**Các thư viện cần cài đặt**

pip install django

pip install django-modeladmin-reorder

pip install python-dotenv

**Chạy môi trường ảo:**

python -m venv venv

.\venv\Scripts\activate

**Lệnh chạy server:**

Python manage.py runserver

**Link truy cập:**

<http://127.0.0.1:8000/>

Tài khoản: admin

**Tích hợp**: boostrap, AJAX

**Extension cho VSCode:**

**Prettier - Code formatter:** Để tự động làm đẹp và sắp xếp code.

**ESLint:** Để bắt lỗi JavaScript.

**djLint:** Để bắt lỗi trong file HTML template của Django.

<https://git-scm.com/downloads/win>

Personal Access Token (PAT) của GIT:   
ghp\_j705r3Ff5q6zQNz0jzYiQ2ynDtyZe80XFanF

#### Cơ hội Cải tiến (Gợi ý cho tương lai)

Dự án của bạn đã rất hoàn chỉnh. Tuy nhiên, nếu bạn muốn phát triển nó lên một tầm cao mới, đây là một vài gợi ý nhỏ:

* **Viết Tests:** File tests.py hiện đang trống. Viết các bài kiểm thử (unit test) cho các logic quan trọng (ví dụ: hàm tính điểm, hàm tạo snapshot, hàm xử lý file Excel) sẽ giúp bạn tự tin hơn khi sửa đổi hoặc thêm tính năng mới mà không làm hỏng các chức năng cũ.
* **Tách biệt JavaScript:** Với các trang có logic JavaScript phức tạp như take\_quiz.html hay topic\_group\_list.html, bạn có thể cân nhắc chuyển mã JavaScript từ thẻ <script> trong file .html ra các file .js riêng và đặt trong thư mục static. Điều này giúp mã nguồn gọn gàng hơn và cho phép trình duyệt cache các file script, tăng tốc độ tải trang cho những lần truy cập sau.

# QUY TRÌNH LÀM VIỆC CHUẨN TRÊN 2 MÁY VỚI GITHUB

### ****Quy tắc vàng và Sơ đồ quy trình****

Quy tắc vàng ở đây là: **GitHub luôn là "nguồn chân lý" (single source of truth)**. Mọi thay đổi đều phải đi qua GitHub. Bạn không bao giờ đồng bộ trực tiếp giữa hai máy tính.

Sơ đồ quy trình sẽ trông như thế này:

[Máy ở nhà] <--- git pull --- **[GitHub]** --- git push ---> [Máy ở nhà]

[Máy cơ quan] <--- git pull --- **[GitHub]** --- git push ---> [Máy cơ quan]

Như bạn thấy, cả hai máy đều chỉ "nói chuyện" với GitHub.

### ****Quy trình làm việc chi tiết****

Đây là chu trình làm việc bạn sẽ lặp lại mỗi ngày.

#### ****1. Khi Bắt đầu một buổi làm việc (trên bất kỳ máy nào)****

Giả sử bạn vừa đến cơ quan và mở máy tính làm việc. Việc đầu tiên bạn cần làm là cập nhật code trên máy cơ quan để nó có tất cả những gì bạn đã làm ở nhà vào tối hôm trước.

* **Mục tiêu:** Đảm bảo bạn đang làm việc trên phiên bản code mới nhất.
* **Hành động:** Mở terminal, di chuyển vào thư mục dự án và chạy lệnh:

Bash

git pull origin main

* **Lệnh này làm gì?** Nó sẽ "kéo" (pull) tất cả các thay đổi mới nhất từ GitHub về máy tính của bạn và hợp nhất chúng vào code hiện tại.
* **Lời khuyên:** Hãy tạo thói quen chạy lệnh này **luôn luôn** trước khi bạn bắt đầu viết code mới.

#### ****2. Khi Kết thúc một buổi làm việc (trên bất kỳ máy nào)****

Giả sử bạn đã làm việc xong ở cơ quan và chuẩn bị về nhà. Bạn cần lưu lại tất cả những thay đổi đã làm và đưa chúng lên GitHub.

* **Mục tiêu:** Lưu trữ an toàn các thay đổi của bạn và sẵn sàng để tiếp tục làm việc ở nhà.
* **Hành động:** Trong terminal, chạy lần lượt 3 lệnh sau:
  1. **Thêm các file đã thay đổi:**

Bash

git add .

* 1. **Ghi nhận các thay đổi với một tin nhắn mô tả:**

Bash

git commit -m "Hoàn thành chức năng báo cáo chi tiết"

* 1. **Đẩy các thay đổi lên GitHub:**

Bash

git push origin main

Sau khi chạy xong 3 lệnh này, toàn bộ công việc của bạn đã được lưu trữ an toàn trên GitHub. Bây giờ, khi về nhà, bạn chỉ cần lặp lại quy trình ở phần 1: mở máy tính ở nhà và chạy git pull origin main để lấy về tất cả những gì bạn vừa làm ở cơ quan.

### ****Trường hợp cần lưu ý: Xung đột (Conflict)****

* **Điều gì xảy ra nếu bạn quên git pull trước khi sửa code?**
  + Nếu bạn sửa cùng một file, cùng một dòng trên cả hai máy mà không đồng bộ, khi bạn cố gắng push hoặc pull, Git có thể sẽ báo lỗi "merge conflict" (xung đột khi hợp nhất).
* **Phải làm gì?**
  + Đây không phải là lỗi nghiêm trọng mà là một tính năng an toàn của Git để báo cho bạn biết rằng nó không biết phải chọn phiên bản nào.
  + VS Code có các công cụ tích hợp rất tốt để giúp bạn so sánh và chọn phiên bản đúng của từng đoạn code bị xung đột.
* **Cách tốt nhất để tránh xung đột:** Luôn nhớ quy tắc vàng: **Pull trước khi code, Push sau khi xong.**

# QUY TRÌNH ĐỒNG BỘ DỮ LIỆU TRÊN WEB

Quy tắc vàng mà bạn luôn nên tuân theo là: **Luôn luôn sửa code ở máy tính cá nhân, sau đó đẩy lên GitHub, và cuối cùng là cập nhật server từ GitHub.**

Tuyệt đối không nên sửa code trực tiếp trên PythonAnywhere, vì điều đó sẽ làm cho mã nguồn của bạn ở 3 nơi không còn đồng bộ và rất khó quản lý.

### ****Quy trình chuẩn để sửa lỗi hoặc thêm tính năng****

Đây là các bước chi tiết cho quy trình làm việc của bạn từ bây giờ:

#### ****Bước 1: Sửa lỗi và Kiểm tra trên máy tính cá nhân (Local)****

* **Sửa code ở đâu?** Luôn luôn là trên máy tính của bạn, sử dụng VS Code. Đây là môi trường phát triển chính của bạn.
* **Thực hiện thay đổi:** Sửa lỗi hoặc viết thêm tính năng mới như bình thường.
* **Kiểm tra kỹ lưỡng:** Sau khi sửa xong, hãy chạy server ở máy tính của bạn (python manage.py runserver) và kiểm tra kỹ lưỡng để đảm bảo lỗi đã được sửa và không có lỗi mới nào phát sinh.

#### ****Bước 2: Ghi nhận thay đổi bằng Git (Commit)****

Khi bạn đã chắc chắn các thay đổi hoạt động tốt ở local, bạn sẽ ghi nhận chúng bằng Git.

* **Hành động:** Mở terminal trên máy tính, di chuyển vào thư mục dự án và chạy các lệnh sau:
  1. Thêm các file đã thay đổi vào Git:

Bash

git add .

(Lệnh này sẽ thêm tất cả các file đã thay đổi. Hoặc bạn có thể chỉ định file cụ thể: *git add quiz\_app/views.py*)

* 1. Tạo một "bản ghi" (commit) với một tin nhắn mô tả rõ ràng về những gì bạn đã làm:

Bash

git commit -m "Sửa lỗi không hiển thị modal xóa đề thi"

#### ****Bước 3: Đẩy mã nguồn lên GitHub****

Bây giờ, bạn sẽ đồng bộ các thay đổi từ máy tính của mình lên kho chứa trung tâm trên GitHub.

* **Hành động:** Trong terminal, chạy lệnh:

Bash

git push origin main

#### ****Bước 4: Cập nhật Server trên PythonAnywhere (Deploy)****

Đây là bước cuối cùng, lấy phiên bản code mới nhất từ GitHub về server của bạn.

* **Hành động:**
  1. Đăng nhập vào PythonAnywhere và mở một **Bash console**.
  2. Kích hoạt môi trường ảo của bạn:

Bash

workon quizapp-env

* 1. Di chuyển vào thư mục dự án trên PythonAnywhere:

Bash

cd ten-repository-cua-ban

* 1. **Kéo (pull)** các thay đổi mới nhất từ GitHub về:

Bash

git pull origin main

* 1. **(Nếu cần) Chạy các lệnh Django:**
     + Nếu bạn có thay đổi trong models.py, hãy chạy: python manage.py migrate
     + Nếu bạn có thay đổi file CSS/JS, hãy chạy: python manage.py collectstatic --noinput
     + (Nếu chỉ sửa file *views.py* như lỗi vừa rồi, bạn không cần chạy hai lệnh này).
  2. **Reload Web App:** Vào tab **"Web"** trên PythonAnywhere và bấm nút **"Reload..."** màu xanh lá.

### ****Sơ đồ quy trình:****

[Máy tính cá nhân (Sửa code & Test)] ---> git push ---> [GitHub (Kho chứa code)] ---> git pull ---> [PythonAnywhere (Server)]

**Tại sao quy trình này lại quan trọng?**

* **An toàn:** Bạn luôn có một phiên bản hoạt động tốt trên GitHub để quay lại nếu có sự cố.
* **Nhất quán:** Mã nguồn ở cả 3 nơi luôn được đồng bộ một cách có kiểm soát.
* **Chuyên nghiệp:** Đây là quy trình làm việc tiêu chuẩn trong ngành phát triển phần mềm.

### Hãy tập làm quen với quy trình này, nó sẽ trở thành một thói quen tốt và giúp dự án của bạn phát triển bền vững.**QUY TRÌNH ĐỂ RESET CODE CỤC BỘ**

Bạn hãy mở terminal, di chuyển vào thư mục dự án trên máy tính đang có lỗi và chạy lần lượt các lệnh sau:

#### ****Bước 1: Lấy về thông tin mới nhất từ GitHub****

Lệnh này sẽ tải về "lịch sử" mới nhất từ GitHub nhưng chưa áp dụng bất kỳ thay đổi nào vào code của bạn.

Bash

git fetch origin

#### ****Bước 2: Reset toàn bộ code cục bộ về giống hệt GitHub****

Đây là lệnh chính. Nó sẽ **buộc** thư mục làm việc của bạn trở thành một bản sao chính xác của phiên bản code đang có trên nhánh main của GitHub, hủy bỏ mọi thay đổi cục bộ của bạn trên các file đã được Git theo dõi.

Bash

git reset --hard origin/main

#### ****Bước 3 (Tùy chọn): Dọn dẹp các file mới chưa được theo dõi****

Lệnh trên chỉ ảnh hưởng đến các file đã được Git theo dõi. Nếu bạn đã tạo ra các file mới hoàn toàn ở local (file "untracked"), chúng vẫn sẽ còn đó. Lệnh sau sẽ xóa chúng đi.

**Cảnh báo: Lệnh này cũng sẽ xóa vĩnh viễn các file không thể khôi phục.**

Bash

git clean -f -d

* -f là viết tắt của --force (bắt buộc).
* -d để xóa cả các thư mục mới chưa được theo dõi.

**QUY TRÌNH TẠO VÀ CÀI FILE REQUIREMENTS.TXT**

1. **Trên máy tính cá nhân của bạn**, trong thư mục dự án và đã kích hoạt môi trường ảo, hãy chạy lệnh sau để tạo file:

Bash

pip freeze > requirements.txt

1. **Đưa file requirements.txt này lên GitHub** (git add requirements.txt, git commit, git push).
2. **Trên PythonAnywhere**, sau khi git pull, bạn chỉ cần chạy một lệnh duy nhất trong console của virtualenv:

Bash

pip install -r requirements.txt