User: username ps name dateofbirth nghề nghiệp description avatar(link) ảnh bìa quê quán nơi sống isOnline

Friend: ID UserID - UserID ( tìm hiểu cấu tạo bảng này)

PersonalPage: UserID PostID

Post: title information userID TimeCreated( tìm hiểu xem cần xây riêng bảng user-post không) likes comments share StatusID

Status: ID statusName (public private only-Friend)

LikePost: PostID UserID(manytomany) time

CommentPost: PostID UserID commentInfor timeCreation

SharePost: PostID UserID time

Message: ID UserID message sendTime BoxID

BoxID: ID userID - userID

( Chức năng chát theo nhóm)

- chức năng đăng bài (dùng cả hình ảnh), chức năng tag @ bạn bè

- hiển thị trang cá nhân

- chức năng tim kiếm bạn bè

- tìm kiếm bài viết( tìm tất cả các bài public post hoặc của bạn bè)

- đăng ký đăng nhập bằng gg

- chỉnh sửa xoá post thay đổi status

- hiển thị ngẫu nhiên post public ở trang chủ khi chưa đăng nhập

Trang internal page

## Kịch bản và Chức năng Chính

Chúng ta sẽ phác thảo các kịch bản người dùng chính và các chức năng tương ứng.

**1. Quản lý người dùng (User Management)**

**Kịch bản:**

* **Đăng ký:** Người dùng mới tạo tài khoản.
* **Đăng nhập/Đăng xuất:** Người dùng truy cập hoặc thoát khỏi ứng dụng.
* **Cập nhật hồ sơ:** Người dùng chỉnh sửa thông tin cá nhân (tên, ảnh đại diện, tiểu sử).
* **Quản lý mật khẩu:** Người dùng thay đổi hoặc khôi phục mật khẩu.

**Chức năng:**

* **Đăng ký tài khoản:** 📧 Email/Username, Mật khẩu, Tên hiển thị.
* **Xác thực tài khoản:** Có thể thông qua email xác nhận để tăng cường bảo mật.
* **Đăng nhập/Đăng xuất:** Sử dụng hệ thống xác thực của Django.
* **Chỉnh sửa hồ sơ cá nhân:** Avatar, Bio, Ngày sinh, Giới tính, Liên hệ.
* **Thay đổi mật khẩu.**
* **Khôi phục mật khẩu:** Qua email.
* **Vô hiệu hóa/Xóa tài khoản (tùy chọn).**

**2. Quản lý bài viết (Post Management)**

**Kịch bản:**

* **Tạo bài viết:** Người dùng đăng tải nội dung.
* **Xem bài viết:** Người dùng xem các bài viết của mình và của người khác (trên bảng tin, trang cá nhân).
* **Chỉnh sửa bài viết:** Người dùng sửa nội dung bài viết của mình.
* **Xóa bài viết:** Người dùng xóa bài viết của mình.
* **Đặt chế độ riêng tư:** Người dùng kiểm soát ai có thể xem bài viết của họ.

**Chức năng:**

* **Tạo bài viết mới:**
  + Nội dung văn bản.
  + Tải lên ảnh/video (nhiều ảnh/video cho một bài viết).
  + Gắn thẻ bạn bè (@mention).
  + Thêm vị trí.
* **Xem bảng tin (News Feed):** Hiển thị các bài viết từ bạn bè và các trang/nhóm mà người dùng theo dõi.
* **Xem bài viết trên trang cá nhân:** Hiển thị tất cả bài viết của một người dùng cụ thể.
* **Chỉnh sửa bài viết:** Chỉ chủ bài viết mới có quyền.
* **Xóa bài viết:** Chỉ chủ bài viết mới có quyền.
* **Chế độ riêng tư cho bài viết:**
  + **Công khai (Public):** Mọi người đều có thể xem.
  + **Bạn bè (Friends Only):** Chỉ bạn bè mới xem được.
  + **Chỉ mình tôi (Only Me):** Chỉ chủ bài viết xem được.
  + **Tùy chỉnh (Custom):** Chọn những người cụ thể hoặc nhóm người được phép xem/không được phép xem.

**3. Tương tác bài viết (Post Interaction)**

**Kịch bản:**

* **Thích/Bỏ thích bài viết:** Người dùng bày tỏ sự quan tâm đến bài viết.
* **Bình luận bài viết:** Người dùng đưa ra ý kiến về bài viết.
* **Chia sẻ bài viết:** Người dùng chia sẻ bài viết lên tường của mình hoặc cho người khác.

**Chức năng:**

* **Thích/Bỏ thích (Like/Unlike):**
  + Đếm số lượt thích.
  + Hiển thị danh sách người đã thích (tùy chọn).
* **Bình luận (Comments):**
  + Viết bình luận.
  + Trả lời bình luận (Nested comments).
  + Chỉnh sửa/Xóa bình luận của mình.
  + Đếm số bình luận.
* **Chia sẻ (Share):**
  + Chia sẻ lại bài viết lên tường của mình (repost).
  + Chia sẻ trực tiếp cho bạn bè qua tin nhắn.
  + Đếm số lượt chia sẻ.

**4. Kết nối xã hội (Social Connections)**

**Kịch bản:**

* **Tìm kiếm người dùng:** Người dùng tìm kiếm bạn bè hoặc người quen.
* **Gửi/Chấp nhận/Hủy yêu cầu kết bạn:** Quản lý mối quan hệ bạn bè.
* **Hủy kết bạn:** Người dùng chấm dứt mối quan hệ bạn bè.
* **Theo dõi/Bỏ theo dõi:** Người dùng theo dõi trang cá nhân công khai mà không cần kết bạn.

**Chức năng:**

* **Tìm kiếm người dùng:** Theo tên, email, v.v.
* **Gửi yêu cầu kết bạn:**
  + Thông báo đến người nhận.
* **Chấp nhận/Từ chối yêu cầu kết bạn:**
  + Cập nhật trạng thái quan hệ.
* **Hủy kết bạn.**
* **Danh sách bạn bè.**
* **Theo dõi (Follow) / Bỏ theo dõi (Unfollow):** Áp dụng cho tài khoản công khai hoặc trang/nhóm (nếu có).

**5. Nhắn tin (Chat/Messaging)**

**Kịch bản:**

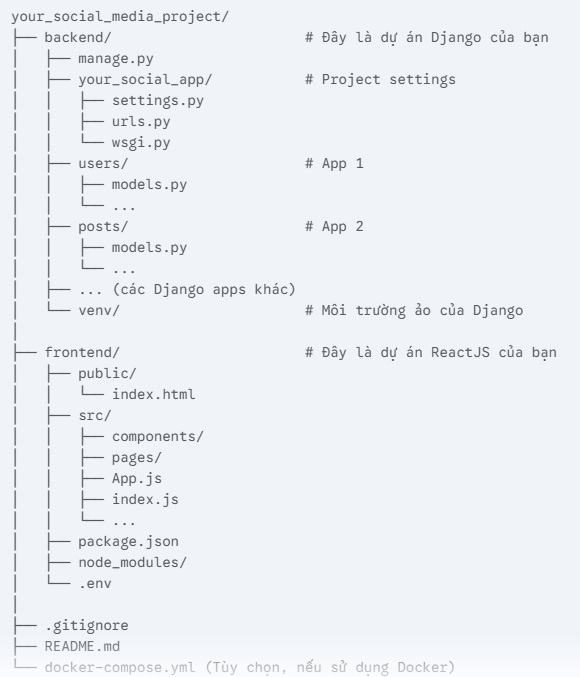
* **Gửi tin nhắn trực tiếp:** Người dùng trò chuyện 1-1.
* **Xem lịch sử tin nhắn:** Người dùng xem lại các cuộc trò chuyện.

**Chức năng:**

* **Gửi/Nhận tin nhắn 1-1 (Direct Message):**
  + Hỗ trợ văn bản.
  + Gửi kèm ảnh/file (tùy chọn).
* **Hiển thị lịch sử trò chuyện.**
* **Trạng thái tin nhắn:** Đã gửi, đã xem.
* **Thông báo tin nhắn mới.**

**6. Các chức năng cơ bản khác**

* **Thông báo (Notifications):**
  + Thông báo khi có người thích/bình luận bài viết của bạn.
  + Thông báo khi có yêu cầu kết bạn mới.
  + Thông báo khi tin nhắn mới.
* **Tìm kiếm:**
  + Tìm kiếm bài viết (theo nội dung, hashtag).
  + Tìm kiếm người dùng.
* **Thiết lập quyền riêng tư chung:**
  + Ai có thể xem hồ sơ của tôi.
  + Ai có thể gửi yêu cầu kết bạn cho tôi.
  + Ai có thể gắn thẻ tôi.



## Các ứng dụng (Apps) cần thiết cho dự án Django

Mỗi ứng dụng sẽ chịu trách nhiệm cho một nhóm chức năng cụ thể và tương tác với các bảng CSDL liên quan.

### 1. users (hoặc accounts)

Ứng dụng này sẽ quản lý tất cả các chức năng liên quan đến người dùng và xác thực.

* **Chức năng chính:**
  + Đăng ký, đăng nhập, đăng xuất người dùng.
  + Quản lý hồ sơ người dùng (chỉnh sửa thông tin cá nhân, ảnh đại diện, ảnh bìa).
  + Thay đổi và khôi phục mật khẩu.
  + Quản lý trạng thái online/offline của người dùng (is\_online).
* **Mô hình (Models) liên quan:**
  + User (hoặc Custom User Model kế thừa từ AbstractUser của Django).

### 2. posts

Ứng dụng này sẽ xử lý việc tạo, quản lý và hiển thị các bài viết.

* **Chức năng chính:**
  + Tạo, chỉnh sửa, xóa bài viết.
  + Hiển thị bảng tin (news feed).
  + Hiển thị bài viết trên trang cá nhân.
  + Quản lý quyền riêng tư cho từng bài viết.
* **Mô hình (Models) liên quan:**
  + Post
  + Status (hoặc có thể dùng choices trong model Post nếu số lượng trạng thái cố định và ít).
  + Media (nếu tách rời để quản lý ảnh/video riêng biệt).

### 3. interactions (hoặc social\_actions)

Ứng dụng này sẽ quản lý tất cả các tương tác với bài viết như lượt thích, bình luận và chia sẻ.

* **Chức năng chính:**
  + Thích/bỏ thích bài viết.
  + Bình luận bài viết và trả lời bình luận.
  + Chia sẻ bài viết.
* **Mô hình (Models) liên quan:**
  + Like (trước đây là LikePost)
  + Comment (trước đây là CommentPost)
  + Share (trước đây là SharePost)

### 4. friends (hoặc connections)

Ứng dụng này sẽ quản lý các mối quan hệ xã hội như kết bạn và theo dõi.

* **Chức năng chính:**
  + Gửi, chấp nhận, từ chối yêu cầu kết bạn.
  + Hiển thị danh sách bạn bè.
  + Hủy kết bạn.
  + Chức năng theo dõi/bỏ theo dõi (nếu có).
* **Mô hình (Models) liên quan:**
  + Friendship
  + FriendRequest

### 5. messaging (hoặc chat)

Ứng dụng này sẽ chịu trách nhiệm cho hệ thống nhắn tin trực tiếp và chat nhóm.

* **Chức năng chính:**
  + Gửi tin nhắn 1-1.
  + Gửi tin nhắn nhóm (nếu triển khai).
  + Hiển thị lịch sử trò chuyện.
* **Mô hình (Models) liên quan:**
  + Conversation
  + ConversationParticipant
  + Message

### 6. notifications (Tùy chọn, nhưng rất nên có)

Ứng dụng này sẽ quản lý và gửi các thông báo cho người dùng.

* **Chức năng chính:**
  + Tạo thông báo khi có tương tác mới (like, comment, share, yêu cầu kết bạn, tin nhắn mới).
  + Hiển thị danh sách thông báo.
  + Đánh dấu thông báo đã đọc.
* **Mô hình (Models) liên quan:**
  + Notification

## Điều chỉnh và Đề xuất Thêm cho Thiết kế Cơ sở dữ liệu

### 1. Bảng User

Bảng User là trung tâm của mọi hoạt động. Dưới đây là các đề xuất:

* **User**:
  + **id**: Kiểu dữ liệu Primary Key, tự động tăng. (Bắt buộc)
  + **username**: VARCHAR(255), UNIQUE. Tên đăng nhập duy nhất. (Bắt buộc)
  + **email**: VARCHAR(255), UNIQUE. Email để đăng nhập và khôi phục mật khẩu. (Bắt buộc)
  + **password\_hash**: VARCHAR(255). Lưu trữ mật khẩu đã được mã hóa (hashed password). **Không bao giờ lưu mật khẩu dưới dạng văn bản thuần!** (Bắt buộc)
  + **full\_name**: VARCHAR(255). Tên đầy đủ của người dùng.
  + **date\_of\_birth**: DATE.
  + **occupation**: VARCHAR(255). Nghề nghiệp.
  + **description / bio**: TEXT. Mô tả bản thân.
  + **avatar\_url**: TEXT. URL đến ảnh đại diện.
  + **cover\_url**: TEXT. URL đến ảnh bìa.
  + **hometown**: VARCHAR(255). Quê quán.
  + **current\_city**: VARCHAR(255). Nơi sống hiện tại.
  + **is\_online**: BOOLEAN, DEFAULT FALSE. Trạng thái online/offline.
  + **created\_at**: DATETIME. Thời gian tạo tài khoản.
  + **updated\_at**: DATETIME. Thời gian cập nhật thông tin gần nhất.
  + **last\_login**: DATETIME. Thời gian đăng nhập gần nhất.
  + **is\_active**: BOOLEAN, DEFAULT TRUE. Trạng thái kích hoạt tài khoản (có thể dùng cho xác minh email hoặc khóa tài khoản).

### 2. Bảng Friend (Mối quan hệ bạn bè)

Đây là cách để thiết kế mối quan hệ 2 người kết bạn với nhau. Bạn cần một bảng để lưu trữ các yêu cầu kết bạn và một bảng để lưu trữ các mối quan hệ bạn bè đã được chấp nhận.

* **Friendship**: Lưu trữ mối quan hệ bạn bè đã được chấp nhận.
  + **id**: Primary Key, tự động tăng.
  + **user1\_id**: Integer, Foreign Key tới User.id. (Bắt buộc)
  + **user2\_id**: Integer, Foreign Key tới User.id. (Bắt buộc)
  + **created\_at**: DATETIME. Thời gian tạo mối quan hệ (khi yêu cầu được chấp nhận).
  + **CONSTRAINT unique\_friendship**: UNIQUE(user1\_id, user2\_id). Đảm bảo không có hai bản ghi trùng lặp cho cùng một cặp bạn bè. Cần thêm một ràng buộc để user1\_id luôn nhỏ hơn user2\_id để tránh trường hợp (A,B) và (B,A) đều tồn tại, hoặc xử lý logic này ở tầng ứng dụng.
* **FriendRequest**: Lưu trữ các yêu cầu kết bạn đang chờ xử lý.
  + **id**: Primary Key, tự động tăng.
  + **sender\_id**: Integer, Foreign Key tới User.id. (Bắt buộc)
  + **receiver\_id**: Integer, Foreign Key tới User.id. (Bắt buộc)
  + **status**: VARCHAR(50), ví dụ: 'pending', 'accepted', 'rejected'. (Bắt buộc)
  + **requested\_at**: DATETIME. Thời gian gửi yêu cầu.
  + **responded\_at**: DATETIME. Thời gian người nhận phản hồi yêu cầu.

### 3. Bảng PersonalPage (Trang cá nhân)

Bạn có thể không cần một bảng PersonalPage riêng biệt như bạn mô tả (UserID PostID). Mỗi người dùng (UserID) sẽ có một trang cá nhân của riêng họ, và các bài viết (PostID) sẽ thuộc về người dùng đó. Mối quan hệ này có thể được quản lý trực tiếp thông qua trường user\_id trong bảng Post.

Nếu bạn muốn có một "trang cá nhân" với các thông tin đặc biệt riêng ngoài thông tin của User, hãy xem xét thêm các trường vào bảng User hoặc tạo một bảng UserProfile liên kết 1-1 với User. Tuy nhiên, với mô tả hiện tại, tôi nghĩ nó không cần thiết.

### 4. Bảng Post

Bạn cần một bảng Post để lưu trữ thông tin về bài viết. **Không cần xây riêng bảng User-Post** vì mối quan hệ 1-nhiều (một User có nhiều Post) sẽ được xử lý bằng Foreign Key từ Post tới User.

* **Post**:
  + **id**: Primary Key, tự động tăng.
  + **user\_id**: Integer, Foreign Key tới User.id. (Bắt buộc)
  + **title**: VARCHAR(255). Tiêu đề bài viết (có thể không bắt buộc).
  + **content**: TEXT. Nội dung chính của bài viết.
  + **media\_url**: TEXT. URL đến ảnh/video (nếu có, có thể là JSON array nếu nhiều file).
  + **created\_at**: DATETIME. Thời gian tạo bài viết.
  + **updated\_at**: DATETIME. Thời gian chỉnh sửa bài viết gần nhất.
  + **status\_id**: Integer, Foreign Key tới Status.id. (Bắt buộc)
  + **location**: VARCHAR(255). Vị trí địa lý (tùy chọn).
  + **is\_active**: BOOLEAN, DEFAULT TRUE. Để quản lý bài viết bị ẩn hoặc xóa mềm.

### 5. Bảng Status

Đây là bảng tham chiếu cho các trạng thái riêng tư của bài viết.

* **Status**:
  + **id**: Primary Key, tự động tăng.
  + **name**: VARCHAR(50), UNIQUE. Ví dụ: 'public', 'private', 'friends\_only', 'custom'. (Bắt buộc)
  + **description**: TEXT. Mô tả chi tiết về trạng thái.

### 6. Bảng LikePost (Mối quan hệ Many-to-Many)

Thiết kế này đã đúng! Bảng này là một bảng trung gian để xử lý mối quan hệ nhiều-nhiều giữa User và Post.

* **Like**:
  + **id**: Primary Key, tự động tăng.
  + **post\_id**: Integer, Foreign Key tới Post.id. (Bắt buộc)
  + **user\_id**: Integer, Foreign Key tới User.id. (Bắt buộc)
  + **created\_at**: DATETIME. Thời gian người dùng like bài viết.
  + **CONSTRAINT unique\_like**: UNIQUE(post\_id, user\_id). Đảm bảo một người dùng chỉ like một bài viết một lần.

### 7. Bảng CommentPost (Đổi tên thành Comment)

Thiết kế này cũng đã đúng.

* **Comment**:
  + **id**: Primary Key, tự động tăng.
  + **post\_id**: Integer, Foreign Key tới Post.id. (Bắt buộc)
  + **user\_id**: Integer, Foreign Key tới User.id. (Bắt buộc)
  + **parent\_comment\_id**: Integer, Foreign Key tới Comment.id, NULLABLE. Để hỗ trợ bình luận lồng nhau (trả lời bình luận khác).
  + **content**: TEXT. Nội dung bình luận. (Bắt buộc)
  + **created\_at**: DATETIME. Thời gian bình luận.
  + **updated\_at**: DATETIME. Thời gian chỉnh sửa bình luận gần nhất.

### 8. Bảng SharePost (Đổi tên thành Share)

Thiết kế này cũng đã đúng.

* **Share**:
  + **id**: Primary Key, tự động tăng.
  + **post\_id**: Integer, Foreign Key tới Post.id. (Bắt buộc)
  + **user\_id**: Integer, Foreign Key tới User.id. Người chia sẻ. (Bắt buộc)
  + **shared\_to\_user\_id**: Integer, Foreign Key tới User.id, NULLABLE. Nếu chia sẻ trực tiếp cho một người dùng.
  + **share\_type**: VARCHAR(50), ví dụ: 'repost', 'direct\_message'.
  + **created\_at**: DATETIME. Thời gian chia sẻ.

### 9. Bảng Message và Conversation (Thay vì BoxID)

Đối với chức năng chat, việc sử dụng khái niệm "Conversation" (hoặc "Box") là hợp lý. Mỗi Conversation sẽ chứa nhiều Message.

* **Conversation**:
  + **id**: Primary Key, tự động tăng.
  + **type**: VARCHAR(50), ví dụ: 'private' (chat 1-1), 'group' (chat nhóm). (Bắt buộc)
  + **name**: VARCHAR(255). Tên nhóm chat (chỉ áp dụng cho nhóm, NULL cho chat 1-1).
  + **created\_at**: DATETIME.
  + **updated\_at**: DATETIME.
* **ConversationParticipant**: Bảng trung gian cho Conversation và User (mối quan hệ nhiều-nhiều).
  + **id**: Primary Key, tự động tăng.
  + **conversation\_id**: Integer, Foreign Key tới Conversation.id. (Bắt buộc)
  + **user\_id**: Integer, Foreign Key tới User.id. (Bắt buộc)
  + **joined\_at**: DATETIME. Thời gian người dùng tham gia cuộc trò chuyện.
  + **left\_at**: DATETIME. Thời gian người dùng rời cuộc trò chuyện (nếu có).
  + **is\_admin**: BOOLEAN, DEFAULT FALSE. Đối với chat nhóm.
  + **CONSTRAINT unique\_participant**: UNIQUE(conversation\_id, user\_id).
* **Message**:
  + **id**: Primary Key, tự động tăng.
  + **conversation\_id**: Integer, Foreign Key tới Conversation.id. (Bắt buộc)
  + **sender\_id**: Integer, Foreign Key tới User.id. (Bắt buộc)
  + **content**: TEXT. Nội dung tin nhắn. (Bắt buộc)
  + **sent\_at**: DATETIME. Thời gian gửi tin nhắn.
  + **is\_read**: BOOLEAN, DEFAULT FALSE. Trạng thái đã đọc (có thể quản lý phức tạp hơn với bảng MessageReadReceipt).

### Bổ sung các bảng tiềm năng khác (cho tương lai):

* **Notification**: Để lưu trữ các thông báo cho người dùng (ai đã like bài viết, ai gửi yêu cầu kết bạn, v.v.).
  + id, user\_id (người nhận thông báo), type (like, comment, friend\_request, message), source\_id (ID của Post, FriendRequest, Message), is\_read, created\_at.
* **Media**: Nếu bạn muốn quản lý file ảnh/video phức tạp hơn, có thể tách riêng bảng này và liên kết đến Post hoặc Message.
  + id, url, type (image, video), file\_size, uploaded\_by\_user\_id, created\_at.

## Thứ tự Hoàn thiện Chức năng Dự án

### Giai đoạn 1: Nền tảng Core (MVP - Minimum Viable Product)

Giai đoạn này tập trung vào các chức năng cốt lõi và tối thiểu để ứng dụng có thể hoạt động và người dùng có thể tương tác cơ bản.

1. **Quản lý người dùng (User Management):**
   * **Đăng ký, Đăng nhập, Đăng xuất:** Nền tảng để người dùng có thể truy cập ứng dụng.
   * **Quản lý hồ sơ cơ bản:** Xem và chỉnh sửa tên hiển thị, ảnh đại diện.
   * **Chỉnh sửa mật khẩu.**
2. **Quản lý bài viết (Post Management):**
   * **Tạo bài viết:** Chỉ văn bản và một ảnh/video đơn lẻ.
   * **Xem bài viết trên trang cá nhân:** Hiển thị bài viết của người dùng hiện tại.
   * **Xóa bài viết:** Chỉ chủ bài viết.
   * **Chế độ riêng tư cơ bản:** Public, Only Me.
3. **Tương tác bài viết (Post Interaction):**
   * **Thích/Bỏ thích bài viết.**
   * **Bình luận bài viết:** Chỉ bình luận cấp 1 (không lồng nhau).
   * **Đếm số lượt thích và bình luận.**
4. **Kết nối xã hội (Social Connections) - Cơ bản:**
   * **Gửi yêu cầu kết bạn.**
   * **Chấp nhận/Từ chối yêu cầu kết bạn.**
   * **Hiển thị danh sách bạn bè.**
   * **Hủy kết bạn.**
   * **Xem bảng tin:** Hiển thị bài viết từ bạn bè.

### Giai đoạn 2: Mở rộng tính năng cốt lõi

Giai đoạn này bổ sung và hoàn thiện các chức năng đã có, làm cho trải nghiệm người dùng phong phú hơn.

1. **Quản lý người dùng (Tiếp):**
   * **Cập nhật hồ sơ đầy đủ:** Thêm bio, ngày sinh, nghề nghiệp, quê quán, nơi sống.
   * **Quên mật khẩu/Khôi phục mật khẩu.**
   * **Trạng thái online/offline.**
2. **Quản lý bài viết (Tiếp):**
   * **Tải lên nhiều ảnh/video cho một bài viết.**
   * **Chỉnh sửa bài viết.**
   * **Chế độ riêng tư mở rộng:** Friends Only, Custom.
3. **Tương tác bài viết (Tiếp):**
   * **Trả lời bình luận (bình luận lồng nhau).**
   * **Chia sẻ bài viết:** Chia sẻ lại trên tường (repost).
4. **Kết nối xã hội (Tiếp):**
   * **Tìm kiếm người dùng.**
   * **Chức năng Theo dõi/Bỏ theo dõi:** Cho phép theo dõi tài khoản công khai mà không cần kết bạn.
5. **Nhắn tin (Messaging) - Cơ bản:**
   * **Tin nhắn 1-1:** Gửi và nhận tin nhắn văn bản.
   * **Hiển thị lịch sử trò chuyện.**

### Giai đoạn 3: Tính năng nâng cao và Hoàn thiện

Giai đoạn này bổ sung các tính năng nâng cao, cải thiện trải nghiệm người dùng và tính ổn định của hệ thống.

1. **Nhắn tin (Tiếp):**
   * **Gửi kèm ảnh/file trong tin nhắn.**
   * **Trạng thái tin nhắn:** Đã gửi, Đã xem.
   * **Thông báo tin nhắn mới.**
   * **Chat nhóm (nếu có yêu cầu):** Tạo nhóm, thêm/xóa thành viên.
2. **Thông báo (Notifications):**
   * **Hệ thống thông báo:** Khi có tương tác mới (like, comment, share), yêu cầu kết bạn, tin nhắn mới.
   * **Đánh dấu đã đọc/chưa đọc.**
   * **Giao diện hiển thị thông báo.**
3. **Tìm kiếm:**
   * **Tìm kiếm bài viết:** Theo nội dung, hashtag.
   * **Tìm kiếm người dùng nâng cao.**
4. **Quyền riêng tư chung:**
   * **Thiết lập quyền riêng tư cho hồ sơ:** Ai có thể xem hồ sơ, ai có thể gửi yêu cầu kết bạn.
   * **Cài đặt gắn thẻ (tagging).**
5. **Tối ưu hiệu suất và bảo mật:**
   * **Tối ưu hóa CSDL và truy vấn.**
   * **Caching.**
   * **Xử lý lỗi và logging.**
   * **Bảo mật API và dữ liệu người dùng.**

**Các thông tin thêm**

**Các options khi tạo field:**

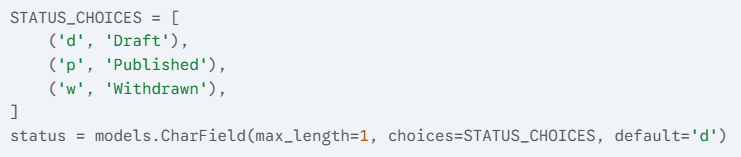
Null = true/false

Blank= true/false – để trống hoặc phải ghi thông tin của trường vào form đăng ký.

Default = value/callable func

unique=True/false – trường này có giá trị duy nhất đối với từng giá trị

choices

****

help\_text: đoạn văn bản trợ giúp ngắn gọn để hiển thị trong biểu mẫu Django (chủ yếu trong Django Admin) hoặc các biểu mẫu tự tạo

verbose\_name: Cung cấp một tên thân thiện, dễ đọc cho trường, được sử dụng trong Django Admin và các thông báo lỗi. Mặc định ví dụ full\_name -> Full Name trong admin

primary\_key = true/false

db\_index = true/false – tạo chỉ mục cho csdl này trog db

editable = true/false - Kiểm soát liệu trường có thể được chỉnh sửa trong Django Admin hoặc trong các model form mặc định hay không.

**Các trường dữ liệu trong Meta khi khai báo class models:**

1. Verbose\_name ; verbose\_name\_plural

2. ordering: ordering = ['category', 'title'] – khai báo thứ tự record khi truy vấn (tương đương với thêm **ORDER BY category, title** với bất kì lệnh truy vấn nào trong model)

3. db\_table – chỉ định tên trong csdl

4. unique\_together = (('author', 'title'),) – chỉ định dữ liệu trong các cặp được khai báo là duy nhất

5. indexes – tạo chỉ mục csdl. Nó định nghĩa một cấu trúc chỉ mục dạng B-tree chứa các địa chỉ trỏ tới các record của 1 trường dữ liệu được định nghĩa.

Ví dụ: **models.Index(fields=['is\_published', '-published\_date'])**

Đây là một **chỉ mục kết hợp** (composite index). Nó được sắp xếp dựa trên cả hai trường. Điều này cực kỳ hữu ích cho các truy vấn vừa lọc vừa sắp xếp, chẳng hạn như **WHERE is\_published = True ORDER BY published\_date DESC**

6. abstract = True/False – không tạo bảng trong csdl; dùng để lớp khác kế thừa và tùy chỉnh Meta

7. app\_label – chỉ định tên ứng dụng thuộc về (ít dùng)

**Tạo form: Chỉ sử dụng với các dự án có FE và BE dính liền (SSR – server side redering). Nếu dùng FE và BE tách dời thì cần xử lí thủ công dùng serializers của DRF (kiến trúc SPA - Single Page Application)**

* + 1. **UserCreationForm:** kế thừa từ ModelForm, được Django cung cấp, nó được cấu hình để làm việc với User mặc định của Django.

Chỉ cần kế thừa và tùy chỉnh các trường cần thiết

Tích hợp các logic để xử lí mật khẩu an toàn(hash validation); code ít

Ít linh hoạt chủ yếu tạo người dùng mới

Class Meta trong UserCreationForm: được cấu hình sẵn để xử lí các trường cần thiết cho việc đăng kí người dùng mới

Model: tên model

Fields: mặc định đã có trường username và password

* + 1. **ModelForm:** form đa năng trong Django, tự động tạo các trường form dựa trên các trường của một model đã có

Linh hoạt: tính tùy chỉnh cao, phù hợp cho cả tạo mới và thay đổi đối tượng model

Cần tự thêm logic validation mật khẩu và hash mật khẩu an toàn

Class Meta trong ModelForm:

**Model** – bắt buộc , tên của Model đang làm việc

**Fields** – liệt kê các trường đưa vào danh sách ( bằng 1 list or 1 tuple)

**Exclude** – loại bỏ các trường không cho vào form (cách an toàn khi không muốn hiển thị trường nhạy cảm)

**labels**: Tùy chỉnh nhãn hiển thị cho các trường. Ví dụ: labels = {'full\_name': 'Họ và Tên'}.

**widgets**: Cung cấp widget HTML tùy chỉnh cho một trường

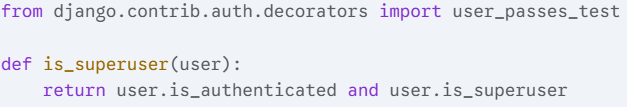
**help\_texts**: Cung cấp văn bản trợ giúp cho các trường.

**Decorators trong việc xác thực(authentication) và quyền truy cập(permission) vào các view**

1. **Phân quyền**
   * **@login\_required**: đảm bảo người dùng đã đăng nhập. Nếu chưa thì chuyển hướng tới settings.LOGIN\_URL
   * **@permission\_required**: đảm bảo đã đăng nhập và có quyền cụ thể để truy cập vào view. Có thể truyền vào một chuổi tên quyền. Nếu không có đủ quyền thì nó chuyển hướng tới trang đăng nhập hoặc trang báo lỗi

**Ví dụ:** @permission\_required(('blog.can\_edit\_article', 'blog.can\_delete\_article'))

* **@staff\_member\_required:** đảm bảo người dùng đã đăng nhập và có is\_staff = True.Thường dành cho nhân viên nội bộ hoặc quản trị viên
* **@superuser\_required** (Không có sẵn, thường tự tạo)

****

* **@user\_passes\_test(test\_func, login\_url=None, redirect\_field\_name=REDIRECT\_FIELD\_NAME)**

Đây là decorator linh hoạt nhất, kiểm tra bất kì điều kiện tùy chỉnh nào đối với người dùng

* **@csrf\_exempt**: Miễn trừ một view khỏi CSRF protection
* **@csrf\_protect**: Buộc CSRF protection cho một view.

1. **Quyền truy cập (permission) sử dụng trong CBV:** trong DRF xử dụng permission thay cho decorators (decorators vẫn dùng được cho FBV)
   * **IsAuthenticated**: Tương tự @login\_required.
   * **IsAdminUser**: Tương tự @staff\_member\_required (kiểm tra is\_staff=True).
   * **IsAuthenticatedOrReadOnly**: Cho phép người dùng đã xác thực thực hiện mọi hành động, nhưng người dùng chưa xác thực chỉ có thể đọc.
   * **DjangoModelPermissions / DjangoObjectPermissions**: Để kiểm tra các quyền dựa trên model hoặc đối tượng cụ thể.

**Nguyên tắc RestFul (Representational State Tranfer)**

**Là một phòng cách kiến trúc để thiết kế hệ thống nối mạng**

* 1. Client – server: tách biệt người dùng client (FE) và server (BE)
  2. Stateless – không trạng thái: mỗi request phải chứa tất cả thông tin để server hiểu yêu cầu đó và đưa ra phản hồi, server không lưu trữ session nào của client
  3. Cacheable: Phản hồi từ server được xác định rõ xem client có cache hay không
  4. Layered system (Hệ thống phân tầng): client không cần biết nó đang kết nối với server hay là một lớp trung gian
  5. Uniform interface (giao diện thống nhất)
     + Tài nguyên được xác định bằng url
     + Client gửi biểu diễn tài nguyên (JSON) để thay đổi thông tin
     + Mỗi thông báo chứa đủ thông tin để client xử lí nó
     + Client tương tác với server thông qua các Url (các components trong giao diện) nhận được từ phản hồi (phản hồi trả về trang homepage sau đó có thể điều hướng đi các trang khác)
  6. Code-on-demand: server có thể tạm thời mở rộng hoặc tùy chỉnh các chức năng của client thông qua mã thực thi

Tác dụng và các loại validate trong **serializers**

* + - Tác dụng:
      * Serialization và deserialization
      * Xác thực (validation): đảm bảo dữ liệu đầu vào là hợp lệ trước khi sử dụng để tạo hoặc cập nhật đổi tượng. Việc này giúp **tách biệt mối quan tâm** (model chỉ chứa logic liên quan tới DB, view chỉ chứa logic liên quan tới request/respond, phân quyền và điều phối)
    - Có 2 loại validate trong serializers:
      * Xác thực cấp trường (validate\_filed\_name): kiểm tra tính hợp lệ của 1 trường
      * Xác thực cấp đối tượng (validate): kiểm tra tính hợp lệ của nhiều tường cùng lúc hoặc ràng buộc giữa chúng

**Các loại Cookie**

Theo thời gian tồn tại: Session cookies/ Persistent Cookies

Theo nguồn gốc: First-party cookies/ Third-party cookies

Theo bảo mật: Secure cookies/ Cookie HttpOnly/ Cookie SameSite