



BÁO CÁO ĐỒ ÁN CUỐI KỲ MÔN

NGÔN NGỮ LẬP TRÌNH JAVA

Lớp: SE330.O21

GVHD: LÊ THANH TRỌNG

ĐỀ TÀI: Video Sharing

THÀNH VIÊN THỰC HIỆN (Nhóm 2):

STT	Họ và tên	MSSV
1	Nguyễn Thành Đăng	21520683
2	Tạ Đức Bảo	21520623
3	Trịnh Tấn Đạt	21520714
4	Nguyễn Trần Bảo Quốc	21520421

TP. HỒ CHÍ MINH, 2024



MỤC LỤC

LỜI CẢM ƠN.....	4
BÁO CÁO CHI TIẾT.....	5
A. CHƯƠNG 1: DANH MỤC HÌNH ẢNH	5
B. CHƯƠNG 2: DANH MỤC BẢNG BIỂU	6
C. CHƯƠNG 3: TỔNG QUAN ĐỀ TÀI	7
1. Giới thiệu	7
2. Mục đích và Chức năng	7
a. Tải lên video	7
b. Xem video	7
c. Tương tác với video.....	7
d. Theo dõi.....	7
e. Kết luận.....	7
D. CHƯƠNG 4: CƠ SỞ LÝ THUYẾT	8
1. Ngôn ngữ sử dụng	8
a. Backend	8
b. Frontend	9
2. Các công nghệ sử dụng	9
a. Database	9
b. Giao thức truyền tin	10
c. Các annotation sử dụng trong Spring Boot.....	10
d. Spring Security	11
e. Spring Multipartfile	12
f. Cross Origin	12
3. Các thư viện của Spring boot	13
a. Spring Boot Starter Security	13
b. Spring Boot Starter Web	13
c. Spring Boot Starter Data MongoDB	13
d. Spring Boot DevTools	14
e. Spring Boot Starter Mail	14
f. Spring Boot Starter Test	15
g. Spring Security Test	15
h. Lombok	15
i. Gson.....	16
E. CHƯƠNG 5: PHÂN TÍCH VÀ THIẾT KẾ ĐỀ TÀI	17
1. Sơ đồ hoạt động	17
a. Mô hình	17

b.	Use-Cases	19
c.	Class Diagram	21
2.	Mô phỏng giao diện ứng dụng	22
a.	Giao diện ứng dụng	22
3.	Phân chia công việc	26
F.	CHƯƠNG 6: HIỆN THỰC ĐỀ TÀI	27
1.	Back-end	27
a.	API cho ứng dụng	27
b.	Kết nối API với các công nghệ sử dụng	27
c.	Viết API	28
2.	Front-end	34
a.	Các giao diện chính	34
3.	Kết nối Front-end và Back-end	38
a.	Giới thiệu	38
b.	Cấu trúc API	38
c.	Quy trình xác thực và phân quyền	39
d.	Quản lý trạng thái và giao tiếp	39
e.	Kết luận	39
G.	CHƯƠNG 7: KIỂM THỬ HỆ THỐNG	40
1.	Kiểm thử Comment Service	40
a.	Kiểm tra comment service có hoạt động không	40
b.	Kiểm tra xem comment service đã kết nối tới cơ sở dữ liệu MongoDB chưa 40	
2.	Kiểm thử Video Service	41
a.	Kiểm tra video service có hoạt động hay không	41
b.	Kiểm tra xem video service đã kết nối tới cơ sở dữ liệu MongoDB chưa	41
3.	Kiểm thử User Service	42
a.	Kiểm tra user service có hoạt động hay không	42
b.	Kiểm tra xem user service đã kết nối tới cơ sở dữ liệu MongoDB chưa	42
H.	CHƯƠNG 8: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN	43
1.	Kết luận	43
2.	Hướng phát triển	43
a.	Xây dựng ứng dụng đa nền tảng (web, iOS, Android):	43
b.	Huấn luyện một mô hình có thể nhận biết và tự động gỡ bỏ các video vi phạm: 43	
c.	Phát triển một hệ thống socket:	43
I.	CHƯƠNG 9: PHỤ LỤC	44
	TÀI LIỆU THAM KHẢO	45

LỜI CẢM ƠN

Lời đầu tiên, chúng em xin gửi lời cảm ơn sâu sắc đến Thầy Lê Thanh Trọng, giảng viên môn Ngôn ngữ lập trình Java tại khoa Công nghệ phần mềm. Thầy đã truyền đạt kiến thức một cách nhiệt tình và tận tâm, giúp chúng em hiểu rõ hơn về những khái niệm cốt lõi của Java và lập trình hướng đối tượng.

Nhờ sự hướng dẫn của Thầy, chúng em không chỉ nắm vững các kiến thức lý thuyết mà còn áp dụng thành công vào các bài tập thực hành. Qua quá trình học tập, chúng em đã phát triển được khả năng tư duy logic, kỹ năng giải quyết vấn đề và sự tự tin khi lập trình. Mỗi bài giảng của Thầy không chỉ cung cấp cho chúng em những hiểu biết sâu rộng về Java mà còn là nguồn cảm hứng để chúng em không ngừng học hỏi và khám phá.

Thầy đã luôn sẵn sàng lắng nghe, giải đáp mọi thắc mắc và hỗ trợ chúng em mọi lúc. Nhờ sự tận tụy và nhiệt tình của Thầy, chúng em không chỉ trưởng thành hơn trong lĩnh vực lập trình mà còn học được cách làm việc nhóm, quản lý thời gian và phân tích vấn đề một cách hiệu quả. Chúng em thật sự biết ơn những bài giảng sinh động, những ví dụ minh họa cụ thể và phương pháp giảng dạy sáng tạo của Thầy.

Chúng em cảm nhận được rằng, không chỉ dừng lại ở việc giảng dạy kiến thức, Thầy còn truyền đạt cho chúng em những giá trị đạo đức nghề nghiệp, lòng đam mê và tinh thần trách nhiệm với công việc. Sự tận tâm và tâm huyết của Thầy đã để lại ấn tượng sâu sắc và là nguồn động lực lớn lao giúp chúng em vững bước trên con đường học tập và phát triển bản thân.

Một lần nữa, chúng em xin chân thành cảm ơn Thầy Lê Thanh Trọng vì những đóng góp quý báu và sự tận tụy trong giảng dạy. Chúng em kính chúc Thầy luôn mạnh khỏe, hạnh phúc và tiếp tục thành công trong sự nghiệp giáo dục. Hy vọng rằng trong tương lai, chúng em sẽ có cơ hội tiếp tục được học hỏi và nhận được sự hướng dẫn từ Thầy.

BÁO CÁO CHI TIẾT

A. CHƯƠNG 1: DANH MỤC HÌNH ẢNH

Hình 1 – Spring boot	8
Hình 2 – ReactJS.....	9
Hình 3 – MongoDB Atlas.....	9
Hình 4 – Spring Security.....	11
Hình 5 – Spring multipartFile	12
Hình 6 – Topology	17
Hình 7 –Use Case	19
Hình 8 – Database.....	21
Hình 9 – Giao diện chính.....	22
Hình 10 – Màn hình đăng nhập	22
Hình 11 – Màn hình đăng ký tài khoản.....	23
Hình 12 – Màn hình Profile cá nhân	23
Hình 13 – Màn hình video đã like	24
Hình 14 – Màn hình khi xem video.....	24
Hình 15 – Màn hình đăng tải video	25
Hình 16 – Màn hình thay đổi mật khẩu.....	25
Hình 17 – Màn hình thay đổi thông tin cá nhân.....	26
Hình 18 – Mô hình MVC	27

B. CHƯƠNG 2: DANH MỤC BẢNG BIỂU

Bảng 1 – Phân công công việc.....	26
Bảng 2 – Màn hình trang chủ.....	34
Bảng 3 – Màn hình Đăng ký tài khoản	34
Bảng 4 – Màn hình Đăng nhập	35
Bảng 5 – Màn hình xem Video	35
Bảng 6 – Màn hình bình luận video	36
Bảng 7 – Màn hình hồ sơ người dùng	36
Bảng 8 – Màn hình tải lên video	37
Bảng 9 – Màn hình đổi mật khẩu.....	37
Bảng 10 – Màn hình đổi thông tin cá nhân	38

C. CHƯƠNG 3: TỔNG QUAN ĐỀ TÀI

1. Giới thiệu

Ngày nay, việc chia sẻ và xem video trực tuyến đã trở thành một phần không thể thiếu trong cuộc sống hàng ngày của mọi người trên khắp thế giới. Nhằm đáp ứng nhu cầu đó, đề tài xây dựng một ứng dụng website chia sẻ video với backend sử dụng Spring Boot và frontend sử dụng React đã được hình thành. Đây là một nền tảng cho phép người dùng tải lên, xem và tương tác với các video một cách dễ dàng và thuận tiện.

2. Mục đích và Chức năng

a. Tải lên video

Ứng dụng cho phép người dùng đăng ký tài khoản và tải lên các video của mình. Các video tải lên sẽ được hiển thị ở chế độ công khai (public), có nghĩa là tất cả người dùng khác đều có thể xem các video này. Việc tải lên video sẽ được thực hiện một cách nhanh chóng và an toàn, đảm bảo chất lượng và tính toàn vẹn của nội dung.

b. Xem video

Tất cả người dùng, dù có tài khoản hay không, đều có thể truy cập và xem các video công khai trên nền tảng. Giao diện xem video được thiết kế trực quan và thân thiện với người dùng, cung cấp trải nghiệm xem video mượt mà và dễ chịu.

c. Tương tác với video

Người dùng có thể tương tác với các video bằng cách:

- Like/Dislike Video: Cho phép người dùng thể hiện cảm xúc và phản hồi về video bằng cách thích hoặc không thích.
- Bình Luận (Comment): Người dùng có thể để lại bình luận dưới mỗi video, tạo cơ hội cho các cuộc thảo luận và phản hồi.
- Like/Dislike Bình Luận: Ngoài việc bình luận, người dùng cũng có thể thích hoặc không thích các bình luận của người khác, tạo nên một hệ thống phản hồi cộng đồng phong phú và đa dạng.

d. Theo dõi

Ứng dụng cho phép người dùng theo dõi (subscribe) các kênh hoặc người dùng mà họ yêu thích. Điều này giúp người dùng dễ dàng cập nhật các video mới nhất từ những người họ quan tâm, tạo nên một hệ thống thông báo và quản lý nội dung cá nhân hoá.

e. Kết luận

Ứng dụng website chia sẻ video với các chức năng tải lên, xem, bình luận, tương tác và theo dõi sẽ mang đến cho người dùng một nền tảng tiện ích, thân thiện và hiện đại để chia sẻ và trải nghiệm các nội dung video. Đây là một đề tài đầy thách thức nhưng cũng rất thú vị, hứa hẹn mang lại nhiều kiến thức và kỹ năng quý báu trong lĩnh vực phát triển ứng dụng web.

D. CHƯƠNG 4: CƠ SỞ LÝ THUYẾT

1. Ngôn ngữ sử dụng

a. Backend

Ngôn ngữ: Java.

Framework: Spring boot.



Hình 1 – Spring boot

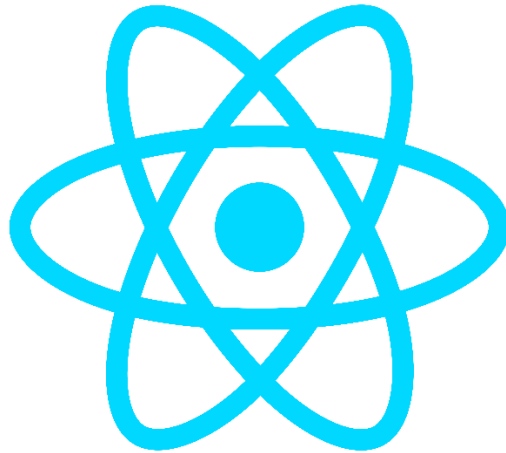
Lý do chọn Spring Boot:

- Spring Boot giúp đơn giản hóa quá trình phát triển ứng dụng bằng cách loại bỏ nhiều cấu hình phức tạp thường gặp trong các ứng dụng Spring truyền thống.
- Spring Boot dễ dàng tích hợp với nhiều công nghệ và thư viện khác nhau, cho phép ứng dụng mở rộng và tích hợp các tính năng mới một cách linh hoạt.
- Spring Boot cung cấp hỗ trợ mạnh mẽ cho việc xây dựng các RESTful API, điều này rất cần thiết cho một ứng dụng chia sẻ video. Các API này sẽ cho phép giao tiếp giữa frontend (React) và backend một cách hiệu quả.
- Spring Boot đi kèm với Spring Security, một module mạnh mẽ cung cấp các giải pháp bảo mật toàn diện. Điều này giúp bảo vệ ứng dụng khỏi các mối đe dọa bảo mật phổ biến.
- Spring Boot tích hợp tốt với nhiều hệ quản trị cơ sở dữ liệu khác nhau như MySQL, PostgreSQL, MongoDB, v.v. Điều này cho phép quản lý và truy xuất dữ liệu người dùng, video và các thông tin liên quan một cách hiệu quả và linh hoạt.
- Spring Boot có một cộng đồng người dùng lớn và tài liệu phong phú, cung cấp nhiều tài nguyên hỗ trợ khi gặp vấn đề trong quá trình phát triển. Các diễn đàn, bài viết, và hướng dẫn trực tuyến giúp giải quyết các vấn đề kỹ thuật nhanh chóng và hiệu quả.

b. Frontend

Ngôn ngữ: Javascript.

Framework: ReactJS



Hình 2 – ReactJS

Lý do chọn ReactJS:

- React có thể dễ dàng tương tác với backend Spring Boot thông qua các API RESTful. Điều này tạo điều kiện thuận lợi cho việc giao tiếp giữa frontend và backend, đảm bảo dữ liệu được trao đổi một cách hiệu quả và bảo mật.
- React có một cộng đồng người dùng rất lớn và tài liệu phong phú. Điều này cung cấp nhiều tài nguyên hỗ trợ khi gặp vấn đề trong quá trình phát triển. Các diễn đàn, bài viết, và hướng dẫn trực tuyến giúp giải quyết các vấn đề kỹ thuật nhanh chóng và hiệu quả.

2. Các công nghệ sử dụng

a. Database

Công nghệ: MongoDB Atlas



Hình 3 – MongoDB Atlas

Lý do lựa chọn:

- MongoDB Atlas cung cấp khả năng mở rộng tự động, cho phép ứng dụng mở rộng theo nhu cầu lưu trữ và tải truy cập một cách linh hoạt mà không cần can

thiệt thủ công. Điều này rất quan trọng đối với một ứng dụng chia sẻ video, nơi dữ liệu video có thể phát triển nhanh chóng và cần khả năng xử lý tải cao từ người dùng.

- MongoDB Atlas dễ dàng tích hợp với các ứng dụng được xây dựng bằng Spring Boot thông qua các thư viện như Spring Data MongoDB. Điều này giúp việc giao tiếp giữa ứng dụng và cơ sở dữ liệu trở nên mượt mà và hiệu quả.

b. Giao thức truyền tin

Công nghệ: HTTP và HTTP Range Request

Lý do lựa chọn:

HTTP: là giao thức cơ bản được sử dụng trên World Wide Web để truyền tải dữ liệu giữa client (như trình duyệt web) và server. Đối với ứng dụng website chia sẻ video, HTTP đóng vai trò quan trọng trong việc trao đổi dữ liệu giữa frontend và backend.

- Truyền Tải Video: HTTP được sử dụng để truyền tải dữ liệu giữa frontend và backend. Khi người dùng truy cập trang web, các yêu cầu HTTP sẽ được gửi đến server để tải danh sách video, thông tin người dùng, và các dữ liệu khác.
- Quản Lý Bình Luận và Tương Tác: Các tính năng bình luận, like, dislike và subscribe cũng sử dụng các yêu cầu HTTP để giao tiếp với server, cập nhật trạng thái và dữ liệu trong thời gian thực.

HTTP Range Request: là một phần mở rộng của giao thức HTTP, cho phép client yêu cầu chỉ một phần của tài nguyên, thay vì toàn bộ tài nguyên. Điều này đặc biệt hữu ích cho việc truyền tải các tệp lớn như video.

- Phát Video Trực Tuyến: HTTP Range Request được sử dụng để phát video trực tuyến. Khi người dùng phát video, trình duyệt sẽ gửi các yêu cầu range để tải các phần của video từng đoạn một. Điều này giúp video phát liên tục mà không bị gián đoạn.

c. Các annotation sử dụng trong Spring Boot

Annotations trong Spring Boot là các dấu chú thích (metadata) được thêm vào mã nguồn để cung cấp thông tin cho Spring Framework. Sử dụng các annotation giúp cấu hình và điều khiển hành vi của ứng dụng mà không cần phải viết mã cấu hình phức tạp. Dưới đây là những tác dụng chính của các annotations trong Spring Boot:

Cấu hình Bean và Dependency Injection:

- **@Component:** Đánh dấu một class là một Spring component. Spring sẽ tự động phát hiện và quản lý các bean này khi thực hiện quét classpath.
- **@Service:** Một biến thể của **@Component**, thường được sử dụng cho các lớp service. Nó giúp dễ dàng phân biệt giữa các loại bean khác nhau.
- **@Repository:** Một biến thể của **@Component**, thường được sử dụng cho các lớp truy cập dữ liệu. Nó cung cấp thêm tính năng xử lý ngoại lệ.

- **@Controller**: Một biến thể của **@Component**, thường được sử dụng cho các lớp controller trong MVC framework.
- **@Autowired**: Được sử dụng để tự động tiêm các phụ thuộc vào một bean.

Cấu hình và khởi tạo ứng dụng:

- **@SpringBootApplication**: Kết hợp ba annotations **@Configuration**, **@EnableAutoConfiguration**, và **@ComponentScan**. Được sử dụng để khởi động ứng dụng Spring Boot.

Xây Dựng RESTful Web Services:

- **@RestController**: Kết hợp **@Controller** và **@ResponseBody**, giúp xây dựng các RESTful web services. Các phương thức trong lớp này sẽ trả về dữ liệu trực tiếp thay vì trả về tên view.
- **@RequestMapping**: Đánh dấu các phương thức xử lý yêu cầu HTTP. Nó có thể được sử dụng trên class và phương thức để chỉ định URL và phương thức HTTP.
- **@GetMapping**, **@PostMapping**, **@PutMapping**, **@DeleteMapping**: Đơn giản hóa việc ánh xạ các phương thức HTTP GET, POST, PUT, DELETE tương ứng.

d. Spring Security

Spring Security là một framework mạnh mẽ và có khả năng tùy biến cao, được sử dụng để bảo vệ các ứng dụng dựa trên nền tảng Java và Spring. Framework này cung cấp các giải pháp bảo mật toàn diện bao gồm xác thực (authentication) và phân quyền (authorization), bảo vệ ứng dụng khỏi các mối đe dọa bảo mật phổ biến.



Hình 4 – Spring Security

- Xác thực người dùng qua nhiều phương thức: form-based login, HTTP Basic/Digest authentication, LDAP, OAuth2, JWT, v.v.
- Hỗ trợ các cơ chế lưu trữ thông tin người dùng khác nhau: in-memory, cơ sở dữ liệu, LDAP.
- Phân Quyền (Authorization): Kiểm soát truy cập dựa trên vai trò và quyền hạn của người dùng.
- Cấu hình phân quyền trên các URL, phương thức HTTP, hoặc các phương thức của class.
- Bảo Vệ Ứng Dụng: Bảo vệ chống lại các cuộc tấn công phổ biến như Cross-Site Request Forgery (CSRF), Clickjacking, Session Fixation, v.v.

- Tích Hợp Dễ Dàng: Tích hợp với các công nghệ và framework khác trong hệ sinh thái Spring như Spring MVC, Spring Boot, Spring Data, v.v.

e. Spring MultipartFile

Spring MultipartFile là một interface trong Spring Framework được sử dụng để xử lý các file được upload từ phía client lên server. Interface này cung cấp các phương thức để dễ dàng thao tác với file như đọc nội dung file, lấy tên file, và lưu file lên hệ thống tệp.



Hình 5 – Spring multipartFile

f. Cross Origin

Cross-Origin Resource Sharing (CORS) là một cơ chế bảo mật được các trình duyệt sử dụng để kiểm soát cách các tài nguyên của một trang web có thể được yêu cầu từ một domain khác với domain mà tài nguyên đó được phục vụ. CORS được sử dụng để cho phép hoặc ngăn chặn các yêu cầu từ các domain khác nhau, giúp bảo vệ dữ liệu người dùng và ngăn chặn các cuộc tấn công Cross-Site Request Forgery (CSRF).

Trong các ứng dụng web hiện đại, việc tách riêng Frontend và Backend là một mô hình phổ biến. Điều này thường dẫn đến việc các yêu cầu HTTP được gửi từ một domain (Frontend) đến một domain khác (Backend), và để cho phép các yêu cầu này, cần phải cấu hình CORS.

- Cấu hình Cross Origin trong Spring boot: Spring Boot cung cấp nhiều cách để cấu hình CORS, từ việc cấu hình trên toàn bộ ứng dụng đến việc cấu hình trên từng controller hoặc endpoint cụ thể. Dưới đây là các cách mà nhóm chúng em cấu hình Cross Origin.

```
@CrossOrigin(origins="**")
```

Việc xử lý CORS đúng cách trong ứng dụng Spring Boot là rất quan trọng để đảm bảo tính bảo mật và hiệu suất của ứng dụng, đồng thời cho phép Frontend và Backend giao tiếp một cách hiệu quả. Bằng cách sử dụng các cấu hình và kiểm tra kỹ càng như đã nêu ở trên, chúng em đã có thể thiết lập CORS một cách chính xác trong dự án chia sẻ video của mình, tạo điều kiện cho sự tương tác mượt mà giữa các thành phần của hệ thống.

3. Các thư viện của Spring boot

a. Spring Boot Starter Security

```
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-security</artifactId>
</dependency>
```

Mô tả:

- Cung cấp các tính năng bảo mật cho ứng dụng, bao gồm xác thực, phân quyền và bảo vệ ứng dụng web.
- Hỗ trợ triển khai các phương thức bảo mật như Basic Authentication, JWT, OAuth2, và các hình thức bảo mật tùy chỉnh.

Ứng dụng:

- Bảo vệ các API và trang web khỏi truy cập trái phép.
- Xác thực người dùng và quản lý phiên đăng nhập.

b. Spring Boot Starter Web

```
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-web</artifactId>
</dependency>
```

Mô tả:

- Cung cấp các thành phần cần thiết để xây dựng các ứng dụng web, RESTful services.
- Tích hợp sẵn với Spring MVC, Tomcat, Jackson để xử lý yêu cầu HTTP và chuyển đổi dữ liệu.

Ứng dụng:

- Tạo các endpoint REST API để giao tiếp giữa frontend và backend.
- Xử lý các yêu cầu và phản hồi HTTP.

c. Spring Boot Starter Data MongoDB

```
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-data-mongodb</artifactId>
</dependency>
```

Mô tả:

- Hỗ trợ kết nối và thao tác với cơ sở dữ liệu MongoDB.
- Cung cấp các phương thức truy vấn và xử lý dữ liệu NoSQL thông qua Spring Data MongoDB.

Ứng dụng:

- Quản lý và lưu trữ dữ liệu người dùng, video, bình luận trong MongoDB Atlas.
- Thực hiện các thao tác CRUD (Create, Read, Update, Delete) trên dữ liệu.

d. Spring Boot DevTools

```
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-devtools</artifactId>
  <scope>runtime</scope>
  <optional>true</optional>
</dependency>
```

Mô tả:

- Cung cấp các công cụ hỗ trợ phát triển như tự động khởi động lại (auto-restart), cập nhật nóng (hot swapping).
- Giúp tăng tốc độ phát triển bằng cách giảm thời gian phản hồi khi có thay đổi mã nguồn.

Ứng dụng:

- Giảm thời gian khởi động lại ứng dụng khi thay đổi mã nguồn.
- Tăng hiệu quả phát triển và thử nghiệm.

e. Spring Boot Starter Mail

```
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-mail</artifactId>
  <version>3.2.1</version>
</dependency>
```

Mô tả:

- Hỗ trợ gửi email từ ứng dụng Spring Boot.
- Tích hợp sẵn với JavaMailSender để gửi các email thông báo, xác thực, hoặc thông tin khác.

Ứng dụng:

- Gửi email xác thực khi người dùng đăng ký tài khoản mới.
- Gửi thông báo hoặc báo cáo qua email.

f. Spring Boot Starter Test

```
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-test</artifactId>
  <scope>test</scope>
</dependency>
```

Mô tả:

- Cung cấp các công cụ và thư viện hỗ trợ kiểm thử ứng dụng Spring Boot.
- Bao gồm JUnit, Mockito, và Spring Test để viết và chạy các test case.

Ứng dụng:

- Viết và thực hiện các kiểm thử đơn vị (unit test) và kiểm thử tích hợp (integration test).
- Đảm bảo chất lượng và độ ổn định của mã nguồn.

g. Spring Security Test

```
<dependency>
  <groupId>org.springframework.security</groupId>
  <artifactId>spring-security-test</artifactId>
  <scope>test</scope>
</dependency>
```

Mô tả:

- Cung cấp các công cụ và lớp hỗ trợ kiểm thử các chức năng bảo mật của ứng dụng.
- Hỗ trợ tạo các test case cho các endpoint bảo mật và xác thực người dùng.

Ứng dụng:

- Kiểm thử các endpoint yêu cầu bảo mật.
- Xác thực tính đúng đắn của các cơ chế bảo mật đã triển khai.

h. Lombok

```
<dependency>
  <groupId>org.projectlombok</groupId>
  <artifactId>lombok</artifactId>
  <optional>true</optional>
</dependency>
```

Mô tả:

- Cung cấp các chú thích (annotations) giúp giảm mã boilerplate trong Java.
- Tự động tạo các getter, setter, constructor, và các phương thức khác.

Ứng dụng:

- Giảm thiểu mã nguồn lặp lại, tăng tính ngắn gọn và dễ đọc của mã nguồn.

i. Gson

```
<dependency>
  <groupId>com.google.code.gson</groupId>
  <artifactId>gson</artifactId>
  <version>2.10.1</version>
</dependency>
```

Mô tả:

- Thư viện của Google cho phép chuyển đổi giữa Java Objects và JSON.
- Dễ dàng sử dụng để phân tích và tạo JSON trong các ứng dụng Java.

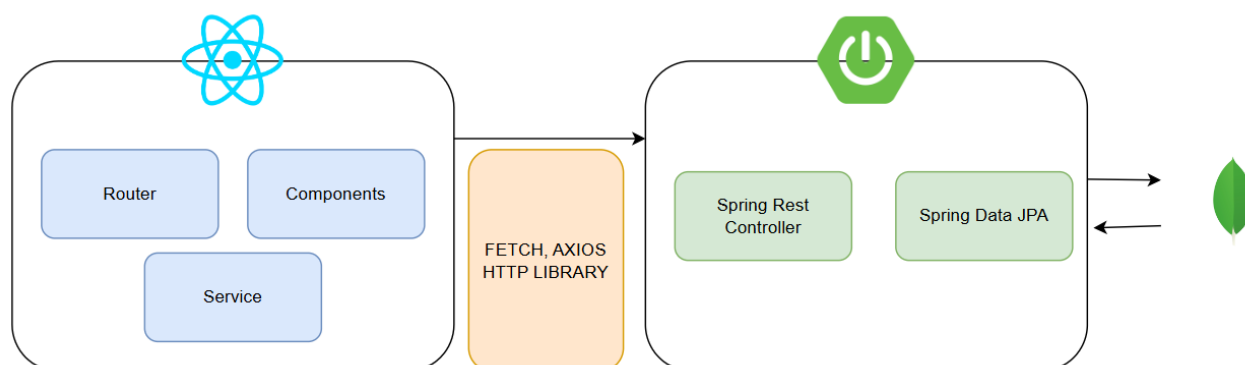
Ứng dụng:

- Chuyển đổi dữ liệu giữa các đối tượng Java và JSON để trao đổi thông tin giữa frontend và backend.

E. CHƯƠNG 5: PHÂN TÍCH VÀ THIẾT KẾ ĐỀ TÀI

1. Sơ đồ hoạt động

a. Mô hình



Hình 6 – Topology

Thành Phần của hệ thống

React (Front-end):

- Router: Quản lý các tuyến đường (routes) trong ứng dụng, giúp điều hướng giữa các trang và các thành phần khác nhau.
- Components: Các thành phần giao diện người dùng được chia nhỏ để dễ quản lý và tái sử dụng. Ví dụ như header, footer, video player, danh sách video, v.v.
- Service: Tầng dịch vụ chịu trách nhiệm gọi API đến back-end, xử lý dữ liệu nhận được từ server và gửi lên server. Dùng các thư viện HTTP như Fetch hoặc Axios để thực hiện các yêu cầu này.

Fetch, Axios HTTP Library:

- Fetch: Một API có sẵn trong trình duyệt để thực hiện các yêu cầu HTTP.
- Axios: Một thư viện HTTP phổ biến của JavaScript để thực hiện các yêu cầu HTTP, hỗ trợ các tính năng như interceptors, bảo mật token, v.v.

Spring Boot (Back-end):

- Spring Rest Controller: Thành phần của Spring Boot chịu trách nhiệm xử lý các yêu cầu HTTP từ client (React). Các controller này định nghĩa các API endpoints để thực hiện các thao tác như upload video, lấy danh sách video, lấy thông tin chi tiết video, v.v.
- Spring Data JPA: Một phần của Spring giúp tương tác với cơ sở dữ liệu. Spring Data JPA giúp đơn giản hóa việc thực hiện các thao tác CRUD (Create, Read, Update, Delete) với cơ sở dữ liệu.

MongoDB:

- Cơ sở dữ liệu NoSQL: MongoDB là một cơ sở dữ liệu NoSQL linh hoạt, được sử dụng để lưu trữ dữ liệu video. Nó cho phép lưu trữ dữ liệu dưới dạng JSON-like documents, giúp việc lưu trữ và truy xuất dữ liệu nhanh chóng và hiệu quả.
- Kết nối với Spring Data JPA: Dữ liệu được lưu trữ trong MongoDB và được truy xuất thông qua Spring Data JPA. Spring Boot sẽ cung cấp các repository để thực hiện các thao tác với MongoDB một cách dễ dàng.

Luồng hoạt động của hệ thống:***Giao diện người dùng (React):***

- Người dùng tương tác với ứng dụng thông qua các components và routes.
- Các yêu cầu từ người dùng (như xem danh sách video, tải lên video mới) sẽ được gửi đến back-end thông qua các service sử dụng Fetch hoặc Axios.

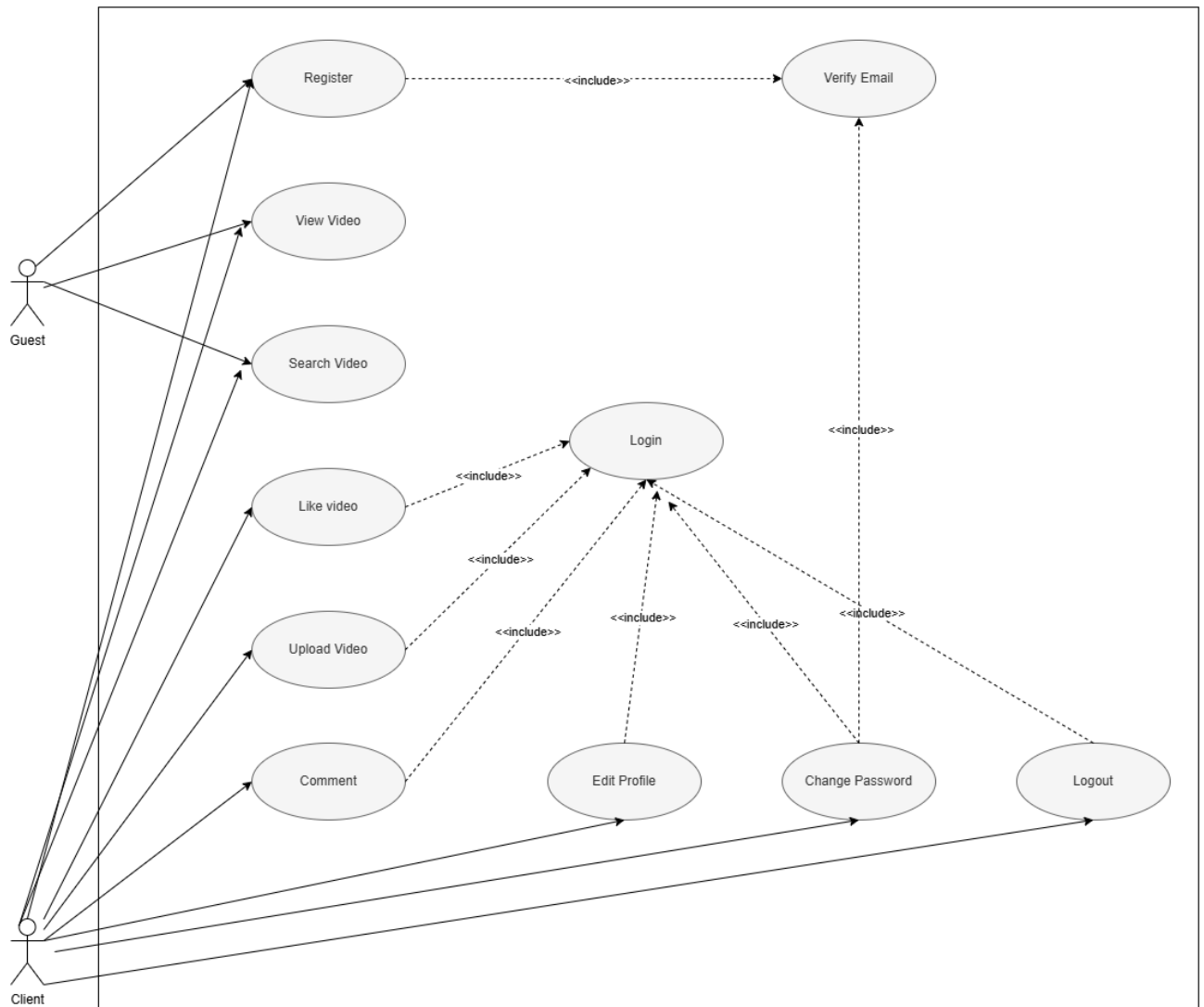
Back-end (Spring Boot):

- Spring Rest Controller nhận các yêu cầu từ front-end, xử lý logic nghiệp vụ và tương tác với cơ sở dữ liệu MongoDB thông qua Spring Data JPA.
- Kết quả xử lý (danh sách video, thông tin chi tiết video, v.v.) sẽ được trả về front-end dưới dạng JSON.

Cơ sở dữ liệu (MongoDB):

- Lưu trữ thông tin video, thông tin người dùng, và các dữ liệu liên quan khác.
- Cung cấp dữ liệu cho Spring Boot khi có yêu cầu.

b. Use-Cases



Hình 7 –Use Case

Các Use Case chính:

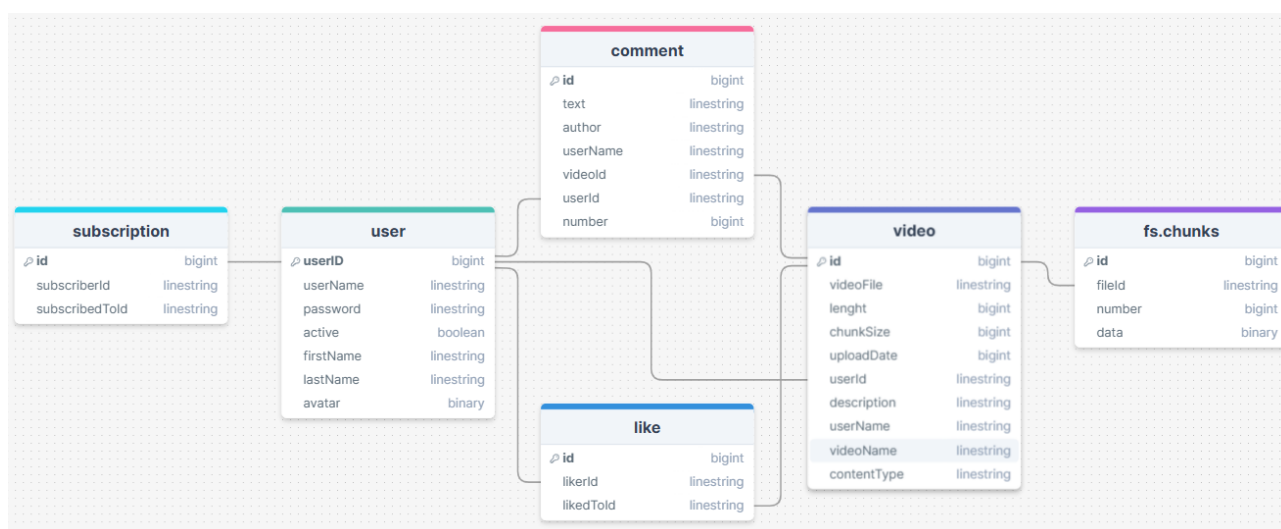
- Register (Đăng ký):
 - Actor: Guest
 - Mô tả: Guest có thể đăng ký tài khoản mới để trở thành Client.
 - Liên kết: Bao gồm Verify Email.
- Verify Email (Xác minh email):
 - Actor: Guest
 - Mô tả: Sau khi đăng ký, Guest phải xác minh email để kích hoạt tài khoản.
- Login (Đăng nhập):
 - Actor: Guest
 - Mô tả: Guest có thể đăng nhập vào hệ thống để trở thành Client.
- View Video (Xem video):

- Actor: Guest, Client
 - Mô tả: Cả Guest và Client có thể xem các video có sẵn trên hệ thống.
- Search Video (Tìm kiếm video):
 - Actor: Guest, Client
 - Mô tả: Cả Guest và Client có thể tìm kiếm video dựa trên từ khóa.
- Like Video (Thích video):
 - Actor: Client
 - Mô tả: Client có thể thích video, yêu cầu phải đăng nhập (bao gồm Login).
- Upload Video (Tải lên video):
 - Actor: Client
 - Mô tả: Client có thể tải lên video mới lên hệ thống, yêu cầu phải đăng nhập (bao gồm Login).
- Comment (Bình luận):
 - Actor: Client
 - Mô tả: Client có thể bình luận trên video, yêu cầu phải đăng nhập (bao gồm Login).
- Edit Profile (Chỉnh sửa hồ sơ):
 - Actor: Client
 - Mô tả: Client có thể chỉnh sửa thông tin cá nhân, yêu cầu phải đăng nhập (bao gồm Login).
- Change Password (Đổi mật khẩu):
 - Actor: Client
 - Mô tả: Client có thể đổi mật khẩu, yêu cầu phải đăng nhập (bao gồm Login).
- Logout (Đăng xuất):
 - Actor: Client
 - Mô tả: Client có thể đăng xuất khỏi hệ thống.

Các mối quan hệ:

- Include (Bao gồm):
- Register bao gồm Verify Email: Khi người dùng đăng ký, họ phải xác minh email để hoàn tất quá trình đăng ký.
- Like Video, Upload Video, Comment, Edit Profile, Change Password bao gồm Login: Các hành động này yêu cầu người dùng phải đăng nhập trước khi thực hiện.

c. Class Diagram



Hình 8 – Database

Mối quan hệ giữa các bảng:

User - Video:

Mỗi video được tải lên bởi một user xác định qua userId.

User - Comment:

Mỗi comment được đăng bởi một user xác định qua userId và liên quan đến một video xác định qua videoId.

User - Like:

Mỗi lượt like được thực hiện bởi một user xác định qua userId và liên quan đến một video xác định qua likedId.

User - Subscription:

Bảng subscription lưu trữ mối quan hệ theo dõi giữa các người dùng, trong đó subscriberId là ID của người theo dõi và subscribedToId là ID của người được theo dõi.

Video - Comment:

Mỗi comment liên quan đến một video xác định qua videoId.

Video - Like:

Mỗi like liên quan đến một video xác định qua likedId.

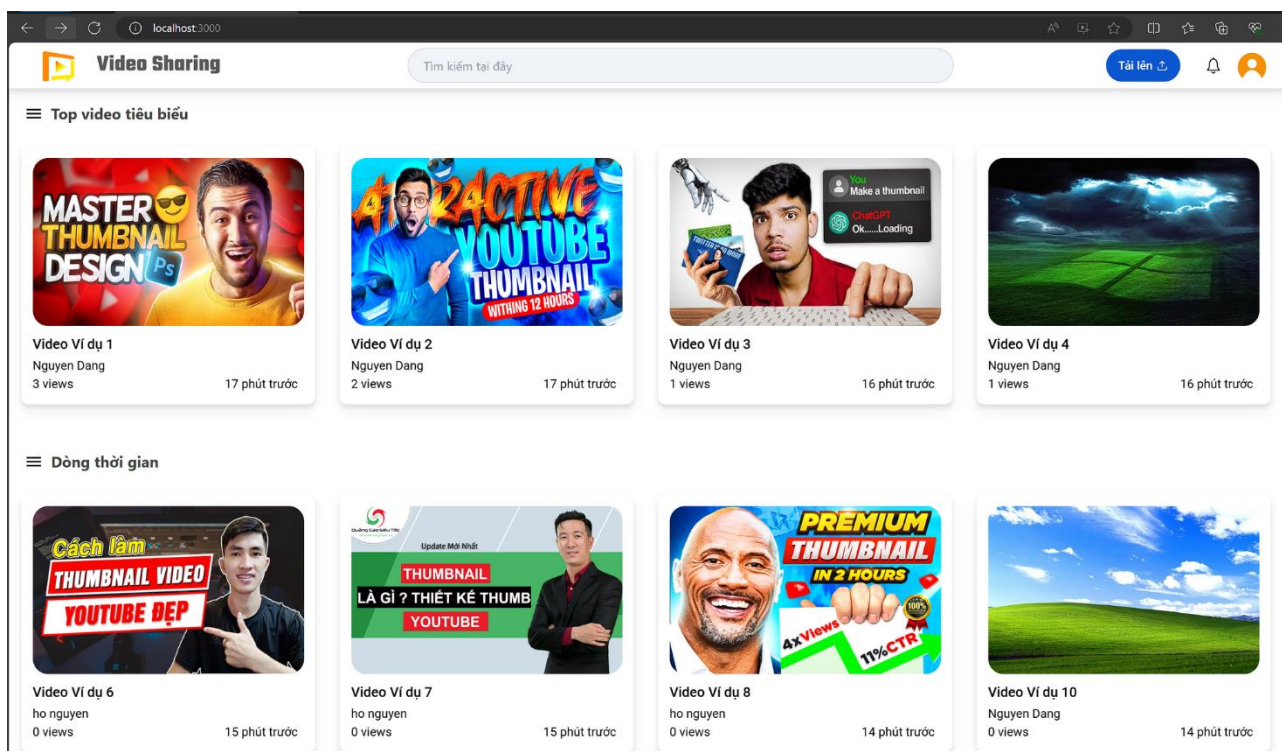
Video - fs.chunks:

Mỗi video có thể được lưu trữ thành nhiều phần (chunks) trong bảng fs.chunks, xác định qua fileId.

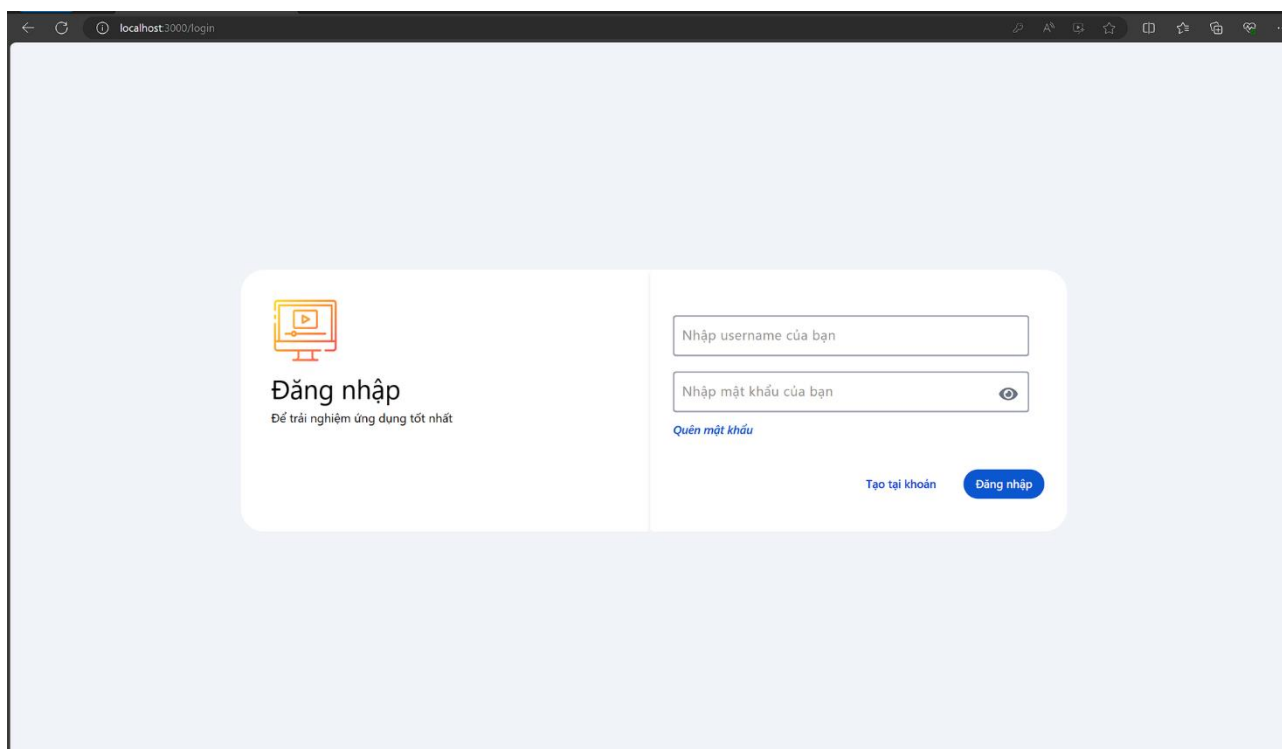
2. Mô phỏng giao diện ứng dụng

a. Giao diện ứng dụng

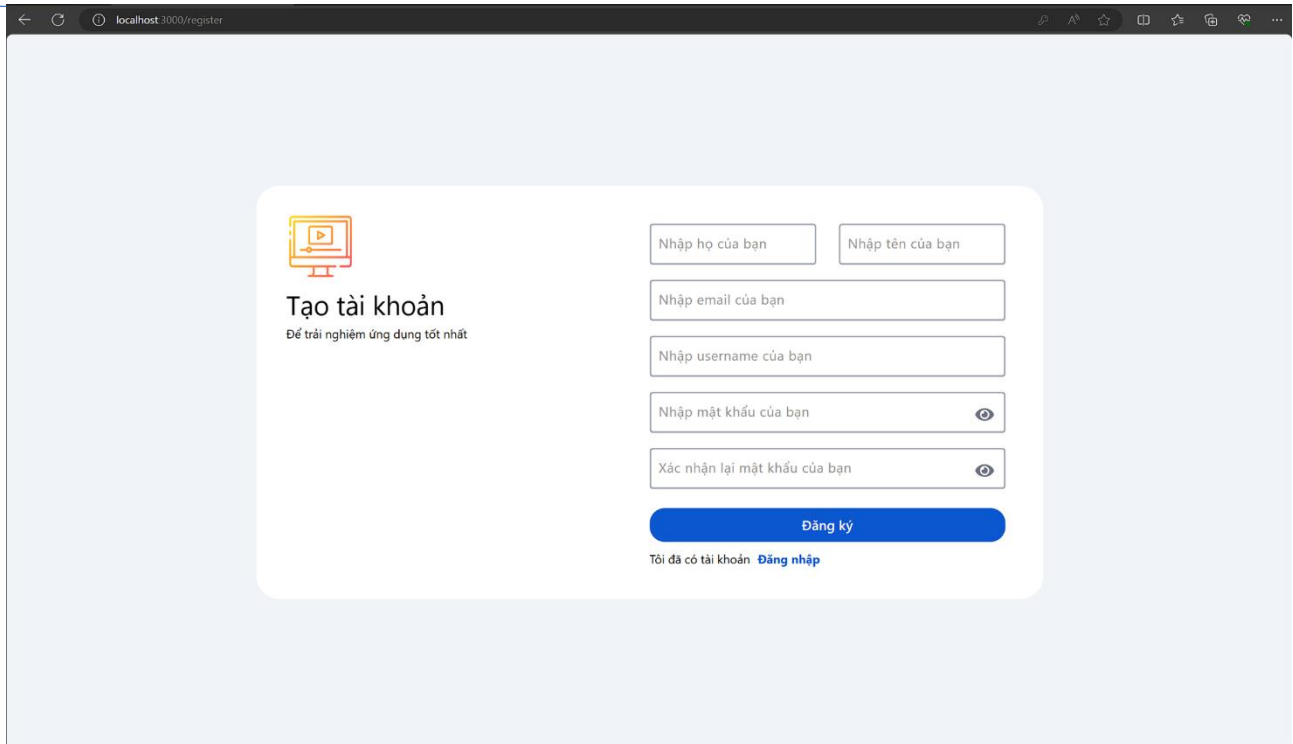
Một số giao diện chính của ứng dụng Video Sharing



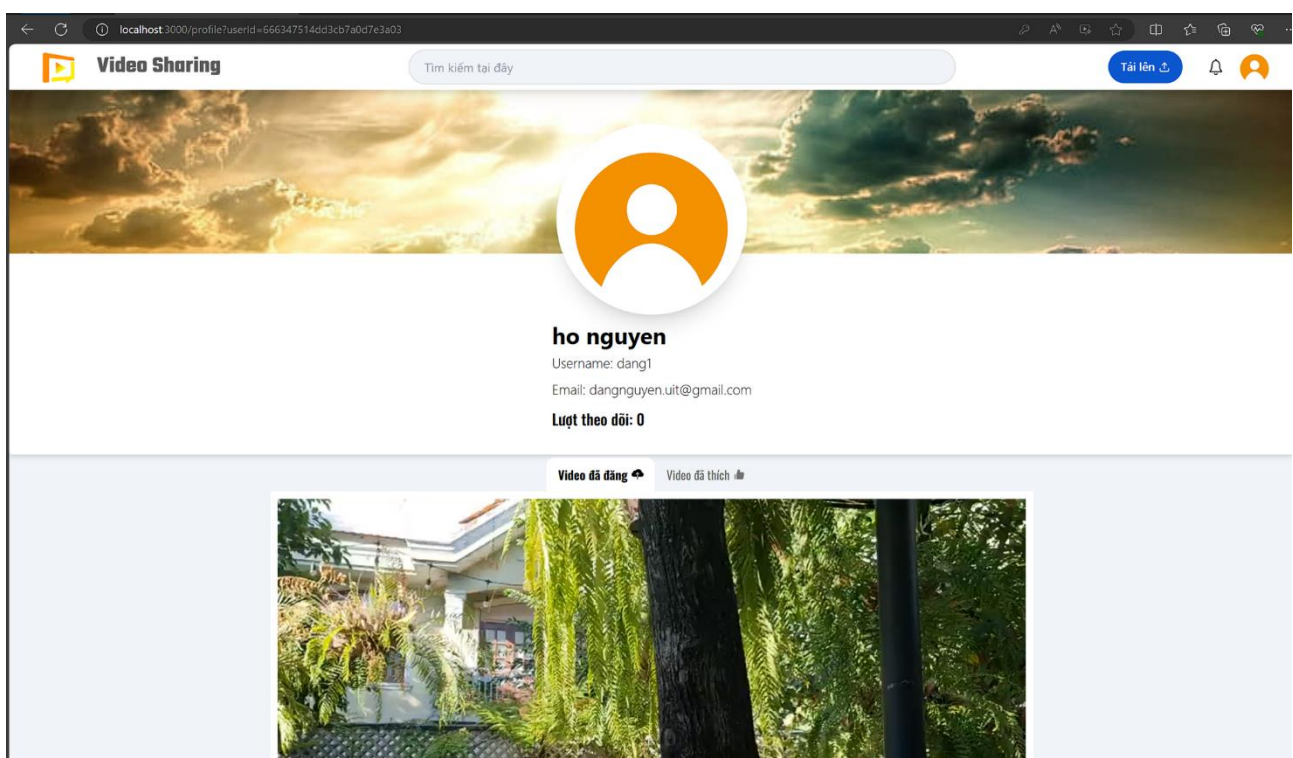
Hình 9 – Giao diện chính



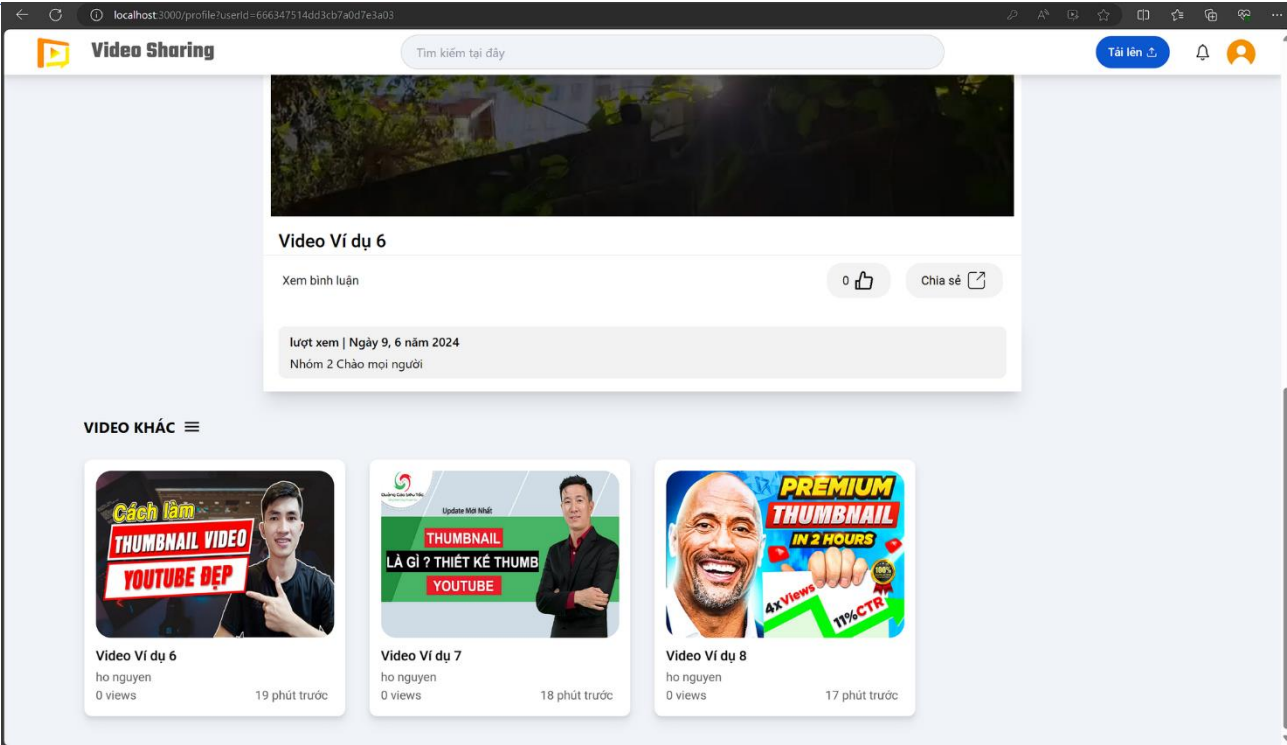
Hình 10 – Màn hình đăng nhập



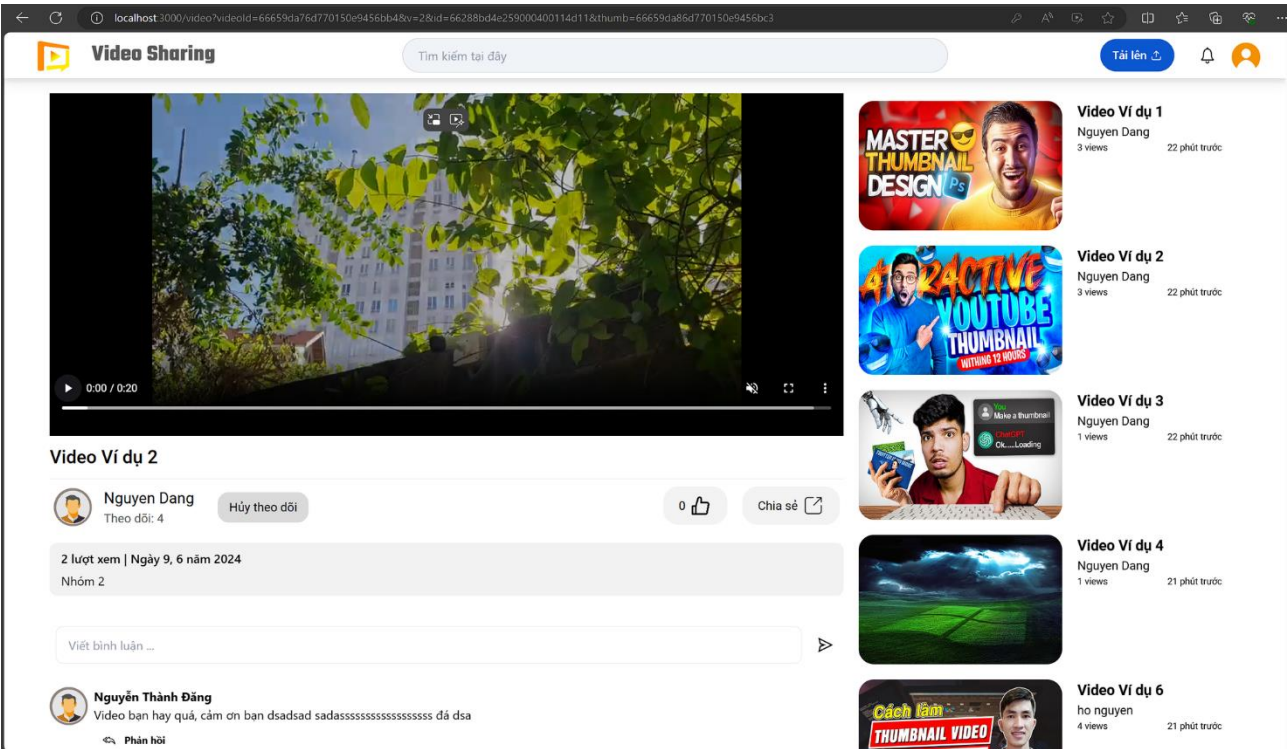
Hình 11 – Màn hình đăng ký tài khoản



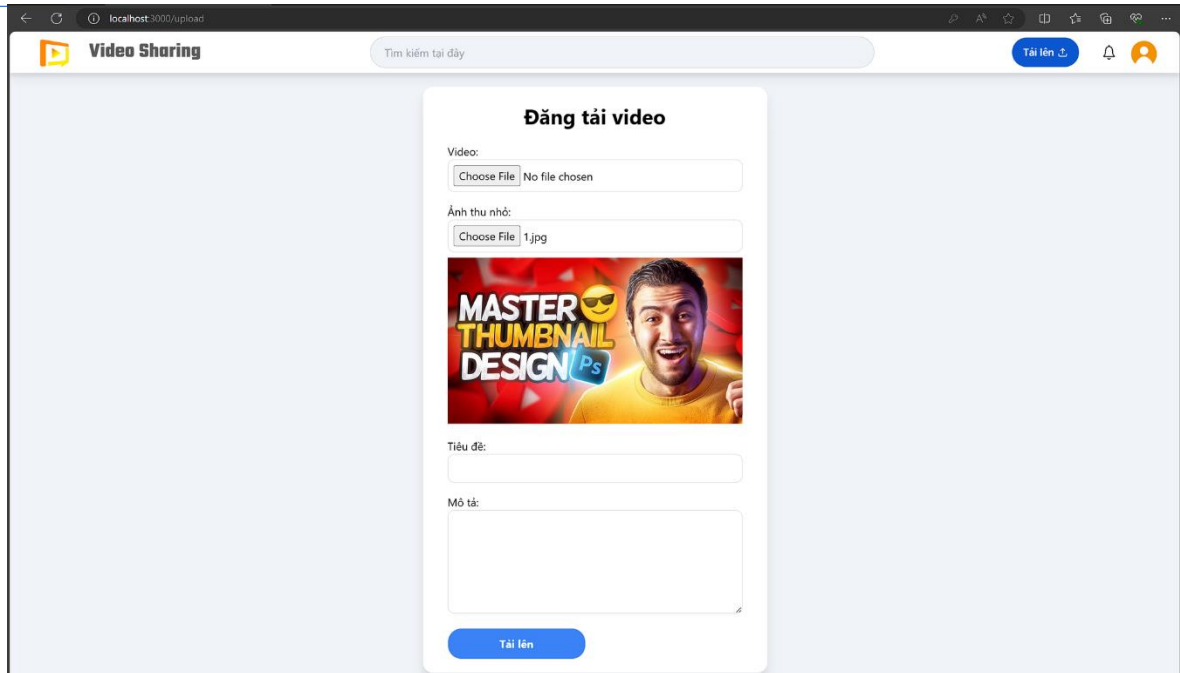
Hình 12 – Màn hình Profile cá nhân



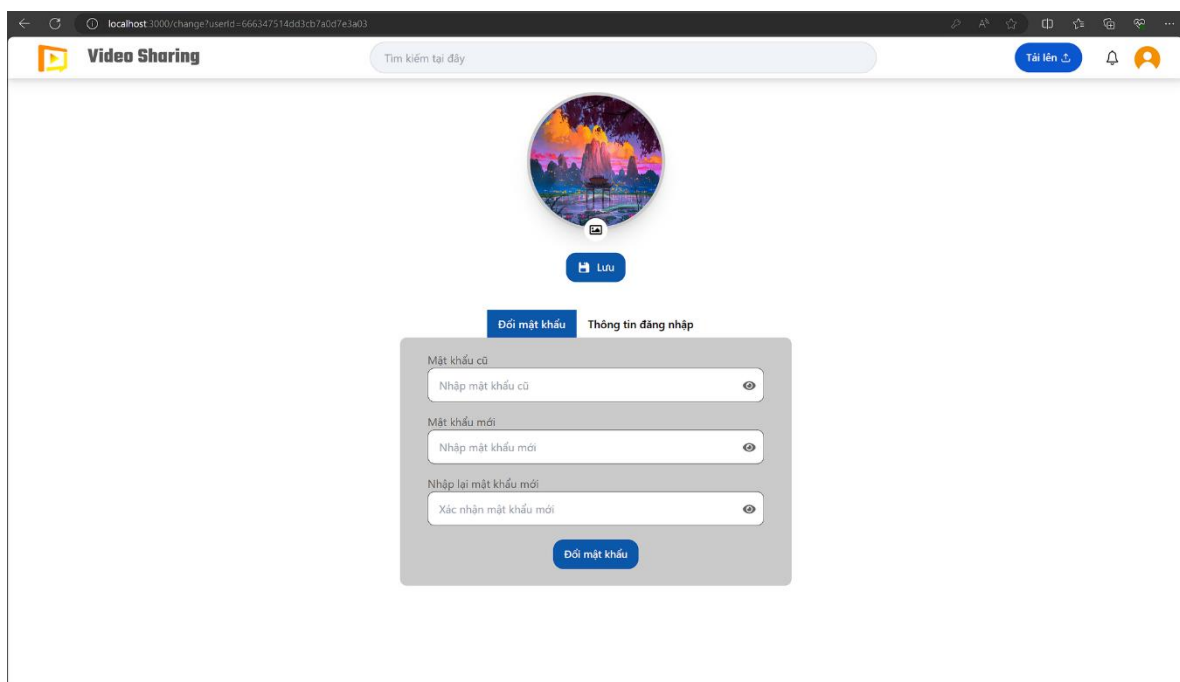
Hình 13 – Màn hình video đã like



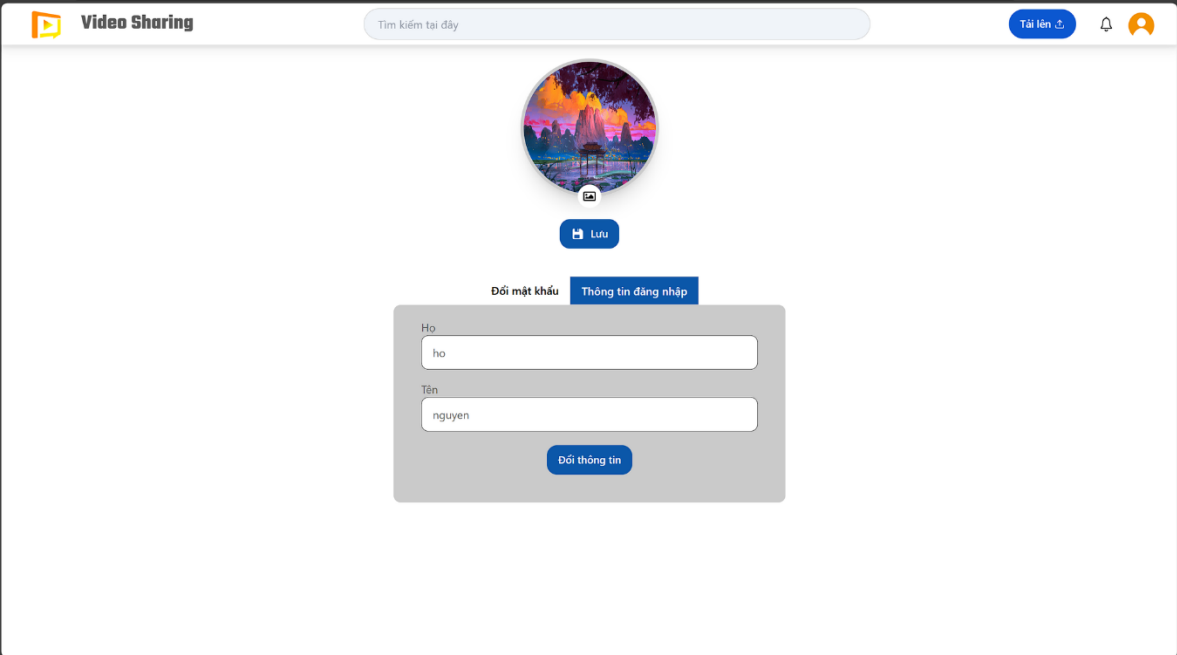
Hình 14 – Màn hình khi xem video



Hình 15 – Màn hình đăng tải video



Hình 16 – Màn hình thay đổi mật khẩu



Hình 17 – Màn hình thay đổi thông tin cá nhân

3. Phân chia công việc

Họ và Tên	Công việc	Hoàn Thành (%)
Nguyễn Thành Đăng	-Thiết kế cơ sở dữ liệu (nosql) -Xây dựng giao diện -Xây dựng API cho User Service -Kiểm thử hệ thống	100 %
Tạ Đức Bảo		
Trịnh Tấn Đạt	-Xây dựng API cho Comment Service, Video Service -Thiết kế giao diện -Kiểm thử hệ thống (Test Case)	100 %
Nguyễn Trần Bảo Quốc		

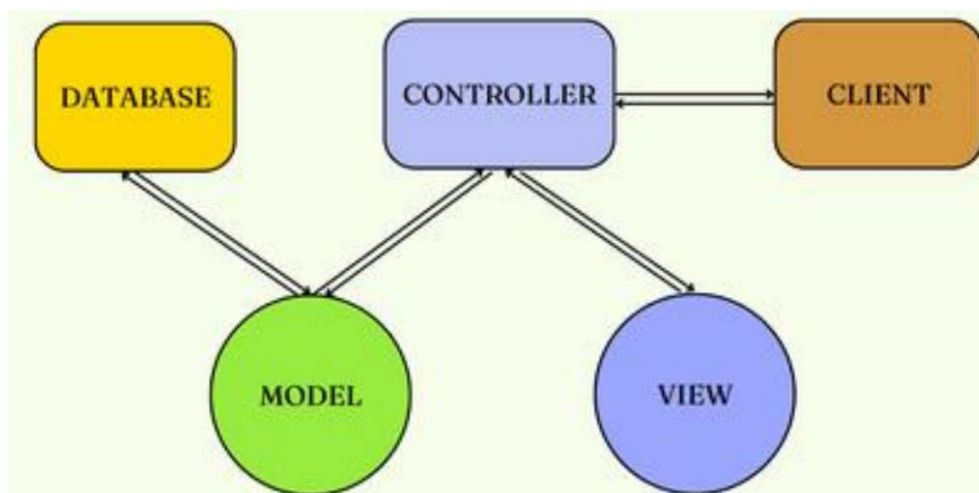
Bảng 1 – Phân công công việc

F. CHƯƠNG 6: HIỆN THỰC ĐỀ TÀI

1. Back-end

a. API cho ứng dụng

Server được xây dựng dựa trên ngôn ngữ Java, thư viện Spring Boot cho phép xây dựng các API một cách nhanh chóng. Server sử dụng mô hình MVC (Model – View – Controller). Một mô hình khá phổ biến hiện nay, mô hình này giúp tổ chức mã nguồn một cách có cấu trúc dễ dàng sửa lỗi và nâng cấp sau này.



Hình 18 – Mô hình MVC

b. Kết nối API với các công nghệ sử dụng

Để có thể kết nối dữ liệu lên MongoDB ta cần kết nối:

```
spring.data.mongodb.uri=mongodb+srv://21520683:<Password>@cluster.sijm2n1.mongodb.net/streaming?retryWrites=true&w=majority&appName=Cluster
```

c. Viết API

Một số API chức năng quan trọng đã viết:

- **API Đăng nhập:** Nếu người dùng đăng nhập thành công sẽ gửi một token để tăng tính bảo mật, còn nếu tài khoản hoặc mật khẩu không chính xác thì sẽ thông báo đăng nhập không thành công và gửi mã là **User not found**.

```
@PostMapping("/login")
public ResponseEntity loginUser(@RequestBody AuthUser user) {
    try {
        AuthUser userFromDb = userRepository.findByUsername(user.getUsername())
            .orElseThrow(() -> new Exception("User not found"));
        if (passwordEncoder.matches(user.getPassword(), userFromDb.getPassword()))
        {
            return ResponseEntity.ok(userFromDb);
        } else {
            return ResponseEntity.status(HttpStatus.UNAUTHORIZED).body("Invalid
credentials");
        }
    } catch (Exception e) {
        return ResponseEntity.internalServerError().body(e.getMessage());
    }
}
```

- **API Đăng ký:** Có 2 quá trình đăng ký tài khoản.
 - **Endpoint /register:** Kiểm tra sự tồn tại của username và email được nhập.

```
@PostMapping("/register")
public ResponseEntity registerUser(@RequestBody AuthUser user) {
    try {
        if (userRepository.findByUsername(user.getUsername()).isPresent())
            return ResponseEntity.status(HttpStatus.CONFLICT).body("Username
already taken. Please try again");
        user.setPassword(passwordEncoder.encode(user.getPassword()));
        user.setFirstName(user.getFirstName());
        user.setLastName(user.getLastName());
        user.setTimestamp(new Date());
        user.setAvatar(getDefaultAvatar());
        AuthUser save = userRepository.save(user);
        return ResponseEntity.ok(save);
    } catch (Exception e) {
        return ResponseEntity.internalServerError().body(e.getMessage());
    }
}
```

Mô tả:

API này nhận thông tin đăng ký từ người dùng, kiểm tra xem username và email đã tồn tại chưa. Nếu chưa tồn tại, mã hóa mật khẩu và lưu thông tin người dùng vào cơ sở dữ liệu.

- **Endpoint /send-verification-email:** Gửi email xác nhận đến email đăng ký.

```
@PostMapping("/send-verification-email")
public String sendVerificationEmail(@RequestBody String emailJson) {
    SimpleMailMessage message = new SimpleMailMessage();
    Gson gson = new Gson();
    Email emailObject = gson.fromJson(emailJson, Email.class);
    String email = emailObject.getEmail();
    message.setTo(email);
    message.setSubject("Xác thực đăng ký");
    message.setText("Xin chào, Bạn đã đăng ký tài khoản thành công!");

    try {
        javaMailSender.send(message);
        return "Email xác thực đã được gửi thành công";
    } catch (MailException e) {
        return "Gửi email xác thực thất bại: " + e.getMessage();
    }
}
```

Mô tả:

API này nhận email đăng ký và gửi một email xác nhận đến người dùng.

- **API thêm 1 bình luận**

```
@PostMapping("/upload")
public ResponseEntity<?> uploadComment(@RequestBody Comment comment) {
    try {
        Comment save = commentRepository.save(comment);
        return ResponseEntity.ok(HttpStatus.CREATED);
    } catch (Exception e) {
        return ResponseEntity.internalServerError().body(e.getMessage());
    }
}
```

Mô tả:

API này nhận dữ liệu bình luận từ người dùng và lưu vào cơ sở dữ liệu.

- **API trả lời bình luận:**

```
@PostMapping("/replyCommentByCommentId/{commentId}")
public ResponseEntity<?> replyCommentByCommentId(@PathVariable String commentId,
@RequestBody Comment comment) {
    try {
        Comment commentFromDb = commentRepository.findById(commentId)
            .orElseThrow(() -> new Exception("Comment not found"));
        comment.setVideoId(commentFromDb.getVideoId());
        comment.setNumber(commentFromDb.getNumber() + 1);
        Comment save = commentRepository.save(comment);
        return ResponseEntity.ok(HttpStatus.CREATED);
    } catch (Exception e) {
        return ResponseEntity.internalServerError().body(e.getMessage());
    }
}
```

Mô tả:

API này nhận commentId của bình luận cần trả lời và dữ liệu bình luận mới, sau đó lưu bình luận mới vào cơ sở dữ liệu.

- **API upload file:**

```
@PostMapping("/upload")
public ResponseEntity<?> upload(@RequestParam("file") MultipartFile file,
    @RequestParam("userID") String userID,
    @RequestParam("thumbnail") MultipartFile thumbnailFile) throws IOException
{
    byte[] thumbnail = thumbnailFile.getBytes();
    Timestamp timestamp = new Timestamp(System.currentTimeMillis());
    return new ResponseEntity<>(videoService.addVideo(file, userID, thumbnail,
timestamp), HttpStatus.OK);
}
```

Mô tả:

API này nhận file video, userID và thumbnail từ người dùng, sau đó lưu vào cơ sở dữ liệu và trả về kết quả.

- **API get file:**

```
@GetMapping("/get/{id}")
public ResponseEntity<> download(@PathVariable String id,
    @RequestHeader(value = HttpHeaders.RANGE, required = false) String
rangeHeader) throws IOException {
    VideoWithStream videoWithStream = videoService.getVideoWithStream(id);
    if (videoWithStream == null) {
        return new ResponseEntity<>(HttpStatus.NOT_FOUND);
    }

    GridFSFile gridFSFile = videoWithStream.getGridFSFile();
    InputStream inputStream = videoWithStream.getInputStream();

    long fileSize = gridFSFile.getLength();
    HttpHeaders headers = new HttpHeaders();

    headers.setContentType(MediaType.parseMediaType(gridFSFile.getMetadata().get("_contentType").toString()));
    headers.setContentDisposition(ContentDisposition.builder("inline")
        .filename(UriUtils.encodePath(gridFSFile.getFilename()),
StandardCharsets.UTF_8))
        .build());

    if (rangeHeader == null) {
        return ResponseEntity.ok()
            .headers(headers)
            .contentType(MediaType.parseMediaType(gridFSFile.getMetadata().get("_contentType").toString()))
            .body(new InputStreamResource(inputStream));
    }

    String[] ranges = rangeHeader.replace("bytes=", "").split("-");
    long rangeStart = Long.parseLong(ranges[0]);
    long rangeEnd = ranges.length > 1 ? Long.parseLong(ranges[1]) : fileSize - 1;
    if (rangeEnd > fileSize - 1) {
        rangeEnd = fileSize - 1;
    }
    long rangeLength = rangeEnd - rangeStart + 1;

    inputStream.skip(rangeStart);
    InputStreamResource inputStreamResource = new InputStreamResource(
        new LimitedInputStream(inputStream, rangeLength));

    headers.add("Content-Range", "bytes " + rangeStart + "-" + rangeEnd + "/" +
fileSize);
    headers.setContentLength(rangeLength);

    return ResponseEntity.status(HttpStatus.PARTIAL_CONTENT)
        .headers(headers)
        .body(inputStreamResource);
}
```

Mô tả:

API này nhận id của video và trả về file video cùng với các thông tin header tương ứng.

- API theo dõi user

```
@CrossOrigin(origins = "*")
@PostMapping("/subscribe")
public ResponseEntity<?> subscribe(@RequestParam("subscriberId") String
subscriberId,
    @RequestParam("subscribedToId") String subscribedToId) {
    videoService.subscribe(subscriberId, subscribedToId);
    return new ResponseEntity<>(HttpStatus.OK);
}
```

Mô tả:

API này nhận subscriberId và subscribedToId để người dùng có thể theo dõi một người dùng khác.

- API bày tỏ cảm xúc với video

```
@CrossOrigin(origins = "*")
@PostMapping("/like")
public ResponseEntity<?> like(@RequestParam("likerToId") String likerToId,
    @RequestParam("likedToId") String likedToId) {
    videoService.like(likerToId, likedToId);
    return new ResponseEntity<>(HttpStatus.OK);
}
```

Mô tả:

API này nhận likerToId và likedToId để người dùng có thể bày tỏ cảm xúc (like) đối với một video.

- API Số lượng subscriber của một user

```
@CrossOrigin(origins = "*")
@GetMapping("/getSubscriberCount")
public ResponseEntity<Long> getSubscriberCount(@RequestParam("userId") String
userId) {
    long subscriberCount = videoService.getSubscriberCount(userId);
    return new ResponseEntity<>(subscriberCount, HttpStatus.OK);
}
```

Mô tả:

API này nhận userId của một người dùng và trả về số lượng người dùng đã subscribe (theo dõi) người này.

- API Số lượng like 1 video

```
@CrossOrigin(origins = "*")
@GetMapping("/getLikeCount")
public ResponseEntity<Long> getLikeCount(@RequestParam("videoId") String videoId) {
    long likeCount = videoService.getLikeCount(videoId);
    return new ResponseEntity<>(likeCount, HttpStatus.OK);
}
```

Mô tả:

API này nhận videoId của một video và trả về số lượng lượt like của video đó.

- API đổi mật khẩu

```
@CrossOrigin(origins = "*")
@PutMapping("/changePassword/{id}")
public ResponseEntity changePassword(@PathVariable("id") String id,
    @RequestBody ChangePasswordRequest changePasswordRequest) {
    try {
        AuthUser userFromDb = userRepository.findById(id)
            .orElseThrow(() -> new Exception("User not found"));

        if (!passwordEncoder.matches(changePasswordRequest.getCurrentPassword(),
            userFromDb.getPassword())) {
            return ResponseEntity.status(HttpStatus.UNAUTHORIZED).body("Current
            password is incorrect");
        }

        userFromDb.setPassword(passwordEncoder.encode(changePasswordRequest.getNewPassword()));
        AuthUser save = userRepository.save(userFromDb);

        return ResponseEntity.ok("Password changed successfully");
    } catch (Exception e) {
        return ResponseEntity.internalServerError().body(e.getMessage());
    }
}
```

Mô tả:

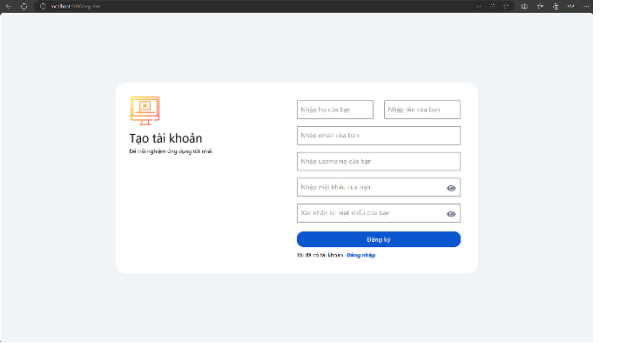
API này nhận id của người dùng cần thay đổi mật khẩu và thông tin yêu cầu thay đổi mật khẩu từ người dùng. Sau đó kiểm tra mật khẩu hiện tại của người dùng, nếu đúng thì thay đổi mật khẩu và lưu vào cơ sở dữ liệu.

2. Front-end

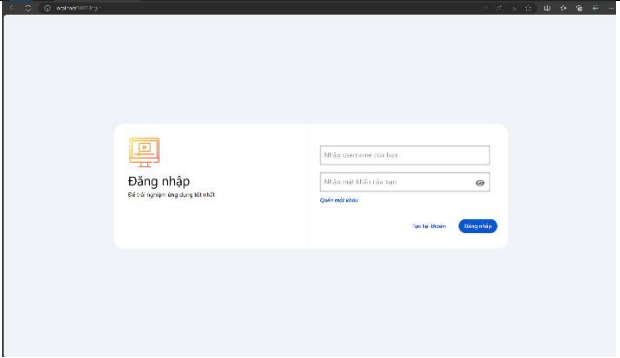
a. Các giao diện chính

Màn hình trang chủ	Mô tả
	<p>Thanh Điều Hướng:</p> <p>Logo/Trang Chủ: Dẫn người dùng trở về trang chủ từ bất kỳ đâu trong ứng dụng.</p> <p>Liên Kết Tới Các Trang Khác: Bao gồm các liên kết đến Trang Chủ, Tải Lên Video, Hồ Sơ Cá Nhân, và Đăng Xuất (nếu người dùng đã đăng nhập) hoặc Đăng Nhập/Đăng Ký (nếu chưa đăng nhập).</p> <p>Danh Sách Video:</p> <p>Hiển Thị Video: Mỗi video hiển thị hình thu nhỏ, tiêu đề, người đăng và số lượt xem.</p> <p>Sắp Xếp Video: Video có thể được sắp xếp theo các tiêu chí như mới nhất, phổ biến nhất, hoặc được đề xuất dựa trên lịch sử xem của người dùng.</p> <p>Thanh Tìm Kiếm:</p> <p>Tìm Kiếm Nhanh: Cho phép người dùng tìm kiếm video dựa trên tiêu đề hoặc từ khóa.</p> <p>Gợi Ý Tìm Kiếm: Hiển thị gợi ý khi người dùng nhập từ khóa tìm kiếm.</p>

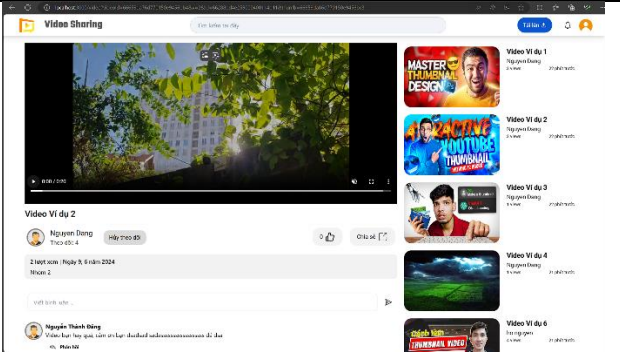
Bảng 2 – Màn hình trang chủ

Màn hình Đăng ký tài khoản	Mô tả
	<p>Trường Nhập Liệu: Bao gồm các trường cho tên, email, tên đăng nhập, mật khẩu và xác nhận mật khẩu.</p> <p>Nút Đăng Ký: Người dùng nhấn vào để gửi thông tin đăng ký.</p> <p>Điều Khoản Dịch Vụ: Người dùng cần chấp nhận điều khoản dịch vụ và chính sách bảo mật trước khi đăng ký.</p>

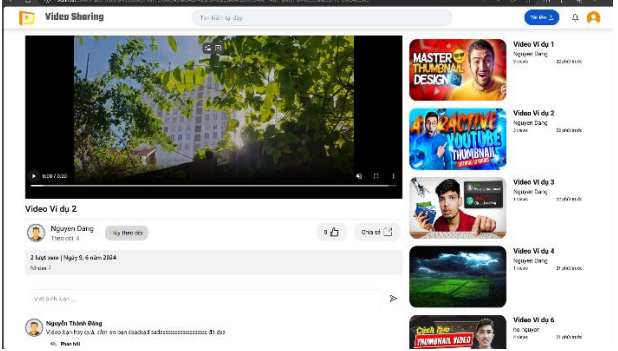
Bảng 3 – Màn hình Đăng ký tài khoản

Màn hình Đăng nhập	Mô tả
	<p>Biểu Mẫu Đăng Nhập:</p> <p>Trường Nhập Liệu: Bao gồm các trường cho tên đăng nhập hoặc email và mật khẩu.</p> <p>Nút Đăng Nhập: Người dùng nhấn vào để gửi thông tin đăng nhập.</p> <p>Nhớ Mật Khẩu: Tùy chọn để người dùng lưu thông tin đăng nhập trên trình duyệt của họ.</p> <p>Liên Kết Đăng Ký:</p> <p>Chuyển Hướng Đăng Ký: Liên kết dành cho những người dùng chưa có tài khoản để chuyển đến màn hình đăng ký.</p>

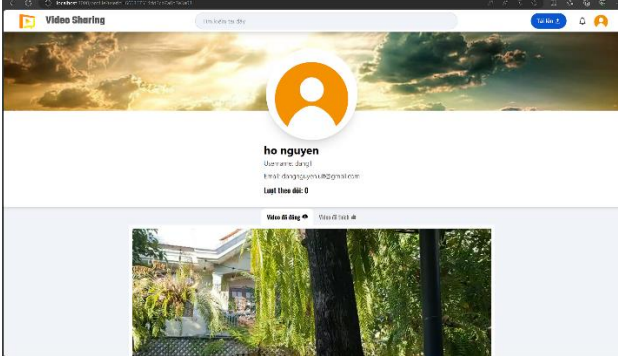
Bảng 4 – Màn hình Đăng nhập

Màn hình xem Video	Mô tả
	<p>Video Player:</p> <p>Trình Phát Video: Hiển thị video được chọn với các nút điều khiển như phát/tạm dừng, tua nhanh/chậm, và âm lượng.</p> <p>Tiêu Đề và Mô Tả Video:</p> <p>Hiển Thị Thông Tin: Hiển thị tiêu đề và mô tả chi tiết của video, cùng với thông tin về người đăng và ngày tải lên.</p> <p>Thích và Không Thích:</p> <p>Nút Thích/Không Thích: Cho phép người dùng thể hiện ý kiến của mình về video.</p> <p>Bình Luận:</p> <p>Danh Sách Bình Luận: Hiển thị các bình luận từ người dùng khác.</p> <p>Biểu Mẫu Thêm Bình Luận: Cho phép người dùng thêm bình luận mới.</p> <p>Danh Sách Video Liên Quan:</p> <p>Hiển Thị Video Liên Quan: Hiển thị các video liên quan hoặc được đề xuất dựa trên nội dung hiện tại.</p>

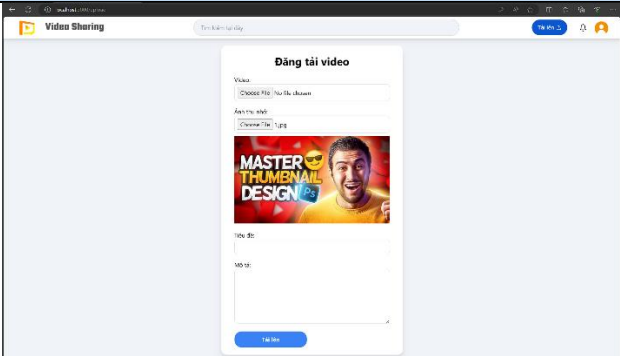
Bảng 5 – Màn hình xem Video

Màn hình bình luận video	Mô tả
	<p>Danh Sách Bình Luận:</p> <p>Hiển Thị Bình Luận: Hiển thị các bình luận đã có cho video, bao gồm tên người bình luận và nội dung bình luận.</p> <p>Sắp Xếp Bình Luận: Bình luận có thể được sắp xếp theo thời gian hoặc lượt thích.</p> <p>Biểu Mẫu Thêm Bình Luận:</p> <p>Trường Nhập Liệu: Bao gồm một trường nhập liệu cho nội dung bình luận.</p> <p>Nút Gửi: Người dùng nhấn vào để gửi bình luận.</p>

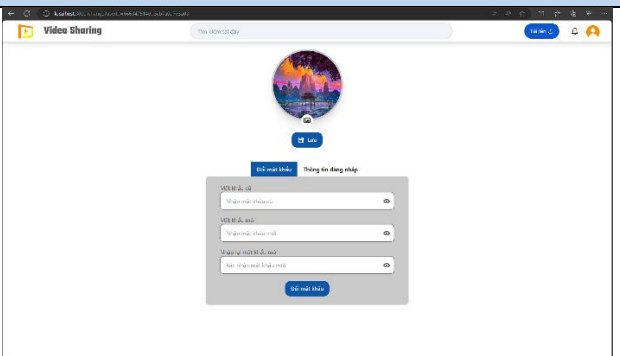
Bảng 6 – Màn hình bình luận video

Màn hình profile	Mô tả
	<p>Thông Tin Cá Nhân:</p> <p>Hiển Thị Thông Tin: Hiển thị tên, email, tên đăng nhập, và ảnh đại diện của người dùng.</p> <p>Chỉnh Sửa Thông Tin: Cung cấp tùy chọn để người dùng chỉnh sửa các thông tin cá nhân của họ.</p> <p>Danh Sách Video Đã Tải Lên:</p> <p>Hiển Thị Video: Hiển thị các video mà người dùng đã tải lên, cùng với các thông tin như lượt xem, lượt thích, và bình luận.</p> <p>Tùy Chọn Bảo Mật:</p> <p>Thay Đổi Mật Khẩu: Cho phép người dùng thay đổi mật khẩu hiện tại.</p> <p>Bảo Mật Tài Khoản: Cung cấp các tùy chọn bảo mật bổ sung</p>

Bảng 7 – Màn hình hồ sơ người dùng

Màn hình tải lên video	Mô tả
	<p>Biểu Mẫu Tải Lên Video:</p> <p>Trường Nhập Liệu: Bao gồm các trường cho tiêu đề video, mô tả, chọn tệp video và thẻ (tags).</p> <p>Nút Tải Lên: Người dùng nhấn vào để tải video lên.</p> <p>Xem Trước Video:</p> <p>Hiển Thị Xem Trước: Cho phép người dùng xem trước video trước khi tải lên để kiểm tra chất lượng và nội dung.</p>

Bảng 8 – Màn hình tải lên video

Màn hình đổi mật khẩu	Mô tả
	<p>Biểu Mẫu Đổi Mật Khẩu:</p> <p>Mật Khẩu Hiện Tại: Trường nhập liệu yêu cầu người dùng nhập mật khẩu hiện tại của họ để xác thực.</p> <p>Mật Khẩu Mới: Trường nhập liệu cho mật khẩu mới mà người dùng muốn đặt.</p> <p>Xác Nhận Mật Khẩu Mới: Trường nhập liệu để người dùng nhập lại mật khẩu mới nhằm xác nhận.</p> <p>Nút Đổi Mật Khẩu: Người dùng nhấn vào để gửi yêu cầu đổi mật khẩu.</p> <p>Yêu Cầu Bảo Mật:</p> <p>Mật Khẩu Mạnh: Gợi ý cho người dùng tạo mật khẩu mạnh bao gồm chữ hoa, chữ thường, số và ký tự đặc biệt.</p> <p>Phản Hồi Người Dùng:</p> <p>Thông Báo Thành Công/Thất Bại: Hiển thị thông báo cho người dùng biết kết quả của yêu cầu đổi mật khẩu.</p>

Bảng 9 – Màn hình đổi mật khẩu

Màn hình thông tin cá nhân	Mô tả
	<p>Biểu Mẫu Thay Đổi Thông Tin Cá Nhân:</p> <p>Tên: Trường nhập liệu cho tên đầy đủ của người dùng.</p> <p>Email: Trường nhập liệu cho địa chỉ email.</p> <p>Tên Đăng Nhập: Trường nhập liệu cho tên đăng nhập.</p> <p>Ảnh Đại Diện: Cho phép người dùng tải lên hoặc thay đổi ảnh đại diện.</p> <p>Nút Lưu Thay Đổi: Người dùng nhấn vào để lưu các thay đổi.</p> <p>Phản Hồi Người Dùng:</p> <p>Thông Báo Thành Công/Thất Bại: Hiển thị thông báo cho người dùng biết kết quả của việc lưu thay đổi thông tin cá nhân.</p>

Bảng 10 – Màn hình đổi thông tin cá nhân

3. Kết nối Front-end và Back-end

a. Giới thiệu

Việc kết nối giữa Frontend và Backend là một phần quan trọng trong việc xây dựng ứng dụng chia sẻ video của nhóm chúng em. Frontend được xây dựng bằng React, cung cấp giao diện người dùng tương tác, trong khi Backend được phát triển bằng Spring Boot, chịu trách nhiệm xử lý logic nghiệp vụ và quản lý cơ sở dữ liệu. Dưới đây là chi tiết về cách chúng em thiết lập và duy trì kết nối giữa hai thành phần này.

b. Cấu trúc API

Backend cung cấp các API RESTful để giao tiếp với Frontend. Các API này tuân thủ các chuẩn REST, giúp việc truyền tải dữ liệu giữa Frontend và Backend trở nên dễ dàng và hiệu quả.

HTTP Methods: Sử dụng các phương thức HTTP như GET, POST, PUT, DELETE để thực hiện các thao tác CRUD (Create, Read, Update, Delete).

Endpoints: Các endpoints được thiết kế rõ ràng và có cấu trúc logic, ví dụ một vài api trong đồ án của chúng em.

- /user/: Quản lý thông tin người dùng
- /video/: Quản lý video
- /comment/: Quản lý bình luận
- /like/: Quản lý lượt thích

c. Quy trình xác thực và phân quyền

Nhóm chúng em sử dụng JSON Web Tokens (JWT) để xác thực và phân quyền người dùng. Khi người dùng đăng nhập, Backend sẽ tạo và trả về một JWT, và Frontend sẽ lưu trữ token này, chúng em đang sử dụng Local Storage.

- **Xác Thực:**

Khi người dùng đăng nhập, Frontend sẽ gửi yêu cầu POST tới `/api/auth/login` với thông tin xác thực.

Backend sẽ kiểm tra thông tin và trả về một JWT nếu thông tin hợp lệ.

JWT được lưu trữ trên Frontend và được gửi kèm theo trong header của các yêu cầu tiếp theo để xác thực.

- **Phân Quyền:**

Backend sẽ kiểm tra JWT trong mỗi yêu cầu để xác định quyền truy cập của người dùng và cho phép hoặc từ chối các thao tác tương ứng.

d. Quản lý trạng thái và giao tiếp

Axios: Chúng em sử dụng thư viện Axios trên Frontend để gửi các yêu cầu HTTP tới Backend.

State Management: Sử dụng Context API để quản lý trạng thái toàn cục trên Frontend, giúp đồng bộ hóa dữ liệu giữa các thành phần giao diện người dùng.

Handling Responses: Frontend xử lý phản hồi từ Backend, cập nhật giao diện người dùng tương ứng và hiển thị thông báo lỗi khi cần thiết.

e. Kết luận

Việc kết nối Frontend và Backend của đồ án video sharing được thiết kế một cách cẩn thận để đảm bảo sự linh hoạt, bảo mật và hiệu suất cao. Các API RESTful, cơ chế xác thực JWT đã giúp chúng em tạo ra một nền tảng mạnh mẽ, hỗ trợ tốt cho người dùng trong việc chia sẻ và tương tác với nội dung video.

G. CHƯƠNG 7: KIỂM THỬ HỆ THỐNG

1. Kiểm thử Comment Service

a. Kiểm tra comment service có hoạt động không

```
@Autowired
private MockMvc mockMvc;

@Test
public void testGetServiceName() throws Exception {
    mockMvc.perform(get("/comment/"))
        .andExpect(status().isOk());
}
```

Mô tả:

- Đoạn mã này kiểm tra xem Comment Service có hoạt động đúng hay không bằng cách gửi yêu cầu GET đến endpoint /comment/ và mong đợi phản hồi HTTP 200 (OK).

b. Kiểm tra xem comment service đã kết nối tới cơ sở dữ liệu MongoDB chưa

```
@Test
public void testUploadComment() throws Exception {
    Comment comment = Comment.builder()
        .text("text")
        .author("author")
        .build();
    String json = "{ \"text\": \"text\", \"author\": \"author\" }";
    mockMvc.perform(post("/comment/upload")
        .contentType("application/json")
        .content(json)
        .andExpect(status().isOk());
}
```

Mô tả:

- Đoạn mã này kiểm tra xem Comment Service có kết nối đến cơ sở dữ liệu MongoDB hay không bằng cách gửi yêu cầu POST với dữ liệu comment đến endpoint /comment/upload và mong đợi phản hồi HTTP 200 (OK).

2. Kiểm thử Video Service

a. Kiểm tra video service có hoạt động hay không

```
@Autowired
private MockMvc mockMvc;

@Test
public void testGetVideoService() throws Exception {
    mockMvc.perform(get("/video/"))
        .andExpect(status().isOk());
}
```

Mô tả:

- Đoạn mã này kiểm tra xem Video Service có hoạt động đúng hay không bằng cách gửi yêu cầu GET đến endpoint /video/ và mong đợi phản hồi HTTP 200 (OK).

b. Kiểm tra xem video service đã kết nối tới cơ sở dữ liệu MongoDB chưa

```
@Autowired
private MockMvc mockMvc;
@Autowired
private ObjectMapper objectMapper;

@Test
public void testUploadVideo() throws Exception {
    Video video = new Video();
    video.setFilename("example.mp4");
    video.setFileType("video/mp4");
    video.setFileSize("1024");
    video.setDescription("Test video");
    video.setUserID("userID");
    video.setUserName("userName");
    video.setVideoName("videoName");
    String json = objectMapper.writeValueAsString(video);
    mockMvc.perform(post("/video/upload")
        .contentType("application/json")
        .content(json)
        .andExpect(status().isOk());
}
```

Mô tả:

- Đoạn mã này kiểm tra xem Video Service có kết nối đến cơ sở dữ liệu MongoDB hay không bằng cách gửi yêu cầu POST với dữ liệu video đến endpoint /video/upload và mong đợi phản hồi HTTP 200 (OK).

3. Kiểm thử User Service

a. Kiểm tra user service có hoạt động hay không

```
@Autowired
private MockMvc mockMvc;

@Test
public void testGetUserService() throws Exception {
    mockMvc.perform(get("/user/"))
        .andExpect(status().isOk());
}
```

Mô tả:

- Đoạn mã này kiểm tra xem User Service có hoạt động đúng hay không bằng cách gửi yêu cầu GET đến endpoint /user/ và mong đợi phản hồi HTTP 200 (OK).

b. Kiểm tra xem user service đã kết nối tới cơ sở dữ liệu MongoDB chưa

```
@Autowired
private MockMvc mockMvc;

@Autowired
private ObjectMapper objectMapper;

@Test
public void testUploadUser() throws Exception {
    AuthUser user = AuthUser.builder()
        .username("testuser")
        .password("password")
        .active(true)
        .firstName("First")
        .lastName("Last")
        .email("test@example.com")
        .build();

    String json = objectMapper.writeValueAsString(user);

    mockMvc.perform(post("/user/register")
        .contentType("application/json")
        .content(json)
        .andExpect(status().isOk());
}
```

Mô tả:

- Đoạn mã này kiểm tra xem User Service có kết nối đến cơ sở dữ liệu MongoDB hay không bằng cách gửi yêu cầu POST với dữ liệu user đến endpoint /user/register và mong đợi phản hồi HTTP 200 (OK).

H. CHƯƠNG 8: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

1. Kết luận

Đồ án video sharing đã hoàn thành việc quan trọng trong việc xây dựng một nền tảng cho phép người dùng đăng ký tài khoản, upload video, bình luận và like video. Chúng em đã hoàn thiện các tính năng cơ bản và đảm bảo rằng người dùng có thể trải nghiệm một cách mượt mà và hiệu quả. Việc triển khai các chức năng này không chỉ mang lại cho người dùng khả năng tương tác đa dạng mà còn tạo nên một cộng đồng chia sẻ video sôi động và phong phú.

Nhờ vào việc sử dụng những thư viện cũng như phương thức hợp lý trong quá trình phát xây dựng ứng dụng, chúng em đã tối ưu hóa hệ thống để đảm bảo tính bảo mật, ổn định và dễ dàng mở rộng. Các thử nghiệm và phản hồi từ người dùng có thể giúp chúng em cải thiện giao diện người dùng và trải nghiệm tổng thể, góp phần vào sự hài lòng và gắn kết của người dùng với nền tảng.

2. Hướng phát triển

Trong tương lai, chúng em sẽ tiếp tục phát triển và mở rộng nền tảng video sharing với các kế hoạch cụ thể như sau:

a. Xây dựng ứng dụng đa nền tảng (web, iOS, Android):

Chúng em sẽ phát triển ứng dụng trên cả ba nền tảng web, iOS và Android để người dùng có thể truy cập và sử dụng dịch vụ ở bất cứ đâu và trên bất kỳ thiết bị nào. Việc này không chỉ giúp mở rộng đối tượng người dùng mà còn tăng cường khả năng tiếp cận và tiện lợi cho người dùng hiện tại. Các tính năng chính như đăng ký tài khoản, upload video, bình luận, và like video sẽ được tối ưu hóa để hoạt động mượt mà trên tất cả các nền tảng này.

b. Huấn luyện một mô hình có thể nhận biết và tự động gỡ bỏ các video vi phạm:

Chúng em sẽ phát triển và huấn luyện một mô hình trí tuệ nhân tạo (AI) nhằm nhận diện các video vi phạm nội quy, chẳng hạn như nội dung không phù hợp, vi phạm bản quyền, hoặc nội dung độc hại. Mô hình này sẽ sử dụng công nghệ machine learning để phân tích nội dung video và tự động gỡ bỏ những video không đạt tiêu chuẩn, đảm bảo môi trường an toàn và lành mạnh cho người dùng. Việc này sẽ giúp giảm tải công việc kiểm duyệt thủ công và nâng cao hiệu quả quản lý nội dung.

c. Phát triển một hệ thống socket:

Chúng em sẽ triển khai một hệ thống socket cho phép người dùng tạo và tham gia vào các phòng xem video chung. Tính năng này sẽ giúp người dùng có thể cùng nhau xem video theo nhóm, chia sẻ cảm xúc và thảo luận trực tiếp trong thời gian thực. Việc phát triển này không chỉ tạo ra một trải nghiệm xem video thú vị và tương tác hơn mà còn giúp xây dựng một cộng đồng gắn kết và năng động. Các tính năng hỗ trợ như chat, gửi biểu tượng cảm xúc, và chia sẻ playlist cũng sẽ được tích hợp để tăng cường trải nghiệm của người dùng trong phòng xem video chung.

Những định hướng phát triển này không chỉ nhằm cải thiện và mở rộng nền tảng, mà còn hướng tới việc xây dựng một cộng đồng chia sẻ video lớn mạnh và bền vững và không ngừng nỗ lực để mang đến những trải nghiệm tốt nhất.

I. CHƯƠNG 9: PHỤ LỤC

Phụ lục 1: Source code Backend: [Video-Sharing-Project-UIT/Video-Sharing-Backend \(github.com\)](https://github.com/Video-Sharing-Project-UIT/Video-Sharing-Backend)

Phụ lục 2: Source code Frontend: [Video-Sharing-Project-UIT/Video-Sharing-Frontend \(github.com\)](https://github.com/Video-Sharing-Project-UIT/Video-Sharing-Frontend)

TÀI LIỆU THAM KHẢO

- [1] [Security with Spring | Baeldung](https://www.baeldung.com/security-spring)
<https://www.baeldung.com/security-spring>

- [2] [Spring Boot with MongoDB: A Step-by-Step Tutorial \(javaguides.net\)](https://www.javaguides.net/2024/05/spring-boot-with-mongodb-step-by-step.html)
<https://www.javaguides.net/2024/05/spring-boot-with-mongodb-step-by-step.html>

- [3] [Spring Boot File upload example with Multipart File - BezKoder](https://www.bezkoder.com/spring-boot-file-upload/)
<https://www.bezkoder.com/spring-boot-file-upload/>

- [4] [Spring Boot @CrossOrigin Annotation Example - Java Code Geeks](https://examples.javacodegeeks.com/java-development/enterprise-java/spring/boot/spring-boot-crossorigin-annotation-example/)
<https://examples.javacodegeeks.com/java-development/enterprise-java/spring/boot/spring-boot-crossorigin-annotation-example/>

- [5] [How to make HTTP requests using RestTemplate in Spring Boot \(attacomsian.com\)](https://attacomsian.com/blog/http-requests-resttemplate-spring-boot)
<https://attacomsian.com/blog/http-requests-resttemplate-spring-boot>

- [6] [React JS + Spring Boot REST API Example Tutorial \(javaguides.net\)](https://www.javaguides.net/2020/07/react-js-spring-boot-rest-api-example-tutorial.html)
<https://www.javaguides.net/2020/07/react-js-spring-boot-rest-api-example-tutorial.html>

- [7] [25+ Spring and Spring Boot Annotations | 3 Hours Full Course | Interview Q&A \(youtube.com\)](https://www.youtube.com/watch?v=AXZkhKTbbWc)
<https://www.youtube.com/watch?v=AXZkhKTbbWc>

- [8] [Connecting Spring Boot to MongoDB Atlas: A Step-by-Step Tutorial \(youtube.com\)](https://www.youtube.com/watch?v=HjDyv7gL4Wg)
<https://www.youtube.com/watch?v=HjDyv7gL4Wg>

- [9] [java - Error occurred while connecting MongoDB Atlas and Spring Boot - Stack Overflow](https://stackoverflow.com/questions/77121129/error-occurred-while-connecting-mongodb-atlas-and-spring-boot)
<https://stackoverflow.com/questions/77121129/error-occurred-while-connecting-mongodb-atlas-and-spring-boot>