|  |  |
| --- | --- |
| **BỘ GIÁO DỤC VÀ ĐÀO TẠO** | **BỘ NÔNG NGHIỆP VÀ PTNT** |

**TRƯỜNG ĐẠI HỌC THỦY LỢI**



NGUYỄN BÁ CHIẾN

**XÂY DỰNG PHẦN MỀM ĐÁNH PHÁCH TỰ ĐỘNG CHO BÀI THI THỰC HÀNH TRÊN MÁY TÍNH**

ĐỒ ÁN TỐT NGHIỆP

|  |  |
| --- | --- |
| **BỘ GIÁO DỤC VÀ ĐÀO TẠO** | **BỘ NÔNG NGHIỆP VÀ PTNT** |

**TRƯỜNG ĐẠI HỌC THỦY LỢI**

NGUYỄN BÁ CHIẾN

**XÂY DỰNG PHẦN MỀM ĐÁNH PHÁCH TỰ ĐỘNG CHO BÀI THI THỰC HÀNH TRÊN MÁY TÍNH**

|  |  |
| --- | --- |
| Ngành : | Kỹ thuật phần mềm |
| Mã số: |  |

|  |  |
| --- | --- |
| NGƯỜI HƯỚNG DẪN | ThS. Nguyễn Thị Phương Dung |

**GÁY BÌA ĐỒ ÁN TỐT NGHIỆP, KHÓA LUẬN TỐT NGHIỆP**

**NGUYỄN BÁ CHIẾN ĐỒ ÁN/KL TỐT NGHIỆP HÀ NỘI, NĂM 2021**



**CỘNG HOÀ XÃ HỘI CHỦ NGHĨA VIỆT NAM**

**Độc lập - Tự do - Hạnh phúc**

«

**NHIỆM VỤ ĐỒ ÁN TỐT NGHIỆP**

**Họ tên sinh viên:** NGUYỄN BÁCHIẾN **Hệ đào tạo:** Đại học chính quy **Lớp:** 59PM1 **Ngành:** Kỹ thuật phần mềm **Khoa:** Công nghệ thông tin

**1. TÊN ĐỀ TÀI:**

Xây dựng phần mềm đánh phách tự động cho bài thi thực hành trên máy tính

**2 - NỘI DUNG CÁC PHẦN THUYẾT MINH VÀ TÍNH TOÁN:** Tỷ lệ %

|  |  |
| --- | --- |
| **Nội dung các phần** | **Tỷ lệ %** |
| **Chương 1: Tổng quan về bài toán giải quyết vần đề nghiệp vụ trên nền tảng website.** |  |
| **Chương 2: Lý thuyết và giải pháp**   * Tìm hiểu về thư viện React JS. * Tìm hiểu về cách xây dựng và hoạt động của Restful API. * Tìm hiểu về NodeJS và Express giúp xây dựng hệ thống Backend. * Tìm hiểu về cơ sở dữ liệu MongoDB. |  |
| **Chương 3: Triển khai xây dựng hệ thống**   * Mô tả hệ thống. * Biểu đồ ca sử dụng. * Biểu đồ tuần tự. * Biểu đồ hoạt động. * Phân tích, xây dựng chức năng. * Kết quả thực tế thu được. |  |

1. **GIÁO VIÊN HƯỚNG DẪN TỪNG PHẦN**

|  |  |
| --- | --- |
| **Phần** | **Họ và tên giáo viên hướng dẫn** |
| **Chương 1: Tổng quan về bài toán giải quyết vần đề nghiệp vụ trên nền tảng website.** | ThS. Nguyễn Thị Phương Dung |
| **Chương 2: Lý thuyết và giải pháp** | ThS. Nguyễn Thị Phương Dung |
| **Chương 3: Triển khai xây dựng hệ thống** | ThS. Nguyễn Thị Phương Dung |

1. **NGÀY GIAO NHIỆM VỤ ĐỒ ÁN TỐT NGHIỆP**

Ngày tháng năm 20

|  |  |
| --- | --- |
| **Trưởng Bộ môn**  *(Ký và ghi rõ Họ tên)* | **Giáo viên hướng dẫn chính**  *(Ký và ghi rõ Họ tên)* |

Nhiệm vụ Đồ án tốt nghiệp đã được Hội đồng thi tốt nghiệp của Khoa thông qua.

Ngày. . . . .tháng. . . . .năm 202..

Chủ tịch Hội đồng

*(Ký và ghi rõ Họ tên)*

Sinh viên đã hoàn thành và nộp bản Đồ án tốt nghiệp cho Hội đồng thi ngày... tháng...

năm 202...

Sinh viên làm Đồ án tốt nghiệp

*(Ký và ghi rõ Họ tên)*

Nguyễn Bá Chiến

|  |  |
| --- | --- |
| logo | TRƯỜNG ĐẠI HỌC THUỶ LỢI  **KHOA CÔNG NGHỆ THÔNG TIN**  BẢN TÓM TẮT ĐỀ CƯƠNG ĐỒ ÁN TỐT NGHIỆP |

Tên đề tài: Xây dựng phần mềm đánh phách tự động cho bài thi thực hành trên máy tính

*Sinh viên thực hiện* : Nguyễn Bá Chiến

*Lớp* : 59PM1

*Giáo viên hướng dẫn*: ThS Nguyễn Thị Phương Dung

**TÓM TẮT ĐỀ TÀI**

Hiện nay các kỳ thi thực hành trên máy tính tại các trường đại học diễn ra khá nhiều và mỗi kỳ thi như vậy lại có rất nhiều sinh viên tham gia. Sau mỗi buổi thi, bài thi sẽ được chuyển đến cán bộ khảo thí thực hiện đánh phách bài thi rồi mới chuyển đến cho cán bộ chấm thi để đảm bảo tính công bằng. Công việc đánh phách bài thi thực hành không giống như đánh phách bài thi trên giấy. Mỗi bài thi sẽ có tên file tương ứng với thông tin của thí sinh dự thi. Việc đánh phách sẽ là đổi tên từng file bài thi đó. Nhưng với việc có quá nhiều bài thi có thể dẫn đến tình trạng đánh phách sai, dẫn đến thông tin bài thi bị sai lệch. Nắm bắt được điều đó em đã lựa chọn đề tài xây dựng phần mềm đánh phách tự động bài thi thực hành trên máy tính, giúp cán bộ đánh phách dễ dàng hơn trong việc đánh mã phách bài thi và chuyển đến cán bộ chấm thi. Việc này cũng sẽ giúp cho công đoạn ghép phách dễ dàng được thực hiện tự động mà không xảy ra sai sót gì.

**CÁC MỤC TIÊU CHÍNH**

* Tìm hiểu về công nghệ sử dụng:
  + Công cụ: Visual Studio Code.
  + Các ngôn ngữ lập trình: HTML, CSS, JS.
  + Cơ sở dữ liệu: MongoDB.
* Xây dựng phần mềm đánh phách tự động gồm các chức năng:
  + Thực hiện đánh phách tự động theo phòng thi, môn thi.
  + Thực hiện ghép phách tự động.
  + Lưu vết quá trình thực hiện
* Xây dựng trang web hỗ trợ với các chức năng sau:
  + Thực hiện giao bài cho cán bộ chấm thi
  + Hỗ trợ cán bộ chấm thi nhập điểm.

**KẾT QUẢ DỰ KIẾN**

* Hoàn thiện hệ thống phần mềm.
* Báo cáo đồ án tốt nghiệp.

**LỜI CAM ĐOAN**

Tác giả xin cam đoan đây là Đồ án tốt nghiệp của bản thân tác giả. Các kết quả trong Đồ án tốt nghiệp này là trung thực và không sao chép từ bất kỳ một nguồn nào và dưới bất kỳ hình thức nào.Việc tham khảo các nguồn tài liệu (nếu có) đã được thực hiện trích dẫn và ghi nguồn tài liệu tham khảo đúng quy định.

**Tác giả DATN/KLTN**

**Nguyễn Bá Chiến**

**LỜI CÁM ƠN**

Trong quá trình thực hiện luận văn này, tôi đã nhận được rất nhiều sự động viên, giúp đỡ của nhiều cá nhân và tập thể.

Trước tiên, tôi xin bày tỏ lòng cảm ơn sâu sắc tới các thầy cô trong khoa Công nghệ thông tin và trung tâm tin học trường Đại học Thủy Lợi đã cung cấp kiến thức, kỹ năng và truyền dạy kinh nghiệm cho tôi trong suốt khóa học, đặc biệt tôi xin gửi lời cảm ơn ThS. Nguyễn Thị Phương Dung, đã nhiệt tình hướng dẫn, góp ý và tạo điều kiện thuận lợi để tôi có thể hoàn thành Đồ án tốt nghiệp một cách tốt nhất.

Tiếp theo, tôi xin gửi lời cảm ơn đến các thầy cô ở Trung tâm tin học trường Đại học Thủy Lợi đã luôn động viên, giúp đỡ và hướng dẫn tôi trong quá trình hoàn thành đồ án tốt nghiệp.

Cuối cùng, tôi cũng xin chân thành cảm ơn các anh, các chị và các bạn học lớp 59PM1 trường Đại học Thủy Lợi đã luôn động viên, giúp đỡ và nhiệt tình chia sẻ với tôi những kinh nghiệm học tập, công tác trong suốt khoá học.

*Hà Nội, ngày tháng năm 2021*

**Sinh viên thực hiện**

**Nguyễn Bá Chiến**

# ­MỤC LỤC

[DANH MỤC CÁC HÌNH ẢNH v](#_Toc75252476)

[DANH MỤC CÁC BẢNG BIỂU viii](#_Toc75252477)

[DANH MỤC CÁC TỪ VIẾT TẮT VÀ GIẢI THÍCH CÁC THUẬT NGỮ ix](#_Toc75252478)

[CHƯƠNG 1 TỔNG QUAN VỀ BÀI TOÁN GIẢI QUYẾT VẤN ĐỀ NGHIỆP VỤ TRÊN NỀN TẢNG WEBSITE TRỰC TUYẾN 1](#_Toc75252479)

[1.1 Tổng quan về website. 1](#_Toc75252480)

[1.1.1 Website là gì? 1](#_Toc75252481)

[1.1.2 Vai trò của website. 2](#_Toc75252482)

[1.1.3 Ứng dụng web là gì? 2](#_Toc75252483)

[1.1.4 Cách hoạt động của ứng dụng web. 3](#_Toc75252484)

[1.1.4 Ưu điểm của ứng dụng web. 3](#_Toc75252485)

[1.1.5 Sự khác nhau giữa website và ứng dụng web 4](#_Toc75252486)

[1.2 Phát triển và triển khai một ứng dụng web. 4](#_Toc75252487)

[1.2.1 Giai đoạn 1: Lên ý tưởng 4](#_Toc75252488)

[1.2.2 Giai đoạn 2: Xây dựng ứng dụng web 5](#_Toc75252489)

[1.2.3 Giai đoạn 3: Triển khai ứng dụng lên server 5](#_Toc75252490)

[CHƯƠNG 2 CÔNG NGHỆ SỬ DỤNG VÀ GIẢI PHÁP 6](#_Toc75252491)

[2.1 Tổng quan về ReactJS 6](#_Toc75252492)

[2.1.1 ReactJS là gì ? 6](#_Toc75252493)

[2.1.2 Tại sao lại là ReactJS? 6](#_Toc75252494)

[2.1.3 Các thư viện sử dụng để xây dựng giao diện? 10](#_Toc75252495)

[2.2 Tổng quan về Restful API 11](#_Toc75252496)

[2.2.1 Restful API là gì? 11](#_Toc75252497)

[2.2.2 Cách thức hoạt động của Resfull API 13](#_Toc75252498)

[2.2.3 Tại sao lại là Restfull API? 14](#_Toc75252499)

[2.3 Tổng quan về NodeJS/ExpressJS 16](#_Toc75252500)

[2.3.1 NodeJS là gì? 16](#_Toc75252501)

[2.3.2 Tại sao lại là NodeJS? 18](#_Toc75252502)

[2.3.3 Cách thức hoạt động của NodeJS 19](#_Toc75252503)

[2.3.4 ExpressJS là gì? 21](#_Toc75252504)

[2.3.5 Tại sao lại là ExpressJS? 22](#_Toc75252505)

[2.3.6 Cách thức hoạt động của ExpressJS 23](#_Toc75252506)

[2.4 Tổng quan về MongoDB 24](#_Toc75252507)

[2.4.1 MongoDB là gì? 24](#_Toc75252508)

[2.4.2 Tại sao lại là MongoDB 26](#_Toc75252509)

[CHƯƠNG 3 TRIỂN KHAI XÂY DỰNG HỆ THỐNG 27](#_Toc75252510)

[3.1 Mô tả hệ thống 27](#_Toc75252511)

[3.1.1 Thành phần chính của hệ thống 27](#_Toc75252512)

[3.1.2 Công nghệ sử dụng 27](#_Toc75252513)

[3.2 Thiết kế hệ thống 28](#_Toc75252514)

[3.2.1 Biểu đồ ca sử dụng 28](#_Toc75252515)

[3.2.2 Biểu đồ tuần tự 30](#_Toc75252516)

[3.2.3 Biểu đồ hoạt động 32](#_Toc75252517)

[3.2.3 Cơ sở dữ liệu 33](#_Toc75252518)

[3.3 Xây dựng chức năng 36](#_Toc75252519)

[3.3.1 Chức năng đăng nhập 36](#_Toc75252520)

[3.3.2 Chức năng đăng xuất 36](#_Toc75252521)

[3.3.3 Chức năng đăng ký 36](#_Toc75252522)

[3.3.4 Chức năng thay đổi mật khẩu tài khoản 37](#_Toc75252523)

[3.3.5 Chức năng tạo mới kỳ thi 38](#_Toc75252524)

[3.3.6 Chức năng tạo mới ca thi 38](#_Toc75252525)

[3.3.7 Chức năng tạo mới môn thi 38](#_Toc75252526)

[3.3.8 Chức năng tạo mới phòng thi 39](#_Toc75252527)

[3.3.9 Chức năng khóa tài khoản 39](#_Toc75252528)

[3.3.10 Chức năng phân công người chấm thi 39](#_Toc75252529)

[3.3.11 Chức năng yêu cầu chấm lại bài thi 40](#_Toc75252530)

[3.3.12 Chức năng phê duyệt bài thi sau khi chấm 40](#_Toc75252531)

[3.3.13 Chức năng tải xuống File Excel điểm thi 40](#_Toc75252532)

[3.3.13 Chức năng chấm thi 41](#_Toc75252533)

[3.4 Kiểm thử 42](#_Toc75252534)

[3.4.1 Chức năng đăng nhập 42](#_Toc75252535)

[3.4.2 Chức năng đăng xuất 42](#_Toc75252536)

[3.4.3 Chức năng đăng ký 43](#_Toc75252537)

[3.4.4 Chức năng thay đổi mật khẩu 43](#_Toc75252538)

[3.4.5 Chức năng tạo mới kỳ thi 44](#_Toc75252539)

[3.4.6 Chức năng tạo mới ca thi 44](#_Toc75252540)

[3.4.7 Chức năng tạo mới môn thi 45](#_Toc75252541)

[3.4.8 Chức năng tạo mới phòng thi 45](#_Toc75252542)

[3.4.9 Chức năng khóa tài khoản 46](#_Toc75252543)

[3.4.10 Chức năng yêu cầu chấm lại bài thi 46](#_Toc75252544)

[3.4.11 Chức năng phê duyệt bài thi sau khi chấm 47](#_Toc75252545)

[3.4.12 Chức năng tải xuống File Excel điểm thi 47](#_Toc75252546)

[3.4.13 Chức năng chấm thi 48](#_Toc75252547)

[3.5 Kết quả sau khi xây dựng hệ thống 49](#_Toc75252548)

[3.5.1 Client 49](#_Toc75252549)

[3.5.2 Server 55](#_Toc75252550)

[CHƯƠNG 4 KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN 56](#_Toc75252551)

[4.1 Kết quả thu được 56](#_Toc75252552)

[4.1.1 Lý thuyết 56](#_Toc75252553)

[4.1.2 Thực hành 56](#_Toc75252554)

[4.1 Hạn chế 56](#_Toc75252555)

[4.1 Hướng phát triển 56](#_Toc75252556)

[TÀI LIỆU THAM KHẢO 57](#_Toc75252557)

# DANH MỤC CÁC HÌNH ẢNH

[Hình 1. 1: cách thức hoạt động của ứng dụng web 3](#_Toc75252147)

[Hình 2. 1 Virual DOM và Real DOM 7](#_Toc75252178)

[Hình 2. 2 Trạng thái vòng đời của một component 8](#_Toc75252179)

[Hình 2. 3 Minh họa về REST API 11](#_Toc75252180)

[Hình 2. 4 Cách thức hoạt động của RestfulAPI 13](#_Toc75252181)

[Hình 2. 5 Thống kê các ngôn ngữ được sử dụng nhiều nhất 16](#_Toc75252182)

[Hình 2. 6 Mô tả kiến trúc của NodeJS 19](#_Toc75252183)

[Hình 2. 7 Minh họa cơ chế Event Loop 20](#_Toc75252184)

[Hình 2. 8 Minh họa mô hình ứng dụng Express 22](#_Toc75252185)

[Hình 2. 9 Cấu trúc cơ bản của một dự án Express JS 23](#_Toc75252186)

[Hình 2. 10 So sanh tốc độ của MongoDB với MYSQL 26](#_Toc75252187)

[Hình 3. 1 Biểu đồ ca sử dụng của quản trị viên 28](#_Toc75252188)

[Hình 3. 2 Biểu đồ ca sử dụng của cán bộ quản và cán bộ chấm thi. 29](#_Toc75252189)

[Hình 3. 4 Biểu đô tuần tự chức năng đăng nhập 30](#_Toc75252190)

[Hình 3. 5 Biểu đô tuần tự các chức năng nghiệp vụ 31](#_Toc75252191)

[Hình 3. 6 Biểu đô tuần tự chức năng đăng nhập 32](#_Toc75252192)

[Hình 3. 7 Chức năng đăng nhập 36](#_Toc75252193)

[Hình 3. 8 Chức năng đăng xuất 36](#_Toc75252194)

[Hình 3. 9 Chức năng đăng ký tài khoản cho quản trị viên 37](#_Toc75252195)

[Hình 3. 10 Chức năng thay đổi mật khẩu 37](#_Toc75252196)

[Hình 3. 11 Chức năng tạo mới kỳ thi 38](#_Toc75252197)

[Hình 3. 12 Chức năng tạo mới ca thi 38](#_Toc75252198)

[Hình 3. 13 Chức năng tạo mới ca thi 38](#_Toc75252199)

[Hình 3. 14 Chức năng tạo mới phòng thi 39](#_Toc75252200)

[Hình 3. 15 Chức năng khóa tài khoản 39](#_Toc75252201)

[Hình 3. 16 Chức năng phân công người chấm thi 39](#_Toc75252202)

[Hình 3. 17 Chức năng yêu cầu chấm lại bài thi 40](#_Toc75252203)

[Hình 3. 18 Chức năng tạo mới kỳ thi 40](#_Toc75252204)

[Hình 3. 19 Chức năng tải xuống file excel điểm thi 41](#_Toc75252205)

[Hình 3. 20 Chức năng chấm thi 41](#_Toc75252206)

[Hình 3. 21 Kiểm thử chức năng đăng nhập 42](#_Toc75252207)

[Hình 3. 22 Kiểm thử chức năng đăng xuất 42](#_Toc75252208)

[Hình 3. 23 Kiểm thử chức năng đăng ký 43](#_Toc75252209)

[Hình 3. 24 Kiểm thử chức năng thay đổi mật khẩu 43](#_Toc75252210)

[Hình 3. 25 Kiểm thử chức năng tạo mới kỳ thi 44](#_Toc75252211)

[Hình 3. 26 Kiểm thử chức năng tạo mới ca thi 44](#_Toc75252212)

[Hình 3. 27 Kiểm thử chức năng tạo mới môn thi 45](#_Toc75252213)

[Hình 3. 28 Kiểm thử chức năng tạo mới phòng thi 45](#_Toc75252214)

[Hình 3. 29 Kiểm thử chức năng khóa tài khoản 46](#_Toc75252215)

[Hình 3. 30 Kiểm thử chức năng yêu cầu chấm lại. 46](#_Toc75252216)

[Hình 3. 31 Kiểm thử chức năng phê duyệt bài thi sau khi chấm 47](#_Toc75252217)

[Hình 3. 32 Kiểm thử chức năng đăng nhập 47](#_Toc75252218)

[Hình 3. 33 Kiểm thử chức năng chấm thi 48](#_Toc75252219)

[Hình 3. 34 Màn hình đăng nhập 49](#_Toc75252220)

[Hình 3. 35 Màn hình quản lý tài khoản 49](#_Toc75252221)

[Hình 3. 36 Màn hình quản lý kỳ thi 50](#_Toc75252222)

[Hình 3. 37 Màn hình quản lý môn học và phòng thi 50](#_Toc75252223)

[Hình 3. 38 Màn hình thông tin cá nhân 51](#_Toc75252224)

[Hình 3. 39 Màn hình thêm mới tài khoản 51](#_Toc75252225)

[Hình 3. 40 Màn hình thay đổi thông tin người dùng 52](#_Toc75252226)

[Hình 3. 41 Màn hình thêm mới ca thi 52](#_Toc75252227)

[Hình 3. 42 Màn hình phân chia ca thi và người chấm 53](#_Toc75252228)

[Hình 3. 43 Màn hình quản lý các ca thi đã phân công chấm 53](#_Toc75252229)

[Hình 3. 44 Màn hình xem điểm của ca thi khi đã chấm xong 54](#_Toc75252230)

[Hình 3. 45 Màn hình các ca thi đã được phân công của cán bộ chấm thi 54](#_Toc75252231)

[Hình 3. 46 Màn hình chấm thi của cán bộ chấm thi 55](#_Toc75252232)

# DANH MỤC CÁC BẢNG BIỂU

[Bảng 1. 1 Sự khác nhau giữa website và web application 4](#_Toc75252089)

[Bảng 2. 1 Minh họa về cách viết enpoint trong Restful API 12](#_Toc75252099)

[Bảng 3. 1 Bảng mô tả database collection bài thi 33](#_Toc75252106)

[Bảng 3. 2 Bảng mô tả database collection người dùng 34](#_Toc75252107)

[Bảng 3. 3 Bảng mô tả database collection kỳ thi 34](#_Toc75252108)

[Bảng 3. 4 Bảng mô tả database collection ca thi 35](#_Toc75252109)

[Bảng 3. 5 Bảng mô tả database collection phòng thi 35](#_Toc75252110)

[Bảng 3. 6 Bảng mô tả database collection môn thi 35](#_Toc75252111)

# DANH MỤC CÁC TỪ VIẾT TẮT VÀ GIẢI THÍCH CÁC THUẬT NGỮ

|  |  |
| --- | --- |
| **Từ viết tắt** | **Thuật ngữ** |
| HTML | Hyper Text Markup Language |
| CSS | Cascading Style Sheets |
| JS | Javascript |
| DOM | Document Object Model |
| SPA | Single Page Application |
| XML | Extensible Markup Language |
| Dev | Development |
| SEO | Search Engine Optimize |
| HTTP | Hypertext Transfer Protocol |
| RDBMS | Relational Database Management Syste |

# CHƯƠNG 1 TỔNG QUAN VỀ BÀI TOÁN GIẢI QUYẾT VẤN ĐỀ NGHIỆP VỤ TRÊN NỀN TẢNG WEBSITE TRỰC TUYẾN

*Chương 1 nói về những vấn đề tổng quan giải quyết vấn đề nghiệp vụ trên nền tảng website trực tuyến bao gồm 2 phần chính: Tổng quan về website, Phát triển và triển khai một ứng dụng web.*

## **Tổng quan về website.**

### ***1.1.1 Website là gì?***

Website, còn gọi là trang web hoặc trang mạng, là một tập hợp trang web, thường chỉ nằm trong một tên miền hoặc tên miền phụ trên World Wide Web của Internet. Một trang web là tập tin HTML hoặc XHTML có thể truy nhập dùng giao thức HTTP. Trang mạng có thể được xây dựng từ các tệp tin HTML (trang mạng tĩnh) hoặc vận hành bằng các CMS chạy trên máy chủ (trang mạng động).

Website có thể được xây dựng bằng nhiều ngôn ngữ lập trình khác nhau (PHP, ASP.NET, Java, Ruby on Rails, Perl,...).

Website được giao tiếp và hiển thị cho người dùng truy cập bằng các phần mềm được gọi là trình duyệt website. Một số trình duyệt website nổi tiếng có thể kể đến như Internet Explorer được cài đặt mặc định vào mỗi máy tính cài hệ điều hành windows và được thay thế bởi Edge trên Windows 10 hay Chrome được phát triển bởi Google và Firefox được phát triển bởi Mozilla

Hai loại website là: website tĩnh và website động:

* Website tĩnh là website mà người quản trị (những người không phải là lập trình viên) không thể tùy ý thay đổi nội dung và hình ảnh mà phải cần kiến thức về HTML cơ bản. Website tĩnh được viết hoàn toàn dựa trên nền tảng HTML CSS và thêm các hiệu ứng từ Javascript nếu muốn.
* Website động là website được viết kèm theo một bộ công cụ quản trị để tùy biến nội dung dành cho webmaster (người quản trị) có thể dễ dàng thay đổi nội dung, hình ảnh. Website động được thiết kế bởi các lập trình viên để làm sao cho phép website có thể thay đổi được nội dung thường xuyên. Một số công nghệ, ngôn ngữ để xây dựng website động bao gồm PHP, ASP.NET, Java,...

### ***1.1.2 Vai trò của website.***

Trong thời đại kỹ thuật số hiện nay thì không thể nói không với Internet. Nó gần như phủ sóng từ thành phố đến nông thôn, từ trẻ đến già đều sử dụng thuần thục. 1 ngày không internet cảm giác như bị lạc trôi về thời kỳ “đồ đá”. Cuộc sống con người giờ đây gần như diễn ra song hành cùng với sự tồn tại của internet. Từ mua sắm, giải trí, xem tin tức, giao tiếp xã hội, kinh doanh, học tập, v.v mọi thứ đều ít nhiều liên quan đến Internet. Vì thế website có vai trò hết sức quan trọng và không thể thiếu đối với đời sống con người.

### ***1.1.3 Ứng dụng web là gì?***

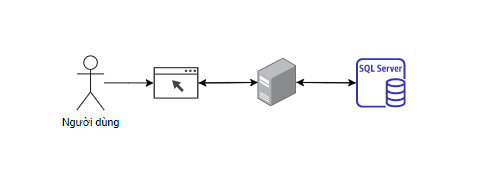
Ứng dụng web là chương trình máy tính sử dụng trình duyệt web và công nghệ web để thực hiện các tác vụ qua Internet.

Các ứng dụng web (web application) sử dụng kết hợp các server-side scripts (PHP và ASP) để xử lý việc lưu trữ và truy xuất thông tin, và client-side scripts (JavaScript và HTML) để trình bày thông tin cho người dùng.

Các ứng dụng web thường được mã hóa bằng ngôn ngữ được trình duyệt hỗ trợ như JavaScript và HTML vì các ngôn ngữ này dựa trên trình duyệt để render chương trình thực thi. Có một số ứng dụng động yêu cầu quá trình xử lý phía máy chủ, còn lại các ứng dụng tĩnh sẽ hoàn toàn không cần xử lý ở phía máy chủ

Ứng dụng web yêu cầu một web server để quản lý các yêu cầu từ máy khách, một application server để thực hiện các tác vụ được yêu cầu và đôi khi, một database để lưu trữ thông tin. Công nghệ application server có các loại từ ASP.NET, ASP và ColdFusion, đến PHP và JSP.

### ***1.1.4 Cách hoạt động của ứng dụng web.***



Hình 1. 1: cách thức hoạt động của ứng dụng web

**B1**: Người dùng kích hoạt request tới web server qua Internet, thông qua trình duyệt web hoặc giao diện người dùng của ứng dụng.

**B2**: Web server chuyển tiếp request này đến web application server thích hợp.

**B3**: Máy chủ ứng dụng Web (web application server) thực hiện nhiệm vụ được yêu cầu - chẳng hạn như truy vấn cơ sở dữ liệu hoặc xử lý dữ liệu - sau đó tạo ra các kết quả của dữ liệu được yêu cầu.

**B4**: Máy chủ ứng dụng web gửi kết quả đến máy chủ web với thông tin được yêu cầu hoặc dữ liệu đã được xử lý

**B5**: Máy chủ web phản hồi response lại cho khách hàng các thông tin được yêu cầu sau đó xuất hiện trên màn hình của người dùng.

### ***1.1.4 Ưu điểm của ứng dụng web.***

* Các ứng dụng web chạy trên nhiều nền tảng bất kể hệ điều hành hay thiết bị, miễn là trình duyệt tương thích.
* Tất cả người dùng đều được truy cập cùng một phiên bản, điều này giúp loại bỏ mọi sự cố tương thích.
* Chúng không được cài đặt trên ổ cứng, do đó loại bỏ được các hạn chế về không gian.
* Chúng làm giảm sự vi phạm bản quyền trong các ứng dụng web dựa trên đăng ký (ví dụ: SaaS)
* Chúng giảm chi phí cho cả doanh nghiệp và người dùng cuối, lý do: doanh nghiệp không cần sự support và bảo trì cho các web application này, chưa kể chúng cũng ko đòi hỏi yêu cầu cao ở máy tính người dùng cuối để có thể thực thi.

### ***1.1.5 Sự khác nhau giữa website và ứng dụng web***

|  |  |
| --- | --- |
| **Website** | **Web App** |
| Tính tương thích thấp, ít chức năng (xem, đọc, click qua lại giữa các link có trong website) | Tính tương tác với người dùng cao, có nhiều chức năng (upload file, xuất hình ảnh, video, báo cáo,…) |
| Được tạo từ các trang HTML tĩnh và một số tài nguyên tĩnh đi kèm như video, hình ảnh hoặc một vài âm thanh | HTML được tạo tự động theo yêu cầu từ request mà người dùng gửi lên và triển khai ở phía Backend Server |
| Được dùng chủ yếu để hiển thị thông tin, lưu trữ là chính. | Được dùng để thực hiện một công việc cụ thể, thực hiện các chức năng của ứng dụng |

Bảng 1. 1 Sự khác nhau giữa website và web application

## **Phát triển và triển khai một ứng dụng web.**

Việc phát triển một ứng dụng web sẽ đều trải qua 3 giai đoạn chính là lên ý tưởng cần những gì cần làm và bắt tay vào giai đoạn xây dựng ứng dụng từ ý tưởng đó, bước cuối cùng là triển khai ứng dụng lên máy chủ.

### ***1.2.1 Giai đoạn 1: Lên ý tưởng***

Lên ý tưởng là một bước rất quan trong trong việc phát triển một ứng dụng web, việc lên ý tưởng và xác định tính khả thi mà ứng dụng đem lại là điều hết sức quan trọng. Nếu ý tưởng quá mơ hồ, không rõ ràng hoặc quá phi lý thì đều dẫn đến thất bại trong quá trình bắt tay vào triển khai xây dựng sau này.

Cần xác định được rõ các mục tiêu mà ứng dụng web cần đạt được như các chức năng chính mà ứng dụng đem lại, đối tượng người sử dụng và sản phẩm này đã có mặt trên thị trường chưa.

Cần vạch ra chi tiết và cụ thể các chức năng của ứng dụng ví dụ như chức năng quản trị người dùng của ứng dụng, trong chức năng quản trị người dùng này cần có các tính năng cơ bản cốt lõi cần có như tạo người dùng, cập nhật thông tin người dùng, thay đổi mật khẩu, thay đổi anh đại diên, …

### ***1.2.2 Giai đoạn 2: Xây dựng ứng dụng web***

Sau khi đã lên ý tưởng và xác đinh được tính khả thi mà ý tưởng đó có thể thực hiện được, chúng ta tiến hành tới giai đoạn tiếp theo, đó là giai đoạn xây dựng ứng dụng.

Các bước cần tiến hành tại giai đoạn này như sau:

* B1: Xây dựng bản phác thảo, thiết kế của ứng dụng web bằng các công cụ hỗ trợ chuyên dụng như Photoshop, Figma, Adobe Illustration,…
* B2: Xây dựng cơ sở dữ liệu để lưu trữ trên hệ thống của ứng dụng, tại đây chúng ta có thể lựa chọn các hệ quản trị Database như SQL Server, MySQL hoặc MongoDB,...
* B3: Xây dựng giao diện cho ứng dụng web, chúng ta có thể chọn các ngôn ngữ để xây dựng giao diện như HTML, CSS, JS hoặc có thể chọn một số bộ khung phát triển giao diện như Angular, React, Vue,…
* B4: Xây dựng hệ thống Backend, chúng ta có thể lựa chọn các ngôn ngữ lập trình hệ thống như Python, Java,… Tại đây chúng ta sẽ xây dựng các endpoint để cho phía giao diện kết nối được tới, có thể triển khai theo mô hình tiêu chuẩn Rest, RestfullAPI.

### ***1.2.3 Giai đoạn 3: Triển khai ứng dụng lên server***

Sau khi đã trải qua giai đoạn phát triển của ứng dụng, chúng ta tiến hành thực hiện giai đoạn cuối cùng đó chính là triển khai ứng dụng của chúng ta lên môi trường Internet.

Tiến hành mua máy chủ lưu trữ (hosting) và tên miền cho ứng dụng và đẩy ứng dụng lên hosting.

**Tiểu kết chương 1:**

Tại chương đầu tiên này luận văn đã trình bày được các vấn đề:

* Định nghĩa về một website và một ứng dụng web (web app) là gì.
* Ưu điểm của ứng dụng web so với một website thông thường.
* Các giai đoạn triển khai một ứng dụng web.

Trong chương tiếp theo, luận văn sẽ trình bày cụ thể về các công nghệ được sử dụng trong việc xây dựng đề tài này.

# CHƯƠNG 2 CÔNG NGHỆ SỬ DỤNG VÀ GIẢI PHÁP

## **2.1 Tổng quan về ReactJS**

### ***2.1.1 ReactJS là gì ?***

Ngày nay với sự phát triển không ngừng của công nghệ nói chung và thế giới lập trình nói riêng, việc xây dựng giao diện cho một ứng dụng web không chỉ đơn thuần sử dụng các ngôn ngữ đánh dấu siêu văn bản như HTML, CSS và một chút JS để tương tác với trình duyệt. Việc đòi hỏi nhu cầu nhiều hơn ở một ứng dụng web đã khiến cho rất nhiều thư viện và framework ra đời nhằm tối ưu và tăng tính trải nghiệm ứng dụng của người dùng. Trong đó phải kể đến sự ra đời của thư viện ReactJS được phát triển bởi tập đoàn Facebook và được công bố rộng dãi đến các lập trình viên vào năm 2013.

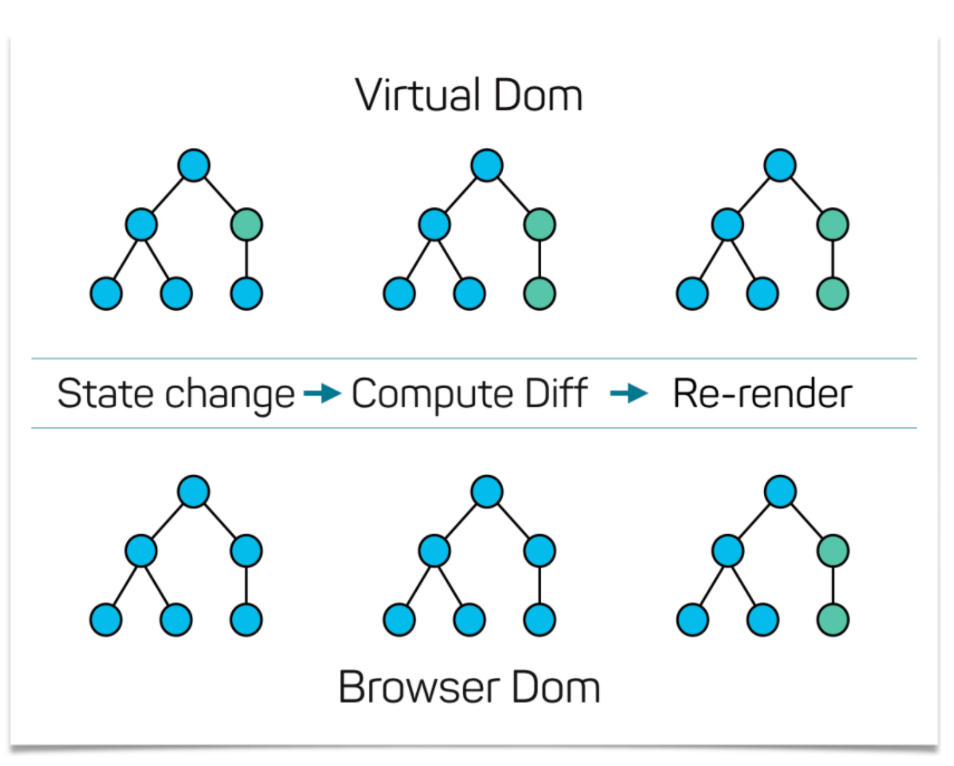
Sự ra đời của React đã làm thay đổi khái niệm đơn thuần về trải nghiệm một ứng dụng trên nền tảng web của người dùng, giờ đây một ứng dụng web có thể làm được nhiều điều hơn trước khi chỉ tương tác một cách nghèo nàn các chức năng.

### ***2.1.2 Tại sao lại là ReactJS?***

ReactJS sử dụng công nghệ Single Page Application (SPA), công nghệ này cho phép người dùng không cần phải tải lại trình duyệt mỗi khi tương tác với ứng dụng web, vì thế làm tăng tính trải nghiệm của người dùng so với ứng dụng web thông thường mỗi lần đều phải tải lại trang. Thay vì sử dụng một Document Object Model (DOM) thật, React đã sử dụng một công nghệ có tên là Virtual DOM, công nghệ ảo hóa này giúp tăng hiệu xuất cho ứng dụng web lên rất nhiều lần so với cách làm thông thường nhờ việc tính toán sự thay đổi của Virtual DOM sau đó chỉ cập nhật những phần thay đổi lên trên giao diện người dùng.

React sử dụng cơ chế one-way data binding – luồng dữ liệu 1 chiều. Dữ liệu được truyền từ thành phần cha đến thành phần con thông qua props. Luồng dữ liệu đơn giản giúp chúng ta dễ dàng kiểm soát cũng như sửa lỗi.

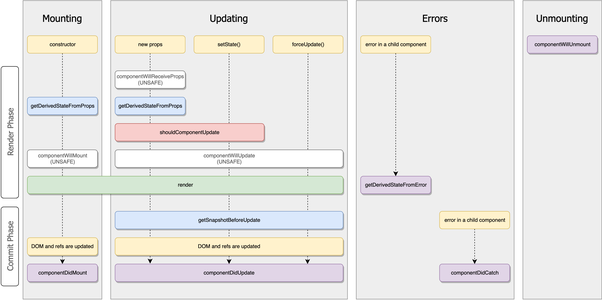
Với các đặc điểm ở trên, React dùng để xây dựng các ứng dụng lớn mà dữ liệu của chúng thay đổi liên tục theo thời gian. Dữ liệu thay đổi thì hầu hết kèm theo sự thay đổi về giao diện. Ví dụ như Facebook: trên trang chủ của bạn cùng lúc sẽ có các status khác nhau và mỗi status lại có số lượt like, share, comment liên tục thay đổi. Khi đó React sẽ rất hữu ích để sử dụng.



Hình 2. 1 Virual DOM và Real DOM

React được xây dựng xung quanh các component, chứ không dùng template như các framework khác. Trong React, chúng ta xây dựng trang web sử dụng những thành phần (component) nhỏ. Chúng ta có thể tái sử dụng một component ở nhiều nơi, với các trạng thái hoặc các thuộc tính khác nhau, trong một component lại có thể chứa thành phần khác. Mỗi component trong React có một trạng thái riêng, có thể thay đổi, và React sẽ thực hiện cập nhật component dựa trên những thay đổi của trạng thái. Mọi thứ trong React đều là component. Chúng giúp bảo trì mã code khi làm việc với các dự án lớn. Một react component đơn giản chỉ cần một phương thức (method) render. Có rất nhiều methods khả dụng khác, nhưng render là method chủ đạo.

Trong một chương trình thì có rất nhiều các component, để đơn giản việc quản lý trạng thái của ứng dụng và các component đó người ta sử dụng redux, redux giống như 1 cái kho chứa toàn bộ trạng thái của ứng dụng và các component và khi ứng dụng ở trạng thái nào thì component tương ứng sẽ được hiển thị trên trình duyệt.



Hình 2. 2 Trạng thái vòng đời của một component

Vòng đời của một React Component:

* constructor(props)
  + Được gọi khi một thể hiện của component được tạo ra.
  + Có thể dùng để khởi tạo state cho component.
  + Nếu không làm gì thì không phải cài đặt hàm này.
  + Nếu phải cài đặt hàm này thì phải khai báo 1 tham số props cho nó và phải gọi super(props) đầu tiên.
* componentWillMount()
  + Được gọi trước khi render().
  + Dùng để đăng ký các sự kiện toàn cục.
  + Dựa vào các props để tính toán và set lại state.
* render()
  + Hàm này bắt buộc phải có trong component().
  + Trả về 1 đối tượng JSX (có thể lồng các đối tượng với nhau nhưng phải có 1 đối tượng gói tất cả các đối tượng lại) để hiển thị hoặc null / false nếu không muốn hiển thị gì.
  + Không được gọi setState() trong hàm này (cũng như trong các hàm mà hàm này gọi đến), bởi khi gọi setState() thì hàm render sẽ được gọi gây ra lặp vô hạn.
* componentDidMount()
  + Ngay sau khi hàm render được gọi đến lần đầu tiên chạy xong thì hàm này sẽ được chạy.
  + Thường dùng để fetch dữ liệu từ server và sau đó setState để render dữ liệu ra.
  + Đến đây thì các phần tử đã được sinh ra rồi, và có thể tương tác với DOM bằng JS trong hàm này.
* componentWillReceiveProps(nextProps)
  + Hàm này được chạy khi mà props của component đã được sinh ra có sự thay đổi.
  + Phải gọi setState() nếu muốn render lại.
* shouldComponentUpdate(nextProps, nextState)
  + Được gọi trước render.
  + Trả về true / false. Nếu false thì sẽ không render lại. Mặc định là true.
* componentWillUpdate(nextProps, nextState)
  + Được gọi ngay sau shouldComponentUpdate() nếu hàm này trả về true.
  + Không gọi setState() trong hàm này bởi hàm này là để chuẩn bị update cho đối tượng chứ không phải tạo ra 1 update mới, sẽ tạo ra lặp vô hạn.
  + Hàm render sẽ được gọi ngay sau hàm này.
* componentDidUpdate(prevProps, prevState)
  + Được gọi ngay sau render() từ lần render thứ 2 trở đi.
  + Đây cũng là 1 cơ hội để thao tác với các phần tử DOM bằng JS.
* componentWillUnmount()
  + Được gọi khi 1 component được loại bỏ khỏi DOM.
  + Thực hiện các thao tác dọn dẹp như huỷ các timer, loại bỏ các phần tử thừa.

Virtual DOM không được tạo ra bởi Reactjs nhưng lại được sử dụng rất nhiều. Đây là một chuẩn của W3C được dùng để truy xuất code HTML hoặc XML. Các Virtual DOM sẽ được tạo ra khi chạy chương trình, đó là nơi chưa các component. Sử dụng DOM sẽ tiết kiệm được hiệu suất làm việc, khi có thay đổi gì Reactjs đều tính toán trước và việc còn lại chỉ là thực hiện chúng lên DOM.

Lợi ích của việc sử dụng React

* ReactJS giúp cho việc viết các đoạn code Javascript sẽ trở nên dễ dàng hơn vì nó sử dụng một cú pháp đặc biệt đó chính là cú pháp JSX. Thông qua JSX cho phép nhúng code HTML vào Javascript.
* ReactJS cho phép Developer phá vỡ những cấu tạo UI phức tạp thành những component độc lập. Dev sẽ không phải lo lắng về tổng thể ứng dụng web, giờ đây Developer dễ dàng chia nhỏ các cấu trúc UI/UX phức tạp thành từng component đơn giản hơn.
* Đi kèm với ReactJS là rất nhiều các công cụ phát triển giúp cho việc debug code một cách dễ dàng hơn.
* Một trong những ưu điểm nữa của ReactJS đó là sự thân thiện với SEO (NextJS). Hầu như các JS Frameworks không thân thiện với các tìm kiếm mặc dù đã được cải thiện nhiều nhưng dưới sự hỗ trợ của các render dữ liệu trả về dưới dạng web page giúp cho SEO chuẩn hơn.

### ***2.1.3 Các thư viện sử dụng để xây dựng giao diện?***

React chỉ đơn thuần là một thư viện để xây dựng giao diện dễ dàng và hiệu quả hơn, vì thế để ứng dụng web trở lên ưu việt hơn, chúng ta cần phải cài đặt thêm một số gói hỗ trợ đi kèm, tùy theo mục đích và chức năng mà có những lựa chọn khác nhau.

Trong đề tài này, có sử dụng thêm một số thư viện hỗ trợ đi kèm, nhằm tăng hiệu suất ứng dụng như:

* Redux Toolkit: Thư viện quản lý trạng thái của ứng dụng.
* ANT Design: Thư viện giúp xây dựng nhanh các component.
* Axios: Thư viện giúp xử lý các yêu cầu tới máy chủ HTTP.
* StyledComponents: Thư viện giúp viết CSS trong JS.
* React Router DOM: Thư viện hỗ trợ quản lý các đường dẫn truy cập của ứng dụng.
* Redux Saga: Thư viện quản lý trạng thái bất động bộ của ứng dụng.
* Redux Persist: Thư viện lưu trữ trạng thái xuống bộ nhớ trình duyệt web.

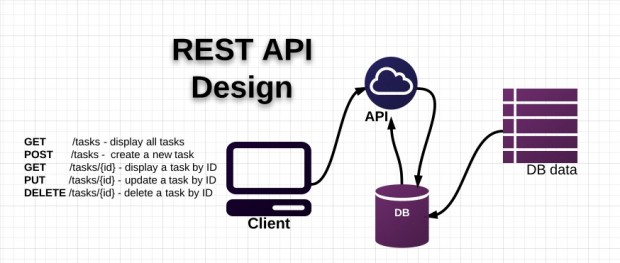
## **2.2 Tổng quan về Restful API**

### ***2.2.1 Restful API là gì?***

API (Application Programming Interface) là một tập các quy tắc và cơ chế mà theo đó, một ứng dụng hay một thành phần sẽ tương tác với một ứng dụng hay thành phần khác. API có thể trả về dữ liệu mà bạn cần cho ứng dụng của mình ở những kiểu dữ liệu phổ biến như JSON hay XML.

REST (REpresentational State Transfer) là một dạng chuyển đổi cấu trúc dữ liệu, một kiểu kiến trúc để viết API. Nó sử dụng phương thức HTTP đơn giản để tạo cho giao tiếp giữa các máy. Vì vậy, thay vì sử dụng một URL cho việc xử lý một số thông tin người dùng, REST gửi một yêu cầu HTTP như GET, POST, DELETE, vv đến một URL để xử lý dữ liệu.

Có thể nói bản thân REST không phải là một loại công nghệ. Nó là phương thức tạo API với nguyên lý tổ chức nhất định. Những nguyên lý này nhằm hướng dẫn lập trình viên tạo môi trường xử lý API request được toàn diện.



Hình 2. 3 Minh họa về REST API

RESTful API là một tiêu chuẩn dùng trong việc thiết kế API cho các ứng dụng web (thiết kế Web services) để tiện cho việc quản lý các resource. Nó chú trọng vào tài nguyên hệ thống (tệp văn bản, ảnh, âm thanh, video, hoặc dữ liệu động…), bao gồm các trạng thái tài nguyên được định dạng và được truyền tải qua HTTP. RESTful là một trong những kiểu thiết kế API được sử dụng phổ biến ngày nay để cho các ứng dụng (web, mobile…) khác nhau giao tiếp với nhau. Chức năng quan trọng nhất của REST là quy định cách sử dụng các HTTP method (như GET, POST, PUT, DELETE…) và cách định dạng các URL cho ứng dụng web để quản các resource. RESTful không quy định logic code ứng dụng và không giới hạn bởi ngôn ngữ lập trình ứng dụng, bất kỳ ngôn ngữ hoặc framework nào cũng có thể sử dụng để thiết kế một RESTful API.

Resource có nghĩa là tài nguyên, nhưng đây là một khái niệm được nhắc đến nhiều trong RESTful. Resource chính là dữ liệu mà chúng ta phải quản lý, có thể là customers, products, posts, images, videos… Mặt khác, tên resource cũng sẽ xuất hiện trong cách viết endpoint, nên nếu đặt tên cho resource một cách khoa học, thì endpoint cũng trở nên dễ hiểu và dễ tiếp cận hơn.

|  |  |
| --- | --- |
| **Endpoint** | **Resource** |
| http://api.example.com/users | users |
| http://api.example.com/users/1/accounts | accounts |
| http://api.example.com/users/1/images | images |

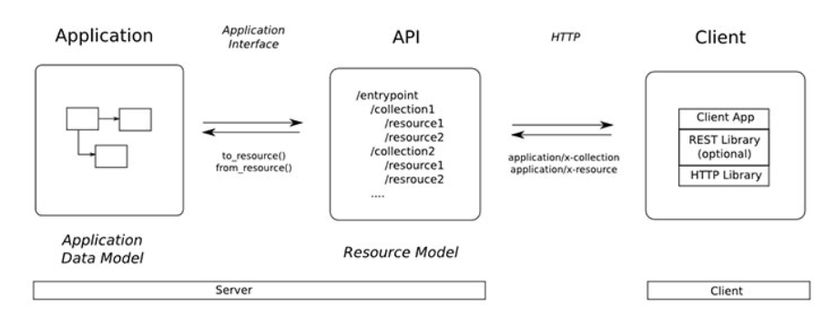
Bảng 2. 1 Minh họa về cách viết enpoint trong Restful API

### **2.2.2 Cách thức hoạt động của Resfull API**

REST hoạt động chủ yếu dựa vào giao thức HTTP. Các hoạt động cơ bản nêu trên sẽ sử dụng những phương thức HTTP riêng.

* GET (SELECT): Trả về một Resource hoặc một danh sách Resource.
* POST (CREATE): Tạo mới một Resource.
* PUT (UPDATE): Cập nhật thông tin cho Resource.
* DELETE (DELETE): Xoá một Resource.
* PATCH: ghi đè các thông tin được thay đổi của đối tượng.
* CONNECT: thiết lập một kết nối tới server theo URI.
* OPTIONS: mô tả các tùy chọn giao tiếp cho resource.
* TRACE: thực hiện một bài test loop – back theo đường dẫn đến resource

Những phương thức hay hoạt động này thường được gọi là CRUD tương ứng với Create, Read, Update, Delete – Tạo, Đọc, Sửa, Xóa.



Hình 2. 4 Cách thức hoạt động của RestfulAPI

Hiện nay đa số viết RESTful API đều chọn JSON là format chính thức nhưng cũng có nhiều người chọn XML làm format, nói chung dùng thế nào cũng được miễn tiện và nhanh.

Khi thiết api cho app IOS hay Client Side, chúng ta nên đặt version cho các api. Ví dụ như endpoint sau: api/v1/users.

Điều này sẽ giúp hệ thống sau khi nâng cấp lên version mới vẫn hộ trợ các api của version cũ, cũng như giúp việc bảo trì, sửa chữa dễ dàng hơn.

### **2.2.3 Tại sao lại là Restfull API?**

Một số ưu điểm chính khi sử dụng RESTFUL API là:

* Hiệu năng: các thành phần đảm bảo được việc giao tiếp theo đúng một quy ước giúp hệ thống có thể vận hành tốt hơn.
* Tính khả biến: với các hệ thống cần thay đổi các tài nguyên liên tục, sử dụng REST với việc tạo request đơn giản sẽ giúp mọi chuyển trở nên đơn giản hơn.
* Tính mở rộng: các hệ thống REST có khả năng mở rộng rất cao nhờ sự tách biệt giữa các thành phần và các quy ước giao tiếp được quy định sẵn.
* Tính linh hoạt: việc chuẩn hoá interface giúp hệ thống trở nên linh hoạt hơn, có thể sử dụng cho cho nhiều nền tảng khác nhau, mobile, web,...
* Trong sáng: trong giao tiếp giữa các thành phần, các request trở nên rất rõ ràng, dễ hiểu.
* Đơn giản: xây dựng rất đơn giản, ví dụ như việc khai báo resource trong node đã giúp xây dựng các uri cho resource.
* Tính tin cậy: khó để xảy ra lỗi trong giao tiếp giữa các thành phần gây sụp đổ hệ thống.

Những trang web ngày nay thường sử dụng REST API để cho phép kết nối đến dữ liệu của họ. Trong đó, facebook cũng cung cấp các REST API để giúp các ứng dụng bên ngoài kết nối đến dữ liệu của họ

Tất cả mọi tài nguyên có trong REST thực chất chính là thực thể, nó có thể hoàn toàn độc lập như:

* GET/users: có thể lấy danh sách của các người dùng.
* GET/users/123: sẽ lấy các thông tin của một người dùng có id=123
* GET/posts: Thường lấy tất cả mọi bài post.

Thường thì chúng cũng có thể sẽ lấy những thực thể hoàn toàn độc lập vào các thực thể khác như:

* GET/users/123/projects: Sẽ lấy toàn bộ các projects của user với id=123.

Dựa vào ví dụ trên nên chú ý thì ta có thể thấy rằng: GET sẽ lấy toàn bộ thông tin về thực thể mà client side đã thực hiện request. Mỗi một request thành công đều sẽ được trả về dữ liệu liên quan đến thực thể cũng như kèm theo status code là 200. Nếu như có lỗi, ta có thể sẽ nhận lại được status code 404, 400 hoặc 5xx.

Với phương thức POST đó là:

* POST/users.

Khi tạo một thực thể mới thì bạn sẽ truyền được dữ liệu vào trong request body ví dụ như sau:

{

“fist\_day”: “Vasyl”,

“last\_day”:Redke”

}

Sau khi bạn gửi request lên với server thì sẽ nhận được kết quả trả về có thể là: status code 201, hàm ý tạo mới cho thành công.

Với phương thức PUT, được sử dụng để có thể cập nhật các thực thể khi bạn đã gửi request thì body nhất thiết cũng cần phải gồm dữ liệu cần phải được cập nhật liên quan tới thực thể.

Những sự thay đổi này cần phải được chỉ ra là cập nhật cho vào các thực thể, nó sẽ được truyền vào thông qua các tham số nhất định. Nếu như được cập nhật thành công thì mã 200 sẽ được trả về và dữ liệu của thực thể sẽ vừa được cập nhật.

Với phương thức Delete, nó thường khá dễ hiểu và được sử dụng để xóa được một thực thể cụ thể đã được chỉ định sẵn thông qua tham số.

* DELETE/users/123\_xóa một user với id=123

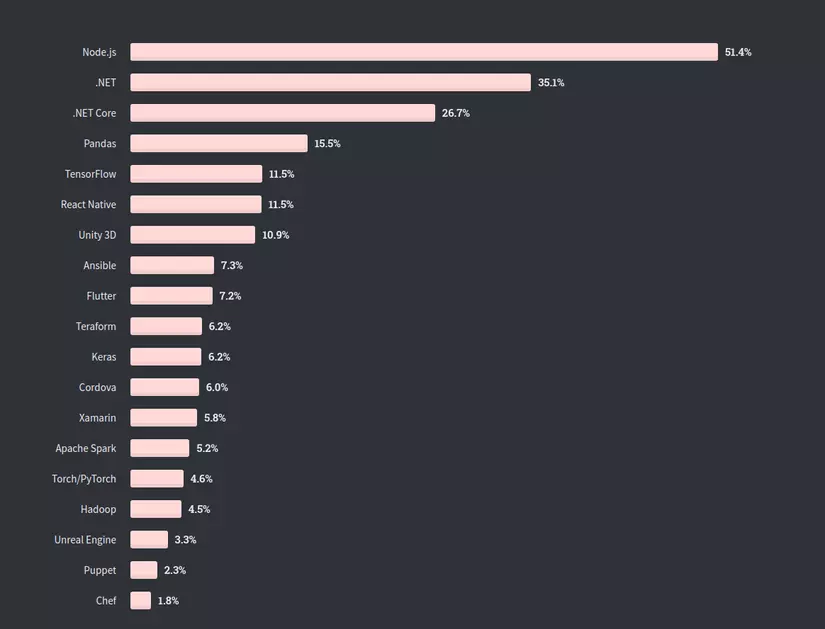
Nếu như đã xóa thành công thì thực thể được trả về status 200 cùng với response sẽ bao gồm các thông tin về trạng thái của chính thực thể.

Hoặc nếu như xóa đi thực thể có trong database luôn thì status code cho request thứ 2 sẽ được trả về là 404, bởi vì đây là thực thể đã được xóa nên không thể nào truy cập được nữa.

## **2.3 Tổng quan về NodeJS/ExpressJS**

### **2.3.1 NodeJS là gì?**

NodeJS là một nền tảng được xây dựng trên V8 JavaScript Engine – trình thông dịch thực thi mã JavaScript, giúp xây dựng các ứng dụng web một cách đơn giản và dễ dàng mở rộng. NodeJS được phát triển bởi Ryan Dahl vào năm 2009 và có thể chạy trên nhiều hệ điều hành khác nhau: OS X, Microsoft Windows, Linux. Giờ đây bạn có thể làm được nhiều thứ với JavaScript hơn là chỉ tương tác với các website.



Hình 2. 5 Thống kê các ngôn ngữ được sử dụng nhiều nhất

Cả trình duyệt JavaScript và Node.js đều chạy trên JavaScript runtime V8 engine. Công cụ này lấy code JavaScript của bạn và convert nó sang mã máy (bytecode) cho việc thực thi nhanh hơn. Mã máy là loại code thấp cấp hơn để máy tính có thể chạy mà không cần biên dịch nó.

Phần core sử dụng Javascript và C++ cho phép xử lý với hiệu năng cao. Phù hợp với các ứng dụng xử lý nhanh, real time hoặc những ứng dụng cần thay đổi công nghệ nhanh. Chạy sigle thread nhưng có cơ chế non blocking giúp chạy bất đồng bộ, tăng khả năng xử lý. Thường phát triển theo hướng module hóa thành các phần nhỏ của ứng dụng và có thể thêm module có sẵn bên ngoài thông qua NPM, yarn. JavaScript suy cho cùng cũng chỉ là một ngôn ngữ - vậy thì không có lý do gì để nói nó không thể sử dụng trên môi trường server tốt như là trong trình duyệt của người dùng được.

Để giải quyết vấn đề kết nối đồng thời cao, Nodejs đã từ bỏ ý tưởng "tạo một luồng mới cho mỗi kết nối người dùng" và thay đổi nó thành mỗi người dùng kết nối với Nodejs và kích hoạt một sự kiện. Điều này giải quyết vấn đề nhiều kết nối người dùng hết bộ nhớ máy chủ.

Như đã đề cập trước đó, Nodejs sử dụng JavaScript engine V8 ở phía máy chủ để đạt được một máy chủ hiệu suất cao. JavaScript là một ngôn ngữ đơn luồng và V8 JavaScript cũng vậy. Khác với JavaScript phía trình duyệt, V8 JavaScript cung cấp cơ chế Nonblocking I/O. Điều đó nghĩa là gì? Nó thực sự là một yêu cầu web JavaScript không đồng bộ.

Trong khi chờ đợi kết quả của yêu cầu, nó sẽ tiếp tục thực thi đoạn mã đằng sau yêu cầu mà không cần đợi dữ liệu trả về, và kết quả mà yêu cầu trả về được xử lý trong hàm gọi lại. Cơ chế này được gọi là cơ chế Nonblocking I/O. Vì các tác vụ tốn thời gian sẽ không chặn mã tiếp theo, nên nó cải thiện đáng kể hiệu quả thực thi. Tất nhiên, yêu cầu mạng được lấy làm ví dụ ở đây, nhưng trên thực tế, sự kiện có thể là: thêm tệp, xóa, kiểm tra, sửa đổi.

### **2.3.2 Tại sao lại là NodeJS?**

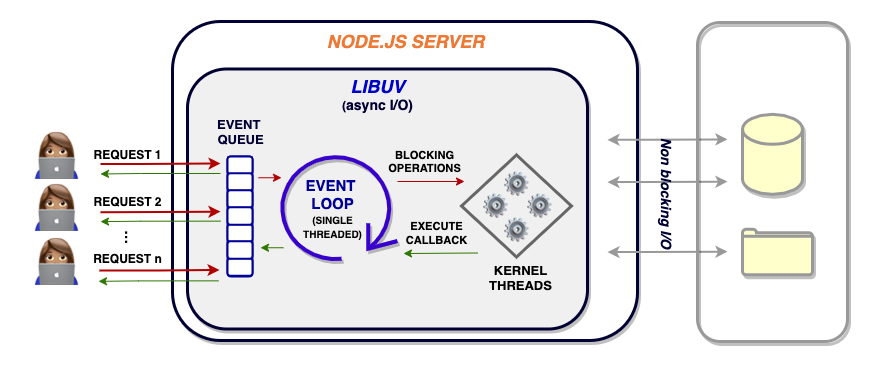
Trong một môi trường server điển hình LAMP (Linux-Apache-MySQL-PHP), bạn có một web server là Apache hoặc NGINX nằm dưới, cùng với PHP chạy trên nó. Mỗi một kết nối tới server sẽ sinh ra một thread mới, và điều này khiến ứng dụng nhanh chóng trở nên chậm chạp hoặc quá tải - cách duy nhất để hỗ trợ nhiều người dùng hơn là bằng cách bổ sung thêm nhiều máy chủ. Đơn giản là nó không có khả năng mở rộng tốt. Nhưng với Node.js thì điều này không phải là vấn đề. Không có một máy chủ Apache lắng nghe các kết nối tới và trả về mã trạng thái HTTP - bạn sẽ phải tự quản lý kiến trúc lõi của máy chủ đó. May mắn thay, có một số module giúp thực hiện điều này được dễ dàng hơn, nhưng công việc này vẫn gây cho bạn một chút khó khăn khi mới bắt đầu. Tuy nhiên, kết quả thu được là một ứng dụng web có tốc độ thực thi cao.

Một khái niệm cốt lõi của Node.js đó là các function bất đồng bộ (asynchronous functions) - vì vậy về cơ bản thì mọi thứ chạy trên nền tảng này. Với hầu hết các ngôn ngữ kịch bản máy chủ, chương trình phải đợi mỗi function thực thi xong trước khi có thể tiếp tục chạy tiếp. Với Node.js, bạn xác định các function sẽ chạy để hoàn thành một tác vụ nào đó, trong khi phần còn lại của ứng dụng vẫn chạy đồng thời.

Tốc độ thực thi và khả năng mở rộng. Node.js có tốc độ rất nhanh. Đó là một yêu cầu khá quan trọng khi cố gắng tạo ra một sản phẩm lớn và muốn đảm bảo có thể mở rộng nhanh chóng, đáp ứng được một lượng lớn người dùng khi trang web của phát triển lên. Node.js có thể xử lý hàng ngàn kết nối đồng thời trong khi PHP sẽ chỉ có nước sụp đổ.

Cơ chế vòng lặp sự kiện trong Nodejs rất giống với cơ chế vòng lặp sự kiện trong JavaScript nhưng có sự khác biệt. Có thêm hai phương thức không đồng bộ nữa là setInstant và process.nextTick.

### **2.3.3 Cách thức hoạt động của NodeJS**



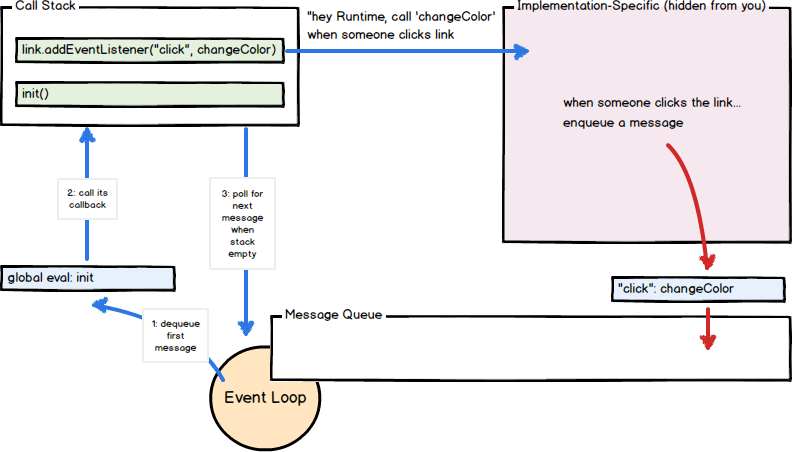
Hình 2. 6 Mô tả kiến trúc của NodeJS

Trong Javascript, hầu hết các lời gọi I/O đều là non-blocking. Nghĩa là khi có HTTP request, truy xuất dữ liệu trong DB hoặc đọc ghi vào bộ nhớ thì hệ thống sẽ không tạm dừng (blocking) các đoạn code tiếp theo (như các ngôn ngữ server khác PHP, Ryby,...) mà sẽ trao quyền thực thi những lời gọi I/O này cho hệ thống và thực thi những đoạn code tiếp theo, khi hệ thống đã thực thi xong những lời gọi hệ thống này thì hàm callback truyền vào sẽ tự động được gọi.

Nodejs giúp code của bạn chạy những đoạn còn lại trong khi vẫn chờ response trả về để thực thi hàm callback truyền vào. Thế nhưng câu hỏi đặt ra là hàm callback này được lưu trữ ở đâu, chúng được thực thi theo thứ tự nào khi có nhiều hàm callback như thế, và cái gì gọi chúng thực thi.

Trình khởi chạy của javascript có một hàng đợi (queue) chứa các messages, các messages này được gắn liền với các hàm callback truyền vào. Mỗi khi gặp câu lệnh có callback truyền vào thì message gắn với callback đó sẽ được đẩy vào queue, và khi các event được trigger( ví dụ như event click, event request có response trả về) thì hệ thống sẽ gọi hàm callback tương ứng để thực thi. Khi một event ví dụ như click vào button nhưng không truyền vào callback thì sẽ không có message nào được đẩy vào queue, nghĩa là chỉ những event có callback tryền vào thì mới được đẩy vào queue.

Hình trên mô tả hoạt động của Event Loop. Khi có trigger(sự kiện click hoặc request ở vd trên thực hiện xong và có response trả về) thì hàm callback truyền vào sẽ được đẩy vào Massage Queue.



Hình 2. 7 Minh họa cơ chế Event Loop

Khi biên dịch đoạn code ban đầu thì các dòng lệnh async của bạn sẽ được đẩy vào stack, khi các dòng lệnh này chạy từ đầu tới cuối (chúng chạy song song với các tiến trình truy cập I/O nói ở trên) và khi chạy xong câu lệnh cuối cùng rồi thì stack sẽ về rỗng. Một khi stack về rỗng (chạy xong thân code của bạn), vòng lặp Event Loop sẽ được khởi chạy. Mỗi khi Event Loop gặp một message trong Message Queue, nó sẽ thực thi hàm callback gắn với message đó bằng cách đẩy các đoạn code trong hàm callback vào stack. Sau khi hàm callback đó thực hiện xong, stack về rỗng, thì Event Loop tiếp tục chạy và lấy message tiếp theo (nếu có) ra và đẩy code của callback vào stack thực thi.

### **2.3.4 ExpressJS là gì?**

ExpressJS là một web framework được xây dựng trên nền tảng NodeJs, là một web application framework cho NodeJS, nó cung cấp cho chúng ta rất nhiều tính năng mạnh mẽ trên nền tảng web.

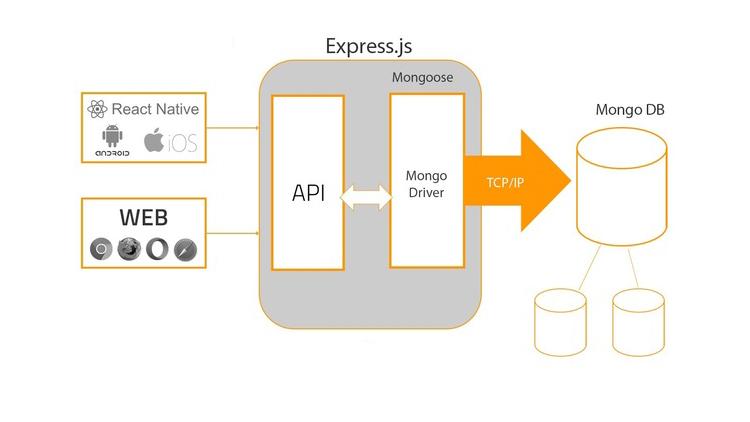
ExpressJS cung cấp các hàm HTTP và midleware để tạo ra API đơn giản và dễ sử dụng. ExpressJS được phát hành theo giấy phép mã nguồn mở, có cộng đồng hỗ trợ lớn, được phép sử dụng cho ứng dụng có mục đích thương mại. Do vậy bạn hoàn toàn có thể sử dụng framework này cho các dự án, từ dự án nhỏ tới lớn.

Expressjs có vô vàn các package hỗ trợ, vì thế bạn không cần phải lo lắng khi làm việc với các Framework này. Expressjs còn cung cấp thêm những tính năng để các lập trình viên lập trình được tốt hơn. Chứ nó không hề làm giảm tốc độ của NodeJS. Các Framework nổi tiếng của NodeJS hiện nay đều đang sử dụng ExpressJS như một core function.

Expressjs được thành lập bởi TJ Holowaychuk, và được phát hành đầu tiên vào ngày 22 tháng 05 năm 2010 với phiên bản 0.12, theo kho lưu trữ GitHub. Tháng 6 năm 2014, quyền quản lý dự án được StrongLoop mua lại. StrongLoop đã được IBM mua lại vào tháng 9 năm 2015. Tháng 1 năm 2016 Expressjs lại được Nodejs Foundation quản lý.

### **2.3.5 Tại sao lại là ExpressJS?**

Express hướng tới sự linh động, không giống như các đối thủ như là Rails hay Django, hướng đến một cách cứng nhắc để có thể xây dựng ứng dụng. Giờ đây việc xây dựng một ứng dụng web trở lên rất dễ dàng. Express sử dụng kiến trúc xây dựng phần mề là MVC vì thế mọi thứ bên trong ứng dụng đều rất rõ ràng và dễ hiểu thuận lợi cho việc debug ứng dụng.



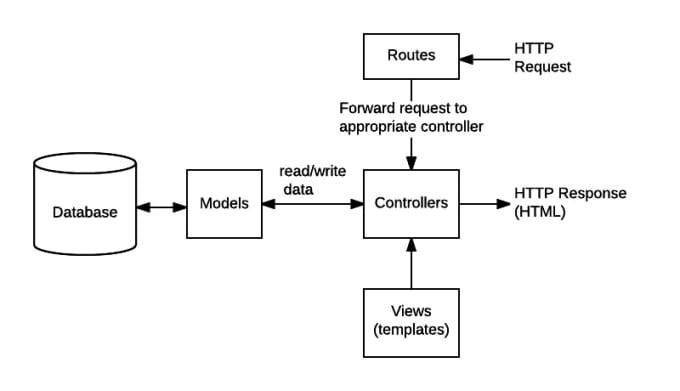
Hình 2. 8 Minh họa mô hình ứng dụng Express

Bằng cách mở rộng những đặc tính của Nodejs, Expressjs cung cấp đến những tính năng như middlewares – hàm xử lý trung gian. Đơn giản hóa khả năng điều hướng của ứng dụng, kèm theo tích hợp hỗ trợ nhiều template engine, có thể giúp bạn tương ứng dễ dàng hơn với HTML.

Express cung cấp các API về Router cho phép lập trình và điều hướng ứng dụng rất linh hoạt, khi xây dựng nó chúng ta có thể dễ dàng cung cấp các tài nguyên phù hợp cho từng yêu cầu được gửi lên từ phía người dùng. Ngoài ra các middleware mà express hỗ trợ giúp cho việc xử lý các yêu cầu tới từ người dùng cần phải xác thực trước khi tới máy chủ rất nhanh chóng. Bên cạnh đó còn rất nhiều tính năng hay ho khác mà framework này hỗ trợ lập trình viên.

### **2.3.6 Cách thức hoạt động của ExpressJS**

Cấu trúc hoạt động của một dư án Backend sử dụng framework Express



Hình 2. 9 Cấu trúc cơ bản của một dự án Express JS

Các yêu cầu HTTP sẽ được gửi tới từ người dùng thông qua trình duyệt, tại đây ứng dụng của chúng ta sẽ thu nhận được các yêu cầu đó qua các Route tương ứng. Sau khi đã truy cập đúng vào các route thích hợp, chúng sẽ được chuyển tới tầng Controller, tại đây ứng dụng Express sẽ trực tiếp xử lý các yêu cầu logic. Nếu yêu cầu cần thiết phải truy vấn về cơ sở dữ liệu Controller sẽ gọi đến các Model thích hợp để xử lý và nhận về dữ liệu được truy vấn từ phía cơ sở dữ liệu, lúc này sau khi đã hoàn tất quá trình xử lý về mặt logic, Controller sẽ chuyển kết quả về lại cho người dùng những thông tin tương ứng mà họ đã gửi yêu cầu lên ứng dụng.

Bằng cách mở rộng các đặc tính của NodeJs, ExpressJs cung cấp các tính năng như middlewares (hàm xử xý trung gian), đơn giản hóa khả năng điều hướng của ứng dụng, kèm theo tích hợp hỗ trợ rất nhiều template engine, giúp bạn tương tác dễ dàng hơn với HTML. Ngoài ra Express còn rất thích hợp cho các ứng dụng hoạt động theo kiểu Máy Chủ và Máy Khách (Client – Server) bằng việc Server cung cấp API và Client sẽ trực tiếp sử dụng chúng .

## **2.4 Tổng quan về MongoDB**

### **2.4.1 MongoDB là gì?**

**Khái niệm về NoSQL**

* NoSQL là 1 dạng CSDL mã nguồn mở và được viết tắt bởi: None-Relational SQL hay có nơi thường gọi là Not-Only SQL.
* NoSQL được phát triển trên Javascript Framework với kiểu dữ liệu là JSON và dạng dữ liệu theo kiểu key và value.
* NoSQL ra đời như là 1 mảnh vá cho những khuyết điểm và thiếu xót cũng như hạn chế của mô hình dữ liệu quan hệ RDBMS (Relational Database Management System - Hệ quản trị cơ sở dữ liệu quan hệ) về tốc độ, tính năng, khả năng mở rộng,...
* Với NoSQL bạn có thể mở rộng dữ liệu mà không lo tới những việc như tạo khóa ngoại, khóa chính, kiểm tra ràng buộc .v.v ...
* NoSQL bỏ qua tính toàn vẹn của dữ liệu và transaction để đổi lấy hiệu suất nhanh và khả năng mở rộng.
* NoSQL được sử dụng ở rất nhiều công ty, tập đoàn lớn, ví dụ như FaceBook sử dụng Cassandra do FaceBook phát triển, Google phát triển và sử dụng BigTable,...

**Khái niệm về MongoDB**

* MongoDB là một hệ quản trị cơ sở dữ liệu mã nguồn mở, là CSDL thuộc NoSql và được hàng triệu người sử dụng.
* MongoDB là một database hướng tài liệu (document), các dữ liệu được lưu trữ trong document kiểu JSON thay vì dạng bảng như CSDL quan hệ nên truy vấn sẽ rất nhanh.
* Với CSDL quan hệ chúng ta có khái niệm bảng, các cơ sở dữ liệu quan hệ (như MySQL hay SQL Server...) sử dụng các bảng để lưu dữ liệu thì với MongoDB chúng ta sẽ dùng khái niệm là collection thay vì bảng
* So với RDBMS thì trong MongoDB collection ứng với table, còn document sẽ ứng với row , MongoDB sẽ dùng các document thay cho row trong RDBMS.
* Các collection trong MongoDB được cấu trúc rất linh hoạt, cho phép các dữ liệu lưu trữ không cần tuân theo một cấu trúc nhất định.
* Thông tin liên quan được lưu trữ cùng nhau để truy cập truy vấn nhanh thông qua ngôn ngữ truy vấn MongoDB

Các trường hợp nên sử dụng MongoDB, ví dụ như các hệ thống realtime (thời gian thực) yêu cầu phản hồi nhanh, Các hệ thống bigdata với yêu cầu truy vấn nhanh hay các hệ thống có lượng request lớn thì MongoDB sẽ là sự lựa chọn ưu tiên hơn CSDL quan hệ. Tùy theo dự án và trường hợp cụ thể để sử dụng CSDL quan hệ hay sử dụng MongoDB đem lại hiệu quả cao.

Một ưu điểm của MongoDB cũng chính là nhược điểm của nó. MongoDB không có các tính chất ràng buộc như trong RDBMS nên khi thao tác với mongoDB thì phải hết sức cẩn thận.

Tốn bộ nhớ do dữ liệu lưu dưới dạng key-value, các collection chỉ khác về value do đó key sẽ bị lặp lại. Không hỗ trợ Join nên dễ bị dữ thừa dữ liệu.

Khi insert/update/remove bản ghi, MongoDB sẽ chưa cập nhật ngay xuống ổ cứng, mà sau 60 giây MongoDB mới thực hiện ghi toàn bộ dữ liệu thay đổi từ RAM xuống ổ cứng điêù này sẽ là nhược điểm vì sẽ có nguy cơ bị mất dữ liệu khi xảy ra các tình huống như mất điện...

### **2.4.2 Tại sao lại là MongoDB**

Do MongoDB sử dụng lưu trữ dữ liệu dưới dạng Document JSON nên mỗi một collection sẽ có các kích cỡ và các document khác nhau, linh hoạt trong việc lưu trữ dữ liệu, nên bạn muốn gì thì cứ insert vào thoải mái.

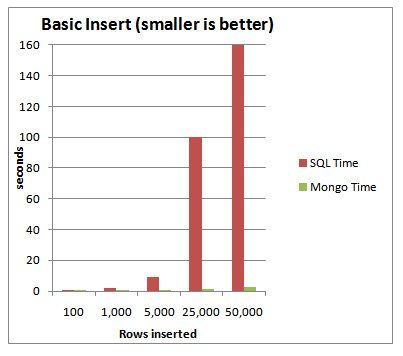
Dữ liệu trong MongoDB không có sự ràng buộc lẫn nhau, không có join như trong RDBMS nên khi insert, xóa hay update nó không cần phải mất thời gian kiểm tra xem có thỏa mãn các ràng buộc dữ liệu như trong RDBMS.

MongoDB rất dễ mở rộng (Horizontal Scalability). Trong MongoDB có một khái niệm cluster là cụm các node chứa dữ liệu giao tiếp với nhau, khi muốn mở rộng hệ thống ta chỉ cần thêm một node với vào cluster:

Trường dữ liệu “\_id” luôn được tự động đánh index (chỉ mục) để tốc độ truy vấn thông tin đạt hiệu suất cao nhất.

Khi có một truy vấn dữ liệu, bản ghi được cached lên bộ nhớ Ram, để phục vụ lượt truy vấn sau diễn ra nhanh hơn mà không cần phải đọc từ ổ cứng.

Hiệu năng cao: Tốc độ truy vấn (find, update, insert, delete) của MongoDB nhanh hơn hẳn so với các hệ quản trị cơ sở dữ liệu quan hệ (RDBMS). Với một lượng dữ liệu đủ lớn thì thử nghiệm cho thấy tốc độ insert của MongoDB có thể nhanh tới gấp 100 lần so với MySQL



Hình 2. 10 So sanh tốc độ của MongoDB với MYSQL

# CHƯƠNG 3 TRIỂN KHAI XÂY DỰNG HỆ THỐNG

## **3.1 Mô tả hệ thống**

### **3.1.1 Thành phần chính của hệ thống**

Phía người dùng (Client): Giao diện tương tác người dùng với hệ thống

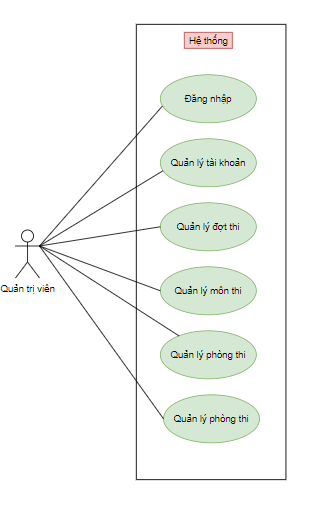
Phía máy chủ (Server): Xử lý các yêu cầu của người dùng khi tương tác với hệ thống thông qua các request API đã được định nghĩa.

### **3.1.2 Công nghệ sử dụng**

* Sử dụng React để xây dựng giao diện người dùng.
* Sử dụng Node và Express để xây dựng hệ thống Restful API.
* Sử dụng MongoDB để làm cơ sở lưu trữ dữ liệu.
* Sử dụng JWT để phân quyền truy cập tài nguyên hệ thống.

## **3.2 Thiết kế hệ thống**

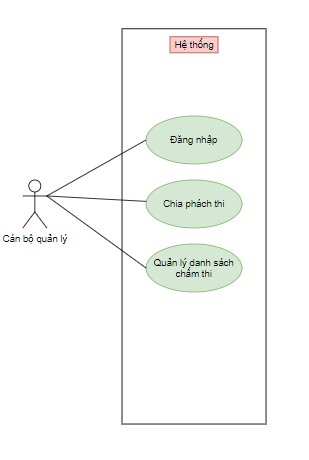
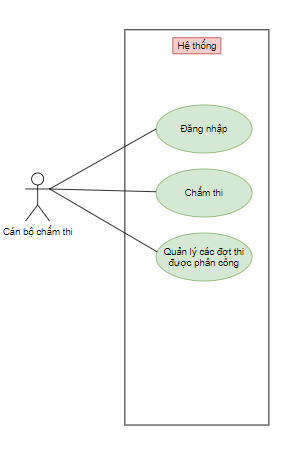
### ***3.2.1 Biểu đồ ca sử dụng***

****

Hình 3. 1 Biểu đồ ca sử dụng của quản trị viên

Quản trị viên có thể sử dụng các chức năng của hệ thống như:

* Đăng nhập
* Quản lý tài khỏan hệ thống
  + Chỉnh sửa thông tin
  + Hủy kích hoạt tài khoản
* Quản lý đợt thi
* Quản lý môn thi
* Quản ly phòng thi

** **

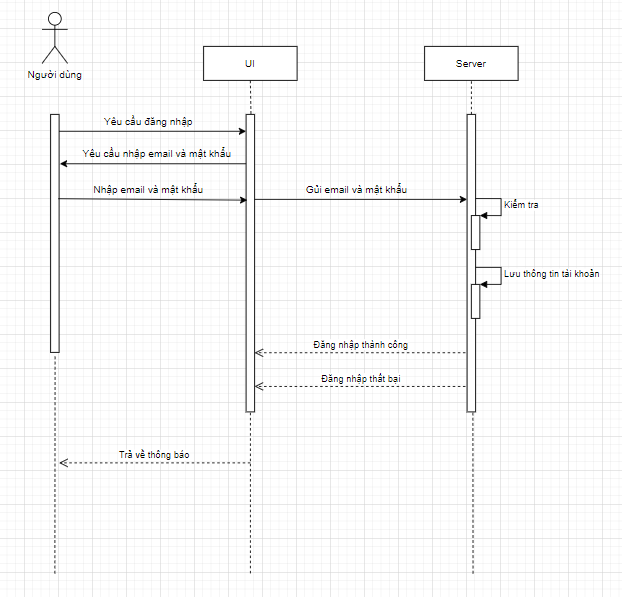
Hình 3. 2 Biểu đồ ca sử dụng của cán bộ quản và cán bộ chấm thi.

Cán bộ quản lý có thể sử dụng các chức năng như: đăng nhập, chia phách bài thi, quản lý danh sách các bài thi đã chia phía.

Cán bộ chấm thi có thể sử dụng các chức năng như: đăng nhập, chấm thi, quản lý các bài thi đã được phân công chấm.

### **3.2.2 Biểu đồ tuần tự**

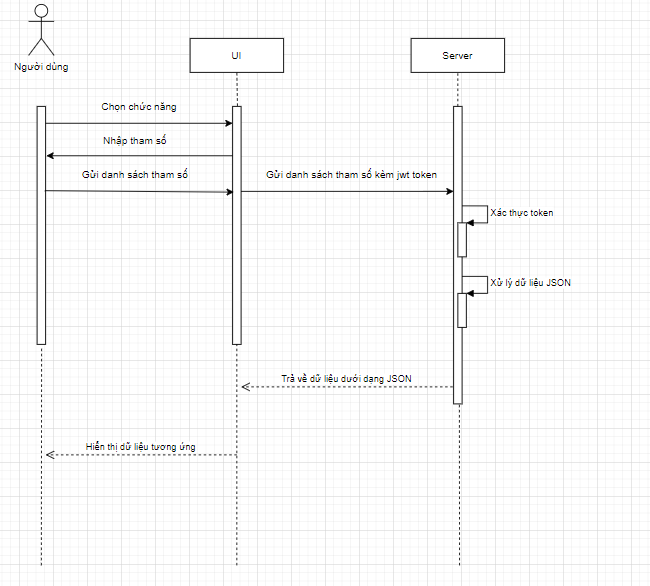
* Chức năng đăng nhập

****

Hình 3. 4 Biểu đô tuần tự chức năng đăng nhập

Người dùng trong hệ thống được cung cấp tài khoản để đăng nhập vào hệ thống. Mỗi khi người dùng đăng nhập, hệ thống sẽ kiểm tra tính hợp lệ của email và mật khẩu người dùng nhập vào, nếu thông tin tài khoản hợp lệ và chính xác, người dùng sẽ đăng nhập thành công, còn người lại nếu thông tin đăng nhập không chính xác hệ thống sẽ trả về thông báo đăng nhập thất bại.

* Chức năng nghiệp vụ hệ thống

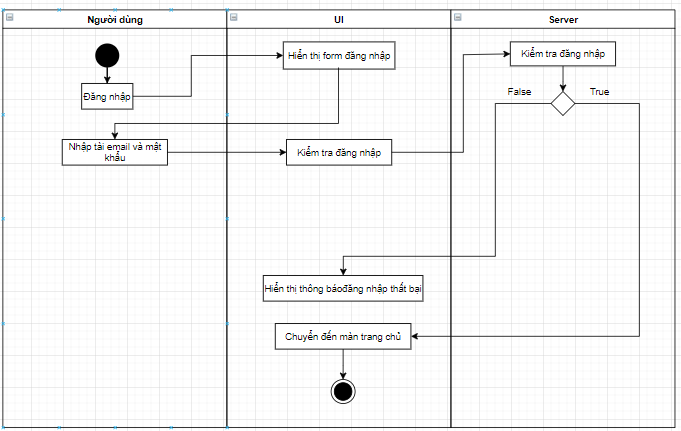
****

Hình 3. 5 Biểu đô tuần tự các chức năng nghiệp vụ

Sau khi đã đăng nhập thành công vào hệ thống, người dùng sẽ có thể sử dụng các tính năng mà hệ thống cung cấp tùy thuộc vào vai trò của tài khoản mà người dùng đăng nhập. Mỗi khi người dùng yêu cầu sử dụng tính năng, hệ thống sẽ gửi một request lên phía Server kèm theo đó là một đoạn mã token của người dùng đã được cấp khi đăng nhập thành công. Dựa vào từng request gửi lên, phía server sẽ biết được yêu cầu của người dùng và đoạn token gửi lên, nếu token hợp lệ và được phân quyền sử dụng, Server sẽ xác nhận yêu cầu sau đó xử lý và trả về dữ liệu về người dùng dưới dạng JSON. Từ đó có thể hiển thị lên trên giao diện ứng dụng web.

### **3.2.3 Biểu đồ hoạt động**

Chức năng đăng nhập



Hình 3. 6 Biểu đô tuần tự chức năng đăng nhập

Người dùng của hệ thống sẽ đăng nhập bằng email và mật khẩu đã được quản trị viên hệ thống cung cấp. Nếu đăng nhập thành công sẽ được chuyển hướng đến trang chủ của tài khoản đó, còn nếu thông tin đăng nhập sai sẽ nhận được thông báo hiển thị lỗi đăng nhập trên màn hình.

Người dùng đăng nhập vào hệ thống sẽ tùy thuộc vào vai trò của tài khỏa mà hệ thống sẽ hiển thị các chức năng khác nhau phù hợp.

### **3.2.3 Cơ sở dữ liệu**

Không giống như cơ sở dữ liệu có quan hệ RDBMS, dữ liệu được lưu dưới dạng các bảng và dữ liệu được lưu trong bảng là các hàng. Trong hệ cơ sở dữ liệu không quan hệ như MongoDB, dữ liệu được lưu dưới dạng các Collection chúng ta có thể hiểu một cách đơn giản nó tương tự với Table trong RDBMS và các dữ liệu trong các Collection được gọi là các Record hay một cách dễ hiểu đó chính là các hàng trong cơ sở dữ liệu RDBMS.

**Collection Bài Thi**

|  |  |  |
| --- | --- | --- |
| **Tên thuộc tính** | **Kiểu dữ liệu** | **Mô tả** |
| ID | ObjectId | Mã định danh của collection |
| OldName | String | Tên bài thi trước khi được chia phách |
| NewName | String | Tên bài thi sau khi được chia phách |
| Mark1 | Number | Điểm của giám thi chấm thứ nhất |
| Mark2 | Number | Điểm của giám thi chấm thứ hai |
| Url | String | Đường dẫn tới file chấm của bài thi |
| CreatedAt | Date | Ngày tạo bài thi |
| UpdatedAt | Date | Ngày chỉnh sửa |
| UserAssign | ObjectID | ID người phân công chấm bài |
| UserAssignMark1 | ObjectID | ID người chấm bài thi thứ nhất |
| UserAssignMark1 | ObjectID | ID người chấm bài thi thứ hai |
| Room | ObjectID | ID phòng thi |
| Subject | ObjectID | ID môn thi |
| ExamSession | ObjectID | ID đợt thi |

Bảng 3. 1 Bảng mô tả database collection bài thi

**Collection Người Dùng**

|  |  |  |
| --- | --- | --- |
| **Tên thuộc tính** | **Kiểu dữ liệu** | **Mô tả** |
| ID | ObjectId | Mã định danh của collection |
| Name | String | Tên người dùng |
| Email | String | Địa chỉ email |
| Password | String | Mật khẩu |
| RefeshToken | String | Token của người dùng khi đăng nhập |
| Role | Enum | Vai trò của tài khoản |
| Active | Boolean | Trạng thái hoạt động của tài khoản |
| CreatedAt | Date | Ngày tạo tài khoản |
| UpdatedAt | Date | Ngày cập nhật tài khoản |
| Subject | ObjectID | ID môn chấm thi (Cán bộ chấm thi) |
| Avatar | String | Đường dẫn danh đại diện |

Bảng 3. 2 Bảng mô tả database collection người dùng

**Collection Kỳ Thi**

|  |  |  |
| --- | --- | --- |
| **Tên thuộc tính** | **Kiểu dữ liệu** | **Mô tả** |
| ID | ObjectId | Mã định danh của collection |
| Name | String | Tên của đợt thi |

Bảng 3. 3 Bảng mô tả database collection kỳ thi

**Collection Ca Thi**

|  |  |  |
| --- | --- | --- |
| **Tên thuộc tính** | **Kiểu dữ liệu** | **Mô tả** |
| ID | ObjectId | Mã định danh của collection |
| Name | String | Tên của ca thi |
| SessionID | ObjectID | ID của kỳ thi |
| CreatedAt | Date | Ngày tạo |
| UpdatedAt | Date | Ngày cập nhật |
| Duration | Date | Ngày diễn ra |
| Times | Number | Ca thi số |
| Subject | ObjectID | ID của môn thi |
| Room | ObjectID | ID của phòng thi |

Bảng 3. 4 Bảng mô tả database collection ca thi

**Collection Phòng Thi**

|  |  |  |
| --- | --- | --- |
| **Tên thuộc tính** | **Kiểu dữ liệu** | **Mô tả** |
| ID | ObjectId | Mã định danh của collection |
| Name | String | Tên phòng thi |

Bảng 3. 5 Bảng mô tả database collection phòng thi

**Collection Môn Thi**

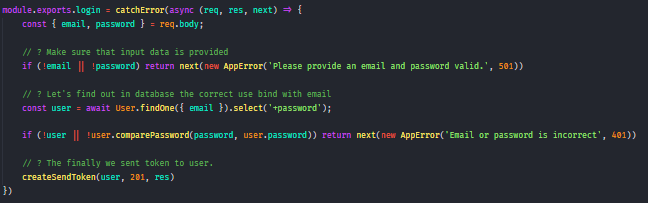
|  |  |  |
| --- | --- | --- |
| **Tên thuộc tính** | **Kiểu dữ liệu** | **Mô tả** |
| ID | ObjectId | Mã định danh của collection |
| Name | String | Tên phòng thi |

Bảng 3. 6 Bảng mô tả database collection môn thi

## **3.3 Xây dựng chức năng**

### **3.3.1 Chức năng đăng nhập**

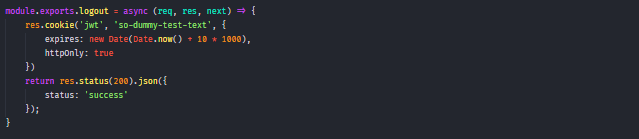
* Điều kiện tiên quyết: Không
* Kịch bản: Người dùng nhập email và mật khẩu vào form đăng nhập. Sau khi đăng nhập thành công với email và mật khẩu vừa nhập thì chuyển hướng người dùng đến trang chủ.
* Mã nguồn chức năng



Hình 3. 7 Chức năng đăng nhập

### **3.3.2 Chức năng đăng xuất**

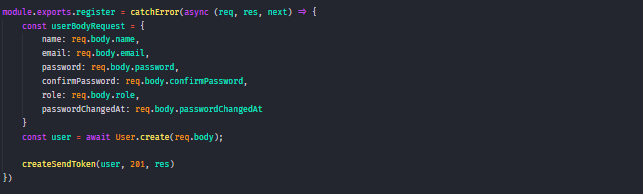
* Điều kiện tiên quyết: Người dùng đã đăng nhập
* Kịch bản: Người dùng nhấn vào Button đăng xuất trên giao diện của ứng dụng web, lúc này hệ thống sẽ xóa đi mã Token đã gửi cho người dùng lúc đăng nhập, sau đó người dùng sẽ được chuyển hướng đến trang đăng nhập.
* Mã nguồn chức năng



Hình 3. 8 Chức năng đăng xuất

### **3.3.3 Chức năng đăng ký**

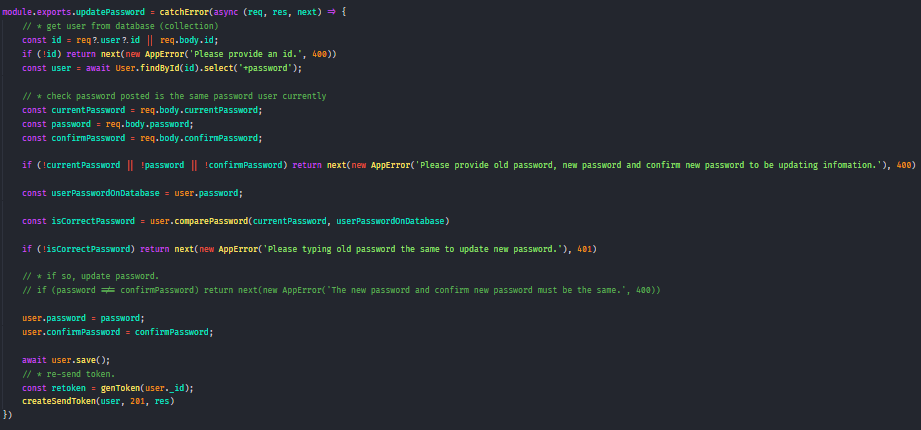
* Điều kiện tiên quyết: Người dùng đã đăng nhập với vai trò là quản trị viên.
* Kịch bản: Quản trị viên yêu cầu chức năng đăng ký tài khoản, sau khi điền đầy đủ thông tin vào form đăng ký và nhấn Button tạo tài khoản trên giao diện người dùng, một request sẽ được gửi lên server kèm theo mã Token định danh lúc đăng nhập thành công với vài trò quản trị viên. Lúc này hệ thống Server sẽ kiểm tra và xác thức mã Token, nếu mã Token hợp lệ thì yêu cầu của người dùng sẽ được xử lý và trả về dữ liệu dưới dạng JSON về cho người dùng với thông báo đăng ký thành công.
* Mã nguồn chức năng



Hình 3. 9 Chức năng đăng ký tài khoản cho quản trị viên

### **3.3.4 Chức năng thay đổi mật khẩu tài khoản**

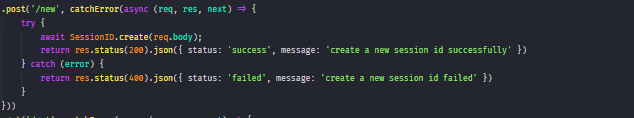
* Điều kiện tiên quyết: Người dùng đã đăng nhập
* Kịch bản: Người dùng nhấn vào Button chức năng thay đổi mật khẩu tại phần thông tin người dùng.
* Mã nguồn chức năng



Hình 3. 10 Chức năng thay đổi mật khẩu

### **3.3.5 Chức năng tạo mới kỳ thi**

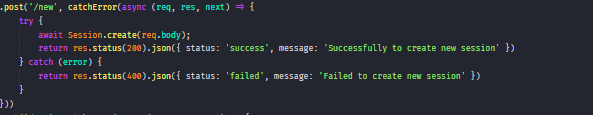
* Điều kiện tiên quyết: Người dùng đã đăng nhập với vai trò quản trị viên.
* Kịch bản: Người dùng nhấn vào Button tạo mới kỳ thi và điền thông tin cần thiết của kỳ thi
* Mã nguồn chức năng



Hình 3. 11 Chức năng tạo mới kỳ thi

### **3.3.6 Chức năng tạo mới ca thi**

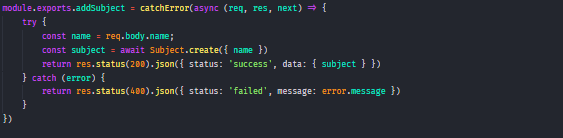
* Điều kiện tiên quyết: Người dùng đã đăng nhập với vai trò quản trị viên.
* Kịch bản: Người dùng nhấn vào Button tạo mới ca thi và điền thông tin.
* Mã nguồn chức năng



Hình 3. 12 Chức năng tạo mới ca thi

### **3.3.7 Chức năng tạo mới môn thi**

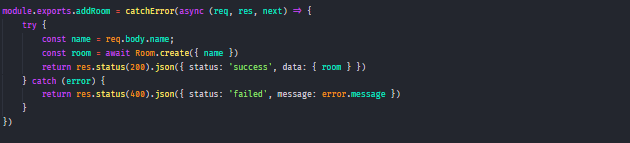
* Điều kiện tiên quyết: Người dùng đã đăng nhập với vai trò quản trị viên.
* Kịch bản: Người dùng nhấn vào Button tạo mới môn thi và điền thông tin cần thiết của môn thi.
* Mã nguồn chức năng



Hình 3. 13 Chức năng tạo mới ca thi

### **3.3.8 Chức năng tạo mới phòng thi**

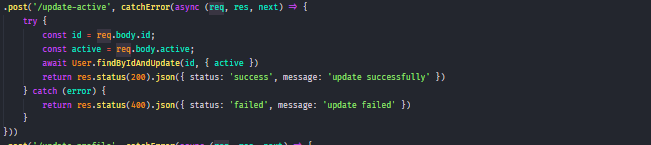
* Điều kiện tiên quyết: Người dùng đã đăng nhập với vai trò quản trị viên.
* Kịch bản: Người dùng nhấn vào Button tạo mới phòng thi và điền thông tin cần thiết của phòng thi.
* Mã nguồn chức năng



Hình 3. 14 Chức năng tạo mới phòng thi

### **3.3.9 Chức năng khóa tài khoản**

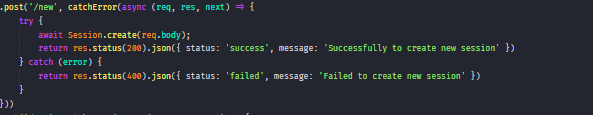
* Điều kiện tiên quyết: Người dùng đã đăng nhập với vai trò quản trị viên.
* Kịch bản: Người dùng nhấn vào Button khóa tài khoản của người dùng nào đó trong phần quản lý người dùng.
* Mã nguồn chức năng



Hình 3. 15 Chức năng khóa tài khoản

### **3.3.10 Chức năng phân công người chấm thi**

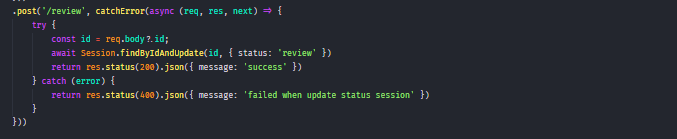
* Điều kiện tiên quyết: Người dùng đăng nhập với vai trò cán bộ quản lý.
* Kịch bản: Người dùng nhấn vào Button phân công bài thi và điền thông tin cần thiết vào form thông tin của ca thi và người chấm thi tương ứng.
* Mã nguồn chức năng



Hình 3. 16 Chức năng phân công người chấm thi

### **3.3.11 Chức năng yêu cầu chấm lại bài thi**

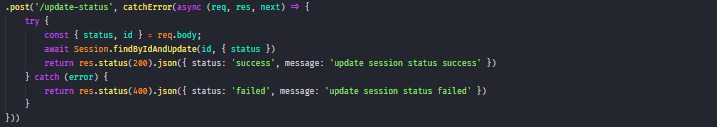
* Điều kiện tiên quyết: Người dùng đăng nhập với vai trò cán bộ quản lý.
* Kịch bản: Sau khi hai cán bộ chấm thi đã chấm xong điểm nhưng kết quả không khớp, cán bộ quản lý có quyền yêu cầu chấm lại bài thi. Sau đó nhấn vào Button chấm lại bài thi.
* Mã nguồn chức năng



Hình 3. 17 Chức năng yêu cầu chấm lại bài thi

### **3.3.12 Chức năng phê duyệt bài thi sau khi chấm**

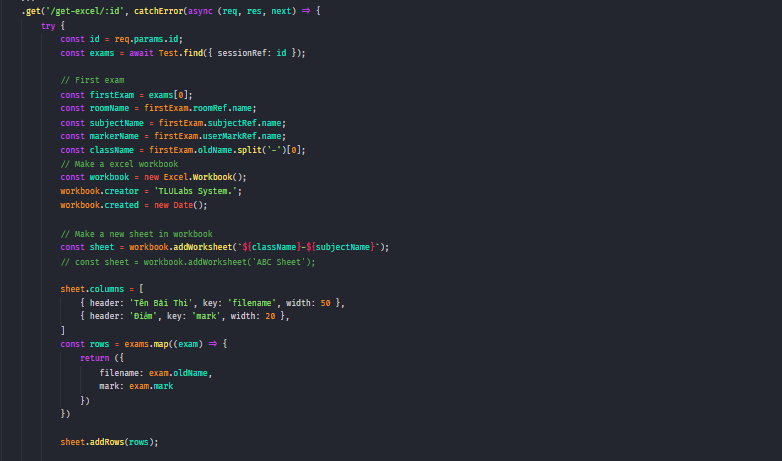
* Điều kiện tiên quyết: Người dùng đăng nhập với vai trò cán bộ quản lý.
* Kịch bản: Sau khi hai cán bộ chấm thi đã chấm xong điểm và kết quả điểm đã khớp, cán bộ quản lý sẽ ca thi đó đã chấm xong. Sau đó nhấn vào Button phê duyệt đã chấm điểm cho ca thi đó.
* Mã nguồn chức năng

******

Hình 3. 18 Chức năng tạo mới kỳ thi

### **3.3.13 Chức năng tải xuống File Excel điểm thi**

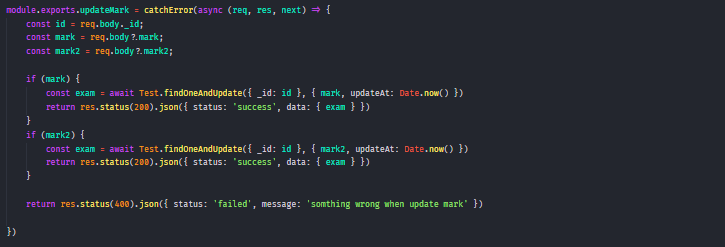
* Điều kiện tiên quyết: Người dùng đăng nhập với vai trò cán bộ quản lý.
* Kịch bản: Sau khi điểm đã khớp và đã được cán bộ quản lý phê duyệt đã chấm xong, lúc này cán bộ quản lý có thể tải xuống file đi
* Mã nguồn chức năng



Hình 3. 19 Chức năng tải xuống file excel điểm thi

### **3.3.13 Chức năng chấm thi**

* Điều kiện tiên quyết: Người dùng đăng nhập với vai trò cán bộ chấm thi.
* Kịch bản: Sau khi được cán bộ quản lý phân công cho bài thi để chấm, cán bộ chấm thi sẽ vào chấm các bài thi đã được phân công
* Mã nguồn chức năng

**

Hình 3. 20 Chức năng chấm thi

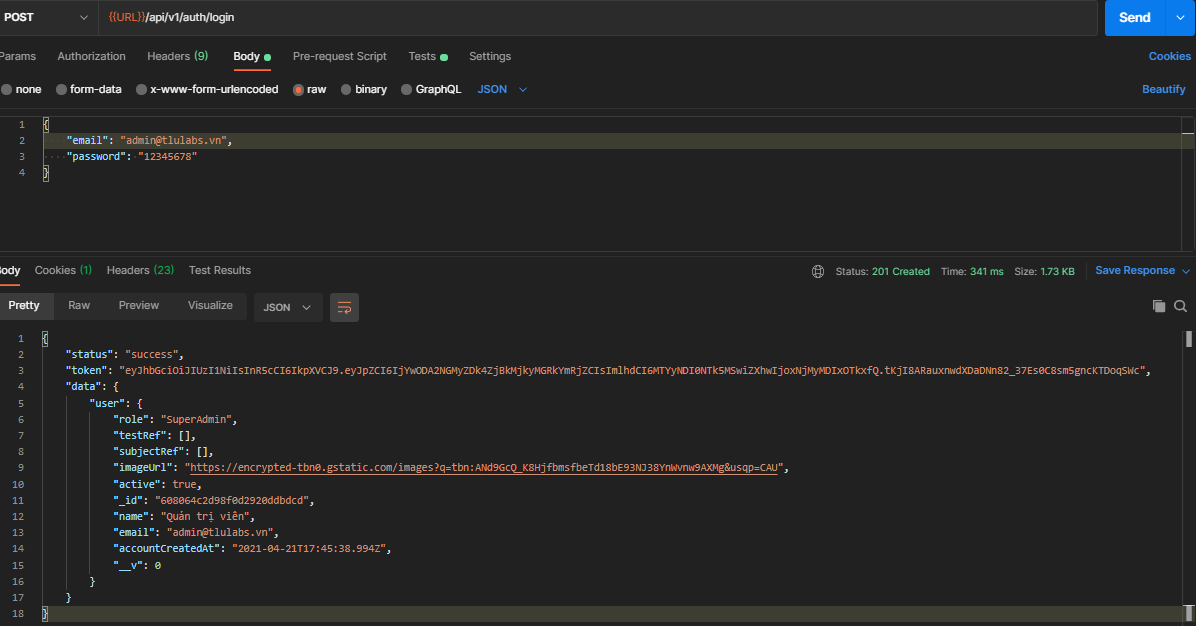
## **3.4 Kiểm thử**

Thực hiện kiểm thử API bằng Postman

### **3.4.1 Chức năng đăng nhập**

Thực hiện truy cập đường dẫn tới API đăng nhập: **/auth/login**

* Cung cấp đầy đủ thông tin đăng nhập cần thiết như Email và Mật khẩu
* Điền đúng thông tin email và mật khẩu, kết quả trả về sẽ là thông tin người dùng kèm theo đó làm một mã Token xác minh đăng nhập thành công.

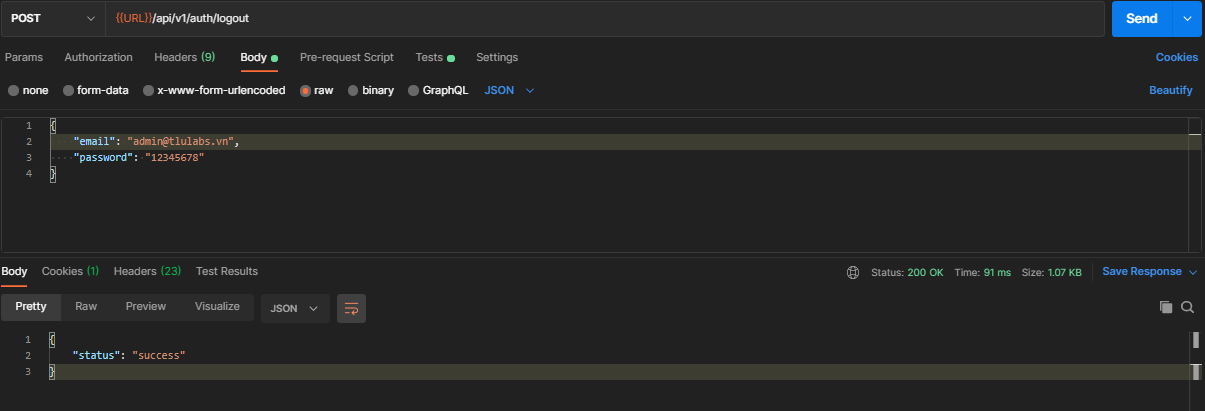


Hình 3. 21 Kiểm thử chức năng đăng nhập

### **3.4.2 Chức năng đăng xuất**

Thực hiện truy cập đường dẫn tới API đăng nhập: **/auth/logout**

* Sau khi truy cập API, người dùng sẽ bị xóa đi Token và dữ liệu trả về “success”.

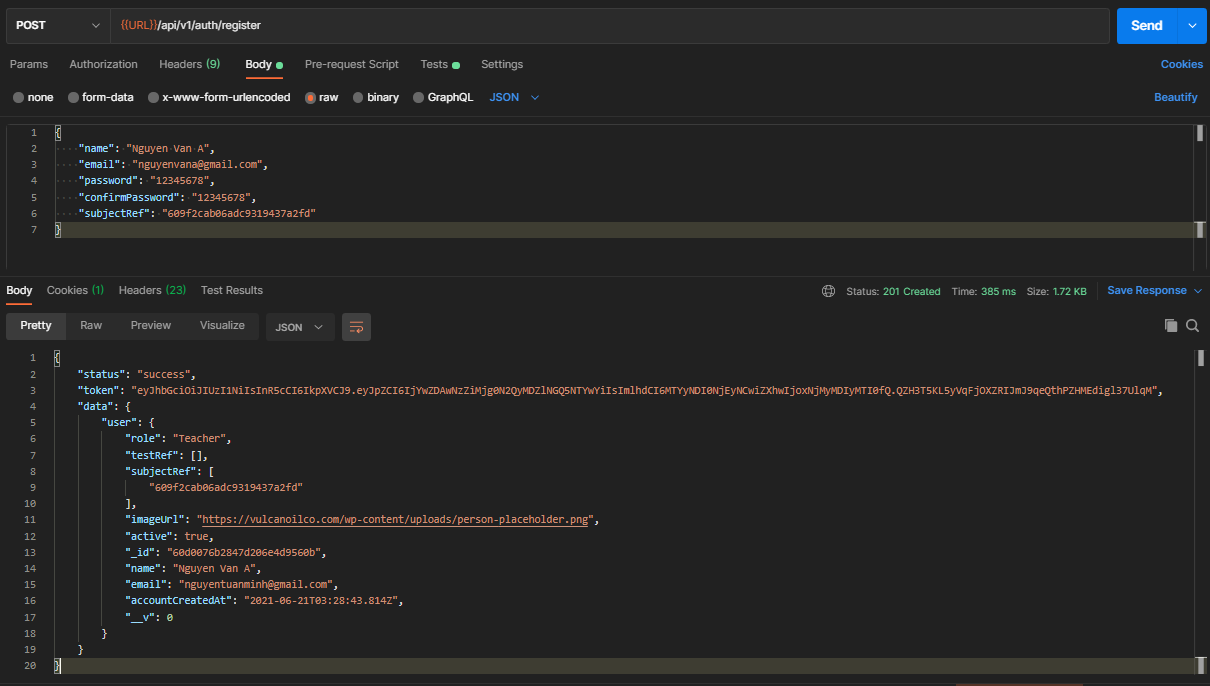


Hình 3. 22 Kiểm thử chức năng đăng xuất

### **3.4.3 Chức năng đăng ký**

Thực hiện truy cập đường dẫn tới API đăng nhập: **/auth/register**

* Cung cấp đầy đủ thông tin của người dùng đăng ký như Email, Mật Khẩu và Môn chấm thi nếu đó là cán bộ chấm thi.
* Sau khi điền đầy đủ thông tin dữ liệu sẽ trả về thông tin người dùng vừa đăng ký.

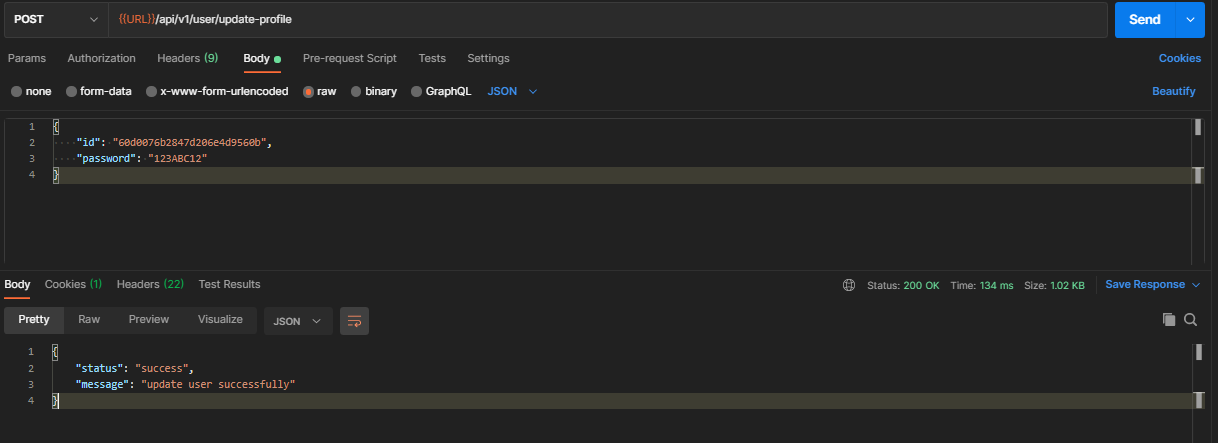


Hình 3. 23 Kiểm thử chức năng đăng ký

### **3.4.4 Chức năng thay đổi mật khẩu**

Thực hiện truy cập đường dẫn tới API đăng nhập: **/user/update-profile**

* Điền thông tin mật khẩu mới và hệ thống trả về thông báo thành công “success”.

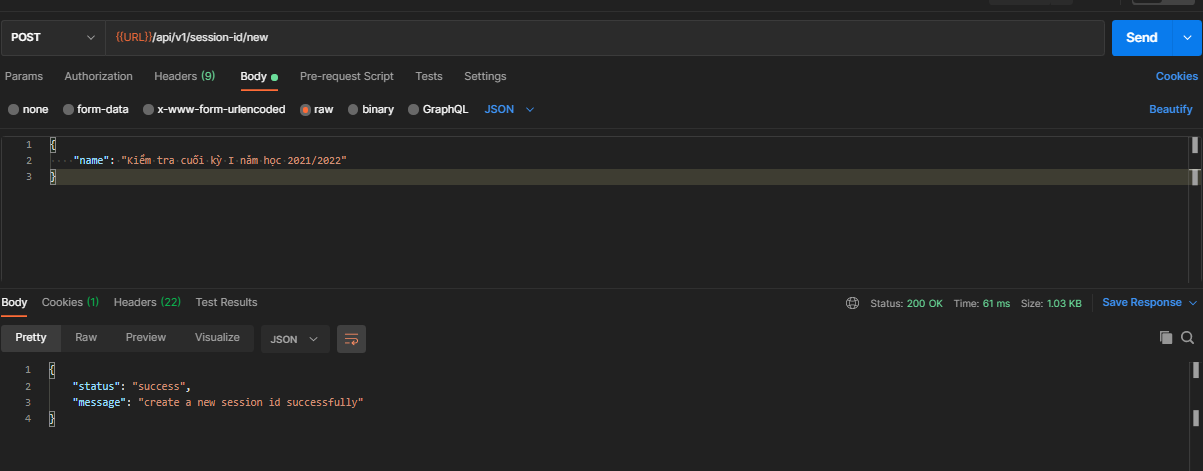


Hình 3. 24 Kiểm thử chức năng thay đổi mật khẩu

### **3.4.5 Chức năng tạo mới kỳ thi**

Thực hiện truy cập đường dẫn tới API đăng nhập: **/session-id/new**

* Cung cấp tên của kỳ thi cần tạo mới.
* Sau khi cung cấp thông tin đầy đủ, hệ thống sẽ trả về tạo mới thành công.

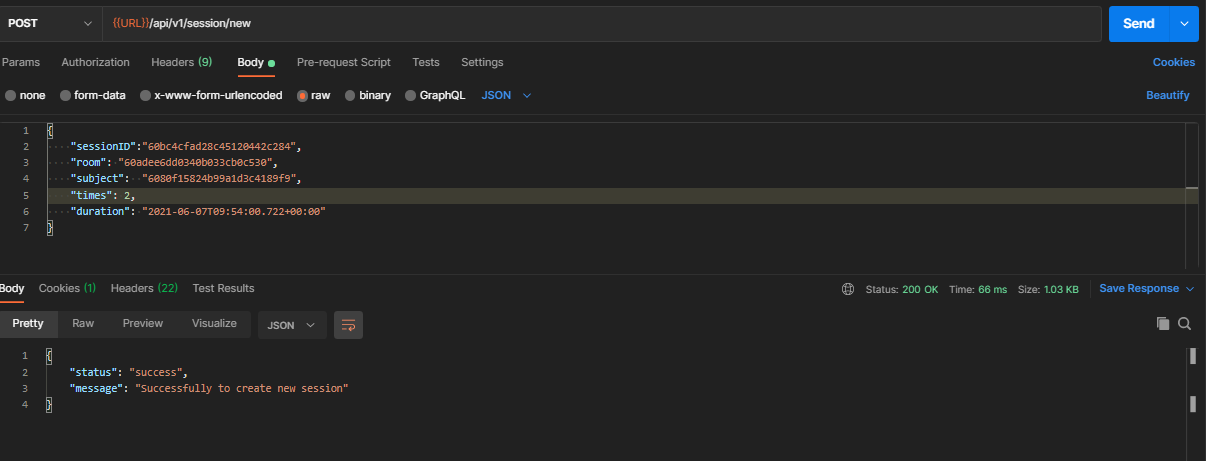


Hình 3. 25 Kiểm thử chức năng tạo mới kỳ thi

### **3.4.6 Chức năng tạo mới ca thi**

Thực hiện truy cập đường dẫn tới API đăng nhập: **/session /login**

* Cung cấp đầy đủ thông tin đăng nhập cần thiết như Email và Mật khẩu
* Điền đúng thông tin email và mật khẩu, kết quả trả về sẽ là thông tin người dùng kèm theo đó làm một mã Token xác minh đăng nhập thành công.

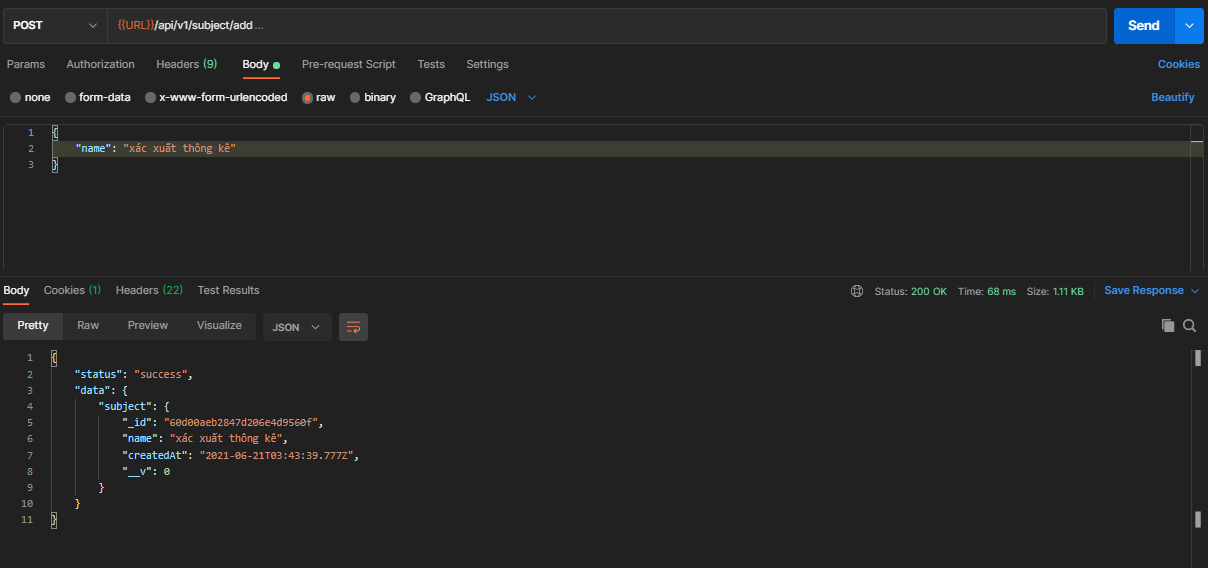


Hình 3. 26 Kiểm thử chức năng tạo mới ca thi

### **3.4.7 Chức năng tạo mới môn thi**

Thực hiện truy cập đường dẫn tới API đăng nhập: **/subject /add**

* Điền đầy đủ thông tin về tên ca thi.
* Sau khi cung cấp đủ thông tin, hệ thống trả về thông tin kỳ thì vừa được tạo thành công.

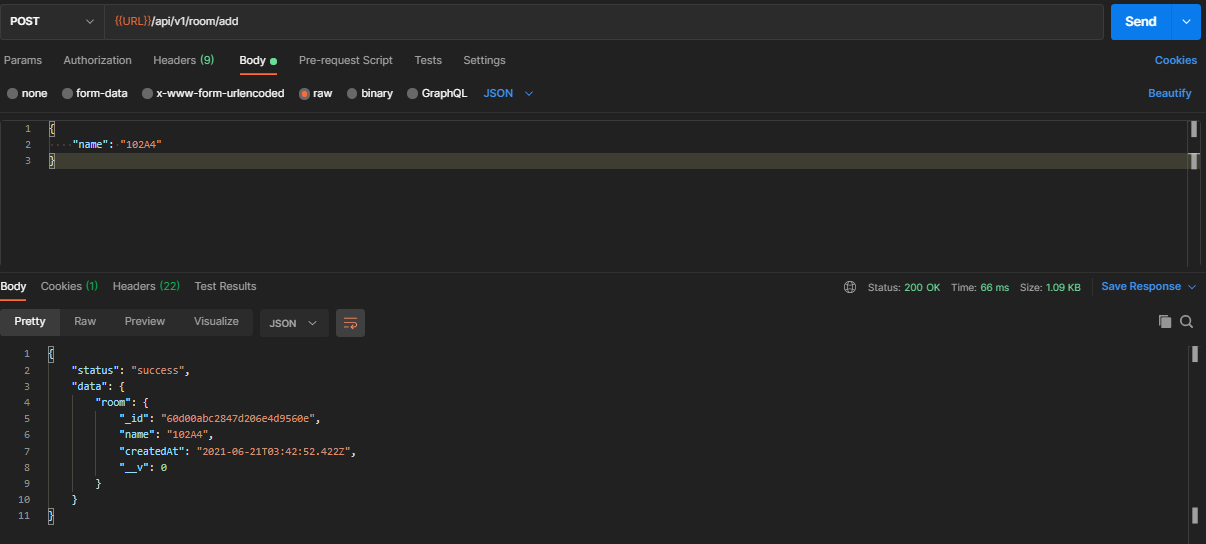


Hình 3. 27 Kiểm thử chức năng tạo mới môn thi

### **3.4.8 Chức năng tạo mới phòng thi**

Thực hiện truy cập đường dẫn tới API đăng nhập: **/room /add**

* Điền đầy đủ thông tin về phòng thi, sau đó hệ thống trả về thông tin vừa tạo.

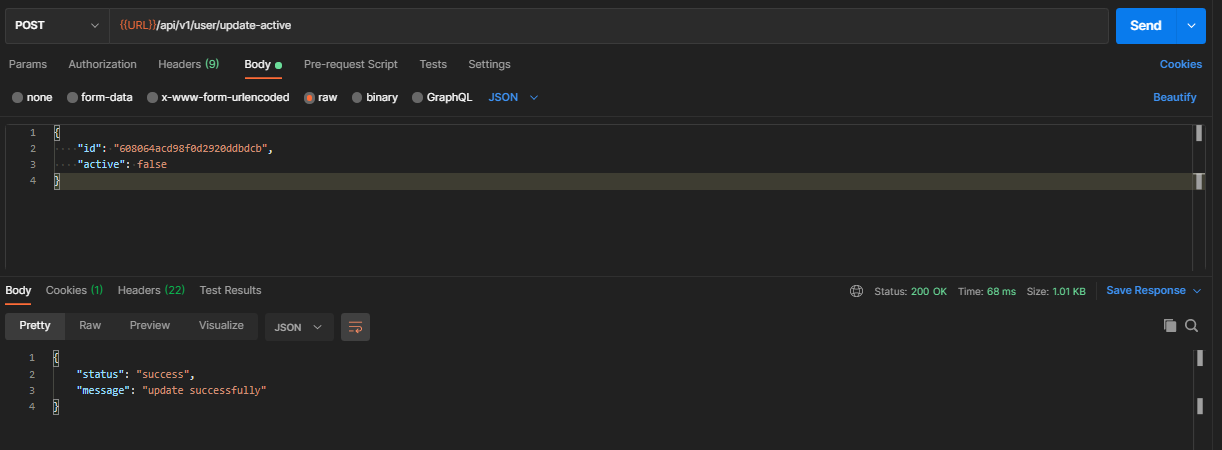


Hình 3. 28 Kiểm thử chức năng tạo mới phòng thi

### **3.4.9 Chức năng khóa tài khoản**

Thực hiện truy cập đường dẫn tới API đăng nhập: **/user /update-active**

* Cung cấp ID của người cần khóa tài khoản và trạng thái [active=False]. Sau đó hệ thống sẽ trả về thông tin thành công “success”.

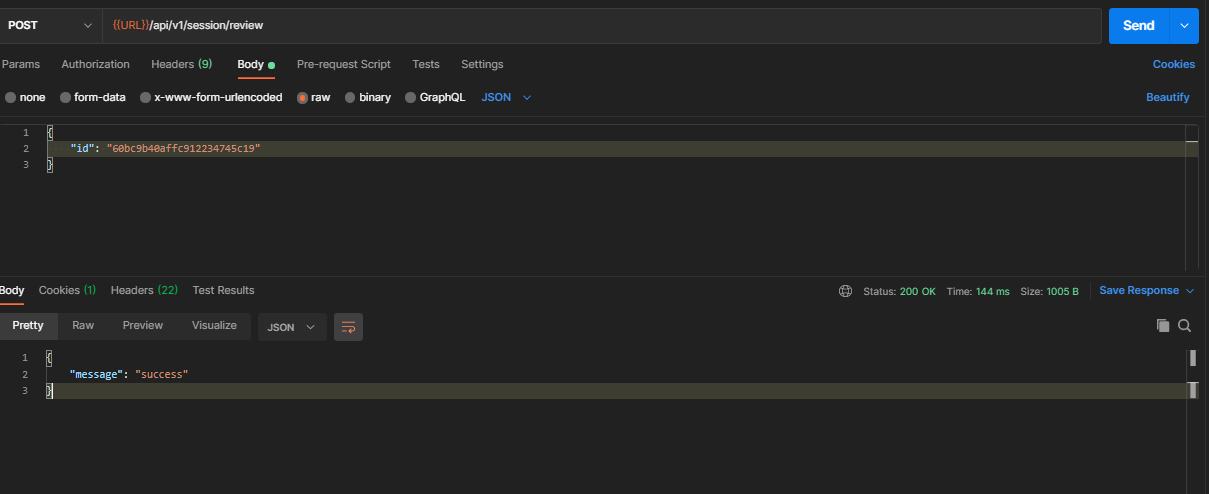


Hình 3. 29 Kiểm thử chức năng khóa tài khoản

### **3.4.10 Chức năng yêu cầu chấm lại bài thi**

Thực hiện truy cập đường dẫn tới API đăng nhập: **/session/review**

* Sau khi hai cán bộ chấm thi chấm bài thành công nhưng điểm của hai giám thị chấm không khớp với nhau, lúc này quản trị viên nhấn vào nút chấm lại và gửi kèm theo mã ID của ca thi.
* Sau khi yêu cầu chấm lại thành công hệ thống trả về thành công “success”.

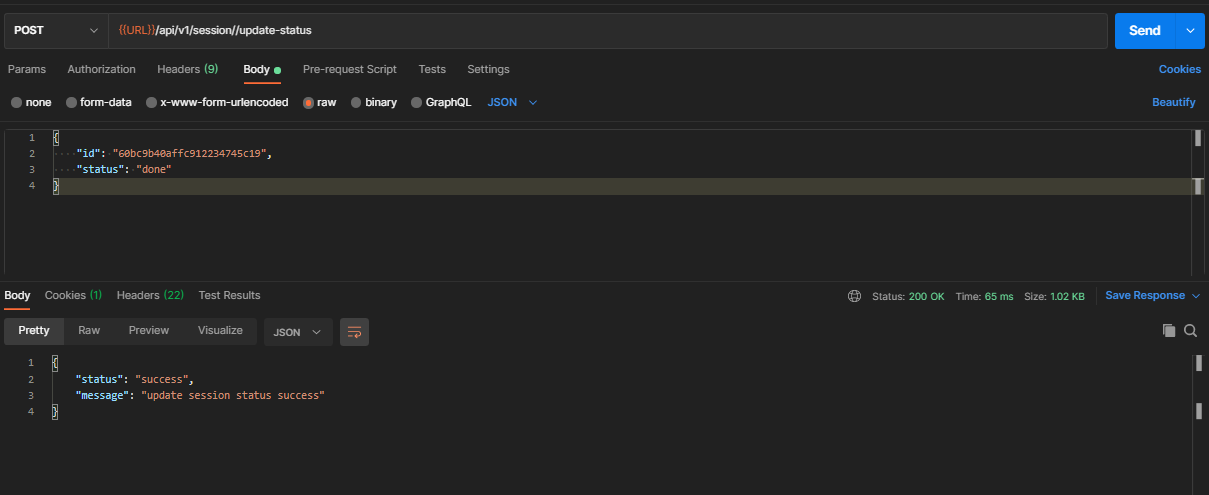


Hình 3. 30 Kiểm thử chức năng yêu cầu chấm lại.

### **3.4.11 Chức năng phê duyệt bài thi sau khi chấm**

Thực hiện truy cập đường dẫn tới API đăng nhập: **/session /update-status**

* Khi kết quả chấm thi của hai cán bộ chấm thi đã khớp, lúc này quản trị viên cung cấp thông tin ID của ca thi kèm theo trạng tái [status=“done”]. Hệ thống trả về thông báo “success”

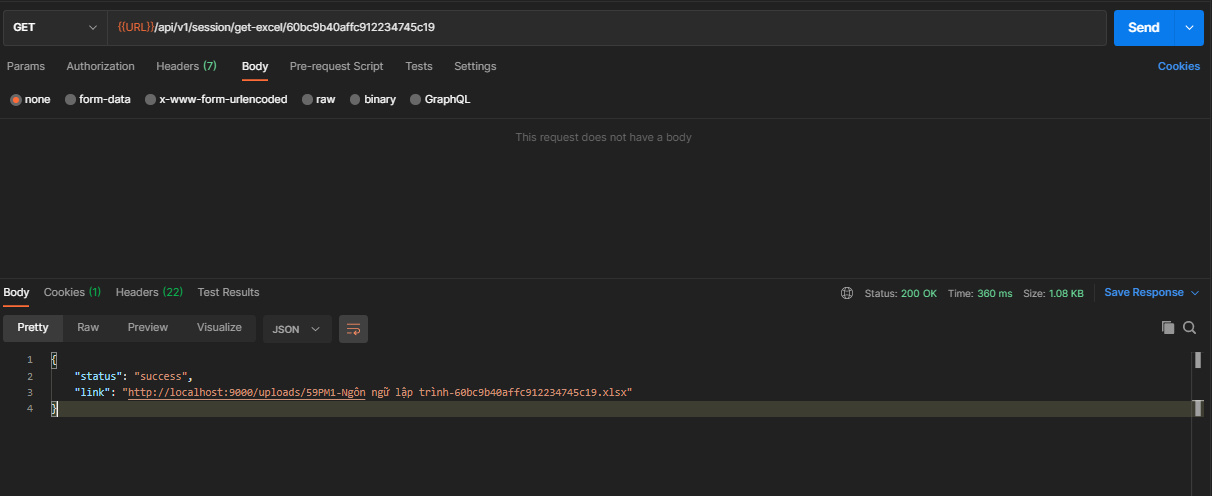


Hình 3. 31 Kiểm thử chức năng phê duyệt bài thi sau khi chấm

### **3.4.12 Chức năng tải xuống File Excel điểm thi**

Thực hiện truy cập đường dẫn tới API đăng nhập: **/session /get-excel/:id**

* Cung cấp thông tin mã ca thi. Khi ca thi đó đã được chấm thành công, hệ thống trả về thông tin kèm theo đường dẫn tải xuống của tệp Excel.

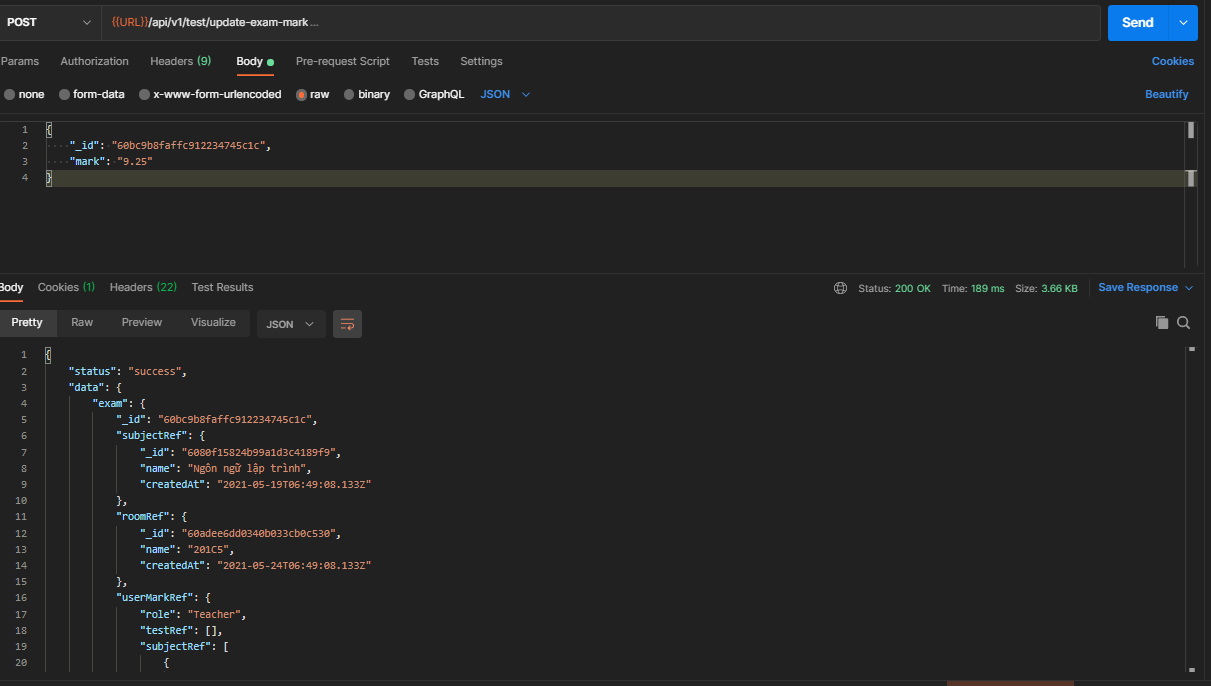


Hình 3. 32 Kiểm thử chức năng đăng nhập

### **3.4.13 Chức năng chấm thi**

Thực hiện truy cập đường dẫn tới API đăng nhập: **/test /update-exam-mark**

* Cán bộ chấm thi và điền điểm tương ứng cho từng bài thi.
* Hệ thống trả về thông tin bài thi với điểm đã được cập nhật.

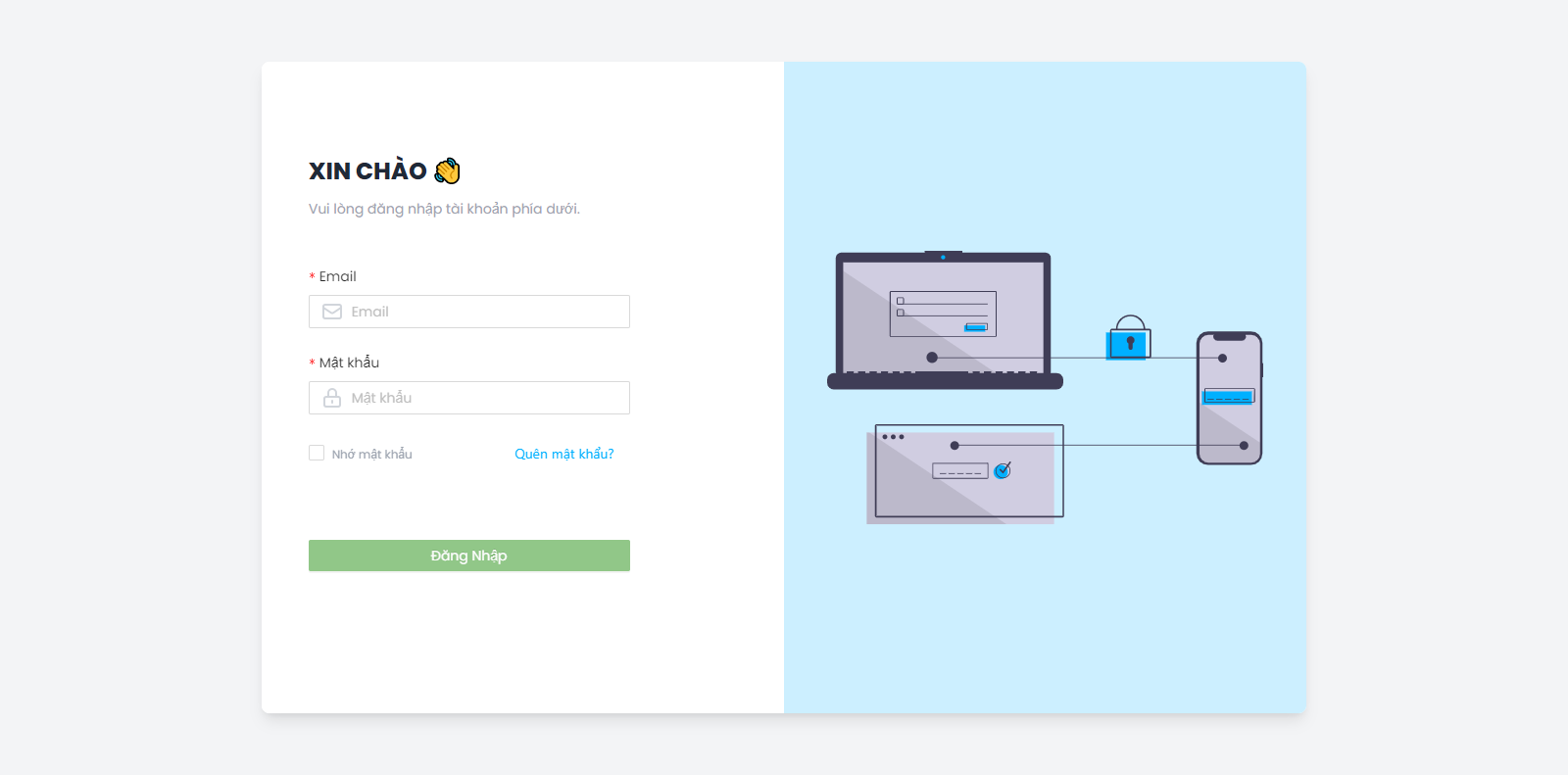


Hình 3. 33 Kiểm thử chức năng chấm thi

## **3.5 Kết quả sau khi xây dựng hệ thống**

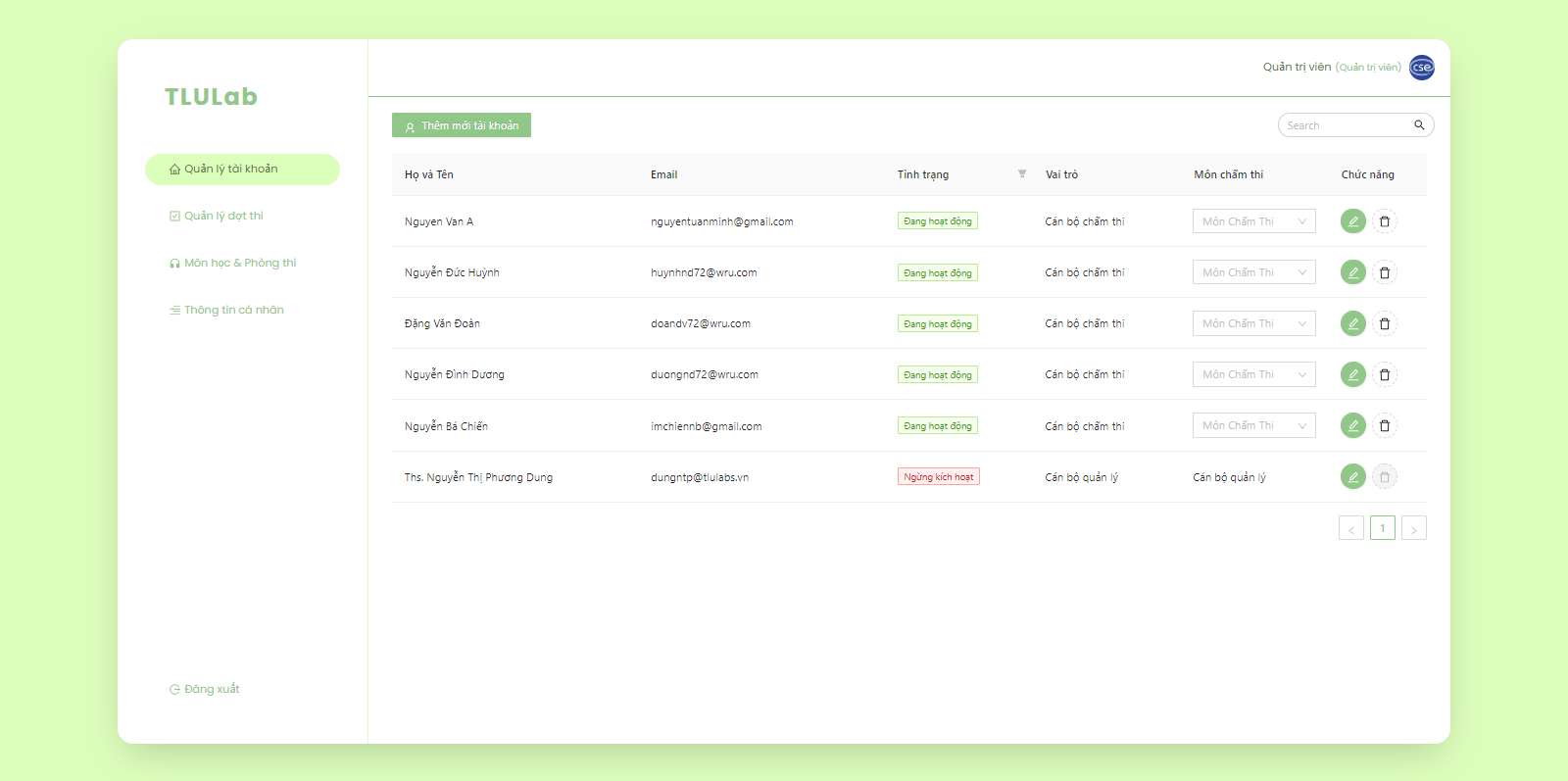
### **3.5.1 Client**

* Màn hình đăng nhập trên ứng dụng, người dùng điền đầy đủ thông tin vào trang đăng nhập phía dưới.



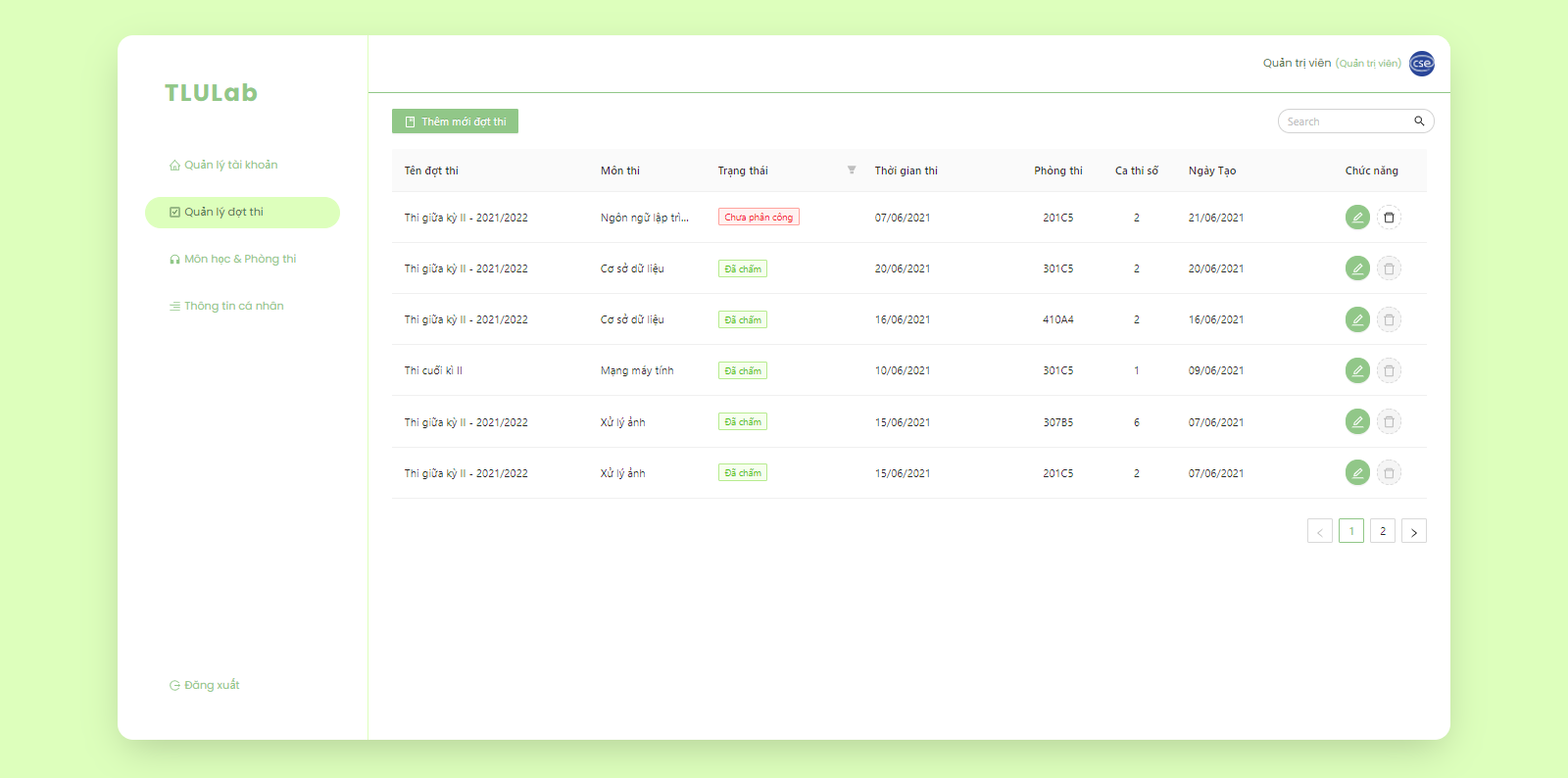
Hình 3. 34 Màn hình đăng nhập

* Giao diện của quản trị viên sau khi đăng nhập thành công. Khi đăng nhập thành công ứng dụng sẽ chuyển được chuyển hướng đến trang quản trị người dùng của hệ thống, tại đây quả trị viên có thể theo dõi và quản lý toàn bộ thông tin người dùng trên hệ thống.



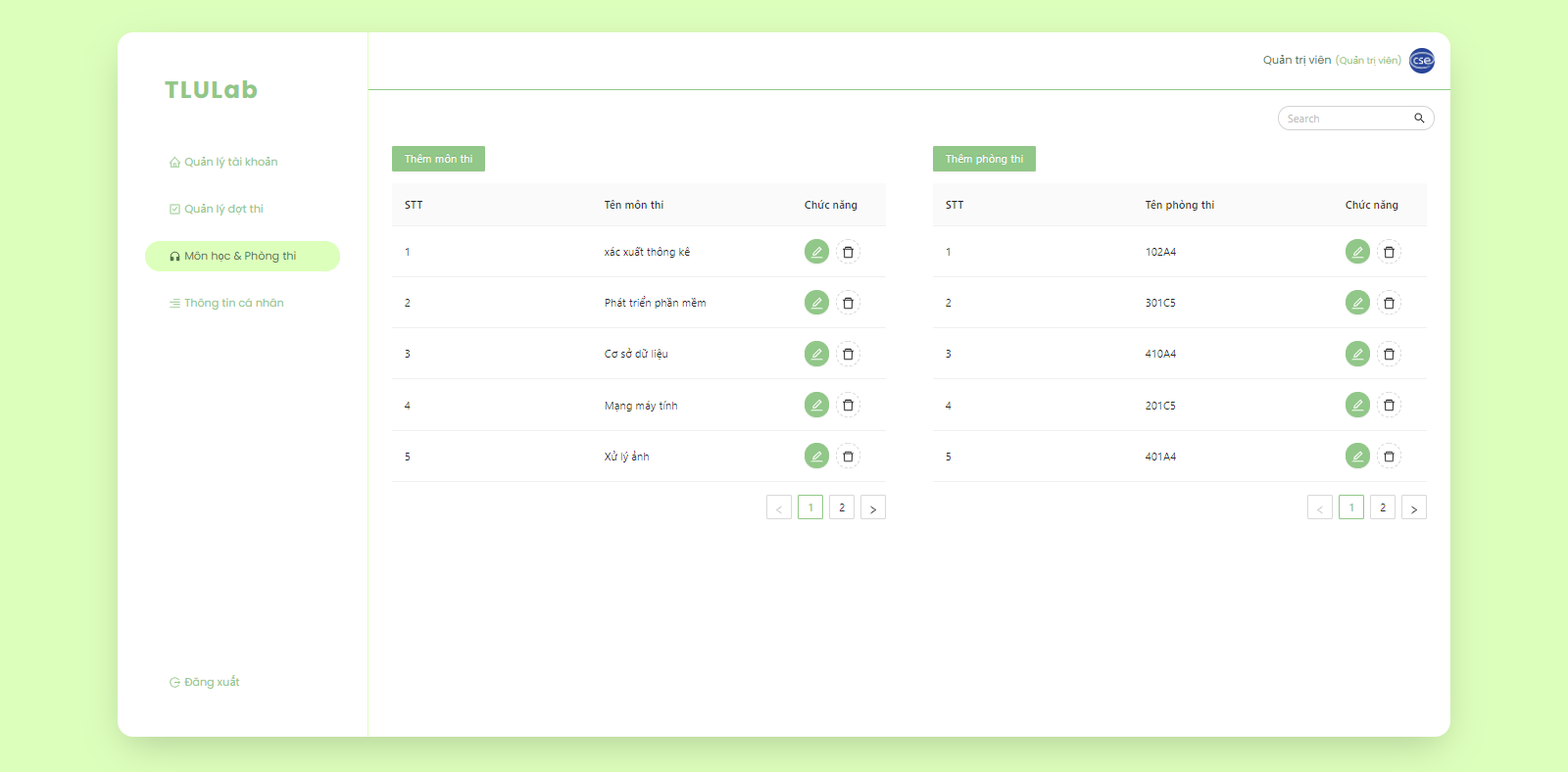
Hình 3. 35 Màn hình quản lý tài khoản

* Giao diện quản lý các kỳ thi của hệ thống, tại đây quản trị viên có thể theo dõi được các kỳ thì đang có trên hệ thống, kèm theo đấy là trạng thái ca thi.



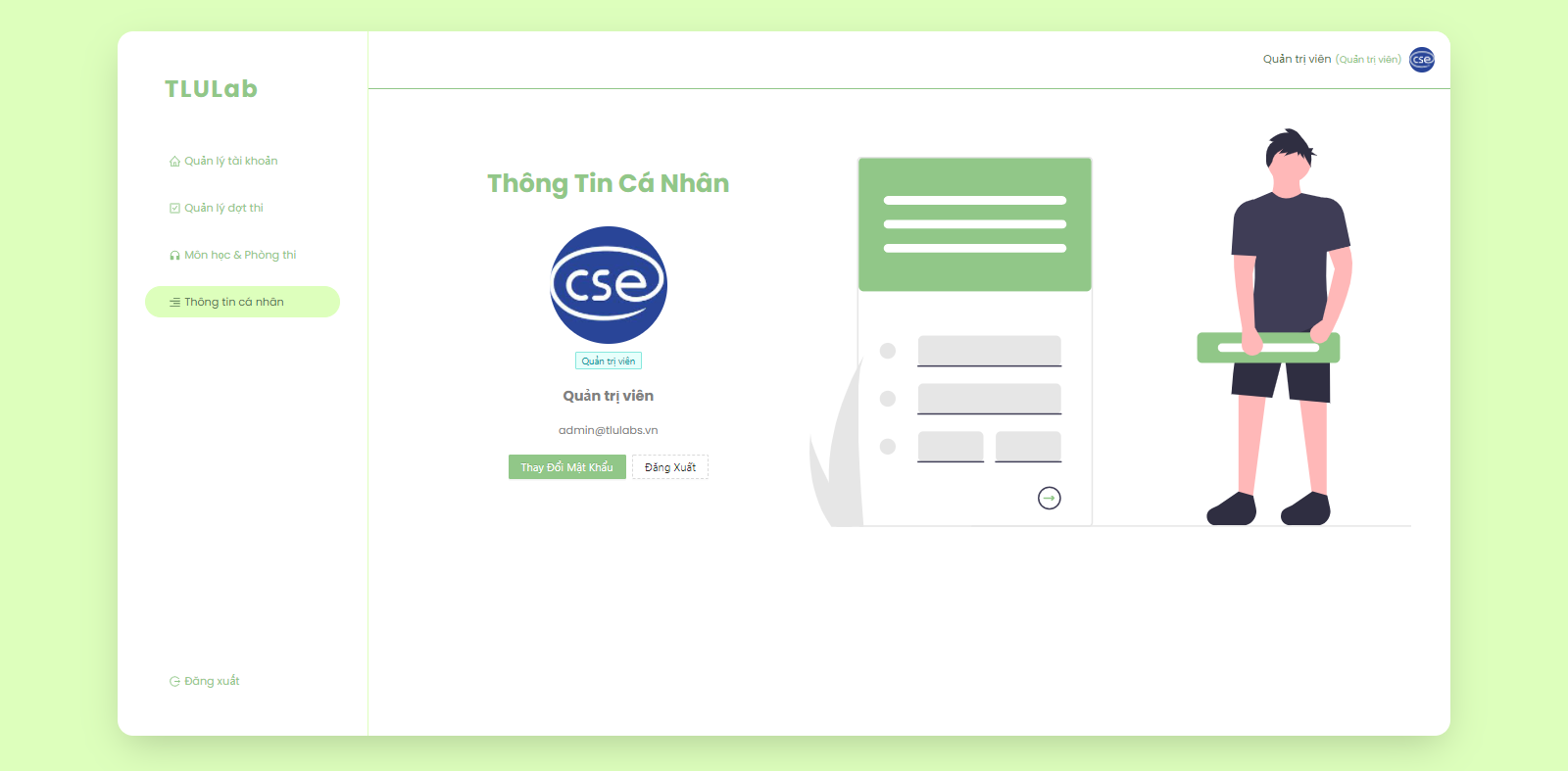
Hình 3. 36 Màn hình quản lý kỳ thi

* Giao diện quản lý các môn học và phòng thi đang có trên hệ thống, tại đây quản trị viên có thể thêm mới xóa hoặc sửa các thông tin môn học và phòng thi tương ứng.



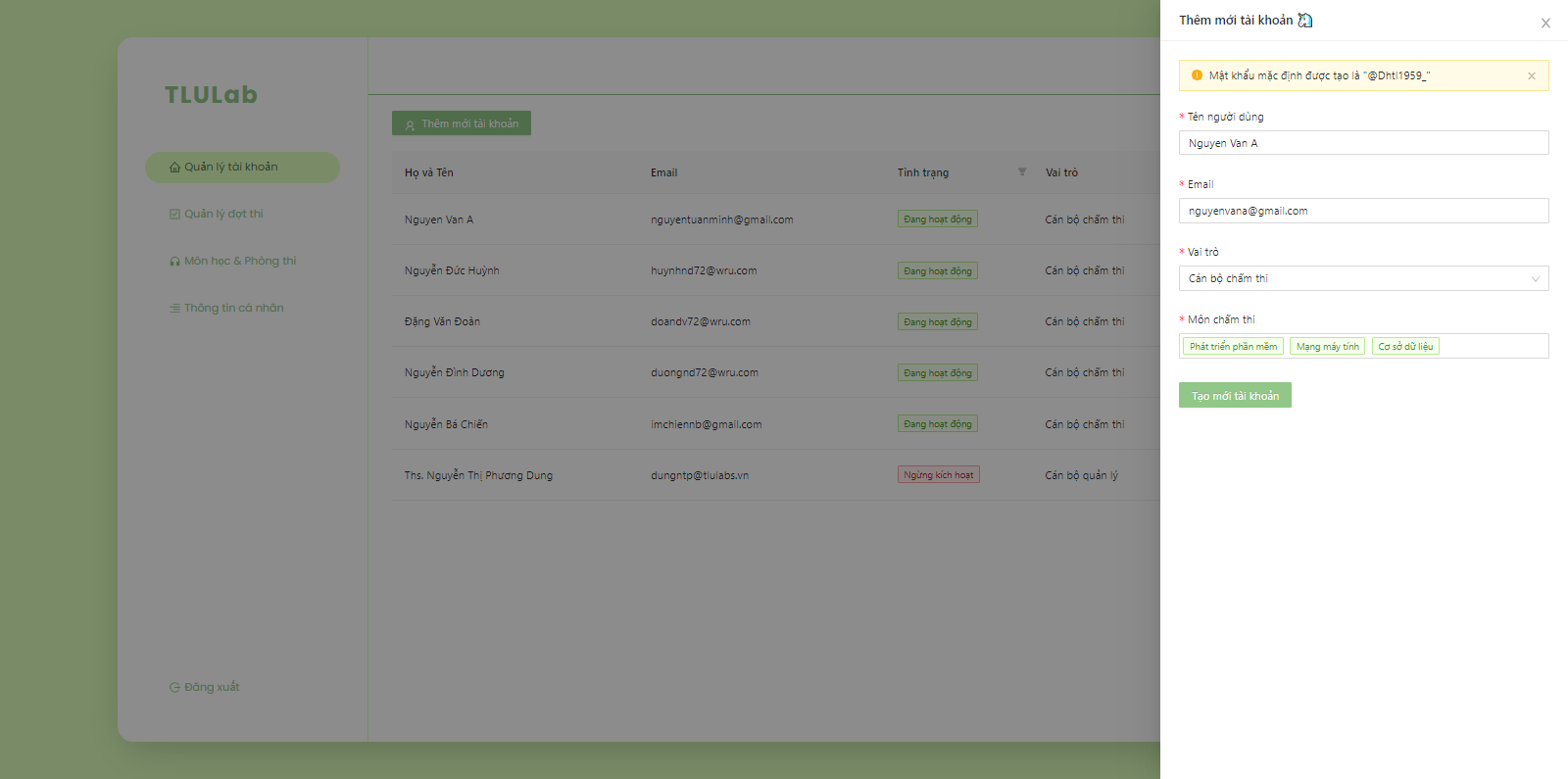
Hình 3. 37 Màn hình quản lý môn học và phòng thi

* Giao diện trang thông tin cá nhân, tại đây người dùng có thể thay đổi mật khẩu và đăng xuất tài khoản. Nhấn vào nút Thay đổi mật khẩu để tiến hành thay đổi mật khẩu.



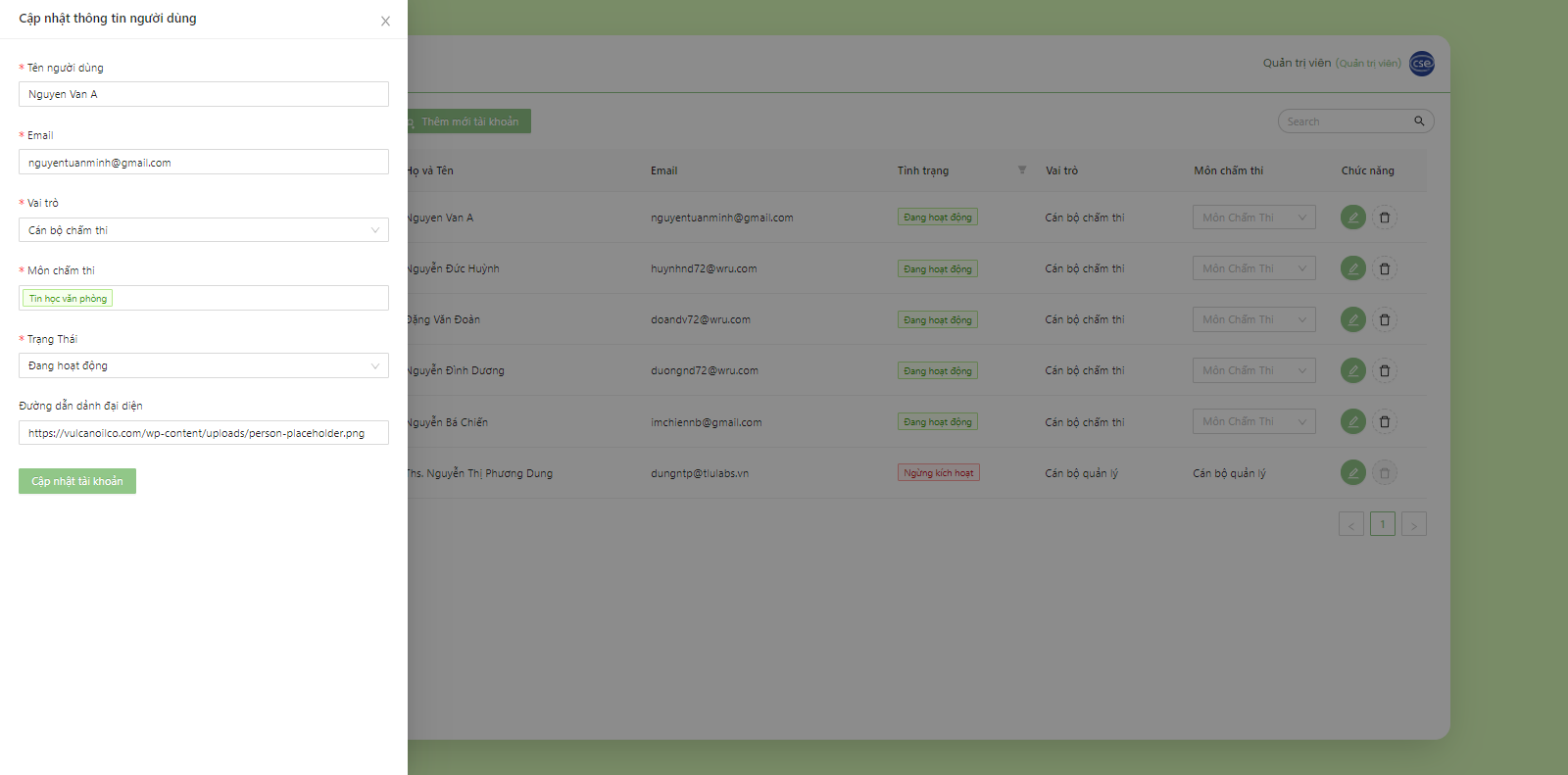
Hình 3. 38 Màn hình thông tin cá nhân

* Giao diện thêm mới tài khoản, tại đây quản trị viên điều đầy đủ thông tin của tài khoản bao gồm: Email, Tên Người Dùng,Vai Trò,... Sau đó nhấn vào nút thêm mới tài khoản để thêm mới tài khoản.



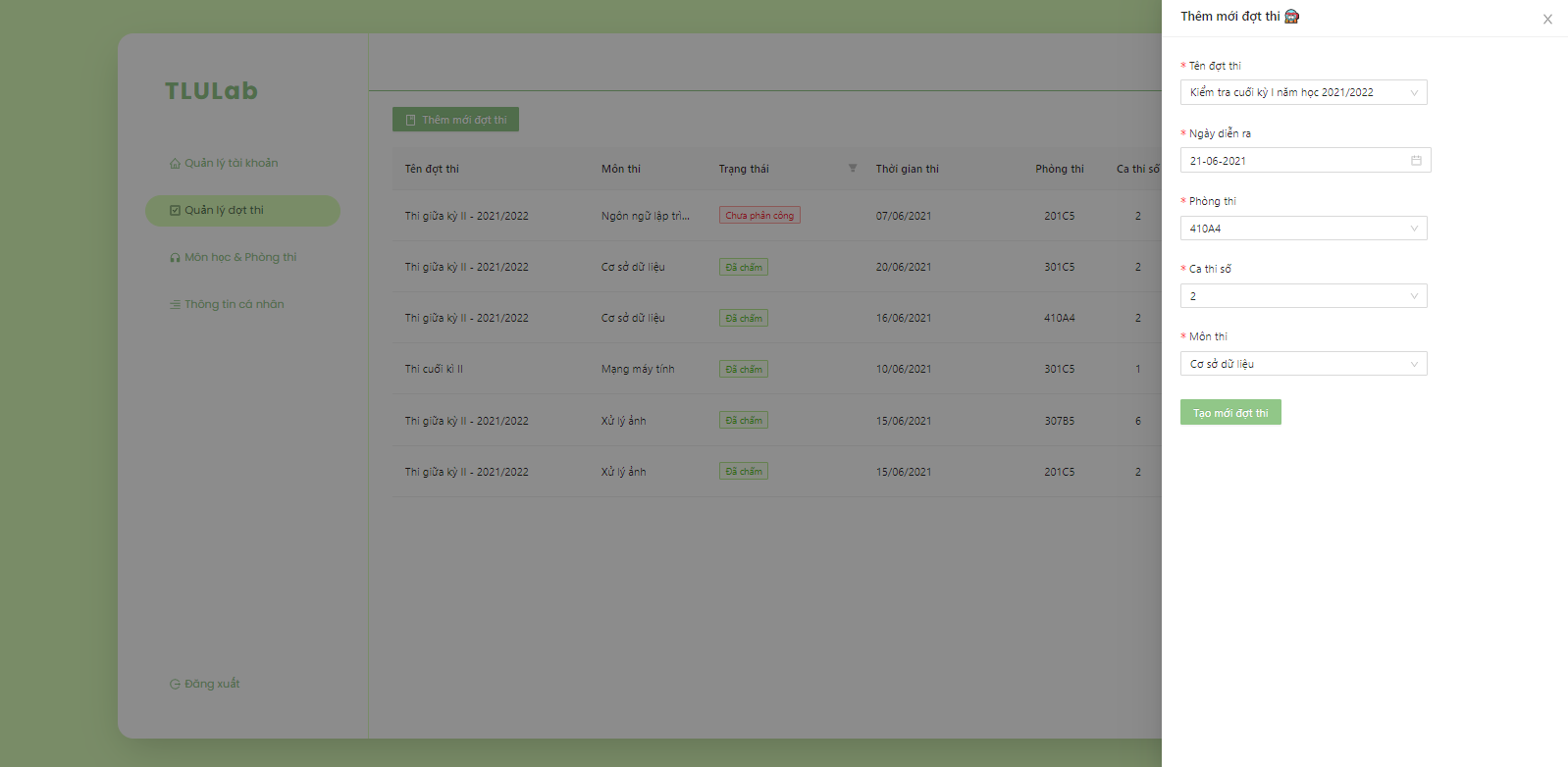
Hình 3. 39 Màn hình thêm mới tài khoản

* Giao diện quản lý các môn học và phòng thi đang có trên hệ thống, tại đây quản trị viên có thể thêm mới xóa hoặc sửa các thông tin môn học và phòng thi tương ứng.



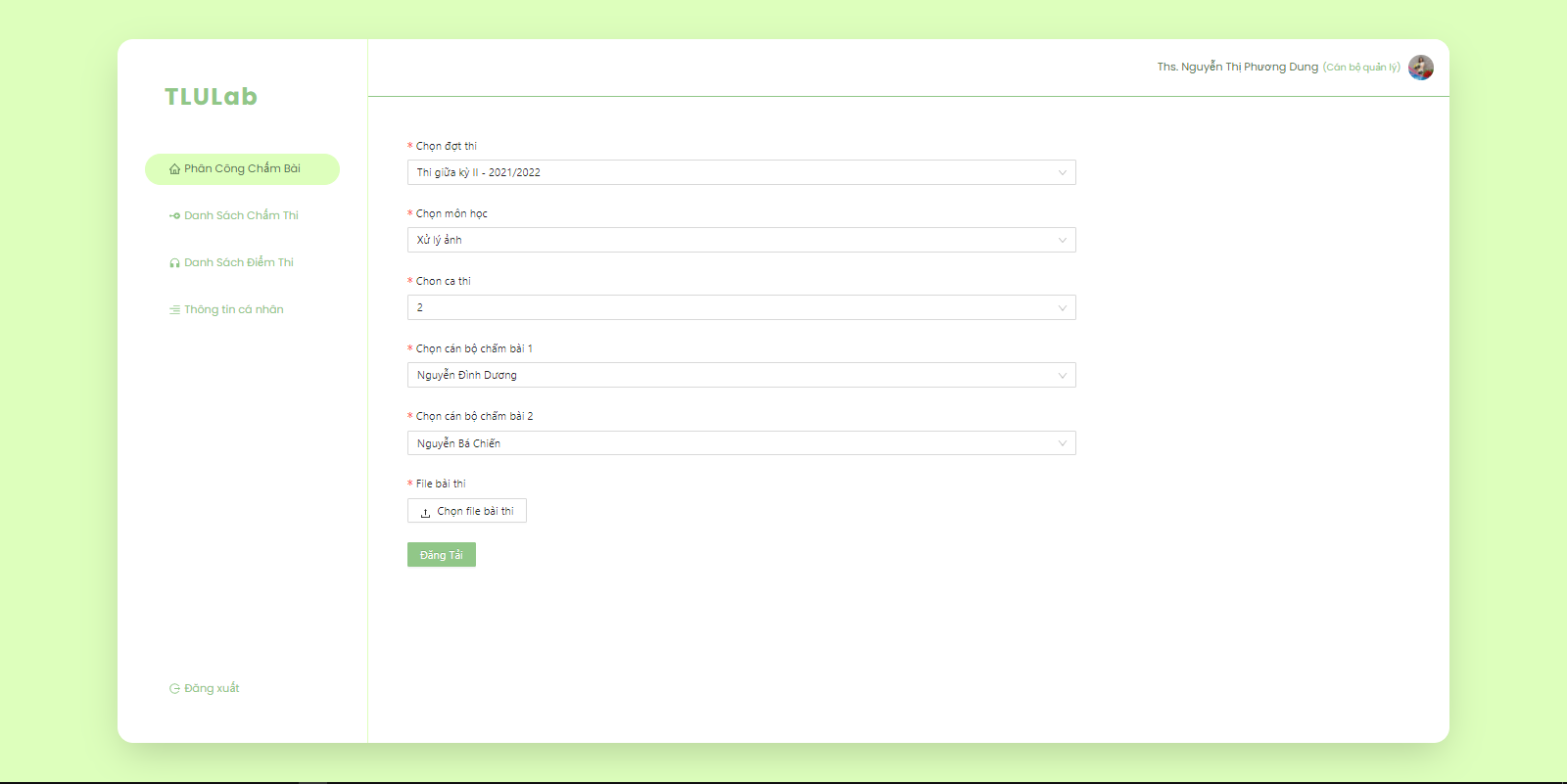
Hình 3. 40 Màn hình thay đổi thông tin người dùng

* Giao diện thêm mới một đợt thi, tại đây quản trị viên sẽ điền thông tin về đợt thi này bao gồm tên môn thi, ngày diễn ra, phòng thi, ca thi và môn thi. Sau khi nhấn vào nút tạo mới kỳ thi sẽ được thêm vào danh sách.



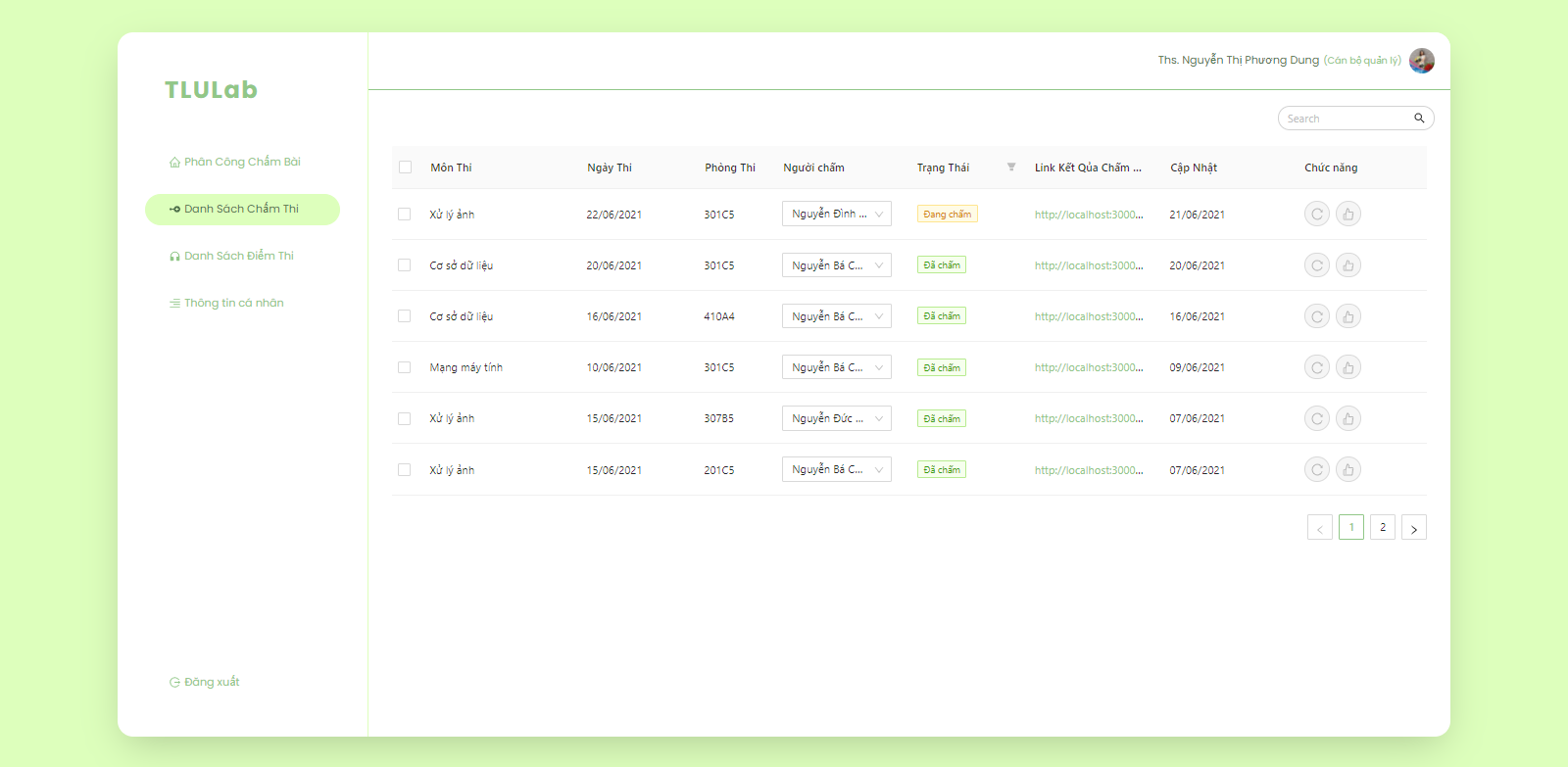
Hình 3. 41 Màn hình thêm mới ca thi

* Giao diện phân công người chấm thi cho ca thi, tại đây cán bộ quản lý sẽ phân công hai người chấm thi cho môn thi và tải lên các tệp bài thi. Hệ thống sẽ lưu trữ và thay đổi tên của bài thi thành một tên mới để tránh tình trạng người thi chấm biết thông tin về bài thi và chấm thi thiên vị (đánh phách).



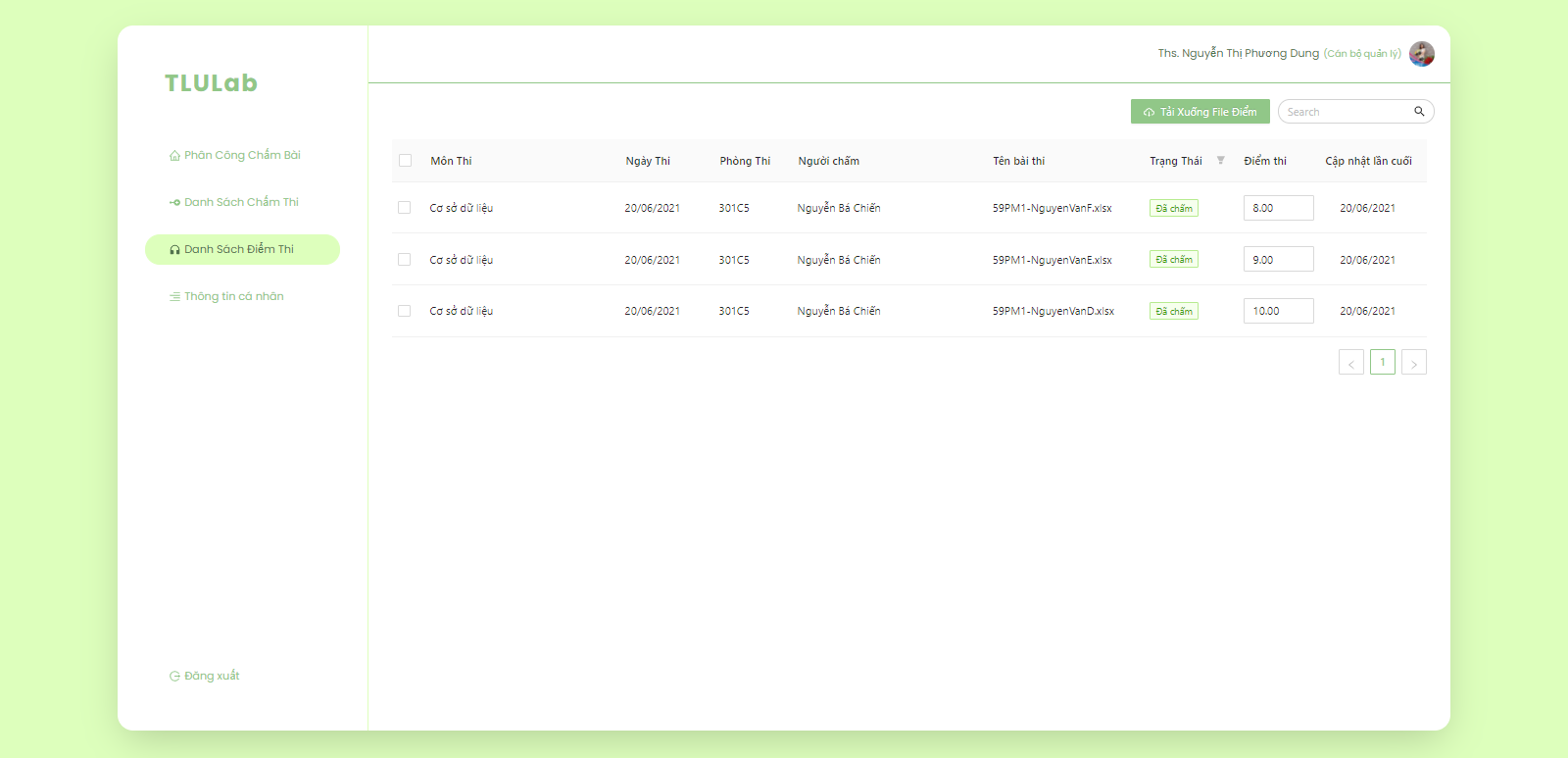
Hình 3. 42 Màn hình phân chia ca thi và người chấm

* Giao diện quản lý các ca thi đã được phân công cho các cán bộ chấm thi. Tai đây cán bộ quản lý có thể theo dõi được trạng thái chấm thị của các cán bộ chấm thi, nếu đã khớp điểm cán bộ quản lý có thể đánh dấu là ca thi đã chấm xong.



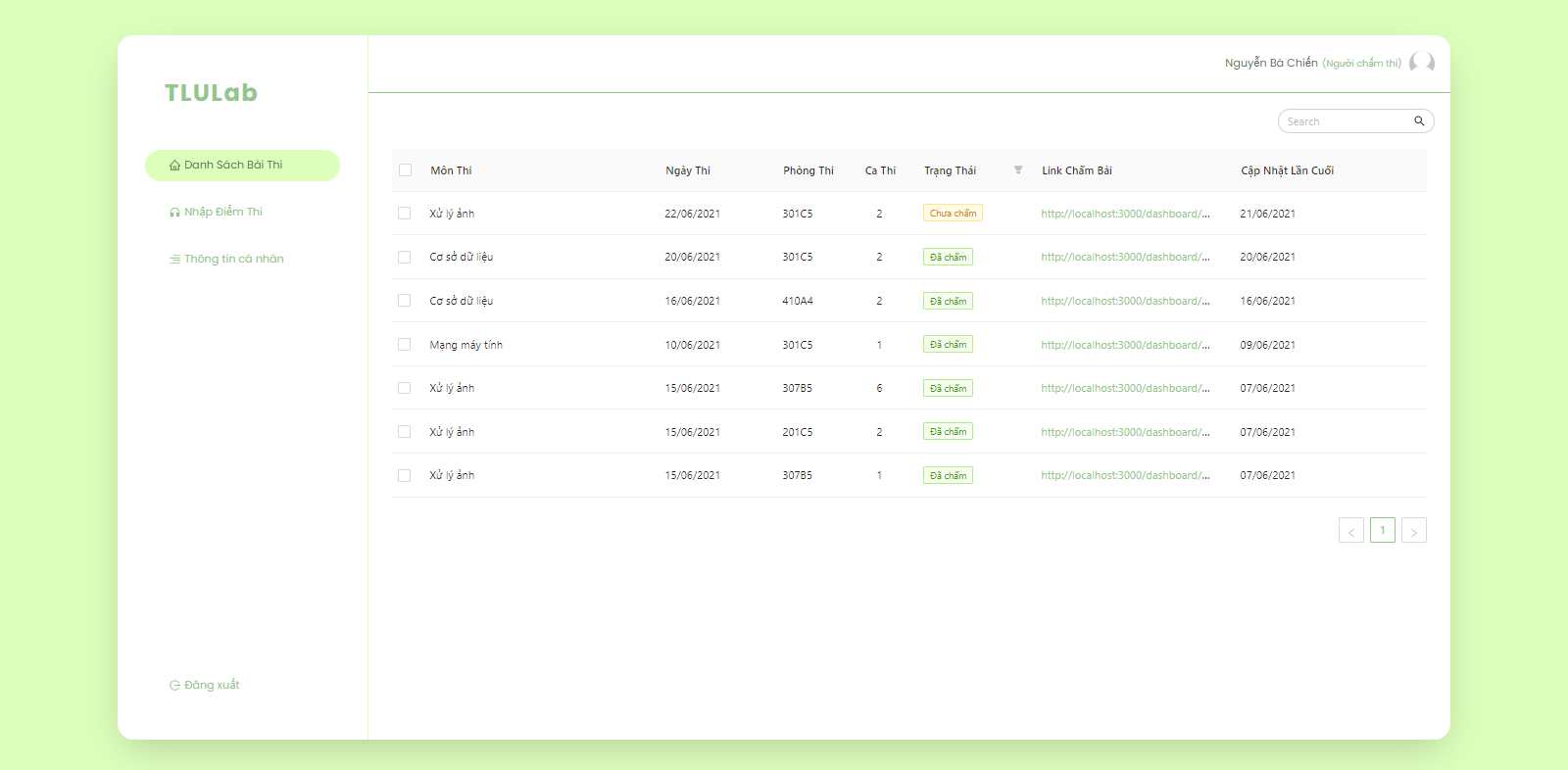
Hình 3. 43 Màn hình quản lý các ca thi đã phân công chấm

* Giao diện xem điểm của các môn thi khi đã được chấm xong.



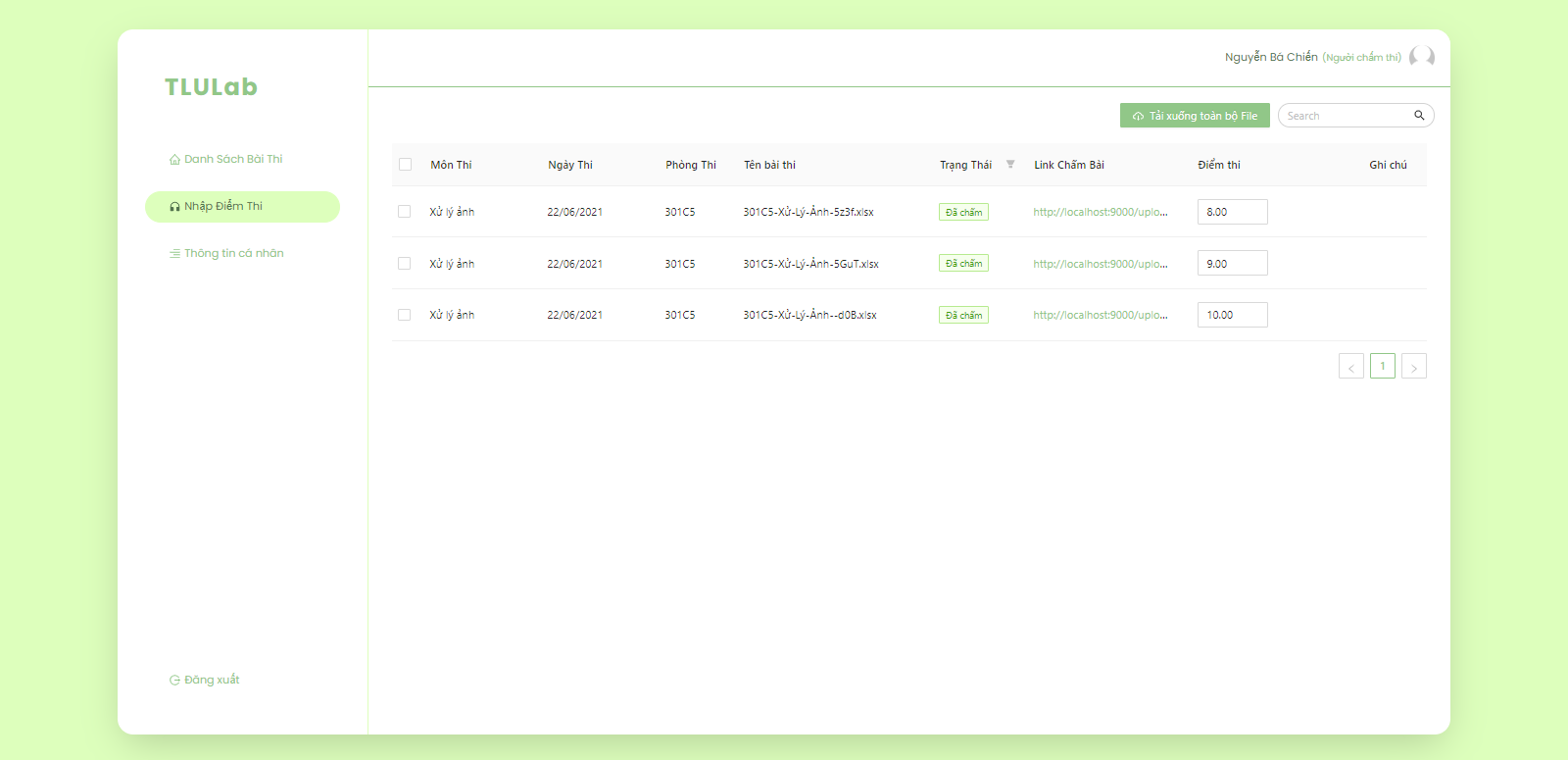
Hình 3. 44 Màn hình xem điểm của ca thi khi đã chấm xong

* Giao diện quản lý các môn thi của cán bộ quản lý đã phân công cho từng cán bộ chấm thi nhất định. Tại đây cán bộ chấm thi có thể quản lý và theo dõi tình trạn của các ca thi mình đã được phân công chấm.



Hình 3. 45 Màn hình các ca thi đã được phân công của cán bộ chấm thi

* Giao diện chấm thi của cán bộ chấm thi, tại đây cán bộ chấm thi có thể tải xuống từng File bài thi để chấm hoặc cũng có thể tải File bài thi đã được nén lại.



Hình 3. 46 Màn hình chấm thi của cán bộ chấm thi

### **3.5.2 Server**

* Đã thực hiện xây dựng được hệ thống Backend có sử dụng NodeJS và ExpresJS
* Xây được mô hình API dưới dạng Restful API.
* Thực hiện bảo mật người dùng bằng cách sử dụng cơ chế Token của JWT.
* Ngăn chặn từ một địa chỉ gửi đi quá nhiều API cùng một lúc nhờ sử dụng thư viện express-rate-limit.
* Thực hiện việc bảo vệ các request gửi lên Server bằng cách sử dụng thư viện Helmet.

# CHƯƠNG 4 KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

## **4.1 Kết quả thu được**

### **4.1.1 Lý thuyết**

* Biết cách sử dụng React để xây dụng một ứng dụng trên nền tảng web.
* Biết cách kết hợp nhiều thư viện xây dựng giao diện người dùng trên ứng dụng web để quá trình xây dựng hiệu quả.
* Sử dụng NodeJS và Express để xây dựng một hệ thống Server có sử dụng Restful API.
* Biết cách sử dụng cách lưu trữ và xử lý dữ liệu bằng hệ quản trị cơ sở không quan hệ MongoDB.

### **4.1.2 Thực hành**

* Xây dựng được ứng dụng trên nền tảng web giúp cán bộ có thể dễ dàng thao tác trong quá trình phân chia phách tới người chấm thi.
* Cung cấp được các API cần thiết để thao tác với giao diện của ứng dụng.
* Tích hợp được tính năng gửi Email mỗi khi cán bộ chấm thi được phân công chấm bài.

## **4.1 Hạn chế**

* Giao diện của ứng dụng còn khá đơn giản và chưa có quá nhiều chức năng.
* Chưa hỗ trợ các thiết bị có màn hình độ phân giải thấp và trình duyệt IE.

## **4.1 Hướng phát triển**

* Xây dựng thêm tính năng trò chuyện và thảo luận trực tiếp giữa người chấm với cán bộ quản lý và người chấm với người chấm.
* Thêm phần thông báo thời gian thực trên hệ thống.
* Thực hiện triển khai trên nhiều loại màn hình độ phân giải khác nhau.

# TÀI LIỆU THAM KHẢO

[1] "Web Application," [Online]. Available: https://websitechuyennghiep.vn/vai-tro-cua-website-doi-voi-doanh-nghiep.html.

[2] "React" [Online]. Available: https://reactjs.org/docs/getting-started.html.

[3] "Express" [Online]. Available: https://en.wikipedia.org/wiki/Express.js..

[4] "NodeJS" [Online]. Available: https://en.wikipedia.org/wiki/Node.js.

[5] "MongoDB" [Online]. Available: https://en.wikipedia.org/wiki/MongoDB.

[6] "MongoDB" [Online]. Available: https://www.mongodb.com/1.

[7] TopDev, "topdev.vn," [Online]. Available: https://topdev.vn/blog/restful-api-la-gi/.