TRƯỜNG ĐẠI HỌC BÁCH KHOA THÀNH PHỐ HỒ CHÍ MINH
KHOA KHOA HỌC & KỸ THUẬT MÁY TÍNH

Đồ Án Công Nghệ Phần Mềm

Báo Cáo Đồ Án

API GRADE

GVHD: Thầy Lê Đình Thuận
HK241 1.1.1 Backend Golang

Nhóm : Code không bao giờ khó

Thành viên: Lý Vĩnh Thái - 2213104

Nguyễn Trung Nhân - 2212372

Lê Tuấn Kiệt - 2211756

Trương Quang Thịnh - 2213313

Phùng Xương Cận - 2210348

Trương Tấn Sang - 2212918

Trinh Khai Toan - 2115036

Tp Hồ Chí Minh, tháng 12/2024

MỤC LỤC

1 Task Managerment **3**
1.1 Task Allocation . . . 3
1.2 Milestones and Deadlines . . . 3
1.3 Reporting and Updates . . . 4

2 Code Organization **6**
2.1 Directory Structure . . . 6
2.2 Code Documentation . . . 6

3 Source Code **8**
3.1 Code Architecture . . . 8
3.2 Libraries and Dependencies . . . 8
3.3 Code Flow . . . 9
3.4 Version Control . . . 9

4 System Design Document (SDD) **10**
4.1 Overview of the System . . . 10
4.2 System Architecture . . . 10
4.3 Class DiagramDiagram . . . 12
4.4 Sequence DiagramDiagram . . . 13
4.5 Use Case Diagram . . . 16
4.6 Deployment DiagramDiagram . . . 17
4.7 Mapping . . . 19

5 Documents **21**
5.1 System Features and Requirements . . . 21
5.1.1 MongoDB . . . 21
5.1.2 Google Auth 2 . . . 21
5.1.3 SendMail . . . 21
5.1.4 Gin Framework . . . 21
5.1.5 JWT (JSON Web Token) . . . 22
5.1.6 gotenv . . . 22
5.1.7 CORS (Cross-Origin Resource Sharing) . . . 22
5.1.8 Tổng hợp hệ thống . . . 22
5.2 User Manuals . . . 22
5.3 API Documentation . . . 22

6 Code style **23**
6.1 System Architecture . . . 23

6.2 Version Control and Workflow 23
6.3 GitHub Actions 23
6.4 Code Quality and Best Practices 24

7 Slide **25**
7.1 Presentation Overview 25

8 CI **26**
8.1 Continuous Integration (CI) 26
8.2 Giải thích Workflow 27
8.3 Lợi ích của CI 27

9 Bonus **28**

1 Task Managerment

1.1 Task Allocation

1. **API cho Web Client**
   - **Mục đích:** Hỗ trợ các tính năng dành cho người dùng bao gồm giảng viên và sinh viên.
   - **Công việc:**
     - Phát triển các endpoint cho xác thực người dùng (ví dụ: đăng nhập, đăng ký).
     - Xây dựng xử lý dữ liệu để hiển thị dashboard và báo cáo.
     - Hỗ trợ xử lý các hành động của người dùng như gửi biểu mẫu và tải lên tệp.
   - **Ưu tiên:** Đảm bảo hiệu suất cao và trải nghiệm người dùng mượt mà.
2. **API cho Web Admin**
   - **Mục đích:** Hỗ trợ các chức năng quản trị.
   - **Công việc:**
     - Tạo các endpoint cho quản lý người dùng và dữ liệu (CRUD).
     - Tạo báo cáo và phân tích dữ liệu.
     - Quản lý cài đặt hệ thống và phân quyền.
   - **Ưu tiên:** Đảm bảo bảo mật và kiểm soát truy cập chặt chẽ.
3. **API cho Tele Client**
   - **Mục đích:** Hỗ trợ các chức năng thời gian thực.
   - **Công việc:**
     - Phát triển các endpoint cho nhắn tin (ví dụ: chat, thông báo).
     - Quản lý kết nối giữa người dùng hoặc thiết bị.
     - Xử lý truyền dữ liệu lớn.
   - **Ưu tiên:** Đảm bảo độ trễ thấp và độ tin cậy cao.

1.2 Milestones and Deadlines

1. **Thiết kế API**
   - Phân tích yêu cầu và xác định các endpoint cần thiết cho **Web Client**, **Web Admin**, và **Tele Client**.
   - Hoàn thành tài liệu thiết kế API trước ngày **DD/MM/YYYY**.

2. **Phát triển API**
    - Bắt đầu phát triển API theo từng phần:
        - **Web Client:** Hoàn thành vào ngày **DD/MM/YYYY**.
        - **Web Admin:** Hoàn thành vào ngày **DD/MM/YYYY**.
        - **Tele Client:** Hoàn thành vào ngày **DD/MM/YYYY**.
    - Mỗi endpoint sẽ được phát triển, kiểm thử, và tích hợp trước thời hạn.
3. **Kiểm thử và Tối ưu hóa API**
    - Tiến hành kiểm thử toàn bộ API để đảm bảo tính chính xác và hiệu suất.
    - Xử lý các lỗi và tối ưu hóa trước ngày **DD/MM/YYYY**.
4. **Triển khai và Bàn giao API**
    - Hoàn thành triển khai API lên môi trường production trước ngày **DD/MM/YYYY**.
    - Bàn giao tài liệu và hướng dẫn sử dụng cho các bên liên quan.

1.3 Reporting and Updates

1. **Báo cáo tiến độ định kỳ**
    - Các thành viên trong nhóm phải gửi báo cáo tiến độ vào cuối mỗi tuần (ví dụ: thứ 6 hàng tuần).
    - Báo cáo phải bao gồm:
        - Các nhiệm vụ đã hoàn thành.
        - Các khó khăn hoặc vấn đề gặp phải trong quá trình phát triển.
        - Kế hoạch cho tuần tiếp theo.
2. **Cập nhật trạng thái dự án**
    - Quản lý dự án sẽ tổng hợp các báo cáo từ các thành viên và cập nhật trạng thái tổng quan của dự án.
    - Cập nhật sẽ được gửi tới toàn bộ nhóm vào đầu mỗi tuần (ví dụ: thứ 2).
3. **Các cuộc họp nhóm**
    - Tổ chức các cuộc họp nhóm định kỳ (ví dụ: hàng tuần hoặc hai tuần một lần) để:
        - Thảo luận về tiến độ và các vấn đề quan trọng.
        - Đưa ra quyết định về các thay đổi hoặc cải tiến trong dự án.
    - Cuộc họp sẽ được ghi lại và chia sẻ cho tất cả các thành viên.

4. **Công cụ báo cáo**

   - Sử dụng các công cụ như **Jira**, **Trello**, hoặc **GitHub Issues** để theo dõi tiến độ và quản lý nhiệm vụ.
   - Gửi thông báo qua email hoặc các nền tảng giao tiếp nhóm (ví dụ: **Slack** hoặc **Microsoft Teams**).

2 Code Organization

2.1 Directory Structure

Cấu trúc thư mục của dự án theo mô hình Model-View-Controller (MVC) để tách biệt các phần công việc và duy trì tính mô-đun trong mã nguồn. Dưới đây là mô tả về cấu trúc thư mục của dự án:

- `.github` - Chứa các cấu hình và workflow liên quan đến GitHub CI/CD.
- `.vscode` - Lưu trữ các thiết lập workspace của Visual Studio Code.
- `API` - Bao gồm các tệp và cấu hình liên quan đến API.
- `docs` - Các tài liệu hướng dẫn, tài liệu dự án.
- `reports` - Thư mục chứa các báo cáo và các tệp được tạo ra.
- `src` - Mã nguồn chính của ứng dụng, tuân theo kiến trúc MVC:
- `.gitignore` - Chỉ định các tệp và thư mục cần bỏ qua trong Git.
- `.env` - Các biến môi trường để cấu hình ứng dụng.
- `Dockerfile` - Cấu hình Docker cho việc thiết lập container cho dự án.
- `go.mod` và `go.sum` - Các tệp quản lý module của Go để quản lý các phụ thuộc.
- `main.go` - Điểm vào của ứng dụng Go.

2.2 Code Documentation

Việc tài liệu hóa mã nguồn là một phần quan trọng trong việc duy trì chất lượng của dự án. Code documentation giúp các lập trình viên khác (hoặc chính bạn trong tương lai) hiểu được cách thức hoạt động của ứng dụng, từ đó dễ dàng bảo trì, mở rộng hoặc sửa lỗi khi cần thiết.

Các nguyên tắc chính trong việc tài liệu hóa mã nguồn trong dự án này bao gồm:

- **Sử dụng comment rõ ràng**: Mỗi hàm, phương thức hoặc class đều cần có các bình luận mô tả mục đích của nó, tham số đầu vào, giá trị trả về và các tình huống ngoại lệ (nếu có). Bình luận cần phải ngắn gọn nhưng đầy đủ thông tin để người đọc có thể hiểu ngay mục đích và cách thức hoạt động của mã.
- **Tài liệu hóa các module**: Mỗi module (ví dụ: controller, model, middleware, helper) cần có một tệp tài liệu riêng hoặc phần mô tả trong đầu tệp để giải thích rõ ràng về chức năng và cách thức hoạt động của nó. Điều này giúp những người mới tham gia dự án dễ dàng hiểu được cấu trúc mã nguồn.
- **Chú thích code sử dụng Docstrings (Đặc biệt đối với Go)**: Đối với mỗi hàm, phương thức hoặc type trong Go, chúng ta sử dụng docstrings để mô tả chi tiết về chức năng của chúng. Ví dụ:

```
// FetchUserById fetches a user from the database by their unique ID.
```

- **Tài liệu API**: Các API cần được tài liệu hóa một cách chi tiết, bao gồm thông tin về các endpoint, phương thức HTTP, các tham số yêu cầu, cấu trúc dữ liệu đầu vào và đầu ra, cũng như các mã trạng thái HTTP có thể trả về. Điều này giúp việc sử dụng API trở nên dễ dàng và minh bạch.
- **Cập nhật tài liệu theo thay đổi mã nguồn**: Tài liệu cần được cập nhật mỗi khi mã nguồn thay đổi. Điều này có thể bao gồm việc thêm mới, sửa đổi hoặc xóa các tính năng. Đảm bảo rằng tài liệu luôn phản ánh chính xác trạng thái hiện tại của dự án.

Lợi ích của việc tài liệu hóa mã nguồn:

- Giúp dễ dàng bảo trì mã nguồn trong tương lai, đặc biệt khi có sự thay đổi về đội ngũ phát triển.
- Tăng khả năng tái sử dụng và mở rộng mã nguồn.
- Cải thiện sự hiểu biết chung về dự án, giúp các lập trình viên mới có thể nhanh chóng tiếp cận và đóng góp vào dự án.

3 Source Code

3.1 Code Architecture

- `config` - Các tệp cấu hình cho việc thiết lập môi trường, cơ sở dữ liệu, và các cài đặt khác.
- `controllers` - Chứa các controller xử lý logic nghiệp vụ của ứng dụng và tương tác với các model và view.
- `helper` - Các hàm tiện ích và tệp helper được sử dụng trong toàn bộ ứng dụng.
- `middlewares` - Các tệp middleware để xử lý yêu cầu, xác thực, kiểm tra dữ liệu,...
- `models` - Định nghĩa các mô hình dữ liệu (representing the database structure) và logic tương tác với cơ sở dữ liệu.
- `routes` - Định nghĩa các tuyến đường (routes) của ứng dụng và ánh xạ chúng đến các controller tương ứng.
- `tmp` - Thư mục chứa các tệp tạm thời, có thể dùng để lưu cache hoặc các tệp tạm thời khác.

3.2 Libraries and Dependencies

Hệ thống của chúng tôi sử dụng một số thư viện và phụ thuộc để tối ưu hóa quá trình phát triển và đảm bảo tính ổn định của ứng dụng. Các thư viện chính bao gồm:

- **Gin:** Là một framework web nhẹ cho Go, giúp xây dựng các API nhanh chóng và hiệu quả.
- **MongoDB:** Sử dụng MongoDB như một cơ sở dữ liệu NoSQL để lưu trữ dữ liệu không có cấu trúc.
- **JWT (JSON Web Token):** Thư viện JWT dùng để tạo và xác thực mã thông báo, đảm bảo an toàn cho các API.
- **Gotenv:** Thư viện giúp dễ dàng tải các biến môi trường từ các tệp '.env', đảm bảo quản lý cấu hình an toàn.
- **Cross-origin Resource Sharing (CORS):** Được sử dụng để cấu hình các chính sách chia sẻ tài nguyên giữa các nguồn gốc khác nhau trong hệ thống.
- **Google Auth 2.0:** Cung cấp phương thức xác thực qua tài khoản Google, giúp người dùng đăng nhập dễ dàng và an toàn.
- **SendMail:** Thư viện hỗ trợ gửi email từ ứng dụng, dùng để gửi thông báo, xác thực hoặc các email liên quan khác.

Mỗi thư viện và phụ thuộc này đều đóng vai trò quan trọng trong việc xây dựng ứng dụng API, đảm bảo tính hiệu quả, bảo mật và khả năng mở rộng.

3.3 Code Flow

Quy trình mã (Code Flow) của hệ thống được chia thành các bước chính để đảm bảo tính dễ hiểu và bảo trì dễ dàng. Quy trình này bao gồm:

1. **Nhận yêu cầu từ client:** API nhận các yêu cầu HTTP từ các client (có thể là web client, mobile client, hoặc các ứng dụng khác).
2. **Xử lý yêu cầu trong Controller:** Các yêu cầu này được gửi đến các controller tương ứng, nơi xử lý logic ứng dụng.
3. **Tương tác với cơ sở dữ liệu:** Controller sẽ gọi các model để tương tác với cơ sở dữ liệu, thực hiện các thao tác như tạo mới, sửa đổi hoặc truy vấn dữ liệu.
4. **Xử lý và trả về phản hồi:** Sau khi thực hiện các thao tác với cơ sở dữ liệu, kết quả sẽ được gửi lại client dưới dạng JSON. Quá trình này có thể bao gồm các bước như xác thực, phân quyền, và xử lý lỗi.
5. **Gửi thông báo qua email (nếu cần):** Một số yêu cầu có thể yêu cầu gửi email thông báo, điều này được thực hiện qua thư viện SendMail.

Quy trình mã này giúp đảm bảo rằng mỗi phần của hệ thống hoạt động một cách độc lập và có thể dễ dàng mở rộng khi cần thiết.

3.4 Version Control

Chúng tôi sử dụng **Git** và **GitHub** để quản lý mã nguồn và phối hợp giữa các thành viên trong nhóm. Các nhánh và quy trình làm việc được quản lý thông qua **GitFlow**, một mô hình phát triển git giúp tổ chức các nhánh hiệu quả. Quy trình phiên bản bao gồm:

- **main**: Chứa mã nguồn ổn định và sẵn sàng để phát hành.
- **develop**: Là nhánh chính để phát triển tính năng, sửa lỗi và chuẩn bị cho các phiên bản mới.
- **feature**: Các nhánh tính năng được tạo ra từ develop để phát triển các tính năng mới.
- **release**: Các nhánh này được sử dụng để chuẩn bị cho việc phát hành.
- **hotfix**: Các nhánh khẩn cấp được tạo ra từ master để sửa lỗi nghiêm trọng trong phiên bản ổn định.

Mỗi khi một tính năng mới được phát triển hoặc một lỗi nghiêm trọng được sửa, chúng tôi tạo một pull request để mã nguồn được kiểm tra và hợp nhất vào nhánh phát triển chính. Quy trình này giúp đảm bảo rằng tất cả các thay đổi đều được kiểm tra kỹ lưỡng trước khi được phát hành.

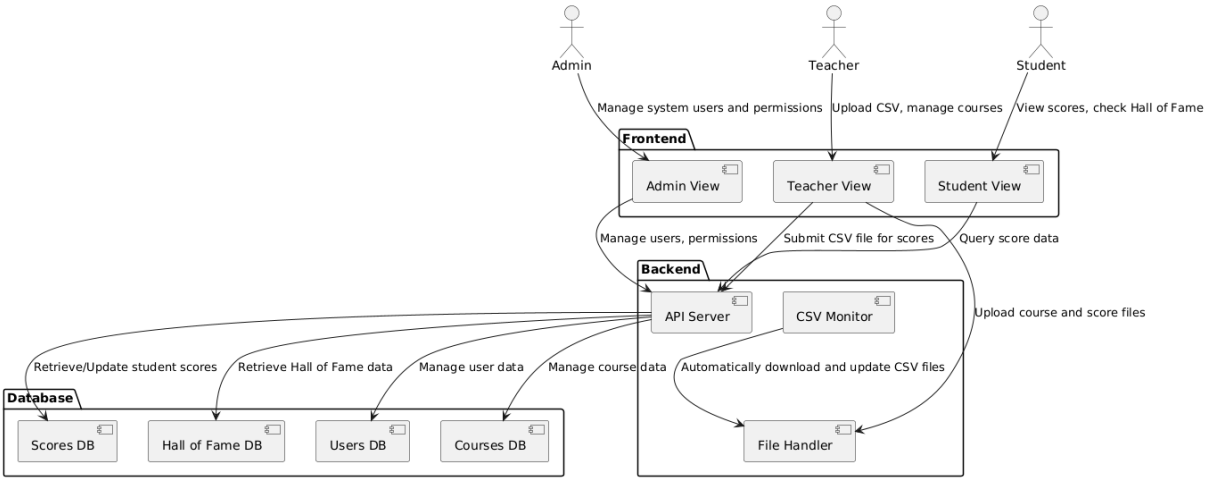
Ngoài ra, chúng tôi cũng sử dụng **GitHub Actions** để tự động hóa quy trình xây dựng, kiểm tra và triển khai mã nguồn lên môi trường sản xuất. Điều này giúp tiết kiệm thời gian và đảm bảo rằng mỗi thay đổi đều được triển khai một cách tự động và an toàn.

4 System Design Document (SDD)

4.1 Overview of the System

Sơ đồ tổng quan minh họa các tương tác giữa các thành phần khác nhau của hệ thống. Các thành phần chính bao gồm StudentView, TeacherView, AdminView, APIServer, UsersDB, CoursesDB, ScoresDB, HallOfFameDB, FileHandler, và CSVMonitor. Dưới đây là mô tả chi tiết về từng tương tác:

- **StudentView** truy vấn **APIServer** để lấy dữ liệu điểm.
- **TeacherView** gửi file CSV chứa điểm lên **APIServer**.
- **AdminView** quản lý người dùng và quyền hạn thông qua **APIServer**.
- **APIServer** quản lý dữ liệu người dùng bằng cách tương tác với **UsersDB**.
- **APIServer** quản lý dữ liệu khóa học bằng cách tương tác với **CoursesDB**.
- **APIServer** lấy và cập nhật điểm của sinh viên bằng cách tương tác với **ScoresDB**.
- **APIServer** lấy dữ liệu Hall of Fame bằng cách tương tác với **HallOfFameDB**.
- **TeacherView** tải lên các file khóa học và điểm lên **FileHandler**.
- **CSVMonitor** tự động tải xuống và cập nhật các file CSV thông qua **FileHandler**.



Hình 1: Sơ đồ tổng quan

4.2 System Architecture

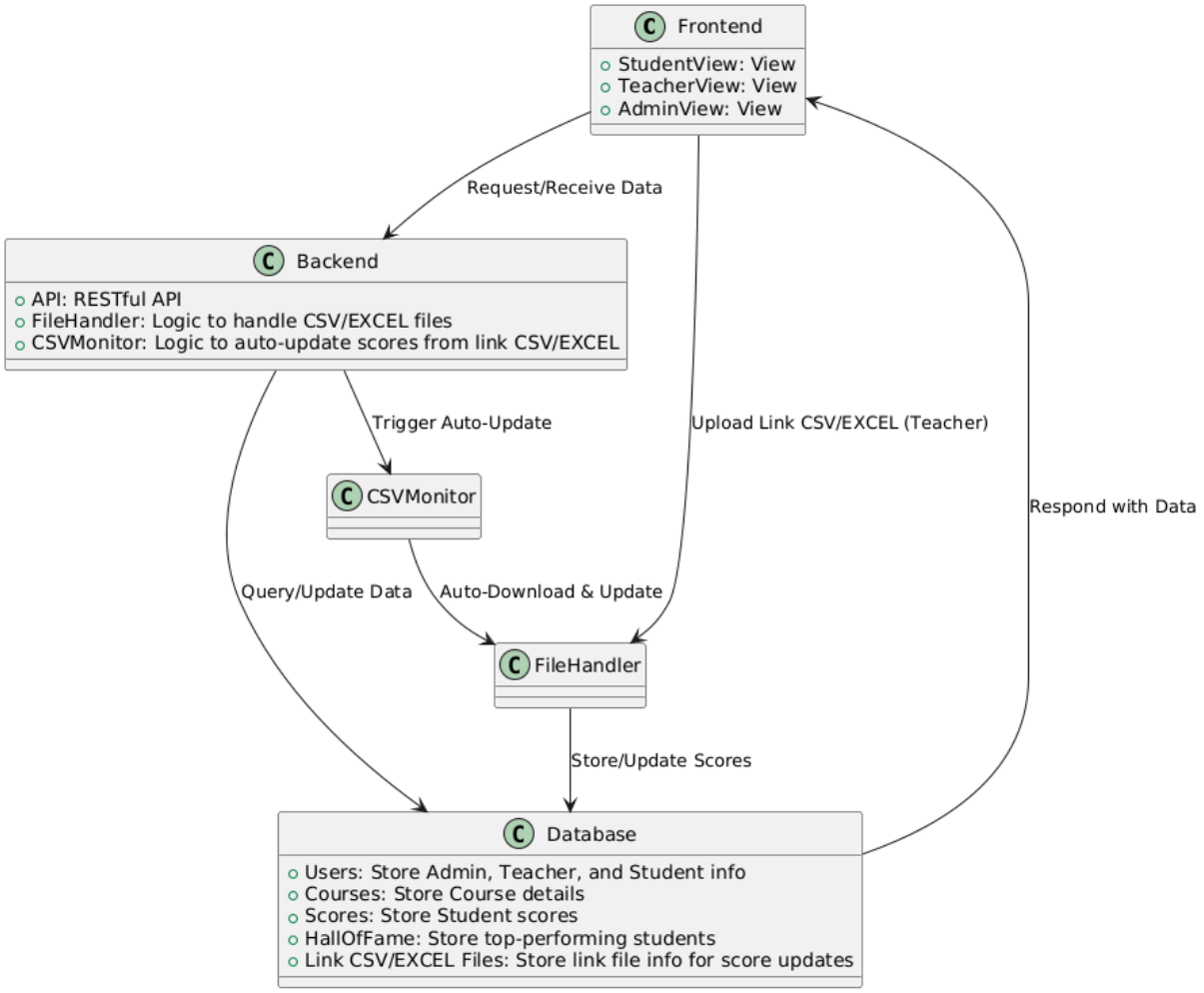
Sơ đồ kiến trúc minh họa các tương tác giữa các thành phần khác nhau của hệ thống. Các thành phần chính bao gồm Frontend, Backend, Database, FileHandler, và CSVMonitor. Dưới đây là mô tả chi tiết về từng tương tác:

- **Frontend** gửi yêu cầu và nhận dữ liệu từ **Backend**.
- **Backend** truy vấn và cập nhật dữ liệu từ **Database**.

- **Database** phản hồi với dữ liệu cho **Frontend**.
- **Frontend** tải lên các liên kết file CSV/EXCEL (giáo viên) lên **FileHandler**.
- **FileHandler** lưu trữ và cập nhật điểm số trong **Database**.
- **CSVMonitor** tự động tải xuống và cập nhật các file thông qua **FileHandler**.
- **Backend** kích hoạt việc tự động cập nhật của **CSVMonitor**.

Các thành phần lưu trữ chính bao gồm:

- **Scores**: Lưu trữ điểm số của sinh viên.
- **HallOfFame**: Lưu trữ thông tin về các sinh viên có thành tích cao.
- **Link CSV/EXCEL Files**: Lưu trữ thông tin liên kết file để cập nhật điểm số.



Hình 2: Sơ đồ kiến trúc hệ thống

4.3 Class DiagramDiagram

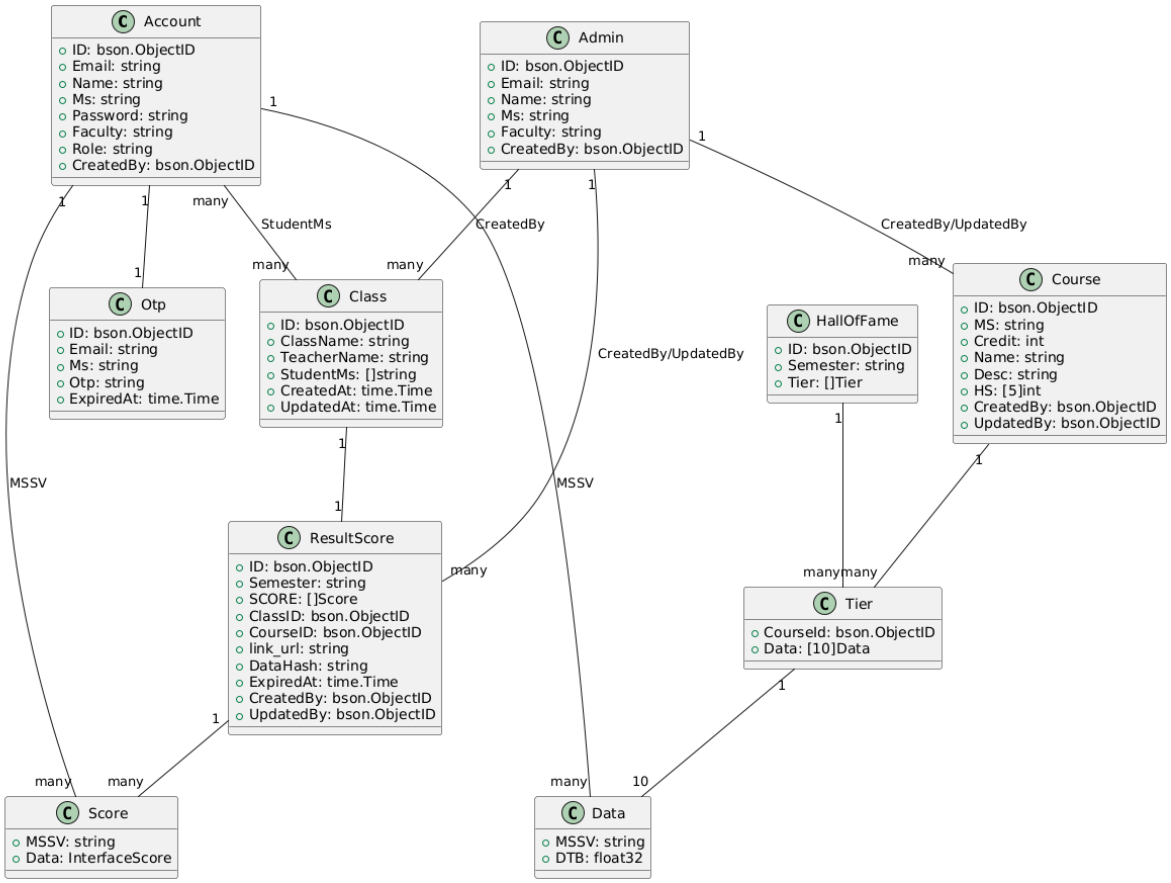
Sơ đồ lớp minh họa các mối quan hệ giữa các lớp trong hệ thống. Các lớp chính bao gồm Account, Admin, Class, Course, HallOfFame, Tier, Data, Otp, ResultScore, và Score. Dưới đây là mô tả chi tiết về từng lớp và mối quan hệ giữa chúng:

- **Account**: Lớp này đại diện cho tài khoản người dùng với các thuộc tính như ID, Email, Name, Ms, Password, Faculty, Role, và CreatedBy.
- **Admin**: Lớp này đại diện cho tài khoản quản trị viên với các thuộc tính như ID, Email, Name, Ms, Faculty, và CreatedBy.
- **Class**: Lớp này đại diện cho lớp học với các thuộc tính như ID, ClassName, TeacherName, StudentMs, CreatedAt, và UpdatedAt.
- **Course**: Lớp này đại diện cho khóa học với các thuộc tính như ID, MS, Credit, Name, Desc, HS, CreatedBy, và UpdatedBy.
- **HallOfFame**: Lớp này đại diện cho bảng vinh danh với các thuộc tính như ID, Semester, và Tier.
- **Tier**: Lớp này đại diện cho cấp bậc trong bảng vinh danh với các thuộc tính như CourseId và Data.
- **Data**: Lớp này đại diện cho dữ liệu sinh viên với các thuộc tính như MSSV và DTB.
- **Otp**: Lớp này đại diện cho mã OTP với các thuộc tính như ID, Email, Ms, Otp, và ExpiredAt.

- **ResultScore**: Lớp này đại diện cho kết quả điểm với các thuộc tính như ID, Semester, SCORE, ClassID, CourseID, $\text{link}_url, DataHash, ExpiredAt, CreatedBy, vUpdatedBy$.**Score** : $LpnyØidinchoØimscasinhvinviccthuctnh$

  Các mối quan hệ giữa các lớp bao gồm:

  - **ResultScore** có mối quan hệ 1-nhiều với **Score**.
  - **Admin** có mối quan hệ 1-nhiều với **Class**, **Course**, và **ResultScore** thông qua thuộc tính CreatedBy và UpdatedBy.
  - **Class** có mối quan hệ 1-1 với **ResultScore**.
  - **Course** có mối quan hệ 1-nhiều với **Tier**.
  - **HallOfFame** có mối quan hệ 1-nhiều với **Tier**.
  - **Tier** có mối quan hệ 1-10 với **Data**.
  - **Account** có mối quan hệ 1-1 với **Otp**.
  - **Account** có mối quan hệ nhiều-nhiều với **Class** thông qua thuộc tính StudentMs.
  - **Account** có mối quan hệ 1-nhiều với **Score** và **Data** thông qua thuộc tính MSSV.



Hình 3: Sơ đồ lớp

4.4 Sequence DiagramDiagram

Sơ đồ trình tự minh họa các quy trình và tương tác giữa các thành phần khác nhau của hệ thống. Các thành phần chính bao gồm User, Admin, Teacher, Account, Otp, Course, Class, Score, ResultScore, Telegram, Google, và Link URL. Dưới đây là mô tả chi tiết về từng quy trình:

- **Quy trình 1: Đăng nhập của người dùng qua Google (Web)**
  - User đăng nhập qua Google.
  - Google xác thực người dùng với Account.
  - Account trả về thông tin người dùng cho Google.
  - Google thông báo đăng nhập thành công cho User.
- **Quy trình 2: Đăng nhập của người dùng qua Telegram (Telegram)**
  - User đăng nhập qua Telegram.
  - Telegram yêu cầu Account tạo OTP.
  - Account gửi OTP cho Telegram.

  - Telegram gửi OTP cho User.
  - User nhập OTP vào Telegram.
  - Telegram xác thực OTP với Account.
  - Account thông báo đăng nhập thành công cho User.

- **Quy trình 3: Sinh viên xem điểm của mình**
  - User yêu cầu ResultScore cung cấp điểm.
  - ResultScore trả về điểm hiện tại cho User.

- **Quy trình 4: Cập nhật điểm tự động**
  - ResultScore kiểm tra nếu có liên kết cập nhật điểm.
  - LinkUrl trả về liên kết cập nhật điểm.
  - ResultScore lấy điểm cập nhật từ LinkUrl (mỗi 10 phút).
  - LinkUrl trả về điểm cập nhật.
  - ResultScore trả về điểm cập nhật cho User.

- **Quy trình 5: Thêm điểm cho học sinh (Admin)**
  - Admin chọn khóa học.
  - Course lấy thông tin lớp học.
  - Admin thêm điểm cho học sinh.
  - Score lưu điểm cho học sinh.
  - ResultScore cập nhật kết quả lớp học.
  - ResultScore hiển thị kết quả điểm cho Admin.

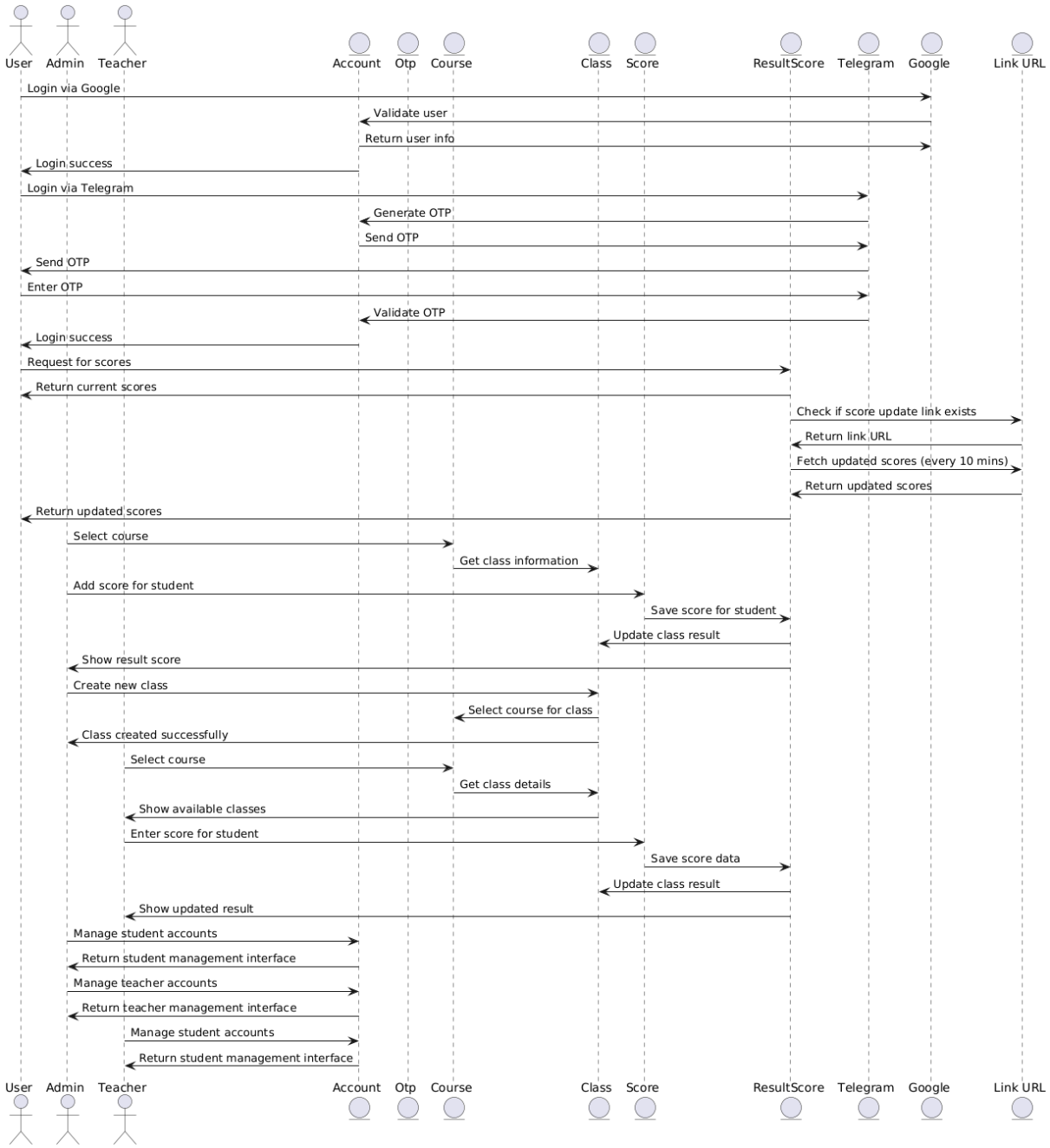
- **Quy trình 6: Tạo lớp học mới (Admin)**
  - Admin tạo lớp học mới.
  - Class chọn khóa học cho lớp học.
  - Class thông báo tạo lớp học thành công cho Admin.

- **Quy trình 7: Giảng viên chọn khóa học và lớp học**
  - Teacher chọn khóa học.
  - Course lấy thông tin lớp học.
  - Class hiển thị các lớp học có sẵn cho Teacher.

- **Quy trình 8: Giảng viên nhập điểm cho sinh viên**

  - Teacher nhập điểm cho sinh viên.
  - Score lưu dữ liệu điểm.
  - ResultScore cập nhật kết quả lớp học.
  - ResultScore hiển thị kết quả cập nhật cho Teacher.

- **Quy trình 9: Admin quản lý sinh viên**
  - Admin quản lý tài khoản sinh viên.
  - Account trả về giao diện quản lý sinh viên cho Admin.
- **Quy trình 10: Admin quản lý giáo viên**
  - Admin quản lý tài khoản giáo viên.
  - Account trả về giao diện quản lý giáo viên cho Admin.
- **Quy trình 11: Giáo viên quản lý sinh viên**
  - Teacher quản lý tài khoản sinh viên.
  - Account trả về giao diện quản lý sinh viên cho Teacher.



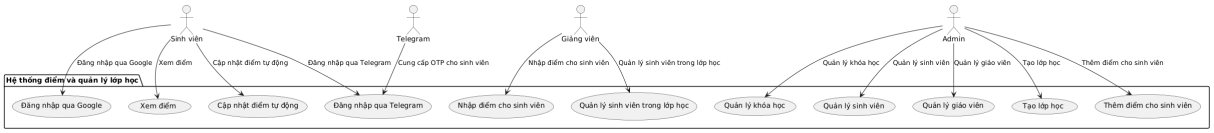
Hình 4: Sơ đồ trình tự

4.5 Use Case Diagram

Sơ đồ Use Case minh họa các trường hợp sử dụng chính trong hệ thống điểm và quản lý lớp học. Các tác nhân chính bao gồm Sinh viên, Giảng viên, Admin, và Telegram. Dưới đây là mô tả chi tiết về từng trường hợp sử dụng:

- **Đăng nhập qua Google**: Sinh viên, Giảng viên, và Admin có thể đăng nhập vào hệ thống thông qua Google.

- **Đăng nhập qua Telegram**: Sinh viên, Giảng viên, và Admin có thể đăng nhập vào hệ thống thông qua Telegram.
- **Xem điểm**: Sinh viên có thể xem điểm của mình.
- **Cập nhật điểm tự động**: Hệ thống tự động cập nhật điểm từ các nguồn dữ liệu.
- **Nhập điểm cho sinh viên**: Giảng viên có thể nhập điểm cho sinh viên.
- **Tạo lớp học**: Admin có thể tạo lớp học mới.
- **Thêm điểm cho sinh viên**: Admin có thể thêm điểm cho sinh viên.
- **Quản lý khóa học**: Admin có thể quản lý các khóa học.
- **Quản lý sinh viên**: Admin có thể quản lý thông tin sinh viên.
- **Quản lý giáo viên**: Admin có thể quản lý thông tin giảng viên.
- **Quản lý sinh viên trong lớp học**: Giảng viên có thể quản lý sinh viên trong lớp học của mình.



Hình 5: Sơ đồ Use Case

4.6 Deployment DiagramDiagram

Sơ đồ triển khai minh họa các thành phần chính của hệ thống và cách chúng tương tác với nhau. Các thành phần chính bao gồm Web Server, Database Server, Telegram Server, Client Device (Web), và Client Device (Telegram). Dưới đây là mô tả chi tiết về từng thành phần và mối quan hệ giữa chúng:

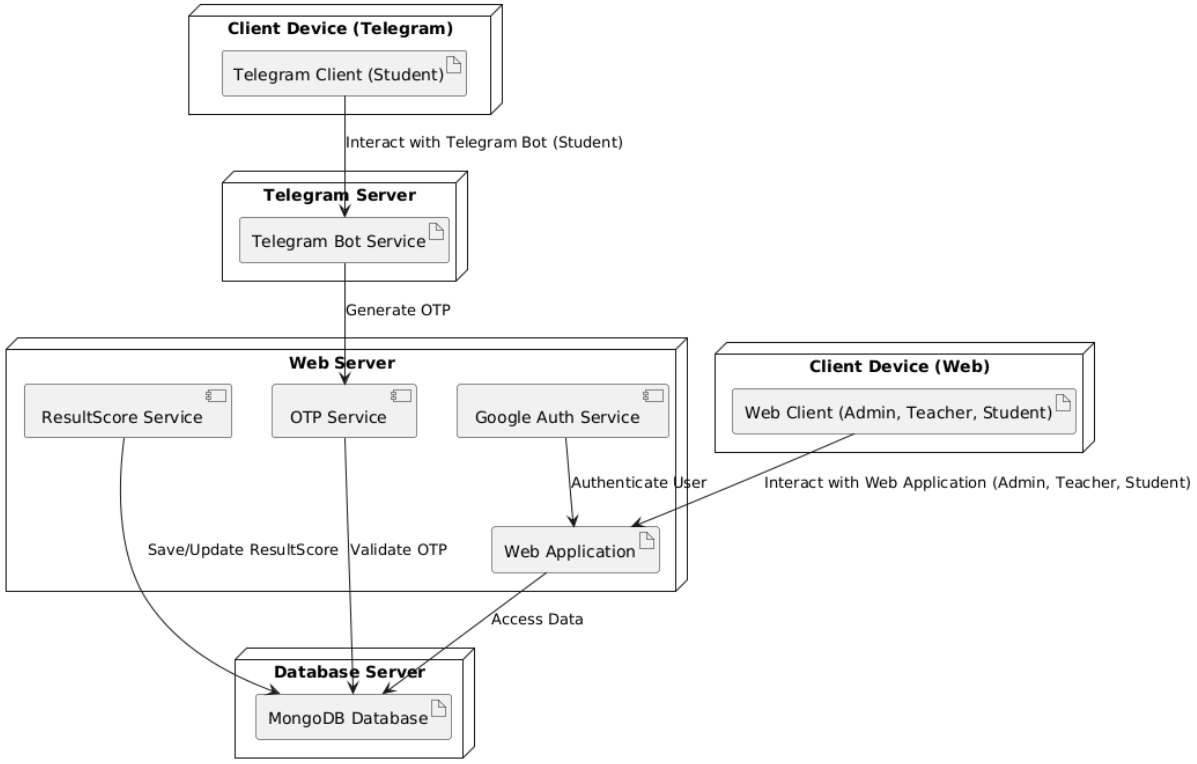
- **Web Server**:
  - **Web Application (WA)**: Ứng dụng web chính.
  - **Google Auth Service (GAS)**: Dịch vụ xác thực người dùng qua Google.
  - **ResultScore Service (RSS)**: Dịch vụ lưu trữ và cập nhật điểm số.
  - **OTP Service (OTP)**: Dịch vụ tạo và xác thực mã OTP.
- **Database Server**:
  - **MongoDB Database (DB)**: Cơ sở dữ liệu MongoDB.
- **Telegram Server**:
  - **Telegram Bot Service (TBS)**: Dịch vụ bot Telegram.
- **Client Device (Web)**:

  - **Web Client (Admin, Teacher, Student) (WC)**: Ứng dụng web dành cho Admin, Giảng viên, và Sinh viên.

- **Client Device (Telegram)**:
  - **Telegram Client (Student) (TC)**: Ứng dụng Telegram dành cho Sinh viên.

Các mối quan hệ giữa các thành phần bao gồm:

- **Web Server và Database Server**:
  - WA truy cập DB để lấy và lưu trữ dữ liệu.
  - GAS xác thực người dùng và tương tác với WA.
  - RSS lưu trữ và cập nhật điểm số trong DB.
  - OTP xác thực mã OTP trong DB.
- **Telegram Service**:
  - TBS tạo mã OTP thông qua dịch vụ OTP.
- **Client Devices**:
  - WC tương tác với WA để thực hiện các chức năng của Admin, Giảng viên, và Sinh viên.
  - TC tương tác với TBS để thực hiện các chức năng của Sinh viên qua Telegram.

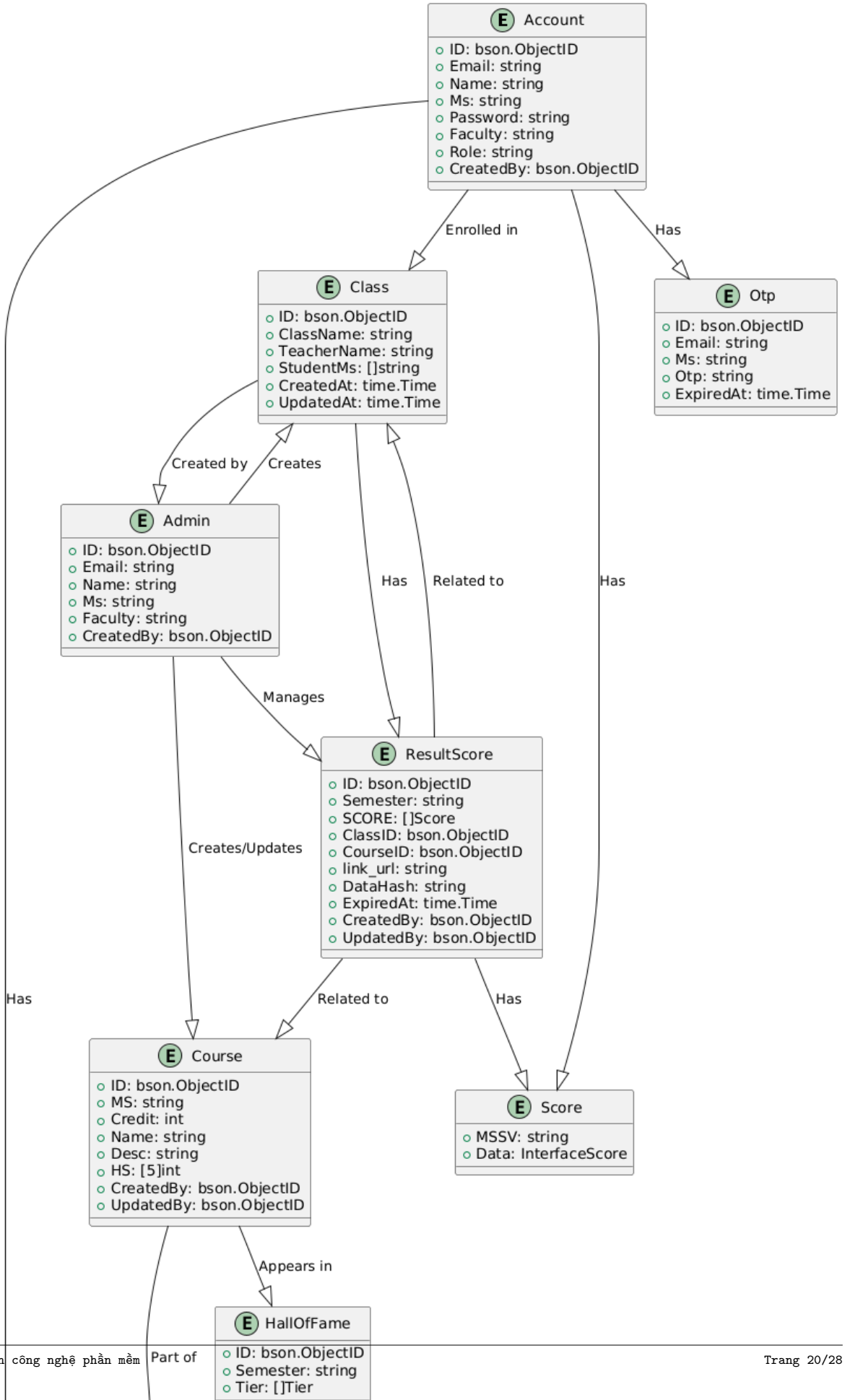


Hình 6: Sơ đồ triển khai

4.7 Mapping

Sơ đồ quan hệ minh họa các mối quan hệ giữa các thành phần chính trong hệ thống. Dưới đây là mô tả chi tiết về từng mối quan hệ:

- **Admin** quản lý **ResultScore**.
- **Class** có **ResultScore**.
- **Class** được tạo bởi **Admin**.
- **Course** là một phần của **Tier**.
- **Course** xuất hiện trong **HallOfFame (hof)**.
- **HallOfFame (hof)** chứa **Tier**.
- **Tier** có **Data**.
- **ResultScore** có **Score**.
- **ResultScore** liên quan đến **Class**.
- **ResultScore** liên quan đến **Course**.


Account
ID: bson.ObjectID
Email: string
Name: string
Ms: string
Password: string
Faculty: string
Role: string
CreatedBy: bson.ObjectID
Enrolled in
Has
Class
ID: bson.ObjectID
ClassName: string
TeacherName: string
StudentMs: []string
CreatedAt: time.Time
UpdatedAt: time.Time
Otp
ID: bson.ObjectID
Email: string
Ms: string
Otp: string
ExpiredAt: time.Time
Created by
Creates
Admin
ID: bson.ObjectID
Email: string
Name: string
Ms: string
Faculty: string
CreatedBy: bson.ObjectID
Has
Related to
Has
Manages
ResultScore
ID: bson.ObjectID
Semester: string
SCORE: []Score
ClassID: bson.ObjectID
CourseID: bson.ObjectID
link_url: string
DataHash: string
ExpiredAt: time.Time
CreatedBy: bson.ObjectID
UpdatedBy: bson.ObjectID
Creates/Updates
Has
Related to
Has
Course
ID: bson.ObjectID
MS: string
Credit: int
Name: string
Desc: string
HS: [5]int
CreatedBy: bson.ObjectID
UpdatedBy: bson.ObjectID
Score
MSSV: string
Data: InterfaceScore
Appears in
HallOfFame
ID: bson.ObjectID
Semester: string
Tier: []Tier
Part of

5 Documents

5.1 System Features and Requirements

Hệ thống của chúng tôi bao gồm các tính năng chính sau đây để đảm bảo tính bảo mật, hiệu suất và khả năng mở rộng. Các công nghệ và tính năng chính được sử dụng trong dự án bao gồm MongoDB, Google Auth 2, SendMail, Gin, JWT, và gotenv và CORS..

5.1.1 MongoDB

MongoDB được sử dụng làm cơ sở dữ liệu chính trong hệ thống. Đây là cơ sở dữ liệu NoSQL, cho phép lưu trữ dữ liệu dưới dạng JSON-like documents, rất linh hoạt trong việc mở rộng và phát triển các tính năng mới. MongoDB được sử dụng để lưu trữ thông tin người dùng, dữ liệu giao dịch và các thông tin quan trọng khác của ứng dụng.

- **Chức năng:** Lưu trữ và truy vấn dữ liệu người dùng, các bản ghi giao dịch.
- **Ứng dụng:** Cung cấp tính năng tìm kiếm nhanh chóng, hỗ trợ truy vấn phức tạp.

5.1.2 Google Auth 2

Google Auth 2 được sử dụng để xác thực người dùng qua tài khoản Google. Điều này giúp đơn giản hóa quy trình đăng nhập, cung cấp một phương thức bảo mật cao, đồng thời giảm thiểu thời gian phát triển hệ thống xác thực.

- **Chức năng:** Đăng nhập thông qua Google, xác thực người dùng và lấy thông tin người dùng từ tài khoản Google.
- **Ứng dụng:** Đảm bảo tính bảo mật và giảm thiểu thao tác đăng ký và đăng nhập thủ công.

5.1.3 SendMail

Tính năng gửi email được tích hợp để hệ thống có thể gửi thông báo cho người dùng, như thông báo xác nhận đăng ký, quên mật khẩu, và các cập nhật quan trọng khác.

- **Chức năng:** Gửi email tới người dùng trong các trường hợp cần thiết.
- **Ứng dụng:** Thông báo cho người dùng qua email, ví dụ như xác nhận tài khoản, thông báo sự kiện hoặc hỗ trợ khách hàng.

5.1.4 Gin Framework

Gin là một framework web nhanh và nhẹ cho Go, được sử dụng để phát triển API cho ứng dụng. Với khả năng xử lý các yêu cầu HTTP hiệu quả, Gin giúp giảm thiểu độ trễ và tối ưu hóa tốc độ của hệ thống.

- **Chức năng:** Xử lý các yêu cầu HTTP, định tuyến các endpoint API.

- **Ứng dụng:** Tạo các RESTful APIs phục vụ cho hệ thống, đảm bảo hiệu suất cao trong việc xử lý các yêu cầu đồng thời.

5.1.5 JWT (JSON Web Token)

JWT được sử dụng để bảo vệ các endpoint API của hệ thống, đảm bảo rằng chỉ những người dùng đã đăng nhập mới có thể truy cập vào các tài nguyên bảo mật. JWT cho phép mã hóa các thông tin cần thiết trong token, giúp kiểm tra tính hợp lệ của người dùng mà không cần truy vấn cơ sở dữ liệu mỗi lần.

- **Chức năng:** Tạo và kiểm tra JWT để xác thực người dùng và bảo vệ các API.
- **Ứng dụng:** Cung cấp bảo mật cho hệ thống bằng cách sử dụng token xác thực trong mỗi yêu cầu API.

5.1.6 gotenv

gotenv được sử dụng để quản lý các biến môi trường trong hệ thống. Các thông tin nhạy cảm như mật khẩu cơ sở dữ liệu, API keys và các thông tin cấu hình khác được lưu trữ trong file .env và được tải vào ứng dụng khi cần thiết.

- **Chức năng:** Tải và sử dụng các biến môi trường từ file .env để cấu hình ứng dụng.
- **Ứng dụng:** Giúp quản lý các thông tin cấu hình và bảo mật mà không cần hard-code vào mã nguồn.

5.1.7 CORS (Cross-Origin Resource Sharing)

CORS là một cơ chế quan trọng trong việc quản lý và xử lý các yêu cầu từ các nguồn gốc khác nhau. CORS cho phép các ứng dụng web truy cập tài nguyên từ các nguồn khác ngoài nguồn gốc của ứng dụng, giúp tăng tính linh hoạt và khả năng tương tác giữa các hệ thống.

- **Chức năng:** Quản lý quyền truy cập từ các miền khác nhau đến các tài nguyên của hệ thống.
- **Ứng dụng:** Cho phép các yêu cầu từ client-side (từ trình duyệt) tiếp cận các API và tài nguyên từ máy chủ của ứng dụng, đảm bảo sự tương thích giữa các dịch vụ và ứng dụng web khác nhau.

5.1.8 Tổng hợp hệ thống

Hệ thống được thiết kế để hỗ trợ các yêu cầu bảo mật, hiệu suất cao và khả năng mở rộng. Các tính năng này cùng nhau tạo ra một hệ thống mạnh mẽ và linh hoạt, có thể đáp ứng các nhu cầu ngày càng tăng của người dùng. Các công nghệ được lựa chọn là hiện đại và phổ biến, đảm bảo tính tương thích và khả năng phát triển trong tương lai.

5.2 User Manuals

5.3 API Documentation

6 Code style

6.1 System Architecture

Hệ thống được phát triển theo mô hình **Model-View-Controller (MVC)** để phân tách rõ ràng giữa các phần logic của ứng dụng. Mỗi phần có một chức năng rõ ràng như sau:

- **Model:** Chứa các mô hình dữ liệu và logic truy vấn cơ sở dữ liệu. Các mô hình này được sử dụng để tương tác với cơ sở dữ liệu và quản lý các dữ liệu trong hệ thống.
- **View:** Chịu trách nhiệm hiển thị dữ liệu cho người dùng. Trong trường hợp của API, phần này không thực sự cần thiết, nhưng có thể có các phản hồi JSON trả về cho người dùng.
- **Controller:** Chứa các hàm xử lý logic cho các yêu cầu HTTP. Các controller này nhận và xử lý các yêu cầu từ người dùng, thực thi các hoạt động cần thiết và trả kết quả cho người dùng.

6.2 Version Control and Workflow

Chúng tôi sử dụng **GitFlow** như một mô hình phát triển để quản lý các nhánh trong quy trình làm việc. GitFlow giúp chúng tôi quản lý các tính năng, sửa lỗi và phát hành một cách có tổ chức. Các nhánh chính trong GitFlow bao gồm:

- **master**: Chứa mã nguồn ổn định đã sẵn sàng để phát hành.
- **develop**: Chứa các tính năng mới và sửa lỗi đã được kiểm tra và sẵn sàng cho việc phát hành.
- **feature**: Các nhánh này được tạo ra từ develop để phát triển các tính năng mới.
- **hotfix**: Các nhánh này được tạo ra từ master để sửa các lỗi khẩn cấp.
- **release**: Các nhánh này được sử dụng để chuẩn bị cho việc phát hành các tính năng mới.

6.3 GitHub Actions

Chúng tôi sử dụng **GitHub Actions** để tự động hóa quá trình xây dựng, kiểm tra và triển khai ứng dụng. Các hành động chính bao gồm:

- **CI (Continuous Integration):** Khi có bất kỳ thay đổi nào được đẩy lên repository, GitHub Actions tự động chạy các kiểm tra (tests) để đảm bảo rằng các thay đổi không làm hỏng chức năng của hệ thống.
- **CD (Continuous Deployment):** Sau khi kiểm tra thành công, GitHub Actions tự động triển khai ứng dụng lên môi trường thử nghiệm hoặc sản xuất.

Các quy trình CI/CD này giúp tăng cường hiệu quả và độ ổn định trong quá trình phát triển phần mềm.

6.4 Code Quality and Best Practices

Để đảm bảo chất lượng mã nguồn và bảo trì dễ dàng, chúng tôi tuân thủ các nguyên tắc sau:

- **Code Style:** Các quy tắc về định dạng mã nguồn như cách thụt lề, cách viết tên biến, hàm và lớp.
- **Naming Conventions:** Quy tắc đặt tên cho các tệp, hàm, biến, và các mô-đun theo một cách có tổ chức và dễ hiểu.
- **Commenting and Documentation:** Mỗi phần mã nguồn cần có chú thích rõ ràng để giải thích chức năng, mục đích và cách sử dụng của nó.
- **Best Practices:** Áp dụng các phương pháp tốt nhất như sử dụng các mô hình thiết kế, kiểm thử đơn vị, và bảo mật trong việc phát triển mã nguồn.

7 Slide

7.1 Presentation Overview

Link to the Canva design: **Canva Design Link**

8 CI

8.1 Continuous Integration (CI)

Listing 1: CI/CD Workflow với GitHub Actions và Docker

```
name: LearnGo
on:
  push:
    branches: [main]
jobs:
  docker:
    runs-on: self-hosted
    steps:
      - name: Checkout
        uses: actions/checkout@v4
      - name: Set up Docker
        run: |
          sudo apt-get update
          sudo apt-get install -y docker-compose
      - name: Build and Push DockerFile
        run: |
          echo "${{ secrets.DOCKERHUB_ACCESS_TOKEN }}" | sudo docker login -u "${{ se
          cd src/
          sudo docker build -t thaily/learngo .
          sudo docker push thaily/learngo
  deploy:
    needs: docker
    runs-on: self-hosted
    steps:
      - name: Pull docker image
        run: sudo docker pull ${{ secrets.DOCKERHUB_USERNAME }}/learngo:latest
      - name: Delete old container
        run: sudo docker rm -f learngo
      - name: Run docker container
        run: |
          cd ~
          sudo docker run -d -p 8080:8080 --name learngo --env-file .env thaily/learng
```

8.2 Giải thích Workflow

Workflow trên thực hiện hai công việc chính: `docker` và `deploy`.

- Trong công việc `docker`, mã nguồn được kiểm tra, Docker được cài đặt, Docker image được xây dựng và đẩy lên DockerHub.
- Công việc `deploy` sẽ kéo Docker image từ DockerHub và chạy container Docker trên máy chủ.

8.3 Lợi ích của CI

- Tự động phát hiện lỗi trong mã nguồn sớm.
- Đảm bảo mã luôn ở trạng thái có thể triển khai.
- Tiết kiệm thời gian và công sức trong việc triển khai phần mềm.

9 Bonus