

**BỘ GIÁO DỤC VÀ ĐÀO TẠO**

**TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT TP.HCM**

**KHOA CÔNG NGHỆ THÔNG TIN**

A logo of hands holding a book

Description automatically generated

-----🙞🙞🙞🙞🙞-----

**BÁO CÁO CUỐI KỲ**

**Chuyên ngành: Hệ Thống Thông Tin**

**Môn: Bảo Mật Cơ Sở Dữ Liệu**

**XÂY DỰNG ỨNG DỤNG WEDSITE CHO**

**QUY TRÌNH GIA HẠN HỘ CHIẾU**

(Áp dụng OracleDB, công nghệ ReactJS và Spring Boot)

**Sinh viên thực hiện:**

Trần Nguyễn Trí Đạt 21110162

Tào Việt Đức 21110169

Phạm Tấn Huy 21110190

Lê Nguyễn Trí Nhân 21110841

**Giáo viên hướng dẫn:** THS. LÊ THỊ MINH CHÂU

Thành phố Hồ Chí Minh, 28 tháng 11, năm 2024

**BẢNG PHÂN CÔNG CÔNG VIỆC**

**HỌC KỲ I NĂM HỌC 2024-2025**

**Nhóm 2**

**Đề tài: XÂY DỰNG ỨNG DỤNG WEDSITE CHO QUY TRÌNH GIA HẠN HỘ CHIẾU**

|  |  |  |
| --- | --- | --- |
| **MSSV** | **Họ và tên** | **Phân chia công việc** |
| 21110162 | Trần Nguyễn Trí Đạt | * Thiết kế Cơ Sở Dữ Liệu. * Phân quyền người dùng. * Giám sát các user trên các bảng. |
| 21110169 | Tào Việt Đức | * Xây dụng Frontend (công nghệ ReactJS). * Xây dựng API gọi dữ liệu từ CSDL lên giao diện. * Tích hợp API (cộng cụ Postman). |
| 21110190 | Phạm Tấn Huy | * Xây dựng Cơ Sở Dữ Liệu. * Trigger kiểm tra và duy trì nhật ký bảo mật. * Hàm và Policy cho nhân viên xét duyệt (XD) và nhân viên xác thực (XT). |
| 21110841 | Lê Nguyễn Trí Nhân | * Chuyển đổi các bảng từ CSDL sang lớp Entity và DTO, Repository, Service. * Xây dựng Backend (công nghệ SpringBoot). |

# **LỜI CẢM ƠN**

Em xin gửi lời cảm ơn chân thành và sâu sắc đến ThS. Lê Thị Minh Châu – cô giáo hướng dẫn, vì sự tận tâm, chỉ bảo và hướng dẫn quý báu trong suốt quá trình phát triển đề tài "Xây dựng ứng dụng website cho quy trình gia hạn hộ chiếu". Cô không chỉ cung cấp những tài liệu chuyên môn bổ ích mà còn giúp em hiểu rõ hơn về các yêu cầu thực tế, cũng như cách áp dụng kiến thức lý thuyết vào công việc thực tế. Sự chỉ dẫn của cô đã giúp em vượt qua những khó khăn trong việc triển khai hệ thống và áp dụng công nghệ OracleDB, ReactJS và Spring Boot một cách hiệu quả.

Em cũng xin chân thành cảm ơn các tổ chức, cá nhân và những người đã tạo điều kiện thuận lợi cho em trong suốt quá trình thực hiện nghiên cứu. Sự giúp đỡ và đóng góp của mọi người đã góp phần không nhỏ vào thành công của dự án này.

Em xin trân trọng cảm ơn!

**NHẬN XÉT**

**ĐIỂM GV KÝ TÊN**

# **MỤC LỤC**

[**LỜI CẢM ƠN** 3](#_Toc185622304)

[**MỤC LỤC** 5](#_Toc185622305)

[**I.** **PHẦN MỞ ĐẦU** 9](#_Toc185622306)

[**II.** **PHẦN NỘI DUNG** 10](#_Toc185622307)

[**Chương 1: TỔNG QUAN VỀ CÔNG NGHỆ ÁP DỤNG.** 10](#_Toc185622308)

[**Chương 2: PHÂN TÍCH YÊU CẦU HỆ THỐNG.** 13](#_Toc185622309)

[**2.1**  **ĐẶC TẢ.** 13](#_Toc185622310)

[**2.2** **PHÂN TÍCH THIẾT KẾ CƠ SỞ DỮ LIỆU.** 14](#_Toc185622311)

[**2.2.1**  **PHÂN TÍCH THIẾT KẾ MỨC LOGIC.** 14](#_Toc185622312)

[**2.2.1.1 Kiểu Dữ Liệu Và Ràng Buộc.** 14](#_Toc185622313)

[**2.2.1.1.1 Resident Table.** 14](#_Toc185622314)

[**2.2.1.1.2 PassportRenewalRequest Table.** 15](#_Toc185622315)

[**2.2.1.1.3 VerificationUnit Table.** 15](#_Toc185622316)

[**2.2.1.1.4 Renewal Table.** 16](#_Toc185622317)

[**2.2.1.1.5 ActivityLog Table.** 17](#_Toc185622318)

[**2.2.1.1.6 Approval Table.** 17](#_Toc185622319)

[**2.2.2.2** **Tạo các hàm chính sách (Policy\_XT, Policy\_XD, Policy\_LT).** 18](#_Toc185622320)

[**2.2.2.3** **Truy vấn thông tin audit.** 20](#_Toc185622321)

[**2.2.3**  **PHÂN TÍCH THIẾT KẾ MỨC VẬT LÝ.** 20](#_Toc185622322)

[**2.2.3.1 Resident Table.** 22](#_Toc185622323)

[**2.2.3.2 PassportRenewalRequest Table.** 23](#_Toc185622324)

[**2.2.3.3 VerificationUnit Table.** 24](#_Toc185622325)

[**2.2.3.4 Renewal Table.** 24](#_Toc185622326)

[**2.2.3.5 ActivityLog Table.** 25](#_Toc185622327)

[**2.2.3.6 ActivityLog Table.** 26](#_Toc185622328)

[**2.2.3.7 Policy.** 27](#_Toc185622329)

[**2.2.3.8 Audit.** 30](#_Toc185622330)

[**2.2.3.9 TRIGGER.** 30](#_Toc185622331)

[**Chương 3: CÀI ĐẶT MẠNG HỆ THỐNG** 32](#_Toc185622332)

[**3.1 Môi trường cài đặt.** 32](#_Toc185622333)

[**3.1.2.1 Frontend Development.** 32](#_Toc185622334)

[**3.1.2.3 Backend Development.** 34](#_Toc185622335)

[**3.1.2.4 IDE và Trình Soạn Thảo.** 36](#_Toc185622336)

[**4.1.2.5 Công cụ bổ trợ.** 37](#_Toc185622337)

[**3.2 Cài đặt cơ sở dữ liệu.** 38](#_Toc185622338)

[**3.3 Triển khai frontend.** 51](#_Toc185622339)

[**3.3.1 User.** 51](#_Toc185622340)

[**3.3.1.1 Trang Chủ.** 51](#_Toc185622341)

[**3.3.1.3 Dịch Vụ.** 52](#_Toc185622342)

[**3.3.1.4 Quy Trình.** 52](#_Toc185622343)

[**3.3.1.5 Liên Hệ.** 53](#_Toc185622344)

[**3.3.1.6 Dashboard.** 53](#_Toc185622345)

[**3.3.1.7 Xác Thực.** 53](#_Toc185622346)

[**3.3.1.8 Xét Duyệt.** 54](#_Toc185622347)

[**3.3.1.9 Lưu Trữ.** 54](#_Toc185622348)

[**3.3.1.10 Giám Sát.** 54](#_Toc185622349)

[**3.3.1.11 Form Điền Thông Tin.** 55](#_Toc185622350)

[**3.4 Triển khai backend.** 55](#_Toc185622351)

[**3.4.1 Cấu hình môi trường triển khai.** 55](#_Toc185622352)

[**3.4.2 Lớp Entity.** 57](#_Toc185622353)

[3.4.2.1 Resident 57](#_Toc185622354)

[3.4.2.2 PASSPORTRENEWALREQUEST 60](#_Toc185622355)

[3.4.2.3 Approval 62](#_Toc185622356)

[3.4.2.4 Renewal 64](#_Toc185622357)

[3.4.2.5 ACTIVITYLOG 67](#_Toc185622358)

[**3.4.3 Lớp DTO.** 69](#_Toc185622359)

[3.4.3.1 ResidentDTO 69](#_Toc185622360)

[3.4.3.2 PassportRenewalRequestDTO 73](#_Toc185622361)

[3.4.3.3 ApprovalDTO 74](#_Toc185622362)

[3.4.3.4 RenewalDTO 76](#_Toc185622363)

[3.4.3.5 ActivityLogDTO 78](#_Toc185622364)

[**3.4.4 Lớp Repository.** 80](#_Toc185622365)

[3.4.4.1 ResidentRepository 80](#_Toc185622366)

[3.4.4.2 PassportRenewalRequestRepository 80](#_Toc185622367)

[3.4.4.3 ApprovalRepository 81](#_Toc185622368)

[3.4.4.4 RenewalRepository 81](#_Toc185622369)

[3.4.4.5 ActivityLogRepository 82](#_Toc185622370)

[**3.4.5 Lớp Service.** 82](#_Toc185622371)

[3.4.5.1 ResidentService 82](#_Toc185622372)

[3.4.5.2 PassportRenewalRequestService 84](#_Toc185622373)

[3.4.5.3 ApprovalService 86](#_Toc185622374)

[3.4.5.4 RenewalService 89](#_Toc185622375)

[3.4.5.5 ActivityLogService 92](#_Toc185622376)

[**3.4.6 Lớp Controller.** 93](#_Toc185622377)

[3.4.6.1 ResidentController 93](#_Toc185622378)

[3.4.6.2 PassportRenewalController 94](#_Toc185622379)

[3.4.6.3 ApprovalController 96](#_Toc185622380)

[3.4.6.4 RenewalController 98](#_Toc185622381)

[3.4.6.5 ActivityLogController 100](#_Toc185622382)

[**3.4.7 Xây dựng API.** 101](#_Toc185622383)

[**3.4.8 Kết nối cơ sở dữ liệu.** 108](#_Toc185622384)

[**3.4.9 Webconfig.** 108](#_Toc185622385)

[**3.4.10 Kế hoạch bảo trì và nâng cấp.** 109](#_Toc185622386)

[**Chương 4: KẾT LUẬN.** 111](#_Toc185622387)

[**TÀI LIỆU THAM KHẢO.** 112](#_Toc185622388)

**Ý kiến của giáo viên hướng dẫn**

………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………

(Ký và ghi rõ họ tên)

# **PHẦN MỞ ĐẦU**

Trong bối cảnh công nghệ thông tin ngày càng phát triển, các quy trình hành chính, đặc biệt là quy trình gia hạn hộ chiếu, cần được số hóa để nâng cao hiệu quả và tiết kiệm thời gian cho người dân. Đề tài "Xây dựng ứng dụng website cho quy trình gia hạn hộ chiếu" nhằm mục đích tạo ra một hệ thống trực tuyến giúp đơn giản hóa và tự động hóa quy trình gia hạn hộ chiếu, từ việc tiếp nhận thông tin đến việc xử lý và cấp phép gia hạn.

Ứng dụng sẽ được xây dựng với OracleDB để quản lý cơ sở dữ liệu một cách hiệu quả và an toàn. ReactJS sẽ được sử dụng để phát triển giao diện người dùng, giúp mang đến một trải nghiệm mượt mà và trực quan. Spring Boot sẽ đảm nhận phần xử lý phía server, đảm bảo hiệu suất cao và bảo mật cho toàn bộ hệ thống.

Qua đề tài này, em sẽ trình bày chi tiết về quy trình phát triển ứng dụng, từ việc phân tích yêu cầu, thiết kế cơ sở dữ liệu, phát triển các tính năng chính, cho đến việc triển khai và kiểm thử hệ thống. Mục tiêu cuối cùng là xây dựng một hệ thống web giúp người dân dễ dàng thực hiện quy trình gia hạn hộ chiếu, đồng thời giảm tải cho các cơ quan hành chính.

# **PHẦN NỘI DUNG**

## **Chương 1: TỔNG QUAN VỀ CÔNG NGHỆ ÁP DỤNG.**

Trong quá trình phát triển ứng dụng website cho quy trình gia hạn hộ chiếu, việc lựa chọn công nghệ phù hợp là yếu tố quan trọng giúp đảm bảo hệ thống hoạt động hiệu quả, bảo mật và có khả năng mở rộng trong tương lai. Mỗi công nghệ trong hệ thống đóng một vai trò riêng biệt, từ việc quản lý cơ sở dữ liệu cho đến xử lý nghiệp vụ và giao diện người dùng. Trong chương này, chúng ta sẽ tìm hiểu tổng quan về các công nghệ chính được áp dụng trong việc xây dựng hệ thống, bao gồm **OracleDB**, **ReactJS**, và **Spring Boot**.

**OracleDB** là một hệ quản trị cơ sở dữ liệu quan hệ (RDBMS) được phát triển bởi Oracle Corporation, nổi tiếng với khả năng xử lý và quản lý dữ liệu quy mô lớn một cách hiệu quả và an toàn. OracleDB cung cấp nhiều tính năng mạnh mẽ, bao gồm khả năng bảo mật cao, khả năng xử lý giao dịch đồng thời, và tính toàn vẹn dữ liệu vượt trội. Đây là một trong những công nghệ cơ sở dữ liệu phổ biến và được sử dụng rộng rãi trong các ứng dụng yêu cầu tính ổn định và bảo mật cao, như trong các hệ thống hành chính công.

**Đặc điểm nổi bật của OracleDB:**

* **Khả năng quản lý dữ liệu lớn**: OracleDB được tối ưu hóa để quản lý một lượng lớn dữ liệu, điều này rất phù hợp với các ứng dụng yêu cầu lưu trữ và xử lý hồ sơ hành chính.
* **Tính bảo mật cao**: OracleDB cung cấp các công cụ mạnh mẽ để bảo vệ dữ liệu, bao gồm mã hóa dữ liệu và các tính năng kiểm soát truy cập, bảo vệ dữ liệu người dùng khỏi các mối đe dọa bên ngoài.
* **Khả năng mở rộng**: OracleDB có khả năng mở rộng mạnh mẽ, giúp hệ thống dễ dàng thích nghi và phát triển khi có sự thay đổi về quy mô người dùng hoặc dữ liệu.

Trong dự án này, OracleDB sẽ được sử dụng để quản lý dữ liệu liên quan đến người dùng, hồ sơ gia hạn hộ chiếu, lịch sử giao dịch, và các thông tin quan trọng khác.

**ReactJS** là một thư viện JavaScript mã nguồn mở được phát triển bởi Facebook, chủ yếu được sử dụng để xây dựng giao diện người dùng (UI) cho các ứng dụng web. Với sự phát triển mạnh mẽ và phổ biến, ReactJS đã trở thành công nghệ chủ đạo trong việc phát triển các ứng dụng frontend hiện đại, nhờ vào các đặc điểm như tính tái sử dụng, khả năng xử lý nhanh chóng và dễ dàng bảo trì.

**Đặc điểm nổi bật của ReactJS:**

* **Component-based architecture**: ReactJS cho phép phát triển giao diện người dùng dưới dạng các thành phần (components) độc lập, dễ tái sử dụng, giúp giảm thiểu việc phải viết lại mã nguồn cho các phần tử giao diện tương tự.
* **Virtual DOM**: ReactJS sử dụng một cấu trúc gọi là Virtual DOM, giúp cải thiện hiệu suất của ứng dụng bằng cách chỉ cập nhật những phần tử cần thiết thay vì làm mới toàn bộ trang web.
* **Tính linh hoạt và mở rộng**: ReactJS hỗ trợ việc tích hợp với nhiều thư viện và công cụ khác nhau, giúp phát triển các tính năng phức tạp và đáp ứng nhu cầu thay đổi linh hoạt trong quá trình phát triển.

Với ReactJS, hệ thống sẽ có một giao diện người dùng trực quan, dễ sử dụng và tương tác mượt mà, giúp người dân dễ dàng thực hiện các thao tác như đăng nhập, điền thông tin, và theo dõi tình trạng hồ sơ gia hạn hộ chiếu.

**Spring Boot** là một framework mã nguồn mở trong hệ sinh thái Spring, được thiết kế để phát triển các ứng dụng Java một cách nhanh chóng và dễ dàng, với mục tiêu đơn giản hóa quá trình cấu hình và triển khai ứng dụng. Spring Boot giúp giảm thiểu các bước cấu hình thủ công, tăng tính hiệu quả trong việc phát triển ứng dụng backend, đồng thời cung cấp các tính năng bảo mật mạnh mẽ và khả năng mở rộng.

**Đặc điểm nổi bật của Spring Boot:**

* **Tự động cấu hình**: Spring Boot cung cấp các cấu hình mặc định, giúp tiết kiệm thời gian cho lập trình viên khi xây dựng ứng dụng.
* **Khả năng tích hợp với nhiều công nghệ khác**: Spring Boot dễ dàng tích hợp với các công nghệ và thư viện khác như Hibernate, Spring Security, và các hệ quản trị cơ sở dữ liệu như OracleDB.
* **Quản lý bảo mật**: Spring Boot hỗ trợ tích hợp các cơ chế bảo mật như xác thực người dùng và phân quyền truy cập, giúp đảm bảo an toàn thông tin cho hệ thống.
* **Khả năng mở rộng và hiệu suất cao**: Với Spring Boot, ứng dụng có thể xử lý hàng nghìn yêu cầu đồng thời mà không ảnh hưởng đến hiệu suất của hệ thống.

Spring Boot sẽ đóng vai trò là nền tảng chính để xử lý các nghiệp vụ phía server, bao gồm xác thực người dùng, quản lý hồ sơ gia hạn hộ chiếu, và cung cấp các API để giao tiếp với frontend (ReactJS).

Sự kết hợp giữa **OracleDB**, **ReactJS**, và **Spring Boot** tạo thành một giải pháp toàn diện cho việc phát triển hệ thống quản lý quy trình gia hạn hộ chiếu. OracleDB sẽ đảm nhiệm việc lưu trữ và quản lý dữ liệu một cách hiệu quả và an toàn, ReactJS mang lại trải nghiệm người dùng mượt mà và linh hoạt, trong khi Spring Boot đảm bảo khả năng xử lý các yêu cầu từ người dùng với hiệu suất cao và bảo mật.

Mỗi công nghệ trong hệ thống được chọn lựa kỹ càng nhằm đáp ứng các yêu cầu về tính bảo mật, hiệu suất, và khả năng mở rộng của ứng dụng. Chúng sẽ giúp xây dựng một hệ thống web không chỉ đáp ứng nhu cầu của người dân mà còn hỗ trợ các cơ quan hành chính trong việc quản lý và xử lý các hồ sơ gia hạn hộ chiếu.

## **Chương 2: PHÂN TÍCH YÊU CẦU HỆ THỐNG.**

### **2.1 ĐẶC TẢ.**

Ứng dụng website gia hạn hộ chiếu được thiết kế để giúp công dân thực hiện quy trình gia hạn hộ chiếu một cách thuận tiện, nhanh chóng và hiệu quả. Hệ thống sử dụng công nghệ ReactJS cho giao diện người dùng, giúp giao diện trực quan, dễ sử dụng và tương tác mượt mà. Phía backend được triển khai bằng Spring Boot, cho phép xử lý các yêu cầu gia hạn hộ chiếu một cách nhanh chóng và hiệu quả, đồng thời đảm bảo tính bảo mật và khả năng mở rộng của hệ thống. Cơ sở dữ liệu của hệ thống được quản lý bằng OracleDB, một hệ quản trị cơ sở dữ liệu mạnh mẽ và bảo mật, giúp lưu trữ và xử lý thông tin người dùng và các yêu cầu gia hạn hộ chiếu một cách an toàn và chính xác.

Quy trình gia hạn hộ chiếu bắt đầu từ việc người dùng đăng nhập vào hệ thống thông qua tài khoản email hoặc mạng xã hội, sau đó điền thông tin vào mẫu đăng ký gia hạn hộ chiếu trực tuyến, bao gồm các thông tin như họ và tên, địa chỉ, số CMND, số điện thoại, email, mã số hộ chiếu, và các thông tin cá nhân khác. Dữ liệu này sau khi được nhập sẽ được gửi đến bộ phận xác thực (XT) để kiểm tra thông tin, đối chiếu với các cơ sở dữ liệu liên quan như CMND, hộ khẩu và các dữ liệu cá nhân khác. Sau khi xác thực xong, yêu cầu sẽ được chuyển tới bộ phận xét duyệt (XD) để phê duyệt hoặc từ chối yêu cầu gia hạn hộ chiếu. Nếu yêu cầu được phê duyệt, thông tin sẽ được gửi đến bộ phận lưu trữ (LT) để cập nhật thời gian gia hạn hộ chiếu mới và thông báo kết quả cho người dùng.

Đặc biệt, hệ thống cũng tích hợp chức năng giám sát (GS), cho phép các cán bộ giám sát theo dõi tiến trình xử lý yêu cầu gia hạn hộ chiếu, bao gồm các trạng thái như yêu cầu đã được xác thực, yêu cầu đang chờ xét duyệt hay đã được duyệt thành công. Quản lý toàn bộ quy trình này giúp giảm thiểu thời gian và công sức cho người dân khi làm thủ tục gia hạn hộ chiếu, đồng thời nâng cao hiệu quả công việc cho các cơ quan chức năng.

Thông qua ứng dụng website này, các cơ quan chức năng sẽ dễ dàng quản lý và giám sát toàn bộ quy trình gia hạn hộ chiếu, đồng thời công dân cũng sẽ được phục vụ nhanh chóng và hiệu quả hơn, góp phần nâng cao chất lượng dịch vụ hành chính công và giảm thiểu sự phiền toái cho người dân trong việc hoàn tất thủ tục hành chính.

### **2.2 PHÂN TÍCH THIẾT KẾ CƠ SỞ DỮ LIỆU.**

#### **2.2.1 PHÂN TÍCH THIẾT KẾ MỨC LOGIC.**

##### **2.2.1.1 Kiểu Dữ Liệu Và Ràng Buộc.**

###### **2.2.1.1.1 Resident Table.**

* ResidentID: Sử dụng kiểu dữ liệu NUMBER với tính năng GENERATED BY DEFAULT ON NULL AS IDENTITY, cho phép tự động sinh ra giá trị duy nhất khi không có giá trị nào được cung cấp trong quá trình thêm dữ liệu. Đây là khóa chính của bảng, giúp nhận diện mỗi cư dân một cách duy nhất.
* FullName: Kiểu dữ liệu VARCHAR2(100) để lưu trữ tên đầy đủ của cư dân, tối đa 100 ký tự. Trường này không được phép để trống (NOT NULL).
* Gender: Kiểu dữ liệu CHAR(1) để lưu trữ giới tính của cư dân. Chỉ cho phép giá trị 'M' (Nam) hoặc 'F' (Nữ) thông qua ràng buộc CHECK (Gender IN ('M', 'F')).
* Address: Kiểu dữ liệu VARCHAR2(255) để lưu trữ địa chỉ cư trú của cư dân, tối đa 255 ký tự.
* District: Kiểu dữ liệu VARCHAR2(50) lưu trữ tên quận hoặc huyện. Trường này không được phép để trống (NOT NULL).
* IDCardNumber: Kiểu dữ liệu VARCHAR2(15) để lưu trữ số CMND hoặc thẻ căn cước công dân, phải là duy nhất (UNIQUE) và không được phép để trống (NOT NULL).
* PhoneNumber: Kiểu dữ liệu VARCHAR2(15) lưu trữ số điện thoại của cư dân, phải duy nhất (UNIQUE), nhưng có thể để trống.
* Email: Kiểu dữ liệu VARCHAR2(100) lưu trữ địa chỉ email, phải duy nhất (UNIQUE).
* PassportNumber: Kiểu dữ liệu VARCHAR2(15) lưu trữ số hộ chiếu, phải duy nhất (UNIQUE).

###### **2.2.1.1.2 PassportRenewalRequest Table.**

* RequestID: Sử dụng kiểu dữ liệu NUMBER với GENERATED BY DEFAULT ON NULL AS IDENTITY, tự động tạo ra giá trị duy nhất khi không có giá trị cung cấp. Đây là khóa chính của bảng.
* ResidentID: Kiểu dữ liệu NUMBER, là khóa ngoại tham chiếu đến bảng Resident(ResidentID). Nếu bản ghi trong bảng Resident bị xóa, các yêu cầu gia hạn liên quan sẽ bị xóa tự động (ON DELETE CASCADE).
* SubmissionDate: Kiểu dữ liệu DATE lưu trữ ngày gửi yêu cầu gia hạn. Mặc định là ngày hiện tại (SYSDATE).
* Status: Kiểu dữ liệu VARCHAR2(20) để lưu trữ trạng thái của yêu cầu. Trạng thái phải nằm trong danh sách hợp lệ với ràng buộc CHECK (Status IN ('Pending Verification', 'Verified', 'Under Review', 'Approved', 'Rejected')).
* VerificationUnit: Kiểu dữ liệu VARCHAR2(10) lưu trữ mã đơn vị xác minh.
* ReviewComments: Kiểu dữ liệu VARCHAR2(255) lưu trữ bình luận của đơn vị xét duyệt.

###### **2.2.1.1.3 VerificationUnit Table.**

* **UnitID**: Kiểu dữ liệu VARCHAR2(10) là khóa chính, đại diện cho mã đơn vị xác minh.
* **District**: Kiểu dữ liệu VARCHAR2(50) lưu trữ tên quận hoặc huyện mà đơn vị xác minh phụ trách, phải duy nhất (UNIQUE) và không thể trống (NOT NULL).
* **Approval Table:**
* **ApprovalID**: Kiểu dữ liệu NUMBER với tính năng tự động tăng, là khóa chính của bảng.
* **RequestID**: Kiểu dữ liệu NUMBER, là khóa ngoại tham chiếu đến bảng PassportRenewalRequest(RequestID). Khi yêu cầu gia hạn bị xóa, bản ghi phê duyệt cũng sẽ bị xóa tự động (ON DELETE CASCADE).
* **IsApproved**: Kiểu dữ liệu CHAR(1), cho phép giá trị 'Y' (Phê duyệt) hoặc 'N' (Không phê duyệt), có ràng buộc CHECK (IsApproved IN ('Y', 'N')).
* **ApprovalDate**: Kiểu dữ liệu DATE lưu trữ ngày phê duyệt, mặc định là ngày hiện tại (SYSDATE).
* **ReviewerComments**: Kiểu dữ liệu VARCHAR2(255) lưu trữ nhận xét của người phê duyệt.

###### **2.2.1.1.4 Renewal Table.**

* RenewalID: Kiểu dữ liệu NUMBER với tính năng tự động tăng, là khóa chính của bảng.
* RequestID: Kiểu dữ liệu NUMBER, là khóa ngoại tham chiếu đến bảng PassportRenewalRequest(RequestID).
* NewExpiryDate: Kiểu dữ liệu DATE lưu trữ ngày hết hạn mới của hộ chiếu sau gia hạn.
* UpdatedAt: Kiểu dữ liệu DATE, mặc định là ngày hiện tại (SYSDATE), ghi nhận thời gian cập nhật hộ chiếu.

###### **2.2.1.1.5 ActivityLog Table.**

* LogID: Kiểu dữ liệu NUMBER với tính năng tự động tăng, là khóa chính của bảng.
* RequestID: Kiểu dữ liệu NUMBER, là khóa ngoại tham chiếu đến bảng PassportRenewalRequest(RequestID).
* Status: Kiểu dữ liệu VARCHAR2(20) ghi nhận trạng thái của yêu cầu gia hạn tại từng thời điểm trong quá trình xử lý.
* ActionTimestamp: Kiểu dữ liệu DATE, mặc định là ngày hiện tại (SYSDATE), ghi nhận thời gian hành động được thực hiện.
* PerformedBy: Kiểu dữ liệu VARCHAR2(50) lưu trữ tên người thực hiện hành động.

###### **2.2.1.1.6 Approval Table.**

* ApprovalID: Kiểu dữ liệu NUMBER với tính năng tự động tăng, là khóa chính của bảng.
* RequestID: Kiểu dữ liệu NUMBER, là khóa ngoại tham chiếu đến bảng PassportRenewalRequest(RequestID). Khi yêu cầu gia hạn bị xóa, bản ghi phê duyệt cũng sẽ bị xóa tự động (ON DELETE CASCADE).
* IsApproved: Kiểu dữ liệu CHAR(1), cho phép giá trị 'Y' (Phê duyệt) hoặc 'N' (Không phê duyệt), có ràng buộc CHECK (IsApproved IN ('Y', 'N')).
* ApprovalDate: Kiểu dữ liệu DATE lưu trữ ngày phê duyệt, mặc định là ngày hiện tại (SYSDATE).
* ReviewerComments: Kiểu dữ liệu VARCHAR2(255) lưu trữ nhận xét của người phê duyệt.

##### **Tạo các hàm chính sách (Policy\_XT, Policy\_XD, Policy\_LT).**

* Các hàm này là những hàm kiểm tra điều kiện truy cập dữ liệu, được sử dụng trong Oracle Database Row-Level Security (RLS) để kiểm soát quyền truy cập cấp dòng (row-level). Mỗi hàm sẽ trả về một chuỗi điều kiện (RETURN VARCHAR2) để xác định quyền truy cập.
* Cách hoạt động:
  + SYS\_CONTEXT('USERENV', 'CURRENT\_USER') lấy tên người dùng hiện tại đang thực hiện truy vấn.
  + Kiểm tra xem người dùng hiện tại có phải là XT\_USER, XD\_USER, hoặc LT\_USER hay không.
  + Nếu đúng:
    - Trả về '1=1' (cho phép xem toàn bộ dữ liệu).
  + Nếu sai:
    - Trả về '1=0' (không cho phép xem dữ liệu nào).

**Gán chính sách truy cập bằng DBMS\_RLS.ADD\_POLICY**

DBMS\_RLS.ADD\_POLICY được dùng để thêm các chính sách bảo mật cấp dòng vào một bảng trong cơ sở dữ liệu. Mỗi bảng có một hàm chính sách tương ứng.

* **Các bảng và chính sách:**
  + **Bảng Resident:**
    - Chỉ XT\_USER mới có thể truy cập dữ liệu.
    - Hàm sử dụng: Policy\_XT.
  + **Bảng PassportRenewalRequest:**
    - Chỉ XD\_USER mới có thể truy cập dữ liệu.
    - Hàm sử dụng: Policy\_XD.
  + **Bảng Approval:**
    - Chỉ LT\_USER mới có thể truy cập dữ liệu (cả **SELECT** và **UPDATE**).
    - Hàm sử dụng: Policy\_LT.
  + **Bảng Renewal:**
    - Chỉ LT\_USER mới có thể truy cập dữ liệu (**SELECT**).
    - Hàm sử dụng: Policy\_LT.
* **Tham số chính:**
  + object\_schema: Schema chứa bảng (ở đây là system hoặc sys).
  + object\_name: Tên bảng áp dụng chính sách.
  + policy\_name: Tên chính sách (mỗi bảng có một chính sách riêng).
  + function\_schema: Schema chứa hàm chính sách (ở đây là sys).
  + policy\_function: Hàm được dùng để kiểm tra quyền truy cập (ví dụ: Policy\_XT).
  + statement\_types: Loại câu lệnh bị áp dụng chính sách (ví dụ: SELECT, UPDATE).

**Gán giám sát hành động (AUDIT)**

Câu lệnh AUDIT trong Oracle Database được sử dụng để theo dõi các hành động của người dùng trên các bảng.

* **Các hành động được giám sát:**
  + **Các hành động trên bảng (TABLE):**
    - SELECT, UPDATE, INSERT, DELETE được giám sát với các user: XT\_USER, GS\_USER, XD\_USER, LT\_USER.
  + **Tạo bảng (CREATE TABLE):**
    - CREATE TABLE được giám sát với các user: XT\_USER, GS\_USER, LT\_USER.
  + **Xóa bảng (DROP ANY TABLE):**
    - DROP ANY TABLE được giám sát với các user: XT\_USER, GS\_USER, LT\_USER.

##### **Truy vấn thông tin audit.**

|  |
| --- |
| SELECT USERNAME, ACTION\_NAME  FROM DBA\_AUDIT\_TRAIL  WHERE USERNAME = 'XT\_USER'; |

Trả về các hành động (cột ACTION\_NAME) mà user XT\_USER đã thực hiện.

|  |
| --- |
| SELECT USERNAME, TIMESTAMP, OBJ\_NAME, ACTION\_NAME  FROM DBA\_AUDIT\_TRAIL  WHERE USERNAME = 'LT\_USER'; |

Trả về thông tin chi tiết hơn: tên user, thời gian thực hiện (TIMESTAMP), tên đối tượng (OBJ\_NAME), và loại hành động (ACTION\_NAME).

#### **2.2.3 PHÂN TÍCH THIẾT KẾ MỨC VẬT LÝ.**

**Mối quan hệ:**

**Bảng Resident và bảng PassportRenewalRequest:**

* Quan hệ: Một cư dân (Resident) có thể có nhiều yêu cầu gia hạn hộ chiếu (PassportRenewalRequest). Đây là một mối quan hệ 1-N (một cư dân có thể có nhiều yêu cầu gia hạn, nhưng mỗi yêu cầu gia hạn chỉ thuộc về một cư dân).
* Khóa ngoại: Trường ResidentID trong bảng PassportRenewalRequest tham chiếu đến ResidentID trong bảng Resident. Khi bản ghi trong bảng Resident bị xóa, các bản ghi liên quan trong bảng PassportRenewalRequest cũng sẽ bị xóa tự động nhờ ràng buộc ON DELETE CASCADE.

**Bảng PassportRenewalRequest và bảng Approval:**

* Quan hệ: Một yêu cầu gia hạn hộ chiếu có thể có một hoặc nhiều phê duyệt (Approval). Đây là mối quan hệ 1-N (một yêu cầu gia hạn có thể có nhiều phê duyệt trong quá trình xử lý, nhưng mỗi phê duyệt chỉ liên kết với một yêu cầu gia hạn cụ thể).
* Khóa ngoại: Trường RequestID trong bảng Approval tham chiếu đến RequestID trong bảng PassportRenewalRequest. Nếu yêu cầu gia hạn bị xóa, các phê duyệt liên quan cũng sẽ bị xóa tự động (ON DELETE CASCADE).

**Bảng PassportRenewalRequest và bảng Renewal:**

* Quan hệ: Mỗi yêu cầu gia hạn hộ chiếu sẽ có một hoặc một vài bản ghi gia hạn tương ứng khi yêu cầu được phê duyệt. Đây là mối quan hệ 1-1 (một yêu cầu gia hạn sẽ có một bản gia hạn sau khi yêu cầu được phê duyệt và thực hiện).
* Khóa ngoại: Trường RequestID trong bảng Renewal tham chiếu đến RequestID trong bảng PassportRenewalRequest. Nếu yêu cầu gia hạn bị xóa, bản ghi gia hạn cũng sẽ bị xóa tự động nhờ ràng buộc ON DELETE CASCADE.

**Bảng PassportRenewalRequest và bảng ActivityLog:**

* Quan hệ: Một yêu cầu gia hạn có thể có nhiều bản ghi hoạt động (ActivityLog), phản ánh các hành động diễn ra trong suốt quá trình xử lý yêu cầu gia hạn. Đây là mối quan hệ 1-N (một yêu cầu gia hạn có thể có nhiều hành động, nhưng mỗi hành động chỉ thuộc về một yêu cầu gia hạn).
* Khóa ngoại: Trường RequestID trong bảng ActivityLog tham chiếu đến RequestID trong bảng PassportRenewalRequest. Nếu yêu cầu gia hạn bị xóa, các bản ghi trong bảng ActivityLog liên quan cũng sẽ bị xóa tự động nhờ ON DELETE CASCADE.

**Bảng VerificationUnit và bảng PassportRenewalRequest:**

* Quan hệ: Mỗi yêu cầu gia hạn sẽ được xử lý bởi một đơn vị xác minh (VerificationUnit) duy nhất. Đây là mối quan hệ N-1 (nhiều yêu cầu gia hạn có thể thuộc về cùng một đơn vị xác minh, nhưng mỗi yêu cầu gia hạn chỉ có một đơn vị xác minh).
* Khóa ngoại: Trường VerificationUnit trong bảng PassportRenewalRequest tham chiếu đến UnitID trong bảng VerificationUnit. Mỗi yêu cầu gia hạn sẽ chỉ được xử lý bởi một đơn vị xác minh.

**Các bước để triển khai trong OracleDB:**

* Cài đặt Oracle Database: Nếu chưa cài đặt, bạn cần cài đặt Oracle Database trên máy chủ hoặc sử dụng dịch vụ Oracle Cloud. Đảm bảo rằng bạn có quyền truy cập vào cơ sở dữ liệu Oracle và có tài khoản người dùng đủ quyền để tạo bảng và thực thi các truy vấn SQL.
* Cài đặt Oracle SQL Developer (hoặc công cụ quản lý khác): Đây là công cụ giúp bạn kết nối và quản lý cơ sở dữ liệu Oracle từ giao diện người dùng.

**Tạo Cơ Sở Dữ Liệu (Schema)**

* **Tạo Schema**: Nếu chưa có schema cho ứng dụng, bạn cần tạo một schema mới để chứa các bảng và dữ liệu của ứng dụng.

|  |
| --- |
| CREATE USER system IDENTIFIED BY 123;  GRANT CONNECT, RESOURCE TO BMCSDL\_Project; |

##### **2.2.3.1 Resident Table.**

|  |
| --- |
| CREATE TABLE Resident (  ResidentID NUMBER GENERATED BY DEFAULT ON NULL AS IDENTITY PRIMARY KEY,  FullName VARCHAR2(100) NOT NULL,  Gender CHAR(1) CHECK (Gender IN ('M', 'F')),  Address VARCHAR2(255),  District VARCHAR2(50) NOT NULL,  IDCardNumber VARCHAR2(15) UNIQUE NOT NULL,  PhoneNumber VARCHAR2(15) UNIQUE,  Email VARCHAR2(100) UNIQUE,  PassportNumber VARCHAR2(15) UNIQUE); |

**Ví dụ:**

|  |
| --- |
| {          "residentID": 4,          "fullName": "Tao Viet Dưc",          "gender": "M",          "address": "Long An",          "district": "Duc Hoa",          "idCardNumber": "567890126",          "phoneNumber": "0971467418",          "email": "taovietduc@gmail.com",          "passportNumber": "P741370"      }, |

##### **2.2.3.2 PassportRenewalRequest Table.**

|  |
| --- |
| CREATE TABLE PassportRenewalRequest (  RequestID NUMBER GENERATED BY DEFAULT ON NULL AS IDENTITY PRIMARY KEY,  ResidentID NUMBER NOT NULL REFERENCES Resident(ResidentID) ON DELETE CASCADE,  SubmissionDate DATE DEFAULT SYSDATE NOT NULL,  Status  VARCHAR2(20) NOT NULL CHECK (Status IN (  'Pending Verification', 'Verified', 'Under Review', 'Approved', 'Rejected'  )),  VerificationUnit VARCHAR2(10),  ReviewComments VARCHAR2(255)  ); |

**Ví dụ:**

|  |
| --- |
| {          "requestID": 1,          "residentID": 1,          "submissionDate": "2024-12-10",          "status": "Pending Verification",          "verificationUnit": "V001",          "reviewComments": "Awaiting verification"      }, |

##### **2.2.3.3 VerificationUnit Table.**

|  |
| --- |
| CREATE TABLE VerificationUnit (  UnitID VARCHAR2(10) PRIMARY KEY,  District VARCHAR2(50) UNIQUE NOT NULL  ); |

##### **2.2.3.4 Renewal Table.**

|  |
| --- |
| CREATE TABLE Renewal (  RenewalID NUMBER GENERATED BY DEFAULT ON NULL AS IDENTITY PRIMARY KEY,  RequestID NUMBER NOT NULL REFERENCES PassportRenewalRequest(RequestID) ON DELETE CASCADE,  NewExpiryDate DATE NOT NULL,  UpdatedAt DATE DEFAULT SYSDATE NOT NULL  ); |

**Ví dụ:**

|  |
| --- |
| {          "renewalID": 1,          "requestID": 2,          "newExpiryDate": "2029-12-11",          "updatedAt": "2024-12-16T09:37:37.000+00:00"      },      {          "renewalID": 2,          "requestID": 3,          "newExpiryDate": "2030-12-12",          "updatedAt": "2024-12-16T09:37:37.000+00:00"      }, |

##### **2.2.3.5 ActivityLog Table.**

|  |
| --- |
| CREATE TABLE ActivityLog (  LogID NUMBER GENERATED BY DEFAULT ON NULL AS IDENTITY PRIMARY KEY,  RequestID NUMBER NOT NULL REFERENCES PassportRenewalRequest(RequestID) ON DELETE CASCADE,  Status VARCHAR2(20) NOT NULL,  ActionTimestamp DATE DEFAULT SYSDATE NOT NULL,  PerformedBy VARCHAR2(50) NOT NULL  ); |

**Ví dụ:**

|  |
| --- |
| {          "requestid": 1,          "status": "Pending Verification",          "performedby": "admin",          "actiontimestamp": "2024-12-16T09:38:10.000+00:00",          "logid": 1      },      {          "requestid": 2,          "status": "Verified",          "performedby": "admin",          "actiontimestamp": "2024-12-16T09:38:10.000+00:00",          "logid": 2      }, |

##### **2.2.3.6 ActivityLog Table.**

|  |
| --- |
| CREATE TABLE Approval (  ApprovalID NUMBER GENERATED BY DEFAULT ON NULL AS IDENTITY PRIMARY KEY,  RequestID NUMBER NOT NULL REFERENCES PassportRenewalRequest(RequestID) ON DELETE CASCADE,  IsApproved CHAR(1) NOT NULL CHECK (IsApproved IN ('Y', 'N')),  ApprovalDate DATE DEFAULT SYSDATE NOT NULL,  ReviewerComments VARCHAR2(255)  ); |

Ví dụ:

|  |
| --- |
| {          "approvalId": 1,          "requestId": 1,          "isApproved": "N",          "approvalDate": "2024-12-12",          "reviewerComments": "Approved after verification"      },      {          "approvalId": 2,          "requestId": 2,          "isApproved": "N",          "approvalDate": "2024-12-13",          "reviewerComments": "Rejected due to missing documents"      }, |

##### **2.2.3.7 Policy.**

|  |
| --- |
| CREATE OR REPLACE FUNCTION Policy\_XT (schema\_name IN VARCHAR2, table\_name IN VARCHAR2)  RETURN VARCHAR2 IS  BEGIN  -- Ki?m tra n?u user hi?n t?i l? XT\_USER  IF SYS\_CONTEXT('USERENV', 'CURRENT\_USER') = 'XT\_USER' THEN  RETURN '1=1'; -- Cho ph?p xem t?t c? d? li?u  ELSE  RETURN '1=0'; -- Kh?ng cho ph?p xem  END IF;  END;  CREATE OR REPLACE FUNCTION Policy\_XD (schema\_name IN VARCHAR2, table\_name IN VARCHAR2)  RETURN VARCHAR2 IS  BEGIN  -- Ki?m tra n?u user hi?n t?i l? XT\_USER  IF SYS\_CONTEXT('USERENV', 'CURRENT\_USER') = 'XD\_USER' THEN  RETURN '1=1'; -- Cho ph?p xem t?t c? d? li?u  ELSE  RETURN '1=0'; -- Kh?ng cho ph?p xem  END IF;  END;  CREATE OR REPLACE FUNCTION Policy\_LT (schema\_name IN VARCHAR2, table\_name IN VARCHAR2)  RETURN VARCHAR2 IS  BEGIN  -- Ki?m tra n?u user hi?n t?i l? XT\_USER  IF SYS\_CONTEXT('USERENV', 'CURRENT\_USER') = 'LT\_USER' THEN  RETURN '1=1'; -- Cho ph?p xem t?t c? d? li?u  ELSE  RETURN '1=0'; -- Kh?ng cho ph?p xem  END IF;  END;  BEGIN  -- G?n policy cho XT  DBMS\_RLS.ADD\_POLICY(  object\_schema => 'system',  object\_name => 'Resident',  policy\_name => 'XT\_only',  function\_schema => 'sys',  policy\_function => 'Policy\_XT',  statement\_types => 'SELECT',  update\_check => TRUE  );  end;  BEGIN  -- G?n policy cho XD  DBMS\_RLS.ADD\_POLICY(  object\_schema => 'sys',  object\_name => 'PassportRenewalRequest',  policy\_name => 'XD\_only',  function\_schema => 'sys',  policy\_function => 'Policy\_XD',  statement\_types => 'SELECT',  update\_check => TRUE  );  end;  BEGIN  -- G?n policy cho LT  DBMS\_RLS.ADD\_POLICY(  object\_schema => 'sys',  object\_name => 'Approval',  policy\_name => 'LT\_only',  function\_schema => 'sys',  policy\_function => 'Policy\_LT',  statement\_types => 'SELECT, UPDATE',  update\_check => TRUE  );  end;  Begin  -- G?n policy cho GS  DBMS\_RLS.ADD\_POLICY(  object\_schema => 'sys',  object\_name => 'Renewal',  policy\_name => 'LT\_only',  function\_schema => 'sys',  policy\_function => 'Policy\_LT',  statement\_types => 'SELECT',  update\_check => TRUE  );  END; |

##### **2.2.3.8 Audit.**

|  |
| --- |
| --Gán giám sát lên các user XT, GS, LT, XD  AUDIT SELECT TABLE, UPDATE TABLE, INSERT TABLE, DELETE TABLE BY XT\_USER, GS\_USER, XD\_USER, LT\_USER;  AUDIT CREATE TABLE BY XT\_USER, GS\_USER, LT\_USER;  AUDIT DROP ANY TABLE BY XT\_USER, GS\_USER, LT\_USER;  --Xem các hành động mà user đó đã thực hiện trên các bảng  SELECT USERNAME, ACTION\_NAME FROM DBA\_AUDIT\_TRAIL  WHERE USERNAME = 'XT\_USER';  select username, timestamp, obj\_name, action\_name from dba\_audit\_trail where username = 'LT\_USER'; |

##### **2.2.3.9 TRIGGER.**

TRIGGER request\_id\_trigger

|  |
| --- |
| CREATE SEQUENCE request\_id\_seq START WITH 1 INCREMENT BY 1;  CREATE OR REPLACE TRIGGER request\_id\_trigger  BEFORE INSERT ON APPROVAL  FOR EACH ROW  BEGIN  IF :new.REQUESTID IS NULL THEN  SELECT request\_id\_seq.NEXTVAL INTO :new.REQUESTID FROM dual;  END IF;  END; |

Khi thêm một dòng mới vào bảng APPROVAL mà không cung cấp giá trị cho cột REQUESTID, trigger sẽ tự động:

* Gọi sequence request\_id\_seq để lấy giá trị tiếp theo.
* Gán giá trị này vào cột REQUESTID của dòng mới.

|  |
| --- |
| CREATE SEQUENCE ActivityLog\_seq  START WITH 1  INCREMENT BY 1  NOCACHE;  -- Triggers to maintain security and audit logs  CREATE OR REPLACE TRIGGER trg\_VerifyStatusChange  BEFORE UPDATE ON PassportRenewalRequest  FOR EACH ROW  BEGIN  INSERT INTO ActivityLog (LogID, RequestID, Status, PerformedBy)  VALUES (ActivityLog\_seq.NEXTVAL, :NEW.RequestID, :NEW.Status, USER);  END;  / |

Mỗi khi một bản ghi trong bảng PassportRenewalRequest được cập nhật:

* Trigger sẽ kích hoạt trước khi thay đổi được thực hiện.
* Thêm một dòng vào bảng ActivityLog với thông tin về thay đổi trạng thái, bao gồm:
  + ID của log (LogID).
  + ID của yêu cầu (RequestID).
  + Trạng thái mới (Status).
  + Người thực hiện thay đổi (PerformedBy).

|  |
| --- |
| CREATE SYNONYM XT\_Resident FOR Resident;  CREATE SYNONYM XT\_PassportRenewalRequest FOR PassportRenewalRequest;  CREATE SYNONYM XD\_PassportRenewalRequest FOR PassportRenewalRequest;  CREATE SYNONYM LT\_Renewal FOR Renewal;  CREATE SYNONYM LT\_Approval FOR Approval; |

sử dụng synonyms (bí danh) để tạo các tên thay thế cho các đối tượng cơ sở dữ liệu

* XT\_Resident: Bí danh cho bảng Resident.
* XT\_PassportRenewalRequest và XD\_PassportRenewalRequest: Hai bí danh khác nhau cùng trỏ đến bảng PassportRenewalRequest.
* LT\_Renewal và LT\_Approval: Bí danh cho bảng Renewal và Approval.

## **Chương 3: CÀI ĐẶT MẠNG HỆ THỐNG**

### **3.1 Môi trường cài đặt.**

##### **3.1.2.1 Frontend Development.**

**Node.js**

* **Phiên bản yêu cầu**: 18.x hoặc mới hơn.
* **Mục đích**: Node.js được sử dụng để cài đặt các thư viện frontend như ReactJS và quản lý chúng thông qua npm (Node Package Manager). Nó cung cấp các công cụ và môi trường cần thiết để phát triển giao diện người dùng và các ứng dụng web hiệu năng cao.
* **Tải xuống**: [Node.js](https://nodejs.org)

A screenshot of a computer

Description automatically generated

**ReactJS**

* **Mục đích**: Trong mạng xã hội, ReactJS được dùng để xây dựng các tính năng như trang cá nhân, diễn đàn, hệ thống thông báo, và các giao diện tương tác khác.
* **Cách cài đặt**: Cài đặt ReactJS thông qua **npm**, đi kèm với Node.js.

**Ví dụ, sử dụng lệnh:**

|  |
| --- |
| npx create-react-app my-app  cd my-app  npm start |

A screenshot of a computer

Description automatically generated

##### **3.1.2.3 Backend Development.**

**Java Development Kit (JDK)**

* **Phiên bản yêu cầu**: 17 hoặc cao hơn.
* **Mục đích**: JDK là bộ công cụ cần thiết để phát triển và chạy các ứng dụng Java, bao gồm trình biên dịch (javac) và môi trường runtime (JRE). Spring Boot, framework chính được sử dụng trong backend của mạng xã hội, yêu cầu JDK để hoạt động. JDK 17 là phiên bản Long Term Support (LTS), đảm bảo hiệu năng và bảo mật cao.
* **Tải xuống**: [Java JDK](https://www.oracle.com/java/technologies/javase-downloads.html)

A screenshot of a computer

Description automatically generated

**Spring Boot**

* **Mục đích**: Spring Boot là một framework phát triển backend được sử dụng để xây dựng các API RESTful. Nó cung cấp nhiều thư viện tích hợp sẵn, giúp giảm bớt thời gian cấu hình và triển khai và để quản lý các tính năng của mạng xã hội như đăng ký, đăng nhập, và xử lý dữ liệu thời gian thực.
* **Cách cài đặt**: Spring Boot được cài đặt và quản lý thông qua **Maven** hoặc **Gradle**, các công cụ build phổ biến cho Java.

**A screenshot of a computer

Description automatically generated**

##### **3.1.2.4 IDE và Trình Soạn Thảo.**

**Visual Studio Code**

* **Phiên bản yêu cầu**: Mới nhất.
* **Mục đích**: Visual Studio Code (VS Code) là một trình soạn nhẹ và mạnh mẽ, hỗ trợ nhiều ngôn ngữ lập trình, bao gồm JavaScript, TypeScript, và Java. VS Code cung cấp nhiều tiện ích như debugging, terminal tích hợp, giúp tối ưu hóa quá trình phát triển frontend.
* **Tải xuống**: [Visual Studio Code](https://code.visualstudio.com)

A screenshot of a computer

Description automatically generated

**IntelliJ IDEA**

* **Phiên bản yêu cầu**: Tương thích với Windows.
* **Mục đích**: IntelliJ IDEA là một IDE mạnh mẽ dành cho Java để phát triển backend Spring Boot nhờ khả năng hỗ trợ cấu hình Maven, Gradle, và các framework Spring.
* **Tải xuống**: [IntelliJ IDEA](https://www.jetbrains.com/idea/)

A screenshot of a computer

Description automatically generated

##### **4.1.2.5 Công cụ bổ trợ.**

**Postman**

* **Mục đích**: Postman là công cụ kiểm thử API dùng để gửi yêu cầu (GET, POST, PUT, DELETE) tới các endpoint của backend và kiểm tra phản hồi từ máy chủ. Trong dự án mạng xã hội, Postman được sử dụng để kiểm thử các API quan trọng như đăng nhập, đăng bài, và tương tác thời gian thực.
* **Tải xuống**: [Postman](https://www.postman.com/)

A screenshot of a computer

Description automatically generated

### **3.2 Cài đặt cơ sở dữ liệu.**

|  |
| --- |
| CREATE TABLE Resident (  ResidentID NUMBER GENERATED BY DEFAULT ON NULL AS IDENTITY PRIMARY KEY, -- S? d?ng IDENTITY cho t? ??ng t?ng  FullName VARCHAR2(100) NOT NULL,  Gender CHAR(1) CHECK (Gender IN ('M', 'F')), -- Ràng bu?c gi?i tính  Address VARCHAR2(255),  District VARCHAR2(50) NOT NULL,  IDCardNumber VARCHAR2(15) UNIQUE NOT NULL,  PhoneNumber VARCHAR2(15) UNIQUE, -- S? ?i?n tho?i ph?i duy nh?t (n?u c?n)  Email VARCHAR2(100) UNIQUE, -- Email ph?i duy nh?t  PassportNumber VARCHAR2(15) UNIQUE -- S? h? chi?u duy nh?t (n?u c?n)  );  CREATE TABLE PassportRenewalRequest (  RequestID NUMBER GENERATED BY DEFAULT ON NULL AS IDENTITY PRIMARY KEY,  ResidentID NUMBER NOT NULL REFERENCES Resident(ResidentID) ON DELETE CASCADE, -- T? ??ng xóa khi Resident b? xóa  SubmissionDate DATE DEFAULT SYSDATE NOT NULL,  Status VARCHAR2(20) NOT NULL CHECK (Status IN (  'Pending Verification', 'Verified', 'Under Review', 'Approved', 'Rejected'  )), -- Tr?ng thái ph?i h?p l?  VerificationUnit VARCHAR2(10),  ReviewComments VARCHAR2(255)  );  CREATE TABLE VerificationUnit (  UnitID VARCHAR2(10) PRIMARY KEY,  District VARCHAR2(50) UNIQUE NOT NULL -- M?i District ch? có m?t ??n v? xác minh  );  CREATE TABLE Approval (  ApprovalID NUMBER GENERATED BY DEFAULT ON NULL AS IDENTITY PRIMARY KEY, -- ID t? ??ng t?ng  RequestID NUMBER NOT NULL REFERENCES PassportRenewalRequest(RequestID) ON DELETE CASCADE,  IsApproved CHAR(1) NOT NULL CHECK (IsApproved IN ('Y', 'N')), -- Ch? Y/N  ApprovalDate DATE DEFAULT SYSDATE NOT NULL,  ReviewerComments VARCHAR2(255)  );  CREATE TABLE Renewal (  RenewalID NUMBER GENERATED BY DEFAULT ON NULL AS IDENTITY PRIMARY KEY, -- ID t? ??ng t?ng  RequestID NUMBER NOT NULL REFERENCES PassportRenewalRequest(RequestID) ON DELETE CASCADE,  NewExpiryDate DATE NOT NULL, -- Ngày h?t h?n m?i  UpdatedAt DATE DEFAULT SYSDATE NOT NULL  );  CREATE TABLE ActivityLog (  LogID NUMBER GENERATED BY DEFAULT ON NULL AS IDENTITY PRIMARY KEY, -- ID t? ??ng t?ng  RequestID NUMBER NOT NULL REFERENCES PassportRenewalRequest(RequestID) ON DELETE CASCADE,  Status VARCHAR2(20) NOT NULL, -- Tr?ng thái liên quan  ActionTimestamp DATE DEFAULT SYSDATE NOT NULL,  PerformedBy VARCHAR2(50) NOT NULL -- Ng??i th?c hi?n hành ??ng  );  CREATE SEQUENCE ActivityLog\_seq  START WITH 1  INCREMENT BY 1  NOCACHE;  --policy  CREATE OR REPLACE FUNCTION Policy\_XT (schema\_name IN VARCHAR2, table\_name IN VARCHAR2)  RETURN VARCHAR2 IS  BEGIN  -- Ki?m tra n?u user hi?n t?i l? XT\_USER  IF SYS\_CONTEXT('USERENV', 'CURRENT\_USER') = 'XT\_USER' THEN  RETURN '1=1'; -- Cho ph?p xem t?t c? d? li?u  ELSE  RETURN '1=0'; -- Kh?ng cho ph?p xem  END IF;  END;  CREATE OR REPLACE FUNCTION Policy\_XD (schema\_name IN VARCHAR2, table\_name IN VARCHAR2)  RETURN VARCHAR2 IS  BEGIN  -- Ki?m tra n?u user hi?n t?i l? XT\_USER  IF SYS\_CONTEXT('USERENV', 'CURRENT\_USER') = 'XD\_USER' THEN  RETURN '1=1'; -- Cho ph?p xem t?t c? d? li?u  ELSE  RETURN '1=0'; -- Kh?ng cho ph?p xem  END IF;  END;  CREATE OR REPLACE FUNCTION Policy\_LT (schema\_name IN VARCHAR2, table\_name IN VARCHAR2)  RETURN VARCHAR2 IS  BEGIN  -- Ki?m tra n?u user hi?n t?i l? XT\_USER  IF SYS\_CONTEXT('USERENV', 'CURRENT\_USER') = 'LT\_USER' THEN  RETURN '1=1'; -- Cho ph?p xem t?t c? d? li?u  ELSE  RETURN '1=0'; -- Kh?ng cho ph?p xem  END IF;  END;  BEGIN  -- G?n policy cho XT  DBMS\_RLS.ADD\_POLICY(  object\_schema => 'system',  object\_name => 'Resident',  policy\_name => 'XT\_only',  function\_schema => 'sys',  policy\_function => 'Policy\_XT',  statement\_types => 'SELECT',  update\_check => TRUE  );  end;  BEGIN  -- G?n policy cho XD  DBMS\_RLS.ADD\_POLICY(  object\_schema => 'sys',  object\_name => 'PassportRenewalRequest',  policy\_name => 'XD\_only',  function\_schema => 'sys',  policy\_function => 'Policy\_XD',  statement\_types => 'SELECT',  update\_check => TRUE  );  end;  BEGIN  -- G?n policy cho LT  DBMS\_RLS.ADD\_POLICY(  object\_schema => 'sys',  object\_name => 'Approval',  policy\_name => 'LT\_only',  function\_schema => 'sys',  policy\_function => 'Policy\_LT',  statement\_types => 'SELECT, UPDATE',  update\_check => TRUE  );  end;  Begin  -- G?n policy cho GS  DBMS\_RLS.ADD\_POLICY(  object\_schema => 'sys',  object\_name => 'Renewal',  policy\_name => 'LT\_only',  function\_schema => 'sys',  policy\_function => 'Policy\_LT',  statement\_types => 'SELECT',  update\_check => TRUE  );  END;  --AUDIT  --Gán giám sát lên các user XT, GS, LT, XD  AUDIT SELECT TABLE, UPDATE TABLE, INSERT TABLE, DELETE TABLE BY XT\_USER, GS\_USER, XD\_USER, LT\_USER;  AUDIT CREATE TABLE BY XT\_USER, GS\_USER, LT\_USER;  AUDIT DROP ANY TABLE BY XT\_USER, GS\_USER, LT\_USER;  --Xem các hành động mà user đó đã thực hiện trên các bảng  SELECT USERNAME, ACTION\_NAME FROM DBA\_AUDIT\_TRAIL  WHERE USERNAME = 'XT\_USER';  select username, timestamp, obj\_name, action\_name from dba\_audit\_trail where username = 'LT\_USER';   CREATE SEQUENCE request\_id\_seq START WITH 1 INCREMENT BY 1;  CREATE OR REPLACE TRIGGER request\_id\_trigger  BEFORE INSERT ON APPROVAL  FOR EACH ROW  BEGIN  IF :new.REQUESTID IS NULL THEN  SELECT request\_id\_seq.NEXTVAL INTO :new.REQUESTID FROM dual;  END IF;  END;  CREATE SEQUENCE ActivityLog\_seq  START WITH 1  INCREMENT BY 1  NOCACHE;  -- Triggers to maintain security and audit logs  CREATE OR REPLACE TRIGGER trg\_VerifyStatusChange  BEFORE UPDATE ON PassportRenewalRequest  FOR EACH ROW  BEGIN  INSERT INTO ActivityLog (LogID, RequestID, Status, PerformedBy)  VALUES (ActivityLog\_seq.NEXTVAL, :NEW.RequestID, :NEW.Status, USER);  END;  /  CREATE SYNONYM XT\_Resident FOR Resident;  CREATE SYNONYM XT\_PassportRenewalRequest FOR PassportRenewalRequest;  CREATE SYNONYM XD\_PassportRenewalRequest FOR PassportRenewalRequest;  CREATE SYNONYM LT\_Renewal FOR Renewal;  CREATE SYNONYM LT\_Approval FOR Approval;  CREATE OR REPLACE FUNCTION renewal\_request\_policy (schema\_name IN VARCHAR2, table\_name IN VARCHAR2)  RETURN VARCHAR2 IS  BEGIN  IF (USER = 'XD\_USER') THEN  RETURN 'Status IN (''Pending Verification'', ''Verified'', ''Under Review'', ''Approved'')'; -- Chỉ xem các yêu cầu chưa phê duyệt hoặc đã xác thực  ELSE  RETURN '1 = 0'; -- Không cho phép bất kỳ dữ liệu nào  END IF;  END renewal\_request\_policy;  /  INSERT INTO PassportRenewalRequest (ResidentID, SubmissionDate, Status, VerificationUnit, ReviewComments)  VALUES (1, TO\_DATE('10-DEC-24', 'DD-MON-YY'), 'Pending Verification', 'V001', 'Awaiting verification');  INSERT INTO PassportRenewalRequest (ResidentID, SubmissionDate, Status, VerificationUnit, ReviewComments)  VALUES (2, TO\_DATE('11-DEC-24', 'DD-MON-YY'), 'Verified', 'V002', 'All details verified');  INSERT INTO PassportRenewalRequest (ResidentID, SubmissionDate, Status, VerificationUnit, ReviewComments)  VALUES (3, TO\_DATE('12-DEC-24', 'DD-MON-YY'), 'Under Review', 'V003', 'Pending approval');  INSERT INTO PassportRenewalRequest (ResidentID, SubmissionDate, Status, VerificationUnit, ReviewComments)  VALUES (5, TO\_DATE('16-DEC-24', 'DD-MON-YY'), 'Under Review', 'V005', 'Review in progress');  INSERT INTO PassportRenewalRequest (ResidentID, SubmissionDate, Status, VerificationUnit, ReviewComments)  VALUES (6, TO\_DATE('17-DEC-24', 'DD-MON-YY'), 'Approved', 'V006', 'Request approved successfully');  INSERT INTO Resident (ResidentID, FullName, Gender, Address, District, IDCardNumber, PhoneNumber, Email, PassportNumber)  VALUES (1, 'Nguyen Minh Tu', 'M', '123 Tran Hung Dao', 'District A', '123456789', '0123456789', 'tu.nguyen@example.com', 'P987654');  INSERT INTO Resident (ResidentID, FullName, Gender, Address, District, IDCardNumber, PhoneNumber, Email, PassportNumber)  VALUES (2, 'Tran Thi Mai', 'F', '456 Le Lai', 'District B', '987654321', '0234567890', 'mai.tran@example.com', 'P321456');  INSERT INTO Resident (ResidentID, FullName, Gender, Address, District, IDCardNumber, PhoneNumber, Email, PassportNumber)  VALUES (3, 'Hoang Thanh Son', 'M', '789 Hoang Hoa Tham', 'District C', '135792468', '0345678901', 'son.hoang@example.com', 'P654789');  INSERT INTO Resident (ResidentID, FullName, Gender, Address, District, IDCardNumber, PhoneNumber, Email, PassportNumber)  VALUES (5, 'Nguyen Hoai Thu', 'F', '456 Nguyen Trai', 'District B', '234567890', '0456789012', 'thu.nguyen@example.com', 'P987321');  INSERT INTO Resident (ResidentID, FullName, Gender, Address, District, IDCardNumber, PhoneNumber, Email, PassportNumber)  VALUES (6, 'Phan Duy Khanh', 'M', '789 Tan Phu', 'District C', '567890123', '0567890123', 'khannhphan@example.com', 'P741369');  INSERT INTO Resident (ResidentID, FullName, Gender, Address, District, IDCardNumber, PhoneNumber, Email, PassportNumber)  VALUES (4, 'Tao Viet D?c', 'M', 'Long An', 'Duc Hoa', '567890126', '0971467418', 'taovietduc@gmail.com', 'P741370');  -- D? li?u cho b?ng VerificationUnit  DELETE FROM VerificationUnit WHERE UnitID IN ('V001', 'V002', 'V003', 'V004', 'V005', 'V006');  INSERT INTO VerificationUnit (UnitID, District) VALUES ('V001', 'District A');  INSERT INTO VerificationUnit (UnitID, District) VALUES ('V002', 'District B');  INSERT INTO VerificationUnit (UnitID, District) VALUES ('V003', 'District C');  INSERT INTO VerificationUnit (UnitID, District) VALUES ('V004', 'District D');  INSERT INTO VerificationUnit (UnitID, District) VALUES ('V005', 'District E');  INSERT INTO VerificationUnit (UnitID, District) VALUES ('V006', 'District F');  SELECT \* FROM VerificationUnit;  -- D? li?u cho b?ng PassportRenewalRequest  INSERT INTO PassportRenewalRequest (RequestID, ResidentID, SubmissionDate, Status, VerificationUnit, ReviewComments)  VALUES (1, 1, TO\_DATE('2024-12-10', 'YYYY-MM-DD'), 'Pending Verification', 'V001', 'Awaiting verification');  INSERT INTO PassportRenewalRequest (RequestID, ResidentID, SubmissionDate, Status, VerificationUnit, ReviewComments)  VALUES (2, 2, TO\_DATE('2024-12-11', 'YYYY-MM-DD'), 'Verified', 'V002', 'All details verified');  INSERT INTO PassportRenewalRequest (RequestID, ResidentID, SubmissionDate, Status, VerificationUnit, ReviewComments)  VALUES (3, 3, TO\_DATE('2024-12-12', 'YYYY-MM-DD'), 'Under Review', 'V003', 'Pending approval');  INSERT INTO PassportRenewalRequest (RequestID, ResidentID, SubmissionDate, Status, VerificationUnit, ReviewComments)  VALUES (5, 5, TO\_DATE('2024-12-16', 'YYYY-MM-DD'), 'Under Review', 'V005', 'Review in progress');  INSERT INTO PassportRenewalRequest (RequestID, ResidentID, SubmissionDate, Status, VerificationUnit, ReviewComments)  VALUES (6, 6, TO\_DATE('2024-12-17', 'YYYY-MM-DD'), 'Approved', 'V006', 'Request approved successfully');  SELECT \* FROM PassportRenewalRequest;  -- D? li?u cho b?ng Approval  INSERT INTO Approval (ApprovalID, RequestID, IsApproved, ApprovalDate, ReviewerComments)  VALUES (1, 1, 'Y', TO\_DATE('2024-12-12', 'YYYY-MM-DD'), 'Approved after verification');  INSERT INTO Approval (ApprovalID, RequestID, IsApproved, ApprovalDate, ReviewerComments)  VALUES (2, 2, 'N', TO\_DATE('2024-12-13', 'YYYY-MM-DD'), 'Rejected due to missing documents');  INSERT INTO Approval (ApprovalID, RequestID, IsApproved, ApprovalDate, ReviewerComments)  VALUES (4, 5, 'Y', TO\_DATE('2024-12-17', 'YYYY-MM-DD'), 'Approved after verification');  SELECT \* FROM Approval;  -- D? li?u cho b?ng Renewal  INSERT INTO Renewal (RenewalID, RequestID, NewExpiryDate, UpdatedAt)  VALUES (1, 2, TO\_DATE('2029-12-11', 'YYYY-MM-DD'), SYSDATE);  INSERT INTO Renewal (RenewalID, RequestID, NewExpiryDate, UpdatedAt)  VALUES (2, 3, TO\_DATE('2030-12-12', 'YYYY-MM-DD'), SYSDATE);  INSERT INTO Renewal (RenewalID, RequestID, NewExpiryDate, UpdatedAt)  VALUES (3, 5, TO\_DATE('2029-12-16', 'YYYY-MM-DD'), SYSDATE);  INSERT INTO Renewal (RenewalID, RequestID, NewExpiryDate, UpdatedAt)  VALUES (4, 6, TO\_DATE('2031-12-17', 'YYYY-MM-DD'), SYSDATE);  SELECT \* FROM Renewal;  -- D? li?u cho b?ng ActivityLog  INSERT INTO ActivityLog (LogID, RequestID, Status, ActionTimestamp, PerformedBy)  VALUES (1, 1, 'Pending Verification', SYSDATE, 'admin');  INSERT INTO ActivityLog (LogID, RequestID, Status, ActionTimestamp, PerformedBy)  VALUES (2, 2, 'Verified', SYSDATE, 'admin');  INSERT INTO ActivityLog (LogID, RequestID, Status, ActionTimestamp, PerformedBy)  VALUES (3, 3, 'Under Review', SYSDATE, 'reviewer1');  INSERT INTO ActivityLog (LogID, RequestID, Status, ActionTimestamp, PerformedBy)  VALUES (5, 5, 'Under Review', SYSDATE, 'reviewer3');  INSERT INTO ActivityLog (LogID, RequestID, Status, ActionTimestamp, PerformedBy)  VALUES (6, 6, 'Approved', SYSDATE, 'reviewer4');  INSERT INTO Resident (ResidentID, FullName, Gender, Address, District, IDCardNumber, PhoneNumber, Email, PassportNumber)  VALUES (11, 'Tao Viet A', 'M', 'Long An', 'Duc Hoa', '567890130', '0971467425', 'taovieta@gmail.com', 'P7414025'); |

### **3.3 Triển khai frontend.**

#### **3.3.1 User.**

A screenshot of a video game

Description automatically generated

##### **3.3.1.1 Trang Chủ.**

A screenshot of a computer

Description automatically generated

##### **3.3.1.3 Dịch Vụ.**

A screenshot of a computer

Description automatically generated

##### **3.3.1.4 Quy Trình.**

**A screenshot of a computer

Description automatically generated**

##### **3.3.1.5 Liên Hệ.**

A screenshot of a computer

Description automatically generated

##### **3.3.1.6 Dashboard.**

A screenshot of a computer

Description automatically generated

##### **3.3.1.7 Xác Thực.**

A screenshot of a computer

Description automatically generated

##### **3.3.1.8 Xét Duyệt.**

A screenshot of a computer

Description automatically generated

##### **3.3.1.9 Lưu Trữ.**

A screenshot of a computer

Description automatically generated

##### **3.3.1.10 Giám Sát.**

**A screenshot of a computer

Description automatically generated**

##### **3.3.1.11 Form Điền Thông Tin.**

A screenshot of a computer

Description automatically generated

### **3.4 Triển khai backend.**

#### **3.4.1 Cấu hình môi trường triển khai.**

Để triển khai backend em đã set up các phần mền sau:

* 1. Cài đặt jdk-17 và setup các lớp.

**A screenshot of a computer

Description automatically generated**

* 1. Cài đặt và setup phần mềm Postman để thực hiện lấy API.

**A screenshot of a computer

Description automatically generated**

* 1. Xác định các chức năng và phân lớp cấu trúc API liên kết Frontend.

**A screenshot of a computer program

Description automatically generated**

#### **3.4.2 Lớp Entity.**

###### 3.4.2.1 Resident

|  |
| --- |
| package com.example.oracle.entity;  import jakarta.persistence.\*;  @Entity @Table(name = "Resident") public class Resident {  @Id  @GeneratedValue(strategy = GenerationType.IDENTITY)  private Long residentID;   @Column(name = "FULLNAME", nullable = false)  private String fullName;   @Column(name = "GENDER", nullable = false)  private String gender;   @Column(name = "ADDRESS")  private String address;   @Column(name = "DISTRICT", nullable = false)  private String district;   @Column(name = "IDCARDNUMBER", unique = true)  private String idCardNumber;   @Column(name = "PHONENUMBER", unique = true)  private String phoneNumber;   @Column(name = "EMAIL", unique = true)  private String email;   @Column(name = "PASSPORTNUMBER", unique = true)  private String passportNumber;   // Getters và setters  public Long getResidentID() {  return residentID;  }   public void setResidentID(Long residentID) {  this.residentID = residentID;  }   public String getFullName() {  return fullName;  }   public void setFullName(String fullName) {  this.fullName = fullName;  }   public String getGender() {  return gender;  }   public void setGender(String gender) {  this.gender = gender;  }   public String getAddress() {  return address;  }   public void setAddress(String address) {  this.address = address;  }   public String getDistrict() {  return district;  }   public void setDistrict(String district) {  this.district = district;  }   public String getIdCardNumber() {  return idCardNumber;  }   public void setIdCardNumber(String idCardNumber) {  this.idCardNumber = idCardNumber;  }   public String getPhoneNumber() {  return phoneNumber;  }   public void setPhoneNumber(String phoneNumber) {  this.phoneNumber = phoneNumber;  }   public String getEmail() {  return email;  }   public void setEmail(String email) {  this.email = email;  }   public String getPassportNumber() {  return passportNumber;  }   public void setPassportNumber(String passportNumber) {  this.passportNumber = passportNumber;  } } |

###### 3.4.2.2 PASSPORTRENEWALREQUEST

|  |
| --- |
| package com.example.oracle.entity;  import jakarta.persistence.\*; import java.util.Date;  @Entity @Table(name = "PASSPORTRENEWALREQUEST") // Tên bảng viết IN HOA public class PassportRenewalRequest {   @Id  @GeneratedValue(strategy = GenerationType.IDENTITY)  @Column(name = "REQUESTID")  private Long requestID;   @Column(name = "RESIDENTID", nullable = false)  private Long residentID;   @Column(name = "SUBMISSIONDATE", nullable = false)  @Temporal(TemporalType.DATE)  private Date submissionDate;   @Column(name = "STATUS", nullable = false, length = 20)  private String status;   @Column(name = "VERIFICATIONUNIT", length = 10)  private String verificationUnit;   @Column(name = "REVIEWCOMMENTS", length = 255)  private String reviewComments;   // Getters và Setters  public Long getRequestID() { return requestID; }  public void setRequestID(Long requestID) { this.requestID = requestID; }   public Long getResidentID() { return residentID; }  public void setResidentID(Long residentID) { this.residentID = residentID; }   public Date getSubmissionDate() { return submissionDate; }  public void setSubmissionDate(Date submissionDate) { this.submissionDate = submissionDate; }   public String getStatus() { return status; }  public void setStatus(String status) { this.status = status; }   public String getVerificationUnit() { return verificationUnit; }  public void setVerificationUnit(String verificationUnit) { this.verificationUnit = verificationUnit; }   public String getReviewComments() { return reviewComments; }  public void setReviewComments(String reviewComments) { this.reviewComments = reviewComments; } } |

###### 3.4.2.3 Approval

|  |
| --- |
| package com.example.oracle.entity;  import jakarta.persistence.\*; import java.util.Date;  @Entity @Table(name = "approval") public class Approval {   @Id  @GeneratedValue(strategy = GenerationType.IDENTITY)  @Column(name = "approvalid") // Ánh xạ chính xác với cột APPROVALID trong cơ sở dữ liệu  private Long approvalId;   @ManyToOne  @JoinColumn(name = "requestid", referencedColumnName = "requestid")  private PassportRenewalRequest passportRenewalRequest;   @Column(name = "isapproved", nullable = false)  private char isApproved;   @Temporal(TemporalType.DATE)  @Column(name = "approvaldate", nullable = false)  private Date approvalDate;   @Column(name = "reviewercomments", length = 255) // Ánh xạ với cột REVIEWERCOMMENTS trong cơ sở dữ liệu  private String reviewerComments;   // Getters and Setters  public Long getApprovalId() {  return approvalId;  }   public void setApprovalId(Long approvalId) {  this.approvalId = approvalId;  }   public PassportRenewalRequest getPassportRenewalRequest() {  return passportRenewalRequest;  }   public void setPassportRenewalRequest(PassportRenewalRequest passportRenewalRequest) {  this.passportRenewalRequest = passportRenewalRequest;  }   public char getIsApproved() {  return isApproved;  }   public void setIsApproved(char isApproved) {  this.isApproved = isApproved;  }   public Date getApprovalDate() {  return approvalDate;  }   public void setApprovalDate(Date approvalDate) {  this.approvalDate = approvalDate;  }   public String getReviewerComments() {  return reviewerComments;  }   public void setReviewerComments(String reviewerComments) {  this.reviewerComments = reviewerComments;  } } |

###### 3.4.2.4 Renewal

|  |
| --- |
| package com.example.oracle.entity;  import jakarta.persistence.\*; import java.util.Date;  @Entity @Table(name = "Renewal") public class Renewal {   @Id  @GeneratedValue(strategy = GenerationType.IDENTITY)  @Column(name = "RENEWALID") // Đổi tên cột thành chữ hoa  private Long renewalID;     @ManyToOne  @JoinColumn(name = "REQUESTID", nullable = false) // Đổi tên cột thành chữ hoa  private PassportRenewalRequest passportRenewalRequest;   @Column(name = "NEWEXPIRYDATE", nullable = false) // Đổi tên cột thành chữ hoa  @Temporal(TemporalType.DATE) // Sử dụng TemporalType.DATE cho ngày (không có thời gian)  private Date newExpiryDate;   @Column(name = "UPDATEDAT", nullable = false) // Đổi tên cột thành chữ hoa  @Temporal(TemporalType.TIMESTAMP) // Sử dụng TemporalType.TIMESTAMP cho ngày và giờ  private Date updatedAt;   // Getters and Setters  public Long getRenewalID() {  return renewalID;  }   public void setRenewalID(Long renewalID) {  this.renewalID = renewalID;  }   public PassportRenewalRequest getPassportRenewalRequest() {  return passportRenewalRequest;  }   public void setPassportRenewalRequest(PassportRenewalRequest passportRenewalRequest) {  this.passportRenewalRequest = passportRenewalRequest;  }   public Date getNewExpiryDate() {  return newExpiryDate;  }   public void setNewExpiryDate(Date newExpiryDate) {  this.newExpiryDate = newExpiryDate;  }   public Date getUpdatedAt() {  return updatedAt;  }   public void setUpdatedAt(Date updatedAt) {  this.updatedAt = updatedAt;  } } |

###### 3.4.2.5 ACTIVITYLOG

|  |
| --- |
| package com.example.oracle.entity;  import jakarta.persistence.\*; import java.util.Date;  @Entity @Table(name = "ACTIVITYLOG") // Bảng với tên IN HOA public class ActivityLog {  @Id  @GeneratedValue(strategy = GenerationType.IDENTITY)  @Column(name = "LOGID") // Tên cột IN HOA  private Long logId;   @Column(name = "REQUESTID", nullable = false) // Tên cột IN HOA  private Long requestId;   @Column(name = "STATUS", length = 20, nullable = false) // Tên cột IN HOA  private String status;   @Column(name = "ACTIONTIMESTAMP", nullable = false) // Tên cột IN HOA  @Temporal(TemporalType.TIMESTAMP)  private Date actionTimestamp;   @Column(name = "PERFORMEDBY", length = 50, nullable = false) // Tên cột IN HOA  private String performedBy;   // Getters and setters  public Long getLogId() {  return logId;  }   public void setLogId(Long logId) {  this.logId = logId;  }   public Long getRequestId() {  return requestId;  }   public void setRequestId(Long requestId) {  this.requestId = requestId;  }   public String getStatus() {  return status;  }   public void setStatus(String status) {  this.status = status;  }   public Date getActionTimestamp() {  return actionTimestamp;  }   public void setActionTimestamp(Date actionTimestamp) {  this.actionTimestamp = actionTimestamp;  }   public String getPerformedBy() {  return performedBy;  }   public void setPerformedBy(String performedBy) {  this.performedBy = performedBy;  } } |

#### **3.4.3 Lớp DTO.**

###### 3.4.3.1 ResidentDTO

|  |
| --- |
| package com.example.oracle.dto;  import com.example.oracle.entity.Resident;  public class ResidentDTO {  private Long residentID;  private String fullName;  private String gender;  private String address;  private String district;  private String idCardNumber;  private String phoneNumber;  private String email;  private String passportNumber;   // Constructor không tham số (optional, cho trường hợp khởi tạo đối tượng rỗng)  public ResidentDTO() {}   // Constructor chuyển từ Entity sang DTO  public ResidentDTO(Resident resident) {  this.residentID = resident.getResidentID();  this.fullName = safeGet(resident.getFullName());  this.gender = safeGet(resident.getGender());  this.address = safeGet(resident.getAddress());  this.district = safeGet(resident.getDistrict());  this.idCardNumber = safeGet(resident.getIdCardNumber());  this.phoneNumber = safeGet(resident.getPhoneNumber());  this.email = safeGet(resident.getEmail());  this.passportNumber = safeGet(resident.getPassportNumber());  }   // Phương thức hỗ trợ kiểm tra null và thay thế bằng "N/A"  private String safeGet(String value) {  return (value != null) ? value : "N/A";  }   // Getters  public Long getResidentID() {  return residentID;  }   public String getFullName() {  return fullName;  }   public String getGender() {  return gender;  }   public String getAddress() {  return address;  }   public String getDistrict() {  return district;  }   public String getIdCardNumber() {  return idCardNumber;  }   public String getPhoneNumber() {  return phoneNumber;  }   public String getEmail() {  return email;  }   public String getPassportNumber() {  return passportNumber;  }   // Setters  public void setResidentID(Long residentID) {  this.residentID = residentID;  }   public void setFullName(String fullName) {  this.fullName = fullName;  }   public void setGender(String gender) {  this.gender = gender;  }   public void setAddress(String address) {  this.address = address;  }   public void setDistrict(String district) {  this.district = district;  }   public void setIdCardNumber(String idCardNumber) {  this.idCardNumber = idCardNumber;  }   public void setPhoneNumber(String phoneNumber) {  this.phoneNumber = phoneNumber;  }   public void setEmail(String email) {  this.email = email;  }   public void setPassportNumber(String passportNumber) {  this.passportNumber = passportNumber;  } } |

###### 3.4.3.2 PassportRenewalRequestDTO

|  |
| --- |
| package com.example.oracle.dto;  public class PassportRenewalRequestDTO {  private Long residentID;  private String status;  private String verificationUnit;  private String reviewComments;   // Getters and Setters  public Long getResidentID() {  return residentID;  }   public void setResidentID(Long residentID) {  this.residentID = residentID;  }   public String getStatus() {  return status;  }   public void setStatus(String status) {  this.status = status;  }   public String getVerificationUnit() {  return verificationUnit;  }   public void setVerificationUnit(String verificationUnit) {  this.verificationUnit = verificationUnit;  }   public String getReviewComments() {  return reviewComments;  }   public void setReviewComments(String reviewComments) {  this.reviewComments = reviewComments;  } } |

###### 3.4.3.3 ApprovalDTO

|  |
| --- |
| package com.example.oracle.dto;  import java.util.Date;  public class ApprovalDTO {   private Long approvalId;  private Long requestId;  private char isApproved;  private Date approvalDate;  private String reviewerComments;   // Getters and Setters  public Long getApprovalId() {  return approvalId;  }   public void setApprovalId(Long approvalId) {  this.approvalId = approvalId;  }   public Long getRequestId() {  return requestId;  }   public void setRequestId(Long requestId) {  this.requestId = requestId;  }   public char getIsApproved() {  return isApproved;  }   public void setIsApproved(char isApproved) {  this.isApproved = isApproved;  }   public Date getApprovalDate() {  return approvalDate;  }   public void setApprovalDate(Date approvalDate) {  this.approvalDate = approvalDate;  }   public String getReviewerComments() {  return reviewerComments;  }   public void setReviewerComments(String reviewerComments) {  this.reviewerComments = reviewerComments;  } } |

###### 3.4.3.4 RenewalDTO

|  |
| --- |
| package com.example.oracle.dto;  import jakarta.persistence.Column; import jakarta.persistence.Entity; import jakarta.persistence.Id; import java.util.Date;  @Entity public class RenewalDTO {   @Id  @Column(name = "RENEWALID")  private Long renewalID;   @Column(name = "REQUESTID")  private Long requestID;   @Column(name = "NEWEXPIRYDATE")  private Date newExpiryDate;   @Column(name = "UPDATEDAT")  private Date updatedAt;   // Constructors  public RenewalDTO() {}   public RenewalDTO(Long renewalID, Long requestID, Date newExpiryDate, Date updatedAt) {  this.renewalID = renewalID;  this.requestID = requestID;  this.newExpiryDate = newExpiryDate;  this.updatedAt = updatedAt;  }   // Getters and Setters  public Long getRenewalID() {  return renewalID;  }   public void setRenewalID(Long renewalID) {  this.renewalID = renewalID;  }   public Long getRequestID() {  return requestID;  }   public void setRequestID(Long requestID) {  this.requestID = requestID;  }   public Date getNewExpiryDate() {  return newExpiryDate;  }   public void setNewExpiryDate(Date newExpiryDate) {  this.newExpiryDate = newExpiryDate;  }   public Date getUpdatedAt() {  return updatedAt;  }   public void setUpdatedAt(Date updatedAt) {  this.updatedAt = updatedAt;  } } |

###### 3.4.3.5 ActivityLogDTO

|  |
| --- |
| package com.example.oracle.dto;  import java.util.Date;  public class ActivityLogDTO {  private Long LOGID;  private Long REQUESTID;  private String STATUS;  private Date ACTIONTIMESTAMP;  private String PERFORMEDBY;   // Constructors, getters and setters  public ActivityLogDTO(Long LOGID, Long REQUESTID, String STATUS, Date ACTIONTIMESTAMP, String PERFORMEDBY) {  this.LOGID = LOGID;  this.REQUESTID = REQUESTID;  this.STATUS = STATUS;  this.ACTIONTIMESTAMP = ACTIONTIMESTAMP;  this.PERFORMEDBY = PERFORMEDBY;  }   public Long getLOGID() {  return LOGID;  }   public void setLOGID(Long LOGID) {  this.LOGID = LOGID;  }   public Long getREQUESTID() {  return REQUESTID;  }   public void setREQUESTID(Long REQUESTID) {  this.REQUESTID = REQUESTID;  }   public String getSTATUS() {  return STATUS;  }   public void setSTATUS(String STATUS) {  this.STATUS = STATUS;  }   public Date getACTIONTIMESTAMP() {  return ACTIONTIMESTAMP;  }   public void setACTIONTIMESTAMP(Date ACTIONTIMESTAMP) {  this.ACTIONTIMESTAMP = ACTIONTIMESTAMP;  }   public String getPERFORMEDBY() {  return PERFORMEDBY;  }   public void setPERFORMEDBY(String PERFORMEDBY) {  this.PERFORMEDBY = PERFORMEDBY;  } } |

#### **3.4.4 Lớp Repository.**

###### 3.4.4.1 ResidentRepository

|  |
| --- |
| package com.example.oracle.repository;  import com.example.oracle.entity.Resident; import org.springframework.data.jpa.repository.JpaRepository; import org.springframework.stereotype.Repository;  @Repository public interface ResidentRepository extends JpaRepository<Resident, Long> {  // JpaRepository đã cung cấp phương thức save() tự động } |

###### 3.4.4.2 PassportRenewalRequestRepository

|  |
| --- |
| package com.example.oracle.repository;  import com.example.oracle.entity.PassportRenewalRequest; import org.springframework.data.jpa.repository.JpaRepository; import org.springframework.stereotype.Repository;  @Repository public interface PassportRenewalRequestRepository extends JpaRepository<PassportRenewalRequest, Long> { } |

###### 3.4.4.3 ApprovalRepository

|  |
| --- |
| package com.example.oracle.repository;  import com.example.oracle.entity.Approval; import org.springframework.data.jpa.repository.JpaRepository; import java.util.List;  public interface ApprovalRepository extends JpaRepository<Approval, Long> {   // Lấy tất cả các phê duyệt  List<Approval> findAll();   Approval findByApprovalId(Long approvalId); } |

###### 3.4.4.4 RenewalRepository

|  |
| --- |
| package com.example.oracle.repository;  import com.example.oracle.entity.Renewal; import org.springframework.data.jpa.repository.JpaRepository; import org.springframework.stereotype.Repository;  import java.util.List;  @Repository public interface RenewalRepository extends JpaRepository<Renewal, Long> {   List<Renewal> findAll(); } |

###### 3.4.4.5 ActivityLogRepository

|  |
| --- |
| package com.example.oracle.repository;  import com.example.oracle.entity.ActivityLog; import org.springframework.data.jpa.repository.JpaRepository; import org.springframework.stereotype.Repository;  @Repository public interface ActivityLogRepository extends JpaRepository<ActivityLog, Long> { } |

#### **3.4.5 Lớp Service.**

###### 3.4.5.1 ResidentService

|  |
| --- |
| package com.example.oracle.service;  import com.example.oracle.entity.Resident; import com.example.oracle.repository.ResidentRepository; import org.slf4j.Logger; import org.slf4j.LoggerFactory; import org.springframework.beans.factory.annotation.Autowired; import org.springframework.stereotype.Service;  import java.util.List;  @Service public class ResidentService {   private static final Logger logger = LoggerFactory.getLogger(ResidentService.class);   @Autowired  private ResidentRepository residentRepository;   public List<Resident> getAllResidents() {  List<Resident> residents = residentRepository.findAll();  residents.forEach(resident -> logger.info("Resident: {}", resident)); // Log từng bản ghi  return residents;  }  public Resident createResident(Resident resident) {  // Sử dụng phương thức save để tạo mới hoặc cập nhật đối tượng Resident  return residentRepository.save(resident);  }   public Resident updateResident(Long residentid, Resident updatedResident) {  // Kiểm tra xem Resident có tồn tại không  Resident resident = residentRepository.findById(residentid)  .orElseThrow(() -> new IllegalArgumentException("Resident not found with id " + residentid));   // Cập nhật các trường thông tin  resident.setAddress(updatedResident.getAddress());  resident.setDistrict(updatedResident.getDistrict());  resident.setEmail(updatedResident.getEmail());  resident.setFullName(updatedResident.getFullName());  resident.setGender(updatedResident.getGender());  resident.setIdCardNumber(updatedResident.getIdCardNumber());  resident.setPassportNumber(updatedResident.getPassportNumber());  resident.setPhoneNumber(updatedResident.getPhoneNumber());   return residentRepository.save(resident); // Lưu thông tin đã cập nhật  }   // Phương thức xóa thông tin  public void deleteResident(Long residentid) {  Resident resident = residentRepository.findById(residentid)  .orElseThrow(() -> new IllegalArgumentException("Resident not found with id " + residentid));  residentRepository.delete(resident); // Xoá resident  } } |

###### 3.4.5.2 PassportRenewalRequestService

|  |
| --- |
| package com.example.oracle.service;  import com.example.oracle.entity.PassportRenewalRequest; import com.example.oracle.repository.PassportRenewalRequestRepository; import org.springframework.beans.factory.annotation.Autowired; import org.springframework.stereotype.Service;  import java.util.List;  @Service public class PassportRenewalRequestService {   @Autowired  private PassportRenewalRequestRepository repository;   public List<PassportRenewalRequest> getAllRequests() {  return repository.findAll();  }   // Thêm mới yêu cầu  public PassportRenewalRequest createRequest(PassportRenewalRequest request) {  return repository.save(request);  }   // Cập nhật yêu cầu  public PassportRenewalRequest updateRequest(Long id, PassportRenewalRequest updatedRequest) {  PassportRenewalRequest existingRequest = repository.findById(id)  .orElseThrow(() -> new RuntimeException("Request not found with id: " + id));   // Cập nhật thông tin  existingRequest.setResidentID(updatedRequest.getResidentID());  existingRequest.setSubmissionDate(updatedRequest.getSubmissionDate());  existingRequest.setStatus(updatedRequest.getStatus());  existingRequest.setVerificationUnit(updatedRequest.getVerificationUnit());  existingRequest.setReviewComments(updatedRequest.getReviewComments());   return repository.save(existingRequest);  } } |

###### 3.4.5.3 ApprovalService

|  |
| --- |
| package com.example.oracle.service;  import com.example.oracle.dto.ApprovalDTO; import com.example.oracle.entity.Approval; import com.example.oracle.repository.ApprovalRepository; import org.springframework.beans.factory.annotation.Autowired; import org.springframework.stereotype.Service; import org.springframework.transaction.annotation.Transactional;  import java.util.List; import java.util.Optional; import java.util.stream.Collectors;  @Service public class ApprovalService {   @Autowired  private ApprovalRepository approvalRepository;   @Transactional(readOnly = true)  public ApprovalDTO getApprovalById(Long approvalId) {  Approval approval = approvalRepository.findByApprovalId(approvalId);   if (approval == null) {  return null; // Return null if no Approval found  }   // Convert entity to DTO  ApprovalDTO approvalDTO = new ApprovalDTO();  approvalDTO.setApprovalId(approval.getApprovalId());  approvalDTO.setRequestId(approval.getPassportRenewalRequest().getRequestID());  approvalDTO.setIsApproved(approval.getIsApproved());  approvalDTO.setApprovalDate(approval.getApprovalDate());  approvalDTO.setReviewerComments(approval.getReviewerComments());   return approvalDTO;  }   @Transactional(readOnly = true)  public List<ApprovalDTO> getAllApprovals() {  List<Approval> approvals = approvalRepository.findAll();   return approvals.stream()  .map(approval -> {  ApprovalDTO approvalDTO = new ApprovalDTO();  approvalDTO.setApprovalId(approval.getApprovalId());  approvalDTO.setRequestId(approval.getPassportRenewalRequest().getRequestID());  approvalDTO.setIsApproved(approval.getIsApproved());  approvalDTO.setApprovalDate(approval.getApprovalDate());  approvalDTO.setReviewerComments(approval.getReviewerComments());  return approvalDTO;  })  .collect(Collectors.toList());  }   public ApprovalDTO createApproval(ApprovalDTO approvalDTO) {  Approval approval = new Approval();  approval.setIsApproved(approvalDTO.getIsApproved());  approval.setApprovalDate(approvalDTO.getApprovalDate());  approval.setReviewerComments(approvalDTO.getReviewerComments());  // Set PassportRenewalRequest (RequestID) logic  approvalRepository.save(approval);  approvalDTO.setApprovalId(approval.getApprovalId());  return approvalDTO;  }   public ApprovalDTO updateApproval(Long approvalId, ApprovalDTO approvalDTO) {  Optional<Approval> approvalOpt = approvalRepository.findById(approvalId);  if (approvalOpt.isPresent()) {  Approval approval = approvalOpt.get();  approval.setIsApproved(approvalDTO.getIsApproved());  approval.setApprovalDate(approvalDTO.getApprovalDate());  approval.setReviewerComments(approvalDTO.getReviewerComments());  approvalRepository.save(approval);  return approvalDTO;  }  return null;  }   public void deleteApproval(Long approvalId) {  approvalRepository.deleteById(approvalId);  } } |

###### 3.4.5.4 RenewalService

|  |
| --- |
| package com.example.oracle.service;  import com.example.oracle.dto.RenewalDTO; import com.example.oracle.entity.Renewal; import com.example.oracle.repository.RenewalRepository; import org.springframework.beans.factory.annotation.Autowired; import org.springframework.stereotype.Service;  import java.util.List; import java.util.Optional; import java.util.stream.Collectors;  @Service public class RenewalService {   @Autowired  private RenewalRepository renewalRepository;   public List<RenewalDTO> getAllRenewals() {  List<Renewal> renewals = renewalRepository.findAll();  return renewals.stream()  .map(renewal -> new RenewalDTO(  renewal.getRenewalID(),  renewal.getPassportRenewalRequest().getRequestID(),  renewal.getNewExpiryDate(),  renewal.getUpdatedAt()))  .collect(Collectors.toList());  }  // public Renewal createRenewal(Renewal renewal) { // PassportRenewalRequest request = renewalRepository.findById(renewal.getPassportRenewalRequest().getRequestId()) // .orElseThrow(() -> new ResourceNotFoundException("Request not found")); // // renewal.setPassportRenewalRequest(request); // return renewalRepository.save(renewal); // }   public RenewalDTO updateRenewal(Long renewalID, RenewalDTO renewalDTO) {  Optional<Renewal> optionalRenewal = renewalRepository.findById(renewalID);   if (optionalRenewal.isPresent()) {  Renewal renewal = optionalRenewal.get();  renewal.setNewExpiryDate(renewalDTO.getNewExpiryDate());  renewal.setUpdatedAt(renewalDTO.getUpdatedAt());  Renewal updatedRenewal = renewalRepository.save(renewal);   return new RenewalDTO(  updatedRenewal.getRenewalID(),  updatedRenewal.getRenewalID(),  updatedRenewal.getNewExpiryDate(),  updatedRenewal.getUpdatedAt()  );  } else {  // Trả về null hoặc xử lý mặc định nếu không tìm thấy  return null;  }  }  public boolean deleteRenewal(Long renewalID) {  Optional<Renewal> optionalRenewal = renewalRepository.findById(renewalID);   if (optionalRenewal.isPresent()) {  renewalRepository.delete(optionalRenewal.get());  return true; // Xóa thành công  } else {  return false; // Không tìm thấy bản ghi để xóa  }  }  } |

###### 3.4.5.5 ActivityLogService

|  |
| --- |
| package com.example.oracle.service;  import com.example.oracle.dto.ActivityLogDTO; import com.example.oracle.entity.ActivityLog; import com.example.oracle.repository.ActivityLogRepository; import org.springframework.beans.factory.annotation.Autowired; import org.springframework.stereotype.Service;  import java.util.List; import java.util.stream.Collectors;  @Service public class ActivityLogService {  @Autowired  private ActivityLogRepository activityLogRepository;   public List<ActivityLogDTO> getAllActivityLogs() {  List<ActivityLog> logs = activityLogRepository.findAll();  return logs.stream().map(log -> new ActivityLogDTO(  log.getLogId(),  log.getRequestId(),  log.getStatus(),  log.getActionTimestamp(),  log.getPerformedBy()  )).collect(Collectors.toList());  } } |

#### **3.4.6 Lớp Controller.**

###### 3.4.6.1 ResidentController

|  |
| --- |
| package com.example.oracle.controller;  import com.example.oracle.entity.Resident; import com.example.oracle.service.ResidentService; import org.springframework.beans.factory.annotation.Autowired; import org.springframework.http.ResponseEntity; import org.springframework.web.bind.annotation.\*;  import java.util.List;  @RestController @RequestMapping("/api/residents") public class ResidentController {   @Autowired  private ResidentService residentService;   // API GET - Lấy danh sách Residents  @GetMapping  public List<Resident> getAllResidents() {  return residentService.getAllResidents(); // Gọi service để lấy danh sách residents  }   // API POST - Thêm mới một Resident   @PostMapping("/create")  public Resident createResident(@RequestBody Resident resident) {  return residentService.createResident(resident);  }   @PutMapping("/{residentid}")  public ResponseEntity<Resident> updateResident(@PathVariable Long residentid, @RequestBody Resident updatedResident) {  Resident resident = residentService.updateResident(residentid, updatedResident);  return ResponseEntity.ok(resident); // Trả về thông tin người cư trú đã được cập nhật  }   @DeleteMapping("/{residentid}")  public ResponseEntity<String> deleteResident(@PathVariable Long residentid) {  residentService.deleteResident(residentid);  return ResponseEntity.ok("Resident deleted successfully"); // Trả về thông báo thành công  } } |

###### 3.4.6.2 PassportRenewalController

|  |
| --- |
| package com.example.oracle.controller;  import com.example.oracle.entity.PassportRenewalRequest; import com.example.oracle.service.PassportRenewalRequestService; import org.springframework.beans.factory.annotation.Autowired; import org.springframework.http.ResponseEntity; import org.springframework.web.bind.annotation.\*;  import java.util.List;  @RestController @RequestMapping("/api/passport-renewal") public class PassportRenewalRequestController {   @Autowired  private PassportRenewalRequestService service;   @GetMapping  public ResponseEntity<List<PassportRenewalRequest>> getAllRequests() {  List<PassportRenewalRequest> requests = service.getAllRequests();  return ResponseEntity.ok(requests);  }   // API thêm mới yêu cầu (POST)  @PostMapping  public ResponseEntity<PassportRenewalRequest> createRequest(@RequestBody PassportRenewalRequest request) {  PassportRenewalRequest createdRequest = service.createRequest(request);  return ResponseEntity.ok(createdRequest);  }   // API cập nhật yêu cầu (PUT)  @PutMapping("/{id}")  public ResponseEntity<PassportRenewalRequest> updateRequest(  @PathVariable Long id, @RequestBody PassportRenewalRequest updatedRequest) {  PassportRenewalRequest request = service.updateRequest(id, updatedRequest);  return ResponseEntity.ok(request);  } } |

###### 3.4.6.3 ApprovalController

|  |
| --- |
| package com.example.oracle.controller;  import com.example.oracle.dto.ApprovalDTO; import com.example.oracle.service.ApprovalService; import org.springframework.beans.factory.annotation.Autowired; import org.springframework.http.HttpStatus; import org.springframework.http.ResponseEntity; import org.springframework.web.bind.annotation.\*;  import java.util.List;  @RestController @RequestMapping("/approvals") public class ApprovalController {   @Autowired  private ApprovalService approvalService;   // GET: /approvals/{approvalId}  @GetMapping("/{approvalId}")  public ResponseEntity<ApprovalDTO> getApproval(@PathVariable Long approvalId) {  ApprovalDTO approvalDTO = approvalService.getApprovalById(approvalId);   if (approvalDTO == null) {  return ResponseEntity.notFound().build(); // Trả về mã 404 nếu không tìm thấy phê duyệt  }   return ResponseEntity.ok(approvalDTO); // Trả về mã 200 cùng dữ liệu ApprovalDTO  }   // GET: /approvals  @GetMapping  public ResponseEntity<List<ApprovalDTO>> getAllApprovals() {  try {  List<ApprovalDTO> approvalDTOs = approvalService.getAllApprovals();  if (approvalDTOs.isEmpty()) {  return ResponseEntity.noContent().build();  }  return ResponseEntity.ok(approvalDTOs);  } catch (Exception e) {  // Log lỗi chi tiết  e.printStackTrace();  return ResponseEntity.status(HttpStatus.INTERNAL\_SERVER\_ERROR).body(null);  }  }   @PostMapping  public ResponseEntity<ApprovalDTO> createApproval(@RequestBody ApprovalDTO approvalDTO) {  ApprovalDTO createdApproval = approvalService.createApproval(approvalDTO);  return ResponseEntity.ok(createdApproval);  }   @PutMapping("/{id}")  public ResponseEntity<ApprovalDTO> updateApproval(@PathVariable Long id, @RequestBody ApprovalDTO approvalDTO) {  ApprovalDTO updatedApproval = approvalService.updateApproval(id, approvalDTO);  return updatedApproval != null ? ResponseEntity.ok(updatedApproval) : ResponseEntity.notFound().build();  }   @DeleteMapping("/{id}")  public ResponseEntity<Void> deleteApproval(@PathVariable Long id) {  approvalService.deleteApproval(id);  return ResponseEntity.noContent().build();  } } |

###### 3.4.6.4 RenewalController

|  |
| --- |
| package com.example.oracle.controller;  import com.example.oracle.dto.RenewalDTO; import com.example.oracle.service.RenewalService; import org.springframework.beans.factory.annotation.Autowired; import org.springframework.http.HttpStatus; import org.springframework.http.ResponseEntity; import org.springframework.web.bind.annotation.\*;  import java.util.List;  @RestController @RequestMapping("/api/renewals") public class RenewalController {   @Autowired  private RenewalService renewalService;   @GetMapping  public ResponseEntity<List<RenewalDTO>> getAllRenewals() {  List<RenewalDTO> renewals = renewalService.getAllRenewals();  return ResponseEntity.ok(renewals); // Trả về ResponseEntity với mã trạng thái 200 OK  }  // @PostMapping // public ResponseEntity<RenewalDTO> createRenewal(@RequestBody RenewalDTO renewalDTO) { // RenewalDTO createdRenewal = renewalService.createRenewal(renewalDTO); // return new ResponseEntity<>(createdRenewal, HttpStatus.CREATED); // }  // @PutMapping("/{renewalID}") // public ResponseEntity<RenewalDTO> updateRenewal(@PathVariable Long renewalID, @RequestBody RenewalDTO renewalDTO) { // RenewalDTO updatedRenewal = renewalService.updateRenewal(renewalID, renewalDTO); // return new ResponseEntity<>(updatedRenewal, HttpStatus.OK); // }    @DeleteMapping("/{renewalID}")  public ResponseEntity<String> deleteRenewal(@PathVariable Long renewalID) {  renewalService.deleteRenewal(renewalID);  return new ResponseEntity<>("Renewal record deleted successfully", HttpStatus.OK);  } } |

###### 3.4.6.5 ActivityLogController

|  |
| --- |
| package com.example.oracle.controller;  import com.example.oracle.dto.ActivityLogDTO; import com.example.oracle.service.ActivityLogService; import org.springframework.beans.factory.annotation.Autowired; import org.springframework.web.bind.annotation.GetMapping; import org.springframework.web.bind.annotation.RequestMapping; import org.springframework.web.bind.annotation.RestController;  import java.util.List;  @RestController @RequestMapping("/api/activity-logs") public class ActivityLogController {  @Autowired  private ActivityLogService activityLogService;   @GetMapping  public List<ActivityLogDTO> getAllActivityLogs() {  return activityLogService.getAllActivityLogs();  } } |

#### **3.4.7 Xây dựng API.**

|  |
| --- |
| @RequestMapping("/api/residents") public class ResidentController {   @Autowired  private ResidentService residentService;   // API GET - Lấy danh sách Residents  @GetMapping  public List<Resident> getAllResidents() {  return residentService.getAllResidents(); // Gọi service để lấy danh sách residents  }   // API POST - Thêm mới một Resident   @PostMapping("/create")  public Resident createResident(@RequestBody Resident resident) {  return residentService.createResident(resident);  }   @PutMapping("/{residentid}")  public ResponseEntity<Resident> updateResident(@PathVariable Long residentid, @RequestBody Resident updatedResident) {  Resident resident = residentService.updateResident(residentid, updatedResident);  return ResponseEntity.ok(resident); // Trả về thông tin người cư trú đã được cập nhật  }   @DeleteMapping("/{residentid}")  public ResponseEntity<String> deleteResident(@PathVariable Long residentid) {  residentService.deleteResident(residentid);  return ResponseEntity.ok("Resident deleted successfully"); // Trả về thông báo thành công  } |

|  |
| --- |
| @RequestMapping("/api/renewals") public class RenewalController {   @Autowired  private RenewalService renewalService;   @GetMapping  public ResponseEntity<List<RenewalDTO>> getAllRenewals() {  List<RenewalDTO> renewals = renewalService.getAllRenewals();  return ResponseEntity.ok(renewals); // Trả về ResponseEntity với mã trạng thái 200 OK  }  // @PostMapping // public ResponseEntity<RenewalDTO> createRenewal(@RequestBody RenewalDTO renewalDTO) { // RenewalDTO createdRenewal = renewalService.createRenewal(renewalDTO); // return new ResponseEntity<>(createdRenewal, HttpStatus.CREATED); // }  // @PutMapping("/{renewalID}") // public ResponseEntity<RenewalDTO> updateRenewal(@PathVariable Long renewalID, @RequestBody RenewalDTO renewalDTO) { // RenewalDTO updatedRenewal = renewalService.updateRenewal(renewalID, renewalDTO); // return new ResponseEntity<>(updatedRenewal, HttpStatus.OK); // }    @DeleteMapping("/{renewalID}")  public ResponseEntity<String> deleteRenewal(@PathVariable Long renewalID) {  renewalService.deleteRenewal(renewalID);  return new ResponseEntity<>("Renewal record deleted successfully", HttpStatus.OK);  } |

|  |
| --- |
| @RequestMapping("/api/renewals") public class RenewalController {   @Autowired  private RenewalService renewalService;   @GetMapping  public ResponseEntity<List<RenewalDTO>> getAllRenewals() {  List<RenewalDTO> renewals = renewalService.getAllRenewals();  return ResponseEntity.ok(renewals); // Trả về ResponseEntity với mã trạng thái 200 OK  }  // @PostMapping // public ResponseEntity<RenewalDTO> createRenewal(@RequestBody RenewalDTO renewalDTO) { // RenewalDTO createdRenewal = renewalService.createRenewal(renewalDTO); // return new ResponseEntity<>(createdRenewal, HttpStatus.CREATED); // }  // @PutMapping("/{renewalID}") // public ResponseEntity<RenewalDTO> updateRenewal(@PathVariable Long renewalID, @RequestBody RenewalDTO renewalDTO) { // RenewalDTO updatedRenewal = renewalService.updateRenewal(renewalID, renewalDTO); // return new ResponseEntity<>(updatedRenewal, HttpStatus.OK); // }    @DeleteMapping("/{renewalID}")  public ResponseEntity<String> deleteRenewal(@PathVariable Long renewalID) {  renewalService.deleteRenewal(renewalID);  return new ResponseEntity<>("Renewal record deleted successfully", HttpStatus.OK);  } |

|  |
| --- |
| @RestController @RequestMapping("/approvals") public class ApprovalController {   @Autowired  private ApprovalService approvalService;   // GET: /approvals/{approvalId}  @GetMapping("/{approvalId}")  public ResponseEntity<ApprovalDTO> getApproval(@PathVariable Long approvalId) {  ApprovalDTO approvalDTO = approvalService.getApprovalById(approvalId);   if (approvalDTO == null) {  return ResponseEntity.notFound().build(); // Trả về mã 404 nếu không tìm thấy phê duyệt  }   return ResponseEntity.ok(approvalDTO); // Trả về mã 200 cùng dữ liệu ApprovalDTO  }   // GET: /approvals  @GetMapping  public ResponseEntity<List<ApprovalDTO>> getAllApprovals() {  try {  List<ApprovalDTO> approvalDTOs = approvalService.getAllApprovals();  if (approvalDTOs.isEmpty()) {  return ResponseEntity.noContent().build();  }  return ResponseEntity.ok(approvalDTOs);  } catch (Exception e) {  // Log lỗi chi tiết  e.printStackTrace();  return ResponseEntity.status(HttpStatus.INTERNAL\_SERVER\_ERROR).body(null);  }  }   @PostMapping  public ResponseEntity<ApprovalDTO> createApproval(@RequestBody ApprovalDTO approvalDTO) {  ApprovalDTO createdApproval = approvalService.createApproval(approvalDTO);  return ResponseEntity.ok(createdApproval);  }   @PutMapping("/{id}")  public ResponseEntity<ApprovalDTO> updateApproval(@PathVariable Long id, @RequestBody ApprovalDTO approvalDTO) {  ApprovalDTO updatedApproval = approvalService.updateApproval(id, approvalDTO);  return updatedApproval != null ? ResponseEntity.ok(updatedApproval) : ResponseEntity.notFound().build();  }   @DeleteMapping("/{id}")  public ResponseEntity<Void> deleteApproval(@PathVariable Long id) {  approvalService.deleteApproval(id);  return ResponseEntity.noContent().build();  } |

|  |
| --- |
| @RestController @RequestMapping("/api/activity-logs") public class ActivityLogController {  @Autowired  private ActivityLogService activityLogService;   @GetMapping  public List<ActivityLogDTO> getAllActivityLogs() {  return activityLogService.getAllActivityLogs();  } } |

|  |
| --- |
| // API chỉnh sửa bài viết @PutMapping("/{id}") public ResponseEntity<Post> updatePost(@PathVariable String id, @RequestBody Post updatedPost) {  Post post = postService.updatePost(id, updatedPost);  return new ResponseEntity<>(post, HttpStatus.OK); } |

#### **3.4.8 Kết nối cơ sở dữ liệu.**

Kết nối cơ sở dữ liệu:

|  |
| --- |
| spring.application.name=oracle  spring.datasource.url=jdbc:oracle:thin:@//26.52.224.14:1521/orcl spring.datasource.username=system spring.datasource.password=123 spring.jpa.hibernate.ddl-auto=none spring.jpa.show-sql=true server.port=8082 |

A screenshot of a computer

Description automatically generated

#### **3.4.9 Webconfig.**

|  |
| --- |
| package com.example.oracle.config;  import org.springframework.context.annotation.Bean; import org.springframework.context.annotation.Configuration; import org.springframework.web.servlet.config.annotation.CorsRegistry; import org.springframework.web.servlet.config.annotation.WebMvcConfigurer;  @Configuration public class WebConfig {   @Bean  public WebMvcConfigurer corsConfigurer() {  return new WebMvcConfigurer() {  @Override  public void addCorsMappings(CorsRegistry registry) {  registry.addMapping("/\*\*")  .allowedOrigins("http://localhost:3000") // Địa chỉ của ReactJS  .allowedMethods("GET", "POST", "PUT", "DELETE", "OPTIONS");  }  };  } } |

#### **3.4.10 Kế hoạch bảo trì và nâng cấp.**

Để đảm bảo hệ thống backend của ứng dụng website cho quy trình gia hạn hộ chiếu vận hành ổn định và đáp ứng nhu cầu phát triển, kế hoạch bảo trì và nâng cấp sẽ tập trung vào các hoạt động chính như kiểm tra và sửa lỗi định kỳ, tối ưu hóa hiệu suất hệ thống, nâng cấp công nghệ, và đảm bảo an toàn bảo mật.

Định kỳ, hệ thống sẽ được theo dõi thông qua việc kiểm tra và phân tích log hệ thống để phát hiện và sửa lỗi kịp thời. Các truy vấn cơ sở dữ liệu sẽ được tối ưu để cải thiện hiệu suất, đồng thời áp dụng các giải pháp caching như Redis hoặc Memcached để giảm tải cho cơ sở dữ liệu và tăng tốc độ phản hồi. Hệ thống sẽ được thiết kế với khả năng cân bằng tải, giúp phân phối đều các yêu cầu và giảm thiểu sự cố về hiệu suất.

Về mặt nâng cấp, hệ thống sẽ được thường xuyên cập nhật các phiên bản mới của các framework và thư viện như ReactJS, Spring Boot và OracleDB để đảm bảo tính ổn định và bảo mật. Cơ sở dữ liệu sẽ được tối ưu hóa định kỳ và tài nguyên hệ thống sẽ được mở rộng khi cần thiết, dựa trên nhu cầu và sự phát triển của người dùng. Các công cụ giám sát hiệu suất như Prometheus và Grafana sẽ được triển khai để theo dõi trạng thái của hệ thống, kết hợp với hệ thống cảnh báo tự động giúp kịp thời phát hiện và xử lý sự cố.

Kế hoạch bảo trì cũng bao gồm việc sao lưu dữ liệu định kỳ và xây dựng phương án dự phòng khi xảy ra sự cố ngoài ý muốn, đảm bảo không mất mát dữ liệu quan trọng. Tất cả các quy trình bảo trì, nâng cấp và khắc phục sự cố sẽ được tài liệu hóa chi tiết để đảm bảo tính minh bạch và dễ dàng cho việc quản lý. Ngoài ra, phản hồi từ người dùng sẽ được thu thập và phân tích để nâng cấp tính năng, đồng thời các tính năng mới sẽ được kiểm thử kỹ càng trước khi triển khai lên hệ thống.

Đội ngũ phát triển và bảo trì sẽ được đào tạo định kỳ để nắm vững các công nghệ và phương pháp mới, đảm bảo hệ thống hoạt động ổn định, hiệu quả, và có thể đáp ứng kịp thời với những thay đổi trong tương lai.

## **Chương 4: KẾT LUẬN.**

Trong khuôn khổ đề tài "Xây dựng ứng dụng website cho quy trình gia hạn hộ chiếu", hệ thống đã được thiết kế và triển khai với mục tiêu tạo ra một nền tảng trực tuyến hiệu quả, dễ sử dụng và bảo mật cao, giúp công dân thực hiện quy trình gia hạn hộ chiếu một cách thuận tiện và nhanh chóng. Hệ thống được xây dựng với các công nghệ hiện đại như OracleDB, ReactJS và Spring Boot, đảm bảo khả năng mở rộng, tính ổn định và bảo mật trong suốt quá trình vận hành.

Quá trình phát triển ứng dụng đã chú trọng đến việc tối ưu hóa hiệu suất và bảo mật. Việc áp dụng các giải pháp như caching, cân bằng tải và giám sát hệ thống thông qua các công cụ như Prometheus và Grafana giúp hệ thống luôn duy trì hiệu suất cao và phản hồi nhanh chóng trước các yêu cầu của người dùng. Cùng với đó, quy trình bảo trì định kỳ và nâng cấp hệ thống sẽ đảm bảo ứng dụng luôn được cập nhật với các công nghệ mới nhất và sẵn sàng mở rộng khi có nhu cầu.

Hệ thống bảo mật của ứng dụng cũng được đặt lên hàng đầu, với các biện pháp như phân quyền truy cập chi tiết, mã hóa thông tin người dùng và dữ liệu quan trọng, cùng với việc sao lưu và xây dựng phương án dự phòng khi gặp sự cố. Điều này giúp bảo vệ dữ liệu của người dùng, đảm bảo tính toàn vẹn và an toàn cho hệ thống trong suốt quá trình hoạt động.

Trong tương lai, hệ thống có thể được mở rộng thêm các tính năng mới, chẳng hạn như hỗ trợ nhiều loại giấy tờ hoặc tích hợp với các hệ thống quản lý khác để phục vụ người dùng một cách tốt nhất. Đội ngũ phát triển cũng sẽ tiếp tục nghiên cứu và áp dụng các công nghệ mới, nhằm không ngừng cải thiện trải nghiệm người dùng và nâng cao hiệu quả công việc.

Tóm lại, ứng dụng website cho quy trình gia hạn hộ chiếu đã đạt được các mục tiêu đề ra, đảm bảo hiệu quả và tính bền vững trong vận hành. Đây là một giải pháp hữu ích cho công dân trong việc thực hiện thủ tục hành chính một cách thuận tiện và nhanh chóng, đồng thời góp phần vào việc cải thiện các dịch vụ công trực tuyến trong tương lai

# **TÀI LIỆU THAM KHẢO.**

[1] FPT Shop - <https://fptshop.com.vn/tin-tuc/danh-gia/reactjs-171664>

[2] FPT Shop - <https://fptshop.com.vn/tin-tuc/danh-gia/angular-la-gi-184969>

[3] FPT Shop - <https://fptshop.com.vn/tin-tuc/danh-gia/node-js-la-gi-169392>

[4] FPT Shop - <https://fptshop.com.vn/tin-tuc/danh-gia/django-la-gi-175507>

[5] FPT Shop - <https://fptshop.com.vn/tin-tuc/danh-gia/spring-boot-la-gi-172291>

[6] FPT Shop - <https://fptshop.com.vn/tin-tuc/danh-gia/tim-hieu-mysql-la-gi-172297>

[7] FPT Shop - <https://fptshop.com.vn/tin-tuc/danh-gia/mongodb-la-gi-167784>

Chat GPT

**Hết**