TRƯỜNG KỸ THUẬT VÀ CÔNG NGHỆ

KHOA CÔNG NGHỆ THÔNG TIN

BÁO CÁO MÔN HỌC

PHÁT TRIỂN ỨNG DỤNG HƯỚNG DỊCH VỤ

HỌC KỲ I, NĂM HỌC 2025-2026

XÂY DỰNG HỆ THỐNG ĐẶT LỊCH SÂN THỂ THAO

Giảng viên hướng dẫn:
ThS. Trịnh Quốc Việt

Sinh viên thực hiện:
Lê Khánh Đăng - 110122047
Châu Gia Bảo - 110122034
Nguyễn Tấn Lợi - 110122014
Lớp: DA22TTA

Vĩnh Long, tháng năm 2026

NHẬN XÉT CỦA GIẢNG VIÊN HƯỚNG DẪN

....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....

Vĩnh Long, ngày ... tháng ... năm ……

Giảng viên hướng dẫn

(Ký tên và ghi rõ họ tên)

NHẬN XÉT CỦA THÀNH VIÊN HỘI ĐỒNG

....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....

Vĩnh Long, ngày ..... tháng ...... năm ......

Thành viên hội đồng

(Ký tên và ghi rõ họ tên)

LỜI CẢM ƠN

Nhóm chúng em xin chân thành cảm ơn quý thầy cô Khoa Công nghệ Thông tin đã tận tình giảng dạy, truyền đạt những kiến thức chuyên môn và kỹ năng cần thiết trong suốt quá trình học tập. Những kiến thức này là nền tảng quan trọng giúp nhóm có thể thực hiện và hoàn thành đồ án kết thúc môn.

Nhóm chúng em xin bày tỏ lòng biết ơn sâu sắc đến thầy Trịnh Quốc Việt - giảng viên hướng dẫn, người đã luôn tận tâm chỉ dẫn, góp ý và định hướng trong quá trình thực hiện đồ án. Những nhận xét và hướng dẫn của thầy đã giúp nhóm khắc phục hạn chế, hoàn thiện nội dung và nâng cao chất lượng bài làm.

Cuối cùng, nhóm chúng em xin cảm ơn gia đình và bạn bè đã luôn quan tâm, động viên và tạo điều kiện thuận lợi để nhóm có thể yên tâm học tập và hoàn thành đồ án đúng tiến độ.

Nhóm chúng em xin chân thành cảm ơn.

Sinh viên 1 **Sinh viên 2** **Sinh viên 3**

MỤC LỤC

DANH MỤC HÌNH ẢNH

DANH MỤC BẢNG BIỂU

CHƯƠNG 1: MỞ ĐẦU

1.1. Lý do chọn đề tài

Trong những năm gần đây, nhu cầu rèn luyện sức khỏe và tham gia các hoạt động thể thao ngày càng gia tăng, đặc biệt là các môn thể thao như bóng đá, cầu lông, tennis,… kéo theo nhu cầu sử dụng và đặt lịch sân thể thao ngày càng phổ biến. Tuy nhiên, trên thực tế, việc quản lý và đặt lịch sân tại nhiều cơ sở vẫn còn được thực hiện thủ công thông qua sổ sách hoặc liên hệ trực tiếp, dẫn đến nhiều hạn chế như mất thời gian, dễ xảy ra trùng lịch, khó quản lý thông tin khách hàng và thiếu tính chuyên nghiệp.

Bên cạnh đó, sự phát triển mạnh mẽ của công nghệ thông tin và các ứng dụng web đã tạo điều kiện thuận lợi cho việc xây dựng các hệ thống quản lý trực tuyến, giúp tự động hóa quy trình đặt lịch, thanh toán và quản lý sân bãi một cách hiệu quả. Việc áp dụng công nghệ vào lĩnh vực quản lý sân thể thao không chỉ giúp nâng cao trải nghiệm người dùng mà còn hỗ trợ chủ sân trong việc quản lý, thống kê và vận hành hệ thống một cách khoa học.

Xuất phát từ những nhu cầu thực tiễn trên, nhóm quyết định lựa chọn đề tài "Xây dựng hệ thống đặt lịch sân thể thao" nhằm nghiên cứu và phát triển một hệ thống ứng dụng công nghệ thông tin vào việc quản lý và đặt lịch sân, góp phần nâng cao hiệu quả hoạt động và đáp ứng nhu cầu ngày càng cao của người sử dụng.

1.2. Mục đích nghiên cứu

Mục đích của đề tài là nghiên cứu, thiết kế và xây dựng một hệ thống đặt lịch sân thể thao trực tuyến nhằm:

- Hỗ trợ người dùng dễ dàng xem thông tin sân, giá thuê, khung giờ trống và thực hiện đặt lịch nhanh chóng.

- Giúp chủ sân và người quản trị quản lý hiệu quả thông tin sân bãi, lịch đặt sân, khách hàng và thanh toán.

- Hạn chế tình trạng trùng lịch, sai sót trong quá trình đặt sân.

- Nâng cao tính chuyên nghiệp, tiện lợi và hiện đại trong hoạt động quản lý sân thể thao.

- Áp dụng kiến thức đã học về lập trình, cơ sở dữ liệu và xây dựng hệ thống thông tin vào thực tế.

1.3. Đối tượng nghiên cứu

Đối tượng nghiên cứu của đề tài bao gồm:

- Các quy trình nghiệp vụ liên quan đến việc quản lý và đặt lịch sân thể thao.

- Người dùng hệ thống, bao gồm: khách hàng đặt sân và người quản trị (admin).

- Các công nghệ và công cụ phục vụ xây dựng hệ thống như: hệ quản trị cơ sở dữ liệu, công nghệ web, API và các kỹ thuật xử lý dữ liệu.

- Dữ liệu liên quan đến sân thể thao, khung giờ, lịch đặt sân, thanh toán và đánh giá.

1.4. Phạm vi nghiên cứu

Trong khuôn khổ đề tài, phạm vi nghiên cứu được giới hạn như sau:

- Hệ thống tập trung vào việc quản lý và đặt lịch các sân thể thao phổ biến như sân bóng đá (sân 5 người, 7 người, 11 người).

- Hệ thống được xây dựng dưới dạng ứng dụng web, cho phép người dùng truy cập và sử dụng thông qua trình duyệt.

- Chức năng chính bao gồm: đăng ký/đăng nhập, xem thông tin sân, đặt lịch theo ngày và khung giờ, thanh toán, quản lý sân và đánh giá chất lượng sân.

- Đề tài không đi sâu vào các vấn đề về phần cứng, thiết bị IoT hay các hệ thống thanh toán phức tạp ngoài phạm vi nghiên cứu.

- Dữ liệu và chức năng của hệ thống phục vụ cho mục đích học tập, nghiên cứu và mô phỏng hoạt động thực tế.

CHƯƠNG 2: NGHIÊN CỨU LÝ THUYẾT

2.1. Giới thiệu về ReactJS

2.1.1. ReactJS là gì ?

ReactJS là một thư viện JavaScript mã nguồn mở do Facebook (nay là Meta) phát triển, được sử dụng để xây dựng giao diện người dùng (User Interface - UI) cho các ứng dụng web hiện đại. ReactJS cho phép xây dựng giao diện theo hướng component-based, trong đó giao diện được chia thành các thành phần nhỏ, độc lập và có khả năng tái sử dụng cao.

ReactJS thường được sử dụng để phát triển các ứng dụng web đơn trang (Single Page Application - SPA), nơi toàn bộ ứng dụng chỉ tải một lần ban đầu, sau đó các nội dung được cập nhật động mà không cần tải lại toàn bộ trang. Điều này giúp cải thiện tốc độ phản hồi và nâng cao trải nghiệm người dùng, đặc biệt phù hợp với các hệ thống có nhiều thao tác tương tác như xem thông tin sân, lựa chọn khung giờ và đặt lịch trực tuyến [1], [2].

2.1.2. Virtual DOM

Document Object Model (DOM) là một cấu trúc dạng cây biểu diễn nội dung của tài liệu HTML, cho phép trình duyệt thao tác và cập nhật giao diện khi dữ liệu thay đổi. Trong các ứng dụng web truyền thống, mỗi lần dữ liệu thay đổi, trình duyệt sẽ thao tác trực tiếp lên DOM thật, điều này có thể gây tốn tài nguyên và làm giảm hiệu năng khi ứng dụng có quy mô lớn.

Để khắc phục hạn chế này, ReactJS sử dụng Virtual DOM, là một bản sao nhẹ của DOM thật được lưu trữ trong bộ nhớ. Khi trạng thái (state) hoặc dữ liệu của ứng dụng thay đổi, ReactJS sẽ tạo ra một Virtual DOM mới và so sánh với Virtual DOM trước đó thông qua thuật toán so sánh (diffing algorithm). Sau đó, ReactJS chỉ cập nhật những phần tử thực sự thay đổi lên DOM thật thông qua quá trình gọi là reconciliation.

Cơ chế Virtual DOM giúp giảm số lần thao tác trực tiếp lên DOM thật, từ đó cải thiện hiệu năng và đảm bảo giao diện được cập nhật nhanh chóng. Điều này đặc biệt quan trọng đối với hệ thống đặt lịch sân thể thao, nơi giao diện cần cập nhật liên tục các trạng thái như khung giờ trống, trạng thái đặt sân và thông tin người dùng.

2.1.3. JSX

JSX (JavaScript XML) là một cú pháp mở rộng của JavaScript, cho phép lập trình viên viết mã có cấu trúc tương tự HTML ngay trong mã JavaScript [1]. JSX không phải là HTML thuần mà sẽ được biên dịch thành JavaScript thông thường thông qua các công cụ như Babel trước khi trình duyệt thực thi.

Việc sử dụng JSX giúp kết hợp logic xử lý và giao diện người dùng trong cùng một component, giúp mã nguồn dễ đọc, dễ hiểu và thuận tiện cho việc bảo trì. Trong JSX, lập trình viên có thể nhúng các biểu thức JavaScript thông qua dấu ngoặc nhọn {}, cho phép hiển thị dữ liệu động như thông tin sân, giá thuê, khung giờ và trạng thái đặt lịch.

JSX là một thành phần quan trọng trong ReactJS, giúp đơn giản hóa quá trình xây dựng giao diện và nâng cao tính trực quan của mã nguồn trong các ứng dụng web hiện đại.

2.1.4. Components

Component là đơn vị cơ bản trong ReactJS, đại diện cho một phần giao diện người dùng cụ thể [1], [2]. Mỗi component có thể bao gồm cấu trúc giao diện, logic xử lý và dữ liệu liên quan. Thay vì xây dựng giao diện dưới dạng một khối lớn, ReactJS cho phép chia giao diện thành nhiều component nhỏ, độc lập và có thể tái sử dụng.

Trong ReactJS, mỗi component được xem như một hàm hoặc một lớp JavaScript, có nhiệm vụ nhận dữ liệu đầu vào và trả về giao diện tương ứng. Việc chia ứng dụng thành các component giúp mã nguồn trở nên rõ ràng, dễ quản lý và dễ mở rộng khi hệ thống phát triển.

Component trong ReactJS có thể được phân loại thành hai loại chính:

- Functional Component: là các hàm JavaScript đơn giản, thường được sử dụng trong các phiên bản ReactJS hiện đại. Functional Component dễ viết, dễ hiểu và kết hợp tốt với các cơ chế quản lý trạng thái.

- Class Component: là các lớp JavaScript kế thừa từ lớp React.Component. Loại component này được sử dụng phổ biến trong các phiên bản ReactJS cũ, hiện nay ít được sử dụng hơn trong các dự án mới.

Một trong những ưu điểm lớn của component là khả năng tái sử dụng. Một component có thể được sử dụng nhiều lần trong các vị trí khác nhau của ứng dụng, giúp giảm sự trùng lặp mã nguồn và tăng hiệu quả phát triển. Ngoài ra, việc chia giao diện thành các component nhỏ giúp dễ dàng kiểm thử và bảo trì hệ thống.

2.1.5. Props & State

Trong ReactJS, Props (Properties) và State là hai khái niệm cốt lõi được sử dụng để quản lý và truyền dữ liệu giữa các component [1]. Việc hiểu rõ sự khác nhau giữa props và state giúp xây dựng ứng dụng có cấu trúc rõ ràng, dễ bảo trì và mở rộng.

Props là các tham số được truyền từ component cha xuống component con nhằm cung cấp dữ liệu hoặc cấu hình cho component. Props có tính chất chỉ đọc, nghĩa là component con không được phép thay đổi giá trị của props. Nhờ cơ chế này, ReactJS đảm bảo luồng dữ liệu một chiều, giúp việc kiểm soát và dự đoán trạng thái của ứng dụng trở nên đơn giản hơn.

State là dữ liệu nội bộ của component, được sử dụng để lưu trữ các thông tin có thể thay đổi trong quá trình tương tác với người dùng. Khi state thay đổi, ReactJS sẽ tự động cập nhật lại giao diện tương ứng thông qua cơ chế Virtual DOM. State thường được dùng để quản lý các trạng thái động như dữ liệu nhập từ biểu mẫu, trạng thái đăng nhập hoặc danh sách bài viết được cập nhật.

Sự khác biệt cơ bản giữa props và state có thể được tóm tắt như sau:

- Props được truyền từ component cha và không thể thay đổi bên trong component con.

- State do chính component quản lý và có thể thay đổi trong quá trình thực thi.

- Props giúp các component giao tiếp với nhau, trong khi state giúp quản lý dữ liệu nội bộ của component.

2.1.6. React Router

React Router là một thư viện phổ biến được sử dụng trong ReactJS để quản lý điều hướng (routing) giữa các trang trong ứng dụng web đơn trang (Single Page Application - SPA) [1]. Thay vì tải lại toàn bộ trang khi người dùng chuyển sang một trang khác, React Router cho phép thay đổi nội dung hiển thị dựa trên đường dẫn URL mà không làm gián đoạn trải nghiệm người dùng.

Trong các ứng dụng ReactJS, React Router đóng vai trò ánh xạ giữa đường dẫn URL và component tương ứng. Khi người dùng truy cập một URL cụ thể, React Router sẽ xác định component nào cần được hiển thị. Cơ chế này giúp ứng dụng có cấu trúc rõ ràng, dễ quản lý và mở rộng.

React Router cung cấp nhiều thành phần hỗ trợ điều hướng, trong đó phổ biến nhất là:

- BrowserRouter: quản lý lịch sử điều hướng của ứng dụng dựa trên URL.

- Route: định nghĩa mối quan hệ giữa một đường dẫn và component tương ứng.

- Link / NavLink: tạo liên kết điều hướng giữa các trang mà không cần tải lại.

- useParams, useNavigate: hỗ trợ lấy tham số từ URL và điều hướng.

Việc sử dụng React Router giúp tổ chức ứng dụng theo dạng nhiều trang logic trong cùng một ứng dụng SPA, đồng thời đảm bảo tính thân thiện với người dùng và dễ phát triển. Ngoài ra, React Router còn hỗ trợ truyền tham số trên URL, giúp hiển thị các nội dung động như trang chi tiết bài viết hoặc trang hồ sơ người dùng.

Ngoài ra, ReactJS có hệ sinh thái phong phú với nhiều thư viện hỗ trợ như React Router, Redux, Axios,… cùng cộng đồng lớn, giúp lập trình viên dễ dàng tìm kiếm tài liệu và giải pháp trong quá trình phát triển.

2.1.7. Ưu điểm

ReactJS mang lại nhiều ưu điểm nổi bật trong quá trình phát triển các ứng dụng web hiện đại:

- Áp dụng mô hình component-based, giúp giao diện rõ ràng và dễ tái sử dụng.

- Virtual DOM giúp tối ưu hiệu năng và tăng tốc độ cập nhật giao diện.

- Hỗ trợ tốt việc xây dựng ứng dụng SPA, nâng cao trải nghiệm người dùng.

- Cộng đồng lớn và hệ sinh thái phong phú, dễ dàng mở rộng chức năng.

Những ưu điểm này giúp ReactJS trở thành lựa chọn phù hợp cho việc xây dựng giao diện hệ thống đặt lịch sân thể thao.

2.1.8. Nhược điểm

Bên cạnh các ưu điểm, ReactJS cũng tồn tại một số hạn chế:

- ReactJS chỉ là thư viện giao diện, cần kết hợp thêm các thư viện khác để xây dựng ứng dụng hoàn chỉnh.

- JSX có thể gây khó khăn cho người mới bắt đầu.

- Yêu cầu kiến thức JavaScript ES6+ tương đối tốt.

Tuy nhiên, với việc học tập và thực hành, các hạn chế này có thể được khắc phục và không làm giảm hiệu quả của ReactJS trong việc phát triển các ứng dụng web hiện đại.

2.2. Giới thiệu về Node.js

2.2.1. NodeJS là gì?

NodeJS là một nền tảng (runtime environment) cho phép thực thi mã JavaScript ở phía máy chủ, được xây dựng dựa trên V8 JavaScript Engine của Google. Trước khi NodeJS ra đời, JavaScript chủ yếu chỉ được sử dụng ở phía trình duyệt để xử lý giao diện người dùng. Sự xuất hiện của NodeJS đã mở rộng phạm vi sử dụng của JavaScript, cho phép lập trình viên xây dựng các ứng dụng phía máy chủ, các dịch vụ web và API một cách hiệu quả.

Một đặc điểm quan trọng của NodeJS là việc sử dụng mô hình bất đồng bộ (asynchronous) và hướng sự kiện (event - driven). Theo mô hình này, các tác vụ như truy vấn cơ sở dữ liệu hoặc xử lý yêu cầu từ người dùng không làm chặn luồng xử lý chính, mà được thực hiện thông qua các cơ chế như callback, promise hoặc async/await. Nhờ đó, NodeJS có khả năng xử lý nhiều yêu cầu đồng thời với hiệu năng cao.

Mặc dù NodeJS hoạt động theo mô hình single-threaded, nhưng nó sử dụng cơ chế event loop để quản lý và điều phối các tác vụ bất đồng bộ. Cơ chế này cho phép NodeJS xử lý hàng nghìn kết nối cùng lúc mà không cần tạo nhiều luồng xử lý riêng biệt, từ đó giúp tiết kiệm tài nguyên hệ thống và nâng cao hiệu suất hoạt động.

Trong kiến trúc ứng dụng web, NodeJS thường được sử dụng để xây dựng phần backend của hệ thống. NodeJS tiếp nhận các yêu cầu từ phía client, xử lý logic nghiệp

vụ, tương tác với cơ sở dữ liệu và trả kết quả về cho client thông qua các API. Với hiệu năng cao, khả năng mở rộng tốt và cộng đồng hỗ trợ lớn, NodeJS phù hợp cho các ứng dụng web có lượng truy cập lớn và yêu cầu thời gian phản hồi nhanh [3].

2.2.2. Giới thiệu về Express.js

Express.js là một framework mã nguồn mở, nhẹ và phổ biến được xây dựng trên nền tảng NodeJS, nhằm hỗ trợ việc phát triển các ứng dụng web và dịch vụ API một cách nhanh chóng và hiệu quả. Express.js cung cấp các công cụ và cấu trúc cần thiết để xử lý các yêu cầu HTTP, định tuyến (routing) và quản lý middleware, giúp đơn giản hóa quá trình xây dựng hệ thống backend [4].

Một trong những đặc điểm nổi bật của Express.js là kiến trúc middleware, cho phép xử lý các yêu cầu theo từng bước thông qua chuỗi middleware. Mỗi middleware đảm nhiệm một chức năng cụ thể như xác thực người dùng, kiểm tra dữ liệu đầu vào, xử lý lỗi hoặc ghi log hệ thống. Cách tiếp cận này giúp mã nguồn rõ ràng, dễ quản lý và dễ mở rộng khi bổ sung các chức năng mới.

Express.js hỗ trợ mạnh mẽ việc xây dựng RESTful API, cho phép định nghĩa các endpoint tương ứng với các phương thức HTTP như GET, POST, PUT và DELETE. Điều này rất phù hợp với kiến trúc client - server, trong đó frontend (ReactJS) giao tiếp với backend thông qua các API. Nhờ Express.js, việc tổ chức và quản lý các API trở nên linh hoạt và hiệu quả hơn.

2.2.3. Ưu điểm

NodeJS mang lại nhiều ưu điểm nổi bật trong việc xây dựng các hệ thống backend hiện đại. Nhờ sử dụng mô hình bất đồng bộ (asynchronous) và hướng sự kiện (event - driven), NodeJS có khả năng xử lý đồng thời nhiều yêu cầu với hiệu năng cao, phù hợp cho các ứng dụng web có lượng truy cập lớn. Việc sử dụng cùng một ngôn ngữ JavaScript cho cả frontend và backend giúp giảm độ phức tạp trong quá trình phát triển và tăng tính nhất quán của hệ thống. Ngoài ra, NodeJS sở hữu cộng đồng lớn và hệ sinh thái phong phú, hỗ trợ nhiều thư viện và công cụ phục vụ phát triển ứng dụng.

Express.js là một framework nhẹ, linh hoạt, giúp đơn giản hóa việc xây dựng backend trên nền NodeJS. Express.js hỗ trợ mạnh mẽ việc phát triển RESTful API, cung cấp cơ chế routing rõ ràng và hệ thống middleware linh hoạt. Nhờ đó, mã nguồn

backend trở nên dễ đọc, dễ bảo trì và dễ mở rộng khi bổ sung thêm các chức năng mới như xác thực, ghi log hay xử lý lỗi.

2.2.4. Nhược điểm

Bên cạnh các ưu điểm, NodeJS tồn tại một số hạn chế. Do hoạt động theo mô hình single - threaded, NodeJS không phù hợp cho các tác vụ tính toán nặng (CPU - intensive), vì các tác vụ này có thể làm chậm event loop và ảnh hưởng đến hệ thống. Ngoài ra, việc lập trình bất đồng bộ đòi hỏi lập trình viên phải nắm vững các khái niệm như callback, promise và async/await để tránh lỗi và khó khăn trong việc debug.

Express.js mặc dù nhẹ và linh hoạt, nhưng không cung cấp một cấu trúc chặt chẽ như các framework lớn. Điều này khiến lập trình viên cần tự tổ chức cấu trúc dự án và lựa chọn các thư viện bổ trợ phù hợp, có thể gây khó khăn cho người mới bắt đầu.d

2.3. Giới thiệu về MongoDB

MongoDB là một hệ quản trị cơ sở dữ liệu NoSQL mã nguồn mở, được thiết kế để lưu trữ và quản lý dữ liệu dưới dạng document theo định dạng JSON (Binary JSON - BSON). Khác với các hệ quản trị cơ sở dữ liệu quan hệ truyền thống sử dụng bảng, hàng và cột, MongoDB tổ chức dữ liệu theo các collection và document, giúp việc lưu trữ và truy xuất dữ liệu trở nên linh hoạt hơn.

2.3.1. Đặc điểm của MongoDB

Một số đặc điểm nổi bật của MongoDB bao gồm:

- Lưu trữ dữ liệu dạng document: Dữ liệu được lưu trữ dưới dạng document có cấu trúc gần giống JSON, dễ đọc và dễ thao tác trong các ứng dụng web.

- Không yêu cầu schema cố định: MongoDB cho phép các document trong cùng một collection có cấu trúc khác nhau, phù hợp với các hệ thống có dữ liệu thay đổi thường xuyên.

- Khả năng mở rộng cao: MongoDB hỗ trợ mở rộng theo chiều ngang (horizontal scaling), đáp ứng tốt khi dữ liệu và số lượng người dùng tăng.

- Hiệu năng tốt: MongoDB được tối ưu cho các thao tác đọc/ghi nhanh, phù hợp với các ứng dụng web có lượng truy cập lớn.

Các đặc điểm trên cho thấy MongoDB phù hợp với các hệ thống web hiện đại có dữ liệu lớn, linh hoạt và yêu cầu khả năng mở rộng cao [6].

2.3.2. MongoDB trong kiến trúc ứng dụng web

Trong kiến trúc ứng dụng web hiện đại, MongoDB thường được sử dụng làm cơ sở dữ liệu phía máy chủ, kết hợp với NodeJS để xây dựng hệ thống backend. MongoDB cho phép lưu trữ dữ liệu dưới dạng JSON, rất tương thích với JavaScript - ngôn ngữ chính được sử dụng trong NodeJS và ReactJS.

Việc kết hợp MongoDB với NodeJS giúp quá trình trao đổi dữ liệu giữa backend và database trở nên đơn giản và hiệu quả, giảm bớt sự phức tạp trong việc chuyển đổi dữ liệu.

2.3.3. Ưu và nhược điểm của MongoDB

Ưu điểm của MongoDB:

- Linh hoạt trong thiết kế cấu trúc dữ liệu.

- Dễ dàng mở rộng khi quy mô hệ thống tăng.

- Tương thích tốt với các công nghệ web hiện đại.

- Hiệu năng cao đối với các ứng dụng có dữ liệu lớn.

Nhược điểm của MongoDB:

- Không phù hợp với các hệ thống yêu cầu tính toàn vẹn quan hệ phức tạp.

- Việc quản lý dữ liệu không có schema cố định đòi hỏi thiết kế cẩn thận để tránh dữ liệu không nhất quán.

- Cần cấu hình và tối ưu hợp lý để đạt hiệu suất tốt nhất.

2.4. Phương pháp nghiên cứu

2.4.1. Phương pháp nghiên cứu tài liệu

Phương pháp nghiên cứu tài liệu được sử dụng để thu thập, phân tích và tổng hợp các tài liệu liên quan đến lĩnh vực nghiên cứu. Các nguồn tài liệu bao gồm giáo trình, sách chuyên ngành, bài báo khoa học và tài liệu kỹ thuật về các công nghệ như ReactJS, NodeJS, Express.js và MongoDB.

Việc nghiên cứu tài liệu giúp đề tài xây dựng cơ sở lý thuyết vững chắc, làm nền tảng cho quá trình phân tích yêu cầu và lựa chọn giải pháp công nghệ phù hợp.

2.4.2. Phương pháp phân tích yêu cầu

Phương pháp phân tích yêu cầu được sử dụng nhằm xác định rõ các chức năng và yêu cầu cần thiết của hệ thống đặt lịch sân thể thao. Quá trình này được thực hiện thông qua việc khảo sát thực tế hoạt động quản lý sân thể thao, phân tích nhu cầu của người dùng và chủ sân, từ đó xác định các chức năng chính của hệ thống.

Cụ thể, đề tài tiến hành:

- Phân tích yêu cầu chức năng của hệ thống như: đăng ký/đăng nhập, xem thông tin sân, đặt lịch theo khung giờ, thanh toán, quản lý sân và đánh giá chất lượng sân.

- Phân tích yêu cầu phi chức năng như: tính ổn định, bảo mật, khả năng mở rộng và tính thân thiện với người dùng.

- Xác định các đối tượng tham gia hệ thống (người dùng và quản trị viên) cùng quyền hạn tương ứng.

Kết quả của phương pháp phân tích yêu cầu là cơ sở quan trọng để xây dựng mô hình hệ thống và thiết kế các thành phần chức năng phù hợp.

2.4.3. Phương pháp thiết kế hệ thống

Trên cơ sở các yêu cầu đã phân tích, đề tài tiến hành thiết kế hệ thống theo mô hình client - server. Các nội dung thiết kế bao gồm:

- Thiết kế kiến trúc tổng thể của hệ thống.

- Thiết kế cơ sở dữ liệu MongoDB.

- Thiết kế giao diện người dùng bằng ReactJS.

- Thiết kế các API phía backend bằng NodeJS và Express.js.

Phương pháp thiết kế giúp hệ thống có cấu trúc rõ ràng, dễ triển khai, bảo trì và mở rộng trong tương lai.

2.4.4. Phương pháp thực nghiệm

Phương pháp thực nghiệm được sử dụng để kiểm tra tính đúng đắn và hiệu quả của hệ thống sau khi xây dựng. Trong quá trình này, hệ thống được triển khai và chạy

thử nghiệm trên môi trường thực tế hoặc mô phỏng, nhằm kiểm tra các chức năng đã thiết kế.

Các nội dung thực nghiệm bao gồm:

- Thử nghiệm các chức năng chính như đặt lịch sân, kiểm tra trùng lịch, thanh toán và quản lý sân.

- Kiểm tra khả năng xử lý dữ liệu và phản hồi của hệ thống khi có nhiều người dùng truy cập.

- Kiểm tra việc lưu trữ và hiển thị dữ liệu, bao gồm cả hình ảnh sân thể thao.

Kết quả thực nghiệm giúp phát hiện các lỗi phát sinh và đánh giá mức độ đáp ứng yêu cầu của hệ thống.

2.4.5. Phương pháp đánh giá và hoàn thiện

Sau khi hoàn thành quá trình thực nghiệm, hệ thống được đánh giá dựa trên các tiêu chí:

- Mức độ đáp ứng yêu cầu đề ra.

- Tính ổn định và bảo mật của hệ thống.

- Trải nghiệm người dùng và mức độ thân thiện của giao diện.

Dựa trên kết quả đánh giá, hệ thống được điều chỉnh và hoàn thiện nhằm nâng cao chất lượng và hiệu quả sử dụng.

2.4.6. Kết luận phương pháp nghiên cứu

Thông qua việc kết hợp các phương pháp nghiên cứu như nghiên cứu tài liệu, phân tích yêu cầu, thiết kế hệ thống, thực nghiệm và đánh giá, đề tài đã xây dựng được một quy trình nghiên cứu khoa học và hợp lý. Các phương pháp này hỗ trợ lẫn nhau, góp phần đảm bảo hệ thống đặt lịch sân thể thao được xây dựng đáp ứng đúng yêu cầu, có tính khả thi cao và phù hợp với thực tiễn ứng dụng.

CHƯƠNG 3: HIỆN THỰC HÓA NGHIÊN CỨU

3.1. Mô tả bài toán

Trong thực tế, việc quản lý và đặt lịch sân thể thao tại nhiều cơ sở hiện nay vẫn chủ yếu được thực hiện theo phương thức thủ công như ghi chép sổ sách hoặc trao đổi trực tiếp qua điện thoại. Cách làm này dễ dẫn đến các vấn đề như trùng lịch đặt sân, khó kiểm soát khung giờ trống, mất thời gian cho cả khách hàng và người quản lý, đồng thời thiếu tính chuyên nghiệp và khả năng thống kê, báo cáo.

Bài toán đặt ra là cần xây dựng một hệ thống đặt lịch sân thể thao trực tuyến cho phép người dùng dễ dàng xem thông tin sân, khung giờ trống và thực hiện đặt lịch mọi lúc, mọi nơi. Đồng thời, hệ thống cần hỗ trợ người quản trị trong việc quản lý sân bãi, lịch đặt sân, thanh toán và theo dõi hoạt động của hệ thống một cách hiệu quả.

Hệ thống được xây dựng dưới dạng ứng dụng web theo mô hình client–server, sử dụng các công nghệ hiện đại nhằm đảm bảo tính ổn định, dễ mở rộng và phù hợp với nhu cầu thực tế.

3.2. Phân tích yêu cầu hệ thống

3.2.1. Yêu cầu chức năng

Hệ thống bao gồm hai nhóm người dùng chính: Người dùng (User) và Quản trị viên (Admin), với các chức năng cụ thể như sau:

Đối với người dùng:

- Đăng ký và đăng nhập vào hệ thống.

- Xem danh sách các sân thể thao và thông tin chi tiết của từng sân (loại sân, giá thuê, hình ảnh).

- Xem các khung giờ trống theo ngày.

- Thực hiện đặt lịch sân theo ngày và khung giờ.

- Thanh toán chi phí thuê sân theo các phương thức hỗ trợ.

- Xem lịch sử đặt sân của cá nhân.

- Hủy lịch đặt sân trong thời gian cho phép.

- Đánh giá và nhận xét chất lượng sân sau khi sử dụng.

Đối với quản trị viên:

- Quản lý thông tin khu sân và sân thể thao (thêm, sửa, xóa).

- Quản lý hình ảnh sân thể thao.

- Quản lý khung giờ hoạt động của sân.

- Quản lý các đơn đặt lịch sân.

- Quản lý tài khoản người dùng.

- Thống kê và báo cáo tình hình hoạt động của hệ thống.

3.2.2. Yêu cầu phi chức năng

Bên cạnh các yêu cầu chức năng, hệ thống cần đáp ứng các yêu cầu phi chức năng sau:

- Tính ổn định: Hệ thống phải hoạt động ổn định, hạn chế lỗi trong quá trình sử dụng.

- Tính bảo mật: Bảo vệ thông tin người dùng, sử dụng cơ chế xác thực và phân quyền truy cập.

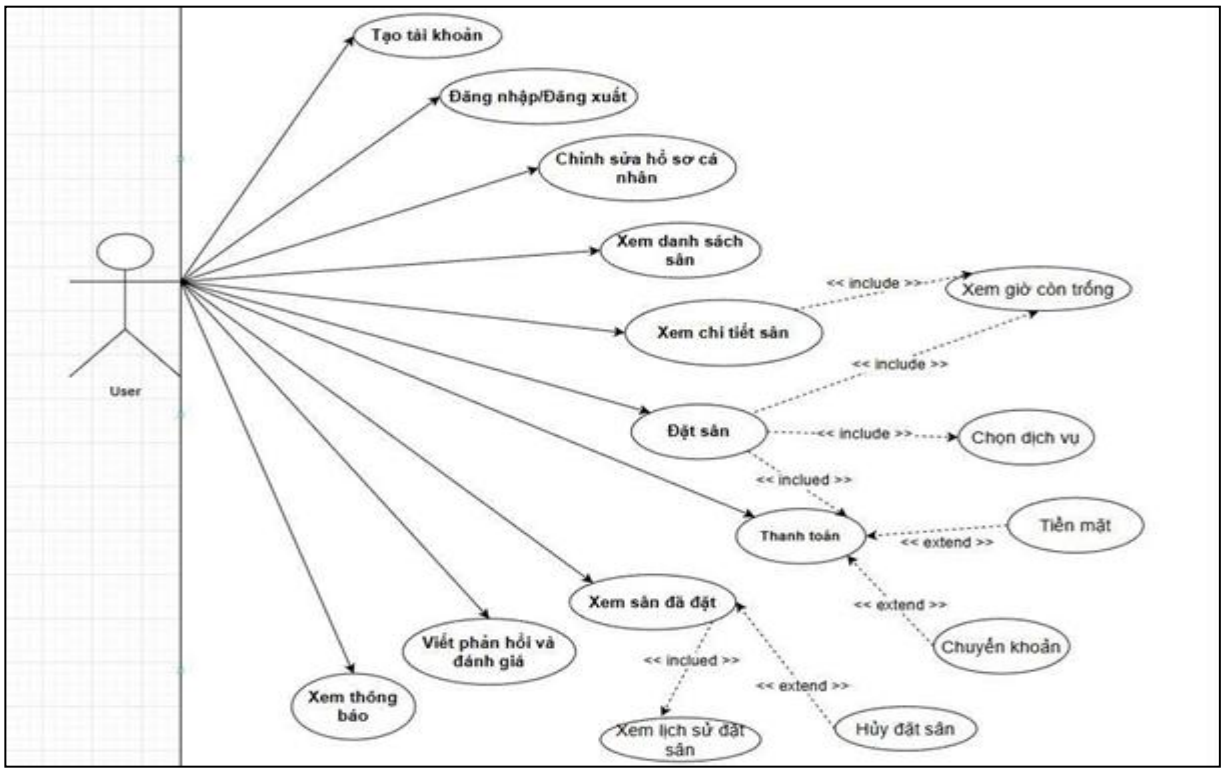
- Hiệu năng: Thời gian phản hồi nhanh, xử lý tốt khi có nhiều người dùng truy cập đồng thời.

- Khả năng mở rộng: Dễ dàng nâng cấp, mở rộng thêm các chức năng hoặc loại sân trong tương lai.

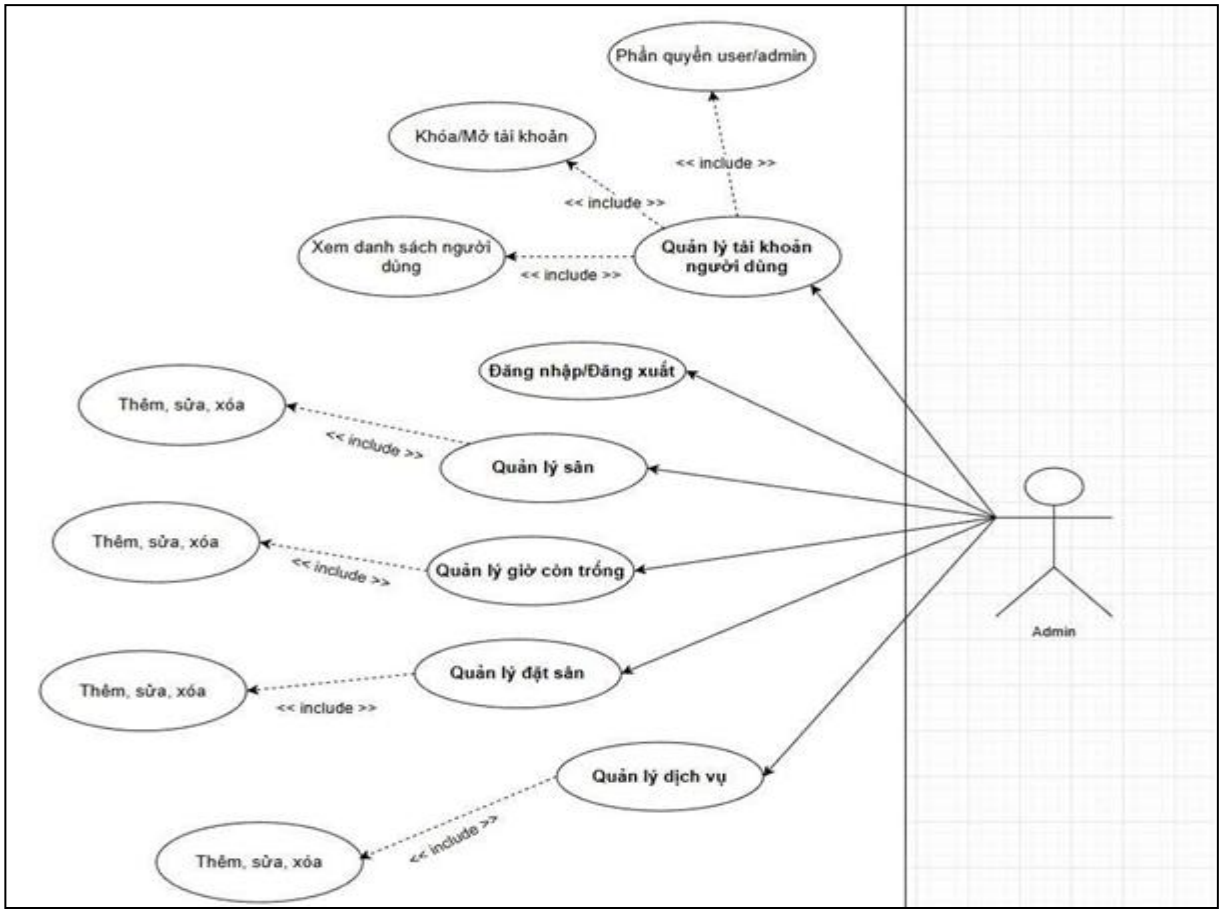
- Tính thân thiện: Giao diện trực quan, dễ sử dụng đối với mọi đối tượng người dùng.

- Khả năng bảo trì: Mã nguồn rõ ràng, dễ bảo trì và sửa lỗi.

3.2.3. Sơ đồ Use Case



Hình 3.1. Sơ đồ Use Case người dùng



Hình 3.2. Sơ đồ Use Case quản trị

3.3. Thiết kế hệ thống

3.3.1. Thiết kế kiến trúc tổng thể hệ thống

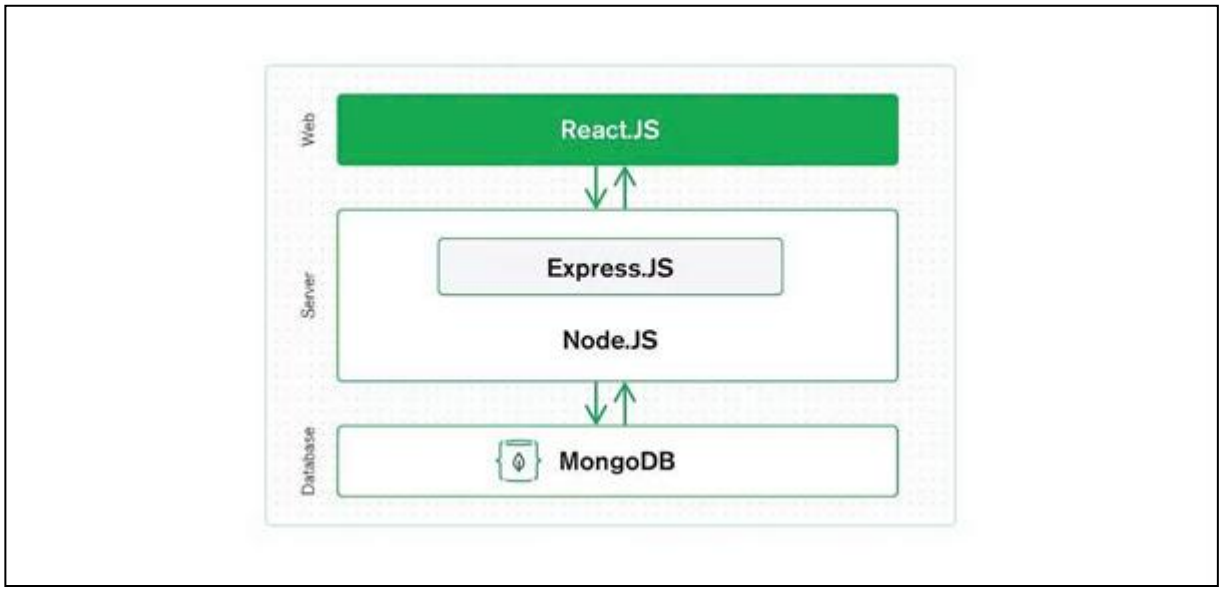
Hệ thống đặt lịch sân thể thao được thiết kế theo mô hình Client - Server, trong đó các thành phần được phân tách rõ ràng nhằm đảm bảo tính linh hoạt, dễ bảo trì và mở rộng trong tương lai.

Frontend (Client): Được xây dựng bằng ReactJS, chịu trách nhiệm hiển thị giao diện người dùng, tiếp nhận các thao tác từ người dùng như đăng nhập, xem sân, đặt lịch và thanh toán.

Backend (Server): Được xây dựng bằng NodeJS kết hợp Express.js, đảm nhiệm xử lý nghiệp vụ, xác thực người dùng, quản lý dữ liệu và cung cấp các API cho frontend.

Cơ sở dữ liệu: MongoDB được sử dụng để lưu trữ dữ liệu người dùng, thông tin sân, lịch đặt sân, thanh toán và đánh giá.

Hệ thống lưu trữ hình ảnh: Hình ảnh sân thể thao được lưu trữ trên hệ thống file hoặc dịch vụ lưu trữ đám mây, trong khi MongoDB chỉ lưu đường dẫn (URL) của hình ảnh.



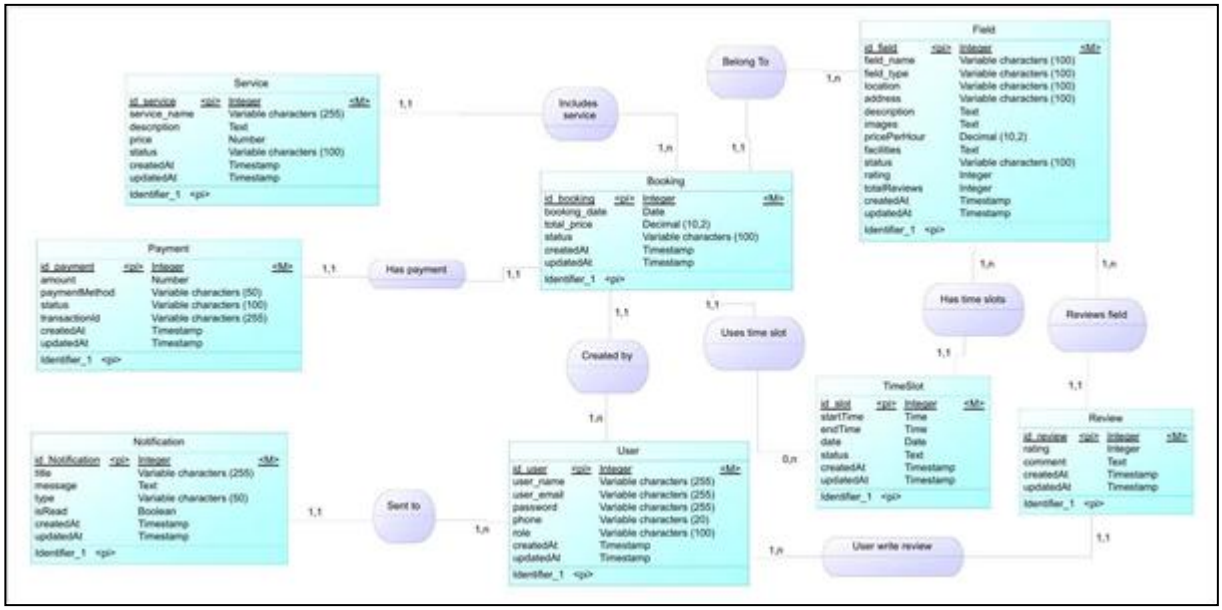
Hình 3.3. Sơ đồ kiến trúc tổng thể hệ thống

3.3.2. Thiết kế cơ sở dữ liệu

Hệ thống sử dụng MongoDB, một hệ quản trị cơ sở dữ liệu NoSQL, phù hợp với mô hình dữ liệu linh hoạt và dễ mở rộng. Các dữ liệu được tổ chức dưới dạng các collection, phản ánh các thực thể chính trong hệ thống đặt lịch sân thể thao.

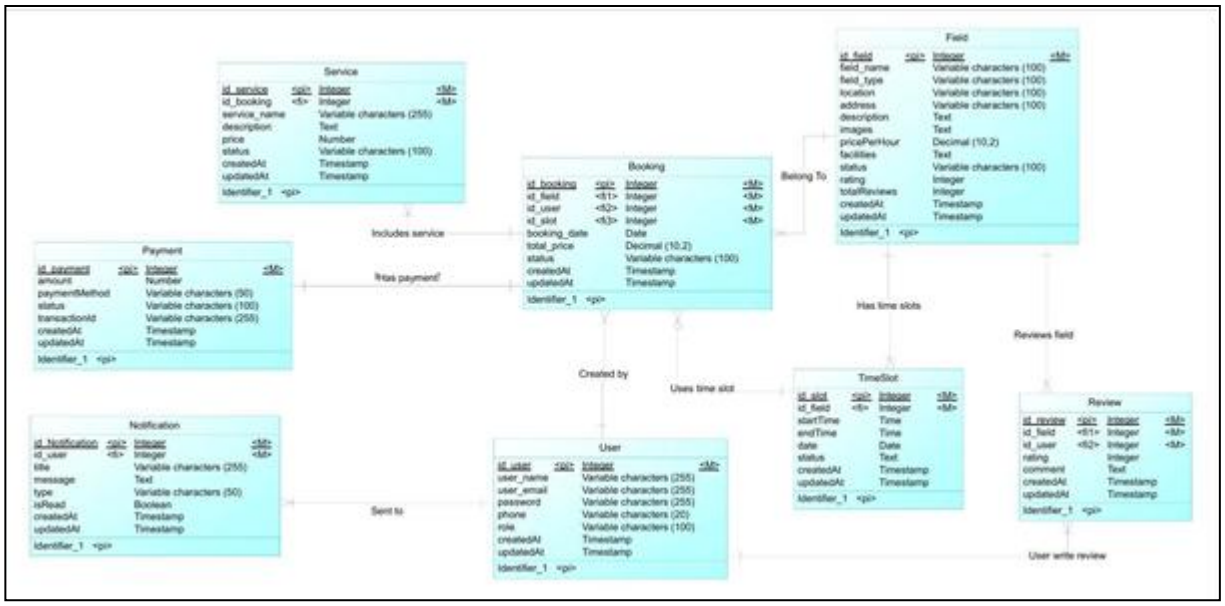
Mô hình ERD :

Mô hình quan niệm (Conceptual Model):



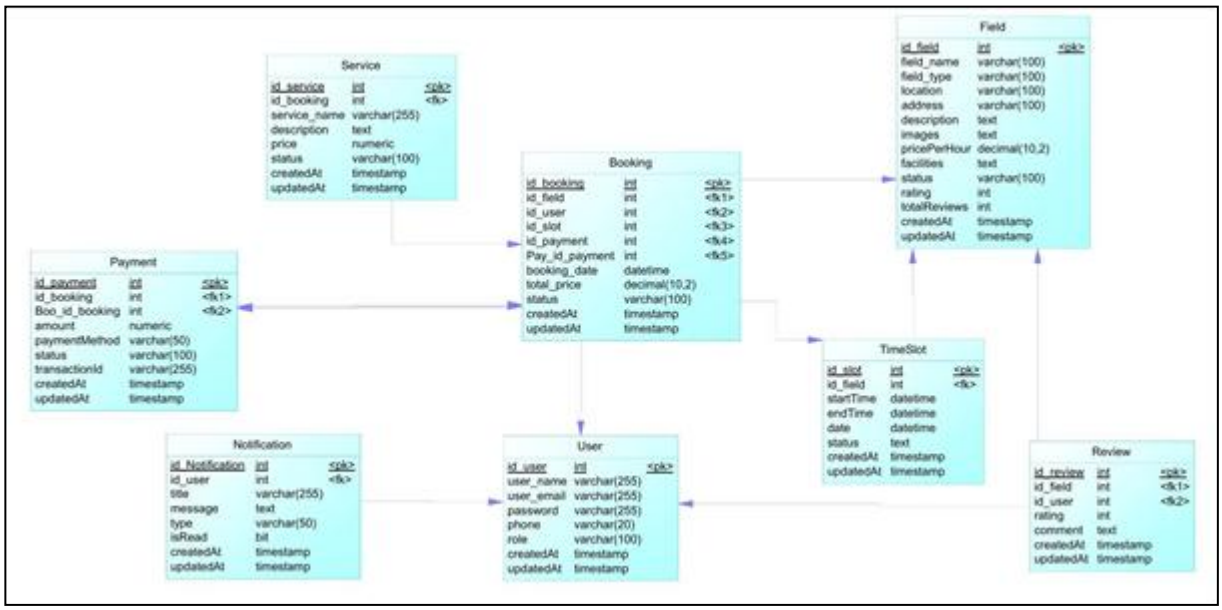
Hình 3.4. Mô hình quan niệm

Mô hình Logic (Logical Model):



Hình 3.5. Mô hình Logic

Mô hình Vật lý (Physical Model):



Hình 3.6. Mô hình vật lý

Bảng 3.1. Chi tiết các thực thể User

Thuộc tính	**Kiểu dữ liệu**	**Mô tả**
_id	ObjectId	Khóa chính
fullName	String	Họ và tên người dùng
email	String	Email đăng nhập (duy nhất)
password	String	Mật khẩu đã mã hóa
phone	String	Số điện thoại
role	String	Vai trò (customer, admin)
avatar	String	Ảnh đại diện

Bảng 3.2. Chi tiết các thực thể Field

Thuộc tính	Kiểu dữ liệu	Mô tả
_id	ObjectId	Khóa chính
name	String	Tên sân
fieldType	String	Loại sân (5vs5, 7vs7, 11vs11)
location	String	Khu vực
address	String	Địa chỉ chi tiết
description	String	Mô tả sân
images	Array[String]	Danh sách hình ảnh sân
pricePerHour	Number	Giá thuê theo giờ
facilities	Array[String]	Tiện ích (đèn, WC, bãi xe…)
status	String	Trạng thái sân
rating	Number	Điểm đánh giá trung bình

Bảng 3.3. Chi tiết các thực thể TimeSlot

Thuộc tính	Kiểu dữ liệu	Mô tả
_id	ObjectId	Khóa chính
field	ObjectId	Tham chiếu sân
date	Date	Ngày đặt sân
startTime	String	Giờ bắt đầu (HH:MM)
endTime	String	Giờ kết thúc (HH:MM)
price	Number	Giá theo khung giờ
status	String	Trạng thái (available, booked)

Bảng 3.4. Chi tiết các thực thể Booking

Thuộc tính	Kiểu dữ liệu	Mô tả
_id	ObjectId	Khóa chính
bookingCode	String	Mã đặt lịch tự động
user	ObjectId	Người đặt sân
field	ObjectId	Sân được đặt
timeSlot	ObjectId	Khung giờ
bookingDate	Date	Ngày đặt sân
startTime	String	Giờ bắt đầu
endTime	String	Giờ kết thúc
totalPrice	Number	Tổng tiền
services	Array	Dịch vụ đi kèm
status	String	Trạng thái đặt sân
paymentStatus	String	Trạng thái thanh toán
customerName	String	Tên người liên hệ
customerPhone	String	SĐT liên hệ
notes	String	Ghi chú
cancelReason	String	Lý do hủy
cancelledAt	Date	Thời điểm hủy

Bảng 3.5. Chi tiết các thực thể Payment

Thuộc tính	Kiểu dữ liệu	Mô tả
_id	ObjectId	Khóa chính
booking	ObjectId	Đơn đặt sân
user	ObjectId	Người thanh toán
amount	Number	Số tiền
paymentMethod	String	Phương thức thanh toán
transactionId	String	Mã giao dịch
status	String	Trạng thái thanh toán

Bảng 3.6. Chi tiết các thực thể Service

Thuộc tính	Kiểu dữ liệu	Mô tả
_id	ObjectId	Khóa chính
name	String	Tên dịch vụ
category	String	Loại dịch vụ
description	String	Mô tả
price	Number	Giá
unit	String	Đơn vị tính
image	String	Hình ảnh
stock	Number	Số lượng tồn

Bảng 3.7. Chi tiết các thực thể Review

Thuộc tính	Kiểu dữ liệu	Mô tả
_id	ObjectId	Khóa chính
user	ObjectId	Người đánh giá
field	ObjectId	Sân
booking	ObjectId	Đơn đặt sân
rating	Number	Số sao
comment	String	Nội dung đánh giá
images	Array[String]	Ảnh minh họa
isVerified	Boolean	Đánh giá xác thực
reply.content	String	Phản hồi admin
reply.createdAt	Date	Ngày phản hồi
likes	Number	Lượt thích

Bảng 3.8. Chi tiết các thực thể Notification

Thuộc tính	Kiểu dữ liệu	Mô tả
_id	ObjectId	Khóa chính
user	ObjectId	Người nhận
title	String	Tiêu đề
message	String	Nội dung
type	String	Loại thông báo
relatedId	ObjectId	ID liên quan

Thuộc tính	Kiểu dữ liệu	Mô tả
relatedModel	String	Model liên kết
isRead	Boolean	Đã đọc
readAt	Date	Thời gian đọc
Thuộc tính	Kiểu dữ liệu	Mô tả

3.4. Thiết kế, kiến trúc ứng dụng

3.4.1. Xây dựng backend

Backend của hệ thống đặt lịch sân thể thao được xây dựng bằng NodeJS kết hợp Express.js, đóng vai trò trung tâm trong việc xử lý nghiệp vụ, quản lý dữ liệu và cung cấp các dịch vụ API cho frontend. Backend được thiết kế theo kiến trúc RESTful API và tổ chức theo mô hình MVC (Model - Controller - Router) nhằm đảm bảo mã nguồn rõ ràng, dễ bảo trì và mở rộng.

Kiến trúc backend

Model: Sử dụng thư viện Mongoose để định nghĩa các schema và thao tác với cơ sở dữ liệu MongoDB. Các model đại diện cho các thực thể chính như User, Field, TimeSlot, Booking, Payment, Service, Review và Notification.

Controller: Chứa các hàm xử lý nghiệp vụ, tiếp nhận yêu cầu từ router, thực hiện các thao tác xử lý dữ liệu và trả kết quả về cho client.

Router: Định nghĩa các endpoint API, ánh xạ các yêu cầu HTTP (GET, POST, PUT, DELETE) đến controller tương ứng.

Các chức năng chính của backend

- Xác thực và phân quyền người dùng bằng JWT, đảm bảo chỉ những người dùng hợp lệ mới có quyền truy cập hệ thống.

- Quản lý thông tin sân thể thao, khung giờ và các dịch vụ đi kèm.

- Xử lý nghiệp vụ đặt lịch sân, kiểm tra trùng khung giờ và cập nhật trạng thái khung giờ.

- Quản lý thanh toán và cập nhật trạng thái đơn đặt sân.

- Quản lý đánh giá, phản hồi và gửi thông báo cho người dùng.

- Xử lý upload và quản lý hình ảnh sân thể thao.

- Bảo mật và kiểm soát truy cập

Backend áp dụng các biện pháp bảo mật cơ bản như:

Mã hóa mật khẩu người dùng.

- Sử dụng JWT để xác thực các yêu cầu API.

- Phân quyền truy cập giữa người dùng và quản trị viên.

- Kiểm tra dữ liệu đầu vào nhằm hạn chế lỗi và tấn công.

Kết nối cơ sở dữ liệu : Backend kết nối trực tiếp đến MongoDB Atlas thông qua chuỗi kết nối bảo mật. Việc sử dụng MongoDB Atlas giúp hệ thống đảm bảo tính ổn định, an toàn dữ liệu và khả năng mở rộng khi số lượng người dùng tăng.

3.4.2. Xây dựng frontend

Frontend của hệ thống đặt lịch sân thể thao được xây dựng bằng ReactJS, chịu trách nhiệm hiển thị giao diện và tương tác trực tiếp với người dùng. Ứng dụng frontend được thiết kế theo mô hình Single Page Application (SPA) nhằm nâng cao trải nghiệm người dùng.

Kiến trúc frontend

Ứng dụng được xây dựng theo mô hình component-based, trong đó mỗi chức năng được tách thành các component độc lập.

Sử dụng cơ chế quản lý state để cập nhật dữ liệu và trạng thái giao diện theo thời gian thực.

Các component được tổ chức rõ ràng theo từng nhóm chức năng như xác thực, hiển thị sân, đặt lịch và quản trị.

Các chức năng chính của frontend

- Giao diện đăng ký, đăng nhập và quản lý tài khoản người dùng.

- Hiển thị danh sách sân thể thao và thông tin chi tiết của từng sân.

- Giao diện lựa chọn ngày, khung giờ và thực hiện đặt lịch sân.

- Giao diện thanh toán và theo dõi trạng thái đơn đặt sân.

- Giao diện đánh giá sân và hiển thị thông báo cho người dùng.

- Giao diện quản trị dành cho quản trị viên để quản lý sân và lịch đặt sân.

Giao tiếp với backend: Frontend giao tiếp với backend thông qua các API RESTful, sử dụng các thư viện hỗ trợ gọi API. Dữ liệu được xử lý và hiển thị động, giúp người dùng nhận được phản hồi nhanh chóng khi thao tác.

Trải nghiệm người dùng và giao diện: Frontend được thiết kế với giao diện trực quan, dễ sử dụng, đảm bảo khả năng hiển thị tốt trên nhiều thiết bị khác nhau. Việc cập nhật dữ liệu động giúp hệ thống phản hồi nhanh và mang lại trải nghiệm người dùng mượt mà.

3.5. Cài đặt và triển khai hệ thống

Sau khi hoàn thành quá trình thiết kế và xây dựng, hệ thống đặt lịch sân thể thao được tiến hành cài đặt và triển khai bằng Docker, kết hợp sử dụng MongoDB Atlas làm cơ sở dữ liệu. Việc sử dụng Docker giúp chuẩn hóa môi trường triển khai, trong khi MongoDB Atlas cung cấp dịch vụ cơ sở dữ liệu đám mây ổn định, bảo mật và dễ mở rộng.

Hệ thống được triển khai theo mô hình phân tách frontend - backend, trong đó cơ sở dữ liệu MongoDB được quản lý trên nền tảng Atlas và kết nối từ backend thông qua Internet.

3.5.1. Môi trường và công cụ phát triển

Hệ thống được phát triển và triển khai trong môi trường sau:

- Hệ điều hành: Windows 11

- Ngôn ngữ lập trình: JavaScript (ES6+)

- Frontend: ReactJS

- Backend: NodeJS, Express.js

- Cơ sở dữ liệu: MongoDB Atlas

- Xác thực và phân quyền: JWT (JSON Web Token)

Công cụ và nền tảng triển khai:

- Visual Studio Code: soạn thảo mã nguồn

- Git & GitHub: quản lý mã nguồn

- Docker & Docker Compose: đóng gói và triển khai hệ thống

- MongoDB Atlas: quản lý cơ sở dữ liệu trên nền tảng đám mây

- Postman: kiểm thử API

- Trình duyệt Google Chrome: kiểm thử giao diện

Việc lựa chọn các công nghệ và công cụ trên giúp hệ thống đảm bảo tính ổn định, bảo mật và khả năng mở rộng trong quá trình vận hành.

3.5.2. Triển khai và chạy thử nghiệm

Mô hình triển khai hệ thống

Hệ thống được triển khai với các thành phần chính sau:

- Frontend container: Chạy ứng dụng ReactJS.

- Backend container: Chạy NodeJS và Express.js, xử lý nghiệp vụ và kết nối đến MongoDB Atlas.

- MongoDB Atlas: Cơ sở dữ liệu được triển khai trên cloud, không chạy trong container.

Trong mô hình này, backend container kết nối trực tiếp đến MongoDB Atlas thông qua chuỗi kết nối bảo mật.

Quy trình triển khai

Quá trình triển khai hệ thống được thực hiện theo các bước sau:

- Chuẩn bị mã nguồn: Mã nguồn frontend và backend được tổ chức rõ ràng, mỗi thành phần có file Dockerfile riêng để cấu hình môi trường chạy.

- Cấu hình biến môi trường: Các biến môi trường được khai báo trong Docker Compose, bao gồm:

+ Chuỗi kết nối MongoDB Atlas

+ JWT Secret Key

+ Cổng chạy backend

+ URL backend cho frontend

- Xây dựng và khởi chạy container: Docker Compose được sử dụng để build image và khởi chạy các container frontend và backend đồng thời.

Chạy thử nghiệm hệ thống

Sau khi triển khai, hệ thống được tiến hành chạy thử nghiệm nhằm đánh giá khả năng hoạt động thực tế. Nội dung thử nghiệm bao gồm:

- Kiểm tra khả năng khởi động và giao tiếp giữa frontend và backend container.

- Kiểm tra kết nối giữa backend và MongoDB Atlas.

- Kiểm tra các chức năng chính như đăng nhập, xem sân, đặt lịch sân và thanh toán.

- Kiểm tra tính ổn định của hệ thống khi thực hiện nhiều thao tác liên tục.

Kết quả thử nghiệm cho thấy hệ thống hoạt động ổn định, các chức năng đáp ứng đúng yêu cầu đề ra và dữ liệu được lưu trữ, truy xuất chính xác trên MongoDB Atlas.

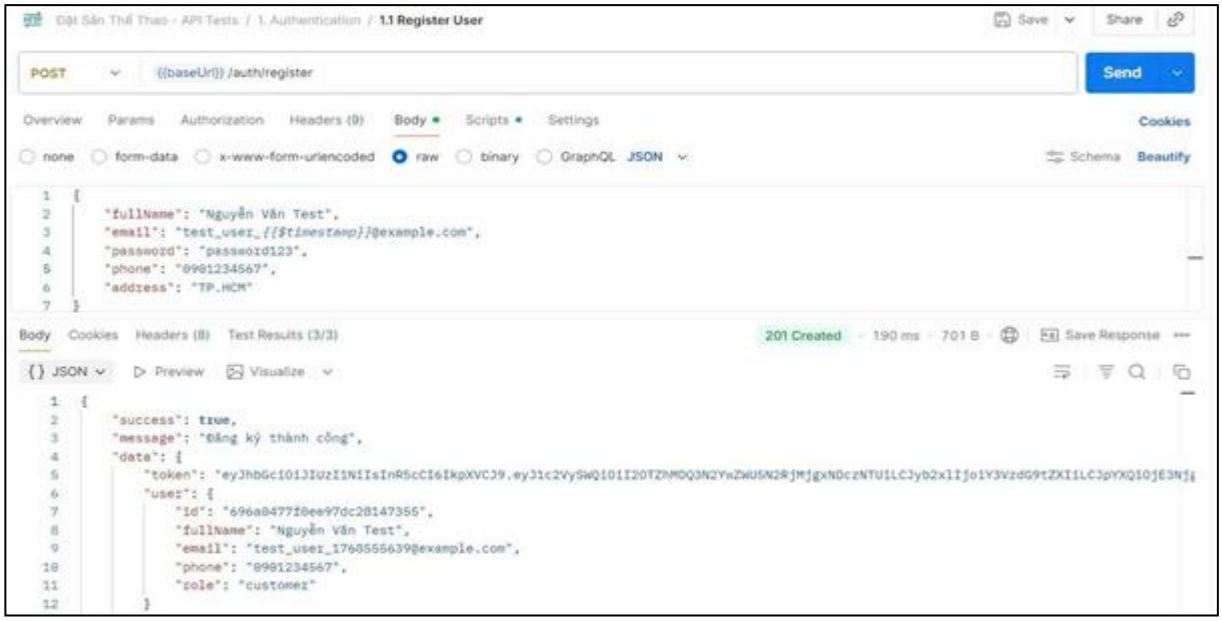
CHƯƠNG 4: KẾT QUẢ NGHIÊN CỨU

4.1. Kiểm thử API với Postman

Trong quá trình phát triển hệ thống backend, các API được kiểm thử bằng công cụ Postman nhằm đảm bảo tính chính xác, ổn định và khả năng đáp ứng yêu cầu của hệ thống. Việc kiểm thử API giúp xác minh các chức năng hoạt động đúng theo thiết kế trước khi tích hợp với giao diện frontend.

Các nhóm API chính đã được kiểm thử bao gồm:

API đăng ký

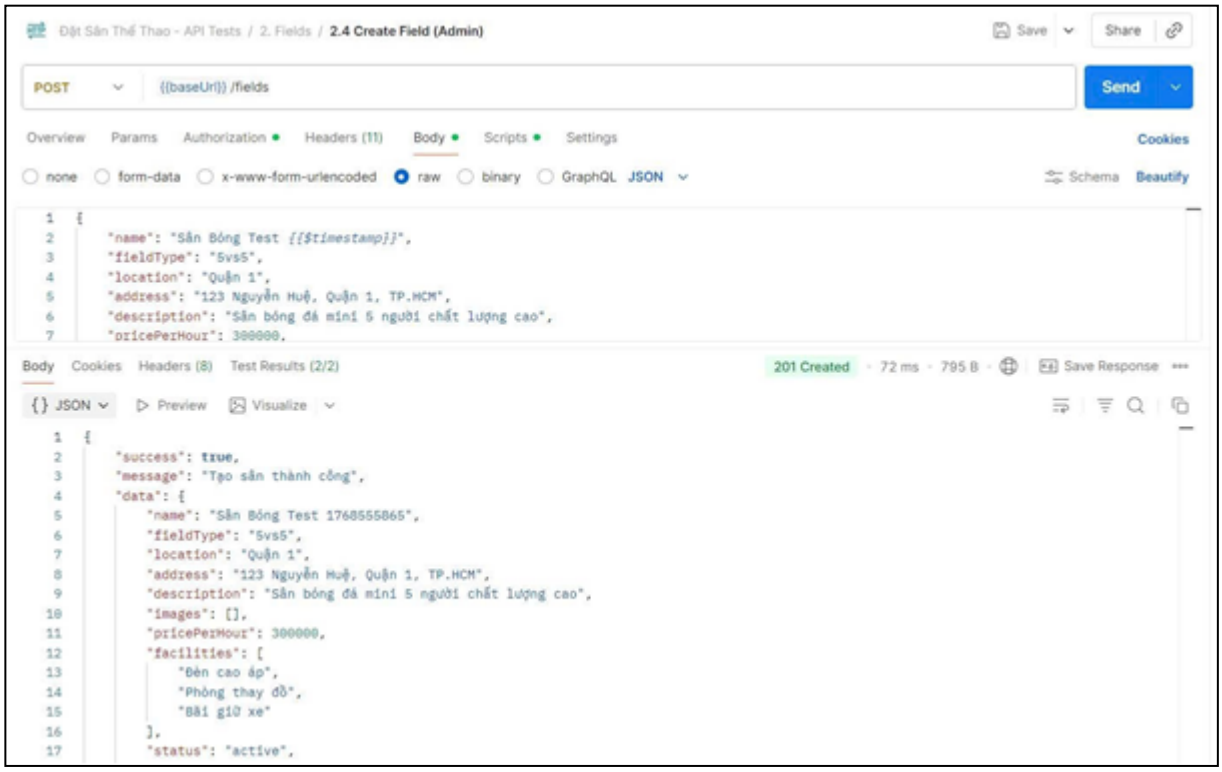


Hình 4.1. Kiểm thử API đăng ký

API đăng nhập

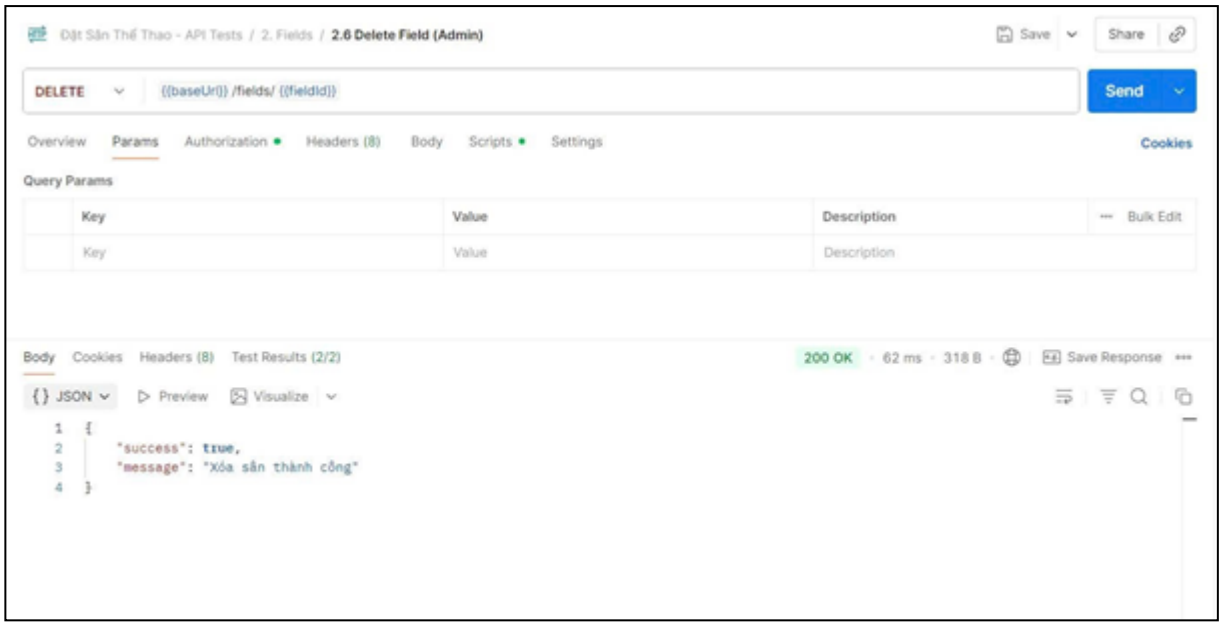
Hình 4.2. Kiểm thử API đăng nhập

API tạo sân



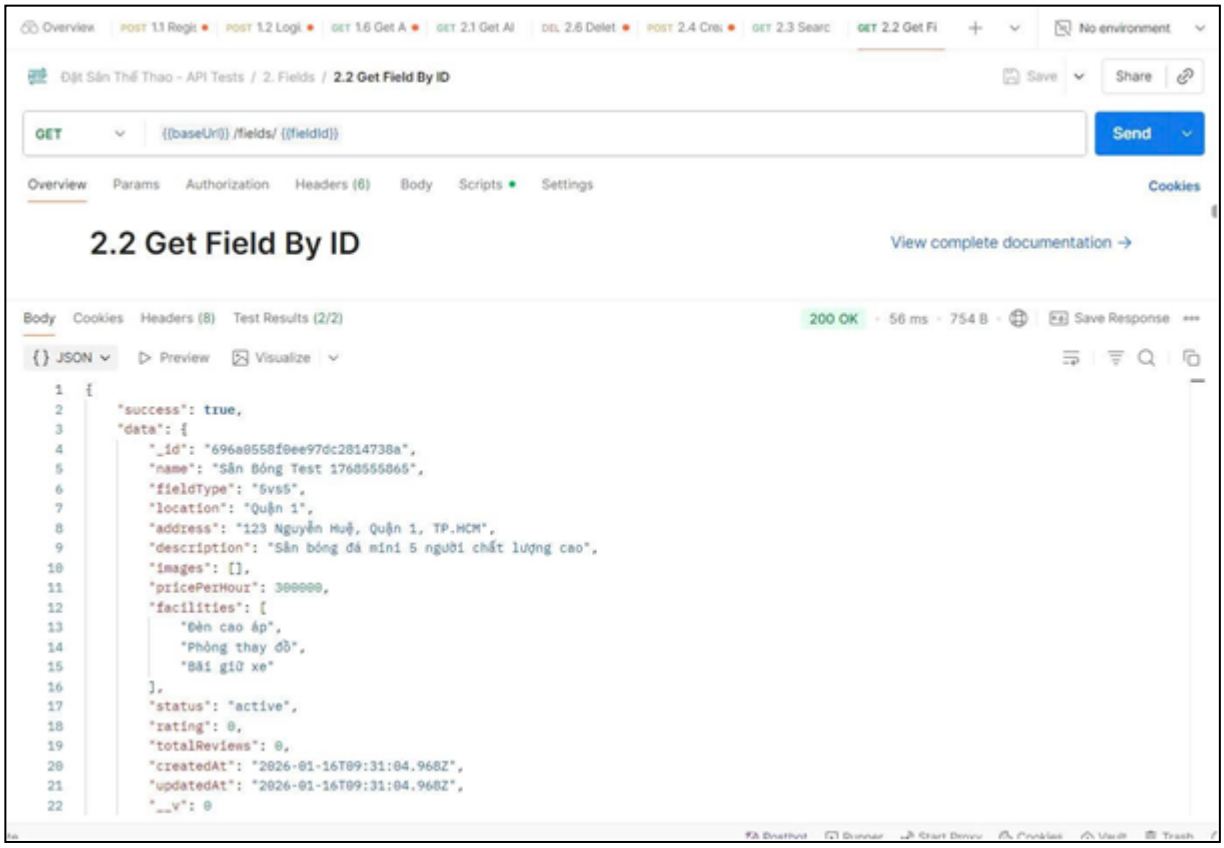
Hình 4.3. Kiểm thử API đặt sân

API xóa sân



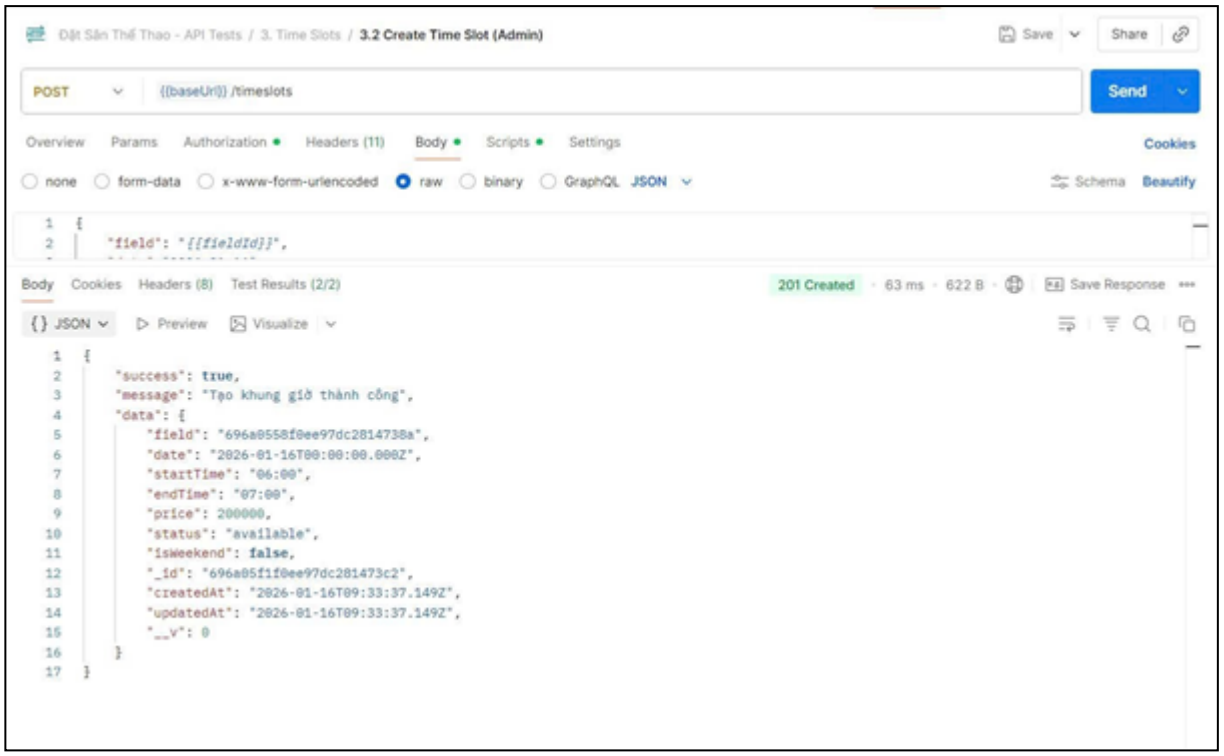
Hình 4.4. Kiểm thử API xóa sân

API lấy sân theo id



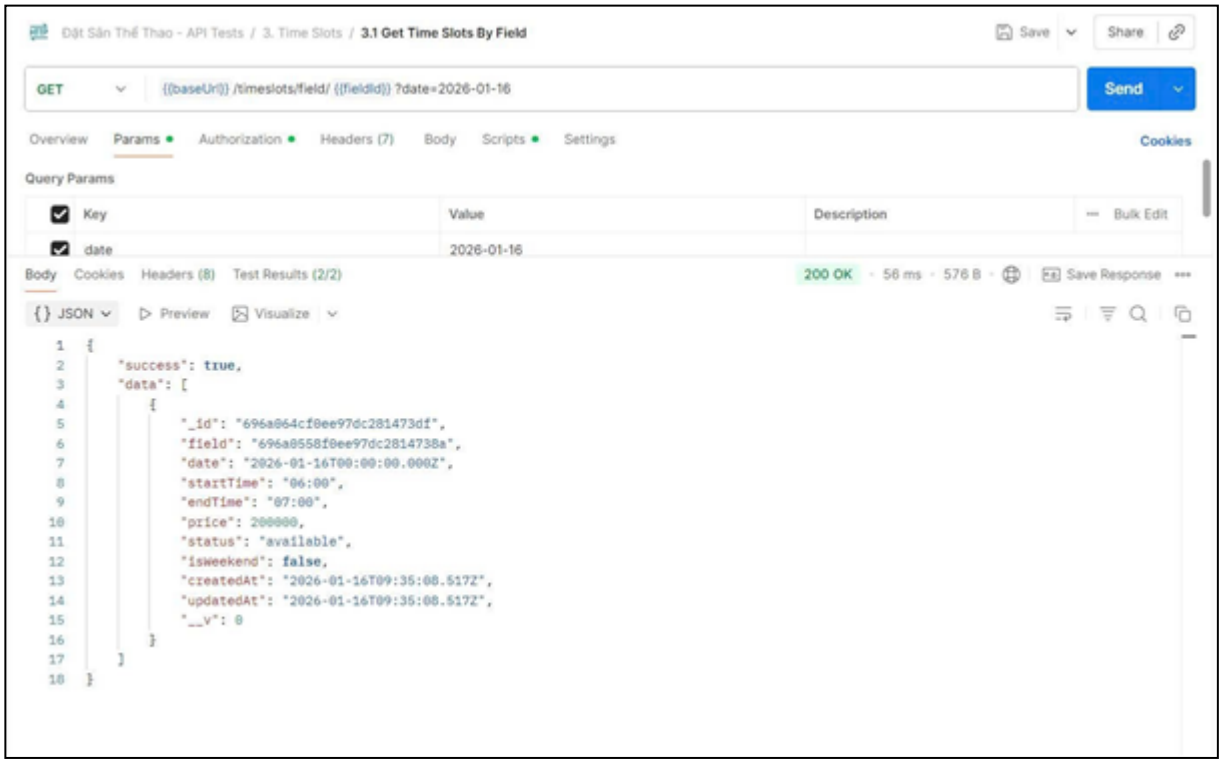
Hình 4.5. Kiểm thử API lấy sân theo id

API tạo khung giờ



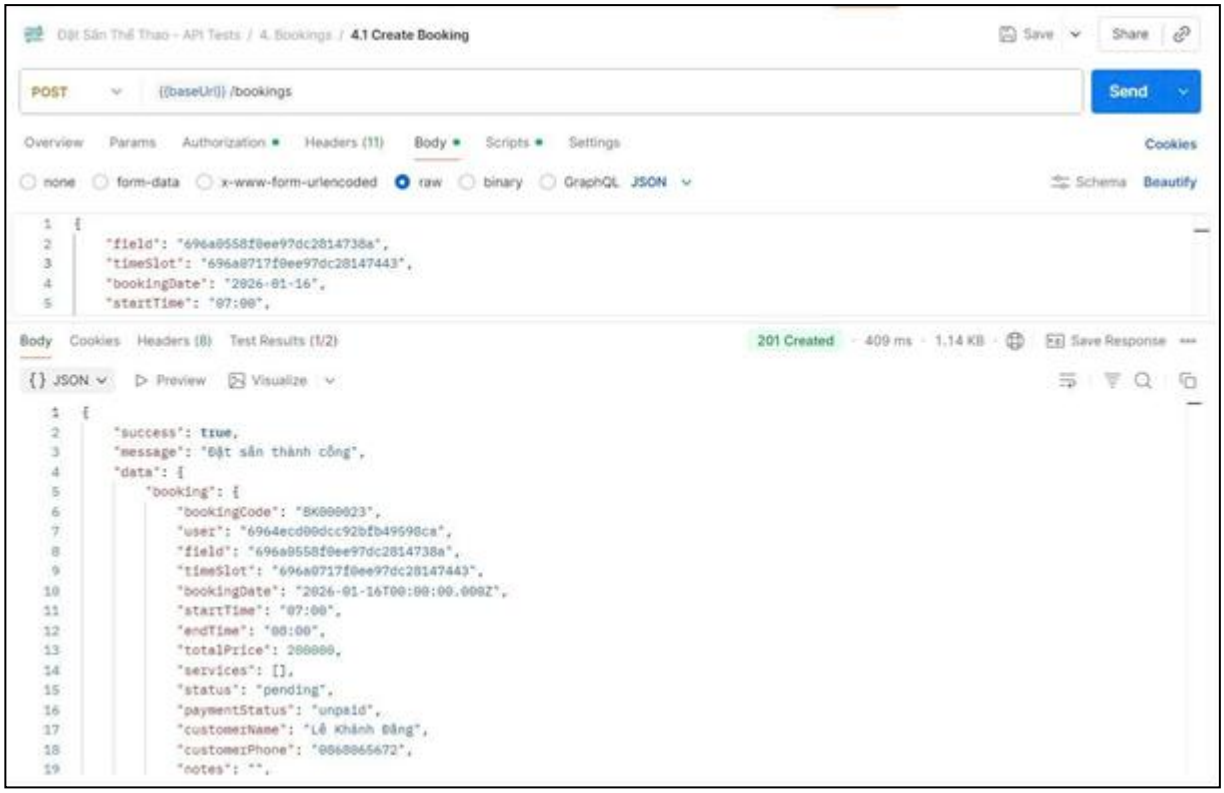
Hình 4.6. Kiểm thử API tạo khung giờ

API lấy khung giờ



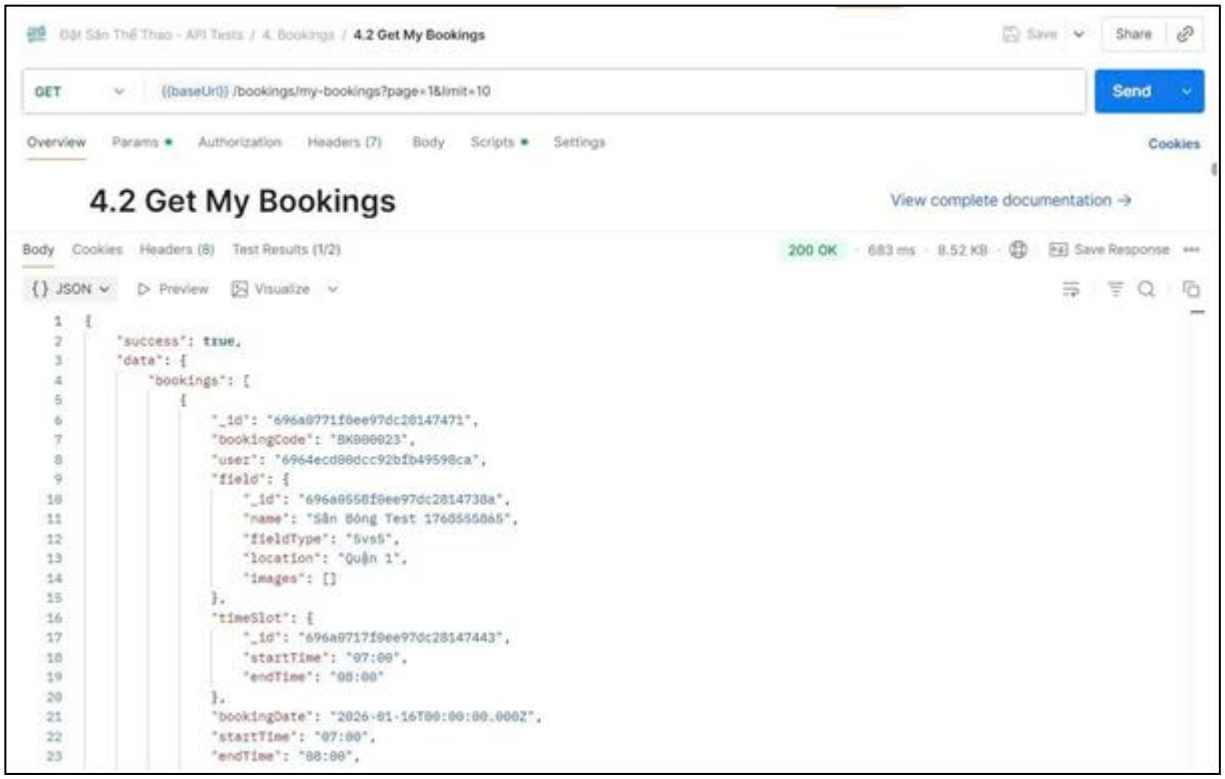
Hình 4.7. Kiểm thử API lấy khung giờ

API đặt sân



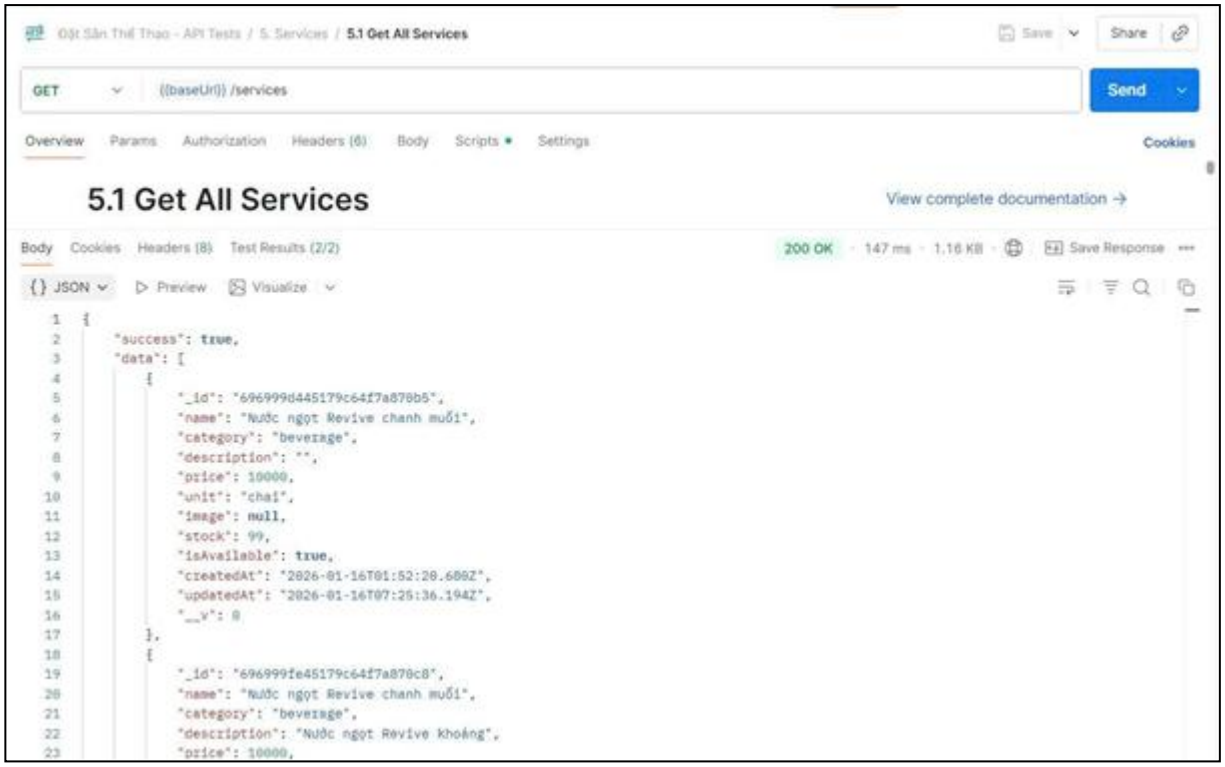
Hình 4.8. Kiểm thử API đặt sân

API lấy danh sách đã đặt



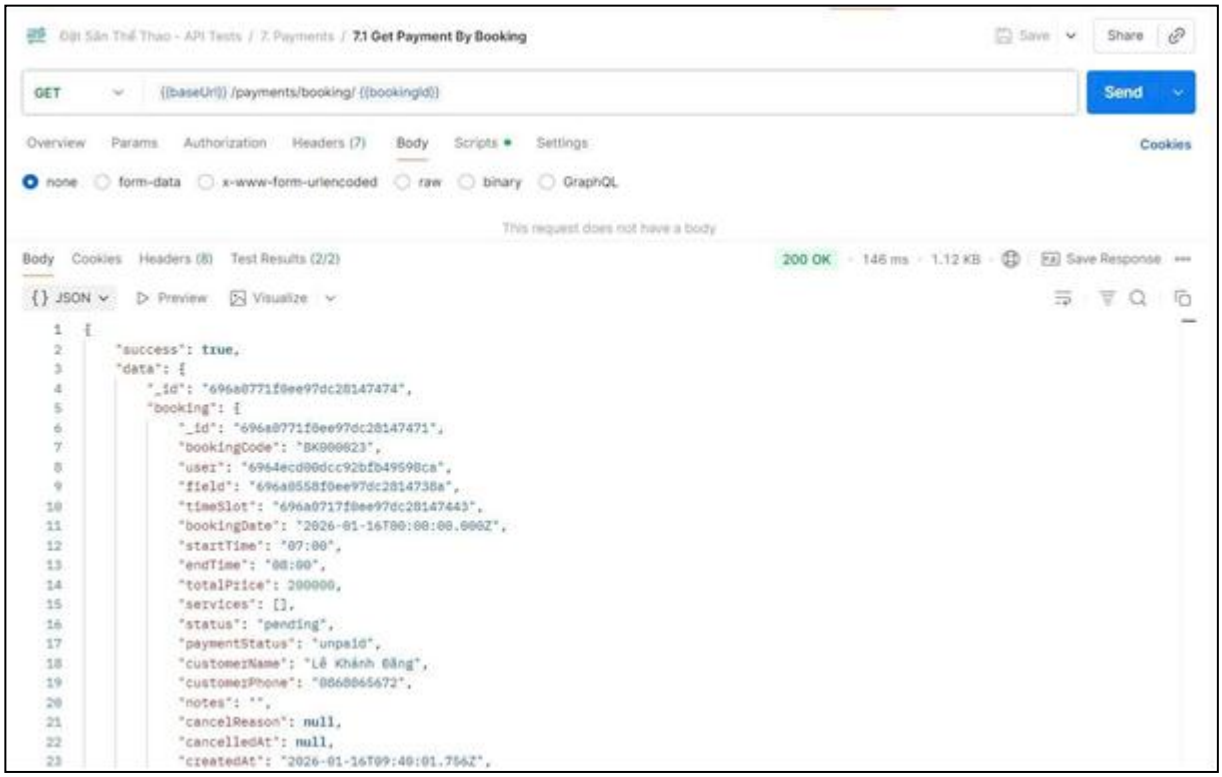
Hình 4.9. Kiểm thử API lấy danh sách sân đã đặt

API lấy danh sách dịch vụ



Hình 4.10. Kiểm thử API lấy danh sách dịch vụ

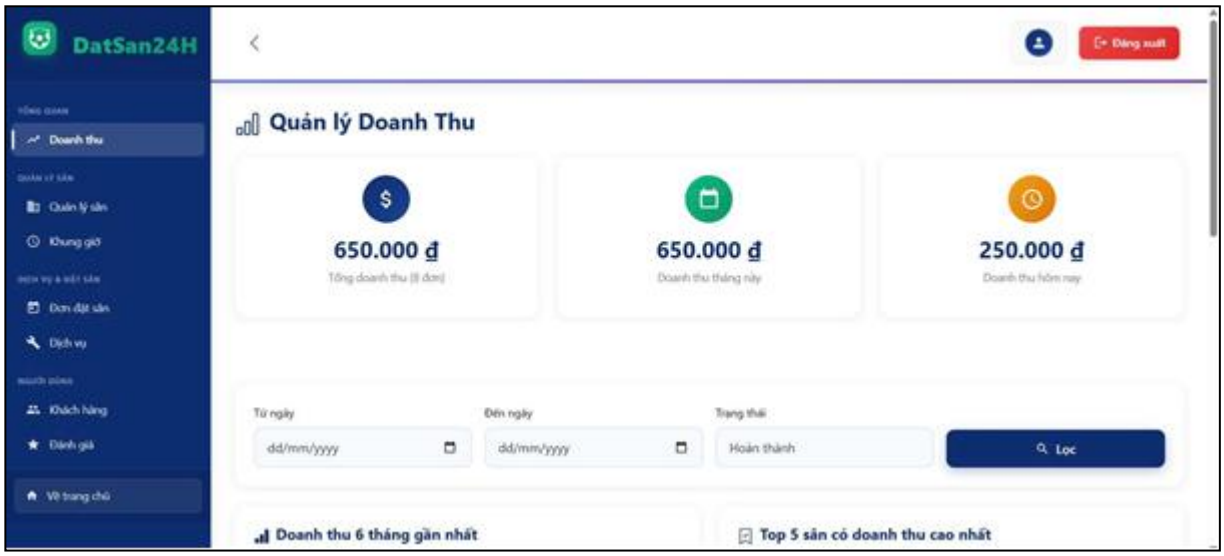
API lấy thanh toán của đơn đã đặt



Hình 4.11. Kiểm thử API lấy thanh toán của đơn đã đặt

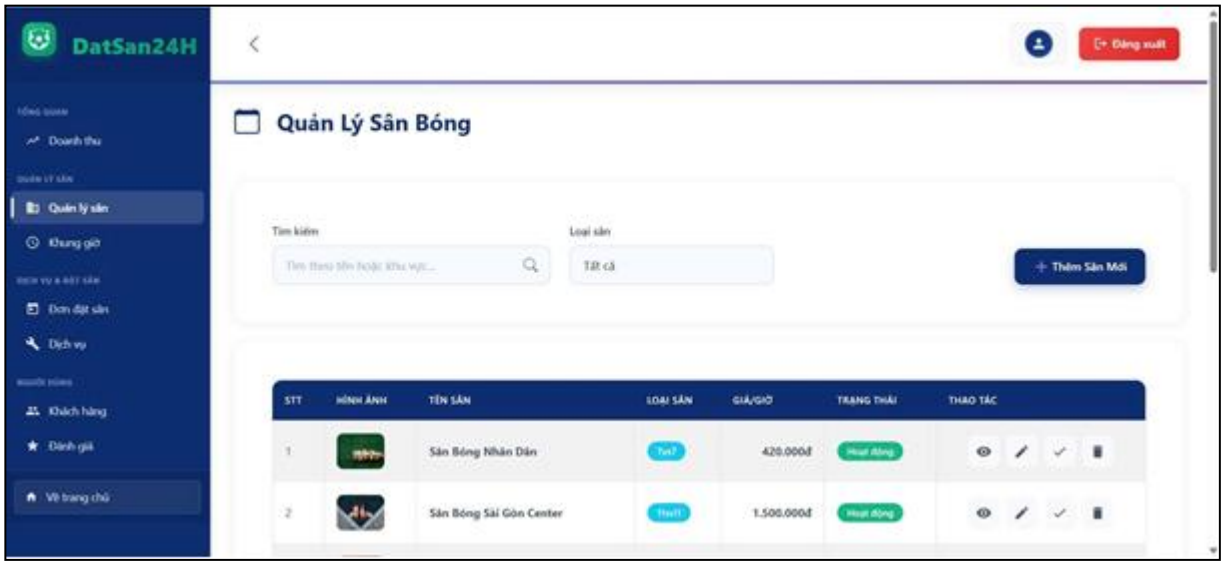
4.2. Giao diện người dùng

Giao diện thống kê: xem doanh thu tổng và danh thu 6 tháng gần nhất , thống kê các sân có danh thu cao nhất.



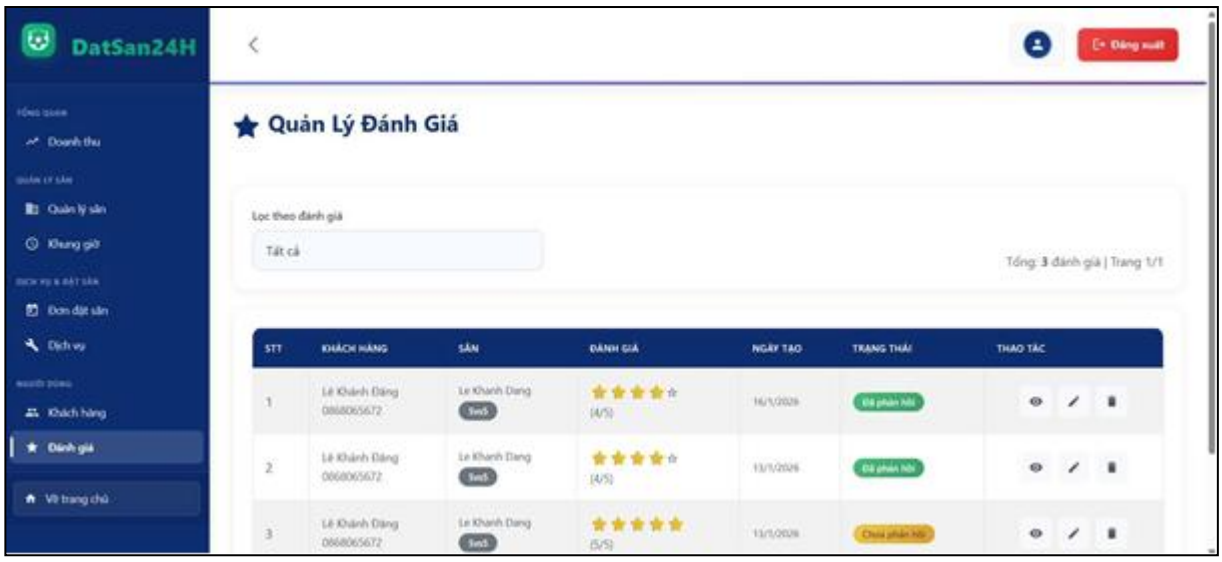
Hình 4.12. Giao diện trang thống kê

Giao diện quản lý sân : quản trị có thể thêm sửa xóa sân



Hình 4.13. Giao diện trang quản lý sân

Giao diện quản lý đánh giá : quản trị có thể xem , trả lời , xóa đánh giá

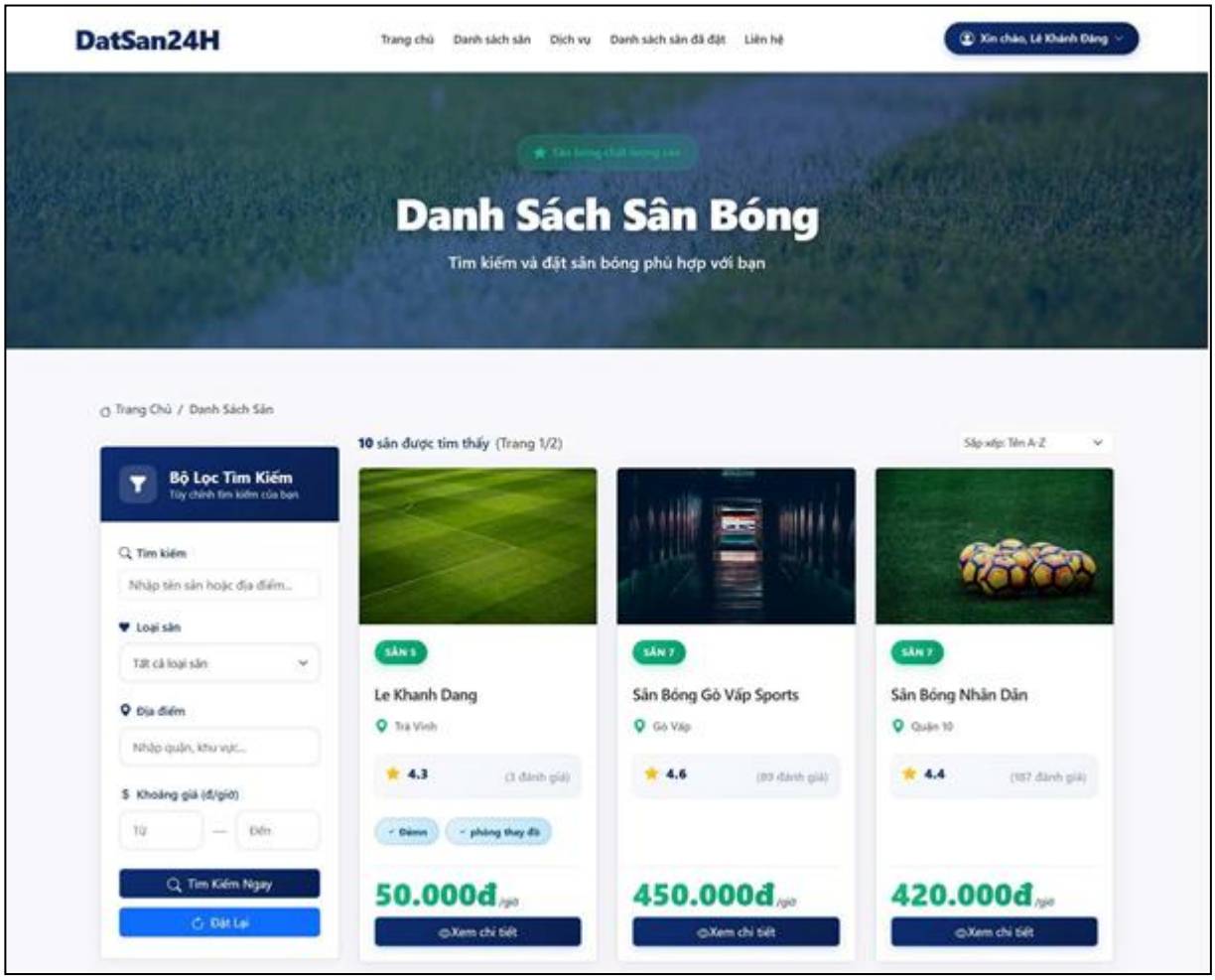


Hình 4.14. Giao diện trang quản lý đánh giá

Giao diện trang chủ: hiển thị thông tin của trang web, danh sách sân, giới thiệu, chính sách

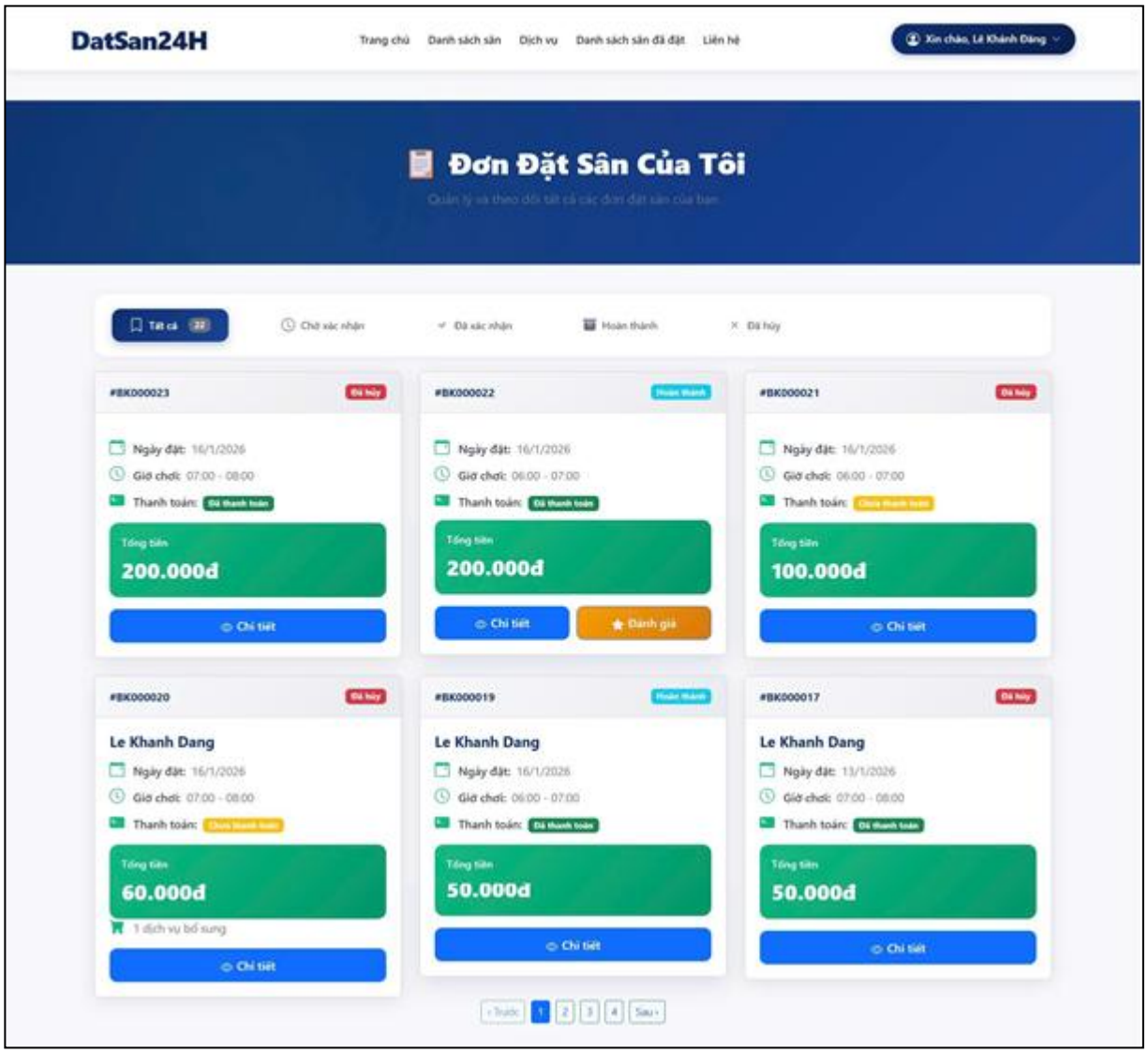
Hình 4.15. Giao diện trang chủ

Giao diện danh sách sân: có thể tìm kiếm, lọc theo loại sân , giá



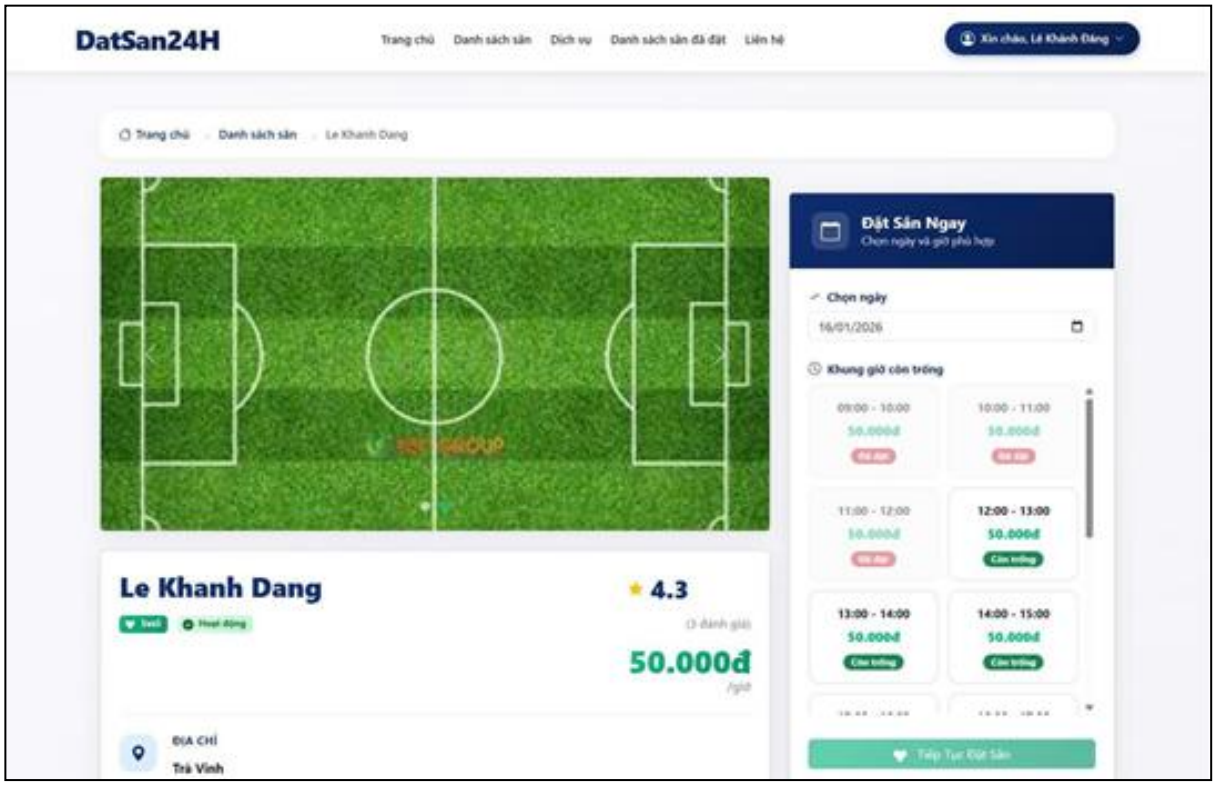
Hình 4.16. Giao diện trang danh sách sân

Giao diện danh sách sân đã đặt: danh sách các sân đã đặt, có thể xem chi tiết, thanh toán nếu chưa thanh toán.



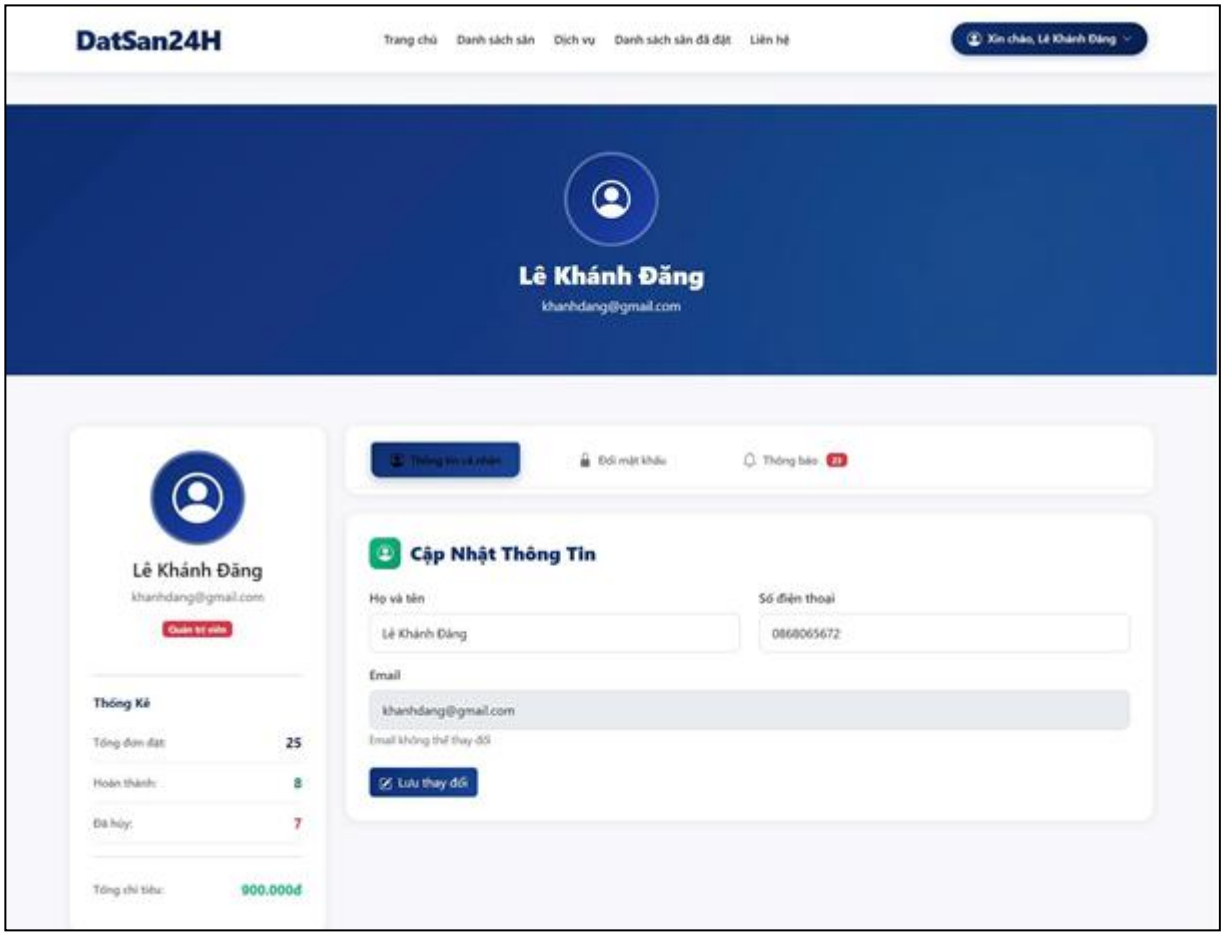
Hình 4.17. Giao diện trang danh sách sân đã đặt

Giao diện đặt sân: xem chi tiết sân , chọn khung giờ để đặt.



Hình 4.18. Giao diện trang đặt sân

Giao diện thông tin cá nhân: thống kê cá nhân, cập nhật thông tin, đổi mật khẩu.



Hình 4.19. Giao diện trang thông tin cá nhân

CHƯƠNG 5: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

5.1. Kết luận

Sau quá trình nghiên cứu, phân tích, thiết kế và triển khai, đề tài "Xây dựng hệ thống đặt lịch sân thể thao" đã hoàn thành các mục tiêu đề ra ban đầu. Hệ thống được xây dựng theo mô hình ứng dụng web hiện đại, tách biệt frontend và backend, sử dụng các công nghệ phổ biến như ReactJS, NodeJS, Express.js và MongoDB Atlas.

Hệ thống cho phép người dùng dễ dàng xem thông tin sân, lựa chọn khung giờ phù hợp và thực hiện đặt lịch trực tuyến. Đồng thời, hệ thống hỗ trợ quản trị viên trong việc quản lý sân bãi, lịch đặt sân, thanh toán và theo dõi hoạt động của hệ thống. Kết quả chạy thử nghiệm cho thấy hệ thống hoạt động ổn định, các chức năng cơ bản đáp ứng đúng yêu cầu đặt ra, giao diện thân thiện và dễ sử dụng.

5.1.1. Ưu điểm

Hệ thống đặt lịch sân thể thao đạt được nhiều ưu điểm nổi bật, cụ thể như sau:

- Hệ thống hoạt động theo mô hình client-server rõ ràng, dễ triển khai và bảo trì.

- Giao diện người dùng trực quan, thân thiện, giúp người dùng dễ dàng thao tác.

- Chức năng đặt lịch sân được tự động hóa, hạn chế tình trạng trùng lịch và sai sót.

- Sử dụng MongoDB Atlas giúp dữ liệu được lưu trữ an toàn, dễ mở rộng và quản lý.

- Áp dụng Docker trong triển khai giúp chuẩn hóa môi trường và đơn giản hóa quá trình cài đặt.

- Hệ thống hỗ trợ phân quyền người dùng rõ ràng giữa khách hàng và quản trị viên.

- Có khả năng tích hợp các dịch vụ đi kèm và mở rộng chức năng trong tương lai.

5.1.2. Nhược điểm

Bên cạnh những ưu điểm đạt được, hệ thống vẫn còn một số hạn chế nhất định:

- Hệ thống mới dừng lại ở mức đáp ứng các chức năng cơ bản, chưa tối ưu cho quy mô người dùng lớn.

- Chưa tích hợp đầy đủ các cổng thanh toán điện tử thực tế.

- Chưa có ứng dụng di động riêng, chủ yếu hoạt động trên nền tảng web.

- Tính năng thống kê và báo cáo chưa được phát triển chuyên sâu.

- Khả năng bảo mật có thể tiếp tục được nâng cao để đáp ứng các tiêu chuẩn cao hơn trong môi trường thực tế.

5.2. Hướng phát triển

Trong thời gian tới, hệ thống đặt lịch sân thể thao có thể được tiếp tục phát triển và hoàn thiện theo các hướng sau:

- Phát triển thêm ứng dụng di động trên nền tảng Android và iOS.

- Tích hợp các cổng thanh toán trực tuyến phổ biến như VNPay, MoMo, ZaloPay.

- Bổ sung chức năng thống kê, báo cáo doanh thu và lịch sử hoạt động chi tiết.

- Áp dụng các cơ chế bảo mật nâng cao như xác thực hai yếu tố (2FA).

- Tối ưu hiệu năng hệ thống để đáp ứng số lượng lớn người dùng truy cập đồng thời.

- Mở rộng hệ thống cho nhiều loại hình sân thể thao khác nhau như cầu lông, tennis, bóng rổ.

- Ứng dụng công nghệ trí tuệ nhân tạo để gợi ý khung giờ và sân phù hợp cho người dùng.

DANH MỤC TÀI LIỆU THAM KHẢO

[1] Meta Platforms, Inc., React - A JavaScript library for building user interfaces. https://react.dev/

[2] M. Banks, Learning React, O'Reilly Media, 2020.

[3] OpenJS Foundation, Node.js Documentation. https://nodejs.org/en/docs/

[4] Express.js Foundation, Express.js Documentation. https://expressjs.com/

[6] MongoDB Inc., MongoDB Manual. https://www.mongodb.com/docs/

[7] Postman Inc., Postman API Testing Documentation. [Online]. Available: https://learning.postman.com/