**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)

**A. Cải thiện Bố cục và Luồng làm việc**

1. **Thiết kế lại Trang Bảng điều khiển (Dashboard Redesign):**
   * **Hiện tại:** Trang Dashboard khá đơn giản, chỉ hiển thị 4 con số thống kê.
   * **Ý tưởng:** Biến nó thành một "trung tâm điều khiển" thực sự. Chúng ta có thể thêm các mục như:
     + Một danh sách "Các bài thi đã làm gần đây".
     + Một danh sách "Các đề thi của bạn được làm nhiều nhất" (dành cho Creators).
     + Một form nhỏ để nhập mã tham gia thi ngay tại dashboard.
   * **Lợi ích:** Làm cho Dashboard trở nên hữu ích và có giá trị hơn, là điểm khởi đầu tốt hơn cho người dùng mỗi khi họ đăng nhập.
2. **Cải thiện "Trạng thái rỗng" (Better Empty States):**
   * **Hiện tại:** Khi người dùng mới đăng ký, các trang danh sách (Quản lý nội dung, Lịch sử,...) chỉ hiển thị một dòng chữ thông báo trống.
   * **Ý tưởng:** Thay thế các dòng chữ đơn điệu bằng các khối (panel) được thiết kế đẹp mắt, có icon minh họa và một nút kêu gọi hành động (Call-to-Action) rõ ràng. Ví dụ: *"Chào mừng! Bạn chưa có chủ đề nào cả. Hãy [Tạo Nhóm chủ đề đầu tiên]!"*
   * **Lợi ích:** Hướng dẫn và khuyến khích người dùng mới tương tác với ứng dụng, làm cho ứng dụng có cảm giác hoàn thiện và thân thiện hơn.
3. **Bố cục 2 cột cho Trang Tìm kiếm:**
   * **Hiện tại:** Trang tìm kiếm câu hỏi (question\_search.html) có bộ lọc ở trên và kết quả ở dưới, gây lãng phí không gian hai bên.
   * **Ý tưởng:** Chuyển bộ lọc tìm kiếm sang một cột sidebar bên trái. Kết quả tìm kiếm sẽ hiển thị ở cột lớn bên phải. Sidebar này có thể ẩn/hiện trên các thiết bị di động.
   * **Lợi ích:** Tận dụng không gian màn hình tốt hơn, là một bố cục rất quen thuộc và hiệu quả cho các trang có chức năng lọc/tìm kiếm.

**B. Tăng cường Tương tác và Phản hồi**

1. **Cảnh báo chống mất dữ liệu khi rời trang (Confirm on Unload):**
   * **Hiện tại:** Người dùng đang điền một form dài (như tạo câu hỏi) có thể vô tình đóng tab và mất hết dữ liệu đã nhập.
   * **Ý tưởng:** Dùng JavaScript để phát hiện nếu form đã bị thay đổi. Nếu người dùng cố gắng rời đi, trình duyệt sẽ hiện một hộp thoại xác nhận: "Các thay đổi bạn đã thực hiện có thể không được lưu. Bạn có chắc muốn rời đi?"
   * **Lợi ích:** Một tính năng rất chuyên nghiệp, bảo vệ công sức của người dùng và giảm thiểu sự thất vọng.
2. **Hệ thống thông báo "Toast":**
   * **Hiện tại:** Các thông báo (ví dụ: tham gia đề thi thành công) đang dùng chung hệ thống Message của Django, hiển thị ở đầu trang và làm xô lệch bố cục.
   * **Ý tưởng:** Xây dựng một hệ thống thông báo "Toast" nhất quán. Các thông báo ngắn như "Đã sao chép mã!", "Xóa thành công!" sẽ xuất hiện ở góc trên bên phải màn hình và tự biến mất.
   * **Lợi ích:** Cung cấp phản hồi tinh tế, hiện đại và không làm gián đoạn luồng làm việc của người dùng.

**C. Nâng cao Trải nghiệm Cốt lõi**

1. **Bộ lọc "Chỉ xem câu sai" trên trang kết quả:**
   * **Hiện tại:** Trang kết quả (attempt\_result.html) hiển thị tất cả các câu hỏi.
   * **Ý tưởng:** Thêm các nút lọc (ví dụ: "Tất cả", "Đúng", "Sai") ở đầu trang kết quả. Khi người dùng nhấn vào "Sai", trang sẽ chỉ hiển thị danh sách các câu hỏi mà họ đã trả lời sai.
   * **Lợi ích:** Một tính năng cực kỳ hữu ích cho việc học tập và ôn lại kiến thức. Người dùng có thể nhanh chóng tập trung vào những phần họ còn yếu.

# 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