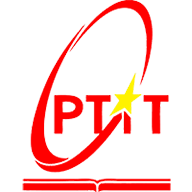


**HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG**

**KHOA CÔNG NGHỆ THÔNG TIN 1**



**MÔN HỌC: THỰC TẬP CƠ SỞ**

**ĐỀ TÀI: FORUM WEB**

**NỘI DUNG: HOÀN THÀNH TRANG WEB**

**Giảng Viên :** Đỗ Thị Bích Ngọc

**Sinh Viên:** Nguyễn Hoàng Tuấn Anh – B19DCCN025

**Hà Nội**

1. **Bảng kết hoạch**

|  |  |
| --- | --- |
| Kết thúc tuần | Công Việc Phải Hoàn Thành |
| Tuần 3 (25/3) | Xây dựng database, nghiên cứu phân tích thuyết kế, tạo đặc tả chi tiết |
| Tuần 4 (1/4) | Xây front-end cơ sở, học kiến thức về java spring boot |
| Tuần 5 (8/4) | Xây dựng đăng ký, đăng nhập,demo qua tính năng post bài |
| Tuần 6 (15/4) | Tính năng post bài |
| Tuần 7 (22/4) | Tương tác giữa các người dùng: cmt, thả cảm xúc, … |
| Tuần 8 (29/4) | Thêm tính năng xếp hạng (hoặc là hoàn thiện các chức năng còn thiếu) |
| Tuần 9 (6/5) | Thiết kế giao diện front-end, học kiến thức về boostrap |
| Tuần 10 (13/5) | Ghép Api , hoàn thiện nốt front-end, sửa các lỗi |
| Tuần 11 (20/5) | Kiểm thử phần mềm |
| Tuần 12 (31/5) | Hoàn thành web |

1. **Tự đánh giá kết quả từ tuần**

Tuần 3: Xây dựng database, nghiên cứu phân tích thiết kết, tạo đặc tả chi tiết

* Đánh giá: Đã hoàn thành tốt việc xây dưng database, nghiên cứu phân tích thiết kế được trang web Forum

Tuần 4: Xây dựng front-end cơ sở, học kiến thức về java spring boot

* Đánh giá: Đã hoàn thành việc xây dưng front-end cơ sở, biết được đâu là những nút cần sử dụng, xây dưng được khung của việc thiết kế ra giao diện trang web, đã tìm hiểu được những thứ cần sử dụng ở trong java spring boot

Tuần 5: Xây dựng đăng ký, đăng nhập,demo qua tính năng post bài

* Đánh giá: Đã hòan thành việc code những chức năng đăng nhập, đăng ký, đã demo thành công tính năng post bài viết. Hoàn thiện được những phần code cả về front-end và back-end về các tính năng trên. Đã hoàn thành tốt.

Tuần 6: Tính năng post bài

* Đánh giá: Đây là tính năng chính của ứng dụng em thực hiện code. Em đã chăm chút về đánh giá về việc đăng bài viết, cũng như những sự kiện tương tác với bài viết sẽ xảy ra như thế nào. Đã hoàn thành được tốt về việc. Hiện tại về bug đã được sửa chữa và phần mềm chạy rất mượt về phần tính năng đăng bài viết.

Tuần 7: Tương tác giữa các người dùng: cmt, thả cảm xúc, …

* Đánh giá: Đã hoàn thành được tính năng này, thực hiện bình luận vào bài viết cũng như thực hiện thả những hành động gọi là upvote downvote cho bài viết.

Tuần 8: Tính năng xếp hạng (hoặc là hoàn thiện các chức năng còn thiếu)

* Đánh giá: Tuần 8 thực hiện tính năng xếp hạng vẫn còn chưa thực hiện thành công việc này. Vì vẫn chưa theo kịp được công nghệ của mình thực hiện nên em chỉ thực hiện hoàn thành và sửa lại những tính năng mà em đã làm sai, hoặc có bug ở những phần trước đó.

Tuần 9: Thiết kế giao diện front-end, học kiến thức về boostrap

* Đánh giá: Mặc dù, thực hiện code phần back-end nhiều hơn. Nhưng em đã làm tốt việc thiết kế giao diện front-end. Trên quan điểm cá nhân của em thì việc code xong phần giao diện front-end đã vừa mắt nhìn và có đủ tính năng của một ứng dụng dành cho sinh viên. Đã học thêm được nhiều kiến thức trong boostrap.

Tuần 10: Ghép Api , hoàn thiện nốt front-end, sửa các lỗi

* Đánh giá: Em không thực hiện chuyển code thành Api vì như lí do đã trình bày ở tuần trước đó. Em không thực hiện code các framework liên quan đến front-end có thể kết nối với Api (như reactJS) nên e sẽ thực hiện không chuyển đổi code phàn back-end đẩy dữ liệu lên API. Thực hiện hoàn thiện nốt những giao diện của phần front-end còn thiếu ở trước đó.

Tuần 11: Kiểm thử phần mềm

* Đánh giá: Thực hiện kiểm tra các tính năng của phần mềm, kiểm tra về bug trong phần mềm, nhận thấy còn khá nhiều lỗi tiềm ẩn, và các chức năng vẫn còn thiếu chức năng xếp hạng, em vẫn đang cố gắng hoàn thiện trong sớm nhớ của tuần 11.

Đánh giá chung: Đã có cố gắng tìm hiểu công nghệ mới, đã có ý thức thực hiện làm báo cáo các tuần. Theo sát sườn kết hoạch tự lập ra. Đã hoàn thành 85% công việc. Học được nhiều công nghệ mới, kĩ thuật lập trình mới. Đã cải thiện được nhiều trong quá trình lập trình của bản thân.   
Khuyết điểm: Vẫn chưa hoàn thành 15% công việc còn lại vẫn còn phải cố gắng nhiều hơn.

1. **Sơ qua về đề tài**

Mục tiêu của em là làm việc một trang web đăng những bài hỏi đáp. Người hỏi có quyền đăng câu hỏi lên ở dạng công khai, và ai cũng có khả năng trả lời câu hỏi đó. Mục tiêu của trang web em hướng để đặt câu hỏi, cũng như trả lời câu hỏi liên quan đến lập trình, hoặc đời sống của một coder, xoay quanh các vấn đề IT.

1. **Những nội dung đã tìm hiểu**
2. **Spring boot**
   1. **Lý do chọn Spring Boot là công cụ phát triển**

Spring Boot sử dụng ngôn ngữ Java để thực hiện phát triển phần mềm. Trong đó ngôn ngữ Java là môn ngữ mà em học ở kì trước và có liên quan đến kì này ở bộ môn lập trình web. Spring Boot cũng là một công cụ rất mạnh hỗ trợ cho công việc lập trình web hiện nay. Nên em quyết định chọn công cụ này để thực hiện code lần thực tập cơ sở này.

Việc ứng dụng Spring Boot giúp các lập trình viên tập trung tốt hơn đến mục tiêu viết code của mình thay vì phải lo nghĩ nhiều đến các kỹ thuật code như với [Node.js](https://itnavi.com.vn/blog/nodejs-la-gi/).Bản thân Spring Boot chứa đựng nhiều các library bên trong, cấu trúc code cũng trở nên chuẩn mực hơn, em không cần phải quá bận tâm đến việc code làm sao cho thật tốt.

Do đó, em có nhiều thời gian hơn để tập trung vào sự logic của sản phẩm.Spring Boot là sự cải tiến tuyệt vời của Spring, cải thiện được rất nhiều khâu rườm rà đến từ Spring. Hơn thế, quá trình học hỏi và tiếp thu Spring Boot cũng trở nên dễ dàng hơn rất nhiều.

* 1. **Tổng quan giới thiệu về Spring Boot**

[**Spring Boot**](https://topdev.vn/viec-lam-it/spring-boot-kt4317) là một dự án phát triển bởi **JAV**(ngôn ngữ java) trong hệ sinh thái Spring framework. Nó giúp cho các lập trình viên chúng ta đơn giản hóa quá trình lập trình một ứng dụng với Spring, chỉ tập trung vào việc phát triển business cho ứng dụng.

Để phát triển một ứng dụng web cơ bản sử dụng Spring framework bạn sẽ cần ít nhất 5 công đoạn sau:

* Tạo một project sử dụng Maven với các dependency cần thiết của Spring MVC và Servlet API.
* Một tập tin web.xml để khai báo DispatcherServlet của Spring MVC.
* Một tập tin cấu hình của Spring MVC.
* Một class Controller trả về một trang khi có request đến.
* Cuối cùng là phải có một web server dùng để triển khai ứng dụng lên chạy.

Trong các công đoạn này, chỉ có công đoạn tạo một class Controller thì có thể khác cho các ứng dụng khác nhau vì mỗi ứng dụng có một yêu cầu khác nhau. Còn các công đoạn khác thì như nhau.

Dưới đây là một số tính năng nổi bật của Spring Boot:

* Tạo các ứng dụng Spring độc lập
* Nhúng trực tiếp Tomcat, Jetty hoặc Undertow (không cần phải deploy ra file WAR)
* Các starter dependency giúp việc cấu hình Maven đơn giản hơn
* Tự động cấu hình Spring khi cần thiết
* Không sinh code cấu hình và không yêu cầu phải cấu hình bằng XML …
  1. **Spring Boot là gì ?**

Spring là một framework được ra đời để giúp các nhà phát triển có thể xây dựng hệ thống và chạy ứng dụng trên JVM một cách thuận tiện, đơn giản và nhanh chóng. Đây là một mã nguồn mở được phát triển và rất nhiều người sử dụng.

Trên thực tế, Spring framework là tập hợp gồm rất nhiều các dự án nhỏ khác nhau như: Spring MVC (sử dụng để xây dựng các ứng dụng trên nền tảng web), Spring Data, Spring Boot,…

Để phát triển một ứng dụng web cơ bản sử dụng [Spring framework](https://itnavi.com.vn/blog/spring-la-gi-spring-framework-cua-java/), bạn cần trải qua ít nhất 5 công đoạn sau:

* Tạo một project sử dụng Maven với các dependency cần thiết.
* Tạo một tập tin web đuôi (.xml) để khai báo DispatcherServlet (thuộc Spring MVC).
* Một tập tin có cấu hình của Spring MVC.
* Trả về một class Controller khi có request đến.
* Cuối cùng là phải có một web server phục vụ triển khai ứng dụng lên chạy.

Để khởi tạo một dự án Spring khá là vất vả và tốn nhiều công sức. Người lập trình phải khai báo những dependency trong pom.xml, cấu hình XML hoặc các annotation cực kỳ phức tạp. Giờ đây, việc tạo ra các ứng dụng này sẽ được thực hiện một cách đơn giản, nhanh chóng hơn rất nhiều chỉ với Spring Boot.



Từ những thông tin ở trên, ta có thể khẳng định Spring Boot chính là một Java framework siêu to và khổng lồ và có nhiều khả năng hữu ích vì nó có thể giúp lập trình viên giải quyết rất nhiều vấn đề. So với framework Spring thông thường, Spring Boot tỏ ra những lợi thế vượt trội. Khi sử dụng Spring Boot, rất nhiều thứ được cải tiến hỗ trợ lập trình viên như:

* Auto config: tự động cấu hình thay lập trình viên, bạn chỉ cần viết code và tiến hành chạy hệ thống là được.
* Dựa trên các Annotation để tạo lập các bean thay vì XML.
* Server Tomcat có thể được nhúng ngay trong file JAR build ra và có thể chạy ở bất kì đâu mà java chạy được.

Khi sử dụng Spring Boot, lập trình viên chỉ cần:

* Sử dụng Spring Initializr: nhập các thông tin của dự án (project), chọn thư viện (Library) rồi tải code về máy.
* Mở mã nguồn (source code) và bắt đầu viết code.
* Có thể chạy ngay trong IDE, hoặc build thành file JAR mà không cần cấu hình config cho server nữa.
  1. **Đặc điểm nổi bật của SpringBoot**

**SpringApplication**: Đây là class được dùng để khởi chạy ứng dụng từ hàm main(). Vì thế, khi khởi chạy ứng dụng, các lập trình viên chỉ cần gọi method run() là được.

**Externalized Configuration**: Spring Boot cho phép người dùng có thể sử dụng cấu hình (config) từ bên ngoài. Chính vì vậy mà mỗi ứng dụng được tạo ra có thể chạy được trên nhiều môi trường khác nhau.

**Profiles**: Dùng để phân chia các loại cấu hình cho các môi trường khác nhau.

**Loggin**: Sử dụng phục vụ cho toàn bộ chức năng log trong phạm vi nội bộ. Những logging này sẽ được quản lý một cách mặc định.

Ngoài ra còn rất nhiều các tính năng tương tự khác của Spring Boot như: Developing web Applications, Working with SQL Technologies, Security, Messaging, Caching, Calling rest Services with RestTemplate/WebClient, Sending Email, Validation,….

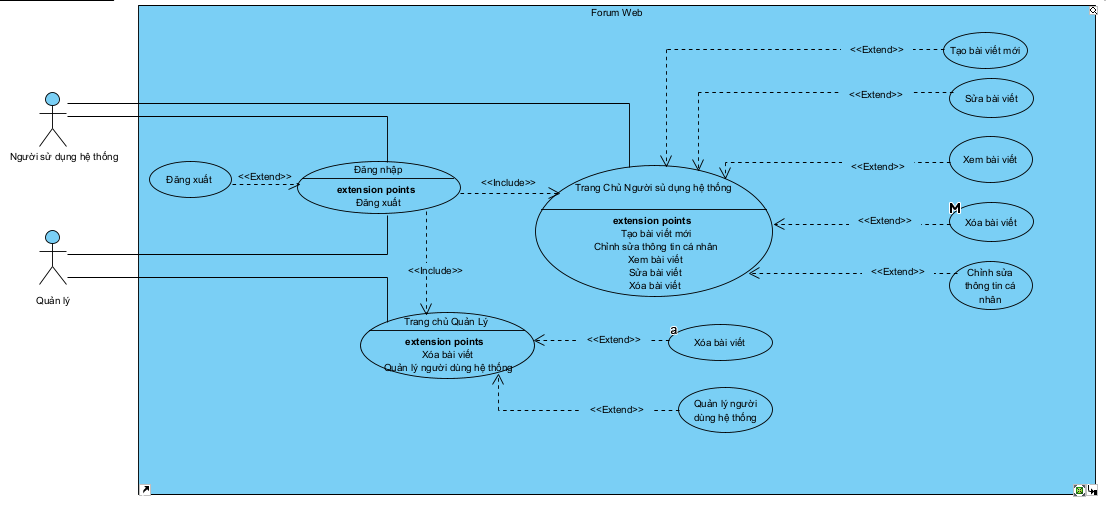
1. **Boostrap**

Cơ chế hoạt động của bootstrap là dựa theo xu hướng [mã nguồn mở](https://mona.media/ma-nguon-mo-la-gi/) HTML,CSS và JavaScript. Người dùng cần trang bị kiến thức cơ bản về 3 loại mã nguồn này mới có thể tiến hành sử dụng Bootstrap hiệu quả. Các mã nguồn này cũng khá đơn giản, có thể dễ dàng thay đổi và tuỳ chỉnh.

Sử dụng công nghệ boostrap và thymeleaf để thực hiện thiết kế giao diện lồng với việc thực hiện code java back-end

Boostrap thực hiện sử dụng file cdn có sẵn không thực hiện tải thư viện về, xài những file css trên mạng và ứng dụng code cho phần cài đặt giao diện

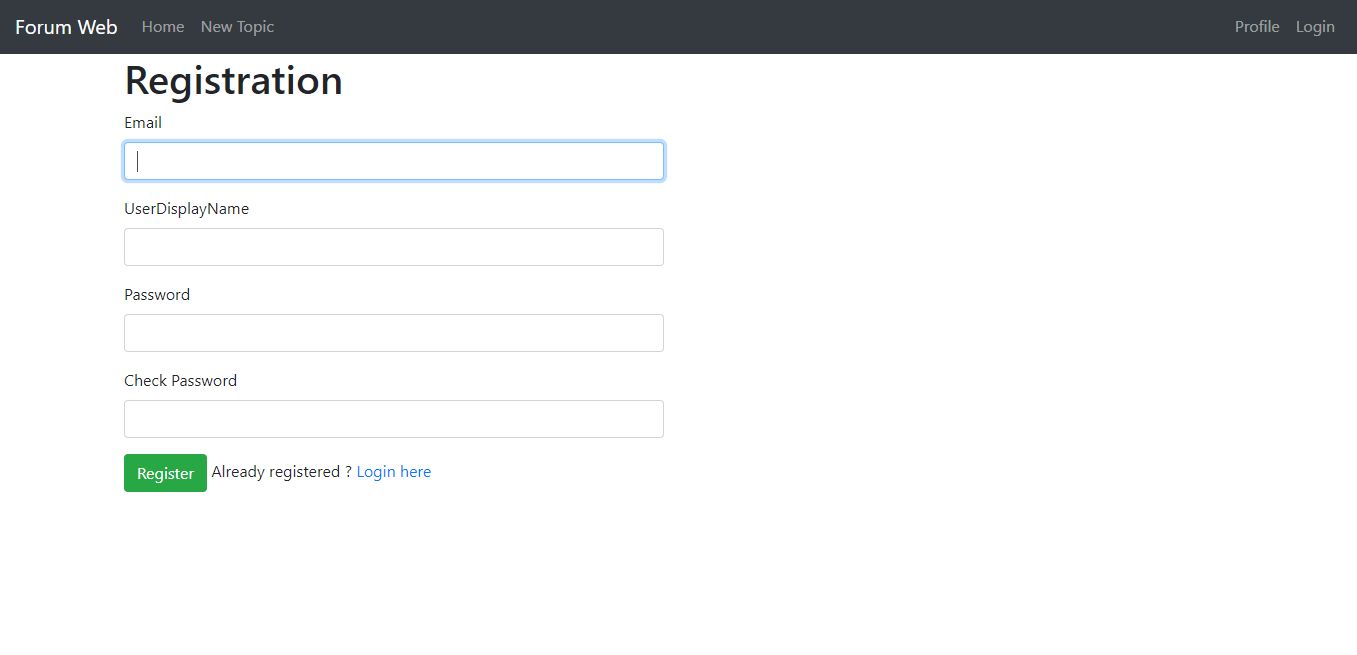
1. **Phân tích thiết kế**
   1. **Sơ đồ tổng quan các UserCase của toàn hệ thống**



* 1. **Đăng ký**

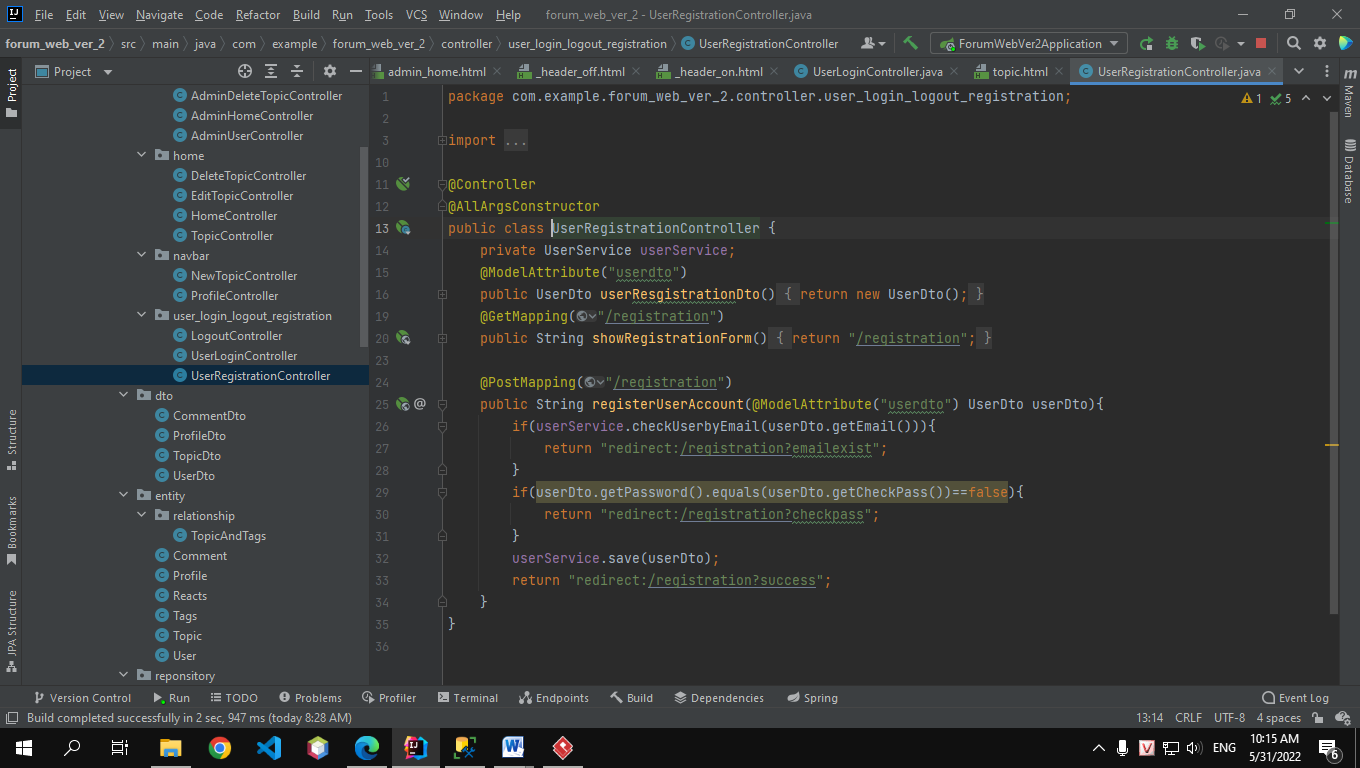
Người dùng đăng ký người dùng hệ thống để nhận được tài khoản đăng nhập hệ thống.

**2.1. Giao diện**

****

Người dùng thực hiện nhập các thông tin của người sử dụng để đăng ký một tài khoản trên hệ thống. Một tài khoản sẽ yêu cầu những thông tin như: Email, UserDisplayName, Password, Check Password.   
Sau khi đã nhập đủ thông tin yêu cầu người dùng hệ thống có thể thực hiện đăng nhập hệ thống.

**2.2. Back-end**

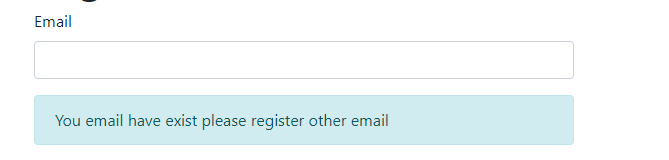


Xử lý phần back-end:

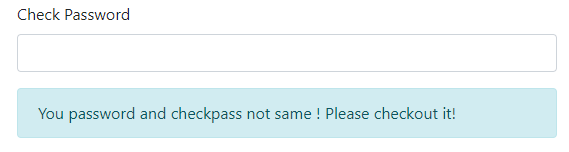
Ở phần get: để thể hiện show form đăng ký

Ở phần post: lọc các thông tin

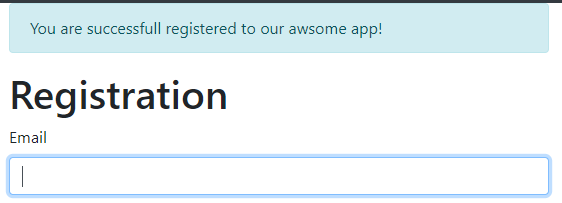
if(userService.checkUserbyEmail(userDto.getEmail())){  
 return "redirect:/registration?emailexist";  
}  
if(userDto.getPassword().equals(userDto.getCheckPass())==false){  
 return "redirect:/registration?checkpass";  
}  
userService.save(userDto);  
return "redirect:/registration?success";

Tại dòng if đầu tiên kiểm tra email này đã được đăng ký hay chưa  


Tại dòng if thứ hai thực hiện kiểm tra password có giống nhau hay không



Thực hiện gọi lớp userService để lưu thông tin và trả về đăng nhập thành công.



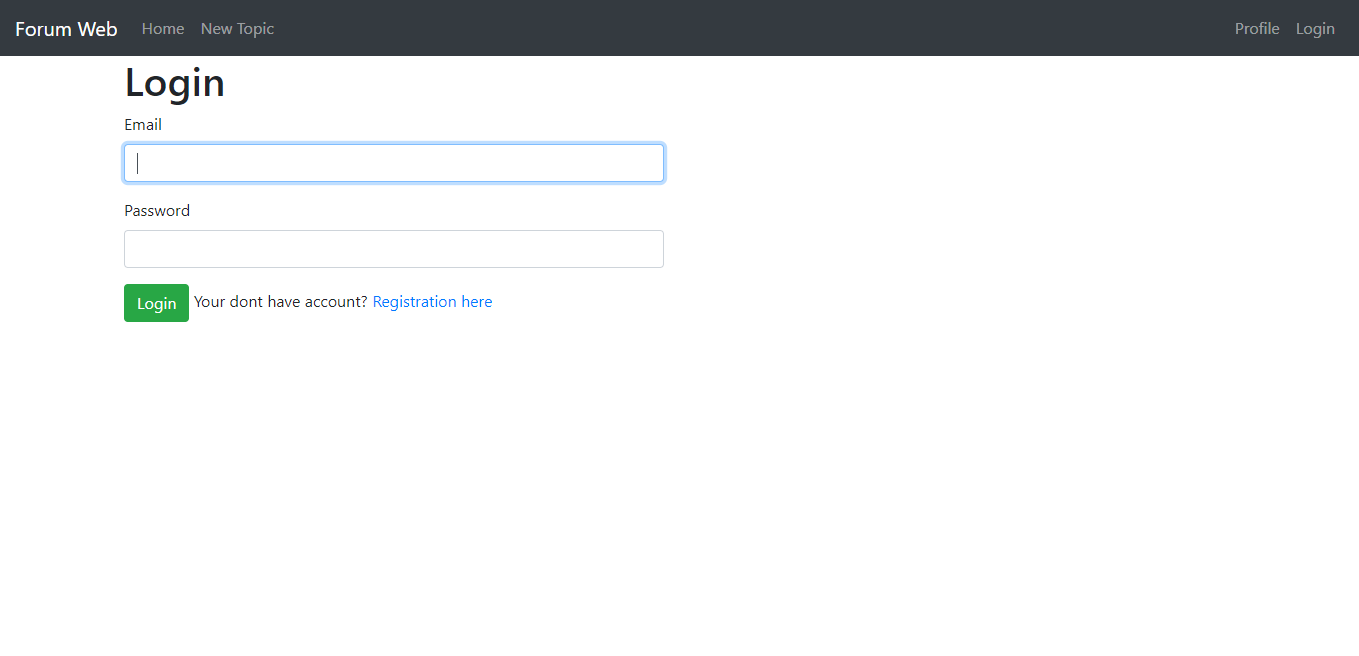
**2.3. Front-end**

<th:block th:include="/\_header\_on" th:if="${userdto.email}"></th:block>  
<th:block th:include="/\_header\_off" th:unless="${userdto.email}"></th:block>  
 <div class="container">  
 <div class="row">  
 <div class="col-md-6 col-md-offset-3">  
 <!-- success message -->  
 <div th:if="${param.success}">  
 <div class="alert alert-info"> You are successfull registered to our awsome app!</div>  
 </div>  
  
 <h1>Registration</h1>  
 <form th:action="@{/registration}" method="post" th:object="${userdto}">  
 <div class="form-group">  
 <label class="control-label" for="Email"> Email </label>  
 <input type="text" id="Email" class="form-control" th:field="\*{email}" required  
 autofocus="autofocus"/>  
 </div>  
 <div th:if="${param.emailexist}">  
 <div class="alert alert-info"> You email have exist please register other email</div>  
 </div>  
  
 <div class="form-group">  
 <label class="control-label" for="UserDisplayName"> UserDisplayName </label>  
 <input type="text" id="UserDisplayName" class="form-control" th:field="\*{userDisplayName}" required/>  
 </div>  
  
 <div class="form-group">  
 <label class="control-label" for="Password"> Password </label>  
 <input type="password" id="Password" class="form-control" th:field="\*{password}" required/>  
 </div>  
  
 <div class="form-group">  
 <label class="control-label" for="checkpass"> Check Password </label>  
 <input type="password" id="checkpass" class="form-control" th:field="\*{checkPass}" required/>  
 </div>  
  
 <div th:if="${param.checkpass}">  
 <div class="alert alert-info"> You password and checkpass not same ! Please checkout it!</div>  
 </div>  
  
 <div class="form-group">  
 <button type="submit" class="btn btn-success">Register</button>  
 <span>Already registered ? <a href="/" th:href="@{/login}">Login here</a></span>  
 </div>  
 </form>  
 </div>  
 </div>  
 </div>

* 1. **Đăng nhập**

Người dùng đăng nhập hệ thống và hệ thống xác định role người dùng và người quản lý trong hệ thống.

**3.1. Giao diện**



Người dùng thực hiện đăng nhập bằng Email và Password. Nếu người dùng có role là User truyền đến trang Home của user. Người dùng có role là Admin truyền đến trang Home của admin.

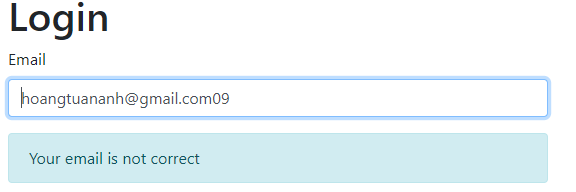
**3.2. Back-end**

@GetMapping("/login")  
public String showLoginForm(){  
 return "/login";  
}  
@PostMapping("/login")  
public String Login(@ModelAttribute("userdto") UserDto userDto, Model model){  
 if(userService.checkUserbyEmail(userDto.getEmail())==false){  
 return "redirect:/login?emailwrong";  
 }  
 User user = userService.getUserbyEmail(userDto.getEmail());  
 if(user.getRole().equals("ADMIN")){  
 return "redirect:/admin\_home";  
 }  
 if(userService.checkPasswordUser(userDto.getEmail(),userDto.getPassword())){  
 return "redirect:/home?success";  
 }  
  
 return "redirect:/login?passwordwrong";  
}

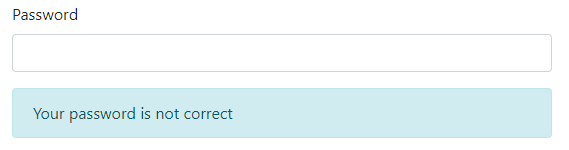
Ở phần Get: Show ra form login để thực hiện điền thông tin

Ở phần Post:

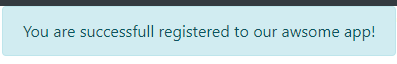
Tại if đầu tiên xử lý dữ liệu xem email có đúng hay không (nghĩa là có tồn tại email này hay không)



Tại if thức 2 xử lý dữ liệu xem mật khẩu của email này có đúng hay không.



Khi đã đúng mật khẩu và tài khoản. Đăng nhập thành công vào hệ thống. Hệ thống sẽ hiển thị là Chúc mừng bạn đã đăng nhập vào hệ thống.



Tại dòng if thứ 3 để thực hiện chuyển luồng nếu người đăng nhập là Admin thì sẽ thực hiện đăng nhập vào một giao diện admin trang chủ còn người là user sẽ thực hiện đăng nhập vào giao diện user trang chủ.

Người dùng sau khi đăng nhập thì tài khoản của người sử dụng được lưu vào trong session để thực hiện thao tác trên trang web.

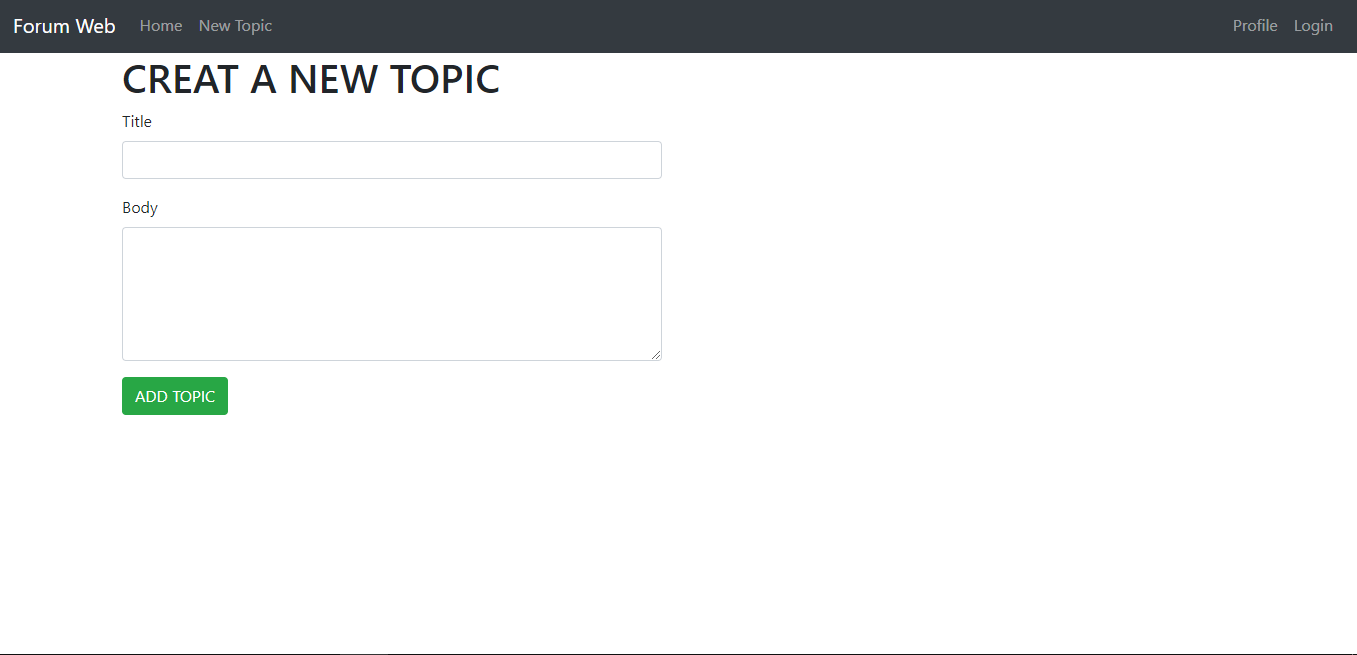
**3.3. Front-end**

<th:block th:include="/\_header\_on" th:if="${userdto.email}"></th:block>  
<th:block th:include="/\_header\_off" th:unless="${userdto.email}"></th:block>  
<div class="container">  
 <div class="row">  
 <div class="col-md-6 col-md-offset-3">  
  
  
 <h1>Login</h1>  
 <form th:action="@{/login}" method="post" th:object="${userdto}">  
 <div class="form-group">  
 <label class="control-label" for="Email"> Email </label>  
 <input type="text" id="Email" class="form-control" th:field="\*{email}" required  
 autofocus="autofocus"/>  
 </div>  
 <div th:if="${param.emailwrong}">  
 <div class="alert alert-info"> Your email is not correct</div>  
 </div>  
  
 <div class="form-group">  
 <label class="control-label" for="Password"> Password </label>  
 <input type="password" id="Password" class="form-control" th:field="\*{password}" required/>  
 </div>  
 <div th:if="${param.passwordwrong}">  
 <div class="alert alert-info"> Your password is not correct</div>  
 </div>  
  
 <div class="form-group">  
 <button type="submit" class="btn btn-success">Login</button>  
 <span>Your dont have account? <a href="/" th:href="@{/registration}">Registration here</a></span>  
 </div>  
 </form>  
 </div>  
 </div>  
</div>

* 1. **Đăng bài viết**

Người dùng thực hiện đăng bài viết thông qua form New Topic. Người dùng có quyền đăng bài viết mới về hỏi đáp thông qua việc phải có tài khoản và đăng nhập thành công.

### 4.1. Giao diện

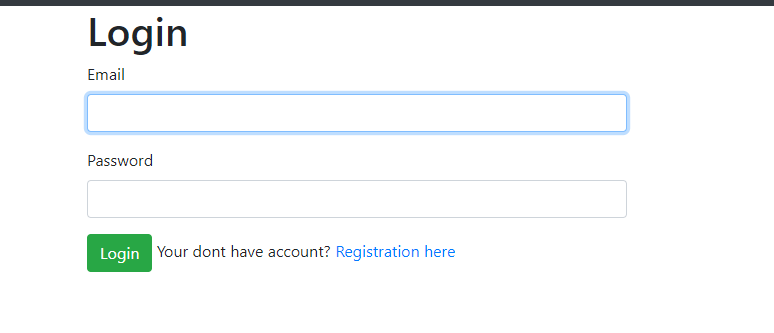


### 4.2. Back-end

@GetMapping("/newtopic")  
public String showNewTopic(Model model){  
 List<Tags> tags = tagsService.getListTag();  
 model.addAttribute("tags",tags);  
 return "/newtopic";  
}  
@PostMapping("/newtopic")  
public String newTopic(@ModelAttribute("userdto") UserDto userDto,@ModelAttribute("topic") TopicDto topicDto,Model model){  
 User user = userReponsitory.getUserByEmail(userDto.getEmail());  
 if(user== null){  
 return "redirect:/login";  
 }  
 topicService.save(topicDto,user);  
 return "redirect:/home";  
}

Phần Get: Thực hiện lấy ra các thuộc tính cần thiết

Phần Post: Lấy thông tin người dùng trong session và thực hiện kiểm tra xem người dùng đã đăng nhập hay chưa. Nếu người dùng vẫn chưa thực hiện đăng nhập. Yêu cầu người dùng đăng nhập.

  
 Nếu người dùng đã thực hiện đăng nhập thành công. Thực hiện lấy các thông tin người dùng nhập vào trong form lưu vào trong database. Và thực hiện chuyển hướng trang đến trang home của user.



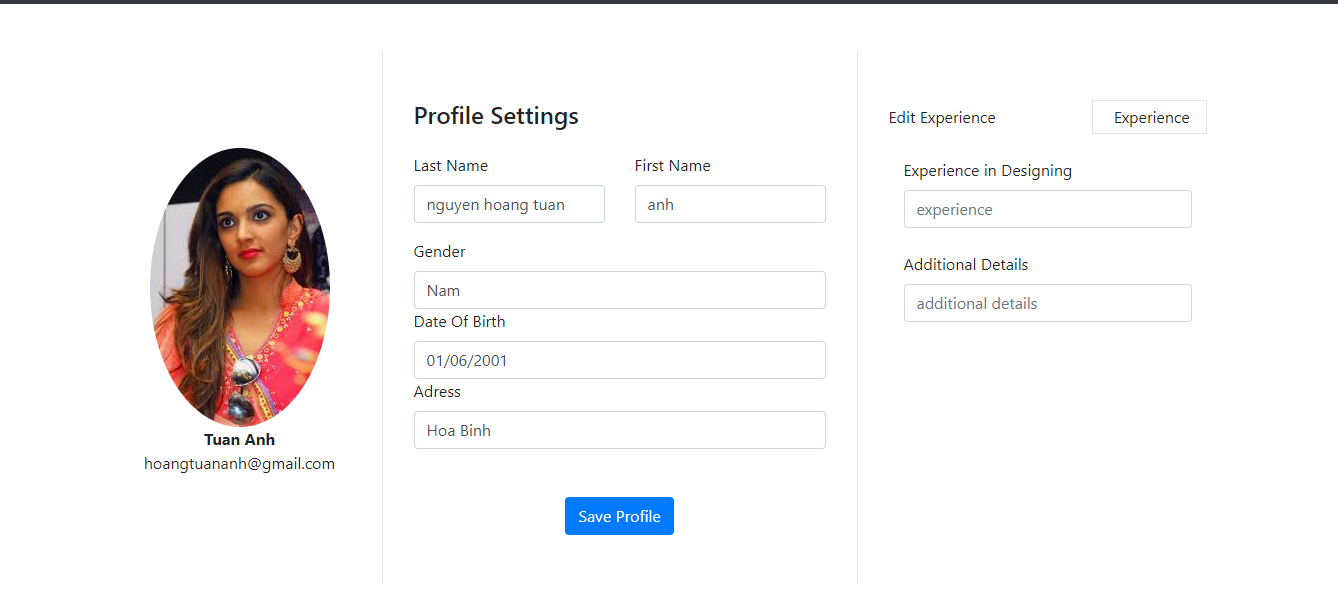
### 4.3. Front-end

<body>  
<th:block th:include="/\_header\_on" th:if="${userdto.email}"></th:block>  
<th:block th:include="/\_header\_off" th:unless="${userdto.email}"></th:block>  
 <div class="container">  
 <div class="row">  
 <div class="col-md-6 col-md-offset-3">  
  
  
 <h1>CREAT A NEW TOPIC</h1>  
 <form th:action="@{/newtopic}" method="post">  
 <div class="form-group">  
 <label class="control-label" for="Title"> Title </label>  
 <input type="text" id="Title" class="form-control" th:field="${topic.title}" required  
 autofocus="autofocus"/>  
 </div>  
  
  
<!-- <div class="form-check">-->  
<!-- <input type="checkbox" class="form-check-input" id="exampleCheck1">-->  
<!-- <label class="form-check-label" for="exampleCheck1">Check me out</label>-->  
<!-- </div>-->  
  
 <div class="form-group">  
 <label class="control-label" for="Body"> Body </label>  
 <textarea id="Body" class="form-control" th:field="\*{topic.body}" rows="5" required/>  
 </div>  
  
 <div class="form-group">  
 <button type="submit" class="btn btn-success">ADD TOPIC</button>  
 </div>  
 </form>  
 </div>  
 </div>  
 </div>  
</body>

* 1. **Thông tin cá nhân**

Người dùng chọn vào nút thông tin cá nhân, hệ thống hiển thị các thông tin của người dùng đã đăng ký. Người dùng có quyền chỉnh sửa thông tin cá nhân của chính bản thân

* 1. **Giao diện**



* 1. **Back-end**

@GetMapping("/profile")  
public String showprofile(@ModelAttribute("userdto")UserDto userDto,  
 Model model){  
 User user = userService.getUserbyEmail(userDto.getEmail());  
 if(user ==null){  
 return "redirect:/login";  
 }  
 model.addAttribute("user",user);  
 Profile profile = profileReponsitory.findProfileByUser(user);  
 model.addAttribute("profile",profile);  
 return "profile";  
}  
  
@PostMapping("/profile/{id}")  
public String getprofile(@SessionAttribute("userdto")UserDto userDto,  
 @PathVariable String id ,  
 Model model,  
 @ModelAttribute("profile") ProfileDto profileDto){  
 Profile profileupdate = profileReponsitory.findProfileById(Integer.*parseInt*(id));  
 profileupdate.setAdress(profileDto.getAdress());  
 profileupdate.setDateofBirth(profileDto.getDateofBirth());  
 profileupdate.setGender(profileDto.getGender());  
 profileupdate.setFirstName(profileDto.getFirstName());  
 profileupdate.setLastName(profileDto.getLastName());  
 profileService.update(profileupdate);  
 return "redirect:/profile";  
}

Phần Get: lấy thông tin của user trong session thực hiện kiểm tra nếu không có user đăng nhập thành công thực hiện chuyển luồng sang form đăng nhập

Nếu đã đăng nhập thành công thực hiện chuyển luồng tìm kiểm thông tin của người đó và thực hiện hiển thị các thông tin của người đó trong phần form Thông tin cá nhân

Phần Post: Được thực hiện khi người dùng nhập mới thông tin và ấn vào nút save. Thông tin sẽ được cập nhật và lưu vào trong database.

Thực hiện trả về trang profile.

### 5.3. Front-end

<body>  
<th:block th:include="/\_header\_on" th:if="${userdto.email}"></th:block>  
<th:block th:include="/\_header\_off" th:unless="${userdto.email}"></th:block>  
 <div class="container rounded bg-white mt-5 mb-5">  
 <div class="row">  
 <div class="col-md-3 border-right">  
<!-- Hiển thị tên người dùng và email-->  
 <div class="d-flex flex-column align-items-center text-center p-3 py-5">  
 <img class="rounded-circle mt-5" src="https://encrypted-tbn0.gstatic.com/images?q=tbn%3AANd9GcQF2psCzfbB611rnUhxgMi-lc2oB78ykqDGYb4v83xQ1pAbhPiB&usqp=CAU">  
 <span class="font-weight-bold" th:text="\*{user.userDisplayName}">Name User</span>  
 <span class="text-black-50" th:text="\*{user.email}">amelly12@bbb.com</span>  
 <span> </span>  
 </div>  
 </div>  
 <div class="col-md-5 border-right">  
 <div class="p-3 py-5">  
 <form th:action="@{'/profile/'+ ${profile.id}}" method="post" th:object="${profile}">  
 <div class="d-flex justify-content-between align-items-center mb-3">  
 <h4 class="text-right">Profile Settings</h4>  
 </div>  
 <div class="row mt-2">  
 <!-- Hiển thị tên thật người dùng-->  
 <div class="col-md-6">  
 <label class="labels">Last Name</label>  
 <input type="text" id="Last Name" class="form-control" th:field="\*{lastName}" required  
 autofocus="autofocus"/>  
 </div>  
 <div class="col-md-6">  
 <label class="labels">First Name</label>  
 <input type="text" class="form-control" value="" placeholder="FirstName" th:field="\*{firstName}" name="firstName">  
 </div>  
 </div>  
 <div class="row mt-3">  
 <div class="col-md-12">  
 <label class="labels">Gender</label>  
 <input type="text" class="form-control" placeholder="Your Gender" th:field="\*{gender}">  
 </div>  
 <div class="col-md-12">  
 <label class="labels">Date Of Birth</label>  
 <input type="text" class="form-control" placeholder="Your Date Of Birth" th:field="\*{dateofBirth}">  
 </div>  
 <div class="col-md-12">  
 <label class="labels">Adress</label>  
 <input type="text" class="form-control" placeholder="Your Adress" th:field="\*{adress}">  
 </div>  
 </div>  
<!-- <div class="row mt-3">-->  
<!-- <div class="col-md-6"><label class="labels">Country</label><input type="text" class="form-control" placeholder="country" value=""></div>-->  
<!-- <div class="col-md-6"><label class="labels">State/Region</label><input type="text" class="form-control" value="" placeholder="state"></div>-->  
<!-- </div>-->  
 <div class="mt-5 text-center">  
 <button class="btn btn-primary profile-button" type="submit">  
 Save Profile  
 </button>  
 </div>  
 </form>  
 </div>  
 </div>  
 <div class="col-md-4">  
 <div class="p-3 py-5">  
 <div class="d-flex justify-content-between align-items-center experience"><span>Edit Experience</span><span class="border px-3 p-1 add-experience"><i class="fa fa-plus"></i>&nbsp;Experience</span></div><br>  
 <div class="col-md-12"><label class="labels">Experience in Designing</label><input type="text" class="form-control" placeholder="experience" value=""></div> <br>  
 <div class="col-md-12"><label class="labels">Additional Details</label><input type="text" class="form-control" placeholder="additional details" value=""></div>  
 </div>  
 </div>  
 </div>  
 </div>  
 </div>  
 </div>  
</body>

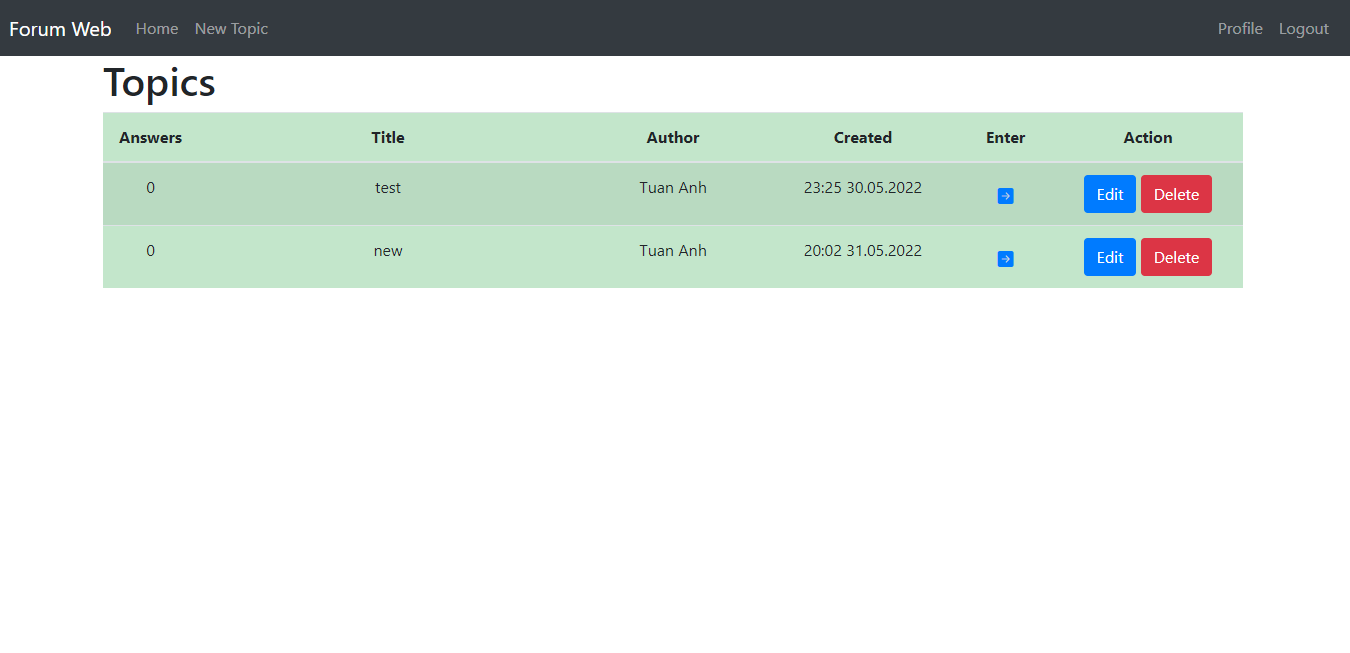
* 1. **Trang chủ**

Người dùng sau khi thực hiện đăng nhập hệ thống vào được trang chủ của hệ thống.

Ở trang chủ nếu người dùng là người đăng bài có khả năng sửa được topic đã tạo, hoặc xóa topic đã tạo.

Người không đăng nhập hệ thống và người đăng nhập hệ thống đều có thể vào xem được topic của người đăng nhập hệ thống đăng bài

* 1. **. Giao diện**



* 1. **Back-end**

@GetMapping("/home")  
public String showHomeForm(@ModelAttribute("userdto") UserDto userDto,Model model){  
 List<Topic> topics = topicReponsitory.findAll();  
 model.addAttribute("topics",topics);  
 model.addAttribute("commentcount",commentService);  
  
 return "/home";  
}

Phần hiển thị thông tin lấy ra những thông tin về các bài viết và số lượng bình luận trong bài viét đó. Và trả về trang home để thực hiện hiển thị

* 1. **Front-end**

<body>  
<th:block th:include="/\_header\_on" th:if="${userdto.email}"></th:block>  
<th:block th:include="/\_header\_off" th:unless="${userdto.email}"></th:block>  
<div class="container">  
 <div class="row">  
  
 <!-- success message -->  
 <div th:if="${param.success}">  
 <div class="alert alert-info"> You are successfull registered to our awsome app!</div>  
 </div>  
  
 <h1>Topics</h1>  
 <table class="table table-striped table-success text-center" th:if="${topics.size() != 0}">  
 <thead>  
 <tr>  
 <th class="col-md-1">Answers</th>  
 <th class="col-md-4">Title</th>  
 <th class="col-md-2">Author</th>  
 <th class="col-md-2">Created</th>  
 <th class="col-md-1">Enter</th>  
 <th class="col-md-2">Action</th>  
 </tr>  
 </thead>  
 <tbody>  
 <th:block th:each="topic : ${topics}">  
 <tr>  
 <td th:text="${commentcount.countComment(topic.id)}">Number of answers</td>  
 <td th:text="\*{topic.title}">Topic's title</td>  
 <td>  
<!-- <a th:href="@{'/profile/' + ${topic.getUser().getId()}}">-->  
 <div class="chip" th:text="${topic.user.userDisplayName}">Topic's author</div>  
<!-- </a>-->  
 </td>  
 <td th:text="\*{topic.creationDate}">Topic's created date</td>  
 <td>  
 <a th:href="@{'/topic/' + ${topic.id}}" class="btn btn-floating pulse">  
 <i class="bi bi-arrow-right-square-fill"></i>  
 </a>  
 </td>  
 <td th:if="${userdto.email == topic.user.email}">  
 <button class="btn btn-primary text-white" type="button">  
 <a th:href="@{'/edit\_topic/'+${topic.id}}" class="text-white">  
 Edit  
 </a>  
 </button>  
 <button class="btn btn-danger" type="button">  
 <a th:href="@{'/deletetopic/'+${topic.id}}" class="text-white" method=RequestMethod.GET>  
 Delete  
 </a>  
 </button>  
 </td>  
 </tr>  
 </th:block>  
 </tbody>  
 </table>  
  
 </div>  
</div>  
</body>

1. **Xóa Bài Viết**

Người dùng đăng nhập hệ thống đúng với tài khoản đăng bài có quyền thực hiện xóa bài viết

**7.1. Back-end**

@GetMapping("/deletetopic/{id}")  
public String deleteTopic(@PathVariable String id){  
 reactService.delete(Integer.*parseInt*(id));  
 commentService.delete(Integer.*parseInt*(id));  
 topicService.delete(Integer.*parseInt*(id));  
 return "redirect:/home";  
}

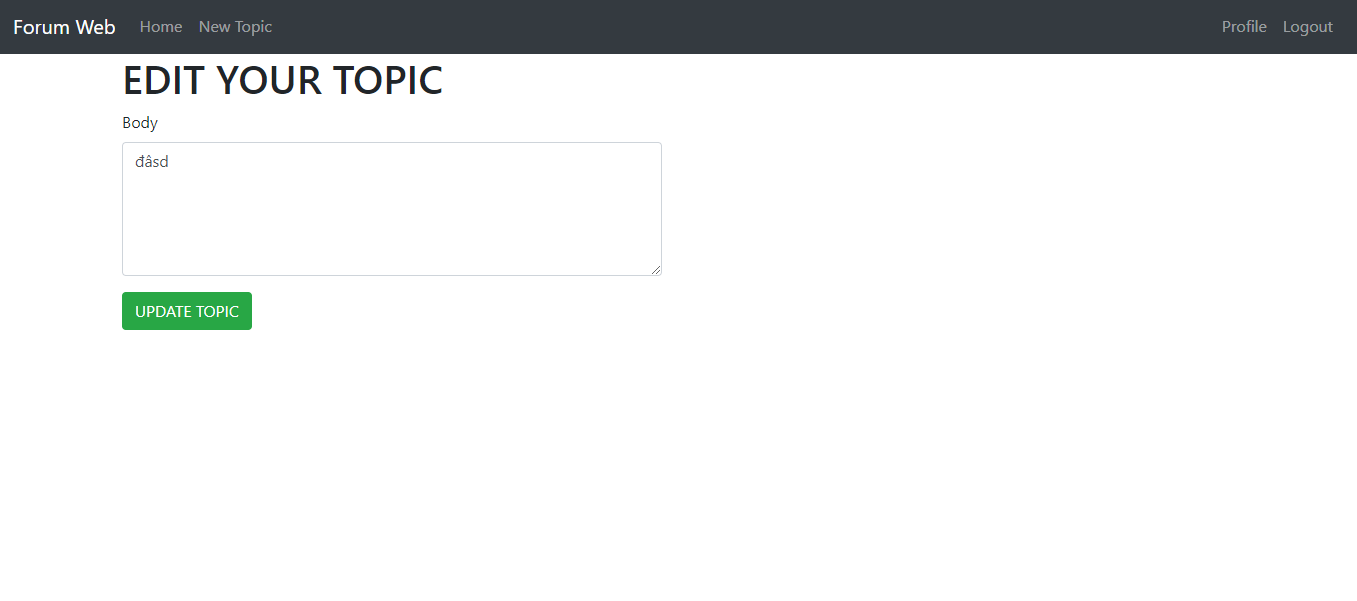
Thực hiện xóa phần tương tác trước, rồi thực hiển xóa phần bình luận, và cuối cùng xóa phần bài viết.

Kết thúc thực hiển trả về trang chủ của người dùng hệ thống

1. **Sửa bài viết**

Người dùng hệ thống, đăng nhập hệ thống đúng với tài khoản thực hiện đăng bài có quyền thực hiện sửa bài viết đã đăng.

**8.1. Phần giao diện**



* 1. **Back-end**

@GetMapping("/edit\_topic/{id}")  
public String showEditTopic(Model model, @PathVariable String id,@ModelAttribute("userdto") UserDto userDto){  
 Topic topic =topicService.findTopicById(Integer.*parseInt*(id));  
 model.addAttribute("topic",topic);  
 return "edit\_topic";  
}  
@PostMapping("/edit\_topic/{id}")  
public String updateTopic(Model model, @PathVariable String id,  
 @ModelAttribute("userdto") UserDto userDto,  
 @ModelAttribute("topic") TopicDto topicDto){  
 Topic updatetopic= topicService.findTopicById(Integer.parseInt(id));  
 updatetopic.setBody(topicDto.getBody());  
 topicService.update(updatetopic);  
 return "redirect:/home";  
}

Phần Get: Thực hiện lấy các thông tin của topic cần chỉnh sửa ra, và lấy session trong đó có người dùng đã đăng nhập hệ thống được lưu thông tin vào trong session của hệ thống

Phần Post: Thực hiện lấy những thông tin đã thay đổi ở trong phần chỉnh sửa bài viết và cập nhật vào Database

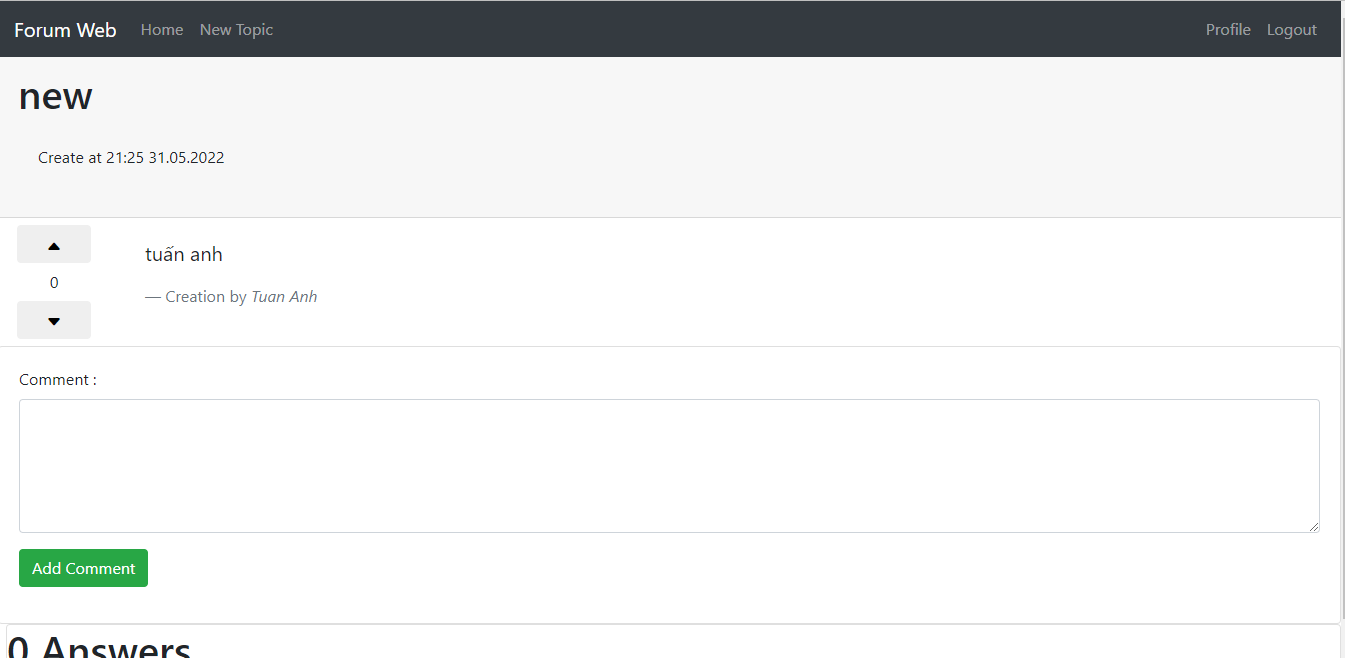
* 1. **Front-end**

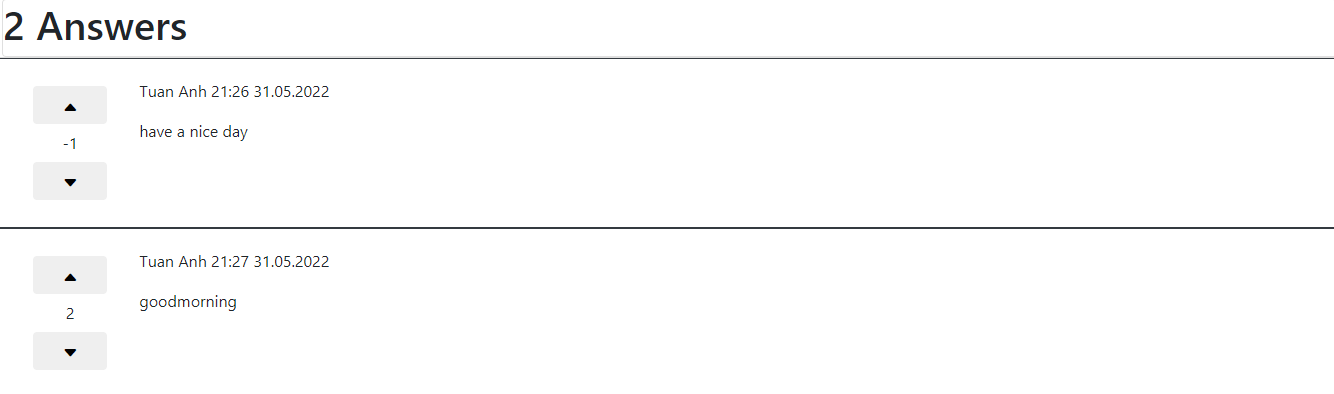
<body>  
<th:block th:include="/\_header\_on" th:if="${userdto.email}"></th:block>  
<th:block th:include="/\_header\_off" th:unless="${userdto.email}"></th:block>  
<div class="container">  
 <div class="row">  
 <div class="col-md-6 col-md-offset-3">  
  
 <h1>EDIT YOUR TOPIC</h1>  
 <form th:action="@{'/edit\_topic/' + ${topic.id}}" method="post" th:object="${topic}">  
 <div class="form-group">  
 <label class="control-label" for="Body"> Body </label>  
 <textarea id="Body" class="form-control" th:field="\*{body}" rows="5" placeholder="Edit your content topic"/>  
 </div>  
  
 <div class="form-group">  
 <button type="submit" class="btn btn-success">UPDATE TOPIC</button>  
 </div>  
 </form>  
 </div>  
 </div>  
</div>  
</body>

1. **Bài viết**

Người sử dụng không thực hiện đăng nhập hệ thống có khả năng xem bài viết. Nhưng không có khả năng thực hiện tương tác với bài viết hoặc bình luận

### 9.1. Giao diện





* 1. **Back-end**

@GetMapping("/topic/{id}")  
public String showTopicById(Model model, @PathVariable String id) {  
 Topic topic = topicService.findTopicById(Integer.*parseInt*(id));  
 List<Comment> comments = commentReponsitory.getAllByTopic\_Id(topic.getId());  
 int countTopicReact = reactService.countReact(topic);  
 Map<Comment, Integer> reactsListComment = new HashMap<>();  
 for(Comment comment: comments){  
 int countCommentReact = reactService.countReactComment(comment);  
 reactsListComment.put(comment,countCommentReact);  
 }  
 model.addAttribute("topic", topic);  
 model.addAttribute("comments", comments);  
 model.addAttribute("commentcount",commentService);  
 model.addAttribute("counttopicreact",countTopicReact);  
 model.addAttribute("reactsListComment",reactsListComment);  
 return "topic";  
}  
@PostMapping("/topic/{id}")  
public String postComment(@ModelAttribute("userdto") UserDto userDto,  
 @PathVariable String id,  
 @ModelAttribute("newcomment") CommentDto commentDto  
) {  
  
 Topic topic = topicService.findTopicById(Integer.*parseInt*(id));  
 User user = userReponsitory.getUserByEmail(userDto.getEmail());  
 if(user == null){  
 return "redirect:/login";  
 }  
 commentService.save(commentDto, user, topic);  
 return "redirect:/topic/"+id;  
}  
@PostMapping("/topic/{id}/down")  
public String downReact(@ModelAttribute("userdto") UserDto userDto,  
 @PathVariable String id){  
 Topic topic = topicService.findTopicById(Integer.*parseInt*(id));  
 User user = userReponsitory.getUserByEmail(userDto.getEmail());  
 if(user == null){  
 return "redirect:/login";  
 }  
 reactService.saveReactTopic("down",topic,user);  
 return "redirect:/topic/"+id;  
}  
  
@PostMapping("/topic/{id}/up")  
public String upReact(@ModelAttribute("userdto") UserDto userDto,  
 @PathVariable String id){  
 Topic topic = topicService.findTopicById(Integer.*parseInt*(id));  
 User user = userReponsitory.getUserByEmail(userDto.getEmail());  
 if(user == null){  
 return "redirect:/login";  
 }  
 reactService.saveReactTopic("up",topic,user);  
 return "redirect:/topic/"+id;  
}  
  
@PostMapping("/comment/{id}/up")  
public String upcmtReact(@ModelAttribute("userdto") UserDto userDto,  
 @PathVariable String id){  
 Comment comment = commentService.findCommentbyId(Integer.*parseInt*(id));  
 User user = userReponsitory.getUserByEmail(userDto.getEmail());  
 if(user == null){  
 return "redirect:/login";  
 }  
 reactService.saveReactComment("up",comment,user);  
 return "redirect:/topic/"+comment.getTopic().getId();  
}  
  
@PostMapping("/comment/{id}/down")  
public String downcmtReact(@ModelAttribute("userdto") UserDto userDto,  
 @PathVariable String id){  
 Comment comment = commentService.findCommentbyId(Integer.*parseInt*(id));  
 User user = userReponsitory.getUserByEmail(userDto.getEmail());  
 if(user == null){  
 return "redirect:/login";  
 }  
 reactService.saveReactComment("down",comment,user);  
 return "redirect:/topic/"+comment.getTopic().getId();  
}

Phần get của bài viết : Thực hiện nhận các thông tin của bài viết, bình luận, số lượng bình luận, cảm xúc, số lượng cảm xúc.

Phần post của bài viết: Thực hiện lưu các thông tin của phần câu trả lời đối với bài viết, những câu trả lời mới sẽ được thêm vào phía sau những câu trả lời cũ.

Phần thả đánh giá của người dùng đối với bài viết và bình luận là như nhau. Phần này thực hiện khi người dùng đã đăng nhập nếu người dùng thích bài viết hoặc bình luận người dùng thả upvote cho bình luận hoặc bài viết đó, Và ngược lại.

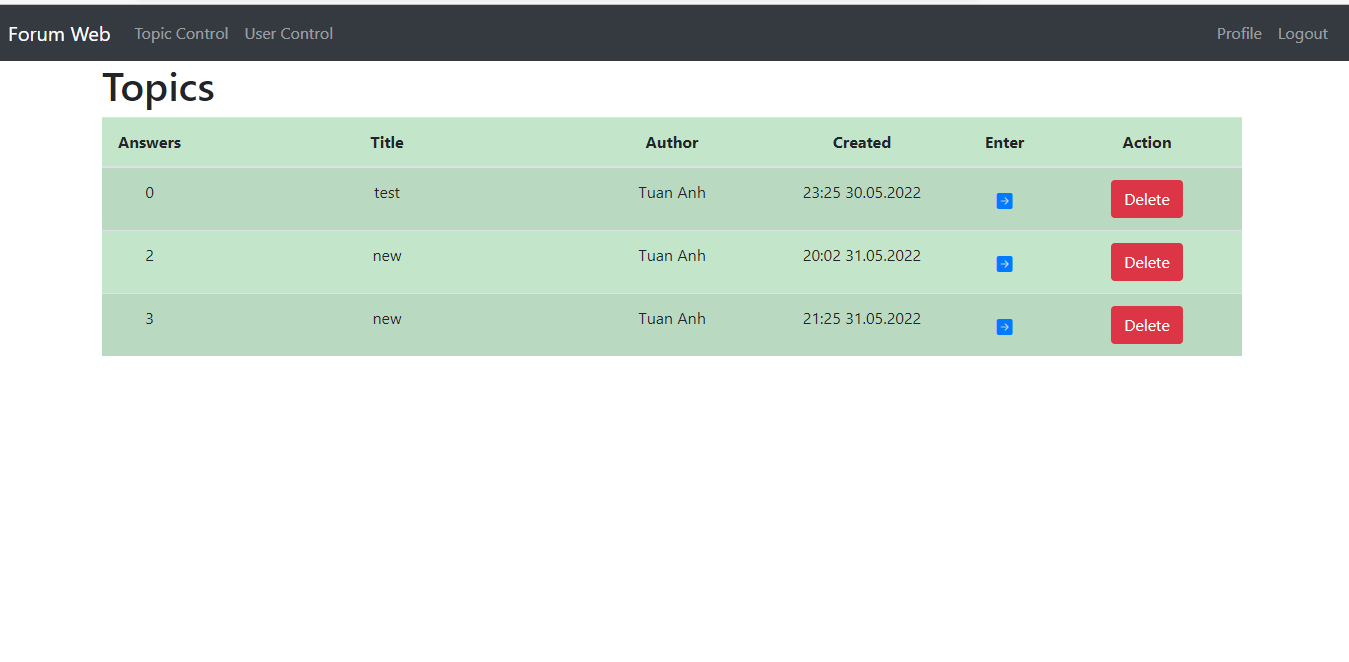
* 1. **Front-End**

<body>  
<th:block th:include="/\_header\_on" th:if="${userdto.email}"></th:block>  
<th:block th:include="/\_header\_off" th:unless="${userdto.email}"></th:block>  
 <div class="card-header">  
 <h1 th:text="\*{topic.title}">Topic's content</h1>  
 <div class="card-body">  
 <p th:text="'Create at ' + \*{topic.creationDate}" class="justify-content-end">Time</p>  
 </div>  
 </div>  
  
 <div class="d-flex">  
 <div class="col-md-1 d-flex flex-column justify-content-center justify-content-around align-items-center">  
 <form class="btn" th:action="@{'/topic/' + ${topic.id} + '/up'}" method="post">  
 <button class="btn">  
 <i class="bi bi-caret-up-fill m-3" > </i>  
 </button>  
 </form>  
 <div>  
 <output id="output" type="number" th:text="${counttopicreact}">0</output>  
 </div>  
 <form class="btn" th:action="@{'/topic/' + ${topic.id} +'/down'}" method="post">  
 <button class="btn" >  
 <i class="bi bi-caret-down-fill m-3" > </i>  
 </button>  
 </form>  
 </div>  
 <div class="col-md-11">  
  
 <div class="card-body">  
 <blockquote class="blockquote mb-0">  
 <p th:text="\*{topic.body}">Question</p>  
 <footer class="blockquote-footer">Creation by <cite th:text="${topic.user.userDisplayName}">Author</cite></footer>  
 </blockquote>  
 </div>  
 </div>  
  
 </div>  
  
 <div class="card">  
 <form th:action="@{'/topic/' + ${topic.id}}" method="post" >  
 <div class="card-body">  
 <div class="form-group">  
 <label class="control-label" for="Body"> Comment : </label>  
 <textarea id="Body" class="form-control" th:field="${newcomment.body}" rows="5" required/>  
 </div>  
 <div class="form-group">  
 <button type="submit" class="btn btn-success">Add Comment</button>  
 </div>  
 </div>  
 </form>  
 </div>  
 <div class="card ml-2">  
 <h1 th:text="${commentcount.countComment(topic.id)}+' Answers'">Answers</h1>  
 </div>  
 <div class ="card" th:if="${comments.size() != 0}">  
 <th:block th:each="comment : ${comments}">  
 <div class="card-body d-flex border border-dark">  
 <div class="col-md-1 d-flex flex-column justify-content-center justify-content-around align-items-center">  
 <form class="btn" th:action="@{'/comment/' + ${comment.id} + '/up'}" method="post">  
 <button class="btn">  
 <i class="bi bi-caret-up-fill m-3" > </i>  
 </button>  
 </form>  
 <div>  
 <output type="number" th:text="${reactsListComment.get(comment)}">0</output>  
 </div>  
 <form class="btn" th:action="@{'/comment/' + ${comment.id} +'/down'}" method="post">  
 <button class="btn">  
 <i class="bi bi-caret-down-fill m-3" > </i>  
 </button>  
 </form>  
 </div>  
 <div class="col-md-11">  
 <p th:text="${comment.user.userDisplayName}+' ' +${comment.creationDate} ">UserDisplayName and Time</p>  
 <p th:text="${comment.body}">Noi Dung</p>  
 </div>  
  
 </div>  
 </th:block>  
 </div>  
  
  
  
</body>

1. **Trang chủ của người quản lý**

Người quản lý sẽ có role là “ADMIN” trong hệ thống. Lúc đăng nhập sẽ được kiểm tra xem có phải là người quản lý hay không. Nếu là người quản lý thực hiện vào một giao diện riêng.

**10.1. Giao diện**



* 1. **Back-end**

@GetMapping("/admin\_home")  
public String showHomeForm(@ModelAttribute("userdto") UserDto userDto, Model model){  
 List<Topic> topics = topicReponsitory.findAll();  
 model.addAttribute("topics",topics);  
 model.addAttribute("commentcount",commentService);  
  
 return "/admin\_home";  
}

Phần Get: lấy các thông tin của bài đăng và bình luận

Người quản lý có khả năng xóa tất cả các bài viết

* 1. **Front-end**

<body>  
<th:block th:include="/\_header\_admin" th:if="${userdto.email}"></th:block>  
<th:block th:include="/\_header\_off" th:unless="${userdto.email}"></th:block>  
<div class="container">  
 <div class="row">  
  
 <!-- success message -->  
 <div th:if="${param.success}">  
 <div class="alert alert-info"> You are successfull registered to our awsome app!</div>  
 </div>  
  
 <h1>Topics</h1>  
 <table class="table table-striped table-success text-center" th:if="${topics.size() != 0}">  
 <thead>  
 <tr>  
 <th class="col-md-1">Answers</th>  
 <th class="col-md-4">Title</th>  
 <th class="col-md-2">Author</th>  
 <th class="col-md-2">Created</th>  
 <th class="col-md-1">Enter</th>  
 <th class="col-md-2">Action</th>  
 </tr>  
 </thead>  
 <tbody>  
 <th:block th:each="topic : ${topics}">  
 <tr>  
 <td th:text="${commentcount.countComment(topic.id)}">Number of answers</td>  
 <td th:text="\*{topic.title}">Topic's title</td>  
 <td>  
 <!-- <a th:href="@{'/profile/' + ${topic.getUser().getId()}}">-->  
 <div class="chip" th:text="${topic.user.userDisplayName}">Topic's author</div>  
 <!-- </a>-->  
 </td>  
 <td th:text="\*{topic.creationDate}">Topic's created date</td>  
 <td>  
 <a th:href="@{'/topic/' + ${topic.id}}" class="btn btn-floating pulse">  
 <i class="bi bi-arrow-right-square-fill"></i>  
 </a>  
 </td>  
 <td> <!--th:if="${userdto.email == topic.user.email}"-->  
 <button class="btn btn-danger" type="button">  
 <a th:href="@{'admin\_home/deletetopic/'+${topic.id}}" class="text-white" method=RequestMethod.GET>  
 Delete  
 </a>  
 </button>  
 </td>  
 </tr>  
 </th:block>  
 </tbody>  
 </table>  
  
 </div>  
</div>  
</body>

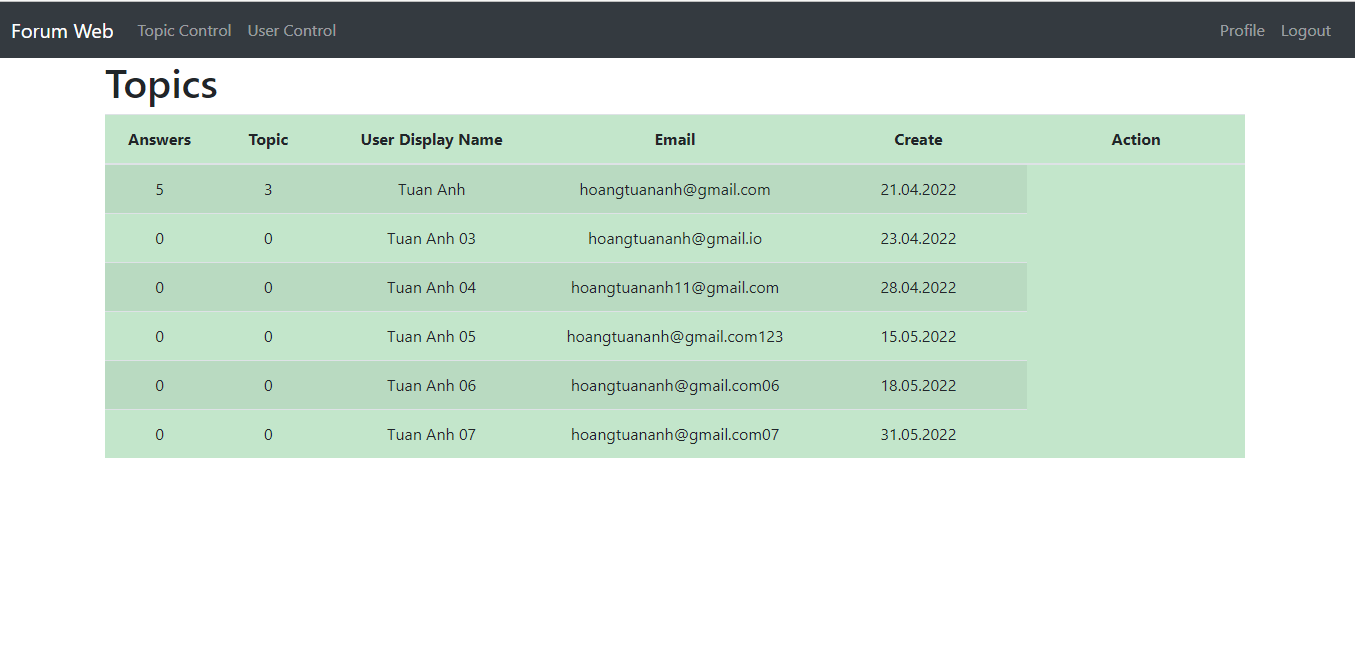
1. **Xóa Topic của Người quản lý**

Phần này thực hiện giống của người dùng sử dụng hệ thống nên không có trình bày thêm.

1. **Quản lý người dùng hệ thống**

Người quản lý có thể đọc được các thông tin của người dùng hệ thống

**12.1. Giao diện**



* 1. **Back-end**

@GetMapping("/table\_user")  
public String showUserControl(@ModelAttribute("userdto") UserDto userDto, Model model){  
 List<User> users = userReponsitory.getAllUser();  
 model.addAttribute("users",users);  
 model.addAttribute("commentcount",commentReponsitory);  
 model.addAttribute("topiccount",topicReponsitory);  
 return "table\_user";  
}

Gọi ra các thông tin của người dùng hệ thống và các hành động của họ. Thực hiện thể hiện các thông tin đó ở trên trang table\_user.

* 1. **Front-end**

<body>  
<th:block th:include="/\_header\_admin" th:if="${userdto.email}"></th:block>  
<th:block th:include="/\_header\_off" th:unless="${userdto.email}"></th:block>  
<div class="container">  
 <div class="row">  
  
 <!-- success message -->  
 <div th:if="${param.success}">  
 <div class="alert alert-info"> You are successfull registered to our awsome app!</div>  
 </div>  
  
 <h1>Topics</h1>  
 <table class="table table-striped table-success text-center" th:if="${users.size() != 0}">  
 <thead>  
 <tr>  
 <th class="col-md-1">Answers</th>  
 <th class="col-md-1">Topic</th>  
 <th class="col-md-2">User Display Name</th>  
 <th class="col-md-2">Email</th>  
 <th class="col-md-2">Create</th>  
 <th class="col-md-2">Action</th>  
 </tr>  
 </thead>  
 <tbody>  
 <th:block th:each="user : ${users}">  
 <tr>  
 <td th:text="${commentcount.countCommentByUser\_ID(user.ID)}">Number of answers</td>  
 <td th:text="\*{topiccount.countTopicByUser\_ID(user.ID)}">Number of topics</td>  
 <td>  
 <!-- <a th:href="@{'/profile/' + ${topic.getUser().getId()}}">-->  
 <div class="chip" th:text="${user.userDisplayName}">User Display Name</div>  
 <!-- </a>-->  
 </td>  
 <td th:text="\*{user.email}">Email</td>  
 <td th:text="\*{user.creationDate}"> User's created date </td>  
<!-- <td th:if="${user.role == 'ROLE\_USER' }">-->  
<!-- <button class="btn btn-danger" type="button">-->  
<!-- <a th:href="" class="text-white" method=RequestMethod.GET>-->  
<!-- Delete-->  
<!-- </a>-->  
<!-- </button>-->  
<!-- </td>-->  
 </tr>  
 </th:block>  
 </tbody>  
 </table>  
  
 </div>  
</div>  
</body>