Báo Cáo Thiết Kế Cơ Sở Dữ Liệu

Mục Lục

I. Giới Thiệu

1. Mục đích và tầm quan trọng của hệ thống cơ sở dữ liệu

Hệ thống cơ sở dữ liệu được thiết kế nhằm mục đích quản lý thông tin một cách hiệu quả, chính xác, và có tổ chức. Với sự phát triển của công nghệ, nhu cầu lưu trữ và xử lý thông tin lớn trở thành một phần không thể thiếu trong quản lý. Hệ thống cơ sở dữ liệu này được xây dựng để hỗ trợ quản lý thông tin liên quan đến người dùng, khoa, sinh viên, môn học, và điểm số trong một tổ chức giáo dục.

2. Phạm vi và đối tượng sử dụng hệ thống

Hệ thống này được thiết kế để phục vụ cho các trường đại học, cao đẳng, hoặc các tổ chức giáo dục khác. Đối tượng sử dụng bao gồm:

- **Quản trị viên**: Quản lý toàn bộ hệ thống.
- **Người quản lý khoa**: Quản lý thông tin sinh viên, môn học, và các hoạt động liên quan đến khoa.
- **Sinh viên**: Tra cứu thông tin cá nhân, điểm số, và môn học.

3. Tổng quan về hệ thống

Đây là một hệ thống quản lý trường học, cung cấp các chức năng như:

- Quản lý người dùng (quản trị viên, người quản lý khoa, sinh viên).
- Quản lý khoa và môn học.
- Lưu trữ và quản lý thông tin sinh viên.
- Ghi nhận và theo dõi điểm số môn học của sinh viên.

II. Yêu Cầu Hệ Thống

1. Yêu cầu chức năng

- Quản lý thông tin người dùng: Tạo, sửa, xóa, và truy xuất danh sách người dùng.
- Quản lý khoa: Lưu trữ thông tin khoa, người quản lý khoa.
- Quản lý sinh viên: Lưu trữ thông tin sinh viên, trạng thái học tập.
- Quản lý môn học: Lưu trữ thông tin các môn học theo từng khoa.
- Quản lý điểm số: Ghi nhận và cập nhật điểm số của sinh viên theo từng môn học.

- Cung cấp các báo cáo thống kê: Số lượng sinh viên, điểm trung bình, các sinh viên tốt nghiệp, v.v.

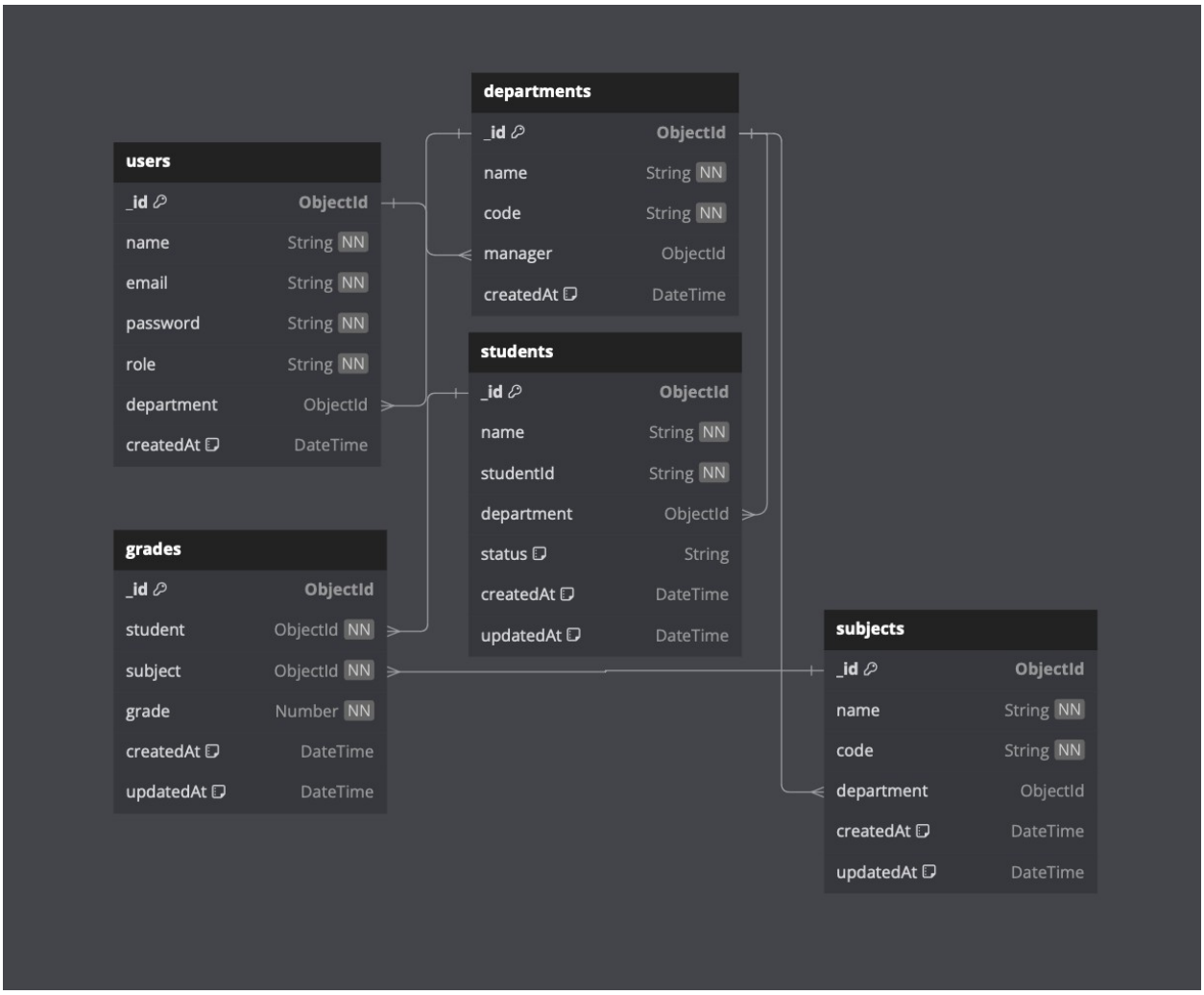
2. Yêu cầu phi chức năng

- **Bảo mật**: Dữ liệu người dùng (như mật khẩu) phải được mã hóa. Phân quyền rõ ràng giữa quản trị viên, quản lý khoa và sinh viên.
- **Hiệu suất**: Đảm bảo truy vấn nhanh chóng, đặc biệt khi số lượng sinh viên và dữ liệu lớn.
- **Khả năng mở rộng**: Hệ thống có thể mở rộng để hỗ trợ thêm nhiều khoa, sinh viên, và môn học.
- **Độ tin cậy**: Đảm bảo dữ liệu không bị mất mát hoặc lỗi khi hệ thống gặp sự cố.

III. Mô Hình Dữ Liệu

1. ER Diagram

Sơ đồ ER (Entity-Relationship) mô tả các thực thể chính trong hệ thống, bao gồm: Users, Departments, Students, Subjects, và Grades. Mỗi thực thể có các thuộc tính cụ thể và mối quan hệ giữa chúng.



2. Mô hình dữ liệu khái niệm

Dưới đây là mô tả tổng quan về các thực thể và mối quan hệ:

- **Users**: Lưu trữ thông tin người dùng, bao gồm quản trị viên và người quản lý khoa.
- **Departments**: Lưu trữ thông tin các khoa trong trường.
- **Students**: Lưu trữ thông tin sinh viên, liên kết với khoa.
- **Subjects**: Lưu trữ thông tin môn học, liên kết với khoa.
- **Grades**: Lưu trữ điểm số của sinh viên theo từng môn học.

IV. Thiết Kế Chi Tiết

1. Mô hình dữ liệu quan hệ

1. Bảng users

Bảng users lưu trữ thông tin người dùng trong hệ thống. Mỗi người dùng có thể là một giảng viên, quản lý, sinh viên hoặc admin. Nếu người dùng là quản lý khoa, họ sẽ có liên kết với một phòng ban.

Tên trường	Kiểu dữ liệu	Mô tả
_id	ObjectId	Khoá chính, duy nhất cho mỗi người dùng.
email	String	Email người dùng, duy nhất và không được null.
password	String	Mật khẩu người dùng, không được null.
role	String	Vai trò của người dùng (có thể là "manager", "admin", "student", mặc định là "student").
department	ObjectId	Liên kết tới bảng departments._id nếu người dùng là quản lý khoa.
createdAt	DateTime	Thời gian tạo tài khoản.

Mối quan hệ:

- Mỗi người dùng có thể là quản lý của một phòng ban (nếu role là "manager").
- Nếu role là "student", không có liên kết tới phòng ban.

2. Bảng departments

Bảng departments lưu trữ thông tin về các phòng ban (khoa) trong hệ thống. Mỗi phòng ban có một người quản lý (quản lý khoa).

Tên trường	Kiểu dữ liệu	Mô tả
_id	ObjectId	Khoá chính, duy nhất cho mỗi phòng ban.
name	String	Tên phòng ban, duy nhất và không được null.
code	String	Mã phòng ban, duy nhất và không được null.
manager	ObjectId	Khoá ngoại tham chiếu tới bảng users._id (quản lý khoa).
createdAt	DateTime	Thời gian tạo phòng ban, mặc định là ngày hiện tại.

Mối quan hệ:

- Mỗi phòng ban có một người quản lý là người dùng trong bảng users.

3. Bảng students

Bảng students lưu trữ thông tin về sinh viên trong hệ thống. Mỗi sinh viên có thể thuộc về một phòng ban (khoa) và có trạng thái riêng.

Tên trường	Kiểu dữ liệu	Mô tả
_id	ObjectId	Khoá chính, duy nhất cho mỗi sinh viên.
name	String	Tên của sinh viên, bao gồm cả tên và họ.
studentId	String	Mã sinh viên, duy nhất và không được null.
department	String	Khoa mà sinh viên thuộc về.
status	String	Trạng thái của sinh viên (có thể là "active", "graduated", "dropped out", mặc định là "active").
createdAt	DateTime	Thời gian tạo bản ghi, tự động thêm.
updatedAt	DateTime	Thời gian cập nhật bản ghi, tự động thêm.

Mối quan hệ:

- Mỗi sinh viên thuộc về một phòng ban và có thể tham gia vào các môn học.

4. Bảng grades

Bảng grades lưu trữ thông tin về điểm của sinh viên trong các môn học. Mỗi điểm của sinh viên được ghi lại cho một môn học.

Tên trường	Kiểu dữ liệu	Mô tả
_id	ObjectId	Khoá chính, duy nhất cho mỗi bản ghi điểm.
studentId	ObjectId	Khoá ngoại tham chiếu tới bảng students._id (sinh viên).
subjectId	ObjectId	Khoá ngoại tham chiếu tới bảng subjects._id (môn học).
grade	String	Điểm của sinh viên trong môn học.
createdAt	DateTime	Thời gian tạo bản ghi, tự động thêm.
updatedAt	DateTime	Thời gian cập nhật bản ghi, tự động thêm.

Mối quan hệ:

- Mỗi bản ghi trong bảng này liên kết một sinh viên với một môn học và điểm của họ.

5. Bảng subjects

Bảng subjects lưu trữ thông tin về các môn học trong hệ thống. Mỗi môn học có thể thuộc về một phòng ban (khoa).

Tên trường	Kiểu dữ liệu	Mô tả
_id	ObjectId	Khoá chính, duy nhất cho mỗi môn học.
name	String	Tên môn học, duy nhất và không được null.
code	String	Mã môn học, duy nhất và không được null.
department	ObjectId	Khoá ngoại tham chiếu tới bảng departments._id (khoa quản lý môn học).
createdAt	DateTime	Thời gian tạo môn học, tự động thêm.
updatedAt	DateTime	Thời gian cập nhật môn học, tự động thêm.

Mối quan hệ:

- Mỗi môn học thuộc về một phòng ban và có thể được giảng dạy bởi nhiều giảng viên.

---

2. Lược đồ cơ sở dữ liệu

Dưới đây là mã nguồn đã chỉnh sửa và viết lại theo cấu trúc chuẩn của Mongoose cho các bảng trong hệ thống:

Bảng User:

```
const mongoose = require('mongoose');

const userSchema = new mongoose.Schema({
  name: { type: String, required: true },
  email: { type: String, unique: true, required: true },
  password: { type: String, required: true },
  role: {
    type: String,
    required: true,
    enum: ['manager', 'student'],
    default: 'student'
  },
  department: { type: mongoose.Schema.Types.ObjectId, ref: 'Department' },
  createdAt: { type: Date, default: Date.now }
});
```

```
const User = mongoose.model('User', userSchema);
module.exports = User;
```

Bảng Department:

```
const mongoose = require('mongoose');

const departmentSchema = new mongoose.Schema({
  name: { type: String, required: true, unique: true },
  code: { type: String, required: true, unique: true },
  manager: { type: mongoose.Schema.Types.ObjectId, ref: 'User' },
  createdAt: { type: Date, default: Date.now }
});

const Department = mongoose.model('Department', departmentSchema);
module.exports = Department;
```

Bảng Student:

```
const mongoose = require('mongoose');

const studentSchema = new mongoose.Schema({
  name: { type: String, required: true },
  studentId: { type: String, unique: true, required: true },
  department: { type: mongoose.Schema.Types.ObjectId, ref: 'Department', required: true },
  status: { type: String, enum: ['active', 'graduated', 'dropped out'], default: 'active' },
  createdAt: { type: Date, default: Date.now },
  updatedAt: { type: Date, default: Date.now }
});

const Student = mongoose.model('Student', studentSchema);
module.exports = Student;
```

Bảng Grade:

```
const mongoose = require('mongoose');

const gradeSchema = new mongoose.Schema({
  student: { type: mongoose.Schema.Types.ObjectId, ref: 'Student', required: true },
  subject: { type: mongoose.Schema.Types.ObjectId, ref: 'Subject', required: true },
  grade: { type: Number, required: true },
  createdAt: { type: Date, default: Date.now },
  updatedAt: { type: Date, default: Date.now }
});

const Grade = mongoose.model('Grade', gradeSchema);
module.exports = Grade;
```

Bảng Subject:

```
const mongoose = require('mongoose');

const subjectSchema = new mongoose.Schema({
  name: { type: String, required: true },
  code: { type: String, unique: true, required: true },
  department: { type: mongoose.Schema.Types.ObjectId, ref: 'Department' },
```

```
  createdAt: { type: Date, default: Date.now },
  updatedAt: { type: Date, default: Date.now }
});

const Subject = mongoose.model('Subject', subjectSchema);
module.exports = Subject;
```

Mỗi schema bao gồm các trường bắt buộc và các tham chiếu tới các bảng khác qua ObjectId. Các trường như createdAt và updatedAt được sử dụng để theo dõi thời gian tạo và cập nhật bản ghi. Các bảng cũng có các thuộc tính mặc định, như default và enum để kiểm soát giá trị nhập vào và giới hạn các giá trị hợp lệ.

Nếu bạn có thêm yêu cầu về các tính năng khác, hoặc cần điều chỉnh thêm về việc tham chiếu giữa các bảng, vui lòng cho biết.

V. Chuẩn Hóa Cơ Sở Dữ Liệu

Cơ sở dữ liệu được chuẩn hóa ở các dạng chuẩn 3NF (Third Normal Form) để loại bỏ sự dư thừa dữ liệu và đảm bảo tính toàn vẹn. Mỗi bảng được thiết kế sao cho không có các thuộc tính phụ thuộc vào thuộc tính không phải là khóa.

VI. Lập Kế Hoạch và Bảo Mật

1. Bảo Mật Dữ Liệu

- **Mã Hóa Mật Khẩu:**
  - Mã hóa mật khẩu người dùng trước khi lưu trữ (sử dụng bcrypt) để đảm bảo an toàn.
- **Phân Quyền Truy Cập:**
  - Phân quyền rõ ràng giữa quản trị viên, người quản lý khoa và sinh viên.
  - Sử dụng Auth Middleware kiểm tra JWT và vai trò của người dùng.
- **Giao Tiếp Bảo Mật:**
  - Sử dụng HTTPS để bảo vệ dữ liệu khi truyền tải giữa client và server.

2. Sao Lưu và Phục Hồi

- **Sao Lưu Định Kỳ:**

    - Lập kế hoạch sao lưu dữ liệu hàng ngày hoặc hàng tuần để đảm bảo không mất mát dữ liệu.
- **Quy Trình Phục Hồi:**
    - Xây dựng quy trình phục hồi dữ liệu nhanh chóng khi gặp sự cố hoặc lỗi hệ thống.

VII. Thiết Kế Luồng (Flow Design) và Giao Tiếp Hệ Thống

1. Luồng Dữ Liệu Tổng Quan

Hệ thống được chia thành 3 container chính:

- **Frontend Container:**
    - Ứng dụng Student Management App được xây dựng bằng React kết hợp với Vite.
    - Sử dụng Axios (với module cấu hình riêng) để giao tiếp với backend.
    - Các module chính: API Service, Auth Module và HTTP Client tùy chỉnh.

- **Backend Container:**

    -
    - Ứng dụng API Server được xây dựng bằng Express.js.
    - Bao gồm các thành phần cốt lõi:
        - **Core Components:** Auth Service (xử lý đăng nhập, mã hoá mật khẩu, tạo JWT), Auth Middleware (xác thực JWT), và cấu hình Database (Mongoose).
        - **Route Components:** Các route cho Auth, User, Department, Student, Subject, Grade và Course Registration.
        - **Controller & Service Components:** Xử lý logic nghiệp vụ và tương tác với cơ sở dữ liệu thông qua các Model.

- **Database Container:**
    - Sử dụng MongoDB Atlas để lưu trữ dữ liệu.
    - Kết nối với backend thông qua Mongoose, đảm bảo các truy vấn và thao tác dữ liệu được thực hiện nhanh chóng và an toàn.

2. Luồng Xử Lý Yêu Cầu (Request Flow)

Người Dùng Tương Tác với Giao Diện:

- Người dùng (sinh viên, quản trị viên, hoặc người quản lý khoa) thực hiện thao tác trên giao diện React.

Giao Tiếp Frontend – Backend:

- API Service sử dụng Axios để gửi request HTTP đến API Server, đính kèm token JWT nếu có (được cấu hình trong `axios-customize.js`).

Xử Lý Yêu Cầu tại Backend:

- **Express.js Server** nhận request và định tuyến tới các route tương ứng (ví dụ: `/api/auth`, `/api/users`, ...).
- **Auth Middleware** kiểm tra tính hợp lệ của JWT trước khi cho phép truy cập các API bảo mật.
- **Controllers** nhận yêu cầu, gọi các **Service** tương ứng để xử lý nghiệp vụ (đăng nhập, quản lý dữ liệu, ...).
- **Service Components** tương tác với MongoDB thông qua các Model được định nghĩa (User, Department, Student, Subject, Grade).

Phản Hồi về Frontend:

- Kết quả từ các thao tác xử lý được trả về cho controller, sau đó gửi về phía client thông qua Express.
- Frontend nhận dữ liệu, hiển thị thông tin hoặc thông báo lỗi cho người dùng.

-
  - Ứng dụng API Server được xây dựng bằng Express.js.
  - Bao gồm các thành phần cốt lõi:
    - Core Components: Auth Service (xử lý đăng nhập, mã hoá mật khẩu, tạo JWT), Auth Middleware (xác thực JWT), và cấu hình Database (Mongoose).
    - Route Components: Các route cho Auth, User, Department, Student, Subject, Grade và Course Registration.
    - Controller & Service Components: Xử lý logic nghiệp vụ và tương tác với cơ sở dữ liệu thông qua các Model.

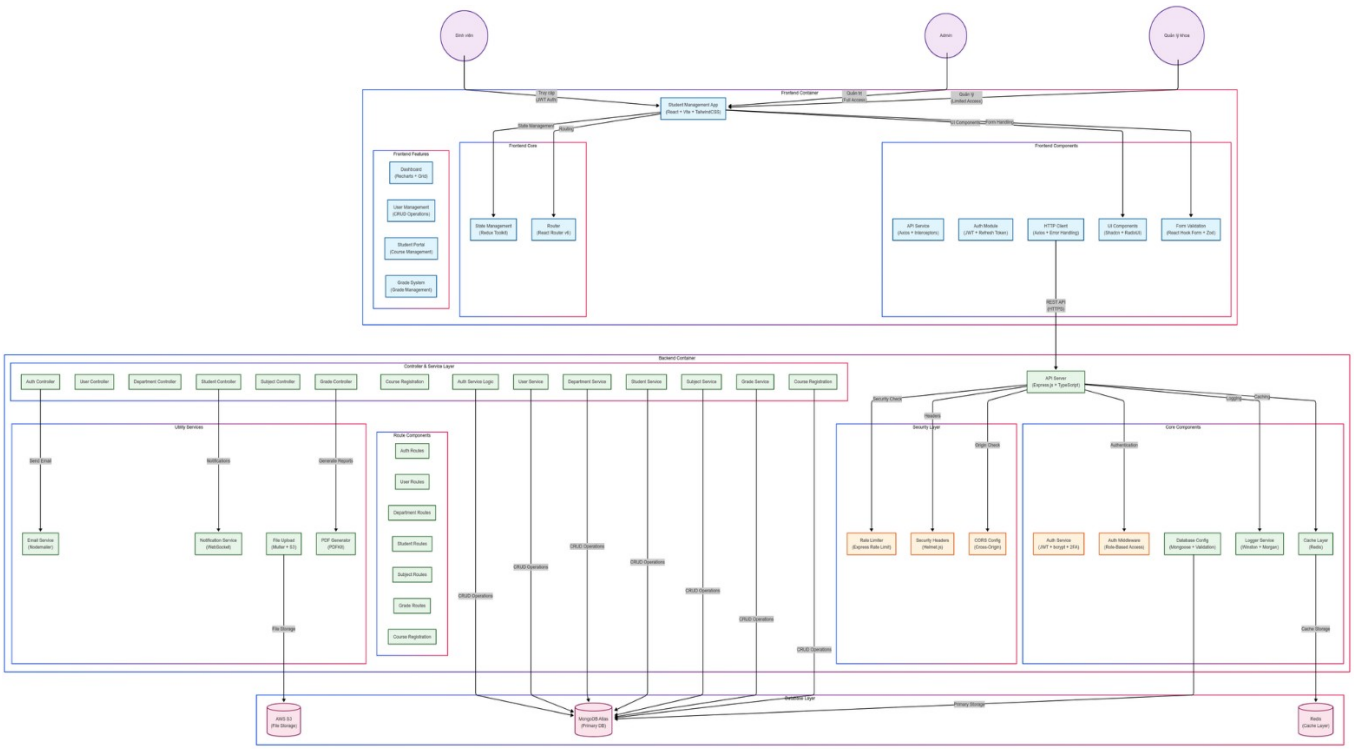
- **Database Container:**

- Sử dụng MongoDB Atlas để lưu trữ dữ liệu.
- Kết nối với backend thông qua Mongoose, đảm bảo các truy vấn và thao tác dữ liệu được thực hiện nhanh chóng và an toàn.

3. Luồng Xử Lý Yêu Cầu (Request Flow)

1. **Người Dùng Tương Tác với Giao Diện:**
   - Người dùng (sinh viên, quản trị viên, hoặc người quản lý khoa) thực hiện thao tác trên giao diện React.
2. **Giao Tiếp Frontend – Backend:**
   - API Service sử dụng Axios để gửi request HTTP đến API Server, đính kèm token JWT nếu có (được cấu hình trong `axios-customize.js`).
3. **Xử Lý Yêu Cầu tại Backend:**
   - Express.js Server nhận request và định tuyến tới các route tương ứng (ví dụ: `/api/auth`, `/api/users`, ...).
   - Auth Middleware kiểm tra tính hợp lệ của JWT trước khi cho phép truy cập các API bảo mật.
   - Controllers nhận yêu cầu, gọi các Service tương ứng để xử lý nghiệp vụ (đăng nhập, quản lý dữ liệu, ...).
   - Service Components tương tác với MongoDB thông qua các Model được định nghĩa (User, Department, Student, Subject, Grade).
4. **Phản Hồi về Frontend:**
   - Kết quả từ các thao tác xử lý được trả về cho controller, sau đó gửi về phía client thông qua Express.
   - Frontend nhận dữ liệu, hiển thị thông tin hoặc thông báo lỗi cho người dùng.

4. Sơ Đồ Luồng Dữ Liệu (Mermaid Diagram)



VIII. Kết Luận

Thiết kế cơ sở dữ liệu này cung cấp một hệ thống quản lý dữ liệu hiệu quả, bảo mật, và dễ mở rộng. Hệ thống hứa hẹn sẽ hỗ trợ tối ưu cho các trường học hoặc tổ chức giáo dục trong việc quản lý thông tin một cách khoa học và chính xác.

IX. Tài Liệu Tham Khảo

1. "Database System Concepts" - Silberschatz, Korth, Sudarshan.
2. PostgreSQL Documentation.
3. MongoDB Schema Design Guidelines.