

HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG



BÁO CÁO KẾT THÚC HỌC PHẦN

HỌC PHẦN: THỰC TẬP CƠ SỞ

TÊN ĐỀ TÀI: XÂY DỰNG WEBSITE MẠNG XÃ HỘI

GIẢNG VIÊN HƯỚNG DẪN: TS.KIM NGỌC BÁCH

SINH VIÊN THỰC HIỆN: NGUYỄN ĐẠI PHÁT

MÃ SINH VIÊN: B22DCVT393

HÀ NỘI, 5/2025

MỤC LỤC

1. TỔNG QUAN VỀ ĐỀ TÀI.....	3
1.1. Giới thiệu chung.....	3
1.2. Mục tiêu đồ án.....	3
1.3. Phạm vi đồ án.....	3
2. PHÂN TÍCH YÊU CẦU.....	4
2.1. Yêu cầu chức năng.....	4
2.2. Yêu cầu phi chức năng.....	5
3. CÔNG NGHỆ VÀ CÔNG CỤ SỬ DỤNG.....	6
3.1. Kiến trúc chính.....	6
3.2. Công nghệ mở rộng.....	6
3.3. Công cụ phát triển và testing.....	7
4. THIẾT KẾ GIAO DIỆN VÀ HỆ THỐNG.....	8
4.1. Kiến trúc tổng thể.....	8
4.2. Thiết kế cơ sở dữ liệu.....	8
4.3. API Design.....	17
4.4. Frontend Architecture.....	18
4.5. Backend Architecture.....	19
4.6. Thiết kế giao diện người dùng (UX/UI).....	20
4.7. Lưu đồ thuật giải.....	25
5. CÀI ĐẶT VÀ TRIỂN KHAI.....	29
5.1. Cài đặt môi trường phát triển.....	30
5.2. Chạy ứng dụng.....	30
6. KẾT LUẬN.....	31
6.1. Tóm tắt đồ án.....	31
6.2. Mục tiêu đạt được.....	31
6.3. Ý nghĩa của đồ án.....	31

1. TỔNG QUAN VỀ ĐỀ TÀI

1.1. Giới thiệu chung

Trong thời đại công nghệ phát triển mạnh mẽ hiện nay, mạng xã hội đã trở thành một phần không thể thiếu trong đời sống hằng ngày của con người. Việc xây dựng một hệ thống mạng xã hội không chỉ giúp kết nối người dùng với nhau mà còn là cơ hội để tiếp cận và áp dụng các công nghệ web hiện đại.

Đề tài “Xây dựng website mạng xã hội” hướng đến việc thiết kế và lập trình một nền tảng mạng xã hội đơn giản, nơi người dùng có thể đăng ký, đăng nhập, trò chuyện, chia sẻ bài viết, bình luận và cập nhật tin tức. Qua đó, sinh viên được tiếp cận và vận dụng kiến thức về lập trình web frontend và backend, quản lý dữ liệu, bảo mật thông tin người dùng và tổ chức kiến trúc phần mềm một cách khoa học.

1.2. Mục tiêu đồ án

Mục tiêu của đồ án bao gồm:

- Xây dựng một hệ thống mạng xã hội hoạt động ổn định, có khả năng mở rộng.
- Cung cấp các chức năng cơ bản như:
 - Đăng ký, đăng nhập, quản lý thông tin người dùng.
 - Gửi và nhận tin nhắn theo thời gian thực (chat).
 - Đăng bài viết, tương tác bài viết (like, comment).
 - Chỉnh sửa thông tin cá nhân.
 - Giao diện người dùng thân thiện, responsive.
- Sử dụng các công nghệ phổ biến:
 - Frontend: ReactJS, Redux, CSS.
 - Backend: Node.js, ExpressJS, MongoDB.
 - Socket.io: để triển khai tính năng nhắn tin thời gian thực.

1.3. Phạm vi đồ án

Phạm vi thực hiện của đồ án tập trung vào các chức năng chính của một nền tảng mạng xã hội ở mức cơ bản:

- Người dùng:
 - Đăng ký và đăng nhập tài khoản.
 - Cập nhật và xem hồ sơ cá nhân.

- Bài viết:
 - Tạo, chỉnh sửa, và xóa bài viết.
 - Bình luận và tương tác với bài viết.
- Tin nhắn:
 - Gửi và nhận tin nhắn giữa các người dùng (real-time chat).
- Giao diện:
 - Thiết kế đơn giản, dễ sử dụng, tương thích với nhiều độ phân giải màn hình.
- Quản lý dữ liệu:
 - Lưu trữ dữ liệu trên MongoDB.
 - Quản lý API hiệu quả với hệ thống route, controller, model.

Các chức năng nâng cao như livestream, nhóm hội, gợi ý bạn bè... chưa được đề cập trong phạm vi đề án này do giới hạn về thời gian và nguồn lực.

2. PHÂN TÍCH YÊU CẦU

2.1. Yêu cầu chức năng

Dựa vào cấu trúc dự án đã triển khai, hệ thống mạng xã hội cần đáp ứng các yêu cầu chức năng chính sau:

1. Chức năng người dùng (User)

- Đăng ký tài khoản: thông qua register.css, Register.jsx, authController.js, auth.js.
- Đăng nhập: sử dụng Login.jsx, login.css.
- Quản lý thông tin cá nhân: chỉnh sửa profile (EditPage.jsx, edit.css), xem thông tin cá nhân (UserProfile).
- Lưu trữ và xác thực người dùng thông qua JWT, xử lý logic tại userController.js, User.js.

2. Chức năng bài viết (Post/News Feed)

- Tạo bài viết mới: Post.js (model), postController.js, post.js (route).
- Hiển thị bài viết trong trang chủ: thông qua HomePage, Feed, FeedNavBar, FeedNavBarBottom, Post.jsx.
- Chỉnh sửa, xóa bài viết: xử lý tại backend qua route post.js và controller postController.js.

3. Chức năng bình luận (Comment)

- Gửi bình luận cho bài viết: `commentController.js`, `Comment.js` (model), `commentSlice.js` (redux).
- Hiển thị và quản lý danh sách bình luận: `Comments.jsx`, `comments.css`.

4. Chức năng nhắn tin (Chat / Message)

- Gửi/nhận tin nhắn thời gian thực: sử dụng thư viện `Socket.io` (thư mục `socket/`).
- Hiển thị cuộc trò chuyện: `ChatOverview`, `RoomLayout`, `Conversation.jsx`, `Message.jsx`.
- Quản lý cuộc trò chuyện qua `conversation.js`, `conversationController.js`, `message.js`, `messageController.js`.

5. Chức năng giao diện người dùng

- Giao diện thân thiện, chia thành các component như `Header`, `Popup`, `Layout`, `News`, `Edit`, `Feed`, `HomePage`.
- Ứng dụng chuẩn thiết kế component-based của `ReactJS`.

6. Quản lý trạng thái ứng dụng

- Dùng `Redux` để quản lý thông tin người dùng, bài viết, điều hướng,... qua các slice như:
 - `authSlice.js`, `postSlice.js`, `userSlice.js`, `navigateSlice.js`, `commentSlice.js`.

2.2. Yêu cầu phi chức năng

1. Hiệu năng

- Hệ thống cần phản hồi nhanh, đảm bảo tốc độ khi xử lý đăng nhập, gửi bài viết, gửi tin nhắn.
- Tin nhắn hoạt động theo thời gian thực nhờ `Socket.io`.

2. Tính ổn định và tin cậy

- Dữ liệu người dùng, bài viết, tin nhắn được lưu trữ trên `MongoDB`.
- Backend xử lý lỗi (middlewares) để tránh ứng dụng bị crash.

3. Tính bảo mật

- Sử dụng `JWT` cho xác thực người dùng.
- Middleware kiểm tra quyền truy cập trước khi xử lý các API yêu cầu đăng nhập.
- Sử dụng `.env` để bảo mật các biến môi trường như `secret key`, `URL database`.

4. Khả năng mở rộng

- Cấu trúc mã rõ ràng theo mô hình MVC (Model-View-Controller).
- Backend phân chia theo Controller, Routes, Models rõ ràng.
- Frontend chia nhỏ theo component, dễ dàng mở rộng chức năng mới.

5. Tính dễ bảo trì

- Quản lý logic phân tán rõ ràng: Redux cho state, Components cho UI, Controller cho logic backend.
- File cấu trúc hợp lý giúp người mới dễ dàng tiếp cận và bảo trì.

6. Tính thân thiện với người dùng

- Giao diện trực quan, màu sắc dễ nhìn.
- Tối ưu hóa hiển thị trên các kích thước màn hình khác nhau.

3. CÔNG NGHỆ VÀ CÔNG CỤ SỬ DỤNG

3.1. Kiến trúc chính.

Website mạng xã hội được phát triển dựa trên một tập hợp các công nghệ JavaScript phổ biến bao gồm:

- MongoDB:
Cơ sở dữ liệu NoSQL được sử dụng để lưu trữ thông tin người dùng, bài viết, bình luận và tin nhắn. Cấu trúc dữ liệu dạng JSON giúp lưu trữ linh hoạt và hiệu quả.
- Express.js:
Framework web phía backend dùng để xây dựng RESTful API. Các route được tổ chức trong thư mục Routes/, xử lý logic thông qua Controllers/.
- React.js:
Thư viện JavaScript xây dựng giao diện người dùng. Ứng dụng sử dụng kiến trúc component, chia nhỏ chức năng theo từng file như HomePage.jsx, Feed.jsx, Comments.jsx,...
- Node.js:
Môi trường runtime cho JavaScript phía server. Kết hợp với Express.js để xây dựng hệ thống backend toàn diện.

3.2. Công nghệ mở rộng

Ngoài các thành phần trong kiến trúc chính, dự án còn tích hợp nhiều công nghệ mở rộng khác để nâng cao tính năng:

- **Redux Toolkit:**
Quản lý trạng thái ứng dụng trong frontend. Hệ thống sử dụng `authSlice.js`, `postSlice.js`, `userSlice.js`, `commentSlice.js` để lưu trữ và cập nhật dữ liệu toàn cục.
- **Socket.IO:**
Dùng cho tính năng nhắn tin thời gian thực, cho phép hai người dùng giao tiếp mà không cần tải lại trang.
- **Multer:**
Middleware để xử lý upload hình ảnh bài viết/avatar người dùng (tích hợp trong `utils/multer.js`).
- **Cloudinary:**
Dịch vụ lưu trữ ảnh trên cloud. Ứng dụng sử dụng Cloudinary API (file `cloudinary.js`) để upload và quản lý hình ảnh.
- **JWT (JSON Web Token):**
Xác thực người dùng, đảm bảo tính bảo mật cho các request API từ client.
- **dotenv:**
Quản lý biến môi trường như chuỗi kết nối MongoDB, token secret.

3.3. Công cụ phát triển và testing

1. Công cụ phát triển

- **Visual Studio Code (VSCode):**
Trình soạn thảo mã chính trong quá trình phát triển.
- **Git:**
Quản lý phiên bản mã nguồn. Kết hợp với GitHub để đồng bộ và theo dõi lịch sử thay đổi.
- **Postman:**
Dùng để test các API trong quá trình phát triển backend.
- **MongoDB Compass:**
Giao diện trực quan để theo dõi dữ liệu trong MongoDB.
- **Vercel:**
Dùng để deploy frontend React một cách nhanh chóng và dễ dàng (file `vercel.json` đã có trong dự án).

2. Testing thủ công

- Kiểm tra chức năng login/logout, tạo bài viết, chỉnh sửa thông tin cá nhân, gửi tin nhắn, bình luận.
- Kiểm tra khả năng phản hồi (responsive) trên các thiết bị và trình duyệt khác nhau.
- Test frontend/backend bằng cách sử dụng dữ liệu mẫu, kiểm tra xử lý lỗi và thông báo người dùng

4. THIẾT KẾ GIAO DIỆN VÀ HỆ THỐNG

4.1. Kiến trúc tổng thể

Hệ thống mạng xã hội được xây dựng theo mô hình client-server với kiến trúc tách biệt:

- Frontend: Ứng dụng React hoạt động phía trình duyệt, chịu trách nhiệm hiển thị giao diện người dùng và tương tác với API.
- Backend: Server Node.js với Express xử lý yêu cầu, xác thực, lưu trữ và truy xuất dữ liệu từ MongoDB.
- Database: MongoDB lưu trữ dữ liệu phi quan hệ bao gồm người dùng, bài viết, bình luận, tin nhắn,...
- Realtime: Sử dụng Socket.IO để xử lý tính năng nhắn tin thời gian thực.

4.2. Thiết kế cơ sở dữ liệu

Hệ thống sử dụng MongoDB với các collection chính:

1. User


```
you, 23 minutes ago | I author (you)
const mongoose = require("mongoose");
const { isEmail } = require("validator");
var uniqueValidator = require("mongoose-unique-validator");

const userSchema = new mongoose.Schema(
  {
    // _id: {
    //   type: String,    // Auto-generated so changed to string
    // },
    createdAt: {      You, 23 minutes ago • Uncommitted changes
      type: Date,
      default: Date.now
    },
    username: {
      type: String,
      required: [true, "Required"],
      minlength: [6, "Must be at least 6 characters"],
      maxlength: [20, "Must be less than 20 characters"],
      unique: true,
    },
    resetPasswordToken: String,
    resetPasswordExpires: Date,
    displayName: {
      type: String,
      default: "New User",
    },
  },

```

```

resetPasswordExpires: Date,
displayName: {
  type: String,
  default: "New User",
},
about: {
  type: String,
  default: "I'm a new user",
},
age: {
  type: Number,
  default: function() {
    // Calculate years since account creation
    const now = new Date();
    const created = this._id.getTimestamp();
    return Math.floor((now - created) / (1000 * 60 * 60 * 24 * 365));
  }
},
email: {
  type: String,
  required: [true, "Required"],
  maxlength: [50, "Must be 50 characters or less"],
  unique: true,
  validate: [isEmail, "Please enter a valid email"],
},

```

```

password: {
  type: String,
  required: [true, "Required"],
  select: false,
  minlength: [8, "Must be 8 characters or more"],
},
isAdmin: {
  type: Boolean,
  default: false,
},
profilePicture: {
  type: String,
  default:
    "https://preview.redd.it/rrz3hmsxc1l71.png?width=640&crop=smart&auto=webp&s=87cc5ed38d8f088ef",
},
theme: {
  type: String,
  default: "#ff9051",
},
karmas: {
  type: Number,
  default: 0,
},

```

```
    },
    followers: {
      type: Array,
      default: [],
    },
    followings: {
      type: Array,
      default: [],
    },
    favorites: {
      type: Array,
      default: [],
    },
    isBot: { type: Boolean, default: false },
    resetPasswordToken: {
      type: String,
      select: false,
    },
    resetPasswordExpires: {
      type: Date,
      select: false,
    },
  ],
},
{ timestamps: true }
```

2. Post

```
const mongoose = require("mongoose");  
  
const postSchema = new mongoose.Schema(  
  {  
    userId: {  
      type: String,  
      required: true,  
    },  
    username: {  
      type: String,  
    },  
    avatarUrl: {  
      type: String,  
    },  
    imageUrl: {  
      type: String,  
    },  
    cloudinaryId: {  
      type: String,  
    },  
    theme: {  
      type: String,  
    },  
  },  
  {  
    strict: false,  
  },  
);
```

```
title: {
  type: String,
  required: true,
  minlength: 10,
  maxlength: 100,
},
description: {
  type: String,
  required: true,
  minlength: 4,
},
tags: {
  type: Number,
  required: true,
  default: 0,
},
upvotes: {
  type: Array,
  default: [],
},
downvotes: {
  type: Array,
  default: [],
},
```

```
    },  
    downvotes: {  
      type: Array,  
      default: [],  
    },  
    comments: {  
      type: Number,  
      default: 0,  
    },  
  },  
  { timestamps: true }  
);
```

3. Comment

You, 2 months ago | 1 author (You)

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

You, 2 months

```
const commentSchema = new mongoose.Schema(  
  {  
    postId: {  
      type: String,  
    },  
    ownerId: {  
      type: String,  
      required: true,  
    },  
    content: {  
      type: String,  
      required: true,  
    },  
    username: {  
      type: String,  
    },  
    avaUrl: {  
      type: String,  
    },  
    theme: {  
      type: String,  
    },  
  },  
  { timestamps: true }
```

4. Message

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

const MessageSchema = new mongoose.Schema(
  {
    conversationId: {
      type: String,
      required: true,
    },
    sender: {
      type: String,
      required: true,
    },
    text: {
      type: String,
      required: true,
    },
  },
  { timestamps: true }
);

module.exports = mongoose.model("Message", MessageSchema);
```

5. Conversation


```
const ConversationSchema = new mongoose.Schema(
  {
    _id: {
      type: String,
      default: () => require('uuid').v4()
    },
    members: {
      type: Array,
    },
    messageCount: {
      type: Number,
      default: 0
    }
  },
  { timestamps: true }
);
```

4.3. API Design

Hệ thống backend cung cấp các RESTful API chia theo các route:

- /api/auth: Đăng ký, đăng nhập, xác thực người dùng
- /api/user: Lấy thông tin người dùng, cập nhật hồ sơ, danh sách người dùng
- /api/post: CRUD bài viết (create, read, update, delete)
- /api/comment: Thêm, xóa, lấy danh sách bình luận theo bài viết
- /api/conversation & /api/message: Quản lý cuộc trò chuyện và tin nhắn

Các phương thức phổ biến:

- POST – tạo mới dữ liệu
- GET – truy vấn dữ liệu
- PUT/PATCH – cập nhật
- DELETE – xóa dữ liệu

Middleware xác thực JWT được sử dụng để bảo vệ các route yêu cầu đăng nhập.

4.4. Frontend Architecture

frontend/

```
|— src/
|   |— pages/           # Các trang chính của ứng dụng
|   |   |— feed/        # Trang hiển thị dòng thời gian (bài viết)
|   |   |— profile/     # Trang cá nhân người dùng
|   |   |— chat/        # Trang nhắn tin thời gian thực
|   |   |— login/       # Trang đăng nhập
|   |   |— register/    # Trang đăng ký
|   |   └— not-found/   # Trang 404
|   |
|   |— components/      # Các component tái sử dụng
|   |   |— ui/          # Các thành phần giao diện (button, modal,...)
|   |   |— posts/       # Component hiển thị bài viết
|   |   |— comments/    # Component bình luận
|   |   |— chatbox/     # Component giao diện chat
|   |   |— navbar/      # Component thanh điều hướng
|   |   └— profile/     # Component avatar, bio,...
|   |
|   |— layout/          # Layout chính (wrapper, sidebar,...)
|   |— lib/             # Thư viện, API request helper, auth helper
|   |— providers/       # React context: auth, theme, socket,...
|   |— stores/          # Redux store (user, post, message)
|   |— types/           # TypeScript type definitions
|   |— App.tsx          # Component gốc của ứng dụng
|   |— main.tsx         # Entry point
|   └— index.css        # Global styles
```

```

├── public/           # Static files: logo, icon,...
├── package.json      # Dependencies và scripts
├── tsconfig.json     # TypeScript config
├── vite.config.ts    # Vite config
├── tailwind.config.js # Tailwind CSS config
└── postcss.config.js # PostCSS config

```

4.5. Backend Architecture

backend/

```

├── src/
|   ├── controllers/      # Xử lý logic cho route (user, post, chat, auth)
|   |   ├── auth.controller.js
|   |   ├── user.controller.js
|   |   ├── post.controller.js
|   |   ├── comment.controller.js
|   |   └── message.controller.js
|
|   ├── routes/           # Định tuyến API
|   |   ├── auth.js
|   |   ├── user.js
|   |   ├── post.js
|   |   ├── comment.js
|   |   └── message.js
|
|   ├── models/           # Mongoose schema định nghĩa dữ liệu MongoDB
|   |   ├── User.js
|   |   ├── Post.js
|   |   └── Comment.js

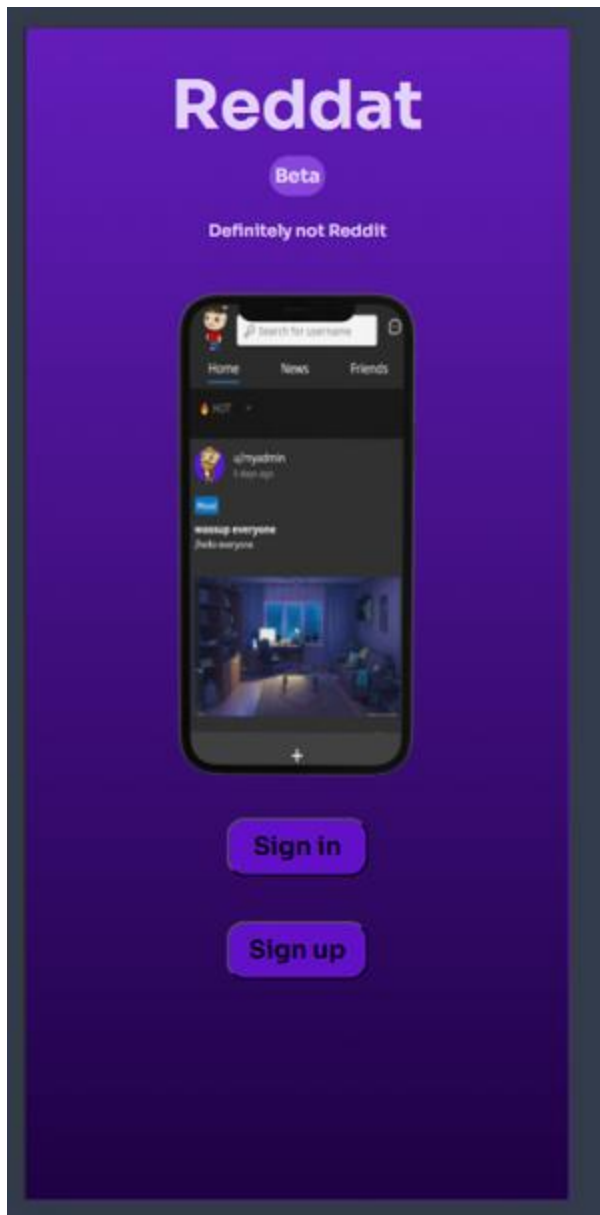
```

```

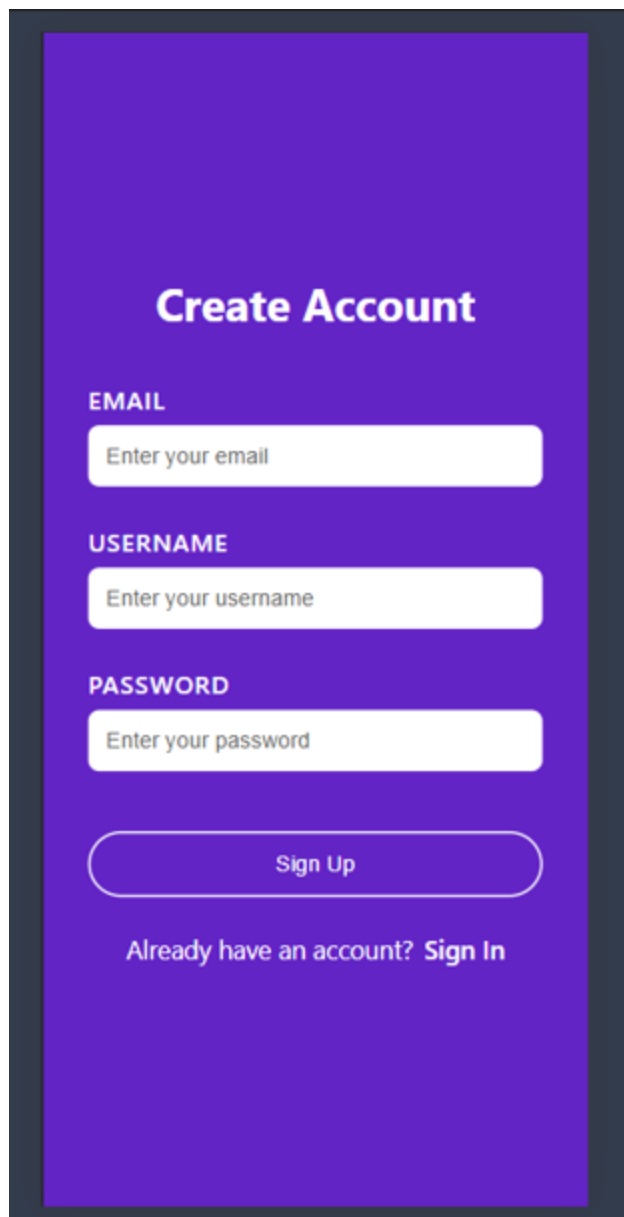
| | | └─ Message.js
| | | └─ Conversation.js
|
| └─ middleware/      # Xác thực JWT, xử lý lỗi,...
| | | └─ auth.middleware.js
| | | └─ error.middleware.js
| | | └─ validate.middleware.js
|
| └─ utils/           # Hàm tiện ích, cloudinary,...
| | | └─ cloudinary.js
| | | └─ multer.js
| index.js
└─ .env               # Biến môi trường
└─ package.json       # Thư viện và script
└─ package-lock.json  # TypeScript config

```

4.6. Thiết kế giao diện người dùng (UX/UI)



Giao diện welcome – khi chưa đăng kí, đăng nhập

A mobile app interface for creating a new account. The background is a solid purple color. At the top, the text "Create Account" is displayed in a bold, white, sans-serif font. Below this, there are three input fields, each with a label in white, uppercase letters above it. The first field is labeled "EMAIL" and contains the placeholder text "Enter your email". The second field is labeled "USERNAME" and contains the placeholder text "Enter your username". The third field is labeled "PASSWORD" and contains the placeholder text "Enter your password". Below these fields is a white, rounded rectangular button with the text "Sign Up" in a dark purple font. At the bottom, there is a line of text in white that reads "Already have an account? Sign In", where "Sign In" is a link.

Giao diện đăng kí

Reddat

BETA

Sign In

EMAIL

minh@gmail.com

PASSWORD

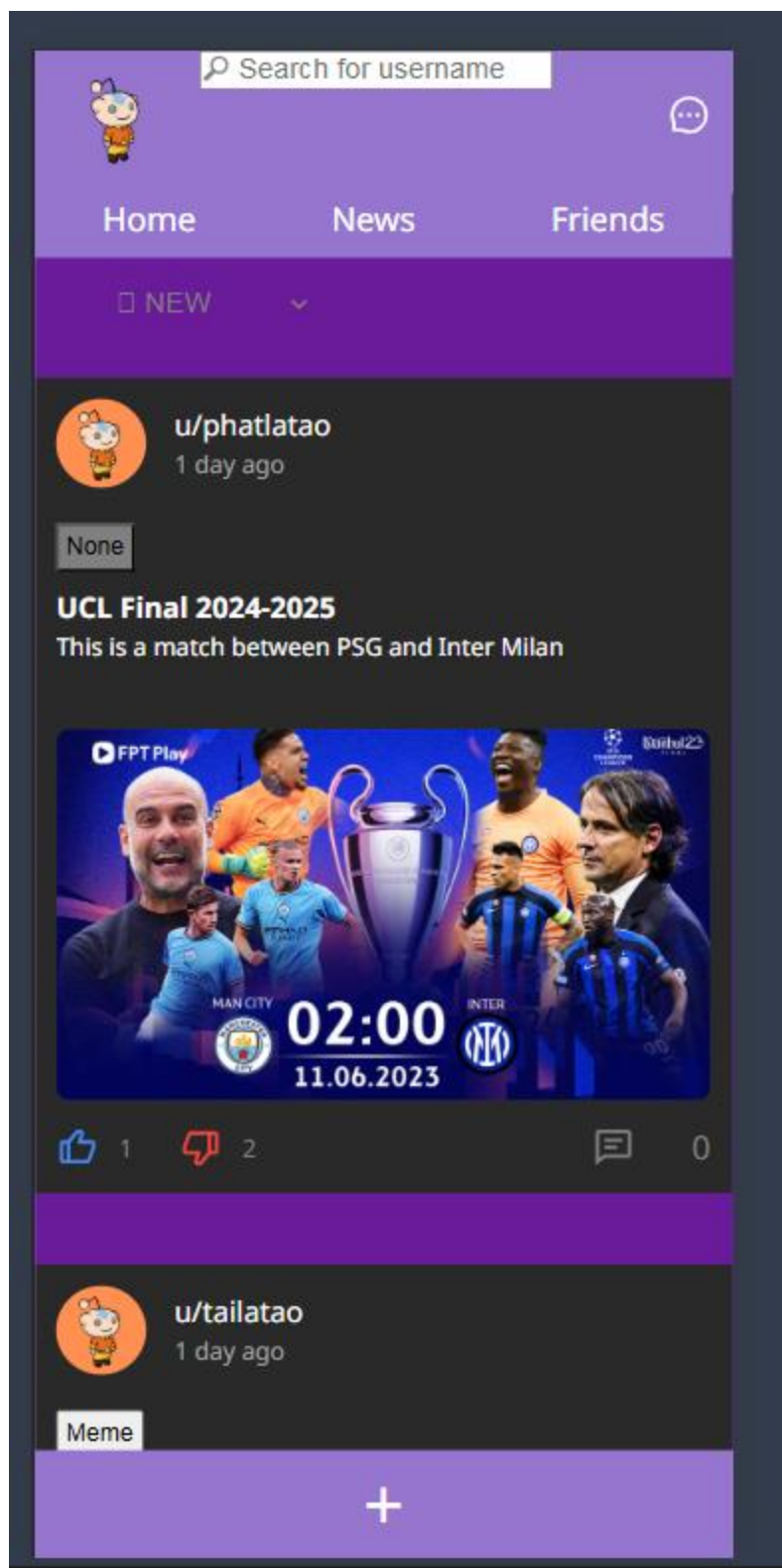
....

Sign In

Incorrect password

Don't have an account? [Sign Up](#)

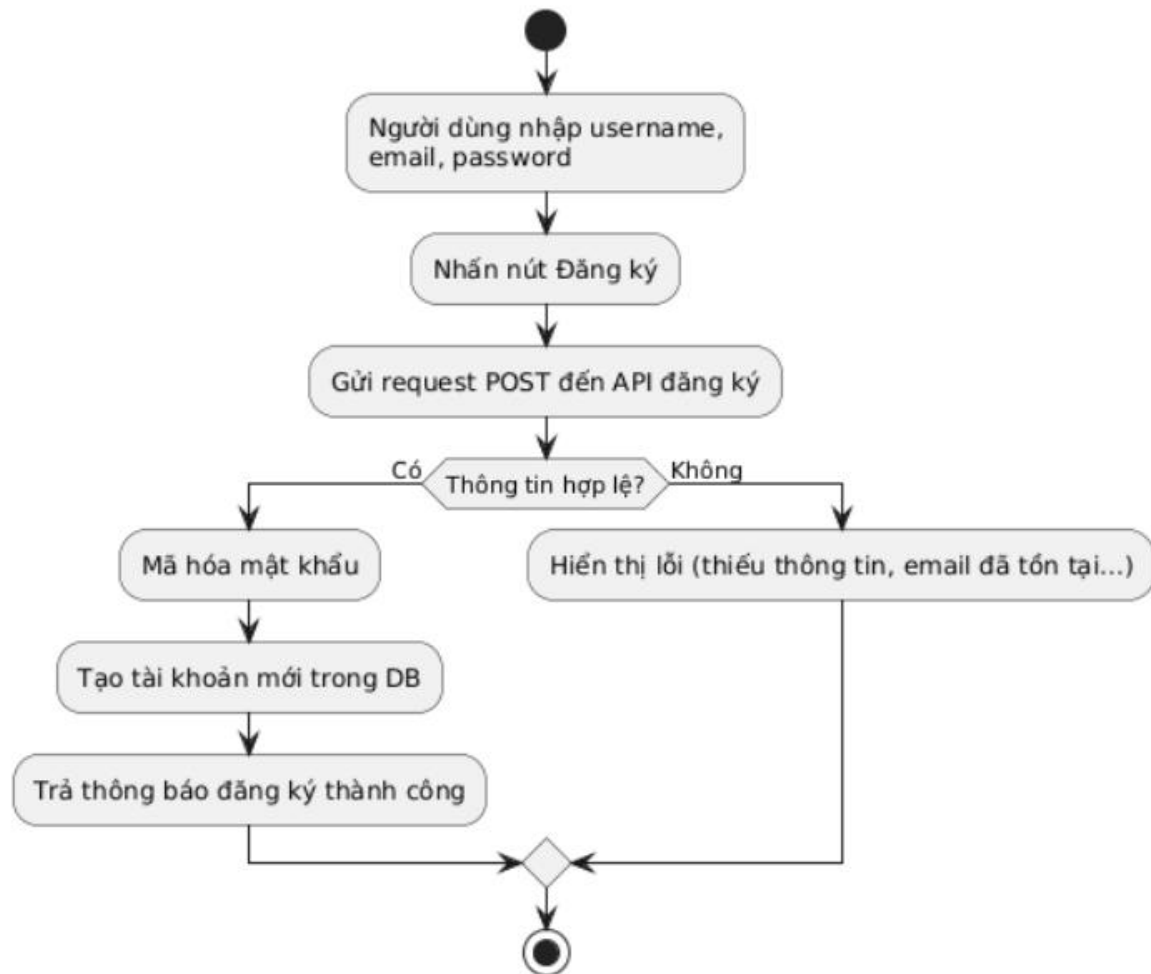
Giao diện đăng nhập



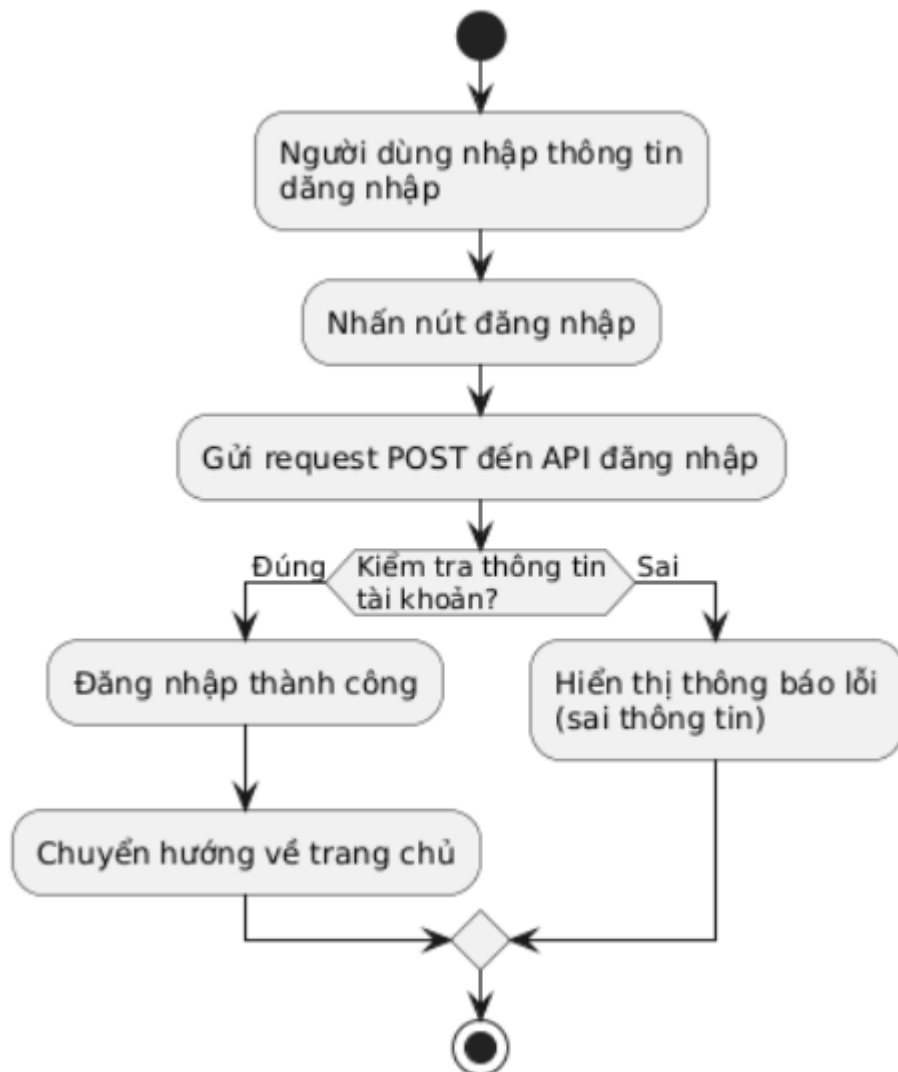
Giao diện trang chủ

4.7 Lưu đồ thuật toán

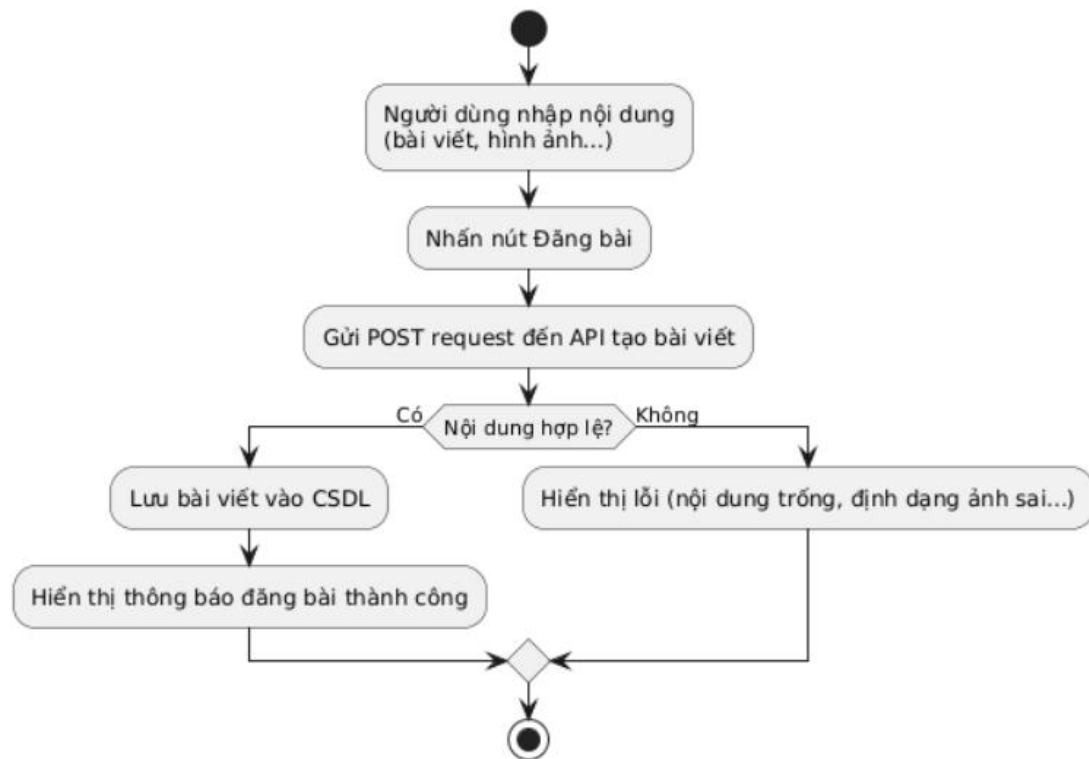
-Chức năng đăng ký:



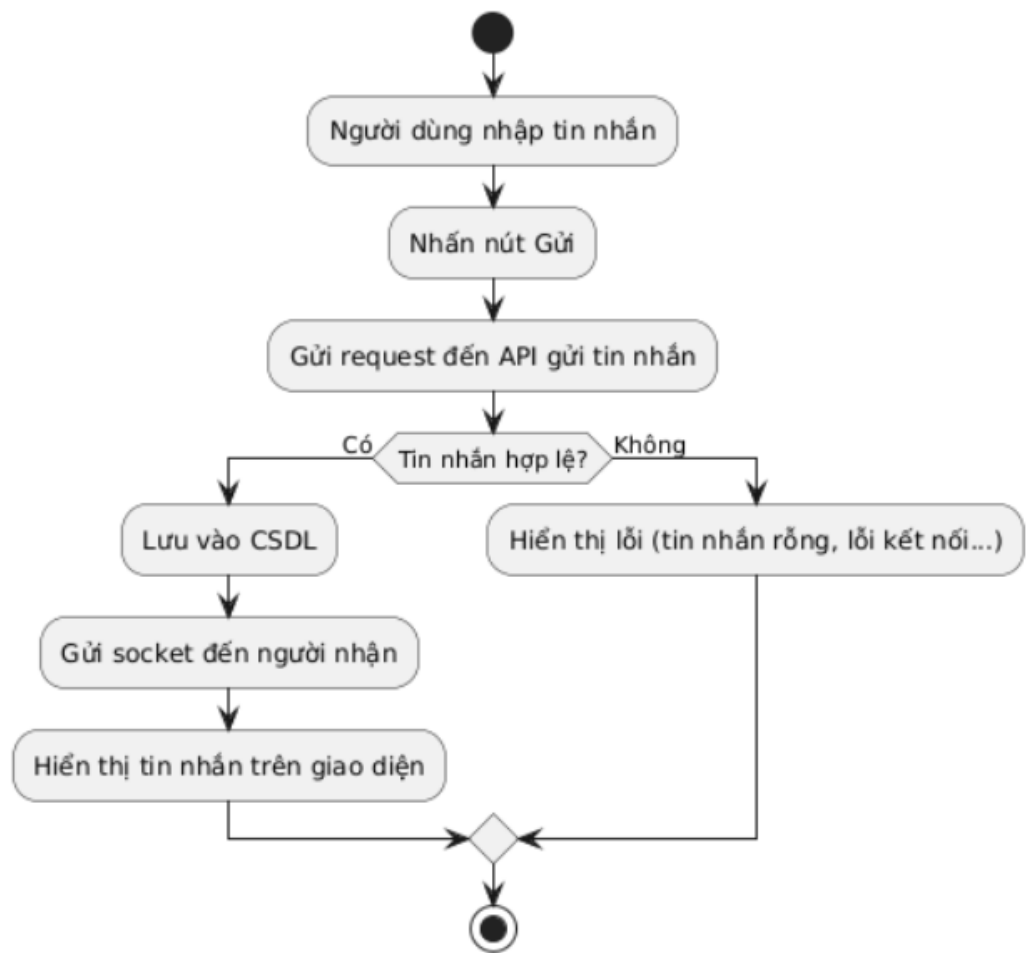
-Chức năng đăng nhập:



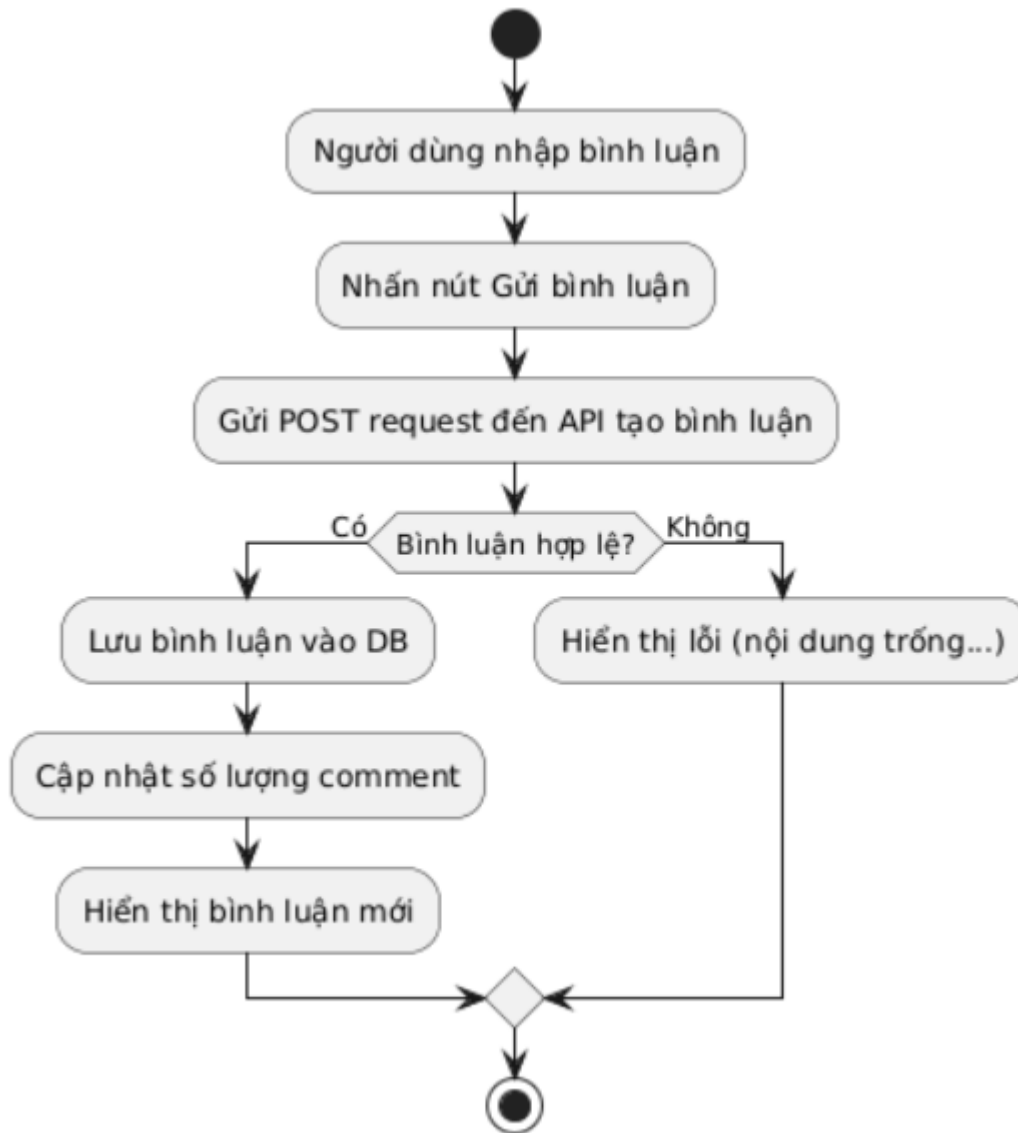
-Chức năng gửi bài viết:



-Chức năng gửi tin nhắn



-Chức năng bình luận:



5. CÀI ĐẶT VÀ TRIỂN KHAI

5.1. Cài đặt môi trường phát triển

Để phát triển và vận hành ứng dụng mạng xã hội, cần chuẩn bị các thành phần sau:

a. Yêu cầu hệ thống

- Node.js $\geq 18.x$
- MongoDB $\geq 6.x$ (hoặc MongoDB Atlas)
- Git
- VSCode (hoặc bất kỳ IDE nào hỗ trợ JavaScript/TypeScript)

- Trình duyệt: Chrome, Firefox, Edge,...

b. Cài đặt dự án Frontend

git clone <https://github.com/your-org/social-network-frontend.git>

cd social-network-frontend

npm install

c. Cài đặt dự án Backend

git clone <https://github.com/your-org/social-network-backend.git>

cd social-network-backend

npm install

d. Cài đặt MongoDB

- Tùy chọn 1: Cài đặt cục bộ trên máy (localhost:27017)
- Tùy chọn 2: Sử dụng MongoDB Atlas (miễn phí)

e. Tạo file cấu hình môi trường

- Tạo file .env ở cả frontend và backend.

5.2. Chạy ứng dụng

a. Khởi động Backend

npm run dev

Ứng dụng backend sẽ chạy tại <http://localhost:8000>

b. Khởi động Frontend

npm run dev

Ứng dụng frontend sẽ chạy tại <http://localhost:3000>

c. Truy cập ứng dụng

Mở trình duyệt và truy cập vào:

<http://localhost:3000> để bắt đầu sử dụng mạng xã hội.

6. KẾT LUẬN

6.1. Tóm tắt đồ án

Đồ án đã xây dựng thành công một website mạng xã hội cơ bản sử dụng kiến trúc MERN Stack (MongoDB, Express.js, React.js, Node.js). Ứng dụng cho phép người dùng thực hiện các chức năng phổ biến như đăng ký, đăng nhập, đăng bài viết, bình luận, nhắn tin, và quản lý thông tin cá nhân.

Hệ thống được chia thành hai phần chính:

- **Frontend:** Phát triển bằng React kết hợp với Tailwind CSS và các thư viện hỗ trợ như React Router, Axios, Zustand,...
- **Backend:** Xây dựng bằng Node.js và Express, sử dụng MongoDB làm cơ sở dữ liệu, JWT cho xác thực bảo mật và Socket.io để xử lý nhắn tin thời gian thực.

Các luồng xử lý chính đều được thiết kế rõ ràng, có tính mở rộng, đáp ứng yêu cầu người dùng cơ bản của một mạng xã hội hiện đại.

6.2. Mục tiêu đạt được

- Nắm vững quy trình xây dựng một ứng dụng web fullstack hoàn chỉnh.
- Áp dụng thành thạo các công nghệ trong MERN Stack vào thực tế.
- Hiểu và triển khai các khái niệm về xác thực người dùng (authentication), phân quyền, và bảo mật cơ bản.
- Thiết kế cơ sở dữ liệu NoSQL phù hợp với logic mạng xã hội.
- Phát triển UI/UX thân thiện với người dùng, có khả năng đáp ứng trên nhiều thiết bị (responsive).
- Triển khai hệ thống hoạt động ổn định trên môi trường máy cục bộ (development environment).

6.3. Ý nghĩa của đồ án

Đồ án không chỉ giúp người thực hiện củng cố kiến thức lập trình web mà còn rèn luyện khả năng:

- Phân tích yêu cầu hệ thống và tổ chức kiến trúc phần mềm hợp lý.
- Tư duy logic và giải quyết vấn đề thông qua thiết kế luồng xử lý thực tế.
- Làm việc nhóm (nếu có), phối hợp frontend-backend hiệu quả.
- Tiếp cận các công nghệ phổ biến hiện nay trong ngành phát triển phần mềm web.

Ngoài ra, sản phẩm của đồ án có thể được phát triển tiếp thành một mạng xã hội thực tế, phục vụ nhu cầu kết nối, chia sẻ và tương tác của cộng đồng người dùng.