dụng khởi chạy không có lỗi, đang có tài khoản đăng nhập ứng dụng.</td></tr>
<tr><td rowspan="2">Hậu điều kiện</td><td>Thành công</td><td colspan="2">Thời gian báo trước ngày bắt đầu một môn thi được thay đổi.</td></tr>
<tr><td>Lỗi</td><td colspan="2">Ứng dụng không có thay đổi.</td></tr>
<tr><td colspan="4">ĐẶC TẢ CHỨC NĂNG</td></tr>
<tr><td colspan="4">Luồng sự kiện chính / kịch bản chính</td></tr>
<tr><td colspan="4">Chức năng bắt đầu khi người dùng chọn chức năng "Cài đặt":
1. Giao diện cài đặt hiển thị lên cho người dùng biết họ đang để thời gian nhắc nhở là bao nhiêu;
2. Thời gian nhắc nhở có thể thay đổi giữa các mốc:
• 1 ngày;
• 3 ngày;
• 7 ngày;
• 10 ngày;
• Không báo.
3. Khi người dùng chọn một mốc thời gian, ứng dụng sẽ lưu lại và tự động nhắc nhở theo mốc thời gian đã được chọn;
4. Nếu người dùng chọn "Không báo", ứng dụng sẽ không nhắc nhở cho người dùng biết khi có môn sắp đến ngày thi.</td></tr>
</table>

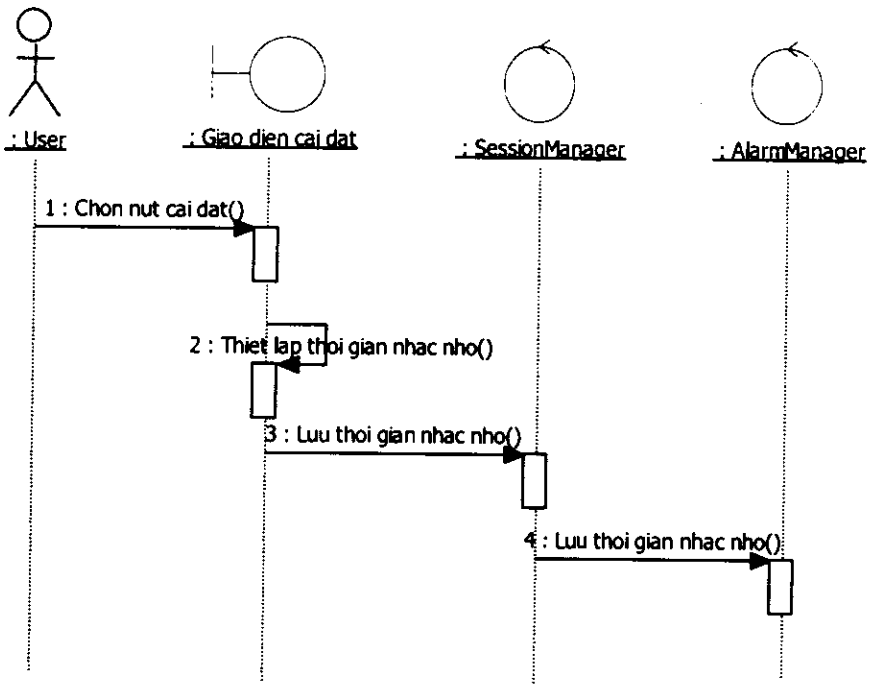
Sơ đồ hành động (Activity diagram)

Luồng sự kiện phát sinh / kịch bản phát sinh
Không có.
Các yêu cầu đặc biệt khác
Không có.
Tình trạng trước khi thực hiện use-case
Ứng dụng phải được khởi động thành công và đang được đăng nhập bởi người dùng.
Tình trạng sau khi thực hiện use-case
1. Nếu use-case thực hiện thành công, thiết lập về thời gian nhắc nhở sẽ được lưu vào ứng dụng. 2. Use-case không thực hiện thành công, hiện trạng ứng dụng không thay đổi.
Điểm mở rộng
Không có.

Thiết kế UML

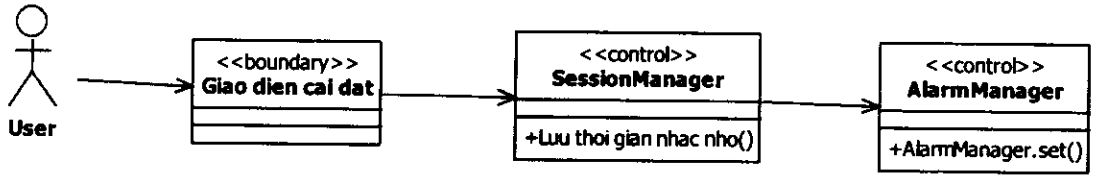
Sơ đồ lớp phân tích (Analysis class diagram)



Sơ đồ trình tự (Sequence diagram)

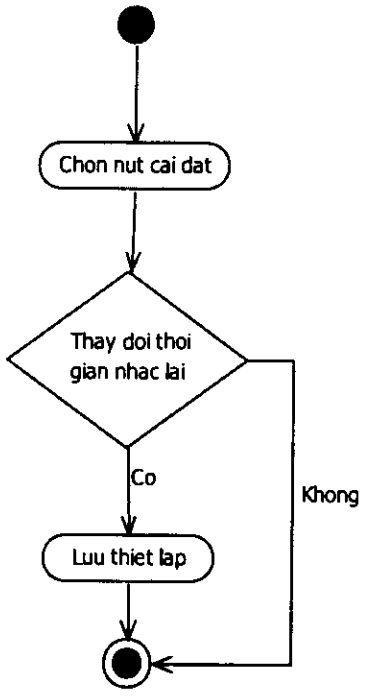


Sơ đồ lớp chi tiết (Class diagram)



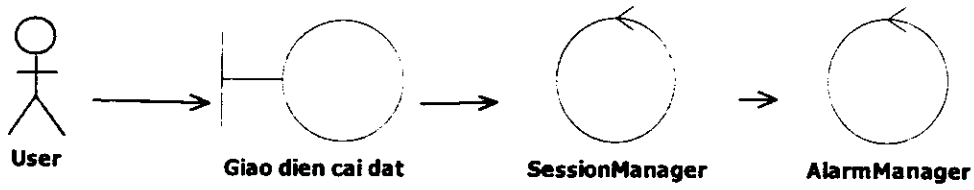
Hình ảnh minh họa
Thời gian hẹn giờ được chọn trong cài đặt    5 phút 30 phút 1 tiếng Thủ công Thời gian hẹn giờ 1 ngày 3 ngày 7 ngày 10 ngày Không báo

<table>
<tr><td colspan="2">UC #5</td><td>CÀI ĐẶT THỜI GIAN LẶP LẠI THÔNG BÁO</td><td>Độ phức tạp: Medium</td></tr>
<tr><td colspan="2">Mô tả</td><td colspan="2">Chức năng cho phép người dùng thay đổi khoảng thời gian lặp lại giữa mỗi lần thông báo báo trước thời điểm diễn ra các môn thi trong lịch thi.</td></tr>
<tr><td rowspan="2">Tác nhân</td><td>Chính</td><td colspan="2">Người dùng.</td></tr>
<tr><td>Phụ</td><td colspan="2">Không có.</td></tr>
<tr><td colspan="2">Tiền điều kiện</td><td colspan="2">Ứng dụng khởi chạy không có lỗi, đang có tài khoản đăng nhập ứng dụng.</td></tr>
<tr><td rowspan="2">Hậu điều kiện</td><td>Thành công</td><td colspan="2">Thời gian báo trước ngày bắt đầu một môn thi được thay đổi.</td></tr>
<tr><td>Lỗi</td><td colspan="2">Ứng dụng không có thay đổi.</td></tr>
<tr><td colspan="4">ĐẶC TẢ CHỨC NĂNG</td></tr>
<tr><td colspan="4">Luồng sự kiện chính / kịch bản chính</td></tr>
<tr><td colspan="4">Chức năng bắt đầu khi người dùng chọn chức năng "Cài đặt":
1. Giao diện cài đặt hiển thị lên cho người dùng biết họ đang để thời gian lặp lại thông báo là bao nhiêu;
2. Thời gian lặp lại thông báo có thể thay đổi giữa các mốc:
• 1 phút;
• 10 phút;
• 30 phút;
• 60 phút;
• Không lặp lại.
3. Khi người dùng chọn một mốc thời gian, ứng dụng sẽ lưu lại và tự động lặp lại thông báo theo mốc thời gian đã được chọn;
4. Nếu người dùng chọn "Không lặp lại", ứng dụng sẽ không thực hiện việc lặp lại thông báo và thông báo đó chỉ được xuất hiện 1 lần duy nhất cho người dùng biết khi có môn sắp đến ngày thi.</td></tr>
</table>

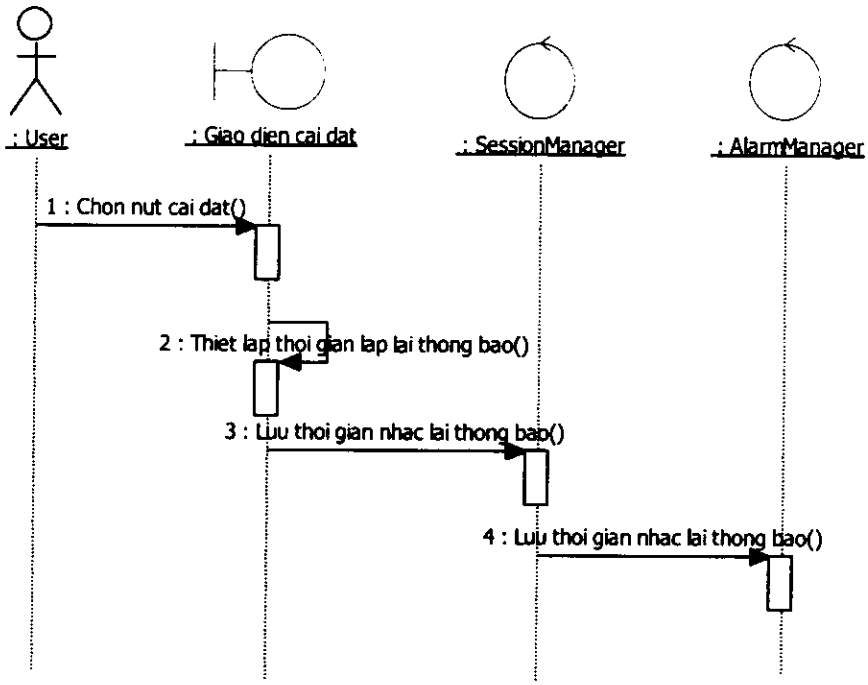
Sơ đồ hành động (Activity diagram)

Luồng sự kiện phát sinh / kịch bản phát sinh
Không có.
Các yêu cầu đặc biệt khác
Không có.
Tình trạng trước khi thực hiện use-case
Ứng dụng phải được khởi động thành công và đang được đăng nhập bởi người dùng.
Tình trạng sau khi thực hiện use-case
1. Nếu use-case thực hiện thành công, thiết lập về thời gian lặp lại thông báo sẽ được lưu vào ứng dụng. 2. Use-case không thực hiện thành công, hiện trạng ứng dụng không thay đổi.
Điểm mở rộng
Không có.

Thiết kế UML

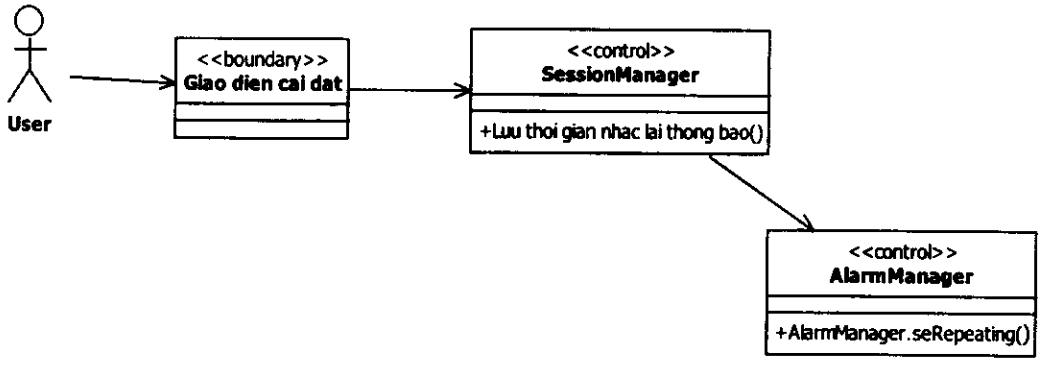
Sơ đồ lớp phân tích (Analysis class diagram)

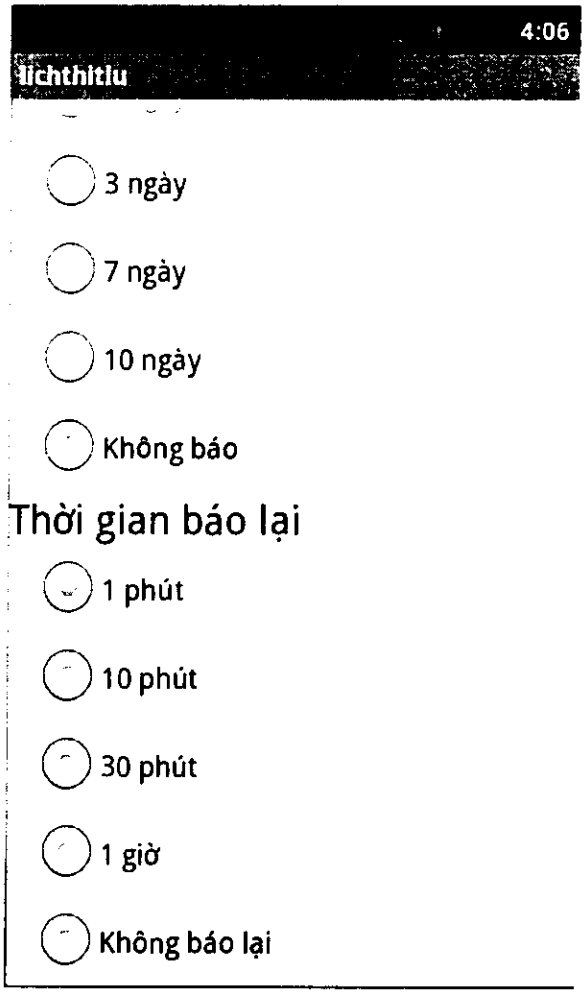


Sơ đồ trình tự (Sequence diagram)

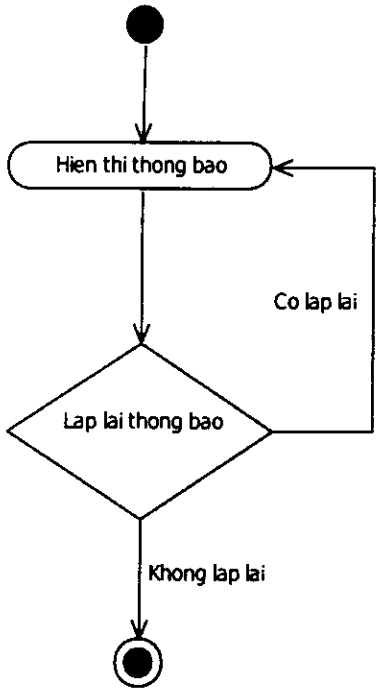


Sơ đồ lớp chi tiết (Class diagram)



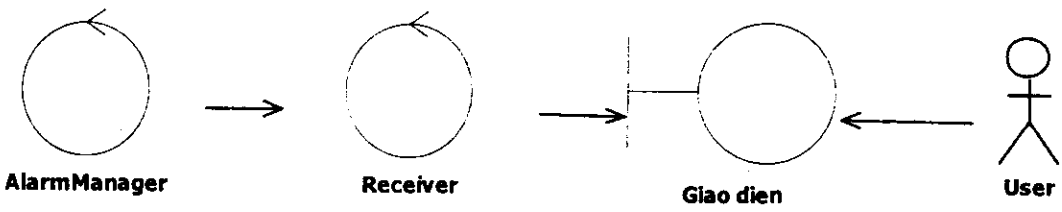
Hình ảnh minh họa
Thời gian hẹn giờ được chọn trong cài đặt 

UC #6		NHẮC LỊCH THI	Độ phức tạp: Medium
Mô tả		Chức năng hiển thị thông báo nhắc nhở về lịch thi cho người dùng	
Tác nhân	Chính	Hệ thống.	
	Phụ	Không có.	
Tiền điều kiện		Ứng dụng khởi chạy không có lỗi, đang có tài khoản đăng nhập ứng dụng.	
Hậu điều kiện	Thành công	Hiển thị thông báo.	
	Lỗi	Ứng dụng không có thay đổi.	
ĐẶC TẢ CHỨC NĂNG			
Luồng sự kiện chính / kịch bản chính			
Chức năng bắt đầu khi người dùng đã thiết lập đầy đủ các cài đặt: 1. Đến thời gian người dùng đã cài đặt, hệ thống đưa ra thông báo nhắc nhở về lịch thi của người dùng. 2. Nếu người dùng cài đặt chế độ nhắc lại thông báo. Hệ thống sẽ tự động nhắc lại thông báo đó theo thời gian định trước. 3. Khi người dùng chọn vào thông báo đó, một thông báo khác được hiển thị. Thông báo này cho phép người dùng chọn có tiếp tục lặp lại thông báo về môn thi đó không. - Khi người dùng chọn tiếp tục lặp lại, thông báo về môn thi sẽ tiếp tục được báo lại theo thời gian đã định trước. - Khi người dùng chọn không báo lại, thông báo đó sẽ không được lặp lại nữa.			

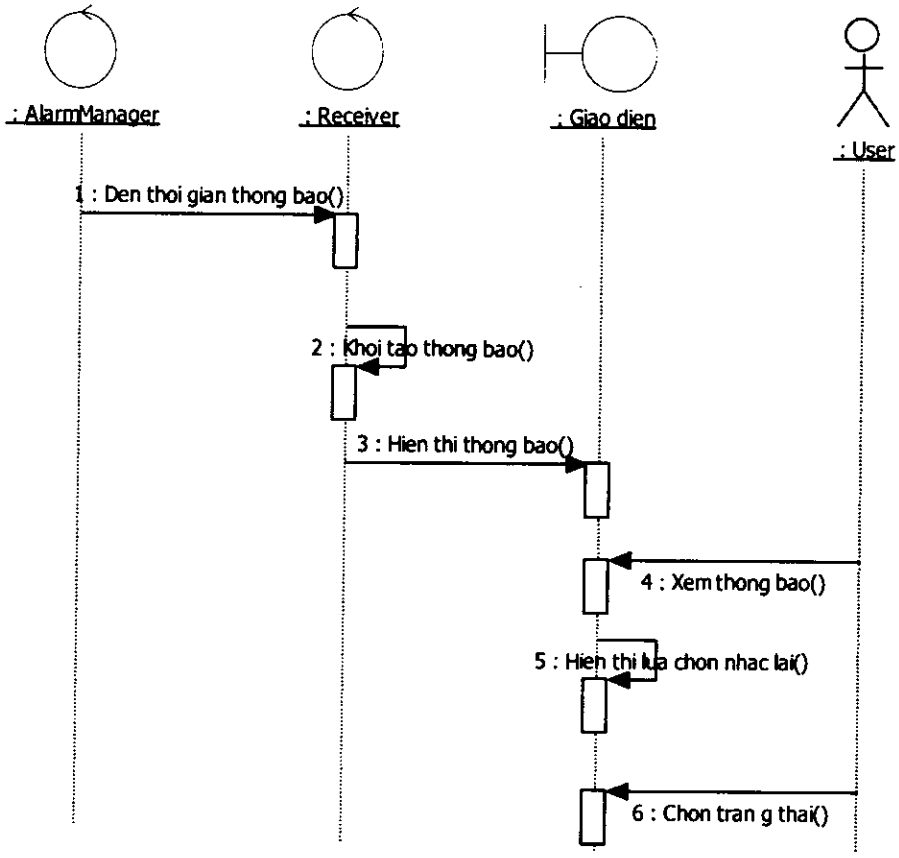
Sơ đồ hành động (Activity diagram)

Luồng sự kiện phát sinh / kịch bản phát sinh
Không có.
Các yêu cầu đặc biệt khác
Không có.
Tình trạng trước khi thực hiện use-case
Ứng dụng phải được khởi động thành công và đang được đăng nhập bởi người dùng.
Tình trạng sau khi thực hiện use-case
1. Nếu use-case thực hiện thành công, thông báo sẽ được hiển thị. 2. Use-case không thực hiện thành công, hiện trạng ứng dụng không thay đổi.
Điểm mở rộng
Không có.

Thiết kế UML

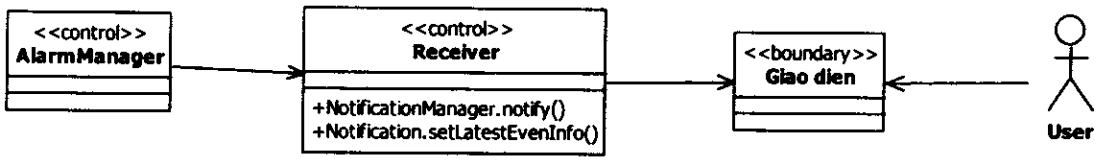
Sơ đồ lớp phân tích (Analysis class diagram)



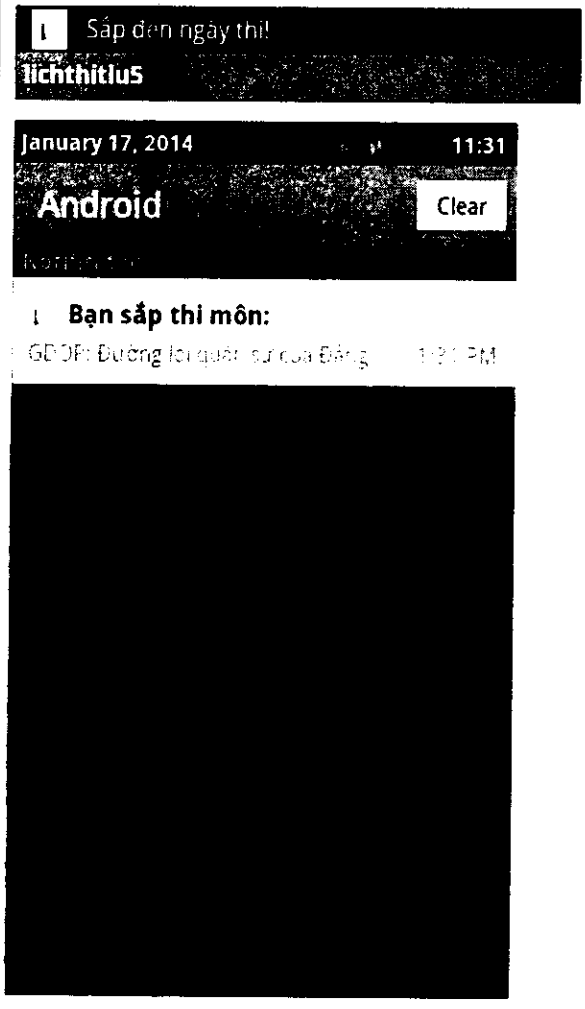
Sơ đồ trình tự (Sequence diagram)

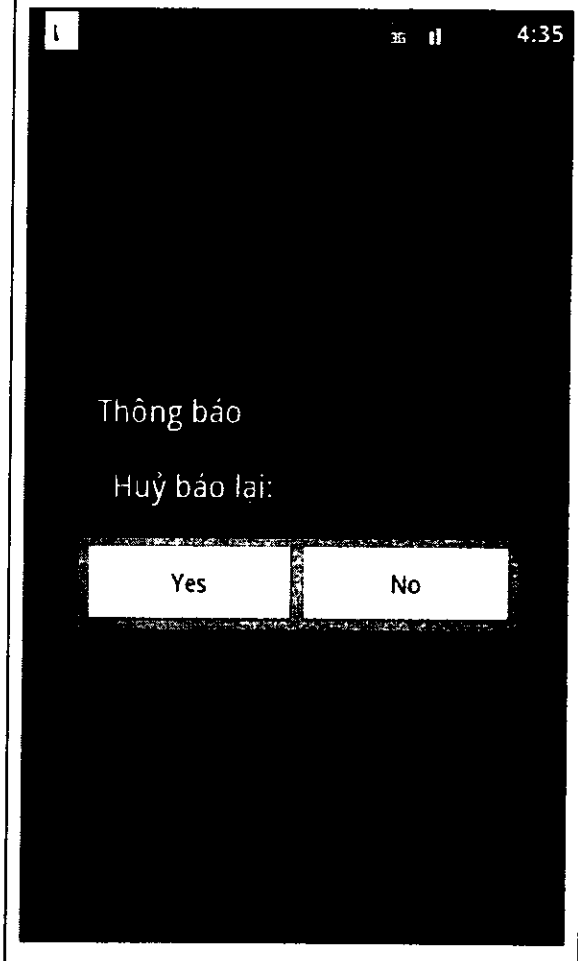


Sơ đồ lớp chi tiết (Class diagram)

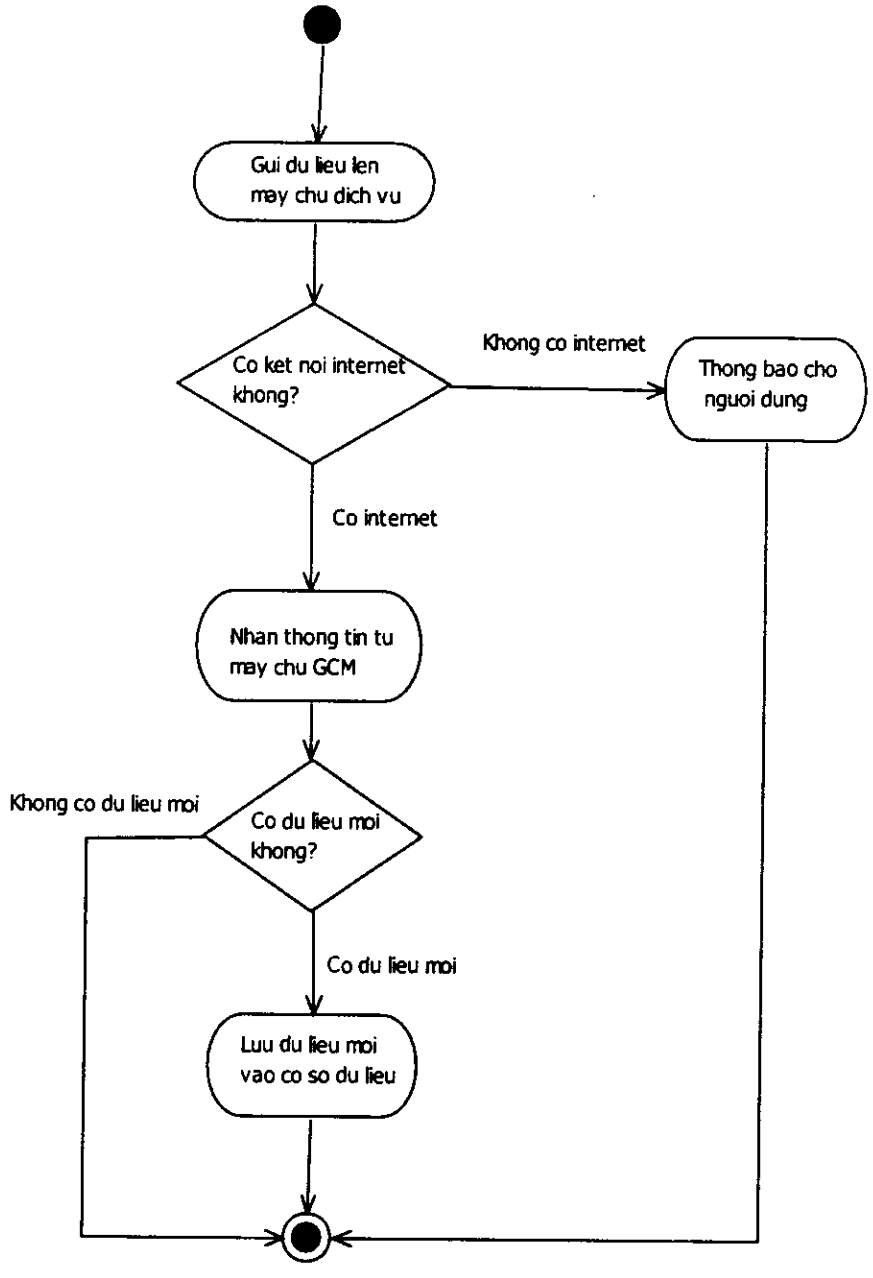


Hình ảnh minh họa	
Thông báo khi sắp đến ngày thi	Thông báo lựa chọn nhắc lại

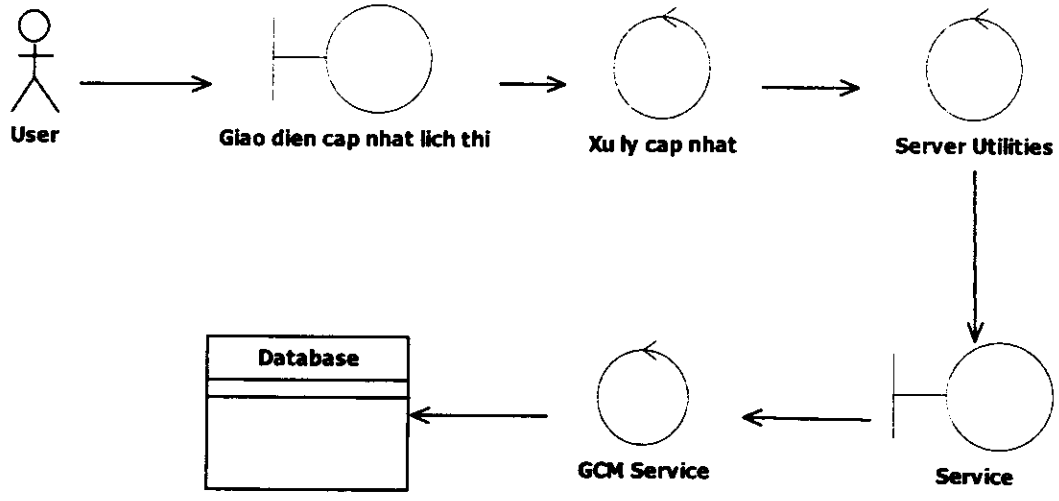




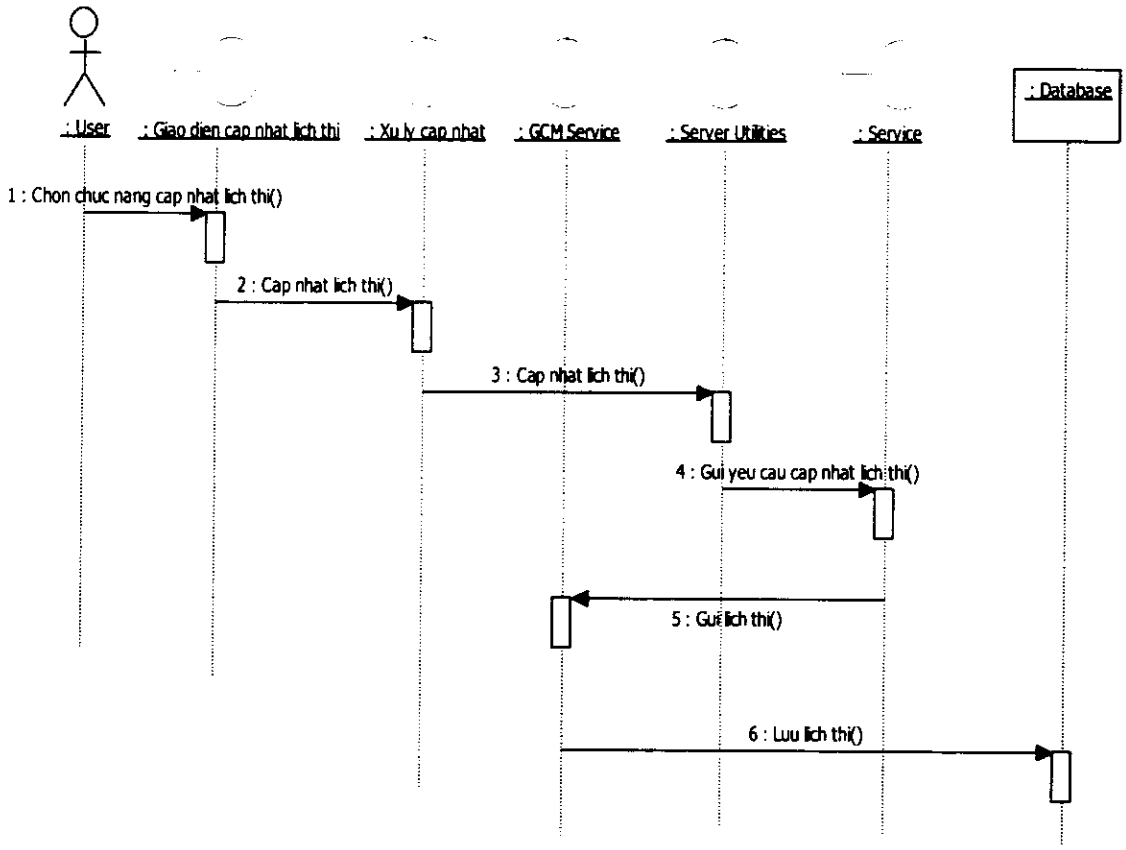
UC #7		CẬP NHẬT LỊCH THI	Độ phức tạp: Medium
Mô tả		Chức năng cho phép người dùng cập nhật lịch thi mới nhất khi chế độ cập nhật thủ công được cài đặt. Ở chế độ cập nhật tự động, chức năng này sẽ được gọi tự động mà không cần tác động của người dùng.	
Tác nhân	Chính	Người dùng.	
	Phụ	Không có.	
Tiền điều kiện		Ứng dụng khởi chạy không có lỗi, đang có tài khoản đăng nhập ứng dụng, có kết nối internet.	
Hậu điều kiện	Thành công	Nếu có thay đổi, lịch thi sẽ được cập nhật.	
	Lỗi	Ứng dụng không có thay đổi.	
ĐẶC TẢ CHỨC NĂNG			
Luồng sự kiện chính / kịch bản chính			
1. Chức năng được bắt đầu khi người dùng đăng nhập vào ứng dụng. 2. Nếu người dùng thiết lập chế độ cập nhật thủ công. Ứng dụng cho phép người dùng có thể cập nhật lịch thi mới nhất khi sử dụng chức năng "Cập nhật lịch thi". 3. Nếu người dùng thiết lập chế độ cập nhật tự động. Ứng dụng sẽ tự động cập nhật lịch thi mới nhất theo thời gian đã cài đặt.			

Sơ đồ hành động (Activity diagram)

Luồng sự kiện phát sinh / kịch bản phát sinh
Không có.
Các yêu cầu đặc biệt khác
Không có.
Tình trạng trước khi thực hiện use-case
Người dùng phải đăng nhập vào ứng dụng thành công.

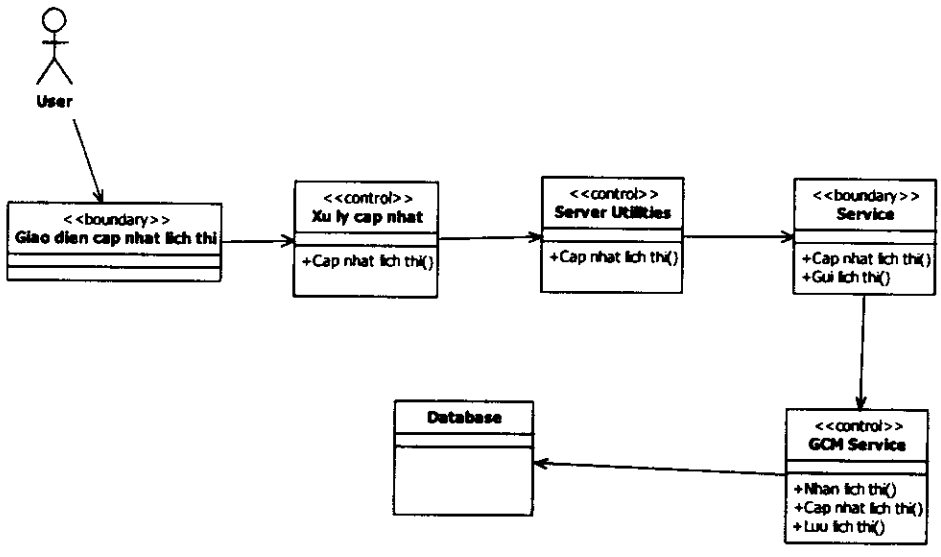
Tình trạng sau khi thực hiện use-case
1. Sau khi thực hiện Use – case thành công, lịch thi mới nhất sẽ được hiển thị lên màn hình. 2. Nếu Use – case thất bại thì trạng thái ứng dụng trước đó không bị thay đổi.
Điểm mở rộng
Không có
Thiết kế UML
Sơ đồ lớp phân tích (Analysis class diagram)

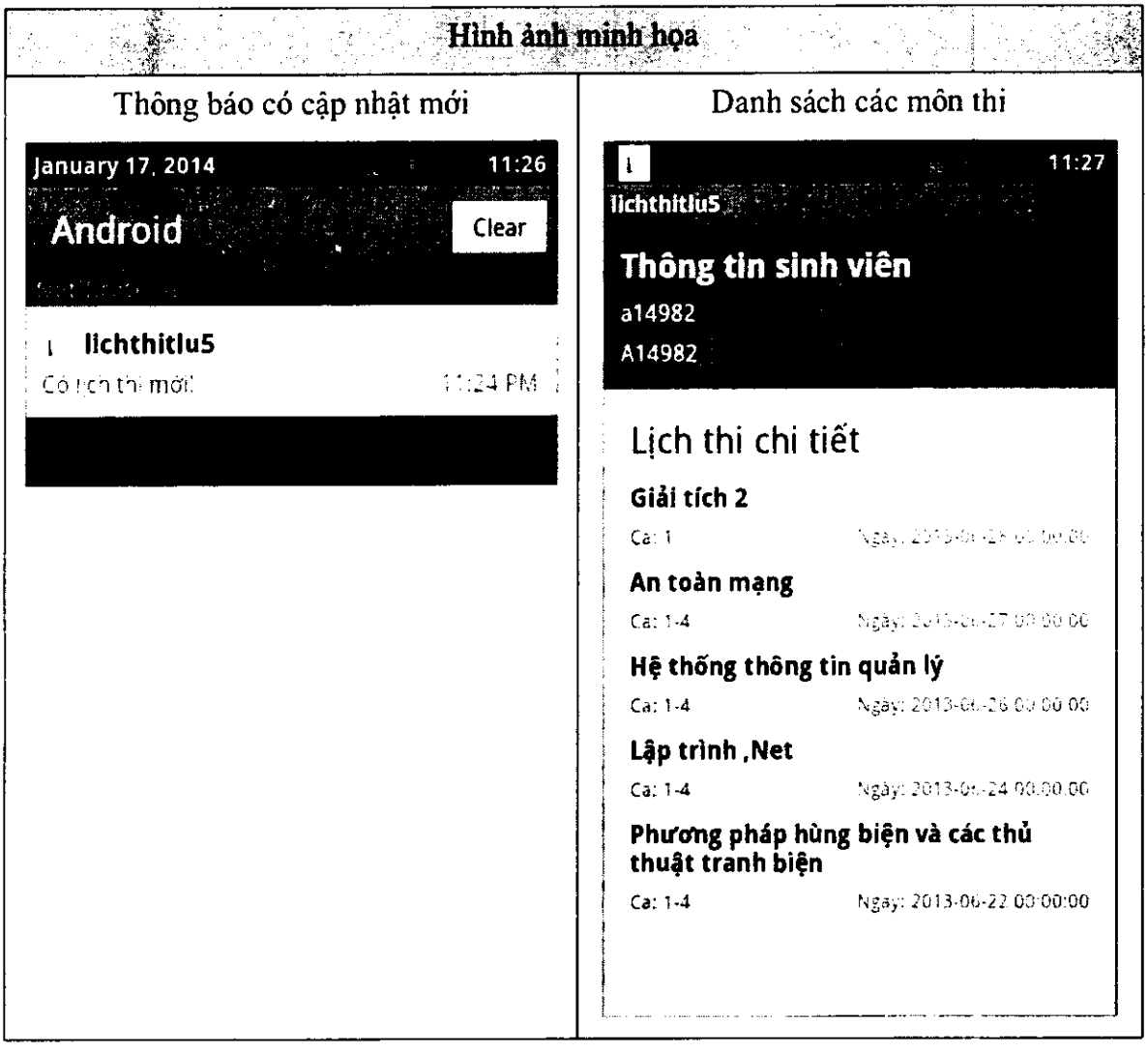


Sơ đồ trình tự (Sequence diagram)



Sơ đồ lớp chi tiết (Class diagram)



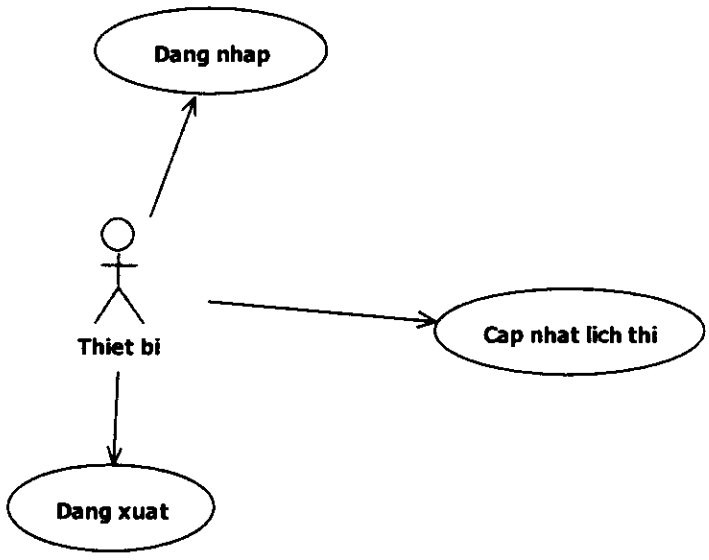


4.3. Ứng dụng máy chủ

4.3.1. Mô tả

Đây là ứng dụng phía máy chủ (server) có nhiệm vụ tiếp nhận thông tin từ ứng dụng (client) nhằm thực hiện các chức năng lưu giữ thông tin và kết nối với hệ thống thông tin của trường Đại học Thăng Long để lấy dữ liệu về lịch thi.

4.3.2. Sơ đồ tổng quan các chức năng của máy chủ



4.3.3. Các tác nhân tham gia

Hệ thống hoạt động tự động mà không cần có sự tham gia của con người.

4.3.4. Các chức năng chính của hệ thống

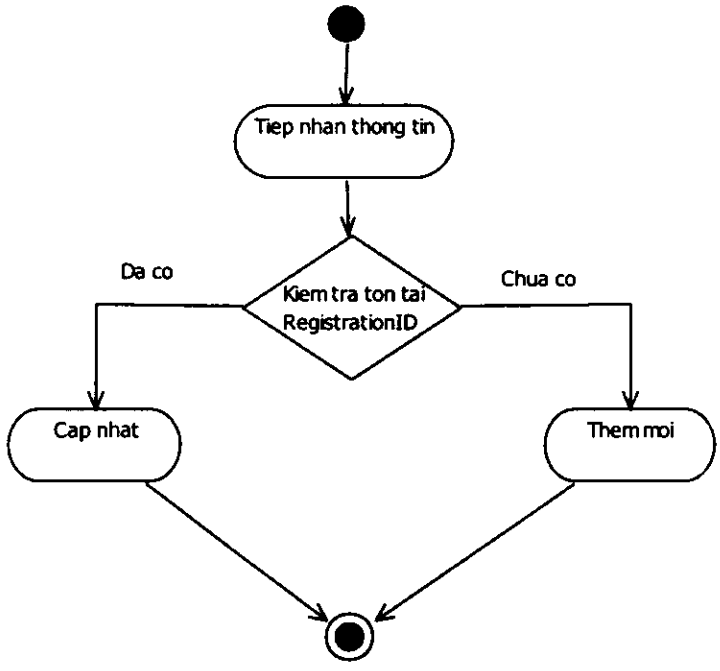
Đăng nhập: chức năng này tiếp nhận và xử lý thông tin đăng nhập của người dùng được yêu cầu từ phía ứng dụng.

Đăng xuất: chức năng này tiếp nhận và xử lý thông tin đăng xuất của người dùng được yêu cầu từ phía ứng dụng.

Cập nhật lịch thi: chức năng này tiếp nhận và xử lý thông tin từ chức năng cập nhật lịch thi được yêu cầu từ phía ứng dụng.

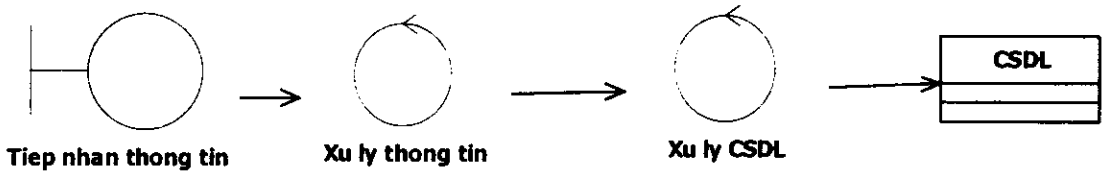
4.3.5. Đặc tả các chức năng của ứng dụng

UC #1		ĐĂNG NHẬP	Độ phức tạp: Medium
Mô tả		Chức năng tiếp nhận thông tin và xử lý thông tin đăng nhập của người dùng được yêu cầu từ phía ứng dụng.	
Tác nhân	Chính	Không có.	
	Phụ	Không có.	
Tiền điều kiện		Hệ thống không có lỗi.	
Hậu điều kiện	Thành công	Thông tin được yêu cầu từ phía ứng dụng và xử lý. Thông tin về người dùng được lưu trữ tại máy chủ.	
	Lỗi	Hệ thống không thay đổi.	

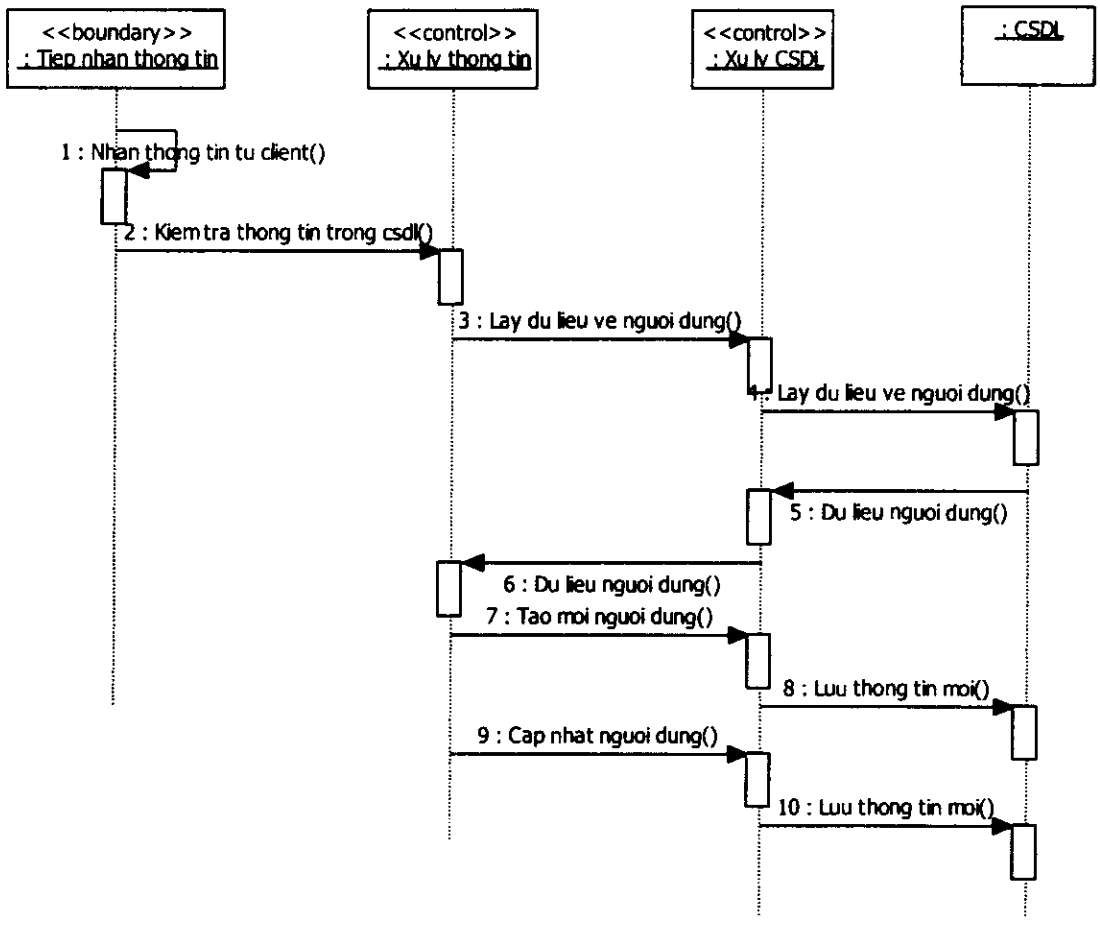
ĐẶC TẢ CHỨC NĂNG
Luồng sự kiện chính / kịch bản chính
Chức năng này hoạt động khi có yêu cầu từ phía ứng dụng: 1. Hệ thống tiếp nhận thông tin được gửi đến từ ứng dụng; 2. Hệ thống kiểm tra thông tin trong cơ sở dữ liệu để tránh trùng lặp dữ liệu; 3. Thông tin về người dùng được lưu trong cơ sở dữ liệu của máy chủ.
Sơ đồ hành động (Activity diagram)

Luồng sự kiện phát sinh / kịch bản phát sinh
Không có
Các yêu cầu đặc biệt khác
Không có
Tình trạng trước khi thực hiện use-case
Server đã được bật và hoạt động bình thường
Tình trạng sau khi thực hiện use-case
Thông tin về người dùng được lưu lại trong cơ sở dữ liệu
Điểm mở rộng
Không có

Thiết kế UML

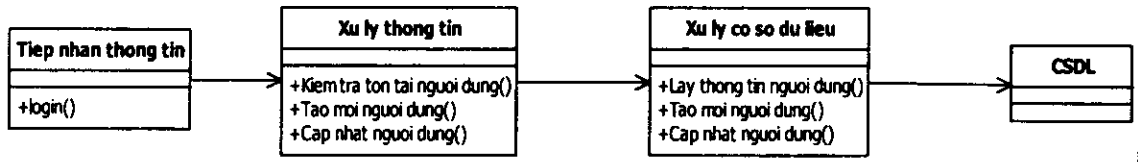
Sơ đồ lớp phân tích (Analysis class diagram)



Sơ đồ trình tự (Sequence diagram)



Sơ đồ lớp chi tiết (Class diagram)



UC #2		ĐĂNG XUẤT	Độ phức tạp: Medium
Mô tả		Chức năng tiếp nhận thông tin và xử lý thông tin đăng xuất của người dùng được yêu cầu từ phía ứng dụng.	
Tác nhân	Chính	Không có.	
	Phụ	Không có.	
Tiền điều kiện		Hệ thống không có lỗi.	
Hậu điều kiện	Thành công	Thông tin được yêu cầu từ phía ứng dụng và xử lý. Thông tin về người dùng được xử lý và lưu trữ tại máy chủ.	
	Lỗi	Hệ thống không thay đổi.	

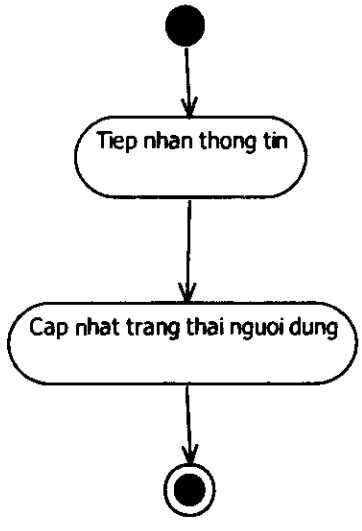
ĐẶC TẢ CHỨC NĂNG

Luồng sự kiện chính / kịch bản chính

Chức năng này hoạt động khi có yêu cầu từ phía ứng dụng:

1. Hệ thống tiếp nhận thông tin được gửi đến từ ứng dụng;
2. Hệ thống kiểm tra thông tin trong cơ sở dữ liệu để tránh trùng lặp dữ liệu;
3. Thông tin về người dùng được thay đổi trong cơ sở dữ liệu của máy chủ, mã sinh viên sẽ được xóa tương ứng với RegistrationID được lưu trong cơ sở dữ liệu.

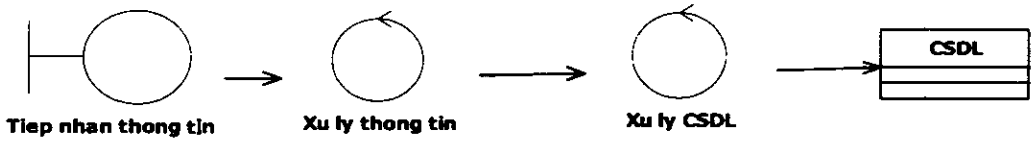
Sơ đồ hành động (Activity diagram)



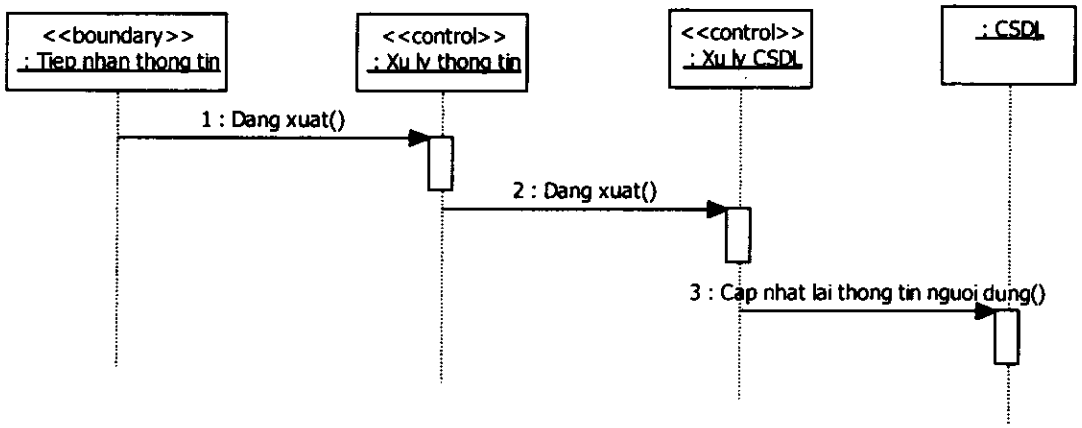
Luồng sự kiện phát sinh / kịch bản phát sinh

Không có

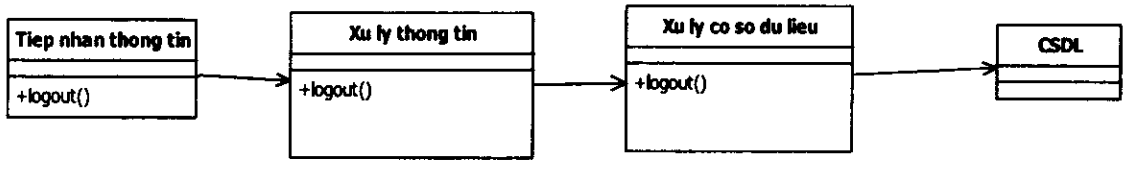
Các yêu cầu đặc biệt khác
Không có
Tình trạng trước khi thực hiện use-case
Mã sinh viên tương ứng với REGID được lưu trữ cùng với mã phiên bản lịch thi
Tình trạng sau khi thực hiện use-case
Thông tin về REGID vẫn được lưu trữ nhưng thông tin về mã sinh viên và mã phiên bản lịch thi bị xóa bỏ
Điểm mở rộng
Không có
Thiết kế UML
Sơ đồ lớp phân tích (Analysis class diagram)



Sơ đồ trình tự (Sequence diagram)



Sơ đồ lớp chi tiết (Class diagram)



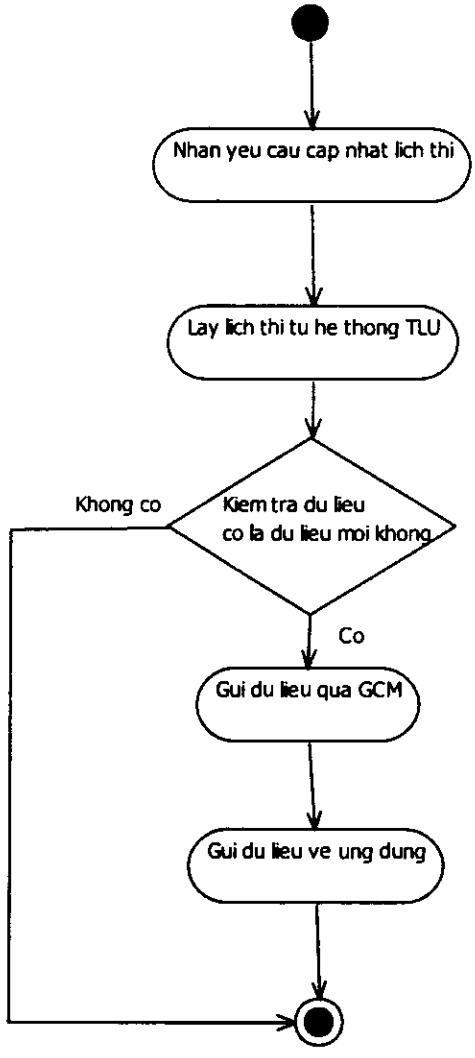
UC #3		CẬP NHẬT LỊCH THI	Độ phức tạp: Cao
Mô tả		Chức năng tiếp nhận thông tin và xử lý thông tin cập nhật lịch thi của người dùng được yêu cầu từ phía ứng dụng.	
Tác nhân	Chính	Không có.	
	Phụ	Không có.	
Tiền điều kiện		Hệ thống không có lỗi.	
Hậu điều kiện	Thành công	Thông tin được yêu cầu từ phía ứng dụng và xử lý. Lịch thi mới được gửi trả về cho ứng dụng.	
	Lỗi	Hệ thống không thay đổi.	

ĐẶC TẢ CHỨC NĂNG

Luồng sự kiện chính / kịch bản chính

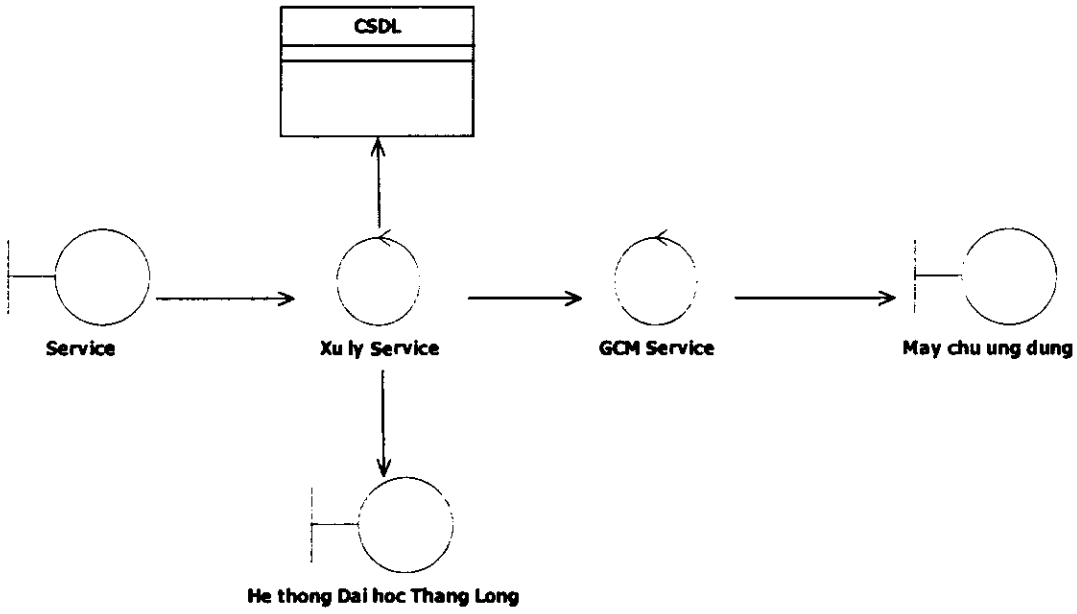
Chức năng này hoạt động khi có yêu cầu từ phía ứng dụng:

1. Hệ thống tiếp nhận thông tin được gửi đến từ ứng dụng;
2. Hệ thống kiểm tra thông tin trong cơ sở dữ liệu;
3. Hệ thống lấy dữ liệu từ hệ thống thông tin trường Đại học Thăng Long dựa vào thông tin được cung cấp từ ứng dụng;
4. Hệ thống kiểm tra tính mới của dữ liệu được lấy về;
5. Gửi trả dữ liệu lịch thi mới tới ứng dụng thông qua GCM;
6. Thông tin về lịch thi và người dùng được lưu trữ trong cơ sở dữ liệu.

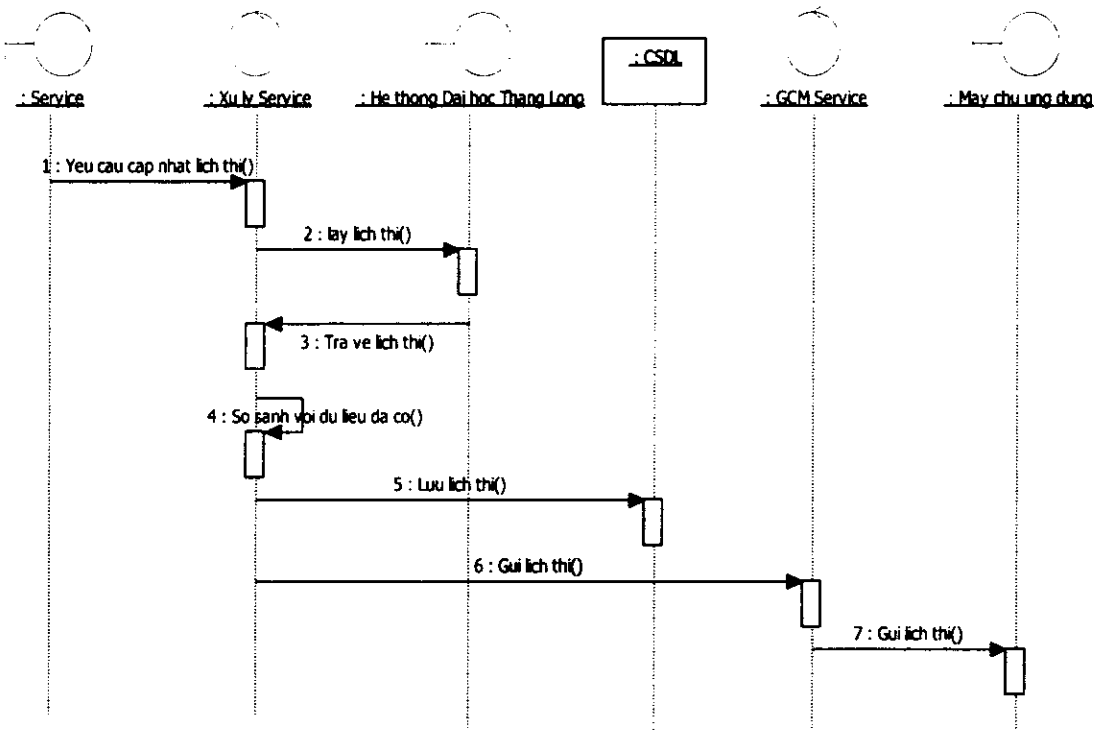
Sơ đồ hành động (Activity diagram)

Luồng sự kiện phát sinh / kịch bản phát sinh
Không có.
Các yêu cầu đặc biệt khác
Không có.
Tình trạng trước khi thực hiện use-case
Hệ thống khởi động thành công.
Tình trạng sau khi thực hiện use-case
Lịch thi mới được gửi tới ứng dụng thành công.
Điểm mở rộng
Không có.

Thiết kế UML

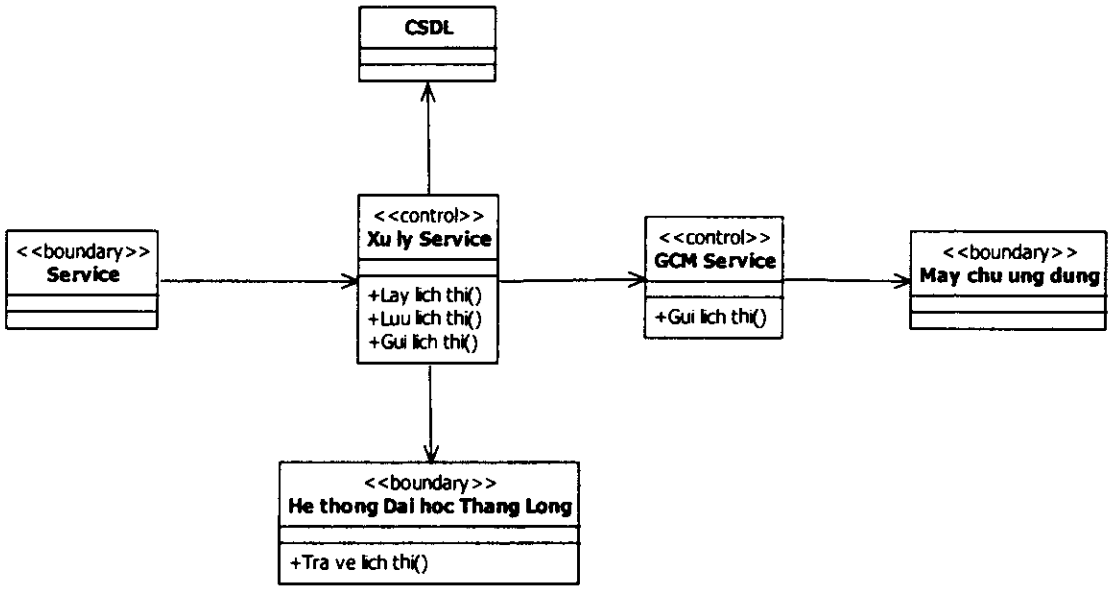
Sơ đồ lớp phân tích (Analysis class diagram)



Sơ đồ trình tự (Sequence diagram)



Sơ đồ lớp chi tiết (Class diagram)



4.4. Phân tích thiết kế dữ liệu

4.4.1. Mô tả phân tích dữ liệu

Hệ thống nhắc lịch thi cần có cơ sở dữ liệu để đảm bảo được các chức năng sau:

– Thông tin về người dùng và thiết bị (mã sinh viên và RegistrationID) được lưu trữ trên hệ thống;

– Thông tin về lịch thi mới nhất được lưu trữ tại thiết bị của người dùng.

Do đó, cần phải có hai cơ sở dữ liệu để lưu trữ thông tin:

– Cơ sở dữ liệu lưu trữ thông tin người dùng và thiết bị được sử dụng cho ứng dụng máy chủ;

– Cơ sở dữ liệu lưu trữ lịch thi được sử dụng cho ứng dụng Android.

Các thông tin cần lưu trữ:

– Ứng dụng máy chủ: thông tin về người dùng

+ Mã sinh viên;

+ Mã thiết bị (RegistrationID);

+ Phiên bản lịch thi.

– Ứng dụng Android: Lịch thi của sinh viên

+ Mã môn thi;

+ Tên môn thi;

+ Ngày thi;

+ Ca thi;

+ Phòng thi;

+ Tình trạng.

4.4.2. Cấu trúc bảng

Ứng dụng máy chủ: Bảng ACCOUNT

Colum name	Description	Datatype	Length	Allow null
ID	Mã định danh	int		No
MASV	Mã sinh viên	Varchar	6	No
REGID	Mã định danh	text		No
VERSION	Phiên bản lịch thi	text		Yes
PRIMARY KEY ID				

Ứng dụng Android: Bảng LICHTHI

Colum name	Description	Datatype	Length	Allow null
ID	Mã định danh	int		No
MAMON	Mã môn thi	Varchar	6	No
TENMON	Tên môn thi	varchar	50	No
NGAYTHI	Ngày diễn ra môn thi	datetime		No
CATHI	Ca thi	Varchar	5	No
PHONGTHI	Phòng thi	varchar	10	No
TINHTRANG	Tình trạng của môn thi	varchar	5	No
PRIMARY KEY ID				

4.5. Kết quả xây dựng ứng dụng

– Ứng dụng hoạt động tốt đúng như yêu cầu đã đặt ra;

– Giao diện thân thiện, dễ dàng sử dụng;

– Lịch thi được cập nhật nhanh chóng, chính xác. Hệ thống nhắc nhở hoạt động chính xác.

CHƯƠNG 5. CÁC KỸ THUẬT XỬ LÝ QUAN TRỌNG

Trong chương trước, nhóm tác giả đã xây dựng hướng phát triển cho hệ thống nhắc lịch thi cho sinh viên Thăng Long. Đó mới là những bước phác thảo, phân tích thiết kế cơ bản trong quá trình phát triển phần mềm. Trong chương này sẽ đi sâu vào thể hiện các kỹ thuật như xử lý thông điệp được gửi về từ GCM, lưu trữ dữ liệu trên thiết bị Android, sử dụng service trên Android...

Để tiện theo dõi, nhóm tác giả sẽ trình bày theo trình tự gửi nhận dữ liệu theo mô hình 3-1.

5.1. Đăng ký thiết bị với GCM để nhận RegistrationID

Trước hết, ta phải khai báo cung cấp quyền cho các thành phần được phép chạy trên ứng dụng trong file AndroidManifest.xml. Những quyền này là cần thiết để ứng dụng có thể sử dụng được GCM.

```
<permission
    android:name="com.pushschedule.permission.C2D_MESSAGE"
    android:protectionLevel="signature">
</permission>

<uses-permission android:name="com.pushschedule.permission.C2D_MESSAGE" />
<uses-permission android:name="com.google.android.c2dm.permission.RECEIVE" />
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.GET_ACCOUNTS" />
<uses-permission android:name="android.permission.WAKE_LOCK" />
<uses-permission
android:name="android.permission.ACCESS_NETWORK_STATE" />
<uses-permission android:name="android.permission.VIBRATE"/>
```

- INTERNET – Khai báo cho phép sửa dụng dịch vụ internet;
- ACCESS_NETWORK_STATE – Để truy cập trạng thái mạng (dùng để xác định trạng thái hiện thời của mạng;
- GET_ACCOUNTS – GCM cần phải có tài khoản Google;
- WAKE_LOCK – Đánh thức thiết bị khi nhận được thông điệp/tin nhắn;
- VIBRATE – Cho phép rung.

Sau khi đã khai báo xong, để có thể sử dụng được dịch vụ GCM, thiết bị cần phải có RegistrattionID. Đăng ký registration ID:

```
GCMRegistrar.checkDevice(_context);
GCMRegistrar.checkManifest(_context);
regId = GCMRegistrar.getRegistrationId(_context,SenderID);
```

GCMRegistrar là một lớp trong thư viện GCM của google và được thêm vào qua khai báo:

```
import com.google.android.gcm.GCMRegistrar;
```

Khi gọi hàm GCMRegistrar.getRegistrationId(), thiết bị sẽ được đăng ký với GCM với tham số truyền vào SenderID, giá trị trả về sẽ là một chuỗi định danh duy nhất thiết bị.

5.2. Gửi thông tin yêu cầu đến máy chủ ứng dụng (application server)

Trong ứng dụng này, để phục vụ cho việc giao tiếp, gửi yêu cầu từ thiết bị đến máy chủ, nhóm tác giả đã xây dựng một lớp riêng biệt để đảm trách nhiệm vụ này. Để có thể giao tiếp với máy chủ, ứng dụng sử dụng giao thức HTTP để truyền dữ liệu.

```
public static void postToServer(List<NameValuePair> nameValuePairs){
    try{
          // url where the data will be posted
          String postReceiverUrl = "http://tlu.hol.es/post_data_receiver.php";
          Log.v(TAG, "postURL: " + postReceiverUrl);

          // HttpClient
          HttpClient httpClient = new DefaultHttpClient();
          HttpPost httpPost = new HttpPost(postReceiverUrl);

          httpPost.setEntity(new UrlEncodedFormEntity(nameValuePairs));

          // execute HTTP post request
          HttpResponse response = httpClient.execute(httpPost);
          HttpEntity resEntity = response.getEntity();

          if (resEntity != null) {

                String responseStr = EntityUtils.toString(resEntity).trim();
                Log.v(TAG, "Response: " +  responseStr);
          }

    } catch (ClientProtocolException e) {
          e.printStackTrace();
    } catch (IOException e) {
          e.printStackTrace();
    }
}
```

Hàm postToServer này có nhiệm vụ nhận đối số truyền vào là danh sách các tham số cần thiết và gửi chúng đến Server để khi nhận được gói tin HTTP, Server sẽ có đủ dữ liệu để xử lý. Để thực thi việc gửi, ứng dụng sử dụng các thư viện sẵn có của java như HttpEntity, HttpResponse, HttpClient... thông qua lệnh khai báo import:

```
import org.apache.http.HttpEntity;
import org.apache.http.HttpResponse;
import org.apache.http.NameValuePair;
import org.apache.http.client.ClientProtocolException;
import org.apache.http.client.HttpClient;
import org.apache.http.client.entity.UrlEncodedFormEntity;
import org.apache.http.client.methods.HttpPost;
import org.apache.http.entity.StringEntity;
import org.apache.http.impl.client.DefaultHttpClient;
import org.apache.http.message.BasicNameValuePair;
import org.apache.http.protocol.HTTP;
```

5.3. Server nhận thông tin và xử lý dữ liệu

Trong ứng dụng này, tác giả sử dụng phương thức POST của giao thức HTTP để gửi dữ liệu đến Server. Do đó, phía server sẽ nhận được dữ liệu thông qua mảng $_POST (Server được xây dựng bằng PHP). Dựa vào các dữ liệu được chứa trong mảng $_POST mà Server sẽ tiến hành phân tích và sử lý dữ liệu. Quy trình được mô tả khái quát như sau:

– Tiến hành phân tách các thành phần của mảng $_POST để có được RegistrationID và hành động thực hiện (đăng nhập, đăng xuất hay cập nhật lịch thi).

– Xử lý hành động:

+ Nếu là đăng nhập thì Server sẽ tiến hành lưu giá trị mã sinh viên được gửi lên cùng với RegistrationId tương ứng;

+ Nếu là đăng xuất thì Server sẽ tiến xóa mã sinh viên với RegistrationId tương ứng;

+ Nếu là cập nhật lịch thi, Server sẽ tiến hành việc cập nhật lịch thi thông qua Webservice của nhà trường để có thể lấy được lịch thi chính xác nhất.

Sử dụng SOAP trong PHP để kết nối đến Webservice và truy vấn dữ liệu

Giới thiệu qua sơ về SOAP, SOAP là gì?

SOAP là viết tắt của Simple Object Access Protocol, là một giao thức giao tiếp có cấu trúc như XML và mã hóa thành định dạng chung cho các ứng dụng trao đổi với nhau. SOAP là một đặc tả việc sử dụng các tài liệu XML theo dạng các thông điệp. Bản thân SOAP không định ra các ngữ nghĩa ứng dụng hoặc cách cài đặt chi tiết. SOAP cung cấp một cơ chế đơn giản và gọn nhẹ cho việc trao đổi thông tin có cấu trúc và định dạng giữa các thành phần trong một môi trường phân tán sử dụng XML. SOAP được thiết kế dựa trên những chuẩn nhằm giảm chi phí tích hợp các hệ thống phân tán xây dựng trên

nhiều nền tảng khác nhau ở mức càng thấp càng tốt. Đặc tả về SOAP định nghĩa một mô hình trao đổi dữ liệu dựa trên 3 khái niệm cơ bản: Các thông điệp là các tài liệu XML, chúng được truyền đi từ bên gửi đến bên nhận, bên nhận có thể chuyển tiếp dữ liệu đến nơi khác.

Khái niệm cơ bản nhất của mô hình SOAP là việc sử dụng các tài liệu XML như những thông điệp trao đổi. Điều này có nhiều ưu điểm hơn các giao thức truyền dữ liệu khác. Các thông điệp XML có thể được tổng hợp và đọc với một bộ soạn thảo text đơn giản, ta có thể làm việc với XML trên hầu hết mọi nền tảng.

Lấy dữ liệu từ Webservice thông qua SOAP:

```
$client = new SoapClient($url);
$lichthi = $client->GetLichThi(array("MaSinhVien"=>$studycode,
                                     "user"=>"test",
                                     "pass"=>"test!@#")
                               )->GetLichThiResult->CLichThi;
```

Hàm SoapClient($url) sẽ khởi tạo nên một đối tượng SOAP client kết nối đến Webservice thông qua đường dẫn $url.Trong trường hợp này,

$url = http://dkonline.thanglong.edu.vn/formobile.asmx?WSDL

$lichthi là một mảng các giá trị được $client lấy về thông qua hàm GetLichThi(). Hàm này được phía Webservice cung cấp để cho phép các máy trạm (client) truy cập và lấy dữ liệu. Dữ liệu được trả về ở dạng mảng JSON.

Xử lý dữ liệu sau khi lấy được từ Webservice của nhà trường

Dữ liệu được trả về ở dạng mảng, do đó sử dụng vòng lặp để có thể bóc tách toàn bộ dữ liệu thu được và định dạng lại dữ liệu chuẩn bị cho quá trình gửi dữ liệu cho GCM.

```
foreach ($lichthi as $monthi) {
    $ngaythi = explode("T", $monthi->NgayThi);
    $ngaythi = $ngaythi[0];
    $date = new DateTime($ngaythi);
    $ngaythi = $date->format("Y-m-d");
    if( $startDay > strtotime($ngaythi))
          break;
    $ct = explode("/",$monthi->CaThiMaPhongThi);
    $pt = $ct[0];
    $ct = $ct[1];
    $subject = array("MASV"=>($monthi->MaSinhVien),
                     "TENSV"=>($monthi->MaSinhVien),
                     "MAMON"=>($monthi->MaHocPhan),
                       "TENMON"=>($monthi->TenHocPhan),
                       "NGAYTHI"=>(str_replace("T"," ",$monthi->NgayThi)),
                       "CATHI"=>($ct),
                       "PHONGTHI"=>($pt),
                       "TINHTRANG"=>($monthi->TinhTrang),
```

```
                        "VERSION"=>null);
    array_push($listSubject, $subject);
)
```

$listSubject sẽ chứa toàn bộ dữ liệu lịch thi đã được tổ chức và định dạng lại để cho phù hợp.

Sau khi đã tổ chức lại dữ liệu tạo nguồn đầu vào cho việc gửi thông điệp, $listSubject sẽ được mã hóa để lưu vào cơ sở dữ liệu nhằm xác định phiên bản sau này. Việc mã hóa sẽ đảm bảo với mỗi bản thông điệp khác nhau đều được xác định duy nhất.

5.4. Gửi thông điệp đến GCM

Để gửi được thông điệp đến GCM. ta tiếp tục sử dụng giao thức HTTP để truyền gói tin. cURL là một thư viện trong PHP cho phép tạo ra các yêu cầu HTTP.

```
public function send_notification($registatoin_ids, $message) {
        // include config
        //include_once './config.php';
        $doc = simplexml_load_file("server_config.xml") or die("Error: Cannot
create object");

        // Set POST variables
        $url = 'https://android.googleapis.com/gcm/send';

        $fields = array(
            'registration_ids' => $registatoin_ids,
            'data' => $message,
        );

        $headers = array(
            'Authorization: key=' . "AIzaSyBW5H0mAxNubGGVDbF74s5-mhoODDnTo9A",
            'Content-Type: application/json'
        );
        /*$headers = array(
            'Authorization: key=' . $GOOGLE_API_KEY,
            'Content-Type: application/json'
        );*/
        // Open connection
        $ch = curl_init();

        // Set the url, number of POST vars, POST data
        curl_setopt($ch, CURLOPT_URL, $url);

        curl_setopt($ch, CURLOPT_POST, true);
        curl_setopt($ch, CURLOPT_HTTPHEADER, $headers);
        curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);

        // Disabling SSL Certificate support temporarly
        curl_setopt($ch, CURLOPT_SSL_VERIFYPEER, false);
```

```
    curl_setopt($ch, CURLOPT_POSTFIELDS, json_encode($fields));

        // Execute post
        $result = curl_exec($ch);
        if ($result === FALSE) {
            die('Curl failed: ' . curl_error($ch));
        }

        // Close connection
        curl_close($ch);
        // echo json_encode($fields);
    }
```

Hàm send_notification() sẽ đảm nhiệm việc gửi dữ liệu đến GCM. Trong tổ chức gói tin gửi đến GCM cần các có các thông tin như RegistrationId để xác định thiết bị nhận, API key để xác thực dịch vụ và thông điệp gửi cần gửi.

5.5. Xử lý thông điệp được gửi đến từ GCM trên thiết bị Android

Đây là bước cuối cùng trong một quá trình gửi nhận thông tin qua GCM. Công nghệ Push Notification sử dụng GCM như một nơi lưu trữ các thông điệp được yêu cầu gửi đến thiết bị. Trong thực tế, không phải lúc nào thiết bị cũng được kết nối internet sẵn sàng, do đó GCM sẽ đóng vai trò là nơi lưu trữ thông tin, ngay khi thiết bị có kết nối internet, GCM sẽ lập tức "đẩy" các thông điệp xuống các thiết bị dựa vào RegistrationId.

Để ứng dụng nhận được thông điệp, cần xây dựng một lớp đảm nhiệm chức năng này và được kế thừa từ lớp GCMBaseIntentService trong thư viện GCM của Android.

Phương thức onMessage() được quá tải lại trong lớp kế thừa có chức năng nhận thông điệp được gửi đến qua GCM.

```
@Override
    protected void onMessage(Context context, Intent intent) {
            Log.i(TAG, "Received message");

            Bundle data = intent.getExtras();
            String message = data.getString("lichthi");
            DatabaseHandler db = new DatabaseHandler(this);
            generateNotification(context, "Có lịch thi mới!");
            Log.i("Chuoi nhan ve:",message);
            try {
                    lichthi = new JSONArray(message);
                    db.saveToDatabase(lichthi);
                    getList();
            } catch (JSONException e) {
                    // TODO Auto-generated catch block
                    e.printStackTrace();
            }
            getVersion(lichthi);
```

```
            getStudyInfo(lichthi);
            displayMessage(context, "new");
    }
```

Khi nhận được dữ liệu gửi về thông qua câu lệnh intent.getExtras(). Dữ liệu sẽ được lưu vào cơ sở dữ liệu trên thiết bị. Do được lưu trữ lại nên sinh viên có thể truy vấn lịch thi bất kỳ lúc nào mà không cần đến mạng internet.

```
public void saveToDatabase(JSONArray _lichthi){
    Log.i("DatabaseHandler","Deleting older record");
    deleteAllRecord();
    Log.d("DatabaseHandler: ", "Inserting ..");
    try{
            for(int i = 0; i<_lichthi.length();i++){
                    JSONObject monthi = _lichthi.getJSONObject(i);
                    MonThi mt = new MonThi(monthi.getString("MAMON"),
monthi.getString("TENMON"), monthi.getString("NGAYTHI"),
monthi.getString("CATHI"), monthi.getString("PHONGTHI"),
monthi.getString("TINHTRANG"));
                    addMonThi(mt);
            }
            }catch (JSONException e) {
                    // TODO Auto-generated catch block
                    e.printStackTrace();
            }
    }
```

Lịch thi được truyền vào dưới dạng mảng JSON, sau đó được bóc tách và tổ chức lại và truyền vào hàm addMonThi() để lưu vào cơ sở dữ liệu. Trong Android sử dụng cơ sở dữ liệu SQLite.

```
public void addMonThi(MonThi mon){
SQLiteDatabase db = this.getWritableDatabase();

            ContentValues values = new ContentValues();
            values.put(KEY_MAMON, mon.getMaMon());
            values.put(KEY_TENMON, mon.getTenMon());
            values.put(KEY_NGAYTHI,mon.getNgayThi());
            values.put(KEY_CATHI, mon.getCaThi());
            values.put(KEY_PHONG, mon.getPhongThi());
            values.put(KEY_TT, mon.getTinhTrang());
            Log.i("Insert mon thi",mon.getTenMon() + " ; " + mon.getNgayThi());
            db.insert(TABLE_SCHEDULE, null, values);
            db.close();
    }
```

5.6. Service trong Android

Công nghệ Push Notification cho phép những thông điệp mới được cập nhật gần như ngay tức thì và đảm bảo tính thời gian thực, có sự thay đổi sẽ có thông báo ngay tức thì. Nhưng để có được sự "tức thì" đó vẫn cần phải có một nguồn yêu cầu thông tin.

Thông thường để thông tin được cập nhật mới thì hệ thống sẽ phải liên tục kiểm tra tính mới của thông tin. Do vậy hệ thống cần liên tục kiểm tra, gửi yêu cầu. Khoảng thời gian gửi yêu cầu kiểm tra giữa hai lần gửi càng nhỏ thì tính cập nhật mới của dữ liệu càng tốt. Để giải quyết vấn đề này, chúng tôi sử dụng Service, một thành phần của ứng dụng Android.

Trong các phần trước đã trình bày về lý thuyết cơ bản của Service trong Andoird. Trong phần này, chúng tôi sẽ tập trung vào kỹ thuật ứng dụng Service vào giải quyết bài toán.

Xây dựng lớp LocalService

Lớp LocalService được kế thừa từ lớp Service có sẵn của Android.

Một Service được khởi tạo bằng hàm onCreate() được bắt đầu bằng onStart() hoặc onStartCommand(). Ở đây chúng tôi sử dụng hàm onStartCommand() để có thể sử dụng giá trị trả về START_STICKY, giúp service không bị ngừng khi ứng dụng bị "killed".

```
public int onStartCommand(Intent intent, int flags, int startId) {
    Toast.makeText(this, "My Service Started", Toast.LENGTH_LONG).show();
    Log.d("Service", "onStart");
    if(session.getTimeRequest() < 0) stopMyThread();
    else startMyThread();
return START_STICKY;
}
```

Service sẽ tạo ra một tiến trình chạy ngầm mà không ảnh hưởng vào tiến trình chính. Cho dùng tiến trình chính bị dừng thì service vẫn có thể tiếp tục.

Tận dụng ưu điểm này, chúng tôi dùng service để có một tiến trình gửi yêu cầu liên tục về phía máy chủ, tiến trình này sẽ không bị gián đoạn hay ảnh hưởng khi đóng ứng dụng do vậy đảm bảo quá trình gửi yêu cầu diễn ra đều đặn và liên tục.

Để tạo ra được chu trình liên tục chúng tôi sử dụng luồng (thread) để có thể làm trễ quá trình lặp.

```
public void startMyThread(){
    timeRequest = session.getTimeRequest();
    if(timeRequest > 0){
          Log.i("Time Request:",String.valueOf(timeRequest));
          if(stopThread) stopThread = false;
          if(myThread == null && !session.getThreadState()){
                myThread = new Thread(){
                      @Override
                      public void run(){
                            while(true && timeRequest > 0)
                           {
                                 try {
                                  if (cd.isConnectingToInternet()){
                                        Log.i("Server","Thread is running");
//                                      getSchedule();
```

```
                                        final HashMap<String, String> user =
session.getUserDetails();
//
   ServerUtilities.getScheduleExam(getApplicationContext(),
user.get(SessionManager.KEY_REGID),user.get(SessionManager.KEY_VERSION));
                                        Thread.sleep(1000*timeRequest);
                                    }
                                   }
                                   catch (InterruptedException e) {
                                       // TODO Auto-generated catch block
                                       e.printStackTrace();
                                   }
                                   if(!session.getThreadState()){
                                    myThread = null;
                                    Log.i("Server","Stoped Thread");
                                    break;
                                   }
                               }
                           }
                   };
                   myThread.start();
                   session.setThreadIsRunning();
                   Log.i("Thread
State",String.valueOf(session.getThreadState()));
           }
   }
   else return;
}
```

Luồng *myThread* được tạo ra và lặp lại với độ trễ tùy chọn của người dùng.

```
Thread.sleep(1000*timeRequest);
```

Và chỉ dừng lại khi có điều kiện tác động

```
if(!session.getThreadState()){
myThread = null;
   Log.i("Server","Stoped Thread");
   break;
}
```

Kết nối và điều khiển Service

Tạo kết nối đến Service:

```
private ServiceConnection mConnection = new ServiceConnection() {
        public void onServiceConnected(ComponentName className, IBinder
service) {
            mBoundService = ((LocalService.LocalBinder)service).getService();
        }

        public void onServiceDisconnected(ComponentName className){
            mBoundService = null;
        }
};
```

Khởi động Service:

```
MainActivity.this.startService(startServiceIntent);
```

5.7. Cài đặt hẹn giờ thông báo

Để phục vụ cho việc thông báo lịch thi đến người dùng, nhóm tác giả sử dụng lớp AlarmManager. AlarmManager có quyền truy cập vào các dịch vụ hệ thống báo động. Với sự giúp đỡ của AlarmManager bạn có thể lên lịch thực thi một đoạn code nào đó trong tương lai. Đối tượng AlarmManager không thể khởi tạo trực tiếp tuy nhiên nó có thể được lấy bằng cách gọi Context.getSystemService (Context.ALARM_SERVICE).

```
AlarmManager mgr =
(AlarmManager)MainActivity.this.getSystemService(Context.ALARM_SERVICE);
```

AlarmManager luôn đăng ký với Intent. Khi đến thời gian báo động, các Intent đã được đăng ký với AlarmManager, được phát sóng bởi hệ thống tự động. Intent này sẽ khởi động ứng dụng nếu ứng dụng đó không chạy. Ứng dụng được khuyên để sử dụng AlarmManager khi bạn muốn code của phần mềm chạy được trong 1 thời gian riêng biệt, thậm chí cả khi phần mềm đó hiện đang không hoạt động.

```
PendingIntent pi =
PendingIntent.getBroadcast(MainActivity.this,notificationCount, i,
PendingIntent.FLAG_UPDATE_CURRENT);
mgr.set(AlarmManager.RTC_WAKEUP,when, pi);
```

Ở đây, "when" chính là khoảng thời gian chờ cho đến khi AlarmManager được gọi, kèm với đó là 1 PendingIntent khởi tạo 1 Broadcast kèm theo Intent chứa dữ liệu cần gửi.

```
Intent i = new Intent(MainActivity.this, Receiver.class);
                i.putExtra("tenMon", tenMon);
                i.putExtra("NotifyCount", notificationCount);
```

Gói dữ liệu bao gồm: tên môn thi và notificationCount. NotificationCount làm nhiệm vụ đếm các môn thi có thể được thông báo. Khi người dùng có nhiều hơn 1 môn sắp đến ngày thi. Mỗi môn sẽ tương ứng với 1 thông báo riêng biệt.

Để người sử dụng nhận biết thông báo được hiển thị trên thiết bị, tác giả sử dụng lớp NotificationManager trong Broadcast. Thông báo có thể có các hình thức khác nhau:

– Một biểu tượng mà đi trong thanh trạng thái và có thể truy cập thông qua các thao tác (khi người dùng chọn nó , một Intent được chỉ định có thể được gọi ra);

– Bật hoặc nhấp nháy đèn LED trên thiết bị;

– Cảnh báo người sử dụng bằng cách nhấp nháy đèn nền , chơi một âm thanh, hoặc rung.

Nó được khai báo như sau:

```
NotificationManager mNotification = (NotificationManager)
context.getSystemService(context.NOTIFICATION_SERVICE);
String tenMon = intent.getStringExtra("tenMon");
String mess ="Sắp đến ngày thi!";
Notification notification = new Notification(R.drawable.ic_launcher, mess,
System.currentTimeMillis());
notification.setLatestEventInfo(context, "Bạn sắp thi môn:", tenMon,
contentIntent);
mNotification.notify(intent.getIntExtra("NotifyCount",0), notification);
```

Khi đến thời gian đã hẹn trước, 1 thông báo về môn thi sắp diễn ra sẽ được hiển thị lên thanh thông báo của thiết bị gồm:

– tenMon: tên của môn thi sắp diễn ra;

– mess: tiêu đề của thông báo đó.

5.8. Cài đặt thời gian nhắc lại thông báo

Ứng dụng cho phép người dùng tuỳ chọn thông báo đó có thể được lặp đi lặp lại theo 1 khoảng thời gian định sẵn hoặc thông báo đó chỉ được hiển thị 1 lần. Khi người dùng thiết lập thông báo được nhắc lại, tương tự như với việc thông báo được hiển thị 1 lần, ta sẽ sử dụng lớp AlarmManager:

```
AlarmManager mgr =
(AlarmManager) MainActivity.this.getSystemService(Context.ALARM_SERVICE);
PendingIntent pi =
PendingIntent.getBroadcast(MainActivity.this, notificationCount, i,
PendingIntent.FLAG_UPDATE_CURRENT);
mgr.setRepeating(AlarmManager.RTC_WAKEUP, when, session.getTimeRepeat(), pi);
```

Ở đây để thiết lập thông báo được lặp đi lặp lại theo 1 khoảng thời gian định sẵn ta dùng hàm setRepeating (int type, long triggerAtMillis, long intervalMillis, PendingIntent operation) với các thông số:

- Type: kiểu thông báo được thiết lập;
- TriggerAtMillis: thời gian thông báo bắt đầu hiển thị (tính theo mili giây);
- IntervalMillis: khoảng cách giữa mỗi lần thông báo được lặp lại;
- Operation: hành động được gọi khi thông báo xuất hiện.

KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

1. Kết luận

Công nghệ di động hiện nay đang phát triển với tốc độ nhanh, nhiều công nghệ mới được ứng dụng, đặc biệt là sự phát triển của điện thoại thông minh cũng như nền tảng di động dần dần càng ngày càng trở nên phổ biến.

Việc phát triển và ứng dụng các công nghệ di động vào đời sống hiện nay đang là xu thế phát triển mạnh của công nghệ thông tin nói chung và của những nhà phát triển phần mềm nói riêng. Do vậy, đề tài này đã cho chúng tôi cơ hội được tìm tòi, nghiên cứu, ứng dụng những công nghệ di động mới nhất, qua đó trao dồi được kinh nghiệm và kỹ năng trong việc phát triển phần mềm, phù hợp với quá trình học tập tại trường Đại học Thăng Long.

Việc thực hiện đề tài này cũng là một tài liệu để cung cấp cho sinh viên những khóa tiếp theo có thể tham khảo, nghiên cứu, phục vụ tốt hơn nữa trong việc học tập tại trường Đại học Thăng Long.

2. Hướng phát triển

Với khả năng truyền dữ liệu nhanh và chính xác, kịp thời của công nghệ Push Notification, ngoài ứng dụng cập nhật lịch thi cho sinh viên, công nghệ này còn mở ra nhiều hướng phát triển ứng dụng tiện ích cho sinh viên hơn nữa như:

- Tự động cập nhật các thông báo mới của nhà trường: những thông báo mới nhất của nhà trường có thể được cập nhật nhanh nhất, mới nhất và kịp thời nhất, đặc biệt là những thông báo khẩn;

- Tự động cập nhật các thông báo của các giảng viên: giảng viên của từng lớp có thể đưa ra các thông báo, nhắc nhở cho sinh viên của lớp mình. Việc nhắc nhở trực tiếp được gửi đến chính xác sinh viên của lớp sẽ tạo hiệu quả tốt hơn việc chỉ đăng thông báo trên website nhà trường, giúp sinh viên nắm bắt tốt hơn thông tin từ giảng viên;

- Phát triển ứng dụng thông báo nhắc nhở đến từng sinh viên, từ đó tăng hiệu quả của việc nhắc nhở như nhắc nhở việc đóng học phí, nộp các giấy tờ cần thiết.

TÀI LIỆU THAM KHẢO

1. **Patrick Niemeyer, Jonathan Knudsen**, *Learning Java 2nd Edition*, O'Reilly, 2002.
2. **Reto Meier**, *Professional Android 4 Application Development*, John Wiley & Sons Inc, 2012.
3. **Samisa Abeysinghe**, *RESTful PHP Web Service*, Packt Publishing Ltd, 2008.
4. Tài liệu dành cho nhà phát triển Android của Google tại trang web: http://developer.android.com