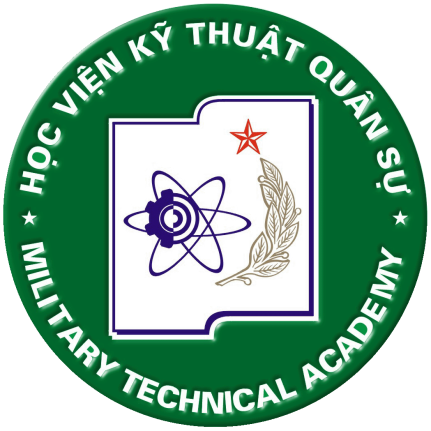
**HỌC VIỆN KỸ THUẬT QUÂN SỰ**

**KHOA CÔNG NGHỆ THÔNG TIN**



**BÁO CÁO KẾT QUẢ**

**HỌC PHẦN THỰC TẬP NHÓM**

**Đề tài: PHỐI HỢP XÂY DỰNG PHẦN MỀM**

**TRÊN GITHUB**

**Thành viên nhóm : Võ Khương Lĩnh (NT)**

**Trần Văn Anh Tuấn**

**Ngô Hồng Quân**

**Nguyễn Quang Trường**

**Khương Huy Hoàng**

**Lê Công Tuấn Anh**

**Hà Nội**

**2015**

**I.TÌM HIỂU VỀ GIT**

**1.Giới thiệu chung**

Git là tên gọi của một hệ thống quản lý phiên bản phân tán (Distributed Version Control System – DVCS) là một trong những hệ thống quản lý phiên bản phổ biến nhất hiện nay. DVCS nghĩa là hệ thống giúp mỗi máy tính có thể lưu trữ nhiều phiên bản khác nhau của một mã nguồn được nhân bản (Clone) từ một kho chứa mã nguồn (Repository), mỗi thay đổi mã nguồn trên máy tính sẽ có thể ủy thác (commit) rồi đưa lên máy chủ nơi đặt kho chứa. Và một máy tính khác (nếu có quyền truy cập) cũng có thể clone lại mã nguồn từ kho chứa hoặc clone lại một tập hợp các thay đổi mới nhất trên máy tính kia.

*Phân biệt Git với Github:*

Git là tên gọi của một mô hình hệ thống còn github chính là một dịch vụ máy chủ repository công cộng, mỗi người có thể tạo tài khoản trên đó để tạo ra các kho chứa của riêng mình để làm việc. Ngoài tính năng nói chung của hệ thống Git, github còn hỗ trợ người dùng những tính năng quan trọng thông qua API sau:

-API to update the repository via HTTP: cho phép người dùng edit file source code trên web browser thông qua HTTP – POST

-API to Access Compare Views: cho phép người dùng review và so sánh code của dự án thông qua việc xem các commit, comments, các dòng khác nhau giữa 2 version của file code… Tính năng này cũng thông qua HTTP – POST, người dùng có thể thực hiện trên web browser.

-Còn nhiều API hữu ích khác mà chúng ta có thể tìm hiểu thêm tại <http://develop.github.com>

Git hỗ trợ 2 giao diện là **GitHub** (giao diện thân thiện với người dùng) và **GitShell** (giao diện code là chủ yếu, với khả năng truy cập sâu vào cơ chế hoạt động của Git hơn).

**2.Cách làm việc của nhóm:**

Khi tạo nhóm , thiết lập và phân quyền tài khoản thì gặp phải vấn đề:

- Nếu muốn phân quyền và dự án private thì phải đóng tiền mua bản quyền của git nên nhóm ko thể làm được.

- Để public và phân quyền tài khoản ngang hàng thì trong quá trình làm việc rất dễ gặp conflict, người này sửa, xóa code của người kia, trùng lặp code…

Chính vì vậy thay vì để public, tạo tài khoản và phân quyền, nhóm chúng tôi đã chọn phương án là chia các branch con để làm việc độc lập nhưng có 1 branch master để đảm bảo việc pull/push là hợp lệ và ít xảy ra conflict nhất và cũng để thuận tiện trong việc sử dụng giao diện GitHub thay cho GitShell

**3.Cài đặt Git**

Chúng ta có thể tải về Git thông qua trang chủ chính thức của Git là “git-scm.com/downloads”. Trong bài báo cáo này, nhóm chúng tôi sử dụng phiên bản Git dành cho hệ điều hành Windows. Khi cài đặt, bộ cài đặt tự động cài cả GitHub lẫn GitShell để tiện cho người sử dụng.

**4.Quy trình làm việc**

Quy trình làm việc là rất quan trọng và cần được rèn luyện nếu người sử dụng muốn thành thạo trong việc sử dụng Git. Quy trình làm việc của Git được biểu diễn bằng sơ đồ sau:

|  |
| --- |
| Add Commit Push  Index  Remote repository  Local repository  Workspace  Fetch    Pull |

Các thành phần trong quy trình

**Workspace**: trạng thái làm việc hiện tại. Các thao tác làm thay đổi mã nguồn, thêm hoặc xóa tài nguyên, … sẽ được lưu trữ ở đây.

**Index**: trạng thái khi các thay đổi đã được lưu lại (bằng các lệnh add)

**Local repository**: sau khi lưu lại, bạn thực hiện thao tác commit, đưa toàn bộ mã nguồn vào local repository. Mỗi commit sẽ có 1 tag (nhãn) riêng để có thể phục hồi lại bất cứ lúc nào.

**Remote repository**: bằng thao tác push, toàn bộ mã nguồn sẽ được đưa lên remote repository và có thể chia sẻ với các thành viên khác thông qua internet.

**Head:** trỏ đến commit gần nhất

Các thao tác

**Khởi tạo git:** Khai báo tên và địa chỉ email để khi commit , git sẽ nhận diện đây là commit của ai, từ đâu .

Cú pháp: git config --global user.name “tên của tài khoản”

git config --global user.email “địa chỉ email của tài khoản”

**Clone**: khi trên remote sever đã có sẵn thư mục và các file mã nguồn, việc cần làm là lấy các thông tin dữ liệu đó về máy tính cá nhân. Lệnh clone sẽ giúp chúng ta thực hiện việc đó.

*Cú pháp: git clone /đường dẫn nơi muốn clone/repository/*

*Ví dụ: git clone https://github.com/KurtLyndon/TTN-TH248*

**Commit**: sau khi thực hiện xong các thay đổi, chúng ta sẽ đưa các thay đổi này vào index và sau đó là Local repository. Trước khi commit chúng ta sẽ sử dụng lệnh add để cập nhật tình trạng các file vào index.

*Cú pháp:*

*git all –all (tự động thêm index tất cả các tập tin thay đổi)*

*git add <tên tập tin> (thêm vào index của tập tin đó)*

*git add \* (thêm vào index tất cả tập tin thay đổi)*

Sau đó thì commit bằng lệnh

*git commit –m “ghi chú commit”=> Head sẽ trỏ đến commit này*

**Push**: sau khi commit, các thay đổi sẽ được lưu trữ tại Local repository, chúng ta sử dụng lệnh push để đưa toàn bộ dữ liệu lên Remote repository.

*Cú pháp: git push origin branch\_name, trong đó branch\_name là tên của nhánh là việc hiện tại, hoặc có thể là master để đẩy trực tiếp lên repository chủ*

**Pull**: là hành động kéo dữ liệu từ trên Remote repository về máy tính cá nhân. Chúng ta có thể kéo toàn bộ dữ liệu hoặc chỉ định một nhánh cụ thể nào đó sau đó tiến hành trộn (merge).

*Cú pháp: git pull <branch\_name>*

**Remote:** Nếu chưa clone một repository hiện có và muốn kết nối repository của hiện tại đến máy chủ remote.

*Cú pháp: git remote add origin <máy-chủ>*

**Fetch**: khi xảy ra lỗi, nếu ta muốn trở lại commit gần nhất để bắt đầu lại công việc, lệnh fetch sẽ giúp ta thực hiện điều đó. Nó sẽ source mới nhất trên sever về và đè lên source hiện tại trên máy local.

*Cú pháp: git fetch <branch\_name>*

**5.Cách tạo tài khoản, phân quyền tài khoản.**

*Tạo tài khoản:* Để đăng kí tài khoản, chúng ta truy cập vào <http://github.com>, sau đó tiến hành đăng kí tài khoản giống như việc đăng kí tài khoản trên các diễn đàn, website khác, chỉ có điều ở khâu cuối cùng hệ thống có yêu cầu lựa chọn hình thức, mức phí thanh toán thì tùy vào tài chính của cá nhân để lựa chọn, với mỗi loại hình, mức phí thanh toán khác nhau chúng ta sẽ được cấp quyền sử dụng khác nhau.

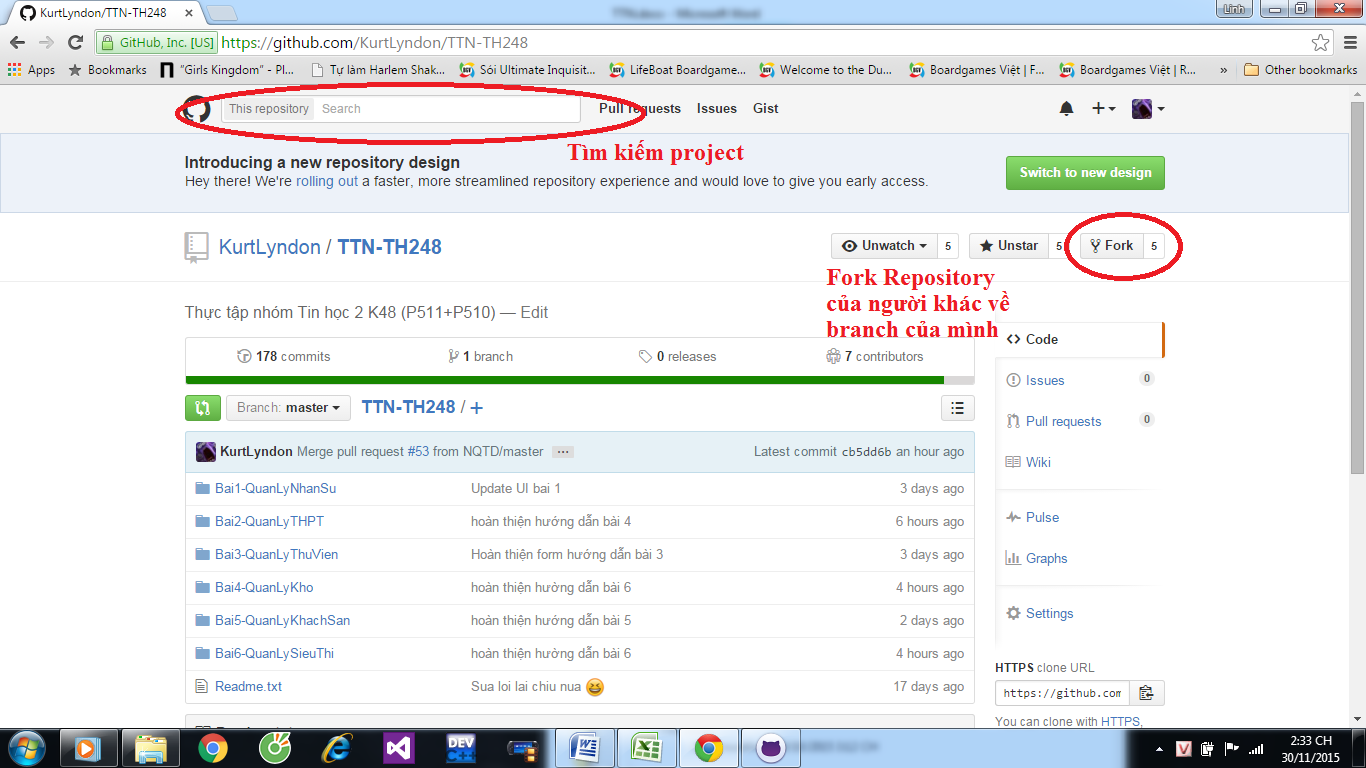
*Phân quyền tài khoản*: sau khi tạo tài khoản thành công, chúng ta có thể tạo một Team (nhóm) bằng cách vào mục Profile của mình chọn mục Organizations và tiến hành tạo Team mới bằng cách chọn mục New Team sau đó điền đầy đủ thông tin về team và chọn cách hiển thị nội dung của team. Sau đó tiến hành mời cách tài khoản khác tham gia vào nhóm của mình và tiến hành phân quyền cho các thành viên trong nhóm.

**6.Cách sử dụng giao diện GitHub.**

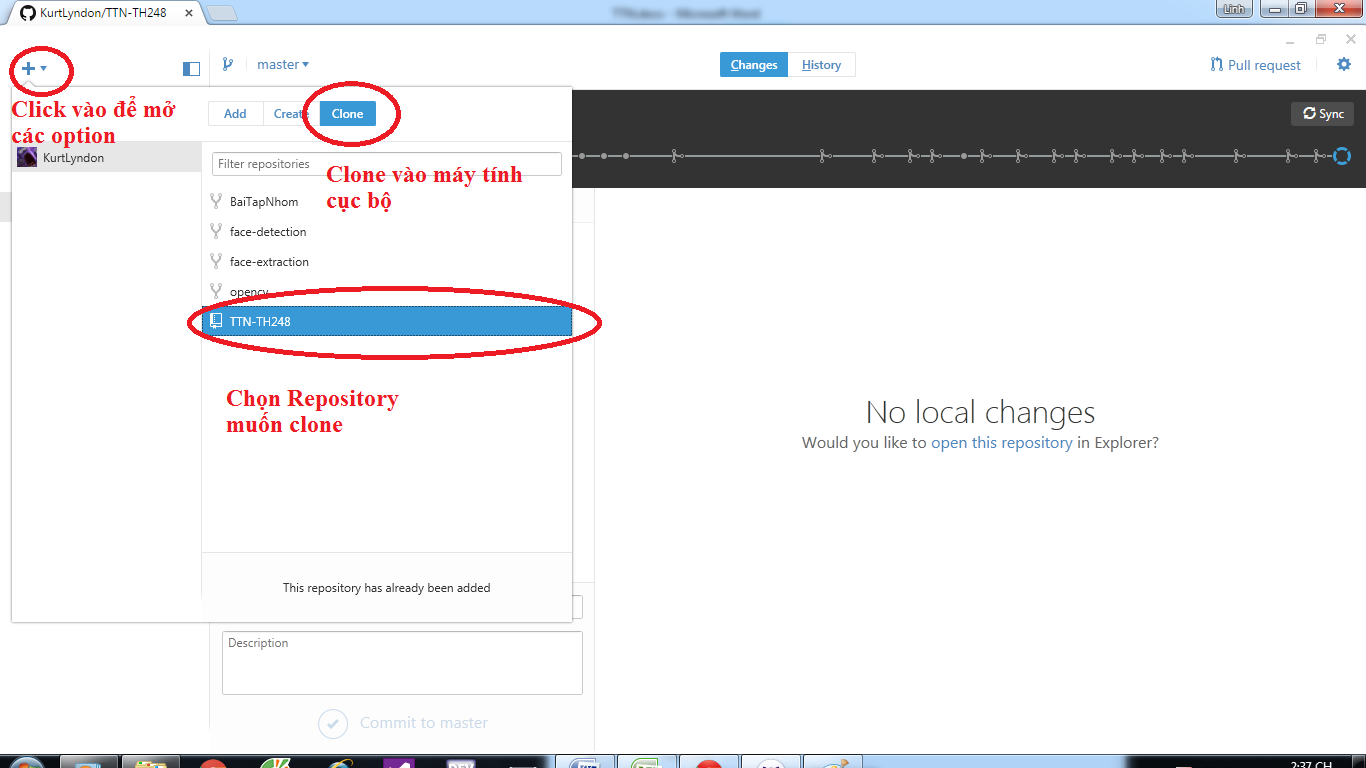
Yêu cầu : - phải có tài khoản trên trang chủ đã đăng ký và đã xác thực trên trang chủ github.com. Và có cài đặt GitHub trên máy tính cá nhân.

***a/Cách tham gia contribute cho 1 dự án nào đó:***

Vào trang GitHub.com, sau đó tìm dự án rồi Fork dự án đó về với Repository của mình

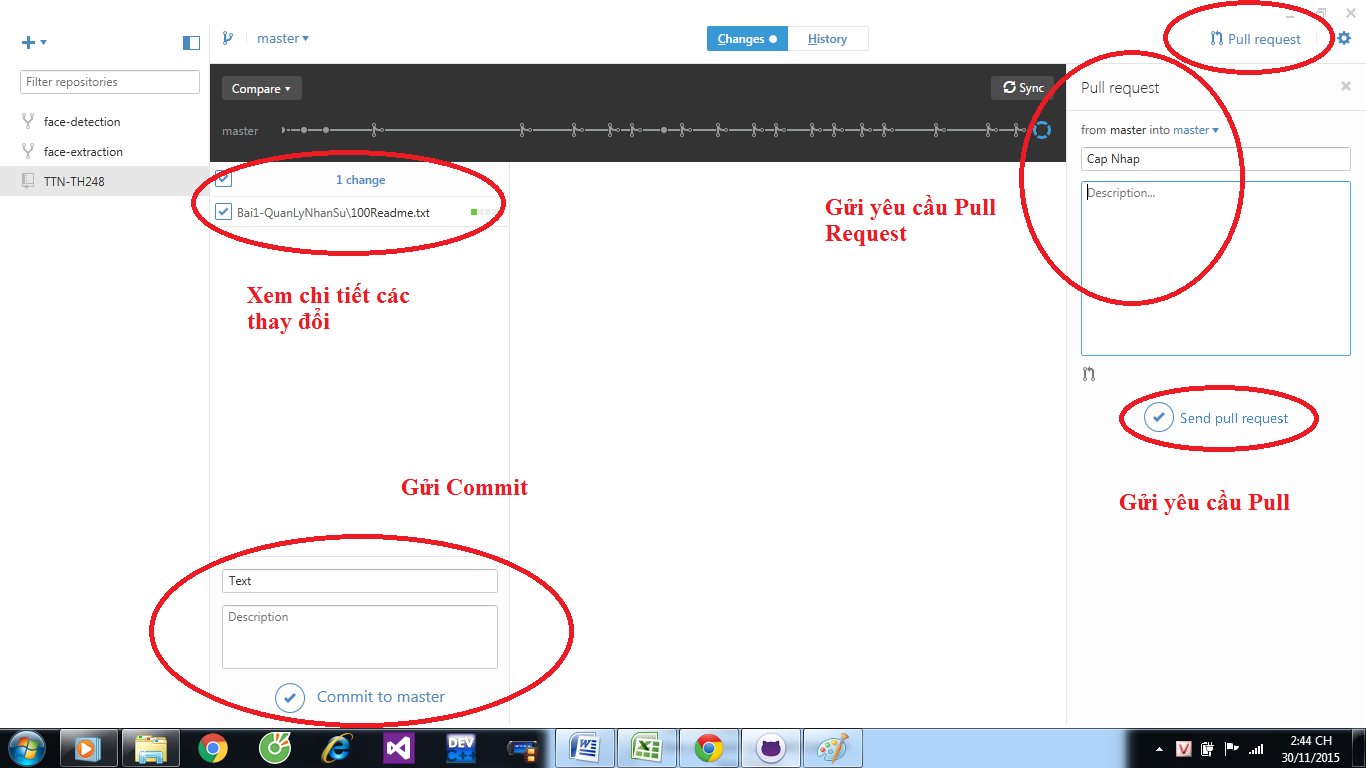


***b/Clone vào máy tính cục bộ cá nhân***



***c/Commit và Pull Request với máy chủ***

Commit để xác nhận các thay đổi của repository ở máy cục bộ lên trên repository ở trên mạng. Pull request để đưa ra yêu cầu Merge branch của mình với branch của người quản lý.



***d/Merge các pull request của các Contributor***

Ở trang remote của máy chủ, vào phần pull requests để xem các pull requites và quyết định xem có cho phép merge các branch làm việc của người gửi yêu cầu đến hay ko

****

***e/Giải quyết các Conflict***

Khi 2 hay nhiều người cùng thay đổi cùng 1 dòng code thì khi commit sẽ gặp conflict

Trường hợp người thứ nhất thay đổi 1 dòng code nào đó và đã Commit/Push(GitShell) hay Commit/PullRequest/Merge(GitHub) thành công, người thứ 2 chưa cập nhật mà lại thay đổi code khác dòng code người thứ nhất đã thay đổi, thì người thứ 2 phải giải quyết conflict:

\*Dùng GitShell:

- dùng lệnh Fetch lại repository của mình rồi từng người commit lại code của mình

- dùng lệnh Pull để check out (cập nhật mới repository của mình) rồi lại Commit lên và Push mới

\*Dùng GitHub:

-Người thứ 2 phải sao lưu backup các thay đổi của mình

-Sau đó phải Sync (đồng bộ) với repository của người QL

-Sau đó Commit/Pull request như bình thường.

Trường hợp cả 2 người (đã check out phiên bản mới nhất của repository của người quản lý) cùng thay đổi một dòng code nào đó:

\*Dùng GitShell:

-Khi dùng gitshell thì lệnh Commit/Push trong trường hợp này sẽ được Gitshell chặn lại để tránh nguy hiểm cho dự án, và đòi hỏi người sử dụng sẽ phải check out dữ liệu mới và xem khi nào người còn lại đã push thành công thì người này mới có thể push hay commit lên được và lúc này thi GitShell sẽ đánh dấu phần code có sự thay đổi của người đầu tiên để người thứ 2 biết và quyết định

\*Dùng GitHub:

-Khi gặp trường hợp này , một là cũng làm tương tự như đã làm với cách ở GitShell, hoặc dùng cách khác thứ 2:

-Cách thứ 2: GitHub hỗ trợ công cụ Merge-Tracking Branch để tự động sinh code của cả 2 người và cùng Pull Request đến branch người quản lý với cả 2 code của người đó với ghi chú (phần nào thêm, phần nào xóa, phần nào thay đổi của cả 2 người) để người quản lý quyết định giữ và bỏ phần code nào hoặc quyết định phần code nào sẽ được lưu vào Remote Repository của dự án

**II.TỔNG QUAN VỀ CÔNG VIỆC**

**1. Giới thiệu:**

*1.1.Công cụ và môi trường thiết kế và xây dựng phần mềm:*

-SQL server 2012, SQL server 2008 R2, ngôn ngữ T-SQL

-Visual Studio 2012, môi trường dotNet, ngôn ngữ C#, HTML

-Môi trường cộng tác làm việc nhóm: Git với công cụ GitHub/GitShell

*1.2.Link đường dẫn đến dự án trên GitHub* <https://github.com/KurtLyndon/TTN-TH248>

Trong đó có chứa:

- File BaoCao.docx : bản báo cáo thực tập nhóm

- File BangPhanCong.xls : bảng phân công chi tiết các công việc cần làm

- File Readme.txt : để làm nơi ghi chú cho các thành viên

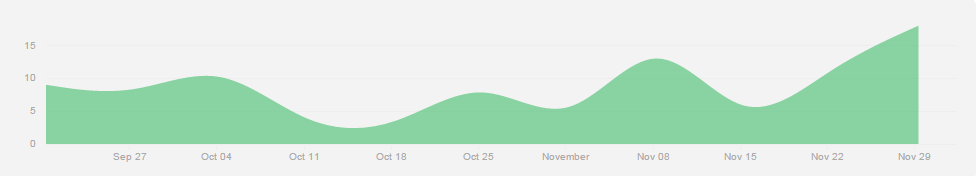
- 6 Folder chứa các dự án cần phải làm

**2. Bảng phân công công việc chủ yếu**

|  |  |  |
| --- | --- | --- |
| STT | Họ và Tên | Công việc chủ yếu |
| 1 | Võ Khương Lĩnh | -Phân công và tổng hợp công việc  -Đưa ra tiêu chuẩn, mục tiêu, bản thảo yêu cầu  -Xây dựng các module  -Hiệu chỉnh sản phẩm đầu ra  -Acceptance/System/Intergrate/Unit Tester |
| 2 | Trần Văn Anh Tuấn | -Thiết kế giao diện  -Xây dựng giao diện tổng  -Thiết kế các module  -System/UI Tester  -Sửa lỗi UI |
| 3 | Ngô Hồng Quân | -Thiết kế các function  -Chỉnh sửa cơ sở dữ liệu  -Xây dựng các module  -System/Intergrate/Unit Tester  -Sửa lỗi Code/Function |
| 4 | Nguyễn Quang Trường | -Thiết kế và xây dựng các document  -Điều chỉnh nội dung các module  -System/Intergrate/UI Tester  -Sửa lỗi Code/UI/System |
| 5 | Khương Huy Hoàng | -Thiết kế cơ sở dữ liệu  -Thiết kế proceduce  -Xây dựng các module  -Acceptance/System Tester |
| 6 | Lê Công Tuấn Anh | -Xây dựng cơ sở dữ liệu  -Xây dựng proceduce  -Xây dựng các module  -Acceptance/System Tester |

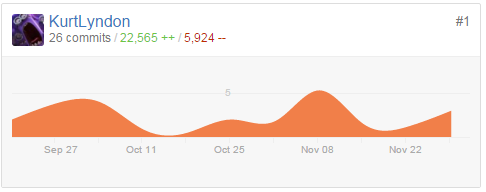
**3.Biểu đồ**

***a.Biểu đồ tổng thể của toàn bộ dự án***

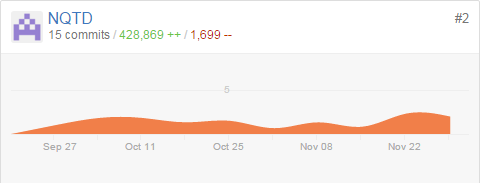


***b.Biểu đồ commit của từng thành viên***

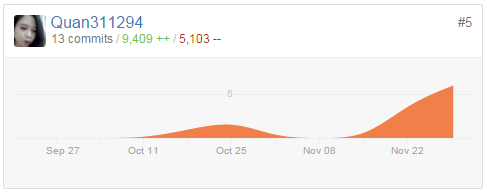
1.Võ Khương Lĩnh



2.Nguyễn Quang Trường



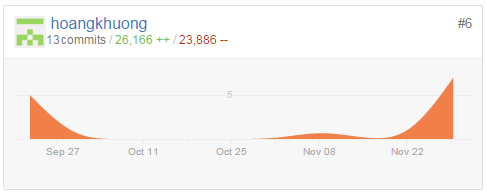
3.Ngô Hồng Quân



4.Trần Văn Anh Tuấn



5.Khương Huy Hoàng



6.Lê Công Tuấn Anh

