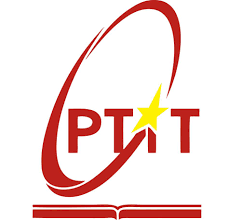
**HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG**

BỘ MÔN CÔNG NGHỆ THÔNG TIN

**[------🙠🕮🙢](https://dangdi.vn/tin-tuc/review-danh-gia-may-chu-dell-emc-poweredge-r240.html)**------



**BÁO CÁO BÀI TẬP LỚN**

**LẬP TRÌNH WEB**

**ĐỀ TÀI : HỆ THỐNG KIỂM TRA ONLINE**

**Giảng viên : Dương Trần Đức**

**Nhóm : 09**

**Nhóm BTL : 17**

Thành Viên**: Đỗ Hoàng Quân - B20DCCN546**

**Phan Anh Hiếu - B20DCCN259**

**Nguyễn Công Mạnh - B20DCCN424**

**LỜI CẢM ƠN**

Em xin gửi lời cảm ơn chân thành đến thầy giáo Dương Trần Đức - giảng viên môn Lập trình Web tại Học Viện Công Nghệ Bưu Chính Viễn Thông đã trang bị giúp nhóm em những kỹ năng cần thiết và những kiến thức cơ bản để hoàn thành được bài tập lớn trong môn học này.

Tuy nhiên, trong quá trình hoàn thiện bài tập lớn, do kiến thức chuyên ngành của nhóm em còn tồn tại một số hạn chế nên không thể tránh khỏi một vài thiếu sót khi xây dựng phần mềm và đánh giá các vấn đề tồn tại. Rất mong nhận được sự góp ý, hướng dẫn của thầy để đề tài của nhóm em sẽ hoàn thiện hơn.

Nhóm xin chân thành cảm ơn thầy ạ!

**Mục lục**

* + 1. **Đặc tả hệ thống 4**
       1. Đặc tả 4
       2. Các chức năng chính 4
    2. **Thiết kế hệ thống 4**
       1. Thiết kế data 4
       2. Thiết kế view 9
       3. Thiết kế controller 10
    3. **Kiến trúc hệ thống 13**
       1. Mô hình 13
       2. Công nghệ áp dụng 13
       3. Database 13
    4. **Kiểm thử, triển khai 14**
       1. Chạy thử trên local 14
       2. Deploy 27
    5. **Phân công công việc 27**

# Đặc tả hệ thống

## Mô tả

Hệ thống kiểm tra online dùng để tạo môi trường chung để giúp giảng viên và sinh viên có thể thực hiện các tác vụ cơ bản với nhau cũng như : Tham khảo , xem thông tin , đánh giá và kiểm tra .

## Các chức năng chính:

* 1. Đăng ký , Đăng nhập , Đăng xuất
  2. Sửa đổi thông tin cá nhân , mật khẩu ,…
  3. Thêm lớp học , xoá lớp học , tham gia lớp học , rời lớp học
  4. Xem thông tin lớp học : Tên lớp , giảng viên , danh sách sinh viên
  5. Làm bài kiểm tra , thêm, xoá , sửa bài kiểm tra
  6. Thêm, sửa, xoá câu hỏi cho bài kiểm tra
  7. Xem kết quả kiểm tra

# Thiết kế hệ thống

## Thiết kế data

### **User**

* Đây là thực thể dùng tham gia vào phiên đăng nhập của trang web.
* Với các thuộc tính: tài khoản, mật khẩu, vai trò.
* Mật khẩu của người dùng được lưu dưới dạng mã hoá (bởi lớp PasswordEncoder)

*@Data  
@Entity  
@NoArgsConstructor*public class User {  
 *@Id  
 @GeneratedValue*(strategy = GenerationType.IDENTITY)  
 private Long id;  
 *@NotNull*(message = "Bạn phải có tài khoản")  
 *@Column*(nullable = false, unique = true)  
 private String username;  
 *@NotNull*(message = "Bạn cần mật khẩu để đăng nhập")  
 *@Size*(min=6, message = "Mật khẩu của bạn cần phải có 6 ký tự")  
 *@Column*(nullable = false)  
 private String password;  
 private String role;  
}

### **Student**

* Thực thể này đại diện cho học sinh, bao gồm các thông tin cơ bản: họ và tên, email, ngày sinh, địa chỉ, số điện thoại.
* Thực thể này liên kết 1 – 1 với User

*@Data  
@Entity  
@NoArgsConstructor*public class Student {  
 *@Id  
 @GeneratedValue*(strategy = GenerationType.IDENTITY)  
 private Long id;  
 *@NotNull*(message = "Bạn phải có tài khoản")  
 *@OneToOne*(targetEntity = User.class)  
 *@OnDelete*(action = OnDeleteAction.CASCADE)  
 private User user;  
 private String email;  
 *@NotNull*(message = "Bạn phải đặt tên")  
 *@Column*(nullable = false)  
 private String name;  
 *@DateTimeFormat*(pattern = "yyyy-MM-dd")  
 private Date dateOfBirth;  
 private String address;  
 private String phoneNumber;  
}

### **Teacher**

* Thực thể này đại diện cho giảng viên, bao gồm các thông tin cơ bản: họ và tên, email, ngày sinh, địa chỉ, số điện thoại.
* Thực thể này liên kết 1 – 1 với User

*@Data  
@Entity  
@NoArgsConstructor*public class Teacher{  
 *@Id  
 @GeneratedValue*(strategy = GenerationType.IDENTITY)  
 private Long id;  
 *@NotNull*(message = "Bạn phải có tài khoản")  
 *@OneToOne*(targetEntity = User.class)  
 *@OnDelete*(action = OnDeleteAction.CASCADE)  
 private User user;  
 private String email;  
 *@NotNull*(message = "Bạn phải đặt tên")  
 *@Column*(nullable = false)  
 private String name;  
 *@DateTimeFormat*(pattern = "yyyy-MM-dd")  
 private Date dateOfBirth;  
 private String address;  
 private String phoneNumber;  
}

### **Course**

* Thực thể này đại diện cho lớp học, mỗi lớp học sẽ có các thông tin như tên lớp, mã mời, giảng viên, danh sách sinh viên.
* Thực thể này liên kết nhiều – 1 với giảng viên, liên kết nhiều – nhiều với học sinh.
* Một giảng viên có thể mở nhiều lớp học, nhưng 1 lớp học chỉ được quản lý bởi 1 giảng viên.
* Một sinh viên có thể join nhiều lớp, và một lớp cũng có nhiều sinh viên.

*@Data  
@Entity  
@NoArgsConstructor*public class Course {  
 private static final String CHARACTERS = "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789";  
 *@Id  
 @GeneratedValue*(strategy = GenerationType.IDENTITY)  
 private Long id;  
 *@NotNull*(message = "Lớp học cần phải có tên")  
 private String name;  
 *@Column*(unique = true)  
 private String code;  
 *@ManyToMany*(fetch = FetchType.EAGER, targetEntity = Student.class)  
 private List<Student> studentList;  
 *@ManyToOne*(targetEntity = Teacher.class)  
 *@OnDelete*(action = OnDeleteAction.CASCADE)  
 *@NotNull*(message = "Lớp học cần có giảng viên")  
 private Teacher teacher;  
 *@PrePersist* void code(){  
 Random random = new Random();  
 StringBuilder sb = new StringBuilder(8);  
 for (int i = 0; i < 8; i++) {  
 int randomIndex = random.nextInt(CHARACTERS.length());  
 char randomChar = CHARACTERS.charAt(randomIndex);  
 sb.append(randomChar);  
 }  
 this.code=sb.toString();  
 }  
}

### **Examination**

* Thực thể này đại diện cho bài kiểm tra. Gồm có các thông tin cơ bản của bài kiểm tra như: tên bài, mô tả, ngày bắt đầu, ngày kết thúc, thời gian làm, điểm số, hiện cho sinh viên ….
* Thực thể này liên kết nhiều – 1 với giảng viên
* Thuộc tính isVisible sẽ cho phép sinh viên được thấy bài kiểm tra hoặc không.

public class Examination {  
 *@Id  
 @GeneratedValue*(strategy = GenerationType.IDENTITY)  
 private Long id;  
 private String title = "Chưa đặt tên";  
 *@Column*(length = 500)  
 private String description = "Bài kiểm tra";  
 *@ManyToOne*(targetEntity = Course.class)  
 *@OnDelete*(action = OnDeleteAction.CASCADE)  
 *@NotNull*(message = "Bài kiểm tra phải có lớp")  
 private Course course;  
 *@NotNull*(message = "Hãy thêm ngày bắt đầu")  
 *@Column*(nullable = false)  
 private Date startDate;  
 *@NotNull*(message = "Hãy thêm ngày kết thúc")  
 *@Column*(nullable = false)  
 private Date endDate;  
 *@NotNull* private Long totalTime = (long) 600; // in seconds  
 *@Column*(nullable = false)  
 *@NotNull*(message = "Bài kiểm tra phải có điểm")  
 private Float points;  
 private Boolean isVisible = false;  
}

### **Question**

* Đây là thực thể đại diện cho câu hỏi. Mỗi câu hỏi sẽ có các thông tin như: đề bài, đáp án, đáp án đúng.
* Để đơn giản việc lưu trữ, các đáp án sẽ được join lại bởi chuỗi “#~~~#”
* Thực thể này liên kết nhiều – 1 với Examination

*@Data  
@Entity  
@NoArgsConstructor*public class Question {  
 *@Id  
 @GeneratedValue*(strategy = GenerationType.IDENTITY)  
 private Long id;  
 *@ManyToOne*(targetEntity = Examination.class)  
 *@OnDelete*(action = OnDeleteAction.CASCADE)  
 *@NotNull*(message = "Câu hỏi phải nằm trong bài kiểm tra")  
 private Examination examination;  
  
 *@Column*(nullable = false, length = 500)  
 *@NotNull*(message = "Cần phải có đề bài")  
 private String description;  
 *@Column*(length = 500)  
 *@NotNull*(message = "Cần phải có câu trả lời")  
 private String answer; // split by #~~~#  
 *@Max*(value = 4, message = "Không có đáp án nào như vậy")  
 *@NotNull*(message = "Câu hỏi cần có câu trả lời đúng")  
 private Integer correctAnswer;  
}

### **Result**

* Thực thể này đại diện cho kết quả của sinh viên sau khi hoàn thành bài kiểm tra
* Thực thể này liên kết nhiều – 1 với Examination và nhiều – 1 với Student.

*@Entity  
@Data  
@NoArgsConstructor*public class Result {  
 *@Id  
 @GeneratedValue*(strategy = GenerationType.IDENTITY)  
 private Long id;  
 *@ManyToOne*(targetEntity = Student.class)  
 *@OnDelete*(action = OnDeleteAction.CASCADE)  
 *@NotNull* private Student student;  
 *@ManyToOne*(targetEntity = Examination.class)  
 *@OnDelete*(action = OnDeleteAction.CASCADE)  
 *@NotNull* private Examination examination;  
 private Date startTime;  
 private Integer numberOfCorrectAnswers = 0;  
 private Boolean done = false;  
 *@PrePersist* void startTime(){  
 this.startTime = new Date();  
 }  
}

## Thiết kế view

### **Page**

* + - 404.html (không tìm thấy), 500.html (lỗi server), 403.html(không có quyền),  login.html, registration.html, success\_registration.html, hometeacher.html, homestudent.html, course\_student.html,
    - course\_teacher.html, profile.html, examination.html, results.html, profile.html, reset-password.html
    - **Với mọi Controller**: nếu session không có dữ liệu đăng nhập thì redirect về login.
    - **Đối với API Controller**: nếu không có dữ liệu hoặc truy cập sai cách thì trả về code 404 hoặc 403
    - 1 navbar, chứa 1 nút home (ở đầu navbar) và, 1 nút profile (ở cuối navbar)

### **login.html**

* 1 form gồm 3 input (tài khoản, mật khẩu, lưu đăng nhập), 1 link dẫn tới registration.html (nếu chưa có tk), 1 button đăng nhập
* Sau khi ấn đăng nhập, nếu oke thì return home.html, nếu không thì return login.html kèo error message.

### **registration.html**

* 1 form gồm 5 input: Họ và tên, email đăng nhập, mật khẩu, nhập lại mật khẩu, Select gồm 2 option vai trò: sinh viên hoặc giảng viên; 1 link dẫn tới login.html (nếu đã có tk); 1 button đăng ký.
* Sau khi ấn button đăng ký, nếu oke thì trả về success\_registration.html với nội dung là đăng kí thành công và 1 nút trở về home.html kèm theo variable role=student hoặc role=teacher.
* Nếu lỗi thì trả lại registration.html kèm error message và vẫn giữ nguyên value các input.

### **home.html**

* + - Chung:
      * Xin chào …
      * Danh sách lớp học dạng bảng, Gồm: id, tên lớp, tên giảng viên, số sinh viên. khi ấn vào tên sẽ dẫn sang course.html
    - Nếu là sinh viên:
      * Có 1 input để nhập mã mời vào lớp học. Nhập xong sẽ call API. nếu không tồn tại thì báo lỗi.
    - Nếu là giảng viên:
      * Sẽ có thêm 1 form tạo course, khi ấn vào sẽ gửi API, nếu succes reload lại

### **course.html**

* + - Chung:
      * tên lớp
      * tên giảng viên
      * 1 bảng hiện danh sách sinh viên, chiều cao cố định, overflow-y: scroll
      * 1 danh sách bài kiểm tra bao gồm tên, trạng thái, số học sinh đã tham gia. Click vào tên sẽ vào examination.html
    - Nếu là giảng viên:
      * Thêm 1 danh sách các sinh viên đang chờ phê duyệt bao gồm: tên sinh viên, email, bên cạnh là nút đồng ý hoặc huỷ. call API. Nếu đồng ý và thành công thì xoá khỏi hàng chờ và thêm vào danh sách sv. Nếu huỷ và thành công thì chỉ xóa khỏi hàng chờ.
      * Thêm input thêm bài kiểm tra, gồm các thông tin cơ bản.
    - Nếu là sinh viên: Danh sách bài kiểm tra chỉ hiện những bài có thuộc tính isVisible = true

### **examination.html**

* Thông tin bài kiểm tra
* Nếu là giảng viên:
  + - Hiện danh sách các câu hỏi.
    - Input thêm câu hỏi, bao gồm mô tả, các đáp án. nút thêm. Sau khi ấn, nếu thành công thì clear input và append câu vào danh sách. Nếu không thì báo lỗi.
* Nếu là sinh viên:
  + Nếu đã nộp bài thì hiện kết quả danh sách các sinh viên đã hoàn thành.
  + Nếu chưa nộp bài, và bài kiểm tra đang trong quá trình diễn ra. Hiện nút bắt đầu làm.
  + Nếu chưa nộp bài và đã bắt đầu, hiện nút tiếp tục.
* Sau khi ấn nút bắt đầu/tiếp tục. valid xem sinh viên còn đủ điều kiện tham gia không. Nếu có thì hiện danh sách bài kiểm tra kèm đồng hồ đếm ngược.

### **profile.html**

* Hiện các thông tin cở bản của người dùng dưới dạng input form. Có nút update để cập nhật thông tin

### **reset-password.html:**

* Gồm 3 form mật khẩu cũ, mật khẩu mới, nhập lại mật khẩu để người dùng có thể cập nhật

## Thiết kế controller

### **Login, Register**

* Login:
  + Nếu đăng nhập rồi thì redirect về home
  + Khi POST sẽ valid tài khoản mật khẩu, lỗi sẽ trả lại trang kèm lỗi. Thành công redirect về home
* Register:
  + Nếu đăng nhập rồi thì redirect về home
  + Khi POST sẽ valid thông tin người dùng, lỗi sẽ trả lại trang kèm lỗi. Thành công thì gửi về trang thành công

### **Home**

* + - * Gồm có 2 phương thức

+ Phương thức xử lý trang home cho học sinh.

* + - Nếu người dùng chưa đăng nhập sẽ chuyển đến trang đăng nhập
    - Lấy danh sách lớp học có chứa học từ trong cơ sở dữ liệu rồi lưu vào model để hiện thị ra bên frontend.

+ Phương thức xử lý home dành cho giáo viên.

* + - Nếu người dùng chưa đăng nhập sẽ chuyển đến trang đăng nhập
    - Lấy danh sách lớp học mà giáo viên đã tạo trong cơ sở dữ liệu rồi lưu vào model để hiển thị ra phía frontend.

### **Course**

* + - * Gồm có 2 phương thức

+ Phương thức xử lý trang course cho học sinh.

* + - Tìm lớp học theo id trên url.
    - Nếu lớp học mà sinh viên đăng nhập chưa tham gia sẽ hiển thị lỗi.
    - Tìm kiếm các bài kiểm tra có trong lớp học và có thuộc tính isVisible = true.
    - Lưu các biến cần hiển thị ở frontend vào model.

+ Phương thức xử lý trang course cho giáo viên.

* + - Tìm lớp học theo id trên url.
    - Nếu lớp học mà không phải giáo viên đăng nhập sẽ hiển thị lỗi.
    - Tím kiếm các bài kiểm tra có trong lớp học mà giáo viên đã tạo.
    - Lưu các biến cần hiển thi ở frontend vào model

### **Examination**

* /examination/id: Khi có yêu cầu GET, kiểm tra xem quyền người dùng là giảng viên hay sinh viên.
  + Nếu là giảng viên: trả về danh sách các câu hỏi đã thêm từ trước. Lúc này giao diện sẽ hiện các form để thao tác API với QuestionApi
  + Nếu là học sinh: trả về thời gian làm bài còn lại, trạng thái bài kiểm tra. Lúc này giao diện sẽ hiện trạng thái bài kiểm tra cho sinh viên. Nếu sinh viên có thể làm bài thì có thể click vào để chuyển sang trang process để thực hiện bài kiểm tra.
* /examination/id/process:
  + Chỉ cho phép sinh viên
  + GET:
    - Khởi tạo một Result cho sinh viên với done = false nếu chưa có.
    - Check xem sinh viên còn đủ điều kiện làm bài không. Nếu không sẽ trả về fobidden.   
      Nếu có sẽ trả về thời gian làm bài còn lại kèm các câu hỏi.
  + POST: khi sinh viên bấm submit sẽ post form lên server
    - Đối chiếu kết quả với đáp án đúng và lưu kết quả của sinh viên với done = true.
* /examination/id/results:
  + Nếu isvisble = false thì chỉ hiện cho giảng viên đã tạo ra bài này
  + Kiểm tra xem sinh viên và giảng viên có ở trong lớp có bài kiểm tra hay không. Nếu không trả về forbidden.
  + Trả về danh sách các bài làm đã hoàn thành của sinh viên trong lớp.
* API:
  + GET: kiểm tra quyền, đối với sinh viên thì chỉ trả những bài có isVisible = true
  + POST: kiểm tra quyền, thêm bài kiểm tra vào lớp học.
  + DELETE: kiểm tra quyền, xoá bài theo id
  + PUT: kiểm tra quyền, sửa bài theo requestbody
  + Thành công sẽ trả về object Examination (trừ delete), thất bại sẽ trả về Forbidden hoặc Notfound code.

### **Question(API):**

* + - POST: kiểm tra quyền, thêm câu hỏi vào bài kiểm tra.
    - PUT: kiểm tra quyền, sửa câu hỏi.
    - DELETE: kiểm tra quyền, xoá câu hỏi.

### **Profile**

* + - GET: gửi về thông tin của người dùng.
    - POST: valid thông tin, nếu hợp lệ sẽ lưu thông tin mới của người dùng, nếu không sẽ trả về lỗi.
    - ChangePassword: Kiểm tra mã hash của mật khẩu cũ có trùng với mật khẩu hiện tại. Đồng thời mật khẩu mới phải có ít nhất 6 kí tự.

# Kiến trúc hệ thống

## Mô hình

Website làm bài trắc nghiệm trực tuyến được xây dựng theo mô hình. Đồng thời web cũng tích hợp thêm kiến trúc REST để tạo ra các service cung cấp API cho việc liên lạc giữa client và server, điều này sẽ giúp trải nghiệm người dùng khi chỉnh sửa cập nhật những dữ liệu lớn được tốt hơn.

## Công nghệ áp dụng

* Ngôn ngữ: Java, Javascript, HTML kết hợp Thymeleaf,
* Framework: Spring, cụ thể là Spring Boot
* Thư viện:
  + Frontend: Boostrap, jquery
  + Backend: spring-boot-starter-web, spring-boot-starter-data-jpa, lombok, spring-boot-starter-validation, spring-boot-starter-security, spring-boot-starter-thymeleaf
* Công cụ:
  + Development: Maven, Spring Initializr, IntelliJ IDEA, Visual Studio Code, Github, Mysql Workbench, Postman, Google Chrome Devtools.
  + Deployment: Ngrok, supervisor
* Môi trường:
  + Development: Windows
  + Deloyment: Linux (Ubuntu)

## Database

Dự án sử dụng hệ quản trị cơ sở dữ liệu Mysql để lưu trữ

Diagram

Description automatically generated

# Kiểm thử, vận hành

## Chạy trên local

**Giao diện chung**

* **Giao diện login**

**Graphical user interface

Description automatically generated**

* **Giao diện đăng ký**

**Graphical user interface

Description automatically generated**

* **Giao diện đăng ký thành công**

**Graphical user interface, text, application, email

Description automatically generated**

* **Giao diện logout thành công**

**Graphical user interface

Description automatically generated**

* **Giao diện thay đổi thông tin**

**Graphical user interface, application

Description automatically generated**

* **Giao diện đổi mật khẩu**

**Graphical user interface, text, application, email

Description automatically generated**

**Giao diện dành cho giáo viên.**

* **Trang chủ**

**Graphical user interface, text, application

Description automatically generated**

* **Giao diện thêm lớp học**

**Graphical user interface

Description automatically generated**

* **Giao diện trang chủ sau khi thêm lớp học**

**Graphical user interface, application

Description automatically generated**

* **Giao diện xóa lớp học thành công ấn ok thì sẽ load lại trang và lớp học sẽ mất**

**Graphical user interface, application

Description automatically generated with medium confidence**

* **Giao diện lớp học.**

**Application

Description automatically generated with low confidence**

* **Giao diện thêm bài kiểm tra**

**Graphical user interface, application

Description automatically generated**

* **Giao diện sau khi thêm thông tin bài kiểm tra**

**Text

Description automatically generated with medium confidence**

* **Giao diện chỉnh sửa thông tin bài kiểm tra**

**Graphical user interface, application

Description automatically generated**

* **Giao diện bài kiểm tra**

**Graphical user interface, text, application

Description automatically generated**

* **Giao diện thêm câu hỏi**

**Graphical user interface

Description automatically generated**

* **Giao diện sau khi thêm câu hỏi**

**Graphical user interface, application

Description automatically generated**

* **Giao diện hiển thị danh sách kết quả của sinh viên làm bài**

**Graphical user interface, text, application, email

Description automatically generated**

**Giao diện dành cho sinh viên.**

* **Giao diện trang chủ.**

**Graphical user interface, application

Description automatically generated**

* **Giao diện lớp học.**

**Graphical user interface, text

Description automatically generated with medium confidence**

* **Giao diện bài kiểm tra**

**Graphical user interface, text, application, Word

Description automatically generated**

* **Giao diện khi đang làm bài kiểm tra**

**Graphical user interface, application, Teams

Description automatically generated**

* **Giao diện kết quả**

**Graphical user interface, text, application, email

Description automatically generated**

* **Giao diện bài kiểm tra chưa bắt đầu.**

**Graphical user interface, text, application

Description automatically generated**

## Deloy

Hiện tại hệ thống đang được host trên: <https://ac61-14-177-236-241.ngrok-free.app/>

# Phân công công việc

* 1. Đỗ Hoàng Quân (lead)
     1. Thiết kế hệ thống, mô tả luồng công việc.
     2. Thiết kế dữ liệu.
     3. Toàn bộ Module Examination: controller, view, api, …
     4. Question API
     5. Thay đổi thông tin người dùng (backend).
     6. Thay đổi mật khẩu.
     7. ExceptionHandler
     8. Review Code
     9. Deploy
  2. Nguyễn Công Mạnh
     1. Backend các Module Đăng nhập, Đăng kí: controller, view (tích hợp dữ liệu từ backend), …
     2. HomeController (backend)
     3. Course API (backend): CRUD + Tham gia + rời course
     4. Security configuration
     5. Thay đổi thông tin người dùng (frontend)
  3. Phan Anh Hiếu
     1. Course Controller, Course view.
     2. View các module đăng nhập, đăng kí.
     3. View home, View exception.