TRƯỜNG ĐẠI HỌC VĂN LANG
KHOA CÔNG NGHỆ THÔNG TIN

THIẾT KẾ KIẾN TRÚC PHẦN MỀM

NGÀNH: CÔNG NGHỆ THÔNG TIN

BÀI THỰC HÀNH BUỔI 1

SVTH: Trần Minh Phúc - 2274802010694
Châu Gia Kiệt - 2274802010449
Võ Đình Ngọc Bình – 2274802010066

LHP: 243_71ITSE41803_0101

GVHD: Nguyễn Thế Quang

TP. Hồ Chí Minh – Năm 2025

MỤC LỤC

Câu 1: Thiết Kế Hệ Thống Cho Chức Năng Xem Tin 1
Câu 2: Thiết Kế Đạt Được Những Chất Lượng..... 1
Câu 3: Lý Do Thiết Kế Như Vậy..... 2

Câu 1: Thiết Kế Hệ Thống Cho Chức Năng Xem Tin

Chức năng "xem tin" được thiết kế theo mô hình kiến trúc nhiều lớp (N-Tier Architecture) để đảm bảo sự phân tách rõ ràng giữa giao diện, xử lý logic và lưu trữ dữ liệu. Cụ thể:

- **Client Layer**: Người dùng truy cập trang tin qua trình duyệt web.
- **Presentation Layer**: Giao diện frontend được xây dựng bằng ReactJS, có nhiệm vụ hiển thị danh sách bài viết, chi tiết bài viết, cho phép tìm kiếm và lọc theo chuyên mục.
- **API Layer**: ExpressJS định tuyến các yêu cầu HTTP và chuyển tiếp đến các dịch vụ phù hợp.
- **Application Layer**: Xử lý logic nghiệp vụ như phân trang bài viết, thống kê lượt xem, kiểm tra dữ liệu đầu vào...
- **Data Access Layer**: Áp dụng mô hình Repository để truy cập cơ sở dữ liệu một cách tách biệt, giúp dễ thay đổi hoặc mở rộng.
- **Database Layer**: Sử dụng MongoDB để lưu trữ bài viết, chuyên mục, tác giả… Ưu điểm là linh hoạt với cấu trúc dữ liệu không cố định (JSON document).

Luồng dữ liệu bắt đầu từ trình duyệt gửi yêu cầu xem tin → frontend gọi API → backend xử lý logic → truy vấn CSDL → trả dữ liệu về và hiển thị ra giao diện người dùng.

Câu 2: Thiết Kế Đạt Được Những Chất Lượng

Thiết kế hệ thống hiện tại đáp ứng nhiều **thuộc tính chất lượng phần mềm (Quality Attributes)**, cụ thể:

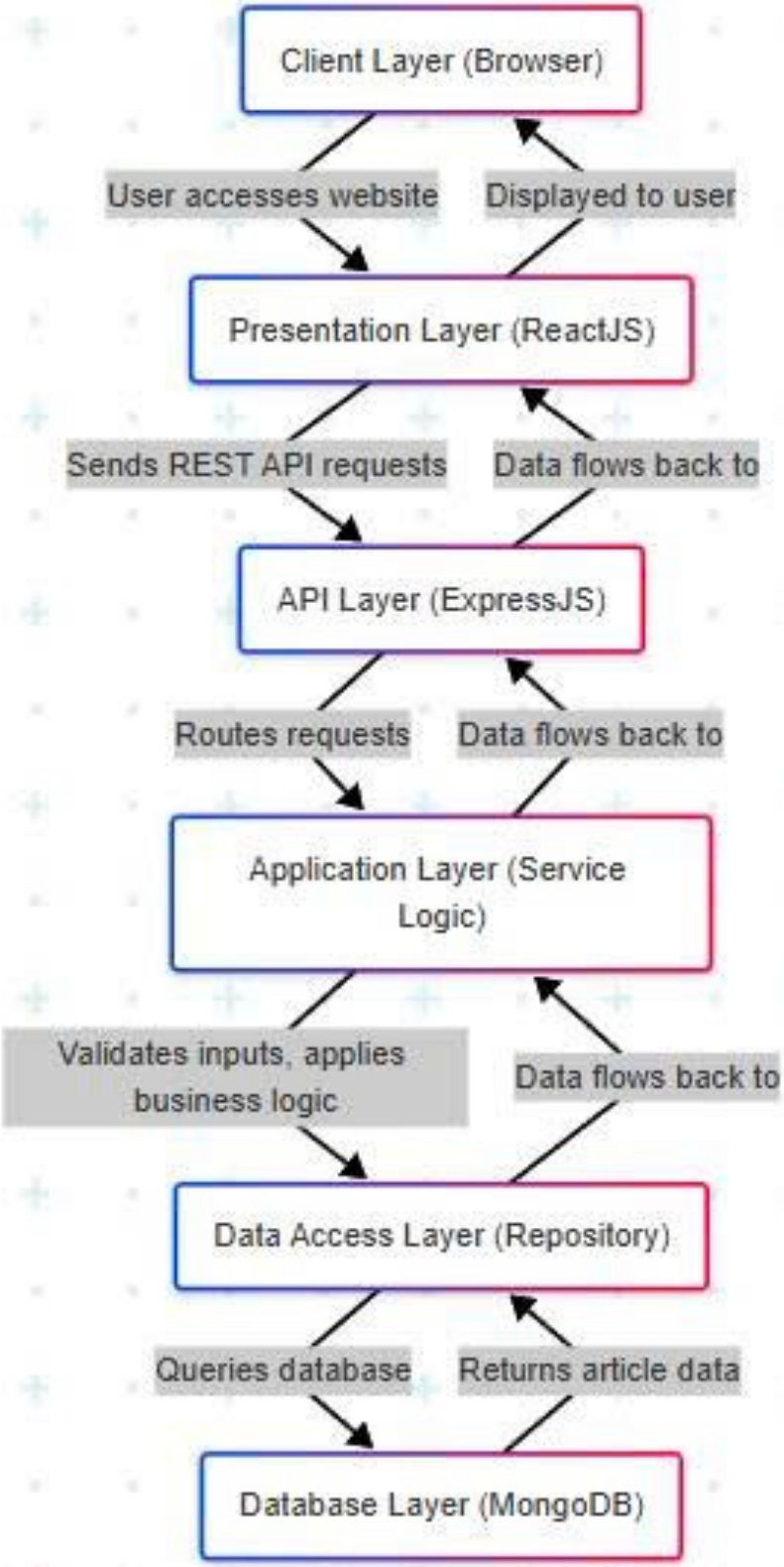
Thành phần	Vai trò
Performance	Dữ liệu được phân trang, truy vấn nhanh, tối ưu payload
Maintainability	Kiến trúc phân lớp giúp dễ bảo trì, dễ sửa chữa
Modularity	Các lớp đảm nhận vai trò riêng biệt (Separation of Concerns)

Scalability	Backend và database có thể scale độc lập (nằm trên cloud)
Security	Hệ thống chỉ cung cấp dữ liệu public, không lộ backend logic
Reusability	API backend có thể dùng lại cho mobile app hoặc hệ thống khác

Câu 3: Lý Do Thiết Kế Như Vậy

Chúng tôi lựa chọn thiết kế kiến trúc này dựa trên các lý do chính sau:

1. **Phù hợp với môi trường sinh viên**:
    - Sử dụng các công nghệ phổ biến (ReactJS, NodeJS, MongoDB) dễ học, dễ triển khai miễn phí (Vercel, MongoDB Atlas).
2. **Dễ mở rộng & bảo trì**:
    - Áp dụng mô hình nhiều lớp giúp dễ nâng cấp chức năng mà không ảnh hưởng toàn bộ hệ thống.
3. **Tuân thủ tư duy kiến trúc phần mềm chuyên nghiệp**:
    - Phân biệt rõ yêu cầu chức năng và phi chức năng.
    - Thiết kế tập trung vào hiệu suất, khả năng bảo trì và khả năng mở rộng.
4. **Đảm bảo chất lượng hệ thống**:
    - Các quyết định kiến trúc đều có thể truy vết về business goals hoặc kỹ thuật (traceability).
    - Luồng dữ liệu được xác định rõ ràng, kiểm soát đầu ra.


Client Layer (Browser)
User accesses website
Displayed to user
Presentation Layer (ReactJS)
Sends REST API requests
Data flows back to
API Layer (ExpressJS)
Routes requests
Data flows back to
Application Layer (Service Logic)
Validates inputs, applies business logic
Data flows back to
Data Access Layer (Repository)
Queries database
Returns article data
Database Layer (MongoDB)