**BỘ THÔNG TIN VÀ TRUYỀN THÔNG**

**HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG**

**KHOA CÔNG NGHỆ THÔNG TIN 2**

---------------------------

**BÁO CÁO CUỐI KỲ MÔN HỌC**

**IOT VÀ ỨNG DỤNG**

***Đề tài:***

**THIẾT KẾ HỆ THỐNG NHẬN DIỆN VIDEO CÓ TÍNH BẠO LỰC BẰNG FRAMEWORK DJANGO**

|  |  |
| --- | --- |
| **Giảng viên hướng dẫn:** | **Đàm Minh Lịnh** |
| **Sinh viên thực hiện:** | **Nguyễn Nhật Minh – N21DCCN053** |

*Thành phố Hồ Chí Minh, ngày 22 tháng 12 năm 2024*

LỜI CẢM ƠN

Để hoàn thành đề tài “Thiết kế hệ thống nhận diện video có tính bạo lực bằng framework Django”, em đã nhận được sự giúp đỡ và hỗ trợ quý báu từ nhiều cá nhân. Nhóm xin bày tỏ lòng biết ơn sâu sắc đến tất cả những người đã đồng hành và hỗ trợ trong suốt quá trình thực hiện nghiên cứu này.

Trong quá trình hoàn thiện đồ án môn “IoT và ứng dụng”. em xin cảm ơn thầy Đàm Minh Lịnh, giảng viên khoa Công Nghệ Thông Tin 2, thầy đã tận tình hướng dẫn, chỉ dạy, truyền đạt kiến thức về môn học một cách tận tình để nhóm có thể hoàn thiện đồ án môn học một cách tốt nhất.

Do những hạn chế về kinh nghiệm và kiến thức nên trong quá trình thực hiện báo cáo đồ án môn học sẽ không tránh khỏi những thiếu sót. Em hi vọng sẽ nhận được những góp ý thêm từ thầy để rút kinh nghiệm trong tương lai.

Cuối cùng, em xin chân thành cảm ơn những đóng góp của thầy và chúc thầy có nhiều niềm vui và sức khỏe!

MỤC LỤC

[DANH MỤC HÌNH ẢNH 1](#_Toc185836272)

[Chương 1. Giới thiệu đề tài 2](#_Toc185836273)

[1.1. Bối cảnh và động lực thực hiện đề tài 2](#_Toc185836274)

[1.1.1. Bối cảnh 2](#_Toc185836275)

[1.1.2. Động lực thực hiện 2](#_Toc185836276)

[1.2. Mục tiêu của dự án 3](#_Toc185836277)

[Chương 2. Tổng quan về framework Django 4](#_Toc185836278)

[2.1. Giới thiệu về Django 4](#_Toc185836279)

[2.2. Nguyên lý thiết kế của Django 4](#_Toc185836280)

[2.3. Ưu điểm và nhược điểm của Django 5](#_Toc185836281)

[2.4. Đặc điểm nổi bật của Django 6](#_Toc185836282)

[2.4.1. Tích hợp ORM 6](#_Toc185836283)

[2.4.2. Hệ thống routing 10](#_Toc185836284)

[2.4.3. Hệ thống và công cụ bảo mật 11](#_Toc185836285)

[2.4.4. Tích hợp và mở rộng hệ thống 13](#_Toc185836286)

[2.4.5. Thư viện và hệ sinh thái hỗ trợ 15](#_Toc185836287)

[Chương 3. Phân tích và thiết kế hệ thống dự án 17](#_Toc185836288)

[3.1. Mô tả và phân tích yêu cầu của hệ thống 17](#_Toc185836289)

[3.2. Thiết kế hệ thống 18](#_Toc185836290)

[3.3. Mô hình nhận diện tính bạo lực 20](#_Toc185836291)

[3.3.1. CNN với MobineNetV2 22](#_Toc185836292)

[3.3.2. CNN kết hợp LSTM hai chiều 23](#_Toc185836293)

[Chương 4. Triển khai hệ thống 28](#_Toc185836294)

[4.1. Môi trường phát triển 28](#_Toc185836295)

[4.2. Các bước triển khai 28](#_Toc185836296)

[4.2.1. Tạo dự án Django và thiết lập các cấu hình cần thiết 28](#_Toc185836297)

[4.2.2. Tạo chức năng người dùng đăng nhập và đăng kí tài khoản 31](#_Toc185836298)

[4.2.3. Tạo chức năng tải video và nhận diện tính bạo lực trong video 34](#_Toc185836299)

[4.2.4. Tạo chức năng đăng kí ứng dụng sử dụng dịch vụ API 37](#_Toc185836300)

[Chương 5. Kết quả và đánh giá 41](#_Toc185836301)

[5.1. Kết quả đạt được 41](#_Toc185836302)

[5.2. Đánh giá hiệu suất 47](#_Toc185836303)

[5.3. Hạn chế 48](#_Toc185836304)

[5.4. Kết luận và hướng phát triển 48](#_Toc185836305)

[TÀI LIỆU THAM KHẢO 49](#_Toc185836306)

# DANH MỤC HÌNH ẢNH

[Hình 1: Giao diện đăng nhập 37](#_Toc185833440)

[Hình 2: Giao diện đăng kí tài khoản 38](#_Toc185833441)

[Hình 3: Giao diện đăng nhập tài khoản admin tích hợp của Django 38](#_Toc185833442)

[Hình 4: Giao diện chính sau khi đăng nhập tài khoản admin hệ thống 38](#_Toc185833443)

[Hình 5: Giao diện quản lí người dùng hệ thống của Django 39](#_Toc185833444)

[Hình 6: Giao diện trang chủ sau khi người dùng đăng nhập thành công 39](#_Toc185833445)

[Hình 7: Giao diện chức năng nhận diện video có tính bạo lực 40](#_Toc185833446)

[Hình 8: Giao diện xem trước video trước khi tải lên hệ thống 40](#_Toc185833447)

[Hình 9: Giao diện xem danh sách các video đã gửi lên hệ thống và kết quả nhận diện tính bạo lực của video 41](#_Toc185833448)

[Hình 10: Giao diện danh sách các ứng dụng đã đăng kí sử dụng API nhận diện bạo lực 41](#_Toc185833449)

[Hình 11: Giao diện tạo ứng dụng sử dụng dịch vụ mới 42](#_Toc185833450)

[Hình 12: Giao diện thông tin ứng dụng đã đăng kí và public key để sử dụng dịch vụ API 42](#_Toc185833451)

[Hình 13: Giao diện xem danh sách các video mà ứng dụng đã tải lên thông qua API 43](#_Toc185833452)

# Giới thiệu đề tài

## Bối cảnh và động lực thực hiện đề tài

### Bối cảnh

Với sự phát triển nhanh chóng của công nghệ thông tin và truyền thông, video đã trở thành một trong những dạng nội dung được sử dụng phổ biến nhất trên Internet. Tuy nhiên, đây cũng là kênh truyền dẫn đến nhiều nguy cơ, bao gồm việc lan truyền các nội dung nhạy cảm, bạo lực và gây tác động tiêu cực đến người xem. Việc kiểm soát nội dung bạo lực trong video đã trở thành một thách thức đối với các nền tảng chia sẻ nội dung và các nhà quản lý.

Song song đó, sự phát triển của các mô hình trí tuệ nhân tạo (AI) và học sâu (Deep Learning) đã mở ra những cách tiếp cận hiệu quả trong việc nhận diện nội dung video. Các mô hình như CNN (Convolutional Neural Network) và LSTM (Long Short-Term Memory) đã chứng minh được khả năng trong việc phân tích và dự đoán dữ liệu video. Việc kết hợp các công nghệ này với các framework phát triển web như Django cung cấp một nền tảng vững chắc cho việc xây dựng các hệ thống tự động phát hiện nội dung bạo lực.

### Động lực thực hiện

* **Vấn đề thực tế:** Trong bối cảnh video trở thành phương tiện truyền tải chính, nguy cơ lan truyền nội dung bạo lực gia tăng, đòi hỏi các giải pháp để phát hiện nhanh chóng và tự động nội dung nhạy cảm.
* **Cơ hội khai thác công nghệ:** Kỹ thuật AI và học sâu có tiềm năng lớn trong việc phân tích video, nhất là với những tiến bộ trong các mô hình như MobileNetV2 và LSTM.
* **Học hỏi và ứng dụng:** Thông qua việc xây dựng ứng dụng, nhóm thực hiện có thể tích lũy kinh nghiệm trong việc kết hợp giữa AI và công nghệ phát triển web như Django, qua đó trang bị thêm kiến thức và kỹ năng làm việc thực tế.
* **Tầm quan trọng của API:** API cung cấp khả năng tích hợp linh hoạt vào nhiều hệ thống ngoài, gia tăng khả năng mở rộng của giải pháp.

Với những bối cảnh và động lực nêu trên, đề tài này được thực hiện nhằm đáp ứng những nhu cầu thực tiễn và từng bước hoàn thiện một giải pháp hiệu quả dựa trên công nghệ mới nhất.

## Mục tiêu của dự án

1. **Mục tiêu tổng quát**

* Xây dựng một hệ thống tự động phát hiện nội dung bạo lực trong video, tạo nền tảng để phát triển các dịch vụ và hệ thống tương tự trong tương lai.

1. **Mục tiêu chi tiết**

* Xây dựng chức năng nhận diện nội dung bạo lực trong video MP4.
* Phát triển API cung cấp dịch vụ dự đoán, cho phép tích hợp vào các ứng dụng bên ngoài.
* Triển khai hệ thống cung cấp public key để đảm bảo mức độ bảo mật khi tích hợp API.
* Tối ưu hóa mô hình học sâu MobileNetV2 kết hợp với LSTM nhằm tăng độ chính xác và hiệu quả xử lý.
* Cung cấp giao diện web trực quan, thân thiện, giúp người dùng dễ dàng upload video và nhận kết quả.

# Tổng quan về framework Django

## Giới thiệu về Django

Django là một framework phát triển web mã nguồn mở được xây dựng bằng ngôn ngữ lập trình Python, nổi bật với khả năng hỗ trợ phát triển nhanh chóng, hiệu quả và an toàn. Ra đời vào năm 2005 bởi Adrian Holovaty và Simon Willison, Django tuân thủ nguyên tắc "Don't Repeat Yourself" (DRY), giúp giảm thiểu việc lặp lại mã nguồn và tối ưu hóa quy trình phát triển. Với cấu trúc rõ ràng, Django cung cấp một hệ thống quản lý cơ sở dữ liệu tích hợp, khả năng mở rộng linh hoạt, cùng các công cụ mạnh mẽ để xử lý bảo mật như chống tấn công SQL Injection, Cross-Site Scripting (XSS), và Cross-Site Request Forgery (CSRF). Nhờ đó, Django đã trở thành sự lựa chọn ưu tiên cho nhiều dự án quy mô từ các ứng dụng doanh nghiệp nhỏ đến các hệ thống lớn như Instagram, Pinterest và Mozilla. Framework này không chỉ hỗ trợ lập trình viên xây dựng các ứng dụng web một cách nhanh chóng mà còn duy trì tính ổn định và bảo mật cao trong quá trình vận hành.

## Nguyên lý thiết kế của Django

Django được thiết kế dựa trên nhiều nguyên lý cốt lõi là tối đa hiệu quả lập trình, dễ bảo trì, và đảm bảo tính bảo mật. Các nguyên lý này được thể hiện qua kiến trúc và các thành phần trong hệ sinh thái của Django.

1. **Kiến trúc MVC (Model-View-Controller)**

Django tuân thủ kiến trúc MVC, nhưng với sự tuý chỉnh nhỏ để phù hợp với Python, gọi là MVT (Model-View-Template):

* **Model:** Quản lý dữ liệu và các logic liên quan đến kinh doanh. Django cung cấp một ORM (Object-Relational Mapping) để tương tác với cơ sở dữ liệu, giúp lập trình viên thao tác dữ liệu bằng Python thay vì SQL.
* **View:** Chịu trách nhiệm xử lý logic ứng dụng, nhận dữ liệu từ Model và chuyển cho Template để hiển thị. View trong Django được thiết kế linh hoạt, giúp xây dựng logic phức tạp nhưng dễ quản lý.
* **Template:** Đáp ứng giao diện người dùng, được tạo thông qua hệ thống ngôn ngữ Template của Django, giúp tách biệt giao diện và logic.

1. **Nguyên lý DRY – Don’t Repeat Yourself**

Django tự hào với việc đảm bảo nguyên lý DRY, giúp giảm thiểu việc lặp lại mã nguồn bằng cách cung cấp các công cụ như forms, middleware, và các thư viện tích hợp sẵn. Hệ thống này giúp tăng hiệu quả và giảm nguy cơ sai sót khi phát triển.

1. **Bảo mật là ưu tiên hàng đầu**

Django tích hợp các cơ chế bảo mật như: chống SQL Injection, Cross-Site Scripting (XSS), và Cross-Site Request Forgery (CSRF). Ngoài ra, hệ thống xác thực và quản lý quyền trong Django cung cấp các công cụ mạnh mẽ để đảm bảo an toàn dữ liệu.

1. **Khả năng mở rộng và tính module hóa**

Django được thiết kế với khả năng mở rộng cao, cho phép người dùng tích hợp các thư viện bên ngoài hoặc tự xây dựng các module riêng để đáp ứng nhu cầu đặc thù. Django cung cấp các "apps" độc lập, cho phép tái sử dụng trong nhiều dự án khác nhau.

1. **Hỗ trợ phát triển nhanh chóng**

Với việc tích hợp sẵn các công cụ như hệ thống quản trị Admin tích hợp, URL dispatcher linh hoạt, và hệ thống testing mạnh mẽ, Django giúp đơn giản hóa quy trình phát triển và kiểm thử ứng dụng.

Tổng hợp các nguyên lý và kiến trúc nêu trên, Django không chỉ mang lại hiệu quả và tính bền vững cho quá trình phát triển, mà còn đáp ứng các nhu cầu bảo mật và duy trì hệ thống trong thời gian dài.

## Ưu điểm và nhược điểm của Django

1. **Ưu điểm**
   * Phát triển nhanh: Django cung cấp các công cụ tích hợp giúp giảm thiểu thời gian phát triển.
   * Bảo mật cao: Django tích hợp nhiều cương chế bảo mật, giúp ngăn chặn các tấn công phổ biến.
   * Khả năng mở rộng: Hệ sinh thái phong phú cùng các "apps" tái sử dụng được.
   * Cộng đồng hỗ trợ mạnh mẽ: Django có cộng đồng lớn, tài liệu phong phú, giúp người mới bắt đầu dễ dàng.
2. **Nhược điểm**
   * Hiệu năng: Django không thích hợp cho các ứng dụng yêu cầu hiệu năng cao như game thời gian thực.
   * Cấu hình phức tạp: Các dự án nhỏ có thể cảm thấy Django quá phức tạp.
   * Hạn chế linh hoạt: Do Django tuân thủ nhiều quy định chặt chẽ, việc tuý biến có thể khó khăn trong một số trường hợp.

**So sánh Django với một số framework khác**

|  |  |  |  |
| --- | --- | --- | --- |
| Tiêu chí đánh giá | Django | Flask | FastAPI |
| Kiểu framework | Full-stack | Microframework | API-first |
| Mục đích sử dụng | Ứng dụng web phức tạp có giao diện | Ứng dụng web nhỏ gọn, linh hoạt | API, Microservice, ứng dụng Real-time |
| ORM tích hợp | Có (Django ORM) | Không | Không |
| Quản trị Admin | Có, tích hợp sẵn | Không | Không |
| Hiệu năng | Trung bình | Cao | Rất cao |
| Hệ sinh thái thư viện | Rất phong phú | Phụ thuộc vào bên thứ ba | Phụ thuộc vào bên thứ ba |
| Bảo mật | Cao, tích hợp sẵn | Tùy thuộc lập trình viên | Tích hợp Open AI Security |
| Dễ bảo trì | Cao, nguyên lý DRY | Phụ thuộc vào quy tắc lập trình | Cao, nhờ typing và bất đồng bộ |
| Hỗ trợ bất đồng bộ | Hạn chế (có nhưng không đầy đủ) | Không tích hợp sẵn | Được hỗ trợ hoàn chỉnh |
| Thời gian tạo hệ thống | Lâu | Nhanh | Rất nhanh |

## Đặc điểm nổi bật của Django

### Tích hợp ORM

Django tích hợp một ORM (Object-Relational Mapping) mạnh mẽ, cho phép ta làm việc với cơ sở dữ liệu một cách thuận tiện mà không cần viết câu lệnh SQL thủ công. ORM của Django chuyển đổi dữ liệu giữa các bảng trong cơ sở dữ liệu (dạng quan hệ) và các đối tượng Python (dạng đối tượng).

**Các đặc điểm chính của ORM trong Django:**

1. **Mô hình hóa dữ liệu**

* Django sử dụng các class Python để định nghĩa bảng cơ sở dữ liệu.
* Mỗi class trong file “models.py” tương ứng với một bảng trong cơ sở dữ liệu.
* Các thuộc tính của class tương ứng với các cột trong bảng.



Lớp Application ở trên sẽ được chuyển thành bảng “applications” tương ứng với các cột dữ liệu là: id, name, description, public\_key, private\_key, status, user\_id, created\_at, deleted\_at.

1. **Quản lý cơ sở dữ liệu**

Django cung cấp các công cụ quản lý cơ sở dữ liệu, chẳng hạn như:

* Migrations: Tự động tạo và quản lý các thay đổi cấu trúc cơ sở dữ liệu.
* Makemigrations: Tạo file migration.
* Migrate: Áp dụng những thay đổi lên cơ sở dữ liệu.

1. **Tương tác với dữ liệu**

Django ORM cung cấp các API truy vấn linh hoạt, bao gồm:

* **Thêm dữ liệu:**



Đây là hàm thêm một dòng dữ liệu mới cho bảng “applications” đã định nghĩa ở trên. Hàm “create” sẽ trả về một đối tượng mới chứa dữ liệu vừa được tạo.

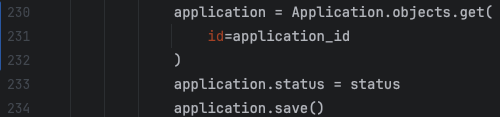
* **Truy vấn dữ liệu:**



Dòng 210: là cách lấy một dòng dữ liệu có mã “id” bằng giá trị mong muốn trong bảng “application”.

Dòng 211 đến dòng 213: là cách lấy một danh sách các media được tải lên hệ thống bằng ứng dụng có mã “id” bằng giá trị mong muốn và sắp xếp kết quả theo thứ tự giảm dần theo cột “created\_at”.

* **Cập nhật dữ liệu:**



Dòng 230 đến 232: là cách lấy một dòng dữ liệu có mã “id” bằng giá trị mong muốn trong bảng “application”.

Dòng 233: sẽ cập nhật thuộc tính “status” của đối tượng vừa tìm được bằng giá trị mới.

Dòng 234: sẽ lưu đối tượng với thuộc tính “status” đã cập nhật xuống CSDL.

* **Xóa dữ liệu:**



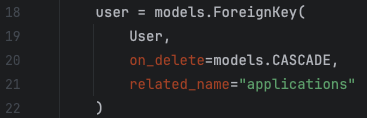
Dòng 236: sẽ tìm một author có mã “id” bằng 1.

Dòng 237: sẽ xóa dòng dữ liệu tương ứng trong CSDL.

1. **Quan hệ giữa các bảng:**

Django hỗ trợ các loại quan hệ giữa bảng như:

* One-to-One: “OneToOneField”
* One-to-Many: “ForeignKey”
* Many-to-Many: “ManyToManyField”



Đây là cách định nghĩa khóa ngoại “user\_id” cho bảng “applications” trong Django.

1. **Truy vấn nâng cao:**

Django hỗ trợ các truy vấn phức tạp như:

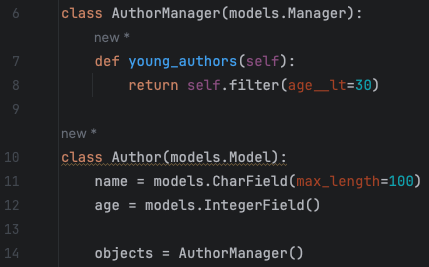
* Gộp nhóm (annotate, aggregate)
* Sắp xếp (order\_by)
* Truy vấn quan hệ (select\_related, prefetch\_related)



Dòng 9: Thực hiện đếm số lượng sách của từng tác giả.

1. **Tùy chỉnh:**

Ta có thể tùy chỉnh ORM của Django bằng cách viết các quản lý tùy chỉnh (Custom Managers) hoặc QuerySet.



Ta có thể định nghĩa một lớp “AuthorManager” là một đối tượng “Manager”. Lúc này, ta có thể lấy danh sách những tác giả có tuổi bé hơn hoặc bằng 30 như sau:



Ưu điểm của Django ORM:

* Tự động hóa nhiều công việc liên quan đến cơ sở dữ liệu.
* Độc lập với hệ quản trị cơ sở dữ liệu (MySQL, PostgreSQL, SQLite...).
* Dễ đọc, dễ bảo trì, giúp giảm thiểu lỗi cú pháp SQL.
* Tích hợp chặt chẽ với các thành phần khác của Django.

### Hệ thống routing

Hệ thống routing trong Django được quản lý thông qua URLconf (URL Configuration), cho phép ta ánh xạ các URL đến các view (hàm hoặc class) tương ứng. Đây là một trong những phần quan trọng nhất của Django Framework, giúp ứng dụng xử lý các yêu cầu HTTP và trả về phản hồi tương ứng.

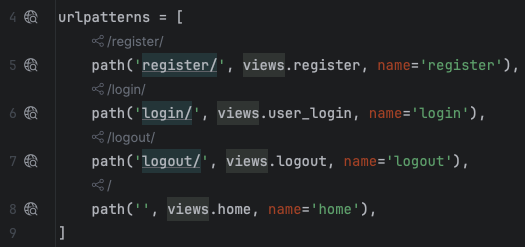
**Các thành phần chính của hệ thống routing trong Django:**

1. **File “urls.py”:**

* Mỗi dự án Django có file“urls.py” chính nằm ở cấp dự án (thư mục chứa settings.py).
* Các ứng dụng con cũng có thể có file “urls.py” riêng để tổ chức URL tốt hơn.

1. **Cách cấu hình URL:**

* File “urls.py” sử dụng danh sách “urlpatterns” để ánh xạ các đường dẫn URL đến các view tương ứng.
* Sử dụng hàm “path()” hoặc “re\_path()” để định nghĩa các quy tắc URL.



Danh sách “urlpatterns” chứa các định nghĩa về các route của app “system”. Các route được định nghĩa bao gồm: Trang chủ, Trang đăng kí tài khoản người dùng, Trang đăng nhập tài khoản người dùng và Trang đăng xuất tài khoản người dùng.

1. **Các hàm định nghĩa URL:**

Hàm “path(route, view, kwargs=None, name=None)”:

* route: Chuỗi mô tả mẫu URL.
* view: Hàm hoặc class xử lý yêu cầu.
* kwargs: Tham số tùy chọn truyền cho view.
* name: Tên để tham chiếu URL.

Hàm “re\_path(route, view, kwargs=None, name=None)”: Dùng regex để định nghĩa mẫu URL phức tạp.

1. **Sử dụng tham số trong URL:**

Ta có thể định nghĩa các tham số trong URL để truyền dữ liệu vào view.



Dòng 8 là cách ta có thể thêm tham số “id” kiểu số nguyên trong URL khi truy cập vào view chi tiết ứng dụng.

1. **Routing theo ứng dụng:**

* Trong dự án lớn, ta có thể tách riêng “urls.py” cho từng ứng dụng.
* Sử dụng hàm “include()” để đưa các cấu hình URL của ứng dụng con vào file “urls.py” chính.

1. **Đặt tên cho URL:**

* Django cho phép bạn đặt tên cho các đường dẫn URL bằng tham số “name”.
* Tên này giúp ta dễ dàng tham chiếu đến URL trong code (template, view, v.v.).

### Hệ thống và công cụ bảo mật

Django cung cấp một loạt các công cụ bảo mật tích hợp nhằm giúp lập trình viên xây dựng các ứng dụng web an toàn hơn. Những tính năng này bao gồm các biện pháp phòng ngừa chống lại tấn công phổ biến như XSS, CSRF, SQL Injection, Clickjacking, và nhiều vấn đề bảo mật khác.

1. **Chống tấn công SQL Injection**

Django ORM tự động sử dụng parameterized queries, giúp ngăn ngừa SQL Injection. Mọi tham số được truyền vào truy vấn sẽ được xử lý an toàn, không chèn trực tiếp vào câu lệnh SQL.

1. **Chống tấn công XSS (Cross-Site Scripting)**

Django tự động escape (lọc) dữ liệu được hiển thị trong các template để ngăn mã JavaScript độc hại chạy trên trình duyệt của người dùng.

1. **Bảo vệ chống CSRF (Cross-Site Request Forgery)**

Django tích hợp CSRF protection thông qua middleware và sử dụng token CSRF để xác minh các yêu cầu POST.

*Cách sử dụng*: Trong các form HTML, thêm thẻ CSRF:



1. **Xác thực và quản lý phiên**

Django có hệ thống xác thực người dùng mạnh mẽ, bao gồm:

* Hash mật khẩu với các thuật toán an toàn (PBKDF2, Argon2, bcrypt, SHA-1).
* Bảo vệ chống lại brute-force với “User.is\_active”.
* Các session được lưu trữ an toàn (cookie-based hoặc database-based).

Ta có thể tùy chỉnh hoặc mở rộng hệ thống xác thực với:

* Custom User Model
* Middleware
* Signals

1. **Clickjacking Protection**

Django tích hợp cơ chế bảo vệ chống Clickjacking thông qua việc thêm header “HTTP X-Frame-Options”.



Dòng 31 sẽ cấu hình chống Clickjacking. Đây là thiết lập mặc định của Django.

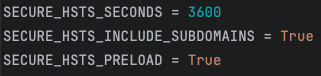
1. **Bảo mật dữ liệu với HTTPS**

Django khuyến khích sử dụng HTTPS để mã hóa dữ liệu truyền tải giữa trình duyệt và máy chủ. Các cấu hình liên quan:

* Bắt buộc HTTPS



* Thiết lập HSTS (HTTP Strict Transport Security):



1. **Bảo vệ mật khẩu**

Django sử dụng trình hash mật khẩu mặc định:

* PBKDF2 (mặc định).
* Argon2.
* bcrypt.

Cấu hình hash mật khẩu trong “settings.py”:



1. **Bảo vệ header HTTP**

Django cung cấp cách thêm các header bảo mật HTTP:

* Content Security Policy (CSP): Django không tích hợp sẵn CSP nhưng ta có thể sử dụng các thư viện như django-csp.
* X-Content-Type-Options: bằng cách cấu hình như hình dưới:



* X-XSS-Protection: Django không tự động thêm header này, nhưng ta có thể thêm thủ công qua middleware.

1. **Session Security**

Django hỗ trợ các tính năng bảo mật session:

* Chỉ cho phép session qua HTTPS:



* Không cho phép JavaScript truy cập session cookie:



### Tích hợp và mở rộng hệ thống

Django là một framework thiết kế theo nguyên tắc "batteries-included", cung cấp sẵn rất nhiều công cụ tích hợp mạnh mẽ. Đồng thời, Django cũng hỗ trợ mở rộng và tùy chỉnh linh hoạt để phù hợp với các yêu cầu đặc thù của ứng dụng.

**Tích hợp các tính năng sẵn có:**

1. Hệ thống quản trị (Admin Interface)

* Django cung cấp một giao diện quản trị sẵn có giúp ta dễ dàng quản lý dữ liệu trong cơ sở dữ liệu.
* Chỉ cần khai báo model trong models.py, và Django Admin sẽ tự động tạo các trang CRUD (Create, Read, Update, Delete).

1. ORM (Object-Relational Mapping)

* Django ORM giúp làm việc với cơ sở dữ liệu mà không cần viết SQL.
* Hỗ trợ nhiều hệ quản trị cơ sở dữ liệu như PostgreSQL, MySQL, SQLite, …
* Cho phép sử dụng các lệnh truy vấn nâng cao.

1. Hệ thống xác thực

Django cung cấp sẵn hệ thống quản lý người dùng và xác thực với các tính năng:

* Đăng ký, đăng nhập, đăng xuất.
* Reset mật khẩu qua email.
* Hỗ trợ nhóm (Group) và quyền (Permission).

1. Tích hợp REST API

Django có thể kết hợp với Django REST Framework (DRF) để xây dựng API RESTful.

DRF cung cấp các tính năng như:

* Serialization (chuyển đổi dữ liệu giữa JSON và Python object).
* Xác thực token, JWT, OAuth2.
* ViewSets và Routers để xây dựng API nhanh chóng.

**Mở rộng hệ thống Django:**

1. Middleware

* Middleware là các lớp xử lý yêu cầu và phản hồi HTTP.
* Ta có thể tạo middleware tùy chỉnh để thêm chức năng vào quy trình xử lý yêu cầu trước khi render view.

1. Tùy chỉnh hệ thống xác thực
2. Tùy chỉnh quản trị Django

Ta có thể tùy chỉnh giao diện quản trị với các công cụ như:

* “list\_display”: Hiển thị cột trong danh sách.
* “list\_filter”: Bộ lọc bên cột phải.
* “fieldsets”: Tổ chức trường trong form.

1. Tạo Template Tags và Filters

* Template tags và filters giúp mở rộng khả năng xử lý dữ liệu trong template.

1. Tùy chỉnh Signals

Signals là cách Django cho phép ta "nghe" các sự kiện xảy ra trong ứng dụng, chẳng hạn như khi một đối tượng được lưu, xóa, ….

1. Tích hợp ứng dụng bên thứ ba

Django hỗ trợ tích hợp dễ dàng với các thư viện của bên thứ ba, ví dụ:

* django-allauth: Quản lý xác thực qua mạng xã hội.
* django-crispy-forms: Tăng cường biểu mẫu HTML.
* django-celery: Xử lý tác vụ không đồng bộ.

1. Tùy chỉnh hệ thống routing

Django sử dụng “urlconf” để định nghĩa URL. Ta có thể mở rộng hoặc viết middleware tùy chỉnh để xử lý URL động.

1. Sử dụng Signal/Hook

Django Signals cho phép ta lắng nghe sự kiện và thực thi mã khi sự kiện xảy ra.

### Thư viện và hệ sinh thái hỗ trợ

Django có một hệ sinh thái phong phú gồm các thư viện và công cụ hỗ trợ giúp tăng tốc phát triển và mở rộng ứng dụng. Hệ sinh thái này bao gồm các công cụ quản lý xác thực, xử lý form, API, tối ưu hiệu suất, và nhiều tính năng khác.

**Quản lý và xác thực người dùng:**

1. django-allauth

* Hỗ trợ xác thực qua email/password và các mạng xã hội như Google, Facebook, GitHub, ….
* Tích hợp sẵn cơ chế đăng nhập, đăng ký, và quản lý tài khoản.

1. django-axes

* Bảo vệ chống lại brute-force attacks bằng cách chặn các IP sau nhiều lần đăng nhập thất bại.

**Xây dựng API:**

1. Django REST framework

* Thư viện mạnh mẽ để xây dựng API RESTful.
* Hỗ trợ Serialization, ViewSets, Routers, và các phương thức xác thực (Token, OAuth2).

1. Graphene-Django

* Thư viện giúp xây dựng API GraphQL với Django.

**Quản lý form và giao diện:**

1. django-crispy-forms

* Tăng cường khả năng xử lý form, hỗ trợ render đẹp hơn với các framework CSS như Bootstrap.

1. django-widget-tweaks

* Cho phép tùy chỉnh widget trong template mà không cần sửa đổi form class.

**Xử lý dữ liệu lớn và công việc không đồng bộ:**

1. django-celery

* Sử dụng Celery để xử lý các công việc không đồng bộ (asynchronous tasks), như gửi email, xử lý file, ….

1. django-redis

* Sử dụng Redis làm backend cache hoặc session.

**Tối ưu hóa và bảo mật:**

1. django-debug-toolbar

* Công cụ debug mạnh mẽ, hiển thị chi tiết thông tin các query SQL, cache, ….

1. django-secure

* Bổ sung các header HTTP bảo mật như Content Security Policy (CSP), HSTS.

**Quản lý tệp và hình ảnh:**

1. django-storages

* Tích hợp với các dịch vụ lưu trữ tệp như AWS S3, Google Cloud Storage.

1. Pillow

* Thư viện xử lý hình ảnh tích hợp với Django.

**Quản lý quốc tế hóa (I18N) và đa ngôn ngữ:**

1. django-modeltranslation

* Dịch các trường trong model sang nhiều ngôn ngữ.

1. django-parler

* Hỗ trợ quản lý nội dung đa ngôn ngữ với cấu trúc dữ liệu tách biệt.

**Thư viện hỗ trợ giao diện:**

1. django-tailwind: Tích hợp Tailwind CSS vào dự án Django.
2. django-suit: Thay đổi giao diện Admin Django thành giao diện đẹp và hiện đại hơn.

# Phân tích và thiết kế hệ thống dự án

## Mô tả và phân tích yêu cầu của hệ thống

Hệ thống nhận diện video có tính bạo lực sẽ quản lí thông tin của người dùng. Các video mà người dùng gửi lên hệ thống sẽ được lưu trữ và tự động nhận diện tính bạo lực. Người dùng có thể đăng kí các ứng dụng để sử dụng dịch vụ API nhận diện tính bạo lực thông qua khóa public mà hệ thống cung cấp cho từng ứng dụng.

**Người dùng (auth\_user):** là người sử dụng hệ thống sẽ được lưu trữ các thông tin như sau.

* Họ và tên người dùng.
* Email đăng kí tài khoản.
* Password của tài khoản.
* Username của tài khoản. Mỗi tài khoản một username khác nhau.
* Last login: thời gian đăng nhập cuối cùng.
* Is staff: định danh nhân viên hệ thống.
* Is superuser: định danh quản lí hệ thống.
* Is active: định danh trạng thái hoạt động.
* Date joined: thời gian đăng kí tài khoản.

**Ứng dụng (application):** là các ứng dụng sử dụng dịch vụ API của hệ thống sẽ được lưu trữ các thông tin dưới đây.

* Tên ứng dụng.
* Mô tả của ứng dụng.
* Khóa bí mật và khóa công khai.
* Trạng thái hoạt động của ứng dụng. Người dùng có thể tắt hoặc mở ứng dụng bằng cách thay đổi trạng thái hoạt động.
* Created at: thời gian tạo ứng dụng.

**Media:** lưu trữ thông tin về các video được gửi lên hệ thống.

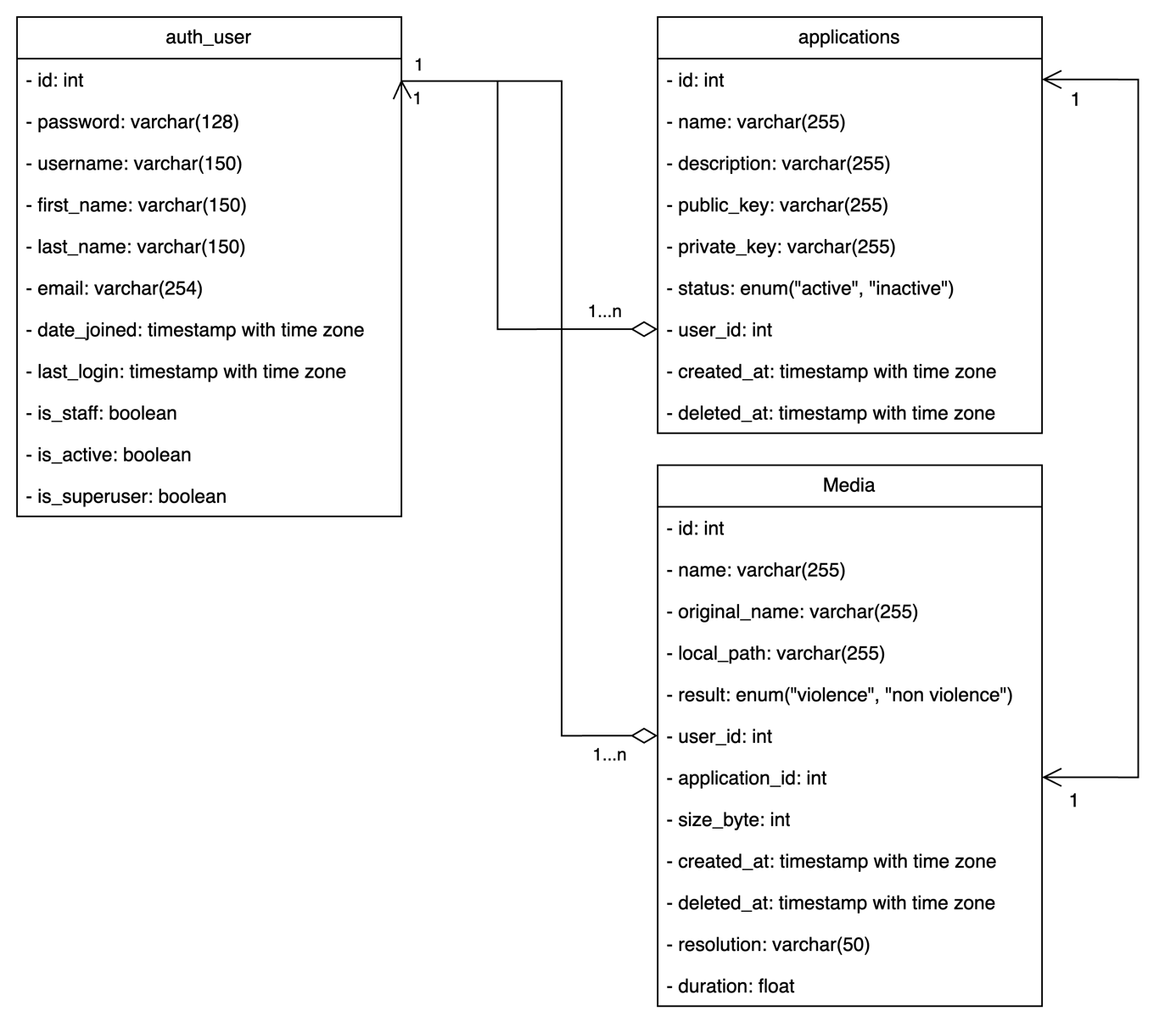
* Tên duy nhất. Tên này sẽ được hệ thống sinh tự động với kiểu UUID V4.
* Tên gốc: Tên gốc của file video người dùng tải lên hệ thống.
* Kích thước video.
* Độ phân giải.
* Địa chỉ lưu trữ trên hệ thống.
* Mã người dùng tải video.
* Mã ứng dụng tải video.
* Thời lượng.
* Kết quả dự đoán tính bạo lực.
* Thời gian video được tải lên.

Các yêu cầu của hệ thống:

* Người dùng có thể đăng kí và đăng nhập với tài khoản đã đăng kí.
* Người dùng có thể tải video lên hệ thống, hệ thống xác định tính bạo lực và trả về kết quả tương ứng.
* Người dùng xem danh sách các video đã tải lên.
* Người dùng có thể đăng kí ứng dụng sử dụng dịch vụ API.
* Ứng dụng bên ngoài có thể gọi API.

## Thiết kế hệ thống

**Sơ đồ ERD:**



**Từ điển dữ liệu:**

Bảng “auth\_user”:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **STT** | **Tên trường** | **Kiểu dữ liệu** | **Ý nghĩa** | **Ràng buộc** |
| 1 | id | int | ID người dùng, tăng tự động | Primary Key |
| 2 | First\_name | Varchar(150) | Tên người dùng | Not null |
| 3 | Last\_name | Varchar(150) | Họ người dùng | Not null |
| 4 | email | Varchar(254) | Email tài khoản | Not null |
| 5 | password | Varchar(128) | Mật khẩu tài khoản | Not null |
| 6 | username | Varchar(150) | Username tài khoản | Not null, Unique Key |
| 7 | Date\_joined | Date time with timezone | Mốc thời gian đăng kí tài khoản. | Not null |
| 8 | Is\_supersuser | boolean | Định danh quản lí hệ thống | Not null, Default: false |
| 9 | Is\_staff | boolean | Định danh nhân viên hệ thống | Not null, Default: false |
| 10 | Is\_active | boolean | Định danh trạng thái hoạt động | Not null, Default: false |
| 11 | Last\_login | Date time with timezone | Thời gian đăng nhập gần đây nhất |  |

Bảng “applications”

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **STT** | **Tên trường** | **Kiểu dữ liệu** | **Ý nghĩa** | **Ràng buộc** |
| 1 | id | int | ID ứng dụng, tăng tự động | Primary Key |
| 2 | name | Varchar(256) | Tên ứng dụng | Not null |
| 3 | description | Varchar(256) | Mô tả ứng dụng | Not null |
| 4 | Public\_key | Varchar(256) | Khóa công khai | Not null, Unique Key |
| 5 | Private\_key | Varchar(256) | Khóa bí mật | Not null, Unique Key |
| 6 | status | Varchar(8) | Trạng thái hoạt động của ứng dụng | Not null, Default: active |
| 7 | Created\_at | Date time with timezone | Mốc thời gian tạo ứng dụng. | Not null |
| 8 | Deleted\_at | Date time with timezone | Mốc thời gian xóa ứng dụng. | Default: null |
| 9 | User\_id | int | Mã người dùng tạo ứng dụng | FK |

Bảng “media”:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **STT** | **Tên trường** | **Kiểu dữ liệu** | **Ý nghĩa** | **Ràng buộc** |
| 1 | id | int | ID ứng dụng, tăng tự động | Primary Key |
| 2 | name | Varchar(256) | Tên hệ thống sinh tự động | Not null, Unique key |
| 3 | Original\_name | Varchar(256) | Tên gốc của dứng dụng | Not null |
| 4 | Local\_path | Varchar(256) | Địa chỉ lưu trữ video | Not null |
| 5 | Size\_byte | Int | Kích thước của video | Not null |
| 6 | Duration | Double | Thời lượng của video | Not null |
| 7 | Resolution | Varchar(50) | Độ phân giải của video | Not null |
| 8 | Result | Varchar(12) | Kết quả dự đoán tính bạo lực. | Default: undefined |
| 9 | User\_id | int | Mã người dùng tải video | FK |
| 10 | Application\_id | Int | Mã ứng dụng tải video | FK |
| 11 | Created\_at | Date time with timezone | Thời gian tải video lên hệ thống | Not null |
| 12 | Deleted\_at | Date time with timezone | Thời gian xóa video | Default null |

## Mô hình nhận diện tính bạo lực

Để xây dựng chức năng nhận diện bạo lực trong hình ảnh/video, hệ thống sử dụng kiến trúc mạng tích chập (Convolutional Neural Network) và LSTM hai chiều (Bidirectional Long-Short Term Memory) để dự đoán tính bạo lực (violence) trong dữ liệu có tính tuần tự của các frame ảnh trích xuất từ video.

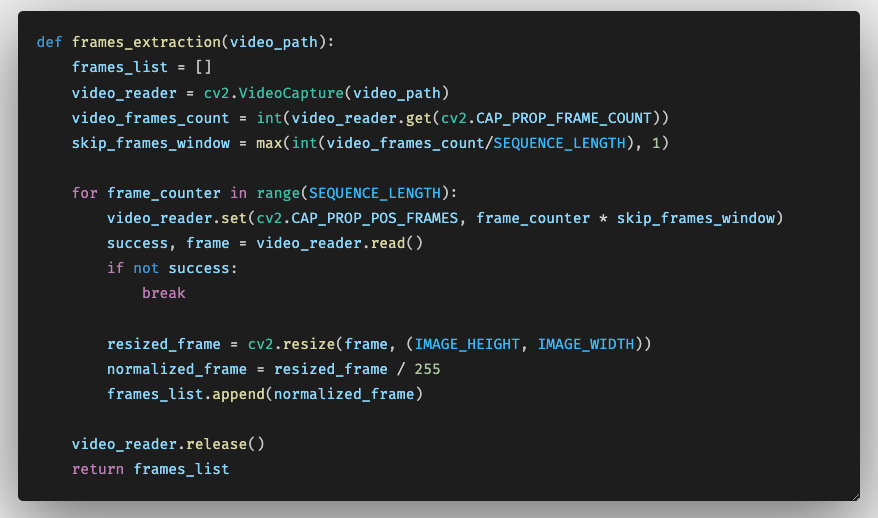
Với mỗi video, ta sẽ chia thành 16 frame ảnh tuần tự đại diện cho video đó, trong đó mỗi ảnh có kích thước 64 × 64. Sau đó, truyền các frame ảnh này vào mạng tích chập để trích xuất thông tin có trong frame ảnh đó. Tiếp theo, ta sẽ sử dụng LSTM hai chiều để so sánh thông tin giữa frame hiện tại với frame liền trước và liền sau nó để đánh giá luồng dữ liệu tuần tự. Cuối cùng, sử dụng bộ phân loại để thực hiện đánh giá tính bạo lực trong video.

Tập dữ liệu gồm hai thư mục chính:

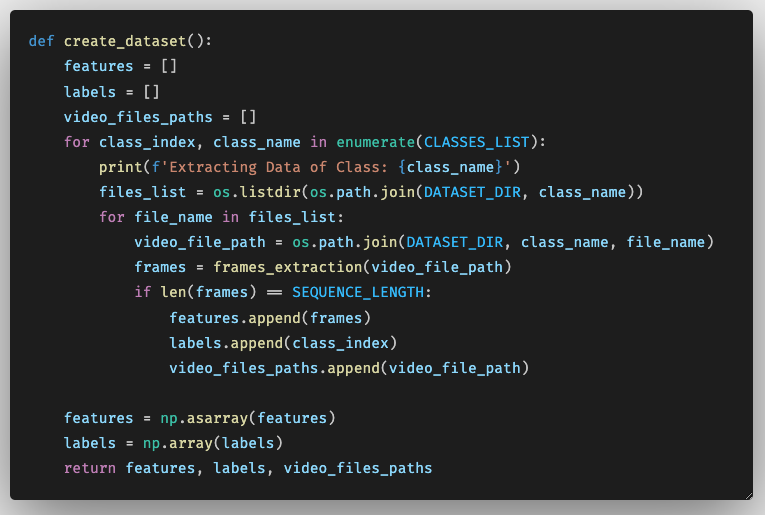
* + - Violence: Chứa 1000 video ở định dạng mp4 chứa nội dung bạo lực dưới nhiều hoàn cảnh khác nhau để làm dữ liệu training cho model.
    - NonViolence: Chứa 1000 video ở định dạng mp4 chứa nội dung không bạo lực dưới nhiều hoàn cảnh khác nhau để làm dữ liệu training cho model.

**Tiền xử lí dữ liệu:**

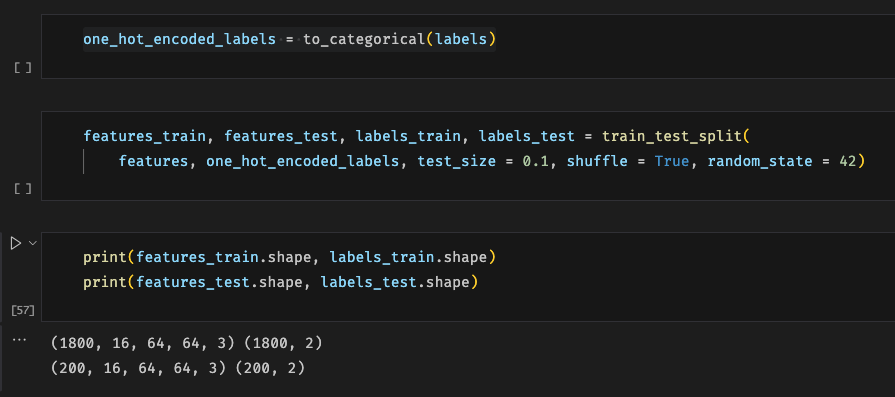
Trước khi tạo tập dataset cho model ta sẽ đi qua bước tiền xử lí dữ liệu. Mỗi video sẽ được trích xuất thành các frame ảnh tuần tự. Mỗi frame ảnh này sẽ được điều chỉnh kích thước và chuẩn hóa lại.



Hàm ***frame\_extraction()*** ở nhận đầu vào là đường dẫn đến video và trả về danh sách các frame tương ứng đã được điều chỉnh kích thước và chuẩn hóa.



Hàm ***create\_dataset()*** dựa trên hai nhãn “Violence” và “NonViolence” để đọc danh sách các video thuộc nhãn tương ứng. Sau đó với mỗi video gọi hàm ***frame\_extraction()*** để lấy danh sách các frame của video tương ứng và sau đó thực hiện đánh index (phân loại video) và lưu danh sách các feature và đường dẫn của video tương ứng.



Sau khi tạo được tập dataset, ta dựa trên đó chia thành dataset thành hai tập train và tập test. Với 90% số video sẽ dành để train và 10% còn lại để test.

### CNN với MobineNetV2

MobileNetV2 là một kiến trúc mạng nơ-ron tích chập (CNN) hiệu quả cao, được thiết kế đặc biệt để chạy trên các thiết bị di động và các thiết bị có tài nguyên tính toán hạn chế. Được giới thiệu bởi nhóm Google AI vào năm 2018, MobileNetV2 cải tiến so với phiên bản đầu tiên (MobileNetV1) bằng cách tăng hiệu quả tính toán mà vẫn giữ được độ chính xác cao trong các tác vụ nhận dạng và phân loại hình ảnh.

**Các đặc điểm chính của MobileNetV2:**

**Inverted Residuals (Phần dư nghịch đảo)**:

* + - MobileNetV2 sử dụng các khối **residual** (phần dư), nhưng với kiến trúc nghịch đảo. Thay vì mở rộng số lượng kênh đầu vào rồi co lại như trong ResNet, nó mở rộng số kênh ở giữa và giảm lại ở đầu ra. Điều này giúp giảm số lượng tham số và chi phí tính toán, đồng thời tăng khả năng khái quát hóa của mô hình.
    - Mỗi khối bao gồm ba lớp: một lớp tích chập 1x1 để mở rộng số lượng kênh, một lớp tích chập độ sâu riêng (depthwise convolution) để lọc, và cuối cùng là một lớp tích chập 1x1 khác để giảm số kênh.

1. **Depthwise Separable Convolution (Tích chập phân tách theo chiều sâu)**:
   * + Giống MobileNetV1, MobileNetV2 cũng sử dụng **depthwise separable convolutions** để giảm số lượng tham số. Thay vì áp dụng tích chập toàn bộ, depthwise separable convolution tách phần tích chập thành hai bước: tích chập trên mỗi kênh riêng lẻ (depthwise convolution) và tích chập gộp các kênh lại (pointwise convolution).
     + Kỹ thuật này giúp giảm số lượng tính toán đáng kể so với các tích chập thông thường mà vẫn giữ được khả năng học đặc trưng tốt.
2. **Linear Bottleneck (Nút thắt cổ chai tuyến tính)**:
   * + Một đặc điểm mới của MobileNetV2 là lớp tuyến tính ở đầu ra của mỗi khối phần dư. Thay vì sử dụng hàm kích hoạt phi tuyến như ReLU, MobileNetV2 sử dụng lớp tuyến tính để giữ nguyên cấu trúc thông tin. Điều này giúp giảm bớt hiện tượng mất thông tin trong các mạng sâu.

**Lợi ích của MobileNetV2:**

* **Tiết kiệm tài nguyên**: Với kiến trúc nhẹ, MobileNetV2 phù hợp cho các ứng dụng trên thiết bị di động, hệ thống nhúng và các thiết bị IoT.
* **Hiệu quả cao**: Mặc dù tiết kiệm tài nguyên, MobileNetV2 vẫn duy trì được độ chính xác cao trong các tác vụ như phân loại ảnh, nhận diện đối tượng, và các ứng dụng học sâu khác.

**Khả năng mở rộng**: MobileNetV2 có thể được tinh chỉnh để phù hợp với các thiết bị và nhu cầu tính toán khác nhau bằng cách điều chỉnh các tham số như độ rộng (width multiplier) và độ phân giải của đầu vào.

**Tạo model MobileNetV2 sử dụng python:**



Đầu tiên ta import model MobileNetV2 từ thư viện keras. Sau đó ta khởi tạo model với các tham số *include\_top = False* và *weights = “imagenet”* vì ta chỉ sử dụng model để lấy feature dựa trên tập các trọng số đã được train trước đó (pre-trained) của tập dataset ImageNet. Sau đó, ta cài đặt lại 40 lớp (layer) cuối của model thành không train được. Điều này giúp giữ lại cấc feature đã học trước đó. Đây gọi là kĩ thuật transfer learning, là khi bắt đầu với một mô hình đã được train sẵn (pre-trained) và sau đó tinh chỉnh lại để cho phù hợp với các tác vụ mới.

### CNN kết hợp LSTM hai chiều

LSTM hai chiều (Bidirectional LSTM) là một phiên bản của mạng LSTM (Long Short-Term Memory) có khả năng học các mối quan hệ tuần tự trong dữ liệu theo cả hai hướng: từ quá khứ đến hiện tại và từ tương lai về hiện tại.

**Cách hoạt động của LSTM hai chiều:**

Trong một LSTM hai chiều, có hai lớp LSTM chạy song song:

* + - * Một lớp xử lý dữ liệu từ đầu đến cuối (forward LSTM).
      * Lớp còn lại xử lý dữ liệu từ cuối về đầu (backward LSTM).

Kết quả của hai lớp LSTM này được kết hợp (thường là bằng cách nối lại) để tạo ra một đầu ra duy nhất, chứa thông tin từ cả hai hướng.

**Lợi ích của LSTM hai chiều:**

* + - * **Hiểu ngữ cảnh tốt hơn**: Trong các bài toán như xử lý ngôn ngữ tự nhiên, nhận dạng giọng nói, hoặc phân tích video, việc có thông tin từ cả trước và sau vị trí hiện tại giúp mô hình hiểu rõ hơn về ngữ cảnh.
      * **Dự đoán chính xác hơn**: Khả năng nhìn về cả hai hướng cho phép LSTM hai chiều có được nhiều thông tin hơn so với LSTM thông thường, giúp cải thiện độ chính xác trong các tác vụ như phân loại chuỗi và dự đoán chuỗi.

**Xây dựng mô hình CNN kết hợp LSTM hai chiều với Python:**



Hàm ***create\_model()*** trên xây dựng mô hình kết hợp giữa MobileNetV2 đã tạo trước đó với LSTM hai chiều để phân loại các chuỗi hình ảnh.

**Xử lý chuỗi hình ảnh với mô hình MobileNet**:

* Sử dụng mô hình MobileNet được huấn luyện trước (pre-trained) để trích xuất đặc trưng từ từng khung hình trong chuỗi.
* Áp dụng MobileNet thông qua lớp TimeDistributed, đảm bảo rằng mỗi khung hình được xử lý riêng biệt, giúp tạo ra các đặc trưng cho từng khung hình.

**Phân tích chuỗi thời gian bằng LSTM hai chiều**:

* + - * Thêm lớp LSTM hai chiều (Bidirectional LSTM) để học và khai thác thông tin từ cả hai hướng của chuỗi hình ảnh (từ đầu đến cuối và ngược lại).
      * LSTM hai chiều giúp mô hình nắm bắt mối quan hệ tuần tự giữa các khung hình và hiểu được ngữ cảnh tổng quát.

**Lớp Dense và Dropout để học đặc trưng và giảm quá khớp**:

* Các lớp Dense với số lượng neuron giảm dần (256, 128, 64, 32) được sử dụng để học các đặc trưng phức tạp từ đầu ra của LSTM.
* Các lớp Dropout xen kẽ giúp ngăn chặn quá khớp bằng cách ngẫu nhiên loại bỏ một tỷ lệ phần trăm các neuron trong quá trình huấn luyện.

**Lớp đầu ra cho phân loại**:

* + - * Lớp cuối cùng là lớp Dense với hàm kích hoạt softmax, trả về xác suất cho mỗi lớp đầu ra dựa trên số lượng lớp cần phân loại (len(CLASSES\_LIST)).
      * Lớp này cho phép mô hình phân loại chuỗi hình ảnh vào các lớp mục tiêu.

Mô hình sẽ lấy đầu vào là danh sách các frame ảnh, trích xuất các feature (đặc trưng) bằng MobileNetV2, phân tích sự thay đổi đặc trưng theo chuỗi thời gian bằng LSTM hai chiều và sử dụng các lớp Dense để phân loại frame ảnh vào các lớp dự đoán.



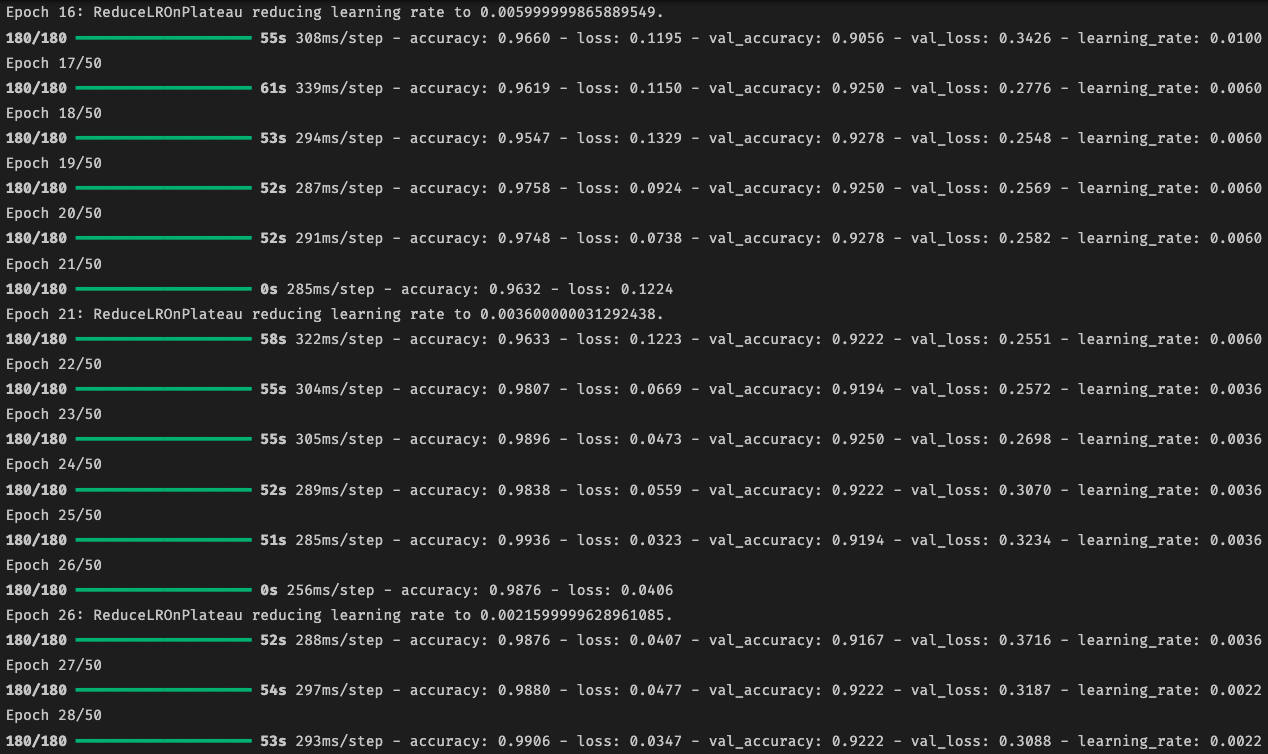
Trước khi thực hiện huấn luyện mô hình, ta định nghĩa hai hàm:

* + - * **Early\_stopping\_callback():** Dừng sớm quá trình huấn luyện nếu mô hình không còn cải thiện về độ chính xác trên tập validation.
      * **Reduce\_lt():** Tự động giảm tốc độ học nếu mất mát trên tập validation không được cải thiện sau những lần huấn luyện.

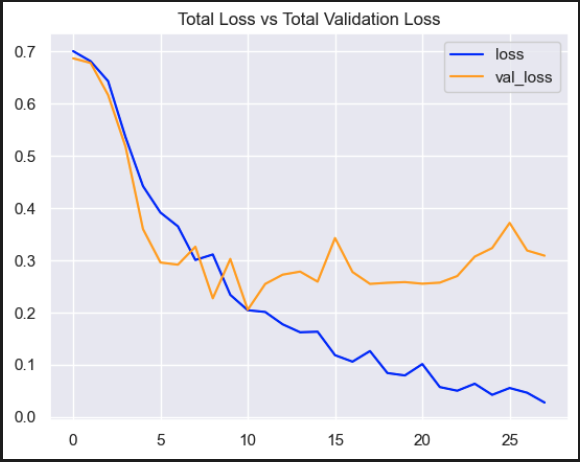
Sau đó ta thực hiện cấu hình mô hình với:

* + - * Hàm mất mát: “categorical\_crossentropy” là hàm mất mát dùng cho bài toán phân loại nhiều lớp.
      * Thuật toán tối ưu: Stochastic Gradient Descent (SGD).
      * Và có theo dõi độ chính xác trong quá trình huấn luyện.

Cuối cùng ta huấn luyện mô hình với tập train đã chia, 50 epochs với 8 mẫu mỗi batch. Mỗi epoch sẽ xáo trộn dữ liệu huấn luyện và sử dụng 20% dữ liệu huấn luyện cho tập validation. Đồng thời sử dụng các hàm callback đã định nghĩa.

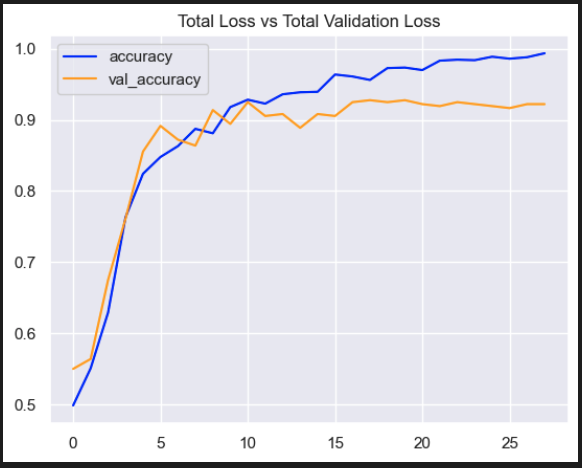


**Đánh giá kết quả huấn luyện:**



Dựa trên biểu đồ trên ta thấy:

* + - * Mất mát trên tập huấn luyện (loss) giảm đều qua các epoch, cho thấy mô hình đang học và tối ưu dần dần.
      * Mất mát trên tập validation (val\_loss) giảm trong giai đoạn đầu nhưng sau đó dao động và có dấu hiệu tăng lên ở các epoch cuối.
      * Hiện tượng này có thể là dấu hiệu của **overfitting** khi mô hình hoạt động tốt trên tập huấn luyện nhưng kém hiệu quả hơn trên tập validation.



Dựa trên biểu đồ ta thấy:

* + - * Độ chính xác trên tập huấn luyện (accuracy) tăng dần và đạt gần mức 1.0, cho thấy mô hình đã học tốt trên tập huấn luyện.
      * Độ chính xác trên tập validation (val\_accuracy) cũng tăng nhanh ở giai đoạn đầu và sau đó ổn định ở mức khoảng 0.9, nhưng không tiếp tục cải thiện.
      * Khoảng cách giữa accuracy và val\_accuracy ở các epoch cuối có thể là dấu hiệu của **overfitting**.

Model sau khi huấn luyện xong có thể lưu lại để dự đoán tính bạo lực.



# Triển khai hệ thống

## Môi trường phát triển

Để xây dựng hệ thống nhận diện video có tính bạo lực, em sử dụng các công cụ và thư viện cần thiết như sau:

* IDE PyCharm 2024.1.4 (Professional Edition).
* Ngôn ngữ lập trình Python phiên bản 3.12.
* Framework Django phiên bản 5.1.4.
* Thư viện cryptography phiên bản 44.0.0.
* Thư viện uuid.
* Thư viện ffmpeg-python phiên bản 0.2.0.
* Thư viện numpy phiên bản 2.0.2
* Thư viện opencv-python phiên bản 4.10.0.84.
* Thư viện tensorflow phiên bản 2.18.0.
* Mô hình MobileNetV2 và LSTM.

Source code của hệ thống và mô hình được sử dụng trong khi viết báo cáo có thể tìm được ở trang GitHub sau: [Violence Prediction Service | nnminh-sam | GitHub](https://github.com/nnminh-sam/violence-prediction-service).

Dataset được sử dụng để huấn luyện mô hình: [Real Life Violence Situations Dataset](https://www.kaggle.com/datasets/mohamedmustafa/real-life-violence-situations-dataset).

## Các bước triển khai

**Yêu cầu:**

* Ngôn ngữ lập trình Python với phiên bản mới nhất.
* Hệ quản trị cơ sở dữ liệu Postgres.

### Tạo dự án Django và thiết lập các cấu hình cần thiết

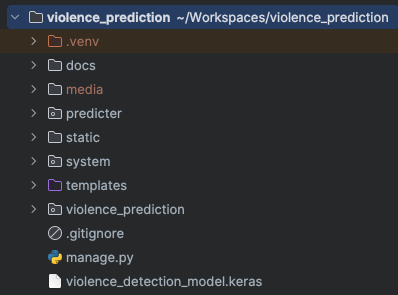
**Tạo dự án Django mới:**

Sử dụng phần mềm PyCharm, ta tạo một dự án mới như sau:



Điền tên dự án mong muốn vào ô “Name” và chọn thư mục lưu dự án ở ô “Location”. PyCharm sẽ tự động tìm kiếm các phiên bản Python đang có sẵn trong hệ thống và tự động chọn phiên bản phù hợp. Các tùy chỉnh cài đặt còn lại giữ nguyên như hình trên.

Sau khi PyCharm tạo dự án xong, ta có thể thấy được cấu trúc cơ bản của một dự án Django sẽ như hình sau:



Dự án sẽ được đặt tại địa chỉ thư mục đã chọn và có tên là “test\_project” (ví dụ). Bên trong thư mục của dự án sẽ có một file được dùng để tương tác với hệ thống là “manage.py”. Đây là file chứa các lệnh để tương tác với dự án Django ta vừa tạo. Một số tương tác chính thường sử dụng bao gồm: chạy server, migrate database, tạo migration, tạo tài khoản superuser, tạo Django app cho dự án.

Ngoài file “manage.py” cũng có một thư mục trùng tên với tên dự án chứa một số file cài đặt của dự án Django. Trong đó:

* File “settings.py” sẽ chứa những cài đặt chung cho toàn bộ dự án.
* File “urls.py” sẽ chứa những cấu hình routing của toàn bộ dự án. File sẽ import những cài đặt cấu hình routing của các app nhỏ trong dự án và đưa vào cấu hình chính của toàn bộ dự án. Đây là bước bắt buộc.
* File “asgi.py” và “wsgi.py” là hai file cấu hình giao thức cho dự án. Vì dự án không có yêu cầu gì đặc biệt nên hai file này sẽ được giữ nguyên cấu hình mặc định.

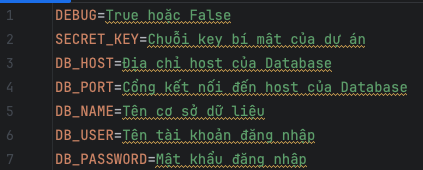
Hệ thống được xây dựng dựa theo kiểu kiến trúc Server Side Rendering, nghĩa là server sẽ trả về trang HTML cho trình duyệt, do đó ta sẽ đặt các bản thiết kế giao diện (file “.html”) ở trong thư mục “template”.

Các Django app của dự án sẽ được lưu trữ trong thư mục cùng cấp với thư mục cài đặt của dự án. Bên cạnh đó một số file và thư mục ngoài cũng được để bên trong thư mục của toàn bộ dự án như hình đã mô tả.

**Cấu hình cần thiết để xây dựng hệ thống:**

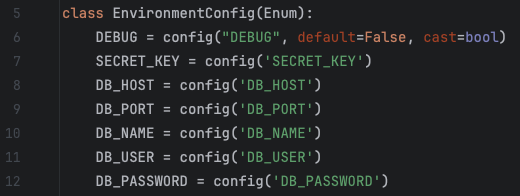
*Biến môi trường (file “.env”) cho dự án:*

Mặc định khi tạo dự án Django thì một số thông tin về bảo mật của dự án sẽ được cấu hình trực tiếp bên trong file “settings.py”. Khi chia sẽ code lên GitHub hay các nền tảng khác sẽ dễ làm rò rỉ thông tin bảo mật. Để hạn chế vấn đề này, ta sẽ tách những thông tin bảo mật đó thành một file “.env” để lưu các cấu hình môi trường cho dự án. Cấu hình môi trường mẫu cho dự án sẽ như sau:

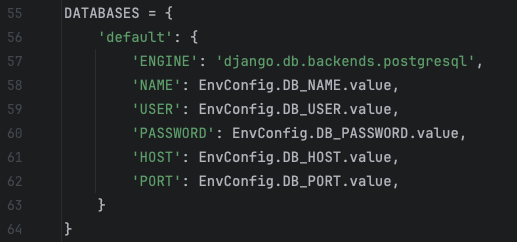


*File ví dụ của dự án:* [*.env*](https://github.com/nnminh-sam/violence-prediction-service/blob/main/violence_prediction/.env)*.*

Sau đó ta sẽ sử dụng thư viện “config” để đọc các giá trị tương ứng từ file “.env” và tạo thành enum class để sử dụng cấu hình file “settings.py”.

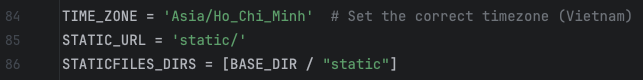


Khi sử dụng Enum class này, ta có thể sử dụng như hình dưới:

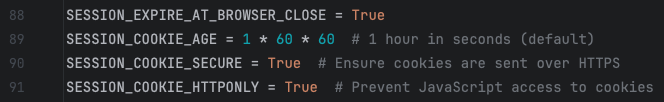


Trong hình, ta đang sử dụng những cấu hình từ enum class “EnvConfig” để cài đặt cấu hình kết nối đến hệ quản trị cơ sở dữ liệu Postgres. Vì không thay đổi hệ quản trị cơ sở dữ liệu nên giá trị của thuộc tính “ENGINE” sẽ được gán cố định.

**Một số cấu hình khác của hệ thống:**



Đây là cấu hình timezone của dự án để hiển thị thời gian chính xác tại Việt Nam. Và cấu hình thư mục “static” để lưu trữ các file “.css” của dự án.



Đây là cấu hình cần thiết về Session và Cookie cho dự án để đảm bảo tính bảo mật.



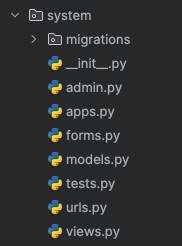
Đây là cấu hình địa chỉ tương đối của thư mục lưu các video sẽ được tải lên hệ thống. Vị trí đường dẫn tuyệt đối sẽ là “*<đường dẫn đến thư mục dự án>/media/*” và các file được tải lên sẽ đặt trong thư mục “uploads” ở trong thư mục “media”.

Sau khi hoàn thành các cài đặt này, và đảm bảo hệ quản trị cơ sở dữ liệu tồn tại như đã cài đặt thì ta có thể chạy dự án bằng PyCharm hoặc bằng cửa sổ dòng lệnh với câu lệnh “runserver”. Sau đó, ta có thể chạy lệnh “makemigrations” và “migrate” để Django tạo các bảng liên quan đến quản lí người dùng hệ thống, phân quyền và quản lí log mặc định cho hệ thống. Ta không cần định nghĩa thủ công các bảng này mà Django đã định nghĩa trước cho ta.

### Tạo chức năng người dùng đăng nhập và đăng kí tài khoản

Mặc định, Django đã cung cấp đầy đủ từ giao diện đến chức năng và bảng để ta quản lí người dùng hệ thống. Tuy nhiên, mặc định chỉ có quản lí hệ thống (superuser) mới có thể tạo người dùng mới. Do đó, ta cần xây dựng giao diện và chức năng đăng kí và đăng nhập cho hệ thống.

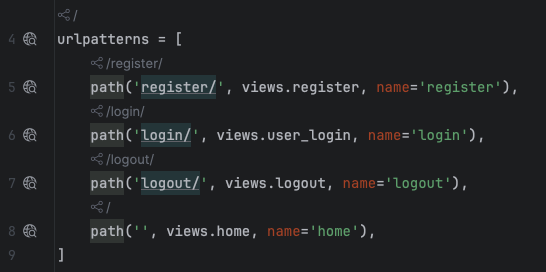
Ta tạo một Django app mới tên gọi là “system” để lưu trữ các view đăng kí và đăng nhập.



Trong đó, ta sẽ:

* Cấu hình app ở file “apps.py”. Xem file ở: [/system/apps.py](https://github.com/nnminh-sam/violence-prediction-service/blob/main/system/apps.py).
* Cấu hình routing cho app “system” ở file “urls.py”. Xem file ở: [/system/urls.py](https://github.com/nnminh-sam/violence-prediction-service/blob/main/system/urls.py).
* Viết logic view ở file “views.py”. Xem file ở: [/system/views.py](https://github.com/nnminh-sam/violence-prediction-service/blob/main/system/views.py).

Khi sử dụng PyCharm, ta sẽ được gợi ý về các routing cho các view mà ta đã cài đặt như hình bên dưới. Do đó, ta sẽ dễ dàng quản lí được route sẽ có tên gì và trả về view nào.



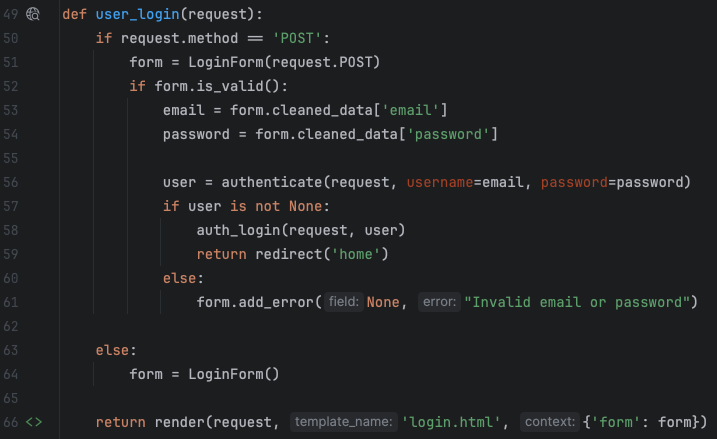
Ta sẽ tạo bốn view: Đăng kí, Đăng nhập, Đăng xuất và Trang chủ. Trong đó hai view Đăng xuất và Trang chủ chỉ được truy cập sau khi đăng nhập.

**View Đăng kí tài khoản người dùng:**



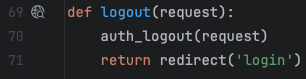
Ta sử dụng “form” của Django để có thể dễ dàng tạo form đăng kí tài khoản và xác thực các trường dữ liệu như email, password, first\_name và last\_name. Sau đó ta có thể lưu thông tin tài khoản mới vừa đăng kí vào cơ sở dữ liệu và truyển người dùng sang Trang chủ.

**View Đăng nhập với tài khoản đã đăng kí:**



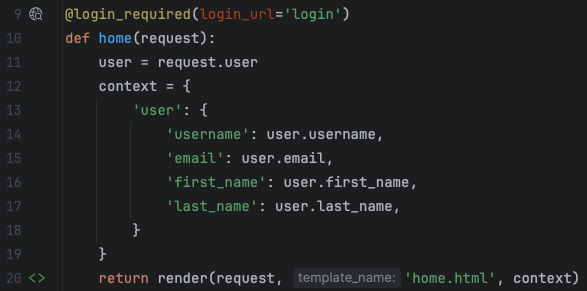
Khi đăng nhập, ta cũng tạo form đăng nhập để xác thực dữ liệu. Sau đó ta xác thực người dùng với API được cung cấp bởi Django.

**View Đăng xuất tài khoản:**

****

Ta sử dụng hàm logout mặc định do API của Django cung cấp để thực hiện đăng xuất và chuyển người dùng sang trang đăng nhập.

**View Trang chủ:**

****

Sau khi đăng nhập, ta lấy những thông tin cần thiết như email, họ tên, username để truyền vào view để Django render ra HTML và trả về cho người dùng.

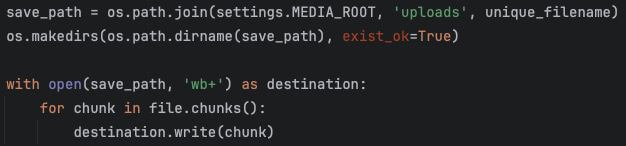
### Tạo chức năng tải video và nhận diện tính bạo lực trong video

Hàm xử lí file video khi file được tải lên hệ thống sẽ thực hiện các bước sau:

1. Tạo tên file ngẫu nhiên cho video.



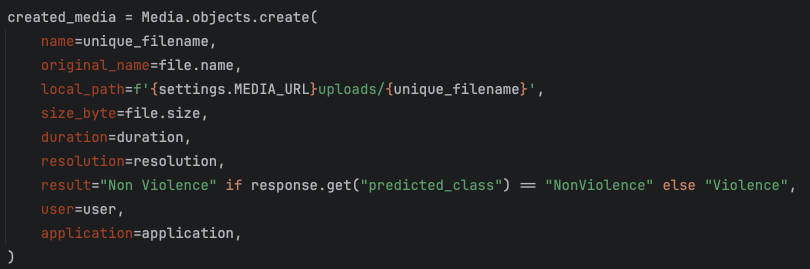
1. Lưu file vào hệ thống.



1. Dự đoán tính bạo lực trong video.



1. Lưu thông tin file vào cơ sở dữ liệu.



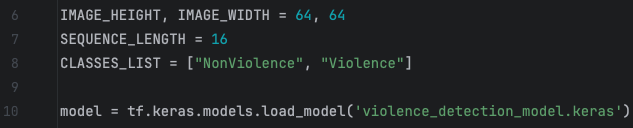
**View tài video lên hệ thống:**

****

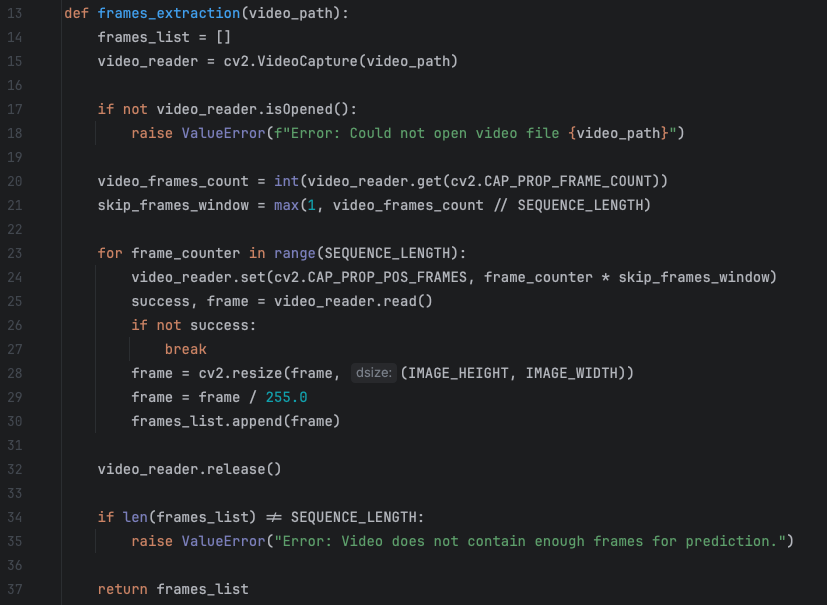
Để truy cập được view này, người dùng cần phải đăng nhập vào hệ thống trước đó. Khi đó ta sẽ lấy danh sách các video đã gửi lên trước đó. Nếu view nhận được một yêu cầu có phương thức POST và có file video thì sẽ thực hiện lời gọi đến hàm xử lí video. Khi video được xử lí thành công thì view sẽ trả về kết quả tương ứng hoặc trả về lỗi nếu có.

**Dịch vụ dự đoán tính bạo lực bằng mô hình đã huấn luyện:**

Trước tiên ta cần định nghĩa một số hằng số cần thiết như kích thước frame dữ liệu, chiều dài chuỗi video, các nhãn và load mô hình đã huấn luyện.

****

Sau đó ta sẽ thực hiện tách dữ liệu từ video thành các frame ảnh và biến đổi các frame ảnh sao cho trùng hợp với các định nghĩa hằng số đã đặt ra trước đó để phù hợp với mô hình đã huấn luyện trước khi đưa vào dự đoán.

****

Khi muốn thực hiện dự đoán tính bạo lực của video, ta chỉ cần gọi lời gọi đến hàm dự đoán tính bạo lực.

****

Hàm sẽ trả về kết quả dự đoán cùng với tỉ lệ phần trăm dự đoán cho mỗi nhãn.

### Tạo chức năng đăng kí ứng dụng sử dụng dịch vụ API

**Hàm sinh cặp khóa bí mật và công khai:**



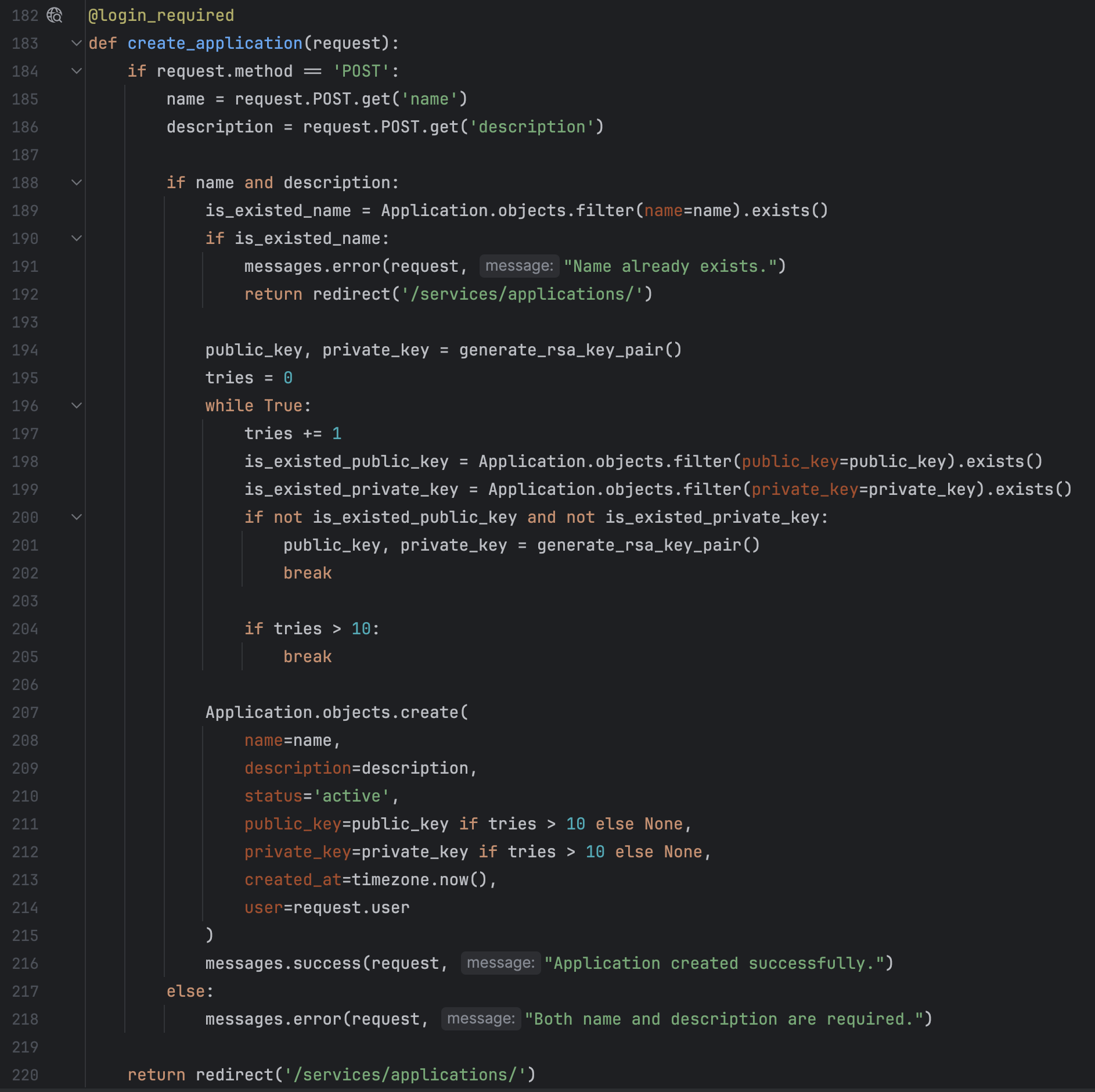
Cặp khóa bí mật và công khai sẽ được hệ thống sinh ra sử dụng thuật toán RSA. Sau đó chuyển sang định dạng PEM để dễ đọc và lưu trữ.

**Hàm xác thực cặp khóa bí mật và khóa công khai:**



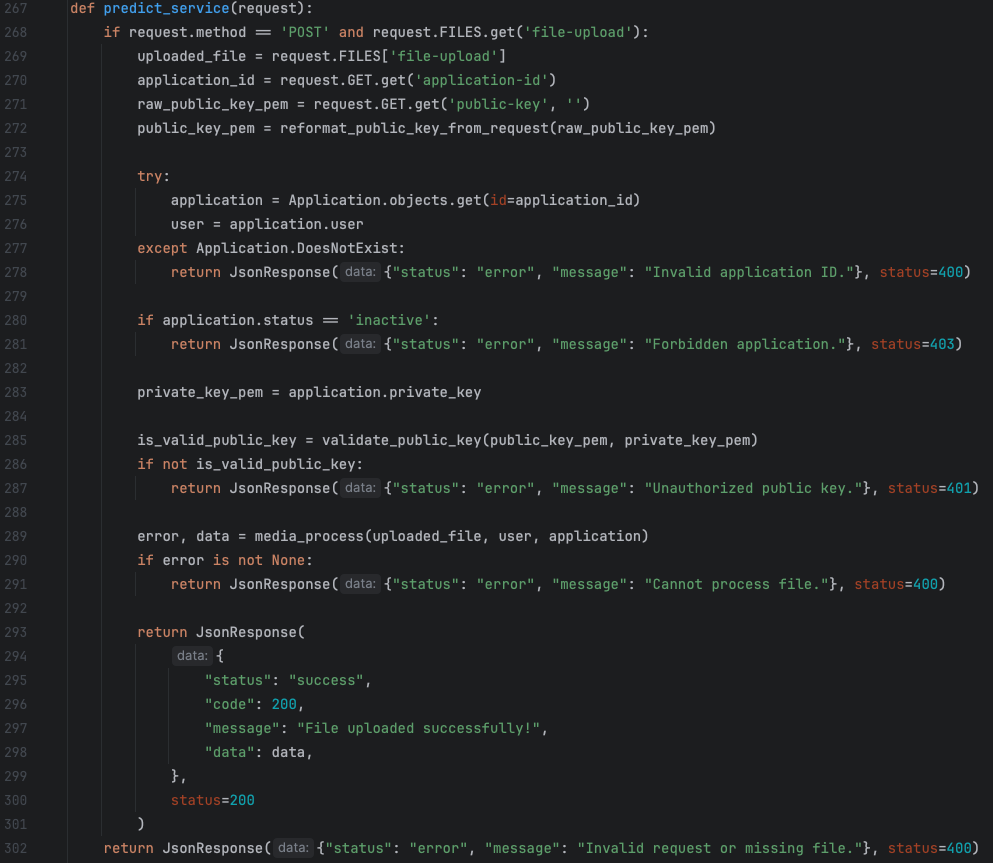
Để xác thực hai khóa có phải là một cặp hay không. Ta thực hiện mã hóa một đoạn tin nhắn bằng khóa bí mật mà hệ thống lưu trữ. Sau đó sử dụng khóa công khai mà hệ thống nhận được từ người dùng và giải mã tin nhắn đã mã hóa đó. Nếu nhận được tin nhắn giải mã tương ứng với tin nhắn gốc thì khóa công khai đó là hợp lệ, nhưng ngược lại khóa công khai đó sẽ không hợp lệ và xác thực thất bại.

**View tạo ứng dụng mới:**



Khi nhận được yêu cầu có phương thức POST, hệ thống sẽ lấy giá trị tham số name và description nhận được từ người dùng. Sau đó, thực hiện tạo cặp khóa bí mật và công khai rồi thực hiện tạo một ứng dụng mới vào cơ sở dữ liệu.

**Dịch vụ API nhận diện tính bạo lực:**



Khi nhận được phương thức POST với các dữ liệu về mã ứng dụng, khóa và file video thì hệ thống sẽ thực hiện xác thực khóa mà người dùng cung cấp, tìm ứng dụng có mã tương ứng, xác thực trạng thái hoạt động của ứng dụng rồi sau đó mới thực hiện quá trình xử lí video tương ứng như tải video lên hệ thống. Khi có kết quả, sẽ phản hồi lại cho người gọi API dưới định dạng JSON.

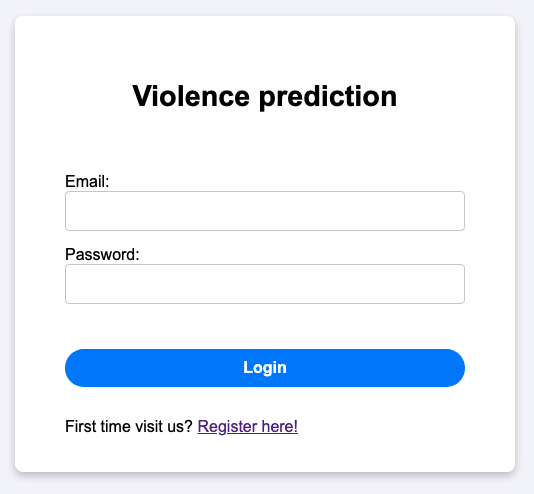
# Kết quả và đánh giá

## Kết quả đạt được

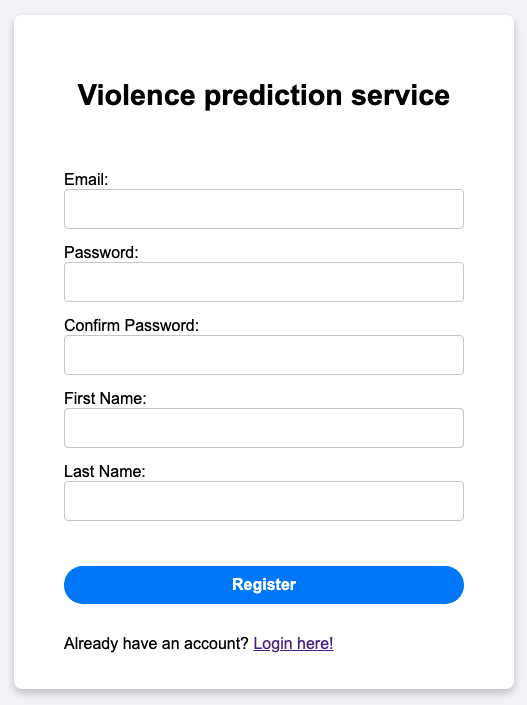
Sau quá trình thực hiện tìm hiểu framekwork Django và xây dựng dự án “hệ thống nhận diện video có tính bạo lực” thì em đạt được kết quả như sau:

1. Về framework Django:
   * Hiểu được nguyên lí thiết kế của framekwork Django.
   * Nắm được những ưu điểm và nhược điểm của framework.
   * Hiểu được những trường hợp có thể sử dụng framework.
   * Sử dụng được một số các đặc điểm nổi bật của framework Django để xây dựng dự án thực tế đã trình bày.
2. Về hệ thống nhận diện video có tính bạo lực:
   * Xây dựng được hệ thống nhận diện video có tính bạo lực đáp ứng được các yêu cầu cơ bản:
     1. Đăng kí/đăng nhập vào hệ thống.
     2. Quản lí thông tin người dùng bằng hệ thống quản lí người dùng tích hợp của Django.
     3. Tải video lên hệ thống và hệ thống nhận được được tính bạo lực có trong video.
     4. Xây dựng chức năng đăng kí ứng dụng ngoài và cung cấp dịch vụ API nhận diện video có tính bạo lực thông qua public key.

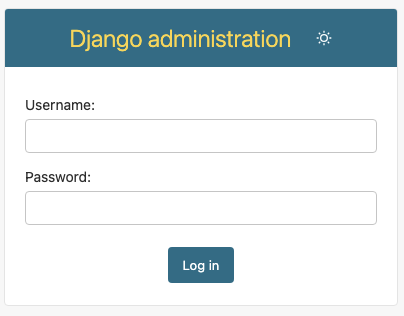
**Một số hình ảnh của giao diện hệ thống đã xây dựng:**

****

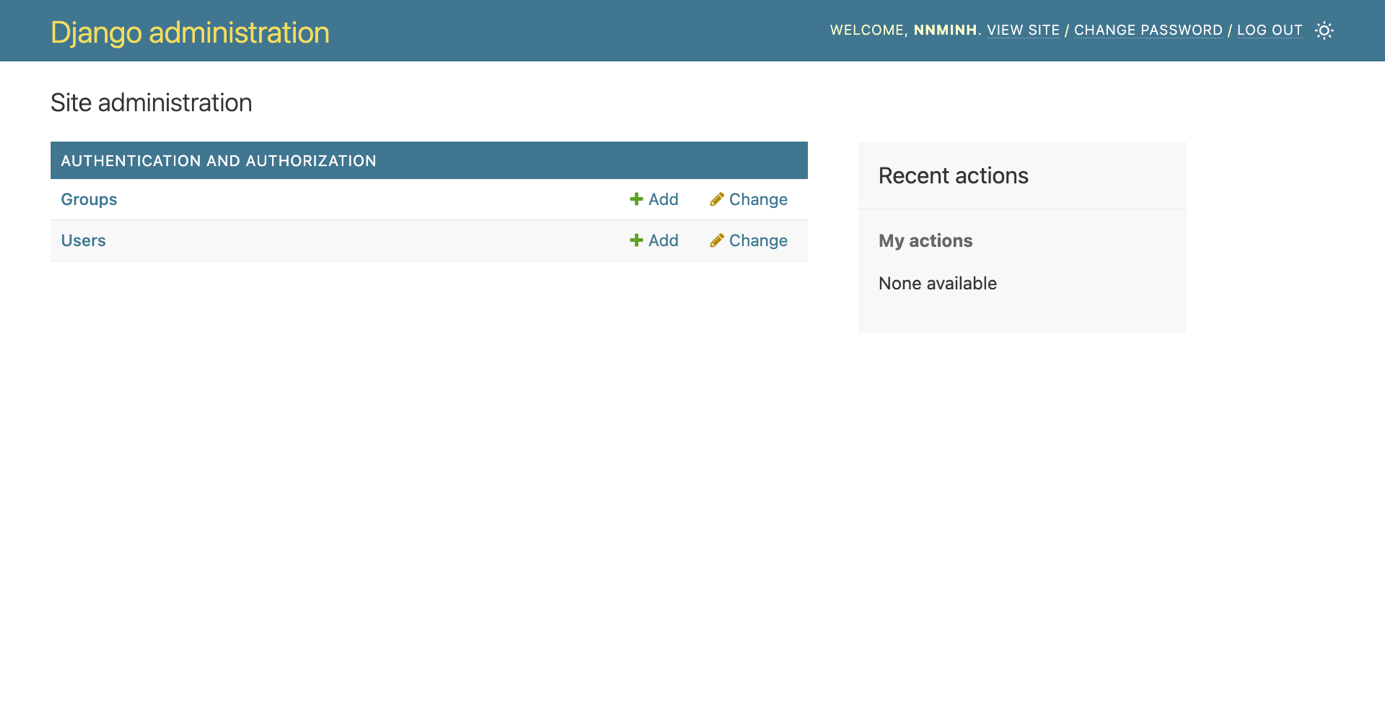
Hình : Giao diện đăng nhập



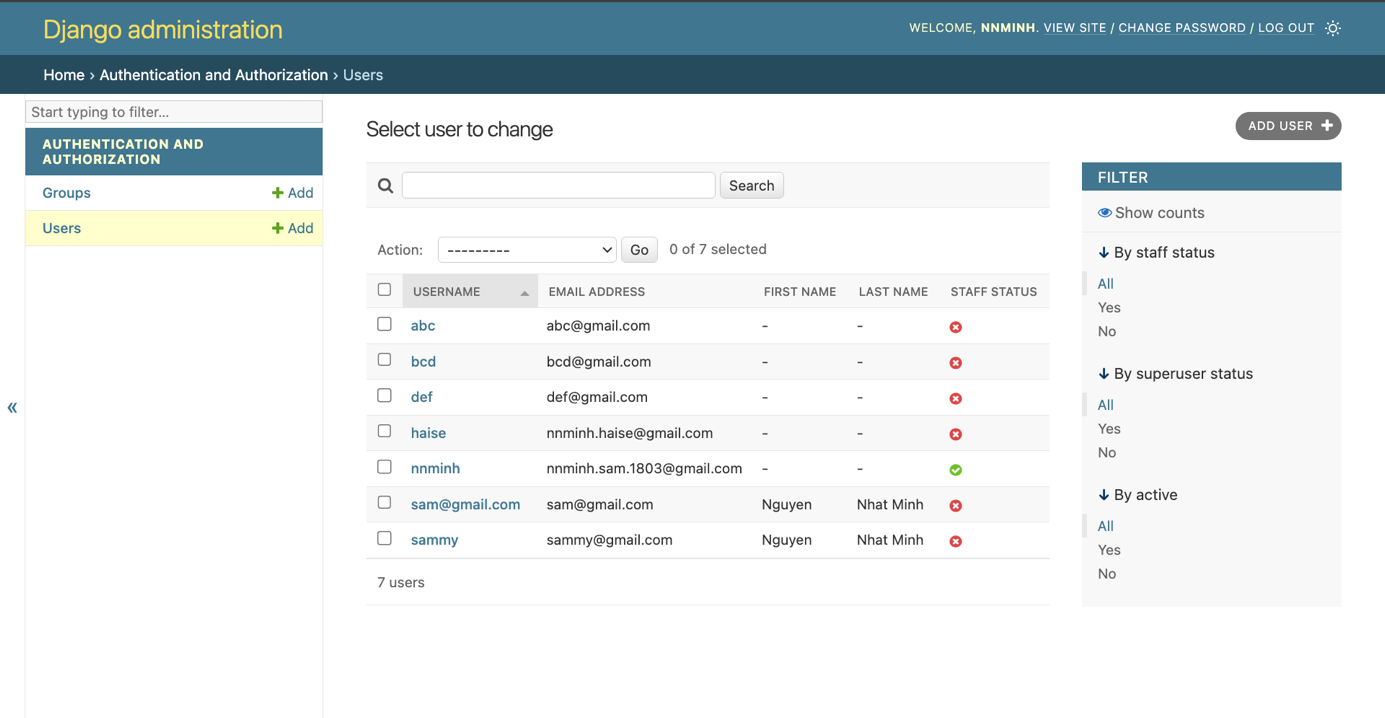
Hình : Giao diện đăng kí tài khoản



Hình : Giao diện đăng nhập tài khoản admin tích hợp của Django



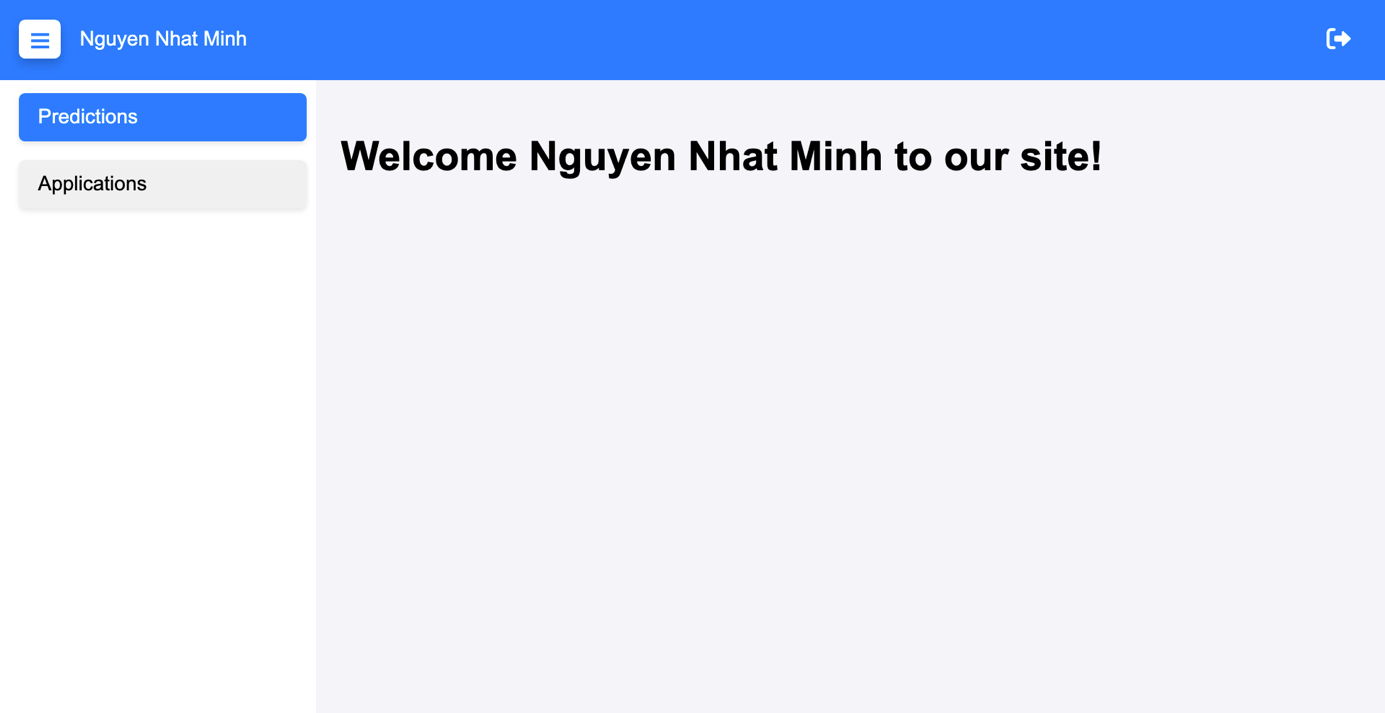
Hình : Giao diện chính sau khi đăng nhập tài khoản admin hệ thống



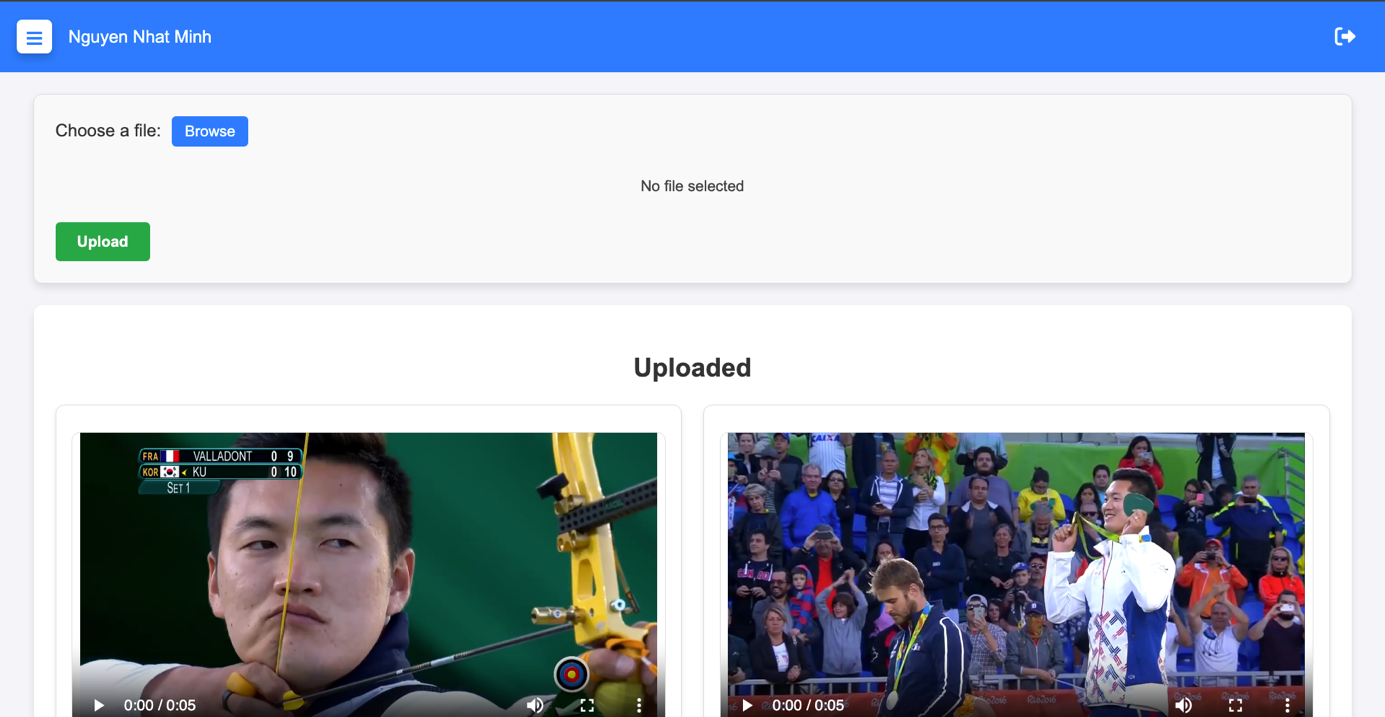
Hình : Giao diện quản lí người dùng hệ thống của Django

Các giao diện ở các hình 3, 4, 5 đều là những giao diện do framework Django cung cấp mặc định khi chạy hệ thống. Điều này giúp tiết kiệm thời gian tạo hệ thống quản lí người dùng khi sử dụng framework Django. Ta chỉ cần đăng kí tài khoản superadmin bằng cửa sổ dòng lệnh khi phát triển ứng dụng và sử dụng tài khoản đó để đăng nhập vào trang quản lí người dùng.

Trong trang này, admin có thể quản lí tài khoản người dùng hệ thống và quản lí phân quyền cho toàn bộ hệ thống. Tất cả những chức năng này đều được cung cấp mặc định bởi Django mà không cần cấu hình thêm.



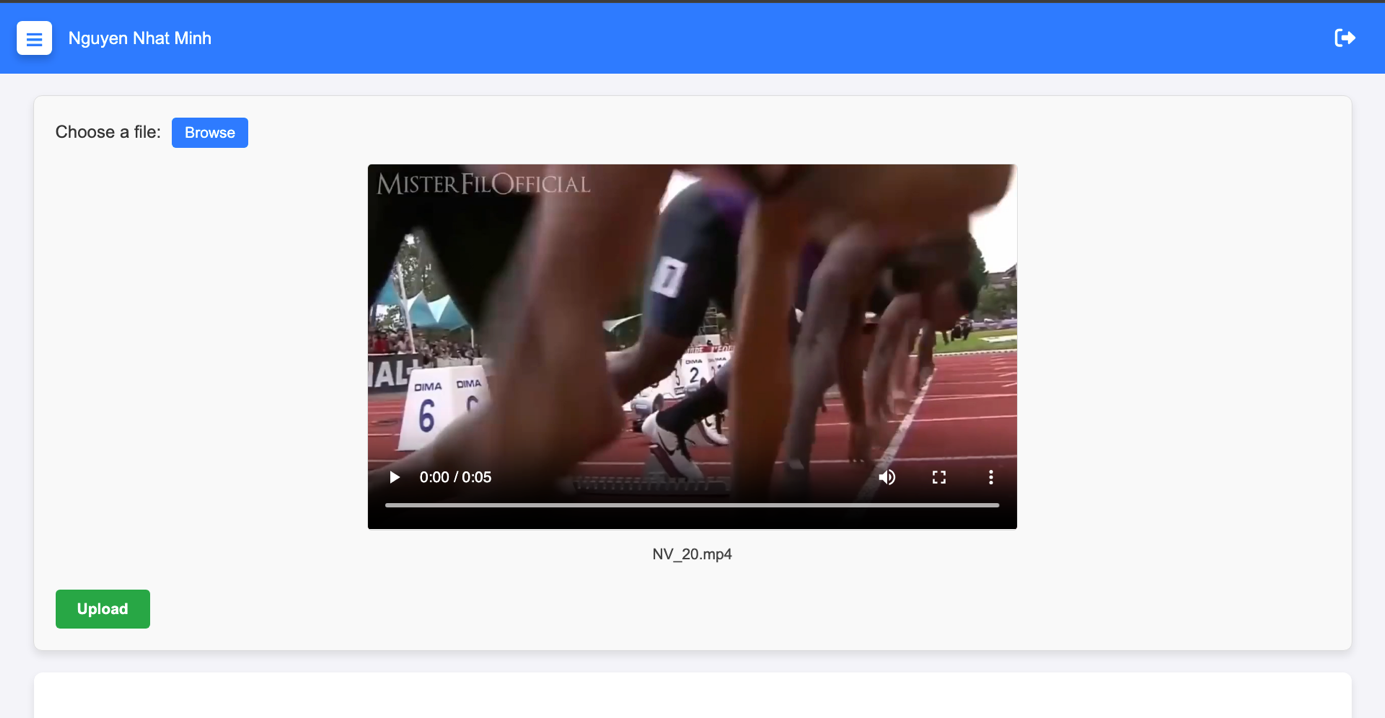
Hình : Giao diện trang chủ sau khi người dùng đăng nhập thành công



Hình : Giao diện chức năng nhận diện video có tính bạo lực

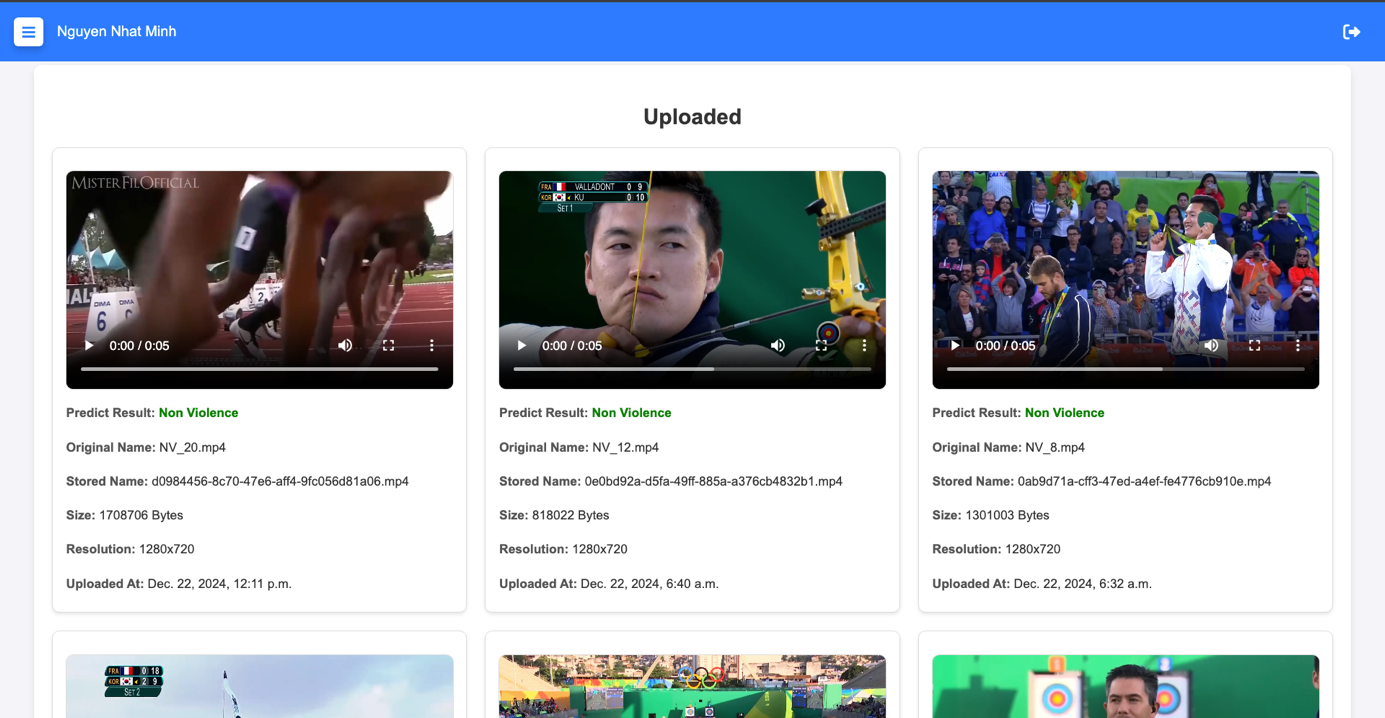
**Mô tả:**

* Người dùng sẽ bấm nút “Browse” để chọn file video muốn tải lên hệ thống.
* Người dùng có thể xem trước video trước khi xác nhận tải video lên hệ thống.



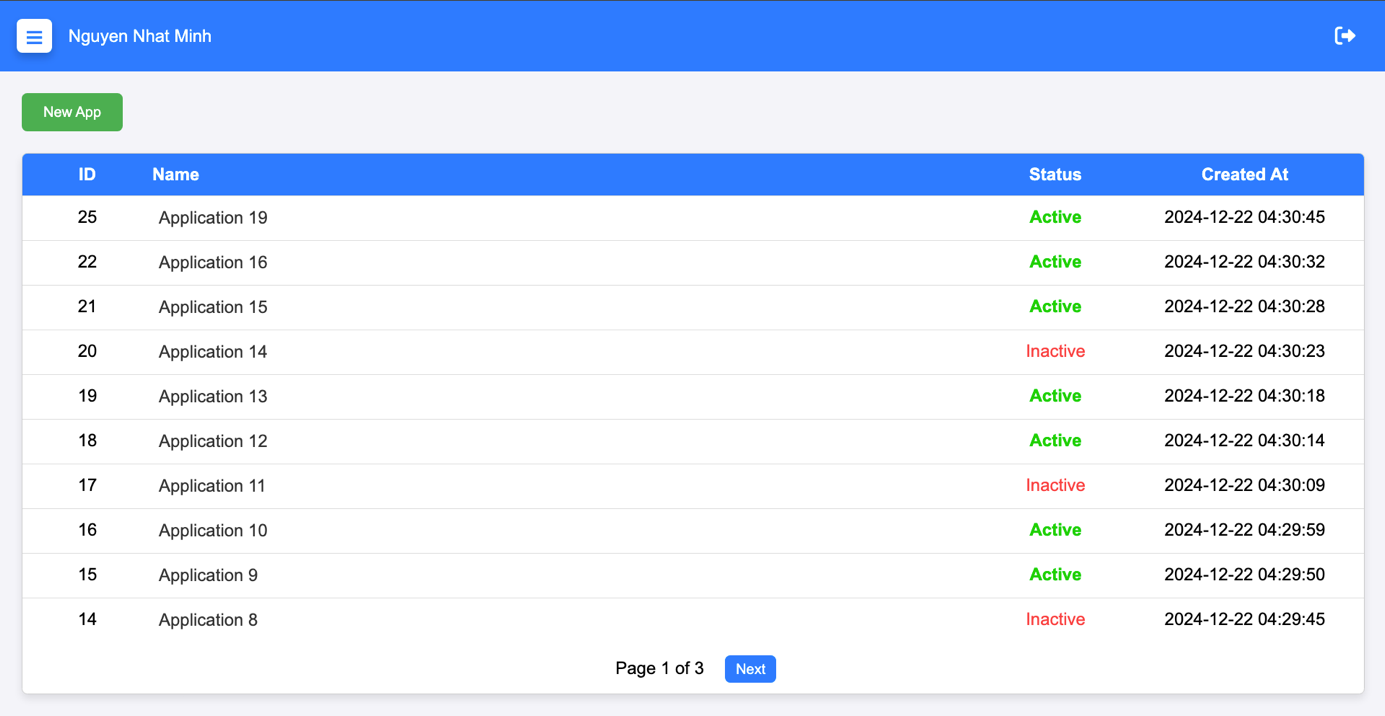
Hình : Giao diện xem trước video trước khi tải lên hệ thống

* Người dùng có thể xem trước video và kiểm tra video đã chính xác như mong muốn chưa dựa trên tên video được hiển thị bên dưới.
* Sau đó có thể bấm nút “Upload” để thực hiện gửi video lên hệ thống.



Hình : Giao diện xem danh sách các video đã gửi lên hệ thống và kết quả nhận diện tính bạo lực của video

Sau khi video được tải lên, hệ thống sẽ thực hiện dự đoán tính bạo lực có trong video để xác định video có tính bạo lực hay không và sau đó hệ thống sẽ trả về kết quả thu được. Người dùng có thể xem danh sách các video đã tải lên hệ thống theo thứ tự video đã tải lên gần đây nhất cùng với kết quả nhận diện.



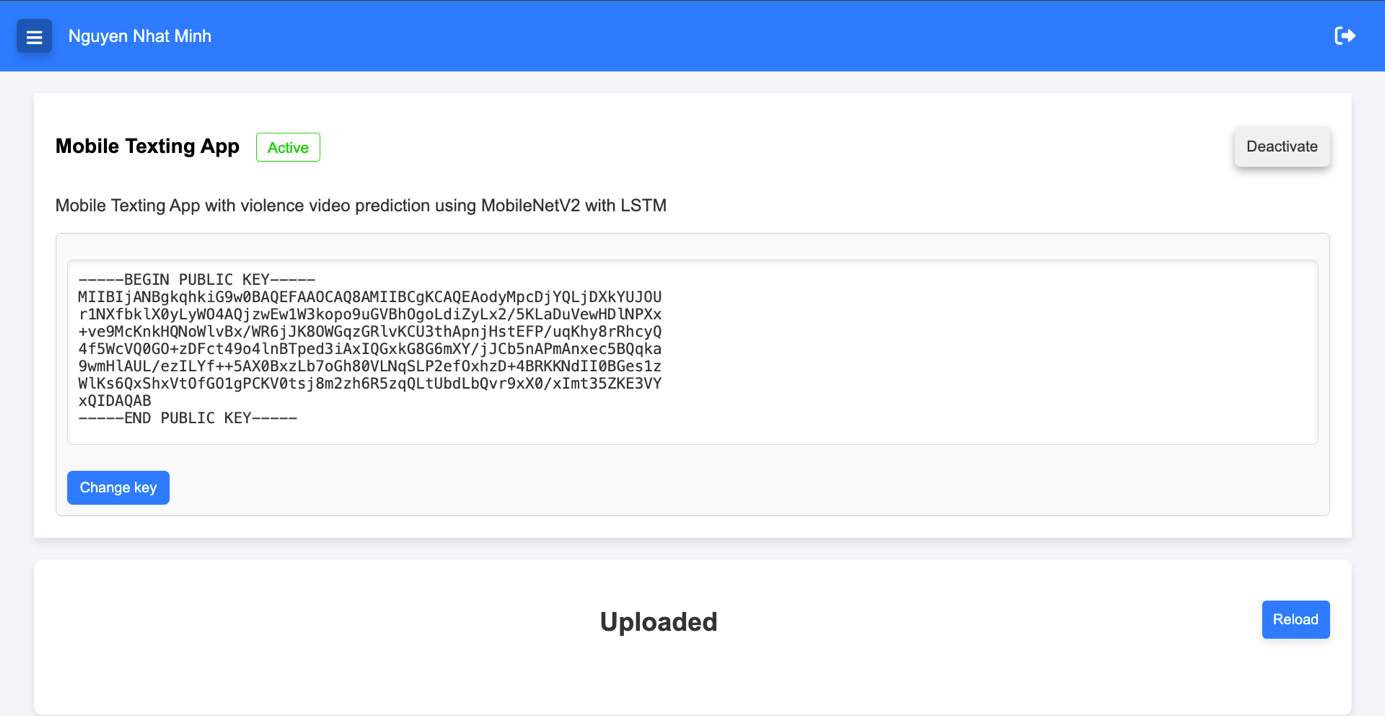
Hình : Giao diện danh sách các ứng dụng đã đăng kí sử dụng API nhận diện bạo lực

Người dùng có thể xem danh sách các ứng dụng đã đăng kí sử dụng dịch vụ API nhận diện tính bạo lực trong video và tạo ứng dụng mới bằng nút “New app”.



Hình : Giao diện tạo ứng dụng sử dụng dịch vụ mới

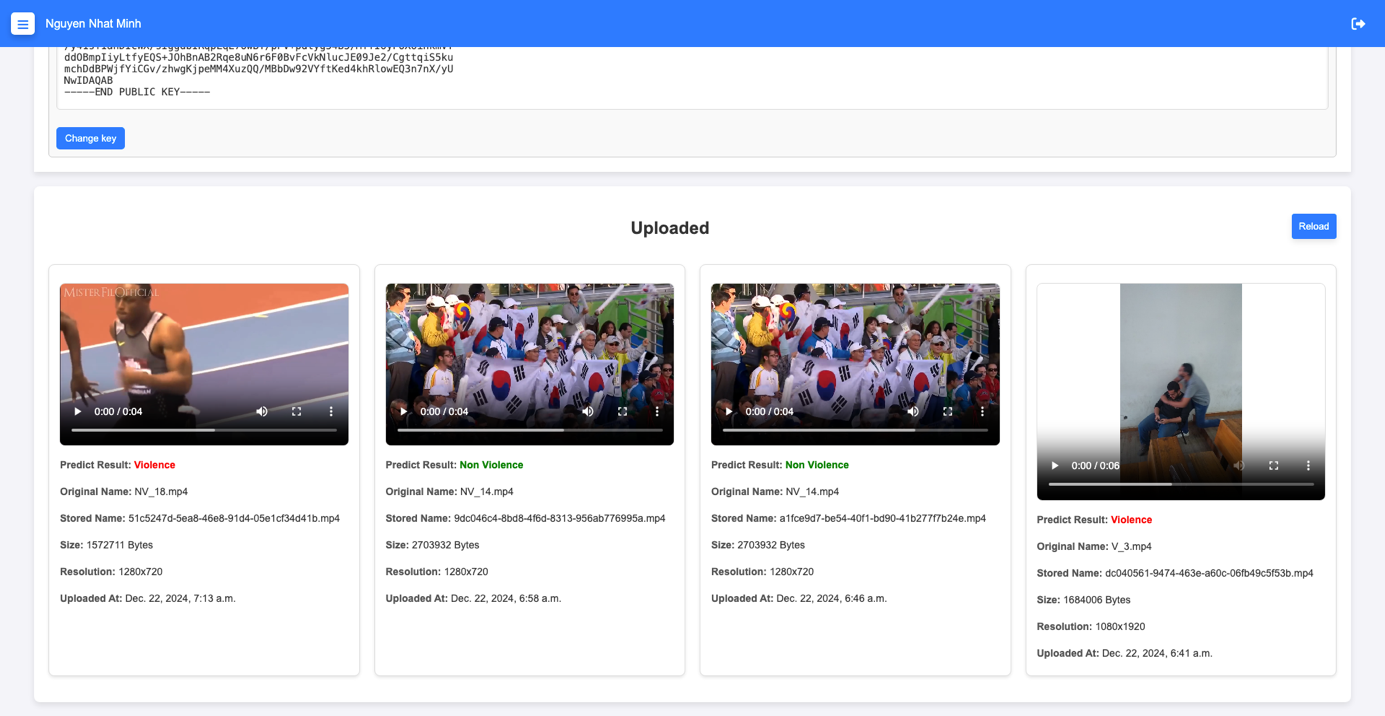
Một form đăng kí ứng dụng sử dụng dịch vụ mới sẽ được hiện ra và người dùng có thể điền tên và mô tả của ứng dụng đó.



Hình : Giao diện thông tin ứng dụng đã đăng kí và public key để sử dụng dịch vụ API

Sau khi ứng dụng được tạo người dùng có thể xem thông tin chi tiết ứng dụng đã đăng kí. Trong trang chi tiết này, người dùng có thể khóa ứng dụng (Deactivate) bằng cách sử dụng nút “Deactivate” ở bên góc phải hoặc mở khóa ứng dụng bằng nút “Activate” ở vị trí tương ứng.

Bên dưới sẽ là nơi hiển thị public key mà hệ thống cung cấp để gọi dịch vụ API. Người dùng có thể đổi cặp key mới bằng nút “Change key” ở dưới.



Hình : Giao diện xem danh sách các video mà ứng dụng đã tải lên thông qua API

Người sử dụng có thể xem danh sách các video gần đây nhất đã tải lên hệ thống thông qua API khi sử dụng public key. Để cập nhật lại danh sách, có thể bấm nút “Reload” ở góc trên bên phải để hệ thống lấy danh sách mới nhất.

## Đánh giá hiệu suất

**Về hiệu suất hệ thống:**

* Thời gian load trang web rất nhanh chóng vì kiến trúc hệ thống đã xây dựng còn đơn giản, chưa có quá nhiều element phức tạp.
* Thời gian tải video sẽ tỉ lệ thuận với số lượng video cần load. Nghĩa là càng nhiều video sẽ càng tốn nhiều thời gian để load video. Tuy nhiên không ảnh hưởng đến việc load các yếu tố trong trang web.
* Thời gian tải video lên hệ thống và nhận diện còn chậm do sử dụng chung một luồng xử lí.

**Về hiệu suất dịch vụ API:**

Để đánh giá hiệu suất của dịch vụ API, em sử dụng phần mêm Postman để thực hiện 10 lời gọi API liên tiếp với cùng 1 file video từ khi hệ thống bắt đầu chạy và lấy trung bình cộng thời gian của 10 lần đó. Kết quả thu được cho thấy thời gian gọi dịch vụ còn tương đối chậm (xem bảng thống kê ở dưới) do thời gian tải file lên hệ thống lâu.

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Test** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** | **Avg** |
| **Predict** | 5000 | 103 | 49 | 41 | 40 | 38 | 41 | 39 | 46 | 40 | 543,7 ms |
| **Response** | 6100 | 1220 | 888 | 923 | 911 | 922 | 954 | 894 | 872 | 972 | 1465,6 ms |

Có thể thấy thời gian dự đoán mô hình ban đầu lâu do hệ thống vừa chạy, nhưng sau vài lần nhận diện thì thời gian nhận diện trung bình đã cải thiện. Tuy nhiên do thời gian xử lí và lưu trữ file còn lâu làm cho thời gian phản hồi của API tăng theo.

## Hạn chế

Do kiến thức và thời gian tiếp xúc với framework chưa nhiều nên hệ thống đã xây dựng còn một số hạn chế như sau:

* Hệ thống chỉ phù hợp sử dụng cho lượng nhỏ người dùng do kiến trúc thiết kế hệ thống không có khả năng chịu tải lớn.
* Người dùng không nắm rõ được quy trình nhận diện tính bạo lực trong video.
* Không theo dõi được các thông số về hiệu suất, lưu lượng sử dụng và thông tin của các ứng dụng sử dụng dịch vụ API.
* Thuật toán và mô hình nhận diện tính bạo lực chưa đủ chặt chẽ và chỉ phù hợp với video ngắn vài chục giây.
* Hiệu suất hệ thống chậm do quá trình tải video và lưu video chưa tối ưu.

## Kết luận và hướng phát triển

Sau khoảng thời gian tìm hiểu framework Django và xây dựng hệ thống nhận diện video có tính bạo lực thì em đã nắm được nguyên lí thiết kế của framework, hiểu được ưu và nhược điểm để có thể sử dụng Django ở những trường hợp với yêu cầu phù hợp. Cuối cùng, xây dựng được hệ thống tương đối hoàn chỉnh hoàn toàn bằng framework Django.

Trong tương lai gần định hướng phát triển cho hệ thống sẽ như sau:

1. Tối ưu hóa thời gian tải file lên hệ thống.
2. Tối ưu hóa thời gian phản hồi dịch vụ API.
3. Mở rộng kiến trúc hệ thống để đáp ứng được nhiều người sử dụng.
4. Tối ưu hóa thuật toán và mô hình nhận diện tính bạo lực của video để cải thiện độ chính xác và nhận diện được video có thời lượng dài hơn.
5. Xem xét đến khía cạnh nhận diện tính bạo lực có trong hình ảnh hay tính bạo lực trong văn bản.

# TÀI LIỆU THAM KHẢO

Foundation, D. S. (2024, 11 12). *Django Official Page*. From Django Documentation: https://docs.djangoproject.com/en/5.1/

Khalid, A. (2024, 10 12). *Kaggle*. From Real Time Violence Detection | MobileNet Bi-LSTM: https://www.kaggle.com/code/abduulrahmankhalid/real-time-violence-detection-mobilenet-bi-lstm/notebook

contributors, I. m. (2024, 11 1). *mdn*. From https://developer.mozilla.org/en-US/